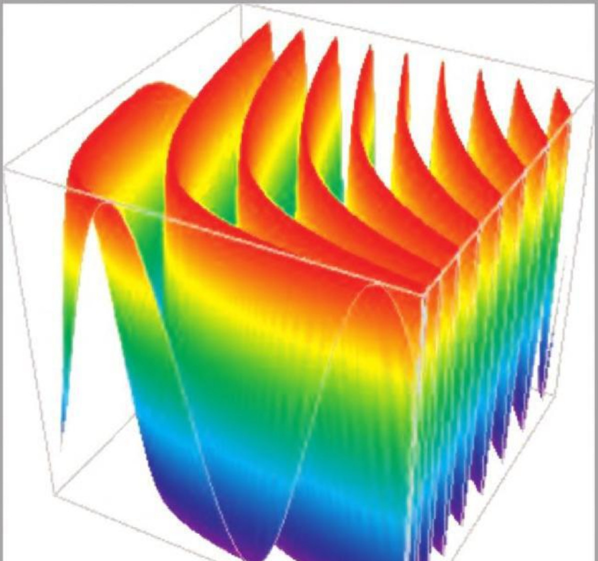
TEXTS IN COMPUTER SCIENCE

Explorations in Quantum Computing



Colin P. Williams

SECOND EDITION

Springer

Texts in Computer Science

Editors
David Gries
Fred B. Schneider

For further volumes:
http://www.springer.com/series/3191

Dr. Colin P. Williams
California Institute of Technology
NASA Jet Propulsion Laboratory
Oak Grove Drive 4800
Pasadena, CA 91109-8099
USA
Colin.P.Williams@jpl.nasa.gov

Series Editors

David Gries	Fred B. Schneider
Department of Computer Science	Department of Computer Science
Upson Hall	Upson Hall
Cornell University	Cornell University
Ithaca, NY 14853-7501, USA	Ithaca, NY 14853-7501, USA

ISSN 1868-0941 e-ISSN 1868-095X
ISBN 978-1-84628-886-9 e-ISBN 978-1-84628-887-6
DOI 10.1007/978-1-84628-887-6
Springer London Dordrecht Heidelberg New York

British Library Cataloguing in Publication Data
A catalogue record for this book is available from the British Library

Library of Congress Control Number: 2010936191

Cover design: SPI, Puducherry, India

Printed on acid-free paper

Springer is part of Springer Science+Business Media (www.springer.com)

Contents

Part I
What is Quantum Computing?

Chapter 1
Introduction

"*The theory of computation has traditionally been studied almost entirely in the abstract, as a topic in pure mathematics. This is to miss the point of it. Computers are physical objects, and computations are physical processes. What computers can or cannot compute is determined by the laws of physics alone, and not by pure mathematics*"
– David Deutsch[1]

Over the past 50 years there has been an astonishing miniaturization in computer technology. Whereas a microprocessor in 1971 contained roughly 2,300 transistors, a modern microprocessor of the same size contains in excess of one billion transistors. Throughout this evolution, even though there have been several changes in how computer hardware is implemented, the *same* underlying mathematical model of a computer has held sway. However, if current trends continue, by the year 2020 the basic components of a computer will be the size of individual atoms. At such scales, the mathematical theory underpinning modern computer science will cease to be valid. Instead, scientists are inventing a new theory, called "quantum computing", which is built upon the recognition that a computing device is a *physical* system governed by *physical* laws, and at very small scales, the appropriate laws are those of quantum mechanics—the most accurate model of reality that is currently known.

There are two attitudes one could adopt regarding the necessity of incorporating quantum mechanical effects into computing machinery. One response it to strive to suppress the quantum effects and still preserve a semblance of classicality even though the computational elements are very small. The other approach is to embrace quantum effects and try to find clever ways to enhance and sustain them to achieve old computational goals in new ways. Quantum computing attempts to pursue the latter strategy by *harnessing* quintessentially quantum effects.

Remarkably, this new theory of quantum computer science predicts that quantum computers will be able to perform certain computational tasks in phenomenally

[1]Source: Opening words of Chap. 5, "Virtual Reality" of "The Fabric of Reality," by David Deutsch, the Penguin Press (1997), ISBN 0-7139-9061-9.

C.P. Williams, *Explorations in Quantum Computing*,
Texts in Computer Science,
DOI 10.1007/978-1-84628-887-6_1, © Springer-Verlag London Limited 2011

fewer steps than *any* conventional ("classical") computer—including any supercomputer yet to be invented! This bold assertion is justified because the algorithms available to quantum computers can harness physical phenomena that are not available to classical computers no matter how sophisticated they may be. As a result, quantum computers can perform computations in fundamentally new ways that can, at best, only be mimicked inefficiently by classical computers. Thus, quantum computing represents a *qualitative* change in how computation is done, making it of a different character than all previous advances in computer science. In particular, quantum computers can perform truly unprecedented tasks such as teleporting information, breaking supposedly "unbreakable" codes, generating true random numbers, and communicating with messages that betray the presence of eavesdropping. Similar counterintuitive capabilities are being discovered, routinely, making quantum computing a very active and exciting field. While no one book can do justice to the myriad of discoveries that have been made so far, I hope to give you a fresh perspective on the capabilities of quantum computers, and to provide you with the tools necessary to make your own foray into this exciting field.

1.1 Trends in Computer Miniaturization

"*I like small gadgets, look at this tiny digital camera … where is it*?"
– Artur Ekert [17]

Computer technology has been driven to smaller and smaller scales because, ultimately, the limiting factor on the speed of microprocessors is the speed with which information can be moved around inside the device. By cramming the transistors closer together, and evolving to ever faster mechanisms for switching, one can speed up the rate of computation. But there is a price to pay. As transistors are packed closer together it becomes more challenging to remove the heat they dissipate. So at any given stage of technological development there has always been an optimal transistor density that trades off size for thermal management.

In 1965 Gordon Moore, a co-founder of Intel, noticed that the most economically favorable transistor densities in integrated circuits seemed to have been doubling roughly every 18 months. He predicted that this trend would continue well into the future. Indeed, as evidenced by Table 1.1, it has, and Moore's anticipated scaling became known as the more official sounding "Moore's Law". However, it is not a Law in the proper scientific sense as Nature does not *enforce* it. Rather, Moore's Law is merely an empirical observation of a scaling regularity in transistor size and power dissipation that industry had achieved, and Gordon Moore extrapolated into the future. However, there is uncertainty in the chip industry today regarding how much longer Moore's Law can be sustained.

Nevertheless, in the 40 years since Moore's Law was invented, successive generations of Intel chips have adhered to it surprisingly. This is all the more surprising when one realizes how just how much the underlying transistor technology has changed (see Fig. 1.1).

Table 1.1 Growth of the clock rate, and the number of transistors per chip in Intel processors from 1971 to 2007. Note that the transistor sizes reduced over the same time period, allowing the chips to remain about the same size. In the table 1 μ = 10^{-6} meter and 1 nm = 10^{-9} meter

Intel microprocessor	Year	Speed	# Transistors	Manufacturing scale
4004	1971	108 kHz	2,300	10 μ
8008	1972	500–800 kHz	3,500	10 μ
8080	1974	2 MHz	4,500	6 μ
8086	1978	5 MHz	29,000	3 μ
8088	1979	5 MHz	29,000	3 μ
286	1982	6 MHz	134,000	1.5 μ
386	1985	16 MHz	275,000	1.5 μ
486	1989	25 MHz	1,200,000	1 μ
Pentium	1993	66 MHz	3,100,000	0.8 μ
Pentium Pro	1995	200 MHz	5,500,000	0.6 μ
Pentium II	1997	300 MHz	7,500,000	0.25 μ
Pentium II Xeon	1997	300 MHz	7,500,000	0.25 μ
Pentium III	1999	500 MHz	9,500,000	0.18 μ
Pentium III Xeon	1999	500 MHz	9,500,000	0.18 μ
Pentium 4	2000	1.5 GHz	42,000,000	0.18 μ
Xeon	2001	1.5 GHz	42,000,000	0.18 μ
Pentium M	2002	1.7 GHz	55,000,000	90 nm
Itanium 2	2002	1 GHz	220,000,000	0.13 μ
Pentium D	2005	3.2 GHz	291,000,000	65 nm
Core 2 Duo	2006	2.93 GHz	291,000,000	65 nm
Core 2 Extreme	2006	2.93 GHz	291,000,000	65 nm
Dual-Core Xeon	2006	2.93 GHz	291,000,000	65 nm
Dual-Core Itanium 2	2006	1.66 GHz	1,720,000,000	90 nm
Quad-Core Xeon	2006	2.66 GHz	582,000,000	65 nm
Quad-Core Core 2 Extreme	2006	2.66 GHz	582,000,000	65 nm
Core 2 Quad	2007	2.66 GHz	582,000,000	65 nm
Quad-Core Xeon	2007	>3 GHz	820,000,000	45 nm
Dual-Core Xeon	2007	>3 GHz	820,000,000	45 nm
Quad-Core Core 2 Extreme	2007	>3 GHz	820,000,000	45 nm

Today, many industry insiders see Moore's Law surviving for just two or three more generations of microprocessors at best. In a valiant effort to sustain Moore's Law chip manufacturers are migrating to multi-core microprocessor architectures, and exotic new semiconductor materials. Beyond these advances, a switch to nanotechnology may be necessary.

Whatever strategy industry adopts to maintain Moore's Law it is clear that as time goes on fewer and fewer atoms will be used to implement more and more bits.

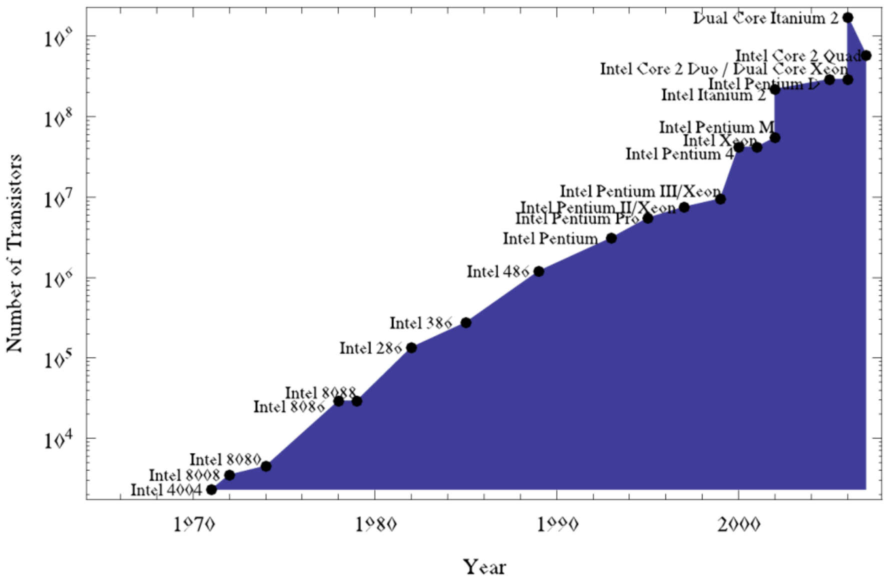


Fig. 1.1 Historical scaling in the numbers of transistors per chip in successive generations of Intel processors. The latest chips use multiple cores

Fig. 1.2 Historical scaling in the number of atoms needed to implement one bit

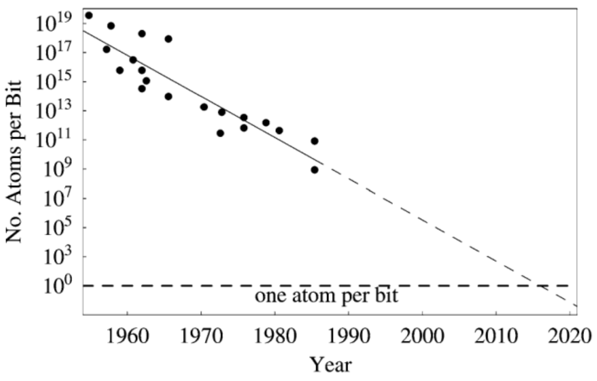


Figure 1.2 shows the scaling in the number of atoms needed to implement a bit as a function of time. Extrapolating this trend shows we will be at the one atom per bit level by about 2020. At the one-atom-per-bit level the appropriate physical model to describe what is going on is that of quantum physics rather than classical physics.

Quantum physics is considerably different from classical physics. Facts that we take as being "common sense" in our everyday (classical) world do not necessarily hold in the quantum realm. For example, in the classical world we are accustomed to thinking of particles (like grains of sand or dust) as having a definite location in space and time. But in the quantum world particles do have a definite location in space and time—in fact they can be in more than one place, or in more than one state, at the same time! More bizarre still, supposed "particles" can interact with one another more in a manner that is more reminiscent of waves than solid objects. Ultimately, as bits must be encoded in the states of physical systems, whether those systems are quantum or classical can therefore affect their properties profoundly.

1.2 Implicit Assumptions in the Theory of Computation

"*Nature isn't classical damn it*!"
– Richard Feynman

Bits, or "binary digits" lie at the heart of all modern digital equipment ranging from computers to iPODs to high-definition television (HDTV). Contemporary computers use voltage levels to encode bits. Old fashioned, mechanical, computers use the position of gear teeth. The only requirement is that the physical system must possess two clearly distinguishable configurations, or states, that are sufficiently stable so that they do not flip, spontaneously, from the state representing the bit 0 into the state representing the bit 1 or vice versa.

Once we have the ability to store 0s and 1s and to manipulate them in a controlled manner we have the basis for making all digital devices. By now, we are all so familiar with digital devices that, to the extent we even think about them at all, we take the properties of the bits within them for granted. For example, I am sure you will agree that the following operations on bits seem eminently reasonable: we can read a bit to learn the value it has; we can copy, erase or negate a bit regardless of whether it is a 0 or a 1; and we can read some of the bits inside a digital device without changing the other bits that we did not read. In fact such properties seem so obvious that we don't even bother to question these assumptions.

However, in his 1959 address "There's Plenty of Room at the Bottom" physicist Richard Feynman alluded to the tremendous opportunity available at the time for further miniaturization of technology [182]. He also anticipated that very small physical devices would be governed by quantum mechanics rather than classical mechanics and, as such, would not necessarily behave the same their larger counterparts. For example, a robot on the quantum scale might pick up and not pick up an object at the same time, and to carry it off left and right simultaneously. You would never know which was the case until you performed an observation as to what he robot had done. Once you did that, and made a permanent record of the result, its behavior would become definite. That sounds crazy, but that is what quantum mechanics tells us can happen.

Likewise, bits are going to be recorded, ultimately, in the state of some physical system. So as devices become miniaturized the sizes of the physical systems used to encode those bits will become smaller. At some point their behavior will need to be described by quantum physics rather than classical physics. At this point, our common sense assumptions about how bits ought to behave, e.g., that we can read, copy, erase, negate them without causing them to change in any way, cease to be

Table 1.2 Assumptions about the properties of bit that are no longer necessarily true at the quantum scale

Assumption	Classically	Quantum mechanically
A bit always has a definite value	True	False. A bit need not have a definite value until the moment after it is read
A bit can only be 0 or 1	True	False. A bit can be in a superposition of 0 and 1 simultaneously
A bit can be copied without affecting its value	True	False. A qubit in an unknown state cannot be copied without necessarily changing its quantum state
A bit can be read without affecting its value	True	False. Reading a qubit that is initially in a superposition will change the qubit
Reading one bit in the computer memory has no affect on any other (unread) bit in the memory	True	False. If the bit being read is entangled with another qubit, reading one qubit will affect the other
To compute the result of a computation, you must run the computer	True	False

valid. In fact, at the quantum scale you cannot necessarily read a bit without changing its value; you cannot necessarily copy, or negate it without perturbing it; you may be unable to erase it; and sometimes when you read one bit your actions can change the state of another bit with which you never interacted. Thus, bits encoded in quantum-scale objects cease to behave like normal bits ought. Some of the differences between normal (classical) and bits encoded at the quantum scale are shown in Table 1.2.

Thus, once computers becomes so small that we are then dealing with quantum bits as opposed to classical bits, we open up a new repertoire of physical effects that can be harnessed to achieve novel functionalities. As a result many new opportunities present themselves.

1.3 Quantization: From Bits to Qubits

Fortunately, quantum systems possess certain properties that lend themselves to encoding bits as physical states. When we measure the "spin" of an electron, for example, we always find it to have one of two possible values. One value, called "spin up" or $|\uparrow\rangle$, means that the spin was found to be parallel to the axis along which the measurement was taken. The other possibility, "spin-down" or $|\downarrow\rangle$, means that the spin was found to be anti-parallel to the axis along which the measurement was taken. This intrinsic discreteness, a manifestation of quantization, allows the spin of an electron to be considered as a natural binary digit or "bit".

Such intrinsic "discreteness" is not unique to spin-systems. Any 2-state quantum system, such as the plane of polarization of a linearly polarized photon, the direction of rotation of a circularly polarized photon, or the discrete energy levels in an excited atom, would work equally well. Whatever the exact physical embodiment chosen, if a quantum system is used to represent a bit, we call the resulting system a quantum bit, or just "qubit" for short.

1.3.1 Ket Vector Representation of a Qubit

As we are talking variously about (classical) bits and (their quantum counterparts) qubits, we'd better find a way of distinguishing them. To do so, we adopt a notation invented by British physicist extraordinaire Paul Dirac, which has since become known as "Dirac-notation".

In Dirac notation, when we are talking about a qubit (a quantum bit) in a physical state that represents the bit value 0, we'll write the qubit state using an angular-looking bracket, $|0\rangle$, which is called a "ket" vector. Likewise, a qubit in a physical state representing the bit value 1 will be written $|1\rangle$. What these notations mean *physically* will depend upon the nature of the system encoding them. For example, a $|0\rangle$ could refer to a polarized photon, or an excited state of an atom, or the direction of circulation of a superconducting current etc. The notation speaks only to the *computational* abstraction that we ascribe to a 2-state quantum system and doesn't give us any direct information about the underlying physical embodiment of the system encoding that qubit.

Mathematically, kets are a shorthand notation for column vectors, with $|0\rangle$ and $|1\rangle$ corresponding to:

$$|0\rangle \equiv \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \qquad |1\rangle \equiv \begin{pmatrix} 0 \\ 1 \end{pmatrix} \tag{1.1}$$

You might ask "Why do we need to represent a *single* quantum bit as a *two*-element column vector?" "Isn't one binary digit enough to specify it completely?" The answer lies in the fact that quantum bits are not constrained to be wholly 0 or wholly 1 at a given instant. In quantum physics if a quantum system can be found to be in one of a discrete set of states, which we'll write as $|0\rangle$ or $|1\rangle$, then whenever it is not being observed it may also exist in a *superposition*, or blend of those states simultaneously, $|\psi\rangle = a|0\rangle + b|1\rangle$ such that $|a|^2 + |b|^2 = 1$.

1.3.2 Superposition States of a Single Qubit

Thus, whereas at any instant a classical bit can be either a 0 *or* a 1, a qubit can be a superposition of both a $|0\rangle$ *and* a $|1\rangle$ simultaneously, i.e., a state such as:

$$|\psi\rangle = a|0\rangle + b|1\rangle \equiv \begin{pmatrix} a \\ b \end{pmatrix} \tag{1.2}$$

where a, and b are complex numbers[2] having the property $|a|^2 + |b|^2 = 1$.

The coefficient "a" is called the *amplitude* of the $|0\rangle$ component and the coefficient "b" is called the *amplitude* of the $|1\rangle$ component. The requirement that $|a|^2 + |b|^2 = 1$ is to ensure the qubit is properly *normalized*. Proper normalization guarantees that when we do finally read a qubit, it will be found, with probability $|a|^2$ to be in state $|0\rangle$ or, with probability $|b|^2$ to be in state $|1\rangle$ and nothing else. Thus the sums of the probabilities of the possible outcomes add up to one.

Dirac notation makes it easy to write down compact descriptions of quantum states and operators. Some common examples are as follows:

Dirac Notation: Bras, Kets, Inner and Outer Products For every "ket" $|\psi\rangle$ (which can be thought of as a shorthand notation for a column vector) there is a corresponding "bra" $\langle\psi|$ (which can be though of as shorthand for a row vector). The ket and the bra contain *equivalent information* about the quantum state in question. Mathematically, they are the dual of one another, i.e.:

$$\begin{aligned} |\psi\rangle &= a|0\rangle + b|1\rangle = \begin{pmatrix} a \\ b \end{pmatrix} \\ \langle\psi| &= a^*\langle 0| + b^*\langle 1| = (a^* \quad b^*) \end{aligned} \tag{1.3}$$

Note that the amplitudes in the bra space are the complex conjugates of the amplitudes in the ket space. That is, if $z = x + iy$ is a complex number with real part x and imaginary part y, then the complex conjugate of z is $z^* = x - iy$.

What is the purpose of introducing bra vectors into the discussion if they don't contain any new information about the quantum state? It turns out that *products* of bras and kets give us insight into the similarities between two quantum states. Specifically, for a pair of qubits in states $|\psi\rangle = a|0\rangle + b|1\rangle$ and $|\phi\rangle = c|0\rangle + d|1\rangle$ we can define their *inner product*, $\langle\psi|\phi\rangle$ as:

$$\langle\psi|\phi\rangle = \underbrace{(\langle\psi|) \cdot (|\phi\rangle)}_{\text{bra (c) ket}} = (a^* \quad b^*) \cdot \begin{pmatrix} c \\ d \end{pmatrix} = a^*c + b^*d \tag{1.4}$$

The inner product $\langle\psi|\phi\rangle$ is also called the overlap between (normalized) states $|\psi\rangle$ and $|\phi\rangle$ because it varies from zero for orthogonal states to one for identical normalized states. We can verify this with a direct calculation: $\langle\psi|\psi\rangle = (a^* \quad b^*) \cdot \binom{a}{b} = a^*a + b^*b = |a|^2 + |b|^2 = 1$.

A second product we can define on states $|\psi\rangle = a|0\rangle + b|1\rangle$ and $|\phi\rangle = c|0\rangle + d|1\rangle$ is their outer product $|\psi\rangle\langle\phi|$:

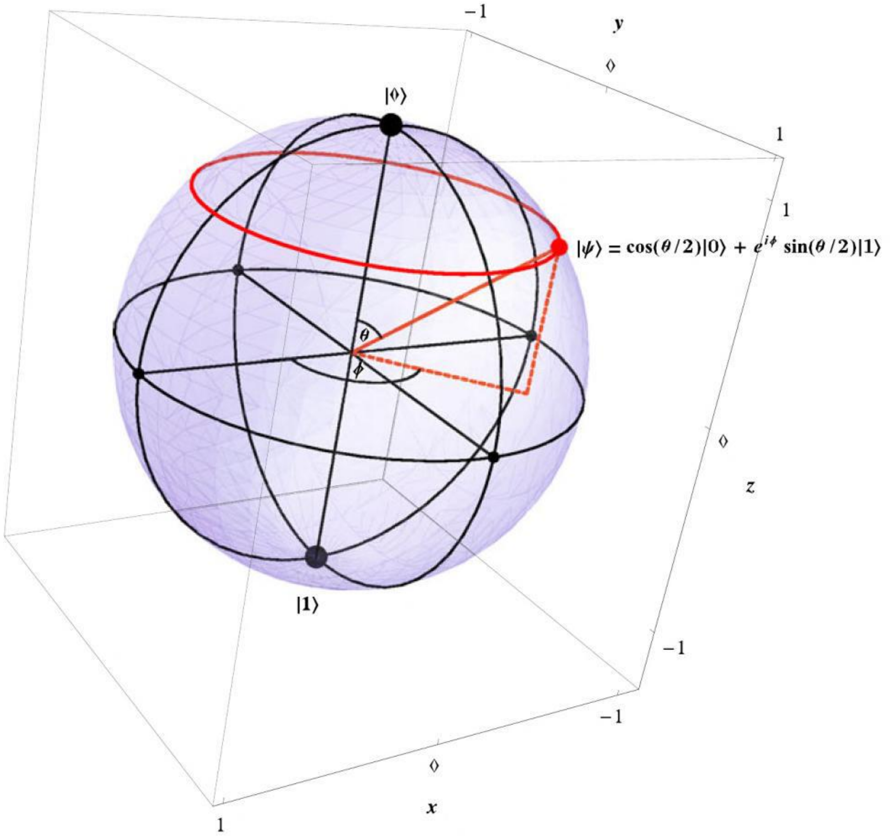
$$|\psi\rangle\langle\phi| = (|\psi\rangle) \cdot (\langle\phi|) = \begin{pmatrix} a \\ b \end{pmatrix} \cdot (c^* \quad d^*) = \begin{pmatrix} ac^* & ad^* \\ bc^* & bd^* \end{pmatrix} \tag{1.5}$$

[2] A complex number $z = x + iy$ is a composite number consisting of two real numbers x and y, and a constant $i = \sqrt{-1}$. $x = \mathrm{Re}(z)$ is called the "real" part of z, and $y = \mathrm{Im}(z)$ is called the "imaginary" part of z. $z^* = x - iy$ denotes the complex *conjugate* of z, and $|z| = \sqrt{x^2 + y^2}$ denotes the *modulus* of z.

which is a matrix. The outer product provides a very nice way of describing the structure of unitary operators, which as will see later, correspond to quantum logic gates. For example, a NOT gate has a corresponding unitary matrix NOT $= \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$. In terms of outer products this can also be written as NOT $= |0\rangle\langle 1| + |1\rangle\langle 0|$. The outer product factorization of the NOT gate shows the transformation it performs explicitly. Indeed, all quantum gates can be best understood as a sum of such outer products.

