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.

C.P. Williams, *Explorations in Quantum Computing,*
Texts in Computer Science,
DOI 10.1007/978-1-84628-887-6_2, © Springer-Verlag London Limited 2011

2.1 Classical Logic Gates

2.1.1 Boolean Functions and Combinational Logic

Logic is a sub-field of mathematics that is principally concerned with the validity of arguments, i.e., determining the truth or falsity of propositions by a process of reasoning from starting assumptions, called axioms, and by applying valid rules of inference to them. Logic is not concerned with determining what is actually true or false in the real world, since the real world is but one of infinitely many possible worlds we may choose to reason about. Rather logic provides the mathematical framework upon which we may draw valid conclusions from given starting assumptions.

The concept of a logic gate arose from efforts to formalize the laws of thought. George Boole (1815–1864) was a British mathematician who lived long before days of transistors and electronic digital computers. Like Babbage and von Leibinitz before him, Boole was interested in formalizing the process of mathematical reasoning. Before Boole, algebra had been thought about, primarily, as a vehicle for performing numerical calculations. However, Boole foresaw a wider opportunity: "[...] hitherto the expression of magnitude, or of operations upon magnitude, has been the express object for which the symbols of Analysis [algebra] have been invented, and for which their laws have been investigated, but this does not mean that the interpretations of algebra can only be quantitative".

Boole went on to provide an interpretation of algebraic expressions as statements about classes of objects. The universe of all objects is a set, and symbols, such as A, B, C, stands for subsets of objects from this set. Then the usual operations on sets, such as intersection ($A \cap B$), union ($A \cup B$), and complement (A^c) can be interpreted as making statements about these subsets of objects as show in Fig. 2.1.

For example, suppose we consider a universe of people with various pizza preferences. If A is the set people who like pepperoni, and B is the set of people who like anchovies, then $A \cap B$ is the set of people who like pepperoni *and* anchovies,

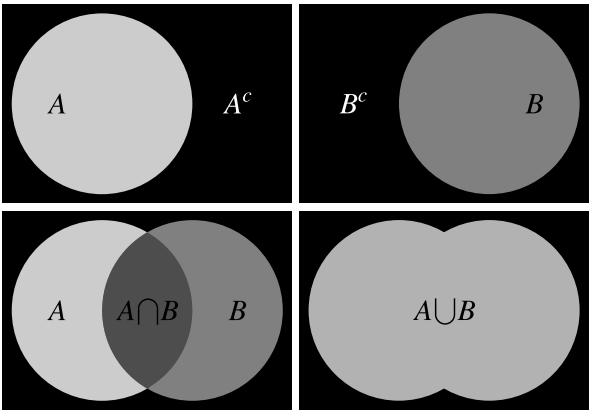


Fig. 2.1 Graphical illustration of the union, intersection and complement operations on sets

$A \cup B$ is the set of people who like pepperoni *or* anchovies or both, and A^c is the set of people who do *not* like pepperoni etc. Algebraic expressions interpreted in this way define what is called a Boolean *algebra*.

As you can see from the example, the *interpretation* of the sets that result from the intersection, union, and complement operations are described in terms of the logical connectives AND, OR, and NOT, indicating that there is a close parallel between set operations and logical operations. For example, if one assumes there are only two objects $1 =$ the set of all objects $=$ TRUE and $0 =$ the empty set of objects $= \varnothing =$ FALSE, we can write algebraic expressions that correctly capture alternate syntactic forms for logically equivalent statements. Hence, the *logical* assertion that a statement and its negation is necessarily contradictory expressed as the logical statement $a \wedge (\neg a) = 0 =$ FALSE (i.e., a AND (NOT a) is necessarily FALSE) mirrors the algebraic statement that the intersection of a set and its complement is necessarily empty, $A \cap A^c = \varnothing$. This restriction of the variables to just 0 and 1 makes the Boolean algebra into a Boolean *logic*.

Once one has the thought of interpreting algebraic statements as logical statements, one can easily define syntactically different forms having the same logical meaning. These are mathematical formulae in which the symbols, $a, b, c, \ldots$ stand for logical propositions that can be either true or false, and the connectives are logical functions. Table 2.1 lists the so-called "De Morgan's Laws" which give syntactically equivalent versions of elementary logical propositions. By using these laws we can systematically eliminate from any logical expression all instances of $\wedge$ or all instances of $\vee$. This means that we can reduce very complicated logical propositions to forms one of two standard forms, i.e., either a disjunction of conjuncts (i.e., Disjunctive Normal Form) or a conjunction of disjuncts (Conjunctive Normal Form).

Thus, if we can create hardware implementations of some very simple elementary gates, e.g., NOT, AND and OR, we can in principle combine those operations into very complex circuits

2.1.2 Irreversible Gates: AND and OR

The logical connectives AND ($\wedge$) and OR ($\vee$) capture, respectively, the notions of logical conjunction and disjunction . That is, for a compound proposition of the form $a \wedge b$ to be true *both* a and b must be true. Conversely, for a compound proposition of the form $a \vee b$ to be true it is sufficient for *either* a or b to be true individually.

Conventionally, a logic *gate* is thought of as a physical device that takes one or more Boolean values (i.e., FALSE or TRUE) as inputs and returns a single Boolean value as output. The Boolean values (FALSE and TRUE) are often used synonymously with the bit values 0 and 1 respectively. Logic gates are the key components of modern computers. Any classical computation can always be decomposed into a sequence of logic gates that act on only a few bits at a time. Hence logic gates lie at the heart of all modern computers.

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

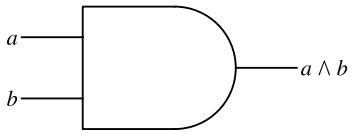


Table 2.3 Truth table of OR

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

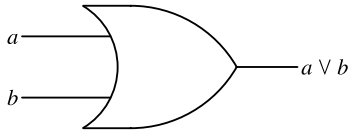


Table 2.4 Truth table of XOR (exclusive-OR)

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-

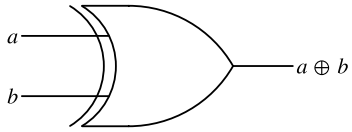


Fig. 2.4 Icon for the XOR gate—a logically irreversible gate

Table 2.5 Truth table of NAND

NAND:

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

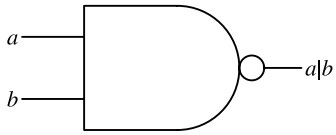


Fig. 2.5 Icon for the NAND gate—a universal gate for classical irreversible computing

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

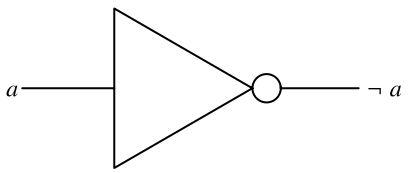


Fig. 2.6 Icon for the XOR gate—a 1-bit logically reversible gate

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

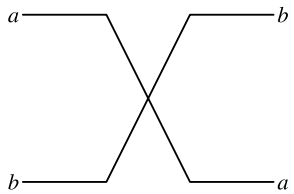


Fig. 2.7 Icon for the SWAP gate—a 2-bit logically reversible gate. The icon conveys the idea that to swap two bits we simply cross the wires on which those bits reside

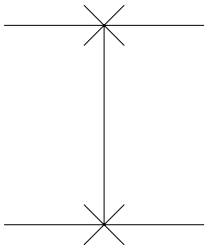


Fig. 2.8 Alternative icon for a SWAP gate that is more common in quantum circuit diagrams. The reason for having a different icon for SWAP in quantum circuits compared to classical circuits is that many implementations of quantum circuits do not have physical wires as such. Hence, it could be misleading to depict a SWAP operation as a crossing of wires. Instead, a SWAP operation can be achieved as the result of a sequence of applied fields

Table 2.10 Truth table of CNOT

	a	b	a'	b'
CNOT:	0	0	0	0
	0	1	0	1
	1	0	1	1
	1	1	1	0

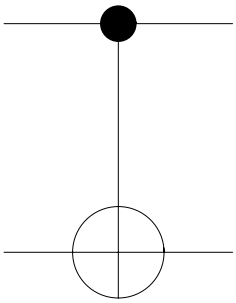


Fig. 2.9 Icon for the CNOT gate—a 2-bit logically reversible gate

That is, the decision to negate or not negate the second bit is controlled by the value of the first bit. Hence, the name "controlled-NOT". Note that, as shown in Fig. 2.10, the SWAP gate can be obtained from a sequence of three CNOT gates.

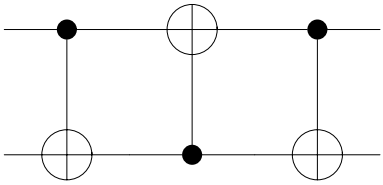


Fig. 2.10 A SWAP gate can be obtained from three CNOT gates

Table 2.11 Truth table of the TOFFOLI gate, which is universal for classical reversible computing

TOFFOLI:

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

Table 2.12 Truth table of the FREDKIN gate, which is universal for classical reversible computing

FREDKIN:

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

2.1.5 Universal Reversible Gates: FREDKIN and TOFFOLI

Just as there can be universal gates for classical irreversible computing, such as the NAND gate (which has two inputs and one output), so too can there be universal gates for classical *reversible* computing. However, the smallest gates that are both reversible *and* universal require *three* inputs and *three* outputs. Two well-known examples are the FREDKIN (controlled-SWAP) gate and the TOFFOLI (controlled-CNOT) gate, whose truth tables are shown in Tables 2.11 and 2.12 respectively.

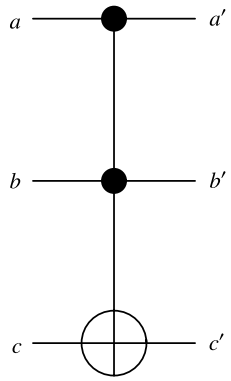


Fig. 2.11 Icon for the TOFFOLI gate also called the controlled-controlled-NOT gate. TOFFOLI is reversible and universal

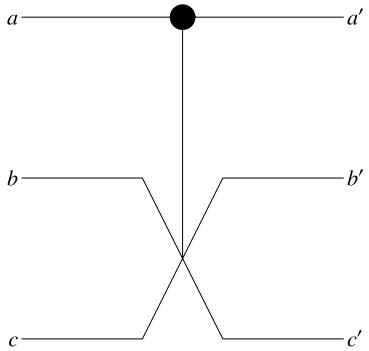


Fig. 2.12 Icon for the FREDKIN gate also called the controlled-SWAP gate. FREDKIN is reversible and universal

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:

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

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.

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

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 $\mathrm{AND}(x, y) = z$ 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 de-

pending 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

of NOT, CNOT , and Toffoli gates (defined in Sect. 2.2) is *universal* for reversible computation. More precisely, he showed that every permutation on $\{0, 1\}^n$ can be realized by means of a reversible circuit over the NOT-CNOT-TOFFOLI basis using at most one ancilla bit.[2]

2.1.9 Ancillae in Reversible Computing

Ancillae are an essential ingredient in classical reversible computing. For example, every circuit with more than 3 inputs over the NOT-CNOT-TOFFOLI basis realizes an *even* permutation on the space of its inputs. Therefore, to realize an *odd* permutation on $\{0, 1\}^n$, we need at least one ancilla bit with fixed constant value in addition to the n variable inputs. Toffoli has shown that one ancilla bit is, in fact, always sufficient [451]. Another way to see ancillae are essential is to consider computing a Boolean function $f : \{0, 1\}^n \longrightarrow \{0, 1\}$ reversibly. Every reversible circuit on m inputs, computing f, has exactly m outputs with one of them considered the value of f. If $m = n$, i.e., there is no ancilla bit, then it is easy to see that every output function must be a *balanced* Boolean function.[3] Therefore, if the function we want to simulate is not balanced, we require $m > n$ and there must therefore be at least one ancilla bit.

In general, we use the model described in Fig. 2.13 to define how a reversible circuit computes a function $f : \{0, 1\}^n \longrightarrow \{0, 1\}$. In this model, it is required that at the end of the computation all ancillae have their initial values, except one ancilla bit, designated as the "answer" bit, that carries the value of the function.

As in the case of reversible Turing machines, we can trade space for time in reversible *circuit* simulations of irreversible computations. But in the circuit picture "space" (i.e., the amount of auxiliary memory) is measured in terms of the *number of ancillae* required to perform the computation, and "time" is measured by the *size*, i.e. total gate count, of the circuit. In some cases allowing more ancillae results in a reversible circuit with smaller net size (i.e., fewer total gates).

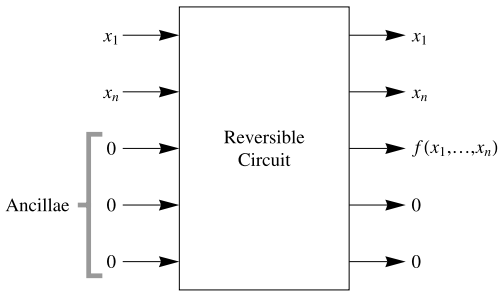


Fig. 2.13 Computing a Boolean function using a reversible circuit

[2] What we call an "ancilla bit" is also referred to as a "storage bit" or a "garbage bit" in the literature.

[3] A balanced function on $\{0, 1\}^n$ returns a value "1" for 2^{n-1} of its inputs and a value "0" for the other 2^{n-1} inputs.

To the best of my knowledge, only Cleve [110, 111] has addressed the space-time (ancillae-size) trade-off of simulation for the reversible circuits. He has shown that any polynomial size *formula* can be simulated by a polynomial size reversible circuit, which uses only 3 ancillae. If his method is applied to a *circuit*, then the result is an exponential size reversible circuit with 3 ancillae.

In contrast, we provide two new methods for simulating general Boolean circuits. In the first method, we show that any irreversible computation having t gates, depth d, and width w, can be implemented in a reversible circuit having $O(t^{2.58})$ gates, and at most $(w+1)\log d + O(1)$ ancillae. The second method deals with the simulation of branching programs. We prove that any branching program of depth d and width w can be simulated by a reversible circuit of size $\leq 4\,w\,2^d$ with $2\,w$ ancillae.

2.2 Universal Reversible Basis

We consider reversible circuits over the NOT-CNOT-TOFFOLI basis. Table 2.13 defines the action of these gates, and the Fig. 2.14 represents their standard icons. Note that the TOFFOLI gate *alone* is universal for reversible computing so, in principle, we do not need the NOT and CNOT gates. However, we allow them to simplify the constructions. Figure 2.15 shows how these reversible gates can simulate the classical (irreversible) standard gates, in some cases with ancillae.

Table 2.13 The action of reversible gates

NOT	CNOT	TOFFOLI
$a \mapsto 1 \oplus a$	$\begin{pmatrix} a \\ b \end{pmatrix} \mapsto \begin{pmatrix} a \\ a \oplus b \end{pmatrix}$	$\begin{pmatrix} a \\ b \\ c \end{pmatrix} \mapsto \begin{pmatrix} a \\ b \\ c \oplus (a \cdot b) \end{pmatrix}$

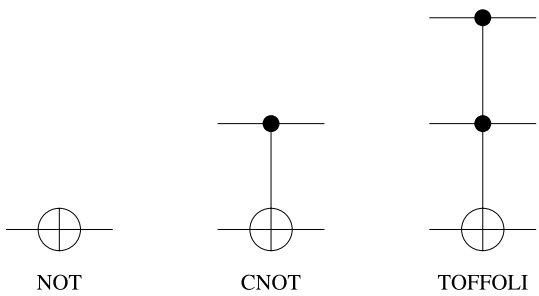


Fig. 2.14 The reversible basis

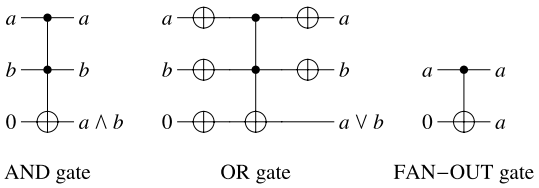


Fig. 2.15 Reversible simulation of classical gates

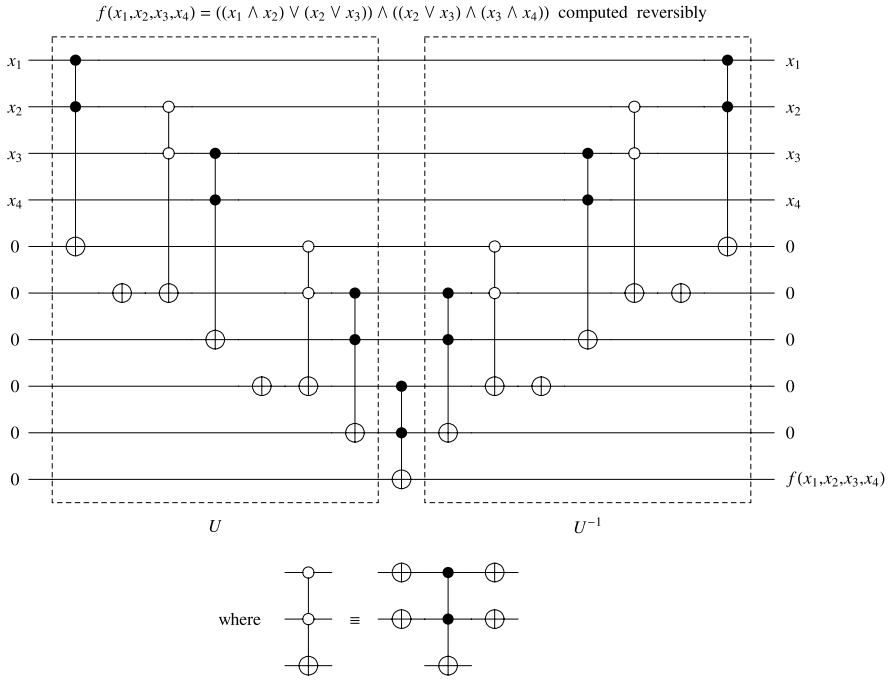


Fig. 2.16 Synthesis via reversible substitution

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.

