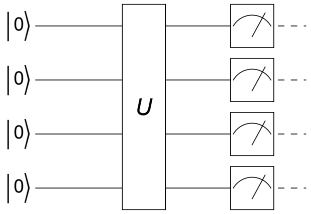
Chapter 2

Quantum Computation: Overview

In the simplest physical terms, quantum computation is an implementation of an arbitrary unitary operation on a finite collection of two-level quantum systems that are called *quantum bits* or *qubits* for short. It is typically depicted in a *quantum circuit diagram* as follows:



(2.1)

Each qubit is associated with a line that indicates the time evolution of the state specified on the left, and time flows from left to right. The *quantum logic gate operations* (or *gates* for short) on single or multiple qubits are denoted by a rectangular box often with labels indicating the types of the gates. Measurements are denoted by square boxes with needles. After a measurement, time-evolution is represented by dashed lines to remind that the information is classical, that is, there is no superposition.

The input state is prepared in one of the states in the logical basis, typically $|0\rangle \otimes \cdots \otimes |0\rangle$. After an overall unitary operation, the resulting state is measured in the logical basis, and the readouts are supposed to be the result of the computation.

In order for a quantum computer to be programmable, a given unitary operator $\hat{U}$ must be implemented as a combination of other more elementary unitary operators

$$\hat{U} = \hat{U}_1 \hat{U}_2 \cdots \hat{U}_L \,, \tag{2.2}$$

where each $\hat{U}_j$ is chosen from a small fixed set of elementary gate operations. The latter operations are the *elementary quantum logic gates* of the quantum computer.

In this chapter, we will examine widely-used choices for elementary gates and illustrate how a set of elementary gates achieve an arbitrary unitary operation to realize the so-called *universal quantum computation*.

Throughout the chapter, we denote by $\mathcal{S}$ the Hilbert space associated with a single qubit. The Hilbert space of an n-qubit system is given by $\mathcal{S}^{\otimes n}$, a tensor-product space of multiple $\mathcal{S}$. Each element $|x\rangle$ in the logical basis of $\mathcal{S}^{\otimes n}$ will be labeled by an integer $x = 0, 1, \cdots, 2^n - 1$, which should be understood to enumerate the tensor product form

$$|x\rangle \equiv |x_1 x_2 \cdots x_n\rangle := |x_1\rangle \otimes |x_2\rangle \otimes \cdots \otimes |x_n\rangle \tag{2.3}$$

in terms of the binary digits x_j $(j = 1, 2, \cdots, n)$ of x, that is, $x \equiv (x_1 x_2 \cdots x_n)_2$.

2.1 Single-Qubit Gates

Unitary operators on the two-dimensional vector space $\mathcal{S}$ associated with a single qubit form the *unitary group* U(2). In the standard logical basis, they are represented by 2×2 unitary matrices. We first take a look at some special examples and discuss the general properties of the single-qubit unitary operations.

2.1.1 Pauli Gates

The Pauli gate operations (or Pauli gates for short) are defined by the corresponding Pauli matrices

$$\hat{X} \doteq \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}, \quad \hat{Y} \doteq \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix}, \quad \hat{Z} \doteq \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}. \tag{2.4}$$

They form the most elementary single-qubit gate operations and are frequently used in many quantum algorithms. In this book, the Pauli gates will be denoted sometimes by $\hat{X}$, $\hat{Y}$, and $\hat{Z}$, and other times by $\hat{S}^x$, $\hat{S}^y$, and $\hat{S}^z$, depending on the context. In quantum circuit model, they are typically depicted by the circuit elements

$$-\!\!\!-\!\!\!-\boxed{X}-\!\!\!-\!\!\!-, \quad -\!\!\!-\!\!\!-\boxed{Y}-\!\!\!-\!\!\!-, \quad -\!\!\!-\!\!\!-\boxed{Z}-\!\!\!-\!\!\!-. \tag{2.5}$$

Pauli operator $\hat{X}$ maps the logical basis states as

$$\hat{X} : |0\rangle \mapsto |1\rangle, \quad |1\rangle \mapsto |0\rangle, \tag{2.6}$$

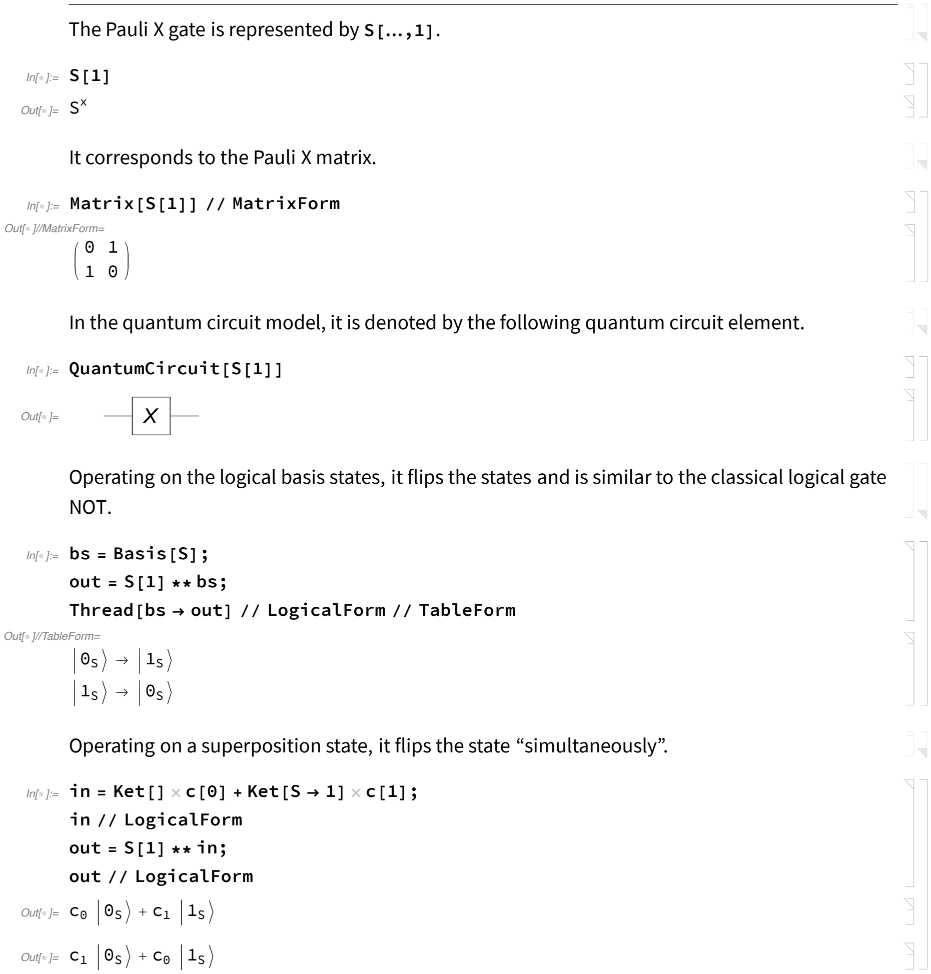
and is similar to the classical logic gate NOT. It is also customary to write $\hat{X}$ in bra-ket notation as

$$\hat{X} = |1\rangle \langle 0| + |0\rangle \langle 1|. \tag{2.7}$$

It is important to remember that like any other quantum gate operations, it can take a linear superposition as input and transform the two logical basis states ‘simultaneously’,

$$\hat{X}(|0\rangle\, c_0 + |1\rangle\, c_1) = |1\rangle\, c_0 + |0\rangle\, c_1\,, \tag{2.8}$$

which is not possible with the classical counterpart NOT.



The Pauli X gate is represented by `S[...,1]`.

```
In[∘]:= S[1]
Out[∘]= Sˣ
```

It corresponds to the Pauli X matrix.

```
In[∘]:= Matrix[S[1]] // MatrixForm
Out[∘]//MatrixForm=
  ( 0  1 )
  ( 1  0 )
```

In the quantum circuit model, it is denoted by the following quantum circuit element.

```
In[∘]:= QuantumCircuit[S[1]]
Out[∘]=
```

Operating on the logical basis states, it flips the states and is similar to the classical logical gate NOT.

```
In[∘]:= bs = Basis[S];
        out = S[1] ** bs;
        Thread[bs → out] // LogicalForm // TableForm
Out[∘]//TableForm=
        |0_S⟩ → |1_S⟩
        |1_S⟩ → |0_S⟩
```

Operating on a superposition state, it flips the state “simultaneously”.

```
In[∘]:= in = Ket[] × c[0] + Ket[S → 1] × c[1];
        in // LogicalForm
        out = S[1] ** in;
        out // LogicalForm
Out[∘]= c₀ |0_S⟩ + c₁ |1_S⟩
Out[∘]= c₁ |0_S⟩ + c₀ |1_S⟩
```

Operating on the logical basis states, Pauli operator $\hat{Z}$ only flips the relative phase of $|1\rangle$,

$$\hat{Z} : |0\rangle \mapsto |0\rangle\,, \quad |1\rangle \mapsto -|1\rangle\,, \tag{2.9}$$

and hence in bra-ket notation, it reads as

$$\hat{Z} = |0\rangle\langle 0| - |1\rangle\langle 1|\ . \tag{2.10}$$

The phase flip is meaningless on classical bits, but it makes a significant difference on a superposition as illustrated in the following example

$$\hat{Z}(|0\rangle\, c_0 + |1\rangle\, c_1) = |0\rangle\, c_0 - |1\rangle\, c_1\,. \tag{2.11}$$

The Pauli Z gate is represented by `S[...,3]`.

```
In[∘]:= S[3]
Out[∘]= S^z
```

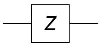
It corresponds to the Pauli Z matrix.

```
In[∘]:= Matrix[S[3]] // MatrixForm
Out[∘]//MatrixForm=
```

$$\begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$$

In the quantum circuit model, it is denoted by the following quantum circuit element.

```
In[∘]:= QuantumCircuit[S[3]]
Out[∘]=
```



Operating on the logical basis states, it "flips the phase", that is, it changes the phase factor to `-1` of the logical basis state $|1\rangle$.

```
In[∘]:= bs = Basis[S];
        out = S[3] ** bs;
        Thread[bs → out] // LogicalForm // TableForm
Out[∘]//TableForm=
```

$$|0_S\rangle \to |0_S\rangle$$
$$|1_S\rangle \to -|1_S\rangle$$

Here is an example how the Pauli Z gate acts on a superposition state.

```
In[∘]:= in = Ket[] × c[0] + Ket[S → 1] × c[1];
        in // LogicalForm
        out = S[3] ** in;
        out // LogicalForm
```

Out[∘]= $c_0 |0_S\rangle + c_1 |1_S\rangle$

Out[∘]= $c_0 |0_S\rangle - c_1 |1_S\rangle$

Pauli operator $\hat{Y}$ combines the bit-flip feature of $\hat{X}$ and the phase-flip feature of $\hat{Z}$ to get

$$|0\rangle \mapsto i\,|1\rangle \ , \quad |1\rangle \mapsto -i\,|0\rangle \ . \tag{2.12}$$

This can also be seen in the operator identity, $\hat{Y} = i\hat{X}\hat{Z}$. In the bra-ket notation, it reads as

$$\hat{Y} = i\,|1\rangle\,\langle 0| - i\,|0\rangle\,\langle 1| \ . \tag{2.13}$$

The Pauli Y gate is represented by `S[...,2]`.

```
In[∘]:= S[2]
Out[∘]= S^y
```

It corresponds to the Pauli Y matrix.

```
In[∘]:= Matrix[S[2]] // MatrixForm
Out[∘]//MatrixForm=
```

$$\begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}$$

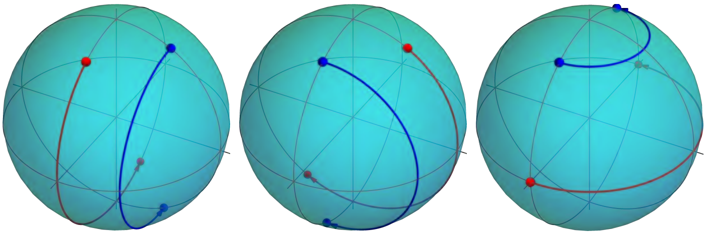
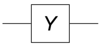


Figure 2.1: Illustration of the actions of Pauli gates as rotations by angle π. From the left, the actions of the Pauli $\hat{X}$, $\hat{Y}$, $\hat{Z}$.

In the quantum circuit model, it is denoted by the following quantum circuit element.

```
In[∘]:= QuantumCircuit[S[2]]
```

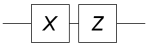
Out[∘]=



Pauli Y is a combination of the bit-flip (Pauli X) and phase-flip (Pauli Z) operation.

```
In[∘]:= qc = QuantumCircuit[S[1], S[3]]
        op = ExpressionFor[qc]
```

Out[∘]=



Out[∘]= $i\, S^y$

This shows more explicitly how Pauli Y “flips” both the bit and phase of the logical basis states.

```
In[∘]:= bs = Basis[S];
        out = S[2] ** bs;
        Thread[bs → out] // LogicalForm // TableForm
```

Out[∘]//TableForm=

$$\left|0_S\right\rangle \to i \left|1_S\right\rangle$$
$$\left|1_S\right\rangle \to -i \left|0_S\right\rangle$$

Here is an example how the Pauli Y gate acts on a superposition state.

```
In[∘]:= in = Ket[] × c[0] + Ket[S → 1] × c[1];
        in // LogicalForm
        out = S[2] ** in;
        out // LogicalForm
```

Out[∘]= $c_0 \left|0_S\right\rangle + c_1 \left|1_S\right\rangle$

Out[∘]= $-i\, c_1 \left|0_S\right\rangle + i\, c_0 \left|1_S\right\rangle$

The Pauli gates can also be regarded as rotations by π around the x-, y-, and z-axis, respectively, in the Bloch sphere as illustrated in Fig. 2.1 and demonstrated in the following:

The Pauli gates also correspond, up to a global phase factor $(-i)$, to rotations by the corresponding axes by angle π. Here `Rotation[ϕ,S[...,μ]]` represents the rotation operator around the μ-axis by angle ϕ.

```
In[∘]:= Rotation[Pi, S[1]]
        Rotation[Pi, S[1]] // Elaborate

Out[∘]= Rotation[π, S^x]

Out[∘]= -i S^x
```

```
In[∘]:= Rotation[Pi, S[2]] // Elaborate

Out[∘]= -i S^y
```

```
In[∘]:= Rotation[Pi, S[3]] // Elaborate

Out[∘]= -i S^z
```

Here we have mainly focused on their roles as unitary operators. However, the Pauli operators play another important role as orthogonal *basis vectors* of the vector space of all linear operators on a two-dimensional vector space (see Section 2.1.3 and Appendix B.1).

2.1.2 Hadamard Gate

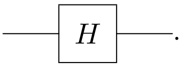
The Hadamard gate is one of the most frequently used elementary gates in many quantum algorithms. The Hadamard gate $\hat{H}$ is defined by the mapping:

$$\hat{H} : |0\rangle \mapsto \frac{|0\rangle + |1\rangle}{\sqrt{2}}, \quad |1\rangle \mapsto \frac{|0\rangle - |1\rangle}{\sqrt{2}}. \tag{2.14}$$

That is, it constructs linear superpositions of the two logical basis states, $|0\rangle$ and $|1\rangle$. This feature makes the Hadamard gate so useful that it is exploited in a wide range of quantum algorithms. In the logical basis, it is represented by the 2×2 Hadamard matrix

$$\hat{H} \doteq \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}. \tag{2.15}$$

In quantum circuit model, the Hadamard gate is depicted by an element with label 'H'



(2.16)

Note that the output states in (2.14) are the eigenstates of the Pauli X operator. One can thus regard the Hadamard gate as a basis transformation from the logical basis to the eigenbasis of the Pauli X gate.

The Hadamard gate is represented by S[..., 6] .

```
In[∘]:= S[1, 6]

Out[∘]= S^H_1
```

Let us consider all the logical basis states.

```
In[∘]:= bs = Basis[S[1]];
        bs // LogicalForm

Out[∘]= {|0_S1⟩, |1_S1⟩}
```

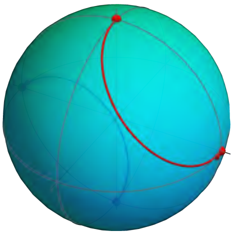


Figure 2.2: Illustration of the action of the Hadamard gate on the Bloch sphere. The Hadamard gate corresponds to a rotation around the axis $(1, 0, 1)$ in the xz-plane by angle π.

Operating the Hadamard gate on them gives the two superposition states.

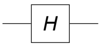
```
In[∘]:= out = S[1, 6] ** bs;
        out // LogicalForm
```

Out[∘]= $\left\{\frac{|0_{S_1}\rangle}{\sqrt{2}} + \frac{|1_{S_1}\rangle}{\sqrt{2}}, \frac{|0_{S_1}\rangle}{\sqrt{2}} - \frac{|1_{S_1}\rangle}{\sqrt{2}}\right\}$

In the quantum circuit model, it is denoted by the following circuit element.

```
In[∘]:= QuantumCircuit[S[1, 6]]
```

Out[∘]=



To provide further insight, note that it can be regarded as a rotation by angle π around the axis $(1, 0, 1)$ on the Bloch sphere (up to a global phase factor). This can be seen from the following:

$$\hat{H} = \frac{1}{\sqrt{2}}\left(\hat{X} + \hat{Z}\right) = i \exp\left[-i\frac{\pi}{2}(\hat{X} + \hat{Z})\right] \tag{2.17}$$

This is illustrated in Fig. 2.2.

Geometrically, the Hadamard gate can be regarded as a rotation around the axis `(1,0,1)` in the Pauli space by angle π.

```
In[∘]:= op = I Rotation[π, S, {1, 0, 1}] // Elaborate
        mat = Matrix[op];
        mat // MatrixForm
```

Out[∘]= $\frac{S^x}{\sqrt{2}} + \frac{S^z}{\sqrt{2}}$

Out[∘]//MatrixForm=

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

An obvious but very useful feature of the Hadamard gate is that it makes a linear superposition of *all* elements in the logical basis. Consider a *quantum* *register* consisting of n qubits. When applied to each qubit in $|0\rangle$, it generates a linear superposition of all states in the logical basis

$$\hat{H}^{\otimes n}\,|0\rangle^{\otimes n} = \frac{|0\rangle + |1\rangle}{\sqrt{2}} \otimes \cdots \otimes \frac{|0\rangle + |1\rangle}{\sqrt{2}} = \frac{1}{2^{n/2}} \sum_{x=0}^{2^n-1} |x\rangle\ , \tag{2.18}$$

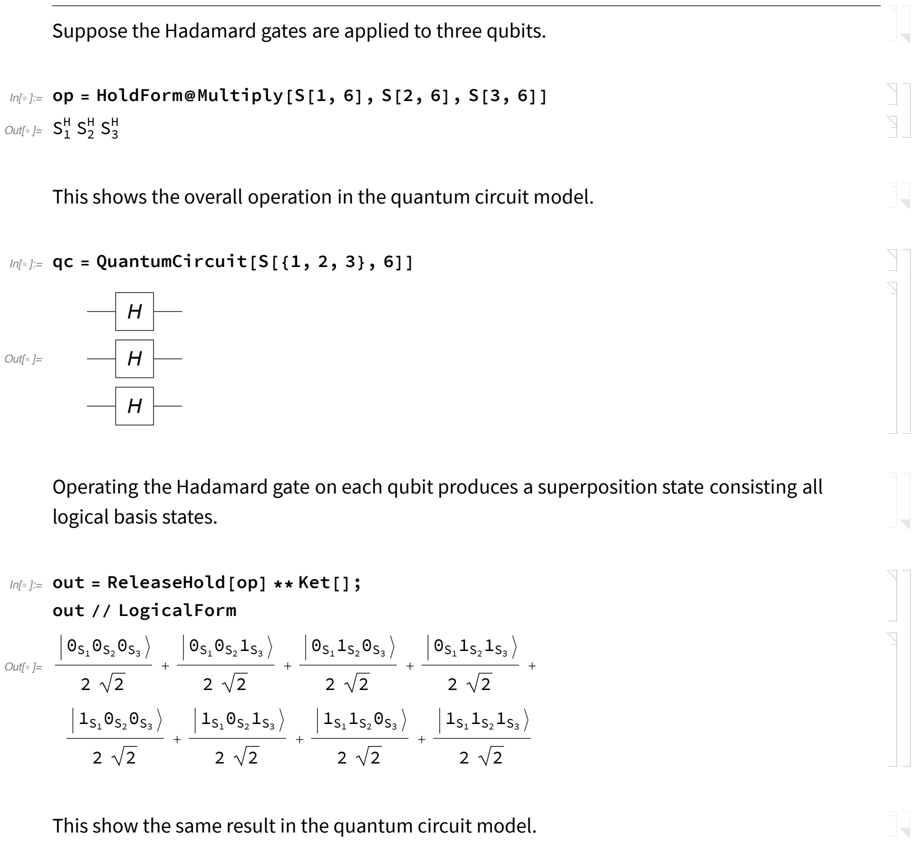
where $|x\rangle := |x_1\rangle \otimes |x_2\rangle \otimes \cdots \otimes |x_n\rangle$ for an integer x represented by $x = (x_1 x_2 \ldots x_n)_2$ in the binary digits. More generally, for an arbitrary state $|y\rangle$ in the logical basis,

$$\hat{H}^{\otimes n}\,|y\rangle = \frac{1}{2^{n/2}} \sum_{x=0}^{2^n-1} |x\rangle\,(-1)^{x\cdot y}\,, \tag{2.19}$$

where we have used a short-hand notation

$$x \cdot y := x_1 y_1 + \cdots + x_n y_n \pmod 2. \tag{2.20}$$

The properties of the Hadamard gate summarized in Eqs. (2.18) and (2.19) are used ubiquitously in almost all quantum algorithms.