1.3.3 Bloch Sphere Picture of a Qubit

An intuitive, albeit approximate, way to visualize the quantum state of a single qubit is to picture it as a unit vector inside a bounding sphere, called the Bloch sphere (see Fig. 1.3). The parameters defining the quantum state are related to the azimuth and



-1
y
0
1
$|0\rangle$
1
$|\psi\rangle = \cos(\theta/2)|0\rangle + e^{i\phi}\sin(\theta/2)|1\rangle$
θ
ϕ
0
z
$|1\rangle$
-1
-1
0
x
1

Fig. 1.3 Bloch sphere showing the computational basis states $|0\rangle$ and $|1\rangle$, and a general qubit state $|\psi\rangle = \cos(\theta/2)|0\rangle + e^{i\phi}\sin(\theta/2)|1\rangle$

elevation angles that determine where the tip of this vector touches the surface of the Bloch sphere. In this picture, the North pole corresponds to the pure state $|0\rangle$ and the South pole corresponds to the (orthogonal) pure state $|1\rangle$. All other points on the surface of the Bloch sphere correspond to the superposition states of the form $a|0\rangle + b|1\rangle$ for all possible values of the complex numbers a and b such that $|a|^2 + |b|^2 = 1$.

In particular, an arbitrary pure state of a single qubit $|\psi\rangle = a|0\rangle + b|1\rangle$ such that $|a|^2 + |b|^2 = 1$ can be written in terms of these azimuth and elevation angles as:

$$|\psi\rangle = e^{i\gamma}\left(\cos\frac{\theta}{2}|0\rangle + e^{i\phi}\sin\frac{\theta}{2}|1\rangle\right) \tag{1.6}$$

where γ, θ, and ϕ are all real numbers. A pair of elevation and azimuth angles (θ, ϕ) in the range $0 \le \theta \le \pi$ and $0 \le \phi \le 2\pi$ pick out a point on the Bloch sphere. Qubit states corresponding to different values of γ are indistinguishable and are all represented by the same point on the Bloch sphere. γ is said to be an overall phase factor that is unobservable.

Students are often confused about the Bloch sphere for three main reasons: first how come the azimuth and elevation angles are expressed in half-angles? Second, how come orthogonal states are not at right angles on the Bloch sphere? Instead they are 180° apart. Third how can it be that the γ parameter has no observable effect?

How might we draw a picture that captures in an intuitive way the complete character of a qubit in a superposition state such as $a|0\rangle + b|1\rangle$? The Bloch sphere provides a way of *visualizing* the quantum mechanical state of a single qubit. “Wait a minute!” you say. “Aren’t orthogonal states supposed to be at right angles? How can the $|0\rangle$ state be the North pole and the $|1\rangle$ be the South Pole? They’re 180° apart!”

Students are often confused by the Bloch sphere representation of a quantum state because orthogonal states are not found to be at right angles on the Bloch sphere. So it is worth a little detour to explain how the Bloch sphere is constructed.

Consider the general quantum state $a|0\rangle + b|1\rangle$. Since a and b are complex numbers they can be written in either Cartesian or Polar coordinates as: $a = x_a + iy_a = r_a e^{i\phi_a}$ and $a = x_b + iy_b = r_b e^{i\phi_b}$ with $i = \sqrt{-1}$ and the x’s, y’s, r’s, and ϕ’s are all real numbers. So, naively, it looks like we need to depict four real numbers x_a, x_b, y_a, y_b or r_a, r_b, ϕ_a, ϕ_b depending on whether we use the Cartesian or polar representation of the complex numbers a and b. Not so!

Write the general state of a qubit $a|0\rangle + b|1\rangle$ as $r_a e^{i\phi_a}|0\rangle + r_b e^{i\phi_b}|1\rangle$. Since an overall phase factor has no observable consequence (you’ll prove this as an exercise later), we can multiply by any global phase we please to obtain an equivalent state. In particular, we could multiply by the phase factor $e^{-i\phi_a}$ to obtain $r_a|0\rangle + r_b e^{i(\phi_b - \phi_a)}|1\rangle$. This allows us to represent the state of the qubit using *three* real numbers r_a, r_b and $\phi = (\phi_b - \phi_a)$. Switching back to Cartesian coordinates for the amplitude of the $|1\rangle$ component we can write this state as $r_a|0\rangle + (x + iy)|1\rangle$. Applying normalization we have $|r_a|^2 + |x + iy|^2 = 1$ or equivalently $r_a^2 + x^2 + y^2 = 1$ which is the equation of a *sphere* in coordinates r_a, x, and y. We can rename $r_a = z$ for aesthetic reasons and it doesn’t change anything but now we have the equation of a sphere in coordinates x, y, and z. Ok so let's switch from these Cartesian coordinates to spherical coordinates. We have,

$$x = r\sin(\theta)\cos(\phi) \tag{1.7}$$

$$y = r\sin(\theta)\sin(\phi) \tag{1.8}$$

$$z = r\cos(\theta) \tag{1.9}$$

But given the constraint $x^2 + y^2 + z^2 = r^2 = 1$, we see $r = 1$. So now the position on the surface of the sphere is specified using only *two* parameters, θ and ϕ. And the general qubit state can be written as $z|0\rangle + (x + iy)|1\rangle$ or equivalently, since $r = 1$, $\cos(\theta)|0\rangle + (\sin(\theta)\cos(\phi) + i\sin(\theta)\sin(\phi))|1\rangle$, or equivalently $\cos(\theta)|0\rangle + e^{i\phi}\sin(\theta)|1\rangle$ since $\cos(\phi) + i\sin(\phi) = e^{i\phi}$. Given that a qubit must lie between the extremes of being wholly $|0\rangle$ (which occurs when $\theta = 0$ and wholly $|1\rangle$ (which occurs when $\theta = 90$ it appears all the qubit states are mapped out over just a hemispherical region of the sphere defined by $x^2 + y^2 + z^2 = 1$. If we want all the possible qubit states to correspond to the points on the surface of a whole sphere, we can simply map this hemisphere or points onto a sphere of points by introducing a new angle $\theta' = 2\theta$. Thus the general qubit state can now be written as $\cos(\frac{\theta'}{2})|0\rangle + e^{i\phi}\sin(\frac{\theta'}{2})|1\rangle$. Thus the complete set of qubit states is now mapped out as θ' runs from $0°$ to $180°$. This final sphere is the Bloch sphere.

An immediate consequence of how the Bloch sphere is constructed is that orthogonal quantum states, i.e., states $|\psi\rangle$ and $|\chi\rangle$ for which $\langle\psi|\chi\rangle = 0$, are represented by antipodal points on the Bloch sphere (rather than being drawn at right angles which is how we usually expect to see orthogonal vectors drawn in 3D space). This is the reason why $|0\rangle$ lies at the North Pole and $|1\rangle$ lies at the South Pole of the Bloch sphere. For a general pure state, represented as a point on the surface of the Bloch sphere, the antipodal state is the one diametrically opposite it on the other side of the Bloch sphere such that a straight line drawn between the original state and its antipodal state would pass through the center of the Bloch sphere. The operation that maps an unknown state to its antipodal state cannot be expressed as a rotation on the Bloch sphere. Rather it is the sum of a rotation (in longitude through 180 degrees) and a reflection (in latitude with respect to the equatorial plane of the Bloch sphere). This inability to express the operation purely as a rotation will turn out to impact our ability to achieve it in a sequence of unitary quantum gates.

Figure 1.4 shows the Bloch sphere labeled with pure 1-qubit states at the extremes of the x-, y-, and z-axes. These are, respectively, $|\nearrow\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$, $|\nwarrow\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$, $|R\rangle = |\circlearrowright\rangle = \frac{1}{\sqrt{2}}(|0\rangle + i|1\rangle)$, $|L\rangle = |\circlearrowleft\rangle = \frac{1}{\sqrt{2}}(|0\rangle - i|1\rangle)$, $|0\rangle$, and $|1\rangle$. Notice that orthogonal states are indeed located at antipodal points on the surface of the Bloch sphere.

1.3.3.1 Other Rotations Having Period 4π

When first encountering the Bloch sphere, students often find it hard to grasp why a rotation of 2π radians (i.e., $360°$) would not restore an object back to its original

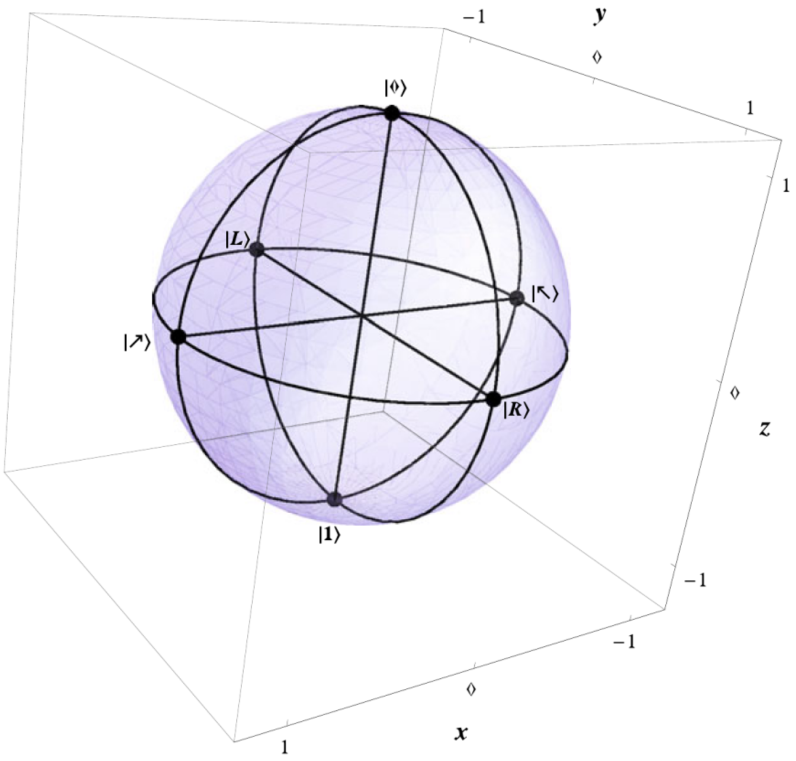


Fig. 1.4 Bloch sphere representation of the states $|0\rangle$, $|1\rangle$, $|\nearrow\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$, $|\nwarrow\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$, $|R\rangle = |\circlearrowright\rangle = \frac{1}{\sqrt{2}}(|0\rangle + i|1\rangle)$, and $|L\rangle = |\circlearrowleft\rangle = \frac{1}{\sqrt{2}}(|0\rangle - i|1\rangle)$. Orthogonal pure states are at antipodal points on the surface of the Bloch sphere

configuration. However, such a phenomenon can also be seen in the motions of certain classical physical systems.

For example, extend your right hand straight out so your palm is face up. *Keeping your palm face up at all times*, rotate your hand clockwise around a vertical axis passing through the center of your palm until your hand returns to its original configuration. The basic contortions you need to do are as follows: starting with your right hand extended straight out palm up, pull your arm inwards (keeping your palm flat), twisting your wrist to the right and pushing your elbow to the left, continue twisting your palm clockwise so your fingertips are pointing towards your right shoulder, and swing your elbow around to the right and upwards, and push your arm out again. Congratulations! Your palm has now been rotated through 2π radians (360°) and it is indeed still face up, *but* your hand is not in its original configuration because your elbow is now on top! To return your hand to its original configuration you need to apply another full rotation of 360° to your palm. To do so, continue turning your wrist to the right (still keeping your palm face up) so that your fingertips point towards your tight armpit, swing you elbow around and downwards in a clockwise rotating arc, whilst twisting your wrist to the right. This will take your arm back to its starting configuration. Thus, your hand requires a total rotation of 4π radians (720°) to return it to its starting configuration. Resting a plate on your palm as you do this ensures you keep your palm face up for fear of dropping the plate. This is sometimes known as "Feynman's Plate Trick".

A more surprising demonstration of the same symmetry property occurs in the rotations of a flat belt that is fixed at one end and rotated at the other. This version if called "Dirac's Belt Trick" and it is always a hit at parties. Take off a long flat belt strap. Have a friend hold the buckle end of the belt and hold the other end yourself. Pull the belt taut so it is flat with the outer face of the belt (as it is normally worn) pointing upwards. Tell your friend to keep hold their end of the belt tightly in a fixed position. Ok now twist (i.e., rotate) your end of the belt through 2π radians (i.e., 360°). Can you remove the kink you have imparted to the belt by passing the belt under and over itself *while keeping the orientation of the ends of the belt fixed* (i.e., flat with the outer face of the belt pointing upwards)? After a little experimentation you will conclude you cannot.

Let us make the problem even harder by applying an *additional* twist to your end of the belt through another 2π radians (i.e., another 360°). Can you remove the *double* kink by passing the belt under and over itself while keeping both ends flat and pointed upwards? Surely if you could not remove *one* kink in this manner, you would expect it would be even harder to remove two! Yet, remarkably, you can! After a rotation of 4π radians (720°) applied to the end of the belt, the belt can be restored to its original configuration by passing it under and over itself while keeping the orientations of the two ends fixed in space! This seems to be more surprising to most people than the plate trick. Yet both are examples of physical systems in which rotations of 2π radians do not restore an object to its original state whereas rotations of 4π radians do! Such examples show that the 4π periodicity of the Bloch sphere has parallels in the classical world around us.

1.3.4 Reading the Bit Value of a Qubit

In the everyday classical world when we read, or measure, or observe, something we don't usually perturb it in the process. For example, when we read a newspaper we don't change the words on the page merely by reading them. Moreover, if ten people read ten different copies of the same edition of the same paper they would all see the same words. However, in the quantum world this is not what happens.

The states $|0\rangle$ and $|1\rangle$ correspond to the North and South poles of the Bloch sphere respectively, and the axis passing through these points is the z-axis (see Fig. 1.5). Thus the act of reading the bit value of a qubit amounts to determining the alignment of its spin with respect to this z-axis. If the particle is aligned "spin-up" it is in the state $|0\rangle$. If it is aligned "spin-down" it is in the state $|1\rangle$.

When a single qubit in state $a|0\rangle + b|1\rangle$ is read (or "measured" or "observed"), with respect to some axis through the center of the Bloch sphere, the probability of

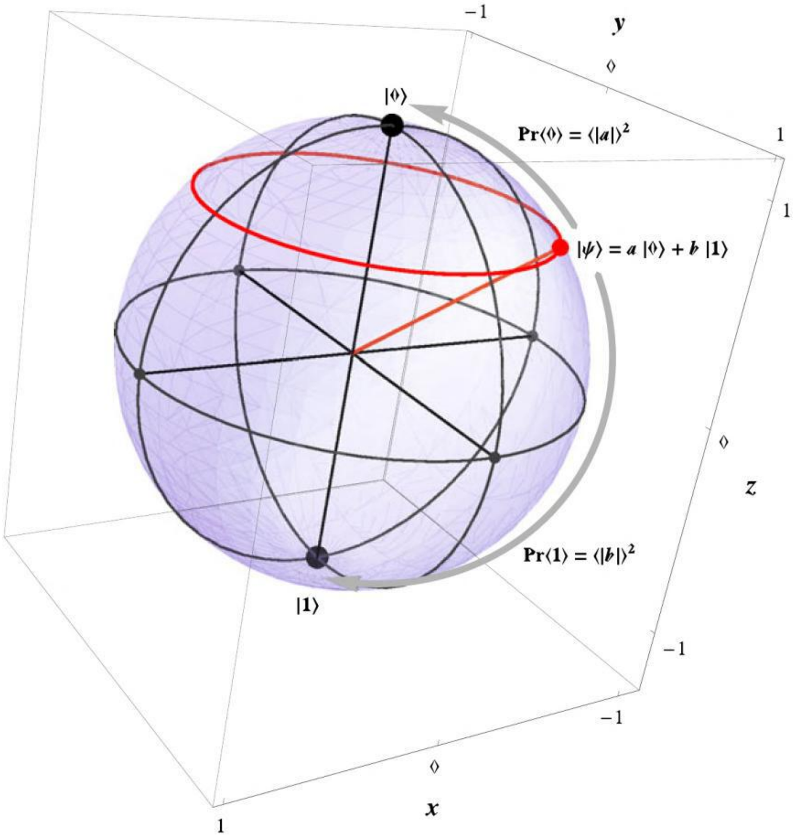


Fig. 1.5 Measuring the bit value of a qubit initially in state $a|0\rangle + b|1\rangle$ yields the answer 0 with probability $|a|^2$ or 1 with probability $|b|^2$, and projects the qubit into either state $|0\rangle$ or state $|1\rangle$ respectively

finding it in state $|0\rangle$ or state $|1\rangle$ depends upon the values of a and b, and on the orientation of this axis. The most commonly used axis is that passing through the North and South poles corresponding to the states $|0\rangle$ and $|1\rangle$. A measurement of a qubit with respect to this axis is called a measurement "in the computational basis" because the answer we get will be one of the bit values $|0\rangle$ or $|1\rangle$. The outcome we obtain is, in general, not certain but depends on the amplitudes a and b. Specifically, measuring the bit value of $a|0\rangle + b|1\rangle$ in the computational basis will yield the answer $|0\rangle$ with probability $|a|^2$ and the answer $|1\rangle$ with probability $|b|^2$. These two probabilities sum to 1, i.e., $|a|^2 + |b|^2 = 1$.

$$\text{Read}(a|0\rangle + b|1\rangle) = \begin{cases} 0 & \text{with probability } |a|^2 \\ 1 & \text{with probability } |b|^2 \end{cases} \tag{1.10}$$

Thus, a single qubit quantum memory register exhibits the interesting property that even though its contents may be definite, i.e., it may be precisely in the state

$|\psi\rangle = a|0\rangle + b|1\rangle$, the outcome we obtain from reading it is non-deterministic. Sometimes we will find it in state $|0\rangle$ and sometimes we will find it in state $|1\rangle$. However, the instant *after* the measurement is made, the state is known with certainty to be $|0\rangle$ or $|1\rangle$ consistent with result we obtained. Moreover, if we rapidly and repeatedly keep measuring the same state we can suppress its evolution and effectively freeze it in a fixed quantum state $|\psi\rangle \overset{\text{read}}{\to} |0\rangle \overset{\text{read}}{\to} |0\rangle \overset{\text{read}}{\to} |0\rangle \cdots$ or $|\psi\rangle \overset{\text{read}}{\to} |1\rangle \overset{\text{read}}{\to} |1\rangle \overset{\text{read}}{\to} |1\rangle \cdots$. This is a variant of the so-called Quantum Zeno Effect.[3] But if we allow time to elapse between measurements the state will, in general, evolve, or "drift off", in accordance with Schrödinger's equation.

1.4 Multi-qubit Quantum Memory Registers

So far we have only been dealing with single qubits, but a useful quantum computational device will need to have a multi-qubit quantum memory register. In general, this is assumed to consist of a collection of n-qubits, which are assumed to be ordered, indexed and addressable so that selective operations can be applied to any single qubit or any pair of qubits at will. If two qubits selected for an operation are not physically adjacent, there is usually an operational sequence that achieves the interaction between them as if they were. This detail is typically omitted from the abstract model of the quantum memory as it is more an implementation issue than anything fundamental to the computational model.

Just as a single qubit can be found in a superposition of the possible bit values it may assume, i.e., $|0\rangle$ and $|1\rangle$, so too can a n-qubit register be found in a superposition of all the 2^n possible bit strings $|00\dots0\rangle, |00\dots1\rangle, \dots, |11\dots1\rangle$ it may assume. However, the most interesting superposition states typically involve non-uniform contributions of eigenstates.

1.4.1 The Computational Basis

When we describe the state of a multi-qubit quantum memory register as a superposition of its possible bit-string configurations, we say the state is represented in the *computational basis*. This is arguably the most natural basis for quantum computing. For example, the most general form for a pure state of a 2-qubit quantum memory register can be written as:

$$|\psi\rangle = c_0|00\rangle + c_1|01\rangle + c_2|10\rangle + c_3|11\rangle \tag{1.11}$$

[3]The Quantum Zeno Effect says that if you repeatedly measure (or observe) a quantum system, you can suppress its quantum mechanical evolution. It is named after Zeno of Elea who devised a paradox that aimed to prove if you continually observe an arrow in flight at any instant it would appear motionless and hence it cannot be moving: "If everything when it occupies an equal space is at rest, and if that which is in locomotion is always occupying such a space at any moment, the flying arrow is therefore motionless."—Aristotle, Physics VI:9, 239b5.

where $|c_0|^2 + |c_1|^2 + |c_2|^2 + |c_3|^2 = 1$. This implies we can think of the register as containing many different bit string configurations at once, each with their own amplitude. Similarly, the general state of a 3-qubit register can be written as:

$$
\begin{aligned}
|\psi\rangle = c_0|000\rangle + c_1|001\rangle + c_2|010\rangle + c_3|011\rangle + c_4|100\rangle + c_5|101\rangle \\
+ c_6|110\rangle + c_7|111\rangle
\end{aligned}
\tag{1.12}
$$

where $|c_0|^2 + |c_1|^2 + |c_2|^2 + |c_3|^2 + |c_4|^2 + |c_5|^2 + |c_6|^2 + |c_7|^2 = 1$. Continuing in this fashion, we see that the most general form for a pure state of an n-qubit quantum memory register is:

$$
|\psi\rangle = c_0|00\ldots0\rangle + c_1|00\ldots1\rangle + \cdots + c_{2^n-1}|11\ldots1\rangle = \sum_{i=0}^{2^n-1} c_i|i\rangle
$$

where $\sum_{i=0}^{2^n-1} |c_i|^2 = 1$ and $|i\rangle$ represents the "computational basis eigenstate" whose bit values match those of the decimal number i expressed in base-2 notation, padded on the left (if necessary) with "0" bits in order to make a full complement of n bits. For example, the 5-qubit computational basis eigenstate corresponding to $|6\rangle$ is $|00110\rangle$. This is because 6 in base-2 is "110" and then we pad on the left with two "0" bits to make a total of 5 bits.

As for the case of single qubits, such ket vectors can always be regarded as a short hand notation for a column vector. The size of these column vectors grow exponentially with the number of qubits, making it computationally intractable to simulate arbitrary quantum computations on classical computers. For example, a 100-qubit quantum memory register requires 2^100 complex amplitudes to specify it completely! In very few qubits, we run out of particle in the known Universe with which to make a classical memory large enough to represent a quantum state.

In a multi-qubit quantum state it is not necessary (and for often not desirable) for every amplitude to be non-zero. For example, if the quantum memory register contains the output from some quantum computation, typically, many of the eigenstates (corresponding) to non-solutions will be absent. For example, a particular 3-qubit quantum state, $|\psi\rangle = a|001\rangle + b|010\rangle + c|100\rangle$ does not contain any contributions from the eigenstates $|000\rangle$, $|011\rangle$, $|101\rangle$, $|110\rangle$, $|111\rangle$. The amplitude of these omitted components is zero by implication. Hence, as a column vector, the aforementioned 3-qubit state would actually be:

$$
|\psi\rangle = a|001\rangle + b|010\rangle + c|100\rangle \equiv \begin{pmatrix} 0 \\ a \\ b \\ 0 \\ c \\ 0 \\ 0 \\ 0 \end{pmatrix} \equiv \begin{pmatrix} \text{amplitude of } |000\rangle \text{ component} \\ \text{''} \quad |001\rangle \quad \text{''} \\ \text{''} \quad |010\rangle \quad \text{''} \\ \text{''} \quad |011\rangle \quad \text{''} \\ \text{''} \quad |100\rangle \quad \text{''} \\ \text{''} \quad |101\rangle \quad \text{''} \\ \text{''} \quad |110\rangle \quad \text{''} \\ \text{''} \quad |111\rangle \quad \text{''} \end{pmatrix}
\tag{1.13}
$$

1.4.2 Direct Product for Forming Multi-qubit States

Suppose we create a quantum memory register from a set of n independent qubits. How the state of the n-qubit register is related to the states of the individual qubits? The answer is provided by way of the *direct product* of the n individual quantum states.