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

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

2.4.1.4 Is Pauli X a NOT Gate for Qubits?

Although the Pauli X gate negates the computational basis states correctly, does it also behave like a true "NOT" gate when acting on a qubit in an *arbitrary* quantum state, i.e., a qubit state corresponding to a point on the Bloch sphere other than the North or South poles? To answer this, we must first specify what we *require* a quantum NOT gate to do, and then determine whether X acts in the appropriate manner.

Since the NOT gate has the effect of mapping a state at the North pole of the Bloch sphere into a state at the South pole and vice versa, it is natural to extend the definition of a NOT gate to be the operation that maps a qubit, $|\psi\rangle$, lying at any point on the surface of the Bloch sphere, into its antipodal state , $|\psi^{\perp}\rangle$, on the opposite side of the Bloch sphere as shown in Fig. 2.17. The antipodal point is that obtained by projecting a straight line from the original state through the origin to intersect the surface of the Bloch sphere on the opposite side. Mathematically, we can assume that our arbitrary starting state $|\psi\rangle$ is given by:

$$|\psi\rangle = \cos\left(\frac{\theta}{2}\right)|0\rangle + e^{i\phi}\sin\left(\frac{\theta}{2}\right)|1\rangle \tag{2.13}$$

where θ is the "latitude" and ϕ the "longitude" angles of $|\psi\rangle$ on the Bloch sphere. To obtain the antipodal point we move, just as we would on Earth, to the equivalent latitude in the opposite hemisphere and shift the longitude by 180° (i.e., π radians). Given the aforementioned definition of $|\psi\rangle$, the mathematical form of the antipodal state, $|\bar{\psi}\rangle$, must therefore be:

$$\begin{aligned}
|\psi^{\perp}\rangle &= \cos\left(\frac{\pi-\theta}{2}\right)|0\rangle + e^{i(\phi+\pi)}\sin\left(\frac{\pi-\theta}{2}\right)|1\rangle \\
&= \cos\left(\frac{\pi-\theta}{2}\right)|0\rangle - e^{i(\phi)}\sin\left(\frac{\pi-\theta}{2}\right)|1\rangle \\
&= \sin\left(\frac{\theta}{2}\right)|0\rangle - e^{i\phi}\cos\left(\frac{\theta}{2}\right)|1\rangle
\end{aligned} \tag{2.14}$$

where we have used the identities $\cos(\frac{\pi-\theta}{2}) = \sin(\frac{\theta}{2})$ and $\sin(\frac{\pi-\theta}{2}) = \cos(\frac{\theta}{2})$.

Having understood the relationship between the mathematical form of an arbitrary starting state, $|\psi\rangle$ to that of its true antipodal state, $|\psi^{\perp}\rangle$, we can now check whether $X|\psi\rangle = |\psi^{\perp}\rangle$, and hence, whether X qualifies as a true NOT gate for an arbitrary qubit. By direct evaluation we have:

$$\begin{aligned}
X|\psi\rangle &= \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \cdot \begin{pmatrix} \cos\left(\frac{\theta}{2}\right) \\ e^{i\phi}\sin\left(\frac{\theta}{2}\right) \end{pmatrix} = \begin{pmatrix} e^{i\phi}\sin\left(\frac{\theta}{2}\right) \\ \cos\left(\frac{\theta}{2}\right) \end{pmatrix} \\
&= e^{i\phi}\sin\left(\frac{\theta}{2}\right)|0\rangle + \cos\left(\frac{\theta}{2}\right)|1\rangle
\end{aligned} \tag{2.15}$$

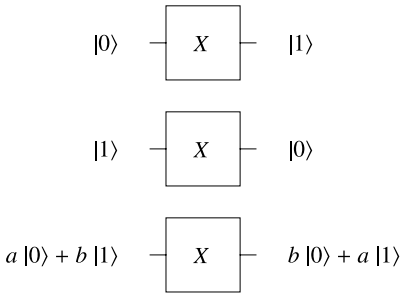


Fig. 2.17 The affect of the Pauli X gate operation on the computational basis states and an arbitrary pure state of a single qubit. The Pauli X gate "negates" the computational basis states correctly, but not an arbitrary superposition state! So the Pauli X gate is not a universal NOT gate for qubits. The universal NOT gate for qubits is discussed in Chap. 11

We are free to multiply by any overall phase factor we please since states that differ only in global phase are indistinguishable. As the amplitude of the $|0\rangle$ component of the true $|\psi^{\perp}\rangle$ state is $\sin(\theta/2)$, we multiply through (2.15) by $e^{-i\phi}$. Hence, the result of $X|\psi\rangle$ can be written as:

$$X|\psi\rangle = \sin\left(\frac{\theta}{2}\right)|0\rangle + e^{-i\phi}\cos\left(\frac{\theta}{2}\right)|1\rangle \neq |\psi^{\perp}\rangle \tag{2.16}$$

This is not $|\psi^{\perp}\rangle$. Hence, it is clear that $X|\psi\rangle$ does *not* negate an arbitrary single qubit state $|\psi\rangle$ since the result we get is not $|\psi^{\perp}\rangle$. Thus although, in classical computing, we can legitimately call the gate whose matrix is $\left(\begin{smallmatrix}0 & 1\\ 1 & 0\end{smallmatrix}\right)$ the "NOT" gate, we really ought not to use this name in the context of quantum computing.

We shall see in Chap. 11 that there is, in fact, *no* universal quantum NOT gate! That is, there is no fixed quantum gate that correctly negates every qubit it is handed.

2.4.1.5 Hadamard Gate

One of the most useful single qubit gates, in fact perhaps *the* most useful one, is the Hadamard gate, H. The Hadamard gate is defined by the matrix:

$$H = \frac{1}{\sqrt{2}}\begin{pmatrix}1 & 1\\ 1 & -1\end{pmatrix} \tag{2.17}$$

It acts, as depicted in Fig. 2.18, so as to map computational basis states into superposition states and vice versa:

$$H|0\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \tag{2.18}$$

$$H|1\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) \tag{2.19}$$

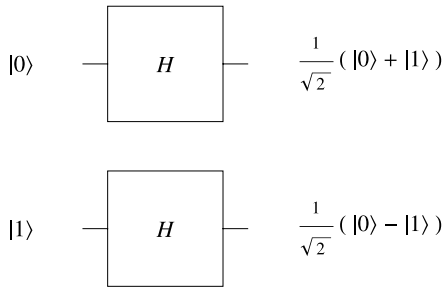


Fig. 2.18 The icon for the 1-qubit Walsh-Hadamard gate, H and its affect on computational basis states

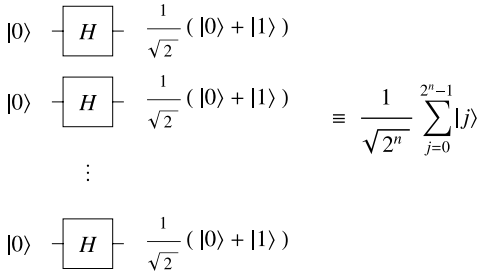


Fig. 2.19 By applying n H gates independently to n qubits, all prepared initially in state $|0\rangle$, we can create an n-qubit superposition whose component eigenstates are the binary representation of all the integers in the range 0 to $2^n - 1$. Thus, a superposition containing *exponentially* many 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:

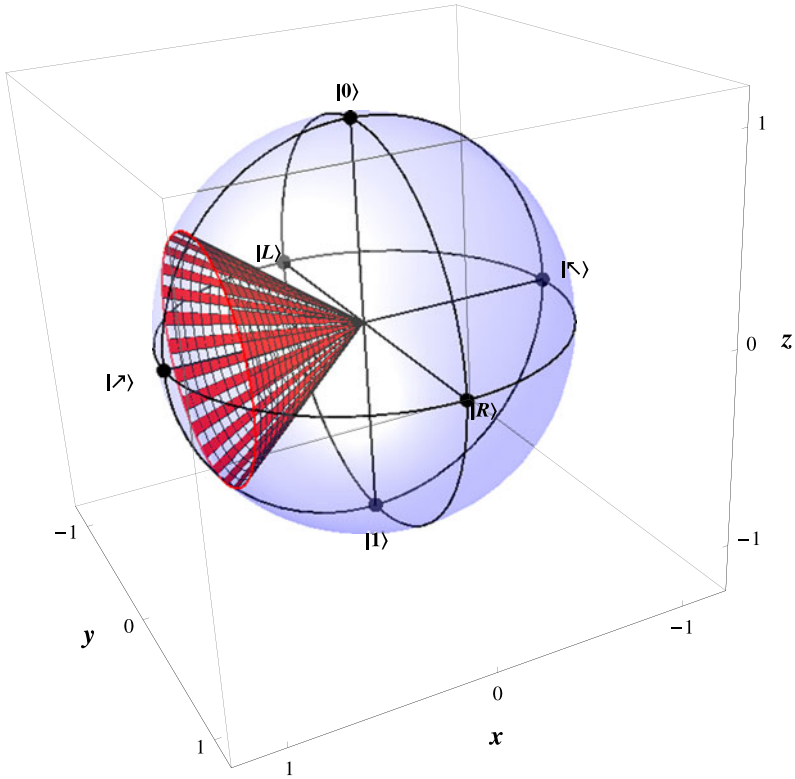


Fig. 2.20 An $R_x(\theta)$ gate maps a state $|\psi\rangle$ on the surface of the Bloch sphere to a new state, $R_x(\theta)|\psi\rangle$, represented by the point obtained by rotating a radius vector from the center of the Bloch sphere to $|\psi\rangle$ through an angle $\frac{\theta}{2}$ around the 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 $|\psi\rangle = \cos(\frac{\theta}{2})|0\rangle + e^{i\phi}\sin(\frac{\theta}{2})|1\rangle$.

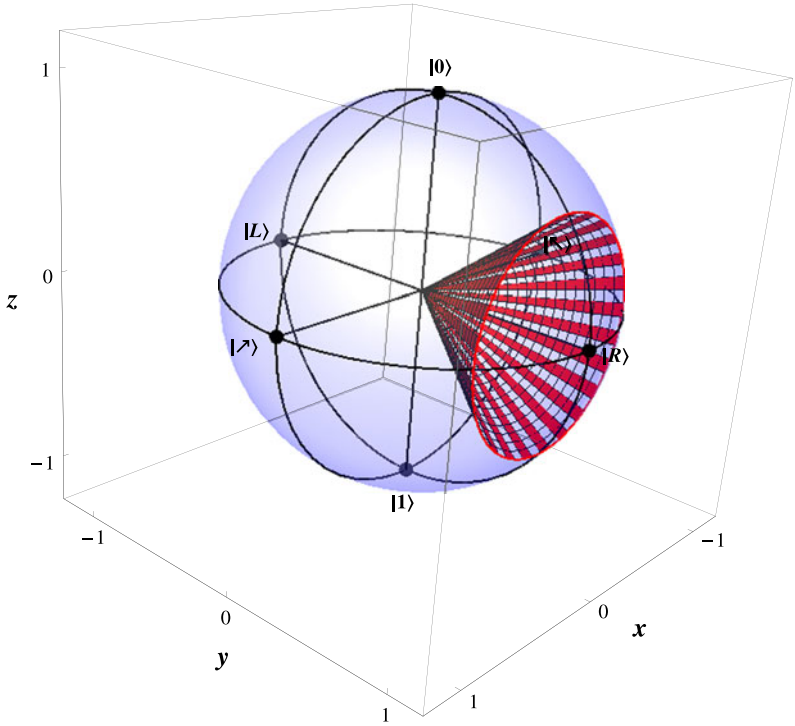


Fig. 2.21 An $R_y(\theta)$ gate maps a state $|\psi\rangle$ on the surface of the Bloch sphere to a new state, $R_y(\theta)|\psi\rangle$, represented by the point obtained by rotating a radius vector from the center of the Bloch sphere to $|\psi\rangle$ through an angle $\frac{\theta}{2}$ around the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

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

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:

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

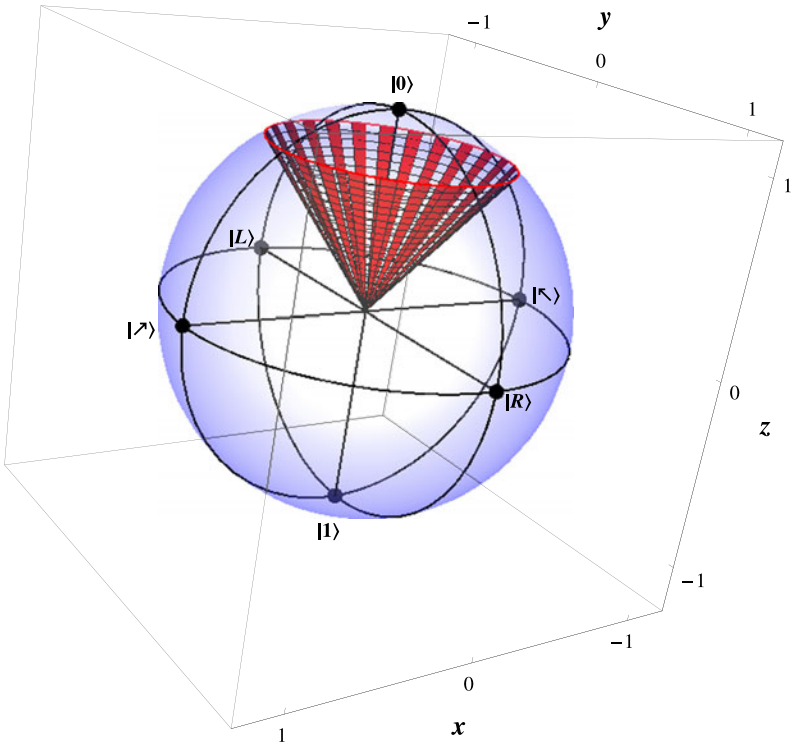


Fig. 2.22 An $R_z(\theta)$ gate maps a state $|\psi\rangle$ on the surface of the Bloch sphere to a new state, $R_z(\theta)|\psi\rangle$, represented by the point obtained by rotating a radius vector from the center of the Bloch sphere to $|\psi\rangle$ through an angle $\frac{\theta}{2}$ around the 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:

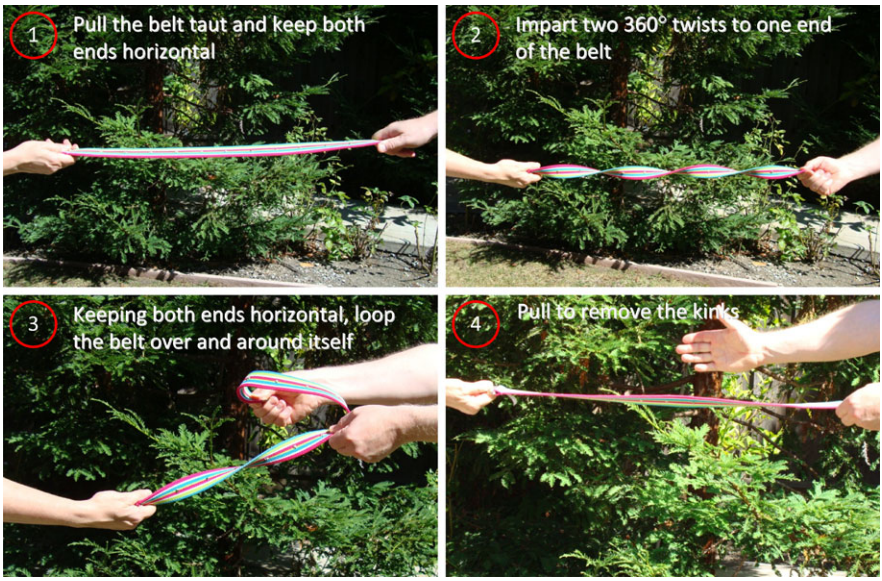


Fig. 2.23 "Dirac's Belt" uses a commonplace belt to illustrate that topology of a single qubit state wherein a rotation of 4π (two full twists) is required to restore the belt to its starting configuration

$$R_z(2\pi)|\psi\rangle = \begin{pmatrix} e^{-i\pi} & 0 \\ 0 & e^{i\pi} \end{pmatrix} \cdot \begin{pmatrix} \cos\left(\frac{\theta}{2}\right) \\ e^{i\phi}\sin\left(\frac{\theta}{2}\right) \end{pmatrix}$$
$$= \begin{pmatrix} -\cos\left(\frac{\theta}{2}\right) \\ -e^{i\phi}\sin\left(\frac{\theta}{2}\right) \end{pmatrix} = -|\psi\rangle \tag{2.30}$$

which has an extra overall phase of -1. To restore a state back to its original form we need to rotate it through 4π on the Bloch sphere.

