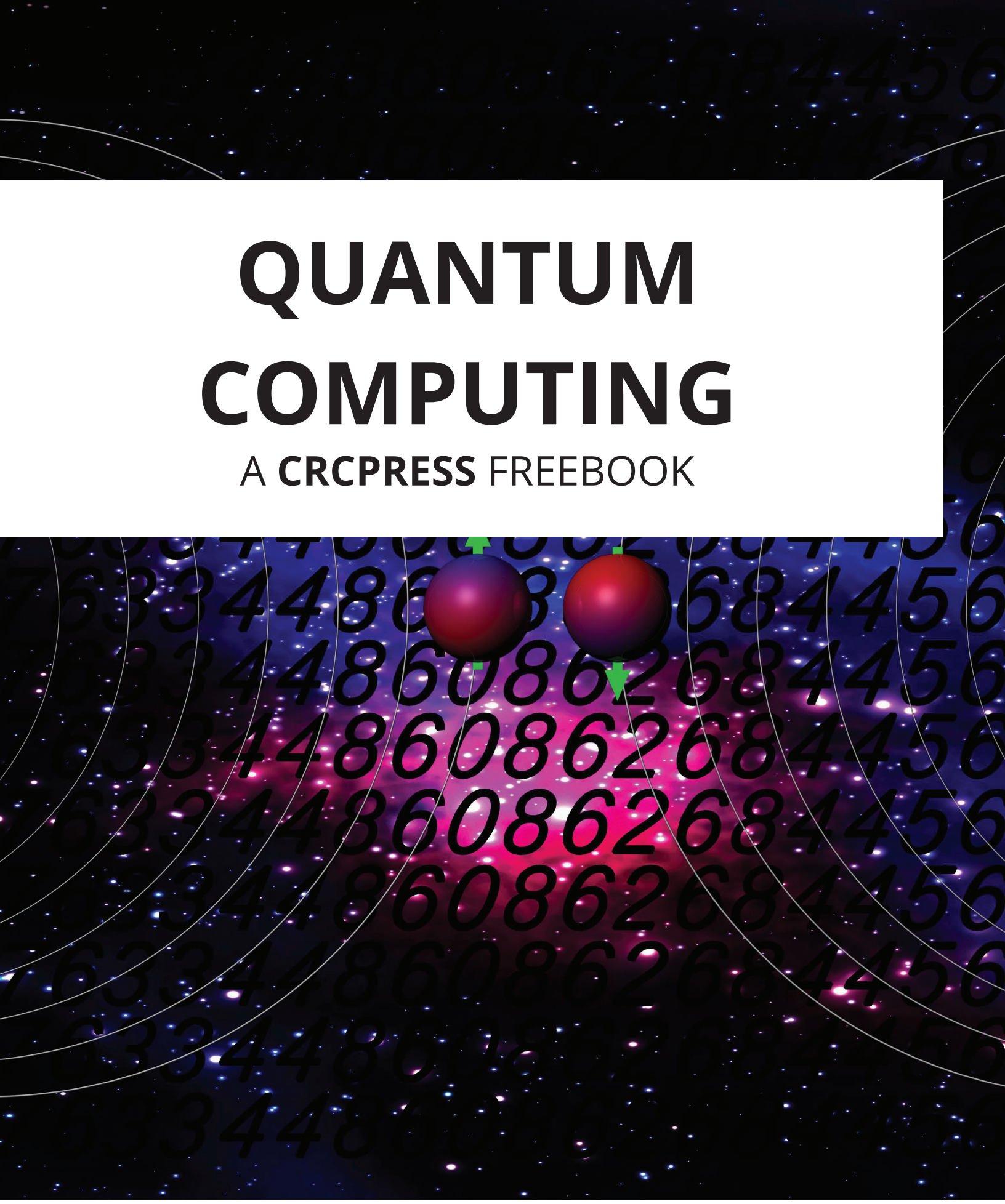
QUANTUM COMPUTING

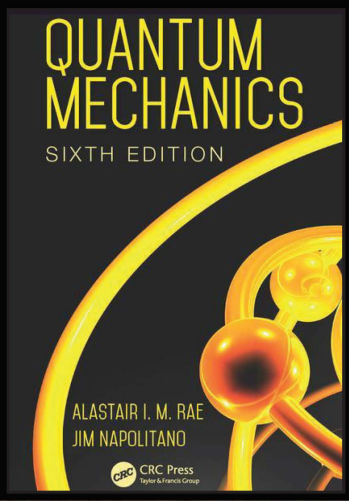
A **CRCPRESS** FREEBOOK

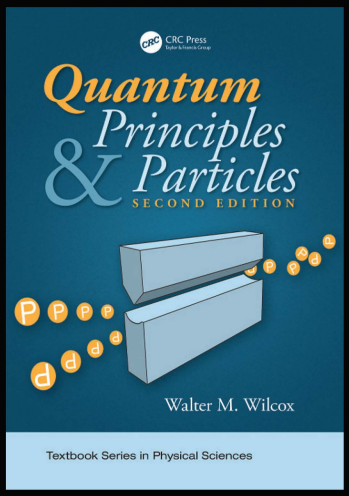


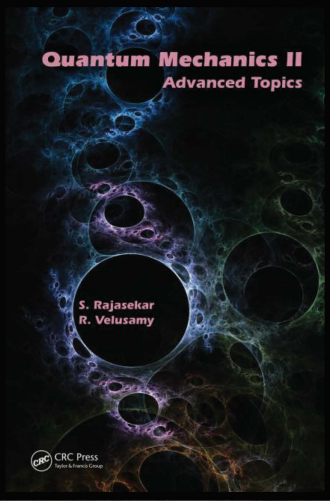
CRC Press
Taylor & Francis Group

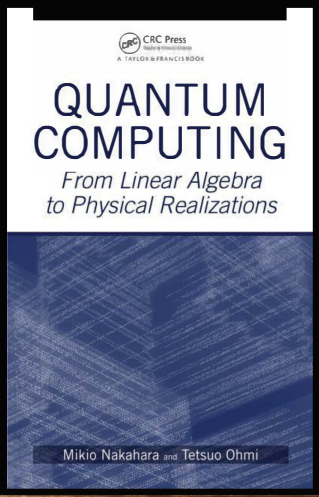
TABLE OF CONTENTS

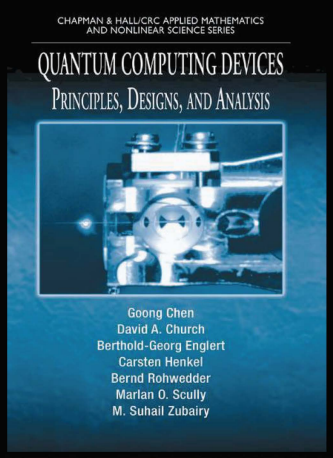
READ THE LATEST ON QUANTUM COMPUTING WITH THESE KEY TITLES

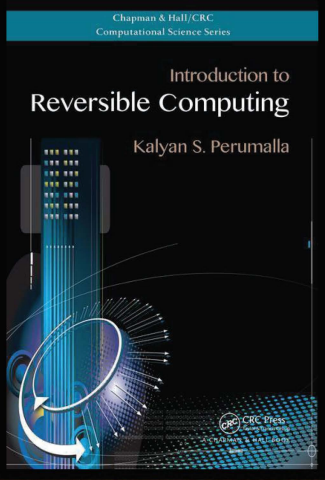












Introduction

CRC Press/Taylor and Francis is a leading publisher in all areas of computing, with a special focus on computational science. We are actively seeking new book projects in the area of quantum computing. Please contact Randi Cohen, Publisher, for more information or to submit a book proposal – Randi.cohen@taylorandfrancis.com.

Continuing to offer an exceptionally clear, up-to-date treatment of the subject, **Quantum Mechanics, Sixth Edition** explains the concepts of quantum mechanics for undergraduate students in physics and related disciplines and provides the foundation necessary for other specialized courses.

Quantum Principles and Particles, Second Edition offers a unique introduction to quantum mechanics progressing gradually from elementary quantum mechanics to aspects of particle physics. It presents the microscopic world by analysis of the simplest possible quantum mechanical system (spin 1/2).

Quantum Mechanics II: Advanced Topics uses more than a decade of research and the authors' own teaching experience to expound on some of the more advanced topics and current research in quantum mechanics.

Covering both theory and progressive experiments, **Quantum Computing: From Linear Algebra to Physical Realizations** explains how and why superposition and entanglement provide the enormous computational power in quantum computing.

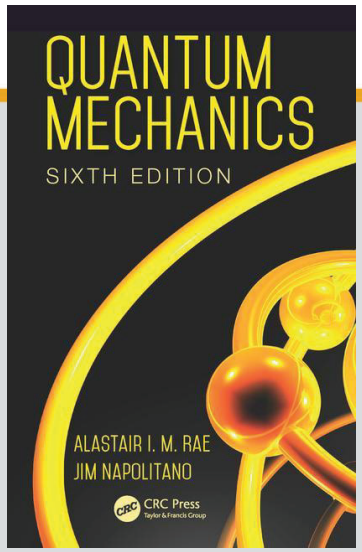
One of the first books to thoroughly examine the subject, **Quantum Computing Devices: Principles, Designs, and Analysis** covers the essential components in the design of a "real" quantum computer. It explores contemporary and important aspects of quantum computation, particularly focusing on the role of quantum electronic devices as quantum gates.

Few books comprehensively cover the software and programming aspects of reversible computing. Filling this gap, **Introduction to Reversible Computing** offers an expanded view of the field that includes the traditional energy-motivated hardware viewpoint as well as the emerging application-motivated software approach.

CHAPTER

1

QUANTUM INFORMATION



This chapter is excerpted from

Quantum Mechanics

by Alastair I. M. Rae, Jim Napolitano

Learn more

CHAPTER 15

Quantum Information

CONTENTS

This chapter aims to introduce some of the applications of the ideas of quantum mechanics that were developed during the last twenty years or so of the twentieth century. The topics chosen reflect our increased understanding of a number of areas and its application to new phenomena that have been predicted and sometimes observed. They all rely on the interplay between the two types of time dependence implicit in the fundamental postulates set out in Chapter 6:

Postulate 5 Between measurements, the development of the wave function with time is governed by the time-dependent Schrödinger equation.

Postulate 2 Immediately after a measurement, the wave function of the system will be identical to the eigenfunction corresponding to the eigenvalue obtained as a result of the measurement.

In practice, there is generally a clear distinction between "unitary evolution" under the influence of the time-dependent Schrödinger and the "collapse" associated with a measurement. In unitary evolution (so called because in a matrix representation, the initial and final states are connected by a unitary matrix) the final state of the system is completely determined by the Hamiltonian operator and the initial state. It follows that unitary evolution is reversible, because the initial state can be generated from the final state by the same time-dependent Schrödinger equation with the sign of the time coordinate reversed and ψ replaced by ψ^*. In contrast, the result of a quantum measurement is generally unpredictable: the system collapses at random

into one of a set of possible outcomes, which is an irreversible change that occurs in one time direction only. Although, in practice, we easily know when to apply which form of time dependence, it is difficult to set objective criteria for this, and we shall discuss the consequent "quantum measurement problem" in more detail in Chapter 16.

Most of our discussion in this chapter will refer to the behaviour of what have become known as "qubits." A classical bit is a system that can exist in one or other of two possible states, conventionally used to represent zero and one; a qubit is a quantum system that can exist in one of two base states, but can also be in a state that is a linear combination of these. A prime example is the spin-half particle, whose component of angular momentum relative to the z axis can have the values $\pm\frac{1}{2}\hbar$, with corresponding eigenstates α_z and β_z. As we saw in Chapter 9, the spin could also be measured relative to some other direction, producing new eigenvectors that could be expressed as linear combinations of the old. For example, if α_x and β_x respectively, represent positive and negative spin measured with respect to the x direction

$$\begin{aligned}\alpha_x &= 2^{-1/2}(\alpha_z+\beta_z)\\ \beta_x &= 2^{-1/2}(\alpha_z-\beta_z)\end{aligned} \tag{15.1}$$

cf. Chapter 9 (9.23).

Another example of a qubit is a plane-polarized photon. Plane polarization is a familiar concept in classical optics where it refers to the plane in which the E vector of the electromagnetic wave vibrates, and when experiments are done on weak light, it is found that this property can also be applied to individual photons. The two states of polarization follow very similar rules to the spin directions in the case of a spin-half particle, but the angles are halved. Thus, if the base states correspond to a photon polarized horizontally and vertically, respectively, the state of a photon polarized at $\pm 45^\circ$ to the horizontal axis can be expressed as linear combinations of the base states using (15.1), where α_z and β_z now represent photons with horizontal and vertical polarization, while α_x and β_x correspond to $\pm 45^\circ$ polarizations.

Atoms can also be used as qubits despite the fact that they generally have an infinite number of bound states! This is achieved by ensuring that they are isolated from all external influences other than radiation whose frequency matches the energy difference between two of the states. To minimize the frequency of unwanted collapses due to spontaneous transitions between the states (cf. Chapter 11), systems are normally chosen where this energy difference is very small. Qubits have also been constructed from semiconductors and from superconductors; in the latter case, the two states typically correspond to opposite directions of current flow around a superconducting ring.

Throughout this chapter, our prime example of a qubit will be the spin-half particle, because that has been discussed in detail in earlier chapters. However, practical applications often use other systems and we shall indicate this from time to time.

15.1 QUANTUM CRYPTOGRAPHY

Cryptography is the science of the exchanging of messages in a coded form that makes them indecipherable to anyone else. We shall not attempt anything like a full survey of this immense subject, but will concentrate on how the particular properties of quantum systems can be used in this area. To transmit any message at all, it must be coded in some way. The pages in this book do so using the alphabet and other symbols, as well as relying on the reader's and writer's knowledge of the English language and of mathematics. When the present edition was written using a computer with a word-processing package, the letters, figures and symbols were further encoded by the computer into a set of binary bits, represented by two symbols "1" and "0." In this way any message at all can be represented by a sequence of 1s and 0s, and any system that is capable of existing in two distinct states can be used to encode such a representation of a message. This procedure is now universally used in electronic transmission where the bits are typically represented by voltage pulses of different sizes.

The essential property of an encrypted message is that the information in it should be understandable only to the sender and the receiver, and be meaningless to a third party. This is achieved by processing the message using some procedure known as a "cypher," which can be decoded only if the reader is in possession of a "key." Early keys were in the form of code books, but modern cryptography uses mathematical procedures that depend on knowing a comparatively small key, typically a few thousand binary bits in length.

For example, a message, M, could be coded as E by the algorithm

$$E = M^s \mod c \tag{15.2}$$

where the right-hand side means that the number M is raised to the power s and the answer expressed as a number to base c. If c is a product of two prime numbers (p and q), it can be shown that the message can be decoded by the algorithm

$$M = E^t \mod c \tag{15.3}$$

where t is a simple function of p and q. However, if p and q are not known, the message is secure and all the other quantities can be exchanged publicly. If c is large enough, breaking the code by searching for its prime factors would take a conventional computer an impossibly long time (more than 10^6 years if c is 1000 binary digits in length). As we shall see later in this chapter, this factorization problem is one where quantum computers could, in theory, make a dramatic contribution, but it may well be 10^6 years before this technology is available!

The present section describes a method whereby a sender and receiver can both acquire knowledge of a number to be used as a key to a code, while being sure that no eavesdropper knows it. As we shall see, it cannot be directly used to transmit previously defined numbers like p and q, because even the sender does not know in advance what the shared number is going to be. However, the commonly held information can be used to decide on which of a number of previously agreed protocols is to be used; e.g. which pair of primes contained in a previously prepared

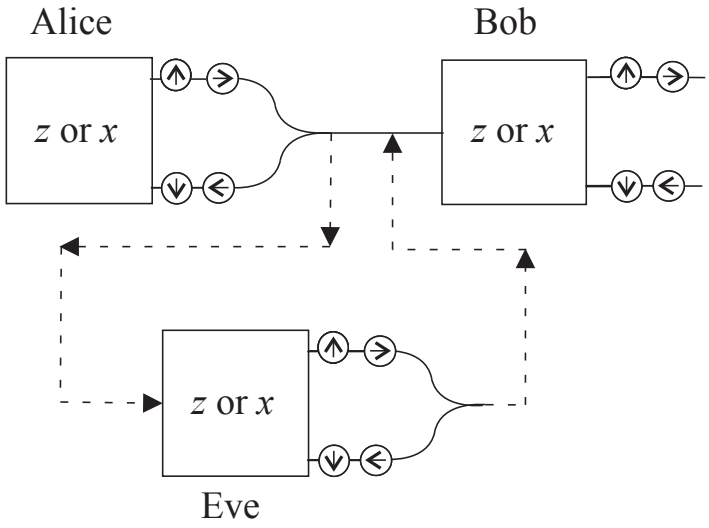


FIGURE 15.1 Alice sends a series of qubits to Bob, which are α_z/β_z or α_x/β_x at random. Bob makes a similar set of randomly selected x or z measurements on the qubits he receives. They then exchange information on the measurement orientations they have chosen and use the results for the subset where these are the same, to define the key to be used. In the absence of Eve's intervention, Alice and Bob have the same key, but this no longer holds if Eve has attempted to make a measurement.

list are to be used as p and q. In this way they can both follow the same coding process, even if they do not know what this is going to be before the exchange takes place. This key distribution process is where quantum mechanics comes in.

A sender (conventionally known as "Alice") transmits a message to a receiver ("Bob") in the form of a stream of qubits where, say, α_z and β_z represent 0 and 1, respectively. As we shall see, the specifically quantum properties of the particles allow us to detect the presence of an eavesdropper ("Eve") who is intercepting the message and retransmitting it to Bob while hoping to do so undetected. The setup is illustrated in Figure 15.1

The central idea is that Alice sends a sequence of qubits where some are in the form α_z or β_z, but others are represented by α_x or β_x. Both α_z and α_x are to be interpreted as 0, while β_z and β_x represent 1. Which orientations Alice uses are decided at random, but she keeps a record of them. When Bob detects the qubits, he also varies the orientation of his apparatus between the z and x directions at random and records which apparatus he uses, as well as the results he obtains. Alice and Bob then publicly tell each other which settings they used in each case and discard those that were not the same—on average half the total. The remainder consists of a set of numbers whose values they both know, and which they can then use to decide which procedure they will follow to encode and decode messages sent openly between them.

We now consider the effect of the possible intervention of Eve. She may intercept the qubits sent by Alice, but she has no means of knowing the orientation of Alice's

TABLE 15.1 The properties of a typical set of 12 qubits analyzed by the apparatus shown in Figure 15.1. The asterisks in the final column mark cases where Eve's intervention has resulted in Bob having the wrong result. (O: apparatus orientation; S: state of sent qubit; M: message; R: state of received qubit; a/r: accepted or rejected; K agreed common key; K′: key corrupted by Eve)

Alice			Bob (Eve inactive)				Eve		Bob (Eve active)	
O	S	M	O	R	a/r	K	O	R/S	R	K′
z	α	0	z	α	a	0	x	α	β	1*
z	β	1	z	β	a	1	x	β	α	0*
x	α	0	z	α	r		z	β	β	
z	β	1	x	α	r		x	α	α	
x	α	0	x	α	a	0	x	α	α	0
x	β	1	z	β	r		x	β	β	
z	β	1	x	β	r		z	β	β	
x	β	1	z	β	r		z	β	β	
x	α	0	x	α	a	0	x	α	α	0
x	α	0	z	α	r		z	α	α	
z	α	0	z	α	a	0	x	α	β	1*
z	β	1	z	β	a	1	z	β	β	1

apparatus when they were sent. The simplest thing she can do is randomly guess which orientation might have been used for a particular qubit, set her apparatus in this orientation, record the result, and send a qubit in the same state as that just recorded on to Bob. On average, she will guess right one-half of the time, but there is only a 50% probability that this guess will be the same as Bob's. The upshot is that about one-half of the qubits that Bob later believes are reliable, have been corrupted by passing through Eve's apparatus set in the "wrong" orientation. About one-half of these (i.e., one-quarter of the final sequence used as the key) will decode as bits that are the opposite of those sent by Alice. Alice and Bob therefore no longer have a set of numbers in common and soon find they cannot decode each other's messages. They have therefore detected the presence of Eve and can take appropriate action.

Worked Example 15.1 Illustrate the above procedure by considering the processing of a string of twelve qubits sent by Alice to Bob in (i) the presence and (ii) the absence of Eve.
Solution See Table 15.1.

This protocol succeeds because of the collapse resulting from a quantum measurement. A necessary concomitant of this is the system's vulnerability to random processes (noise). Even in the absence of Eve, Alice and Bob will successfully end up with a common key only if their apparatuses are accurately aligned when they believe they are, and if the polarization state has not been changed (e.g., rotated slightly) during the transmission. Moreover, even though Eve cannot accurately read the message and her attempt to do so can be detected, her presence apparently makes the channel of communication useless. There are strategies for

combatting this by further public exchange of information, as a result of which Alice and Bob can end up with a shared key considerably shorter than the length of the shared message, but with a very low probability of Eve also knowing it.

Unlike the cases to be discussed in the rest of this chapter, quantum cryptography is reasonably easy to carry out experimentally. Secure key interchange has been demonstrated by sending polarized photons over distances of up to about 100 km of standard optical communication fibre and there is active research aimed at extending this to worldwide distances.

15.2 ENTANGLEMENT

The remaining examples discussed in this chapter involve two or more particles in what is called an "entangled state." In quantum mechanics, the word "entanglement" refers to a quantum state of two or more particles, where the probabilities of the outcome of measurements on one of them depend on the state of the other—even though there is no interaction between them.

As an example of entanglement, consider two spin-half particles in a state where they are far apart, their spins are opposite and we have no other information about them. The spin part of their wave function has the form

$$\psi(1,2) = 2^{-1/2}[\alpha_z(1)\beta_z(2) - \beta_z(1)\alpha_z(2)] \tag{15.4}$$

where $\alpha_z(i)$ and $\beta_z(i)$ represent particle i with a positive and negative spin component, respectively, relative to the z axis. However, the only information this wavefunction contains is that the spins are opposite—it does not tell us the absolute direction of either spin. This is because $\psi(1,2)$ is independent of the direction of the axis of quantization: to see this, we apply the transformation (15.1) to express $\psi(1,2)$ in terms of $\alpha_x(i)$ and $\beta_x(i)$ and get

$$\begin{aligned}\psi(1,2) &= 2^{-3/2}\Big([\alpha_x(1)+\beta_x(1)][\alpha_x(2)-\beta_x(2)] - [\alpha_x(1)-\beta_x(1)][\alpha_x(2)+\beta_x(2)]\Big) \\ &= -2^{-1/2}[\alpha_x(1)\beta_x(2) - \beta_x(1)\alpha_x(2)]\end{aligned} \tag{15.5}$$

which is the same as (15.4) with z replaced by x, apart from an irrelevant change of sign. We can therefore drop the suffixes on α and β when we describe qubits entangled in this way.

We now consider the effect of performing a measurement of the z component of either of the spins when the system is in the state $\psi(1,2)$. Clearly we expect to get a positive or a negative result at random with equal probability. But suppose we measure the spin of particle 2 *after* we have measured the spin of particle 1 and obtained (say) a positive result. As a result of this first measurement, the system will have collapsed into an eigenstate of the spin-component operator with positive eigenvalue. It therefore has the form

$$\psi(1,2) = \alpha_z(1)\beta_z(2) \tag{15.6}$$

and we now know that the particles are in eigenstates of $\hat{S}_z$, because the state has

been disentangled. A measurement of the z component of the spin of the second particle can now only yield a negative result, whereas there would have been equal probability of a positive or negative result if we had made this measurement before that on particle 1. It follows that the probabilities of obtaining particular values of the spin of one particle can depend on what measurements have been previously carried out on the other. It should be noted that this result is independent of other properties of the particles, in particular their position: particles that have interacted and become entangled can maintain this be entanglement even when they have moved a long distance apart. This apparent influence of the operations carried out on one particle on the properties of a distant particle is known as "nonlocality" and we discuss its implications for our understanding of the conceptual basis of quantum mechanics in Chapter 16.

Various techniques for creating entangled states have been developed in recent years. Probably the most popular is now the application of "parametric down conversion." This involves the use of nonlinear optics in which a photon incident on a crystal is converted into two photons, the sum of whose frequencies equals that of the original photon (so that energy is conserved); the two emitted photons are found to have perpendicular polarizations, although their absolute polarization is unknown.

15.3 CLONING AND TELEPORTATION

"Cloning" is a word used in biology to describe the creation of a living organism that has identical genetic make up to another existing organism. In quantum physics, the term refers to attempts to transfer the quantum state of one "reference" system to another "target" system, without relying on prior knowledge of the reference state and without disturbing it in the process. As we shall see, quantum cloning defined in this way is impossible. To prove this, we consider two quantum systems with the same composition; they might be qubits, but they could be more complex systems such as two hydrogen atoms. Initially the target (system 1) is in a state $\phi(1)$ and we are looking for a general procedure that will transform its state to be the same as the state, $\psi(2)$, of the reference system (system 2) without affecting the state of system 2 and without relying on any prior knowledge of the state of system 2. That is, we are looking for a procedure that will perform the following transformations on the composite state of the two systems

$$\phi(1)\psi(2) \rightarrow \psi(1)\psi(2) \tag{15.7}$$

where $\psi(2)$ is any allowed state of system 2.

First consider the effect of making a measurement on this system. By definition, cloning requires (15.7) to hold whatever the state of system (2), including states that are not eigenstates of the measurement operator. However, all measurements result in the wave function collapsing into an eigenstate of the measurement operator, which in general results in a change in the state of system 2. As we have defined cloning

as a process that operates *whatever* state is represented by $\psi(2)$, we conclude that cloning by measurement is impossible.

Now suppose that we allow the composite system to evolve in a unitary manner following the time-dependent Schrödinger equation under the influence of some Hamiltonian $H(\hat{1},2)$. We consider two of the possible states represented by $\psi(2)$ and label these as $\psi_A(2)$ and $\psi_B(2)$. Substituting these in turn into (15.7), we take the product of one of the resulting left-hand sides with complex conjugate of that of the other. The time dependence of this product is then:

$$\begin{aligned}\frac{\partial}{\partial t}\int\int \phi^*(1)\psi_A^*(2)\phi(1)\psi_B(2)d\tau_1 d\tau_2 = \\ -(i/\hbar)\int \phi^*(1)\psi_A^*(2)H(\hat{1},2)\phi(1)\psi_B(2)d\tau_1 d\tau_2 \\ +(i/\hbar)\int\int \phi(1)\psi_B(2)\hat{H}^*(1,2)\phi^*(1)\psi_A^*(2)d\tau_1 d\tau_2 \\ =0 \end{aligned} \tag{15.8}$$

where we have used the time-dependent Schrodinger equation and its complex conjugate—c.f. (11.1) and (11.2)— along with the fact that $\hat{H}$ is Hermitian. (15.8) means that the integral of the product of the wave functions does not change with time. It therefore follows from (15.8) that if (15.7) holds

$$\int\int \phi^*(1)\psi_A^*(2)\phi(1)\psi_B(2)d\tau_1 d\tau_2 = \int\int \psi_A^*(1)\psi_A^*(2)\psi_B(1)\psi_B(2)d\tau_1 d\tau_2 \tag{15.9}$$

or

$$\int \psi_A^*\psi_B d\tau = \left[\int \psi_A^*\psi_B d\tau\right]^2 \tag{15.10}$$

This is true only in the special cases where the integral on the left-hand side is zero or unity, which means that there is no such general procedure and the no-cloning theorem is proved.

If the no-cloning theorem were not true, we could in principle create a large number of cloned systems whose properties could be measured without significantly affecting the ensemble as a whole and this would potentially be in breach of the uncertainty principle. Suppose, for example, we have a spin-half particle in a state where the direction of spin is unknown. If a large number of identical copies of these could be made by cloning, then all three components of spin could be accurately measured on the resulting ensemble, even though the operators representing them do not commute.

Teleportation

The word "teleportation" entered the English language via the science-fiction television series *Star Trek*. In this fantasy, a person or object could be transported to a distant destination using a transmitter that measured all the properties of the object

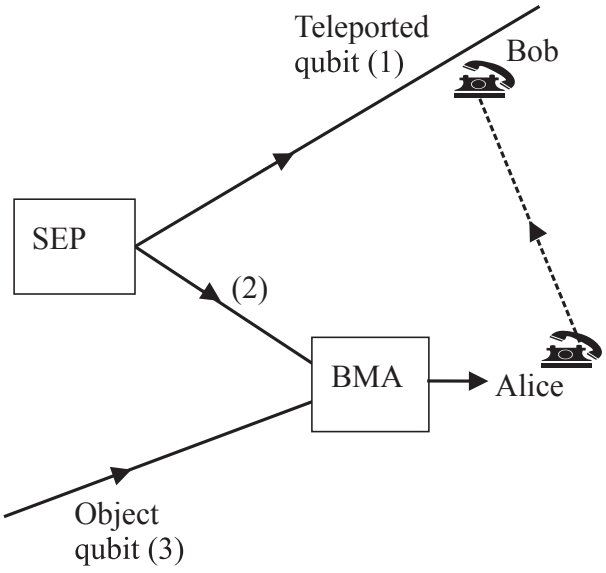


FIGURE 15.2 Quantum teleportation. SEP represents a source of entangled pairs of particles. One member of a pair is sent to Bob while the other is sent to Alice, who allows it to interact with the object qubit and uses the Bell measurement apparatus (BMA) to project their state into one of the four Bell states. Alice then uses a classical communication channel to tell Bob which of these results she obtained. Bob can then transform the state of the teleported qubit into one identical to the initial state of the object qubit.

and sent them (presumably by some radio link) to a receiver that reassembled the information to recreate the object. In the process the original object was destroyed (dematerialized?). In the quantum analogue, teleportation is the same as quantum cloning except that the state of the reference system is inevitably changed in the process. We consider, as a simple example, an object qubit, such as a spin-half particle, in a state given by

$$\psi(3) = A\alpha(3) + B\beta(3) \tag{15.11}$$

where the values of A and B are unknown. An experimentalist (Alice, of course) wants to transfer these values to a target qubit (1), which she is to send to another experimenter (Bob). She first performs an experiment to entangle qubit (1) with another qubit (2) so that their quantum state has the form given by (15.4). The total wave function of all three particles is therefore

$$\Psi(1,2,3) = 2^{-1/2}[A\alpha(3) + B\beta(3)][\alpha(1)\beta(2) - \beta(1)\alpha(2)] \tag{15.12}$$

This is algebraically identical to the expression

$$\Psi(1,2,3) = -\frac{1}{2}[A\alpha(1) + B\beta(1)]\psi_1(2,3) + \frac{1}{2}[A\alpha(1) - B\beta(1)]\psi_2(2,3)$$
$$-\frac{1}{2}[A\beta(1) + B\alpha(1)]\psi_3(2,3) - \frac{1}{2}[A\beta(1) - B\alpha(1)]\psi_4(2,3) \qquad (15.13)$$

where

$$\psi_1(2,3) = 2^{-1/2}[\alpha(2)\beta(3) - \beta(2)\alpha(3)]$$
$$\psi_2(2,3) = 2^{-1/2}[\alpha(2)\beta(3) + \beta(2)\alpha(3)]$$
$$\psi_3(2,3) = 2^{-1/2}[\alpha(2)\alpha(3) - \beta(2)\beta(3)]$$
$$\psi_4(2,3) = 2^{-1/2}[\alpha(2)\alpha(3) + \beta(2)\beta(3)] \qquad (15.14)$$

as can be checked by expanding the right-hand sides of (15.12) and (15.13). This set of states can be shown to have the maximum possible entanglement for two spin-half particles and are known as "Bell states" (after John Bell—see Chapter 16) .

Alice sends particle 1 to Bob and then performs a measurement on particles 2 and 3. The essential feature of this "Bell state measurement" is that it is represented by an operator whose eigenfunctions are the four ψ_is, so that the whole system, including particle 1, collapses into one of these states. Alice communicates the result of this to Bob, using a *classical* channel of communication—cf. Figure 15.2. Thus, if Alice obtains the result corresponding to ψ_1, Bob's particle (1) must have the same spin function as was originally possessed by particle 3, which of course was Alice's original intention. If she obtains one of the other results and lets Bob know, then Bob can transform the state function into that required, simply by rotating the spin using a magnetic field B for an appropriate time. For example, to generate the second square-bracketed expression in (15.13) from the first, we use (11.14) in Chapter 11 to get

$$A(t) = A(0)\exp(-\frac{1}{2}i\omega_p t) \text{ and } B(t) = B(0)\exp(\frac{1}{2}i\omega_p t) \qquad (15.15)$$

where $\omega_p = eB/m$. We get the required result (apart from an irrelevant phase factor) if $t = \pi m/eB$. We emphasize that teleportation transfers the full quantum state of the object qubit to the target, but, as always in quantum mechanics, only part of this is revealed in any single measurement. However, the statistical properties of the teleported state of the target are predicted to be the same as those of the object qubit in the absence of teleportation, and this has been verified experimentally.

A major challenge to the practical realization of quantum teleportation is making a measurement that collapses the system into one of the Bell states. If, for example, we were simply to measure a component of spin of one of the particles, we would cause the system to collapse into the corresponding eigenstate, which is not one of the Bell states. To perform a Bell-state measurement successfully, we must address both qubits 2 and 3 simultaneously and execute a measurement that is sensitive to their relative properties, but does not produce any information about their individual

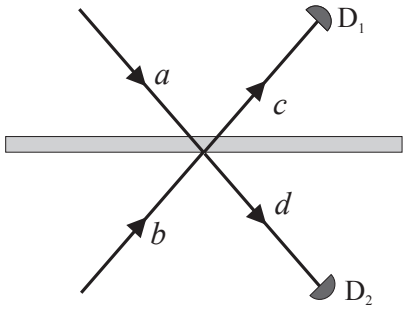


FIGURE 15.3 Two photons (a and b) are simultaneously incident on a semitransparent mirror where each is reflected and/or transmitted with equal probability, and then detected by one of the detectors, D. If photons are simultaneously detected in both detectors, this constitutes a Bell-state measurement.

states. Figure 15.3 illustrates how this could be partially achieved for two qubits, which in practice are normally photons. Qubits 2 and 3 are represented by polarized photons, which strike a semitransparent mirror simultaneously (i.e., within a time short compared with the length of the associated wave packets) from opposite sides. To understand how this can lead to a Bell-state measurement, we have to go beyond the spin states set out in (15.14) and include a representation of the spatial parts of the wave functions. Referring to figure 15.3, we denote the spatial parts of the wave functions of the two beams approaching the beam splitter by a and b. As photons are bosons, the total wave function (15.13) must be symmetric with respect to exchange of labels 2 and 3. Referring to (15.14), we see that the spin function ψ_1 is antisymmetric and the others are symmetric. It follows that when constructing the total wave function the $\psi_i(2,3)$ must be multiplied by

$$2^{-1/2}\,[a(2)b(3) \pm b(2)a(3)] \tag{15.16}$$

where the negative sign applies when $i = 1$ and the positive sign for $i = 2,3,4$.