Definition *Direct Product of Quantum States* of qubit states. Let $|\phi\rangle = \sum_{j=0}^{2^m-1} a_j|j\rangle$ be an m-qubit pure state, and $|\psi\rangle = \sum_{k=0}^{2^n-1} b_k|k\rangle$ be an n-qubit pure state. The quantum state of a memory register formed by considering $|\phi\rangle$ and $|\psi\rangle$ together is computed by taking their direct product, $|\phi\rangle \otimes |\psi\rangle$ (sometimes called "tensor" or "Kroenecker" product too):

$$|\phi\rangle \otimes |\psi\rangle = \sum_{j=0}^{2^m-1} a_j|j\rangle \otimes \sum_{k=0}^{2^n-1} b_k|k\rangle = \begin{pmatrix} a_0 \\ a_1 \\ \vdots \\ a_{2^m-1} \end{pmatrix} \otimes \begin{pmatrix} b_0 \\ b_1 \\ \vdots \\ b_{2^n-1} \end{pmatrix}$$

$$= \begin{pmatrix} a_0 \begin{pmatrix} b_0 \\ b_1 \\ \vdots \\ b_{2^n-1} \end{pmatrix} \\ a_1 \begin{pmatrix} b_0 \\ b_1 \\ \vdots \\ b_{2^n-1} \end{pmatrix} \\ \vdots \\ a_{2^m-1} \begin{pmatrix} b_0 \\ b_1 \\ \vdots \\ b_{2^n-1} \end{pmatrix} \end{pmatrix} = \begin{pmatrix} a_0 b_0 \\ a_0 b_1 \\ \vdots \\ a_0 b_{2^n-1} \\ \vdots \\ a_1 b_0 \\ a_1 b_1 \\ \vdots \\ a_1 b_{2^n-1} \\ \vdots \\ a_{2^m-1} b_0 \\ a_{2^m-1} b_1 \\ \vdots \\ a_{2^m-1} b_{2^n-1} \end{pmatrix} \tag{1.14}$$

For example, let $|\phi\rangle = a|0\rangle + b|1\rangle$ and $|\psi\rangle = c|0\rangle + d|1\rangle$. Then the direct product

$$|\phi\rangle \otimes |\psi\rangle = \begin{pmatrix} a \\ b \end{pmatrix} \otimes \begin{pmatrix} c \\ d \end{pmatrix} = \begin{pmatrix} a \begin{pmatrix} c \\ d \end{pmatrix} \\ b \begin{pmatrix} c \\ d \end{pmatrix} \end{pmatrix}$$

$$= \begin{pmatrix} ac \\ ad \\ bc \\ bd \end{pmatrix} = ac|00\rangle + ad|01\rangle + bc|10\rangle + bd|11\rangle \tag{1.15}$$

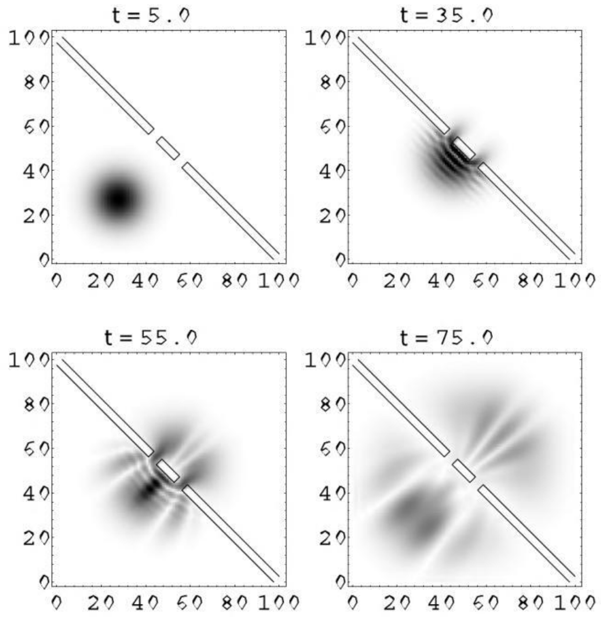


Fig. 1.6 A particle impinging on a double slit seen at four different times $t = 5.0$, 35.0, 55.0 and 75.0. Notice the interference pattern beyond the double slit (upper right quadrant of lower right frame). This, and several other stunning animations of quantum mechanical interference effects, can be found in [420]

1.4.3 Interference Effects

One of the most striking differences between quantum memory registers and classical memory registers is the possibility of encountering "quantum interference" effects in the quantum case that are absent in the classical case. In general terms, quantum interference can occur whenever there is more than one way to obtain a particular computational result. The different pathways can interfere constructively to increase the net probability of that result, or they can interfere destructively to reduce the probability of that result. For example, if a quantum mechanical particle impinges on a double slit it will, as shown in Fig 1.6, pass through both slits and self-interfere beyond the slit, resulting in an oscillatory pattern of probability amplitude for where the particle will be found. To understand this quantitatively, let's consider the probability of obtaining a particular computational result first by pretending that our quantum register behaves like a classical probabilistic register and then by treating it (correctly) as a true quantum memory register.

Let $|j\rangle$ and $|k\rangle$ be two eigenstates of an n-qubit quantum memory register that hold two different bit strings corresponding to integers j and k respectively. These states are orthogonal ($\langle j|k\rangle = 0$) and normalized ($\langle j|j\rangle = \langle k|k\rangle = 1$). So long as it is not being observed, it is possible for the quantum memory register to exist in a superposition of any of its allowed eigenstates such as a superposition of $|j\rangle$ and $|k\rangle$, i.e., $|\psi\rangle = c_j|j\rangle + c_k|k\rangle$. If we observed this state in the computational basis we would find it in state $|j\rangle$ with probability $|c_j|^2$ and in state $|k\rangle$ with probability $|c_k|^2 = 1 - |c_j|^2$ (since these are the only two possibilities).

Thus, on the face of it, one might think that the quantum memory register holding the state $|\psi\rangle = c_j|j\rangle + c_k|k\rangle$ behaves just the same as if it were a classical probabilistic memory register that outputs state $|j\rangle$ with probability p_j $(= |c_j|^2)$ and state $|k\rangle$ with probability p_k $(= |c_k|^2)$. But as we now show, this is not the case.

Specifically, let A be some observable that can act on an n-qubit register. Suppose one of the eigenvalues of this observable is "a" when the corresponding state of the memory register is $|\psi_a\rangle$. In other words we have $A|\psi_a\rangle = a|\psi_a\rangle$.

The question is, with what probability would be obtain the value "a" when we measure the observable A when the quantum memory register is in state $|\psi\rangle = c_j|j\rangle + c_k|k\rangle$?

Well in the (erroneous) "classical" view, the register *really* holds either state $|j\rangle$ or state $|k\rangle$ but we are ignorant about which is the case. The probability of getting "a" if the register is in state $|j\rangle$ is $P_j(a) = |\langle\psi_a|j\rangle|^2$. Similarly, the probability of getting "a" if the register is in state $|k\rangle$ is $P_k(a) = |\langle\psi_a|k\rangle|^2$. As we are ignorant about whether the register really holds state $|j\rangle$ or state $|k\rangle$ the probability with which we expect to see "a" is:

$$\begin{aligned} P^{\text{CLASSICAL}}(a) &= P_j(a)p_j + P_k(a)p_k = |c_j|^2 P_j(a) + |c_k|^2 P_k(a) \\ &= |c_j|^2|\langle\psi_a|j\rangle|^2 + |c_k|^2|\langle\psi_a|k\rangle|^2 \end{aligned} \tag{1.16}$$

So this is our prediction for the probability with which we see result "a" if our memory register behaves "classically".

In the case of the "quantum" interpretation of the register, however, we're not ignorant of anything! The register *truly* exists in the superposition state $|\psi\rangle = c_j|j\rangle + c_k|k\rangle$, and the probability of getting "a" is therefore:

$$\begin{aligned} P^{\text{QUANTUM}}(a) &= |\langle\psi_a|\psi\rangle|^2 = |c_j\langle\psi_a|j\rangle + c_k\langle\psi_a|k\rangle|^2 \\ &= |c_j|^2|\langle\psi_a|j\rangle|^2 + |c_k|^2|\langle\psi_a|k\rangle|^2 + 2\,\text{Re}(c_j c_k^*\langle\psi_a|j\rangle\langle\psi_a|k\rangle^*) \end{aligned} \tag{1.17}$$

Thus, in the quantum case there is an addition term contributing to the probability of obtaining result "a". This is the result of quantum interference between the different computational pathways by which result "a" can be obtained.

1.4.4 Entanglement

"*I would not call* [*entanglement*] *one but rather the characteristic trait of quantum mechanics, the one that enforces its entire departure from classical lines of thought.*"
– Erwin Schrödinger

Another way in which quantum memory registers can differ from classical memory registers is in their ability to exist in *entangled* states. This is a state of a composite quantum system that involves unusually strong correlations between parts of the system. There is considerable debate at present about the nature of entanglement, especially in systems involving more than two particles, and whether entanglement is strictly necessary to obtain a complexity advantage over a classical computer. However, at this time it appears that entanglement is *crucial* to obtaining the exponential speedups seen in some quantum algorithms.

So what is an entangled state exactly? In its simplest terms we can define an entangled state as follows:

Definition: Entangled Pure State A multi-qubit pure state is *entangled* if and only if it cannot be factored into the direct product of a definite state for each qubit individually. Thus, a pair of qubits, A and B, are entangled if and only if their joint state $|\psi\rangle_{AB}$ cannot be written as the product of a state for qubit A and a state for qubit B, i.e., if and only if $|\psi\rangle_{AB} \neq |\psi\rangle_A \otimes |\psi\rangle_B$ for any choice of states $|\psi\rangle_A$ and $|\psi\rangle_B$.

In a multi-qubit memory register if qubits are entangled then actions performed on one subset of qubits can have an impact on another, "untouched", subset of qubits. For example, consider a 2-qubit memory register comprised of qubits A and B, in state $\frac{1}{\sqrt{2}}(|0\rangle_A \otimes |0\rangle_B + |1\rangle_A \otimes |1\rangle_B)$. If qubit A is measured in the computational basis and found to be in state $|1\rangle$ then even though qubit B has not yet been touched, its quantum state is now determined to be $|1\rangle$ too. Thus a measurement of qubit A has had a side effect on the value of qubit B!

For notational compactness entangled state are more commonly written by dropping the particle label (A, B, etc.) because this is implied by position, and by dropping the $\otimes$ product as this is implied by simply abutting ket vectors. So the aforementioned entangled state could also be written as $\frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$

Entanglement is a pervasive phenomenon in multi-qubit quantum memory registers, It is also the cornerstone of many quantum algorithms. For example, we can prepare two entangled quantum registers, A and B say, such that register A contains a set of indices running from 0 to $2^n - 1$ and register B contains a set of values of a function who behavior depends upon the value of the index in register A. So the joint state (ignoring the normalization factor) can be something like $\sum_{i=0}^{2^n-1} |i\rangle_A |f(i)\rangle_B$. By measuring the value of the function (in register B) to be value "c" say, we can project out the set of indices (in register A) consistent with the observed function value, giving us a superposition of the form $\sum_{\{i':f(i')=c\}} |i'\rangle_A |c\rangle$. That's a neat trick because in one shot we get all the index values (in register A) that give the same value for the function (in register B).

1.4.4.1 Entanglement and Quantum States in Different Number Bases

One of the most interesting aspects of entanglement is how it is tied to our choice of representation of numbers. Traditionally, we think of quantum computing using the base-2 number system. Showing the number base as a subscript we have $|0_{10}\rangle = |0_2\rangle, |1_{10}\rangle = |1_2\rangle, |2_{10}\rangle = |10_2\rangle, |3_{10}\rangle = |11_2\rangle, \ldots$.

If the quantum gate, represented by the unitary matrix U, is to act on n qu*bits*, U will have dimensions of that are a power of two, specifically, $2^n \times 2^n$. Likewise, the unitary matrix corresponding to a quantum gate that acts on qu*trits* (i.e., base 3 quantum computation), will have dimensions that are a power of three, i.e., $3^n \times 3^n$. Typically, most researchers use a base 2 (qubit) model of quantum computation. This is partly out of habit, and partly because quantum gates that manipulate qubits (and which therefore require 2-body interactions) are assumed to be simpler to build than those that manipulate qutrits (and which therefore require 3-body interactions). But in principle, one could use whatever base one wants.

Does the choice of base matter? Well, not from a computability perspective. Any computation that can be done using qubits can also be done using qutrits. However, it does raise some interesting issues when we consider the degree to which entanglement is critical to quantum computation. For example, suppose you wanted to create a superposition of two numbers "1" and "2" in some quantum memory register. Using qubits, such a superposition could be coded as $\frac{1}{\sqrt{2}}(|01\rangle + |10\rangle)$ (which is entangled). However, using qutrits, the equivalent state could be encoded as $\frac{1}{\sqrt{2}}(|1\rangle + |2\rangle)$ (a plain, unentangled, superposition). So the choice of base affects the degree to which entanglement is needed.

Some researchers misinterpreted the implications of this by proposing that quantum computation can be implemented without entanglement. For example, suppose we consider using a single atom (or perhaps artificial quantum dot) that has a huge spectrum of energy levels available to it. We could imagine associating each energy level with a different computational state: the ground state of the atom could be "$|0\rangle$", the first excited state "$|1\rangle$", the second excited state "$|2\rangle$" etc. We could then regard a quantum computation as a sequence of operations that maps some initial state of this atom (represented as an unentangled superposition of states) into a final state (represented as an unentangled superposition of states). And it would seem as though entanglement is unnecessary.

The problem with this approach is that it neglects a hidden exponential cost. To do universal (i.e., arbitrary) quantum computation we need to be able to access exponentially many different energy levels. However, as the total energy of the atom is finite, this means we will need to "fit" exponentially many energy levels into a finite energy interval. Hence, we will require exponentially increasing precision in order to address a specific energy level. Hence, although in principle one could perform quantum computation in higher bases, and perhaps lower the degree to which entanglement is needed, in practice it is very hard to imagine doing away with entanglement entirely.

1.5 Evolving a Quantum Memory Register: Schrödinger's Equation

So far we have been discussing the properties of individual quantum bits (such as superposition), and those of multi-qubit quantum memory registers (such as superposition, entanglement and interference). Our working assumption has been that the instantaneous state of a quantum memory register, $|\psi(t)\rangle$, holds the instantaneous state of the quantum computation. But how does this state evolve with time, and how can we control this evolution to enact a purposeful quantum computation?

1.5.1 Schrödinger's Equation

Remarkably in 1929, long before anyone had ever thought of quantum computers, physicist Erwin Schrödinger discovered an equation that describes how *any* isolated quantum system evolves in time. Since a quantum memory register is nothing more than an isolated quantum system, it too must be described by Schrödinger's equation.

Schrödinger's equation is a linear first order deterministic partial differential equation that involves the instantaneous state of the quantum memory register $|\psi(t)\rangle$, a time independent Hermitian matrix $\mathcal{H}$, called the Hamiltonian (the observable for the total energy of the system), and a constant $\hbar$ equal to Planck's constant divided by 2π. The fact that Schrödinger's equation is "linear" means that sums of solution to the equation are also solutions to the equation, which is the fundamental origin of the superposition principle. The fact that the Schrödinger equation is deterministic means that if you know its instantaneous state at any moment you can predict its future and past states with certainty (provided the system is not observed).

Regardless of the precise details of the physical system, Schrödinger's equation always takes the form:

$$i\hbar\frac{\partial|\psi(t)\rangle}{\partial t} = \mathcal{H}|\psi(t)\rangle \tag{1.18}$$

As $\hbar$ is a constant, and $|\psi(t)\rangle$ describes the instantaneous state of the quantum memory register, the form of this equation implies that all of the details pertaining to the particular physical system in question must be bundled into the operator $\mathcal{H}$—the Hamiltonian. So what does this Hamiltonian mean exactly?

1.5.2 Hamiltonians

In quantum mechanics observables are described by operators, which in turn can be represented as Hermitian matrices. The allowed values for an observable are the eigenvalues of its associated Hermitian matrix. The Hamiltonian, $\mathcal{H}$ is the observable corresponding to the total energy of the system, and its eigenvalues are the possible values one can obtain when one measures (or "observes") the total energy of the system. Depending on the physical situation such a Hamiltonian may be time dependent or time independent.

The Hamiltonian $\mathcal{H}$ for a particular quantum physical system is built up from knowledge of the elementary interactions available in the system, and it can be written in terms of operator products like those we encountered in Sect. 1.3.2. For example, the Hydra superconducting quantum processor [423] has the Hamiltonian:

$$\mathcal{H}(t) = \sum_{i=1}^{N} h_i Z_i + \sum_{i<j=2}^{N} J_{ij} Z_i Z_j + \sum_{i=1}^{N} \Delta_i(t) X_i \tag{1.19}$$

where $Z_i = \sigma_z^i$ and $X_i = \sigma_x^i$ are the Paul-Z and Pauli-X matrices for qubit i, h_i is the bias applied to qubit i, $\Delta_i(t)$ is the tunneling matrix element for qubit i, and J_{ij} is the coupling between qubits i and j.

The fact that $\mathcal{H}$ is the observable for the total energy of the n-qubit system means that $\mathcal{H}$ is a $2^n \times 2^n$ dimensional Hermitian matrix such that there exist energy eigenstates $|\psi_i\rangle$, and energy eigenvalues E_i such that $\mathcal{H}|\psi_i\rangle = E_i|\psi_i\rangle$. The eigenvalues E_i are the only allowed values for the total energy of the system. Thus there is always some basis (the energy eigenbasis $\{|\psi_i\rangle\}$) in which $\mathcal{H}$ is a diagonal matrix, $\mathcal{H} = \sum_i E_i|\psi_i\rangle\langle\psi_i|$.

$$\mathcal{H} = \begin{pmatrix} E_0 & 0 & 0 & 0 \\ 0 & E_1 & 0 & 0 \\ 0 & 0 & \ddots & 0 \\ 0 & 0 & 0 & E_{2^n-1} \end{pmatrix} \tag{1.20}$$

However, the Hamiltonian if often stated with respect to some other basis, e.g., the computational basis, $\{|00\ldots 0\rangle, |00\ldots 1\rangle, \ldots, |1\ldots 1\rangle\}$. Hence, it is sometimes necessary to change the basis used to describe states and operators in quantum computing. We will come back to this issue and discuss it in detail in Sect. 1.6.4.

1.5.3 Solution as a Unitary Evolution of the Initial State

Once the Hamiltonian is known the Schrödinger equation can be solved. The simplest case is that of a time-independent Hamiltonian. In this case the solution to the Schrödinger equation is:

$$U(t) = \exp(-i\mathcal{H}t/\hbar) \tag{1.21}$$

This says that if you know the initial state of the system, $|\psi(0)\rangle$, you can determine its state at a later time, t, by acting on the initial state with the operator $\exp(-i\mathcal{H}t/\hbar)$. Or, in other words, the system is described by some Hamiltonian $\mathcal{H}$ and you let it "run" for a length of time t, then the result you get is $|\psi(t)\rangle = U(t)|\psi(0)\rangle = \exp(-i\mathcal{H}t/\hbar)|\psi(0)\rangle$.

The matrix $U(t)$ is therefore the matrix exponential of $-i\mathcal{H}t\ \hbar$. If A is any matrix, its matrix exponential is:

$$e^A = \mathbb{1} + A + \frac{A^2}{2!} + \frac{A^3}{3!} + \frac{A^4}{4!} + \frac{A^5}{5!} + \cdots \tag{1.22}$$

As $\mathcal{H}$ is an Hermitian matrix, its matrix exponential $\exp(-i\mathcal{H}t/\hbar)$ is a *unitary* matrix. A unitary matrix has the property that its inverse is equal to its conjugate transpose, i.e. $U^{-1} = U^{\dagger}$. Therefore, a unitary matrix is always invertible which means that the evolution it describes is reversible, i.e., there is no loss of information. Hence, the closest classical analog to quantum computing is classical reversible computing, as it too preserves information about the computational history.

1.5.4 Computational Interpretation

A classical computer follows essentially a LOAD-RUN-READ cycle wherein one *loads* data into the machine, *runs* a program using this data as input, and then *reads* out the result. This becomes an analogous PREPARE-EVOLVE-MEASURE cycle for a quantum computer. That is, one *prepares* a quantum state, *evolves* it on the quantum computer, and *measures* the result.

Each aspect of the quantum computer's operation offers new opportunities unavailable in the analogous phase of a classical computer's operation. For example, whereas in a classical computer you can only *load* one input at a time, in a quantum computer you can *prepare* exponentially many inputs in the same amount of time. The whereas a classical computer can only run a computation on one input, a quantum computer can evolve a superposition of computations on all inputs in the same time. Finally, whereas a classical computer can only read one output, we can perform more sophisticated measurements of the output state from a quantum computer to compute certain *joint properties* of all the answers to a particular computational problem in the time it takes a classical computer to find just one of the answers. This gives quantum computers the potential to be much faster than any classical computer, even a state-of-the supercomputer.

1.6 Extracting Answers from Quantum Computers

The process of extracting answers from quantum computers can be more tricky than one might imagine. In order to learn the result of a quantum computation we must *read* the quantum memory register that contains it. Such an act is more properly thought of as performing a *measurement* on a certain quantum state (i.e., the result of the quantum computation) in a certain basis (typically, but not necessarily, the *computational* basis).

1.6.1 Observables in Quantum Mechanics

A measurement of a quantum memory register couples the quantum computer to the measuring device, temporarily, causing information from the quantum memory register to be transferred to the measuring apparatus, whereupon it is converted to classical information and amplified to a scale detectable by human senses. At this point we say the observable has been “read” or “measured”. Therefore, the act of reading a quantum memory register is more properly thought of as an experimental determination of the value of some *observable* of the system.

In quantum mechanics, an *observable* for some property of an n-qubit system is represented by a $2^n \times 2^n$ dimensional Hermitian matrix, $\mathcal{O}$ say. The Hermitian property means that $\mathcal{O} = \mathcal{O}^\dagger$ and so the eigenvalues of $\mathcal{O}$ are guaranteed to be real. the significance of this is that quantum mechanics says that when the property associated with observable $\mathcal{O}$ is measured that the answer we obtain has to be one of the *eigenvalues* of $\mathcal{O}$, and the state immediately after the measurement is the eigenvector that pairs with this eigenvalue. Thus, if $\{|\psi_i\rangle\}$ are the family of eigenvectors of $\mathcal{O}$ and $\{\lambda_i\}$ are the corresponding family of eigenvalues, such that:

$$\mathcal{O}|\psi_i\rangle = \lambda_i|\psi_i\rangle \tag{1.23}$$

then the only possible values we can ever obtain for the property associated with observable $\mathcal{O}$ are one of the λ_i’s and, having obtained such a result, the state immediately after this measurement will be $|\psi_i\rangle$. Moreover, if we repeatedly prepared and measured several preparations of the state $|\psi\rangle$ then the *average* value we would obtain would be:

$$\langle\mathcal{O}\rangle = \langle\psi|\mathcal{O}|\psi\rangle \tag{1.24}$$

where $|\psi\rangle$ and $\mathcal{O}$ should be described with respect to the same basis.

Many students find this measurement formalism perplexing. Why should acts of measurement be associated with matrices? And why should the values obtained from acts of measurements be associated with eigenvalues? What motivates this formalism?

The answer lies in our desire to have a mathematical way of describing acts of measurement that reflects, faithfully, the phenomena experimentalists encounter when they perform real measurements on quantum systems. As we shall see in the next section, by associating observables with Hermitian matrices, and the allowed values of observables with eigenvalues of those operators, we can conjure up a relatively simple and concise mathematical model of the measurement process that naturally has all the requisite properties.

1.6.1.1 Observables as Hermitian Operators

Let us start by summarizing the phenomena scientists encounter when they try to make observations on quantum systems, as this will make the subsequent mathematical account of observation that is used in quantum mechanics far more intuitive.

The first idea is that when we measure some property of a system we obtain a real number for the answer. So *measurement results need to be real numbers*.

Secondly, for quantum-scale objects, the act of observing the system can change its state. For example, to find the position of an electron you need to bounce light off it. The shorter the wavelength of the light used the more precisely you can determine position. But the shorter the wavelength of the light the greater the momentum kick the light imparts to the electron as it scatters off it. Hence, a very precise measurement of position necessarily induces a large uncertainty in momentum and vice versa. So the second idea is that *acts of observation can change the state*.

Third, the measured values one obtains do not usually span a continuous range of possibilities but instead may take on only certain discrete values. For example, if we measure the spin of an electron it is always found to be aligned or anti-aligned with the measurement axis. Even if you try to "cheat" by setting up an experiment with the electron spinning at some known angle to the axis of measurement, when you make the measurement the spin jumps into alignment or anti-alignment with the measurement axis. These are the only two values allowed. So the third idea is that *measured values are discrete rather than continuous*.

Fourth, the order in which we make a sequence of observations can affect the outcome we obtain. So an experiment that measures property A first and then property B does not always yield the same results, even statistically, as if we measured property B first and then property A. So the fourth idea is that *the order in which we perform measurements can affect the outcome we obtain*.

Fifth, when we measure certain pairs of observables, the more accurately we can pin down one, the less accurately we can pin down the other. That is there is a fundamental quantifiable limit to how accurately we can measure certain pairs of observables. In particular, defining:

$$\begin{aligned} \Delta\mathcal{O}_{\mathcal{A}} &= \mathcal{O}_{\mathcal{A}} - \langle\mathcal{O}_{\mathcal{A}}\rangle \\ \Delta\mathcal{O}_{\mathcal{B}} &= \mathcal{O}_{\mathcal{B}} - \langle\mathcal{O}_{\mathcal{B}}\rangle \end{aligned} \tag{1.25}$$

it can be shown that

$$\Delta\mathcal{O}_A \Delta\mathcal{O}_B \geq \text{constant} \tag{1.26}$$

where the inequality is strict if the order in which observations are made makes a difference. The mathematical machinery used in quantum mechanics to describe acts of observation has to reflect the phenomena scientists encounter when they do actual measurements.