Suppose the Hadamard gates are applied to three qubits.

```
In[◦]:= op = HoldForm@Multiply[S[1, 6], S[2, 6], S[3, 6]]
```

Out[◦]= $S_1^H\, S_2^H\, S_3^H$

This shows the overall operation in the quantum circuit model.

```
In[◦]:= qc = QuantumCircuit[S[{1, 2, 3}, 6]]
```

Out[◦]=

Operating the Hadamard gate on each qubit produces a superposition state consisting all logical basis states.

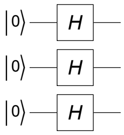
```
In[◦]:= out = ReleaseHold[op] ** Ket[];
        out // LogicalForm
```

Out[◦]= $\frac{|0_{S_1}0_{S_2}0_{S_3}\rangle}{2\sqrt{2}} + \frac{|0_{S_1}0_{S_2}1_{S_3}\rangle}{2\sqrt{2}} + \frac{|0_{S_1}1_{S_2}0_{S_3}\rangle}{2\sqrt{2}} + \frac{|0_{S_1}1_{S_2}1_{S_3}\rangle}{2\sqrt{2}} + \frac{|1_{S_1}0_{S_2}0_{S_3}\rangle}{2\sqrt{2}} + \frac{|1_{S_1}0_{S_2}1_{S_3}\rangle}{2\sqrt{2}} + \frac{|1_{S_1}1_{S_2}0_{S_3}\rangle}{2\sqrt{2}} + \frac{|1_{S_1}1_{S_2}1_{S_3}\rangle}{2\sqrt{2}}$

This show the same result in the quantum circuit model.

```
In[•]:= qc = QuantumCircuit[LogicalForm[Ket[], S@{1, 2, 3}], S[{1, 2, 3}, 6]]
        ExpressionFor[qc] // LogicalForm
```

Out[•]=



Out[•]=

$$\frac{\left|0_{S_1}0_{S_2}0_{S_3}\right\rangle}{2\sqrt{2}}+\frac{\left|0_{S_1}0_{S_2}1_{S_3}\right\rangle}{2\sqrt{2}}+\frac{\left|0_{S_1}1_{S_2}0_{S_3}\right\rangle}{2\sqrt{2}}+\frac{\left|0_{S_1}1_{S_2}1_{S_3}\right\rangle}{2\sqrt{2}}+\frac{\left|1_{S_1}0_{S_2}0_{S_3}\right\rangle}{2\sqrt{2}}+\frac{\left|1_{S_1}0_{S_2}1_{S_3}\right\rangle}{2\sqrt{2}}+\frac{\left|1_{S_1}1_{S_2}0_{S_3}\right\rangle}{2\sqrt{2}}+\frac{\left|1_{S_1}1_{S_2}1_{S_3}\right\rangle}{2\sqrt{2}}$$

Let us compare it with an explicit construction. To do it first prepare the indices of the logical basis states in binary digits.

```
In[•]:= nn = Range[0, 2^3 - 1];
        bit = IntegerDigits[nn, 2, 3]

Out[•]= {{0, 0, 0}, {0, 0, 1}, {0, 1, 0},
          {0, 1, 1}, {1, 0, 0}, {1, 0, 1}, {1, 1, 0}, {1, 1, 1}}
```

This gives an explicit construction (unnormalized) of the superposition of all local basis states.

```
In[•]:= vec = Total[Ket[S@{1, 2, 3} → #] & /@ bit];
        vec // LogicalForm
```

Out[•]=

$$\left|0_{S_1}0_{S_2}0_{S_3}\right\rangle+\left|0_{S_1}0_{S_2}1_{S_3}\right\rangle+\left|0_{S_1}1_{S_2}0_{S_3}\right\rangle+\left|0_{S_1}1_{S_2}1_{S_3}\right\rangle+\left|1_{S_1}0_{S_2}0_{S_3}\right\rangle+\left|1_{S_1}0_{S_2}1_{S_3}\right\rangle+\left|1_{S_1}1_{S_2}0_{S_3}\right\rangle+\left|1_{S_1}1_{S_2}1_{S_3}\right\rangle$$

On other elements of the logical basis, the sign of each term is determined by the bitwise dot product of the its bit-string with that of the input state.

```
In[•]:= in = Ket[S[{1, 2, 3}] → {1, 0, 1}];
        in = LogicalForm[in, S@{1, 2, 3}]
```

Out[•]= $\left|1_{S_1}0_{S_2}1_{S_3}\right\rangle$

```
In[•]:= out = ReleaseHold[op] ** in;
        out // LogicalForm
```

Out[•]=

$$\frac{\left|0_{S_1}0_{S_2}0_{S_3}\right\rangle}{2\sqrt{2}}-\frac{\left|0_{S_1}0_{S_2}1_{S_3}\right\rangle}{2\sqrt{2}}+\frac{\left|0_{S_1}1_{S_2}0_{S_3}\right\rangle}{2\sqrt{2}}-\frac{\left|0_{S_1}1_{S_2}1_{S_3}\right\rangle}{2\sqrt{2}}-\frac{\left|1_{S_1}0_{S_2}0_{S_3}\right\rangle}{2\sqrt{2}}+\frac{\left|1_{S_1}0_{S_2}1_{S_3}\right\rangle}{2\sqrt{2}}-\frac{\left|1_{S_1}1_{S_2}0_{S_3}\right\rangle}{2\sqrt{2}}+\frac{\left|1_{S_1}1_{S_2}1_{S_3}\right\rangle}{2\sqrt{2}}$$

2.1.3 Rotations

Any unitary operator $\hat{U}$ can always be written in the form $\hat{U} = \exp(-i\hat{H})$ with a Hermitian operator $\hat{H}$. On a two-dimensional vector space $\mathcal{S}$ associated with a qubit, any Hermitian operator $\hat{H}$ can be expanded in terms of the Pauli operators $\hat{S}^\mu$ as

$$\hat{H} = \hat{I}\phi_0 + \hat{S}^x B_x + \hat{S}^y B_y + \hat{S}^z B_z \tag{2.21}$$

where ϕ_0, B_x, B_y, B_z are real parameters. Regarding $\boldsymbol{B} := (B_x, B_y, B_z)$ as a three-dimensional vector, we consider the unit vector $\boldsymbol{n}$ pointing to the same direction

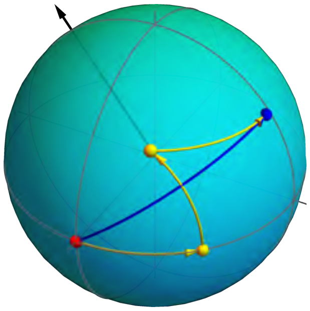


Figure 2.3: Visualization of the transformations of states under single-qubit operations. Up to a global phase factor, a single-qubit unitary operation is a rotation on the Bloch sphere. The rotation around the axis indicated by the black arrow is depicted by the blue arrow. The same rotation can be achieved by combining three rotations around the y- or z-axis depicted by the yellow arrows.

as $\boldsymbol{B}$ and another real parameter $\phi := 2|\boldsymbol{B}|$, where the factor 2 is just for later convenience. In terms of these new parameterization, $\hat{H}$ reads as

$$\hat{H} = \hat{I}\phi_0 + \hat{\boldsymbol{S}} \cdot \boldsymbol{n}\,\phi/2\,, \tag{2.22}$$

where $\hat{\boldsymbol{S}} := (\hat{S}^x, \hat{S}^y, \hat{S}^z)$. In short, any unitary operator on $\mathcal{S}$ has the form $\hat{U} = e^{-i\phi_0}\hat{U}_{\boldsymbol{n}}(\phi)$ with

$$\hat{U}_{\boldsymbol{n}}(\phi) := \exp(-i\hat{\boldsymbol{S}} \cdot \boldsymbol{n}\,\phi/2). \tag{2.23}$$

Here $e^{-i\phi_0}$ changes the global phase factor and is physically irrelevant. The more important and interesting part is $\hat{U}_{\boldsymbol{n}}(\phi)$, which as we will see below, describes a "rotation" around the axis $\boldsymbol{n}$ by the angle ϕ. The rotations here are not in the real three-dimensional world but on the Bloch sphere, as illustrated in Fig. 2.3, corresponding to the two-dimensional vector space $\mathcal{S}$. We will further denote the rotations around the μ-axis—$\boldsymbol{n}$ parallel to the μ-axis—of the Bloch sphere by $\hat{U}_\mu(\phi)$.

To see that the unitary operation $\hat{U}_{\boldsymbol{n}}(\phi)$ in (2.23) corresponds to a rotation, recall that the Pauli operators $\hat{S}^\mu$ are the spin angular momentum operators of spin 1/2. That is, they are the generators of rotations and satisfy the commutation relations

$$[\hat{S}^\mu, \hat{S}^\nu] = 2i\sum_\lambda \hat{S}^\lambda \epsilon_{\lambda\mu\nu}, \tag{2.24}$$

where $\epsilon_{\lambda\mu\nu}$ is the Levi-Civita symbol. The connection of unitary operator $\hat{U}_\lambda(\phi)$ to rotation is seen more explicitly in the equivalent relation

$$\hat{U}_\lambda(\phi)\hat{S}^\nu\hat{U}_\lambda^\dagger(\phi) = \sum_\mu \hat{S}^\mu[R_\lambda(\phi)]_{\mu\nu}\,, \tag{2.25}$$

where $R_\lambda(\phi)$ is the 3×3 orthogonal matrix describing the rotation of three-dimensional coordinates around the λ-axis by angle ϕ.

The Pauli operators are generators of the rotational transformations in a two-dimensional complex vector space, and hence satisfy the fundamental commutation relations of angular momentum operators (up to a normalization factor).

```
In[∘]:= op = S[All]
Out[∘]= {Sˣ, Sʸ, Sᶻ}
```

```
In[∘]:= in = Outer[HoldForm@*Commutator, op, op];
        out = ReleaseHold[in];
        Thread[Flatten[in] → Flatten[out]] // TableForm
Out[∘]//TableForm=
        Commutator[Sˣ, Sˣ] → 0
        Commutator[Sˣ, Sʸ] → 2 ⅈ Sᶻ
        Commutator[Sˣ, Sᶻ] → -2 ⅈ Sʸ
        Commutator[Sʸ, Sˣ] → -2 ⅈ Sᶻ
        Commutator[Sʸ, Sʸ] → 0
        Commutator[Sʸ, Sᶻ] → 2 ⅈ Sˣ
        Commutator[Sᶻ, Sˣ] → 2 ⅈ Sʸ
        Commutator[Sᶻ, Sʸ] → -2 ⅈ Sˣ
        Commutator[Sᶻ, Sᶻ] → 0
```

In $\mathbb{R}^3$, any 3×3 rotation matrix can be decomposed into three factors

$$R = R_z(\alpha)R_y(\beta)R_z(\gamma)\,, \tag{2.26}$$

where α, β, γ are the so-called *Euler angles* and such a combination of rotations is called the *Euler rotation*. In the same manner, any unitary operator on $\mathcal{S}$ can also be written as

$$\hat{U} = e^{-i\phi_0}\hat{U}_z(\alpha)\hat{U}_y(\beta)\hat{U}_z(\gamma)\,, \tag{2.27}$$

that is, a combination of elementary "rotations" around the y- and z-axis and an additional overall phase shift. The unitary operator $\hat{U}(\alpha,\beta,\gamma) := \hat{U}_z(\alpha)\hat{U}_y(\beta)\hat{U}_z(\gamma)$ is called the Euler rotation in the two-dimensional vector space $\mathcal{S}$. Figure 2.3 illustrates an Euler rotation $\hat{U}(\pi/3, -\pi/3, \pi/4)$. It transforms $|v\rangle = (|0\rangle + |1\rangle)/\sqrt{2}$ (the red dot in Fig. 2.3) into $|w\rangle = \hat{U}\,|v\rangle$ (the blue dot in Fig. 2.3). The transformation is displayed by the blue arrow. The yellow dots and arrows depicts the transformations under $\hat{U}_z(\alpha)$, $\hat{U}_y(\beta)$, and $\hat{U}_z(\gamma)$, which combine to reproduce $\hat{U}$.

Consider a unitary operator represented by the following matrix.

```
In[∘]:= mat = {
           {3 - I Sqrt[3], I - Sqrt[3]},
           {I + Sqrt[3], 3 + I Sqrt[3]}
         } / 4;
        mat // MatrixForm
Out[∘]//MatrixForm=
```

$$\begin{pmatrix} \frac{1}{4}\left(3 - i\sqrt{3}\right) & \frac{1}{4}\left(i - \sqrt{3}\right) \\ \frac{1}{4}\left(i + \sqrt{3}\right) & \frac{1}{4}\left(3 + i\sqrt{3}\right) \end{pmatrix}$$

This gives its expression in terms of the Pauli operators.

```
In[•]:= op = Elaborate@ExpressionFor[mat, S]

         3    ⅈ Sˣ   1              1
Out[•]=  ─ + ──── - ─ ⅈ √3 Sʸ - ─ ⅈ √3 Sᶻ
         4     4     4              4

In[•]:= Dagger[op] ** op

Out[•]= 1
```

This gives the Euler angles of the unitary operator.

```
In[•]:= angs = TheEulerAngles[mat]

          π   π
Out[•]= { ─ , ─ , 0}
          3   3
```

Indeed, the Euler angles reproduces the original unitary operator.

```
In[•]:= new = EulerRotation[angs, S] // Elaborate
        op - new

         3    ⅈ Sˣ   1              1
Out[•]=  ─ + ──── - ─ ⅈ √3 Sʸ - ─ ⅈ √3 Sᶻ
         4     4     4              4

Out[•]= 0
```

The rotations $\hat{U}_z(\phi)$ around the z-axis in the Bloch space induces the relative phase difference ϕ between the two logical basis states, $|0\rangle$ and $|1\rangle$. In this sense, such rotations are called (relative) phase gates by phase angle ϕ. Two phase gates $\hat{Q}$ and $\hat{O}$ by angles $2\pi/4$ and $2\pi/8$, respectively, are particularly common, and they are often called the *quadrant* and *octant phase gate*. In the logical basis, they are represented by the following diagonal matrices

$$\hat{Q} \doteq \begin{bmatrix} 1 & 0 \\ 0 & i \end{bmatrix}, \quad \hat{O} \doteq \begin{bmatrix} 1 & 0 \\ 0 & e^{i\pi/4} \end{bmatrix}, \tag{2.28}$$

respectively. Obviously, $\hat{O}^2 = \hat{Q}$ and $\hat{Q}^2 = \hat{Z}$.

Among (relative) phase gates, the quadrant and octant phase gates are most common.

```
In[•]:= qd = S[1, 7]
        Matrix[qd] // MatrixForm

Out[•]= S₁ˢ

Out[•]//MatrixForm=
        (1 0)
        (0 ⅈ)

In[•]:= qd ** qd

Out[•]= S₁ᶻ

In[•]:= oc = S[1, 8]
        Matrix[oc] // MatrixForm

Out[•]= S₁ᵀ

Out[•]//MatrixForm=
        (1  0  )
        (   ⅈπ )
        (   ── )
        (0  ⅇ 4)

In[•]:= oc ** oc

Out[•]= S₁ˢ
```

2.2 Two-Qubit Gates

Next, let us consider quantum logic gate operations acting on two qubits. Such operations are represented by 4×4 unitary matrices. We will see that any two-qubit gate operations can be decomposed into controlled-unitary gates. A controlled-unitary gate acts a unitary operator on one qubit depending on the logical state of the other qubit. A controlled-unitary gate on two qubits can be further decomposed into factors including only the CNOT gate and single-qubit rotation gates. In this sense, the CNOT gate alone is sufficient for any two-qubit gate.

The controlled-unitary and CNOT gates have various interesting properties that make them useful in implementing quantum algorithms. In this section, we first examine the basic properties of the CNOT gate, and in particular, how it is used to generate an entanglement between two qubits. Then, we discuss the properties of the controlled-unitary gate and describe how to implement a controlled-unitary gate in terms of the CNOT gate and single-qubit rotations. Finally, we discuss how an arbitrary two-qubit unitary operation can be decomposed into controlled-unitary gates.

2.2.1 CNOT, CZ, and SWAP

The CNOT or controlled-NOT gate is a quantum logic gate on two qubits that maps the logical basis states as

$$|c\rangle \otimes |t\rangle \mapsto |c\rangle \otimes |c \oplus t\rangle \;; \quad c, t \in \{0, 1\} \;, \tag{2.29}$$

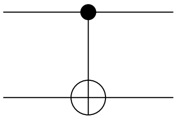
where the first qubit is typically called the *control qubit* (c) and the second qubit the *target qubit* (t). It has the following matrix representation in the logical basis

$$\mathrm{CNOT} \doteq \begin{bmatrix} 1 & & & \\ & 1 & & \\ & & 0 & 1 \\ & & 1 & 0 \end{bmatrix} , \tag{2.30}$$

and is expressed in terms of the Pauli operators on the control and target qubit as

$$\mathrm{CNOT} = \frac{1}{2}\left(\hat{I} + \hat{S}_c^z + \hat{S}_t^z - \hat{S}_c^z \hat{S}_t^x\right) . \tag{2.31}$$

In quantum circuit model, it is represented as the following circuit element



$$, \tag{2.32}$$

where the smaller filled circle indicates the dependence on the state of the control qubit and the circled-plus sign denotes the conditional NOT action on the target qubit.

CNOT[*control*, *target*] denotes the CNOT gate in the quantum circuit model.

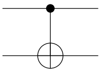
```
In[ ]:= op = CNOT[S[1], S[2]]
```

Out[ ]= $\mathrm{CNOT}[\{S_1\}, \{S_2\}]$

This displays a quantum circuit including the CNOT gate.

```
In[ ]:= qc = QuantumCircuit[op]
```

Out[ ]=



This is the explicit expression of the CNOT gate in terms of the Pauli operators.

```
In[ ]:= op = ExpressionFor[qc]
```

Out[ ]= $\frac{1}{2} - \frac{1}{2} S_1^z S_2^x + \frac{S_1^z}{2} + \frac{S_2^x}{2}$

This is the matrix representation of the CNOT gate in the logical basis.

```
In[ ]:= Matrix[op] // MatrixForm
```

Out[ ]//MatrixForm=

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

This shows how the CNOT gate operates on the logical basis states.

```
In[ ]:= in = Basis[S@{1, 2}];
        out = op ** in;
        Thread[in → out] // LogicalForm // TableForm
```

Out[ ]//TableForm=

$|0_{S_1}0_{S_2}\rangle \rightarrow |0_{S_1}0_{S_2}\rangle$
$|0_{S_1}1_{S_2}\rangle \rightarrow |0_{S_1}1_{S_2}\rangle$
$|1_{S_1}0_{S_2}\rangle \rightarrow |1_{S_1}1_{S_2}\rangle$
$|1_{S_1}1_{S_2}\rangle \rightarrow |1_{S_1}0_{S_2}\rangle$

A simple yet important feature of CNOT is to copy the logical state of the control qubit to the target bit provided that the target bit is initially set to $|0\rangle$; or the reversed state when the target qubit is in $|1\rangle$. A vital implication is that CNOT generates an entangled state when the control qubit is in a superposition:

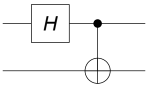
$$(|0\rangle c_0 + |1\rangle c_1) \otimes |0\rangle \mapsto |0\rangle \otimes |0\rangle c_0 + |1\rangle \otimes |1\rangle c_1. \tag{2.33}$$

Applying a single qubit rotation such the Hadamard gate on the control qubit prior to CNOT, one can thus generate entangled states from logical states. In this sense, such a circuit is called a *quantum entangler circuit*.

As an example of the application of the CNOT gate, this shows an entangler quantum circuit.

```
In[ ]:= entangler = QuantumCircuit[S[1, 6], CNOT[S[1], S[2]]]
```

Out[ ]=

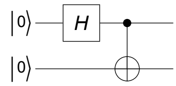


For example, when the input state is $|0\rangle \otimes |0\rangle$, the outcome of the circuit is given by one of the so-called Bell states.

This demonstrates the generation of an entangled state from a product state.

```
In[•]:= new = QuantumCircuit[LogicalForm[Ket[S@{1, 2} → {0, 0}], S@{1, 2}], entangler]
        vec = ExpressionFor[new];
        vec // LogicalForm
```

Out[•]=



Out[•]= $\frac{|0_{S_1}0_{S_2}\rangle}{\sqrt{2}} + \frac{|1_{S_1}1_{S_2}\rangle}{\sqrt{2}}$

In general, the different product states in the logical basis are transformed to different Bell states.