We now consider the effect of the semitransparent mirror from which the photons emerge in states c and d—see Figure 15.3. Each photon has an equal probability of transmission or reflection, so it is impossible to tell which path was followed by any particular detected photon. The wave function in beam c is made up of a reflected component that originated in beam a and a transmitted component that originated in b. Similarly, d is a superposition of a transmitted part from a and a reflected part from b; moreover, because this last reflection was at the interface with an optically denser medium, a change of sign is introduced at this point. We represent these relations by

$$\begin{aligned} a &\rightarrow 2^{-\frac{1}{2}}\,(c+d) \\ b &\rightarrow 2^{-\frac{1}{2}}\,(c-d) \end{aligned} \tag{15.17}$$

After substitution from (15.17) the expressions in (15.16) become

$$\begin{aligned} & 2^{-1/2}\,[d(2)c(3) - c(2)d(3)]\text{, if } i = 1 \\ \text{and} \quad & 2^{-1/2}\,[c(2)c(3) - d(2)d(3)]\text{, otherwise.} \end{aligned} \tag{15.18}$$

It follows that, if we detect one photon in detector D_1 and the other in D_2, then one must have been in c and the other in d, so only the first alternative can apply. This means that photons 2 and 3 have been successfully measured to be in the first Bell state and the initial state of photon 3 has been successfully teleported to photon 1. However, if both photons are detected in either D_1 or D_2 no Bell state is uniquely determined by this measurement and no teleportation is possible. As ψ_1, is one of four equally weighted Bell states that make up the original wave function, teleportation occurs only 25% of the time. When a photon is registered in each of the two detectors, Alice can tell Bob that the properties of the object photon have been successfully teleported to his target photon. More sophisticated experimental arrangements, with four possible outcomes, each of which corresponds to one of the Bell states, have been devised. One of these is based on the parametric up-conversion of two photons into a single photon followed by manipulation of its state.

We note that a classical communication channel by which Alice communicates the result of her measurement to Bob is an essential part of the experiment and, as she must identify one of four possibilities, two bits of classical information must be transmitted in this way. This is an example of a general principle that forbids the transmission of information using entanglement alone. Otherwise we could transmit information instantaneously in breach of the principles of special relativity. We shall return to this point in Chapter 16.

The principles of quantum teleportation have been experimentally demonstrated for polarized photons and also using atomic beams. However, this is a long way from *Star Trek*! Before any significantly complex object could be teleported, it would have to be coupled in some way to a system consisting of as many entangled particle pairs as the number of parameters needed to define the reference state. The practicability of this is certainly well beyond our present technology.

15.4 QUANTUM COMPUTING

A conventional computer manipulates binary bits according to a set of rules, the operation performed on one bit often depending on the state of one or more of the others. In a "quantum computer" the binary bits are qubits and the operations proceed by unitary evolution (as discussed earlier in this chapter, this means that the states of the qubits evolve according to the time-dependent Schrödinger equation) until we make measurements on them. All the basic operations of a classical computer can be carried out using a quantum computer, though sometimes more elaborately for reasons to be discussed later. There is, therefore, no obstacle *in principle* to the construction of a quantum computer that would carry out the same computations as a classical computer. However, the emphasis must be on the phrase "in principle." As we shall see, a useful quantum computation involves the entanglement of a

significant number of qubits; such entangled states are extremely sensitive to noise and decoherence, so the practical obstacles to constructing a useful device are immense, and overcoming them would almost certainly involve the development of a completely new technology.

Of course, there is no reason why anyone would go to such trouble to construct something that could do no more than a conventional computer. The great interest in quantum computers is that, although they would carry out most computing tasks no more efficiently than conventional machines, there are some processes that would be performed at an immensely faster speed. An often-quoted example is the factorization of a large number into its prime-number components by a method known as Shor's algorithm, after the person who devised it; as mentioned earlier, if this operation could be efficiently performed it would endanger the security of some commonly used cryptographic protocols. The factorization of a number consisting of N decimal digits using a conventional computer requires a number of computing steps that is proportional to $\exp(2L^{1/3}\ln L^{2/3})$, where $L = \ln N$. In contrast, the number of operations required by a quantum computer has been shown to be about $300L^3$. For L equal to 10, the classical algorithm is faster, but the quantum computer takes over for larger numbers: for $L = 200$, the calculation would take 10^9 years classically, but could be performed by a quantum computer in about 8 hours.

As a first step to understanding the principles of quantum computation, we consider a particular simple operation, known as a NOT gate. This has the property of reversing the state of a bit: if we input 0 we get out 1, while if we input 1 we get out 0. (Because it acts on one bit at a time, a NOT gate is an example of a "single-bit" gate). It is very easy to devise a unitary operation that will carry out this procedure on a qubit. We represent 0 and 1 by the spin-up (α_z) and spin-down (β_z) states of a spin-half particle, which we subject to a field directed in the y direction. The spin precesses according to (11.17) in Chapter 11 and if we apply the field for the correct length of time, its direction will be reversed: i.e. $\alpha_z \rightarrow \beta_z$ and $\beta_z \rightarrow -\alpha_z$, which is just what we expect from the operation of a NOT gate (NB, the sign change is a result of the phase change associated with spin rotation discussed in Chapter 11 and is not relevant to the present discussion).

Suppose now we perform the same operation, but starting from a state where the spin is not in an eigenstate of S_z, but in one that is a linear combination of α_z and β_z. (To simplify the notation, from now on we shall drop the subscript z when referring to eigenstates of spin measured along the z axis.) We now get

$$A\alpha + B\beta \rightarrow A\beta - B\alpha \tag{15.19}$$

We see that both the operations $\alpha \rightarrow \beta$ and $\beta \rightarrow -\alpha$ are now contained in this single operation. This procedure is central to quantum computation and is known as the operation of a "Hadamard gate."

A single NOT gate can be considered as a very simple computer, programmed to perform a single operation. If this were a conventional device, we could fully determine its properties only by running it twice: once with each of its possible inputs (0 or 1). In the quantum case, however, if we run the program once only using, say,

α_x as input, then the wave function of the output is a linear combination of both outcomes. Thus, both calculations have been performed in one step!

We might be forgiven for being less than impressed by this, when we realize that we could do the same thing by inverting a classical pointer held at an angle to the horizontal—because reversing the direction of any vector reverses the direction of all its components. In one sense, a quantum computer has much in common with a classical analogue computer, but the power of a quantum computer is realized only when it operates on a quantum state made up from a number of qubits simultaneously, and this could be modelled classically only if our pointer existed in a many dimensional space —2^n dimensions in the case of n qubits.

An important restriction on the power of quantum computing is that, in order to access the result of a calculation, we must make a measurement and this inevitably causes wave function collapse. This means that in the case of the simple example above, although both α and β are operated on simultaneously, only one of these components can be measured in one operation, so there is apparently no practical gain in performing the calculation in this way. For this reason, quantum computation often has little or no advantage over conventional methods. However, there are important exceptions to this rule and these relate to cases where the aim is to extract a single piece, or a small number of pieces, of information that would require a long and complex computation to determine classically. One example is searching a large database for a single match and another is the determination of the period of a periodic function. (For reasons we shall not go into, the latter is a key part of the algorithm to factorize a product of two prime numbers mentioned above). We shall give an outline of the principles of how the latter calculation could be carried out, provided the correct unitary operations could be performed and the appropriate measurements made. We shall see that a subtle interplay of unitary evolution and collapse can be employed to perform this calculation with very high efficiency

We first note a further important constraint on the operation of a quantum computer. This arises from the fact that the Schrödinger equation is time-reversible, by which we mean that if we make the substitution $t \rightarrow -t$, the equation is still valid. Thus, if a system undergoes a unitary evolution from one state to another, it must also be able to reverse the process in a similar manner; an example of this is the rotation of a spin state in a magnetic field as discussed above, which can be reversed by applying a field of the same strength and for the same time, but in the opposite direction. A corollary of this is that each operation must have the same number of outputs as it has inputs. To understand this, think of an operation where the input consists of two numbers and the output consists of their sum: it is impossible to reconstruct the original pair uniquely if the only information we have is the final result. This does not mean that a quantum computer cannot perform an addition, but the process is more complex than that employed classically and involves retaining more information.

Suppose we have a quantum computer that contains a number of qubits, which we shall continue to think of as spin-half particles. We shall use n of these to construct what is known as a *register* that can be used to represent all numbers from 0 to $2^n - 1$. We first want to set our register to zero, which means that all qubits must be in the state α; this can be achieved by applying a magnetic field in the z direction

and allowing the spins to relax into their ground state. (NB: this is an example of an irreversible measurement). Now consider the result of applying a magnetic field in the y direction for a time sufficient to rotate the spins through 90 degrees so that they are all in the state α_x; in the case where $n = 3$, this state is

$$\begin{aligned}\chi &= \alpha_x(1)\alpha_x(2)\alpha_x(3)\\ &= 2^{-3/2}(\alpha_1+\beta_1)(\alpha_2+\beta_2)(\alpha_3+\beta_3)\\ &= 2^{-3/2}[\alpha_1\alpha_2\alpha_3+\alpha_1\alpha_2\beta_3+\alpha_1\beta_2\alpha_3+\alpha_1\beta_2\beta_3\\ &\quad +\beta_1\alpha_2\alpha_3+\beta_1\alpha_2\beta_3+\beta_1\beta_2\alpha_3+\beta_1\beta_2\beta_3] \end{aligned} \tag{15.20}$$

where $\alpha_x(i)$ represents atom i in state α_x and we have made a small change in notation from the second line onwards, where α_i represents atom i in state α_z. These final eight combinations of α and β are just the binary representations of the numbers zero to seven, so generalizing, we have

$$|\chi\rangle = N^{-\frac{1}{2}} \sum_{j=0}^{N-1} |j\rangle \tag{15.21}$$

where $N = 2^N$ and $|j\rangle$ represents the state of the three qubits equivalent to the binary representation of the number j; our example treated the case where $N = 8$, but (15.21) holds for any number of qubits. We see therefore that the application of the magnetic field has transformed the state from one representing the number zero, to one that is a linear combination of the representations of all the numbers from zero to seven. We note that this operation is unitary, because the system is governed by the time-dependent Schrödinger equation. It is also reversible as the original state can be restored simply by applying a field in the opposite direction.

We shall call the register just discussed the x-register and we now suppose that we have another register (the y-register) containing two qubits (numbered 4, 5), which are initially in the state α. We now perform what is called a *controlled NOT* (or CNOT) operation; this acts on the state of a particular qubit in the y-register (known as the "target qubit") leaving it alone or reversing its state, depending on the state of a particular qubit in the x register (the "control qubit"). In particular, if qubit(2) is zero, qubit(4) remains zero, while if qubit(2) is one qubit(4) is changed to one. (In principle this could be achieved by focusing the magnetic field created by the magnetic dipole associated with qubit(2) onto qubit(4) and applying a further field to cancel this out or reinforce it, depending on whether qubit(2) is in the state α or β, respectively.) The state of qubit(5) is subjected to the same procedure, using qubit(3) as the control. Because the state of the first three qubits is the linear combination (15.21), the five qubits end up in an entangled state that can be expressed as

$$\begin{aligned}|\psi\rangle &= N^{-\frac{1}{2}}(|0\rangle|0\rangle+|1\rangle|1\rangle+|2\rangle|2\rangle+|3\rangle|3\rangle+|4\rangle|0\rangle+|5\rangle|1\rangle+|6\rangle|2\rangle+|7\rangle|3\rangle\\ &= N^{-\frac{1}{2}}\sum_{j=0}^{N-1}|j\rangle|y_j\rangle)\end{aligned} \tag{15.22}$$

where the first and second kets in a product represents the state of the x and y registers, respectively, and $|y_j\rangle$ is defined by the first line. Because the last four states of the y-register are the same as the first four, we have an example of a function that is periodic in the variable defined by the values in the x-register, and this has been created by unitary operations on a set of qubits.

The above is a simple example of the generation of a periodic function by performing unitary operations on qubits. Moreover, we know in advance what its periodicity is going to be. A more interesting case is where a state similar to (15.22) is created by some other process, such that we do not know what the period is. In this case we would expect to be able to determine its period by subjecting it to a Fourier transform: this will have maxima corresponding to its fundamental frequency and/or some or all of its harmonics. It turns out that this period can be found very efficiently by a quantum computation as we shall now explain.

Worked Example 15.2 If the y register contains a single qubit, explain how to construct a state similar to (15.22), but with period 2.

Solution We first place the qubit in state α, then subject it to a controlled NOT, where the control bit is qubit 3 of the x register. This produces the state

$$|\psi\rangle = N^{-\frac{1}{2}}(|0\rangle|0\rangle + |1\rangle|1\rangle + |2\rangle|0\rangle + |3\rangle|1\rangle + |4\rangle|0\rangle + |5\rangle|1\rangle + |6\rangle|0\rangle + |7\rangle|1\rangle$$

so the state of the y register has period 2 as required.

The quantum Fourier transform

Classically, the Fourier transform $\tilde{y}_k$ of a function y_j can be written as

$$\tilde{y}_k = N^{-\frac{1}{2}} \sum_{j=0}^{N-1} \exp\left(\frac{2\pi i}{N} jk\right) y_j \tag{15.23}$$

where the functions $\tilde{y}_k$ and y_j are evaluated at the points $k = 0, 1...(N-1)$ and $j = 0, 1...(N-1)$, respectively. We note that this expression is generally applicable, provided the spaces spanned by j and k are appropriately scaled. It is convenient to assume that $N = 2^n$ where n is an integer and we shall do so from now on. If y_j is periodic with period r and $N/r = m$ where m is an integer, (15.23) becomes

$$\begin{aligned}
\tilde{y}_k =& N^{-\frac{1}{2}} \sum_{l=0}^{m-1} \sum_{j=0}^{r-1} \exp\left(\frac{2\pi i}{N} k(j+lr)\right) y_{j+lr} \\
=& N^{-\frac{1}{2}} \sum_{l=0}^{m-1} \exp\left(\frac{2\pi i}{N} klr\right) \sum_{j=0}^{r-1} \exp\left(\frac{2\pi i}{N} jk\right) y_j \\
=& N^{-\frac{1}{2}} \left[\frac{1-\exp(2\pi ik))}{1-\exp\left(\frac{2\pi ik}{m}\right)}\right] \sum_{j=0}^{r-1} \exp\left(\frac{2\pi i}{N} jk\right) y_j
\end{aligned} \tag{15.24}$$

where the expression in square brackets equals the sum over l in the line above because the latter is a geometric series. Unless $k = pm$, where p is an integer $\leq r$, this term equals zero, because the numerator is zero and the denominator is finite. However, if $k = pm$ both are zero and their ratio equals m. If N/r is not an integer, these statements are true to a good approximation provided $N >> r$. It follows that the period r can be determined from the positions of the peaks in the Fourier transform. Classical "fast Fourier transform" methods have been developed that allow this calculation to be performed with considerable efficiency, but this could be greatly exceeded by a quantum calculation as we shall see.

Consider the quantum transformation

$$|j\rangle \rightarrow N^{-\frac{1}{2}} \sum_{k=0}^{N-1} \exp\left(\frac{2\pi i}{N} jk\right)|k\rangle \tag{15.25}$$

where each of $|j\rangle$ and $|k\rangle$ describe the states of N qubits. If $|\psi\rangle$ represents the state of two registers entangled as in (15.22) and we apply the transformation (15.25) to the x-register only, we get

$$\begin{aligned}
|\psi\rangle &= N^{-\frac{1}{2}} \sum_{j=0}^{N-1} |j\rangle|y_j\rangle \\
&\rightarrow N^{-\frac{1}{2}} \sum_{j=0}^{N-1} \left(N^{-\frac{1}{2}} \sum_{k=0}^{N-1} \exp\left(\frac{2\pi i}{N} jk\right)|k\rangle \right)|y_j\rangle \\
&= N^{-\frac{1}{2}} \sum_{k=0}^{N-1} \left(N^{-\frac{1}{2}} \sum_{j=0}^{N-1} \exp\left(\frac{2\pi i}{N} jk\right)|y_j\rangle \right)|k\rangle \\
&= \sum_{k=0}^{N-1} |\tilde{y}_k\rangle|k\rangle
\end{aligned} \tag{15.26}$$

where

$$|\tilde{y}_k\rangle = N^{-1} \sum_{j=0}^{N-1} \exp\left(\frac{2\pi i}{N} jk\right)|y_j\rangle \tag{15.27}$$

$|\tilde{y}_k\rangle$ represents the state of the qubits in the y register, which are now entangled with the transformed states, $|k\rangle$, of the x-register. We note that each $|\tilde{y}_k\rangle$ is a linear combination of n products of the spin eigenfunctions, while $|k\rangle$ is a single product that represents the number k. Comparing (15.27) with (15.23), we see that $|\tilde{y}_k\rangle$ is a representation of the Fourier transform of $|y_j\rangle$. We can also use the fact that y_j is periodic so that $|y_{j+r}\rangle = |y_j\rangle$ to follow a similar procedure as we did for the classical case above and write (15.27) as

$$|\tilde{y}_k\rangle = N^{-1} \left[\frac{1 - \exp(2\pi ik)}{1 - \exp\left(\frac{2\pi ik}{m}\right)} \right] \sum_{j=0}^{r-1} \exp\left(\frac{2\pi i}{N} jk\right)|y_j\rangle \tag{15.28}$$

We may now think that we have calculated the whole Fourier transform of $|y_j\rangle$ in a single operation. However, as is the case for all quantum computations, only a small part of the outcome can be extracted by a single measurement. Suppose we were to attempt to measure the state of the y-register by measuring the z components of its component atoms; this would cause the state to collapse into a single product of spin eigenfunctions, which would give us little or no information about the previous state of the y-register. Remembering that the single piece of information we are seeking is the period, r, of the function y_k, we shall show that this can be obtained, not by operating on $|\tilde{y}_k\rangle$, but by measuring the state of the x-register, which contains the values of k. The probability of obtaining a particular value of k following such a measurement is $|\langle k|\psi\rangle|^2 = \langle\tilde{y}_k|\tilde{y}_k\rangle$, using (15.26). Referring to (15.28) we see that the term in square brackets is zero unless $k = pm$, where p is an integer, in which case it equals m, so that

$$\begin{aligned}\langle\tilde{y}_k|\tilde{y}_k\rangle &= \frac{m^2}{N^2}\sum_{j=0}^{r-1}\sum_{j'=0}^{r-1}\exp\left(\frac{2\pi i}{N}(j-j')k\right)\langle y_j|y_{j'}\rangle \\ &= \frac{m^2 r}{N^2} \\ &= \frac{1}{r}\end{aligned} \tag{15.29}$$

remembering that $N = mr$ and that the $|y_j\rangle$ are orthonormal. We note that the probabilities are all equal and they sum to unity, because there are r maxima less than N in the Fourier transform. We also note that this result is exact only if r is a submultiple of N, but is a good approximation in the general case, provided $N >> r$. Thus, a measurement of k is almost certain to yield a value equal to a multiple of N/r. To obtain r itself, we would have to repeat the procedure a number of times and look for the lowest common multiple of the measured values of N/k. Alternatively, we could test each result we obtain against the original data as this often involves a very short conventional calculation: for example, if the quantum Fourier transform is used to determine prime factors, their correctness can be tested by a single multiplication. We note that once the entangled state representing the periodic function has been established, all subsequent transformations and measurements, which actually lead to a value of the period of the function represented by the y-register, are carried on the x-register only.

We illustrate the above by considering the particular case defined in (15.22); using (15.26) and (15.27) we see that the wave function $|\psi\rangle$ can be written as the

sum of the following eight terms:

$$\left.\begin{aligned}
|\tilde{y}_0\rangle|0\rangle &= (1/4)(\alpha_4\alpha_5 + \alpha_4\beta_5 + \beta_4\alpha_5 + \beta_4\beta_5)\alpha_1\alpha_2\alpha_3 \\
|\tilde{y}_1\rangle|1\rangle &= 0 \\
|\tilde{y}_2\rangle|2\rangle &= (1/4)(\alpha_4\alpha_5 + i\alpha_4\beta_5 - \beta_4\alpha_5 - i\beta_4\beta_5)\alpha_1\beta_2\alpha_3 \\
|\tilde{y}_3\rangle|3\rangle &= 0 \\
|\tilde{y}_4\rangle|4\rangle &= (1/4)(\alpha_4\alpha_5 - \alpha_4\beta_5 + \beta_4\alpha_5 - \beta_4\beta_5)\beta_1\alpha_2\alpha_3 \\
|\tilde{y}_5\rangle|5\rangle &= 0 \\
|\tilde{y}_6\rangle|6\rangle &= (1/4)(\alpha_4\alpha_5 - i\alpha_4\beta_5 - \beta_4\alpha_5 + i\beta_4\beta_5)\beta_1\beta_2\alpha_3 \\
|\tilde{y}_7\rangle|7\rangle &= 0
\end{aligned}\right\} \tag{15.30}$$

where we have represented $|k\rangle$ by products of the spin states α and β, which represent 0 and 1 respectively. Because $|y_j\rangle = |y_{j+4}\rangle$, the contributions from these terms to $|\tilde{y}_k\rangle|k\rangle$ have doubled up in the case where k is even and cancelled out when k is odd. The probability of obtaining one of the allowed values of k is proportional to $\langle\tilde{y}_k|\tilde{y}_k\rangle$ and equals 0.25 in all cases. When we measure the x-register we therefore obtain a value of k equal to 0, 2, 4 or 6. If we get zero, we get no information, but otherwise we can deduce that $r = 4, 2$ or $4/3$. Repeating the whole calculation a small number of times is very likely to lead us to conclude that $r = 4$.

We now describe in a little more detail how, in principle, the quantum Fourier transform (15.25) could be carried out. We first write the number k in its binary form $\sum_{l=1}^{n} k_l 2^{n-l}$, where $k_l = 0$ or 1. Substituting into (15.25) we get

$$\begin{aligned}
|j\rangle &\to 2^{-n/2} \sum_{k_1=0}^{1} \sum_{k_2=0}^{1} \ldots \sum_{k_n=0}^{1} \exp\left(\frac{2\pi i}{2^n} j \sum_{l=1}^{n} k_l 2^{n-l}\right) |k_1\rangle|k_2\rangle..|k_n\rangle \\
&= 2^{-n/2} \prod_{l=1}^{n} \sum_{k_l=0}^{1} \exp\left(2\pi i j k_l 2^{-l}\right) |k_l\rangle \\
&= 2^{-n/2} \prod_{l=1}^{n} \left[\alpha_l + \exp\left(2\pi i j 2^{-l}\right)\beta_l\right]
\end{aligned} \tag{15.31}$$

where, in the last line, we have again replaced $|0\rangle$ and $|1\rangle$ by the corresponding spin states α_l and β_l. We now write j in its expanded binary form, so that the exponent in (15.31) can be written as

$$2\pi i j 2^{-l} = 2\pi i \sum_{m=0}^{n} j_m 2^{n-m-l} \tag{15.32}$$

where $j_m = 0$ or 1. We substitute (15.32) into (15.31) dropping all terms with $m \le n - l$ in the above summation because the exponential of $2\pi i$ times an integer equals one. This yields

$$|j\rangle \to 2^{-n/2} \prod_{l=1}^{n} \left[\alpha_l + \exp\left(2\pi i \sum_{m=n-l+1}^{n} j_m 2^{n-m-l}\right)\beta_l\right] \tag{15.33}$$

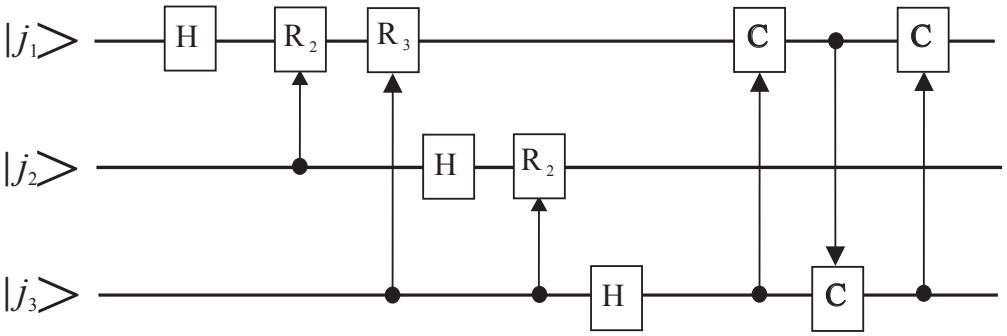


FIGURE 15.4 Quantum Fourier transform. The boxes labelled H represent Hadamard gates that perform a $\pi/2$ rotation of spin about the y axis. The boxes labelled R_2 and R_3 rotate the spin about the z axis so as to introduce phase changes of $\pi/2$ and $\pi/4$, respectively if the relevant control bit equals 1. The three boxes labelled C are CNOT gates, whose combined effect is to swap the states of qubits 1 and 3.

Our earlier example treated the case where the x register contained three qubits in which case (15.33) can be written as

$$|j\rangle \rightarrow 2^{-3/2}[\alpha_1 + \exp(i\pi j_3)\beta_1][\alpha_2 + \exp(i(j_2\pi + j_3\pi/2))\beta_2] \times [\alpha_3 + \exp(i(j_1\pi + j_2\pi/2 + j_3\pi/4))\beta_3] \quad (15.34)$$

We see, therefore, that the transformation (15.25) can be effected by changing the phase of each β by an amount specified in (15.34) and leaving the phases of the αs unchanged. This can be achieved using a "controlled phase gate" by which we mean one that applies a specified phase change to a target bit if and only if the value of the control bit equals one. Referring again to Chapter 11, we see from (11.14) that the application of a magnetic field parallel to z for a time t changes the relative phase of β and α by $\omega_p t$ and, because the overall phase of the wave function is arbitrary, we can attribute all of this phase change to β. In principle, therefore, we can generate the desired phase change by applying the field created by the control bit to the target bit for a given time. (NB: we should also apply an external field adjusted to cancel out the control field when the control bit is zero—cf. the earlier discussion of the controlled NOT gate).

Hopefully this will all become clearer when we discuss the particular example illustrated in figure 15.4, which represents the effect of applying several spin rotations to a 3-qubit number. We start by considering the effect of the operations on the first qubit whose initial state is $|j_1\rangle$. The first Hadamard gate, H, rotates the spin through $\pi/2$ about the y axis to produce the state, $2^{-\frac{1}{2}}(\alpha_1 \pm \beta_1)$, the sign being plus or minus depending on whether $|j_1\rangle$ is α_1 or β_1 respectively. The following operation R_2 applies a field parallel to z so as to change the phase of β_1 by $\pi/2$ if and only if $j_2 = 1$; R_3 changes the phase by $\pi/4$ if and only if $j_3 = 1$. The state of $|j_1\rangle$ has therefore now become

$$\alpha_1 + \exp[i(j_1\pi + j_2\pi/2 + j_3\pi/4)]\beta_1$$

The qubit $|j_2\rangle$ is operated on by the Hadamard gate H and by R_2 and changes in a similar way to become

$$\alpha_2 + \exp[i(j_2\pi + j_3\pi/2)]\beta_2$$

while $|j_3\rangle$ is operated on only by H to become

$$\alpha_3 + \exp[i(j_3\pi)]\beta_2$$

If we compare the above expressions with (15.34), we see that these gates have created a state of the three qubits similar to the one we were aiming at, but with the labels 1 and 3 interchanged. The states of these qubits are then swapped by the operation of the three CNOT gates shown in figure 15.4 to produce the state (15.34).

The total number of operations involved in a quantum Fourier transform is easily estimated. Referring to figure 15.4, we see that the first qubit is subject to one H followed by $(n-1)$ R_2's, the second to one H plus $(n-2)$ R_2's, and so on. There are then about $n/2$ swap gates making a total number of around n^2 operations. This is in contrast to the fastest classical computation, which involves $2^n n$ operations. The potential gain is huge for large values of n (faster by a factor of about 5×10^4 if $n = 20$ and 2×10^{13} for $n = 50$) and this has motivated the considerable scientific effort that has been devoted to the possible realization of such a quantum calculation.

As we stressed from the beginning our discussion has been confined to how "in principle" a quantum Fourier transform could be carried out, but we now briefly turn to what might be possible in practice. The practical problems involved in using the spins associated with single atoms are formidable, but some progress has been made using liquids made up of molecules that contain atoms with unpaired nuclear spins. When a magnetic field is applied to such a liquid, the nuclear spins tend to line up parallel to it to create a macroscopic magnetic moment. If a radio frequency field tuned to the energy difference between the parallel and antiparallel nuclear spin states is applied, this magnetic moment can be rotated. This process, which is known as nuclear magnetic resonance (NMR), allows the magnetic moment to be treated as a qubit much as described above. As the nuclei contained in the molecules making up the liquid have different resonant frequencies, they can be manipulated independently, and it is also possible to arrange for them to interact in such a way as to create a CNOT and other quantum gates. Because the states are represented by macroscopically sized spins, they are not collapsed by a measurement in the way described earlier and this part of the calculation has to be performed in a different way. Indeed, it can be argued that a calculation based on NMR is closer to a many-dimensional classical analogue calculation than to a true quantum calculation. Using NMR techniques, a quantum calculation based on seven qubits has been carried out to implement Shor's algorithm for the factorization of a number into its prime factors. This involved generating a four-qubit periodic function, finding that its period is four and deducing that the factors of the number fifteen are five and three! However, although NMR has provided a robust realization of a quantum computation, seven is probably close to the maximum number of qubits that can be employed in this way.

This is because each qubit is identified with the resonance of a particular nucleus in the molecule and the number of these is equal to the number of different nuclear sites in the molecule. If molecules with a larger variety of spins were used, their resonant frequencies could be too close to each other to allow them to be distinguished by an applied field—unless this is applied for a time greater than the inverse of the frequency difference, in which case the process is slowed to the point where its advantage is lost.

This example illustrates the enormous potential power of the quantum computer, the essential feature being the almost miraculous way in which a superposition of the results of n calculations can be generated by a single operation. The measurement process limits the amount of this information we can retrieve, but in appropriate cases very powerful results can be achieved. We should, however, remember the immense obstacles that lie in the way of any practical application of these ideas. Although entangled states of two particles can be generated almost routinely, extending this to more than a few is difficult, and a hundred or so really seems impossible. However, there are many examples in the past of the seemingly impossible being achieved and we should beware of underestimating human technological capacity. Whether more "in principle" objections may arise if and when we understand the measurement process better will be speculated on in the next chapter.

15.5 PROBLEMS

15.1 Construct a variant of Table 15.1 in which Eve makes different guesses about the settings of Alice's apparatus.

15.2 Confirm that the state defined by Equation (15.5) is the same as that given in (15.4)

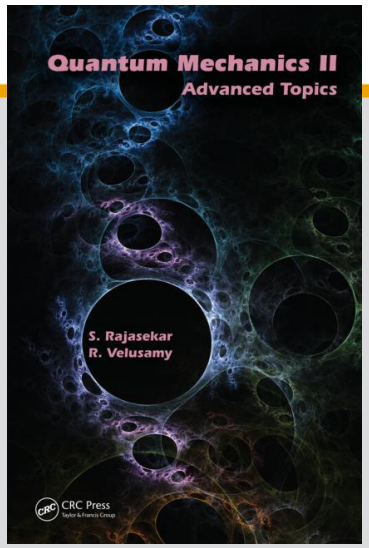
15.3 Verify the equivalence of equations (15.12) and (15.13).

15.4 Describe the operation of a quantum Fourier transform on the system described in worked example 12.3 and explain how it can be used to obtain the period of this function.

CHAPTER

2

QUANTUM COMPUTERS



This chapter is excerpted from

Quantum Mechanics II, Advanced Topics

by S. Rajasekar, R. Velusamy

Learn more

CHAPTER 7

Quantum Computers

7.1 INTRODUCTION

Present day computers perform computations using two-state binary logic. They led to an amazing revolution in data manipulation and processing. The components of a computer are subject to various laws of physics. *What will happen if the components of a computer become very small such that they are subjected to the principles of quantum mechanics?* Alternatively, can a real quantum system be used to build a computer functioning in the quantum mechanical regime? This is one of the major issues in quantum computing. The name *quantum computing* refers to calculations using logic based on the probability amplitude concept.

Classical computers of the present decade are able to answer one question at a time. In contrast a quantum computer will have the ability to carry out more than one problem simultaneously. Essentially quantum computers manipulate quantum states instead of classical bits. In a quantum computer the eigenstates of, for example, a two-level system are renamed 0 and 1. Now, the two level quantum system becomes a qubit (that is, quantum binary digit) [1-4]. The concept of the quantum computer was introduced first by Paul Benioff (1980) [5]. Richard Feynman [6,7] contributed to the early development of quantum computation. The first paper on quantum computing was published by David Deutsch [8] in the year 1985.

During the past one decade or so, many quantum algorithms have emerged. Among them the most remarkable successes of quantum computation are Shor's efficient algorithms for integer factorization and the computation of discrete logarithms [9,10]. Peter Williston Shor has shown that quantum computers would solve the problem of finding discrete logarithms (mod N). He predicted that a quantum computer can perform prime factoring in polynomial time: $t \propto k^p$ where p is a constant and k is the number of bits in the number to be factored. For this problem a classical computer is believed to take $e^{ck^{1/3}}$ time where c is a constant. Shor's breakthrough created an avalanche of research activity in quantum computation and quantum information theory. In

addition to Shor's factorization algorithm, Deutsch–(Richard)Jozsa algorithm [8,11] and Lov Kumar Grover's rapid search algorithm [12] are capable of performing certain computational tasks exponentially faster compared to their classical counterparts. In the present chapter we discuss the basic aspects of quantum computing.

7.2 WHAT IS A QUANTUM COMPUTER?

In a classical information theory 'bit' is an indivisible unit. It takes the values such as yes or no, true or false or simply 0 or 1. A sequence of bits is used to represent classical information. In a classical computer, logical gates are employed to evaluate Boolean functions of a set of input bits.

7.2.1 Qubits

Quantum information can be represented by the elementary units called *quantum bits* abbreviated as *qubits* or *qbits*. A qubit is two levels of a quantum system (like the spin of an electron). For example, spin-up, $|\uparrow\rangle$, represents 1 (true) and spin-down, $|\downarrow\rangle$, represents 0 (false). Note that $|\uparrow\rangle$ to $|\downarrow\rangle$ can be achieved by a magnetic field and dissipates no heat. All information can be encoded into a sequence of qubits. In principle, any two-state system can be used as a quantum bit. Some examples are presented in table 7.1.

What is the difference between a classical bit and a qubit? A qubit can be in a state other than $|0\rangle$ and $|1\rangle$. It is possible to form a combination of states called *superposition states* given by $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$. Denoting $|0\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$ and $|1\rangle = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$ the quantum state $|\psi\rangle$ is written in vector notation as $\begin{bmatrix} \alpha \\ \beta \end{bmatrix}$. For example, we can represent the spin of an electron

TABLE 7.1 Examples of two-state quantum systems. Here $|V\rangle$, $|H\rangle$, $|L\rangle$ and $|R\rangle$ represent vertical, horizontal, left-circular and right-circular polarizations respectively. $|+\rangle$ and $|-\rangle$ denote spin-up and spin-down respectively. $|E_0\rangle$ and $|E_1\rangle$ represent ground and excited states respectively.

S.No.	$\|0\rangle$	$\|1\rangle$	Qubit
1.	$\|V\rangle$	$\|H\rangle$	Photon – Linear polarization
2.	$\|L\rangle$	$\|R\rangle$	Photon – Circular polarization
3.	$\|+\rangle$	$\|-\rangle$	Electron, nucleus – Spin
4.	$\|E_0\rangle$	$\|E_1\rangle$	Atoms, quantum dots – Energy levels

in the horizontal direction as the sum of the up and down states. When we measure a qubit, the result will be either 0 with probability $|\alpha|^2$ or 1 with probability $|\beta|^2$. Hence, $|\alpha|^2 + |\beta|^2 = 1$. A classical bit has either 0 state or 1 state whereas a qubit can exist between $|0\rangle$ and $|1\rangle$ until it is observed.

