Chapter 2

Boolean Algebra and its Applications

From a field of mathematics called *modern algebra*, *classical* computers are designed and maintained. Over a hundred years, algebraists have studied for mathematical systems that are called *boolean algebras*. The name *boolean algebra* honors a fascinating English mathematician, George Boole because in 1854 he published a *classic* book, *An Investigation of the Laws of Thought, on which Are Founded Mathematical Theories of Logic and Probabilities*. Boole's stated objective (intention) was to complete a mathematical analysis of logic. The *calculus of propositions* and *algebra of sets* were based on Boole's investigation. In this book we designate the algebra now used in the design and maintenance of quantum logical circuitry as *boolean algebra*.

There are several advantages in having a mathematical technique for the illustration of the internal workings of a quantum algorithm (circuit) for solving each different kind of applications in **IBM**'s quantum computers. The first advantage is to that it is often far more convenient to calculate with algebraic expressions used to describe the internal workings of a quantum algorithm (circuit) than it is to apply schematic or even logical diagrams. The second advantage is to that an ordinary algebraic expression that describes the internal workings of a quantum algorithm (circuit) may be reduced or simplified. This enables that the designer of quantum algorithms (circuits) achieves economy of construction and reliability of quantum operation. Boolean algebra also provides an economical and straightforward way of designing quantum algorithms (circuits) for solving each different kind of applications. In all, a knowledge of boolean algebra is indispensable in the computing field. In this chapter, we describe how to com-

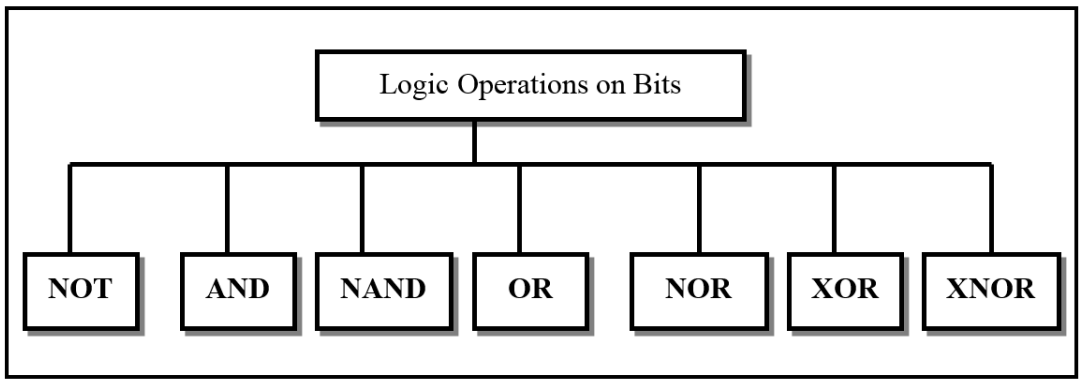

Figure 2.1: Logic operations on bits.

plete logic operations that appear in Figure 2.1 and include **NOT**, **AND**, **NAND**, **OR**, **NOR**, **Exclusive-OR** (**XOR**) and **Exclusive-NOR** (**XNOR**) with quantum logic gates in the backend *ibmqx4* or a simulator in **IBM**'s quantum computers. We also illustrate how to complete several applications from boolean algebra.

2.1 Illustration to NOT Operation

The **NOT** operation acquires a single input and yields one single output. It inverts the value of a bit into the one's complement of the bit. This is to say that the **NOT** operation for a bit provides the following result:

$$\begin{aligned} &\textbf{NOT } 1 = 0 \\ &\textbf{NOT } 0 = 1 \end{aligned} \tag{2.1}$$

The value of a Boolean variable (a bit) is only zero (0) or one (1). Therefore, **NOT** of a Boolean variable (a bit) q[0], written as $\overline{q[0]}$ is equal to one (1) if and only if q[0] is equal to zero (0). Similarly, $\overline{q[0]}$ is equal to zero (0) if and only if q[0] is equal to one (1). The rules in (2.1) for the **NOT** operation may also be expressed in the form of a truth table that is shown in Table 2.1.

Input	Output
q[0]	$\overline{q[0]}$
0	1
1	0

Table 2.1: The truth table for the **NOT** operation.

From (2.1) and Table 2.1, the **NOT** operation of a bit is to invert the value of the bit into its one's complement. The **NOT** operation of *n* bits is to provide the corresponding one's complement for each input in *n* inputs by means of implementing the **NOT** operation of a bit of *n* times. The following subsections will be used to illustrate how to design the quantum programs to complete the **NOT** operation of a bit and the **NOT** operation of two bits.

2.1.1 Quantum Program to the One's Complement of a Bit

Consider that two values for unsigned integer of one bit are, respectively, 0 (0_{10})

and $1(1_{10})$, where 0_{10} is the decimal representation of zero and 1_{10} is the decimal representation of one. We want to simultaneously take the one's complement of those two values.

In Listing 2.1, the program in the backend *ibmqx4* with five quantum bits in **IBM**'s quantum computer is the *first* example of the *second* chapter in which we illustrate how to write a quantum program to invert 0 (0_{10}) and $1(1_{10})$ into their one's complement. Figure 2.2 is the corresponding quantum circuit of the program in Listing 2.1. The statement "OPENQASM 2.0;" on line one of Listing 2.1 is to indicate that the program is written with version 2.0 of Open QASM. Then, the statement "include "qelib1.inc";" on line two of Listing 2.1 is to continue parsing the file "qelib1.inc" as if

```
1. OPENQASM 2.0;
2. include "qelib1.inc";
3. qreg q[5];
4. creg c[5];
5. h q[0];
6. x q[0];
7. measure q[0] -> c[0];
```

Listing 2.1: The program of taking the one's complement to the input of a bit.

the contents of the file were pasted at the location of the include statement, where the file "qelib1.inc" is **Quantum Experience (QE) Standard Header** and the path is specified relative to the current working directory.

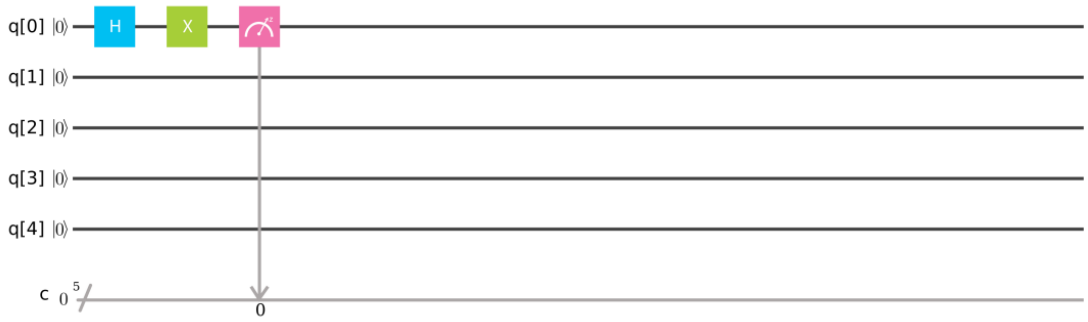

Figure 2.2: The corresponding quantum circuit of the program in Listing 2.1.

Next, the statement "qreg q[5];" on line three of Listing 2.1 is to declare that in the program there are five quantum bits. In the left top of Figure 2.2, five quantum bits are subsequently q[0], q[1], q[2], q[3] and q[4]. The initial value of each quantum bit is set

to |0>. We use a quantum bit q[0] to encode the input of a bit that is unsigned integer of a bit. Next, the statement "creg c[5];" on line four of Listing 2.1 is to declare that there are five classical bits in the program. In the left bottom of Figure 2.2, five classical bits are respectively c[0], c[1], c[2], c[3] and c[4]. The initial value of each classical bit is set to 0.