This lists the mapping between the standard tensor-product basis states and the Bell states.

```
In[•]:= bs = Basis@S@{1, 2};
        op = ExpressionFor[entangler];
        out = op ** bs;
        table = Thread[bs → out];
        table // LogicalForm // TableForm
```

Out[•]//TableForm=

$|0_{S_1}0_{S_2}\rangle \to \frac{|0_{S_1}0_{S_2}\rangle}{\sqrt{2}} + \frac{|1_{S_1}1_{S_2}\rangle}{\sqrt{2}}$

$|0_{S_1}1_{S_2}\rangle \to \frac{|0_{S_1}1_{S_2}\rangle}{\sqrt{2}} + \frac{|1_{S_1}0_{S_2}\rangle}{\sqrt{2}}$

$|1_{S_1}0_{S_2}\rangle \to \frac{|0_{S_1}0_{S_2}\rangle}{\sqrt{2}} - \frac{|1_{S_1}1_{S_2}\rangle}{\sqrt{2}}$

$|1_{S_1}1_{S_2}\rangle \to \frac{|0_{S_1}1_{S_2}\rangle}{\sqrt{2}} - \frac{|1_{S_1}0_{S_2}\rangle}{\sqrt{2}}$

One can further generalize the above procedure to generate maximally entangled states between larger systems: Consider two *quantum registers*, each consisting of n qubits. We respectively call them the "control" and "target" register for the reason that will be clear below. Upon applying the CNOT gate on each pair of the corresponding qubits in the two registers as in the quantum circuit shown in Fig. 2.4 (a), the logical basis states are transformed as

$$|c\rangle \otimes |t\rangle \mapsto |c\rangle \otimes |c \oplus t\rangle \ . \tag{2.34}$$

The above association rule is *formally* the same as the one in (2.29) for a single pair of qubits. Here, however, $|c\rangle := |c_1\rangle \otimes |c_2\rangle \otimes \cdots \otimes |c_n\rangle$ and $|t\rangle := |t_1\rangle \otimes |t_2\rangle \otimes \cdots \otimes |t_n\rangle$, and $c \oplus t$ denotes the bit-wise exclusive OR (or XOR),

$$c \oplus t := (c_1 \oplus t_1, c_2 \oplus t_2, \ldots, c_n \oplus t_n), \tag{2.35}$$

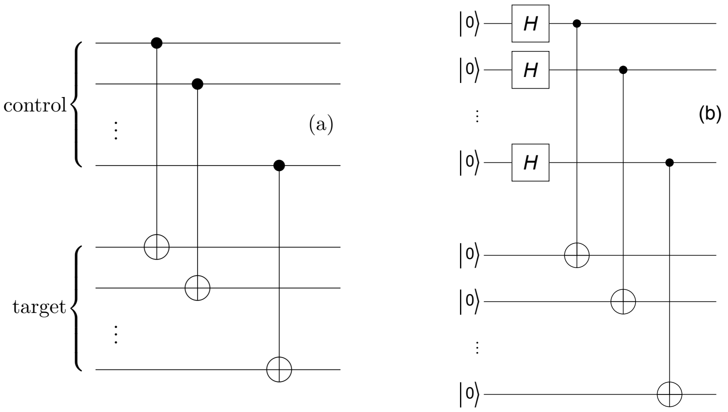


Figure 2.4: (a) A quantum circuit which makes a copy of the logical state of the "control" quantum register to the "target" quantum register. The quantum circuit transforms the logical basis states as $|c\rangle \otimes |t\rangle \mapsto |c\rangle \otimes |c \oplus t\rangle$, where c and t are n-bit strings. (b) A quantum circuit generating maximally entangled states between two quantum registers each of which consists of n qubits.

for the n-bit strings $c \equiv (c_1, c_2, \ldots, c_n)$ and $t \equiv (t_1, t_2, \ldots, t_n)$. When the target register is prepared in the state $|\mathbf{0}\rangle \equiv |0\rangle^{\otimes n}$, the logical basis state of the control register is copied to the target register,[2.1]

$$|x\rangle \otimes |\mathbf{0}\rangle \mapsto |x\rangle \otimes |x\rangle \tag{2.36}$$

under the set of CNOT gates on paired qubits. More interestingly, when the control register is prepared in a superposition, say, $2^{-n/2} \sum_{x=0}^{2^n-1} |x\rangle$, and the target register is in the state $|\mathbf{0}\rangle \equiv |0\rangle$, the transformation in (2.34) makes a copy of each logical state of the control register to the target register, leading to a maximally entangled state,

$$\frac{1}{2^{n/2}} \sum_{x=0}^{2^n-1} |x\rangle \otimes |0\rangle \mapsto |\Phi\rangle := \frac{1}{2^{n/2}} \sum_{x=0}^{2^n-1} |x\rangle \otimes |x\rangle \tag{2.37}$$

between the two registers (rather than single qubits). Since the superposition $2^{-n/2} \sum_{x=0}^{2^n-1} |x\rangle$ is obtained just by the Hadamard gates as shown in (2.18), the maximally entangled state $|\Phi\rangle$ in (2.37) can be generated from the logical state $|0\rangle \otimes |0\rangle$ through the quantum circuit illustrated in Fig. 2.4 (b). It is also interesting to note that even though $|\Phi\rangle$ is a maximally entangled state between the control and target registers, it is a product state of n maximally entangled pairs

$$\frac{1}{2^{n/2}} \sum_{x=0}^{2^n-1} |x\rangle \otimes |x\rangle = \bigotimes_{j=1}^{n} \left(\frac{|0\rangle_j \otimes |0\rangle_{n+j} + |1\rangle_j \otimes |1\rangle_{n+j}}{\sqrt{2}} \right). \tag{2.38}$$

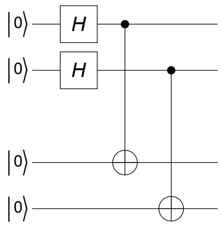
[2.1] It is important to note here that only *logical basis states* can be copied, but not a superposition of them. The latter is forbidden by the no-cloning theorem (Wooters & Zurek, 1982; Zurek, 2000) discussed in Section 1.3.

The entanglement strongly depends on how you partition the system.

Here we want to generate a maximally entangled state between two quantum *registers* (not just qubits) each of which is consisting of two qubits.

```
In[•]:= qc = QuantumCircuit[LogicalForm[Ket[], S@{1, 2, 3, 4}],
          S[{1, 2}, 6], CNOT[S[1], S[3]], CNOT[S[2], S[4]], "Invisible" → S[2.5]]
```

Out[•]=



```
In[•]:= out = ExpressionFor[qc];
        out // LogicalForm
```

Out[•]= $\frac{1}{2}\left|0_{S_1}0_{S_2}0_{S_3}0_{S_4}\right\rangle + \frac{1}{2}\left|0_{S_1}1_{S_2}0_{S_3}1_{S_4}\right\rangle + \frac{1}{2}\left|1_{S_1}0_{S_2}1_{S_3}0_{S_4}\right\rangle + \frac{1}{2}\left|1_{S_1}1_{S_2}1_{S_3}1_{S_4}\right\rangle$

This example illustrate that the entanglement depends on the partition of the system. Indeed, the above system is a product state for the partition (1,3) and (2,4) qubits.

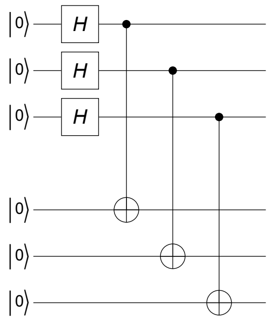
```
In[•]:= KetFactor[out]
```

Out[•]= $\frac{1}{2}\left(\left|0_{S_1}0_{S_3}\right\rangle + \left|1_{S_1}1_{S_3}\right\rangle\right) \otimes \left(\left|0_{S_2}0_{S_4}\right\rangle + \left|1_{S_2}1_{S_4}\right\rangle\right)$

The above construction can be generalized for a pair of *n*-qubit registers.

```
In[•]:= n = 3;
        qc = QuantumCircuit[LogicalForm[Ket[], S@Range[2 n]], S[Range[n], 6],
          Sequence @@ Table[CNOT[S[j], S[n + j]], {j, 1, n}], "Invisible" → S[n + 1 / 2]]
```

Out[•]=



```
In[•]:= out = ExpressionFor[qc];
        out // LogicalForm
```

Out[•]= $\frac{\left|0_{S_1}0_{S_2}0_{S_3}0_{S_4}0_{S_5}0_{S_6}\right\rangle}{2\sqrt{2}} + \frac{\left|0_{S_1}0_{S_2}1_{S_3}0_{S_4}0_{S_5}1_{S_6}\right\rangle}{2\sqrt{2}} + \frac{\left|0_{S_1}1_{S_2}0_{S_3}0_{S_4}1_{S_5}0_{S_6}\right\rangle}{2\sqrt{2}} + \frac{\left|0_{S_1}1_{S_2}1_{S_3}0_{S_4}1_{S_5}1_{S_6}\right\rangle}{2\sqrt{2}} + \frac{\left|1_{S_1}0_{S_2}0_{S_3}1_{S_4}0_{S_5}0_{S_6}\right\rangle}{2\sqrt{2}} + \frac{\left|1_{S_1}0_{S_2}1_{S_3}1_{S_4}0_{S_5}1_{S_6}\right\rangle}{2\sqrt{2}} + \frac{\left|1_{S_1}1_{S_2}0_{S_3}1_{S_4}1_{S_5}0_{S_6}\right\rangle}{2\sqrt{2}} + \frac{\left|1_{S_1}1_{S_2}1_{S_3}1_{S_4}1_{S_5}1_{S_6}\right\rangle}{2\sqrt{2}}$

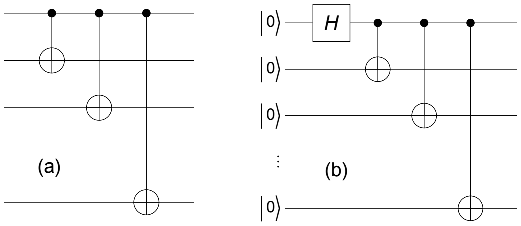


Figure 2.5: (a) A quantum circuit copying the logical basis state of a single control qubit to multiple target qubits. (b) A quantum circuit model to generate the Greenberger-Horne-Zeilinger state among multiple qubits.

Another generalization of copying logical basis states to a series of qubits by consecutively applying CNOT gates with a single control qubit on multiple target qubits is shown in the quantum circuit in Fig. 2.5 (a), and another equivalent quantum circuit is given in Problem 2.5. When the target qubits are all prepared in the logical basis state $|0\rangle$, the series of CNOT gates makes the transformation

$$|x\rangle \otimes |0\rangle \otimes \cdots \otimes |0\rangle \mapsto |x\rangle \otimes |x\rangle \otimes \cdots \otimes |x\rangle \ , \tag{2.39}$$

where $x = 0, 1$. Therefore, a simple application of the Hadamard gate in the control qubit before the CNOT gates can generate a state of the form

$$|\mathrm{GHZ}\rangle = \frac{|0\rangle \otimes |0\rangle \otimes \cdots \otimes |0\rangle + |1\rangle \otimes |1\rangle \otimes \cdots \otimes |1\rangle}{\sqrt{2}} \,. \tag{2.40}$$

The state in (2.40), which is known as the Greenberger-Horne-Zeilinger (GHZ) state (Greenberger *et al.*, 1989), exhibits a quantum entanglement of more than two particles. It has stimulated tests of non-locality of quantum mechanics beyond Bell inequalities (Bouwmeester *et al.*, 1999; Pan *et al.*, 2000).

Quantum entanglement is a valuable resource in quantum information processing and in quantum communication. The most popular example of such is quantum teleportation, which will be discussed in Section 4.1. The features elucidated in (2.36), (2.37) and (2.39)—and the related quantum circuits in Figs. 2.4 and 2.5—will be used frequently in later parts of the book.

In the above consideration, CNOT gates with the same control qubit are applied on different target qubits. The opposite setting brings forth additional applications of CNOT as illustrated in Problem 2.9.

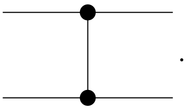
An interesting variant of CNOT gates is the so-called CZ or controlled-Z gate. This is a quantum logic gate on two qubits that maps the logical basis states as

$$\mathrm{CZ} : |c\rangle \otimes |t\rangle \mapsto |c\rangle \otimes |t\rangle \, (-1)^{ct} \,. \tag{2.41}$$

The matrix representation of the CZ gate in the logical basis is given by

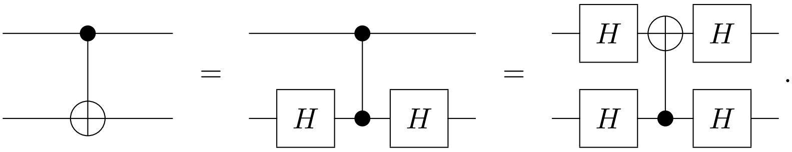
$$\mathrm{CZ} \doteq \begin{bmatrix} 1 & & & \\ & 1 & & \\ & & 1 & \\ & & & -1 \end{bmatrix}. \tag{2.42}$$

Since it is symmetric for the two qubits, a distinction of the control and target qubit is meaningless. Accordingly, in quantum circuit model, the gate is depicted by the following quantum circuit element



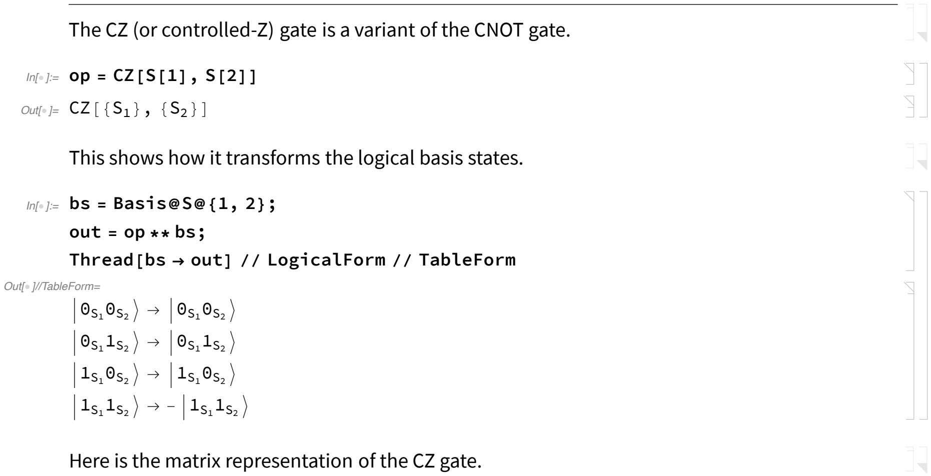
(2.43)

The filled circles on both qubit lines (rather than a square box on either qubit) indicate that the bit values of both qubits remain unchanged. Noting the identity $\hat{H}\hat{Z}\hat{H} = \hat{X}$, one can regard that the CZ gate is equal to the CNOT gate up to the Hadamard gate on the target qubit. The relation between the CNOT and CZ gate is illustrated in the following quantum circuits



(2.44)

Depending on the Hamiltonian of a particular physical system, direct realization of the CNOT gate may be considerably difficult while the CZ gate is relatively easier to implement. In such a case, identity (2.44) offers a straightforward workaround for a physical implementation of the CNOT gate.



The CZ (or controlled-Z) gate is a variant of the CNOT gate.

```
In[ ]:= op = CZ[S[1], S[2]]
Out[ ]= CZ[{S1}, {S2}]
```

This shows how it transforms the logical basis states.

```
In[ ]:= bs = Basis@S@{1, 2};
        out = op ** bs;
        Thread[bs → out] // LogicalForm // TableForm
Out[ ]//TableForm=
```

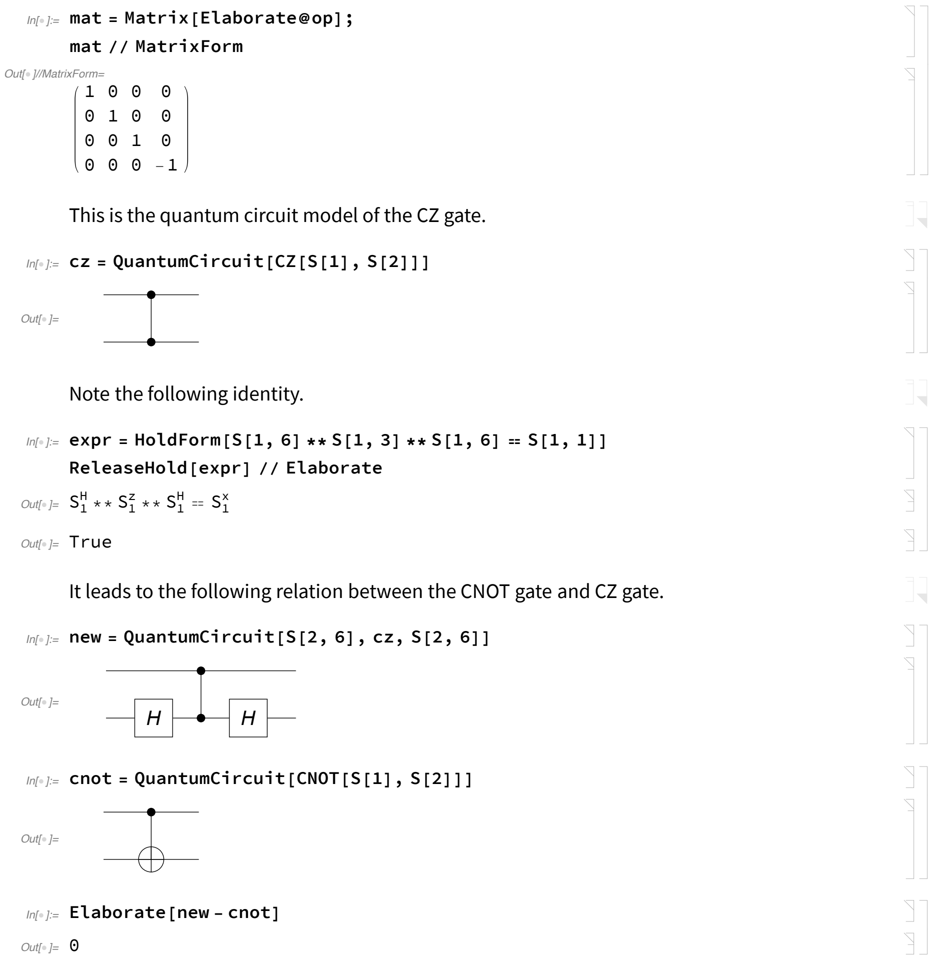
$|0_{S_1}0_{S_2}\rangle \to |0_{S_1}0_{S_2}\rangle$

$|0_{S_1}1_{S_2}\rangle \to |0_{S_1}1_{S_2}\rangle$

$|1_{S_1}0_{S_2}\rangle \to |1_{S_1}0_{S_2}\rangle$

$|1_{S_1}1_{S_2}\rangle \to -|1_{S_1}1_{S_2}\rangle$

Here is the matrix representation of the CZ gate.



```
In[ ]:= mat = Matrix[Elaborate@op];
        mat // MatrixForm
Out[ ]//MatrixForm=
        1 0 0  0
        0 1 0  0
        0 0 1  0
        0 0 0 -1
```

This is the quantum circuit model of the CZ gate.

```
In[ ]:= cz = QuantumCircuit[CZ[S[1], S[2]]]
Out[ ]=
```

Note the following identity.

```
In[ ]:= expr = HoldForm[S[1, 6] ** S[1, 3] ** S[1, 6] == S[1, 1]]
        ReleaseHold[expr] // Elaborate
```

Out[ ]= $S_1^H ** S_1^z ** S_1^H == S_1^x$

```
Out[ ]= True
```

It leads to the following relation between the CNOT gate and CZ gate.

```
In[ ]:= new = QuantumCircuit[S[2, 6], cz, S[2, 6]]
Out[ ]=
```

```
In[ ]:= cnot = QuantumCircuit[CNOT[S[1], S[2]]]
Out[ ]=
```

```
In[ ]:= Elaborate[new - cnot]
Out[ ]= 0
```

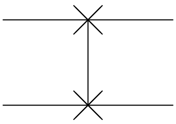
Another interesting two-qubit gate is the SWAP gate that 'swaps' the states of the two qubits and maps the logical basis states as

$$\mathrm{SWAP} : |x_1\rangle \otimes |x_2\rangle \mapsto |x_2\rangle \otimes |x_1\rangle \ . \tag{2.45}$$

Given that the states $|0\rangle \otimes |0\rangle$ and $|1\rangle \otimes |1\rangle$ are not altered by the operation, the matrix representation is given by

$$\mathrm{SWAP} \doteq \begin{bmatrix} 1 & & & \\ & 0 & 1 & \\ & 1 & 0 & \\ & & & 1 \end{bmatrix} . \tag{2.46}$$

In quantum circuit model, it is depicted as



(2.47)

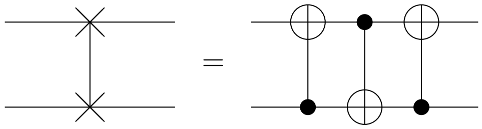
The SWAP gate can be implemented using the CNOT gate. To see this, first note that a simultaneous exchange of the second and fourth columns and rows of the matrix in (2.46) leads to

$$\begin{bmatrix} 1 & & & \\ & 1 & & \\ & & 0 & 1 \\ & & 1 & 0 \end{bmatrix}, \tag{2.48}$$

which is nothing other than the matrix representation of the CNOT gate. On the other hand, the exchange of the second and fourth columns and rows is described by the transformation matrix

$$\begin{bmatrix} 1 & & & \\ & 0 & & 1 \\ & & 1 & \\ & 1 & & 0 \end{bmatrix}, \tag{2.49}$$

which flips the bit values of the first qubit only when the second qubit is set to $|1\rangle$. Hence, it corresponds to the CNOT gate with the second and first qubit as the control and target qubit, respectively. In short, the SWAP gate can be achieved by a combination of the CNOT gates as follows



(2.50)

The SWAP gate exchanges the states of two qubits.

In[ ]:=
```
op = SWAP[S[1], S[2]]
```
Out[ ]=
```
SWAP[S₁, S₂]
```

In[ ]:=
```
bs = Basis@S@{1, 2};
new = op ** bs;
Thread[bs → new] // LogicalForm // TableForm
```
Out[ ]//TableForm=

$|0_{S_1}0_{S_2}\rangle \to |0_{S_1}0_{S_2}\rangle$

$|0_{S_1}1_{S_2}\rangle \to |1_{S_1}0_{S_2}\rangle$

$|1_{S_1}0_{S_2}\rangle \to |0_{S_1}1_{S_2}\rangle$

$|1_{S_1}1_{S_2}\rangle \to |1_{S_1}1_{S_2}\rangle$

This is the matrix representation of the SWAP gate in the logical basis.

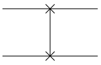
In[ ]:=
```
Matrix[Elaborate@op] // MatrixForm
```
Out[ ]//MatrixForm=

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

In the quantum circuit model, the SWAP gate is represented as following.