Have you ever encountered anything like this in your everyday world? You probably think not, but you'd be wrong! Find yourself a thick leather belt. Have a friend hold one end flat and apply a rotation of 2π to the other end, i.e., one full twist (see Fig. 2.23). Now try to loop the belt around itself without tilting either end. In so doing, can you remove the twist? After some experimentation you should be convinced that the twist is there to stay and there is no way to remove it and yet keep the orientations of the ends of the belt fixed relative to one another. By analogy, a rotation of 2π has not restored the belt to its initial (flat and twist free) state. Ok so let's try again. Have a friend hold one end flat and apply a rotation of 4π to the other end, i.e., two full twists. Now try to loop the belt around itself without tilting either end. After a little experimentation you should find, to the surprise of most people, that the twist has gone! In other words, a rotation of 4π to one end of the belt has resulted in a state that is equivalent to the original (flat and twist free) state of the belt.

2.4.2.1 NOT, $\sqrt{\text{NOT}}$, and Hadamard from Rotation Gates

The NOT, $\sqrt{\text{NOT}}$, and Hadamard gates can all be obtained via sequences of rotation gates. For example,

$$\text{NOT} \equiv R_x(\pi) \cdot Ph\left(\frac{\pi}{2}\right) \tag{2.31}$$

$$\text{NOT} \equiv R_y(\pi) \cdot R_z(\pi) \cdot Ph\left(\frac{\pi}{2}\right) \tag{2.32}$$

$$\sqrt{\text{NOT}} \equiv R_x\left(\frac{\pi}{2}\right) \cdot Ph\left(\frac{\pi}{4}\right) \tag{2.33}$$

$$\sqrt{\text{NOT}} \equiv R_z\left(-\frac{\pi}{2}\right) \cdot R_y\left(\frac{\pi}{2}\right) \cdot R_z\left(\frac{\pi}{2}\right) \cdot Ph\left(\frac{\pi}{4}\right) \tag{2.34}$$

$$H \equiv R_x(\pi) \cdot R_y\left(\frac{\pi}{2}\right) \cdot Ph\left(\frac{\pi}{2}\right) \tag{2.35}$$

$$H \equiv R_y\left(\frac{\pi}{2}\right) \cdot R_z(\pi) \cdot Ph\left(\frac{\pi}{2}\right) \tag{2.36}$$

2.4.3 Arbitrary 1-Qubit Gates: The Pauli Decomposition

So far we have seen how *specific* 1-qubit gates can be decomposed into sequences of rotation gates, i.e., $R_x(\cdot)$, $R_y(\cdot)$, $R_z(\cdot)$, and phase gates, i.e., $Ph(\cdot)$. Next we consider how to decompose an arbitrary, maximally general, 1-qubit gate.

A maximally general 1-qubit gate will correspond to some 2×2 unitary matrix, U. As U is unitary the magnitude of its determinant must be unity, i.e., $|\det(U)| = 1$. This equation can be satisfied by $\det(U)$ taking on any of the values $+1, -1, +i$, or $-i$. If $\det(U) = +1$ then U is said to be "special unitary". If not, we can always write U in the form $U = e^{i\delta}V$ where V is a special unitary matrix, i.e., $\det(V) = +1$. So to find a circuit for the unitary matrix U it is sufficient to find a circuit for the special unitary matrix V, because simply appending a phase shift gate $Ph(\delta)$ to the circuit for V will give a circuit for U. This is easily seen by realizing

$$U = e^{i\delta}V = e^{i\delta}\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \cdot V = \begin{pmatrix} e^{i\delta} & 0 \\ 0 & e^{i\delta} \end{pmatrix} \cdot V = Ph(\delta) \cdot V$$

As V is a 2×2 special unitary matrix its rows and columns are orthonormal and, its elements, most generally, are complex numbers. Hence, V must have the form:

$$V = \begin{pmatrix} \alpha & -\bar{\beta} \\ \beta & \bar{\alpha} \end{pmatrix} \tag{2.37}$$

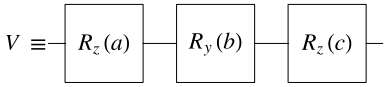


Fig. 2.24 Any 1-qubit special unitary gate can be decomposed into a rotation about the z-axis, the y-axis, and the z-axis

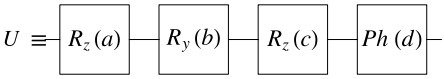


Fig. 2.25 Any 1-qubit unitary gate can be decomposed into a rotation about the z-axis, the y-axis, the z-axis, followed by a phase shift

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.

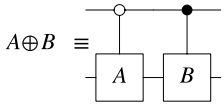


Fig. 2.26 The quantum circuit corresponding to a gate that performs different control actions according to whether the top qubit is $|0\rangle$ or $|1\rangle$

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:

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

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:

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

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.

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

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

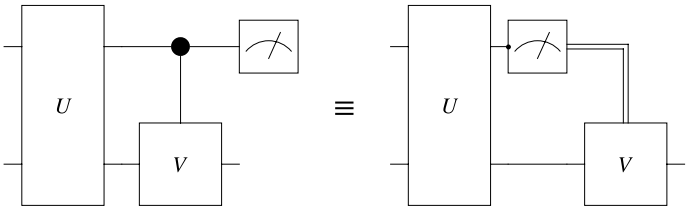


Fig. 2.27 Semi-classical quantum gates. Measurements of a control qubit made after a controlled gate can be moved before the gate and the subsequent controlled gate then be classically controlled. Griffiths and Niu used this trick in their semi-classical QFT [213], and Brassard used it in his quantum teleportation circuit [75]

quantum circuit are never read. However, we don't have to operate quantum circuits this way. If we want, we can move the final measurements on control qubits to earlier parts of the quantum circuit, and use the resulting classical bits to determine which gate operation to apply to the corresponding target qubits. Such a strategy will, of course, change the final state produced by the quantum circuit on any particular run, but it won't change their *statistical* properties averaged over many repetitions. Such intermediate measurements have been used to make a "semi-classical Fourier transform" [213] and also within a quantum circuit for teleportation [75].

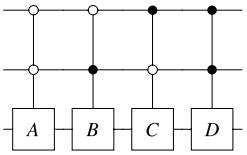
For example, as shown in Fig. 2.27 the control qubits of the controlled gates in the quantum Fourier transform can be measured immediately after they have acted and the resulting classical bit used to classically condition a subsequent controlled gate operation. The ability to move some final measurements to earlier stages of a quantum circuit and then condition subsequent gate operations on their (classical) outcomes can be of practical value by lowering the engineering complexity required to achieve practical quantum computational hardware.

2.5.3 Multiply-Controlled Gates

Controlled gates can be generalized to have multiple controls as shown in Fig. 2.28. Here a different operation is performed on the third qubit depending on the state of the top two qubits. Such multiply-controlled quantum gates are quite common in practical quantum circuits. Note, however, that the number of distinct states of the controls grows exponentially with the number of controls. So it becomes more difficult to actually build multiply-controlled gates beyond just a few control qubits.

2.5.4 Circuit for Controlled-U

Regardless of when qubits are to be read, we should like to know how to decompose these controlled gates into a simpler set of standard gates. Factoring a controlled gate



$$\equiv \begin{pmatrix} A & 0 & 0 & 0 \\ 0 & B & 0 & 0 \\ 0 & 0 & C & 0 \\ 0 & 0 & 0 & D \end{pmatrix} \equiv A \oplus B \oplus C \oplus D$$

Fig. 2.28 The quantum circuit corresponding to a gate that performs different control actions according to whether the top two qubits are $|00\rangle$, $|01\rangle$, $|10\rangle$, or $|1\rangle$

as in $A \oplus B = (\mathbb{1} \otimes A) \cdot (\mathbb{1} \otimes A^{-1} \cdot B)$ where $\mathbb{1} = \left(\begin{smallmatrix} 1 & 0 \\ 0 & 1 \end{smallmatrix}\right)$, we can see that the core "controlled" component of the gate is really a gate of the form:

$$\text{controlled-}U \equiv \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & U_{11} & U_{12} \\ 0 & 0 & U_{21} & U_{22} \end{pmatrix} \tag{2.67}$$

where the U_{ij} are the elements of an arbitrary 1-qubit gate $U = A^{-1} \cdot B$. We call a 2-qubit gate of the form $\left(\begin{smallmatrix} \mathbb{1} & \hat{0} \\ \hat{0} & U \end{smallmatrix}\right)$ a controlled-U gate.

We can construct a quantum circuit for a 2-qubit controlled-U gate in terms of CNOT gates and 1-qubit gates as follows. Let U be an arbitrary 1-qubit gate having a single qubit (Pauli) decomposition of the form $U = e^{ia} R_z(b) \cdot R_y(c) \cdot R_z(d)$. The action of the controlled-U gate is to do nothing to the target qubit when the control qubit is $|0\rangle$ and to apply U to the target qubit when the control qubit is $|1\rangle$. The act of "doing nothing" is mathematically equivalent to applying the identity gate to the target. So given the quantum circuit decomposition for computing U, what is a quantum circuit that computes controlled-U?

By (2.40) there exist angles a, b, c, and d such that:

$$U = e^{ia} R_z(b) \cdot R_y(c) \cdot R_z(d) \tag{2.68}$$

Given these angles, define matrices A, B, C as follows:

$$A = R_z\left(\frac{d-b}{2}\right) \tag{2.69}$$

$$B = R_y\left(-\frac{c}{2}\right) \cdot R_z\left(-\frac{d+b}{2}\right) \tag{2.70}$$

$$C = R_z(b) \cdot R_y\left(\frac{c}{2}\right) \tag{2.71}$$

$$\Delta = \text{diag}(1, e^{ia}) \tag{2.72}$$

We claim that the circuit shown in Fig. 2.29 computes controlled-U. Here is how it works. When the control qubit is in state $|0\rangle$ the Δ gate does change it because $\Delta|0\rangle = |0\rangle$ (with no phase addition). The control qubits of the CNOT gates are

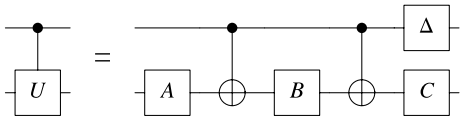


Fig. 2.29 A quantum circuit for a controlled-U gate, where U is an arbitrary 1-qubit gate

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:

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

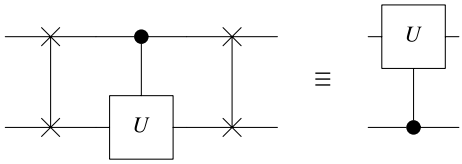


Fig. 2.30 A quantum circuit for an upside down controlled-U gate, where U is an arbitrary 1-qubit gate

Hence the circuit for controlled-U performs as follows:

$$\begin{aligned}
\text{controlled-}U\ |0\rangle(a|0\rangle + b|1\rangle) &= |0\rangle \otimes C \cdot B \cdot A(a|0\rangle + b|1\rangle) \\
&= |0\rangle \otimes (a|0\rangle + b|1\rangle) \\
\text{controlled-}U\ |1\rangle(a|0\rangle + b|1\rangle) &= e^{ia}|1\rangle \otimes C \cdot X \cdot B \cdot X \cdot A(a|0\rangle + b|1\rangle) \\
&= |1\rangle \otimes e^{ia} C \cdot X \cdot B \cdot X \cdot A(a|0\rangle + b|1\rangle) \\
&= |1\rangle \otimes U(a|0\rangle + b|1\rangle)
\end{aligned} \tag{2.75}$$

Thus U is applied to the target qubit if and only if the control qubit is set to $|1\rangle$.

2.5.5 Flipping the Control and Target Qubits

The control qubit does not have to be the topmost qubit in a quantum circuit. An upside down controlled-U gate would be given by SWAP · controlled-U · SWAP as shown in Fig. 2.30.

$$\text{upside-down-controlled-}U = \text{SWAP} \cdot \text{controlled-}U \cdot \text{SWAP} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & U_{11} & 0 & U_{12} \\ 0 & 0 & 1 & 0 \\ 0 & U_{21} & 0 & U_{22} \end{pmatrix} \tag{2.76}$$

The second qubit is now the control qubit and the first qubit the target qubit. The result is the matrix corresponding to a 2-qubit controlled quantum gate inserted into a circuit "upside down".

2.5.6 Control-on-$|0\rangle$ Quantum Gates

Furthermore, in a controlled quantum gate the value that determines whether or not a special action is performed does not have to be $|1\rangle$; it can be $|0\rangle$ (or any other state) too. A 2-qubit quantum gate with the special action conditioned on the value of the

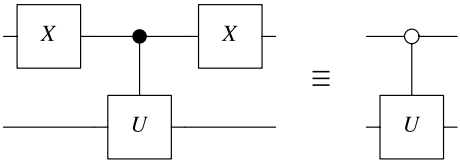


Fig. 2.31 A quantum circuit for a controlled quantum gate that acts when its control qubit is in state $|0\rangle$ (as indicated by the open circle on the control qubit) rather than state $|1\rangle$

first qubit being $|0\rangle$ instead of $|1\rangle$ is related to the usual controlled gate as follows:

$$\text{controlled}[1]\text{-}U = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & U_{11} & U_{12} \\ 0 & 0 & U_{21} & U_{22} \end{pmatrix} \tag{2.77}$$

$$\begin{aligned} \text{controlled}[0]\text{-}U &= (\text{NOT} \otimes \mathbb{1}_2) \cdot \text{controlled}[1]\text{-}U \cdot (\text{NOT} \otimes \mathbb{1}_2) \\ &= \begin{pmatrix} U_{11} & U_{12} & 0 & 0 \\ U_{21} & U_{22} & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \end{aligned} \tag{2.78}$$

as illustrated in Fig. 2.31.

2.5.7 Circuit for Controlled-Controlled-U

We can carry on in a similar fashion by, e.g., allowing multiple control qubits and/or target qubits. For example, earlier we interpreted the TOFFOLI gate as a controlled-controlled-NOT gate. Generalizing leads us to consider a controlled-controlled-U gate, where U is an arbitrary 1-qubit gate.

As a matrix, the controlled-controlled-U gate has the form:

$$\text{controlled-controlled-}U \equiv \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & U_{11} & U_{12} \\ 0 & 0 & 0 & 0 & 0 & 0 & U_{21} & U_{22} \end{pmatrix} \tag{2.79}$$

We can decompose a controlled-controlled-U gate into a circuit built from only CNOT gates and 1-qubit gates of the form shown in Fig. 2.32 (see [33]). Here $V = U^{1/2}$. The operation of this circuit can be understood by considering what it does to

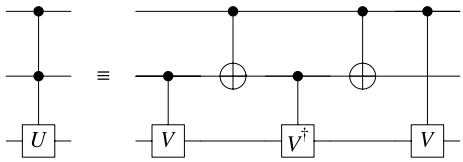


Fig. 2.32 Quantum circuit for the controlled-controlled-U operation. Here V is any quantum gate such that $V^2 = U$

the eight possible computational basis states of a three qubit system.

$$|000\rangle \xrightarrow{\text{ctrl-ctrl-}U} |000\rangle \tag{2.80}$$

$$|001\rangle \xrightarrow{\text{ctrl-ctrl-}U} |001\rangle \tag{2.81}$$

$$|010\rangle \xrightarrow{\text{ctrl-ctrl-}U} |01\rangle \otimes (V^\dagger \cdot V|0\rangle) = |010\rangle \tag{2.82}$$

$$|011\rangle \xrightarrow{\text{ctrl-ctrl-}U} |01\rangle \otimes (V^\dagger \cdot V|1\rangle) = |011\rangle \tag{2.83}$$

$$|100\rangle \xrightarrow{\text{ctrl-ctrl-}U} |10\rangle \otimes (V \cdot V^\dagger|0\rangle) = |100\rangle \tag{2.84}$$

$$|101\rangle \xrightarrow{\text{ctrl-ctrl-}U} |10\rangle \otimes (V \cdot V^\dagger|1\rangle) = |101\rangle \tag{2.85}$$

$$|110\rangle \xrightarrow{\text{ctrl-ctrl-}U} |11\rangle \otimes V^2|0\rangle = |11\rangle \otimes U|0\rangle \quad (\text{since } V^2 = U) \tag{2.86}$$

$$|111\rangle \xrightarrow{\text{ctrl-ctrl-U}} |11\rangle \otimes V^2|1\rangle = |11\rangle \otimes U|1\rangle \quad (\text{since } V^2 = U) \tag{2.87}$$

2.6 Universal Quantum Gates

A set of gates, $\mathcal{S}$, is "universal" if any feasible computation can be achieved in a circuit that uses solely gates from $\mathcal{S}$. The most interesting universal sets of gates are those containing a single gate. The NAND gate, the NOR gate, and the NMAJORITY gate, are all known, individually, to be universal for classical irreversible computing. Similarly, the TOFFOLI and FREDKIN gates are each known to be universal for classical reversible computing. Are there similar universal gates for *quantum* computing? If so, how many qubits does the *smallest* universal quantum gate have to have?