Next, the statement "h q[0];" on line five of Listing 2.1 actually completes $\begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{pmatrix} \times \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{pmatrix} = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 \\ 1 \end{pmatrix} = \frac{1}{\sqrt{2}} (\begin{pmatrix} 1 \\ 0 \end{pmatrix} + \begin{pmatrix} 0 \\ 1 \end{pmatrix}) = \frac{1}{\sqrt{2}} (|0> + |1>)$. This is to say that the statement "h q[0];" on line five of Listing 2.1 is to apply the Hadamard gate to convert q[0] from one state |0> to another state $\frac{1}{\sqrt{2}}$ (|0> + |1>) (its superposition). In its superposition, |0> with the amplitude $\frac{1}{\sqrt{2}}$ encodes the value 0 (zero) to the input of a bit and |1> with the amplitude $\frac{1}{\sqrt{2}}$ encodes the value 1 (one) to the input of a bit. Next, the statement "x q[0];" on line six of Listing 2.1 actually completes $\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \times \begin{pmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{pmatrix} = \begin{pmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{pmatrix} = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 \\ 1 \end{pmatrix} = \frac{1}{\sqrt{2}} (\begin{pmatrix} 0 \\ 1 \end{pmatrix} + \begin{pmatrix} 1 \\ 0 \end{pmatrix}) = \frac{1}{\sqrt{2}} (|1> + |0>)$. This indicates that the statement "x q[0];" on line six of Listing 2.1 inverts |0> with the amplitude $\frac{1}{\sqrt{2}}$ (the input zero of a bit) into |1> with the amplitude $\frac{1}{\sqrt{2}}$ (its corresponding one's complement) and also inverts |1> with the amplitude $\frac{1}{\sqrt{2}}$ (the input one of a bit) into |0> with the amplitude $\frac{1}{\sqrt{2}}$ (its corresponding one's complement). This also implies that two instructions (two **NOT** operations) of taking one's complement to the input of a bit are completed by means of using one quantum instruction "x q[0];".

Next, the statement "measure q[0] -> c[0];" on line seven of Listing 2.1 is to measure the first quantum bit q[0] and to record the measurement outcome by overwriting the first classical bit c[0]. In the backend *ibmqx4* with five quantum bits in **IBM**'s quantum computers, we use the command "simulate" to execute the program in Listing 2.1. The measured result is shown in Figure 2.3. From Figure 2.3, we obtain the

answer 00001 (c[4] = q[4] = |0>, c[3] = q[3] = |0>, c[2] = q[2] = |0>, c[1] = q[1] = |0> and c[0] = q[0] = |1>) with the probability 0.530. This is to say that we obtain the one's complement (q[0] = |1>) with the probability 0.530 to the input zero (0) of a bit. Or we obtain the answer 00000 with the probability 0.470 (c[4] = q[4] = |0>, c[3] = q[3] = |0>, c[2] = q[2] = |0>, c[1] = q[1] = |0> and c[0] = q[0] = |0>). This is also to say that we obtain the one's complement (q[0] = |0>) with the probability 0.470 to the input one (1) of a bit.

Figure 2.3: After the measurement to the program in Listing 2.1 is completed, we obtain the answer 00001 with the probability 0.530 or the answer 00000 with the probability 0.470.

2.1.2 Quantum Program to the One's Complement of Two Bits

Consider that four values to unsigned integer of two bits are subsequently 00 (0_{10}), 01 (1_{10}), 10 (2_{10}) and 11(3_{10}), where 0_{10} is the decimal representation of zero, 1_{10} is the decimal representation of one, 2_{10} is the decimal representation of two and 3_{10} is the decimal representation of three. We want to simultaneously take the one's complement of those four values.

In Listing 2.2, the program in the backend *ibmqx4* with five quantum bits in **IBM**'s quantum computer is the *second* example of the *second* chapter in which we describe how to write a quantum program to take the one's complement of 00 (0_{10}), 01 (1_{10}), 10 (2_{10}) and 11 (3_{10}). Figure 2.4 is the corresponding quantum circuit of the program in Listing 2.2. The statement "OPENQASM 2.0;" on line one of Listing 2.2 is to point to that the program is written with version 2.0 of Open QASM. Next, the statement "include "qelib1.inc";" on line two of Listing 2.2 is to continue parsing the file "qelib1.inc" as if the contents of the file were pasted at the location of the include statement, where the file "qelib1.inc" is **Quantum Experience (QE) Standard Header** and the path is specified relative to the current working directory.

```
1. OPENQASM 2.0;
```

```
2. include "qelib1.inc";
3. qreg q[5];
4. creg c[5];
5. h q[0];
6. h q[1];
7. x q[0];
8. x q[1];
9. measure q[0] -> c[0];
10. measure q[1] -> c[1];
```

Listing 2.2: The program of taking the one's complement to the input of two bits.

Then, the statement "qreg q[5];" on line three of Listing 2.2 is to declare that in the program there are five quantum bits. In the left top of Figure 2.4, five quantum bits are subsequently q[0], q[1], q[2], q[3] and q[4]. The initial value of each quantum bit is set to |0>. We use two quantum bits q[0] and q[1] to encode the input of two bits that are unsigned integer of two bits. Next, the statement "creg c[5];" on line four of Listing 2.2 is to declare that there are five classical bits in the program. In the left bottom of Figure 2.4, five classical bits are respectively c[0], c[1], c[2], c[3] and c[4]. The initial value of each classical bit is set to 0.

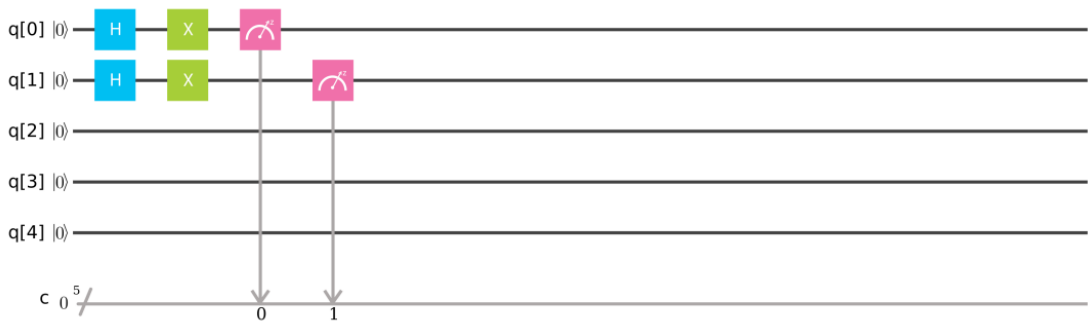

Figure 2.4: The corresponding quantum circuit of the program in Listing 2.2.

Then, the statement "h q[0];" on line five of Listing 2.2 actually completes $\begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{pmatrix} \times \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{pmatrix} = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 \\ 1 \end{pmatrix} = \frac{1}{\sqrt{2}} (\begin{pmatrix} 1 \\ 0 \end{pmatrix} + \begin{pmatrix} 0 \\ 1 \end{pmatrix}) = \frac{1}{\sqrt{2}} (|0> + |1>)$. This indicates that the statement "h q[0];" on line five of Listing 2.2 is to apply the Hadamard gate to convert q[0] from one state |0> to another state $\frac{1}{\sqrt{2}} (|0> + |1>)$ (its superposition).

Next, the statement “h q[1];” on line six of Listing 2.2 actually completes $\begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{pmatrix} \times \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{pmatrix} = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 \\ 1 \end{pmatrix} = \frac{1}{\sqrt{2}} (\begin{pmatrix} 1 \\ 0 \end{pmatrix} + \begin{pmatrix} 0 \\ 1 \end{pmatrix}) = \frac{1}{\sqrt{2}}$ (|0> + |1>). This impiles that the statement “h q[1];” on line six of Listing 2.2 is to use the Hadamard gate to convert q[1] from one state |0> to another state $\frac{1}{\sqrt{2}}$ (|0> + |1>) (its superposition). Hence, the superposition of the two quantum bits q[0] and q[1] is ($\frac{1}{\sqrt{2}}$ (|0> + |1>)) ($\frac{1}{\sqrt{2}}$ (|0> + |1>)) = $\frac{1}{2}$ (|0> |0> + |0> |1> + |1> |0> + |1> |1>) = $\frac{1}{2}$ (|00> + |01> + |10> + |11>). In the superposition, |00> with the amplitude $\frac{1}{2}$ encodes the value 00 (zero) to the input of two bits, |01> with the amplitude $\frac{1}{2}$ encodes the value 1 (one) to the input of two bits, |10> with the amplitude $\frac{1}{2}$ encodes the value 2 (two) to the input of two bits and |11> with the amplitude $\frac{1}{2}$ encodes the value 3 (three) to the input of two bits.