$|\psi_1\rangle \otimes |\psi_2\rangle$ denotes tensor product of $|\psi_1\rangle$ and $|\psi_2\rangle$. If $|\psi_1\rangle = \begin{bmatrix} 1 \\ 2\mathrm{i} \end{bmatrix}$ and $|\psi_2\rangle = \begin{bmatrix} 2 \\ 3 \end{bmatrix}$ then

$$|\psi_1\rangle \otimes |\psi_2\rangle = \begin{bmatrix} 1 \\ 2\mathrm{i} \end{bmatrix} \otimes \begin{bmatrix} 2 \\ 3 \end{bmatrix} = \begin{bmatrix} 1 \times 2 \\ 1 \times 3 \\ 2\mathrm{i} \times 2 \\ 2\mathrm{i} \times 3 \end{bmatrix} = \begin{bmatrix} 2 \\ 3 \\ 4\mathrm{i} \\ 6\mathrm{i} \end{bmatrix}. \tag{7.1}$$

If $|\psi_1\rangle = a|0\rangle + b|1\rangle$ and $|\psi_2\rangle = c|0\rangle + d|1\rangle$ then

$$\begin{aligned} |\psi_1\rangle \otimes |\psi_2\rangle &= |\psi_1\psi_2\rangle \\ &= ac|0\rangle|0\rangle + ad|0\rangle|1\rangle + bc|1\rangle|0\rangle + bd|1\rangle|1\rangle \\ &= ac|00\rangle + ad|01\rangle + bc|10\rangle + bd|11\rangle. \end{aligned} \tag{7.2}$$

Multiple bits have more states. With two classical bits 0 and 1 there are four possible states 00, 01, 10, 11. But a general two qubit system can be represented by

$$|\psi\rangle = \alpha_{00}|00\rangle + \alpha_{01}|01\rangle + \alpha_{10}|10\rangle + \alpha_{11}|11\rangle \tag{7.3}$$

with $\sum |\alpha_x|^2 = 1$ where $x = 00, 01, 10, 11$.

As infinite range of values of α and β are possible with $|\alpha|^2 + |\beta|^2 = 1$, in principle a qubit can store an infinite amount of data. But this is misleading because a measurement of the qubit changes its state to yield either 0 or 1. Measurement collapses the state of qubit from the superposition of $|0\rangle$ and $|1\rangle$ to $|0\rangle$ with a probability $|\alpha|^2$ or $|1\rangle$ with probability $|\beta|^2$. So, from a measurement we can obtain only a single bit of information about the qubit's state. Only if infinitely many identical qubits are prepared and then measurements are performed we can determine α and β. As no quantum state can be copied because such an act will lead to the collapse of the superposition state into one of its constituent state, it is impossible to set-up identical states. Hence, in principle, it is impossible to find α and β exactly. The information contained in a qubit is enormous if we do not measure it. That is, nature conceals a great deal of information. This hidden quantum information falls at the center of what makes quantum mechanics a powerful modern emerging tool for information processing.

Solved Problem 1:

Write the Pauli matrices σ_x, σ_y and σ_z in operator form and state their effect on a qubit.

Defining $|0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$ and $\langle 0| = \begin{pmatrix} 1 & 0 \end{pmatrix}$ one can find $|0\rangle\langle 0|$ as

$$|0\rangle\langle 0| = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \begin{pmatrix} 1 & 0 \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix}. \tag{7.4}$$

Then

$$\sigma_x = |0\rangle\langle 1| + |1\rangle\langle 0|, \tag{7.5}$$
$$\sigma_y = -\mathrm{i}|0\rangle\langle 1| + \mathrm{i}|1\rangle\langle 0|, \tag{7.6}$$
$$\sigma_z = |0\rangle\langle 0| - |1\rangle\langle 1|. \tag{7.7}$$

The action of the Pauli matrices σ_x and σ_z on a qubit is

$$\sigma_x|0\rangle = |1\rangle, \quad \sigma_x|1\rangle = |0\rangle, \tag{7.8}$$
$$\sigma_z|0\rangle = |0\rangle, \quad \sigma_z|1\rangle = -|1\rangle. \tag{7.9}$$

We note that σ_x gives rise to bit flip while σ_z causes phase flip. What is the effect of σ_y?

7.2.2 Quantum Gates

An elementary quantum logic gate is a unitary transformation. A quantum gate acts on a qubit or pair of qubits. Quantum gates are represented by matrices or operators. Any unitary matrix can specify a valid gate. We can represent them in Dirac notation also. In quantum computers a unitary transformation is applied to a given initial state of a set of qubits through several quantum gates. The final outcome of a quantum computation is contained in the final state of the qubits.

There are certain major differences between classical and quantum gates.

1. Fan-in[1] is not possible in quantum circuits.
2. In classical circuits wires are joined together to form a single wire. In quantum circuits this irreversible operation is not possible.
3. Fan-out[2] is also not possible. That is, making a number of copies of a bit is not possible.
4. Quantum gates do not permit feedback loops from one part of the circuit to another part.

[1] A term defining the maximum number of digital inputs allowed by a logic gate.

[2] It defines the maximum number of digital inputs that the output of logic gates can feed.

7.2.3 Quantum Computer

Quantum computation is defined as an arbitrary transformation on a Hilbert space spanned by the complete set of all possible states of bits. The difference between quantum computers and a system of interacting spins is that the computation must be modular, each logical operation considers only a few spins. Essentially, quantum computers will perform computations at the atomic scale.

In a quantum computer, the execution of a program can be thought of as a dynamical process and is governed by the Schrödinger equation. Consequently, a state vector ψ is used to describe the state of the computer. Here ψ is a linear superposition of all the binary states of the bits $x_m \in \{0, 1\}$:

$$\psi(t) = \sum_{x_m \in \{0,1\}} \alpha_x \, |x_1, x_2, \cdots, x_m\rangle \ , \quad \sum_x |\alpha_x|^2 = 1 \ . \tag{7.10}$$

The time evolution of the state is governed by a unitary operator U on a vector space.

7.3 WHY IS A QUANTUM COMPUTER?

Anything a quantum computer can perform can also be done in a classical computer. Then, *why should one think of a quantum computer?* In the following we list some of the difficulties with classical computers and the advantages of quantum computers.

1. A quantum computer is very efficient over a classical computer.

   Example 1:

   Consider a quantum state of a modest number of qubits, for example 100 lines in a Hilbert space of dimensions $2^{100} \sim 10^{30}$. To perform a computation a classical computer has to work with matrices of exponentially large size and this would take a very long time.

   Example 2:

   A single computation acting on say, 300 qubits can achieve the same effect as 2^{300} computations acting simultaneously on classical bits.

   Example 3:

   To factor a 400 digit number, a powerful workstation would require about 10 years. But a quantum computer could complete the same task in just a few minutes.

2. There are a number of problems for which the underlying process can be sped-up tremendously through quantum algorithms. Consider a number N of L digits so that $N \approx 10^L$. To determine its factors, in the least case,

it is required to divide N by numbers up to $\sqrt{N}$. That is $\sqrt{N} \sim 10^{L/2}$ operations are essential. Hence, the number of operations would increase with L exponentially. The best known classical algorithm requires $s = Ae^{1.9L^{1/3}(\log L)^{2/3}}$ (A is a constant) number of operations for factorizing an L digit number. Therefore, it is not considered an efficient algorithm. To factorize a 130 digit number at the rate of $\sim 10^{12}$ operations per second, a classical computer would require ~ 42 days. It would require ~ 10 years for a 400 digit number. However, a quantum algorithm of Peter Williston Shor requires time $\propto L^3$.

3. Computations cannot be reversible in a classical computer (why can't a computer run backwards?). In quantum theory, reverse time evolution is specified by the unitary operator $U^{-1} = U^\dagger$. A consequence of this is that computations can be reversible in a quantum computer.

4. Calculations in a classical computer lead to dissipation in order to damp out an attempt by the system to make a transition. In contrast, in a quantum computer dissipation cannot be used and further the accuracy of a computation is built-in.

5. In a classical computer errors in the initial data may grow exponentially with the number of steps involved. This is because, the classical dynamics involves the symplectic group that is noncompact. In quantum mechanics, inaccuracies in the initial data do not grow. This is because it uses the compact unitary group.

7.4 FUNDAMENTAL PROPERTIES

In the following we discuss the fundamental properties of quantum systems that are relevant to information processing.

7.4.1 Software, Hardware and CPU [1,13]

Superposition:

A quantum computer can exist in an arbitrary linear combination of classical Boolean states. These states evolve in parallel as per a unitary transformation.

Interference:

Parallel computation paths in the superposition, like a particle's path through an interferometer, can cancel one another or reinforce depending on their relative phase.

Entanglement:

Certain states of a complete quantum system do not form definite states of its parts. For more details see sec.10.2.

Nonlocality and uncertainty:

An unknown quantum state cannot be copied (cloned) accurately. It cannot be observed without being disturbed.

A quantum computer has a register with n qubits. A qubit has 0 and 1 classical states so that the register has 2^n classical states. The state of a quantum computer is described by a 2^n-dimensional vector, $\mathbf{x}$, indexed by $i = 000\cdots00, 000\cdots01, 000\cdots10, \cdots, 111\cdots11$ in binary notation. Moreover,

$$||\mathbf{x}||^2 = \sqrt{\sum_j |x_j|^2} = 1 \tag{7.11}$$

and $|x_j|^2$ is the probability that the register is in state j. We call $\mathbf{x}$ the wave function (ψ) of the register.

In a quantum computer, the software is represented by ψ and the hardware by a Hamiltonian. The Hamiltonian describes the dynamics of the central processing unit (CPU). The hardware generates a unitary evolution of $\psi(t)$ representing the state of the software at time t. The software is a finite string of bits with a logical meaning. It includes the inputs (such as programs and data), the output and a scratch pad necessary to store intermediate results. The states, for example, $|\downarrow\rangle$ and $|\uparrow\rangle$ represent a bit with logical values 0 and 1 respectively.

7.4.2 Two-Bit Gates for Universal Computation

In a classical computer logic gates are used to process information. David Deutsch has shown a way to obtain a universal quantum computation. Tommaso Toffoli [14] showed how the AND and XOR gates can be implemented reversibly. Recall that conventional AND and XOR gates are not reversible because a reversible gate must contain the same number of input and output bits. However, XOR can be implemented reversibly with a two-bit gate where one output bit may return the conventional XOR. $\oplus$ is used to denote for the exclusive-or operation. $a_1 \oplus a_2$ (a_1 and a_2 are the binary values of the two input bits) is given by a one output bit, while the second output bit returns the original value of a_1 (or a_2). To implement AND reversibly, a three-bit gate is used where a_1 and a_2 are passed through unchanged and the third bit is XORed with the AND of the first two, returning $(a_1 \cdot a_2) \oplus a_3$. Because this three-bit gate has both the XOR and the AND functions, it can be considered as a universal reversible gate. This gate is called *Toffoli gate*.

7.4.3 NOT, Z and Hadamard Gates

A simple classical gate is the NOT gate that changes 0 to 1 and 1 to 0. An analogous quantum NOT gate transforms states in a particular basis into

TABLE 7.2 The truth table of the quantum NOT gate.

Input	Output
$\|0\rangle$	$\|1\rangle$
$\|1\rangle$	$\|0\rangle$
$\alpha\|0\rangle + \beta\|1\rangle$	$\alpha\|1\rangle + \beta\|0\rangle$

states orthogonal to them. The unitary operation $U_{\rm NOT}$ is given by

$$U_{\rm NOT}|0\rangle = |1\rangle\,, \tag{7.12a}$$
$$U_{\rm NOT}|1\rangle = |0\rangle\,. \tag{7.12b}$$

Unlike the digital gates, the quantum gates are assumed to act on superposition states. The $U_{\rm NOT}$ provides the transformation

$$U_{\rm NOT}(\alpha|0\rangle + \beta|1\rangle) = \alpha|1\rangle + \beta|0\rangle\,. \tag{7.13}$$

Here α and β are the amplitudes of the states. If we represent $|0\rangle$ and $|1\rangle$ in column matrix then the output of the NOT gate is

$$X\begin{pmatrix}\alpha\\ \beta\end{pmatrix} = \begin{pmatrix}0 & 1\\ 1 & 0\end{pmatrix}\begin{pmatrix}\alpha\\ \beta\end{pmatrix} = \begin{pmatrix}\beta\\ \alpha\end{pmatrix}\,. \tag{7.14}$$

The truth table of the quantum NOT gate is given in the table 7.2.

In classical gates, the NOT gate is the only nontrivial single-bit gate. In quantum mechanics there are many nontrivial qubit gates with $|\alpha|^2+|\beta|^2 = 1$. Examples are the Z gate and Hadamard (H) gate. They are given by

$$\mathrm{Z} = \begin{pmatrix}1 & 0\\ 0 & -1\end{pmatrix}, \quad \mathrm{H} = \frac{1}{\sqrt{2}}\begin{pmatrix}1 & 1\\ 1 & -1\end{pmatrix}. \tag{7.15}$$

These gates are very useful. The truth table of them are given in tables 7.3 and 7.4 respectively. These gates are represented pictorially as shown in Fig. 7.1. Consider the Hadamard transformation given by

$$\mathrm{H} = \frac{1}{\sqrt{2}}\begin{pmatrix}1 & 1\\ 1 & -1\end{pmatrix}, \quad \mathrm{H}\begin{pmatrix}1\\ 0\end{pmatrix} = \frac{1}{\sqrt{2}}\begin{pmatrix}1\\ 1\end{pmatrix} = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle),$$
$$\mathrm{H}\begin{pmatrix}0\\ 1\end{pmatrix} = \frac{1}{\sqrt{2}}\begin{pmatrix}1\\ -1\end{pmatrix} = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle). \tag{7.16}$$

It acts on a single qubit. Its effect is to rotate the state about the y-axis.

Interestingly, there are infinitely many 2×2 unitary matrices and so infinitely many qubit gates. Remember that the classical gate NOT gate is the

TABLE 7.3 The truth table of the quantum Z gate.

Input	Output
$\|0\rangle$	$\|0\rangle$
$\|1\rangle$	$-\|1\rangle$
$\alpha\|0\rangle + \beta\|1\rangle$	$\alpha\|0\rangle - \beta\|1\rangle$

TABLE 7.4 The truth table of the Hadamard gate.

Input	Output
$\|0\rangle$	$\frac{1}{\sqrt{2}}(\|0\rangle + \|1\rangle)$
$\|1\rangle$	$\frac{1}{\sqrt{2}}(\|0\rangle - \|1\rangle)$
$\frac{1}{\sqrt{2}}(\|0\rangle + \|1\rangle)$	$\|0\rangle$
$\frac{1}{\sqrt{2}}(\|0\rangle - \|1\rangle)$	$\|1\rangle$
$\alpha\|0\rangle + \beta\|1\rangle$	$\frac{1}{\sqrt{2}}(\alpha+\beta)\|0\rangle + \frac{1}{\sqrt{2}}(\alpha-\beta)\|1\rangle$

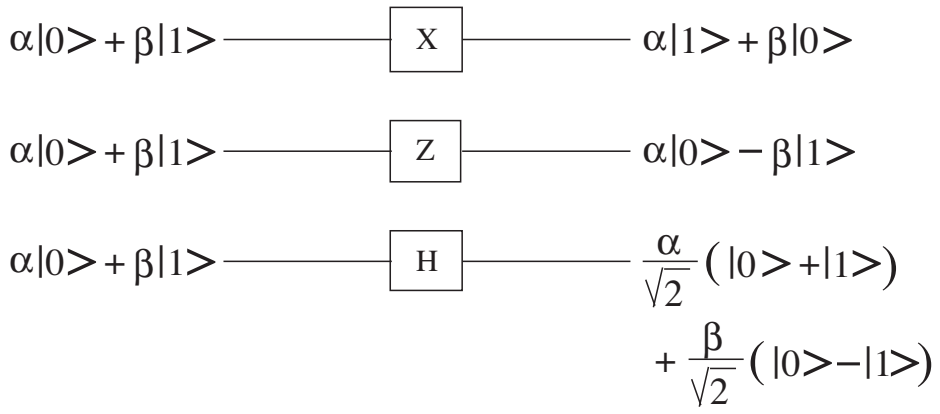


FIGURE 7.1 Qubit logic gates.

single one-bit. Any single qubit unitary gate can be decomposed as

$$U = \mathrm{e}^{\mathrm{i}\alpha} \begin{bmatrix} \mathrm{e}^{-\mathrm{i}\beta/2} & 0 \\ 0 & \mathrm{e}^{\mathrm{i}\beta/2} \end{bmatrix} \begin{bmatrix} \cos(\gamma/2) & -\sin(\gamma/2) \\ \sin(\gamma/2) & \cos(\gamma/2) \end{bmatrix} \times \begin{bmatrix} \mathrm{e}^{-\mathrm{i}\delta/2} & 0 \\ 0 & \mathrm{e}^{\mathrm{i}\delta/2} \end{bmatrix}, \tag{7.17}$$

where α, β, γ and δ are real numbers. In fact, one can build-up a qubit gate using a finite set of quantum gates called *universal gates*.

Solved Problem 2:

If X, H and Z denote the quantum NOT, Hadamard and Z gates respectively, show that $HXH = Z$.

We obtain

$$\begin{aligned} HXH &= \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \\ &= \frac{1}{2} \begin{pmatrix} 1 & 1 \\ -1 & 1 \end{pmatrix} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \\ &= \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \\ &= Z. \end{aligned} \tag{7.18}$$

7.4.4 CNOT (XOR) and Toffoli Gates

CNOT stands for controlled NOT. It is a two qubit gate that modifies the state of one of the qubits depending on the state of the other control qubit. The effect of CNOT on the target state is shown in table 7.5. In this table if we treat the first two columns as the input and third as the output then this table is the truth table of classical XOR gate.

In operator form CNOT is

$$\mathrm{CNOT} = |00\rangle\langle 00| + |01\rangle\langle 01| + |10\rangle\langle 11| + |11\rangle\langle 10| \,. \tag{7.19}$$

TABLE 7.5 The truth table of the quantum CNOT gate.

Control	Target initial	Final
0	0	0
0	1	1
1	0	1
1	1	0

TABLE 7.6 The truth table of Toffoli gate.

Inputs			Outputs		
a	b	c	a'	b'	c'
0	0	0	0	0	0
0	0	1	0	0	1
0	1	0	0	1	0
0	1	1	0	1	1
1	0	0	1	0	0
1	0	1	1	0	1
1	1	0	1	1	1
1	1	1	1	1	0

The first qubit is the control and the target is the second qubit. The first two terms in Eq. (7.19) indicates that if the control qubit is in the state $|0\rangle$ then the target state remains the same. The target state is changed when the control state is $|1\rangle$. The last two terms in Eq. (7.19) represent these. Because the state of the second qubit is dependent on the first qubit's state, the two qubits become entangled on passing through the CNOT gate. Combination of the CNOT and the Hadamard gates can be used to realize both quantum superposition and entanglement. We notice that the control and the target qubits are XORed and stored in the target qubit. The matrix representation of the CNOT operation is given by

$$U_{\text{NOT}} \begin{pmatrix} |00\rangle \\ |01\rangle \\ |10\rangle \\ |11\rangle \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} |00\rangle \\ |01\rangle \\ |10\rangle \\ |11\rangle \end{pmatrix} . \tag{7.20}$$

A reversible quantum gate is Toffoli gate. It has three input bits, say a, b and c and three outputs a', b' and c'. The truth table of Toffoli gate is given in table 7.6. a and b are treated as control bits and are unaffected by the target bit c. But c is inverted if both a and b are 1.

Solved Problem 3:

Given $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$ and an EPR pair $(|00\rangle + |11\rangle)/\sqrt{2}$ find the state of the complete system and the effect of CNOT on it.

We obtain

$$\frac{1}{\sqrt{2}} \left[\alpha|0\rangle \left(|00\rangle + |11\rangle\right) + \beta|1\rangle \left(|00\rangle + |11\rangle\right)\right] = \frac{1}{\sqrt{2}} \begin{pmatrix} u_1 \\ u_2 \end{pmatrix}, \tag{7.21a}$$

where

$$u_1 = \begin{pmatrix} \alpha \\ 0 \\ 0 \\ \alpha \end{pmatrix}, \quad u_2 = \begin{pmatrix} \beta \\ 0 \\ 0 \\ \beta \end{pmatrix}. \tag{7.21b}$$

Performing CNOT we get

$$\frac{1}{\sqrt{2}} \left[\alpha|0\rangle \left(|00\rangle + |11\rangle\right) + \beta|1\rangle \left(|10\rangle + |01\rangle\right)\right] = \frac{1}{\sqrt{2}} \begin{pmatrix} u_1 \\ u_3 \end{pmatrix}, \quad u_3 = \begin{pmatrix} 0 \\ \beta \\ \beta \\ 0 \end{pmatrix}. \tag{7.22}$$

7.4.5 Symbols for Quantum Circuits

The schematic symbols used to denote some unitary operations in quantum circuits are given in Fig. 7.2 with their matrix representations. The connections are represented by the symbols shown in Fig. 7.3. If a gate U acts on an n-qubit, we depict it as in Fig. 7.4.

By a measurement on the n-qubit register of a quantum computer, we mean measuring the observable

$$x = \sum_{i=0}^{2^n-1} i|i\rangle\langle i| \tag{7.23}$$

and it is represented in circuits by the ammeter symbol as in Fig. 7.4. In a measurement we get two quantities, a collapsed state $|k\rangle$ and its probability $|\langle k|U|\psi\rangle|^2$, and hence it is indicated by a double-line in Fig. 7.4.

Solved Problem 4:

Find the output state $|\psi_1\rangle$ of the circuit given in Fig. 7.5 for the input state $|\psi_0\rangle = \alpha|0\rangle + \beta|1\rangle$.

We have

$$Z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}, \quad X = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \tag{7.24}$$

and

$$|\psi_0\rangle = \alpha|0\rangle + \beta|1\rangle = \alpha \begin{bmatrix} 1 \\ 0 \end{bmatrix} + \beta \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} \alpha \\ \beta \end{bmatrix}. \tag{7.25}$$

Further,

$$Z|\psi_0\rangle = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \begin{bmatrix} \alpha \\ \beta \end{bmatrix} = \begin{bmatrix} \alpha \\ -\beta \end{bmatrix} = \alpha|0\rangle - \beta|1\rangle. \tag{7.26}$$

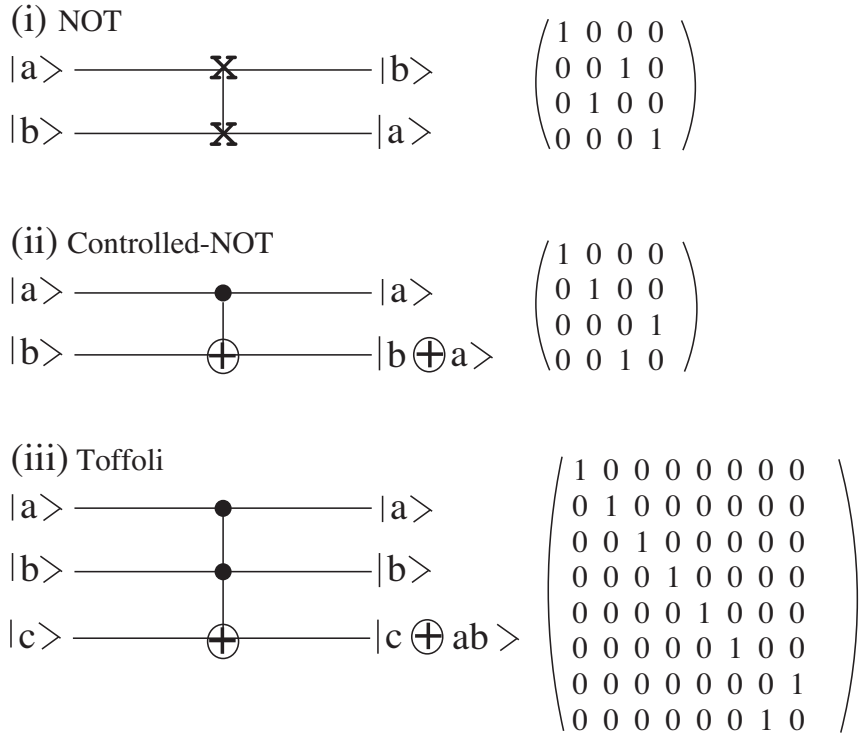


FIGURE 7.2 Circuit symbols and matrix representations of logic gates.

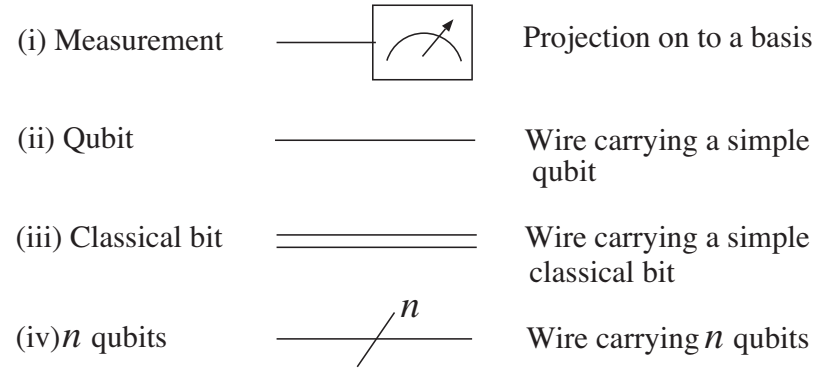


FIGURE 7.3 Symbols for connections in quantum circuits.

FIGURE 7.4 A quantum circuit.

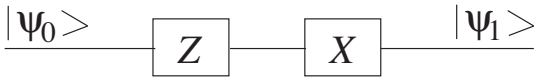


FIGURE 7.5 A quantum circuit with Z and X gates.

Then

$$|\psi_1\rangle = XZ|\psi_0\rangle = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} \alpha \\ -\beta \end{bmatrix} = \begin{bmatrix} -\beta \\ \alpha \end{bmatrix} = \alpha|1\rangle - \beta|0\rangle \ . \tag{7.27}$$

7.4.6 Evaluation of Functions

Let us describe the calculation of functions by quantum computers. Consider a function

$$f : \{0, 1, \cdots, 2^m - 1\} \to \{0, 1, \cdots 2^n - 1\} \ , \tag{7.28}$$

where m and n are positive integers. A classical computer calculates f by evolving the inputs $0, 1, \cdots, 2^m - 1$ into the outputs $f(0), f(1), \cdots, f(2^m - 1)$. Quantum computers use two registers. Input is stored in the first register and the second is for output. The quantum state of the first register is represented as $|x\rangle$. Output may be represented by $|y\rangle$. The function evaluation is computed by a unitary evolution operator U_f that acts on the two registers, that is,

$$U_f|x\rangle|0\rangle = |x\rangle|f(x)\rangle = |x, f(x)\rangle \ , \tag{7.29}$$

where the output is initially set to 0. The values of $f(0), \cdots, f(2^m - 1)$ found by applying U_f only once to a superposition of all input as

$$\begin{aligned} |\psi\rangle &= U_f \left(\frac{1}{2^{m/2}} \sum_{x=0}^{2^m-1} |x\rangle \right) |0\rangle \\ &= \frac{1}{2^{m/2}} \sum_{x=0}^{2^m-1} |x\rangle|f(x)\rangle \ . \end{aligned} \tag{7.30}$$

7.5 QUANTUM ALGORITHMS

One can simulate a classical circuit using a quantum circuit. That is, it is possible to perform classical computations on quantum computers. But the advantage of quantum computing is that powerful functions may be computed by making use of *quantum parallelism* the fundamental feature of many quantum algorithms. It allows a quantum computer to evaluate $f(x)$ for many different values of x simultaneously. This parallelism is not immediately useful because a measurement would give $f(x)$ for a single value of x only as in the case of a classical computer.

Quantum computation needs much more than that quantum parallelism to be useful. This is achieved in Deutsch's algorithm [8,15] which combines quantum parallelism with *interference*. Using Deutsch's algorithm, information about a $f(x)$ can be obtained very quickly compared with a classical computer. Deutsch's algorithm is a simple case of a more general Deutsch–Jozsa algorithm [11]. It suggests that quantum computers may be capable of solving certain problems more efficiently than classical computers.

There are three classes of quantum algorithms that provide an advantage over classical algorithms.

1. There is the class of algorithms based on quantum Fourier transform. Examples are the Deutsch–Jozsa algorithm of finding whether a given function is a constant or not and the Shor's algorithms for factoring and discrete logarithm.

2. The second class is quantum search algorithms. Their principles were discovered by Grover [16]. The goal of a quantum search algorithm is given a search space of size N finding an element of that search space having a known property. A quantum search algorithm achieves this in approximately $\sqrt{N}$ operations whereas a classical computer requires about N operations.

3. Another class of quantum algorithms is quantum simulation, where a quantum computer is explored to simulate a quantum system.

7.5.1 Deutsch's Algorithm

The first and the simplest quantum algorithm is Deutsch's problem. Let $f(x)$ denote one bit functions with $x = 0$ or 1. There are only four possibilities:

$$\begin{array}{ll} f_1(0) = 0, & f_1(1) = 0. \\ f_2(0) = 0, & f_2(1) = 1. \\ f_3(0) = 1, & f_3(1) = 0. \\ f_4(0) = 1, & f_4(1) = 1. \end{array} \tag{7.31}$$

Given an unknown f the problem is to determine which one of the above four classes it belongs to. In a classical algorithm we can calculate $f(0)$ and $f(1)$ and then find its class by comparing the values of $f(0)$ and $f(1)$ with the Eqs. (7.31). Therefore, we wish to evaluate f at 0 and 1. But a quantum algorithm requires only one evaluation. Deutsch proposed a quantum algorithm for the above problem which is based on the principle that the superposition of quantum states provide the possibility to perform computation on many states simultaneously. (The generalization of the Deutsch algorithm is the Deutsch–Jozsa algorithm.) The algorithm consists of three steps.

Consider the operation U_A

$$U_A|0\rangle = \frac{1}{\sqrt{2}}\left[|0\rangle + |1\rangle\right] \ , \quad U_A|1\rangle = \frac{1}{\sqrt{2}}\left[|0\rangle - |1\rangle\right] \ . \tag{7.32}$$

Applying n times U_A to an n-bit quantum register in the state $|0\rangle$ we have

$$\begin{aligned}
|\psi\rangle &= U_A \otimes U_A \otimes \cdots U_A |000\cdots 0\rangle \\
&= \frac{1}{\sqrt{2}}\left(|0\rangle + |1\rangle\right) \otimes \frac{1}{\sqrt{2}}\left(|0\rangle + |1\rangle\right) \otimes \cdots \frac{1}{\sqrt{2}}\left(|0\rangle + |1\rangle\right) \\
&= \frac{1}{2^{n/2}}\left[|00\cdots 0\rangle + |00\cdots 1\rangle + \cdots + |11\cdots 1\rangle\right] \\
&= \frac{1}{2^{n/2}} \sum_{i=0}^{2^n-1} |i\rangle \ . \qquad (7.33)
\end{aligned}$$

That is, n applications of U_A yields a register state that has 2^n distinct terms. Note that in classical case n elementary operations can only give one state of the register giving one number.

Let us start with two qubits. One bit is set to the state $|0\rangle$ and the other is to the state $|1\rangle$. The total state is $|01\rangle$. In the first step we apply the gate U_A to each qubit. This gives

$$\frac{1}{\sqrt{2}}\left[\,|0\rangle + |1\rangle\,\right] \otimes \frac{1}{\sqrt{2}}\left[\,|0\rangle - |1\rangle\,\right] = \frac{1}{2}\left[\,|00\rangle - |01\rangle + |10\rangle - |11\rangle\,\right] \ . \qquad (7.34)$$

In the second step compute f on the superposition state given by Eq. (7.34). This is realized by a two-bit gate U_f (Eq. (7.29)) acting on the basis vector

$$|x, y\rangle \to |x, y \oplus f(x)\rangle \ , \quad x, y = 0, 1 \qquad (7.35)$$

where $\oplus$ denotes addition mod 2. The last step is to apply U_A again on each qubit.

Let us apply the above algorithm assuming $f = f_1$. The first step yields the superposition state given by Eq. (7.34). (This is independent of the function.) The second step gives

$$\begin{aligned}
|\psi\rangle &= U_f \left(\frac{1}{2}\left[\,|00\rangle - |01\rangle + |10\rangle - |11\rangle\,\right]\right) \\
&= \frac{1}{2}\left[\,|0, 0 \oplus f(0)\rangle - |0, 1 \oplus f(0)\rangle + |1, 0 \oplus f(1)\rangle - |1, 1 \oplus f(1)\rangle\,\right] \\
&= \frac{1}{2}\left[\,|00\rangle - |01\rangle + |10\rangle - |11\rangle\,\right] \ . \qquad (7.36)
\end{aligned}$$

The final step is the application of U_A on $|\psi\rangle$ given by Eq. (7.36). We obtain

$$\begin{aligned}
|\psi\rangle &= \frac{1}{2}\Bigg[\frac{1}{\sqrt{2}}\left(|0\rangle + |1\rangle\right)\frac{1}{\sqrt{2}}\left(|0\rangle + |1\rangle\right) - \frac{1}{\sqrt{2}}\left(|0\rangle + |1\rangle\right)\frac{1}{\sqrt{2}}\left(|0\rangle - |1\rangle\right) \\
&\qquad + \frac{1}{\sqrt{2}}\left(|0\rangle - |1\rangle\right)\frac{1}{\sqrt{2}}\left(|0\rangle + |1\rangle\right) - \frac{1}{\sqrt{2}}\left(|0\rangle - |1\rangle\right)\frac{1}{\sqrt{2}}\left(|0\rangle - |1\rangle\right)\Bigg] \\
&= \frac{1}{4}\left[\,|00\rangle + |01\rangle + |10\rangle + |11\rangle - |00\rangle + |01\rangle - |10\rangle + |11\rangle\right. \\
&\qquad \left. + |00\rangle + |01\rangle - |10\rangle - |11\rangle - |00\rangle + |01\rangle + |10\rangle - |11\rangle\right] \\
&= |01\rangle \ . \qquad (7.37)
\end{aligned}$$

If $f = f_2$ then the second step gives

$$\begin{aligned} |\psi\rangle &= \frac{1}{2}\left[\,|0,0\oplus 0\rangle - |0,1\oplus 0\rangle + |1,0\oplus 1\rangle - |1,1\oplus 1\rangle\,\right] \\ &= \frac{1}{2}\left[\,|00\rangle - |01\rangle + |11\rangle - |10\rangle\,\right]\ . \end{aligned} \tag{7.38}$$

Then U_A on $|\psi\rangle$ results in

$$\begin{aligned} |\psi\rangle &= \frac{1}{4}\left[\,|00\rangle + |01\rangle + |10\rangle + |11\rangle - |00\rangle + |01\rangle - |10\rangle + |11\rangle\right. \\ &\quad \left. +|00\rangle - |01\rangle - |10\rangle + |11\rangle - |00\rangle - |01\rangle + |10\rangle + |11\rangle\,\right] \\ &= |11\rangle\ . \end{aligned} \tag{7.39}$$

In a similar manner for $f = f_3$ and $f = f_4$ we obtain $|\psi\rangle = -|11\rangle$ and $|\psi\rangle = -|01\rangle$ respectively. Therefore, the final state of the two qubits is

$$\begin{aligned} |01\rangle \quad &\text{if} \quad f = f_1 \\ |11\rangle \quad &\text{if} \quad f = f_2 \\ -|11\rangle \quad &\text{if} \quad f = f_3 \\ -|01\rangle \quad &\text{if} \quad f = f_4. \end{aligned} \tag{7.40}$$

Thus, by comparing the final state of $|\psi\rangle$ with Eq. (7.40) we can identify whether the unknown f is f_1 or f_2 or f_3 or f_4.