It turns out that all these properties fall out naturally if we associate observables with Hermitian operators. If an observable A is associated with an Hermitian operator $\mathcal{O}_A$, then:

1. Quantized values: the only allowed outcomes for the measurement are the eigenvalues of $\mathcal{O}_A$.
2. Real values: as $\mathcal{O}_A$ is Hermitian its eigenvalues must be real.
3. Observation changes the state: if the system is in a superposition state just prior to a measurement then upon obtaining the result λ_i the system will be projected into the state $|\psi_i\rangle$. This is the eigenstate of $\mathcal{O}_A$ such that $\mathcal{O}_A|\psi_i\rangle = \lambda_i|\psi_i\rangle$.
4. Non-commuting Measurements: if we are interested in two observables A and B represented by Hermitian matrices $\mathcal{O}_A$ and $\mathcal{O}_B$ then the order in which measurements are made will make a difference whenever $\mathcal{O}_A \cdot \mathcal{O}_B \neq \mathcal{O}_B \cdot \mathcal{O}_A$.

5. Uncertainty principle: as we show in Chap. 12, for any pair of observables $\mathcal{O}_{\mathcal{A}}$ and $\mathcal{O}_{\mathcal{B}}$ there is a minimum uncertainty with which the $\mathcal{A}$ and $\mathcal{B}$ properties can be measured simultaneously given by $\Delta\mathcal{O}_{\mathcal{A}}\Delta\mathcal{O}_{\mathcal{B}} \geq \frac{1}{4}|\langle[\mathcal{O}_{\mathcal{A}}, \mathcal{O}_{\mathcal{B}}]\rangle|$.

Hence, although the quantum mechanical account of observables appears quite alien to most people when they first encounter it, remember that the reason it is set up this way is simply to capture the empirically determined properties of measurements and observations on quantum scale objects.

1.6.2 Observing in the Computational Basis

The most common kind of measurement that is made in quantum computing is to measure a set of qubits "in the computational basis". By this we mean that the spin orientation of each qubit in the quantum memory register is measured along an axis parallel to the z-axis of the Bloch sphere, which is the axis passing through its North and South poles. When such a measurement is made, each qubit will be found to be aligned or anti-aligned with the z-axis corresponding to being "spin-up" (i.e., in state $|0\rangle$) or "spin-down" (i.e., in state $|1\rangle$) respectively. When such a measurement is applied to each qubit in an n-qubit quantum memory register one will obtain one of the 2^n possible bit string configurations that the register can take on. The probably of obtaining the different outcomes depends upon the amplitude with which each bit string configuration appears in the superposition state of the register just prior to it being measured.

Consider, for example, an n-qubit quantum memory register in the (normalized) state $\sum_{i=0}^{2^n-1} c_i|i\rangle$. Here we use the shorthand notation that $|i\rangle$ really stands for a bit string, $|i\rangle \equiv |i_{n-1}i_{n-2}\dots i_2 i_1 i_0\rangle$, such that $i = 2^0 i_0 + 2^1 i_1 + \cdots + 2^{n-1} i_{n-1}$. The outcome we obtain will depend on the amplitudes c_i and on whether we measure some or all of the qubits.

1.6.2.1 Complete Readout

If *all* the qubits are measured in the computational basis one will obtain the result $|i\rangle$ with probability $|c_i|^2$. Consequently, if one of the amplitudes is zero, i.e., there exists an index value i' such that $c_{i'} = 0$, there is no chance whatsoever of obtaining the answer $|\psi_{i'}\rangle$ from the measurement. Conversely, if one of the amplitudes is unity, i.e., there exists an index value i'' such that $c_{i''} = 1$, then if the state is properly normalized, the result of the measurement is guaranteed to be the corresponding eigenstate, $|\psi_{i''}\rangle$.

Consider a 3-qubit quantum memory register that initially is in the state

$$\begin{aligned}|\psi\rangle &= c_0|000\rangle + c_1|001\rangle + c_2|010\rangle + c_3|011\rangle + c_4|100\rangle + c_5|101\rangle \\ &\quad + c_6|110\rangle + c_7|111\rangle \end{aligned} \tag{1.27}$$

Table 1.3 Probabilities of obtaining the eight distinct triples of values when three qubits are read in the computational basis

Qubit A	Qubit B	Qubit C	Probability
$\|0\rangle$	$\|0\rangle$	$\|0\rangle$	$\|c_0\|^2$
$\|0\rangle$	$\|0\rangle$	$\|1\rangle$	$\|c_1\|^2$
$\|0\rangle$	$\|1\rangle$	$\|0\rangle$	$\|c_2\|^2$
$\|0\rangle$	$\|1\rangle$	$\|1\rangle$	$\|c_3\|^2$
$\|1\rangle$	$\|0\rangle$	$\|0\rangle$	$\|c_4\|^2$
$\|1\rangle$	$\|0\rangle$	$\|1\rangle$	$\|c_5\|^2$
$\|1\rangle$	$\|1\rangle$	$\|0\rangle$	$\|c_6\|^2$
$\|1\rangle$	$\|1\rangle$	$\|1\rangle$	$\|c_7\|^2$

where $\sum_{i=0}^{7} |c_i|^2 = 1$. For convenience imagine labeling the leftmost qubit A, the middle qubit B, and the rightmost qubit C. When we do a complete measurement of all the qubits in this memory register, we expect to find the result $|i\rangle$ with probability $|c_i|^2$. That is we obtain the results shown in Table 1.3.

1.6.2.2 Partial Readout

Alternatively, suppose we measure only the middle qubit, B, and find it to be in state "$|1\rangle$". Such a measurement projects the qubits into a form that constrains the middle qubit to be $|1\rangle$, but leaves the other qubits indeterminate (since neither qubits A nor C were measured). Moreover, the resulting state must still be properly normalized. Hence, after the measurement, the state of the 3-qubit memory register is $\frac{c_2|010\rangle+c_3|011\rangle+c_6|110\rangle+c_7|111\rangle}{\sqrt{|c_2|^2+|c_3|^2+|c_6|^2+|c_7|^2}}$.

1.6.3 Alternative Bases

We do not have to view the contents of a quantum memory register as being in the computational basis however. A basis for an n-qubit quantum memory register is any complete orthonormal set of eigenstates such any n-qubit state can be written as a superposition of states taken from only this set. The computational basis states for a single qubit memory register are $|0\rangle$ and $|1\rangle$, and for an n-qubit quantum memory register the tensor product of all combinations of these, i.e. $\{|0\rangle, |1\rangle\}^{\otimes n} \equiv \{|00\ldots0\rangle, |00\ldots1\rangle, \ldots, |11\ldots1\rangle\}$. However, many other bases are possible, including those related to rotations of the single qubit computational basis states and tensor products thereof, and entirely unusual choices such as entangled multi-qubit states, e.g., the Bell basis. Table 1.4 shows some possible bases for a rudimentary (2-qubit) quantum memory register. The first three bases are related to rotations of the single computational basis states, but the fourth basis is a basis consisting of purely 2-qubit states, which is nevertheless a proper basis for 2-qubit states.

Table 1.4 Some examples of different bases for 2-qubit quantum memory register. Note that the Bell basis is defined over entangled 2-qubit states. The other bases shown are unitary transformations of the computational basis states $|0\rangle$ and $|1\rangle$

Basis	Eigenstates
θ° Rotated	$|\bar{0}\rangle = \cos\theta|0\rangle + \sin\theta|1\rangle$
	$|\bar{1}\rangle = \cos\theta|0\rangle - \sin\theta|1\rangle$
Diagonal	$|\nearrow\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$
	$|\nwarrow\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$
Chiral	$|\circlearrowright\rangle = \frac{1}{\sqrt{2}}(|0\rangle + i|1\rangle)$
	$|\circlearrowleft\rangle = \frac{1}{\sqrt{2}}(|0\rangle - i|1\rangle)$
Bell	$|\beta_{00}\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$
	$|\beta_{01}\rangle = \frac{1}{\sqrt{2}}(|01\rangle + |10\rangle)$
	$|\beta_{10}\rangle = \frac{1}{\sqrt{2}}(|00\rangle - |11\rangle)$
	$|\beta_{11}\rangle = \frac{1}{\sqrt{2}}(|01\rangle - |10\rangle)$

However, the proper way to think of this is that there is an *observable*, $\mathcal{A}$ say, whose eigenvectors correspond to the possible n-bit computational eigenstates, $|00\ldots0\rangle$, $|00\ldots1\rangle, \ldots, |11\ldots1\rangle$. To remind ourselves that these are eigenvectors of observable $\mathcal{A}$ we'll rename these eigenvectors $|a_i\rangle$ and call them the "a"-basis.

However, we do not *have* to use the computational basis to represent a state. Any complete orthonormal set of eigenvectors for an n-qubit state will do. In some circumstances, it is convenient to re-represent a given state in a new basis that simplifies some subsequent calculation. For example, suppose we are interested in calculating the expected outcomes of an observable property of an n-qubit state other than its bit values. Let us call the observable operator in which we are interested $\mathcal{B}$ having eigenvectors $|b_j\rangle$. Measuring observable $\mathcal{B}$ amounts to measuring the state $|\psi\rangle$ in the "b"-basis. The question is given a representation of a particular state $|\psi\rangle$ in the "a"-basis, how would this same state be represented in the "b"-basis? Knowing this we can then calculate the expected outcome from measuring observable $\mathcal{B}$ of $|\psi\rangle$.

First we need to know how the eigenvectors in the two bases are related. In particular, imagine creating the operator, U, defined as follows:

$$U = \sum_k |b_k\rangle\langle a_k| \tag{1.28}$$

An operator, U, of this form is unitary and induces the following mapping between the "a"-basis and the "b"-basis:

$$\begin{aligned} |b_1\rangle &= U|a_1\rangle \\ |b_2\rangle &= U|a_2\rangle \\ &\vdots \\ |b_{2^n}\rangle &= U|a_{2^n}\rangle \end{aligned} \tag{1.29}$$

Hence for each eigenvector in the "a"-basis there is a corresponding eigenvector in the "b"-basis.

1.6.4 Change of Basis

A given quantum state is not wedded to any particular basis. The same state can be interpreted as different superposition states of eigenstates from a completely different basis. Once this is understood, it makes it easier to appreciate why we might choose to observe a given state in a basis other than the computational basis.

Typically, in this book, we represent the states of a quantum memory register in the *computational* basis. That is, we write an n-qubit pure state in the form:

$$|\psi\rangle = \sum_i c_i |i\rangle \tag{1.30}$$

where $|i\rangle$ is the binary representation of integer i padded on the left with zeroes, if necessary, to make a full complement of n bits, and c_i is the complex amplitude with which eigenstate $|i\rangle$ contributes to the superposition, such that $\sum_i |c_i|^2 = 1$.

In the computational basis representation it is easy to calculate the probability of measuring the quantum memory register to be in a certain bit-string configuration, since configuration $|i\rangle$ will be found with probability $|c_i|^2$.

1.6.4.1 Change of Basis for a State

Thus a given state $|\psi\rangle$ can be written in either the "a"-basis or the "b"-basis. Specifically, we have:

$$|\psi\rangle = \sum_i \alpha_i |a_i\rangle = \sum_j \beta_j |b_j\rangle \tag{1.31}$$

where the amplitudes α_i and β_j are given by:

$$\alpha_i = \langle a_i|\psi\rangle \tag{1.32}$$

$$\beta_j = \langle b_j|\psi\rangle \tag{1.33}$$

Equation (1.29) tells us how to compute each "b"-basis vector $|b_k\rangle$ given its corresponding "a"-basis vector, $|a_k\rangle$, and U. So all we need to do now is to learn how to compute β_j. We can rewrite β_j as follows:

$$\begin{aligned}\beta_j &= \langle b_j|\psi\rangle = \langle b_j|\left(\sum_i |a_i\rangle\langle a_i|\right)|\psi\rangle = \sum_i \langle b_j|a_i\rangle\langle a_i|\psi\rangle \\ &= \sum_i \langle a_j|U^\dagger|a_i\rangle\langle a_i|\psi\rangle = \sum_i (U^\dagger)_{ji}\alpha_i\end{aligned} \tag{1.34}$$

where we have used the facts that $(\sum_i |a_i\rangle\langle a_i|) = \mathbb{1}$, the identity operator, and $|b_j\rangle = U|a_j\rangle$, which implies $\langle b_j| = \langle a_j|U^\dagger$. The last line of (1.34) is the usual form for the dot product of a matrix (i.e., $U^\dagger$) with a column vector (i.e., the column vector of amplitudes α_i). Hence, the column vector of amplitudes β_j representing the state $|\psi\rangle$ in the $\{|b_j\rangle\}$ (i.e.,"new") basis is related to column vector of amplitudes c_i in the $\{|a_i\rangle\}$ (i.e., "old") basis via the matrix equation:

$$|\psi\rangle_{\text{“}b\text{”-basis}} = U^\dagger |\psi\rangle_{\text{“}a\text{”-basis}} \tag{1.35}$$

where U is the operator define by (1.28) and which induced the connection between the bases given in (1.29).

Example: Linear versus Diagonal Polarization Bases Imagine a qubit encoded in the linear polarization state of a photon. By this we mean if we think of light as an oscillating electromagnetic wave, the plane in which the electric field component of that wave is oscillating, i.e., the state of its linear polarization, encodes our qubit. If the plane is "vertical" (with respect to some arbitrary axis in physical space) we say the qubit is a logical 0 $|\updownarrow\rangle$. Conversely, if the plane in which the electric field is oscillating is "horizontal" (with respect to the same axis in physical space) we say the qubit is a logical 1 $|\leftrightarrow\rangle$. Note, just to reinforce your understanding of the geometry on the Bloch sphere, on the Bloch sphere the states representing vertical and horizontal polarization ($|0\rangle \equiv |\updownarrow\rangle$ and $|1\rangle \equiv |\leftrightarrow\rangle$) correspond to the North Pole and South Pole respectively (i.e., at 180° separation). But in physical space the planes representing vertical and horizontally polarized photons lie at 90° to one another.

Now let's imagine switching to a polarization basis that it tilted at 45° with respect to the original basis. The new basis kets are $|\nearrow\!\!\!\swarrow\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$ (corresponding to a photon whose plane of polarization is tipped at +45° to the old plane of polarization) and $|\nwarrow\!\!\!\searrow\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$ (corresponding to a photon whose plane of polarization is tipped at −45° to the old plane of polarization). Thus following the recipe given above, the unitary matrix that maps a state in the old basis to its equivalent in the new basis is

$$\begin{aligned} U &= |\nearrow\!\!\!\swarrow\rangle\langle 0| + |\nwarrow\!\!\!\searrow\rangle\langle 1| = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)\langle 0| + \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)\langle 1| \\ &= \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \end{aligned} \tag{1.36}$$

1.6.4.2 Change of Basis for an Operator

Just as we can view quantum states in different bases, so too can we view quantum operators in different bases. Consider some operator, $\mathcal{O}$ say, given initially in the

"a"-basis. By inserting the identity operator twice we can write:

$$\begin{aligned}
\langle b_k|\mathcal{O}|b_\ell\rangle &= \langle b_k|\left(\sum_m |a_m\rangle\langle a_m|\right)\cdot X\cdot\left(\sum_m |a_m\rangle\langle a_m|\right)|b_\ell\rangle \\
&= \sum_m\sum_n \langle b_k|a_m\rangle\langle a_m|\mathcal{O}|a_n\rangle\langle a_n|b_\ell\rangle \\
&= \sum_m\sum_n \langle a_k|U^\dagger|a_m\rangle\langle a_m|\mathcal{O}|a_n\rangle\langle a_n|U|a_\ell\rangle \\
&= \sum_m\sum_n (U^\dagger)_{km}\mathcal{O}_{mn}U_{n\ell} \qquad (1.37)
\end{aligned}$$

This has the form of a "similarity transform", which is encountered routinely in linear algebra. That is, in matrix form, we can write:

$$\mathcal{O}_{\text{"}b\text{"-basis}} = U^\dagger\cdot\mathcal{O}_{\text{"}a\text{"-basis}}\cdot U \qquad (1.38)$$

Thus, given an operator in the "a"-basis equation (1.38) shows how to transform it into the "b"-basis.

1.6.5 Observing in an Arbitrary Basis

So far we have equated the act of observing a quantum memory register with the act of reading its bit values, or equivalently, measuring its qubits in the computational basis. However, a given quantum state does not have a unique interpretation: any state—even the state of a quantum memory register—can be pictured as different superposition states over different bases. Consequently, although most of the time in quantum computing it seems natural to read a quantum memory register in the computational basis, in some circumstances it might be more natural to read the quantum memory register with respect to some other basis.

Consider what this means in the case of a single qubit. Although a qubit might be defined initially with respect to the computational basis, i.e., as a state of the form $a|0\rangle + b|1\rangle$, where $|0\rangle$ is the North pole, and $|1\rangle$ the South pole, of the Bloch sphere, this same state can be re-represented in infinitely many other ways simply by changing which vectors we regard as the "basis" vectors.

Picture the state of a single qubit as a point on the surface of the Block sphere. Define a vector whose origin lies at the center of the Bloch sphere and whose tip touches this same point on the surface. Imagine keeping this vector in a fixed orientation but rotating the Bloch sphere surrounding it. Although the vector has not changed, the coordinates of its tip with respect to the x-, y-, and z-axes of the Bloch sphere have changed, and so the state at the tip of the vector appears to have changed. But however we rotate the axes around we can always pick an observation-axis that is on a line from the qubit state, $|\psi\rangle$ say (on the surface of the Bloch sphere),

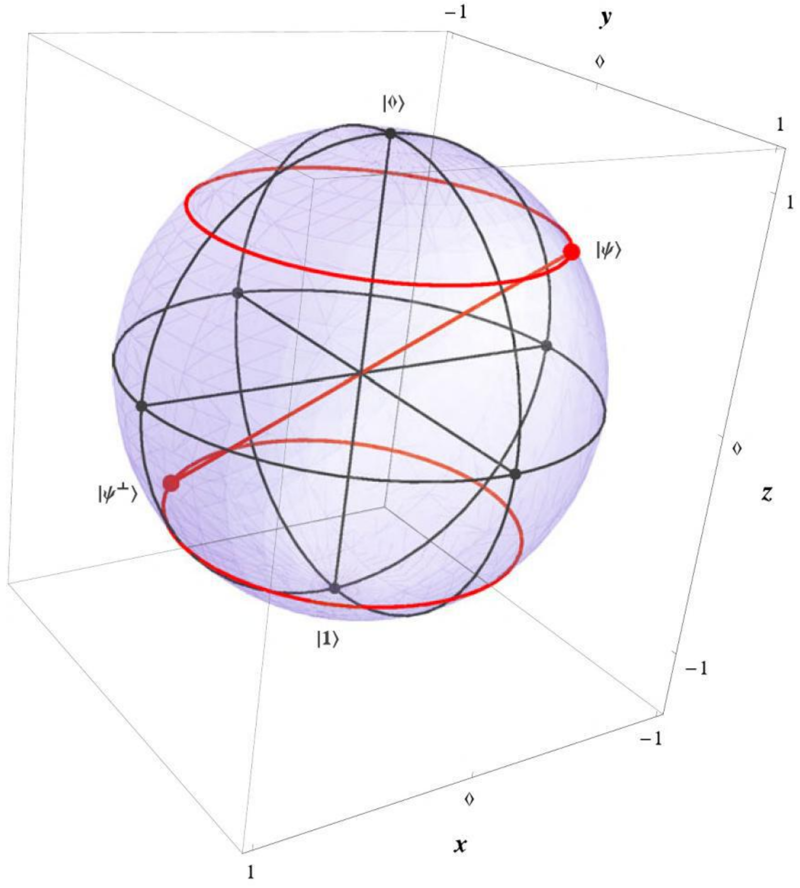


Fig. 1.7 Measuring the state of a qubit initially in state $a|0\rangle + b|1\rangle$ along an axis passing through states $|\psi\rangle$ and $|\psi^{\perp}\rangle$ corresponds to measuring the qubit in the $\{|\psi\rangle, |\psi^{\perp}\rangle\}$ basis

through the center of the Bloch sphere piercing the opposite side. The (antipodal) point where this line pierces the Bloch sphere corresponds to the quantum state $|\psi_{\perp}\rangle$, which is orthogonal to $|\psi\rangle$. Thus the basis made from states $\{|\psi\rangle, |\psi^{\perp}\rangle\}$ is equally as good as the computational basis, $\{|0\rangle, |1\rangle\}$, for describing single qubit states. Thus, it is possible to measure our qubit in this alternate $\{|\psi\rangle, |\psi^{\perp}\rangle\}$ basis too. Such a measurement is illustrated in Fig. 1.7.

1.7 Quantum Parallelism and the Deutsch-Jozsa Algorithm

Having introduced the main ideas of quantum computing we end this chapter by describing our first quantum computation—deciding whether a given function has a certain property using the Deutsch-Jozsa quantum algorithm. This computation cannot be solved as efficiently using any classical computer. It is not an especially *useful* computation, mind you. In fact, it is rather contrived. Nevertheless it illustrates many of the key steps in a typical quantum computation.

1.7.1 The Problem: Is $f(x)$ Constant or Balanced?

The problem, originally formulated by Cleve, Ekert, Macchiavello, and Mosca [112] as a variant of one by Deutsch and Jozsa [138] is this: Let x be any n-bit binary number and let $f(x)$ be a function that returns a single binary output (i.e., 0 or 1) for each value of x. Furthermore, we are promised that $f(x)$ behaves in only one of two possible ways: either $f(x)$ returns the same value for all binary inputs (in which case $f(x)$ is said to be *constant*), else $f(x)$ returns one bit value for half its inputs and the other bit value for the other half of its inputs (in which case $f(x)$ is said to be *balanced*). Finally, we are not allowed to inspect the mathematical definition of $f(x)$. Instead, we imagine $f(x)$ is given to us as a "black-box" function that acts in such a way that, when given the input x, the black box responds with the correct value for $f(x)$. Our task is to decide, using the fewest calls to the black-box, whether $f(x)$ is constant or balanced. Note that the decision does require us to exhibit the values of $f(x)$. Rather it only concerns a property those values possess, namely, whether they are all the same, or whether half have one bit value and half the other.

Using our conventional (classical) thinking, the number of times we would seem to need to call the black box is clear. There are a total of 2^n possible bit string inputs that can be made from n bits. Thus, we will need to check at least one more than half of them, i.e., $(\frac{1}{2} \times 2^n) + 1 = 2^{n-1} + 1$, to be able to decide with certainty whether $f(x)$ is constant or balanced. Note that we don't have to check all the 2^n input bit strings because we were promised that $f(x)$ is either constant or balanced. Thus, discovering $f(x)$ is non-constant is enough to conclude it must be the other possibility, namely, balanced. Even though we can avoid checking all inputs, classically, as larger and larger decision problems are considered the number of elementary calls to the black box would still seem to have to grow *exponentially* in the length of the input bit string n. In contrast, as we shall show, using a quantum computer, and a quantum implementation of the black-box that encodes $f(x)$, we can decide the question of whether $f(x)$ is constant or balanced in just *one* call to the black-box! This represents an *exponential speedup* in obtaining the decision—which is amazing!

Let's begin by looking at the simplest instance of such a decision problem when the input bit string is just a single bit (i.e., when $n = 1$). In this case, the decision problem can be stated as:

$$\text{decision}(f) = \begin{cases} \text{constant} & \text{iff } f(0) = f(1) \\ \text{balanced} & \text{iff } f(0) \neq f(1) \end{cases} \tag{1.39}$$

Using a classical computer we could decide the matter by first computing $f(0)$ and then computing $f(1)$ and then comparing the results to determine whether $f(0) =$

$f(1)$ or $f(0) \neq f(1)$. This approach would require *two* calls to the black box to make the decision regarding whether $f(x)$ is constant or balanced.

A quantum computer can solve this problem differently using a technique called *quantum parallelism*. To understand how quantum parallelism works we must first figure out how to define the action of the black-box that encodes knowledge of the function $f(x)$ in a manner that is consistent with quantum mechanics.

1.7.2 Embedding $f(x)$ in a Quantum Black-Box Function

On the face of it you might think that the black-box could be defined as performing the mapping $|x\rangle \rightarrow |f(x)\rangle$ since, in the special (i.e., $n = 1$) case we are considering both x and $f(x)$ are single bits. However, this won't do because as we saw in Sect. 1.5 quantum mechanical evolutions are described by unitary, and hence logically reversible, operations. For an operation to be logically reversible, each distinct input ought to be mapped to a distinct output and vice versa. Unfortunately, describing the black-box as performing the mapping $|x\rangle \rightarrow |f(x)\rangle$ is not necessarily logically reversible. If $f(x)$ happens to be constant then both possible values for $|x\rangle$ would be mapped into the same value for $f(x)$. So if the operation performed by our black-box is to be described quantum mechanically, the specification $|x\rangle \rightarrow |f(x)\rangle$ won't do. Strike one!

Ok well how about introducing an extra register—one to hold the input and the other to hold the output? The starting configuration could be $|x\rangle|0\rangle$, with the second register initialized to $|0\rangle$, which we can think of as analogous to a blank piece of paper on which the correct answer for $f(x)$ is to be written. In this case, our black-box would perform the operation $|x\rangle|0\rangle \rightarrow |x\rangle|f(x)\rangle$. Since the input, $|x\rangle$, is now recorded explicitly in the output we can always invert this mapping unambiguously, whatever the value of $f(x)$. Unfortunately, we're still not done because for a mapping between bit strings to be unitary (as quantum mechanics requires) we need to a complete mapping, i.e., a specification how each possible binary input is mapped to a distinct output. Since, the specification of the black-box as performing the operation $|x\rangle|0\rangle \rightarrow |x\rangle|f(x)\rangle$ only accounts for inputs that end in $|0\rangle$ it is missing half the possible inputs that could be given to it. Hence, the specification is incomplete, and therefore, won't do either. Strike two!