In[∘]:=
```
qc = QuantumCircuit[SWAP[S[1], S[2]]]
```

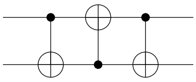
Out[∘]=



The SWAP gate can be implemented by means of the CNOT gate.

In[∘]:=
```
new = QuantumCircuit[CNOT[S[1], S[2]], CNOT[S[2], S[1]], CNOT[S[1], S[2]]]
```

Out[∘]=



In[∘]:=
```
Elaborate[qc - new]
```

Out[∘]=
```
0
```

Interestingly, the SWAP gate itself is not universal, but the $\sqrt{\text{SWAP}}$ gate—the gate that equals SWAP when squared—is universal. That is, any quantum gate on a multi-qubit system can be implemented by combining $\sqrt{\text{SWAP}}$ and single-qubit rotations (see also Section 2.4 and Section 3.2.2). Indeed, one can combine the $\sqrt{\text{SWAP}}$ gate with single-qubit rotations to construct the CZ gate (Section 3.2.2). As discussed above, the CZ gate requires just two more Hadamard gates to implement the CNOT gate, and it is hence universal.

Let us construct the CZ gate with the $\sqrt{\text{SWAP}}$ gate. This is the matrix representation of the the $\sqrt{\text{SWAP}}$ gate.

In[∘]:=
```
mat = MatrixPower[Matrix@Elaborate@SWAP[S[1], S[2]], 1 / 2];
mat // MatrixForm
```

Out[∘]//MatrixForm=

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

This is an explicit operator expression of the $\sqrt{\text{SWAP}}$ gate in terms of the Pauli operators.

In[∘]:=
```
sqrtSWAP = {ExpressionFor[mat, S@{1, 2}], "Label" → " √U "}
```

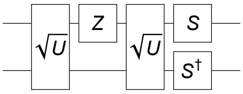
Out[∘]=

$$\left\{\left(\frac{3}{4}+\frac{i}{4}\right)+\left(\frac{1}{4}-\frac{i}{4}\right) S_1^z S_2^z+\left(\frac{1}{2}-\frac{i}{2}\right) S_1^+ S_2^-+\left(\frac{1}{2}-\frac{i}{2}\right) S_1^- S_2^+,\ \text{Label}\to\sqrt{\text{U}}\right\}$$

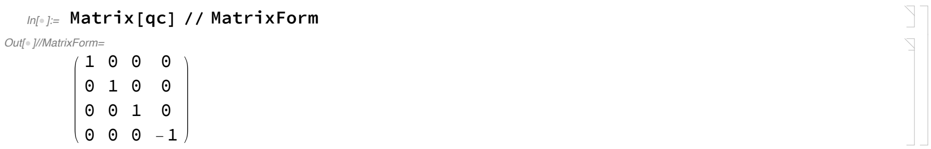
This is a quantum circuit model to construct the CZ gate from the $\sqrt{\text{SWAP}}$ gate and single-qubit rotation gates. In this diagram, $\sqrt{U}$ denotes the $\sqrt{\text{SWAP}}$ gate and S the quadrant phase gate (or, equivalently, the rotation around the z-axis by angle $\pi/2$).

In[∘]:=
```
qc = QuantumCircuit[sqrtSWAP, S[1, 3], sqrtSWAP, {S[1, 7], Dagger@S[2, 7]}]
```

Out[∘]=



To check if it indeed implements the CZ gate, take a look at the matrix representation of the quantum circuit model.



```
In[∘]:= Matrix[qc] // MatrixForm
Out[∘]//MatrixForm=
 1 0 0  0
 0 1 0  0
 0 0 1  0
 0 0 0 -1
```

2.2.2 Controlled-Unitary Gates

Consider two qubits that are once again called control and target qubits. Let $\hat{U}$ be a unitary operator on the target qubit. The controlled-$\hat{U}$ gate is a unitary operator on the two-qubit Hilbert space defined analogously to CNOT by

$$\mathrm{Ctrl}(\hat{U}) : |c\rangle \otimes |t\rangle \mapsto |c\rangle \otimes \hat{U}^c |t\rangle \ , \tag{2.51}$$

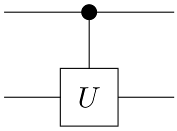
or equivalently, by

$$\mathrm{Ctrl}(\hat{U}) := |0\rangle \langle 0| \otimes \hat{I} + |1\rangle \langle 1| \otimes \hat{U} \,. \tag{2.52}$$

If the control qubit is in $|0\rangle$, then it does nothing. On the other hand, if the control qubit is in $|1\rangle$, then it operates unitary operator $\hat{U}$ on the target qubit. Analogously to the CNOT gate, the matrix representation of the controlled-unitary gate is thus given by

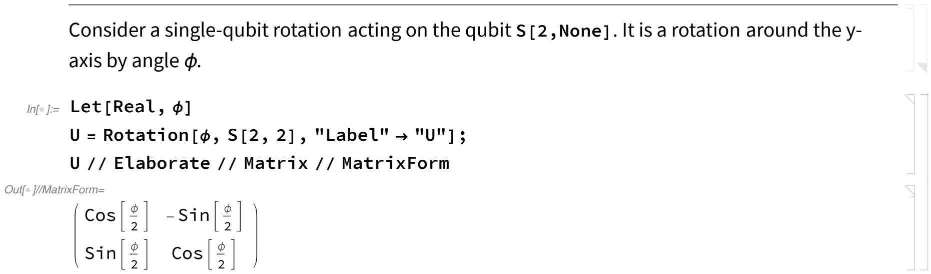
$$\mathrm{Ctrl}(\hat{U}) \doteq \begin{bmatrix} 1 & & & \\ & 1 & & \\ & & U_{11} & U_{12} \\ & & U_{21} & U_{22} \end{bmatrix} , \tag{2.53}$$

where U is the matrix representation of $\hat{U}$. In quantum circuit model, a controlled-unitary gate is depicted as



$$. \tag{2.54}$$

The filled circle connected to the quantum circuit element on the target qubit indicates the conditional operation of the element conditioned on the state of the control qubit.



Consider a single-qubit rotation acting on the qubit `S[2,None]`. It is a rotation around the y-axis by angle ϕ.

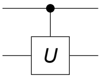
```
In[∘]:= Let[Real, ϕ]
        U = Rotation[ϕ, S[2, 2], "Label" → "U"];
        U // Elaborate // Matrix // MatrixForm
Out[∘]//MatrixForm=
```

$$\begin{pmatrix} \mathrm{Cos}\left[\frac{\phi}{2}\right] & -\mathrm{Sin}\left[\frac{\phi}{2}\right] \\ \mathrm{Sin}\left[\frac{\phi}{2}\right] & \mathrm{Cos}\left[\frac{\phi}{2}\right] \end{pmatrix}$$

This is a quantum circuit featuring the controlled-unitary gate.

In[◦]:=
```
qc = QuantumCircuit[ControlledU[S[1], U]]
```

Out[◦]=



This is the explicit expression of the controlled-U gate operation in terms of the Pauli operators.

In[◦]:=
```
op = Elaborate[qc]
```

Out[◦]=
$$\text{Cos}\left[\frac{\phi}{4}\right]^2 + S_1^z\,\text{Sin}\left[\frac{\phi}{4}\right]^2 + \frac{1}{2}\,i\,S_1^z S_2^y\,\text{Sin}\left[\frac{\phi}{2}\right] - \frac{1}{2}\,i\,S_2^y\,\text{Sin}\left[\frac{\phi}{2}\right]$$

The controlled-U gate maps the logical basis states as following.

In[◦]:=
```
bs = Basis[S@{1, 2}];
bs // LogicalForm
out = op ** bs;
out // LogicalForm
```

Out[◦]=
$$\left\{\left|0_{S_1}0_{S_2}\right\rangle, \left|0_{S_1}1_{S_2}\right\rangle, \left|1_{S_1}0_{S_2}\right\rangle, \left|1_{S_1}1_{S_2}\right\rangle\right\}$$

Out[◦]=
$$\left\{\left|0_{S_1}0_{S_2}\right\rangle, \left|0_{S_1}1_{S_2}\right\rangle, \text{Cos}\left[\frac{\phi}{2}\right]\left|1_{S_1}0_{S_2}\right\rangle + \left|1_{S_1}1_{S_2}\right\rangle\text{Sin}\left[\frac{\phi}{2}\right], \text{Cos}\left[\frac{\phi}{2}\right]\left|1_{S_1}1_{S_2}\right\rangle - \left|1_{S_1}0_{S_2}\right\rangle\text{Sin}\left[\frac{\phi}{2}\right]\right\}$$

To make the mapping clearer, this tabulates the above result.

In[◦]:=
```
new = KetFactor[#, S[1]] & /@ out;
Thread[bs → new] // LogicalForm // TableForm
```

Out[◦]//TableForm=
$$\left|0_{S_1}0_{S_2}\right\rangle \to \left|0_{S_1}\right\rangle \otimes \left|0_{S_2}\right\rangle$$
$$\left|0_{S_1}1_{S_2}\right\rangle \to \left|0_{S_1}\right\rangle \otimes \left|1_{S_2}\right\rangle$$
$$\left|1_{S_1}0_{S_2}\right\rangle \to \left|1_{S_1}\right\rangle \otimes \left(\text{Cos}\left[\frac{\phi}{2}\right]\left|0_{S_2}\right\rangle + \left|1_{S_2}\right\rangle\text{Sin}\left[\frac{\phi}{2}\right]\right)$$
$$\left|1_{S_1}1_{S_2}\right\rangle \to \left|1_{S_1}\right\rangle \otimes \left(\text{Cos}\left[\frac{\phi}{2}\right]\left|1_{S_2}\right\rangle - \left|0_{S_2}\right\rangle\text{Sin}\left[\frac{\phi}{2}\right]\right)$$

Let us take a look at the mapping more closely. When the first qubit is set to `Ket[0]`, it does nothing.

In[◦]:=
```
Let[Complex, c]
vec = Ket[] × c[0] + Ket[S[2] → 1] × c[2];
LogicalForm[vec, S@{1, 2}]
```

Out[◦]=
$$c_0\left|0_{S_1}0_{S_2}\right\rangle + c_2\left|0_{S_1}1_{S_2}\right\rangle$$

In[◦]:=
```
new = op ** vec;
LogicalForm[new, S@{1, 2}]
```

Out[◦]=
$$c_0\left|0_{S_1}0_{S_2}\right\rangle + c_2\left|0_{S_1}1_{S_2}\right\rangle$$

When the control qubit -- the first qubit in this case -- is set to $\left|1\right\rangle$, it operates the unitary operator on the second qubit.

In[◦]:=
```
vec = Ket[S[1] → 1] ** (Ket[] × c[0] + Ket[S[2] → 1] × c[2]);
LogicalForm[vec, S@{1, 2}]
```

Out[◦]=
$$c_0\left|1_{S_1}0_{S_2}\right\rangle + c_2\left|1_{S_1}1_{S_2}\right\rangle$$

```
In[•]:= new = op ** vec;
        LogicalForm[new, S@{1, 2}]
```

Out[•]= $\left|1_{S_1}1_{S_2}\right\rangle \left(c_2\,\mathrm{Cos}\left[\frac{\phi}{2}\right] + c_0\,\mathrm{Sin}\left[\frac{\phi}{2}\right]\right) + \left|1_{S_1}0_{S_2}\right\rangle \left(c_0\,\mathrm{Cos}\left[\frac{\phi}{2}\right] - c_2\,\mathrm{Sin}\left[\frac{\phi}{2}\right]\right)$

When the control qubit is in a superposition, the resulting state is an entangled state in general.

```
In[•]:= vec = Ket[] + Ket[S[1] → 1];
        LogicalForm[vec, S@{1, 2}]
```

Out[•]= $\left|0_{S_1}0_{S_2}\right\rangle + \left|1_{S_1}0_{S_2}\right\rangle$

```
In[•]:= new = op ** vec;
        LogicalForm[new, S@{1, 2}]
```

Out[•]= $\left|0_{S_1}0_{S_2}\right\rangle + \mathrm{Cos}\left[\frac{\phi}{2}\right] \left|1_{S_1}0_{S_2}\right\rangle + \left|1_{S_1}1_{S_2}\right\rangle \mathrm{Sin}\left[\frac{\phi}{2}\right]$

An important aspect of a controlled-unitary operator is that it induces relative phase shifts on the *control* qubit when the target qubit has been prepared in an eigenstate of $\hat{U}$. At first glace, it may sound counterintuitive since the definition in (2.51) seems to indicate that it changes only the target qubit depending on the state of the control qubit and keeps the latter intact. This is another feature distinguishing quantum gates from their classical counterparts. Let us take a closer look to see how this works. Prepare the control qubit in a superposition $|\psi\rangle = |0\rangle + |1\rangle$ and the target qubit in an eigenstate $|\phi\rangle$ of $\hat{U}$ with eigenvalue $e^{i\phi}$. The controlled-unitary gate transforms the state to

$$\begin{aligned} |\psi\rangle \otimes |\phi\rangle &\to |0\rangle \otimes |\phi\rangle + |1\rangle \otimes \hat{U}\,|\phi\rangle \\ &= |0\rangle \otimes |\phi\rangle + |1\rangle \otimes |\phi\rangle\, e^{i\phi} = \left(|0\rangle + |1\rangle\, e^{i\phi}\right) \otimes |\phi\rangle\,. \end{aligned} \tag{2.55}$$

This feature is extended to multi-control unitary gates and plays a crucial role in many quantum algorithms. In particular, the quantum phase estimation algorithm (Section 4.4) is a direct consequence of this feature.

When the target qubit is set to an eigenstate of the unitary operator, it does not change but the control qubit acquires the phase factor given by the eigenvalue of the target state.

```
In[•]:= vec = (Ket[] + Ket[S[1] → 1]) ** (Ket[] - I Ket[S[2] → 1]);
        LogicalForm[KetFactor@vec, S@{1, 2}]
```

Out[•]= $\left(\left|0_{S_1}\right\rangle + \left|1_{S_1}\right\rangle\right) \otimes \left(\left|0_{S_2}\right\rangle - i\left|1_{S_2}\right\rangle\right)$

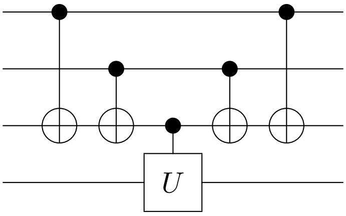
```
In[•]:= new = op ** vec // TrigToExp;
        LogicalForm[KetFactor@new, S@{1, 2}]
```

Out[•]= $\left(\left|0_{S_1}\right\rangle + e^{\frac{i\phi}{2}}\left|1_{S_1}\right\rangle\right) \otimes \left(\left|0_{S_2}\right\rangle - i\left|1_{S_2}\right\rangle\right)$

The CNOT gate and the controlled-unitary gate can be combined to achieve a variety of conditional gate operations. For example, consider a system consisting of a control register of n qubits and a target register of a single qubit. Suppose that you want to operate a unitary gate $\hat{U}$ on the target qubit only when an odd number of the control qubits is set to $|1\rangle$,

$$|c\rangle \otimes |t\rangle \mapsto |c\rangle \otimes \hat{U}^{c_1 \oplus \cdots \oplus c_n} |t\rangle \ . \tag{2.56}$$

First, operate the CNOT gates consecutively with the first $(n-1)$ qubits in the control register as the control qubit and the last qubit in the control register as the target qubit. This transforms the nth qubit to $|c_1 \oplus \cdots \oplus c_n\rangle$ (Problem 2.9). Then, the desired operation is implemented by applying the controlled-$\hat{U}$ gate controlled by the nth qubit. To get the control qubits back to the original state, operate the CNOT gates in the reverse order. Overall, the following quantum circuit implements the conditional operation



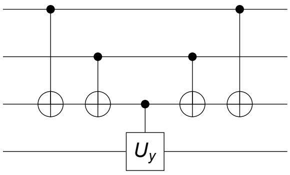
(2.57)

for the case of $n = 3$. This method is used to implement a multi-control unitary gate based on the Gray code (see Section 2.3).

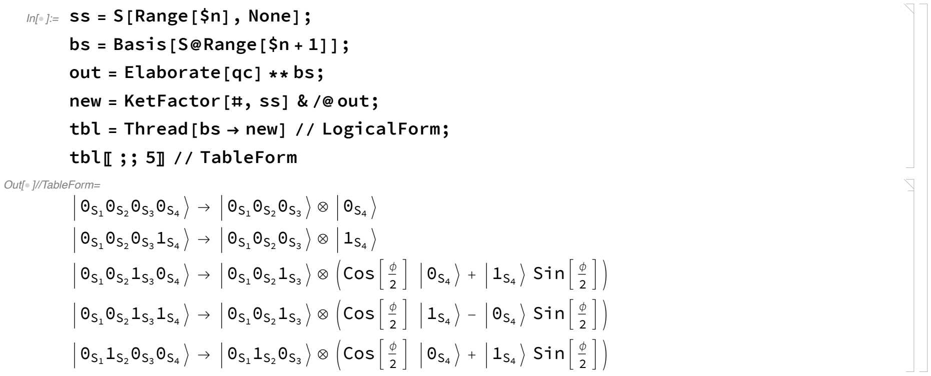
This shows a quantum circuit model conditionally operating the logic gate U on the target qubit.

In[•]:=
```
$n = 3;
cc = Table[CNOT[S[j], S[$n]], {j, 1, $n - 1}];
op = Rotation[ϕ, S[$n + 1, 2]];
cU = ControlledU[S[$n], op];
qc = QuantumCircuit[Sequence @@ Flatten@{cc, cU, Reverse@cc}]
```

Out[•]=



This shows how the above quantum circuit model maps the logical basis states. It affects the target qubit only when $c_1 \oplus c_2 \oplus c_3 = 1$.



```
In[ ]:= ss = S[Range[$n], None];
        bs = Basis[S@Range[$n + 1]];
        out = Elaborate[qc] ** bs;
        new = KetFactor[#, ss] & /@ out;
        tbl = Thread[bs → new] // LogicalForm;
        tbl⟦ ;; 5⟧ // TableForm
```

```
Out[ ]//TableForm=
        |0_{S_1}0_{S_2}0_{S_3}0_{S_4}⟩ → |0_{S_1}0_{S_2}0_{S_3}⟩ ⊗ |0_{S_4}⟩
        |0_{S_1}0_{S_2}0_{S_3}1_{S_4}⟩ → |0_{S_1}0_{S_2}0_{S_3}⟩ ⊗ |1_{S_4}⟩
        |0_{S_1}0_{S_2}1_{S_3}0_{S_4}⟩ → |0_{S_1}0_{S_2}1_{S_3}⟩ ⊗ (Cos[ϕ/2] |0_{S_4}⟩ + |1_{S_4}⟩ Sin[ϕ/2])
        |0_{S_1}0_{S_2}1_{S_3}1_{S_4}⟩ → |0_{S_1}0_{S_2}1_{S_3}⟩ ⊗ (Cos[ϕ/2] |1_{S_4}⟩ - |0_{S_4}⟩ Sin[ϕ/2])
        |0_{S_1}1_{S_2}0_{S_3}0_{S_4}⟩ → |0_{S_1}1_{S_2}0_{S_3}⟩ ⊗ (Cos[ϕ/2] |0_{S_4}⟩ + |1_{S_4}⟩ Sin[ϕ/2])
```

How can you implement a controlled-unitary gate? The operation involves only two qubits, and in principle, it should be possible to implement any specific controlled-unitary gate. However, it will become clear in Chapter 3, the requirements for a physical implementation of two-qubit gates is far more difficult to fulfill on realistic systems than for single-qubit gates. Fortunately, any controlled-unitary gate can be implemented using only a CNOT gate and single-qubit gates. This is one of the basic steps in establishing universal quantum computation.