Next, the statement “x q[0];” on line seven of Listing 2.2 actually completes $\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \times \begin{pmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{pmatrix} = \begin{pmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{pmatrix} = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 \\ 1 \end{pmatrix} = \frac{1}{\sqrt{2}} (\begin{pmatrix} 0 \\ 1 \end{pmatrix} + \begin{pmatrix} 1 \\ 0 \end{pmatrix}) = \frac{1}{\sqrt{2}}$ (|1> + |0>) and the statement “x q[1];” on line eight of Listing 2.2 actually completes $\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \times \begin{pmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{pmatrix} = \begin{pmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{pmatrix} = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 \\ 1 \end{pmatrix} = \frac{1}{\sqrt{2}} (\begin{pmatrix} 0 \\ 1 \end{pmatrix} + \begin{pmatrix} 1 \\ 0 \end{pmatrix}) = \frac{1}{\sqrt{2}}$ (|1> + |0>). This is to say that the two statements “x q[0];” and “x q[1];” on line seven and line eight of Listing 2.2 inverts |00> with the amplitude $\frac{1}{2}$ (the input zero of two bits) into |11> with the amplitude $\frac{1}{2}$ (its one’s complement), inverts |01> with the amplitude $\frac{1}{2}$ (the input one of two bits) into |10> with the amplitude $\frac{1}{2}$ (its one’s complement), inverts |10> with the amplitude

$\frac{1}{2}$ (the input two of two bits) into |01> with the amplitude $\frac{1}{2}$ (its one's complement) and inverts |11> with the amplitude $\frac{1}{2}$ (the input three of two bits) into |00> with the amplitude $\frac{1}{2}$ (its one's complement). This indicates that eight instructions (eight **NOT** operations) of taking one's complement to the input of two bits are completed by means of applying two quantum operations "x q[0];" and "x q[1];".

Next, the two statements "measure q[0] -> c[0];" and "measure q[1] -> c[1];"on line nine and line ten of Listing 2.2 is to measure the first quantum bit q[0] and the second quantum bit q[1] and to record the measurement outcome by overwriting the first classical bit c[0] and the second classical bit c[1]. In the backend *ibmqx4* with five quantum bits in **IBM**'s quantum computers, we apply the command "simulate" to execute the program in Listing 2.2. The measured result is shown in Figure 2.5. From Figure 2.5, we obtain the answer 00001 (c[4] = q[4] = |0>, c[3] = q[3] = |0>, c[2] = q[2] = |0>, c[1] = q[1] = |0> and c[0] = q[0] = |1>) with the probability 0.330. This is to say that we obtain the one's complement (q[1] = |0> and q[0] = |1>) with the probability 0.330 to the input two (10) of two bits. Or we obtain the answer 00010 (c[4] = q[4] = |0>, c[3] = q[3] = |0>, c[2] = q[2] = |0>, c[1] = q[1] = |1> and c[0] = q[0] = |0>) with the probability 0.260. This indicates that we obtain the one's complement (q[1] = |1> and q[0] = |0>) with the probability 0.260 to the input one (01) of two bits. Or we obtain the answer 00011 (c[4] = q[4] = |0>, c[3] = q[3] = |0>, c[2] = q[2] = |0>, c[1] = q[1] = |1> and c[0] = q[0] = |1>) with the probability 0.210. This implies that we obtain the one's complement (q[1] = |1> and q[0] = |1>) with the probability 0.210 to the input zero (00) of two bits. Or we obtain the answer 00000 (c[4] = q[4] = |0>, c[3] = q[3] = |0>, c[2] = q[2] = |0>, c[1] = q[1] = |0> and c[0] = q[0] = |0>) with the probability 0.200. This is to say that we obtain the one's complement (q[1] = |0> and q[0] = |0>) with the probability 0.200 to the input three (11) of two bits.

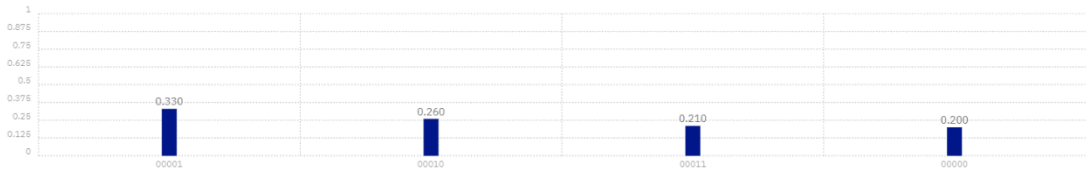

Figure 2.5: After the measurement to the program in Listing 2.2 is completed, we obtain the answer 00001 with the probability 0.330, the answer 00010 with the probability 0.260, the answer 00011 with the probability 0.210 or the answer 00000 with the probability 0.200.

2.2 The Toffoli Gate of Three Quantum Bits

The Toffoli gate that is also known as the ***controlled-controlled-NOT*** or ***CCNOT*** gate of three quantum bits is

$$U_{CCN} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix}. \tag{2.2}$$

It is assumed that U_{CCN}^{+} is the conjugate-transpose matrix of U_{CCN} and is equal to $((U_{CCN})^{*})^{t} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix}$, where the * is the complex conjugation and the t is the transpose operation. Because $U_{CCN} \times ((U_{CCN})^{*})^{t} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix} \times \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix} = ((U_{CCN})^{*})^{t} \times$

$$U_{CCN} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix} \times \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix} =$$

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$, U_{CCN} is a unitary matrix or a unitary operator. This indicates that the ***controlled-controlled-NOT*** or ***CCNOT*** gate (the Toffoli gate) U_{CCN} is one of quantum gates with three quantum bits. If the quantum state l_0 |000> + l_1 |001> + l_2 |010> + l_3 |011> + l_4 |100> + l_5 |101> + l_6 |110> + l_7 |111> is written in a vector notation as

$$\begin{pmatrix} l_0 \\ l_1 \\ l_2 \\ l_3 \\ l_4 \\ l_5 \\ l_6 \\ l_7 \end{pmatrix}, \tag{2.3}$$

with the first entry l_0 is the amplitude for |000>, the second entry l_1 is the amplitude for |001>, the third entry l_2 is the amplitude for |010>, the fourth entry l_3 is the amplitude for |011>, the fifth entry l_4 is the amplitude for |100>, the sixth entry l_5 is the amplitude for |101>, the seventh entry l_6 is the amplitude for |110> and the eighth entry l_7 is the amplitude for |111>, then the corresponding output from the ***CCNOT*** gate U_{CCN} is

$$\begin{pmatrix} l_0 \\ l_1 \\ l_2 \\ l_3 \\ l_4 \\ l_5 \\ l_7 \\ l_6 \end{pmatrix} = l_0 \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} + l_1 \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} + l_2 \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} + l_3 \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} + l_4 \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \end{pmatrix} + l_5 \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \end{pmatrix} + l_7 \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \end{pmatrix} +$$

$$l_6 \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{pmatrix} = l_0 \text{ |000>} + l_1 \text{ |001>} + l_2 \text{ |010>} + l_3 \text{ |011>} + l_4 \text{ |100>} + l_5 \text{ |101>} + l_7 \text{ |110>} +$$

l_6 |111>. (2.4)

This is to say that the ***CCNOT*** gate U_{CCN} converts three quantum bits from one state l_0 |000> + l_1 |001> + l_2 |010> + l_3 |011> + l_4 |100> + l_5 |101> + l_6 |110> + l_7 |111> to another state l_0 |000> + l_1 |001> + l_2 |010> + l_3 |011> + l_4 |100> + l_5 |101> + l_7 |110> + l_6 |111>. This implies that in the ***CCNOT*** gate U_{CCN} if the two control quantum bits (the *first* quantum bit and the second quantum bit) is set to 0, then the target quantum bit (the *third* quantum bit) is left alone. If the two control quantum bits (the *first* quantum bit and the *second* quantum bit) is both set to 1, then the target quantum bit (the *third* quantum bit) is flipped. The probability of measuring a |000>, |001>, |010>, |011>, |100> or |101> is unchanged, the probability of measuring a |110> is $|l_7|^2$ and the probability of measuring a |111> is $|l_6|^2$ after using the ***CCNOT*** gate U_{CCN}. Since

$$(U_{CCN})^2 = U_{CCN} \times U_{CCN} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix} \times \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix},$$

using the ***CCNOT*** gate U_{CCN} twice to a state is equivalent to do nothing to it. The graphical representation of the ***CCNOT*** gate U_{CCN} is shown in Figure 2.6. In Figure 2.6, the left top two wires are *control bits* that are unchanged by the action of the ***CCNOT*** gate. The bottom wire is a *target bit* that is flipped if both control bits are set to 1, and otherwise is left alone, where ⊕ is addition modulo two.