The fact that the closest classical gates to the quantum gates are the classical reversible gates, and these need a minimum of three bits to be universal, might lead you to expect that the smallest universal quantum gate will be a 3-qubit gate too. Indeed, there is a 3-qubit gate that is universal for quantum computing. It is called a DEUTSCH gate, and any feasible quantum computation can be achieved in a circuit built only from DEUTSCH gates acting on various triplets of qubits [137]. This gate

has the form:

$$\text{DEUTSCH} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & i\cos(\theta) & \sin(\theta) \\ 0 & 0 & 0 & 0 & 0 & 0 & \sin(\theta) & i\cos(\theta) \end{pmatrix} \tag{2.88}$$

where θ is any constant angle such that $2\theta/\pi$ is an irrational number. However, circuits for an arbitrary $2^n \times 2^n$ unitary matrix built from this gate are typically very inefficient in gate count.

Surprisingly, however, Deutsch's gate is not the smallest possibility. David DiVincenzo and John Smolin showed that DEUTSCH's gate could be built from only 2-qubit gates [149], and Adriano Barenco showed it could be obtained using only just a single type of 2-qubit gate—the BARENCO gate [32], which has the form:

$$\text{BARENCO} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & e^{i\alpha}\cos(\theta) & -ie^{i(\alpha-\phi)}\sin(\theta) \\ 0 & 0 & -ie^{i(\alpha+\phi)}\sin(\theta) & e^{i\alpha}\cos(\theta) \end{pmatrix} \tag{2.89}$$

where ϕ, α and θ are fixed irrational multiples of π and each other.

Thus, quantum gates are very different from classical gates in terms of universality. Whereas in classical reversible computing there is no 2-bit gate that is both reversible and universal, in quantum computing *almost all* 2-qubit gates are universal [80, 147]. This is quite remarkable. In particular, it means that certain *classical* reversible computations (which are described by permutation matrices and are, therefore, unitary) can potentially be implemented more efficiently using *quantum* gates than using only *classical reversible* gates. Ironically, it is conceivable that one of the nearest term large scale applications of *quantum* gates will be in implementations of (perhaps spintronic-based) "classical" reversible computers for fast, low power, reversible microprocessors.

The primary reason to study universal gates is to make the life of the experimentalist a little easier. If all quantum computations can be built from a single type of gate, then an experimentalist need only focus on how to achieve that gate in order to be guaranteed that any quantum computation is, in principle, attainable. Unfortunately, in practice, it is quite hard to use the Barenco gate as a primitive gate as it requires a 2-qubit Hamiltonian having three "tunable" parameters, ϕ, α and θ. However, luckily, the BARENCO gate is clearly a controlled-U gate and can therefore be further decomposed, using the methods of Sect. 2.9, into a sequence of 1-qubit gates and a single (fixed) 2-qubit gate such as CNOT. Hence, the set of gates $\mathcal{S} = \{R_x(\alpha), R_y(\beta), R_z(\gamma), Ph(\delta), \text{CNOT}\}$ must be a universal set of gates for quantum computing (and we can even drop one of the rotation gates if we wanted

Table 2.14 Families of gates that are universal for quantum computing

Universal gate family	Meaning	Noteworthy properties
$\{R_x, R_y, R_z, Ph, \text{CNOT}\}$	The union of the set of 1-qubit gates and CNOT is universal	The most widely used set of gates in current quantum circuits
$\text{BARENCO}(\phi, \alpha, \theta)$	A single type of 2-qubit gate is universal	The surprise here is that whereas in classical reversible computing no 2-bit classical reversible gate is universal, in quantum computing almost *all* 2-qubit gates are universal
$\{H, S, T, \text{CNOT}\}$ where $H = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$ is the Walsh-Hadamard gate, $S = \begin{pmatrix} 1 & 0 \\ 0 & i \end{pmatrix}$ is the "phase gate", and $T = \begin{pmatrix} 1 & 0 \\ 0 & \exp(i\pi/4) \end{pmatrix}$ is the "$\pi/8$ gate"	Three fixed-angle 1-qubit gates together with CNOT	The surprise here is that fixed angle gates can form a universal set. In fact, the Solvay-Kitaev theorem [284] implies that any 1-qubit gate can be approximated to accuracy ϵ using $\mathcal{O}(\log^c 1/\epsilon)$ gates from the set $\{H, S, T, \text{CNOT}\}$ where c is a positive constant

to). In fact, the set of all 1-qubit gates and CNOT is the most common set of gates used in constructing practical quantum circuits. Other universal gate sets are known, summarized in Table 2.14, that involve only fixed-angle gates. However, these do not typically lead to efficient quantum circuits due to the need to repeat fixed angle rotations many times to approximate a desired 1-qubit gate to adequate precision. Moreover, even if a given set of gates is universal, and therefore in principle all that is needed to achieve any quantum circuit, in practice, certain computations can be done more efficiently if an "over-complete" family of universal gates is used.

2.7 Special 2-Qubit Gates

The decision to use the set of all 1-qubit gates and CNOT as the universal family of gates, might not be the best choice depending on your type of quantum computing hardware. Different types of quantum computing hardware are associated with different Hamiltonians. So while a CNOT gate (say) may be easy to obtain in one embodiment, it might not be easy in another. For this reason, the next sections describe several different families of 1-qubit and 2-qubit gates that are more "natural" with respect to different types of quantum computing hardware. We give rules for inter-changing between these types of 2-qubit gates so that experimentalists can look at a quantum circuit expressed using one gate family and map it into another, perhaps easier to attain, family.

2.7.1 CSIGN, SWAP$^\alpha$, iSWAP, Berkeley B

The physical interactions available within different types of quantum computer hardware can give rise to different "natural" 2-qubit gates such as iSWAP, SWAP$^\alpha$, CSIGN etc. These are typically easier to achieve than CNOT in the particular physical embodiment, and if maximally entangling, provide no less efficient decompositions of arbitrary 2-qubit operations.

The four most common alternatives to CNOT are shown below:

$$\begin{aligned}
\text{CSIGN} &= \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \end{pmatrix}, \\
\text{SWAP}^\alpha &= \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \frac{1}{2}\left(1+e^{i\pi\alpha}\right) & \frac{1}{2}\left(1-e^{i\pi\alpha}\right) & 0 \\ 0 & \frac{1}{2}\left(1-e^{i\pi\alpha}\right) & \frac{1}{2}\left(1+e^{i\pi\alpha}\right) & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \\
\text{iSWAP} &= \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & i & 0 \\ 0 & i & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}, \\
B &= \begin{pmatrix} \cos\left(\frac{\pi}{8}\right) & 0 & 0 & i\sin\left(\frac{\pi}{8}\right) \\ 0 & \cos\left(\frac{3\pi}{8}\right) & i\sin\left(\frac{3\pi}{8}\right) & 0 \\ 0 & i\sin\left(\frac{3\pi}{8}\right) & \cos\left(\frac{3\pi}{8}\right) & 0 \\ i\sin\left(\frac{\pi}{8}\right) & 0 & 0 & \cos\left(\frac{\pi}{8}\right) \end{pmatrix}
\end{aligned} \tag{2.90}$$

Figure 2.33 shows the special icons for some of these gates and summarizes their properties with respect to qubit reversal and their relationship to their own inverse.

2.7.1.1 CSIGN

CSIGN arises naturally in Linear Optical Quantum Computing (LOQC).

$$\text{CSIGN} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \end{pmatrix} \tag{2.91}$$

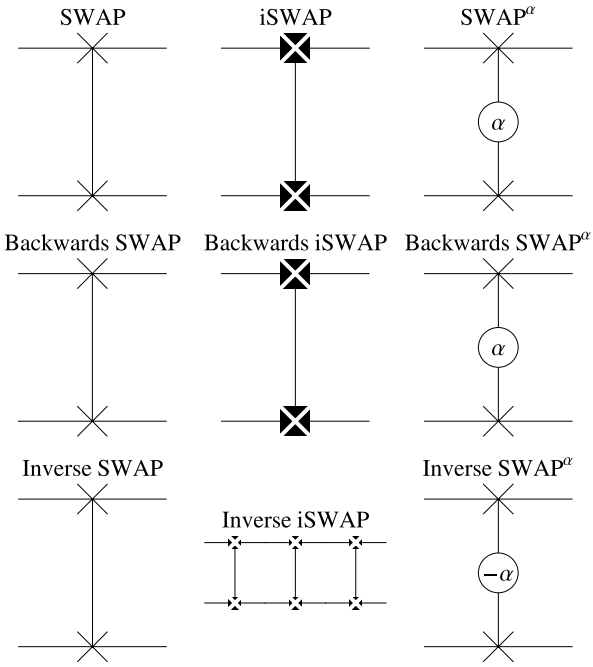


Fig. 2.33 Icons for the special quantum gates SWAP, iSWAP, and SWAP$^\alpha$. The first row shows the basic gate icon. The second row emphasizes that, unlike CNOT, these gates do not have a preferred "control" qubit and can be inserted "right way up" or "upside down" without it affecting the operation the gate performs. However, whereas CNOT is its own inverse, the same is not true for iSWAP (for which $\text{iSWAP}^\dagger = \text{iSWAP}^{-1} = \text{iSWAP}^3$) and SWAP$^\alpha$ (for which $(\text{SWAP}^\alpha)^\dagger = (\text{SWAP}^\alpha)^{-1} = \text{SWAP}^{-\alpha}$)

2.7.1.2 iSWAP

iSWAP arises naturally in superconducting quantum computing via Hamiltonians implementing the so-called XY model.

$$\text{iSWAP} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & i & 0 \\ 0 & i & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{2.92}$$

2.7.1.3 $\sqrt{\text{SWAP}}$

$\sqrt{\text{SWAP}}$ arises naturally in spintronic quantum computing as that approach employs the "exchange interaction".

$$\sqrt{\text{SWAP}} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \frac{1}{2}+\frac{i}{2} & \frac{1}{2}-\frac{i}{2} & 0 \\ 0 & \frac{1}{2}-\frac{i}{2} & \frac{1}{2}+\frac{i}{2} & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{2.93}$$

2.7.1.4 SWAP$^{\alpha}$

SWAP$^{\alpha}$ also arises naturally in spintronic quantum computing. The duration of the exchange operation determines the exponent achieved in SWAP$^{\alpha}$.

$$\text{SWAP}^{\alpha} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \frac{1}{2}\left(1+e^{i\pi\alpha}\right) & \frac{1}{2}\left(1-e^{i\pi\alpha}\right) & 0 \\ 0 & \frac{1}{2}\left(1-e^{i\pi\alpha}\right) & \frac{1}{2}\left(1+e^{i\pi\alpha}\right) & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{2.94}$$

2.7.1.5 The Berkeley B Gate

Hamiltonian is $\mathcal{H} = \frac{\pi}{8}(2X \otimes X + Y \otimes Y)$. Gate is $U = \exp(i\mathcal{H})$.

$$\begin{aligned} \text{B} &= e^{i\frac{\pi}{8}(2X\otimes X + Y\otimes Y)} \\ &= \begin{pmatrix} \cos\left(\frac{\pi}{8}\right) & 0 & 0 & i\sin\left(\frac{\pi}{8}\right) \\ 0 & \cos\left(\frac{3\pi}{8}\right) & i\sin\left(\frac{3\pi}{8}\right) & 0 \\ 0 & i\sin\left(\frac{3\pi}{8}\right) & \cos\left(\frac{3\pi}{8}\right) & 0 \\ i\sin\left(\frac{\pi}{8}\right) & 0 & 0 & \cos\left(\frac{\pi}{8}\right) \end{pmatrix} \\ &= \frac{\sqrt{2-\sqrt{2}}}{2} \begin{pmatrix} 1+\sqrt{2} & 0 & 0 & i \\ 0 & 1 & i\left(1+\sqrt{2}\right) & 0 \\ 0 & i\left(1+\sqrt{2}\right) & 1 & 0 \\ i & 0 & 0 & 1+\sqrt{2} \end{pmatrix} \end{aligned} \tag{2.95}$$

2.7.2 Interrelationships Between Types of 2-Qubit Gates

In experimental quantum computing one is faced with having to work with the physical interactions Nature provides. A priori, there is no reason to expect that the most accessible and controllable physical interactions should happen to permit a quantum mechanical evolution that can be interpreted as a CNOT gate. However, if one

looks at the Hamiltonians available in different types of physical systems one can always find 2-qubit gates from which we can, in conjunction with 1-qubit gates, build CNOT gates. In the following sections we give explicit constructions for how to build CNOT gates out of the kinds of 2-body interactions that are commonly available in real physical systems.

2.7.2.1 CNOT from CSIGN

We can obtain a CNOT gate given the ability to achieve 1-qubit gates and CSIGN.

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \equiv \left(\mathbb{1}_2 \otimes R_y\left(\frac{\pi}{2}\right)\right) \cdot \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \end{pmatrix} \cdot \left(\mathbb{1}_2 \otimes R_y\left(-\frac{\pi}{2}\right)\right) \tag{2.96}$$

An equivalent quantum circuit diagram is shown in Fig. 2.34.

2.7.2.2 CNOT from $\sqrt{\text{SWAP}}$

We can obtain a CNOT gate given the ability to achieve 1-qubit gates and $\sqrt{\text{SWAP}}$.

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \equiv \left(R_z\left(-\frac{\pi}{2}\right) \otimes \left(R_y\left(-\frac{\pi}{2}\right) \cdot R_z\left(-\frac{\pi}{2}\right)\right)\right)$$

$$\cdot \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \frac{1}{2}+\frac{i}{2} & \frac{1}{2}-\frac{i}{2} & 0 \\ 0 & \frac{1}{2}-\frac{i}{2} & \frac{1}{2}+\frac{i}{2} & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

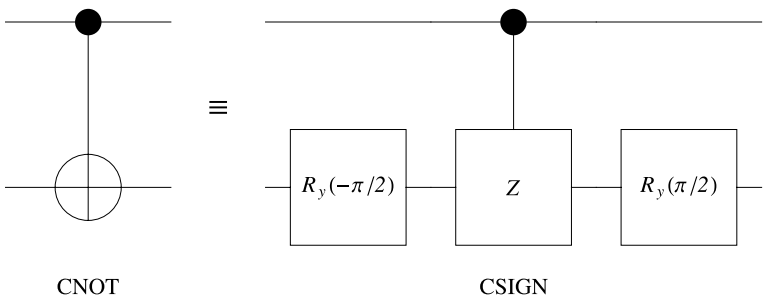


Fig. 2.34 Quantum circuit for obtaining a CNOT gate given the ability to achieve 1-qubit gates and CSIGN

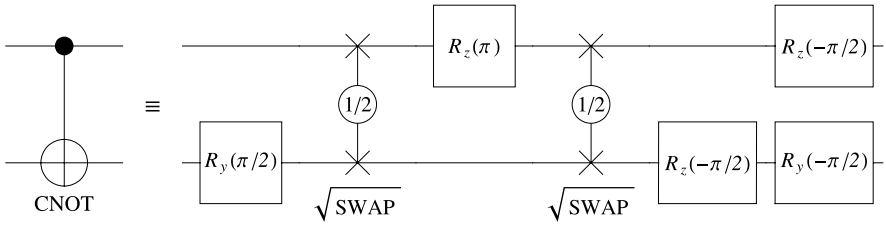


Fig. 2.35 Quantum circuit for obtaining a CNOT gate given the ability to achieve 1-qubit gates and $\sqrt{\text{SWAP}}$

$$\cdot\,(R_z(\pi)\otimes \mathbb{1}_2)\cdot \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \frac{1}{2}+\frac{i}{2} & \frac{1}{2}-\frac{i}{2} & 0 \\ 0 & \frac{1}{2}-\frac{i}{2} & \frac{1}{2}+\frac{i}{2} & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$
$$\cdot\left(\mathbb{1}I_2 \otimes R_y\left(\frac{\pi}{2}\right)\right) \tag{2.97}$$

An equivalent quantum circuit diagram is shown in Fig. 2.35.