Let $\hat{U}$ be a unitary gate on the second (target) qubit controlled by the first (control) qubit. Suppose that $\hat{U}$ has Euler angles α, β, and γ and an additional phase factor $e^{i\varphi}$ [see (2.27)], $\hat{U} = e^{i\varphi}\hat{U}_z(\alpha)\hat{U}_y(\beta)\hat{U}_z(\gamma)$. Then one can always find three unitary operators $\hat{A}$, $\hat{B}$, and $\hat{C}$ such that

$$\hat{U} = e^{i\varphi}\hat{A}\hat{X}\hat{B}\hat{X}\hat{C}\,, \quad \hat{A}\hat{B}\hat{C} = \hat{I}\,, \tag{2.58}$$

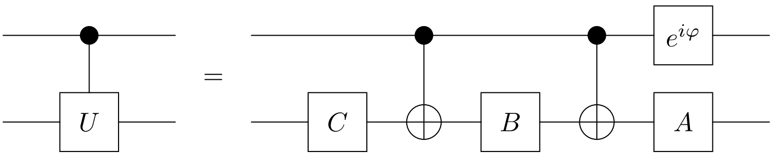
where $\hat{X}$ is the Pauli X operator. More explicitly, one common choice is

$$\hat{A} = \hat{U}_z(\alpha)\hat{U}_y(\beta/2)\,, \tag{2.59a}$$

$$\hat{B} = \hat{U}_y(-\beta/2)\hat{U}_z(-(\alpha+\gamma)/2)\,, \tag{2.59b}$$

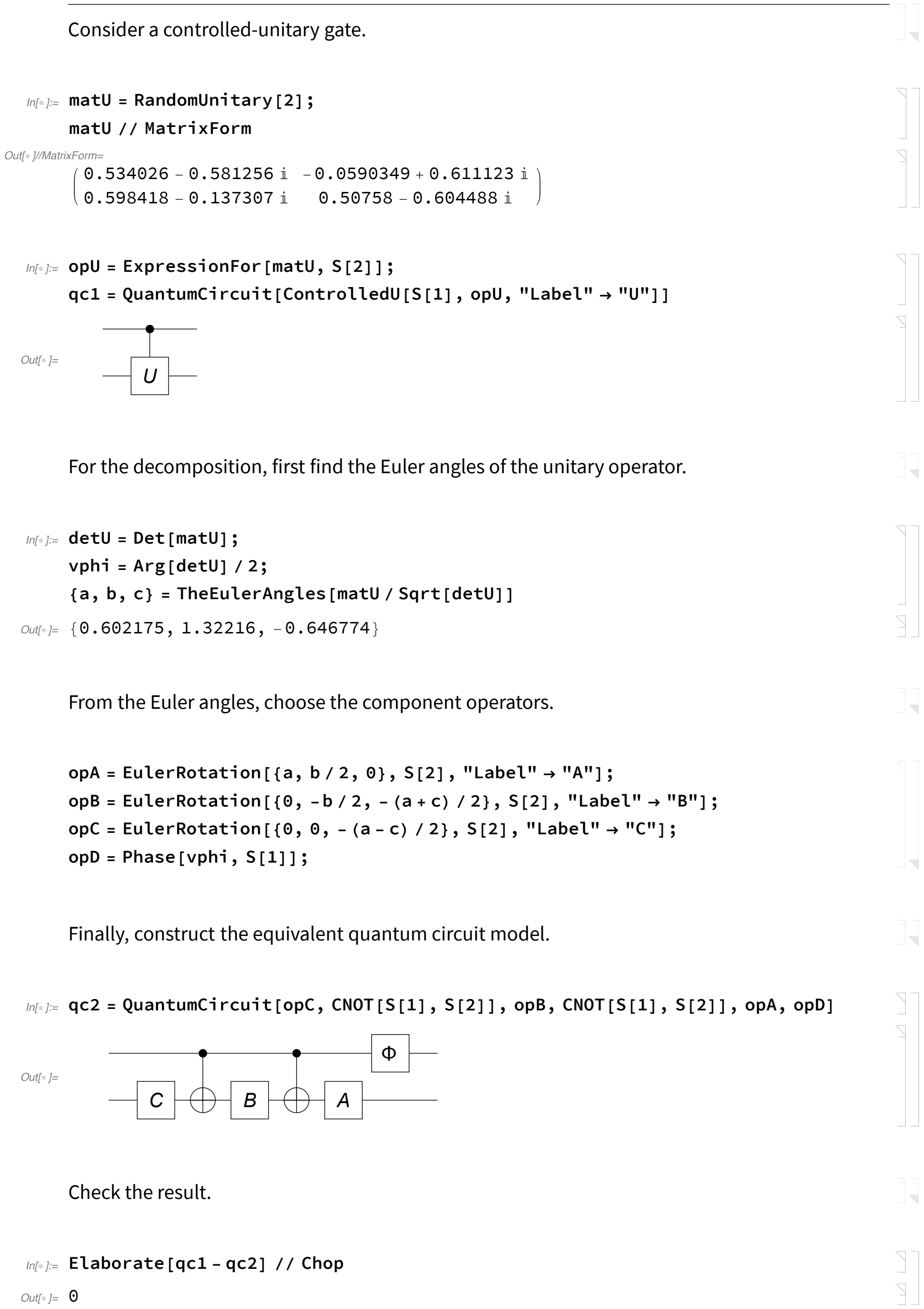
$$\hat{C} = \hat{U}_z(-(\alpha-\gamma)/2). \tag{2.59c}$$

Since $\hat{U}_{y/z}(\phi) = \hat{X}\hat{U}_{y/z}(-\phi)\hat{X}$ for any ϕ, the above choice satisfies the desired properties in (2.58). These properties imply that the controlled-unitary gate can be implemented as in the following quantum circuit



, (2.60)

where the last gate on the first qubit is the *relative* phase shift by φ, $|0\rangle\langle 0| + |1\rangle\langle 1| e^{i\varphi}$. Indeed, when the control qubit is in $|0\rangle$, the two CNOT gates in the middle do nothing, and the combined operator $\hat{A}\hat{B}\hat{C}$ on the target qubit ends up with the identity operator. With the control qubit in $|1\rangle$, on the other hand, the two CNOT gates are operational and the overall operator on the target qubit becomes $\hat{A}\hat{X}\hat{B}\hat{X}\hat{C} = e^{-i\varphi}\hat{U}$, where the phase factor is canceled by the opposite phase from the control qubit.



Consider a controlled-unitary gate.

```
In[]:= matU = RandomUnitary[2];
        matU // MatrixForm
Out[]//MatrixForm=
        ( 0.534026 - 0.581256 i  -0.0590349 + 0.611123 i )
        ( 0.598418 - 0.137307 i    0.50758 - 0.604488 i   )
```

```
In[]:= opU = ExpressionFor[matU, S[2]];
        qc1 = QuantumCircuit[ControlledU[S[1], opU, "Label" → "U"]]
Out[]=
```

For the decomposition, first find the Euler angles of the unitary operator.

```
In[]:= detU = Det[matU];
        vphi = Arg[detU] / 2;
        {a, b, c} = TheEulerAngles[matU / Sqrt[detU]]
Out[]= {0.602175, 1.32216, -0.646774}
```

From the Euler angles, choose the component operators.

```
opA = EulerRotation[{a, b / 2, 0}, S[2], "Label" → "A"];
opB = EulerRotation[{0, -b / 2, - (a + c) / 2}, S[2], "Label" → "B"];
opC = EulerRotation[{0, 0, - (a - c) / 2}, S[2], "Label" → "C"];
opD = Phase[vphi, S[1]];
```

Finally, construct the equivalent quantum circuit model.

```
In[]:= qc2 = QuantumCircuit[opC, CNOT[S[1], S[2]], opB, CNOT[S[1], S[2]], opA, opD]
Out[]=
```

Check the result.

```
In[]:= Elaborate[qc1 - qc2] // Chop
Out[]= 0
```

2.2.3 General Unitary Gates

Here we decompose an arbitrary two-qubit unitary gate into factors of controlled-unitary gates only. This is a remarkable advantage when one tries to build a quantum computer as you can just focus on how to implement the CNOT gate and single-qubit gates. Moreover, the same idea eventually leads to the proof of universal quantum computation on a larger system, which will be discussed in Section 2.4.

Consider an arbitrary two-qubit unitary operator $\hat{U}$. In the logical basis, it is represented by a unitary matrix

$$U = \begin{bmatrix} U_{11} & U_{12} & U_{13} & U_{14} \\ U_{21} & U_{22} & U_{23} & U_{24} \\ U_{31} & U_{32} & U_{33} & U_{34} \\ U_{41} & U_{42} & U_{43} & U_{44} \end{bmatrix} . \tag{2.61}$$

To break the unitary operation $\hat{U}$ down into more elementary quantum logic gates, we make use of *two-level unitary transformations*. A two-level unitary transformation is a unitary operation with matrix representation that acts only on the two columns and rows of other matrices or column or row vectors. The descriptive word "two-level" should not be confused with "two-qubit". A two-level transformation acts on multiple qubits, but it just transforms only two rows or columns at a time in the representation. For example, consider a two-level unitary transformation of the form

$$T_1 = \begin{bmatrix} 1 & & & \\ & 1 & & \\ & & \tilde{U}_{13}^* & \tilde{U}_{14} \\ & & \tilde{U}_{14}^* & -\tilde{U}_{13} \end{bmatrix} , \tag{2.62}$$

where $\tilde{U}_{ij} \propto U_{ij}$ with normalization factor (unspecified) such that $T_1^\dagger T_1 = T_1 T_1^\dagger = I$. When it multiplies U from the right, it does not change the first two columns of U. It only alters the last two columns, and hence the name "two-level transformation". The elements in the lower-right sub-block of T_1 have been chosen so that the first element of the last column is canceled:

$$UT_1 = \begin{bmatrix} U_{11} & U_{12} & U'_{13} & 0 \\ U_{21} & U_{22} & U'_{23} & U'_{24} \\ U_{31} & U_{32} & U'_{33} & U'_{34} \\ U_{41} & U_{42} & U'_{43} & U'_{44} \end{bmatrix} . \tag{2.63}$$

Now take another two-level unitary transformation, this time, of the form

$$T_2 = \begin{bmatrix} 1 & & & \\ & \tilde{U}_{12}^* & \tilde{U}'_{13} & \\ & \tilde{U}'^*_{13} & -\tilde{U}_{12} & \\ & & & 1 \end{bmatrix} . \tag{2.64}$$

It removes U'_{13} while keeping the first and last column as they are:

$$UT_1T_2 = \begin{bmatrix} U_{11} & U''_{12} & 0 & 0 \\ U_{21} & U''_{22} & U''_{23} & U'_{24} \\ U_{31} & U''_{32} & U''_{33} & U'_{34} \\ U_{41} & U''_{42} & U''_{43} & U'_{44} \end{bmatrix}. \tag{2.65}$$

Similarly, we go further with the two-level unitary matrix

$$T_2 = \begin{bmatrix} \tilde{U}^*_{12} & \tilde{U}'_{13} & & \\ \tilde{U}'^*_{13} & -\tilde{U}_{12} & & \\ & & 1 & \\ & & & 1 \end{bmatrix}. \tag{2.66}$$

to remove the element U''_{12} and get

$$UT_1T_2T_3 = \begin{bmatrix} U'''_{11} & 0 & 0 & 0 \\ 0 & U'''_{22} & U''_{23} & U'_{24} \\ 0 & U'''_{32} & U''_{33} & U'_{34} \\ 0 & U'''_{42} & U''_{43} & U'_{44} \end{bmatrix}. \tag{2.67}$$

At this stage, all elements except for the first of the first column vanish —$U'''_{21} = U'''_{31} = U'''_{41} = 0$—because the product $UT_1T_2T_3$ is a unitary matrix. Repeating the procedure, we can remove all off-diagonal elements to get

$$UT_1T_2 \dots T_L = I. \tag{2.68}$$

As T_j are all unitary, it enables us to rewrite U in a combination of two-level unitary transformations, that is,

$$U = T_L^\dagger \dots T_2^\dagger T_1^\dagger. \tag{2.69}$$

Consider a Hermitian operator on a two-qubit system. Physically, it corresponds to a transverse-field Ising model.

```
In[∘]:= H = S[1, 1] ** S[2, 1] + S[1, 2] + S[2, 2] + S[1, 3] + S[2, 3]
        matH = Matrix[H];
        matH // MatrixForm
```

Out[∘]= $S_1^x S_2^x + S_1^y + S_1^z + S_2^y + S_2^z$

Out[∘]//MatrixForm=

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

This is the unitary operator generated by the above Hermitian operator.

```
In[∘]:= U = MultiplyExp[-I H Pi / 4]
```

Out[∘]= $e^{-\frac{1}{4} i \pi \left(S_1^x S_2^x + S_1^y + S_1^z + S_2^y + S_2^z\right)}$

This shows the matrix representation of `U` in the logical basis.

```
In[∘]:= matU = Matrix[U] // Simplify;
        matU // MatrixForm
```

Out[∘]//MatrixForm=

$$\begin{pmatrix} -\frac{\frac{1}{2}+\frac{5 i}{6}}{\sqrt{2}} & -\frac{\frac{1}{6}-\frac{i}{2}}{\sqrt{2}} & -\frac{\frac{1}{6}-\frac{i}{2}}{\sqrt{2}} & \frac{\frac{1}{2}-\frac{i}{2}}{\sqrt{2}} \\ \frac{\frac{1}{6}-\frac{i}{2}}{\sqrt{2}} & \frac{\frac{1}{2}+\frac{i}{6}}{\sqrt{2}} & -\frac{\frac{1}{2}+\frac{5 i}{6}}{\sqrt{2}} & -\frac{\frac{1}{2}+\frac{i}{2}}{\sqrt{2}} \\ \frac{\frac{1}{6}-\frac{i}{2}}{\sqrt{2}} & -\frac{\frac{1}{2}+\frac{5 i}{6}}{\sqrt{2}} & \frac{\frac{1}{2}+\frac{i}{6}}{\sqrt{2}} & -\frac{\frac{1}{2}+\frac{i}{2}}{\sqrt{2}} \\ \frac{\frac{1}{2}-\frac{i}{2}}{\sqrt{2}} & \frac{\frac{1}{2}+\frac{i}{2}}{\sqrt{2}} & \frac{\frac{1}{2}+\frac{i}{2}}{\sqrt{2}} & -\frac{\frac{1}{2}-\frac{i}{2}}{\sqrt{2}} \end{pmatrix}$$

This shows the decomposition of the unitary matrix into two-level matrices (displaying the first three elements). For the purpose of efficiency, the resulting list is given in terms of `TwoLevelU`.

```
In[∘]:= twl = TwoLevelDecomposition[matU] // Simplify;
        twl⟦ ;; 3⟧
```

Out[∘]=

$$\Big\{\text{TwoLevelU}[\{\{1,0\},\{0,1\}\},\{3,4\},4],$$
$$\text{TwoLevelU}\Big[\Big\{\Big\{\frac{\frac{1}{2}-\frac{3 i}{2}}{\sqrt{7}},-\frac{\frac{3}{2}+\frac{3 i}{2}}{\sqrt{7}}\Big\},\Big\{\frac{\frac{3}{2}-\frac{3 i}{2}}{\sqrt{7}},\frac{\frac{1}{2}+\frac{3 i}{2}}{\sqrt{7}}\Big\}\Big\},\{3,4\},4\Big],$$
$$\text{TwoLevelU}\Big[\Big\{\Big\{-\frac{1-3 i}{\sqrt{38}},\sqrt{\frac{14}{19}}\Big\},\Big\{-\sqrt{\frac{14}{19}},-\frac{1+3 i}{\sqrt{38}}\Big\}\Big\},\{2,3\},4\Big]\Big\}$$

For a more intuitive form, you can convert `TwoLevelU` into the normal matrix form using `Matrix`. Here the first three are shown.

```
In[∘]:= MatrixForm /@ Matrix /@ twl⟦ ;; 3⟧
```

Out[∘]=

$$\left\{\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}, \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \frac{\frac{1}{2}-\frac{3 i}{2}}{\sqrt{7}} & -\frac{\frac{3}{2}+\frac{3 i}{2}}{\sqrt{7}} \\ 0 & 0 & \frac{\frac{3}{2}-\frac{3 i}{2}}{\sqrt{7}} & \frac{\frac{1}{2}+\frac{3 i}{2}}{\sqrt{7}} \end{pmatrix}, \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & -\frac{1-3 i}{\sqrt{38}} & \sqrt{\frac{14}{19}} & 0 \\ 0 & -\sqrt{\frac{14}{19}} & -\frac{1+3 i}{\sqrt{38}} & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}\right\}$$

Indeed, they reconstruct the original matrix.

```
In[∘]:= new = Dot @@ Matrix /@ twl;
        matU - new // Simplify // MatrixForm
```

Out[∘]//MatrixForm=

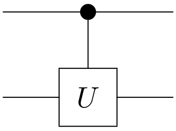
$$\begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

We are still to express the two-level unitary transformation in terms of a controlled-unitary gate. The two-level unitary matrix—for example, T_1 in (2.62)—of the form

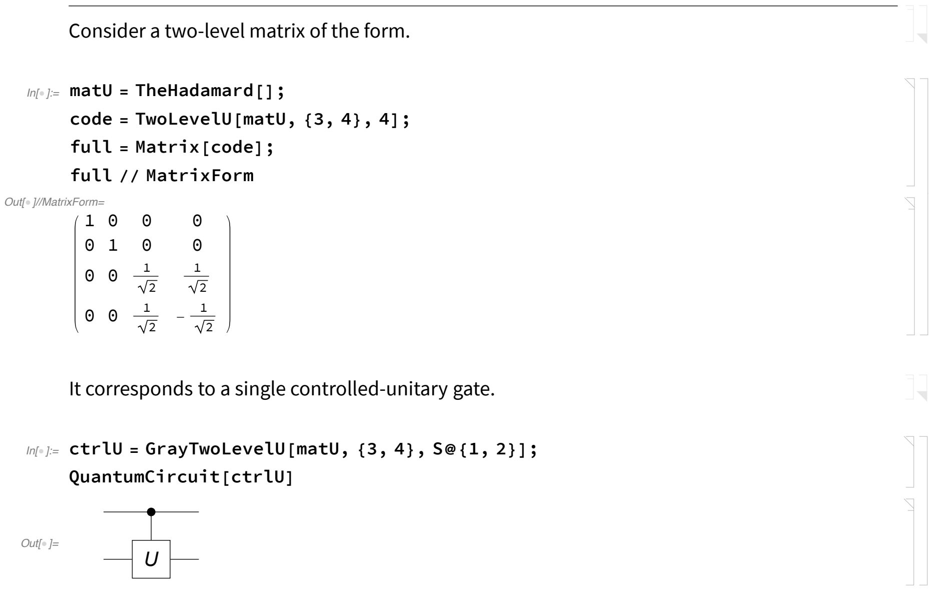
$$\begin{bmatrix} 1 & & & \\ & 1 & & \\ & & U_{11} & U_{12} \\ & & U_{21} & U_{22} \end{bmatrix}, \tag{2.70}$$

is already in the form of the matrix representation of a controlled-unitary gate in (2.53), and just corresponds to a single controlled-unitary:

$$\begin{bmatrix} 1 & & & \\ & 1 & & \\ & & U_{11} & U_{12} \\ & & U_{21} & U_{22} \end{bmatrix} = \tag{2.71}$$



.



Consider a two-level matrix of the form.

```
In[ ]:= matU = TheHadamard[];
        code = TwoLevelU[matU, {3, 4}, 4];
        full = Matrix[code];
        full // MatrixForm
```

Out[ ]//MatrixForm=

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ 0 & 0 & \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{pmatrix}$$

It corresponds to a single controlled-unitary gate.

```
In[ ]:= ctrlU = GrayTwoLevelU[matU, {3, 4}, S@{1, 2}];
        QuantumCircuit[ctrlU]
```

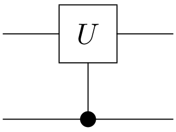
Out[ ]=

A two-level unitary matrix of the form

$$\begin{bmatrix} 1 & & & \\ & U_{11} & & U_{12} \\ & & 1 & \\ & U_{21} & & U_{22} \end{bmatrix} \tag{2.72}$$

also corresponds to a single controlled-unitary gate with the control and target qubit exchanged. Here, the first qubit is the target qubit and the second the control qubit:

$$\begin{bmatrix} 1 & & & \\ & U_{11} & & U_{12} \\ & & 1 & \\ & U_{21} & & U_{22} \end{bmatrix} = \tag{2.73}$$



.

Consider a two-level matrix of the form.

```
In[◦]:= matU = TheHadamard[];
        code = TwoLevelU[matU, {2, 4}, 4];
        full = Matrix[code];
        full // MatrixForm
```

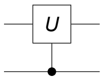
Out[◦]//MatrixForm=

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \frac{1}{\sqrt{2}} & 0 & \frac{1}{\sqrt{2}} \\ 0 & 0 & 1 & 0 \\ 0 & \frac{1}{\sqrt{2}} & 0 & -\frac{1}{\sqrt{2}} \end{pmatrix}$$

It corresponds to a single controlled-unitary gate with the control and target qubit exchanged.

```
In[◦]:= ctrlU = GrayTwoLevelU[matU, {2, 4}, S@{1, 2}];
        QuantumCircuit[ctrlU]
```

Out[◦]=

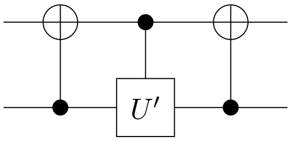


More complicated is the two-level unitary matrix of the form

$$\begin{bmatrix} 1 & & & \\ & U_{11} & U_{12} & \\ & U_{21} & U_{22} & \\ & & & 1 \end{bmatrix}. \tag{2.74}$$

As it affects both qubits simultaneously, it cannot be represented by a single controlled-unitary gate. However, it is possible to bring it to the form of (2.71) by exchanging the second and fourth columns and rows. The specified exchanges correspond to flipping the bit values of the first qubit only when the second qubit has a value of 1, that is, the CNOT gate with the second qubit as the control qubit and the first qubit as the target qubit. Through this exchange, the unitary matrix U itself is modified and the rows and columns are exchanged, which corresponds to the basis change by the Pauli X matrix, $U \to U' = XUX$. Putting all together, the two-level unitary matrix is implemented as

$$\begin{bmatrix} 1 & & & \\ & U_{11} & U_{12} & \\ & U_{21} & U_{22} & \\ & & & 1 \end{bmatrix} =$$



. (2.75)

Consider a two-level matrix of the form.

```
In[◦]:= matU = TheHadamard[];
        code = TwoLevelU[matU, {2, 3}, 4];
        full = Matrix[code];
        full // MatrixForm
```

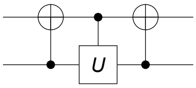
Out[◦]//MatrixForm=

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 \\ 0 & \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

In this case, we need to apply the CNOT gate before and after the controlled-unitary gate.

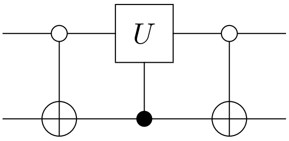
```
In[◦]:= ctrlU = GrayTwoLevelU[matU, {2, 3}, S@{1, 2}];
        QuantumCircuit @@ ctrlU
```

Out[◦]=

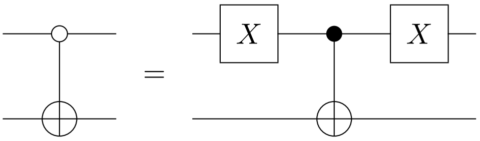


In a similar manner, we can implement another form of two-level unitary matrix as

$$\begin{bmatrix} U_{11} & & & U_{12} \\ & 1 & & \\ & & 1 & \\ U_{21} & & & U_{22} \end{bmatrix} = \tag{2.76}$$



Here we have adopted a short-hand diagram for the modified-CNOT gate, which flips the bit value of the target qubit when the control qubit is in the state $|0\rangle$ rather than $|1\rangle$. It is achieved by operating the Pauli X gate before and after the usual CNOT gate,



(2.77)

Consider a two-level matrix of the form.