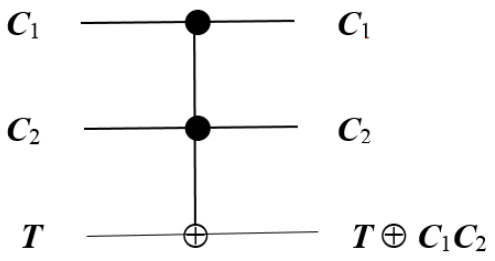

Figure 2.6: The circuit representation of the ***CCNOT*** gate.

2.2.1 Implementing the Toffoli Gate of Three Quantum Bits

The Toffoli gate has three input bits and three output bits. Its truth table is shown in

Input			Output		
C_1	C_2	T	C_1	C_2	T
0	0	0	0	0	0
0	1	0	0	1	0
1	0	0	1	0	0
1	1	0	1	1	1
0	0	1	0	0	1
0	1	1	0	1	1
1	0	1	1	0	1
1	1	1	1	1	0

Table 2.2: The truth table for the Toffoli gate with three input bits and three output bits.

Table 2.2. In **IBM Q** Experience, it does not provide one quantum instruction (operation) of implementing the ***CCNOT*** gate (the Toffoli gate) with three quantum bits. We decompose **CCNOT** gate into *six* **CNOT** gates and *nine* gates of one quantum bits that are shown in Figure 2.7. In Figure 2.7, H is the Hadamard gate, $T = \begin{bmatrix} 1 & 0 \\ 0 & e^{\sqrt{-1}\times\frac{\pi}{4}} \end{bmatrix}$ and $T^{+} = \begin{bmatrix} 1 & 0 \\ 0 & e^{-1\times\sqrt{-1}\times\frac{\pi}{4}} \end{bmatrix}$. In **IBM Q** Experience, the available gates are that **CNOT** is the only gate with two quantum bits and the other gates act on single quantum bit and they are introduced in the previous chapter. In the backend *ibmqx4* with five quantum bits, there are only six pairs of quantum bits to which a **CNOT** gate can be applied. Connectivity of the ***CNOT*** gate in the backend *ibmqx4* with five quantum bits is shown in Figure 1.21 in Subsection 1.11.1.

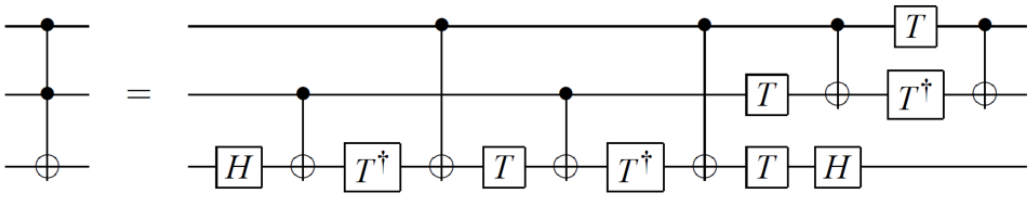

Figure 2.7: Decomposing **CCNOT** gate into six **CNOT** gates and nine gates of one bit.

In Listing 2.3, the program in the backend *ibmqx4* with five quantum bits in **IBM**'s quantum computer is the *third* example of the *second* chapter in which we illustrate how to write a quantum program to implement a Toffoli gate (a ***CCNOT*** gate) of three quantum bits. Figure 2.8 is the corresponding quantum circuit of the program in Listing 2.3. The statement "OPENQASM 2.0;" on line one of Listing 2.3 is to indicate that the program is written with version 2.0 of Open QASM. Next, the statement "include "qelib1.inc";" on line two of Listing 2.3 is to continue parsing the file "qelib1.inc" as if the contents of the file were pasted at the location of the include statement, where the file "qelib1.inc" is **Quantum Experience (QE) Standard Header** and the path is specified relative to the current working directory.

```
1.  OPENQASM 2.0;
2.  include "qelib1.inc";
3.  qreg q[5];
4.  creg c[5];
5.  h q[1];
6.  h q[2];
7.  h q[0];
8.  cx q[1],q[0];
9.  tdg q[0];
10. cx q[2],q[0];
11. t q[0];
12. cx q[1],q[0];
13. tdg q[0];
14. cx q[2],q[0];
15. t q[0];
16. t q[1];
17. h q[0];
18. cx q[2],q[1];
19. tdg q[1];
20. t q[2];
21. cx q[2],q[1];
22. measure q[0] -> c[0];
23. measure q[1] -> c[1];
24. measure q[2] -> c[2];
```

Listing 2.3: The program of implementing a ***CCNOT*** gate of three quantum bits.

Next, the statement “qreg q[5];” on line three of Listing 2.3 is to declare that in the program there are five quantum bits. In the left top of Figure 2.8, five quantum bits are subsequently q[0], q[1], q[2], q[3] and q[4]. The initial value of each quantum bit is set to |0>. We apply three quantum bits q[2], q[1] and q[0] to subsequently encode the first control bit, the second control bit and the target bit. For the convenience of our explanation, $q[k]^0$ for $0 \le k \le 4$ is to represent the value 0 of q[k] and $q[k]^1$ for $0 \le k \le 4$ is to represent the value 1 of q[k]. Similarly, for the convenience of our explanation, an initial state vector of implementing a Toffoli gate is as follows:

$$|\Phi_0\rangle = |q[2]^0\rangle\, |q[1]^0\rangle\, |q[0]^0\rangle = |0\rangle\, |0\rangle\, |0\rangle = |000\rangle.$$

Then, the statement “creg c[5];” on line four of Listing 2.3 is to declare that there are five classical bits in the program. In the left bottom of Figure 2.8, five classical bits are respectively c[0], c[1], c[2], c[3] and c[4]. The initial value of each classical bit is set to 0.

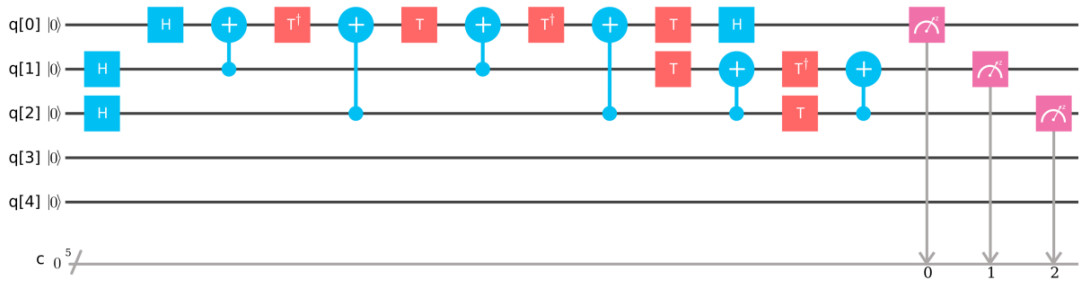

Figure 2.8: The corresponding quantum circuit of the program in Listing 2.3.