The essential features of the above quantum algorithm are:

1. The crucial elements are the superposition and linearity of quantum mechanics. $|\psi\rangle$ in Eq. (7.40) is computed on the superposition states $|00\rangle$, $|01\rangle$, $|10\rangle$ and $|11\rangle$ simultaneously.

2. The final state $|\psi\rangle$ is due to an interference of various parts of the superposition.

3. If it is desired to know whether f is a constant ($f = f_1$ or f_4) or balanced ($f = f_2$ or f_3) then it is enough to measure the final state of the first qubit. If the first qubit is $|0\rangle$ then f is a constant. f is balanced if the first qubit is $|1\rangle$. Notice that the algorithm does not say whether f is f_1 or f_4 and f_2 or f_3. However, nowhere we learn about either $f(0)$ or $f(1)$. We are able to find out that the f is a constant or not by computing f once. In classical computation we must evaluate f twice before making a decision.

The quantum circuit to implement Deutsch's algorithm is given in Fig. 7.6. The input state is $|\psi_i\rangle = |01\rangle$ and the output is

$$|\psi_f\rangle = \pm|f(0)\oplus f(1)\rangle\left[\frac{|0\rangle - |1\rangle}{\sqrt{2}}\right]\ . \tag{7.41}$$

So, by measuring the first qubit, we may determine $f(0)\oplus f(1)$ and hence

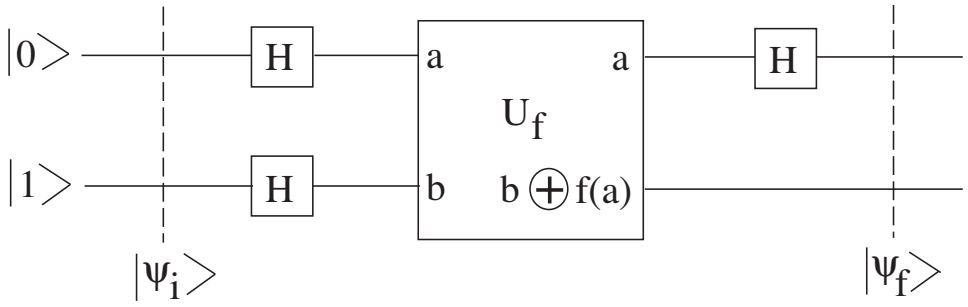


FIGURE 7.6 Quantum circuit to implement Deutsch's algorithm.

whether $f(x)$ is balanced or not. We cannot determine $f(x)$. But determine a global property of $f(x)$, namely, $f(0) \oplus f(1)$ with one evaluation of $f(x)$ only. So, a clever choice of function and final transformation allows efficient determination of useful information about the function. The above is achieved much faster compared to a classical computer.

7.5.2 Grover's Quantum Search Algorithm

In this subsection we explain the quantum search algorithm [12,16,17] and describe some of the exciting ways it can be used.

The kind of search problem that can be solved by a quantum search algorithm is following. Consider a function $f(x)$ with integer arguments 0 to N. Let the value of it be 0 everywhere except for $x = \omega$. The problem is to find ω using few calls to $f(x)$. This is analogous to finding the name of person in a telephone directory with the telephone number given. The data-base we wish to search is of size N. Classically, the probability of the value of a randomly chosen element to be ω is $1/N$. Therefore, to have a $50-50$ chance of getting ω we must call the data-base at least $N/2$ times. But a quantum algorithm can reduce the calls to approximately $\sqrt{N}$.

Lov Kumar Grover, a computer scientist at Lucent Technologies Bell Laboratories proposed a quantum search algorithm. In the following we discuss Grover's algorithm following mainly the review of Sudarshan [17]. We can model an oracle or a unitary operator $U_\omega(\lambda)$ as a black-box function $f(x)$. It computes $f(x)$ for an input x. It will return 1 if and only if $x = \omega$ and return 0 if $x \neq \omega$. A quantum circuit that has the ability to recognize solutions to the search problem is called a *quantum oracle* which is represented by the unitary operator U_ω.

We begin with the state $|0\,0\rangle$. The two zero's represent two registers of qubits where all the qubits are set to the 0 state. We can use Hadamard transformation to bring this initial state into superposition of states

$$|\phi\rangle = \frac{1}{\sqrt{N}} \left[\, |00\rangle + |10\rangle + |20\rangle + \cdots + |N-10\rangle \right] . \tag{7.42}$$

In matrix form, the transformation is given by Eq. (7.16). In Grover's algorithm the first register is assumed to be big enough to represent the largest element. In the second register there is only one qubit. By applying the Hadamard transformations on the individual qubits of the initial state we get

$$|\phi\rangle = \frac{1}{\sqrt{N}} \sum_{i=0}^{N-1} |i\rangle \left(\frac{|1\rangle - |0\rangle}{\sqrt{2}} \right). \tag{7.43}$$

The number of steps needed for this is $O(\log N)$. The second register is initialized to a state different from $|0\rangle$. The action of the oracle U_ω is

$$U_\omega |i, j\rangle = |i, j \oplus f(i)\rangle \, . \tag{7.44}$$

$|i\rangle$ is the index register, $|j\rangle$ is the oracle single qubit which is flipped if $f(i) = 1$ and unchanged otherwise. We can find whether i is a solution of the problem by preparing $|i\rangle|0\rangle$, applying the oracle and checking whether the oracle qubit is flipped to $|1\rangle$. We have

$$U_\omega |\phi\rangle = \frac{1}{\sqrt{N}} \left[\sum_{i \neq \omega} |i\rangle \left(\frac{|1\rangle - |0\rangle}{\sqrt{2}} \right) - |\omega\rangle \left(\frac{|1\rangle - |0\rangle}{\sqrt{2}} \right) \right] . \tag{7.45}$$

The action of U_ω on $|\phi\rangle$ is to change the sign of the component in the direction of $|\omega\rangle$. This reflects $|\phi\rangle$ in the Hilbert space of dimension N about the hyperplane orthogonal to $|\omega\rangle$. At this instant the value of ω is unknown to us. We can find the value of ω by consulting the oracle a certain minimum number of times.

Now, we construct another operator U_s which performs a reflection in such a way that the component of $|\phi\rangle$ along $|s\rangle$ is preserved and the signs of the component in the hyperplane perpendicular to $|s\rangle$ is changed. Here one iteration is the unitary transformation $R_{\text{Grov}} = U_s U_\omega$. Let θ be the angle between $|s\rangle$ and $|\omega\rangle$. Then the action of one iteration on $|\phi\rangle$ is to rotate its component along $|s\rangle$ through an angle 2θ that is away from the hyperplane perpendicular to the vector $|\omega\rangle$. Successive iterations with various choices of $|s\rangle$ makes $|\phi\rangle$ close to $|\omega\rangle$ and moreover away from the hyperplane perpendicular to $|\omega\rangle$. The number of queries required to obtain the correct value of $|\omega\rangle$ with large probability when $|\phi\rangle$ is measured after the iterations is $\pi\sqrt{N}/4$. So, Grover's algorithm has a quadratic speedup compared to the best classical algorithm.

7.5.3 Quantum Fourier Transform

The discrete Fourier transform is defined by

$$f_j = \frac{1}{\sqrt{N}} \sum_{k=1}^{N-1} \mathrm{e}^{\mathrm{i}2\pi jk/N} g_k. \tag{7.46}$$

This transforms a set of N numbers $\{g_0, g_1, \cdots, g_{N-1}\}$ (can be complex) into another set of numbers $\{f_0, f_1, \cdots, f_{N-1}\}$. The quantum Fourier transform $U_{\rm FT}$ is defined, on n qubits by its action on basis states $|j\rangle$ where $0 \leq j \leq 2^n - 1$, as

$$U_{\rm FT}|j\rangle \rightarrow \frac{1}{\sqrt{N}} \sum_{k=0}^{2^n-1} \mathrm{e}^{\mathrm{i}2\pi jk/N}|k\rangle \ . \tag{7.47}$$

It can be easily verified that $U_{\rm FT}$ is a unitary operator: The matrix of the transformation is $M(U_{\rm FT}) = [M_{jk}] = \mathrm{e}^{\mathrm{i}2\pi jk/N}/\sqrt{N}$. This transformation can be realized as a quantum circuit.

Many of the quantum algorithms are based on quantum Fourier transform. Shor's fast algorithm for factoring and discrete logarithm are two most interesting examples of algorithms based on the quantum Fourier transform. Classically, the fast Fourier transform takes about $N \log N = n2^n$ steps to Fourier transform $N = 2^n$ numbers. A quantum computer requires only n^2 steps. So, there is an exponential saving of time with a quantum computer compared to a classical computer. But it is to be noted that the set $\{f_j\}$ cannot be measured directly because a measurement would collapse each qubit into $|0\rangle$ or $|1\rangle$. Though quantum computation can be done more efficiently, creating the initial state $\{g_k\}$ and measuring the result are difficult.

7.5.4 Applications of Quantum Search

Let us point out some of the applications of quantum search.

1. An effective search algorithm for hard problems, like constrained optimization, is the so-called randomized algorithm. In it a set of random numbers is used to find a trajectory through some search space. Quantum search is able to speed-up randomized algorithms.

2. Quantum search can be applied to determine the statistical properties mean, variance, maxima and minima of functions, etc.

3. With quantum Fourier transform one can count effectively the number of possible solutions of a problem without finding them.

4. Quantum search is useful for experimental physicists to prepare desired superposition states. For example, to create a superposition of indices corresponding to prime numbers we can design an oracle $f(x)$ which returns 1 if x is a prime and 0 otherwise.

7.5.5 Shor's Algorithm

In 1994 Shor [9] developed an efficient quantum algorithm to compute the period of a periodic function. The period finding routine can be used to factorize large numbers in polynomial time. Consider the problem of factorizing a large

number N into exactly two large prime numbers [17]. Classically, to find the two prime numbers we have to check all the numbers from 1 to $\sqrt{N}$.

In Shor's algorithm, randomly a number $a < N$, $a^r = 1 \pmod N$ for an even integer value of r is chosen. It can be shown that for most choices of a, N shares a common factor having $a^{r/2} + 1$ or $a^{r/2} - 1$. Once r is found then applying a classical Euclid's algorithm one can easily compute the common factor of N and also $a^{r/2} \pm 1$. Thus, the problem of factorizing N is solved.

Let us choose

$$f_{N,a}(x) = a^x \pmod N, \quad x = 0, 1, 2, \cdots . \tag{7.48}$$

Because $a^r = 1 \pmod N$ the period of $f_{N,a}$ is r. Evaluate $f_{N,a}$ on a $|\phi\rangle$ given by

$$|\phi\rangle = \frac{1}{\sqrt{N}} \sum_{i=0}^{N-1} |i0\rangle \, . \tag{7.49}$$

Next, we setup a unitary oracle $U_{f_{N,a}}$ such that

$$U_{f_{N,a}}|\phi\rangle = \frac{1}{\sqrt{N}} \sum_{i=0}^{N-1} |ia^x (\text{mod } N)\rangle = |\psi\rangle \, . \tag{7.50}$$

The second register has a function with period r. Therefore, if we perform a measurement on it and obtain $|u\rangle$ then the first register will collapse to a linear combination of the values of x. This results in $f(x) = u$. Due to the periodicity of f these values of x form $x_0 + jr$, $j = 0, 2, \cdots, x_{\max}/r$ where $x_{\max}$ is the biggest number contained in the first register. We have $f(x_0) = u$. Suppose $t = x_{\max}/r$ is an integer. The measurement is found to reduce $|\psi\rangle$ to

$$|\phi\rangle = \frac{1}{\sqrt{x_{\max}/r}} \sum_{j=0}^{x_{\max}-1} |x_0 + jr\rangle \, |u\rangle \, . \tag{7.51}$$

Now, to get the value of r apply a quantum Fourier transform on the first register (of the state $|\phi\rangle$). The effect is

$$U_{\text{FT}}|\phi\rangle \rightarrow \sum_k \overline{f}(k) \, |k\rangle \, |u\rangle \, , \tag{7.52}$$

where $\overline{f}(k) = 1$ if k is a multiple of $x_{\max}/r$ otherwise 0. The value of k determined by the measurement of the first register will be of the form $k = \lambda x_{\max}/r$. λ and r are unknown. But if λ and r do not have a common factor then $k/x_{\max} = \lambda/r$ can be reduced to an irreducible fraction to read r and λ. On the other hand, if λ and r have a common factor then we conclude that the algorithm fails. In this case we repeat the analysis with another value of a. It is possible to show that the number of steps taken by the algorithm to get the correct answer is $O(\log N)$. This is indeed an exponential speed-up compared to the classical case.

7.5.6 Quantum Factorization of Integers

Let us describe the quantum factorization of integers by considering the number 20. First, choose a number a randomly such that the greatest common divisor of it and N is 1. Consider the periodic function

$$f(x) = a^x \pmod{N}, \quad x = 0, 1, \cdots . \tag{7.53}$$

For $N = 20$, select $a = 9$. Then from Eq. (7.53) we have

$$\begin{aligned} f(0) &= 1 \pmod{20} = 1. \\ f(1) &= 9 \pmod{20} = 9. \\ f(2) &= 9^2 \pmod{20} = 81 \pmod{20} = 1. \\ f(3) &= 9^3 \pmod{20} = 729 \pmod{20} = 9. \\ f(4) &= 9^4 \pmod{20} = 6561 \pmod{20} = 1. \end{aligned}$$

From the above, the period T of $f(x)$ is obtained as $T = 2$. This period can be determined by employing the method described earlier. To find N, calculate $Z = a^{T/2} = 9^1 = 9$. The greatest common divisor of $(Z+1, N) = (9+1, 20) = (10, 20)$ is 10. The greatest common divisor of $(Z-1, N) = (9-1, 20) = (8, 20)$ is 4. These two numbers 4 and 10 are factors of 20. In this way two factors of a number N can be obtained if the quantum algorithm gives the period T of $f(x)$.

7.6 FEATURES OF QUANTUM COMPUTATION

Some of the essential (but not sufficient) features of quantum computers [18] are summarized below:

1. Input, output, program and memory are represented by qubits.
2. A unitary transformation of the computer can represent a computation step.
3. All computations are reversible.
4. Only one-to-one operations are possible and therefore qubits cannot be copied.
5. The values of qubits may depend on the method used to infer them and on the co-measured qubits.
6. A measurement may be performed on any qubit at any stage of computation. However, a qubit cannot be measured by an experiment with a desired accuracy.
7. During a computation, a quantum computer proceeds all paths at once which when managed cleverly may speed-up the computation.

8. A subroutine should not leave any qubits over its computed answer. This is because the computational paths with different information cannot interfere.

In order to perform a quantum computation one should make proper use of the above features.

7.7 QUANTUM COMPUTATION THROUGH NMR

The essential requirement of a quantum computer are two-level isolated quantum systems. The physical systems explored so far to build quantum hardwares range from optical photons, cavity quantum electrodynamics, quantum dots, trapped ions to nuclear spins. The basic requirements of a quantum computer are:

1. The quantum states must be sufficiently isolated from the surroundings so that they have very low decoherence.

2. They must be made to evolve as per the unitary transformations performed.

3. It should be possible to prepare the initial state.

4. Suitable measurement technique must be devised for measuring quantum information because a measurement destroys quantum information and replaces it with classical information.

As a candidate for quantum computing, nuclear magnetic resonance (NMR) is attractive because of the spin's long coherence times and also due to the complexity of operations performed on modern spectrometers. Most atomic nuclei have spin and it causes them to act like tiny magnets. These nuclear magnets interact with magnetic fields thereby allowing them to be controlled with high precision. In certain cases, such as in hydrogen, this nuclear spin can assume two values, spin-up and spin-down. This is a two-state quantum system. Therefore, a hydrogen nucleus can be regarded as a qubit. In a molecule, different nuclei can be differentiated by their different resonance frequencies. Consequently the molecule can act as a quantum computer with each hydrogen providing one qubit. For example, naturally occurring cytosine has five hydrogen atoms in each molecule. It is in fact easy to replace three of them with deuterium, thereby leaving the two hydrogens to serve a two-qubit computer. This system with a conventional NMR spectrometer has been used to demonstrate certain quantum algorithms. Various nonselective pulses, transition and spin-selective pulses, rf gradients, coherence transfer via J-coupling and simultaneous multi-site excitation have been proposed to construct universal quantum gates and implement quantum algorithms for qubit systems. For some details see refs.[13,19-22].

7.8 WHY IS MAKING A QUANTUM COMPUTER EXTREMELY DIFFICULT?

If quantum computers would be so marvelous, *why don't we just build one?* There are several technical problems in setting up a quantum computer. We list some of them:

1. A notable serious problem is decoherence. It is the modification of the quantum state due to interaction with an environment. It can alter the value of a qubit that is uncontrollable.

2. Errors in classical information are discrete. In quantum information they are continuous.

3. To check whether errors have occurred, we must perform a quantum measurement. But a measurement will affect the state of the system. That is, errors cannot be diagnosed without introducing further errors.

4. To obtain the outcome of a computation, a readout system must carry out a measurement. Any imperfection in the measurement process gives rise to a readout error.

5. A transistor or any conventional computer element cannot be useful to perform quantum computation.

6. The various degrees of freedom of the device (such as the elastic vibrations of the device, the excitation of its conduction electrons, etc.) interact strongly with one another and also with the state of the device. As a result even approximate unitary evolution is impossible.

7.9 CONCLUDING REMARKS

Quantum algorithms for solving both linear and nonlinear differential, equations [23-27], quantum field theories [28] and simulation of sparse Hamiltonian systems [29], chemical dynamics [30] and electronic structure Hamiltonians [31] have been proposed.

Several groups working on quantum computation are focusing on lengthening the lifetime of the quantum bits of information and also quickening the pace of computation. Quantum computing using coherent photon conversion [32], fullerene based electron spin [33,34], trapped polar molecules [35], Josephson junction arrays [36], scanning tunneling microscopy [37], antiferromagnetic rings [38], one-dimensional optical lattice [39] and quantum walk [40] have been proposed. Simulation of electronic structure Hamiltonians [41], many-body Fermi systems [42], calculations of molecular properties [43], molecular energies [44] using quantum computers were reported. Implementation of Deutsch's algorithm on an ion-trap quantum computer [45] and experimental realization of it in a one-way quantum computer [46] have been

achieved. Magnetic resonance realization of decoherence-free quantum computation [47], performance of adiabatic quantum computation subject to decoherence [48], role of entanglement and correlations in mixed-state quantum computation [49], quantum discord and the power of one qubit [50] enhancement of quantum computation using quantum chaos [51] and geometric phase shift in quantum computation using superconducting nano circuits [52] were analyzed.

7.10 BIBLIOGRAPHY

[1] B. Schumacher, *Phys. Rev. A* 51:2738, 1995.

[2] G.P. Berman, G.D. Doolen, R. Mainieri and V.I. Tsifrinovich, *Introduction to Quantum Computers.* World Scientific, Singapore, 1998.

[3] M. Nielsen and I.L. Chuang, *Quantum Computation and Quantum Information.* Cambridge University Press, Cambridge, 2002.

[4] V. Sahni, *Quantum Computing.* Tata McGraw–Hill, New Delhi, 2007.

[5] P.A. Benioff, *Phys. Rev. Lett.* 48:1581, 1980.

[6] R. Feynman, *Found. Phys.* 16:507, 1986.

[7] R. Feynman, *Int. J. Theor. Phys.* 21:467, 1982.

[8] D. Deutsch, *Proc. Roy. Soc. London A* 400:97, 1985.

[9] P.W. Shor, "Algorithms for quantum computation: Discrete logarithms and factoring," in the Proceedings of 35th Annual Symposium on the Foundations of Computer Science. IEEE Press, Los Alamos, 1994.

[10] P.W. Shor, *SIAM J. Comp.* 26:1484, 1997.

[11] D. Deutsch and R. Jozsa, *Proc. Royal Soc. London A* 439:553, 1992.

[12] L.K. Grover, in Proceedings, 28th Annual ACM Symposium on the *Theory of Computation.* ACM Press, New York, 1996 pp. 212.

[13] C.H. Bennett, *Physics Today*, October 1995, pp. 24.

[14] T. Toffoli, *Reversible Computing. Technical Report MIT/LCS/TM-151*, 1980.

[15] D. Deutsch, A. Berenco and A. Ekert, *Proc. Royal Soc. London A* 449:669, 1995.

[16] L.K. Grover, *Phys. Rev. Lett.* 79:325, 1997.

[17] E.C.G. Sudarshan, *Current Science* 84:511, 2003.

[18] K. Svozil, *J. Univ. Comp. Sci.* 2:311, 1996.

[19] K. Dorai, T.S. Maohesh Arvind and A. Kumar, *Current Science* 79:1447, 2000.

[20] D.P. DiVincenzo, *Phys. Rev. A* 51:1015, 1995.

[21] D.G. Cory, M.D. Price and J.F. Havel, *Physica D* 120:82, 1998.

[22] N.A. Gershenfeld and I.L. Chuang, *Science* 275:350, 1997.

[23] S.K. Leyton and T.J. Osborne, arXiv:0812.4423, 2008.

[24] A.W. Harrow, A. Hassidin and S.L. Lloyd, *Phys. Rev. Lett.* 103:150502, 2009.

[25] X.D. Cai, Z.E. Su., M.C. Chen, M. Gu, M.J Zhu, L. Li, N.L. Liu, C.Y. Lu and J.W. Pan, *Phys. Rev. Lett.* 110:230501, 2013.

[26] B.D. Clader, B.C. Jacobs and C.R. Spouse, *Phys. Rev. Lett.*, 110:250504, 2013.

[27] D.W. Berry, *J. Phys. A: Math. Theor.* 4:105301, 2014

[28] S.P. Jordan, K.S.M. Lee and J. Preskill, *Science* 336:1130, 2012.

[29] D.W. Berry, G. Ahokas, R. Cleve and B.C. Sanders, *Commun. Math. Phys.* 270:359, 2007.

[30] J. Karsal, S.P. Jordan, P.J. Love, M. Mohseni and A.A. Guzik, *Proc. Natl. Acad. Sci.* 105:18681, 2008.

[31] J.D. Whitfield, J.D. Biamonte and A.A. Guzik, *Mol. Phys.* 109:735, 2011.

[32] N.K. Langford, S. Ramelow, R. Prevedel, W.J. Munro, G.J. Milburn and A. Zeilinger, *Nature* 478:360, 2011.

[33] W. Harneit, *Phys. Rev. A* 65:032322, 2002.

[34] S.C. Benjamin, A. Ardavan, G.A.D. Briggs, D.A. Britz, D. Gunlycke, J. Jefferson, M.A.G. Jones, D.F. Leigh, B.W. Lovett, A.N. Khobystov, S.A. Lyon, J.J.L. Morton, K.Porfyrakis, M.R. Sambrook and A.M. Tyryshkin, *J. Phys.: Condens. Matter* 18:S867, 2006.

[35] D. DeMille, *Phys. Rev. Lett.* 88:067901, 2002.

[36] L.B. Ioffe and M.V. Feigelman, *Phys. Rev. Lett.* 66:224503, 2002.

[37] G.P. Berman, G.W. Brown, M.E. Hawley and V.I. Tsifrinovich, *Phys. Rev. Lett.* 87:097902, 2001.

[38] F. Troiani, A. Ghirri, M. Affronte, S. Carretta, P. Santini, G. Amoretti, S. Piligkos, G. Timco and R.E.P. Winpenny, *Phys. Rev. Lett.* 94:207208, 2005.

[39] J.K. Pachos and P.L. Knight, *Phys. Rev. Lett.* 91:107902, 2003.

[40] A.M. Childs, *Phys. Rev. Lett.* 102:180501, 2009.

[41] J.D. Whitfield, J. Biamonte and A. Aspuru-Guzik, *Mol. Phys.* 109:735, 2011.

[42] D.S. Abrams and S. Lloyd, *Phys. Rev. Lett.* 79:2586, 1997.

[43] B.P. Lanyon, J.D. Whiifield, G.G. Gillett, M.E. Goggin, M.P. Almeida, I. Kassal, J.D. Biamonte, M. Mohseni, B.J. Powell, M. Barbieri, A. Aspuru-Guzik and A.G. White, *Nature Chemistry* 2:106, 2010.

[44] A. Aspuru-Guzik, A.D. Dutoi, P.J. Love and M. Head-Gordon, *Science* 309:1704, 2005.

[45] S. Gulde, M. Riebe, G.P.T. Lancaster, C. Becher, J. Eschner, H. Haffner, F. Schmidt Kaler, I.L. Chuang and R. Blatt, *Nature* 421:48, 2003.

[46] M.S. Tame, R. Prevedel, M. Paternostro, P. Bohi, M.S. Kim and A. Zeilinger, *Phys. Rev. Lett.* 98:140501, 2007.

[47] J.E. Ollerenshaw, D.A. Lidar and L.E. Kay, *Phys. Rev. Lett.* 91:217904, 2003.

[48] M.S. Sarandy and D.A. Lidar, *Phys Rev. Lett.* 95:250503, 2005.

[49] A. Datta and G. Vital, *Phys. Rev. A* 75:042310, 2007.

[50] A. Datta, A. Shaji and C.M. Caves, *Phys. Rev. Lett.* 100:050502, 2008.

[51] T. Prosen and M. Znidaric, *J. Phys. A: Math. Gen.* 34:L681, 2001.

[52] S.L. Zhu and Z.D. Wang, *Phys. Rev. A* 66:042322, 2002.

7.11 EXERCISES

7.1 Assume that a qubit can be expressed as $|\psi\rangle = \cos(\theta/2)\,|0\rangle + \sin(\theta/2)\,\mathrm{e}^{\mathrm{i}\phi}|1\rangle$ with $\theta \in [0, \pi]$ while $\phi \in [0, 2\pi]$. Express the two qubits $|\psi\rangle|\phi\rangle$ in separable form and also find $|\psi\rangle^{\otimes 2}$.

7.2 Obtain the matrix representation of the Hadamard gate.

7.3 Express the Hadamard gate in terms of the Pauli matrices $\sigma_1 = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$ and $\sigma_3 = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$.

7.4 Determine α, β, γ and δ of the decomposition matrices for the Hadamard gate.

7.5 Find the unitary matrix for the two qubit gate given in Fig. 7.7 and show that it is equivalent to controlled NOT gate.

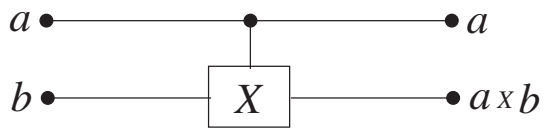


FIGURE 7.7 An equivalent controlled NOT gate.

7.6 If X, H and Z denote the quantum NOT, Hadamard and Z gates respectively, find HZH.

7.7 Consider the initial state

$$|\psi\rangle = \frac{\lambda}{\sqrt{2}}(|000\rangle + |011) + \frac{\mu}{\sqrt{2}}(|100\rangle + |111\rangle).$$

Find the state after applying a CNOT gate (using the first qubit as the control qubit and the second as the target) followed by Hadamard gate on first qubit.

7.8 Suppose $M_0 = |0\rangle\langle 0| = \begin{pmatrix} 1 \\ 0 \end{pmatrix}\begin{pmatrix} 0 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix}$ and $M_2 = \begin{pmatrix} 0 \\ 1 \end{pmatrix}\begin{pmatrix} 0 & 1 \end{pmatrix} = \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix}$ are two measurement operators. If $|\psi\rangle = a|0\rangle + b|1\rangle$ find the probability of measuring $|0\rangle$.

7.9 If $S = \begin{pmatrix} 1 & 0 \\ 0 & \mathrm{i} \end{pmatrix}$ is a quantum phase gate then form its truth table.

7.10 Form the truth table for the T gate defined as $T = \begin{pmatrix} 1 & 0 \\ 0 & \mathrm{e}^{\mathrm{i}\pi/4} \end{pmatrix}$.

7.11 Consider the two circuits shown in Fig. 7.8. The left-side circuit is the swap gate which exchanges the state of two qubits. Show that the two circuits in Fig. 7.8 are equivalent.

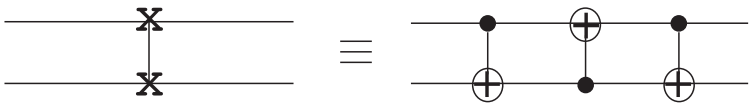


FIGURE 7.8 Two equivalent circuits.

7.12 In the notation $|a\rangle|b\rangle = |ab\rangle$ where a is the control bit and b is the target bit, find the outputs of the quantum circuit shown in Fig. 7.9 for the inputs $|00\rangle$, $|01\rangle$, $|10\rangle$ and $|11\rangle$.

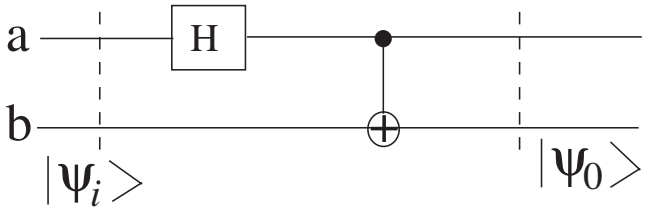


FIGURE 7.9 A quantum circuit with a Hadamard gate.

7.13 Construct the truth table for the Toffoli gate shown in Fig. 7.10 and show that it stimulates NAND gate.

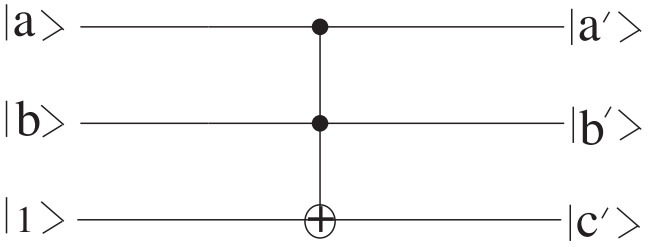


FIGURE 7.10 A Toffoli gate.

7.14 Find the output of the circuit given in Fig. 7.11, which can be used to implement the Deutsch's algorithm, for the input $|\psi_i\rangle = |0\rangle\,|1\rangle = |01\rangle$.

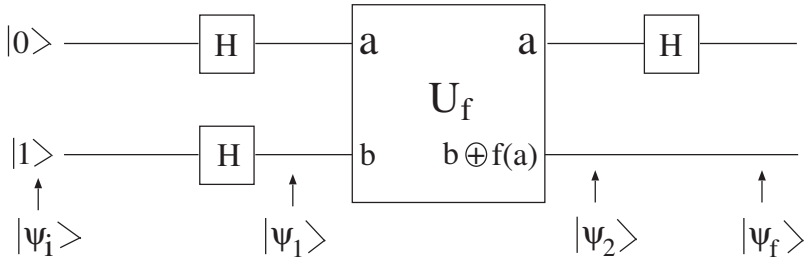
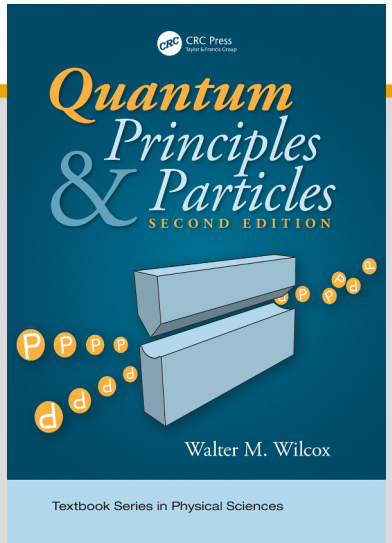


FIGURE 7.11 A quantum circuit of Deutsch's algorithm.

CHAPTER

3

QUANTUM COMPUTING



This chapter is excerpted from

Quantum Principles and Particles, Second Edition

by Walter Wilcox

Learn more

Appendix G: Quantum Computing

G.1 INTRODUCTION

It is strongly suspected that computers that function as quantum mechanics simulators, so-called quantum computers, will prove to be inherently more powerful than conventional or classical computers. The main reason for this is that quantum computers, whose functions are based on manipulating coherent quantum amplitudes, will have a huge advantage in algorithmic speed. A number of fundamental algorithms require fewer logical steps to solve on a quantum computer than its conventional counterpart. Another intrinsic advantage is in the expansion of the memory space of computations. The state space of such machines will someday overwhelm conventional machines in terms of the storage and manipulation of large data sets.

Quantum computers are now a reality, but we are only in the initial stages of understanding the things such machines are capable of doing. I cannot in any way do justice to the burgeoning field of quantum computation and information here, but I will attempt to impart some of the excitement of the concepts involved and make connections to helpful references that can paint a more complete picture. In any case, a student in quantum mechanics is actually in a very advantageous position to understand, utilize, and manipulate such new resources, and so a modest introductory tutorial on this topic seems very appropriate.

G.2 COMPUTATIONAL BASICS

What is a quantum computer?* First, let us distinguish it from what we are terming a classical computer. A classical computer works its way serially through a problem, one step at a time. This mode of operation has been greatly improved upon over the past several decades by incorporating the concept of *parallel processing*: a problem can be broken into pieces and many independent calculations can be done simultaneously. However, a quantum computer will do better: it can manipulate all possibilities at once without the need to break the problem into pieces with the attendant overheads. The classical computer works on irreversible steps of logic, pulling items out of an inventory, whereas the quantum computer works on reversible global amplitudes that are immediately accessible. The logic necessary for a quantum computer is a generalization of the ways we learned how to manipulate spin 1/2 amplitudes in Chapter 1. Quantum computation involves all the concepts we learned about in our quantum excursions: indeterminism, interference, uncertainty, and superposition. There will only be a change in terminology here, with no change in the underlying quantum concepts. Mathematically, quantum computation is just an application of linear algebra, but it is important to get used to the language in this new field.

Let us first think about the basic algorithmic elements of a quantum computer. The basic computational component of a classical computer is the bit, which is a logical unit that can take on only two values, conventionally labeled 0 and 1, whereas the basic unit on a quantum computer

* See the early overview article for a more eloquent orientation: L. Grover, "Quantum Computing," *The Sciences*, July/August 1999, 24–30.

is a two-valued *qubit* (*qu*antum *bit*), whose general structure we learned about in Chapter 1. The general state of this qubit $|\psi\rangle$ is the linear superposition

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle, \tag{G.1}$$

where bra–ket notation is being used, and the states are orthonormal: $\langle 0|0\rangle = \langle 1|1\rangle = 1$, $\langle 0|1\rangle = \langle 1|0\rangle = 0$. α and β are complex numbers with the normalization condition: $\langle\psi|\psi\rangle = |\alpha|^2 + |\beta|^2 = 1$. Thus $|0\rangle$ is simply "spin up" ($|0\rangle \equiv |+\rangle$ from Chapter 1), and $|1\rangle$ is "spin down" ($|1\rangle \equiv |-\rangle$):

$$|0\rangle \leftrightarrow \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \; |1\rangle \leftrightarrow \begin{pmatrix} 0 \\ 1 \end{pmatrix}. \tag{G.2}$$

The states $|0\rangle$ and $|1\rangle$ are called the *computational basis states*, and are the analog of the bits 0 and 1 in conventional computer logic. The column–matrix equivalent form of the state in Equation G.1 is of course

$$\psi = \begin{pmatrix} \alpha \\ \beta \end{pmatrix}. \tag{G.3}$$

After normalization, the qubit parameter space (or manifold) is characterized by two numbers. Going back to bra–ket notation, the most general state may be written

$$|\psi\rangle = \left(\cos\frac{\theta}{2}|0\rangle + e^{i\phi} \sin\frac{\theta}{2}|1\rangle \right) \tag{G.4}$$

(cf. Equations 1.80 and 1.222 in Chapter 1 and Equation 10.100 in Chapter 10) outside of an irrelevant overall phase, for the real parameters θ and ϕ. Based upon our experience with spin 1/2, we know that θ and ϕ may be interpreted as the spherical coordinate direction angles associated with the vector pointing in the geometrical "up" spin direction; θ is the polar angle and ϕ is the azimuthal one. This one-to-one mapping of the state in Equation G.4 to a direction in an internal space is embodied in the so-called *Bloch sphere*, shown in Figure G.1.