Thus to ensure our description of the black-box is unitary we need to specify how input states ending in $|0\rangle$ and states ending in $|1\rangle$ are to be mapped to outputs. Thus the right way to define the black-box operation is as

$$|x\rangle|y\rangle \longrightarrow |x\rangle|y \oplus f(x)\rangle. \tag{1.40}$$

The $y \oplus f(x)$ operation is the exclusive-OR operation, and is computed as shown in Table 1.5. When $y = 0$, $y \oplus f(x) = f(x)$, so the definition $|x\rangle|y\rangle \rightarrow |x\rangle|y \oplus f(x)\rangle$ includes the case $|x\rangle|0\rangle \rightarrow |x\rangle|f(x)\rangle$. But by defining the operation with the second qubit allowed to be either $|0\rangle$ or $|1\rangle$ we ensure that our description of the action of the black-box is a unitary (reversible) operation, which specifies a complete mapping between all possible 2-qubit binary inputs and all possible 2-qubit

Table 1.5 Truth table of the exclusive-OR ($\oplus$) operation. This is different from the usual OR operation ($\vee$) in that $1 \vee 1 = 1$ whereas $1 \oplus 1 = 0$

x	$f(x)$	$y \oplus f(x)$
0	0	0
0	1	1
1	0	1
1	1	0

binary outputs, and hence is implementable quantum mechanically. The operation $|x\rangle|y\rangle \xrightarrow{f\text{-c-N}} |x\rangle|y \oplus f(x)\rangle$ is sometimes called an "f-controlled-NOT" operation (f-c-N) since one way to think of it is that the value of $f(x)$ controls whether or not the value of y is negated.

1.7.3 *Moving Function Values Between Kets and Phase Factors*

Armed with our quantum black-box, which encapsulates the knowledge of $f(x)$, we are now ready to tackle the decision problem regarding whether $f(x)$ is constant or balanced.

If we restricted ourselves to inputting only quantum states corresponding to the "classical" binary inputs $|0\rangle|0\rangle$, $|0\rangle|1\rangle$, $|1\rangle|0\rangle$, and $|1\rangle|1\rangle$, to our quantum black box then our quantum method would confer no advantage over what we can do classically. The magic happens when we use quantum states corresponding to non-classical inputs. Specifically, consider what happens under the action of the f-controlled-NOT operation when the input is $|x\rangle \otimes \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$. The transformation effected is shown in (1.41)

$$|x\rangle\frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) \xrightarrow{f\text{-c-N}} |x\rangle\frac{1}{\sqrt{2}}(|0 \oplus f(x)\rangle - |1 \oplus f(x)\rangle) \tag{1.41}$$

As we are only considering the simplest ($n = 1$) instance of the decision problem at this time, the argument of $f(x)$, i.e., x, can be only 0 or 1, and the value of $f(x)$ is also only 0 or 1. So we can write out a table showing how the values of x, $f(x)$, and the right hand side of (1.41) are related: Notice that the table also contains a fourth column corresponding to the value of the expression $(-1)^{f(x)}|x\rangle\frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$. Remarkably, for all pairs of 2-bit binary inputs, the value returned by the expression $|x\rangle\frac{1}{\sqrt{2}}(|0 \oplus f(x)\rangle - |1 \oplus f(x)\rangle)$ is, as shown in Table 1.6, identical to the value returned by the expression $(-1)^{f(x)}|x\rangle\frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$. Hence—drum roll please—the two expressions are equally good mathematical descriptions of the output quantum state after the f-controlled-NOT operation has been applied. Thus, we could equally well describe the transformation the f-controlled-NOT operation has achieved as:

$$|x\rangle\frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) \xrightarrow{f\text{-c-N}} (-1)^{f(x)}|x\rangle\frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) \tag{1.42}$$

Table 1.6 By considering the possible values of x, $f(x)$ and the right hand side of (1.41) recognize an equivalent way to write the equation

x	$f(x)$	$\lvert x\rangle \frac{1}{\sqrt{2}}(\lvert 0\oplus f(x)\rangle - \lvert 1\oplus f(x)\rangle)$	$(-1)^{f(x)}\lvert x\rangle\frac{1}{\sqrt{2}}(\lvert 0\rangle - \lvert 1\rangle)$
0	0	$\lvert 0\rangle(\lvert 0\rangle - \lvert 1\rangle)$	$\lvert 0\rangle(\lvert 0\rangle - \lvert 1\rangle) = \lvert 0\rangle(\lvert 0\rangle - \lvert 1\rangle)$
0	1	$\lvert 0\rangle(\lvert 1\rangle - \lvert 0\rangle)$	$-\lvert 0\rangle(\lvert 0\rangle - \lvert 1\rangle) = \lvert 0\rangle(\lvert 1\rangle - \lvert 0\rangle)$
1	0	$\lvert 1\rangle(\lvert 0\rangle - \lvert 1\rangle)$	$\lvert 1\rangle(\lvert 0\rangle - \lvert 1\rangle) = \lvert 1\rangle(\lvert 0\rangle - \lvert 1\rangle)$
1	1	$\lvert 1\rangle(\lvert 1\rangle - \lvert 0\rangle)$	$-\lvert 1\rangle(\lvert 0\rangle - \lvert 1\rangle) = \lvert 1\rangle(\lvert 1\rangle - \lvert 0\rangle)$

Thus, with no physical action whatsoever taking place, we can simply re-interpret what mathematical transformation we have achieved. This re-interpretation of the output state allows us to regard the value of the function $f(x)$ as being moved from inside the ket (in (1.41)) to being in the phase factor (in (1.42)). This is very important because we saw in Sect. 1.4.3 that quantum mechanical interference effects can change the relative probabilities of various outcomes. What we will do next is engineer these interference effects to enhance or suppress various possible outcomes depending on whether $f(x)$ is constant or balanced!

1.7.4 Interference Reveals the Decision

To achieve our desired interference effect we take the interpretation of the $f =$ controlled-NOT transformation defined in (1.42) and we specialize the input $|x\rangle$ to be $|x\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$. We can create this state by applying a Walsh-Hadamard gate to just the first qubit prepared initially in the state $|0\rangle$, i.e., $H|0\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$. With this specialization, the transformation we perform is therefore:

$$\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)\frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) \xrightarrow{f\text{-c-N}} \frac{1}{\sqrt{2}}\left((-1)^{f(0)}|0\rangle + (-1)^{f(1)}|1\rangle\right)\frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) \tag{1.43}$$

Next we apply a Walsh-Hadamard gate to just the first qubit again. This results in the transformation:

$$\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)\frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) \xrightarrow{f\text{-c-N}} \frac{1}{\sqrt{2}}\left((-1)^{f(0)}|0\rangle + (-1)^{f(1)}|1\rangle\right)\otimes\frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) \tag{1.44}$$

Summarizing all the steps in the Deutsch-Jozsa algorithm:

$$\begin{aligned}|0\rangle|1\rangle &\xrightarrow{H\otimes H} \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)\frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)\\ &\xrightarrow{f\text{-c-N}} \frac{1}{\sqrt{2}}\left((-1)^{f(0)}|0\rangle + (-1)^{f(1)}|1\rangle\right)\otimes\frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)\end{aligned}$$

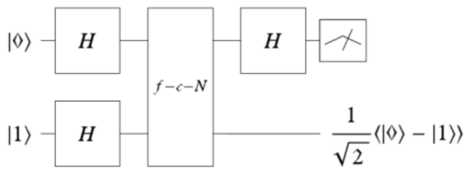


Fig. 1.8 Quantum circuit implementing the Deutsch-Jozsa algorithm. The black-box function $f(x)$ accepts a single bit x and returns 0 or 1. If the returned values are the same $f(x)$ is "constant". Otherwise $f(x)$ is "balanced". The function $f(x)$ is implemented by way of the Deutsch-Jozsa oracle f-controlled-NOT (*f-c-N*). This implements the transformation $|x\rangle|y\rangle \overset{f\text{-c-N}}{\rightarrow} |x\rangle|y \oplus f(x)\rangle \equiv (-1)^{f(x)}|x\rangle|y\rangle$ when $|y\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$. Using the Deutsch-Jozsa algorithm we can decide whether $f(x)$ is constant or balanced using a single call to the oracle

$$\begin{aligned}\xrightarrow{H\otimes\mathbb{1}} \Bigg[&\left(\frac{1}{2}(-1)^{f(0)} + \frac{1}{2}(-1)^{f(1)}\right)|0\rangle \\ &+ \left(\frac{1}{2}(-1)^{f(0)} - \frac{1}{2}(-1)^{f(1)}\right)|1\rangle\Bigg] \otimes \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)\end{aligned} \tag{1.45}$$

Inspection of the amplitudes of the $|0\rangle$ and $|1\rangle$ components of the first qubit suggest that if this qubit is read (in the computational basis) then if $f(x)$ is constant, i.e., if $f(0) = f(1)$, then we will find the first qubit in state $|0\rangle$. Else if $f(x)$ is balanced, i.e., $f(0) \neq f(1)$, then we will find the first qubit in state $|1\rangle$. This means we can determine whether $f(x)$ is constant or balanced in just one call to the black-box (when using quantum inputs) versus two calls to the black-box (if using classical bit value inputs). The quantum circuit implementing the Deutsch-Jozsa algorithm is shown in Fig. 1.8. This is interesting but not that dramatic. To determine what scaling we're actually seeing we need to consider the relative costs of the quantum and classical methods as we scale up to larger problem instances.

1.7.5 Generalized Deutsch-Jozsa Problem

The aforementioned decision problem only pertains to a function $f(x)$ that has a single bit input and a single bit output. In this case we obtain a factor of two speedup over the naive classical algorithm for the solving the same problem. How does this speedup change if we allow f to accept an n-bit input instead of just a single bit input?

To formalize the question, let $\boldsymbol{x} = x_1 x_2 \cdots x_n$ be an n-bit binary string with binary values $x_1, x_2, \ldots, x_n$. Thus, $\boldsymbol{x}$ represents the bit string corresponding to any integer in the range 0 to $2^n - 1$ inclusive. Let $f(\boldsymbol{x})$ be a function that accepts and n-bit input $\boldsymbol{x}$ and returns a single bit output, i.e., 0 or 1. We are promised that $f(\boldsymbol{x})$ is one of only two kinds of function, namely, "constant" or "balanced". An n-bit function $f(\boldsymbol{x})$ is "constant" if it returns the same bit value on all 2^n possible inputs, and "balanced" if it returns 0 on exactly half its possible inputs, and 1 on the other half of inputs. Note that the promise is our guarantee that the only types of functions under consideration are constant functions and balanced functions and no others. With this promise in mind, our challenge is to decide whether $f(\boldsymbol{x})$ is constant or balanced using the fewest calls the oracle.

Classically, in the worst case, we will have to call the oracle a total of $\frac{1}{2}2^n + 1 = 2^{n-1} + 1$ times. This is because we cannot know for sure that $f(\boldsymbol{x})$ is constant until we have evaluated $f(\boldsymbol{x})$ on one more than half its possible inputs. At that point if all the returned values are the same, the promise allows us to conclude that $f(\boldsymbol{x})$ is constant. However, if in the course of performing these $2^{n-1} + 1$ evaluations we find any two inputs that yield different values for the function, then the promise allows us to conclude the given $f(\boldsymbol{x})$ is balanced. So, given the promise, on average deciding that $f(\boldsymbol{x})$ is balanced is easier than deciding it is constant. But in the worst case (when we are unlucky enough that even though $f(\boldsymbol{x})$ is balanced the first 2^{n-1} inputs we tried happened to be those for which $f(\boldsymbol{x})$ returned the same value), we need to test one more than half the values to be sure. Since the classical algorithm needs $2^{n-1} + 1$ calls to the oracle, the classical complexity is *exponential* in the number of bits, n.

Can do better using a quantum algorithm? As you will see shortly, it turns out that there is a quantum algorithm, the "Generalized Deutsch-Jozsa Algorithm", for solving this same decision problem that only needs to make a *single* call to the oracle, regardless of n. This amounts to an exponential speedup over what is possible classically! This is an astonishing difference in complexity between a quantum computer and a classical computer on the same problem. So even though the actual problem solved is rather arcane and esoteric, nevertheless, it illustrates the enormous potential of quantum computers to outperform classical computers on certain computational problems.

The best way to see how the Generalized Deutsch-Jozsa algorithm works is to start with the quantum circuit that implements it and to walk through the state transformations it enacts. This will allow us to compute the mathematical form of the final state that is synthesized by the circuit and hence determine how a measurement made upon this final state can reveal the decision regarding whether $f(\boldsymbol{x})$ is constant or balanced.

The quantum circuit for the generalized Deutsch-Jozsa algorithm is shown in Fig. 1.9 and the associated algorithm is as follows:

Generalized Deutsch-Jozsa Algorithm Given an oracle, or black-box quantum function, $f(\boldsymbol{x})$ that accepts an n-bit binary string input, $\boldsymbol{x} = x_1 x_2 \cdots x_n$, and the promise that $f(x)$ is either constant or balanced, decide which is the case using the fewest calls to the oracle.

1. Create an $(n+1)$-qubit quantum register having n control qubits, each in state $|0\rangle$, and one ancilla qubit in state $|1\rangle$.

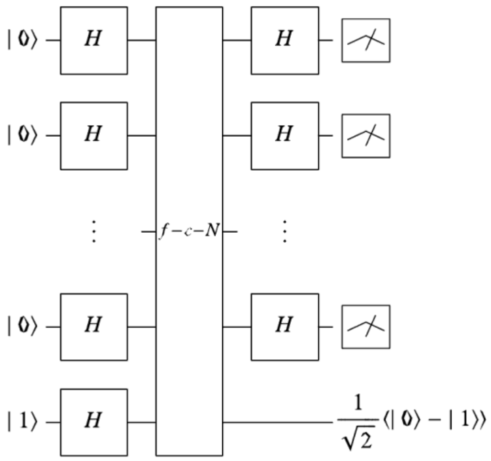


Fig. 1.9 Quantum circuit implementing the Generalized Deutsch-Jozsa algorithm. The black-box function $f(\boldsymbol{x})$ accepts an n-bit input $\boldsymbol{x}$ and returns the single bit 0 or 1. We are promised that $f(\boldsymbol{x})$ is either "constant" (i.e., returns the same value on all its possible 2^n inputs) or "balanced" (i.e., returns 0 on half of its possible inputs and 1 on the other half of its possible inputs). The function $f(\boldsymbol{x})$ is implemented by way of the Generalized Deutsch-Jozsa oracle f-controlled-NOT (f-c-N). This implements the transformation $|\boldsymbol{x}\rangle|y\rangle \overset{f\text{-c-N}}{\rightarrow} |\boldsymbol{x}\rangle|y \oplus f(\boldsymbol{x})\rangle$. In turn, this is equivalent to $(-1)^{f(x)}|x\rangle|y\rangle$ when $|y\rangle$ is specialized to be the state $|y\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$. Using the Generalized Deutsch-Jozsa algorithm we can decide whether $f(\boldsymbol{x})$ is constant or balanced using a *single* call to the oracle. A classical computer would need to use $\frac{1}{2}2^n + 1$ calls to the oracle to arrive at the same decision. Hence, in this case, a quantum computer running the Generalized Deutsch-Jozsa algorithm is exponentially more efficient than a classical computer. Hence, the Generalized Deutsch-Jozsa algorithm, although not particularly useful as a practical algorithm, illustrates the potential for enormous complexity advantages of quantum computers over classical computers on certain problems

2. Apply a Walsh-Hadamard gate to each qubit. That is, perform the operation:

$$|00\ldots0\rangle|1\rangle \overset{H^{\otimes(n+1)}}{\longrightarrow} \frac{1}{\sqrt{2^n}} \sum_{\boldsymbol{x}=0}^{2^n-1} |\boldsymbol{x}\rangle \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) \tag{1.46}$$

3. Then apply the Generalized Deutsch-Jozsa oracle.

$$\overset{f\text{-c-N}}{\longrightarrow} \frac{1}{\sqrt{2^n}} \sum_{\boldsymbol{x}=0}^{2^n-1} (-1)^{f(\boldsymbol{x})} |\boldsymbol{x}\rangle \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) \tag{1.47}$$

4. Apply a Walsh-Hadamard gate to the top n qubits.

$$\overset{H^{\otimes n}\otimes\mathbb{1}}{\longrightarrow} \frac{1}{\sqrt{2^n}} \sum_{\boldsymbol{x}=0}^{2^n-1} (-1)^{f(\boldsymbol{x})} \underbrace{(H \otimes H \otimes \cdots \otimes H)}_{n} |\boldsymbol{x}\rangle \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) \tag{1.48}$$

$$\equiv \frac{1}{\sqrt{2^n}}\left(\sum_{\boldsymbol{x}=0}^{2^n-1}(-1)^{f(\boldsymbol{x})}\frac{1}{\sqrt{2^n}}\sum_{z=0}^{2^n-1}(-1)^{\boldsymbol{x}\cdot\boldsymbol{z}}|z\rangle\right)\frac{1}{\sqrt{2}}(|0\rangle-|1\rangle) \tag{1.49}$$

$$\equiv \frac{1}{2^n}\left(\sum_{\boldsymbol{x}=0}^{2^n-1}\sum_{z=0}^{2^n-1}(-1)^{f(\boldsymbol{x})}(-1)^{\boldsymbol{x}\cdot\boldsymbol{z}}|z\rangle\right)\frac{1}{\sqrt{2}}(|0\rangle-|1\rangle) \tag{1.50}$$

5. Measure the top n qubits in the computational basis. If the first n qubits are found to be in state $|\mathbf{0}\rangle = |00\ldots0\rangle$, $f(\boldsymbol{x})$ is "constant". If any other pattern of values is obtained for the first n qubits, then $f(\boldsymbol{x})$ is "balanced".

The algorithm works as follows: in the first step we initialize n control qubits to be in state $|00\ldots0\rangle$ and we initialize a single ancilla qubit to be in state $|1\rangle$. Next we apply our oracle, i.e., the f-controlled-NOT gate. This acts on n control bits (which hold the value of the input "$\boldsymbol{x}$") and one target qubit (which starts off in state $\frac{1}{\sqrt{2}}(|0\rangle-|1\rangle)$). The transformation the oracle performs is:

$$\underbrace{|\boldsymbol{x}\rangle}_{n\text{ qubits}} \otimes \underbrace{|y\rangle}_{1\text{ qubit}} \xrightarrow{f\text{-c-N}} |\boldsymbol{x}\rangle\otimes|y\oplus f(\boldsymbol{x})\rangle \equiv (-1)^{f(\boldsymbol{x})}|\boldsymbol{x}\rangle|y\rangle \tag{1.51}$$

Next we apply a Walsh-Hadamard gate to the top n qubits only. This is perhaps the hardest part of the algorithm to understand because it is not immediately obvious why $H^{\otimes n}|\boldsymbol{x}\rangle = \frac{1}{\sqrt{2^n}}\sum_{z=0}^{2^n-1}(-1)^{\boldsymbol{x}\cdot\boldsymbol{z}}|z\rangle$. To see why this is true let's start by considering a simple 3-qubit instance of the problem.

$$\begin{aligned}
(H\otimes H\otimes H)|\boldsymbol{x}\rangle &= (H\otimes H\otimes H)|x_1x_2x_3\rangle \\
&= H|x_1\rangle\otimes H|x_2\rangle\otimes H|x_3\rangle \\
&= \frac{1}{\sqrt{2^3}}(|0\rangle+(-1)^{x_1}|1\rangle)\otimes(|0\rangle+(-1)^{x_2}|1\rangle) \\
&\quad \otimes(|0\rangle+(-1)^{x_3}|1\rangle) \\
&= \frac{1}{\sqrt{2^3}}\big(|000\rangle+(-1)^{x_3}|001\rangle+(-1)^{x_2}|010\rangle \\
&\quad +(-1)^{x_2+x_3}|011\rangle+(-1)^{x_1}|100\rangle \\
&\quad +(-1)^{x_1+x_3}|101\rangle+(-1)^{x_1+x_2}|110\rangle+(-1)^{x_1+x_2+x_3}|111\rangle\big) \\
&= \frac{1}{\sqrt{2^3}}\sum_{z_1=0}^{1}\sum_{z_2=0}^{1}\sum_{z_3=0}^{1}(-1)^{x_1z_1+x_2z_2+x_3z_3}|z_1z_2z_3\rangle \\
&= \frac{1}{\sqrt{2^3}}\sum_{z=0}^{2^3-1}(-1)^{\boldsymbol{x}\cdot\boldsymbol{z}}|z\rangle
\end{aligned} \tag{1.52}$$

(a) $|0\rangle + e^{i\theta}|1\rangle$
(b) $\frac{1}{2}|0\rangle - \frac{2}{3}|1\rangle$
(c) $i|00\rangle + |01\rangle + |00\rangle$
(d) $(|0\rangle + i|1\rangle) \otimes (|0\rangle - i|1\rangle)$

1.3 Compute the probability with which each of the following qubits is found in the state $|0\rangle$ when measured in the computational basis. Be careful as the given state may or may not be properly normalized as given.

1. $\frac{1}{2}|0\rangle + \frac{\sqrt{3}}{2}|1\rangle$
2. $-\frac{i}{3}|0\rangle - \frac{\sqrt{3}}{2}|1\rangle$
3. $\frac{2\sqrt{2}}{3}|0\rangle + \frac{1}{3}|1\rangle$
4. $\frac{i}{2}|0\rangle - \frac{\sqrt{3}}{2}|1\rangle$

1.4 Let $|\bar{0}\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$ and $|\bar{1}\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$. Prove that the states $|\bar{0}\rangle$ and $|\bar{1}\rangle$ are orthogonal. Write the state $a|0\rangle + b|1\rangle$ in the $\{|\bar{0}\rangle, |\bar{1}\rangle\}$ basis.

1.5 The memory register of a 3-qubit quantum computer evolves into the state $\frac{1}{3}|001\rangle + \frac{\sqrt{5}}{3}|010\rangle + \frac{1}{\sqrt{3}}|100\rangle$. What is the probability of:

1. Finding the first qubit to be $|1\rangle$?
2. Finding the second qubit to be $|0\rangle$?
3. Finding the last two qubits to be $|00\rangle$?

1.6 Which of the following states are entangled, and which are unentangled?

(a) $\frac{1}{\sqrt{2}}(|000\rangle + |111\rangle)$
(b) $\frac{1}{3\sqrt{2}}|00\rangle + \frac{2}{3}|01\rangle + \frac{1}{3\sqrt{2}}|10\rangle + \frac{2}{3}|11\rangle$
(c) $\frac{1}{6}|00\rangle - \frac{1}{2\sqrt{3}}|01\rangle + \frac{\sqrt{2}}{3}|10\rangle - \sqrt{\frac{2}{3}}|11\rangle$
(d) $\frac{1}{\sqrt{3}}(|001\rangle + |010\rangle + |100\rangle)$
(e) $\frac{1}{\sqrt{2}}(|00\rangle - i|10\rangle)$

1.7 Prove that the quantum state $|\psi\rangle$ defined by:

$$\begin{aligned} |\psi\rangle = {} & \frac{6}{\sqrt{181}}|000\rangle - \frac{4}{\sqrt{181}}|001\rangle + \frac{3}{\sqrt{181}}|010\rangle - 4\sqrt{\frac{3}{181}}|011\rangle \\ & + \sqrt{\frac{2}{181}}|100\rangle - \sqrt{\frac{6}{181}}|101\rangle + \frac{4}{\sqrt{181}}|110\rangle - 4\sqrt{\frac{3}{181}}|111\rangle \end{aligned} \tag{1.55}$$

is properly normalized. Given the state $|\psi\rangle$, what is the probability, when you read $|\psi\rangle$ in the computational basis, of obtaining:

(a) the result $|010\rangle$?

(b) the result $|001\rangle$?
(c) finding the first qubit to be in state $|1\rangle$?
(d) finding the first and third qubits to both be in state $|0\rangle$?
(e) finding the first and second qubits to be the same?

1.8 Consider a single qubit in state $|\psi\rangle = \cos(\frac{\theta}{2})|\psi\rangle + e^{i\phi}\sin(\frac{\theta}{2})|1\rangle$ such that $0 \leq \theta \leq \pi$ and $0 \leq \phi \leq 2\pi$. Prove that the state $|\psi\rangle^{\perp}$ at the antipodal point of the Bloch sphere is orthogonal to $|\psi\rangle$. The antipodal point is found by projecting a straight line from the point on the surface of the Bloch sphere representing $|\psi\rangle$ through the origin to intersect the surface of the Bloch sphere on the opposite side.

1.9 Prove that the expectation value of any observable $\mathcal{A}$, $\langle\psi|\mathcal{A}|\psi\rangle$, for a quantum system in state $|\psi\rangle$ is no different from that obtained if the state where $e^{i\phi}|\psi\rangle$ instead. That is, prove the claim made in this chapter that overall phase factors have no observable consequence.