Next, the two statements “h q[1];” and “h q[2];”on line five and line six of Listing 2.3 implement two Hadamard gates of the *first* time slot of the quantum circuit in Figure 2.8 and both actually complete $\begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{pmatrix} \times \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{pmatrix} = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 \\ 1 \end{pmatrix} = \frac{1}{\sqrt{2}} (\begin{pmatrix} 1 \\ 0 \end{pmatrix} + \begin{pmatrix} 0 \\ 1 \end{pmatrix}) = \frac{1}{\sqrt{2}} (|0\rangle + |1\rangle)$. This is to say that converting q[1] from one state |0> to another state $\frac{1}{\sqrt{2}} (|0\rangle + |1\rangle)$ (its superposition) and converting q[2] from one state |0> to another state $\frac{1}{\sqrt{2}} (|0\rangle + |1\rangle)$ (its superposition) are completed. Thus, the superposition of the two quantum bits q[2] and q[1] is $(\frac{1}{\sqrt{2}} (|0\rangle + |1\rangle)) (\frac{1}{\sqrt{2}} (|0\rangle + |1\rangle))$

$= \frac{1}{2}$ (|0> |0> + |0> |1> + |1> |0> + |1> |1>) $= \frac{1}{2}$ (|00> + |01> + |10> + |11>). Because in the *first* time slot of the quantum circuit in Figure 2.8 there is no quantum gate to act on the quantum bit q[0], its state |0> is not changed. Therefore, after two Hadamard gates in the *first* time slot of the quantum circuit in Figure 2.8 are implemented by means of using the two statements "h q[1];" and "h q[2];"on line five and line six of Listing 2.3, the following new state vector is obtained:

$$|\Phi_1\rangle = \frac{1}{2} (|0\rangle |0\rangle |0\rangle + |0\rangle |1\rangle |0\rangle + |1\rangle |0\rangle |0\rangle + |1\rangle |1\rangle |0\rangle)$$

$$= \frac{1}{2} (|000\rangle + |010\rangle + |100\rangle + |110\rangle).$$

The next 12 time slots in the quantum circuit of Figure 2.8 implement a Toffoli gate. Next, the statement "h q[0];" on line seven of Listing 2.3 takes the new state vector $|\Phi_1\rangle = \frac{1}{2} (|000\rangle + |010\rangle + |100\rangle + |110\rangle)$ as its input and completes one Hadamard gate for q[0] in the *second* time slot. This is to say that the statement "h q[0];" converts q[0] from one state |0> to another state $\frac{1}{\sqrt{2}} (|0\rangle + |1\rangle)$ (its superposition). Because there is no other quantum gate in the *second* time slot to act on quantum bits q[2] and q[1], their states are not changed. Therefore, after one Hadamard gate for q[0] in the *second* time slot is completed by the statement "h q[0];" on line seven of Listing 2.3, the following new state vector is obtained:

$|\Phi_2\rangle = \frac{1}{2} (|0\rangle |0\rangle \frac{1}{\sqrt{2}} (|0\rangle + |1\rangle) + |0\rangle |1\rangle \frac{1}{\sqrt{2}} (|0\rangle + |1\rangle) + |1\rangle |0\rangle \frac{1}{\sqrt{2}} (|0\rangle + |1\rangle) + |1\rangle |1\rangle \frac{1}{\sqrt{2}} (|0\rangle + |1\rangle)) = \frac{1}{2\sqrt{2}} (|000\rangle + |001\rangle + |010\rangle + |011\rangle + |100\rangle + |101\rangle + |110\rangle + |111\rangle)$.

Next, the statement "cx q[1],q[0];" on line eight of Listing 2.3 takes the new state vector $|\Phi_2\rangle = \frac{1}{2\sqrt{2}} (|000\rangle + |001\rangle + |010\rangle + |011\rangle + |100\rangle + |101\rangle + |110\rangle + |111\rangle)$ as its input and completes one ***CNOT*** gate for q[1] and q[0] in the *third* time slot. If the value of the control bit q[1] is equal to one, then the value of the target bit q[0] is flipped. Otherwise, it is not changed. Because there is no other quantum gate in the *third* time slot to act on quantum bit q[2], its state is not changed. Hence, after one ***CNOT*** gate for

q[1] and q[0] in the *third* time slot is completed by the statement "cx q[1],q[0];" on line eight of Listing 2.3, the following new state vector is obtained:

$$|\Phi_3\rangle = \frac{1}{2\sqrt{2}} (|000\rangle + |001\rangle + |011\rangle + |010\rangle + |100\rangle + |101\rangle + |111\rangle + |110\rangle).$$

Next, the statement "tdg q[0];" on line nine of Listing 2.3 takes the new state vector $|\Phi_3\rangle = \frac{1}{2\sqrt{2}} (|000\rangle + |001\rangle + |011\rangle + |010\rangle + |100\rangle + |101\rangle + |111\rangle + |110\rangle)$ as its input and completes one T^+ gate for q[0] in the *fourth* time slot. If the value of q[0] is equal to one, then its phase is changed as $e^{-1\times\sqrt{-1}\times\frac{\pi}{4}}$. Otherwise, its phase is not changed. There is no other quantum gate in the *fourth* time slot to act on quantum bits q[2] and q[1], so their states are not changed. Thus, after one T^+ gate for q[0] in the *fourth* time slot is completed by the statement "tdg q[0];" on line nine of Listing 2.3, the following new state vector is obtained:

$$|\Phi_4\rangle = \frac{1}{2\sqrt{2}} (|000\rangle + e^{-1\times\sqrt{-1}\times\frac{\pi}{4}}|001\rangle + e^{-1\times\sqrt{-1}\times\frac{\pi}{4}}|011\rangle + |010\rangle + |100\rangle + e^{-1\times\sqrt{-1}\times\frac{\pi}{4}}|101\rangle + e^{-1\times\sqrt{-1}\times\frac{\pi}{4}}|111\rangle + |110\rangle).$$

Next, the statement "cx q[2],q[0];" on line ten of Listing 2.3 takes the new state vector $|\Phi_4\rangle = \frac{1}{2\sqrt{2}} (|000\rangle + e^{-1\times\sqrt{-1}\times\frac{\pi}{4}}|001\rangle + e^{-1\times\sqrt{-1}\times\frac{\pi}{4}}|011\rangle + |010\rangle + |100\rangle + e^{-1\times\sqrt{-1}\times\frac{\pi}{4}}|101\rangle + e^{-1\times\sqrt{-1}\times\frac{\pi}{4}}|111\rangle + |110\rangle)$ as its input and completes one ***CNOT*** gate for q[2] and q[0] in the *fifth* time slot. If the value of the control bit q[2] is equal to one, then the value of the target bit q[0] is flipped. Otherwise, it is not changed. Because there is no other quantum gate in the *fifth* time slot to act on quantum bit q[1], its state is not changed. Thus, after one ***CNOT*** gate for q[2] and q[0] in the *fifth* time slot is completed by the statement "cx q[2],q[0];" on line ten of Listing 2.3, the following new state vector is obtained:

$$|\Phi_5\rangle = \frac{1}{2\sqrt{2}} (|000\rangle + e^{-1\times\sqrt{-1}\times\frac{\pi}{4}}|001\rangle + e^{-1\times\sqrt{-1}\times\frac{\pi}{4}}|011\rangle + |010\rangle + |101\rangle + e^{-1\times\sqrt{-1}\times\frac{\pi}{4}}|100\rangle + e^{-1\times\sqrt{-1}\times\frac{\pi}{4}}|110\rangle + |111\rangle).$$

Next, the statement "t q[0];" on line eleven of Listing 2.3 takes the new state vector $|\Phi_5\rangle = \frac{1}{2\sqrt{2}} (|000\rangle + e^{-1\times\sqrt{-1}\times\frac{\pi}{4}} |001\rangle + e^{-1\times\sqrt{-1}\times\frac{\pi}{4}} |011\rangle + |010\rangle + |101\rangle + e^{-1\times\sqrt{-1}\times\frac{\pi}{4}} |100\rangle + e^{-1\times\sqrt{-1}\times\frac{\pi}{4}} |110\rangle + |111\rangle)$ as its input and completes one *T* gate for q[0] in the *sixth* time slot. If the value of q[0] is equal to one, then its phase is changed as $e^{1\times\sqrt{-1}\times\frac{\pi}{4}}$. Otherwise, its phase is not changed. Since there is no other quantum gate in the *sixth* time slot to act on quantum bits q[2] and q[1], their states are not changed. Hence, after one *T* gate for q[0] in the *sixth* time slot is completed by the statement "t q[0];" on line eleven of Listing 2.3, the following new state vector is obtained:

$|\Phi_6\rangle = \frac{1}{2\sqrt{2}} (|000\rangle + (e^{1\times\sqrt{-1}\times\frac{\pi}{4}} \times e^{-1\times\sqrt{-1}\times\frac{\pi}{4}}) |001\rangle + (e^{1\times\sqrt{-1}\times\frac{\pi}{4}} \times e^{-1\times\sqrt{-1}\times\frac{\pi}{4}}) |011\rangle + |010\rangle + e^{1\times\sqrt{-1}\times\frac{\pi}{4}} |101\rangle + e^{-1\times\sqrt{-1}\times\frac{\pi}{4}} |100\rangle + e^{-1\times\sqrt{-1}\times\frac{\pi}{4}} |110\rangle + e^{1\times\sqrt{-1}\times\frac{\pi}{4}} |111\rangle = \frac{1}{2\sqrt{2}} (|000\rangle + |001\rangle + |011\rangle + |010\rangle + e^{1\times\sqrt{-1}\times\frac{\pi}{4}} |101\rangle + e^{-1\times\sqrt{-1}\times\frac{\pi}{4}} |100\rangle + e^{-1\times\sqrt{-1}\times\frac{\pi}{4}} |110\rangle + e^{1\times\sqrt{-1}\times\frac{\pi}{4}} |111\rangle)$.