Defining a set of labels x_i=(0,1) for qubit i, a direct product of qubits, $|x_1, x_2, ...\rangle \equiv |x_1\rangle \otimes |x_2\rangle \otimes ...$, may be used as computational basis states for the computer state just as a conventional spin basis for a set of particles, $|m_1, m_2, ...\rangle \equiv |m_1\rangle \otimes |m_2\rangle \otimes ...$, can be used for the spin state, just as we studied in Chapter 8. So, for a two-qubit system, the basis states can be taken to be $|0,0\rangle$, $|1,0\rangle$, $|0,1\rangle$, and $|1,1\rangle$. The state of such a system can be in a mixture of such basis states, for example

$$|\psi\rangle = \alpha|0,0\rangle + \beta|1,0\rangle + \chi|0,1\rangle + \delta|1,1\rangle, \tag{G.5}$$

where $|\alpha|^2 + |\beta|^2 + |\chi|^2 + |\delta|^2 = 1$. Such a multiqubit state is said to be "entangled."

Now consider a system of n qubits. The number of different states of such a system is 2^n. The computational memory of a conventional one terabyte (10^{12} bytes) machine would be approximately equivalent to a quantum computer with only $n = 40$ qubits. Computer hardware has improved at an exponential rate for well over 50 years. A conventional encapsulation of this

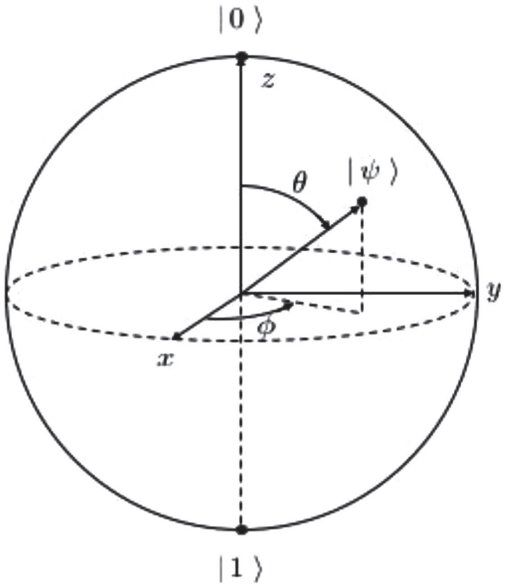


FIGURE G.1 The Bloch sphere representation of the state of a single qubit.

progress is embodied in Moore's law,* which states that market forces drive a doubling in computational power roughly every 2 years. However, this level of advancement in conventional memory would be equivalent to simply adding a single additional qubit to a quantum computer in the same amount of time!

A new computational field has emerged in which conventional computers are used to simulate quantum ones. Such simulations can be used for testing and reliability considerations. Conventional parallel computers with better algorithms have so far kept pace with their quantum cousins. However, a point will come where quantum computers, which are capable of doing 2^n mathematical operations simultaneously, will outpace all conventional ones. This point of separation is termed *quantum supremacy*.

G.3 SOME LOGIC GATE CONCEPTS

In conventional computers, logic is performed by so-called *logic gates* on bits. Four examples are shown in Figure G.2. The NOT gate is the only single-bit logic gate. The three others are called AND, OR, and NOR gates. They have incoming bits labeled *a* and *b*; the output bit is determined by the logic of the gate, which is represented by a symbol. Notice that in the last three cases the action is irreversible: 2 bits in, 1 bit out. In each case, an associated *truth table* is also given for the gate. The horizontal lines represent the causal flow from left to right.

Quantum logic gates are fundamentally different. The number of gates on input and output are always the same. In contrast to the conventional computer single-bit gate, there are an unlimited number of single-qubit logic gates. In fact, the operations on single-qubit gates are equivalent to 2×2 unitary matrices U in linear algebra, and the action on the gates is always reversible.

* Named after Gordon Moore, a computer entrepreneur who made his observation/forecast in 1975. More technically, the law involves the number of transistors on a central processing unit (CPU). This rate has held roughly constant until the present (2019), but this trend must end eventually, as microprocessors are reaching the limits of the classical world, where quantum indeterminacy is definitely not an advantage. The latest prognostication is that the law will last until around 2025.

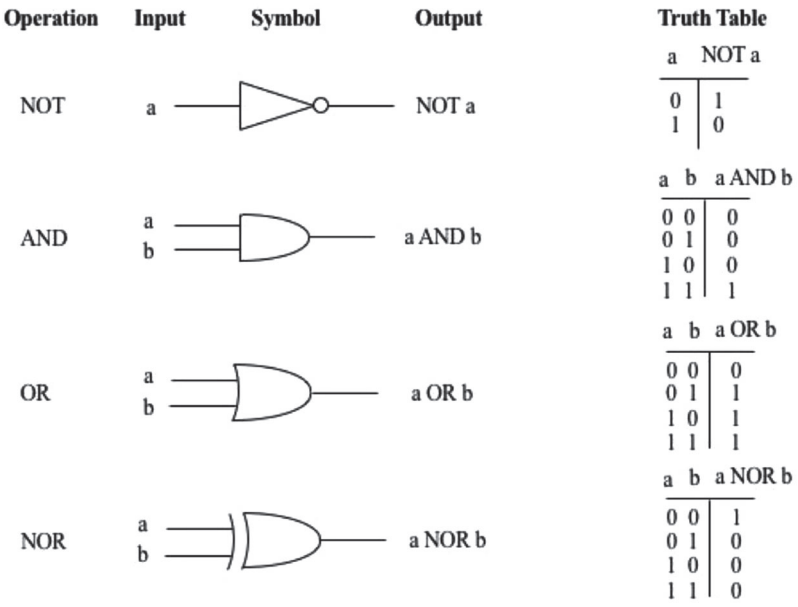
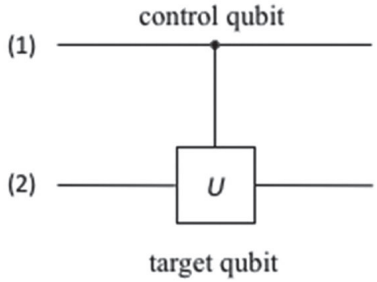


FIGURE G.2 Some standard logic gates for conventional computers.

Some standard single-qubit operations and their associated unitary transformations are given in Figure G.3. Note that such operations are specified by the action on the computational basis $|x_1,x_2\rangle \rightarrow U\,|x_1,x_2\rangle$, which is analogous to the truth tables for conventional logic gates. Multiple operations may be performed on a qubit, with the right-flowing causal order indicated by a horizontal line. We found it useful in Chapter1 to explain the action of operators on quantum ket-states by the use of Process Diagrams, in which there was an understood causal flow of symbols from right to left. Similarly, workers in the quantum information field have found it useful to illustrate multiple qubit operations as such a flow. Qubits are conventionally shown as flowing horizontal lines, with operations taking place on them in causal order from left to right.

In addition to the single-qubit operations, there are also operations involving multiple qubits. One of the two-qubit types of logic gates is called *controlled* operations, in that one qubit state is used to control the output of another qubit. The symbolic form of the *controlled-U* operation is shown next. Note that the gates are labeled from top to bottom.



The *control qubit* is used to set a condition, whereas the *target qubit* is conditionally modified with one of the single-qubit logic gates, U. The control qubit is said to be "set" when its computational basis x-value is 1. That is, when qubit 1 is the control and qubit 2 is the target, one has the controlled $U^{(1)}$ unitary operation:

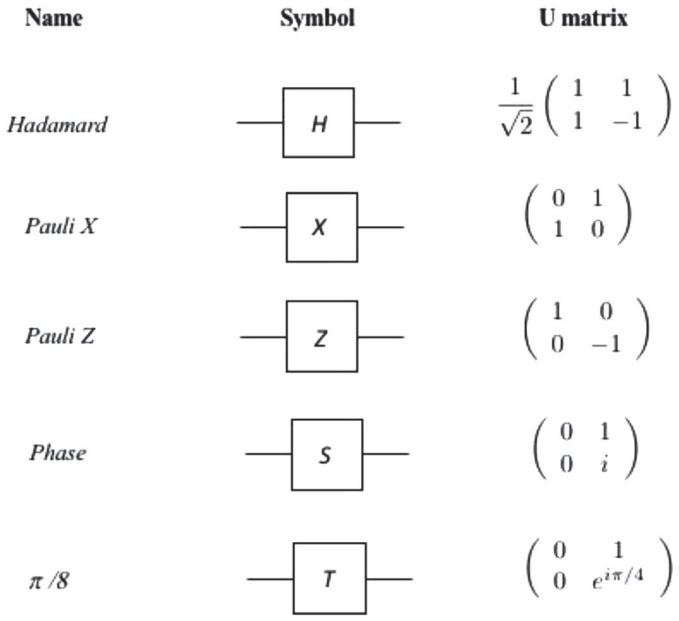


FIGURE G.3 Some single-qubit logic gates and associated matrices.

$$|x_1,x_2\rangle \xrightarrow{U^{(1)}} \begin{cases} |x_1,x_2\rangle,\ x_1=0, \\ |x_1\rangle U|x_2\rangle,\ x_1=1. \end{cases} \tag{G.6}$$

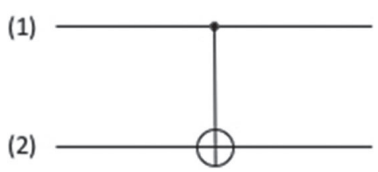
When control and target are reversed, one has the controlled $U^{(2)}$ operation:

$$|x_1,x_2\rangle \xrightarrow{U^{(2)}} \begin{cases} |x_1,x_2\rangle,\ x_2=0, \\ U|x_1\rangle|x_2\rangle,\ x_2=1. \end{cases} \tag{G.7}$$

A fundamental controlled two-qubit operation, called CNOT, is defined by

$$|x_1,x_2\rangle \xrightarrow{\text{CNOT(1)}} |x_1,x_1\oplus x_2\rangle, \tag{G.8}$$

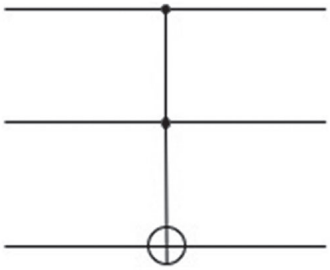
when the first qubit is the control, and where the addition of the labels on the target qubit, $x_1 \oplus x_2$, is done in *mod(2)* arithmetic. It is symbolized by



This operation is actually equivalent to a controlled *Pauli X* operation:

The equivalence is the subject of Problem G.3.

There are many other conditional qubit operations involving three or more qubits. For example, the following gate symbol shows a *Toffoli gate*, which is just a multiqubit generalization of the two-qubit CNOT gate. In this case, both control gates must be "set" for the third gate to act.



G.4 OUTLINE OF ALGORITHMS

What are some of the known algorithms that quantum computers are expected to excel at?*

- **Quantum Fourier transform**. We have already encountered the quantum version of the traditional Fourier transform in Appendix B, Equations B.9 and B.10. It is a unitary transformation on qubits, and so can be considered a logic gate for quantum information. For a set of 2^n qubits, the number of logic gates necessary is of order n^2. This beats the traditional *Fast Fourier Transform* (FFT) that requires $n2^n$ classical logic gates. The efficient implementation of the quantum Fourier transform is a key ingredient in many quantum algorithms, including the following.

- **Factorization of numbers**. Calculating the prime factors of a very large odd composite number N is considered essentially impossible for a conventional computer, but turns out to be considerably less computationally intensive for a quantum one. A quantum computer uses $\sim (\log N)^2(\log\log N)(\log\log\log N)$ (base 2 logarithms) logic gates, whereas a conventional computer, using a so-called *number field sieve*, uses $\sim \exp(c(\log N)^{1/3}(\log\log N)^{2/3})$ gates, where $c \approx 1.9$. Such factorizations, which are easy to assign but hard to decode, are used every day for commercial encryption purposes, which means that quantum computers could decipher such communications. This is obviously an impetus for the development of study of such machines! The quantum factorization code was developed by Peter Shor in 1994.†

- **Quantum search algorithms**. The problem of doing searches through data is intrinsically more efficient on quantum computers than classical ones. It can be shown that a search through N items can be done with $\sqrt{N}$ logic steps as opposed to the classical

* See the Wiki "Quantum Algorithms" (https://en.wikipedia.org/wiki/Quantum_algorithm) for a more detailed list.

† P. Shor, "Algorithms for Quantum Computation: Discrete Logarithms and Factoring," *Proceedings of the 35th Annual Symposium on Foundations of Computer Science*, IEEE Press (1994).

number of steps N. This algorithm can also be used to speed up cryptographic searches. The algorithm was developed by Lov Grover in 1996.*

- **Quantum simulations**. Is it possible to do an actual quantum simulation† of quantum systems that arise in atomic, nuclear, or particle physics? Classical simulations of quantum systems use the Monte Carlo algorithm described in Appendix E. This is a way of summing over a huge set of possible states. Is it possible to simulate such theories directly on a set of spatially arrayed qubits? This possibility was speculated upon by Feynman in a visionary talk in 1981.‡ His point was that to simulate the responses of quantum systems, the best thing to do is to make them quantum mechanical also. There is an impressive amount of work in this direction already.§

G.5 COMPUTER REALIZATIONS

How does one build a quantum computer? One needs a physical realization of the qubit, an ability to maintain its quantum state, as well as an ability to control their interactions. In addition, a quantum computer has to be completely isolated from its environment if it is to maintain its quantum state. One also needs a way to control input and output. Such machines will in fact be hybrid devices with classical interfaces controlling and interacting with the quantum computing device. Many companies and organizations are working on such devices, and many different ideas are being tested. The race is on to build these machines and demonstrate their usefulness.

It is likely that anything one writes concerning the operational state of these devices will be found to be obsolete within a few years. However, there will always be a need to summarize existing technologies for the incoming student. As of the writing of this appendix (2019), the leading edge of the maturing technologies seems to be based upon the following types of devices.

- **Superconducting Josephson circuits**. Superconducting circuits called Josephson circuits or junctions may be put into a coherent state between various isolated low-lying charge, flux, or phase states. Such devices can be conveniently designed and implemented on integrated chips.
- **Trapped ions**. Individual charged atoms or ions can be trapped using an oscillating electric field in a so-called Paul trap. Tuning with lasers can also place the atoms in a superposition of two of their standard atomic states.
- **Quantum dots**. Small spheres can be fabricated on superconducting silicon chips on which an electron is placed, whose operation is controlled by microwaves. The electronic spin state or electron position state in a quantum potential well can form the qubits.
- **Topological quasiparticles**. Two-dimensional quasiparticle combinations called *anyons* may be formed from electrons on braided integrated circuits. The braiding forms topological structures with conserved quantum numbers.

---

* L. Grover, "A Fast Quantum Mechanical Algorithm for Database Search", *Proceedings of the 28th Annual ACM Symposium on the Theory of Computing*, ACM Press, 212 (1996). L. Grover, "Quantum Mechanics Helps in Searching for a Needle in a Haystack," *Phys. Rev. Lett.* **79**, 325 (1997).

† T. Johnson, S. Clark, and D. Jaksch, "What Is a Quantum Simulator?" *EPJ Quantum Technology* **1**, 10 (2014).

‡ R. Feynman, "Simulating Physics with Computers," *Int. J. Theor. Phys.* **21**, 467 (1982). Such devices were also considered in Yu. Manin, "Computable and Noncomputable" (in Russian), *Sov. Radio.* **13** (1980).

§ See, for example, the articles in A. Trabesinger, ed., "Quantum Simulation," *Nat. Phys.* **8** (2012). A recent imposing simulation is described in J. Zhang et al., "Observation of a Many-Body Dynamical Phase Transition with a 53-Qubit Quantum Simulator," *Nature* **551**, 601 (2017).

This is just a mere sampling of the technologies that are being tested.* The list given now for the leading candidates may be completely different in just a few years!

A major problem with such machines is the tendency for the quantum state created to lose quantum coherence, or "decohere." It turns out the system decoherence time decreases as the number of qubits increases. The decoherence time for n qubits is roughly $1/n$ times shorter than the decoherence time for an individual qubit.† Maintaining a quantum state in the macroscopic world is hard to do! The decoherence time for Josephson circuits is the shortest of the technologies listed, on the order of microseconds, whereas the decoherence time for trapped ions is the greatest, on the order of seconds. Decoherence times for quantum dots falls somewhere in between, and it is thought that the quasiparticle approach could provide a very stable system. Besides the coherence issue, one must also consider the timescale necessary for individual gate processes to occur, which vary greatly over the various technologies. The time necessary for the algorithm to complete is the gate process speed times the number of gates, and this must be much smaller than the system decoherence time for a workable machine.

Part of the cure for the decoherence issue are quantum error-correction and fault tolerance codes. A major theoretical discovery in the 1990s was that quantum computers could reliably compute despite the imperfections and faults introduced by quantum noise. The theory of such quantum error protection is remarkably well developed. The basic idea is to fight entanglement with entanglement and redundancy. Amazingly, it turns out that reliable computation can be performed with faulty logic gates, as long as the probability of error on gates is below a certain threshold. However, the reliability encoding thus introduced means many more qubits must be introduced to control the "working" ones. In order to factor a 300 digit number ($\approx 2^{10^3}$) using Shor's algorithm, it is optimistically estimated‡ to require about 10^4 qubits, but with error correction and encoding, it could take $10^3 \times 10^4 = 10^7$ qubits to actually work. Thus, unless qubits turn out to be easily protectable (as is possible with the quasiparticle approach) or as easy to form as transistors on an integrated circuit, this could be a problem!

G.6 REFERENCES

A short list of some popular and recent introductory textbooks on quantum computing follows.

P. Kaye, R. Laflamme, and M. Mosca, *An Introduction to Quantum Computing* (Oxford University Press, 2007).

N. Mermin, *Quantum Computer Science: An Introduction* (Cambridge University Press, 2007).

M. Nielson and I. Chuang, *Quantum Computation and Quantum Information* (Cambridge University Press, 2010).

E. Rieffel and W. Polak, *Quantum Computing: A Gentle Introduction* (MIT Press, 2011).

N. Yanofsky and M. Mannucci, *Quantum Computing for Computer Scientists* (Cambridge University Press, 2008).

PROBLEMS

G.1 The qubit state

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

* See the Wiki "Quantum computing" (https://en.wikipedia.org/wiki/Quantum_computing).

† M. Dyakonov, "Is Fault-Tolerant Quantum Computation Really Possible?" *Future Trends in Microelectronics Workshop*, 4–18 (Wiley, 2007).

‡ Dyakonov, "Fault-Tolerant Quantum Computation."

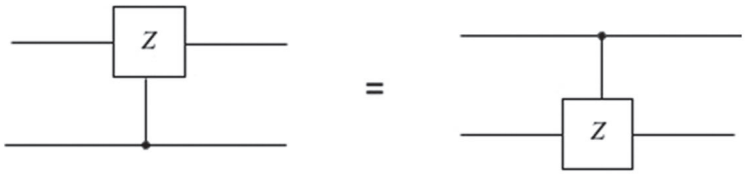
is represented on the Bloch sphere at the point (1,0,0) in (x, y, z) coordinate space, because comparing to Equation G.4 one has $\theta = \pi/2$ and $\phi = 0$. Given this initial state, find the action of the single-qubit operators H, Z, S, and T on $|\psi\rangle$ and the associated final position on the Bloch sphere in (x, y, z) space.

G.2 Show the single-qubit identity

$$HZH = X.$$

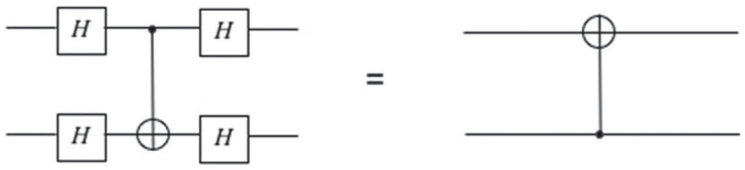
G.3 Show that the CNOT operation is equivalent to the controlled *Pauli X* operation, as discussed and illustrated in Section G.3.

G.4 Show the identity



That is, it is possible to interchange control and target qubits for this operation?

G.5 Show the symbolic identity



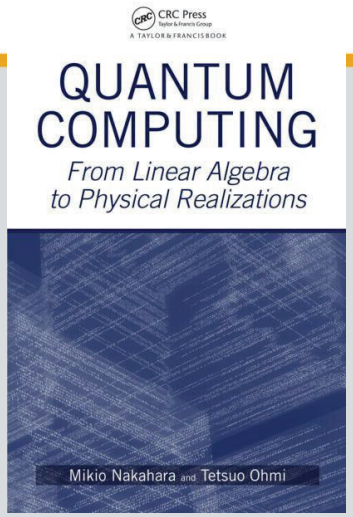
Notice the CNOT operation has switched sides.

G.6 Prove directly that the quantum Fourier transform defined in Appendix B (Equations B.9 and B.10) is unitary and therefore can be considered a qubit gate operation.

CHAPTER

4

QUANTUM GATES, QUANTUM CIRCUIT AND QUANTUM COMPUTATION



This chapter is excerpted from

Quantum Computing

From Linear Algebra to Physical Realizations

by Mikio Nakahara, Tetsuo Ohmi

Learn more

4

Quantum Gates, Quantum Circuit and Quantum Computation

4.1 Introduction

Now that we have introduced qubits to store information, it is time to consider operations acting on them. If they are simple, these operations are called gates, or more precisely **quantum gates**, in analogy with those in classical logic circuits. More complicated **quantum circuits** are composed of these simple gates. A collection of quantum circuits for executing a complicated algorithm, a **quantum algorithm**, is a part of a quantum computation.

DEFINITION 4.1 (**Quantum Computation**) A quantum computation is a collection of the following three elements:

(1) A register or a set of registers,

(2) A unitary matrix U, which is taylored to execute a given quantum algorithm, and

(3) Measurements to extract information we need.

More formally, we say a quantum computation is the set $\{\mathcal{H}, U, \{M_m\}\}$, where $\mathcal{H} = \mathbb{C}^{2^n}$ is the Hilbert space of an n-qubit register, $U \in \mathrm{U}(2^n)$ represents the quantum algorithm and $\{M_m\}$ is the set of measurement operators.

The hardware (1) along with equipment to control the qubits is called a quantum computer.

Suppose the register is set to a fiducial initial state, $|\psi_{\mathrm{in}}\rangle = |00\ldots0\rangle$, for example. A unitary matrix U_{alg} is designed to represent an algorithm which we want to execute. Operation of U_{alg} on $|\psi_{\mathrm{in}}\rangle$ yields the output state $|\psi_{\mathrm{out}}\rangle = U_{\mathrm{alg}}|\psi_{\mathrm{in}}\rangle$. Information is extracted from $|\psi_{\mathrm{out}}\rangle$ by appropriate measurements.

Actual quantum computation processes are very different from those of a classical counterpart. In a classical computer, we input the data from a keyboard or other input devices and the signal is sent to the I/O port of the computer, which is then stored in the memory, then fed into the microprocessor, and the result is stored in the memory before it is printed or it is

displayed on the screen. Thus information travels around the circuit. In contrast, information in quantum computation is stored in a register, first of all, and then external fields, such as oscillating magnetic fields, electric fields or laser beams are applied to produce gate operations on the register. These external fields are designed so that they produce desired gate operation, i.e., unitary matrix acting on a particular set of qubits. Therefore the information sits in the register and they are updated each time the gate operation acts on the register.

One of the other distinctions between classical computation and quantum computation is that the former is based upon digital processing and the latter upon hybrid (digital + analogue) processing. A qubit may take an arbitrary superposition of $|0\rangle$ and $|1\rangle$, and hence their coefficients are continuous complex numbers. A gate is also an element of a relevant unitary group, which contains continuous parameters. An operation such as "rotate a specified spin around the x-axis by an angle π" is implemented by applying a particular pulse of specified amplitude, angle and duration. These parameters are continuous numbers and always contain errors. These aspects might cause challenging difficulties in a physical realization of a quantum computer.

Parts of this chapter depend on [1, 2] and [3].

4.2 Quantum Gates

We have so far studied the change of a state upon measurements. When measurements are not made, the time evolution of a state is described by the Schrödinger equation. The system preserves the norm of the state vector during time evolution. Thus the time development is unitary. Let U be such a time-evolution operator; $UU^\dagger = U^\dagger U = I$. We will be free from the Schrödinger equation in the following and assume there exist unitary matrices which we need. Physical implementation of these unitary matrices is another important area of quantum information processing and is a subject of the second part of this book.

One of the important conclusions derived from the unitarity of gates is that the computational process is reversible.

4.2.1 Simple Quantum Gates

Examples of quantum gates which transform a one-qubit state are given below. We call them one-qubit gates in the following. Linearity guarantees that the action of a gate is completely specified as soon as its action on the basis $\{|0\rangle,\ |1\rangle\}$ is given. Let us consider the gate I whose action on the basis vectors are defined by $I:\ |0\rangle \to |0\rangle,\ |1\rangle \to |1\rangle$. The matrix expression of this

gate is easily found as

$$I = |0\rangle\langle 0| + |1\rangle\langle 1| = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}. \tag{4.1}$$

Similarly we introduce $X:\ |0\rangle \to |1\rangle,\ |1\rangle \to |0\rangle$, $Y:\ |0\rangle \to -|1\rangle,\ |1\rangle \to |0\rangle$, and $Z:\ |0\rangle \to |0\rangle,\ |1\rangle \to -|1\rangle$, whose matrix representations are

$$X = |1\rangle\langle 0| + |0\rangle\langle 1| = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} = \sigma_x, \tag{4.2}$$

$$Y = |0\rangle\langle 1| - |1\rangle\langle 0| = \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix} = -i\sigma_y, \tag{4.3}$$

$$Z = |0\rangle\langle 0| - |1\rangle\langle 1| = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} = \sigma_z. \tag{4.4}$$

The transformation I is the trivial (identity) transformation, while X is the negation (NOT), Z the phase shift and $Y = XZ$ the combination of them. It is easily verified that these gates are unitary.

The **CNOT** (**controlled-NOT**) gate is a two-qubit gate, which plays quite an important role in quantum computation. The gate flips the second qubit (the **target qubit**) when the first qubit (the **control qubit**) is $|1\rangle$, while leaving the second bit unchanged when the first qubit state is $|0\rangle$. Let $\{|00\rangle, |01\rangle, |10\rangle, |11\rangle\}$ be a basis for the two-qubit system. In the following, we use the standard basis vectors with components

$$|00\rangle = (1,0,0,0)^t,\ |01\rangle = (0,1,0,0)^t,\ |10\rangle = (0,0,1,0)^t,\ |11\rangle = (0,0,0,1)^t.$$

The action of the CNOT gate, whose matrix expression will be written as U_{CNOT}, is

$$U_{\mathrm{CNOT}} : |00\rangle \mapsto |00\rangle,\ |01\rangle \mapsto |01\rangle,\ |10\rangle \mapsto |11\rangle,\ |11\rangle \mapsto |10\rangle.$$

It has two equivalent expressions

$$\begin{aligned} U_{\mathrm{CNOT}} &= |00\rangle\langle 00| + |01\rangle\langle 01| + |11\rangle\langle 10| + |10\rangle\langle 11| \\ &= |0\rangle\langle 0| \otimes I + |1\rangle\langle 1| \otimes X, \end{aligned} \tag{4.5}$$

having a matrix form

$$U_{\mathrm{CNOT}} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}. \tag{4.6}$$

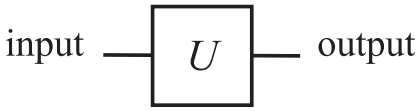
The second expression of the RHS in Eq. (4.5) shows that the action of U_{CNOT} on the target qubit is I when the control qubit is in the state $|0\rangle$, while it is σ_x when the control qubit is in $|1\rangle$. Verify that U_{CNOT} is unitary and, moreover, idempotent, i.e., $U_{\mathrm{CNOT}}^2 = I$.

Let $\{|i\rangle\}$ be the basis vectors, where $i \in \{0, 1\}$. The action of CNOT on the input state $|i\rangle|j\rangle$ is written as $|i\rangle|i \oplus j\rangle$, where $i \oplus j$ is an addition mod 2, that is, $0 \oplus 0 = 0, 0 \oplus 1 = 1, 1 \oplus 0 = 1$ and $1 \oplus 1 = 0$.

EXERCISE 4.1 Show that the U_{CNOT} cannot be written as a tensor product of two one-qubit gates.

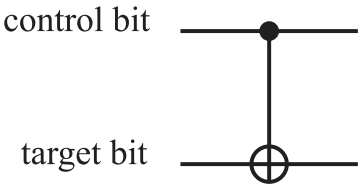
EXERCISE 4.2 Let $(a|0\rangle + b|1\rangle) \otimes |0\rangle$ be an input state to a CNOT gate. What is the output state?

It is convenient to introduce graphical representations of quantum gates. A one-qubit gate whose unitary matrix representation is U is depicted as



The left horizontal line is the input qubit state, while the right horizontal line is the output qubit state. Therefore the time flows from the left to the right.

A CNOT gate is expressed as

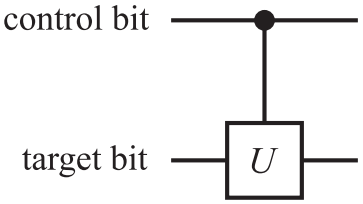


where $\bullet$ denotes the control bit, while $\bigoplus$ denotes the conditional negation. There may be many control bits (see CCNOT gate below).

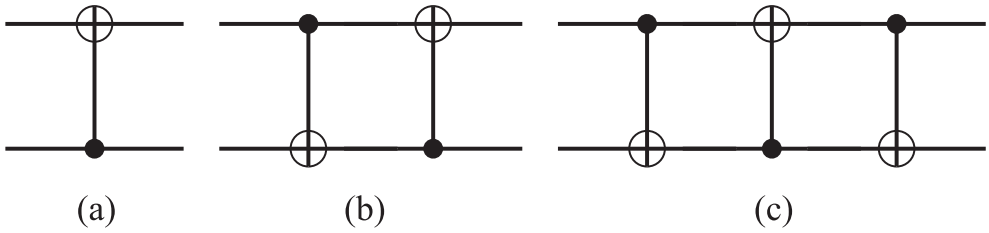
More generally, we consider a controlled-U gate,

$$V = |0\rangle\langle 0| \otimes I + |1\rangle\langle 1| \otimes U, \tag{4.7}$$

in which the target bit is acted on by a unitary transformation U only when the control bit is $|1\rangle$. This gate is denoted graphically as



EXERCISE 4.3 (1) Find the matrix representation of the "upside down" CNOT gate (a) in the basis $\{|00\rangle, |01\rangle, |10\rangle, |11\rangle\}$.

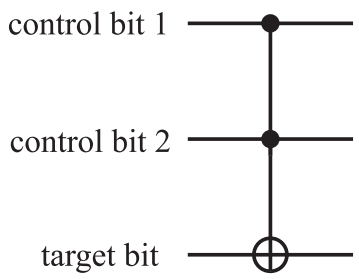


(2) Find the matrix representation of the circuit (b).
(3) Find the matrix representation of the circuit (c). Find the action of the circuit on a tensor product state $|\psi_1\rangle \otimes |\psi_2\rangle$.

The **CCNOT** (**Controlled-Controlled-NOT**) gate has three inputs, and the third qubit flips when and only when the first two qubits are both in the state $|1\rangle$. The explicit form of the CCNOT gate is

$$U_{\text{CCNOT}} = (|00\rangle\langle 00| + |01\rangle\langle 01| + |10\rangle\langle 10|) \otimes I + |11\rangle\langle 11| \otimes X. \tag{4.8}$$

This gate is graphically expressed as



The CCNOT gate is also known as the **Toffoli gate**.

4.2.2 Walsh-Hadamard Transformation

The **Hadamard gate** or the **Hadamard transformation** H is an important unitary transformation defined by

$$\begin{aligned} U_{\text{H}} : |0\rangle &\rightarrow \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \\ |1\rangle &\rightarrow \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle). \end{aligned} \tag{4.9}$$

It is used to generate a superposition state from $|0\rangle$ or $|1\rangle$. The matrix representation of H is

$$U_{\text{H}} = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)\langle 0| + \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)\langle 1| = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}. \tag{4.10}$$

A Hadamard gate is depicted as

There are numerous important applications of the Hadamard transformation. All possible 2^n states are generated, when U_{H} is applied on each qubit

of the state $|00\ldots0\rangle$:

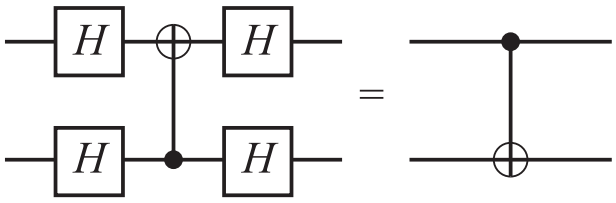
$$\begin{aligned}
&(H \otimes H \otimes \ldots \otimes H)|00\ldots0\rangle \\
&= \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \otimes \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \otimes \ldots \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \\
&= \frac{1}{\sqrt{2^n}} \sum_{x=0}^{2^n-1} |x\rangle. \qquad (4.11)
\end{aligned}$$

Therefore, we produce a superposition of all the states $|x\rangle$ with $0 \leq x \leq 2^n - 1$ simultaneously. This action of H on an n-qubit system is called the **Walsh transformation**, or **Walsh-Hadamard transformation**, and denoted as W_n. Note that

$$W_1 = U_{\mathrm{H}}, \quad W_{n+1} = U_{\mathrm{H}} \otimes W_n. \qquad (4.12)$$

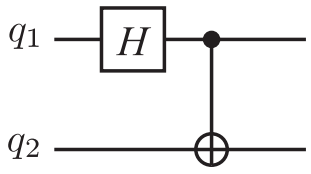
EXERCISE 4.4 Show that W_n is unitary.

EXERCISE 4.5 Show that the two circuits below are equivalent:



This exercise shows that the control bit and the target bit in a CNOT gate are interchangeable by introducing four Hadamard gates.

EXERCISE 4.6 Let us consider the following quantum circuit



(4.13)

where q_1 denotes the first qubit, while q_2 denotes the second. What are the outputs for the inputs $|00\rangle, |01\rangle, |10\rangle$ and $|11\rangle$?

4.2.3 SWAP Gate and Fredkin Gate

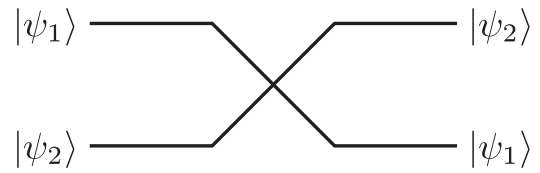
The SWAP gate acts on a tensor product state as

$$U_{\mathrm{SWAP}}|\psi_1, \psi_2\rangle = |\psi_2, \psi_1\rangle. \qquad (4.14)$$

The explict form of $U_{\rm SWAP}$ is given by

$$\begin{aligned} U_{\rm SWAP} &= |00\rangle\langle 00| + |01\rangle\langle 10| + |10\rangle\langle 01| + |11\rangle\langle 11| \\ &= \begin{pmatrix} 1\,0\,0\,0 \\ 0\,0\,1\,0 \\ 0\,1\,0\,0 \\ 0\,0\,0\,1 \end{pmatrix}. \end{aligned} \tag{4.15}$$

Needless to say, it works as a linear operator on a superposition of states. The SWAP gate is expressed as



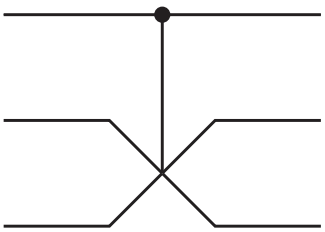
Note that the SWAP gate is a special gate which maps an arbitrary tensor product state to a tensor product state. In contrast, most two-qubit gates map a tensor product state to an entangled state.