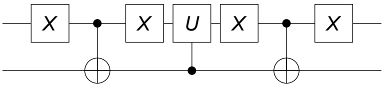
```
In[◦]:= matU = TheHadamard[];
        code = TwoLevelU[matU, {1, 4}, 4];
        full = Matrix[code];
        full // MatrixForm
```

Out[◦]//MatrixForm=

$$\begin{pmatrix} \frac{1}{\sqrt{2}} & 0 & 0 & \frac{1}{\sqrt{2}} \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ \frac{1}{\sqrt{2}} & 0 & 0 & -\frac{1}{\sqrt{2}} \end{pmatrix}$$

```
In[◦]:= ctrlU = GrayTwoLevelU[matU, {1, 4}, S@{1, 2}];
        QuantumCircuit @@ ctrlU
```

Out[◦]=



So far, we have determined the implementations of 4 × 4 two-level unitary matrices of different forms just by simple inspection. For more than two qubits, the size of the two-level unitary matrices is much larger and it is difficult to find proper implementations in such a way. Fortunately, there is a systematic way based on the Gray code, which will be discussed later in Section 2.4.

In summary, an arbitrary two-qubit unitary gate can be carried out by first decomposing its matrix representation into two-level unitary matrices and then implementing the two-level unitary matrices by means of the CNOT gate and the controlled-unitary gate.

Let us consider again the two-qubit model demonstrated before.

```
In[∘]:= H = S[1, 1] ** S[2, 1] + S[1, 2] + S[2, 2] + S[1, 3] + S[2, 3]
        U = MultiplyExp[-I H Pi / 4]
        matU = Matrix[U];
```

Out[∘]=
$$S_1^x S_2^x + S_1^y + S_1^z + S_2^y + S_2^z$$

Out[∘]=
$$e^{-\frac{1}{4} i \pi \left(S_1^x S_2^x+S_1^y+S_1^z+S_2^y+S_2^z\right)}$$

This is the decomposition into the two-level matrices, just showing the first four.

```
In[∘]:= twl = TwoLevelDecomposition[matU] // Simplify;
        MatrixForm /@ Matrix /@ twl[[ ;; 3]]
```

Out[∘]=
$$\left\{\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}, \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \frac{\frac{1}{2}-\frac{3 i}{2}}{\sqrt{7}} & -\frac{\frac{3}{2}+\frac{3 i}{2}}{\sqrt{7}} \\ 0 & 0 & \frac{\frac{3}{2}-\frac{3 i}{2}}{\sqrt{7}} & \frac{\frac{1}{2}+\frac{3 i}{2}}{\sqrt{7}} \end{pmatrix}, \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & -\frac{1-3 i}{\sqrt{38}} & \sqrt{\frac{14}{19}} & 0 \\ 0 & -\sqrt{\frac{14}{19}} & -\frac{1+3 i}{\sqrt{38}} & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}\right\}$$

The two-level matrices are written in terms of the controlled-unitary and CNOT gate, again showing the first four. The first of the list `twl` happens to be the identity matrix in this case, and it is excluded in the further analysis.

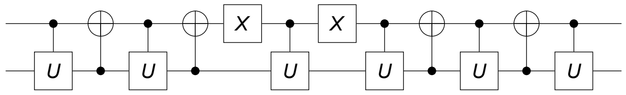
```
In[∘]:= gates = GrayTwoLevelU[#, S@{1, 2}] & /@ Rest[twl];
        gates[[ ;; 3]]
```

Out[∘]=
$$\left\{\left\{\text{ControlledU}\left[\{S_1\}, \frac{1}{2\sqrt{7}} - \frac{3 i S_2^z}{2\sqrt{7}} - \frac{\left(\frac{3}{2}+\frac{3 i}{2}\right) S_2^+}{\sqrt{7}} + \frac{\left(\frac{3}{2}-\frac{3 i}{2}\right) S_2^-}{\sqrt{7}}, \text{Label} \to U\right]\right\},\right.$$
$$\left\{\text{CNOT}[\{S_2\}, \{S_1\}],\right.$$
$$\left.\text{ControlledU}\left[\{S_1\}, -\frac{1}{\sqrt{38}} - i\sqrt{\frac{14}{19}}\, S_2^y - \frac{3 i S_2^z}{\sqrt{38}}, \text{Label} \to U\right], \text{CNOT}[\{S_2\}, \{S_1\}]\right\},$$
$$\left.\left\{\text{ControlledU}\left[\{S_1\}, -\frac{5}{14\sqrt{2}} - \frac{5 i S_2^z}{14\sqrt{2}} + \frac{3}{14}\sqrt{19}\, S_2^+ - \frac{3}{14}\sqrt{19}\, S_2^-, \text{Label} \to U\right]\right\}\right\}$$

This shows the overall quantum circuit model. Here the same label "U" has been used for different controlled-unitary gates just for convenience. Do not forget to reverse the order before plugging the gates in the quantum circuit model.

```
In[∘]:= qc = QuantumCircuit @@ Reverse@Flatten[gates]
```

Out[∘]=



Check the above quantum circuit by converting it into the matrix representation.

```
In[∘]:= new = Matrix[qc] // Normal // Simplify;
        new // MatrixForm
```

Out[∘]//MatrixForm=

$$\begin{pmatrix} -\frac{\frac{1}{2}+\frac{5 i}{6}}{\sqrt{2}} & -\frac{\frac{1}{6}-\frac{i}{2}}{\sqrt{2}} & -\frac{\frac{1}{6}-\frac{i}{2}}{\sqrt{2}} & \frac{\frac{1}{2}-\frac{i}{2}}{\sqrt{2}} \\ \frac{\frac{1}{6}-\frac{i}{2}}{\sqrt{2}} & \frac{\frac{1}{2}+\frac{i}{6}}{\sqrt{2}} & -\frac{\frac{1}{2}+\frac{5 i}{6}}{\sqrt{2}} & -\frac{\frac{1}{2}+\frac{i}{2}}{\sqrt{2}} \\ \frac{\frac{1}{6}-\frac{i}{2}}{\sqrt{2}} & -\frac{\frac{1}{2}+\frac{5 i}{6}}{\sqrt{2}} & \frac{\frac{1}{2}+\frac{i}{6}}{\sqrt{2}} & -\frac{\frac{1}{2}+\frac{i}{2}}{\sqrt{2}} \\ \frac{\frac{1}{2}-\frac{i}{2}}{\sqrt{2}} & \frac{\frac{1}{2}+\frac{i}{2}}{\sqrt{2}} & \frac{\frac{1}{2}+\frac{i}{2}}{\sqrt{2}} & -\frac{\frac{1}{2}-\frac{i}{2}}{\sqrt{2}} \end{pmatrix}$$

```
In[∘]:= new - matU // Simplify // MatrixForm
```

Out[∘]//MatrixForm=

$$\begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

2.3 Multi-Control Unitary Gates

Let $\mathcal{S}^{\otimes m}$ and $\mathcal{S}^{\otimes n}$ be the Hilbert spaces of the control and target register consisting of m and n qubits, respectively. Suppose that $\hat{U}$ is a unitary operator on the target register. The multi-control unitary operator is defined by

$$\mathrm{Ctrl}(\hat{U}) : |c\rangle \otimes |t\rangle \mapsto |c\rangle \otimes \hat{U}^{c_1 c_2 \cdots c_m} |t\rangle \ , \tag{2.78}$$

where $c \equiv (c_1 c_2 \ldots c_m)_2$. The unitary transformation $\hat{U}$ acts on the target qubits only when every control qubit is set to $|1\rangle$. We will be most interested in the case with $n = 1$, where the matrix representation takes the form

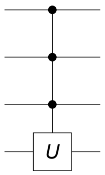
$$\mathrm{Ctrl}(\hat{U}) \doteq \begin{bmatrix} 1 & & & & \\ & 1 & & & \\ & & \ddots & & \\ & & & U_{11} & U_{12} \\ & & & U_{21} & U_{22} \end{bmatrix} . \tag{2.79}$$

The above is the prototype form of a two-level unitary matrix on $(m+1)$ qubits. Indeed, any two-level unitary matrix can be put into this form by exchanging columns and rows—equivalent to basis changes. The multi-control unitary gates thus arise naturally as we discuss the universal quantum computation in Section 2.4.

This is a three-qubit-control unitary gate.

```
In[∘]:= qc = QuantumCircuit[ControlledU[S@{1, 2, 3}, S[4, 1], "Label" → "U"]]
```

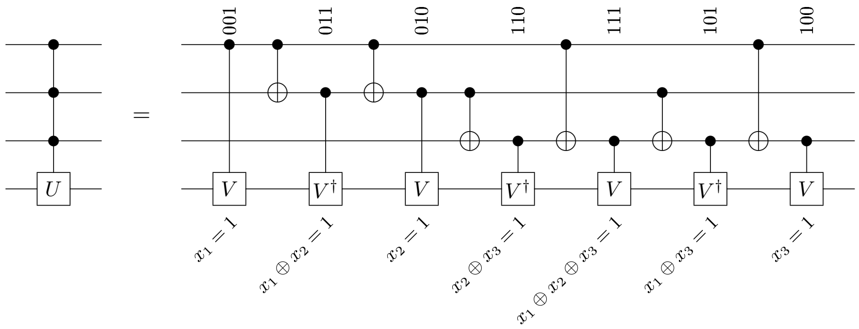
Out[∘]=



A reliable implementation of multi-control unitary gates is essential in many quantum algorithms. A notable example is the quantum oracle (see Section 4.2.1), which is a key component of quantum decision problems and quantum search problems. Here, we introduce some widely known methods to implement it systematically.

2.3.1 Gray-Code Method

We first discuss a systematic method based on the Gray code to decompose a multi-control unitary gate into factors of either the single-qubit controlled-unitary or CNOT gate. For example, consider a 3-control unitary gate as shown in the following quantum circuit (Barenco *et al.*, 1995)



, (2.80)

where $\hat{V}$ is another unitary operator such that $\hat{V}^4 = \hat{U}$. When every control qubit is set to $|1\rangle$, all $\hat{V}$ gates in the diagram take effects on the target qubit while $\hat{V}^\dagger$ gates are ineffective. To examine the case with the control qubits in a general state $|x_1\rangle \otimes \cdots \otimes |x_n\rangle$, note that the gates on the target qubit are effective under the conditions specified in terms of the bit values at the bottom of the diagram [see also Eq. (2.57)]. The conditions are systematically fulfilled by following the *Gray code sequence*, an arrange of bits such that two successive values differ in only one bit, the bit strings of which are indicated at the top of the diagram. The identity for the bitwise AND,

$$\begin{aligned}2^{n-1}(x_1 x_2 \cdots x_n) &= \sum_{k_1} x_{k_1} - \sum_{k_1<k_2} (x_{k_1} \oplus x_{k_2}) \\ &+ \sum_{k_1<k_2<k_3} (x_{k_1} \oplus x_{k_2} \oplus x_{k_3}) - \cdots + (-1)^{n-1}(x_1 \oplus x_2 \oplus \cdots \oplus x_n)\,, \quad (2.81)\end{aligned}$$

ensures that the quantum circuit on the right-hand side of (2.80) reproduces the desired multi-control unitary gate on the left-hand side.

In general, an n-control unitary gate can be implemented by combining 2^{n-1} controlled-$\hat{V}$ gates, $(2^{n-1} - 1)$ controlled-$\hat{V}^\dagger$ gates, and $(2^n - 2)$ CNOT gates, where $\hat{V}$ is a unitary operator satisfying $\hat{V}^{2^{n-1}} = \hat{U}$. For a relatively small n ($n \leq 8$), the Gray code is known to be the most efficient method. However, it grows exponentially and eventually becomes impractical. For such cases, several alternative methods have been proposed where the computational cost increases quadratically with the size of the control register (Barenco *et al.*, 1995; Nielsen & Chuang, 2011).

Consider a three-qubit register as an example.

```
$L = 3;
jj = Range[$L];
cc = S[jj, None];
```

This is the gate to operate on the target qubit.

```
In[◦]:= T = S[$L + 1, None];
        op = Rotation[Pi / 3, T[1]] // Elaborate
```

Out[◦]= $$\frac{\sqrt{3}}{2}-\frac{i\, S_4^x}{2}$$

This decomposes the multi-control unitary gate based on the Gray code -- showing only a part of it. Each component is either CNOT or a two-qubit-control unitary gate.

```
In[◦]:= gc = GrayControlledU[cc, op]; gc⟦ ;; 3⟧
```

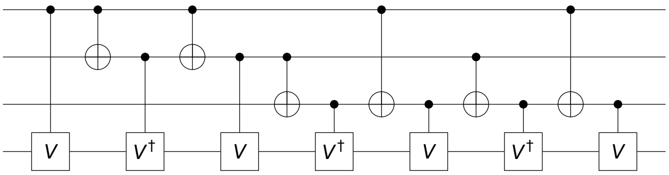
Out[◦]= $$\Big\{\texttt{ControlledU}\Big[\{S_1\},\ \frac{2^{1/4}e^{-\frac{i\pi}{24}}+2^{1/4}e^{\frac{i\pi}{24}}}{2\times 2^{1/4}}+\frac{\left(2^{1/4}e^{-\frac{i\pi}{24}}-2^{1/4}e^{\frac{i\pi}{24}}\right)S_4^+}{2\times 2^{1/4}}+\frac{\left(2^{1/4}e^{-\frac{i\pi}{24}}-2^{1/4}e^{\frac{i\pi}{24}}\right)S_4^-}{2\times 2^{1/4}},\ \texttt{Label}\to V\Big],$$

$$\texttt{CNOT}[\{S_1\},\{S_2\}],\ \texttt{ControlledU}\Big[\{S_2\},\ \frac{2^{1/4}e^{-\frac{i\pi}{24}}+2^{1/4}e^{\frac{i\pi}{24}}}{2\times 2^{1/4}}+\frac{\left(-2^{1/4}e^{-\frac{i\pi}{24}}+2^{1/4}e^{\frac{i\pi}{24}}\right)S_4^+}{2\times 2^{1/4}}+\frac{\left(-2^{1/4}e^{-\frac{i\pi}{24}}+2^{1/4}e^{\frac{i\pi}{24}}\right)S_4^-}{2\times 2^{1/4}},\ \texttt{Label}\to V^\dagger\Big]\Big\}$$

This is a quantum circuit model of the decomposition.

```
In[◦]:= qc = QuantumCircuit @@ gc
```

Out[◦]=



Finally check the result.

```
In[◦]:= expr = ExpressionFor[qc];
        expr2 = ControlledU[cc, op] // Elaborate;
        expr - expr2 // Simplify
Out[◦]= 0
```

2.3.2 Multi-Control NOT Gate

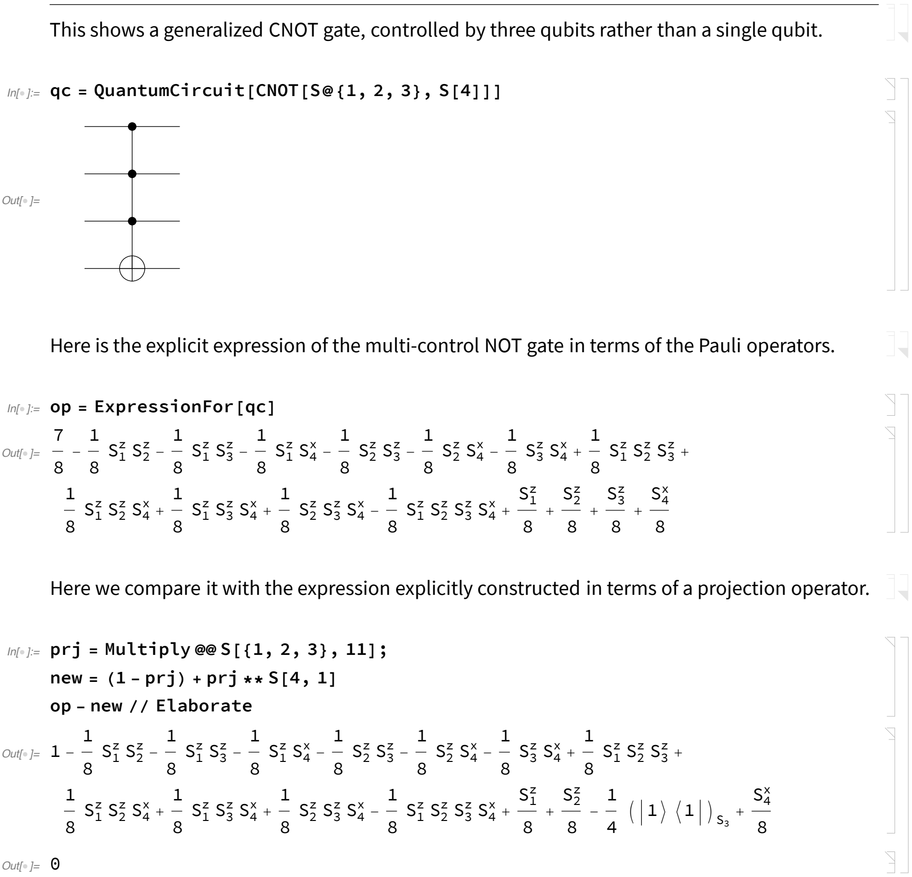
The multi-control NOT gate is an important special case of the multi-control unitary gate, where the unitary operator $\hat{U}$ equals to the Pauli $\hat{X}$. It transforms the logical basis states as [see Eq. (2.78)]

$$|c\rangle \otimes |t\rangle \mapsto |c\rangle \otimes \hat{X}^{c_1 c_2 \cdots c_m} |t\rangle \ , \tag{2.82a}$$

or equivalently,

$$|c\rangle \otimes |t\rangle \mapsto |c\rangle \otimes |(c_1 c_2 \cdots c_n) \oplus t\rangle \ . \tag{2.82b}$$

The multi-control NOT gate commonly occurs when one converts the two-level unitary transformation (see Section 2.2.3) on more than two qubits. Indeed, we will generalize the procedure and discuss a systematic way of conversion based on the Gray code in Section 2.4.



This shows a generalized CNOT gate, controlled by three qubits rather than a single qubit.

```
In[ ]:= qc = QuantumCircuit[CNOT[S@{1, 2, 3}, S[4]]]
```

Out[ ]=

Here is the explicit expression of the multi-control NOT gate in terms of the Pauli operators.

```
In[ ]:= op = ExpressionFor[qc]
```

Out[ ]= $\frac{7}{8} - \frac{1}{8} S_1^z S_2^z - \frac{1}{8} S_1^z S_3^z - \frac{1}{8} S_1^z S_4^x - \frac{1}{8} S_2^z S_3^z - \frac{1}{8} S_2^z S_4^x - \frac{1}{8} S_3^z S_4^x + \frac{1}{8} S_1^z S_2^z S_3^z + \frac{1}{8} S_1^z S_2^z S_4^x + \frac{1}{8} S_1^z S_3^z S_4^x + \frac{1}{8} S_2^z S_3^z S_4^x - \frac{1}{8} S_1^z S_2^z S_3^z S_4^x + \frac{S_1^z}{8} + \frac{S_2^z}{8} + \frac{S_3^z}{8} + \frac{S_4^x}{8}$

Here we compare it with the expression explicitly constructed in terms of a projection operator.

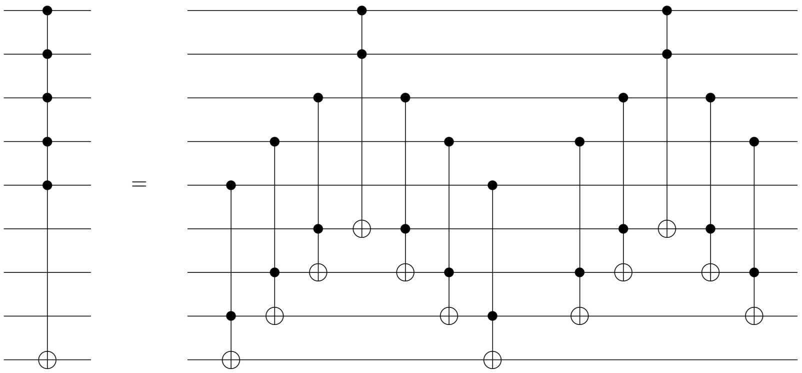
```
In[ ]:= prj = Multiply @@ S[{1, 2, 3}, 11];
        new = (1 - prj) + prj ** S[4, 1]
        op - new // Elaborate
```

Out[ ]= $1 - \frac{1}{8} S_1^z S_2^z - \frac{1}{8} S_1^z S_3^z - \frac{1}{8} S_1^z S_4^x - \frac{1}{8} S_2^z S_3^z - \frac{1}{8} S_2^z S_4^x - \frac{1}{8} S_3^z S_4^x + \frac{1}{8} S_1^z S_2^z S_3^z + \frac{1}{8} S_1^z S_2^z S_4^x + \frac{1}{8} S_1^z S_3^z S_4^x + \frac{1}{8} S_2^z S_3^z S_4^x - \frac{1}{8} S_1^z S_2^z S_3^z S_4^x + \frac{S_1^z}{8} + \frac{S_2^z}{8} - \frac{1}{4} \left(|1\rangle\langle 1|\right)_{S_3} + \frac{S_4^x}{8}$

Out[ ]= 0

An efficient implementation of a multi-control NOT gate uses additional qubits not directly involved in the gate operation itself as in the following quantum circuit

(Barenco *et al.*, 1995):



(2.83)

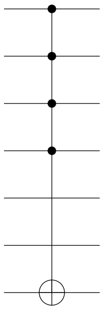
It is emphasized that the quantum states of the extra qubits should not be confused with the so-called "ancillary qubits" in the sense that they do not have to be initialized in a certain fixed quantum state and their state is not altered. The desired gate operation is performed properly regardless of the initial state of the extra qubits and their quantum state is restored after the gate operation.