2.7.2.3 CNOT from iSWAP and one SWAP

We can obtain a CNOT gate given the ability to achieve 1-qubit gates, iSWAP, and SWAP.

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \equiv \left(\mathbb{1}_2 \otimes R_y\left(-\frac{\pi}{2}\right)\right)\cdot \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & i & 0 \\ 0 & i & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$
$$\cdot\left(R_z\left(\frac{\pi}{2}\right)\otimes\left(R_z\left(-\frac{\pi}{2}\right)\cdot R_y\left(\frac{\pi}{2}\right)\right)\right) \tag{2.98}$$

An equivalent quantum circuit diagram is shown in Fig. 2.36.

2.7.2.4 CNOT from Two iSWAPs

We can obtain a CNOT gate given the ability to achieve 1-qubit gates and iSWAP.

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \equiv \left(\mathbb{1}_2 \otimes \left(R_z\left(\frac{\pi}{2}\right)\cdot R_y\left(-\frac{\pi}{2}\right)\right)\right)\cdot \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & i & 0 \\ 0 & i & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

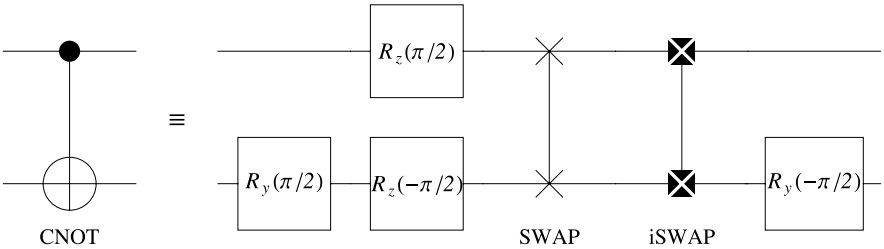


Fig. 2.36 Quantum circuit for obtaining a CNOT gate given the ability to achieve 1-qubit gates, iSWAP, and SWAP

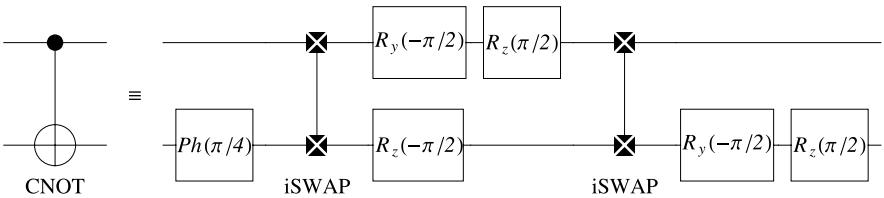


Fig. 2.37 Quantum circuit for obtaining a CNOT gate given the ability to achieve 1-qubit gates and iSWAP

$$\cdot \left(\left(R_z\left(\frac{\pi}{2}\right) \cdot R_y\left(-\frac{\pi}{2}\right) \right) \otimes R_z\left(-\frac{\pi}{2}\right) \right) \cdot \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & i & 0 \\ 0 & i & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$
$$\cdot \left(\mathbb{1}_2 \otimes Ph\left(\frac{\pi}{4}\right) \right) \tag{2.99}$$

An equivalent quantum circuit diagram is shown in Fig. 2.37.

2.8 Entangling Power of Quantum Gates

A set of qubits is *entangled* if the operations performed on one subset of qubits affect the complementary subset of qubits, even though those qubits are not operated upon directly. For example, imagine partitioning a set of n qubits $\mathcal{S}$ into two subsets $\mathcal{A} \subset \mathcal{S}$ and $\mathcal{B} = \mathcal{S} \setminus \mathcal{A}$. If operations performed on the qubits in $\mathcal{A}$ affect the state of the qubits in $\mathcal{B}$ then there is entanglement between the qubits in $\mathcal{A}$ and those in $\mathcal{B}$. In such a circumstance, the state of the system cannot be written as the direct product of a state for the qubits in subset $\mathcal{A}$ and a state for the qubits in the complementary subset $\mathcal{B}$. Such entanglement is unmediated and undiminished by distance and gives rise to so-called "non-local" effects which Einstein dubbed "spooky action at a distance".

The most striking difference between quantum logic gates and classical logic gates lies in the fact that quantum logic gates can cause the qubits upon which they

act to become more or less entangled, whereas classical gates cannot. In fact, the entire notion of entanglement is absent in classical computing and classical gates can neither entangle nor disentangle the bits upon which they act. Thus entanglement is a quintessentially quantum resource that is only available to quantum computers. Consequently, entanglement is believed to be essential in achieving the exponential speedups seen in quantum algorithms without other computational resources, such as space (memory), time and energy, ballooning exponentially.

Given the apparent importance of entanglement in quantum computing, it is natural to wonder whether all 2-qubit gates are equally good at generating entanglement or whether some are better than others? A little thought should convince you that some 2-qubit gates, such as those built as the direct product of two 1-qubit gates, cannot generate any entanglement whatsoever. But other gates, such as CNOT, seem able to map unentangled inputs into maximally entangled outputs. So clearly there is some variability in the potential for 2-qubit gates to generate entanglement. To make our study precise, however, we need a way to quantify the degree of entanglement within a state, i.e., we need an *entanglement measure*, and we need to define an ensemble of input states over which we would like to average this entanglement measure. Intuitively, if we pick an ensemble of initially unentangled inputs, i.e., product states, then we ought to be able to characterize how effective a given gate is at generating entanglement by seeing how entangled, on average, its outputs will be given it received initially unentangled inputs. This is the essential idea between the notion of the "entangling power" of a quantum gate. Intuitively, the more the output is entangled, the greater the entangling power of the gate.

2.8.1 "Tangle" as a Measure of the Entanglement Within a State

It turns out that there are many ways one could characterize the degree of entanglement within a 2-qubit quantum state. Fortunately, in the case of 2-qubit states, all the different entanglement measures turn out to be equivalent to one another. However, no such equivalence is found for entanglement measures of n-qubit states and attempts to find a unifying entanglement measure for n-qubit states have been fraught with difficulties spawning a cottage industry of"entanglement monotones" on which many Ph.D. theses have been written. For us, however, here we are concerned only with the entangling power of 2-qubit gates, and so any of the equivalent 2-qubit entanglement measures will serve us equally well.

Specifically, the *tangle* provides a quantitative measure of the degree of entanglement within a quantum state. Formally, the *tangle is the square of the concurrence*, which for a 2-qubit pure state, $|\psi\rangle$, is given by:

$$\text{Concurrence}(|\psi\rangle) = |\langle\psi|\widetilde{\psi}\rangle| \tag{2.100}$$

where $|\widetilde{\psi}\rangle$ is the spin-flipped version of $|\psi\rangle$. This is defined as $|\widetilde{\psi}\rangle = (Y \otimes Y)|\psi^*\rangle$, where Y is the Pauli-Y matrix, and $|\psi^*\rangle$ is $|\psi\rangle$ with its amplitudes complex conjugated. Thus, if $|\psi\rangle = a|00\rangle + b|01\rangle + c|10\rangle + d|11\rangle$, then $|\psi^*\rangle = a^*|00\rangle + b^*|01\rangle +$

$c^*|10\rangle + d^*|11\rangle$ and $|\widetilde{\psi}\rangle = -d^*|00\rangle + c^*|01\rangle + b^*|10\rangle - a^*|11\rangle$. Hence, the concurrence of a general 2-qubit state $|\psi\rangle$ is given by:

$$\text{Concurrence}(a|00\rangle + b|01\rangle + c|10\rangle + d|11\rangle) = |2b^*c^* - 2a^*d^*| \tag{2.101}$$

The "spin-flip" transformation maps the state of each component qubit into its orthogonal state. Hence the spin-flip transformation is not unitary and cannot, therefore, be performed deterministically by any isolated quantum system. So there can be no such thing as a perfect spin-flip "gate" as such. (If there were it would be a universal NOT gate.) Nevertheless, the spin-flip transformation is a perfectly legitimate mathematical specification of a transformation. One of the properties of the spin-flip transformation is that, if the 2-qubit state $|\psi\rangle$ happens to be a product state (i.e., an unentangled state) its spin-flipped version, $|\widetilde{\psi}\rangle$, will be orthogonal to $|\psi\rangle$. Hence, the overlap $\langle\psi|\widetilde{\psi}\rangle$ will be zero and hence the concurrence of state $|\psi\rangle$ will be zero. So unentangled states have a concurrence of zero.

At the other extreme, under the spin-flip transformation maximally entangled states, such as Bell states, remain invariant up to an unimportant overall phase. To see this, the four Bell states are given by: Bell states

$$\begin{aligned}
|\beta_{00}\rangle &= \tfrac{1}{\sqrt{2}}(|00\rangle + |11\rangle) \\
|\beta_{01}\rangle &= \tfrac{1}{\sqrt{2}}(|01\rangle + |10\rangle) \\
|\beta_{10}\rangle &= \tfrac{1}{\sqrt{2}}(|00\rangle - |11\rangle) \\
|\beta_{11}\rangle &= \tfrac{1}{\sqrt{2}}(|01\rangle - |10\rangle)
\end{aligned} \tag{2.102}$$

Under the spin-flip transformation these states transform, respectively, into:

$$\begin{aligned}
|\beta_{00}\rangle &\xrightarrow{\text{spin-flip}} -|\beta_{00}\rangle \\
|\beta_{01}\rangle &\xrightarrow{\text{spin-flip}} |\beta_{01}\rangle \\
|\beta_{10}\rangle &\xrightarrow{\text{spin-flip}} |\beta_{10}\rangle \\
|\beta_{11}\rangle &\xrightarrow{\text{spin-flip}} -|\beta_{11}\rangle
\end{aligned} \tag{2.103}$$

Hence, the overlap between a maximally entangled state and its spin-flipped counterpart is unity, which is the most it can be, implying that maximally entangled states have a concurrence of one.

Thus the tangle, as defined above, provides a quantitative measure for the degree of entanglement within a pure 2-qubit state. Generalizations of tangle to mixed states and multi-partite states are discussed in Chap. 11.

2.8.2 "Entangling Power" as the Mean Tangle Generated by a Gate

Having quantified the degree of entanglement within a state, it becomes possible to quantify the degree to which different gates generate entanglement when acting upon initially unentangled inputs. Specifically we can define the *entangling power* of a gate as follows [559]:

Entangling Power The entangling power of a 2-qubit gate U, $\mathrm{EP}(U)$, is the mean tangle that U generates averaged over all input product state inputs sampled uniformly on the Bloch sphere.

Mathematically this is expressed as:

$$\mathrm{EP}(U) = \langle E(U|\psi_1\rangle \otimes |\psi_2\rangle)\rangle_{|\psi_1\rangle,|\psi_2\rangle} \tag{2.104}$$

where $E(\cdot)$ is the tangle of any other 2-qubit entanglement measure such as the linear entropy (as all the 2-qubit entanglement measures are equivalent to one another), and $|\psi_1\rangle$ and $|\psi_2\rangle$ are single qubit states sampled uniformly on the Bloch sphere.

Although formally correct, the definition of entangling power given in (2.104) is not an easy thing to compute. However, since we have fixed the probability distribution over which the samples $|\psi_1\rangle$ and $|\psi_2\rangle$ are to be taken to be the uniform distribution on the surface of the Bloch sphere, we can build this assumption into the definition of entangling power and derive a more explicit, and effectively computable, formula for entangling power.

Let's begin by writing the arbitrary pure states $|\psi_1\rangle$ and $|\psi_2\rangle$ as:

$$|\psi_1\rangle = \cos\left(\frac{\theta_1}{2}\right)|0\rangle + e^{i\phi_1}\sin\left(\frac{\theta_1}{2}\right)|1\rangle \tag{2.105}$$

$$|\psi_2\rangle = \cos\left(\frac{\theta_2}{2}\right)|0\rangle + e^{i\phi_2}\sin\left(\frac{\theta_2}{2}\right)|1\rangle \tag{2.106}$$

For state $|\psi_1\rangle$, θ_1 is the angle between the z-axis and the state vector, and ϕ_1 is the angle around the z-axis in the x–y plane. Hence, as we desire to compute an average over the product of such states sampled *uniformly* over the Bloch sphere, we need to weight the contributions depending on the values of θ_1 and θ_2. Otherwise, the samples would be biased towards product states in the vicinity of the poles. To see this imagine that the density of states around the circumference of the Bloch sphere in the x–y plane is N states in a distance $2\pi R$, where R is the radius of the Bloch sphere, so the density of states at the equator is $N/(2\pi R)$. As we ascend the z-axis, to be unbiased, we still want to sample points around the circumference of a plane parallel to the x–y plane at height z at the same density. Hence we require $n/(2\pi r) = N/(2\pi R)$ which implies $n/N = r/R = \sin(\theta_1)$. Thus we must dilute states by a factor of $\sin(\theta_1)$ as we ascend the z-axis to maintain constant density.

Likewise for $|\psi_2\rangle$, giving an overall weighting function of $\sin(\theta_1)\sin(\theta_2)$. Hence, we have:

$$\begin{aligned}\mathrm{EP}(U) &= \langle E(U|\psi_1\rangle \otimes |\psi_2\rangle)\rangle_{|\psi_1\rangle,|\psi_2\rangle} \\ &= 2\mathrm{tr}\left((U\otimes U)\cdot \Omega_p \cdot (U^\dagger \otimes U^\dagger)\cdot \frac{1}{2}(\mathbb{1}_{16} - \mathrm{SWAP}_{1,3;4})\right)\end{aligned} \tag{2.107}$$

where $\mathbb{1}_{16}$ is the 16×16 identity matrix, and $\mathrm{SWAP}_{i,j;k}$ is the operator that swaps the i-th and j-th of k qubits.

$$\Omega_p = \frac{1}{16\pi^2}\int_0^{2\pi}\int_0^{2\pi}\int_0^{\pi}\int_0^{\pi} \sin(\theta_1)\sin(\theta_2)(|\psi_1\rangle\langle\psi_1|\otimes|\psi_2\rangle\langle\psi_2|)^{\otimes 2} d\theta_2 d\theta_1 d\phi_2 d\phi_1 \tag{2.108}$$

and the normalization factor $1/(16\pi^2)$ comes from the average of the weighting function:

$$\int_0^{2\pi}\int_0^{2\pi}\int_0^{\pi}\int_0^{\pi} \sin(\theta_1)\sin(\theta_2) d\theta_2 d\theta_1 d\phi_2 d\phi_1 = 16\pi^2 \tag{2.109}$$

With these definitions, Ω_p evaluates to the matrix

$$\Omega_p = \begin{pmatrix}
\frac{1}{9} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & \frac{1}{18} & 0 & 0 & \frac{1}{18} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & \frac{1}{18} & 0 & 0 & 0 & 0 & 0 & \frac{1}{18} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & \frac{1}{36} & 0 & 0 & \frac{1}{36} & 0 & 0 & \frac{1}{36} & 0 & 0 & \frac{1}{36} & 0 & 0 & 0 \\
0 & \frac{1}{18} & 0 & 0 & \frac{1}{18} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & \frac{1}{9} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & \frac{1}{36} & 0 & 0 & \frac{1}{36} & 0 & 0 & \frac{1}{36} & 0 & 0 & \frac{1}{36} & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{18} & 0 & 0 & 0 & 0 & 0 & \frac{1}{18} & 0 & 0 \\
0 & 0 & \frac{1}{18} & 0 & 0 & 0 & 0 & 0 & \frac{1}{18} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & \frac{1}{36} & 0 & 0 & \frac{1}{36} & 0 & 0 & \frac{1}{36} & 0 & 0 & \frac{1}{36} & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{9} & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{18} & 0 & 0 & \frac{1}{18} & 0 \\
0 & 0 & 0 & \frac{1}{36} & 0 & 0 & \frac{1}{36} & 0 & 0 & \frac{1}{36} & 0 & 0 & \frac{1}{36} & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{18} & 0 & 0 & 0 & 0 & 0 & \frac{1}{18} & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{18} & 0 & 0 & \frac{1}{18} & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{9}
\end{pmatrix} \tag{2.110}$$

Table 2.15 Entangling power of some common 2-qubit gates. Here $\mathbb{1}_2 \oplus U$ is a controlled gate with U defined as $U = R_x(a) \cdot R_y(b) \cdot R_z(c) \cdot Ph(d)$, and $\mathbb{1}_2 \oplus V$ is a controlled gate with V defined as $V = R_z(a) \cdot R_y(b) \cdot R_z(c) \cdot Ph(d)$. Notice that there can be no angle α that would make the SWAP$^\alpha$ a maximally entangling gate