1.10 Let Ω be an observable operator for a single qubit described as:

$$\Omega = \begin{pmatrix} a & b \\ c & d \end{pmatrix}$$

Answer the following questions:

(a) Which elements of Ω must be real numbers?
(b) Which elements of Ω can be complex numbers?
(c) Which two elements of Ω are related?
(d) What are the eigenvalues of Ω?
(e) What is the expectation value $\langle\psi|\Omega|\psi\rangle$ when $|\psi\rangle = \alpha|0\rangle + \sqrt{1-|\alpha|^2}|1\rangle$?

1.11 A qubit in an arbitrary pure quantum state is described mathematically by the state vector $|\psi\rangle = e^{i\gamma}(\cos(\frac{\theta}{2})|0\rangle + e^{i\phi}\sin(\frac{\theta}{2})|1\rangle)$. Equivalently, as you will see in Sect. 11.2.2 the *same* state can be described by the density operator $\rho = |\psi\rangle\langle\psi|$. On how many free parameters does the state $|\psi\rangle$ depend? Compute the density operator ρ corresponding to the state $|\psi\rangle$. On how many free parameters does the density operator ρ depend? Explain what role the parameter γ plays in the density operator representation of the state.

1.12 Look up the definition of the quantum Fourier transform (QFT) matrix defined in Sect. 3.4.6. Prove that the 1-qubit Walsh-Hadamard gate, $H = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$, can be thought of as a 1-qubit quantum Fourier transform.

1.13 Find the unitary matrix that changes a state represented in the $\{|+\rangle, |-\rangle\}$ basis to one represented in the $\{|R\rangle, |L\rangle\}$ basis. You may assume $|+\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$, $|-\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$, $|R\rangle = \frac{1}{\sqrt{2}}(|0\rangle + i|1\rangle)$, and $|L\rangle = \frac{1}{\sqrt{2}}(|0\rangle - i|1\rangle)$.

1.14 Find the unitary matrix that changes an arbitrary 2-qubit gate,

$$U = \begin{pmatrix} u_{11} & u_{12} & u_{13} & u_{14} \\ u_{21} & u_{22} & u_{23} & u_{24} \\ u_{31} & u_{32} & u_{33} & u_{34} \\ u_{41} & u_{42} & u_{43} & u_{44} \end{pmatrix},$$

in the $\{|00\rangle, |01\rangle, |10\rangle, |11\rangle\}$ basis to one represented in the $\{|RR\rangle, |RL\rangle, |LR\rangle, |LL\rangle\}$ basis. You may assume $|R\rangle = \frac{1}{\sqrt{2}}(|0\rangle + i|1\rangle)$, and $|L\rangle = \frac{1}{\sqrt{2}}(|0\rangle - i|1\rangle)$, and $|RL\rangle = |R\rangle \otimes |L\rangle$ etc.

1.15 Consider a 2-qubit Hamiltonian having a block diagonal structure when expressed in the computational basis:

$$\mathcal{H} = \begin{pmatrix} a & e & 0 & 0 \\ e^* & b & 0 & 0 \\ 0 & 0 & c & g \\ 0 & 0 & g^* & d \end{pmatrix} \tag{1.56}$$

What are the eigenvalues and *normalized* eigenvectors of $\mathcal{H}$?

1.16 Suppose we are promised that we are given either a known state $|\psi\rangle$ or a known state $|\varphi\rangle$ and we have to decide, by making some measurement, which is the case. If $|\psi\rangle$ and $|\varphi\rangle$ are non-orthogonal quantum states there is no single measurement that can distinguish between them 100% of the time. However, given knowledge of the forms for $|\psi\rangle$ and $|\varphi\rangle$ we can choose a measurement basis in which to measure our mystery state that optimizes our chances of guessing correctly. For example, consider the pair of quantum states defined by:

$$\begin{aligned} |\psi\rangle &= |0\rangle \\ |\varphi\rangle &= \alpha(\theta)|0\rangle + \beta(\theta)|1\rangle \end{aligned} \tag{1.57}$$

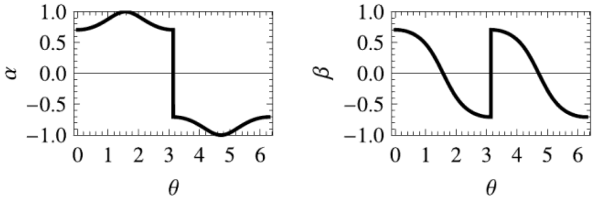
where

$$\begin{aligned} \alpha(\theta) &= \frac{\csc\theta}{\sqrt{|\csc\theta|^2 + |\cot\theta|^2}} \\ \beta(\theta) &= \frac{\cot\theta}{\sqrt{|\csc\theta|^2 + |\cot\theta|^2}} \end{aligned} \tag{1.58}$$

The amplitudes of the $|\varphi\rangle$ state are certainly peculiar, having the form over the interval $0 \le \theta \le 2\pi$ shown in Fig. 1.10.

(a) Nevertheless, prove that $|\varphi\rangle$ is a properly normalized state.
(b) With what probability can we guess correctly if we measure the mystery state in the computational, i.e., $\{|0\rangle, |1\rangle\}$, basis?
(c) In what basis *ought* we to make the measurement to maximize our chances of guessing correctly whether we were given $|\psi\rangle$ and $|\varphi\rangle$?

Fig. 1.10 Amplitudes of the state $|\varphi\rangle = \alpha|0\rangle - \beta|1\rangle$ where $\alpha = \frac{\csc\theta}{\sqrt{|\csc\theta|^2+|\cot\theta|^2}}$ and $\beta = \frac{\cot\theta}{\sqrt{|\csc\theta|^2+|\cot\theta|^2}}$



(d) What is the state $|\varphi\rangle$ at $\theta = \pi$? Is there any ambiguity?

(e) What is the density operator corresponding to $|\varphi\rangle$, i.e., $\rho = |\varphi\rangle\langle\varphi|$?

(f) What is the density operator ρ at $\theta = \pi$? Is there any ambiguity? How do you reconcile your answers to parts (d) and (f)?

Chapter 2
Quantum Gates

"When we get to the very, very small world—say circuits of seven atoms—we have a lot of new things that would happen that represent completely new opportunities for design. Atoms on a small scale behave like nothing on a large scale, for they satisfy the laws of quantum mechanics. So, as we go down and fiddle around with the atoms down there, we are working with different laws, and we can expect to do different things. We can manufacture in different ways. We can use, not just circuits, but some system involving the quantized energy levels, or the interactions of quantized spins."
– Richard P. Feynman[1]

Currently, the circuit model of a computer is the most useful abstraction of the computing process and is widely used in the computer industry in the design and construction of practical computing hardware. In the circuit model, computer scientists regard any computation as being equivalent to the action of a circuit built out of a handful of different types of Boolean logic gates acting on some binary (i.e., bit string) input. Each logic gate transforms its input bits into one or more output bits in some deterministic fashion according to the definition of the gate. By composing the gates in a graph such that the outputs from earlier gates feed into the inputs of later gates, computer scientists can prove that any feasible computation can be performed.

In this chapter we will look at the types of logic gates used within circuits and how the notions of logic gates need to be modified in the quantum context.

[1]Source: Opening words of the "Atoms in a SmallWorld" section of Richard Feynman's classic talk "There's Plenty of Room at the Bottom," given on 29th December 1959 at the annual meeting of the American Physical Society at the California Institute of Technology. The full transcript of the talk is available at http://www.zyvex.com/nanotech/feynman.html.

DOI 10.1007/978-1-84628-887-6_2, © Springer-Verlag London Limited 2011

Table 2.1 Logically equivalent propositions. Note by using De Morgan's laws any proposition can be expressed using NOT and AND alone or using NOT and OR alone

Logically equivalent forms	
$a \wedge 0 = 0$	Zero of $\wedge$
$a \wedge 1 = a$	Identity of $\wedge$
$a \vee 0 = a$	Zero of $\vee$
$a \vee 1 = 1$	Identity of $\vee$
$a \wedge a = a$	Indempotence
$a \vee a = a$	Indempotence
$a \wedge \neg a = 0$	Law of Contradiction
$a \vee \neg a = 1$	Tautology
$\neg\neg a = a$	Double Negation
$a \wedge b = b \wedge a$	Commutativity of $\wedge$
$a \vee b = b \vee a$	Commutativity of $\vee$
$a \vee (b \vee c) = (a \vee b) \vee c$	Associativity
$a \wedge (b \wedge c) = (a \wedge b) \wedge c$	Associativity
$a \wedge (b \vee c) = (a \wedge b) \vee (a \wedge c)$	Distributivity
$a \vee (b \wedge c) = (a \vee b) \wedge (a \vee c)$	Distributivity
$a \wedge (a \vee b) = a$	Absorption
$a \vee (a \wedge b) = a$	Absorption
$a \vee (\neg a \wedge b) = a \vee b$	Absorption
$a \wedge (\neg a \vee b) = a \wedge b$	Absorption
$\neg(a \wedge b) = (\neg a) \vee (\neg b)$	De Morgan's Law
$\neg(a \vee b) = (\neg a) \wedge (\neg b)$	De Morgan's Law
$(a \wedge b) \vee (a \wedge \neg b) = a$	
$a \implies b = \neg a \vee b$	
$a \implies b = \neg(a \wedge \neg b)$	

The best way to describe the action of a logic gate is in terms of its "truth table". In a truth table we write down all the possible logical values of the inputs together with their corresponding outputs. For example, the truth table for the AND gate is given in Table 2.2. The corresponding icon for the AND gate as seen in circuit diagrams is shown in Fig. 2.2. The AND gate is logically irreversible, which means that you cannot determine unique inputs for all outputs. Specifically, if the output is 0 (i.e. FALSE), you cannot tell whether the input values where 00, 01, or 10. It "erases" some information when it acts whenever the output from the AND gate is 0.

Similarly, the truth table for the OR gate is shown in Table 2.3. The corresponding circuit icon for the OR gate is shown in Fig. 2.3. The OR gate is also logically irreversible because when its output is 1 (i.e., TRUE) it is impossible to say whether the inputs were 01, 10, or 11. Hence, again the OR gate erases some information when it acts whenever the output is a 1.

There is a variant of the OR gate, called exclusive-OR (often written "XOR" or "$\oplus$") that turns out to be very useful. The XOR gate is like the OR gate except that

Table 2.2 Truth table of AND

AND:

a	b	$a \wedge b$
0	0	0
0	1	0
1	0	0
1	1	1

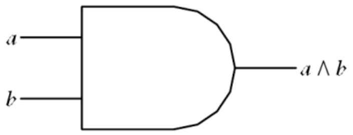


Fig. 2.2 Icon for the AND gate—a logically irreversible gate

Table 2.3 Truth table of OR

OR:

a	b	$a \vee b$
0	0	0
0	1	1
1	0	1
1	1	1

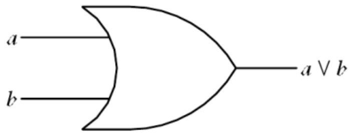


Fig. 2.3 Icon for the OR gate—a logically irreversible gate

Table 2.4 Truth table of XOR (exclusive-OR)

XOR:

a	b	$a \oplus b$
0	0	0
0	1	1
1	0	1
1	1	0

it returns 0 (i.e., FALSE) when both its inputs are 1 (i.e., TRUE). The truth table for XOR is shown in Table 2.4. The corresponding circuit icon for XOR is shown in Fig. 2.4.

2.1.3 Universal Gates: NAND and NOR

There is a special class of logic gates, called *universal* gates, any one of which is alone sufficient to express any desired computation. The possibility of such uni-

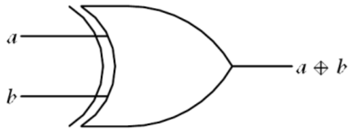


Fig. 2.4 Icon for the XOR gate—a logically irreversible gate

Table 2.5 Truth table of NAND

NAND:

a	b	$a\|b$
0	0	1
0	1	1
1	0	1
1	1	0

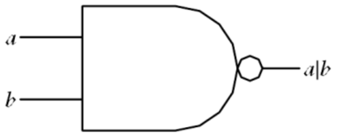


Fig. 2.5 Icon for the NAND gate—a universal gate for classical irreversible computing

versal gates accounts, in part, for the remarkable miniaturization of modern computers since computer designers need only focus on miniaturizing a single type of gate. Nowadays, the logic gates that manipulate these values are implemented using transistors, but in future computers even smaller, and faster, devices are being considered in an effort to maintain the pace of Moore's Law.

You can see why such universal gates are possible from Table 2.1. The rules in the table show that any Boolean function can be reduced to an expression involving only $\neg$ and $\wedge$ or only $\neg$ and $\vee$. Hence, any Boolean function can be computed by means of a circuit comprising NOT and AND gates, or NOT and OR gates. Nevertheless, the construction of large scale logic circuits would be greatly streamlined if manufacturers only had to use a *single* type of gate. Such a gate is said to be "universal" since from it circuits for any Boolean function can be derived. Restricting circuits to using a single type of universal gate does not necessarily lead to the smallest circuit for computing a desired Boolean function but it does allow chip manufacturers to perfect the design and manufacturing process for the universal gate, which, in practice, tends to make it easier to improve yield, reliability, and boost speed. Today, the microprocessor industry pursues this strategy by basing their circuits on the NAND ("NOT AND") gates. Mathematically, $a\text{NAND}b \equiv \neg(a \wedge b)$, often written as $a|b$, and is universal for classical irreversible computing. The truth table for the NAND gate is shown in Table 2.5: The corresponding circuit icon for the NAND gate is shown in Fig. 2.5.

To convince you that the NAND gate is truly universal, given that we already know we can compute any Boolean function in a circuit comprising only NOT and AND gates, it is sufficient to show we can obtain NOT from NAND gates and AND from NAND gates. Table 2.6 shows how to obtain $\neg a$ from $a|a$: Likewise, Table 2.7 shows we can obtain $a \wedge b$ from two $a|b$ gates. Since we proved that any logical

Table 2.6 A NOT gate can be obtained using a NAND gate since $a|a$ has precisely the same truth values as $\neg a$

NOT in terms of NAND:

a	a	$a\|a$	$\neg a$
0	0	1	1
1	1	0	0

Table 2.7 An AND gate can be obtained using only NAND gates since $a \wedge b$ has precisely the same truth values as $(a|b)|(a|b)$

AND in terms of NAND:

a	b	$a\|b$	$(a\|b)\|(a\|b)$	$a \wedge b$
0	0	1	0	0
0	1	1	0	0
1	0	1	0	0
1	1	0	1	1

proposition can be written in terms of only $\neg$ and $\wedge$, and that $\neg$ and $\wedge$ can, in turn, each be written in terms of | (NAND) we have proved that any logical proposition can be written only in terms of | (NAND) gates. This is good news for chip manufacturers because it means they need only perfect the implementation of just one type of gate, the NAND gate, to be sure that they can build a circuit that can perform any feasible computation.

There are other universal gates for classical irreversible computing including the NOR gate ("NOT OR") and the NMAJORITY gate ("NOT MAJORITY"). The NMAJORITY gate is a relatively new universal gate. It is especially interesting because it is implementable in a new transistor design and leads to highly compact circuits.

Unfortunately, logical irreversibility comes at a price. Fundamental physics dictates that energy must be dissipated when information is erased, in the amount $kT \ln 2$ per bit erased, where k is Boltzman's constant ($k = 1.3805 \times 10^{-23}$ JK^{-1}) and T is the absolute temperature (in degrees Kelvin). Thus, even if all other energy loss mechanisms were eliminated from any NAND based circuit, the circuit would still dissipate energy when it operated due to the unavoidable energy losses that occur when information is erased.

Today energy losses in NAND-based logic circuits due to logical irreversibility are dwarfed by other loss mechanisms. However, as these other loss mechanisms are tamed, someday the energy losses due solely to information erasure (in turn a consequence of using irreversible logic gates) will become the significant contribution. At this point if nothing is done, further miniaturization of computer technology will be impeded by the difficulty of removing this unwanted waste heat from deep within the irreversible circuitry.

2.1.4 Reversible Gates: NOT, SWAP, and CNOT

One way chip manufacturers can suppress the unwanted heat produced as a side effect of running irreversible logic gates is to modify their chip designs to use only *reversible* logic gates. In a reversible logic gate there is always a unique input associated with a unique output and vice versa. So reversible gates never erase any information when they act, and consequently, a computation based on reversible logic can be run forward to obtain an answer, the answer copied, and then the whole computation undone to recover all the energy expended apart from the small amount used to copy the answer at the mid-way point.

The simplest example of a reversible logic gate is the NOT gate. NOT is a 1-input/1-output gate that simply inverts the bit value it is handed. The truth table for the NOT gate is shown in Table 2.8. The circuit icon for the NOT gate is shown in Fig. 2.6. If one knows the output bit value, one can infer the input bit value unambiguously and vice versa.

A slightly more complicated example, is the 2-input/2-output SWAP gate. SWAP simply exchanges the bit values it is handed. Its truth table is shown in Table 2.9: The circuit icon for the SWAP gate is shown in Fig. 2.7. In quantum computing a circuit may not have any physical wires connecting the gates together. Instead a circuit can be merely a visual specification of a sequence of gate operations with time increasing from left to right in the circuit diagram as successive gates are applied. Consequently, in quantum computing we sometimes use a different icon for a SWAP gate (showing in Fig. 2.8, that is more suggestive that some operation (other than crossing wires) needs to occur to achieve the effect of a SWAP operation.

A reversible gate of considerable importance in quantum computing is the 2-bit controlled-NOT gate (CNOT). The truth table for CNOT is shown in Table 2.10. The circuit icon for the CNOT gate is shown in Fig. 2.9. The effect of the "controlled"-NOT gate is to flip the bit value of the second bit if and only if the first bit is set to 1.

Table 2.8 Truth table of NOT

NOT:

a	$\neg a$
0	1
1	0

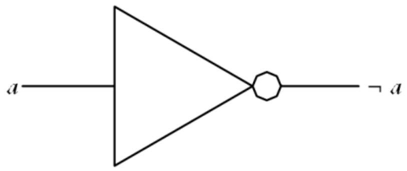


Fig. 2.6 Icon for the XOR gate—a 1-bit logically reversible gate

Table 2.9 Truth table of SWAP

SWAP:

a	b	a'	b'
0	0	0	0
0	1	1	0
1	0	0	1
1	1	1	1

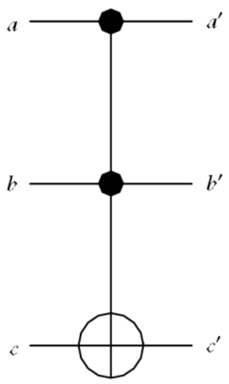


Fig. 2.11 Icon for the TOFFOLI gate also called the controlled-controlled-NOT gate. TOFFOLI is reversible and universal

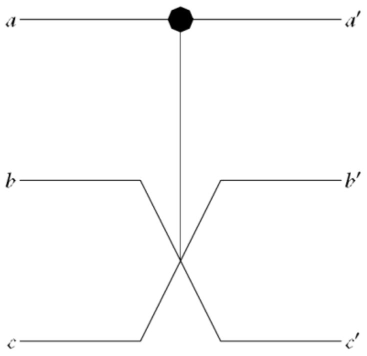


Fig. 2.12 Icon for the FREDKIN gate also called the controlled-SWAP gate. FREDKIN is reversible and universal

2.1.5.1 TOFFOLI (a.k.a. "Controlled-Controlled-NOT")

The TOFFOLI gate is also called the controlled-controlled-NOT gate since it can be understood as flipping the third input bit if, and only if, the first two input bits are both 1. In other words, the values of the first two input bits control whether the third input bit is flipped. The icon for the TOFFOLI gate is shown in Fig. 2.11.

2.1.5.2 FREDKIN (a.k.a. "Controlled-SWAP")

Another famous reversible gate is the FREDKIN (controlled-SWAP) gate. The truth table for the FREDKIN gate is: The icon for the FREDKIN gate is shown in Fig. 2.12. The FREDKIN gate can also be seen as a controlled-SWAP gate in that it swaps the values of the second and third bits, if, and only if, the first bit is set to 1.

2.1.6 Reversible Gates Expressed as Permutation Matrices

Any n-bit reversible gate must specify how to map each distinct bit string input into a distinct bit string output of the same length. Thus no two inputs are allowed to be mapped to the same output and vice versa. This ensures the mapping is reversible. Consequently, one can think of a reversible gate as encoding a specification for how to permute the 2^n possible bit strings inputs expressible in n bits. In the case of the 2-bit SWAP gate, for example, the four possible input bit strings are 00, 01, 10, 11 and these are mapped, respectively, into $00 \to 00$, $01 \to 10$, $10 \to 01$, $1 \to 11$. In the case of CNOT gate, the inputs 00, 01, 10, and 11 are mapped into 00, 01, 11, and 10 respectively. Thus a natural way to represent an n-bit reversible gate is as an array whose rows and columns are indexed by the 2^n possible bit strings expressible in n bits. The (i, j)-th element of this array is defined to be 1 if, and only if, the input bit string corresponding to the i-th row is mapped to the output bit string corresponding to the j-th column. The resulting array will contain a single 1 in each row and column and zeroes everywhere else, and will therefore be a permutation matrix. As arrays, the NOT, SWAP and CNOT gates would be described as follows:

$$\text{NOT} = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}; \qquad \text{SWAP} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix};$$
$$\text{CNOT} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \tag{2.1}$$

Likewise, the TOFFOLI gate could be represented as:

$$\text{TOFFOLI:} \quad \begin{array}{c} \\ 000 \\ 001 \\ 010 \\ 011 \\ 100 \\ 101 \\ 110 \\ 111 \end{array} \begin{array}{c} \begin{array}{cccccccc} 000 & 001 & 010 & 011 & 100 & 101 & 110 & 111 \end{array} \\ \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix} \end{array} \tag{2.2}$$

Similarly, the action of the FREDKIN gate could be represented as:

$$\text{FREDKIN:} \quad \begin{array}{c} \\ 000 \\ 001 \\ 010 \\ 011 \\ 100 \\ 101 \\ 110 \\ 111 \end{array} \begin{array}{c} \begin{array}{cccccccc} 000 & 001 & 010 & 011 & 100 & 101 & 110 & 111 \end{array} \\ \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \end{pmatrix} \end{array} \tag{2.3}$$

In fact, the matrices corresponding to classical reversible gates are always permutation matrices, i.e., $0/1$ matrices having a single 1 in each row and column, and permutation matrices are also always unitary matrices.

To calculate the effect of a reversible gate, e.g., the FREDKIN or TOFFOLI gate, on an input bit string, we simply prepare the column vector corresponding to that bit string, and then perform the usual matrix vector product operation. For example, since the FREDKIN and TOFFOLI gates act on three bits, we can imagine a column vector consisting of $2^3 = 8$ slots, one of which (the i-th say) contains a single 1, and all the other elements are 0.

$$000 \equiv \begin{pmatrix} 1\\0\\0\\0\\0\\0\\0\\0 \end{pmatrix}, \quad 001 \equiv \begin{pmatrix} 0\\1\\0\\0\\0\\0\\0\\0 \end{pmatrix}, \quad 010 \equiv \begin{pmatrix} 0\\0\\1\\0\\0\\0\\0\\0 \end{pmatrix}, \quad \ldots \quad 111 \equiv \begin{pmatrix} 0\\0\\0\\0\\0\\0\\0\\1 \end{pmatrix} \tag{2.4}$$

etc. We can calculate the effect of, e.g., the TOFFOLI gate on such an input by vector-matrix multiplication.

$$\text{TOFFOLI}|110\rangle = \begin{pmatrix} 1&0&0&0&0&0&0&0\\ 0&1&0&0&0&0&0&0\\ 0&0&1&0&0&0&0&0\\ 0&0&0&1&0&0&0&0\\ 0&0&0&0&1&0&0&0\\ 0&0&0&0&0&1&0&0\\ 0&0&0&0&0&0&0&1\\ 0&0&0&0&0&0&1&0 \end{pmatrix} \cdot \begin{pmatrix} 0\\0\\0\\0\\0\\0\\1\\0 \end{pmatrix} = \begin{pmatrix} 0\\0\\0\\0\\0\\0\\0\\1 \end{pmatrix} = |111\rangle \tag{2.5}$$

2.1.7 Will Future Classical Computers Be Reversible?

The computer industry has done a truly remarkable job at squeezing more and more computation out of fewer and fewer physical resources. For example, the energy per logical operation has decreased pretty much exponentially since the inception of the microchip, in lock step with a similar reduction in the size of transistors. As a result a given *volume* of microprocessor has, over successive generations, been made to perform exponentially more computation.

However, chip designers are now finding it harder to increase performance without incurring the need to dissipate more energy per unit area of chip. You can sense this quite directly if you spend any time working with a notebook computer on your lap. After a while you will notice it becoming quite warm. This is because the microprocessor is dissipating heat as it runs. Indeed, modern chips can consume 100

Watts or more. Since it is impractical to allow them to dissipate more power than this, this problem could ultimately stall Moore's Law.

Today, power losses arise from the non-zero electrical resistance of the conductors used inside microprocessors and some leakage of current through materials that are supposed to be insulators. This chip designers are working feverishly to lessen such losses by using fewer and fewer electrons and avoiding large voltage swings, which cuts down leakage. Once these stratagems have been played out to the maximum extent possible chip designers will have to consider various methods, such as charge recovery, to recapture energy, much like a flywheel recaptures energy in mechanical devices. Beyond this, what options remain to further reduce energy dissipation during computation?