Here we demonstrate a multi-control NOT gate.

```
$n = 4;
$m = 2;
cc = Range[$n];
aa = Range[$n + 1, $n + $m];
tt = $n + $m + 1;
```

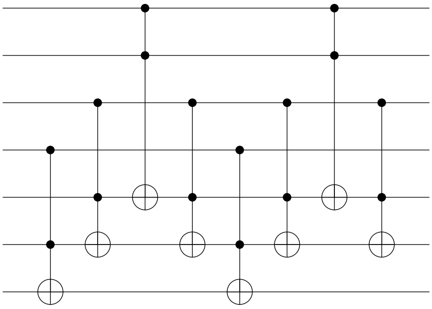
```
In[ ]:= qc = QuantumCircuit[CNOT[S[cc], S[tt]], "Visible" → S[aa]]
```

Out[ ]=



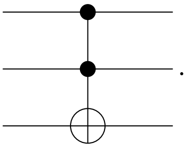
```
In[∘]:= tofa = Table[Toffoli[S[$n - j + 1], S[$n + $m - j + 1], S[tt - j + 1]], {j, 1, $m}];
        tofb = Toffoli[S[1], S[2], S[$n + 1]];
        tofc = Rest@tofa;
        new = QuantumCircuit[
          Sequence @@ Flatten@{tofa, tofb, Reverse@tofa, tofc, tofb, Reverse@tofc}]
```

Out[∘]=



```
In[∘]:= Timing[Elaborate[qc - new]]
Out[∘]= {6.6162, 0}
```

In the quantum circuit shown in (2.83), we have introduced the *Toffoli gate*, another special case of multi-control NOT gates denoted by the quantum circuit element



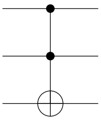
(2.84)

The Toffoli gate has attracted interest because it is universal for classical reversible computation. Unfortunately, however, it is not universal for quantum computation.

This is a quantum circuit model of the Toffoli gate.

```
In[∘]:= qc = QuantumCircuit[Toffoli[S[1], S[2], S[3]]]
        toff = ExpressionFor[qc]
```

Out[∘]=

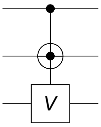


Out[∘]= $$\frac{3}{4} - \frac{1}{4} S_1^z S_2^z - \frac{1}{4} S_1^z S_3^x - \frac{1}{4} S_2^z S_3^x + \frac{1}{4} S_1^z S_2^z S_3^x + \frac{S_1^z}{4} + \frac{S_2^z}{4} + \frac{S_3^x}{4}$$

It can be implemented by a combination of two-qubit gates.

In[∘]:=
```
gray = GrayControlledU[S@{1, 2}, S[3, 1]];
qc = QuantumCircuit[gray]
expr = ExpressionFor[qc]
toff - expr
```

Out[∘]=



Out[∘]= $\frac{3}{4} - \frac{1}{4} S_1^z S_2^z - \frac{1}{4} S_1^z S_3^x - \frac{1}{4} S_2^z S_3^x + \frac{1}{4} S_1^z S_2^z S_3^x + \frac{S_1^z}{4} + \frac{S_2^z}{4} + \frac{S_3^x}{4}$

Out[∘]= 0

Here V = $\sqrt{X}$, and its matrix representation looks like this.

In[∘]:=
```
matV = MatrixPower[ThePauli[1], 1 / 2];
matV * 2 / (1 + I) // MatrixForm
opV = Elaborate@ExpressionFor[matV, S[3]]
```

Out[∘]//MatrixForm=

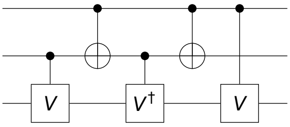
$$\begin{pmatrix} 1 & -i \\ -i & 1 \end{pmatrix}$$

Out[∘]= $\left(\frac{1}{2} + \frac{i}{2}\right) + \left(\frac{1}{2} - \frac{i}{2}\right) S_3^x$

Reversing the above circuit gives the identical result.

In[∘]:=
```
qc = QuantumCircuit @@ Reverse@gray
expr = ExpressionFor[qc]
toff - expr
```

Out[∘]=



Out[∘]= $\frac{3}{4} - \frac{1}{4} S_1^z S_2^z - \frac{1}{4} S_1^z S_3^x - \frac{1}{4} S_2^z S_3^x + \frac{1}{4} S_1^z S_2^z S_3^x + \frac{S_1^z}{4} + \frac{S_2^z}{4} + \frac{S_3^x}{4}$

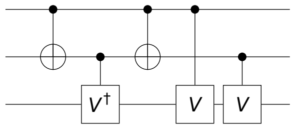
Out[∘]= 0

As noted by Smolin & DiVincenzo (1996), it can also be reordered as follows, and this rearrangement is useful in optimizing the *Fredkin gate*, another universal gate for classical reversible computation.

This shows another slightly different implementation of the Toffoli gate.

```
In[◦]:= qc = QuantumCircuit @@ Permute[gray, Cycles[{{4, 3, 2, 1}}]]
        expr = ExpressionFor[qc]
        toff - expr
```

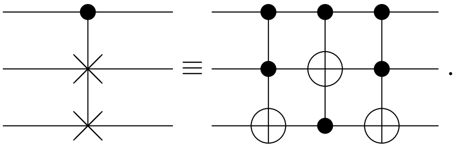
Out[◦]=



Out[◦]= $\frac{3}{4} - \frac{1}{4} S_1^z S_2^z - \frac{1}{4} S_1^z S_3^x - \frac{1}{4} S_2^z S_3^x + \frac{1}{4} S_1^z S_2^z S_3^x + \frac{S_1^z}{4} + \frac{S_2^z}{4} + \frac{S_3^x}{4}$

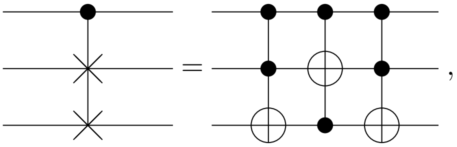
Out[◦]= `0`

The Fredkin gate "swaps" the states of the two target qubits when the control qubit is set in the state $|1\rangle$, as depicted by the following two equivalent quantum circuit elements



(2.85)

In fact, the relation between the SWAP gate and the CNOT gate suggests that there is another equivalent quantum circuit of the Fredkin gate,



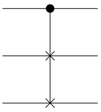
(2.86)

which is simpler than the one given above. This equivalent quantum circuit was used by Smolin & DiVincenzo (1996) for an efficient implementation of the Fredkin gate in terms of the elementary gates: Like the Toffoli gate, the Fredkin gate is not universal for quantum computation.

This shows the quantum circuit model of the quantum Fredkin gate.

```
In[◦]:= qc = QuantumCircuit[Fredkin[S[1], S[2], S[3]]]
```

Out[◦]=



This is an explicit operator expression of the Fredkin gate in terms of the Pauli operators.

```
In[◦]:= op = Elaborate[qc]
```

Out[◦]= $\frac{3}{4} + \frac{1}{4} S_2^x S_3^x + \frac{1}{4} S_2^y S_3^y + \frac{1}{4} S_2^z S_3^z - \frac{1}{4} S_1^z S_2^x S_3^x - \frac{1}{4} S_1^z S_2^y S_3^y - \frac{1}{4} S_1^z S_2^z S_3^z + \frac{S_1^z}{4}$

This is the matrix representation of the Fredkin gate in the logical basis.

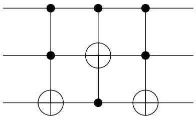
```
In[∘]:= mat = Matrix[qc];
        mat // MatrixForm
Out[∘]//MatrixForm=
```

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

The Fredkin gate is equivalent to a combination of three Toffoli gates.

```
In[∘]:= qc2 = QuantumCircuit[
          Toffoli[S[1], S[2], S[3]],
          Toffoli[S[1], S[3], S[2]],
          Toffoli[S[1], S[2], S[3]]]
```

Out[∘]=



```
In[∘]:= op2 = Elaborate[qc2]
```

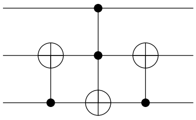
Out[∘]= $\frac{3}{4}+\frac{1}{4}S_2^x S_3^x+\frac{1}{4}S_2^y S_3^y+\frac{1}{4}S_2^z S_3^z-\frac{1}{4}S_1^z S_2^x S_3^x-\frac{1}{4}S_1^z S_2^y S_3^y-\frac{1}{4}S_1^z S_2^z S_3^z+\frac{S_1^z}{4}$

```
In[∘]:= op - op2
Out[∘]= 0
```

In fact, the relation between the SWAP gate and the CNOT gate suggests that there is another equivalent quantum circuit model of the Fredkin gate.

```
In[∘]:= new = QuantumCircuit[
          CNOT[S[3], S[2]],
          Toffoli[S[1], S[2], S[3]],
          CNOT[S[3], S[2]]
         ]
```

Out[∘]=



```
In[∘]:= qc - new // Elaborate
Out[∘]= 0
```

2.4 Universal Quantum Computation

In classical computation, it is known that a finite set of logic gates—typically including AND, OR, and NOT—is sufficient to calculate any binary function. The set is said to be *universal* for classical computation. Does there exist a universal set of elementary quantum logic gates for quantum computation that enables to implement any arbitrary quantum unitary operation? In this section, we examine this question.

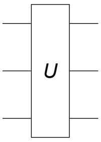
Suppose that we want to implement an arbitrary unitary operation $\hat{U}$ on an n-qubit register. We start by decomposing $\hat{U}$ into a set of two-level unitary transformations $\hat{T}_k$; that is, $\hat{U} = \hat{T}_L \cdots \hat{T}_2\hat{T}_1$. We have already discussed the decomposition procedure for two-qubit systems in Section 2.2.3. The procedure is still the same for a larger number of qubits. First remove the off-diagonal element U_{1n} at the top-right corner of the matrix representation of $\hat{U}$ by multiplying $\hat{U}$ with a two-level unitary transformation $\hat{T}_1^\dagger$ on the right, and repeat it along the top row and then the next consecutive rows to zero all off-diagonal elements and to set every diagonal element to unity, so that $\hat{U}\hat{T}_1^\dagger\hat{T}_2^\dagger \cdots \hat{T}_L^\dagger = \hat{I}$. The decomposition of $\hat{U}$ is then given by $\hat{U} = \hat{T}_L \cdots \hat{T}_2\hat{T}_1$ as desired.

Consider a three-qubit system, and an arbitrary unitary operation on it.

```
In[ ]:= $n = 3;
        SS = S[Range[$n], None];
        mat = RandomUnitary[2^$n];
        mat⟦ ;; 3,  ;; 3⟧ // MatrixForm
Out[ ]//MatrixForm=
 ( -0.0394109 + 0.298858 i   -0.169212 - 0.530373 i    0.105654 - 0.469469 i  )
 ( -0.295389 + 0.0625766 i   -0.310104 + 0.289566 i   0.0152336 + 0.295724 i  )
 (  -0.315498 - 0.279878 i  -0.0866186 - 0.201536 i  -0.161459 + 0.0770453 i  )
```

```
In[ ]:= op = ExpressionFor[mat, SS];
        qc = QuantumCircuit[{op, "Label" → "U"}]
```

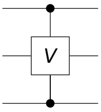
Out[ ]=



```
In[ ]:= twl = TwoLevelDecomposition[mat];
        twl⟦3⟧ // Matrix // MatrixForm
Out[ ]//MatrixForm=
 ( 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  0.0879404 - 0.109594 i          0.990079          0 )
 ( 0  0  0  0  0       -0.990079          0.0879404 + 0.109594 i  0 )
 ( 0  0  0  0  0           0                         0              1 )
```

Some two-level unitary transformations are themselves multi-control unitary gates, which can be implemented in terms of elementary gates as shown in Section 2.3. The most common example is shown in Eq. (2.79). On three qubits, another example is

$$\begin{bmatrix} 1 &&&&&&& \\ & 1 &&&&&& \\ && 1 &&&&& \\ &&& 1 &&&& \\ &&&& 1 &&& \\ &&&&& V_{11} && V_{12} \\ &&&&&& 1 & \\ &&&&& V_{21} && V_{22} \end{bmatrix} \doteq \tag{2.87}$$



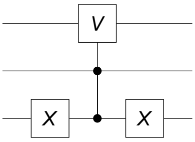
,

where the gate with label 'V' on the second qubit indicates unitary operator

$$\hat{V} \doteq \begin{bmatrix} V_{11} & V_{12} \\ V_{21} & V_{22} \end{bmatrix}. \tag{2.88}$$

Some other two-level unitary transformations are not multi-control unitary gates and yet simple variants. An example is

$$\begin{bmatrix} 1 &&&&&&& \\ & 1 &&&&&& \\ && V_{11} &&&& V_{12} & \\ &&& 1 &&&& \\ &&&& 1 &&& \\ &&&&& 1 && \\ && V_{21} &&&& V_{22} & \\ &&&&&&& 1 \end{bmatrix} \doteq \tag{2.89}$$



.

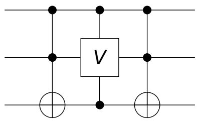
In this case, unitary operator $\hat{V}$ acts conditioned on the second qubit in $|1\rangle$ and the third qubit in $|0\rangle$ (rather than $|1\rangle$), and the above two-level unitary transformation is a simple variant of multi-control unitary gate. Through these examples, we see that any two-level unitary gate with the two non-trivial columns and rows attributed to two logical basis states differing only in one qubit is either a multi-control unitary gate or a simple variant of it. It is thus clear that such a two-level unitary transformation can be implemented in terms of elementary gates.

Two-level unitary transformations that do not belong to the above class requires additional manipulation. In Section 2.2.3, we examined an implementation on two qubits. The idea was to simultaneously exchange columns and rows so that the resulting matrix corresponds to a controlled-unitary gate such as in Eq. (2.62). An exchange of columns and rows was implemented by the CNOT gate or its variant. For an arbitrary number of qubits, the idea is essentially the same in that a series of simultaneous exchanges of columns and rows bring any two-level unitary transformation to a multi-control unitary gate and that each exchange of columns of rows can be carried by the multi-control NOT gate or a simple variant of it. For example, consider a two-level unitary transformation $\hat{T}$ on three qubits of the following form,

$$\hat{T} \doteq \begin{bmatrix} 1 &&&&&&& \\ & 1 &&&&&& \\ && 1 &&&&& \\ &&& 1 &&&& \\ &&&& 1 &&& \\ &&&&& V_{11} & V_{12} & \\ &&&&& V_{21} & V_{22} & \\ &&&&&&& 1 \end{bmatrix}. \tag{2.90}$$

The simultaneous exchange of the last two columns and rows brings the above two-level unitary matrix to the form in (2.87). On the other hand, given that the last two columns (or rows) in the matrix representation correspond to the logical basis states $|110\rangle$ and $|111\rangle$, respectively, the simultaneous exchange of the last two columns and rows can be carried out by the multi-control NOT gate with the first two qubits as controls and the third qubit as target. Putting these two observations together, we find that

$\hat{T} =$

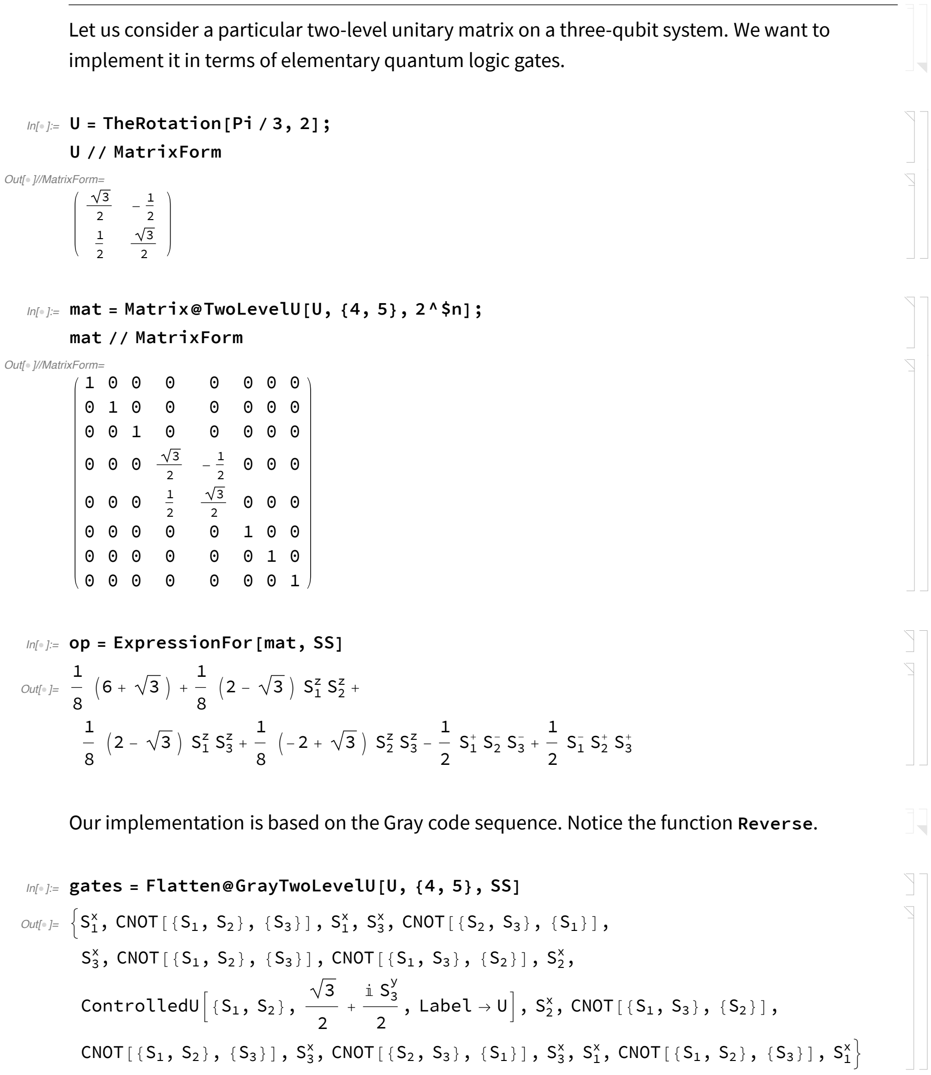


. (2.91)

So far, the exchanges of columns and rows have been performed on an *ad hoc* basis. An irritating issue is that every exchange of columns or rows cannot be carried out by a single multi-control NOT gate or a simple variant of it. Fortunately, however, one can achieve any desired exchange by combining a sequence of other exchanges that can be carried out by a multi-control NOT gate (or a simple variant). For example, suppose that we want to exchange the sixth and seventh columns and rows. Noting that these columns and rows in the matrix representation correspond to logical basis states $|101\rangle$ and $|110\rangle$, it is clear that the exchange cannot be fulfilled by a single multi-control NOT gate. However, it can be achieved by first exchanging the sixth and eighth columns and rows (respectively corresponding to logical basis states $|101\rangle$ and $|111\rangle$) and then the eighth and seventh columns and rows (respectively corresponding to $|111\rangle$ and $|110\rangle$). The first exchange can be carried out by the multi-control NOT gate with the first and third qubit as controls and the second qubit as target while the second exchange can be implemented by the multi-control NOT gate with the first two qubits as controls and the third qubit as target. The key point is that each exchange must be between a pair of columns (and rows) corresponding to logical basis states that differ in only one qubit. For this purpose, a systematic and efficient way is to use the Gray code sequence. In terms of classical bits, it is a sequence of bit strings where any two successive strings are different only in one bit.[2.2] As an example, the Gray code sequence from $|010\rangle$ to $|101\rangle$ is given by

$$|010\rangle \to |110\rangle \to |111\rangle \to |101\rangle\,. \tag{2.92}$$

The first step in the above sequence corresponds to the NOT gate on the first qubit conditioned on the second qubit in $|1\rangle$ and the third qubit in $|0\rangle$, which is a simple variant of the multi-control NOT gate with the last two qubits as control and the first qubit as target—it just needs additional NOT gates on the third qubit before and after the multi-control NOT gate. The other steps can be carried out simply by multi-control NOT gates.



Let us consider a particular two-level unitary matrix on a three-qubit system. We want to implement it in terms of elementary quantum logic gates.

```
In[∘]:= U = TheRotation[Pi / 3, 2];
        U // MatrixForm
```

Out[∘]//MatrixForm=

$$\begin{pmatrix} \frac{\sqrt{3}}{2} & -\frac{1}{2} \\ \frac{1}{2} & \frac{\sqrt{3}}{2} \end{pmatrix}$$

```
In[∘]:= mat = Matrix@TwoLevelU[U, {4, 5}, 2^$n];
        mat // MatrixForm
```

Out[∘]//MatrixForm=

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{\sqrt{3}}{2} & -\frac{1}{2} & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{1}{2} & \frac{\sqrt{3}}{2} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

```
In[∘]:= op = ExpressionFor[mat, SS]
```

Out[∘]=

$$\frac{1}{8}\left(6+\sqrt{3}\right)+\frac{1}{8}\left(2-\sqrt{3}\right) S_1^z S_2^z + \frac{1}{8}\left(2-\sqrt{3}\right) S_1^z S_3^z + \frac{1}{8}\left(-2+\sqrt{3}\right) S_2^z S_3^z - \frac{1}{2} S_1^+ S_2^- S_3^- + \frac{1}{2} S_1^- S_2^+ S_3^+$$

Our implementation is based on the Gray code sequence. Notice the function `Reverse`.

```
In[∘]:= gates = Flatten@GrayTwoLevelU[U, {4, 5}, SS]
```

Out[∘]=

$$\left\{S_1^x,\ \text{CNOT}[\{S_1, S_2\}, \{S_3\}],\ S_1^x,\ S_3^x,\ \text{CNOT}[\{S_2, S_3\}, \{S_1\}],\ S_3^x,\ \text{CNOT}[\{S_1, S_2\}, \{S_3\}],\ \text{CNOT}[\{S_1, S_3\}, \{S_2\}],\ S_2^x,\ \text{ControlledU}\left[\{S_1, S_2\}, \frac{\sqrt{3}}{2}+\frac{i\, S_3^y}{2}, \text{Label}\to U\right],\ S_2^x,\ \text{CNOT}[\{S_1, S_3\}, \{S_2\}],\ \text{CNOT}[\{S_1, S_2\}, \{S_3\}],\ S_3^x,\ \text{CNOT}[\{S_2, S_3\}, \{S_1\}],\ S_3^x,\ S_1^x,\ \text{CNOT}[\{S_1, S_2\}, \{S_3\}],\ S_1^x\right\}$$

[2.2] In Section 2.3.1, the same method was also used in a slightly different context.