Next, the statement "cx q[1],q[0];" on line twelve of Listing 2.3 takes the new state vector $|\Phi_6\rangle = \frac{1}{2\sqrt{2}} (|000\rangle + |001\rangle + |011\rangle + |010\rangle + e^{1\times\sqrt{-1}\times\frac{\pi}{4}} |101\rangle + e^{-1\times\sqrt{-1}\times\frac{\pi}{4}} |100\rangle + e^{-1\times\sqrt{-1}\times\frac{\pi}{4}} |110\rangle + e^{1\times\sqrt{-1}\times\frac{\pi}{4}} |111\rangle)$ as its input and performs one ***CNOT*** gate for q[1] and q[0] in the *seventh* time slot. If the value of the control bit q[1] is equal to one, then the value of the target bit q[0] is flipped. Otherwise, it is not changed. There is no other quantum gate in the *seventh* time slot to act on quantum bit q[2], so its state is not changed. Therefore, after one ***CNOT*** gate for q[1] and q[0] in the *seventh* time slot is implemented by the statement "cx q[1],q[0];" on line twelve of Listing 2.3, the following new state vector is obtained:

$|\Phi_7\rangle = \frac{1}{2\sqrt{2}} (|000\rangle + |001\rangle + |010\rangle + |011\rangle + e^{1\times\sqrt{-1}\times\frac{\pi}{4}} |101\rangle + e^{-1\times\sqrt{-1}\times\frac{\pi}{4}} |100\rangle + e^{-1\times\sqrt{-1}\times\frac{\pi}{4}} |111\rangle + e^{1\times\sqrt{-1}\times\frac{\pi}{4}} |110\rangle)$.

Next, the statement “tdg q[0];” on line thirteen of Listing 2.3 takes the new state vector $|\Phi_7\rangle = \frac{1}{2\sqrt{2}}$ ($|000\rangle + |001\rangle + |010\rangle + |011\rangle + e^{1\times\sqrt{-1}\times\frac{\pi}{4}} |101\rangle + e^{-1\times\sqrt{-1}\times\frac{\pi}{4}} |100\rangle + e^{-1\times\sqrt{-1}\times\frac{\pi}{4}} |111\rangle + e^{1\times\sqrt{-1}\times\frac{\pi}{4}} |110\rangle$) as its input and finish one T^+ gate for q[0] in the *eighth* time slot. If the value of q[0] is equal to one, then its phase is changed as $e^{-1\times\sqrt{-1}\times\frac{\pi}{4}}$. Otherwise, its phase is not changed. Because there is no other quantum gate in the *eighth* time slot to act on quantum bits q[2] and q[1], their states are not changed. Hence, after one T^+ gate for q[0] in the *eighth* time slot is completed by the statement “tdg q[0];” on line thirteen of Listing 2.3, the following new state vector is obtained:

$$|\Phi_8\rangle = \frac{1}{2\sqrt{2}} (|000\rangle + e^{-1\times\sqrt{-1}\times\frac{\pi}{4}} |001\rangle + |010\rangle + e^{-1\times\sqrt{-1}\times\frac{\pi}{4}} |011\rangle + (e^{-1\times\sqrt{-1}\times\frac{\pi}{4}} \times e^{1\times\sqrt{-1}\times\frac{\pi}{4}}) |101\rangle + e^{-1\times\sqrt{-1}\times\frac{\pi}{4}} |100\rangle + (e^{-1\times\sqrt{-1}\times\frac{\pi}{4}} \times e^{-1\times\sqrt{-1}\times\frac{\pi}{4}}) |111\rangle + e^{1\times\sqrt{-1}\times\frac{\pi}{4}} |110\rangle).$$

Next, the statement “cx q[2],q[0];” on line fourteen of Listing 2.3 takes the new state vector $|\Phi_8\rangle = \frac{1}{2\sqrt{2}}$ ($|000\rangle + e^{-1\times\sqrt{-1}\times\frac{\pi}{4}} |001\rangle + |010\rangle + e^{-1\times\sqrt{-1}\times\frac{\pi}{4}} |011\rangle + (e^{-1\times\sqrt{-1}\times\frac{\pi}{4}} \times e^{1\times\sqrt{-1}\times\frac{\pi}{4}}) |101\rangle + e^{-1\times\sqrt{-1}\times\frac{\pi}{4}} |100\rangle + (e^{-1\times\sqrt{-1}\times\frac{\pi}{4}} \times e^{-1\times\sqrt{-1}\times\frac{\pi}{4}}) |111\rangle + e^{1\times\sqrt{-1}\times\frac{\pi}{4}} |110\rangle$) as its input and completes one ***CNOT*** gate for q[2] and q[0] in the *ninth* time slot. If the value of the control bit q[2] is equal to one, then the value of the target bit q[0] is flipped. Otherwise, it is not changed. Since there is no other quantum gate in the *ninth* time slot to act on quantum bit q[1], its state is not changed Thus, after one ***CNOT*** gate for q[2] and q[0] in the *ninth* time slot is completed by the statement “cx q[2],q[0];” on line fourteen of Listing 2.3, the following new state vector is obtained:

$$|\Phi_9\rangle = \frac{1}{2\sqrt{2}} (|000\rangle + e^{-1\times\sqrt{-1}\times\frac{\pi}{4}} |001\rangle + |010\rangle + e^{-1\times\sqrt{-1}\times\frac{\pi}{4}} |011\rangle + |100\rangle + e^{-1\times\sqrt{-1}\times\frac{\pi}{4}} |101\rangle + (e^{-1\times\sqrt{-1}\times\frac{\pi}{4}} \times e^{-1\times\sqrt{-1}\times\frac{\pi}{4}}) |110\rangle + e^{1\times\sqrt{-1}\times\frac{\pi}{4}} |111\rangle).$$

Next, the statement “t q[0];” on line fifteen of Listing 2.3 takes the new state vector $|\Phi_9\rangle = \frac{1}{2\sqrt{2}} (|000\rangle + e^{-1\times\sqrt{-1}\times\frac{\pi}{4}} |001\rangle + |010\rangle + e^{-1\times\sqrt{-1}\times\frac{\pi}{4}} |011\rangle + |100\rangle + e^{-1\times\sqrt{-1}\times\frac{\pi}{4}} |101\rangle + (e^{-1\times\sqrt{-1}\times\frac{\pi}{4}} \times e^{-1\times\sqrt{-1}\times\frac{\pi}{4}}) |110\rangle + e^{1\times\sqrt{-1}\times\frac{\pi}{4}} |111\rangle)$ as its input and performs one *T* gate for q[0] in the *tenth* time slot. If the value of q[0] is equal to one, then its phase is changed as $e^{1\times\sqrt{-1}\times\frac{\pi}{4}}$. Otherwise, its phase is not changed. There is no other quantum gate in the *tenth* time slot to act on quantum bit q[2], so its state is not changed. Hence, after one *T* gate for q[0] in the *tenth* time slot is completed by the statement “t q[0];” on line fifteen of Listing 2.3, the following new state vector is obtained:

$|\Phi_{10}\rangle = \frac{1}{2\sqrt{2}} (|000\rangle + (e^{1\times\sqrt{-1}\times\frac{\pi}{4}} \times e^{-1\times\sqrt{-1}\times\frac{\pi}{4}}) |001\rangle + |010\rangle + (e^{1\times\sqrt{-1}\times\frac{\pi}{4}} \times e^{-1\times\sqrt{-1}\times\frac{\pi}{4}}) |011\rangle + |100\rangle + (e^{1\times\sqrt{-1}\times\frac{\pi}{4}} \times e^{-1\times\sqrt{-1}\times\frac{\pi}{4}}) |101\rangle + (e^{-1\times\sqrt{-1}\times\frac{\pi}{4}} \times e^{-1\times\sqrt{-1}\times\frac{\pi}{4}}) |110\rangle + (e^{1\times\sqrt{-1}\times\frac{\pi}{4}} \times e^{1\times\sqrt{-1}\times\frac{\pi}{4}}) |111\rangle) = \frac{1}{2\sqrt{2}} (|000\rangle + |001\rangle + |010\rangle + |011\rangle + |100\rangle + |101\rangle + (e^{-1\times\sqrt{-1}\times\frac{\pi}{4}} \times e^{-1\times\sqrt{-1}\times\frac{\pi}{4}}) |110\rangle + (e^{1\times\sqrt{-1}\times\frac{\pi}{4}} \times e^{1\times\sqrt{-1}\times\frac{\pi}{4}}) |111\rangle)$.