The answer could lie in the use of classical reversible gates, such as FREDKIN and TOFFOLI gates that we discussed earlier. This is because, as Rolf Landauer showed, energy need only be dissipated when information is erased, and the minimum amount that Nature demands is $k_B T \ln 2$ per bit erased, where k_B is Blotzmann's constant and T is the temperature in degrees Kelvin. At room temperature (300 Kelvin) this is about 3×10^{-21} Joules per bit erased. Therefore, if we were to use reversible computing, the only energy that must be dissipated is related to that required to initialize the computer, or to make a permanent record on an answer, because these operations must take a memory register in one state, and reset it, regardless of what that state was, in a fixed configuration. Hence this operation is necessarily irreversible. But apart from that, in principle, it takes no energy to compute!

2.1.8 Cost of Simulating Irreversible Computations Reversibly

Today, most computing hardware employs, at its lowest level, gates that are logically irreversible. Logical irreversibility means that certain outputs from a logic gate are consistent with more than one set of inputs, preventing one from inferring a unique input for each output. For example, the logic gate $\text{AND}(x, y) = z$ that maps two input bits, x and y, into a single bit, z, is logically irreversible because an output $z = 0$ (false) could be accounted for by any of the three input pairs $(x = 0, y = 0)$, $(x = 0, y = 1)$ and $(x = 1, y = 0)$. Hence, for this particular output, the input is ambiguous and the operation is therefore logically irreversible.

It has long been known that such logical irreversibility has a thermodynamic consequence, namely, that energy must be dissipated, in the amount $k_B T \log 2$ per bit erased, whenever a logically irreversible operation is performed [299]. However, the converse of this is also true. If we were to employ only *logically reversible* gates inside our chips, then no net energy need be dissipated in performing those gate operations. The only thermodynamic cost to computing would then be the cost of creating the initial input, reading the output, and re-setting the computer.

For a computation to be logically reversibility each "step" of the computation must be logically reversible. However, the exact meaning of a "step" changes depending on the model of computation being used. For example, in the Turing machine model one step of computation is a transition of the finite control of the machine [44], which maps one "configuration" of the machine to another configuration. Likewise, in the circuit model, a step of computation is the execution of one gate of the circuit (see, e.g., [187, 494]). Thus, a *reversible Turing machine* is a machine mapping distinct input configurations to distinct output configurations, and a *reversible circuit* is a circuit comprised of gates each mapping distinct input bit patterns to distinct output bit patterns.

There are two important questions concerning reversible computing. The first is the practical question of how to find the optimal reversible circuit implementing a desired Boolean function [343, 451, 494]. This approach boils down to understanding how to implement permutations by reversible circuits, and is mainly concerned with generic functions.

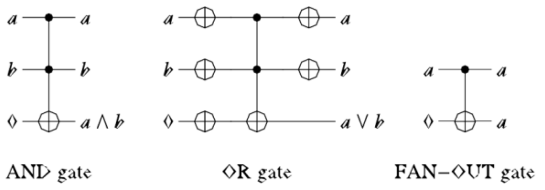
The second question concerning reversible computing is to determine with what efficiency a reversible computer can simulate an irreversible computation [44, 45, 88, 119, 302, 311, 312]. Most previous studies of this question have addressed it in the context of the Turing machine model of computation. In this paper we present a similar analysis in the context of the circuit model. In order to aid comparison we first recap the insights gleaned from these Turning machine studies.

Initially it was believed that the only way to simulate an irreversible computation on a reversible Turing machine was to keep all the intermediate calculations. Consequently, the size of the memory (i.e., "space") needed to perform the computation reversibly was proportional to the time (i.e., number of steps) of the corresponding irreversible computation. Bennett, however, [44] discovered that the history of a reversible computation could be cleared in a reversible fashion, leaving only the input and the output in memory, and recording the configuration of certain *checkpoints* of the irreversible computation. This reduced the space needed to simulate an irreversible computation reversibly but at the expense of increasing the time of the reversible computation. Specifically, in [45] Bennett proposed a method which uses time $S\,T^{\log 3}$ and space $S \log T$, when the irreversible computation uses T time and S space. In this case the space complexity of the simulation is S^2 in the worst case. Later it was shown that it is possible to have a reversible simulation in space $O(S)$ but at the cost of requiring the simulation to run in exponential time [302]. The best tradeoff for reversible simulation of an irreversible computation was provided by Li [312]. It uses time $\Theta(T^{1+\varepsilon}/S^{\varepsilon})$ and space $\Theta(c(\varepsilon)S[1+\log(T/S)])$, for any $\varepsilon > 0$, where $c(\varepsilon) \approx \varepsilon 2^{1/\varepsilon}$. Similarly, in [119] it is shown that any *nondeterministic* Turing machine running in space S can be simulated by a reversible machine using space $O(S^2)$.

The foregoing studies of the efficiency with which a reversible computer can simulate an irreversible computation were all based on the deterministic or nondeterministic Turing machine models. As best we can tell there has been no similar direct study in the literature based on the circuit model of computation. This is the main contribution of our paper.

Toffoli and Fredkin [187, 494] performed some of the first systematic studies of reversible circuits. Toffoli showed, for example, that the reversible basis consisting

Fig. 2.15 Reversible simulation of classical gates



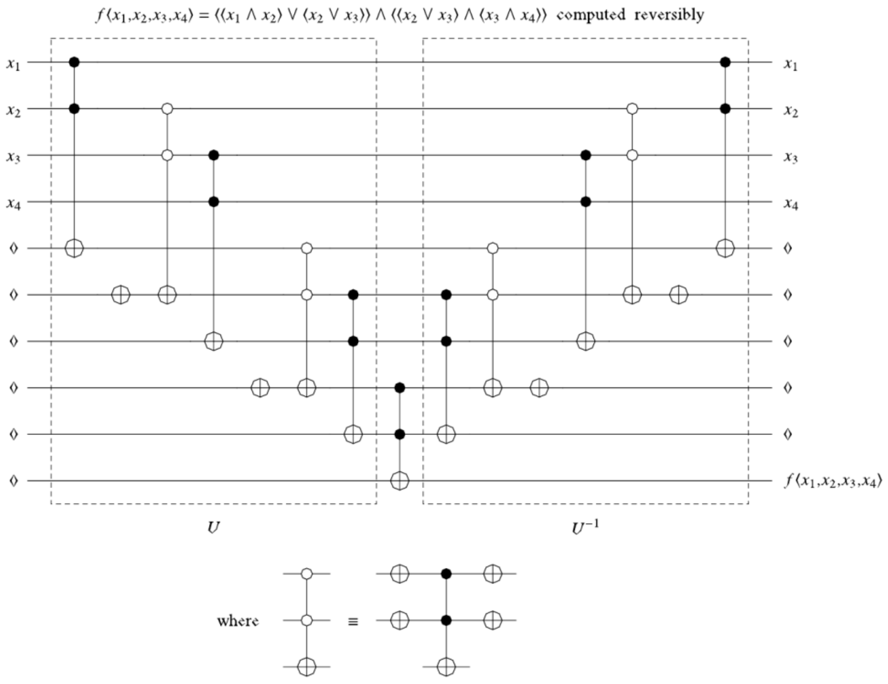

f(x1,x2,x3,x4) = ((x1 ∧ x2) ∨ (x2 ∨ x3)) ∧ ((x2 ∨ x3) ∧ (x3 ∧ x4)) computed reversibly
x1
x2
x3
x4
0
f(x1,x2,x3,x4)
U
U−1
where
≡

Fig. 2.16 Synthesis via reversible substitution

2.2.1 Can All Boolean Circuits Be Simulated Reversibly?

The constructions of Fig. 2.15 suggest a simple (naive) method for simulating any Boolean (irreversible) circuit: simply replace each irreversible gate in the circuit with its reversible counterpart. Figure 2.16 shows an example of this method.

However, this naive method is hardly efficient and we now present a better scheme. Before we begin, we define some useful terminology. A synchronous circuit is one in which all paths from the inputs to any gate have the same length. Synchronous circuits may have delay (identity) gates, and gates at level m get inputs from gates at level $m - 1$. Thus, without loss of generality, we can assume that our desired irreversible circuit is synchronous. For a Boolean circuit, the *size* is the total number of gates, the *depth* is the number of levels, and the *width* is the maximum number of gates in any level.

The following procedure shows how to create a reversible circuit that simulates and irreversible circuit while making substantial savings in the number of ancillae used.

- First simulate the gates in the first-half levels.
- Keep the results of the gates in the level $d/2$ separately.
- Clean up the ancillae bits.
- Use them to simulate the gates in the second-half levels.
- After computing the output, clean up the ancillae bits.
- Clean up the result of the level $d/2$.

Note This method needs roughly *half* the number of ancillae used by the previous (naive) method. Figure 2.16 shows the circuit of this procedure.

By applying the above procedure recursively, on a circuit of size t, depth d, and width w we obtain the following recursive relations for S, the size, and A, the number of the ancillae needed:

$$S(t) \leq 6S(t/2) + O(1),$$
$$A(d) \leq A(d/2) + w + 1.$$

Solving these recursion relations leads to the following result.

Efficiency of Reversible Simulation *Any irreversible computation* (*in the synchronous form*) *having t gates, depth d, and width w, can be simulated by a reversible circuit having $O(t^{2.58})$ gates, and at most $(w+1)\log d + O(1)$ ancillae.*

Thus, most of the irreversible computations going on inside your notebook computer could, in principle, be implemented using reversible logic gates, which in turn need no *net* energy to run apart from any operations that require erasure of information, such as overwriting a memory register to make a copy of an answer! This is surprise to many people because their perception is that computers are making something new. But in reality, they don't. They just take the known information given as input and re-arrange it. The vast majority of the operations employed along the way can be done reversibly, and hence, don't generate any more information in their output than they had in their input. There is no truly creative act as such. As Pablo Picasso once said, "Computers are useless—they only give answers!"

2.3 Quantum Logic Gates

Now that we have looked at classical irreversible and classical reversible gates, we have a better context in which to appreciate the benefits of quantum gates.

Just as any classical computation can be broken down into a sequence of classical logic gates that act on only a few classical bits at a time, so too can any quantum computation can be broken down into a sequence of quantum logic gates that act on only a few qubits at a time. The main difference is that whereas classical logic gates manipulate the classical bit values, 0 or 1, quantum gates can manipulate arbitrary multi-partite quantum states including arbitrary superpositions of the computational basis states, which are frequently also entangled. Thus the logic gates of quantum computation are considerably more varied than the logic gates of classical computation.

2.3.1 From Quantum Dynamics to Quantum Gates

The physical phenomena used to achieve the desired manipulation of a quantum state can be very varied. For example, if qubits are encoded in particles having quantum mechanical spin, the logic is effected by spin-manipulation brought about by varying an applied magnetic field at various orientations. Or if the qubit is encoded in an internal excitation state of an ion, the gate operation can be achieved by varying the time a laser beam is allowed to irradiate the ion or by varying the wavelength of that laser light.

As any quantum gate must be implemented physically as the quantum mechanical evolution of an isolated quantum system, the transformation it achieves is governed by Schrödinger's equation, $i\hbar\partial|\psi\rangle/\partial t = \mathcal{H}|\psi\rangle$, where $\mathcal{H}$ is the Hamiltonian, specifying the physical fields and forces at work. Thus, the unitary matrices describing quantum gates are related to the physical processes by which they are achieved via the equation $U = \exp(-i\mathcal{H}t/\hbar)$. Here $\mathcal{H}$ is the Hamiltonian which specifies the interactions that are present in the physical system.

As we saw in Chap. 1, the quantum mechanical evolution induced by this equation is unitary provided no measurements are made, and no unwanted stray interactions occur with the environment. In this case, starting from some initial state, $|\psi(0)\rangle$, the quantum system will evolve, in time t, into the state $|\psi(t)\rangle = \exp(-i\mathcal{H}t/\hbar)|\psi(0)\rangle = U|\psi(0)\rangle$ where U is some unitary matrix. Thus the evolution, in time t, of an isolated quantum system is described by a unitary transformation of an initial state $|\psi(0)\rangle$ to a final state $|\psi(t)\rangle = U|\psi(0)\rangle$. This means that a quantum logic gate acting on an isolated quantum computer, will transform that state unitarily up until the point at which an observation is made. Hence, quantum logic gates are described, mathematically, by unitary matrices, and their action is always logically reversible.

The parallels between classical reversible gates and quantum gate were not lost the early quantum computer pioneers Richard Feynman and David Deutsch. They recognized that since the matrices corresponding to reversible (classical) gates were permutation matrices, they were also unitary matrices and hence could be interpreted as operators that evolved some initial quantum state representing the input to a gate into some final quantum state representing its output in accordance with Schrödinger's equation. Thus, the closest classical analogs to quantum logic gates are the classical reversible gates such as the NOT, SWAP, CNOT, TOFFOLI and FREDKIN. However, whereas the repertoire of gates available in classical reversible computing is limited to the unitary gates whose matrix representations correspond to permutation matrices, in deterministic quantum computing any gate is allowed whose matrix is unitary whether or not it is also a permutation matrix.

2.3.2 Properties of Quantum Gates Arising from Unitarity

The essential properties of quantum logic gates flow immediately from that fact that they are described by unitary matrices. A matrix, U, is unitary if and only if its inverse[4] equals its conjugate transpose, i.e., if and only if $U^{-1} = U^\dagger$. If U is unitary the following facts hold:

- $U^\dagger$ is unitary.
- U^{-1} is unitary.
- $U^{-1} = U^\dagger$ (which is the criterion for determining unitarity).
- $U^\dagger U = \mathbb{1}$
- $|\det(U)| = 1$.
- The columns (rows) of U form an orthonormal set of vectors.
- For a fixed column, $\sum_{i=1}^{2^n} |U_{ij}|^2 = 1$.
- For a fixed row, $\sum_{j=1}^{2^n} |U_{ij}|^2 = 1$.
- $U = \exp(i\mathcal{H})$ where $\mathcal{H}$ is an hermitian matrix, i.e., $\mathcal{H} = \mathcal{H}^\dagger$.

The fact that, for any quantum gate U, $U^\dagger U = \mathbb{1}$ ensures that we can always undo a quantum gate, i.e., that a quantum gate is logically reversible. Moreover, that fact that for a fixed column $\sum_{i=1}^{2^n} |U_{ij}|^2 = 1$ and for a fixed row $\sum_{j=1}^{2^n} |U_{ij}|^2 = 1$ guarantee that if you start with a properly normalized quantum state and act upon it with a quantum gate, then you will end up with a properly normalized quantum state. Thus, there are no probability "leaks". The fact that it is the magnitude $|\det(U)|$ that is constrained to be unity means that the constraint on the determinant can be satisfied with $\det(U) = \pm 1$ or $\pm i$. Thus the elements of a general unitary matrix are generically allowed to be complex numbers.

2.4 1-Qubit Gates

2.4.1 Special 1-Qubit Gates

2.4.1.1 Pauli Spin Matrices

For single qubits, the "Pauli matrices" ($\mathbb{1}$, X, Y, Z), which happen to be both hermitian and unitary, are of special interest since any 1-qubit Hamiltonian can always be

[4]If A and B are two matrices B is the inverse of A when $A.B = \mathbb{1}$ where $\mathbb{1}$ is the identity matrix, i.e., a matrix having only ones down the main diagonal.

written as a weighted sum of the Pauli matrices:

$$\mathbb{1} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \qquad X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \qquad Y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}, \qquad Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \tag{2.6}$$

Some common forms for Hamiltonians that arise in practice are $\mathcal{H} = Z^{(1)}Z^{(2)}$ (the Ising interaction) and $\mathcal{H} = X^{(1)} \otimes X^{(2)} + Y^{(1)} \otimes Y^{(2)}$ (the XY interaction) and $\mathcal{H} = 2X^{(1)} \otimes X^{(2)} + Y^{(1)} \otimes Y^{(2)}$ where the parenthetical superscripts labels which of two qubits the operator acts upon.

2.4.1.2 NOT Gate

The Pauli X matrix is synonymous with the classical (reversible) NOT gate, i.e.,

$$X \equiv \text{NOT} = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \tag{2.7}$$

Thus, it is not surprising that X negates the computational basis states $|0\rangle$ and $|1\rangle$, correctly as these correspond to the classical bits, 0 and 1, respectively. Specifically, we have:

$$X|0\rangle = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \cdot \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \end{pmatrix} = |1\rangle \tag{2.8}$$

$$X|1\rangle = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \cdot \begin{pmatrix} 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \end{pmatrix} = |0\rangle \tag{2.9}$$

2.4.1.3 $\sqrt{\text{NOT}}$ Gate

One of the simplest 1-qubit non-classical gates one can imagine is a fractional power the of NOT gate, such as $\sqrt{\text{NOT}}$:

$$\sqrt{\text{NOT}} = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}^{\frac{1}{2}} = \begin{pmatrix} \frac{1}{2} + \frac{i}{2} & \frac{1}{2} - \frac{i}{2} \\ \frac{1}{2} - \frac{i}{2} & \frac{1}{2} + \frac{i}{2} \end{pmatrix} \tag{2.10}$$

The $\sqrt{\text{NOT}}$ gate has the property that a repeated application of the gate, i.e., $\sqrt{\text{NOT}} \cdot \sqrt{\text{NOT}}$, is equivalent to the NOT operation, but a single application results in a quantum state that neither corresponds to the classical bit 0, or the classical bit 1. So $\sqrt{\text{NOT}}$ it is the first truly non-classical gate we have encountered.

$$|0\rangle \xrightarrow{\sqrt{\text{NOT}}} \left(\frac{1}{2} + \frac{i}{2}\right)|0\rangle + \left(\frac{1}{2} - \frac{i}{2}\right)|1\rangle \xrightarrow{\sqrt{\text{NOT}}} |1\rangle \tag{2.11}$$

$$|1\rangle \xrightarrow{\sqrt{\text{NOT}}} \left(\frac{1}{2} - \frac{i}{2}\right)|0\rangle + \left(\frac{1}{2} + \frac{i}{2}\right)|1\rangle \xrightarrow{\sqrt{\text{NOT}}} |0\rangle \tag{2.12}$$

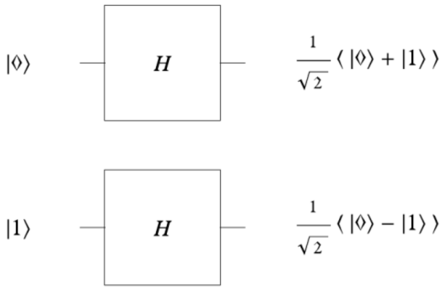


Fig. 2.18 The icon for the 1-qubit Walsh-Hadamard gate, H and its affect on computational basis states

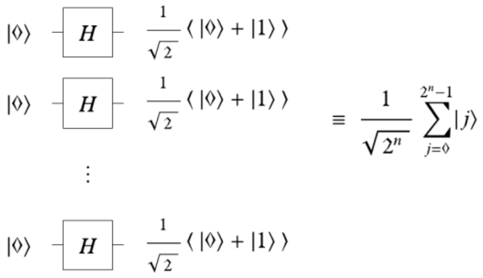


Fig. 2.19 By applying n H gates independently to n qubits, all prepared initially in state $|0\rangle$, we can create an n-qubit superposition whose component eigenstates are the binary representation of all the integers in the range 0 to $2^n - 1$. Thus, a superposition containing *exponentially* many terms can be prepared using only a *polynomial* number of operations. This trick is used in a great many quantum algorithms to load a quantum memory register efficiently with an equally weighted superposition of all the numbers it can contain

When the Hadamard gate H acts on a computational basis state $|x\rangle$ it transforms the input according to $H|x\rangle = \frac{1}{\sqrt{2}}(|0\rangle + (-1)^x|1\rangle)$.

The Hadamard is one of the unsung heroes of quantum computing. It is a deceptively simple looking gate but it harbors a remarkable property that, if you think about it, turns out to be of vital importance to quantum computing. If you prepare n qubits each in the state $|0\rangle$ and you apply to each qubit, in parallel, its own Hadamard gate, then, as shown in Fig. 2.19, the state produced is an equal superposition of all the integers in the range 0 to $2^n - 1$.

$$H|0\rangle \otimes H|0\rangle \otimes \cdots \otimes H|0\rangle = \frac{1}{\sqrt{2^n}} \sum_{j=0}^{2^n-1} |j\rangle \tag{2.20}$$

where $|j\rangle$ is the computational basis state indexed by the binary number that would correspond to the number j in base-10 notation. For example, in a 7-qubit register the state "$|19\rangle$" corresponds to the computational basis state $|0010011\rangle$. The first two bits (00) are padding to make the binary number 7 bits in length, and 10011_2 (i.e., 10011 in base 2) corresponds to 19_{10} (i.e. 19 in base-10).

The utility of the Hadamard gate derives from that fact that by applying, in parallel, a separate Hadamard gate to each of n qubits, each initially in the state $|0\rangle$, we can create an n-qubit superposition containing 2^n component eigenstates. These eigenstates represent all the possible bit strings one can write using n bits. The importance of this capability is often overlooked. But, in reality, it is one of the most important tricks of quantum computing as it gives is the ability to load exponentially many indices into a quantum computer using only polynomially many operations. Had Nature been unkind, and had we had to enter the different bit-strings individually, as we do in classical computing, then quantum computing would have had far less potential for breakthroughs in computational complexity.

2.4.2 *Rotations About the x-, y-, and z-Axes*

Having seen a couple of examples of special quantum logic gates (i.e., $\sqrt{\text{NOT}}$ and H) we next turn to the question of what is the most general kind of quantum gate for a single qubit. To address this, we must first introduce the family of quantum gates that perform rotations about the three mutually perpendicular axes of the Bloch sphere.

A single qubit pure state is represented by a point on the surface of the Bloch sphere. The effect of a single qubit gate that acts in this state is to map it to some other point on the Bloch sphere. The gates that rotate states around the x-, y-, and z-axes are of special significance since we will be able to decompose an arbitrary 1-qubit quantum gate into sequences of such rotation gates.

First, let's fix our reference frame with respect to which arbitrary single qubit pure states is defined. We choose three mutually perpendicular axes, x-, y-, and z-, or equivalently, three polar coordinates, a radius r (which is unity for all points on the surface of the Bloch sphere) and two angles θ (the latitude, measured monotonically from the North pole to the South pole over the interval $0 \leq \theta \leq \pi$) and ϕ the longitude (measured monotonically as we rotate around the z-axis in a clockwise fashion. So any point on the surface of the Bloch sphere can be specified using its (x, y, z) coordinates or, equivalently, its (r, θ, ϕ) coordinates. Right? Well actually not quite right since a general qubit state also must specify an overall phase factor. But let's ignore this for now. These two coordinate systems are related via the equations:

$$x = r\sin(\theta)\cos(\phi) \tag{2.21}$$

$$y = r\sin(\theta)\sin(\phi) \tag{2.22}$$

$$z = r\cos(\theta) \tag{2.23}$$

So what are the quantum gates that rotate this state about the x-, y-, or z-axes? We claim that these gates, illustrated in Figs. 2.20, 2.21, and 2.22, can be built from the Pauli X, Y, Z, matrices, and the fourth Pauli matrix, $\mathbb{1}$, can be used to achieve a global overall phase shift. Specifically, let's define the following unitary matrices, $R_x(\theta)$, $R_y(\theta)$, $R_z(\theta)$, and Ph from Hamiltonians chosen to be, respectively, the four Pauli matrices, X, Y, Z, and I (the identity matrix). That is, we have:

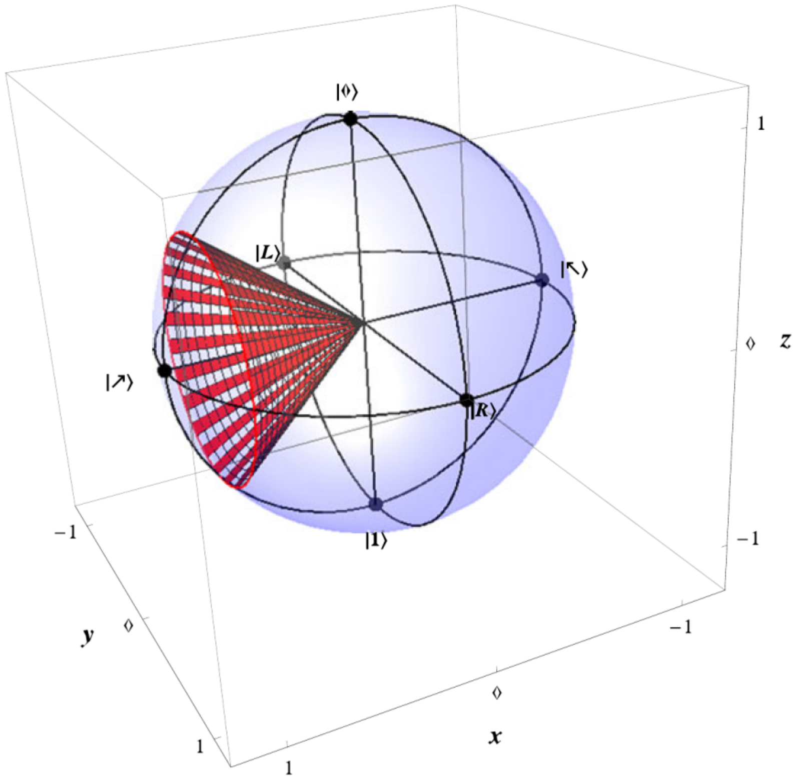


Fig. 2.20 An $R_x(\theta)$ gate maps a state $|\psi\rangle$ on the surface of the Bloch sphere to a new state, $R_x(\theta)|\psi\rangle$, represented by the point obtained by rotating a radius vector from the center of the Bloch sphere to $|\psi\rangle$ through an angle $\frac{\theta}{2}$ around the x-axis. Note that a rotation of 4π is needed to return to the original state

$$R_x(\alpha) = \exp(-i\alpha X/2) = \begin{pmatrix} \cos(\frac{\alpha}{2}) & -i\sin(\frac{\alpha}{2}) \\ -i\sin(\frac{\alpha}{2}) & \cos(\frac{\alpha}{2}) \end{pmatrix} \tag{2.24}$$

$$R_y(\alpha) = \exp(-i\alpha Y/2) = \begin{pmatrix} \cos(\frac{\alpha}{2}) & -\sin(\frac{\alpha}{2}) \\ \sin(\frac{\alpha}{2}) & \cos(\frac{\alpha}{2}) \end{pmatrix} \tag{2.25}$$

$$R_z(\alpha) = \exp(-i\alpha Z/2) = \begin{pmatrix} e^{-i\alpha/2} & 0 \\ 0 & e^{i\alpha/2} \end{pmatrix} \tag{2.26}$$

$$Ph(\delta) = e^{i\delta}\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \tag{2.27}$$

Consider the gate $R_z(\alpha)$. Let's see how this gate transforms an arbitrary single qubit state $|\psi\rangle = \cos(\frac{\theta}{2})|0\rangle + e^{i\phi}\sin(\frac{\theta}{2})|1\rangle$.