U	EP(U)
$U \otimes V$	0
CNOT	$\frac{2}{9}$
iSWAP	$\frac{2}{9}$
B	$\frac{2}{9}$
SWAP	0
$\sqrt{\text{SWAP}}$	$\frac{1}{6}$
SWAP$^\alpha$	$\frac{1}{6}\sin^2(\pi\alpha)$
$R_x(a) \oplus R_x(b)$	$\frac{1}{9}(1-\cos(a-b))$
$R_x(a) \oplus R_y(b)$	$\frac{1}{18}(-\cos(b)-\cos(a)(\cos(b)+1)+3)$
$R_x(a) \oplus R_z(b)$	$\frac{1}{18}(-\cos(b)-\cos(a)(\cos(b)+1)+3)$
$R_y(a) \oplus R_x(b)$	$\frac{1}{18}(-\cos(b)-\cos(a)(\cos(b)+1)+3)$
$R_y(a) \oplus R_y(b)$	$\frac{1}{9}(1-\cos(a-b))$
$R_y(a) \oplus R_z(b)$	$\frac{1}{18}(-\cos(b)-\cos(a)(\cos(b)+1)+3)$
$R_z(a) \oplus R_x(b)$	$\frac{1}{18}(-\cos(b)-\cos(a)(\cos(b)+1)+3)$
$R_z(a) \oplus R_y(b)$	$\frac{1}{18}(-\cos(b)-\cos(a)(\cos(b)+1)+3)$
$R_z(a) \oplus R_z(b)$	$\frac{1}{9}(1-\cos(a-b))$
$\mathbb{1}_2 \oplus U$	$\frac{1}{6}+\frac{1}{18}(\sin(a)\sin(b)\sin(c)-\cos(a)\cos(b)-\cos(c)\cos(b)-\cos(a)\cos(c))$
$\mathbb{1}_2 \oplus V$	$\frac{1}{6}-\frac{1}{18}(\cos(a+c)\cos(b)+\cos(b)+\cos(a+c))$

Although it is non-obvious, an equivalent way to compute EP(U) is from the formula:

$$\begin{aligned}\text{EP}(U) = \frac{5}{9} - \frac{1}{36}[&\text{tr}((U\otimes U)^\dagger \cdot \text{SWAP}_{1,3;4} \cdot (U\otimes U) \cdot \text{SWAP}_{1,3;4}) \\ &+ \text{tr}(((\text{SWAP}_{1,2;2} \cdot U \otimes \text{SWAP}_{1,2;2} \cdot U))^\dagger \cdot \text{SWAP}_{1,3;4} \\ &\cdot (\text{SWAP}_{1,2;2} \cdot U \otimes \text{SWAP}_{1,2;2} \cdot U) \cdot \text{SWAP}_{1,3;4})] \end{aligned} \tag{2.111}$$

The entangling power of a gate ranges from 0 for non-entangling gates (such as SWAP), to $\frac{2}{9}$ for maximally entangling gates (such as CNOT, iSWAP, and Berkeley B). Other gates, such as $\sqrt{\text{SWAP}}$ and more generally SWAP$^\alpha$, have intermediate values of entangling power. Table 2.15 lists the entangling powers for some common types of 2-qubit gates. Typically, the entangling powers of parameterized gates, such as SWAP$^\alpha$ and $R_y(a) \oplus R_y(b)$, varies with the parameter values used.

2.8.3 CNOT from any Maximally Entangling Gate

In experimental quantum computing, one often needs to find a way to obtain a CNOT gate from whatever physically realizable 2-qubit interaction, is available. It turns out that the ease with which a CNOT can be obtained from the physically available 2-qubit gate, U, is intimately connected to the entangling power of U. In particular, if $\mathrm{EP}(U) = \frac{2}{9}$, i.e., maximal, but U is itself not a CNOT gate, then we can always create a CNOT gate from just *two* calls to U via a decomposition of the form:

$$\mathrm{CNOT} \equiv (A_1 \otimes A_2) \cdot U \cdot (H \otimes \mathbb{1}_2) \cdot U \tag{2.112}$$

where H is the Hadamard gate and A_1 and A_2 are 1-qubit gates.

This result is of practical importance to experimentalists since it may not always possible to achieve a CNOT gate directly from whatever interaction Hamiltonian is attainable within some physical context. Nevertheless, this result shows that once it is understood how a maximally entangling operation can be achieved from the available interaction Hamiltonians, then we can use it, in conjunction with 1-qubit gates, to achieve a CNOT.

2.8.4 The Magic Basis and Its Effect on Entangling Power

As you might recall, a quantum gate with unitary matrix U in the computational basis can be viewed as the matrix $V \cdot U \cdot V^\dagger$ in the "V-basis". In the case of 2-qubit gates there is a special basis, called the *magic basis*, that turns out to several have remarkable properties [54, 232, 296].

The "magic basis" is a set of 2-qubit states that are phase shifted versions of the Bell states. In particular, we have:

$$|00\rangle \xrightarrow{\mathcal{M}} |\mathcal{M}_1\rangle = |\beta_{00}\rangle \tag{2.113}$$

$$|01\rangle \xrightarrow{\mathcal{M}} |\mathcal{M}_2\rangle = i|\beta_{10}\rangle \tag{2.114}$$

$$|10\rangle \xrightarrow{\mathcal{M}} |\mathcal{M}_3\rangle = i|\beta_{01}\rangle \tag{2.115}$$

$$|11\rangle \xrightarrow{\mathcal{M}} |\mathcal{M}_4\rangle = |\beta_{11}\rangle \tag{2.116}$$

where $|\beta_{00}\rangle$, $|\beta_{01}\rangle$, $|\beta_{10}\rangle$, and $|\beta_{11}\rangle$ are the Bell states defined by:

$$\begin{aligned}
|\beta_{00}\rangle &= \tfrac{1}{\sqrt{2}}(|00\rangle + |11\rangle) \\
|\beta_{01}\rangle &= \tfrac{1}{\sqrt{2}}(|01\rangle + |10\rangle) \\
|\beta_{10}\rangle &= \tfrac{1}{\sqrt{2}}(|00\rangle - |11\rangle) \\
|\beta_{11}\rangle &= \tfrac{1}{\sqrt{2}}(|01\rangle - |10\rangle)
\end{aligned} \tag{2.117}$$

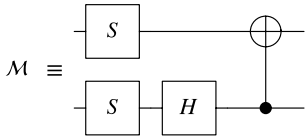


Fig. 2.38 Quantum circuit that implements the magic basis transformation. Here $S = Ph(\frac{\pi}{4}) \cdot R_z(\frac{\pi}{2})$ and $H = Z \cdot R_y(-\frac{\pi}{2})$

Thus, the matrix, $\mathcal{M}$, which maps the computational basis into the "magic" basis is:

$$\begin{aligned}\mathcal{M} &= |\mathcal{M}_1\rangle\langle 00| + |\mathcal{M}_2\rangle\langle 01| + |\mathcal{M}_3\rangle\langle 10| + |\mathcal{M}_4\rangle\langle 11| \\ &= \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & i & 0 & 0 \\ 0 & 0 & i & 1 \\ 0 & 0 & i & -1 \\ 1 & -i & 0 & 0 \end{pmatrix}\end{aligned} \tag{2.118}$$

The reason this basis is called the "magic basis" is because it turns out that any partially or maximally entangling 2-qubit gate, described by a *purely real* unitary matrix, U, becomes an *non-entangling* gate in the "magic" basis. In other words, no matter how entangling U may be, $\mathcal{M} \cdot U \cdot \mathcal{M}^\dagger$ is always a non-entangling gate, and hence $\mathrm{EP}(\mathcal{M} \cdot U \cdot \mathcal{M}^\dagger) = 0$.

We can make use of this observation in order to find a circuit for any 2-qubit gate described by a purely real unitary matrix, U, because either $\mathcal{M} \cdot U \cdot \mathcal{M}^\dagger = A \otimes B$ (one kind of non-entangling circuit) or else is related to a single SWAP gate (another non-entangling gate). And it is pretty easy to spot which is the case. Therefore, if we know the simplest quantum circuit implementing the magic basis transformation, we can then invert $\mathcal{M} \cdot U \cdot \mathcal{M}^\dagger = A \otimes B$ (or the similar one involving SWAP) to find a circuit for U. Luckily, it is easy to find a quantum circuit for the magic basis transformation. A simple quantum circuit that achieves the magic basis gate is show in Fig. 2.38.

If that was not magical enough, we can also use the magic basis transformation to relate a given purely real unitary, via a mathematical procedure involving $\mathcal{M}$, to gate that is guaranteed to be *maximally* entangling! Specifically, for any purely real 4×4 unitary matrix, U, then, regardless of its entangling power, the entangling power of the gate defined by $\mathcal{M} \cdot U \cdot \mathcal{M}$ is maximal, i.e., $\frac{2}{9}$. Amazing!

2.9 Arbitrary 2-Qubit Gates: The Krauss-Cirac Decomposition

Given that qubit-qubit interactions are essential to performing non-trivial quantum computations, it is important to understand how an *arbitrary* 2-qubit gate can be decomposed into more elementary gates such as CNOTs and 1-qubit gates. A priori it is not at all obvious how many CNOTs we will need. As we shall see the answer depends on the structure of the 2-qubit gate in question, but in no case do we ever need to use more than three CNOT gates [90, 452, 512, 517].

The key to finding a general circuit that can implement *any* 2-qubit gate is to use the magic basis transformation in conjunction with a factorization of an arbitrary

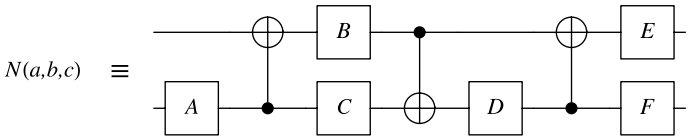


Fig. 2.39 Quantum circuit for the core entangling gate $N(a,b,c)$. Here $A = R_z(-\frac{\pi}{2})$, $B = R_z(\frac{\pi}{2} - 2c)$, $C = R_y(2a - \frac{\pi}{2})$, $D = R_y(\frac{\pi}{2} - 2b)$, $E = R_z(\frac{\pi}{2})$, and $F = Ph(\frac{\pi}{4})$

2-qubit gate discovered by Krauss and Cirac. Krauss and Cirac found that any 4×4 unitary matrix can be factored into the form:

$$U \equiv (A_1 \otimes A_2) \cdot e^{i(aX \otimes X + bY \otimes Y + cZ \otimes Z)} \cdot (A_3 \otimes A_4) \tag{2.119}$$

where X, Y, and Z are the three Pauli matrices, and $e^M = \mathbb{1} + M + \frac{1}{2!}(M \cdot M) + \frac{1}{3!}(M \cdot M \cdot M) + \frac{1}{3!}(M \cdot M \cdot M \cdot M) + \cdots$ is the *matrix* exponential[5] and $a, b, c \in \mathbb{R}$ [277, 296, 562]. Since we already know how to find quantum circuits for any 1-qubit gate, we can always find decompositions for the A_j whatever they may happen to be. We also know that the 1-qubit gates cannot change the entangling power of the core 2-qubit gate $N(a,b,c)$. So all the action is really concentrated in the 2-qubit gate $N(a,b,c)$, which is equivalent to the following unitary matrix:

$$N(a,b,c) \equiv \begin{pmatrix} e^{ic}\cos(a-b) & 0 & 0 & ie^{ic}\sin(a-b) \\ 0 & e^{-ic}\cos(a+b) & ie^{-ic}\sin(a+b) & 0 \\ 0 & ie^{-ic}\sin(a+b) & e^{-ic}\cos(a+b) & 0 \\ ie^{ic}\sin(a-b) & 0 & 0 & e^{ic}\cos(a-b) \end{pmatrix} \tag{2.120}$$

A quantum circuit for $N(a,b,c)$ is shown in Fig. 2.39. Algebraically, we have: $N(a,b,c) = (E \otimes F) \cdot \mathrm{CNOT}_{2,1;2} \cdot (\mathbb{1} \otimes D) \cdot \mathrm{CNOT}_{1,2;2} \cdot (B \otimes C) \cdot \mathrm{CNOT}_{2,1;2} \cdot (\mathbb{1} \otimes A)$ where $A = R_z(-\frac{\pi}{2})$, $B = R_z(\frac{\pi}{2} - 2c)$, $C = R_y(2a - \frac{\pi}{2})$, $D = R_y(\frac{\pi}{2} - 2b)$, $E = R_z(\frac{\pi}{2})$, and $F = Ph(\frac{\pi}{4})$.

The matrix, U, corresponding to any 2-qubit quantum gate is always unitary, and the *magnitude* of its determinant is always unity, i.e., $|\det(U)| = 1$. However, the ease with which we can implement U depends upon whether its elements are real or complex and whether its determinant is $+1$ or one of the other possibilities, consistent with $|\det(U)| = 1$, namely $-1, +i$, or $-i$. We classify the possibilities as follows:

1. $U \in \mathbf{SU}(2^n)$ implies U is a $2^n \times 2^n$ dimensional special unitary matrix containing real or complex elements and having a determinant $|\det(U)| = 1$, i.e., $\det(U) = \pm 1$ or $\pm i$.

[5] N.B. the leading "1" in the series expansion of the exponential function is replaced with the identity matrix, $\mathbb{1}$.

2. $U \in \mathbf{U}(2^n)$ implies U is a $2^n \times 2^n$ dimensional unitary matrix containing real or complex elements and having a determinant $|\det(U)| = 1$, i.e., $\det(U) = \pm 1$ or $\pm i$.
3. $U \in \mathbf{SO}(2^n)$ implies U is a $2^n \times 2^n$ dimensional special unitary matrix containing only real elements and having a determinant $\det(U) = +1$.
4. $U \in \mathbf{O}(2^n)$ implies U is a $2^n \times 2^n$ dimensional unitary matrix containing only real elements and having a determinant $\det(U) = \pm 1$.

The number of CNOT gates needed to implement U depends upon which the class into which U falls.

Using the upside down CNOT, we can write a circuit that implements the core entangling gate $N(a, b, c)$:

$$\begin{aligned} N(a,b,c) \equiv & \left(R_z\left(\frac{\pi}{2}\right) \otimes Ph\left(\frac{\pi}{4}\right)\right) \\ & \cdot \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix} \cdot \left(\mathbb{1}_2 \otimes R_y\left(\frac{\pi}{2} - 2b\right)\right) \\ & \cdot \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \cdot \left(R_z\left(\frac{\pi}{2} - 2c\right) \otimes R_y\left(2a - \frac{\pi}{2}\right)\right) \\ & \cdot \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix} \cdot \left(\mathbb{1}_2 \otimes R_z\left(-\frac{\pi}{2}\right)\right) \end{aligned} \tag{2.121}$$

2.9.1 Entangling Power of an Arbitrary 2-Qubit Gate

An *arbitrary* 2-qubit gate, U, can be factored according to the Krauss-Cirac decomposition as $U = (A_1 \otimes A_2) \cdot N(a, b, c) \cdot (A_3 \otimes A_4)$, where the A_j are 1-qubit gates, and $N(a, b, c) = \exp(i(aX \otimes X + bY \otimes Y + cZ \otimes Z))$ is the core entangling operation. As the entangling power of any gate is not affected by 1-qubit operations, the entangling power of an arbitrary 2-qubit gate must be determined entirely by the entangling power of its core factor $N(a, b, c)$. Using the formulae given earlier, we can calculate the entangling power of $N(a, b, c)$. In particular, one finds $\mathrm{EP}(N(a, b, c))$ is given by:

$$\begin{aligned} \mathrm{EP}(N(a,b,c)) = & -\frac{1}{18}\cos(4a)\cos(4b) - \frac{1}{18}\cos(4c)\cos(4b) \\ & - \frac{1}{18}\cos(4a)\cos(4c) + \frac{1}{6} \end{aligned} \tag{2.122}$$

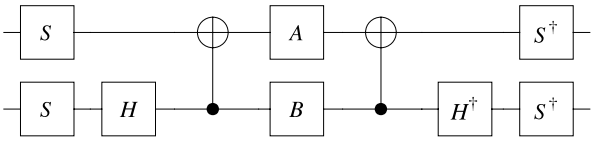


Fig. 2.40 Quantum circuit sufficient to implement any 2-qubit gate $U \in \mathbf{SO}(4)$. The unitary matrix for such a gate is purely real and has a determinant of $+1$

Notice that this immediately gives us a way of proving that the greatest entangling power of any 2-qubit gate is the largest value that $\mathrm{EP}(N(a,b,c))$ can assume, namely, $\frac{2}{9}$. The CNOT, iSWAP, and Berkeley B gates introduced earlier are all maximally entangling gates in this sense. However, the SWAP^{α} gate is not a maximally entangling gate.

2.9.2 Circuit for an Arbitrary Real 2-Qubit Gate

2.9.2.1 Case of $U \in \mathbf{SO}(4)$

If $U \in \mathbf{SO}(4)$ then the elements of U are purely real numbers and $\det(U) = +1$.