Next, the statement “t q[1];” on line sixteen of Listing 2.3 takes the new state vector $|\Phi_{10}\rangle = \frac{1}{2\sqrt{2}} (|000\rangle + |001\rangle + |010\rangle + |011\rangle + |100\rangle + |101\rangle + (e^{-1\times\sqrt{-1}\times\frac{\pi}{4}} \times e^{-1\times\sqrt{-1}\times\frac{\pi}{4}}) |110\rangle + (e^{1\times\sqrt{-1}\times\frac{\pi}{4}} \times e^{1\times\sqrt{-1}\times\frac{\pi}{4}}) |111\rangle)$ as its input and completes one *T* gate for q[1] in the *tenth* time slot. If the value of q[1] is equal to one, then its phase is changed as $e^{1\times\sqrt{-1}\times\frac{\pi}{4}}$. Otherwise, its phase is not changed. Because there is no other quantum gate in the *tenth* time slot to act on quantum bit q[2], so its state is not changed. Therefore, after one *T* gate for q[1] in the *tenth* time slot is completed by the statement “t q[1];” on line sixteen of Listing 2.3, the following new state vector is obtained:

$|\Phi_{11}\rangle = \frac{1}{2\sqrt{2}} (|000\rangle + |001\rangle + e^{1\times\sqrt{-1}\times\frac{\pi}{4}} |010\rangle + e^{1\times\sqrt{-1}\times\frac{\pi}{4}} |011\rangle + |100\rangle + |101\rangle +$

$(e^{1\times\sqrt{-1}\times\frac{\pi}{4}} \times e^{-1\times\sqrt{-1}\times\frac{\pi}{4}} \times e^{-1\times\sqrt{-1}\times\frac{\pi}{4}})$ |110> + $(e^{1\times\sqrt{-1}\times\frac{\pi}{4}} \times e^{1\times\sqrt{-1}\times\frac{\pi}{4}} \times e^{1\times\sqrt{-1}\times\frac{\pi}{4}})$ |111>).

Next, the statement "h q[0];" on line seventeen of Listing 2.3 takes the new state vector $|\Phi_{11}\rangle = \frac{1}{2\sqrt{2}}$ (|000> + |001> + $e^{1\times\sqrt{-1}\times\frac{\pi}{4}}$ |010> + $e^{1\times\sqrt{-1}\times\frac{\pi}{4}}$ |011> + |100> + |101> + $(e^{1\times\sqrt{-1}\times\frac{\pi}{4}} \times e^{-1\times\sqrt{-1}\times\frac{\pi}{4}} \times e^{-1\times\sqrt{-1}\times\frac{\pi}{4}})$ |110> + $(e^{1\times\sqrt{-1}\times\frac{\pi}{4}} \times e^{1\times\sqrt{-1}\times\frac{\pi}{4}} \times e^{1\times\sqrt{-1}\times\frac{\pi}{4}})$ |111>) as its input and completes one Hadamard gate for q[0] in the *eleventh* time slot. If the value of q[0] is equal to one, then its state is changed as $\frac{1}{\sqrt{2}}$ (|0> – |1>). Otherwise, its state is changed as $\frac{1}{\sqrt{2}}$ (|0> + |1>). Hence, after one Hadamard gate for q[0] in the *eleventh* time slot is completed by the statement "h q[0];" on line seventeen of Listing 2.3, the following new state vector is obtained:

$|\Phi_{12}\rangle = \frac{1}{2\sqrt{2}}$ $(\frac{2}{\sqrt{2}}$ |000> + $(\frac{2}{\sqrt{2}} \times e^{1\times\sqrt{-1}\times\frac{\pi}{4}})$ |010> + $\frac{2}{\sqrt{2}}$ |100> + $(e^{-1\times\sqrt{-1}\times\frac{\pi}{4}} \times \frac{2}{\sqrt{2}})$ |111>) = $\frac{1}{2}$ (|000> + $e^{1\times\sqrt{-1}\times\frac{\pi}{4}}$ |010> + |100> + $e^{-1\times\sqrt{-1}\times\frac{\pi}{4}}$ |111>).

Next, the statement "cx q[2],q[1];" on line eighteen of Listing 2.3 takes the new state vector $|\Phi_{12}\rangle = \frac{1}{2}$ (|000> + $e^{1\times\sqrt{-1}\times\frac{\pi}{4}}$ |010> + |100> + $e^{-1\times\sqrt{-1}\times\frac{\pi}{4}}$ |111>) as its input and performs one ***CNOT*** gate for q[2] and q[1] in the *eleventh* time slot. If the value of the control bit q[2] is equal to one, then the value of the target bit q[1] is flipped. Otherwise, its value is not changed. Hence, after one ***CNOT*** gate for q[2] and q[1] in the *eleventh* time slot is completed by the statement "cx q[2],q[1];" on line eighteen of Listing 2.3, the following new state vector is obtained:

$$|\Phi_{13}\rangle = \frac{1}{2} (|000\rangle + e^{1\times\sqrt{-1}\times\frac{\pi}{4}} |010\rangle + |110\rangle + e^{-1\times\sqrt{-1}\times\frac{\pi}{4}} |101\rangle).$$

Next, the statement "tdg q[1];" on line nineteen of Listing 2.3 takes the new state vector $|\Phi_{13}\rangle = \frac{1}{2}$ (|000> + $e^{1\times\sqrt{-1}\times\frac{\pi}{4}}$ |010> + |110> + $e^{-1\times\sqrt{-1}\times\frac{\pi}{4}}$ |101>) as its input

and completes one T^+ gate for q[1] in the *twelfth* time slot. If the value of q[1] is equal to one, then its phase is changed as $e^{-1\times\sqrt{-1}\times\frac{\pi}{4}}$. Otherwise, its phase is not changed. There is no other quantum gate in the *twelfth* time slot to act on quantum bit q[0], so its state is not changed. Hence, after one T^+ gate for q[1] in the *twelfth* time slot is completed by the statement "tdg q[1];" on line nineteen of Listing 2.3, the following new state vector is obtained:

$|\Phi_{14}\rangle = \frac{1}{2}\ (|000\rangle + (e^{-1\times\sqrt{-1}\times\frac{\pi}{4}} \times e^{1\times\sqrt{-1}\times\frac{\pi}{4}})\,|010\rangle + e^{-1\times\sqrt{-1}\times\frac{\pi}{4}}\,|110\rangle + e^{-1\times\sqrt{-1}\times\frac{\pi}{4}}\,|101\rangle)$.