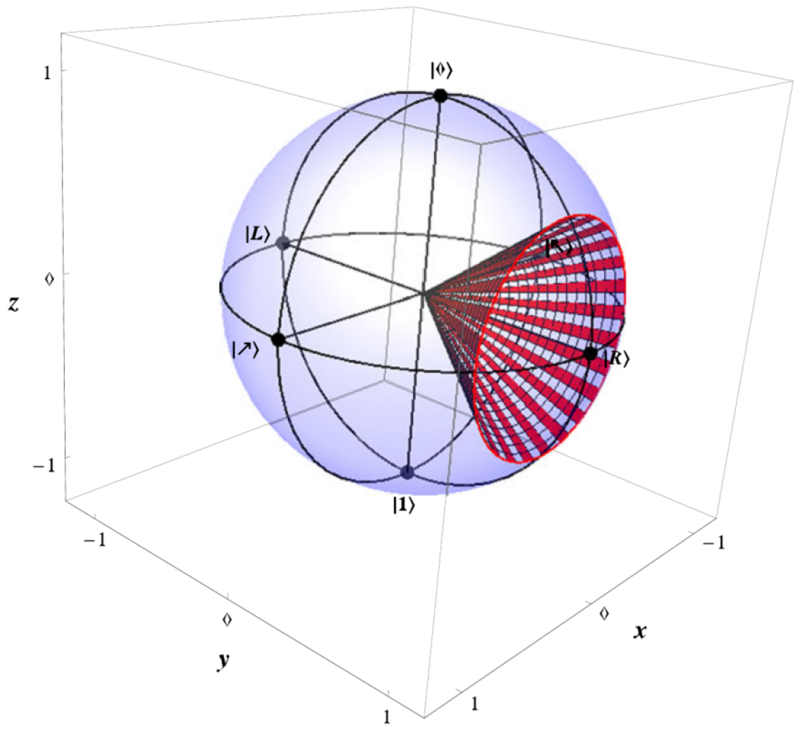


Fig. 2.21 An $R_y(\theta)$ gate maps a state $|\psi\rangle$ on the surface of the Bloch sphere to a new state, $R_y(\theta)|\psi\rangle$, represented by the point obtained by rotating a radius vector from the center of the Bloch sphere to $|\psi\rangle$ through an angle $\frac{\theta}{2}$ around the y-axis. Note that a rotation of 4π is needed to return to the original state

$$\begin{aligned}R_z(\alpha)|\psi\rangle &= \begin{pmatrix} e^{-i\alpha/2} & 0 \\ 0 & e^{i\alpha/2} \end{pmatrix} \cdot \begin{pmatrix} \cos\left(\frac{\theta}{2}\right) \\ e^{i\phi}\sin\left(\frac{\theta}{2}\right) \end{pmatrix} \\ &= \begin{pmatrix} e^{-i\alpha/2}\cos\left(\frac{\theta}{2}\right) \\ e^{i\alpha/2}e^{i\phi}\sin\left(\frac{\theta}{2}\right) \end{pmatrix} \\ &= e^{-i\alpha/2}\cos\left(\frac{\theta}{2}\right)|0\rangle + e^{i\alpha/2}e^{i\phi}\sin\left(\frac{\theta}{2}\right)|1\rangle \qquad (2.28)\end{aligned}$$

We are free to multiply this state by any overall phase factor we please since for any quantum state $|\chi\rangle$, the states $|\chi\rangle$ and $e^{i\gamma}|\chi\rangle$ are indistinguishable. So let's multiply by an overall phase factor of $\exp(i\alpha/2)$, which gives us the state:

$$R_z(\alpha)|\psi\rangle \equiv \cos\left(\frac{\theta}{2}\right)|0\rangle + e^{i(\phi+\alpha)}\sin\left(\frac{\theta}{2}\right) \qquad (2.29)$$

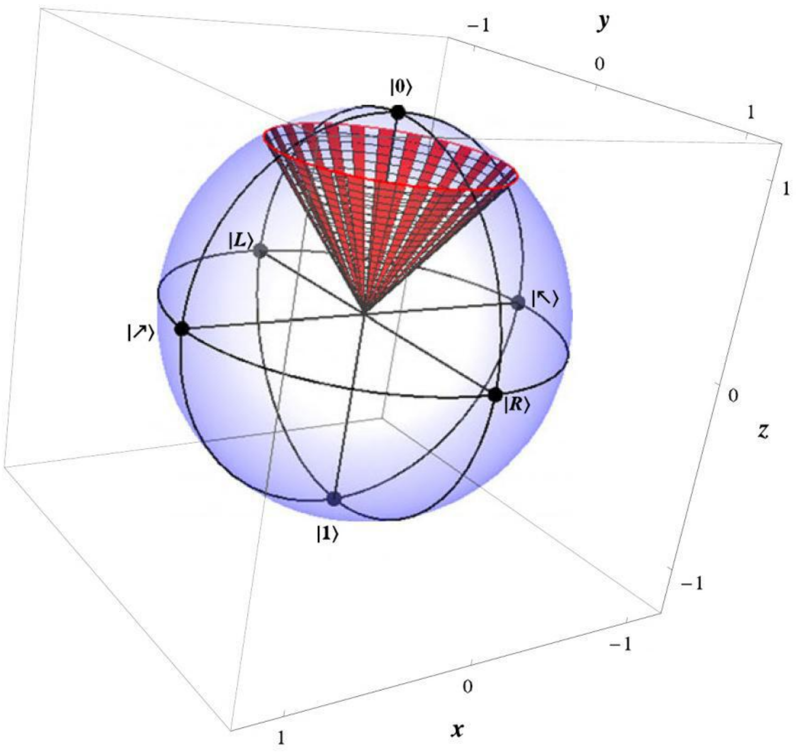


Fig. 2.22 An $R_z(\theta)$ gate maps a state $|\psi\rangle$ on the surface of the Bloch sphere to a new state, $R_z(\theta)|\psi\rangle$, represented by the point obtained by rotating a radius vector from the center of the Bloch sphere to $|\psi\rangle$ through an angle $\frac{\theta}{2}$ around the z-axis. Note that a rotation of 4π is needed to return to the original state

where $\equiv$ is to be read as "equal up to an unimportant arbitrary overall phase factor". Hence the action of the $R_z(\alpha)$ gate on $|\psi\rangle$ has been to advance the angle ϕ by α and hence rotate the state about the z-axis through angle α. This is why we call $R_z(\alpha)$ a z-rotation gate. We leave it to the exercises for you to prove that $R_x(\alpha)$ and $R_y(\alpha)$ rotate the state about the x- and y-axes respectively.

Rotations on the Bloch sphere do not conform to commonsense intuitions about rotations that we have learned from our experience of the everyday world. In particular, usually, a rotation of 2π radians (i.e., 360 degrees) of a solid object about any axis, restores that object to its initial orientation. However, this is not true of rotations on the Bloch sphere! When we rotate a quantum state through 2π on the Bloch sphere we don't return it to its initial state. Instead we pick up a phase factor. To see this, let's compute the effect of rotating our arbitrary single qubit pure state, $|\psi\rangle$ about the z-axis through 2π radians. We have:

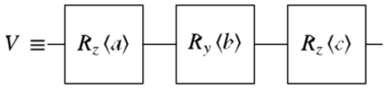


Fig. 2.24 Any 1-qubit special unitary gate can be decomposed into a rotation about the z-axis, the y-axis, and the z-axis

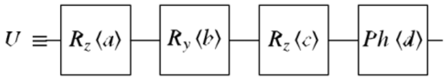


Fig. 2.25 Any 1-qubit unitary gate can be decomposed into a rotation about the z-axis, the y-axis, the z-axis, followed by a phase shift

where α and β are arbitrary complex numbers that satisfy the determinant equation $\det(V) = \alpha\bar{\alpha} - \beta(-\bar{\beta}) = |\alpha|^2 + |\beta|^2 = 1$. This equation can be satisfied by picking $\alpha = e^{i\mu}\cos(\theta/2)$, and $\beta = e^{i\xi}\sin(\theta/2)$. This means we can also write the matrix for V as:

$$\begin{aligned} V &= \begin{pmatrix} \alpha & -\bar{\beta} \\ \beta & \bar{\alpha} \end{pmatrix} \quad \text{with } \alpha \to e^{i\mu}\cos(\theta/2) \text{ and } \beta \to e^{i\xi}\sin(\theta/2) \\ &= \begin{pmatrix} e^{i\mu}\cos(\theta/2) & -e^{-i\xi}\sin(\theta/2) \\ e^{i\xi}\sin(\theta/2) & e^{-i\mu}\cos(\theta/2) \end{pmatrix} \end{aligned} \tag{2.38}$$

But this matrix can also be obtained as the product of the three gates $R_z(a) \cdot R_y(b) \cdot R_z(c)$ with $a \to -(\mu - \xi)$, $b \to \theta$, and $c \to -(\mu + \xi)$.

$$\begin{aligned} R_z(a) \cdot R_y(b) \cdot R_z(c) &= \begin{pmatrix} e^{-\frac{ia}{2}-\frac{ic}{2}}\cos\left(\frac{b}{2}\right) & -e^{\frac{ic}{2}-\frac{ia}{2}}\sin\left(\frac{b}{2}\right) \\ e^{\frac{ia}{2}-\frac{ic}{2}}\sin\left(\frac{b}{2}\right) & e^{\frac{ia}{2}+\frac{ic}{2}}\cos\left(\frac{b}{2}\right) \end{pmatrix} \\ &\qquad \text{with } a \to -(\mu - \xi), b \to \theta, \text{ and } c \to -(\mu + \xi) \\ &= \begin{pmatrix} e^{i\mu}\cos(\theta/2) & -e^{-i\xi}\sin(\theta/2) \\ e^{i\xi}\sin(\theta/2) & e^{-i\mu}\cos(\theta/2) \end{pmatrix} = V \end{aligned} \tag{2.39}$$

Thus, any 1-qubit special unitary gate V can be decomposed into the form $R_z(a) \cdot R_y(b) \cdot R_z(c)$ as shown in Fig. 2.24. Hence, any 1-qubit unitary gate, U can be decomposed into the form:

$$U \equiv R_z(a) \cdot R_y(b) \cdot R_z(c) \cdot Ph(d) \tag{2.40}$$

as shown in Fig. 2.25.

2.4.4 Decomposition of R_x Gate

Lest it seem peculiar that we can achieve an arbitrary 1-qubit gate without performing a rotation about the x-axis, we note that it is possible to express rotations about the x-axis purely in terms of rotations about the y- and z-axes. Specifically, we have the identities:

$$\begin{aligned} R_x(\theta) = \exp(-i\theta X/2) &= \begin{pmatrix} \cos(\frac{\theta}{2}) & i\sin(\frac{\theta}{2}) \\ i\sin(\frac{\theta}{2}) & \cos(\frac{\theta}{2}) \end{pmatrix} \\ &\equiv R_z(-\pi/2) \cdot R_y(\theta) \cdot R_z(\pi/2) \\ &\equiv R_y(\pi/2) \cdot R_z(\theta) \cdot R_y(-\pi/2) \end{aligned} \tag{2.41}$$

2.5 Controlled Quantum Gates

To perform non-trivial computations it is often necessary to change the operation applied to one set of qubits depending upon the values of some other set of qubits. The gates that implement these "if-then-else" type operations are called *controlled* gates. Some examples of controlled gates that appeared earlier in this chapter are CNOT (controlled-NOT), FREDKIN (controlled-SWAP), and TOFFOLI (controlled-controlled-NOT). The justification for calling these gates "controlled" stems from their effect on the computational basis states. For example, CNOT transforms the computational basis states such that the second qubit is negated if and only if the first qubit is in state $|1\rangle$.

$$|00\rangle \xrightarrow{\text{CNOT}} |00\rangle \tag{2.42}$$

$$|01\rangle \xrightarrow{\text{CNOT}} |01\rangle \tag{2.43}$$

$$|10\rangle \xrightarrow{\text{CNOT}} |11\rangle \tag{2.44}$$

$$|11\rangle \xrightarrow{\text{CNOT}} |10\rangle \tag{2.45}$$

Hence, the value of the second qubit (called the "target" qubit) is *controlled* by the first qubit (called the "control" qubit).

Likewise, under the action of the FREDKIN gate the second and third qubits are swapped if and only if the first qubit is in state $|1\rangle$. So the FREDKIN gate performs a controlled-SWAP operation.

$$|000\rangle \xrightarrow{\text{FREDKIN}} |000\rangle \tag{2.46}$$

$$|001\rangle \xrightarrow{\text{FREDKIN}} |001\rangle \tag{2.47}$$

$$|010\rangle \xrightarrow{\text{FREDKIN}} |010\rangle \tag{2.48}$$

$$|011\rangle \xrightarrow{\text{FREDKIN}} |011\rangle \tag{2.49}$$

$$|100\rangle \xrightarrow{\text{FREDKIN}} |100\rangle \tag{2.50}$$

$$|101\rangle \xrightarrow{\text{FREDKIN}} |110\rangle \tag{2.51}$$

$$|110\rangle \xrightarrow{\text{FREDKIN}} |101\rangle \tag{2.52}$$

$$|111\rangle \xrightarrow{\text{FREDKIN}} |111\rangle \tag{2.53}$$

It is also possible to have controlled gates with multiple control qubits and multiple target qubits. The action of the TOFFOLI gate is to negate the third qubit (i.e., the target qubit) if and only if the first two qubits (the control qubits) are in state $|11\rangle$. Thus the TOFFOLI gate has two control qubits and one target qubit.

$$|000\rangle \xrightarrow{\text{TOFFOLI}} |000\rangle \tag{2.54}$$

$$|001\rangle \xrightarrow{\text{TOFFOLI}} |001\rangle \tag{2.55}$$

$$|010\rangle \xrightarrow{\text{TOFFOLI}} |010\rangle \tag{2.56}$$

$$|011\rangle \xrightarrow{\text{TOFFOLI}} |011\rangle \tag{2.57}$$

$$|100\rangle \xrightarrow{\text{TOFFOLI}} |100\rangle \tag{2.58}$$

$$|101\rangle \xrightarrow{\text{TOFFOLI}} |101\rangle \tag{2.59}$$

$$|110\rangle \xrightarrow{\text{TOFFOLI}} |111\rangle \tag{2.60}$$

$$|111\rangle \xrightarrow{\text{TOFFOLI}} |110\rangle \tag{2.61}$$

Now all this is very well, but aren't CNOT, FREDKIN and TOFFOLI not just classical reversible gates? Well yes they are! But in addition they are also quantum gates because the transformations they perform (i.e., permutations of computational basis states) also happen to be unitary. But indeed, controlled *quantum* gates can be far more sophisticated than controlled classical gates. For example, the natural quantum generalization of the controlled-NOT gate is the controlled-U gate:

$$\text{controlled-}U \equiv \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & U_{11} & U_{12} \\ 0 & 0 & U_{21} & U_{22} \end{pmatrix} \tag{2.62}$$

where $U = \left(\begin{smallmatrix} U_{11} & U_{12} \\ U_{21} & U_{22} \end{smallmatrix}\right)$ is an arbitrary 1-qubit gate.

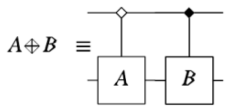


Fig. 2.26 The quantum circuit corresponding to a gate that performs different control actions according to whether the top qubit is $|0\rangle$ or $|1\rangle$

2.5.1 Meaning of a "Controlled" Gate in the Quantum Context

If we are using CNOT, FREDKIN or TOFFOLI gates within the context of classical reversible computing their inputs are only ever classical bits. Hence, there is no problem imagining *reading* each control bit to determine what action to perform on the target bit. But if we use these gates in the context of quantum computing, where they may be required to act on arbitrary superposition states, we ought to question whether it continues to make sense to speak of "controlled" gates because, in the quantum case, the act of reading the control qubit will, in general, perturb it.

The answer is that *we do not need to read control bits* during the application of a controlled quantum gate! Instead if a controlled quantum gate acts on a superposition state *all* of the control actions are performed in parallel to a degree commensurate with the amplitude of the corresponding control qubit eigenstate within the input superposition state.

For example, suppose A and B are a pair of unitary matrices corresponding to arbitrary 1-qubit quantum gates. Then the gate defined by their direct sum:

$$A \oplus B = \begin{pmatrix} A & 0 \\ 0 & B \end{pmatrix} = \begin{pmatrix} A_{11} & A_{12} & 0 & 0 \\ A_{21} & A_{22} & 0 & 0 \\ 0 & 0 & B_{11} & B_{12} \\ 0 & 0 & B_{21} & B_{22} \end{pmatrix} \tag{2.63}$$

performs a "controlled" operation in the following sense. If the first qubit is in state $|0\rangle$ then the operation A is applied to the second qubit. Conversely, if the first qubit is in state $|1\rangle$ then the operation B is applied to the second qubit. And if the control qubit is some superposition of $|0\rangle$ and $|1\rangle$ then both control actions are performed to some degree. The quantum circuit for such a gate is shown in Fig. 2.26. Don't believe me? Let's work it out explicitly.

If the first qubit is in state $|0\rangle$ we can write the input as a state of the form $|0\rangle(a|0\rangle + b|1\rangle)$, and if the first qubit is in state $|1\rangle$ we write the input as a state of the form $|1\rangle(a|0\rangle + b|1\rangle)$. For the first case, when the gate acts we therefore obtain:

$$\begin{aligned}
(A \oplus B)(|0\rangle \otimes (a|0\rangle + b|1\rangle)) &= \begin{pmatrix} A_{11} & A_{12} & 0 & 0 \\ A_{21} & A_{22} & 0 & 0 \\ 0 & 0 & B_{11} & B_{12} \\ 0 & 0 & B_{21} & B_{22} \end{pmatrix} \cdot \begin{pmatrix} a \\ b \\ 0 \\ 0 \end{pmatrix} \\
&= \begin{pmatrix} aA_{11} + bA_{12} \\ aA_{21} + bA_{22} \\ 0 \\ 0 \end{pmatrix} \\
&= (aA_{11} + bA_{12})|00\rangle + (aA_{21} + bA_{22})|01\rangle \\
&= |0\rangle \otimes A(a|0\rangle + b|1\rangle)
\end{aligned} \tag{2.64}$$

Likewise, for the second case, when the gate acts on an input of the form $|1\rangle \otimes (a|0\rangle + b|1\rangle)$ we obtain:

$$\begin{aligned}
(A \oplus B)(|1\rangle \otimes (a|0\rangle + b|1\rangle)) &= \begin{pmatrix} A_{11} & A_{12} & 0 & 0 \\ A_{21} & A_{22} & 0 & 0 \\ 0 & 0 & B_{11} & B_{12} \\ 0 & 0 & B_{21} & B_{22} \end{pmatrix} \cdot \begin{pmatrix} 0 \\ 0 \\ a \\ b \end{pmatrix} \\
&= \begin{pmatrix} 0 \\ 0 \\ aB_{11} + bB_{12} \\ aB_{21} + bB_{22} \end{pmatrix} \\
&= (aB_{11} + bB_{12})|10\rangle + (aB_{21} + bB_{22})|11\rangle \\
&= |1\rangle \otimes B(a|0\rangle + b|1\rangle)
\end{aligned} \tag{2.65}$$

Putting these results together, when the 2-qubit controlled gate $(A \oplus B)$ acts on a *general* 2-qubit superposition state $|\psi\rangle = a|00\rangle + b|01\rangle + c|10\rangle + d|11\rangle$ the control qubit is no longer purely $|0\rangle$ or purely $|1\rangle$. Nevertheless, the linearity of quantum mechanics guarantees that the correct control actions are performed, in the correct proportions, on the target qubit.

$$(A \oplus B)|\psi\rangle = |0\rangle \otimes A(a|0\rangle + b|1\rangle) + |1\rangle \otimes B(c|0\rangle + d|1\rangle) \tag{2.66}$$

2.5.2 Semi-Classical Controlled Gates

Note that although we do not *have* to read the values of control qubits in order for controlled actions to be imposed on target qubits, we *may* do so if we wish. Specifically, in the traditional model of quantum computation one prepares a quantum state, evolves it unitarily through some quantum circuit, and then makes a *final* measurement on the output qubits. The values of the control qubits contained within such a

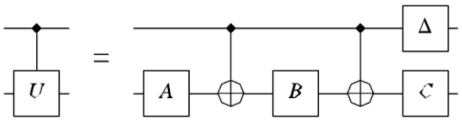


Fig. 2.29 A quantum circuit for a controlled-U gate, where U is an arbitrary 1-qubit gate

therefore also $|0\rangle$ and so the CNOTs do not do anything to the target qubit. Hence, the transformation to which the target qubit will be subject when the control qubit in the circuit is $|0\rangle$ is $C \cdot B \cdot A$. Note that the order is reversed with respect to the left to right sequence in the circuit diagram because, mathematically, if the A gate acts first, then the B gate, and then the C gate, the matrices must be multiplied in the order $C \cdot B \cdot A$ since when this object acts in an input state $|\psi\rangle$ we want the grouping to be $(C \cdot (B \cdot (A|\psi\rangle)))$ (gate A first then gate B then gate C). A little algebra shows that the net effect of these three operations is the identity (as required).

$$C \cdot B \cdot A \equiv R_z(b) \cdot R_y\left(\frac{c}{2}\right) \cdot R_y\left(-\frac{c}{2}\right) \cdot R_z\left(-\frac{d+b}{2}\right) \cdot R_z\left(\frac{d-b}{2}\right) = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \tag{2.73}$$

Next we consider what happens when the control qubit is in state $|1\rangle$. In this case the control qubit first picks up a phase factor since $\Delta|1\rangle = e^{ia}|1\rangle$. The control qubits of the CNOT gates will all be set to $|1\rangle$, and so they will apply a NOT gate (equivalent to a Pauli-X gate) to the target qubit when the CNOT gate acts. Hence, the transformation to which the target qubit will be subject when the control qubit is $|1\rangle$ is $e^{ia} C \cdot X \cdot B \cdot X \cdot A$. To simplify this expression we need to notice that $X \cdot R_y(\theta) \cdot X \equiv R_y(-\theta)$ and $X \cdot R_z(\theta) \cdot X \equiv R_z(-\theta)$. Hence we obtain:

$$\begin{aligned} C \cdot X \cdot B \cdot X \cdot A &= R_z(b) \cdot R_y\left(\frac{c}{2}\right) \cdot X \cdot R_y\left(-\frac{c}{2}\right) \cdot R_z\left(-\frac{d+b}{2}\right) \\ &\quad \cdot X \cdot R_z\left(\frac{d-b}{2}\right) \\ &= R_z(b) \cdot R_y\left(\frac{c}{2}\right) \cdot X \cdot R_y\left(-\frac{c}{2}\right) \cdot X \cdot X \cdot R_z\left(-\frac{d+b}{2}\right) \\ &\quad \cdot X \cdot R_z\left(\frac{d-b}{2}\right) \\ &= R_z(b) \cdot R_y\left(\frac{c}{2}\right) \cdot X.R_y\left(-\frac{c}{2}\right) \cdot X \cdot X \cdot R_z\left(-\frac{d+b}{2}\right) \\ &\quad \cdot X \cdot R_z\left(\frac{d-b}{2}\right) \\ &= R_z(b) \cdot R_y\left(\frac{c}{2}\right) \cdot R_y\left(\frac{c}{2}\right) \cdot R_z\left(\frac{b+d}{2}\right) \cdot R_z\left(\frac{d-b}{2}\right) \\ &= R_z(b) \cdot R_y(c) \cdot R_z(d) \end{aligned} \tag{2.74}$$