EXERCISE 4.7 Show that the above $U_{\rm SWAP}$ is written as

$$\begin{aligned} U_{\rm SWAP} = (|0\rangle\langle 0| \otimes I + |1\rangle\langle 1| \otimes X)(I \otimes |0\rangle\langle 0| + X \otimes |1\rangle\langle 1|) \\ (|0\rangle\langle 0| \otimes I + |1\rangle\langle 1| \otimes X). \end{aligned} \tag{4.16}$$

This shows that the SWAP gate is implemented with three CNOT gates as given in Exercise 4.3 (3).

The controlled-SWAP gate



is also called the **Fredkin gate**. It flips the second (middle) and the third (bottom) qubits when and only when the first (top) qubit is in the state $|1\rangle$. Its explicit form is

$$U_{\rm Fredkin} = |0\rangle\langle 0| \otimes I_4 + |1\rangle\langle 1| \otimes U_{\rm SWAP}. \tag{4.17}$$

4.3 Correspondence with Classical Logic Gates

Before we proceed further, it is instructive to show that all the elementary logic gates, NOT, AND, XOR, OR and NAND, in classical logic circuits can

be implemented with quantum gates. In this sense, quantum information processing contains the classical one.

4.3.1 NOT Gate

Let us consider the **NOT gate** first. It is defined by the following logic function,

$$\mathrm{NOT}(x) = \neg x = \begin{cases} 0 & x = 1 \\ 1 & x = 0 \end{cases} \tag{4.18}$$

where $\neg x$ stands for the **negation** of x. Under the correspondence $0 \leftrightarrow |0\rangle,\ 1 \leftrightarrow |1\rangle$, we have already seen in Eq. (4.2) that the gate X negates the basis vectors as

$$X|x\rangle = |\neg x\rangle = |\mathrm{NOT}(x)\rangle,\ (x = 0, 1). \tag{4.19}$$

Now let us measure the output state. We employ the following measurement operator:

$$M_1 = |1\rangle\langle 1|. \tag{4.20}$$

M_1 has eigenvalues 0 and 1 with the eigenvectors $|0\rangle$ and $|1\rangle$, respectively. When the input is $|0\rangle$, the output is $|1\rangle$ and the measurement gives the value 1 with the probability 1. If, on the other hand, the input is $|1\rangle$, the output is $|0\rangle$ and the measurement yields 1 with probability 0, or in other words, it yields 0 with probability 1. It should be kept in mind that the operator X acts on an arbitrary linear combination $|\psi\rangle = a|0\rangle + b|1\rangle$, which is classically impossible. The output state is then $X|\psi\rangle = a|1\rangle + b|0\rangle$.

We show in the following that the CCNOT gate implements all classical logic gates. The first and the second input qubits are set to $|1\rangle$ to obtain the NOT gate as

$$U_{\mathrm{CCNOT}}|1, 1, x\rangle = |1, 1, \neg x\rangle. \tag{4.21}$$

4.3.2 XOR Gate

Since a quantum gate has to be reversible, we cannot construct a unitary gate corresponding to the classcial **XOR gate** whose function is $x, y \mapsto x \oplus y$ ($x, y \in \{0, 1\}$), where $x \oplus y$ is an addition mod 2; $0 \oplus 0 = 0,\ 0 \oplus 1 = 1 \oplus 0 = 1,\ 1 \oplus 1 = 0$. Clearly this operation has no inverse. This operation may be made reversible if we keep the first bit x during the gate operation, namely, if we define

$$f(x, y) = (x, x \oplus y), \quad x, y \in \{0, 1\}. \tag{4.22}$$

We call this function f, also the XOR gate. The quantum gate that does this operation is nothing but the CNOT gate defined by Eq. (4.5),

$$U_{\mathrm{XOR}} = U_{\mathrm{CNOT}} = |0\rangle\langle 0| \otimes I + |1\rangle\langle 1| \otimes X. \tag{4.23}$$

Note that the XOR gate may be also obtained from the CCNOT gate. Suppose the first qubit of the CCNOT gate is fixed to $|1\rangle$. Then it is easy to verify that

$$U_{\text{CCNOT}}|1, x, y\rangle = |1, x, x \oplus y\rangle. \tag{4.24}$$

Thus the CCNOT gate can be used to construct the XOR gate.

4.3.3 AND Gate

The logical **AND gate** is defined by

$$\text{AND}(x, y) \equiv x \wedge y \equiv \begin{cases} 1 & x = y = 1 \\ 0 & \text{otherwise} \end{cases} \quad x, y \in \{0, 1\}. \tag{4.25}$$

Clearly this operation is not reversible and we have to introduce the same sort of prescription which we employed in the XOR gate.

Let us define the logic function

$$f(x, y, 0) \equiv (x, y, x \wedge y), \tag{4.26}$$

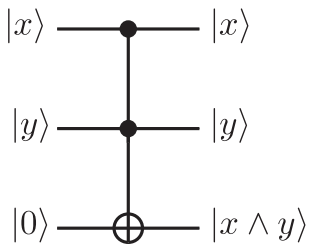
which we also call AND. Note that we have to keep both x and y for f to be reversible since $x = x \wedge y = 0$ implies *both* $x = y = 0$ and $x = 0, y = 1$. The unitary matrix that computes f is

$$\begin{aligned} U_{\text{AND}} = &(|00\rangle\langle 00| + |01\rangle\langle 01| + |10\rangle\langle 10|) \otimes I \\ &+ |11\rangle\langle 11| \otimes X. \end{aligned} \tag{4.27}$$

It is readily verified that

$$U_{\text{AND}}|x, y, 0\rangle = |x, y, x \wedge y\rangle, \quad x, y \in \{0, 1\}. \tag{4.28}$$

Observe that the third qubit in the RHS is 1 if and only if $x = y = 1$ and 0 otherwise. Thus the CCNOT gate may be employed to implement the AND gate. It follows from Eq. (4.28) that the AND gate is denoted graphically as



4.3.4 OR Gate

The **OR gate** represents the logical function

$$\text{OR}(x, y) = x \vee y = \begin{cases} 0 & x = y = 0 \\ 1 & \text{otherwise} \end{cases} \quad x, y \in \{0, 1\}. \tag{4.29}$$

This function OR is not reversible either and special care must be taken.

Let us define

$$f(x, y, 0) \equiv (\neg x, \neg y, x \vee y), \quad x, y \in \{0, 1\}, \tag{4.30}$$

which we also call OR. Although the first and the second bits are negated, it is not essential in the construction of the OR gate. These negations appear due to our construction of the OR gate based on the de Morgan theorem

$$x \vee y = \neg(\neg x \wedge \neg y). \tag{4.31}$$

They may be removed by adding extra NOT gates if necessary.

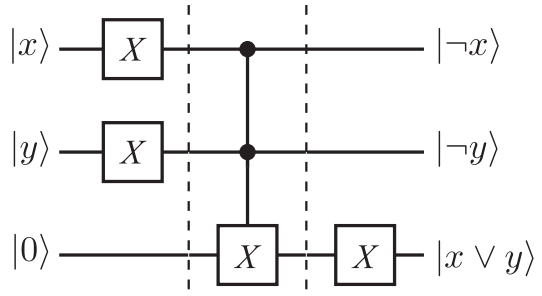
Let $|x, y, 0\rangle$ be the input state. The unitary matrix that represents f is

$$U_{\rm OR} = |00\rangle\langle 11| \otimes X + |01\rangle\langle 10| \otimes X + |10\rangle\langle 01| \otimes X + |11\rangle\langle 00| \otimes I. \tag{4.32}$$

EXERCISE 4.8 Verify that the above matrix $U_{\rm OR}$ indeed satisfies

$$U_{\rm OR}|x, y, 0\rangle = |\neg x, \neg y, x \vee y\rangle, \quad x, y \in \{0, 1\}. \tag{4.33}$$

Now it is obvious why negations in the first and the second qubits appear in the OR gate. Since we have already constructed the NOT gate and AND gate, we take advantage of this in the construction of the OR gate. The equality (4.31) leads us to the following diagram:



Accordingly, the first and the second qubits are negated. The unitary matrix obtained from this diagram is

$$\begin{aligned} U_{\rm OR} = {} & (I \otimes I \otimes X) \\ & \cdot(|00\rangle\langle 00| \otimes I + |01\rangle\langle 01| \otimes I + |10\rangle\langle 10| \otimes I + |11\rangle\langle 11| \otimes X) \\ & \cdot(X \otimes X \otimes I). \end{aligned} \tag{4.34}$$

The matrix products are readily evaluated to yield

$$\begin{aligned} U_{\rm OR} &= (|00\rangle\langle 00| \otimes X + |01\rangle\langle 01| \otimes X + |10\rangle\langle 10| \otimes X + |11\rangle\langle 11| \otimes I) \\ &\quad \cdot(X \otimes X \otimes I) \\ &= |00\rangle\langle 11| \otimes X + |01\rangle\langle 10| \otimes X + |10\rangle\langle 01| \otimes X + |11\rangle\langle 00| \otimes I, \end{aligned}$$

which verifies Eq. (4.32).

Observe that the OR gate is implemented with the X and the CCNOT gates and, moreover, the X gate is obtained from the CCNOT gate by putting the first and the second bits to $|1\rangle$.

If we want to have a gate $V_{\rm OR}|x, y, 0\rangle = |x, y, x \vee y\rangle$, we may multiply $X \otimes X \otimes I$ to $U_{\rm OR}$ from the left so that $V_{\rm OR} = (X \otimes X \otimes I)U_{\rm OR}$.

EXERCISE 4.9 Show that the NAND gate can be obtained from the CCNOT gate. Here NAND is defined by the function

$$\mathrm{NAND}(x, y) = \neg(x \wedge y) = \begin{cases} 0 & x = y = 1 \\ 1 & \text{otherwise} \end{cases} \qquad x, y \in \{0, 1\}. \tag{4.35}$$

In summary, we have shown that all the classical logic gates, NOT, AND, OR, XOR and NAND gates, may be obtained from the CCNOT gate. Thus all the classical computation may be carried out with a quantum computor. Note, however, that these gates belong to a tiny subset of the set of unitary matrices.

In the next section, we show that copying unknown information is impossible in quantum computing. However, it is also shown that this does not restrict the superiority of quantum computing over the classical counterpart.

4.4 No-Cloning Theorem

We copy classical data almost every day. In fact, this is amongst the most common functions with digital media. (Of course we should not copy media that are copyright protected.) This cannot be done in quantum information theory! We cannot clone an unknown quantum state with unitary operations.

THEOREM 4.1 (Wootters and Zurek [4], Dieks [5]) An unknown quantum system cannot be cloned by unitary transformations.

Proof. Suppose there would exist a unitary transformation U that makes a clone of a quantum system. Namely, suppose U acts, for any state $|\varphi\rangle$, as

$$U : |\varphi 0\rangle \to |\varphi\varphi\rangle. \tag{4.36}$$

Let $|\varphi\rangle$ and $|\phi\rangle$ be two states that are linearly independent. Then we should have $U|\varphi 0\rangle = |\varphi\varphi\rangle$ and $U|\phi 0\rangle = |\phi\phi\rangle$ by definition. Then the action of U on $|\psi\rangle = \dfrac{1}{\sqrt{2}}(|\varphi\rangle + |\phi\rangle)$ yields

$$U|\psi 0\rangle = \frac{1}{\sqrt{2}}(U|\varphi 0\rangle + U|\phi 0\rangle) = \frac{1}{\sqrt{2}}(|\varphi\varphi\rangle + |\phi\phi\rangle).$$

If U were a cloning transformation, we must also have

$$U|\psi 0\rangle = |\psi\psi\rangle = \frac{1}{2}(|\varphi\varphi\rangle + |\varphi\phi\rangle + |\phi\varphi\rangle + |\phi\phi\rangle),$$

which contradicts the previous result. Therefore, there does not exist a unitary cloning transformation. ∎

Clearly, there is no way to clone a state by measurements. A measurement is probabilistic and non-unitary, and it gets rid of the component of the state which is in the orthogonal complement of the observed subspace.

EXERCISE 4.10 Suppose U is a cloning unitary transformation, such that

$$|\Psi\rangle \equiv U|\psi\rangle|0\rangle = |\psi\rangle|\psi\rangle$$
$$|\Phi\rangle \equiv U|\phi\rangle|0\rangle = |\phi\rangle|\phi\rangle$$

for arbitrary $|\psi\rangle$ and $|\phi\rangle$.
(1) Write down $\langle\Psi|\Phi\rangle$ in all possible ways.
(2) Show, by inspecting the result of (1), that such U does not exist.

It was mentioned in the end of the previous section that a quantum computer can simulate arbitrary classical logic circuits. Then how about copying data? It should be kept in mind that the no-cloning theorem states that we cannot copy an *arbitrary* state $|\psi\rangle = a|0\rangle + b|1\rangle$. The loophole is that the theorem does not apply if the states to be cloned are limited to $|0\rangle$ and $|1\rangle$. For these cases, the copying operator U should work as

$$U : |00\rangle \mapsto |00\rangle, \quad : |10\rangle \mapsto |11\rangle.$$

We can assign arbitrary action of U on a state whose second input is $|1\rangle$ since this case does not happen. What we have to keep in our mind is only that U be unitary. An example of such U is

$$U = (|00\rangle\langle 00| + |11\rangle\langle 10|) + (|01\rangle\langle 01| + |10\rangle\langle 11|), \tag{4.37}$$

where the first set of operators renders U the cloning operator and the second set is added just to make U unitary. We immediately notice that U is nothing but the CNOT gate introduced in §4.2.

Therefore, if the data under consideration are limited within $|0\rangle$ and $|1\rangle$, we can copy the qubit states even in a quantum computer. This fact is used to construct quantum error correcting codes.

4.5 Dense Coding and Quantum Teleportation

Now we are ready to introduce two simple applications of qubits and quantum gates: **dense coding** and **quantum teleportation**. The Bell state has

been delivered beforehand, and one of the qubits carries two classical bits of information in the dense coding system. In the quantum teleportation, on the other hand, two classical bits are used to transmit a single qubit. At first glance, the quantum teleportation may seem to be in contradiction with the no-cloning theorem. However, this is not the case since the original state is destroyed.

Entanglement is the keyword in both applications. The setting is common for both cases. Suppose Alice wants to send Bob information. Each of them has been sent each of the qubits of the Bell state

$$|\Phi^+\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle) \tag{4.38}$$

in advance. Suppose Alice has the first qubit and Bob has the second.

4.5.1 Dense Coding

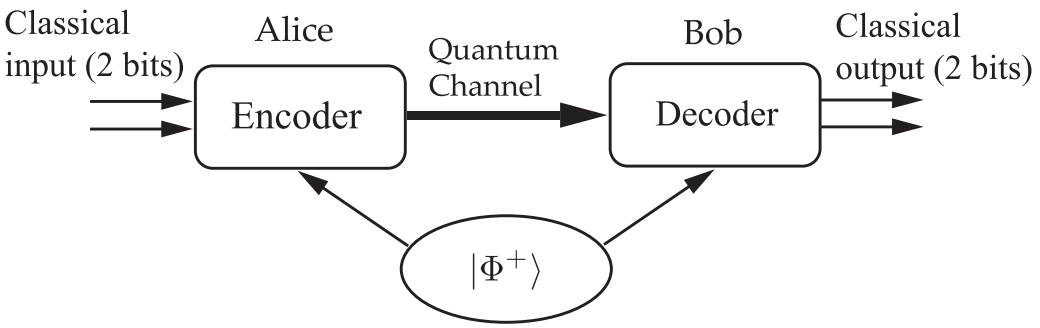


FIGURE 4.1
Communication from Alice to Bob using dense coding. Each qubit of the Bell state $|\Phi^+\rangle$ has been distributed to each of them beforehand. Then two bits of classical information can be transmitted by sending a single qubit through the quantum channel.

Alice: Alice wants to send Bob a binary number $x \in \{00, 01, 10, 11\}$. She picks up one of $\{I, X, Y, Z\}$ according to x she has chosen and applies the transformation on her qubit (the first qubit of the Bell state). Applying the transformation to only her qubit means she applies an identity transformation

to the second qubit which Bob keeps with him. This results in

$$\begin{array}{lll} x & \text{transformation } U & \text{state after transformation} \\ 0 = 00 & I \otimes I & |\psi_0\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle) \\ 1 = 01 & X \otimes I & |\psi_1\rangle = \frac{1}{\sqrt{2}}(|10\rangle + |01\rangle) \\ 2 = 10 & Y \otimes I & |\psi_2\rangle = \frac{1}{\sqrt{2}}(|10\rangle - |01\rangle) \\ 3 = 11 & Z \otimes I & |\psi_3\rangle = \frac{1}{\sqrt{2}}(|00\rangle - |11\rangle). \end{array} \tag{4.39}$$

Alice sends Bob her qubit after the transformation given above is applied. Note that the set of four states in the rightmost column is nothing but the four Bell basis vectors.

Bob: Bob applies CNOT to the entangled pair in which the first qubit, the received qubit, is the control bit, while the second one, which Bob keeps, is the target bit. This results in a tensor-product state:

$$\begin{array}{llll} \text{Received state} & \text{Output of CNOT} & \text{1st qubit} & \text{2nd qubit} \\ |\psi_0\rangle & \frac{1}{\sqrt{2}}(|00\rangle + |10\rangle) & \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) & |0\rangle \\ |\psi_1\rangle & \frac{1}{\sqrt{2}}(|11\rangle + |01\rangle) & \frac{1}{\sqrt{2}}(|1\rangle + |0\rangle) & |1\rangle \\ |\psi_2\rangle & \frac{1}{\sqrt{2}}(|11\rangle - |01\rangle) & \frac{1}{\sqrt{2}}(|1\rangle - |0\rangle) & |1\rangle \\ |\psi_3\rangle & \frac{1}{\sqrt{2}}(|00\rangle - |10\rangle) & \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) & |0\rangle \end{array} \tag{4.40}$$

Note that Bob can measure the first and second qubits independently since the output is a tensor-product state. The number x is either 00 or 11 if the measurement outcome of the second qubit is $|0\rangle$, while it is either 01 or 10 if the meansurement outcome is $|1\rangle$.

Finally, a Hadamard transformation H is applied on the first qubit. Bob obtains

$$\begin{array}{lll} \text{Received state} & \text{1st qubit} & U_{\mathrm{H}}|\text{1st qubit}\rangle \\ |\psi_0\rangle & \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) & |0\rangle \\ |\psi_1\rangle & \frac{1}{\sqrt{2}}(|1\rangle + |0\rangle) & |0\rangle \\ |\psi_2\rangle & \frac{1}{\sqrt{2}}(|1\rangle - |0\rangle) & -|1\rangle \\ |\psi_3\rangle & \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) & |1\rangle \end{array} \tag{4.41}$$

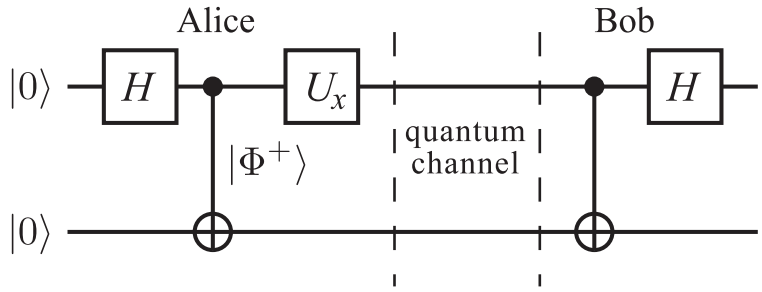


FIGURE 4.2
Quantum circuit implementation of the dense coding system. The leftmost Hadamard gate and the next CNOT gate generate the Bell state. Then a unitary gate U, depending on the bits Alice wants to send, is applied to the first qubit. Bob applies the rightmost CNOT gate and the Hadamard gate to decode Alice's message.

The number x is either 00 or 01 if the measurement of the first qubit results in $|0\rangle$, while it is either 10 or 11 if it is $|1\rangle$. Therefore, Bob can tell what x is in every case.

Quantum circuit implementation for the dense coding is given in Fig. 4.2

4.5.2 Quantum Teleportation

The purpose of **quantum teleportation** is to transmit an unknown quantum *state* of a qubit using two classical bits such that the recipient reproduces exactly the same state as the original qubit state. Note that the qubit itself is not transported but the information required to reproduce the quantum state is transmitted. The original state is destroyed such that quantum teleportation should not be in contradiction with the no-cloning theorem. Quantum teleportation has already been realized under laboratory conditions using photons [6, 7, 8, 9], coherent light field [10], NMR [11], and trapped ions [12, 13]. The teleportation scheme introduced in this section is due to [11]. Figure 4.3 shows the schematic diagram of quantum teleportation, which will be described in detail below.

Alice: Alice has a qubit, whose state she does not know. She wishes to send Bob the quantum state of this qubit through a classical communication channel. Let

$$|\phi\rangle = a|0\rangle + b|1\rangle \tag{4.42}$$

be the state of the qubit. Both of them have been given one of the qubits of the entangled pair

$$|\Phi^+\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$$

as in the case of the dense coding.

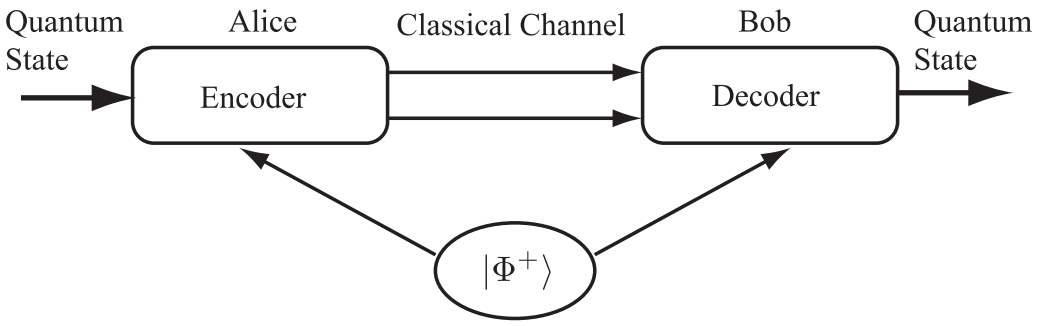


FIGURE 4.3
In quantum teleportation, Alice sends Bob two classical bits so that Bob reproduces a qubit state Alice used to have.

Alice applies the decoding step in the dense coding to the qubit $|\phi\rangle = a|0\rangle + b|1\rangle$ to be sent and her qubit of the entangled pair. They start with the state

$$\begin{aligned} |\phi\rangle \otimes |\Phi^+\rangle &= \frac{1}{\sqrt{2}}\left[a|0\rangle \otimes (|00\rangle + |11\rangle) + b|1\rangle \otimes (|00\rangle + |11\rangle)\right] \\ &= \frac{1}{\sqrt{2}}\left(a|000\rangle + a|011\rangle + b|100\rangle + b|111\rangle\right), \end{aligned} \tag{4.43}$$

where Alice has the first two qubits while Bob has the third. Alice applies $U_{\text{CNOT}} \otimes I$ followed by $U_{\text{H}} \otimes I \otimes I$ to this state, which results in

$$\begin{aligned} &(U_{\text{H}} \otimes I \otimes I)(U_{\text{CNOT}} \otimes I)(|\phi\rangle \otimes |\Phi^+\rangle) \\ &= (U_{\text{H}} \otimes I \otimes I)(U_{\text{CNOT}} \otimes I)\frac{1}{\sqrt{2}}\left(a|000\rangle + a|011\rangle + b|100\rangle + b|111\rangle\right) \\ &= \frac{1}{2}\left[a(|000\rangle + |011\rangle + |100\rangle + |111\rangle) + b(|010\rangle + |001\rangle - |110\rangle - |101\rangle)\right] \\ &= \frac{1}{2}[|00\rangle(a|0\rangle + b|1\rangle) + |01\rangle(a|1\rangle + b|0\rangle) \\ &\qquad + |10\rangle(a|0\rangle - b|1\rangle) + |11\rangle(a|1\rangle - b|0\rangle)]. \end{aligned} \tag{4.44}$$

If Alice measures the two qubits in her hand, she will obtain one of the states $|00\rangle, |01\rangle, |10\rangle$ or $|11\rangle$ with equal probability 1/4. Bob's qubit (a qubit from the Bell state initially) collapses to $a|0\rangle + b|1\rangle, a|1\rangle + b|0\rangle, a|0\rangle - b|1\rangle$ or $a|1\rangle - b|0\rangle$, respectively, depending on the result of Alice's measurement. Alice then sends Bob her result of the measurement using two classical bits.

Notice that Alice has totally destroyed the initial qubit $|\phi\rangle$ upon her measurement. This makes quantum teleportation consistent with the no-cloning theorem.

Bob: After receiving two classical bits, Bob knows the state of the qubit in

his hand;

Received bits	Bob's state	Decoding
00	$a\|0\rangle + b\|1\rangle$	I
01	$a\|1\rangle + b\|0\rangle$	X
10	$a\|0\rangle - b\|1\rangle$	Z
11	$a\|1\rangle - b\|0\rangle$	Y

(4.45)

Bob reconstructs the intial state $|\phi\rangle$ by applying the decoding process shown above. Suppose Alice sends Bob the classical bits 10, for example. Then Bob applies Z to his state to reconstruct $|\phi\rangle$ as follows:

$$Z: \ (a|0\rangle - b|1\rangle) \mapsto (a|0\rangle + b|1\rangle) = |\phi\rangle.$$

Figure 4.4 shows the actual quantum circuit for quantum teleportation.

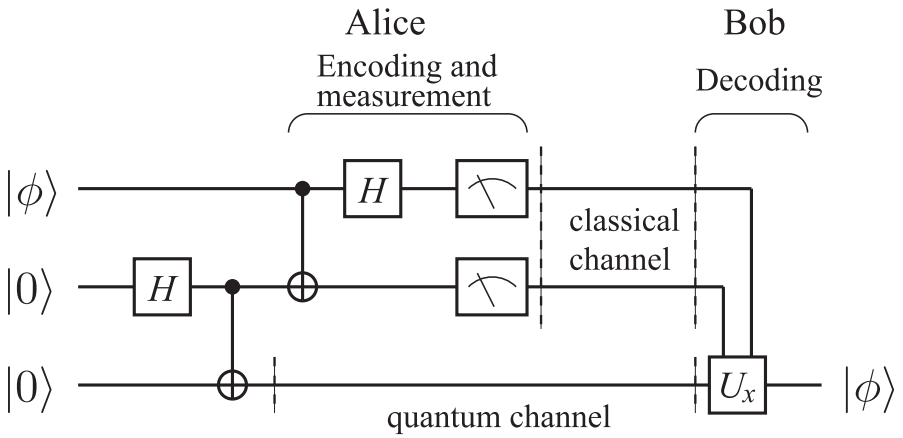


FIGURE 4.4
Quantum circuit implementation of quantum teleportation. Alice operates gates in the left side. The first Hadamard gate and the next CNOT gates generate the Bell state $|\Phi^+\rangle$ from $|00\rangle$. The bottom qubit is sent to Bob through a quantum channel while the first and the second qubits are measured after applying the second set of the CNOT gate and the Hadamard gate on them. The measurement outcome x is sent to Bob through a classical channel. Bob operates a unitary operation U_x, which depends on the received message x, on his qubit.

EXERCISE 4.11 Let $|\psi\rangle = a|00\rangle + b|11\rangle$ be a two-qubit state. Apply a Hadamard gate to the first qubit and then measure the first qubit. Find the second qubit state after the measurement corresponding to the outcome of the first qubit measurement.

4.6 Universal Quantum Gates

It can be shown that any classical logic gate can be constructed by using a small set of gates, AND, NOT and XOR, for example. Such a set of gates is called the *universal* set of classical gates. Since the CCNOT gate can simulate these classical gates, quantum circuits simulate any classical circuits. It should be noted that the set of quantum gates is much larger than those classical gates which can be simulated by quantum gates. Thus we want to find a universal set of *quantum* gates from which any quantum circuits, i.e., any unitary matrix, can be constructed.

In the following, it will be shown that
(1) the set of single qubit gates and
(2) CNOT gate
form a universal set of quantum circuits (**universality theorem**).

We will prove the following Lemma before stating the main theorem. Let us start with a definition. A **two-level unitary matrix** is a unitary matrix which acts non-trivially only on two vector components. Suppose V is a two-level unitary matrix. Then V has the same matrix elements as those of the unit matrix except for certain four elements V_{aa}, V_{ab}, V_{ba} and V_{bb}. An example of a two-level unitary matrix is

$$V = \begin{pmatrix} \alpha^* & 0 & 0 & \beta^* \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -\beta & 0 & 0 & \alpha \end{pmatrix}, \quad (|\alpha|^2 + |\beta|^2 = 1),$$

where $a = 1$ and $b = 4$.

LEMMA 4.1 Let U be a unitary matrix acting on $\mathbb{C}^d$. Then there are $N \leq d(d-1)/2$ two-level unitary matrices $U_1, U_2, \ldots, U_N$ such that

$$U = U_1 U_2 \ldots U_N. \tag{4.46}$$

Proof. The proof requires several steps. It is instructive to start with the case $d = 3$. Let

$$U = \begin{pmatrix} a & d & g \\ b & e & h \\ c & f & j \end{pmatrix}$$

be a unitary matrix. We want to find two-level unitary matrices U_1, U_2, U_3 such that

$$U_3 U_2 U_1 U = I.$$

Then it follows that

$$U = U_1^\dagger U_2^\dagger U_3^\dagger.$$

(Never mind the daggers! If U_k is two-level unitary, $U_k^\dagger$ is also two-level unitary.) We prove the above decomposition by constructing U_k explicitly.

(i) Let

$$U_1 = \begin{pmatrix} \frac{a^*}{u} & \frac{b^*}{u} & 0 \\ -\frac{b}{u} & \frac{a}{u} & 0 \\ 0 & 0 & 1 \end{pmatrix},$$

where $u = \sqrt{|a|^2 + |b|^2}$. Verify that U_1 is unitary. Then we obtain

$$U_1 U = \begin{pmatrix} a' & d' & g' \\ 0 & e' & h' \\ c' & f' & j' \end{pmatrix},$$

where $a', \ldots, j'$ are some complex numbers, whose details are not necessary. Observe that, with this choice of U_1, the first component of the second row vanishes.

(ii) Let

$$U_2 = \begin{pmatrix} \frac{a'^*}{u'} & 0 & \frac{c'^*}{u'} \\ 0 & 1 & 0 \\ -\frac{c'}{u'} & 0 & \frac{a'}{u'} \end{pmatrix} = \begin{pmatrix} a'^* & 0 & c'^* \\ 0 & 1 & 0 \\ -c' & 0 & a' \end{pmatrix},$$

where $u' = \sqrt{|a'|^2 + |c'|^2} = 1$. Then

$$U_2 U_1 U = \begin{pmatrix} 1 & d'' & g'' \\ 0 & e'' & h'' \\ 0 & f'' & j'' \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & e'' & h'' \\ 0 & f'' & j'' \end{pmatrix},$$

where the equality $d'' = g'' = 0$ follows from the fact that $U_2 U_1 U$ is unitary, and hence the first row must be normalized.

(iii) Finally let

$$U_3 = (U_2 U_1 U)^\dagger = \begin{pmatrix} 1 & 0 & 0 \\ 0 & e''^* & f''^* \\ 0 & h''^* & j''^* \end{pmatrix}.$$

Then, by definition, $U_3 U_2 U_1 U = I$ is obvious. This completes the proof for $d = 3$.

Suppose U is a unitary matrix acting on $\mathbb{C}^d$ with a general dimension d. Then by repeating the above arguments, we find two-level unitary matrices $U_1, U_2, \ldots, U_{d-1}$ such that

$$U_{d-1} \ldots U_2 U_1 U = \begin{pmatrix} 1 & 0 & 0 & \ldots & 0 \\ 0 & * & * & \ldots & * \\ 0 & * & * & \ldots & * \\ & & \ldots\ldots & & \\ 0 & * & * & \ldots & * \end{pmatrix},$$

namely the $(1,1)$ component is unity and other components of the first row and the first column vanish. The number of matrices $\{U_k\}$ to achieve this form is the same as the number of zeros in the first column, hence $(d-1)$.

We then repeat the same procedure to the $(d-1)\times(d-1)$ block unitary matrix using $(d-2)$ two-level unitary matrices. After repeating this, we finally decompose U into a product of two-level unitary matrices

$$U = V_1 V_2 \dots V_N,$$

where $N \leq (d-1)+(d-2)+\ldots+1 = d(d-1)/2$. ∎

EXERCISE 4.12 Let U be a general 4×4 unitary matrix. Find two-level unitary matrices U_1, U_2 and U_3 such that

$$U_3 U_2 U_1 U = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & * & * & * \\ 0 & * & * & * \\ 0 & * & * & * \end{pmatrix}.$$

EXERCISE 4.13 Let

$$U = \frac{1}{2}\begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & i & -1 & -i \\ 1 & -1 & 1 & -1 \\ 1 & -i & -1 & i \end{pmatrix}. \tag{4.47}$$

Decompose U into a product of two-level unitary matrices.

Let us consider a unitary matrix acting on an n-qubit system. Then this unitary matrix is decomposed into a product of at most $2^n(2^n-1)/2 = 2^{n-1}(2^n-1)$ two-level unitary matrices. Now we are in a position to state the main theorem.

THEOREM 4.2 (Barenco *et al.*)[14] The set of single qubit gates and CNOT gate are universal. Namely, any unitary gate acting on an n-qubit register can be implemented with single qubit gates and CNOT gates.

Proof. We closely follow [1] for the proof here. Thanks to the previous Lemma, it suffices to prove the theorem for a two-level unitary matrix. Let U be a two-level unitary matrix acting nontrivially only on $|s\rangle$ and $|t\rangle$ basis vectors of an n-qubit system, where $s = s_{n-1}2^{n-1}+\ldots+s_1 2+s_0$ and $t = t_{n-1}2^{n-1}+\ldots+t_1 2+t_0$ are binary expressions for decimal numbers s and t. This means that matrix elements U_{ss}, U_{st}, U_{ts} and U_{tt} are different from those of the unit matrix, while all the others are the same, where $|s\rangle$ stands for $|s_{n-1}\rangle|s_{n-2}\rangle\dots|s_0\rangle$, for example. We can construct $\tilde{U}$, the non-trivial 2×2 unitary submatrix of U. $\tilde{U}$ may be thought of as a unitary matrix acting on a single qubit, whose basis is $\{|s\rangle, |t\rangle\}$.

STEP 1: U is reduced to $\tilde{U} \in \mathrm{U}(2)$.

The basis vectors $|s\rangle$ and $|t\rangle$ may be put together to form a basis for a single qubit using the following trick. This is done by introducing **Gray codes**. For two binary numbers $s = s_{n-1} \dots s_1 s_0$ and $t = t_{n-1} \dots t_1 t_0$, a Gray code connecting s and t is a sequence of binary numbers $\{g_1, \dots, g_m\}$ where the adjacent numbers, g_k and g_{k+1}, differ in exactly one bit. Moreover, the sequence satisfies the boundary conditions $g_1 = s$ and $g_m = t$.

Suppose $s = 100101$ and $t = 110110$, for example. An example of a Gray code connecting s and t is

$$\begin{aligned} s = g_1 &= 100101 \\ g_2 &= 1\hat{1}0101 \\ g_3 &= 1101\hat{1}1 \\ g_4 &= 11011\hat{0} = t, \end{aligned}$$

where the digit with $\hat{}$ has been renewed. It is clear from this construction that if s and t differ in p bits, the shortest Gray code is made of $p+1$ elements. It should be also clear that if s and t are of n digits, then $m \leq (n+1)$ since s and t differ at most in n bits.