Theorem 2.1 *In the magic basis, $\mathcal{M}$, any purely real special unitary matrix $U \in \mathbf{SO}(4)$, can be factored as the tensor product of two special unitary matrices, i.e., we always have $\mathcal{M} \cdot U \cdot \mathcal{M}^{\dagger} = A \otimes B$ where $A, B \in \mathbf{SU}(2)$.*

A quantum circuit implementing the magic basis transformation (2.118) was shown in Fig. 2.38. Therefore, every 2-qubit quantum gate in $\mathbf{SO}(4)$ can be realized in a circuit consisting of 12 elementary 1-qubit gates and two CNOT gates (see Fig. 2.40).

2.9.2.2 Case of $U \in \mathbf{O}(4)$

If $U \in \mathbf{O}(4)$ then the elements of U are purely real numbers and $\det(U) = \pm 1$.

Theorem 2.2 *In the magic basis, $\mathcal{M}$, any purely real unitary matrix $U \in \mathbf{O}(4)$ with $\det(U) = -1$, can be factored as the tensor product of two special unitary matrices, i.e., we always have $\mathcal{M} \cdot U \cdot \mathcal{M}^{\dagger} = (A \otimes B) \cdot \mathrm{SWAP} \cdot (\mathbb{1} \otimes Z)$ where $A, B \in \mathbf{U}(2)$ and Z is the Pauli-Z matrix.*

Every 2-qubit quantum gate in $\mathbf{O}(4)$ with $\det(U) = -1$ can be realized in a circuit consisting of 12 elementary gates, two CNOT gates, and one SWAP gate (see Fig. 2.41). As you will show in Exercise 2.29 this circuit can be simplified further to one involving at most *three* CNOT gates.

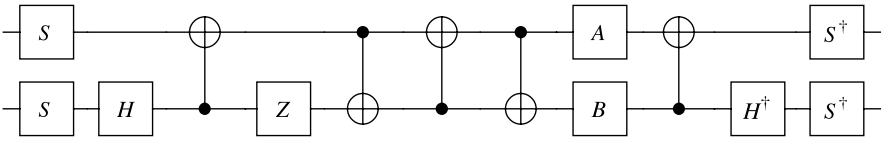


Fig. 2.41 Quantum circuit sufficient to implement any 2-qubit gate $U \in \mathbf{O}(4)$. The unitary matrix for such a gate is purely real and has a determinant of ± 1. Those gates having a determinant of $+1$ can be implemented using at most two CNOT gates. Those having a determinant of -1 can be implemented in a circuit of the form shown. In Exercise 2.29 you will simplify this circuit further to show that an arbitrary 2-qubit gate $U \in \mathbf{O}(4)$ requires at most *three* CNOT gates

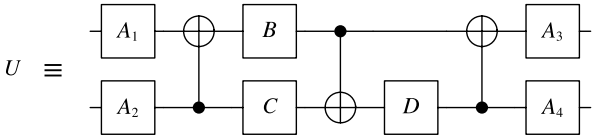


Fig. 2.42 Quantum circuit for an arbitrary 2-qubit gate, U. By the Kraus-Cirac decomposition U can be written in the form $(A_1 \otimes A_2) \cdot N(a,b,c) \cdot (A_3 \otimes A_4)$. As in the quantum circuit for $N(a,b,c)$, $B = R_z(\frac{\pi}{2} - 2c)$, $C = R_y(2a - \frac{\pi}{2})$, $D = R_y(\frac{\pi}{2} - 2b)$. The leftmost and rightmost single qubit gates needed to obtain $N(a,b,c)$ can be absorbed into the single qubit gates A_1, A_2, A_3, A_4

2.9.3 Circuit for an Arbitrary Complex 2-Qubit Gate

An arbitrary 2-qubit gate SWAP^α and can therefore have elements whose values are complex numbers. Every 2-qubit quantum gate in $\mathbf{U}(4)$ can be realized, up to an overall global phase factor, in a circuit consisting of 15 elementary 1-qubit gates, three CNOT gates (see Fig. 2.42).

2.9.4 Circuit for an Arbitrary 2-Qubit Gate Using SWAP^α

Ponder for a moment whether you would expect the quantum circuit for an arbitrary 2-qubit using the CNOT $\cup$ 1-qubit gates family to require more, less, or the same number of 2-qubit gates than the equivalent circuits based on different a gate family relying on a less than maximally entangling gate, such as SWAP^α. Since a general 2-qubit gate needs three CNOTs (and CNOT is a maximally entangling gate) one might expect that one needs more than three SWAP^α gates to implement a general 2-qubit gate. Surprisingly, this is not the case! In fact, three SWAP^α gates, having three different values for the exponents, are sufficient. The proof is by explicit

construction of the central entangling gate of the Krauss-Cirac decomposition:

$$\begin{aligned} N(a,b,c) \equiv (Ph(a+b-c) \otimes \mathbb{1}_2) \cdot \left(R_z\left(\frac{\pi}{2}\right) \otimes R_z\left(-\frac{\pi}{2}\right) \cdot R_y(\pi) \right) \\ \cdot \mathrm{SWAP}^{1-\frac{2(b-c)}{\pi}} \cdot (R_y(\pi) \cdot R_z(-\pi) \otimes R_y(\pi)) \\ \cdot \mathrm{SWAP}^{\frac{2(c-a)}{\pi}} \cdot (R_z(\pi) \otimes R_y(\pi) \cdot R_z(-\pi)) \\ \cdot \mathrm{SWAP}^{1-\frac{2(a+b)}{\pi}} \cdot \left(R_z\left(\frac{\pi}{2}\right) \otimes R_z\left(\frac{\pi}{2}\right) \right) \end{aligned} \tag{2.123}$$

2.10 Summary

Quantum gates are not always to be thought of in the same way we picture classical gates. In a conventional electronic circuits we are used to thinking of bits passing through logic gates. In quantum circuits this notion may or may not be accurate depending on how qubits are actually encoded within the physical system. If one is using photons to encode qubits and optical elements (such as beam-splitters or phase shifters) to perform gate operations, then the qubits are quite literally moving through the quantum gates. However, if we are using say trapped ions to encode the qubits, the logical state of the qubits is encoded within the internal excitation state of the ions, and the ions are held more or less in place. This distinction illustrates that a quantum gate is really nothing more than a deliberate manipulation of a quantum state.

In this chapter we introduced the idea of a quantum gate, and contrasted it with logically irreversible and logically reversible classical gates. Quantum gates are, like classical reversible gates, logically reversible, but they differ markedly on their universality properties. Whereas the smallest universal classical reversible gates have to use three bits, the smallest universal quantum gates need only use two bits. As the classical reversible gates are also unitary, it is conceivable that one of the first practical applications of quantum gates is in non-standard (e.g., "spintronic") implementations of classical reversible computers.

We described some of the more popular quantum gates and why they are useful, explained how these gates can be achieved via the natural evolution of certain quantum systems, and discussed quantum analogs of controlled and universal gates. Controlled gates are key to achieving non-trivial computations, and universal gates are key to achieving practical hardware.

In the theory of classical computing you would interpret a controlled gate operation as implying that you *read* (i.e., measure) the control bit and, depending on its value, perform the appropriate action on the target bits. However, such explicit measurement operations on the control qubits are neither implied nor necessary in *quantum* controlled gates. Instead, the controlled quantum gates

apply *all* the control actions consistent with the quantum state of the control qubits.

We showed that there are several 2-qubit gates that are as powerful as the CNOT gate when used in conjunction with 1-qubit gates, and gave explicit intercoversions between these types of gates. Such alternatives to CNOT gates may be easier to achieve than CNOT in specific schemes for quantum computing hardware. For example, iSWAP, SWAP$^{\alpha}$, and CSIGN are more naturally suited to superconducting, spintronic, and optical quantum computers than CNOT.

We introduced the "tangle" as a way of quantifying the entanglement within a quantum state, and used it to define the "entangling power" of a quantum gate. We also introduced the magic basis and demonstrated its effects on entangling power.

We ended the chapter with exact minimal quantum circuits sufficient to implement an arbitrary 2-qubit gate and gave an analytic scheme for converting a given unitary matrix into a minimal 2-qubit circuit.

2.11 Exercises

2.1 Which of the following matrices are unitary?

1. $\begin{pmatrix} \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \\ -\frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{pmatrix}$

2. $\begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix}$

3. $\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 \end{pmatrix}$

4. $\begin{pmatrix} 0 & 0 & i & 0 \\ 0 & i & 0 & 0 \\ i & 0 & 0 & 0 \\ 0 & 0 & 0 & i \end{pmatrix}$

Which of those could describe quantum gates that act on qubits? Explain your answer.

2.2 What is the output state from the quantum circuit shown in Fig. 2.43.

2.3 How would a CNOT gate transform an entangled input state of the form $|\psi\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$? Are the qubits still entangled after the CNOT has been applied? Explain your answer by making reference to the definition of an entangled state.

2.4 Show that X does not negate a general quantum state $|\psi\rangle = \cos(\frac{\theta}{2})|0\rangle + \exp(i\phi)\sin(\frac{\theta}{2})|1\rangle$.

2.5 Given a qubit whose state is known to lie in the equatorial x–y plane in the Bloch sphere is it possible to find a quantum gate that will always negate this qubit? If so, exhibit such a gate. If not, explain why it is impossible.

2.6 The circuit for controlled-controlled-U that was given earlier in this chapter assumed the existence of a controlled-V gate defined such that $V^2 = U$ with V unitary. Prove, for any unitary matrix U, that such a V always exists, i.e. that there exists a unitary matrix V such that $V^2 = U$.

2.7 Decompose the Hadamard gate, $H = \frac{1}{\sqrt{2}}\left(\begin{smallmatrix} 1 & 1 \\ 1 & -1 \end{smallmatrix}\right)$, in terms of $R_y(\theta)$ and $R_z(\phi)$ gates.

2.8 The "magic" basis is defined by the matrix....

2.9 Given real numbers x, y, and z and the Pauli matrices defined as

$$\mathbb{1} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \qquad X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \qquad Y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}, \qquad Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \tag{2.124}$$

prove the identity

$$e^{i(xX+yY+zZ)} = \cos(r)\mathbb{1} + \frac{\sin(r)}{r} i(xX + yY + zZ) \tag{2.125}$$

where $r = \sqrt{x^2 + y^2 + z^2}$. You might find the following identities to be useful: $\cos(\alpha) = \cosh(i\alpha)$ and $\sin(\beta) = -i\sinh(i\beta)$, and $i\sqrt{x^2 + y^2 + z^2} = \sqrt{-x^2 - y^2 - z^2}$.

2.10 Prove any 2×2 hermitian matrix can be written as a sum of Pauli matrices. This shows that any 1-qubit Hamiltonian can be expressed in terms of just Pauli matrices.

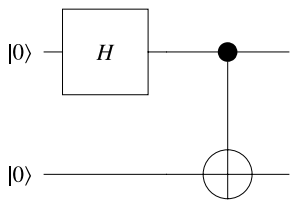


Fig. 2.43 This quantum circuit applies a Hadamard gate to the first qubit followed by a CNOT gate to both qubits

2.11 Show that a state, $|\psi\rangle$, is orthogonal to its antipodal state, $|\psi^{\perp}\rangle$, i.e., show $\langle\psi|\psi^{\perp}\rangle = 0$.

2.12 Prove that $R_x(\alpha)$ and $R_y(\alpha)$ rotate a general single qubit pure state about the x- and y-axes respectively through angle α.

2.13 Show that the NOR gate defined by a NOR $b \equiv \neg(a \vee b)$ is, like the NAND gate, also universal for classical irreversible computing. [Hint: Show that any logical proposition can be written in terms of just $\neg$ and $\vee$, and that both $\neg$ and $\vee$ can be expressed using only NOR gates.]

2.14 One of the most fundamental tasks we might imagine a computer doing is to decide whether two items in memory are the same and, if so, to output TRUE and, if not, to output FALSE. If we imagine the items in memory are represented by bit strings, our task becomes on of determining whether two bit strings are the same. Show that you can accomplish this task in a circuit that uses only $\neg$ and $\wedge$ gates. That is, provide a Boolean circuit for the $\Leftrightarrow$ (equivalence) relation in terms of just $\neg$ and $\wedge$ gates.

2.15 Quantum gates are supposed to be unitary and hence logically reversible. How then, do you explain why, when you apply a Hadamard gate to state $|0\rangle$ and observe what state you obtain, that some of the time you find the result to be $|0\rangle$ and some of the time you find the result to be $|1\rangle$? How can a Hadamard gate be logically reversible if it is not producing a deterministic output. Where has our logic failed us?

2.16 What measurement, or repeated measurements, might you make on a quantum system in order to verify that the action of a box purported to enact a Hadamard gate is functioning correctly. The Hadamard gate enacts the transformations $|0\rangle \to \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$ and $|1\rangle \to \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$? How many measurements would you need to make to have a 99% confidence in your answer?

2.17 The Hadamard gate, $H = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$ can be obtained, up to an overall global phase factor, using one R_x gate and one R_y gate, or using one R_y gate and one R_z gate. Can you obtain a Hadamard gate, up to an overall global phase factor, using just one R_x gate and one R_z gate? If so, exhibit the construction, else explain why it is impossible.

2.18 The FREDKIN and TOFFOLI gates are not the only (3-bit)-to-(3-bit) universal gates for reversible computing. For example, consider the reversible gate having the truth table given in Table 2.16 or, equivalently, the reversible gate represented by the matrix:

$$\text{NAND/NOR} \equiv \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \tag{2.126}$$

If the first bit in the input is set to 0, then the gate computes the NAND of the second and third input bits. Conversely, if the first bit in the input is set to 1, the gate computes the NOR of the second and third qubits.

Find (a) a classical reversible circuit and (b) a quantum circuit that implements the NAND/NOR gate.

2.19 If U is a maximally entangling gate, show that a CNOT gate can always be obtained from U via a decomposition of the form $\text{CNOT} \equiv (A_1 \otimes A_2) \cdot U \cdot (R_y(\frac{\pi}{2}) \otimes \mathbb{1}) \cdot U^{-1}$ where A_1 and A_2 are single qubit gates.

2.20 Find the general form for a 2-qubit circuit, which uses only 1-qubit gates and Berkeley B gates, that will implement an arbitrary 2-qubit gate, U. How many Berkeley B gates are necessary? How does this compare to the number of CNOT gates needed for an arbitrary 2-qubit gate?

2.21 Given an arbitrary 1-qubit gate defined as $U = \left(\begin{smallmatrix} u_{11} & u_{12} \\ u_{21} & u_{22} \end{smallmatrix}\right)$, what is the unitary matrix for the multiply controlled-U gate shown in Fig. 2.44?

2.22 What are the unitary matrices implied by the circuits shown in Fig. 2.45?

2.23 Determine the eigenvalues and normalized eigenvectors of the following operators built from the Pauli matrices:

(a) $\frac{1}{\sqrt{3}}(X + Y + Z)$

Table 2.16 Truth table of the NAND/NOR gate, which is a reversible gate containing the NAND and NOR gates quite explicitly

NAND/NOR:

Input bits	Output bits
000	111
001	101
010	110
011	011
100	100
101	001
110	010
111	000

(b) $\frac{1}{\sqrt{2}}(X \cdot Y + Y \cdot Z)$
(c) $\mathbb{1} \oplus X \oplus Y \oplus Z$
(d) $e^{i\alpha(X \otimes X + Y \otimes Y)}$ (N.B. this is a *matrix* exponential).

2.24 Construct the unitary matrix, $U = e^{-i\mathcal{H}t/\hbar}$, of the quantum gate one would obtain from the Hamiltonian, $\mathcal{H}$, at time $t = 1$, assuming you are working in units of $\hbar = 1$, for each of the following Hamiltonians:

(a) $\mathcal{H} = \alpha\ X \otimes \mathbb{1}$,
(b) $\mathcal{H} = \alpha\ X \otimes X$,
(c) $\mathcal{H} = \alpha\ X \otimes X + \beta\ Y \otimes Y$,
(d) $\mathcal{H} = \alpha\ X \otimes Y + \beta\ Y \otimes X$,

where $X, Y, \mathbb{1}$ are Pauli matrices, and $\alpha, \beta \in \mathbb{R}$.