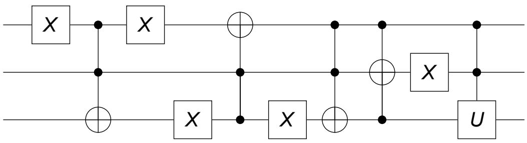
```
In[◦]:= new = Apply[Dot, Matrix[#, SS] & /@ Elaborate /@ gates] // Normal // Simplify;
        new // MatrixForm
```

Out[◦]//MatrixForm=

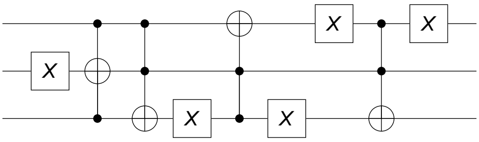
$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{\sqrt{3}}{2} & -\frac{1}{2} & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{1}{2} & \frac{\sqrt{3}}{2} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

```
In[◦]:= qc1 = QuantumCircuit @@ gates⟦ ;; 10⟧
        qc2 = QuantumCircuit @@ gates⟦11 ;;⟧
```

Out[◦]=



Out[◦]=



```
In[◦]:= new = Matrix[qc2].Matrix[qc1] // Normal // Simplify;
        new // MatrixForm
```

Out[◦]//MatrixForm=

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{\sqrt{3}}{2} & -\frac{1}{2} & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{1}{2} & \frac{\sqrt{3}}{2} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

We have seen that one can implement any unitary operation combining the CNOT and single-qubit gates. A subtle issue is that single-qubit gates form a continuum of unitary operators. It is impractical to implement them to an infinite accuracy. Fortunately, it is known that a discrete set of gates is sufficient to perform universal quantum computation. The Hadamard, quadrant phase, octant phase, and CNOT gates form a universal set of gates in the sense that an arbitrary gate on n qubits can be approximated to arbitrary accuracy using a quantum circuit composed of only such gates. Furthermore, the set remains universal if octant phase gates are replaced with Toffoli gates.

2.5 Measurements

We conclude this chapter with a few discussions on measurement. In quantum computers, measurement is assumed to be performed independently on individual

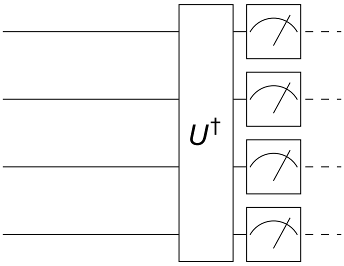
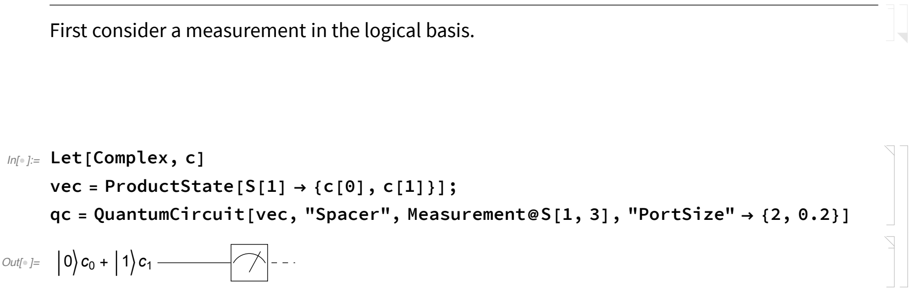


Figure 2.6: Measurement in a basis $\{|\alpha_x\rangle\}$ other than the logical basis. The unitary operator $\hat{U}$ here corresponds to the basis change $|\alpha_y\rangle = \hat{U}\,|y\rangle = \sum_x |x\rangle\,\langle x|\alpha_y\rangle$.

qubits in the logical basis, $\{|x\rangle : x = 0, 1, \ldots, 2^n - 1\}$.

What if a measurement in another basis, say, $\{|\alpha_x\rangle = \hat{U}\,|x\rangle\}$ is required? We require that the input state $|\alpha_x\rangle$ should end up with the logical basis state $|x\rangle$ with unit probability so that the measurement yields outcome x. This process is described by the measurement operators $\hat{M}_x := |x\rangle\,\langle\alpha_x| = |x\rangle\,\langle x|\,\hat{U}^\dagger$. Evidently, they satisfy the condition $\sum_x \hat{M}_x^\dagger \hat{M}_x = \hat{I}$ for measurement operators (see Postulate 1.3). Now we note that the operators $\hat{P}_x := |x\rangle\,\langle x|$ describe nothing but the measurement in the logical basis. This implies that by simply applying the inverse unitary operation $\hat{U}^\dagger$ before the measurement, the measurement in the new basis $\{|\alpha_x\rangle\}$ can be achieved through measurement in the logical basis (see Fig. 2.6).

For example, suppose that a qubit is in the state $|\psi\rangle = |0\rangle\,c_0 + |1\rangle\,c_1$. By default, a measurement is assumed to be in the logical basis and the measurement statistics reflect the probability distribution $P_0 = |c_0|^2$ and $P_1 = |c_1|^2$ as illustrated in the demonstration below.



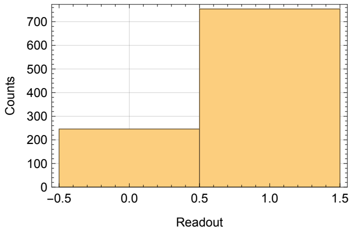
First consider a measurement in the logical basis.

```
In[]:= Let[Complex, c]
       vec = ProductState[S[1] → {c[0], c[1]}];
       qc = QuantumCircuit[vec, "Spacer", Measurement@S[1, 3], "PortSize" → {2, 0.2}]
```

Out[]= $|0\rangle c_0 + |1\rangle c_1$

```
In[ ]:= Block[
         {c, data},
         c[0] = 1 / 2;
         c[1] = Sqrt[3] / 2;
         data = Table[ExpressionFor[qc]; Readout@S[1, 3], 1000];
         Histogram[data, FrameLabel → {"Readout", "Counts"}]
        ]
```

Out[ ]=



Next, suppose that we want to make a measurement, say, in the eigenbasis $\{|\pm\rangle\}$ of the Pauli X operator. In this case, it is the Hadamard gate that gives the desired basis change. In the new basis, the state vector $|\psi\rangle$ is given by

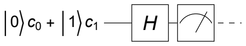
$$|\psi\rangle = |+\rangle \frac{c_0 + c_1}{\sqrt{2}} + |-\rangle \frac{c_0 - c_1}{\sqrt{2}}, \tag{2.93}$$

In this case, the measurement statistics are in accordance with the probability distribution $P_\pm = |c_0 \pm c_1|^2/2$.

Now consider a measurement in the eigen-basis of the Pauli X operator.

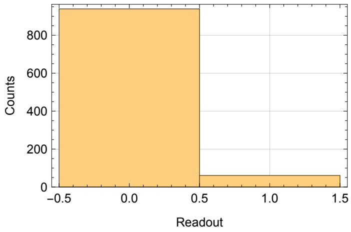
```
In[ ]:= Let[Complex, c]
        vec = ProductState[S[1] → {c[0], c[1]}];
        qc = QuantumCircuit[vec, S[1, 6], Measurement@S[1, 3], "PortSize" → {2, 0.2}]
```

Out[ ]=



```
In[ ]:= Block[
         {c, data},
         c[0] = 1 / 2;
         c[1] = Sqrt[3] / 2;
         data = Table[ExpressionFor[qc]; Readout@S[1, 3], 1000];
         Histogram[data, FrameLabel → {"Readout", "Counts"}]
        ]
```

Out[ ]=



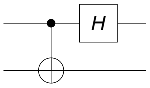
Another interesting example is the *Bell measurement*. The Bell measurement is a measurement on two qubits in the basis of Bell states,

$$\begin{aligned} |\beta_0\rangle &= \frac{|00\rangle + |11\rangle}{\sqrt{2}}, \\ |\beta_1\rangle &= \frac{|01\rangle + |10\rangle}{\sqrt{2}}, \\ |\beta_2\rangle &= \frac{|01\rangle - |10\rangle}{\sqrt{2}}, \\ |\beta_3\rangle &= \frac{|00\rangle - |11\rangle}{\sqrt{2}}. \end{aligned} \tag{2.94}$$

Recall that the Bell states can be generated from the logical basis states by the so-called quantum entangler circuit, a combination of the Hadamard gate and the CNOT gate discussed in Section 2.2.1. Therefore, the Bell measurement can be achieved by applying the inverse of the entangling operation.

This is the "disentangler" quantum circuit, which is the inverse of the entangler quantum circuit
.

```
In[∘]:= disentangler = QuantumCircuit[CNOT[S[1], S[2]], S[1, 6]]
Out[∘]=
```



This shows that the disentangler quantum circuit maps the Bell states into the logical basis states.

```
In[∘]:= op = ExpressionFor[disentangler];
        bs = BellState@S@{1, 2};
        Thread[bs → op ** bs] // LogicalForm // TableForm
Out[∘]//TableForm=
```

$$\frac{|0_{S_1}0_{S_2}\rangle + |1_{S_1}1_{S_2}\rangle}{\sqrt{2}} \to |0_{S_1}0_{S_2}\rangle$$

$$\frac{|0_{S_1}1_{S_2}\rangle + |1_{S_1}0_{S_2}\rangle}{\sqrt{2}} \to |0_{S_1}1_{S_2}\rangle$$

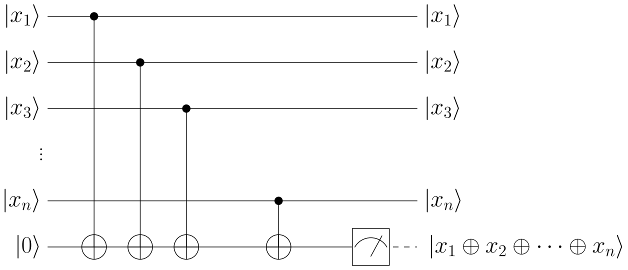
$$\frac{|0_{S_1}1_{S_2}\rangle - |1_{S_1}0_{S_2}\rangle}{\sqrt{2}} \to |1_{S_1}1_{S_2}\rangle$$

$$\frac{|0_{S_1}0_{S_2}\rangle - |1_{S_1}1_{S_2}\rangle}{\sqrt{2}} \to |1_{S_1}0_{S_2}\rangle$$

In quantum error-correction codes and stabilizer circuits to be discussed in Chapter 6, one often encounters the so-called *Pauli measurements*, the measurements of tensor products of the single-qubit Pauli operators. To be specific, let us consider the measurement of $\hat{Z} \otimes \hat{Z} \otimes \cdots \otimes \hat{Z}$ on n qubits; the measurement of general Pauli operators can be achieved by incorporating the trick of basis change discussed above. Before going further, it is important to note that the measurement of $\hat{Z} \otimes \hat{Z} \otimes \cdots \otimes \hat{Z}$ is a collective measurement and must be distinguished from the sequential measurements of $\hat{Z}$ on individual qubits. For example, when an *even* number of qubits are prepared in quantum state

$$|\psi\rangle = |00\cdots0\rangle\, c_0 + |11\cdots1\rangle\, c_1, \tag{2.95}$$

the measurement of $\hat{Z}\otimes\hat{Z}\otimes\cdots\otimes\hat{Z}$ always gives outcome 1 (out of two eigenvalues ±1 of the observable), and the quantum state remains intact even after the measurement. On the other hand, the measurement of $\hat{Z}$, say, on the first qubit produces outcome 0 or 1 (corresponding to eigenvalues ±1, respectively, of $\hat{Z}$), and the quantum state accordingly collapses to either $|00\cdots0\rangle$ or $|11\cdots1\rangle$. The measurement of $\hat{Z}\otimes\hat{Z}\otimes\cdots\otimes\hat{Z}$ can be implemented within the quantum circuit model of quantum computation by using an ancillary qubit initialized in state $|0\rangle$ and by coupling the qubits in the native register to the ancillary qubits as follows



. (2.96)

The CNOT gates transform the state of the ancillary qubit from the initial state $|0\rangle$ to the final state $|x_1\oplus x_2\oplus\cdots\oplus x_n\rangle$ involving the logical values of the n qubits in the native register (see Problem 2.9), and hence a simple measurement on the ancillary qubits in the computational basis leads to the desired result.

More sophisticated measurements may also be necessary for fast quantum algorithms and quantum error corrections. A common example is the quantum phase estimation, which is one of the core parts of the quantum factorization algorithm. We will discuss it in Section 4.4.

Problems

2.1. Let $\hat{\Phi}(\phi)$ be the *phase gate*, which gives rise to a *relative* phase shift by $\phi\in[0,2\pi)$,

$$\hat{\Phi}(\phi) : |0\rangle \mapsto |0\rangle\ , \quad |1\rangle \mapsto |1\rangle\, e^{i\phi}\,. \tag{2.97}$$

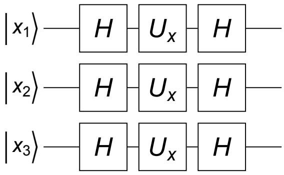
(a) Show that on a n-qubit quantum register,

$$[\hat{\Phi}(\phi)]^{\otimes n}\,|x\rangle = |x\rangle\, e^{i\phi(x_1+\cdots+x_n)} \tag{2.98}$$

for any $x = 0, 1, \ldots, 2^n - 1$.

(b) Explicitly evaluate the state $[\hat{\Phi}(\phi)]^{\otimes n}\hat{H}^{\otimes n}\,|0\rangle$, where $\hat{H}$ is the Hadamard gate.

2.2. Let $\hat{U}_x(\phi)$ be the rotation around the x-axis by angle ϕ on a single qubit. Explicitly analyze and evaluate the following quantum circuit



, (2.99)

where $|x\rangle := |x_1\rangle \otimes |x_2\rangle \otimes |x_3\rangle$ is a 3-qubit logical basis state and the labels "H" and "U_x" indicate the Hadamard gate $\hat{H}$ and the rotation gate $\hat{U}_x(\phi)$, respectively. Generalize the result and show that

$$\hat{H}^{\otimes n}[\hat{U}_x(\phi)]^{\otimes n}\hat{H}^{\otimes n}\,|x\rangle = e^{-i\phi n/2}\,|x\rangle\, e^{i\phi(x_1+\cdots x_n)} \tag{2.100}$$

for any $x = 0, 1, \ldots, 2^n - 1$.

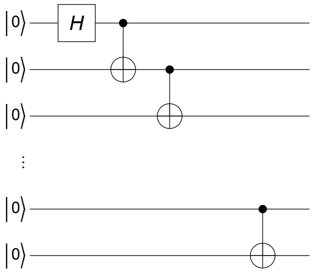
2.3. Let $\hat{S}^\mu$ be the Pauli operators. Show that

$$e^{i\hat{S}^\mu\phi/2}\hat{S}^\nu e^{-i\hat{S}^\mu\phi/2} = \hat{S}^\nu\cos(\phi) + \sum_\lambda \hat{S}^\lambda \epsilon_{\lambda\mu\nu}\sin(\phi) \tag{2.101}$$

for all $\mu, \nu = x, y, z$ and $\mu \neq \nu$.

2.4. The lever on Bob's quantum computer is stuck in the "forward" position, so it can only perform CNOT gates controlled by qubit 1 on qubit 2 (target). His computer can still perform single-qubit operations normally. How can he perform a CNOT gate controlled by qubit 2 on qubit 1?[2.3]

2.5. Consider the following quantum circuit with n qubits:



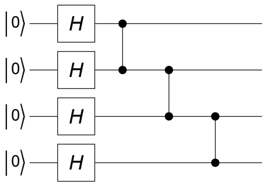
(2.102)

Show that it generates the Greenberger-Horne-Zeilinger state in Eq. (2.40), identical to the one from the quantum circuit in Fig. 2.5 (b).

2.6. Consider a quantum register of four qubits.

[2.3] This problem was originally published in Gottesman (1999).

(a) Analyze the following quantum circuit



(2.103)

and evaluate the resulting state $|\Psi\rangle$ explicitly. The state is a so-called *cluster state* or *graph state*, a crucial resource in the measurement-based quantum computation (see Section 3.4).

(b) Show that in the state $|\Psi\rangle$ from (b), every qubit is *maximally entangled* with the rest of the qubits. That is, the *reduced density matrix* $\hat{\rho}_j$ of the jth qubit, $\hat{\rho}_j := \mathrm{Tr}_{k\neq j} |\Psi\rangle \langle\Psi|$, is given by $\hat{\rho}_j = \hat{I}/2$, exhibiting coherence in no basis.

2.7. Suppose that a qubit is known to be in one of the two eigenstates of the unitary operator

$$\hat{U} = \hat{\sigma}^0 \cos(\phi/2) - i\hat{\sigma}^x \sin(\phi/2) \tag{2.104}$$

with the angle ϕ *known*. Construct a quantum circuit to figure out the unknown state using an additional qubit.

Hint: Use a controlled-unitary gate to acquire the one-bit information. This is a simplified version of the quantum phase estimation procedure (see Section 4.4), but it can be worked out without resorting to it.

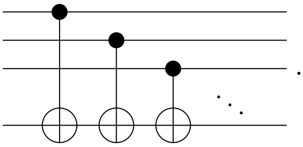
2.8. Suppose that a two-qubit system is known to be in one of the four eigenstates of the unitary operator

$$\hat{U} = e^{i\phi} \left(|0\rangle \langle 0| + i\,|1\rangle \langle 1| - |2\rangle \langle 2| - i\,|3\rangle \langle 3| \right). \tag{2.105}$$

Construct a quantum circuit to figure out the unknown state using two additional qubits.

Hint: Use the property (2.18) of the Hadamard gate and those of the controlled-unitary gates, where a unitary operator acts on a two-qubit system controlled by a single qubit.

2.9. Consider the following quantum circuit consisting of the CNOT gates on an n-qubit quantum register



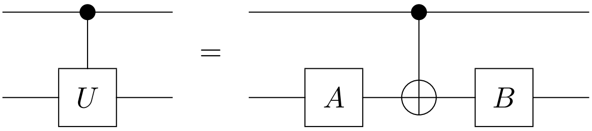
(2.106)

(a) Find the output state for the input of a logical basis state $|x_1\rangle \otimes \cdots \otimes |x_n\rangle \in \mathcal{S}^{\otimes n}$.

(b) Find the output state for the input state $|+\rangle \otimes \cdots \otimes |+\rangle \otimes |-\rangle$, where $|\pm\rangle := (|0\rangle \pm |1\rangle)/\sqrt{2}$.

2.10. Let $\hat{U}$ be a unitary operator on a single qubit. Show that the following three statements are equivalent:

(a) There exist unitary gates $\hat{A}$ and $\hat{B}$ such that



(2.107)

(b) There exists a unitary operator $\hat{W}$ such that $\hat{U} = \hat{W}\hat{Z}\hat{W}^\dagger$.

(c) $\operatorname{Tr}\hat{U} = 0$ and $\det\hat{U} = -1$.

2.11. Let $P(i \leftrightarrow j)$ be the matrix exchanging the ith and jth rows (columns) of vectors/matrices. For example, on a four-dimensional space,

$$P(1 \leftrightarrow 2) = \begin{bmatrix} 0 & 1 & & \\ 1 & 0 & & \\ & & 1 & \\ & & & 1 \end{bmatrix}. \tag{2.108}$$

$\hat{P}(i \leftrightarrow j)$ is the corresponding operator.

(a) Find a quantum circuit for $\hat{P}(2 \leftrightarrow 3)$ on a two-qubit system (four-dimensional vector space $\mathcal{S} \otimes \mathcal{S}$) using CNOT gates only.

(b) Find a quantum circuit for $\hat{P}(2 \leftrightarrow 3)$ on a three-qubit system ($\mathcal{S}^{\otimes 3}$) using multi-control NOT gates only.

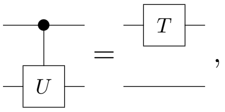
(c) Find a quantum circuit for $\hat{P}(4 \leftrightarrow 7)$ on a three-qubit system ($\mathcal{S}^{\otimes 3}$) using multi-control NOT gates only.

2.12. Let $\hat{U} = e^{i\phi}\hat{I}$ be a single-qubit unitary operator that *globally* shifts the phase by ϕ. Also define

$$\hat{T} = |0\rangle\langle 0| + e^{i\phi}|1\rangle\langle 1| \doteq \begin{bmatrix} 1 & 0 \\ 0 & e^{i\phi} \end{bmatrix}, \tag{2.109}$$

which shifts the *relative* phase by ϕ.

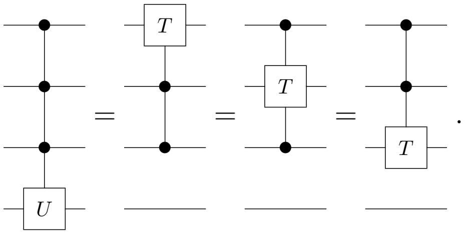
(a) Show that



(2.110)

where the labels “U” and “T” denote the unitary operators $\hat{U}$ and $\hat{T}$, respectively.

(b) Show also that



(2.111)