With these preparations, we consider the implementation of U. The strategy is to find gates providing the sequence of state changes

$$|s\rangle = |g_1\rangle \to |g_2\rangle \to \dots \to |g_{m-1}\rangle. \tag{4.48}$$

Then g_{m-1} and g_m differ only in one bit, which is identified with the single qubit on which $\tilde{U}$ acts. In the example above, we have $|g_3\rangle = |11011\rangle \otimes |1\rangle$ and $|t\rangle = |g_4\rangle = |11011\rangle \otimes |0\rangle$. Now the operator $\tilde{U}$ may be introduced so that it acts on a two-dimensional subspace of the total Hilbert space, in which the first five qubits are in the state $|11011\rangle$. Then we undo the sequence (4.48) so that $|g_{m-1}\rangle \to |g_{m-2}\rangle \to \dots \to |g_1\rangle = |s\rangle$. Each of these steps can be easily implemented using simple gates that have been introduced previously (see below).

Let us consider the following example of a three-qubit system, whose basis is $\{|000\rangle, |001\rangle, \dots, |111\rangle\}$. Let

$$U = \begin{pmatrix} a & 0 & 0 & 0 & 0 & 0 & 0 & c \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ b & 0 & 0 & 0 & 0 & 0 & 0 & d \end{pmatrix}, \quad (a, b, c, d \in \mathbb{C}) \tag{4.49}$$

be a two-level unitary matrix which we wish to implement. Note that U acts non-trivially only in the subspace spanned by $|000\rangle$ and $|111\rangle$. The unitarity

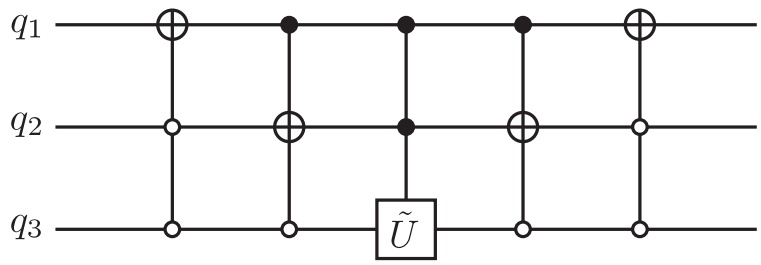


FIGURE 4.5
Example of circuit implementing the gate U.

of U ensures that the matrix

$$\tilde{U} = \begin{pmatrix} a & c \\ b & d \end{pmatrix} \tag{4.50}$$

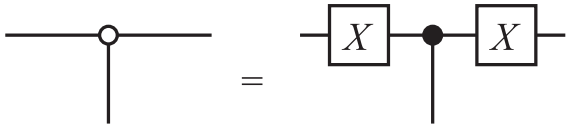
is also unitary. An example of a Gray code connecting 000 and 111 is

$$\begin{array}{rccc} & q_1 & q_2 & q_3 \\ g_1 = & 0 & 0 & 0 \\ g_2 = & 1 & 0 & 0 \\ g_3 = & 1 & 1 & 0 \\ g_4 = & 1 & 1 & 1 \end{array} \tag{4.51}$$

Since g_3 and g_4 differ only in the third qubit, which we call q_3, we have to bring g_1 to g_3 and then operate $\tilde{U}$ on the qubit q_3 provided that the first and the second qubits are in the state $|11\rangle$. (Namely we have a controlled-$\tilde{U}$ gate with the target bit q_3 and the control bits q_1 and q_2.) After this controlled operation is done, we have to put $|g_3\rangle = |110\rangle$ back to the state $|000\rangle$ as

$$|110\rangle \rightarrow |100\rangle \rightarrow |000\rangle.$$

This operation is graphically shown in Fig. 4.5. Here ○ denotes the negated control node. This means that the unitary gate acts on the target bit only when the control bit is in the state $|0\rangle$. This is easily implemented by adding two X gates as



It is easy to see that this gate indeed implements U. Suppose the input is $|101\rangle$, for example. Figure 4.6 shows that the gate has no effect on this basis vector since U should act as a unit matrix on this vector. The operation of U

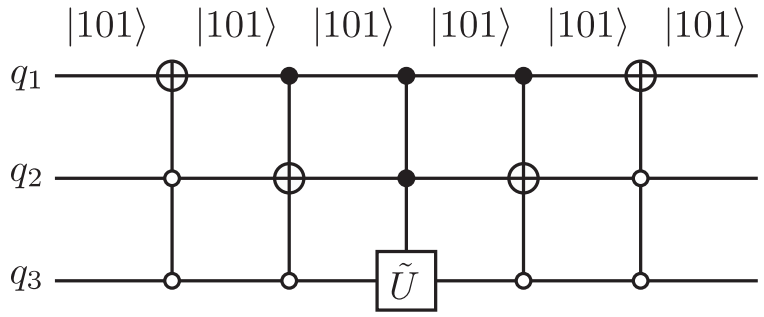


FIGURE 4.6
U-gate has no effect on the vector $|101\rangle$.

on the input $\alpha|000\rangle + \beta|111\rangle$ is

$$U(\alpha|000\rangle + \beta|111\rangle) = \begin{pmatrix} a & 0 & 0 & 0 & 0 & 0 & 0 & c \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ b & 0 & 0 & 0 & 0 & 0 & 0 & d \end{pmatrix} \begin{pmatrix} \alpha \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ \beta \end{pmatrix} = \begin{pmatrix} \alpha a + \beta c \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ \alpha b + \beta d \end{pmatrix}. \quad (4.52)$$

If we use the circuit shown in Fig. 4.5, we produce the same result as shown in Fig. 4.7

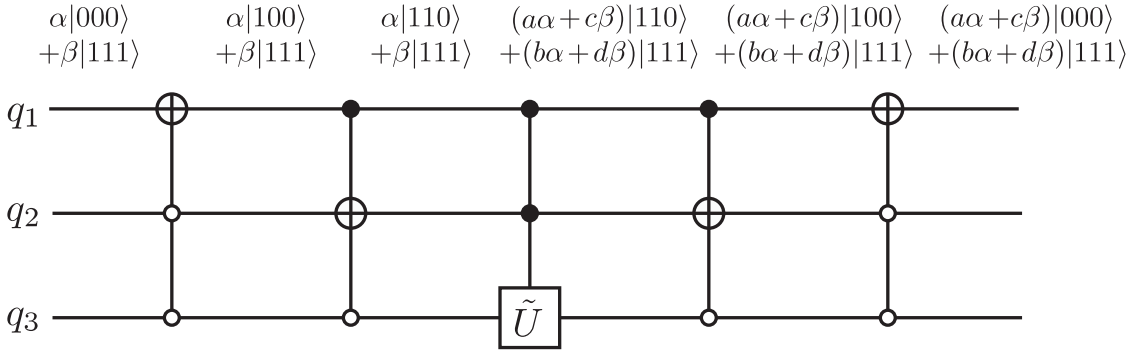


FIGURE 4.7
U-gate acting on $\alpha|000\rangle + \beta|111\rangle$ yields the desired output $(a\alpha + c\beta)|000\rangle + (b\alpha + d\beta)|111\rangle$.

This construction is easily generalized to any two-level unitary matrix $U \in \mathrm{U}(2^n)$. It will be shown below that all the gates in the above circuit can

be implemented with single-qubit gates and CNOT gates, which proves the universality of these gates.

EXERCISE 4.14 (1) Find the shortest Gray code which connects 000 with 110.
(2) Use this result to find a quantum circuit, such as Fig. 4.5, implementing a two-level unitary gate

$$U = \begin{pmatrix} a & 0 & 0 & 0 & 0 & 0 & c & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ b & 0 & 0 & 0 & 0 & 0 & d & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}, \quad \tilde{U} \equiv \begin{pmatrix} a & c \\ b & d \end{pmatrix} \in U(2).$$

You may use various controlled-NOT gates and controlled-$\tilde{U}$ gates.

STEP 2: Two-level unitary gates are decomposed into single-qubit gates and CNOT gates.

A controlled-U gate can be constructed from at most four single-qubit gates and two CNOT gates for any single-qubit unitary $U \in \mathrm{U}(2)$. Let us prove several Lemmas before we prove this statement.

LEMMA 4.2 Let $U \in \mathrm{SU}(2)$. Then there exist $\alpha, \beta, \gamma \in \mathbb{R}$ such that $U = R_z(\alpha)R_y(\beta)R_z(\gamma)$, where

$$R_z(\alpha) = \exp(i\alpha\sigma_z/2) = \begin{pmatrix} e^{i\alpha/2} & 0 \\ 0 & e^{-i\alpha/2} \end{pmatrix},$$
$$R_y(\beta) = \exp(i\beta\sigma_y/2) = \begin{pmatrix} \cos(\beta/2) & \sin(\beta/2) \\ -\sin(\beta/2) & \cos(\beta/2) \end{pmatrix}.$$

Proof. After some calculation, we obtain

$$R_z(\alpha)R_y(\beta)R_z(\gamma) = \begin{pmatrix} e^{i(\alpha+\gamma)/2}\cos(\beta/2) & e^{i(\alpha-\gamma)/2}\sin(\beta/2) \\ -e^{i(-\alpha+\gamma)/2}\sin(\beta/2) & e^{-i(\alpha+\gamma)/2}\cos(\beta/2) \end{pmatrix}. \quad (4.53)$$

Any $U \in \mathrm{SU}(2)$ may be written in the form

$$U = \begin{pmatrix} a & b \\ -b^* & a^* \end{pmatrix} = \begin{pmatrix} \cos\theta e^{i\lambda} & \sin\theta e^{i\mu} \\ -\sin\theta e^{-i\mu} & \cos\theta e^{-i\lambda} \end{pmatrix}, \quad (4.54)$$

where we used the fact that $\det U = |a|^2 + |b|^2 = 1$. Now we obtain $U = R_z(\alpha)R_y(\beta)R_z(\gamma)$ by making identifications

$$\theta = \frac{\beta}{2}, \lambda = \frac{\alpha+\gamma}{2}, \mu = \frac{\alpha-\gamma}{2}. \quad (4.55)$$

∎

LEMMA 4.3 Let $U \in \mathrm{SU}(2)$. Then there exist $A, B, C \in \mathrm{SU}(2)$ such that $U = AXBXC$ and $ABC = I$, where $X = \sigma_x$.

Proof. Lemma 4.2 states that $U = R_z(\alpha)R_y(\beta)R_z(\gamma)$ for some $\alpha, \beta, \gamma \in \mathbb{R}$. Let

$$A = R_z(\alpha)R_y\left(\frac{\beta}{2}\right), B = R_y\left(-\frac{\beta}{2}\right)R_z\left(-\frac{\alpha+\gamma}{2}\right), C = R_z\left(-\frac{\alpha-\gamma}{2}\right).$$

Then

$$\begin{aligned} AXBXC &= R_z(\alpha)R_y\left(\frac{\beta}{2}\right)XR_y\left(-\frac{\beta}{2}\right)R_z\left(-\frac{\alpha+\gamma}{2}\right)XR_z\left(-\frac{\alpha-\gamma}{2}\right) \\ &= R_z(\alpha)R_y\left(\frac{\beta}{2}\right)\left[XR_y\left(-\frac{\beta}{2}\right)X\right]\left[XR_z\left(-\frac{\alpha+\gamma}{2}\right)X\right]R_z\left(-\frac{\alpha-\gamma}{2}\right) \\ &= R_z(\alpha)R_y\left(\frac{\beta}{2}\right)R_y\left(\frac{\beta}{2}\right)R_z\left(\frac{\alpha+\gamma}{2}\right)R_z\left(-\frac{\alpha-\gamma}{2}\right) \\ &= R_z(\alpha)R_y(\beta)R_z(\gamma) = U, \end{aligned}$$

where use has been made of the identities $X^2 = I$ and $X\sigma_{y,z}X = -\sigma_{y,z}$.

It is also verified that

$$\begin{aligned} ABC &= R_z(\alpha)R_y\left(\frac{\beta}{2}\right)R_y\left(-\frac{\beta}{2}\right)R_z\left(-\frac{\alpha+\gamma}{2}\right)R_z\left(-\frac{\alpha-\gamma}{2}\right) \\ &= R_z(\alpha)R_y(0)R_z(-\alpha) = I. \end{aligned}$$

This proves the Lemma. ∎

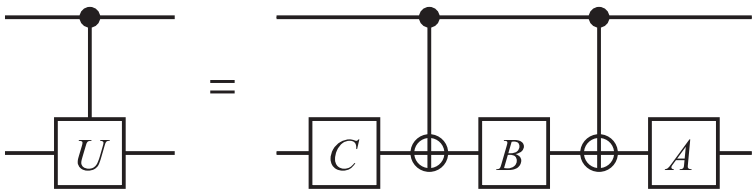


FIGURE 4.8
Controlled-U gate is made of at most three single-qubit gates and two CNOT gates for any $U \in \mathrm{SU}(2)$.

LEMMA 4.4 Let $U \in \mathrm{SU}(2)$ be factorized as $U = AXBXC$ as in the previous Lemma. Then the controlled-U gate can be implemented with at most three single-qubit gates and two CNOT gates (see Fig. 4.8).

Proof. The proof is almost obvious. When the control bit is 0, the target bit $|\psi\rangle$ is operated by C, B and A in this order so that

$$|\psi\rangle \mapsto ABC|\psi\rangle = |\psi\rangle,$$

while when the control bit is 1, we have

$$|\psi\rangle \mapsto AXBXC|\psi\rangle = U|\psi\rangle.$$

∎

So far, we have worked with $U \in \mathrm{SU}(2)$. To implement a general U-gate with $U \in \mathrm{U}(2)$, we have to deal with the phase. Let us first recall that any $U \in \mathrm{U}(2)$ is decomposed as $U = e^{i\alpha}V,\ V \in \mathrm{SU}(2), \alpha \in \mathbb{R}$.

LEMMA 4.5 Let

$$\Phi(\phi) = e^{i\phi}I = \begin{pmatrix} e^{i\phi} & 0 \\ 0 & e^{i\phi} \end{pmatrix}$$

and

$$D = R_z(-\phi)\Phi\left(\frac{\phi}{2}\right) = \begin{pmatrix} e^{-i\phi/2} & 0 \\ 0 & e^{i\phi/2} \end{pmatrix}\begin{pmatrix} e^{i\phi/2} & 0 \\ 0 & e^{i\phi/2} \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\phi} \end{pmatrix}.$$

Then the controlled-$\Phi(\phi)$ gate is expressed as a tensor product of single qubit gates as

$$U_{\mathrm{C}\Phi(\phi)} = D \otimes I. \tag{4.56}$$

.

Proof. The LHS is

$$\begin{aligned} U_{\mathrm{C}\Phi(\phi)} &= |0\rangle\langle 0| \otimes I + |1\rangle\langle 1| \otimes \Phi(\phi) = |0\rangle\langle 0| \otimes I + |1\rangle\langle 1| \otimes e^{i\phi}I \\ &= |0\rangle\langle 0| \otimes I + e^{i\phi}|1\rangle\langle 1| \otimes I, \end{aligned}$$

while the RHS is

$$\begin{aligned} D \otimes I &= \begin{pmatrix} 1 & 0 \\ 0 & e^{i\phi} \end{pmatrix} \otimes I \\ &= \left[|0\rangle\langle 0| + e^{i\phi}|1\rangle\langle 1|\right] \otimes I = U_{\mathrm{C}\Phi(\phi)}, \end{aligned}$$

which proves the lemma. ∎

Figure 4.9 shows the statement of the above lemma.

EXERCISE 4.15 Let us consider the controlled-V_1 gate $U_{\mathrm{C}V_1}$ and the controlled-V_2 gate $U_{\mathrm{C}V_2}$. Show that the controlled-V_1 gate followed by the controlled-V_2 gate is the controlled-V_2V_1 gate $U_{\mathrm{C}(V_2V_1)}$ as shown in Fig. 4.10.

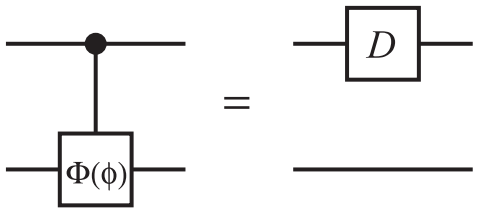


FIGURE 4.9
Equality $U_{C\Phi(\phi)} = D \otimes I$.

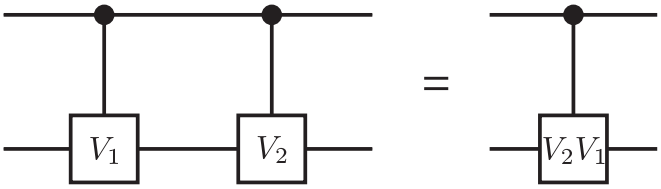


FIGURE 4.10
Equality $U_{CV_2}U_{CV_1} = U_{C(V_2V_1)}$.

Now we are ready to prove the main proposition.

PROPOSITION 4.1 Let $U \in \mathrm{U}(2)$. Then the controlled-U gate U_{CU} can be constructed by at most four single-qubit gates and two CNOT gates.

Proof. Let $U = \Phi(\phi)AXBXC$. According to the exercise above, the controlled-U gate is written as a product of the controlled-$\Phi(\phi)$ gate and the controlled-$AXBXC$ gate. Moreover, Lemma 4.5 states that the controlled-$\Phi(\phi)$ gate may be replaced by a single-qubit phase gate acting on the first qubit. The rest of the gate, the controlled-$AXBXC$ gate is implemented with three SU(2) gates and two CNOT gates as proved in Lemma 4.3. Therefore we have the following decomposition:

$$U_{CU} = (D \otimes A)U_{\mathrm{CNOT}}(I \otimes B)U_{\mathrm{CNOT}}(I \otimes C), \tag{4.57}$$

where

$$D = R_z(-\phi)\Phi(\phi/2)$$

and use has been made of the identity $(D \otimes I)(I \otimes A) = D \otimes A$. ∎

Figure 4.11 shows the statement of the proposition.

STEP 3: CCNOT gate and its variants are implemented with CNOT gates and their variants.

Now our final task is to prove that controlled-U gates with $n-1$ control bits are also constructed using single-qubit gates and CNOT gates. Let us start with the simplest case, in which $n = 3$.

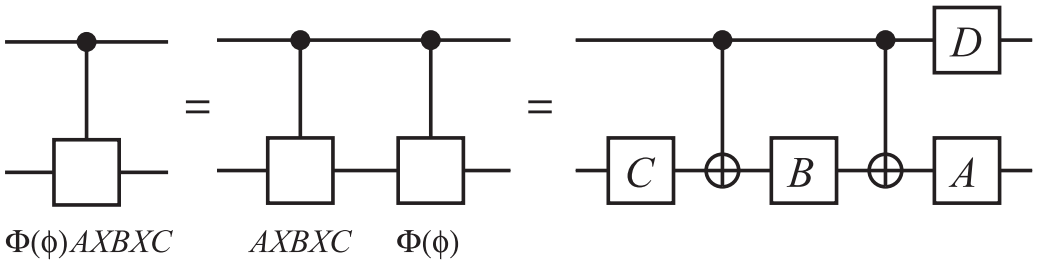


FIGURE 4.11
Controlled-U gate is implemented with at most four single-qubit gates and two CNOT gates.

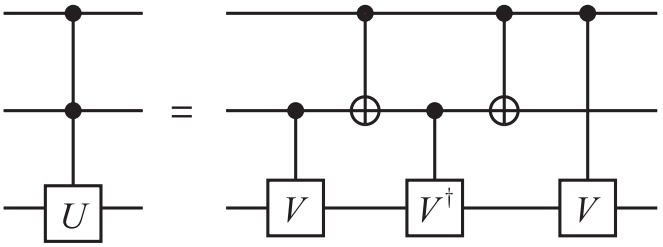


FIGURE 4.12
Controlled-controlled-U gate is equivalent to the gate made of controlled-V gates with $U = V^2$ and CNOT gates.

LEMMA 4.6 The two quantum circuits in Fig. 4.12 are equivalent, where $U = V^2$.

Proof. If both the first and the second qubits are 0 in the RHS, all the gates are ineffective and the third qubit is unchanged; the gate in this subspace acts as $|00\rangle\langle 00| \otimes I$. In case the first qubit is 0 and the second is 1, the third qubit is mapped as $|\psi\rangle \mapsto V^\dagger V|\psi\rangle = |\psi\rangle$; the gate is then $|01\rangle\langle 01| \otimes I$. When the first qubit is 1 and the second is 0, the third qubit is mapped as $|\psi\rangle \mapsto VV^\dagger|\psi\rangle = |\psi\rangle$; hence the gate in this subspace is $|10\rangle\langle 10| \otimes I$. Finally let both the first and the second qubits be 1. Then the action of the gate on the third qubit is $|\psi\rangle \mapsto VV|\psi\rangle = U|\psi\rangle$; namely the gate in this subspace is $|11\rangle\langle 11| \otimes U$. Thus it has been proved that the RHS of Fig. 4.12 is

$$(|00\rangle\langle 00| + |01\rangle\langle 01| + |10\rangle\langle 10|) \otimes I + |11\rangle\langle 11| \otimes U, \tag{4.58}$$

namely the controlled-controlled-U gate. ∎

This decomposition is explained intuitively as follows. The first V operates on the third qubit $|\psi\rangle$ if and only if the second qubit is 1. $V^\dagger$ is in action if and only if $x_1 \oplus x_2 = 1$, where x_k is the input bit of the kth qubit. The second V operation is applied if and only if the first qubit is 1. Thus the action of this gate on the third qubit is $V^2 = U$ only when $x_1 \wedge x_2 = 1$ and

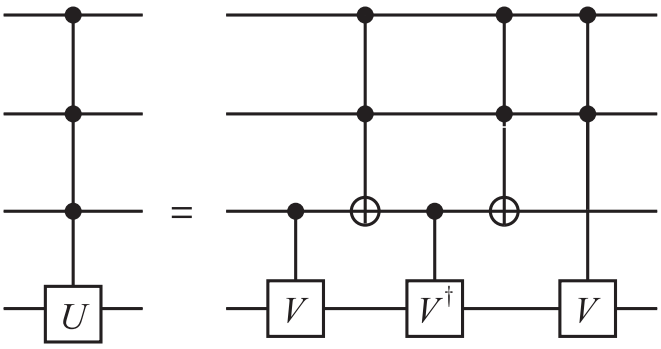


FIGURE 4.13
Decomposition of the C^3U gate.

I otherwise. This intuitive picture is of help when we implement the U gate with more control qubits.

EXERCISE 4.16 Prove Lemma 4.6 by writing down the action of each gate in the RHS of Fig. 4.12 explicitly using bras, kets and $I, U, V, V^\dagger$. (For example, $U_{\mathrm{CNOT}} = |0\rangle\langle 0| \otimes I + |1\rangle\langle 1| \otimes X$ for a two-qubit system.)

A simple generalization of the above construction is applied to a controlled-U gate with three control bits as the following exercise shows.

EXERCISE 4.17 Show that the circuit in Fig. 4.13 is a controlled-U gate with three control bits, where $U = V^2$.

Now it should be clear how these examples are generalized to gates with more control bits.

PROPOSITION 4.2 The quantum circuit in Fig. 4.14 with $U = V^2$ is a decomposition of the controlled-U gate with $n - 1$ control bits.

The proof of the above proposition is very similar to that of Lemma 4.6 and Exercise 4.17 and is left as an exercise to the readers.

Theorem 4.2 has been now proved. ∎

Other types of gates are also implemented with single-qubit gates and the CNOT gates. See Barenco *et al.* [14] for further details. A few remarks are in order. The above controlled-U gate with $(n-1)$ control bits requires $\Theta(n^2)$ elementary gates.*† Let us write the number of the elementary gates required

*We call single-qubit gates and the CNOT gates elementary gates from now on.

†We will be less strict in the definition of "the order of." In the theory of computational complexity, people use three types of "order of magnitude." One writes "$f(n)$ is $O(g(n))$" if there exist $n_0 \in \mathbb{N}$ and $c \in \mathbb{R}$ such that $f(n) \leq cg(n)$ for $n \geq n_0$. In other words, O sets the asymptotic upper bound of $f(n)$. A function $f(n)$ is said to be $\Omega(g(n))$ if there exist

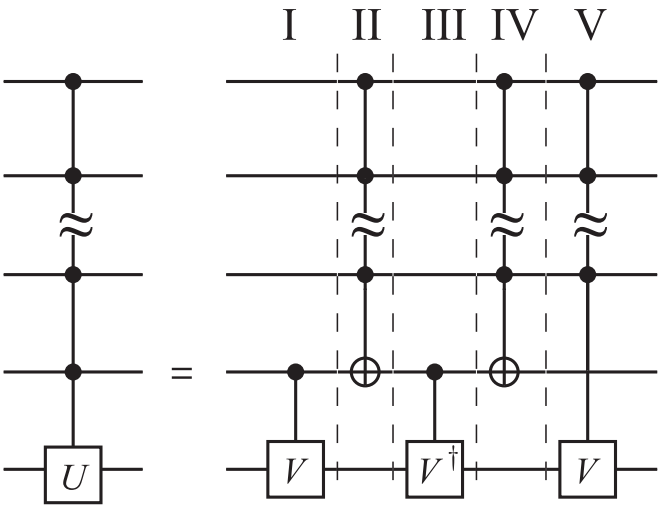


FIGURE 4.14
Decomposition of the $\mathrm{C}^{(n-1)}U$ gate. The number on the top denotes the layer refered to in the text.

to construct the gate in Fig. 4.14 by $C(n)$. Construction of layers I and III requires elementary gates whose number is independent of n. It can be shown that the number of the elementary gates required to construct the controlled NOT gate with $(n-2)$ control bits is $\Theta(n)$ [14]. Therefore layers II and IV require $\Theta(n)$ elementary gates. Finally the layer V, a controlled-V gate with $(n-2)$ control bits, requires $C(n-1)$ basic gates by definition. Thus we obtain a recursion relation

$$C(n) - C(n-1) = \Theta(n). \tag{4.59}$$

The solution to this recursion relation is

$$C(n) = \Theta(n^2). \tag{4.60}$$

Therefore, implementation of a controlled-U gate with $U \in \mathrm{U}(2)$ and $(n-1)$ control bits requires $\Theta(n^2)$ elementary gates.

$n_0 \in \mathbb{N}$ and $c \in \mathbb{R}$ such that $f(n) \geq cg(n)$ for $n \geq n_0$. In other words, Ω sets the asymptotic lower bound of $f(n)$. Finally $f(n)$ is said to be $\Theta(f(n))$ if $f(n)$ behaves asymptotically as $g(n)$, namely if $f(n)$ is both $O(g(n))$ and $\Omega(g(n))$.

4.7 Quantum Parallelism and Entanglement

Given an input x, a typical quantum computer computes $f(x)$ in such a way as

$$U_f : |x\rangle|0\rangle \mapsto |x\rangle|f(x)\rangle, \tag{4.61}$$

where U_f is a unitary matrix that implements the function f.

Suppose U_f acts on the input which is a superposition of many states. Since U_f is a linear operator, it acts simultaneously on all the vectors that constitute the superposition. Thus the output is also a superposition of all the results;

$$U_f : \sum_x |x\rangle|0\rangle \mapsto \sum_x |x\rangle|f(x)\rangle. \tag{4.62}$$

Namely, when the input is a superposition of n states, U_f computes n values $f(x_k)$ $(1 \leq k \leq n)$ simultaneously. This feature, called the *quantum parallelism*, gives a quantum computer an enormous power. A quantum computer is advantageous compared to a classical counterpart in that it makes use of this quantum parallelism and also entanglement.

A unitary transformation acts on a superposition of all possible states in most quantum algorithms. This superposition is prepared by the action of the Walsh-Hadamard transformation on an n-qubit register in the state $|00\ldots0\rangle = |0\rangle \otimes |0\rangle \otimes \ldots \otimes |0\rangle$ resulting in

$$\frac{1}{\sqrt{2^n}} \left(|00\ldots0\rangle + |00\ldots1\rangle + \ldots |11\ldots1\rangle\right) = \frac{1}{\sqrt{2^n}} \sum_{x=0}^{2^n-1} |x\rangle. \tag{4.63}$$

This state is a superposition of vectors encoding all the integers between 0 and $2^n - 1$. Then the linearity of U_f leads to

$$U_f \left(\frac{1}{\sqrt{2^n}} \sum_{x=0}^{2^n-1} |x\rangle|0\rangle \right) = \frac{1}{\sqrt{2^n}} \sum_{x=0}^{2^n-1} U_f|x\rangle|0\rangle = \frac{1}{\sqrt{2^n}} \sum_{x=0}^{2^n-1} |x\rangle|f(x)\rangle. \tag{4.64}$$

Note that the superposition is made of $2^n = e^{n \ln 2}$ states, which makes quantum computation exponentially faster than the classical counterpart in a certain kind of computation.

What about the limitation of a quantum computer? Let us consider the CCNOT gate, for example. This gate flips the third qubit if and only if the first and the second qubits are both in the state $|1\rangle$, while it leaves the third qubit unchanged otherwise. Let us fix the third input qubit to $|0\rangle$. It was shown in §4.3.3 that the third output is $|x \wedge y\rangle$, where $|x\rangle$ and $|y\rangle$ are the first and the second input qubit states, respectively. Suppose the input state is a superposition of all possible states while the third qubit is fixed to $|0\rangle$. This

can be achieved by the Walsh-Hadamard transformation as

$$U_{\rm H}|0\rangle \otimes U_{\rm H}|0\rangle \otimes |0\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \otimes \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \otimes |0\rangle$$
$$= \frac{1}{2}(|000\rangle + |010\rangle + |100\rangle + |110\rangle). \qquad (4.65)$$

By operating CCNOT on this state, we obtain

$$U_{\rm CCNOT}(U_{\rm H}|0\rangle \otimes U_{\rm H}|0\rangle \otimes |0\rangle) = \frac{1}{2}(|000\rangle + |010\rangle + |100\rangle + |111\rangle). \qquad (4.66)$$

This output may be thought of as the truth table of AND: $|x, y, x \wedge y\rangle$. It is extremely important to note that the output is an entangled state and the measurement projects the state to *one line* of the truth table, i.e., a single term in the RHS of Eq. (4.66). The order of the measurements of the three qubits does not matter at all. The measurement of the third qubit projects the state to the superposition of the states with the given value of the third qubit. Repeating the measurements on the rest of the qubits leads to the collapse of the output state to one of $|x, y, x \wedge y\rangle$.

There is no advantage of quantum computation over classical at this stage. This is because only *one* result may be obtained by a single set of measurements. What is worse, we cannot choose a specific vector $|x, y, x \wedge y\rangle$ at our will! Thus any quantum algorithm should be programmed so that the particular vector we want to observe should have larger probability to be measured compared to other vectors. This step has no classical analogy and is very special in quantum computation. The programming strategies to deal with this feature are [2]

1. to amplify the amplitude, and hence the probability, of the vector that we want to observe. This strategy is employed in the Grover's database search algorithm.

2. to find a common property of all the $f(x)$. This idea was employed in the quantum Fourier transform to find the order[‡] of f in the Shor's factoring algorithm.

Now we consider the power of entanglement. Suppose we have an n-qubit register, whose Hilbert space is 2^n-dimensional. Since each qubit has two basis vectors $|0\rangle$ and $|1\rangle$, there are $2n$ basis vectors (n $|0\rangle$'s and n $|1\rangle$'s) involved to span this 2^n-dimensional Hilbert space. Imagine that we have a single quantum system, instead, which has the same Hilbert space. One might think that the system may do the same quantum computation as the n-qubit register does. One possible problem is that one cannot measure the "kth digit"

[‡]Let $m, N \in \mathbb{N}$ $(m < N)$ be numbers coprime to each other. Then there exists $P \in \mathbb{N}$ such that $m^P \equiv 1 \pmod{N}$. The smallest such number P is called the **period** or the **order**. It is easily seen that $m^{x+P} \equiv m^x \pmod{N}$, $\forall x \in \mathbb{N}$.

leaving other digits unaffected. Even worse, consider how many different basis vectors are required for this system. This single system must have an enormous number, 2^n, of basis vectors! Let us consider 20 spin-1/2 particles in a magnetic field. We can employ the spin-up and spin-down energy eigenstates of each particle as the qubit basis vectors. Then there are merely 40 energy eigenvectors involved. Suppose we use energy eigenstates of a certain molecule to replace this register. Then we have to use $2^{20} \sim 10^6$ eigenstates. Separation and control of so many eigenstates are certainly beyond current technology. These simple consideration shows that multipartite implementation of a quantum algorithm requires an exponentially smaller number of basis vectors than monopartite implementation since the former makes use of entanglement as a computational resource.

References

[1] M. A. Neilsen and I. L. Chuang, *Quantum Computation and Quantum Information*, Cambridge University Press (2000).

[2] E. Riefell and W. Polak, ACM Computing Surveys (CSUR) **32**, 300 (2000).

[3] Y. Uesaka, *Mathematical Principle of Quantum Computation*, Corona Publishing, Tokyo, in Japanese (2000).

[4] W. K. Wootters and W. H. Zurek, Nature **299**, 802 (1982).

[5] D. Dieks, Phys. Lett. A **92**, 271 (1982).

[6] D. Bouwmeester *et al.*, Nature **390**, 575 (1997).

[7] D. Boschi *et al.*, Phys. Rev. Lett. **80**, 1121 (1998).

[8] I. Marcikic *et al.*, Nature **421**, 509 (2003).

[9] R. Ursin *et al.*, Nature **430**, 849 (2004).

[10] A. Furusawa *et al.*, Science **282**, 706 (1998).

[11] M. A. Nielsen *et al.*, Nature **396**, 52 (1998).

[12] M. Riebe *et al.*, Nature **429**, 734 (2004).

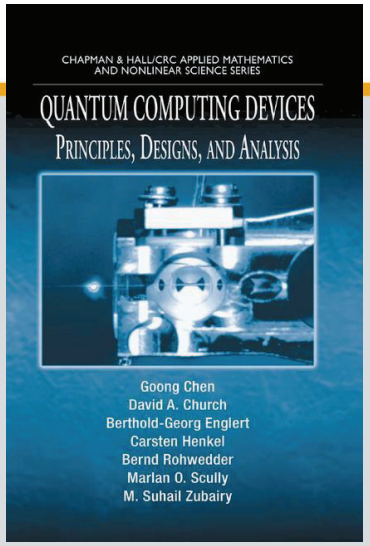
[13] M. D. Barret *et al.*, Nature **429**, 737 (2004).

[14] A. Barenco *et al.*, Phys. Rev. A **52**, 3457 (1995).

CHAPTER

5

SUPERCONDUCTING QUANTUM COMPUTING DEVICES



This chapter is excerpted from

Quantum Computing Devices, Principles, Designs, and Analysis

by Goong Chen, David A. Church, Berthold-Georg Englert, Carsten Henkel, Bernd Rohwedder, Marlan O. Scully, M. Suhail Zubairy

Learn more

Chapter 9

Superconducting Quantum Computing Devices

- Superconductivity history
- Components of superconducting circuits
- Quantization of superconducting circuits
- Types of qubits: charge, flux, phase and charge-flux
- Universality of quantum gates
- Measurements

Superconductivity manifests macroscopic quantum phenomena. A superconducting circuit composed of Josephson junctions, Cooper pair boxes, and rf-/dc-SQUID, properly miniaturized, becomes quantized and demonstrates Rabi oscillations and entanglement. In this chapter, we begin with an introduction of the history and elementary theory of superconductivity. Then we describe the building blocks of superconducting classical circuits and derive their canonical quantizations. The set up of qubits and superconducting quantum logic gates are then examined. Finally, ways to make measurements are discussed. We should remark that SQUID are widely regarded as the *most scalable* quantum computing device.