2.25 Decompose the following 2×2 unitary matrices into sequences of $R_y(\alpha)$, $R_z(\beta)$, and $Ph(\gamma)$ gates:

$$\text{(a)} \quad \begin{pmatrix} \frac{1}{2}i\sqrt{\frac{1}{2}(5+\sqrt{5})} & \frac{1}{4}i(1-\sqrt{5}) \\ -\frac{1}{4}i(-1+\sqrt{5}) & -\frac{1}{2}i\sqrt{\frac{1}{2}(5+\sqrt{5})} \end{pmatrix}$$

$$\text{(b)} \quad \begin{pmatrix} \frac{\sqrt{\frac{3}{2}}}{2} + \frac{1}{2\sqrt{2}} & \frac{\sqrt{\frac{3}{2}}}{2} - \frac{1}{2\sqrt{2}} \\ \frac{\sqrt{\frac{3}{2}}}{2} - \frac{1}{2\sqrt{2}} & -\frac{\sqrt{\frac{3}{2}}}{2} - \frac{1}{2\sqrt{2}} \end{pmatrix}$$

$$\text{(c)} \quad \begin{pmatrix} \frac{1}{4}(3+i\sqrt{3}) & \frac{1}{4}(1-i\sqrt{3}) \\ \frac{1}{4}(1-i\sqrt{3}) & \frac{1}{4}(3+i\sqrt{3}) \end{pmatrix}$$

2.26 Assess the degree to which the following 2-qubit states are entangled by computing their "tangle", i.e., tangle($|\psi\rangle$) where:

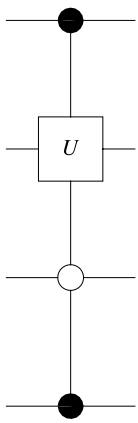


Fig. 2.44 A single qubit gate having an unusual pattern of control qubits

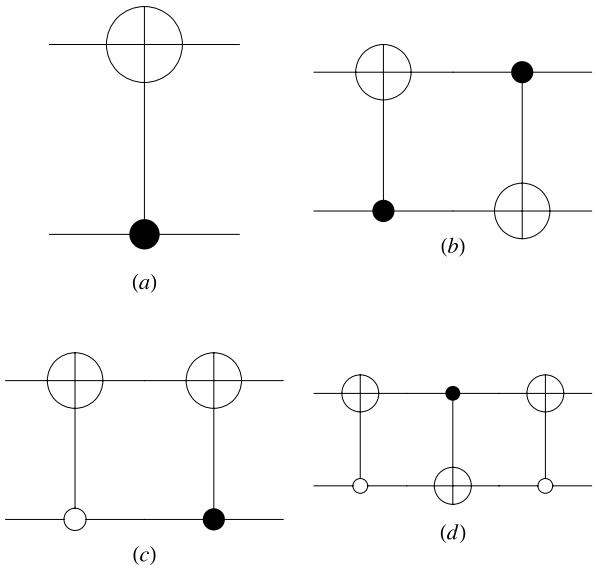


Fig. 2.45 Some 2-qubit gates involving "control-on-$|0\rangle$" CNOT gates and reversed embeddings

(a) $|\psi\rangle = \frac{1}{\sqrt{3}}|00\rangle + \frac{1}{\sqrt{3}}|01\rangle + \frac{1}{\sqrt{3}}|11\rangle$. Is the state entangled?
(b) $|\psi\rangle = \frac{1}{3\sqrt{5}}|00\rangle + \frac{2}{3\sqrt{5}}|01\rangle + \frac{2}{3}\sqrt{\frac{2}{5}}|10\rangle + \frac{4}{3}\sqrt{\frac{2}{5}}|11\rangle$. Is the state entangled?
(c) $|\psi\rangle = \frac{3}{2\sqrt{31}}|00\rangle + \frac{5}{2\sqrt{31}}|01\rangle + \frac{9}{2\sqrt{31}}|10\rangle + \frac{3}{2\sqrt{31}}|11\rangle$. Is the state entangled?
(d) $|\psi\rangle = \frac{1}{\sqrt{2}}|01\rangle - \frac{i}{\sqrt{2}}|10\rangle$. Is the state entangled?
(e) $|\psi\rangle = -\frac{e^{i\frac{\pi}{3}}}{\sqrt{2}}|00\rangle - \frac{1}{\sqrt{2}}|10\rangle$. Is the state entangled?

2.27 Consider the Bloch sphere with perpendicular axes x, y, and z. What 1-qubit gates, up to overall phase factors, perform the following operations on the Bloch sphere:

(a) Map the state at the North pole of the Bloch sphere to the state at the South pole?
(b) Map the state at $(x, y, z) = (0, 0, 1)$ to the state at $(x, y, z) = (0, 1, 0)$?
(c) Map the state at $(x, y, z) = (0, 0, 1)$ to the state at $(x, y, z) = (\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}}, 0)$?
(d) Map the state at $(x, y, z) = (0, \frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}})$ to the state at $(x, y, z) = (0, -1, 0)$?
(e) Map the state at $(x, y, z) = (0, 0, 1)$ to the state at $(x, y, z) = (\frac{1}{\sqrt{3}}, \frac{1}{\sqrt{3}}, \frac{1}{\sqrt{3}})$?

2.28 Compute the entangling power of the following 2-qubit quantum gates, and determine which ones are maximal entanglers:

(a) $$\begin{pmatrix} \cos\left(\frac{\alpha}{2}\right) & 0 & 0 & -\sin\left(\frac{\alpha}{2}\right) \\ \sin\left(\frac{\alpha}{2}\right) & 0 & 0 & \cos\left(\frac{\alpha}{2}\right) \\ 0 & \cos\left(\frac{\alpha}{2}\right) & -\sin\left(\frac{\alpha}{2}\right) & 0 \\ 0 & \sin\left(\frac{\alpha}{2}\right) & \cos\left(\frac{\alpha}{2}\right) & 0 \end{pmatrix}$$

(b) $$\begin{pmatrix} e^{-\frac{i\alpha}{2}} & 0 & 0 & 0 \\ 0 & \left(\frac{1}{2}+\frac{i}{2}\right)e^{\frac{i\alpha}{2}} & \left(\frac{1}{2}-\frac{i}{2}\right)e^{\frac{i\alpha}{2}} & 0 \\ 0 & \left(\frac{1}{2}-\frac{i}{2}\right)e^{-\frac{i\alpha}{2}} & \left(\frac{1}{2}+\frac{i}{2}\right)e^{-\frac{i\alpha}{2}} & 0 \\ 0 & 0 & 0 & e^{\frac{i\alpha}{2}} \end{pmatrix}$$

(c) $$\begin{pmatrix} \cos\left(\frac{\alpha}{2}\right) & 0 & -\sin\left(\frac{\alpha}{2}\right) & 0 \\ \sin\left(\frac{\alpha}{2}\right) & 0 & \cos\left(\frac{\alpha}{2}\right) & 0 \\ 0 & \cos\left(\frac{\alpha}{2}\right) & 0 & -\sin\left(\frac{\alpha}{2}\right) \\ 0 & \sin\left(\frac{\alpha}{2}\right) & 0 & \cos\left(\frac{\alpha}{2}\right) \end{pmatrix}$$

(d) $$\begin{pmatrix} e^{-\frac{i\alpha}{2}} & 0 & 0 & 0 \\ 0 & 0 & 0 & e^{\frac{i\alpha}{2}} \\ 0 & 0 & ie^{-\frac{i\alpha}{2}} & 0 \\ 0 & ie^{\frac{i\alpha}{2}} & 0 & 0 \end{pmatrix}$$

(e) $$\begin{pmatrix} \cos\left(\frac{\pi}{18}\right) & -\sin\left(\frac{\pi}{18}\right) & 0 & 0 \\ 0 & 0 & \cos\left(\frac{\pi}{18}\right) & \sin\left(\frac{\pi}{18}\right) \\ \sin\left(\frac{\pi}{18}\right) & \cos\left(\frac{\pi}{18}\right) & 0 & 0 \\ 0 & 0 & -\sin\left(\frac{\pi}{18}\right) & \cos\left(\frac{\pi}{18}\right) \end{pmatrix}$$

2.29 In Fig. 2.41 we show a circuit sufficient to implement an arbitrary real unitary $U \in \mathbf{O}(4)$ that uses four CNOT gates. However, this circuit is not in its simplest form. Prove the following circuit identities and use them to show an arbitrary purely real unitary matrix having $\det(U) = -1$ can be implemented in a circuit requiring at most *three* CNOT gates:

(a) $(\mathbb{1} \otimes Z) \cdot \mathrm{CNOT}_{2,1;2} \equiv \mathrm{CNOT}_{2,1;2} \cdot (\mathbb{1} \otimes R_z(\pi) \cdot \mathrm{Ph}(\frac{\pi}{2}))$ (i.e., prove the identity illustrated in Fig. 2.46)

(b) $\mathrm{CNOT}_{1,2;2} \cdot \mathrm{CNOT}_{2,1;2} \cdot \mathrm{CNOT}_{1,2;2} \cdot \mathrm{CNOT}_{2,1;2} \equiv \mathrm{CNOT}_{2,1;2} \cdot \mathrm{CNOT}_{1,2;2}$ (i.e., prove the circuit identity illustrated in Fig. 2.47)

(c) Hence, prove that any $U \in \mathbf{O}(4)$ with $\det(U) = -1$ can be implemented in a quantum circuit requiring at most three CNOT gates.

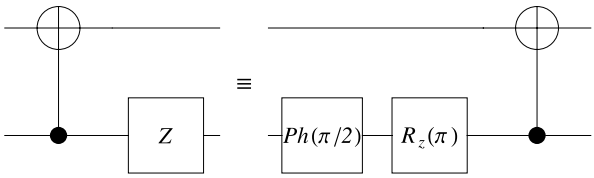


Fig. 2.46 A circuit identity that allows a Z gate to be moved through the control qubit of a CNOT gate

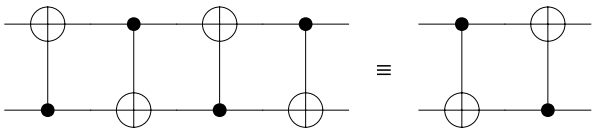


Fig. 2.47 A circuit identity that allows four CNOT gates to be contracted to two CNOT gates

2.30 Let $\mathcal{M}$ be the 2-qubit gate that maps the computational basis states $|00\rangle$, $|01\rangle$, $|10\rangle$, and $|11\rangle$, into the "magic basis":

$$|00\rangle \xrightarrow{\mathcal{M}} |\mathcal{M}_1\rangle = |\beta_{00}\rangle \tag{2.127}$$

$$|01\rangle \xrightarrow{\mathcal{M}} |\mathcal{M}_2\rangle = i\ |\beta_{10}\rangle \tag{2.128}$$

$$|10\rangle \xrightarrow{\mathcal{M}} |\mathcal{M}_3\rangle = i\ |\beta_{01}\rangle \tag{2.129}$$

$$|11\rangle \xrightarrow{\mathcal{M}} |\mathcal{M}_4\rangle = |\beta_{11}\rangle \tag{2.130}$$

where $|\beta_{00}\rangle$, $|\beta_{01}\rangle$, $|\beta_{10}\rangle$, and $|\beta_{11}\rangle$ are the Bell states defined by:

$$\begin{aligned} |\beta_{00}\rangle &= \tfrac{1}{\sqrt{2}}(|00\rangle + |11\rangle) \\ |\beta_{01}\rangle &= \tfrac{1}{\sqrt{2}}(|01\rangle + |10\rangle) \\ |\beta_{10}\rangle &= \tfrac{1}{\sqrt{2}}(|00\rangle - |11\rangle) \\ |\beta_{11}\rangle &= \tfrac{1}{\sqrt{2}}(|01\rangle - |10\rangle) \end{aligned} \tag{2.131}$$

(a) Verify that the matrix, $\mathcal{M}$, which maps the computational basis into the magic basis, is given by:

$$\mathcal{M} = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & i & 0 & 0 \\ 0 & 0 & i & 1 \\ 0 & 0 & i & -1 \\ 1 & -i & 0 & 0 \end{pmatrix} \tag{2.132}$$

(b) Prove that if U is a purely real unitary matrix then, regardless of the entangling power of U, the entangling power of $\mathcal{M} \cdot U \cdot \mathcal{M}$ is maximal, i.e., $\frac{2}{9}$.

(c) Prove that if U is a purely real unitary matrix then, regardless of the entangling power of U, the entangling power of $\mathcal{M} \cdot U \cdot \mathcal{M}^\dagger$ is zero.
(d) Check these claims by computing the entangling powers of U, $\mathcal{M} \cdot U \cdot \mathcal{M}$, and $\mathcal{M} \cdot U \cdot \mathcal{M}^\dagger$ for U given by:

$$U = \begin{pmatrix} \frac{\sqrt{\frac{3}{2}}}{2} & \frac{1}{2\sqrt{2}} & -\frac{\sqrt{\frac{3}{2}}}{2} & -\frac{1}{2\sqrt{2}} \\ \frac{1}{2\sqrt{2}} & -\frac{\sqrt{\frac{3}{2}}}{2} & -\frac{1}{2\sqrt{2}} & \frac{\sqrt{\frac{3}{2}}}{2} \\ \frac{1}{2\sqrt{2}} & \frac{\sqrt{\frac{3}{2}}}{2} & \frac{1}{2\sqrt{2}} & \frac{\sqrt{\frac{3}{2}}}{2} \\ \frac{\sqrt{\frac{3}{2}}}{2} & -\frac{1}{2\sqrt{2}} & \frac{\sqrt{\frac{3}{2}}}{2} & -\frac{1}{2\sqrt{2}} \end{pmatrix} \tag{2.133}$$

These remarkable properties explain why the vectors $|\mathcal{M}_1\rangle$, $|\mathcal{M}_2\rangle$, $|\mathcal{M}_3\rangle$ and $|\mathcal{M}_4\rangle$ are called the "magic basis".

2.31 The nice properties of the "magic basis", $\mathcal{M}$, do not, in general, carry over to complex unitary matrices.

(a) Experiment by generating a complex 4×4 unitary matrix, U, at random, and compute $\det(U)$, $|\det(U)|$, $\mathrm{EP}(U)$, $\mathrm{EP}(\mathcal{M} \cdot U \cdot \mathcal{M})$, and $\mathrm{EP}(\mathcal{M} \cdot U \cdot \mathcal{M}^\dagger)$. Such a matrix is most easily generated by guessing a quantum circuit containing a few R_x, R_z, Ph, and CNOT gates. After a few experiments you should convince yourself that the nice properties of the magic basis do not hold, in general, for complex unitaries.
(b) Show that $\det(\mathrm{SWAP}^\alpha) = (-1)^\alpha$, rather than ± 1 as is the case for all real unitary matrices. Based on this, would you expect $\mathrm{EP}(\mathcal{M} \cdot \mathrm{SWAP}^\alpha \cdot \mathcal{M})$ to be maximal? Compute $\mathrm{EP}(\mathcal{M} \cdot \mathrm{SWAP}^\alpha \cdot \mathcal{M})$ to check your answer.
(c) Given that $\det(\mathrm{iSWAP}) = 1$ (just like many real unitaries), would you expect $\mathrm{EP}(\mathcal{M} \cdot \mathrm{iSWAP} \cdot \mathcal{M}^\dagger)$ to be non-entangling? Compute $\mathrm{EP}(\mathcal{M} \cdot \mathrm{iSWAP} \cdot \mathcal{M}^\dagger)$ to check your answer.

2.32 Prove each of the following identities:

(a) $\mathrm{SWAP} \cdot \mathrm{SWAP} \cdot \mathrm{SWAP} = \mathrm{SWAP}$
(b) $\mathrm{SWAP} \cdot \mathrm{iSWAP} \cdot \mathrm{SWAP} = \mathrm{iSWAP}$
(c) $\mathrm{SWAP} \cdot \mathrm{SWAP}^\alpha \cdot \mathrm{SWAP} = \mathrm{SWAP}^\alpha$
(d) $\mathrm{SWAP}^\dagger = \mathrm{SWAP}$
(e) $\mathrm{iSWAP}^\dagger = \mathrm{iSWAP}^3$
(f) $(\mathrm{SWAP}^\alpha)^\dagger = \mathrm{SWAP}^{-\alpha}$

The first three identities show that it makes not difference which way around you insert a SWAP, iSWAP, and SWAP^α gate into a quantum circuit. The last two identities show that, whereas SWAP and CNOT are their own inverses, iSWAP and SWAP^α are not.

2.33 Invent an icon for the Berkeley B gate. In choosing your icon, decide whether you need to make it asymmetric so that you can distinguish between embedding the gate one way around or upside down, or whether this is immaterial. Then express the inverse of the Berkeley B gate in terms of itself and 1-qubit gates if necessary.

2.34 Invent an icon for the CSIGN gate. In choosing your icon, decide whether you need to make it asymmetric so that you can distinguish between embedding the gate one way around or upside down, or whether this is immaterial. Then express the inverse of the CSIGN gate in terms of itself and 1-qubit gates if necessary. Is the CSIGN gate a maximal entangling gate?

2.35 What matrix do you obtain when you raise the matrix representing the Berkeley B gate to the sixteenth power, i.e., B^{16}?

http://www.springer.com/978-1-84628-886-9

Explorations in Quantum Computing
Williams, C.P.
2011, XXII, 717 p., Hardcover
ISBN: 978-1-84628-886-9