9.1 Introduction

Even though quantum effects are mostly observed in microscopic scales, they also manifest macroscopically. A particular case of such is *superconductivity*. Superconducting devices composed of Josephson junctions (JJ),

Cooper-pair boxes and SQUID (superconducting quantum interference devices) have been developed since the 1980s as magnetometers, gradiometers, gyroscopes, sensors, transistors, voltmeters, etc., to perform measurements on small magnetic fields, and to demonstrate the quantum effects of tunneling, resonance and coherence [6, 11, 18, 27, 30, 35]. Many industrial and medical applications have also resulted: maglev trains, superconducting power generator, cables and transformers, MRI and NMR for medical scans, to mention a few. With the advances in solid-state lithography and thin-film technology, superconducting devices have the great advantage of being easily scalable and engineering-designable. For a bulk superconductor, if its size is reduced smaller and smaller, then the quasi-continuous electron conduction band therein turns into discrete energy levels. In principle, such energy levels can be used to constitute a qubit. The first demonstration of quantum-coherent oscillations of a Josephson "charge qubit" in a superposition of eigenstates was made by Nakamura, et al. [20] in 1999. Ever since, theoretically and experimentally there has been steady progress. New proposals for qubits based on *charges, flux, phase* and *charge-flux* have been made, with observations of microwave-induced Rabi oscillations of 2-level populations in those qubit systems [8, 9, 10, 37, 38].

The organization of this chapter is made as follows: we will first introduce superconductivity in Section 9.2, the Josephson junction in Section 9.3, and the elementary superconducting circuits in Section 9.4. Superconducting quantum circuits and gates are studied in Sections 9.5 and 9.6, and conclude with measurements in Section 9.7.

9.2 Superconductivity

We begin by giving a brief historical account [13, 23, 33]. Superconductivity was discovered in 1911 by the Dutch physicist Heike Kamerlingh Onnes (1853–1926), who dedicated his career to the exploration of extremely cold refrigeration. In 1908, he successfully liquefied helium by cooling it to -452° F (4 K). In 1911, he began the investigation of the electrical properties of metals in extremely cold temperatures, using liquid helium. He noticed that for solid mercury at cryogenic temperature of 4.2 K, its electric resistivity abruptly disappeared (as if there were a jump discontinuity). This is the discovery of superconductivity, and Onnes was awarded the Nobel Prize of Physics in 1913.

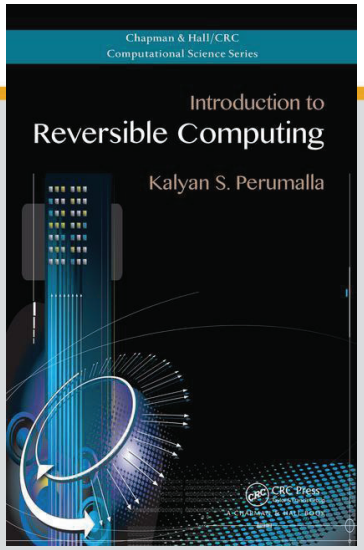
Subsequently, superconductivity was found in other materials. For example, lead was found to superconduct at 7 K, and (in 1941) niobium nitride was found to superconduct at 16 K.

Important understanding of superconductivity was made by Meissner and Ochsenfeld in 1933 who discovered that superconductors expelled applied magnetic fields, a phenomenon which has come to be known as the *Meissner effect*. In 1935, F. and H. London showed that the Meissner effect was a consequence of the minimization of the electromagnetic free energy

CHAPTER

6

APPLICATION AREAS



This chapter is excerpted from
Introduction to Reversible Computing
by Kalyan S. Perumalla

Learn more

Chapter 2

Application Areas

2.1 General Reversible Computing Problem

At a high level, the challenge of reversible computing may be stated as follows: Given a program, the program must be executed in such a way that its forward progress could be paused at any moment and the direction of its execution can be reversed to retrace its previous steps exactly. The backward execution can also be paused at any moment and the forward execution of the program can be resumed. This ability to pause the forward or backward execution at any point and the ability to switch the direction of execution is the generalized challenge of reversible computing. Specific variants and specializations of this general reversible computing problem arise in different contexts. Historically, low power computing has been a major motivating factor behind the development of reversible computing. Over time, additional areas have emerged in which reversible computing has found applications.

2.2 Energy-Optimal Computing

When contemplating ways to reduce the energy consumed to accomplish a certain computation, the thought process naturally leads to the determination of the fundamental lower bound for the same. In this process, it was discovered that the theoretical lower bound on energy is directly related to the reversibility of computation [Landauer, 1961]. More precisely, there are ways in which the original (forward) computation could be relaxed into a more generalized setting of *reversible* computing, (these ideas are elaborated more later in Chapter 4 and Chapter 5). Arguments based on thermodynamics helped keep the treatment beyond any specific hardware technologies [Bennett, 2003]. In an ideal setting, it was shown that no energy need be fundamentally lost in performing any given computation [Bennett, 1973, 1982]. As a corollary, this implies that, in theory, arbitrarily large fractions of the energy consumed for a computation should be recoverable for useful work again, although there would be an underlying trade-off between the efficiency of such recovery and the speed of computation (i.e., computation time approaches infinity as the recoverable fraction approaches unity). Based on the thermodynamics of computation, various hardware technologies have been proposed, designed, and explored, and some have even been implemented as proofs of concept [Ren and Semenov, 2011].

The theory about the minimality of energy spent for computation is built over the most basic bit operation, namely *bit erasure* (e.g., resetting a bit to 0 independent of whether its current bit value is 0 or 1 [Bub, 2001]). First, it is argued that bit erasure is the only operation that fundamentally consumes energy in any computation. An equivalent way of saying the same is the following: any computation can be carried out in such a way that only the bit erasures performed as part of that computation consume irrecoverable energy, while the energy used in all other bit operations is fully recoverable and reusable.

Given that the bit erasures form the sole cause of irrecoverable energy consumption, the important question that follows is whether every computation can be performed without bit erasures. This question was answered in the positive by introducing reversible computing. A notion of the *reverse* of any computation is introduced, which is then used in a higher-level algorithm to accomplish any computation without bit erasures (see Chapter 4 and Chapter 6). In this way, the reduction and recycling of energy is one of the fundamental applications of reversible computing, holding far-reaching potential in the future of computing.

2.3 Parallel Computing and Synchronization

Reversible computing finds use in addressing some of the challenges in large-scale parallel computing such as reduction of synchronization overheads and increasing the concurrency dynamically. As of this writing, parallel computers with millions of processing units is a reality [TOP500.org, 2013], and the scale is only envisioned to increase in the future.

Synchronization is that critical aspect of parallel computation which ensures that the overall execution obeys and incorporates all the inter-processor data dependencies of the application [Grama et al., 2003, Raynal, 2012]. A runtime component of parallel computing is needed in all applications to realize proper synchronization, and this runtime synchronization costs the application some wasted processing time (also called *blocked time*) at some processors during different periods of execution. Some of this synchronization is fundamental in nature, in the sense that the inter-processor data dependencies impose a theoretical total parallel execution time (given by critical path analysis). However, this lower bound is often difficult to achieve in practice, and the programs are written with very conservative order of execution. For example, global barriers are invoked in the program to ensure all processors reach known points in the program before proceeding further. The cost for each such synchronization operation becomes significantly large as the number of processors increases, with some applications spending more than 50% of their time in synchronization versus actual, useful computation.

Note that any synchronization operation is pure overhead, in the sense that it only adds a component of execution time (and energy) that is simply absent in sequential computation. The synchronization problem is an important issue at the highest scales of hardware, despite some advances in underlying network hardware technologies to speed up the synchronization operations themselves.

2.3.1 Asynchronous Computing

The synchronization problem can be solved by finding ways to increase concurrency across processors such that there is increasingly more useful computation performed per synchronization operation. One approach is to incrementally reorganize operations to overlap some computation with communication or synchronization operations. The more general method is to use a relaxation to a generalized asynchronous reversible execution, allowing the application to uncover maximum concurrency at runtime.

- **Computation–communication overlap** One specialized way to increase concurrency is by using non-blocking synchronization operations, by which processors perform some local operation while a synchronization operation is in progress. For example, non-blocking collective operations such as global barriers or global reductions may be initiated

and completed asynchronously. However, the amount of increased concurrency uncovered by this approach is limited to the amount of local computation that can be performed during synchronization.

- **Generalized asynchronous reversible execution** A generalized way to increase concurrency is to relax the parallel execution model to one in which all processors can execute the application both forward and backward, and let processors execute local computation asynchronously with respect to each other. Communication and synchronization proceed in the background, and any violations of data dependency order are detected at runtime and corrected by relying on reversible execution. Executing backward, the processor is rolled back from its current point of execution to the point of dependency violation, and, after incorporating the correction (e.g., new data arriving from another processor), forward execution is restarted from the corrected point.

In the specialized communication–computation overlap approach, execution follows conventional computation, that is, it is still fundamentally forward execution-based, with no tolerance for any data dependency violation. In the generalized asynchronous reversible execution, the view of execution is fundamentally different, that is, execution is assumed to be reversible from the outset, with a runtime component (supervisor, coordinator, controller, or engine) orchestrating the global execution such that the overall execution eventually provides the same answers as one that incorporated all inter-processor data dependencies. Without reversibility, it is not possible to relax execution to uncover any more concurrency than the most conservative one that avoids even *transient* violations of dependencies.

The best demonstrations of the potential for the reversible execution to reduce synchronization overheads in large-scale parallel computing has been in the area of parallel discrete event simulation (PDES). The Time Warp algorithm [Jefferson, 1985] for PDES has been gainfully employed in multiple large-scale applications (e.g., epidemiological outbreak simulations) using rollback-based synchronization implemented on supercomputers. The algorithm is built on relaxation to a reversible execution, in which local events at a processor are processed optimistically even while lower bound guarantees on the global virtual time are being computed across all processors. Also, theoretical analyses have been performed to show the gains with rollback dynamics in the presence of primary and secondary rollbacks in Time Warp [Akyildiz et al., 1992].

2.3.2 Supercriticality

Critical Path (CP) analysis of a parallel application is the determination of the inherently sequential path(s) of data dependencies from the start to end of the computation [Yang and Miller, 1988, Hollingsworth, 1998]. Clearly, the longest path from start to end is the most critical one. It is well-known that the

critical path(s) in any application determine(s) a fundamental lower bound on the parallel application's total computation time. In particular, CP imposes an absolute upper bound on the achievable parallel efficiency or speedup. However, an important aspect about this CP-bounded parallel efficiency is that it rests on the fundamental assumption of conventional forward-only (irreversible) computation. If the assumption of irreversibility is relaxed, and the lower bound is examined with a new assumption of reversible execution, it is in fact possible to sometimes exceed the theoretical CP-based lower bound on computation time (and, consequently, upper bound on parallel efficiency and speedup).

To illustrate, suppose computations appear as a sequence of updates, of the form

$$U_{ij} : x_i \leftarrow f_j(\mathcal{R}_{ij}),$$

where x_i is a variable being updated, $\mathcal{R}_{ij}$ is the set of all variables that x_i depends on for update j, and f_j is a computable expression on that set of variables. In normal execution, the exact values of all variables in $\mathcal{R}_{ij}$ should be available before the update can be performed. In this model, the chain of these dependencies gives the CP-bounded computation time.

Now, consider the new computation model in which the computation is relaxed to be reversible, i.e., f_j^{-1} exists and is available for every j. In this reversible model, suppose U_{ij} is computed without waiting for the most recent values of $\mathcal{R}_{ij}$. After this out-of-order update, when new values R_{ij}^* for $\mathcal{R}_{ij}$ are available, the correct new value for x_i must be recomputed. This is accomplished by first invoking f_j^{-1}, then incorporating R_{ij}^*, and finally recomputing U_{ij}. This reversal and recomputation can reduce synchronization effects via reversible execution, yet it does not defeat the fundamental critical path-based lower bound on computation time.

To improve the execution beyond the CP limit, consider the following variation: when new values R_{ij}^* of $\mathcal{R}_{ij}$ are received, a temporary value $x_i^* = f_j(R_{ij}^*)$ is first computed and compared with the current value of x_i that was computed out of order. If they are equal, that is, if $x_i = x_i^*$, then there is no need to overwrite/update x_i because it already has the correct value. There are two extremely important implications of this obviation of update to x_i:

- No succeeding nodes in the dependency graph will need updates, that is, no U_{ik}, $j < k$, needs to be recomputed (in case they have already been computed).

- Due to computation of U_{ij} ahead of its position in the dependency graph, its computation is overlapped with other preceding computations U_{ik} for some $k < j$, thereby reducing the overall completion time of the graph.

This phenomenon of the potential to defeat the lower bound dictated by the critical path is called supercriticality [Jefferson and Reiher, 1991]. It not only preserves the correctness of the final answer, but, more importantly, defeats

the lower bound on computation time if U_{ij} is in the critical path of the overall computation Note that supercriticality is only possible due to reversible execution. In the example, if x_i is not equal to x_i^*, then f^{-1} would have to be invoked to correct the execution to conform to the dependencies. Thus, reversible computing can be used to sometimes provide supercritical parallel computation that exceeds the parallel efficiency and speedup otherwise prescribed by the critical path.

2.3.3 Performance Effects

When reversibility is introduced to relieve synchronization costs, the program must be instrumented to enable forward as well as backward modes of computation. Two broad approaches are (1) saving copies of variables to a memory trace before they are modified in the forward mode and copying back from the trace in reverse mode, or (2) avoid saving values of variables in forward mode, but instead invoke inverses of individual operations in reverse mode. In hardware technologies that have high memory latencies (i.e., time taken to read from or write to memory locations) compared to speed of computation (such as computing arithmetic operations), it is vastly more efficient to perform inverse computations rather than store and retrieve values from memory.

The so-called "memory wall" being faced in the 2000s-2010s exhibits this situation in which the increasing speeds of central processing units (CPUs) make memory operations relatively very expensive. This is because of the increasing ratio of CPU clock rates compared to those of memory units, and also because of multi-core technologies that result in an increase in the number of CPUs accessing the same number of memory units. This widening gap between the speeds of arithmetic/logic operations and memory storage/retrieval operations can be bridged via reversible computing.

Operationally, the performance differences manifest themselves in terms of memory behavior such as caching effects at all levels (L1, L2, L3), and Translation Look-aside Buffer (TLB) effects. Memory-based (checkpointing) reversal suffers from the following drawbacks: (1) the total amount of memory needed for the reversible program is larger than that without reversible execution; (2) the time cost of copying values before being modified in the forward mode adds to the forward execution overhead, potentially slowing down forward execution; and (3) memory subsystem behavior (cache and TLB) can be negatively impacted due to decrease in locality and due to memory accesses spilling over cache size limits. Computation-based reversal can relieve these drawbacks because (1) little additional memory is needed for reversal of arithmetic operations; (2) the forward code is largely unaffected, thus retaining the original forward execution speed; and (3) memory subsystem effects are not significantly altered from the original forward mode.

2.4 Processor Architectures

In processor architecture technology, the need for reversible execution arises in two distinct ways: (1) speculative execution, and (2) very large instruction word. In both cases, reversible execution is used to correctly uncover dynamic parallelism from an otherwise sequentially specified computation.

2.4.1 Speculative Execution

Consider a single sequence of instructions $\mathcal{S} =< \ldots, I_i, \ldots >$ being fetched, decoded, and executed by a processor. In general, instruction I_i must be executed only after all instructions $I_j, j < i$ are completed, because the set of variables I_i depends on may be modified by those prior instructions. However, such dependency may not always be actually present in a small window of the sequence, and some of the instructions may be safely processed concurrently without waiting for the others to finish. For example, I_1=$\boxed{a \leftarrow b \times c}$ and I_2=$\boxed{d \leftarrow e + f}$ can be processed concurrently even if specified sequentially because there is no intersection of variables being read or modified. More complex instances involve implied dependencies such as from array operations, dereferences, and register conflicts. The problem is that the correct execution order cannot be determined statically, a priori. In general, the set of variables being modified by an instruction is not necessarily known until it is actually decoded and executed (e.g., when indirect references such as array indices or pointers are used to refer to the variable being read or modified). Thus, there is a conflict between the possibility of executing a few instructions concurrently to increase the overall processing speed and the possibility of the concurrently executed instructions incorrectly affecting each other that might result in wrong results.

A way to dynamically exploit the potential concurrency is via *speculative execution* in which a later instruction is issued even before the earlier instructions are completed [Dubois et al., 2012]. After the speculatively executed instruction is executed, conflict detection is performed to see whether there was an intersection in the set of affected variables (i.e., a violation in read/write dependencies). If a conflict is detected, the results of the speculatively executed instruction are quashed and that instruction is restarted from the beginning. Alternatively, fix-up code may be invoked to repair the incorrectly (speculatively) computed results, instead of discarding everything and restarting the speculatively executed instructions from scratch. A fair amount of processor infrastructure and compiler support is usually needed to accomplish speculative execution. For example, at the processor-level, instruction execution must be relaxed so that modifications to registers are held in temporary (shadow) registers for every speculatively executed instruction, and the changes are committed from the temporary to actual registers only when

conflict resolution succeeds. Reverse computation could be used to avoid these temporary registers by invoking an inverse instruction to restore the affected register value if the speculative instruction was found to be conflicting. Speculative execution is widely used in many modern processors to increase the processor instruction throughput.

2.4.2 Very Large Instruction Word

In a manner complementary to speculative execution, suppose each instruction I_i is in fact a vector of sub-instructions, $I_i = [s_{i1}, \ldots, s_{ij}, \ldots, s_{in}]$, where the sub-instructions s_{ij} can all be processed concurrently to constitute the execution of I_i. Because the number of sub-instructions can be large, the width of each instruction becomes large, giving the name *Very Large Instruction Word* (VLIW) [Fisher, 1983, Dubois et al., 2012]. Determination of which sub-instructions are possible to execute as an aggregate instruction is accomplished within the compiler that generates the VLIW code. While the *processor* is responsible for determining the concurrency and reversal of instructions in speculative execution systems, the *compiler* is responsible to determine the concurrency and reversal of sub-instructions in VLIW systems. The VLIW compiler attempts to generate instructions that have a dense packing of sub-instructions; however, it may have to reorder sub-instructions in generating such dense packings. The compiler is then responsible for generating compensation code that reverses the effects of incorrectly (speculatively) ordered sub-instructions if they result in data conflicts at runtime. The most well-known packing method is the Trace Scheduling approach [Fisher, 1981, 1983] in which the path taken by the code in an execution is used to generate the basic instruction sequence (independent of branch conditions), and reversal or compensatory code is added to this sequence to recover from deviations from the assumed path.

2.4.3 Anti-Memoization

A form of reversible computing can be employed in recovering temporary values that are usually pushed from registers to main memory when a register conflict occurs (due to register file size limitation or due to named registers being reused and overwritten across operations). If a register value is potentially about to be lost, it is usually pushed to memory and later loaded back from memory when it is again needed/used. Because memory operations are orders of magnitude slower than register speeds, it is desirable to keep operations confined to register accesses as much as possible. Thus, when the intermediate value that is lost in a register (due to being overwritten) is again needed, it is sometimes possible to either recompute or reverse compute earlier operations to recover the lost register value, instead of relying on storage to/retrieval from main memory.

A generalization of such memory *versus* computation trade-off is the con-

cept of memoization and anti-memoization. The method of storing the value of an intermediate computation in memory for reuse in later computation is called memoization [Hoffmann, 1992] (note that the term *memoization* is different from *memorization*). Such an optimization that trades off memory for computation arises either from explicit programmer intent or from automation techniques such as code generation by compilers. While memoization works best when memory is cheaper than recomputation, it can in fact degrade performance compared to recomputation when memory access cost is much higher than recomputation cost. However, if the code has already been written using memoization, the memoized values must be recovered using recomputation. This can be done via reverse computing to the most recent position at which the variable was last stored to memory, and then recomputing the expression that led to the stored value. We call this approach *anti-memoization*, which is the process of undoing memoization. An instance of this high-level approach has been applied at the level of register value recovery (called register rematerialization) [Bahi and Eisenbeis, 2011, 2012] to improve the overall application performance.

2.5 Debugging

Reversibility of execution is very useful in the process of debugging programs in an efficient and convenient fashion. When running a program, if an unexpected condition or undesirable results occur, the program's execution needs to be retraced to find the precise point where the deviation from desired behavior actually originated. In order to be able to step backward, the program state needs to be saved before every forward operation. However, the amount of saved state can become extremely large because the computer executes millions of instructions per second; in fact, the accumulated state grows so quickly that it may be infeasible to store the program trace for long program execution length. The trace for only a small window of execution may be stored. Yet, the distance between the point of the bug's manifestation and the original source location of the bug may be large, making it infeasible to step backward to the correct origin of the bug via trace traversal. This is where reversible computing can be applied—instead of saving/restoring the values of the variables to/from memory, the inverses of the instructions can be executed backward to traverse the program in reverse from the bug manifestation point until the bug's source is determined.

The difficulty in debugging is especially pronounced in (1) assembly-level debugging, due to very large sizes of traces; and (2) parallel computing. Reversible computing is either a significantly more efficient method or the only feasible method in these cases.

In debugging programs at the level of its assembly language, since the

number of assembly instructions or machine code is extremely large, the trace sizes needed to enable backward traversal in assembly code grows extremely quickly. Reversible computing is the only feasible method to enable bi-directional movement in assembly instruction streams for efficient debugging. Without reversible computing, assembly-level debugging either slows down forward execution tremendously (due to the introduction of the high cost of trace generation operations before every assembly instruction) or is infeasible because the trace does not fit in the available computer memory.

The problem of debugging has been described and various methods surveyed extensively in the context of high-level programs such as text editors and user interface systems [Teitelman, 1975, 1984, Archer et al., 1984, Leeman, 1986]. General bi-directional movement for debugging in general has been studied [Boothe, 2000], especially taking care of logging all the relevant states including system call data to provide determinism in repeated bi-directional movement across already-executed instructions. Assembly-level debugging via reverse execution was studied and optimizations proposed to significantly reduce the amount of memory trace needed for backward traversal [Akgul and Mooney III, 2004, Lee, 2007]. While reversible execution has been successfully applied in parallel computing applications (e.g., [Carothers et al., 1999]), full-scale application of reversible debugging in large parallel systems is a relatively open item of research.

2.6 Source Code Control Systems

Source code control systems provide a world view in which modifications to a set of objects can be tracked, traversed, and manipulated along different logical timelines. The timelines are formed by sequences of individual or grouped modifications to objects. In general, the timelines are related to each other in the form of directed acyclic graphs (DAGs). Most often, each object is a named file in a file system. Modifications to the file are characterized in terms of edits or changes to the individual lines, assuming that the file is a text file. Popular source code control systems such as `git` [Loeliger and McCullough, 2012, Somasundaram, 2013] and `mercurial` [O'Sullivan, 2009], as well as older systems such as `svn` [Collins-Sussman et al., 2009] and `cvs` [Thomas and Hunt, 2003], provide reversible sequences of edits to sets of files. They also provide ways to identify and extract, via user-specified or system-generated naming, individual paths along the DAG of timelines.

Overall, the systems provide a way to view the evolution of the object values as a reversible computation that can be traversed forward or backward and also merge different timelines. When each object is viewed as a program variable holding data, changes to the values of the variables can be made in a reversible fashion to save and restore the data values. Thus, although

the granularity of the objects is different from the variables in a conventional computer program (files often have much larger bit lengths than typical program variables), the notion of reversibility in source code control systems is analogous to that of a program's state evolution. Concepts common to both include the *undo* and *commit* operations, while the *branching* and *merging* are operations that are somewhat specific to source code control systems.

2.7 Fault Detection

In fault detection, the idea is to utilize reversible computing to periodically verify if the forward computation was performed correctly, as follows. Given a code fragment P, after it has been executed forward as $F(P)$, it is to be ascertained if there had been a faulty execution of any portion of P (for example, values of some variables may become incorrect due to low-probability errors in the memory subsystem that flip one or more bits randomly due to electrical faults). If the code is executed backward as $R(F(P))$, then the initial values of the variables prior to execution of $F(P)$ will not match the values restored by $R(F(P))$; such a mismatch can be used to signal an error condition. This method of detection relies on two important, reasonable assumptions:

1. The type of errors encountered in the forward path are rare events, and hence the backward path is not susceptible to the same errors as well. Because $F(P)$ executed under errors and $R(P)$ did not, their net results can be expected to differ from each other.

2. In the rare event that the backward path also encounters errors, we can reasonably assume that the forward errors and backward errors do not cancel each other.

Reversible computing can be used for fault detection and, in some cases, fault correction. Reverse execution can be used in trapping errors in either forward code or the reverse code, or even in the implementation of the reverse compiler. This is achieved by checking the following simple *necessary* correctness condition at runtime [Bishop, 1997], which, when applied on every local variable and global variable, is useful in detecting errors in the code or the compiler.

> Suppose a variable `var` is initialized to `expression` in the forward execution. At the end of the reverse execution, the final value of the variable `var` must equal `expression`.

Another simple correctness condition is the following, which is a generalization of the end-of-tape condition described in [Bishop, 1997]:

Suppose the bit/byte tape is at position P before the forward execution of a function f, and later the reverse of the function is executed. Then, by the end of the reverse execution, the tape must be rewound to the same position P.

2.8 Fault Tolerance

Fault tolerant computation in a parallel or distributed system is the ability to gracefully continue execution of an application despite transient faults or failures of system components at runtime. Fault tolerance is an extremely difficult capability to achieve in parallel systems, particularly when the number of components in the system is very large. Simplistic schemes rely on periodically saving the entire application state to persistent storage and restoring this state at all processors for recovering from failures. However, such schemes are woefully non-scalable and break down with large numbers of processors. More scalable solutions do not rely on global checkpoint/restart views, but use in-memory solutions. Among them, rollback-based recovery is an important algorithmic core underlying scalable parallel computing, appearing in the form of system support, middleware, or applications. For example, efficient rollback-based fault tolerance approaches (e.g., [Manivannan and Singhal, 1996, Kim et al., 1996], among many others) rely critically on the ability of processors to revert their state back to a point in the past.

Thus, processors need the ability to go back to a previous point in execution dynamically on demand, when they are informed of a *fault*. The definition of a fault varies with application. Often, a fault is the detection of a failure of a processor. In other software-level rollback schemes, a fault is the detection of a violation of application-specific event order for correctness. For example, in large-scale Time Warp [Jefferson, 1985, Perumalla, 2007], a *primary rollback* results when an event is received with timestamp smaller than the current virtual time of the receiving processor. When previously sent messages are taken back as a result of primary rollback, further rollbacks, called *secondary rollbacks*, are transitively propagated to other processors.

Figure 2.1 shows the schematic for this general setting [Perumalla and Park, 2013]. When a processor P_f encounters a fault, it restarts from the most recently saved checkpoint LC_f and informs all other processors $\{P_r\}$ to roll back to the point corresponding to the program state of LC_f. Every rolling back processor P_r can invoke reverse code to recover the state corresponding to LC_f.

Note that LC_f and LC_r are in general different because, for maximum efficiency, each processor is allowed to asynchronously and *infrequently* initiate a checkpoint of its own state to persistent storage. Note also that every rolling

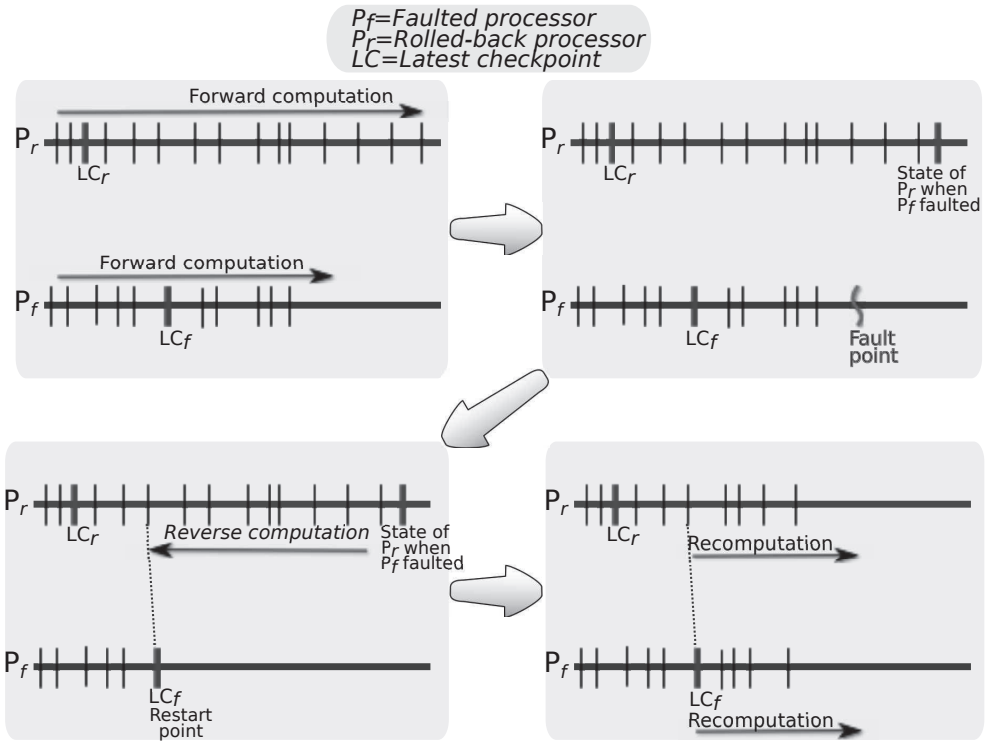


FIGURE 2.1: Schematic of asynchronous recovery sequence using reverse computation-based rollback.

Each thin vertical line denotes an update to the local state of a processor. Each thick vertical line marks a full checkpoint of the processor state to persistent storage. Note that there are very many processors indicated by P_r that are affected due to the fault at P_f. All of them need to be rolled back, although, for simplicity, only one processor is shown in the illustration.

back (non-faulted) processor P_r need only use reverse computation, but does *not* have to access its own checkpoint LC_r. The checkpoint is only used by a processor if and only if it is the faulted processor. This particular aspect *dramatically* relieves congestion in the system in terms of lowered pressure on the memory bus and on the file system.

In Figure 2.1, P_r's corresponding state of LC_f is joined by the dashed line between the processors. The faulted processor P_f restarts from LC_f, because the state from LC_f to the fault point is assumed to be lost or unavailable due to failure.

2.9 Database Transactions

The concept of reversal of operations is fundamental to the backbone of databases, namely, database transactions. Databases provide the key properties of atomicity, consistency, isolation, and durability (ACID properties) for any groups of operations called transactions [Date, 2003]. Applications can be written correctly and conveniently using these properties. Database systems internally provide sophisticated and elaborate implementations to provide support for *transactions* [Berstein and Newcomer, 2009], fundamentally relying on the concepts of reversal and replay of indivisible groups of operations. Operations are logged into persistent storage, and complex algorithms ensure that the state of the logs and the state of the data stored in the database are always consistent with each other. Reversibility of a transaction is key to correct operation because a transaction may be aborted at any time (either intentionally by the user due to change of mind, or unintentionally by the user such as from loss of network connection, or automatically by the system due to faults such as power failures). When a transaction is aborted mid-way, all operations performed as part of the transaction must be undone to preserve the critical ACID properties.

In a common example often used to illustrate the reversal of operations for transactions, two transactions T_1 and T_2 operate on a database of two bank accounts A_1 and A_2. Transaction T_1 attempts to transfer x dollars from A_1 to A_2, while T_2 attempts to transfer y dollars from A_2 to A_1. To transfer, T_1 first debits A_1 by x dollars and then credits A_2 by the same amount. Similarly, T_2 first debits A_2 by y dollars and credits A_1 by the same amount. Thus, T_1=$\boxed{A_1 \leftarrow A_1 - x; A_2 \leftarrow A_2 + x}$ and T_2=$\boxed{A_2 \leftarrow A_2 - y; A_1 \leftarrow A_2 + y}$. If the database state before either transaction is $\boxed{S}$, then the transaction system ensures that the final system state after the transactions is only one among $\boxed{S \mapsto T_1}$, $\boxed{S \mapsto T_2}$, $\boxed{S \mapsto T_1 \mapsto T_2}$, or $\boxed{S \mapsto T_2 \mapsto T_1}$, where $\boxed{P \mapsto Q}$ denotes the state obtained by application of transaction Q on state P. In other words, it ensures that at most one transaction succeeds, or if both succeed, the state is exactly *as though* one transaction is entirely preceded by the other (i.e., not interleaved). To achieve these semantics, reversal of operations is employed if and when any transaction is aborted before it is executed to completion (i.e., "committed"). For example, if T_1 is aborted after its first step $\boxed{A_1 \leftarrow A_1 - x}$, this partially executed transaction can be undone by executing the inverse $\boxed{A_1 \leftarrow A_1 + x}$. Similarly, if T_1 completes both steps, but somehow fails before it is committed, both steps are to be undone, in reverse order; That is, by executing the inverse $\boxed{A_2 \leftarrow A_2 - x; A_1 \leftarrow A_1 + x}$. While this simple example illustrates the reversal of aborted transactions in databases, in practice the database system infrastructure is much more elaborate and complex to support very fast operation of a large number of transactions containing

a richer set of operations. Different reversal technologies are employed to roll back transactions, the superset of which is in the realm of reversible computing.

2.10 Quantum Computing

Reversible computing is an inherent feature of Quantum Computing [Bennett et al., 1997, Rieffel and Polak, 2011]. In Quantum Computing, computation is a sequence of *unitary* operation of the computer state. Because every unitary matrix is reversible by definition, the entire sequence is inherently reversible.

2.11 Additional Applications

Reversible computing finds use in different forms and to varying degrees in several other applications. The adjoint methods in Automatic Differentiation (AD) can exploit reversible computation in the so-called reverse mode of evaluation [Griewank and Walther, 2008]. User-friendly graphical interface-based applications commonly provide interfaces to allow users to perform certain operations that can be undone on demand, allowing the user to explore different operations. Sports scoreboard maintenance and recording systems are built using reversible computing principles to enable automated generation of inverse actions for normal actions, to deal with inherently error-prone processes in real-time scoring [Briggs, 1987]. Computer file systems, such as the Apple Macintosh Operating System's Time Machine® functionality, provide a reversible view of all changes to the file system contents.

Interesting Contexts

Software for reversible execution found an interesting application in the 1980s when J. Briggs reported a way to use reverse code generation and reversible execution to undo incorrect updates to the scoreboard in cricket matches [Briggs, 1987]. Reversal code was automatically generated with the objective of minimization of the state information to be stored to enable reversal.

In the game of cricket [Knight et al., 2007], as in other sports, mistakes and corrections inevitably occur in recording and publishing the scores even as the game is in progress. Errors can appear in two ways: (1) the scoreboard operators may commit errors of omission or commission in entering events into the computing system, or (2) the game itself may experience reversals of decisions and other sport-specific updates, such as umpire's corrections. In each of these types, there are many occasions where updates are rolled back and corrected, naturally warranting reversible execution in the scoring program software.

In an unrelated context, reversibility appears in musical compositions in the concepts of the *mirror canon* and the *crab canon* (also called *cancrizans*) in which the musical notes are mirror images of themselves (or palindromic) [Hugo, 1904]. A popular reference to crab canons is the collection by J. S. Bach titled "The Musical Offering" [Bach, 1747]. The notes of a crab canon written over a Möbius strip [Pickover, 2007] can be played back and forth ad infinitum.