Next, the statement "t q[2];" on line twenty of Listing 2.3 takes the new state vector $|\Phi_{14}\rangle = \frac{1}{2}\ (|000\rangle + (e^{-1\times\sqrt{-1}\times\frac{\pi}{4}} \times e^{1\times\sqrt{-1}\times\frac{\pi}{4}})\,|010\rangle + e^{-1\times\sqrt{-1}\times\frac{\pi}{4}}\,|110\rangle + e^{-1\times\sqrt{-1}\times\frac{\pi}{4}}\,|101\rangle)$ as its input and completes one *T* gate for q[2] in the *twelfth* time slot. If the value of q[2] is equal to one, then its phase is changed as $e^{1\times\sqrt{-1}\times\frac{\pi}{4}}$. Otherwise, its phase is not changed. There is no other quantum gate in the *twelfth* time slot to act on quantum bit q[0], so its state is not changed. Hence, after one *T* gate for q[2] in the *twelfth* time slot is completed by the statement "t q[2];" on line twenty of Listing 2.3, the following new state vector is obtained:

$|\Phi_{15}\rangle = \frac{1}{2}\ (|000\rangle + |010\rangle + (e^{1\times\sqrt{-1}\times\frac{\pi}{4}} \times e^{-1\times\sqrt{-1}\times\frac{\pi}{4}})\,|110\rangle + (e^{1\times\sqrt{-1}\times\frac{\pi}{4}} \times e^{-1\times\sqrt{-1}\times\frac{\pi}{4}})\,|101\rangle) = \frac{1}{2}\ (|000\rangle + |010\rangle + |110\rangle + |101\rangle)$.

Next, the statement "cx q[2],q[1];" on line twenty-one of Listing 2.3 takes the new state vector $|\Phi_{15}\rangle = \frac{1}{2}\ (|000\rangle + |010\rangle + |110\rangle + |101\rangle)$ as its input and performs one ***CNOT*** gate for q[2] and q[1] in the *thirteenth* time slot. If the value of the control bit q[2] is equal to one, then the value of the target bit q[1] is flipped. Otherwise, its value is not changed. Because there is no other quantum gate in the *thirteenth* time slot to act on quantum bit q[0], its state is not changed. Thus, after one ***CNOT*** gate for q[2] and q[1] in the *thirteenth* time slot is completed by the statement "cx q[2],q[1];" on line twenty-one of Listing 2.3, the following new state vector is obtained:

$$|\Phi_{16}> = \frac{1}{2} (|000> + |010> + |100> + |111>).$$

Next, the three statements “measure q[0] -> c[0];”, “measure q[1] -> c[1];” and “measure q[2] -> c[2];” on line twenty-two through line twenty-four of Listing 2.3 is to measure the first quantum bit q[0], the second quantum bit q[1] and the third quantum bit q[2] and to record the measurement outcome by overwriting the first classical bit c[0], the second classical bit c[1] and the third classical bit c[2]. In the backend *ibmqx4* with five quantum bits in **IBM**’s quantum computers, we use the command “simulate” to execute the program in Listing 2.3. The measured result is shown in Figure 2.9. From

Figure 2.9: After the measurement to the program in Listing 2.3 is completed, we obtain the answer 00100 with the probability 0.260, the answer 00010 with the probability 0.250, the answer 00111 with the probability 0.250 or the answer 00000 with the probability 0.240.

Figure 2.9, we obtain the answer 00100 (c[4] = q[4] = |0>, c[3] = q[3] = |0>, c[2] = q[2] = |1>, c[1] = q[1] = |0> and c[0] = q[0] = |0>) with the probability 0.260. Because the value of the first control bit q[2] is equal to one and the value of the second control bit q[1] is equal to zero, the value of the target bit q[0] is not changed and is equal to zero with the probability 0.260.

Or we obtain the answer 00010 (c[4] = q[4] = |0>, c[3] = q[3] = |0>, c[2] = q[2] = |0>, c[1] = q[1] = |1> and c[0] = q[0] = |0>) with the probability 0.250. The value of the first control bit q[2] is equal to zero and the value of the second control bit q[1] is equal to one, so the value of the target bit q[0] is not changed and is equal to zero with the probability 0.250. Or we obtain the answer 00111 (c[4] = q[4] = |0>, c[3] = q[3] = |0>, c[2] = q[2] = |1>, c[1] = q[1] = |1> and c[0] = q[0] = |1>) with the probability 0.250. Since the value of the first control bit q[2] is equal to one and the value of the second control bit q[1] is also equal to one, the value of the target bit q[0] is flipped and is equal to one with the probability 0.250. Or we obtain the answer 00000 (c[4] = q[4] = |0>, c[3] = q[3] = |0>, c[2] = q[2] = |0>, c[1] = q[1] = |0> and c[0] = q[0] = |0>) with the probability 0.240. Because the value of the first control bit q[2] is equal to zero and the value of the second control bit q[1] is also equal to zero, the value of the target bit q[0]

is not changed and is equal to zero with the probability 0.240.

2.3 Introduction to AND Operation

The **AND** operation of a bit obtains two inputs of a bit and produces one single output of a bit. If the value of the first input is 1 (one) and the value of the second input is also 1 (one), then it generates a result (output) of 1 (one). Otherwise, the result is 0 (zero). A symbol "$\wedge$" is used to represent the **AND** operation. Therefore, the **AND** operation of a bit that has two inputs of a bit is the following four possible combinational results:

$$\begin{aligned} 0 \wedge 0 &= 0 \\ 0 \wedge 1 &= 0 \\ 1 \wedge 0 &= 0 \\ 1 \wedge 1 &= 1 \end{aligned} \tag{2.5}$$

The value of a Boolean variable (a bit) is only 0 (zero) or 1 (one). Thus, **AND** of two Boolean variables (two inputs of a bit) q[2] and q[1], written as q[2] $\wedge$ q[1] is equal to 1 (one) if and only if q[2] and q[1] are both 1 (one). Similarly, q[2] $\wedge$ q[1] is equal to 0 (zero) if and only if either q[2] or q[1], or both, are 0 (zero). A truth table is often used with logic operation to represent all possible combinations of inputs and the corresponding outputs. Hence, the rules in (2.5) for the **AND** operation of a bit that has two inputs of a bit and generates one output of a bit may also be expressed in the form of a truth table that is shown in Table 2.3.

Input		Output
q[2]	q[1]	q[2] $\wedge$ q[1]
0	0	0
0	1	0
1	0	0
1	1	1

Table 2.3: The truth table for the **AND** operation of a bit that has two inputs of a bit and generates one output of a bit.

2.3.1 Quantum Program of Implementing AND Operation

We use one ***CCNOT*** gate that has three quantum input bits and three quantum output

bits to implement **AND** operation of a *classical* bit that has two inputs of a classical bit and generates one output of a classical bit. We use the two control bits C_1 and C_2 of the ***CCNOT*** gate to encode two inputs q[2] and q[1] of a classical bit in **AND** operation of a *classical* bit and apply the target bit ***T*** of the ***CCNOT*** gate to store one output q[2] ∧ q[1] of a classical bit in **AND** operation of a *classical* bit. The rule of using one ***CCNOT*** gate to implement **AND** operation may also be expressed in the form of a truth table that is shown in Table 2.4. Its graph representation is shown in Figure 2.10. The initial state of the target bit in the ***CCNOT*** gate in Figure 2.10 is set to |0>.

Input			Output		
C_1	C_2	T	C_1	C_2	$T = q[2] \wedge q[1]$
0	0	0	0	0	0
0	1	0	0	1	0
1	0	0	1	0	0
1	1	0	1	1	1

Table 2.4: The truth table of using one ***CCNOT*** gate to implement **AND** operation.

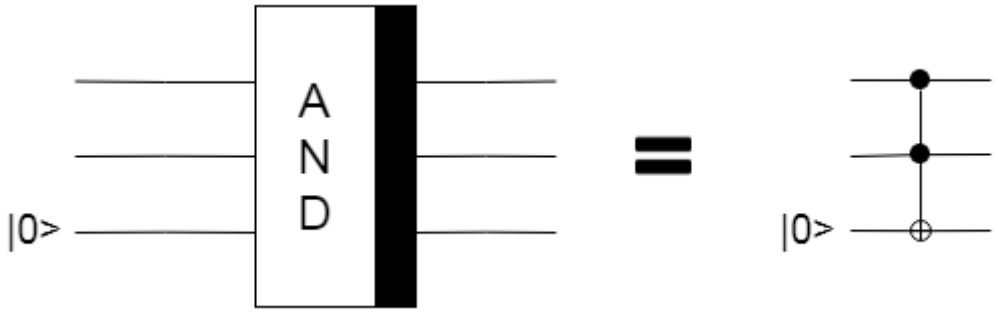

Figure 2.10: The quantum circuit of implementing **AND** operation of a *classical* bit.

In Listing 2.4, the program in the backend *ibmqx4* with five quantum bits in **IBM**'s quantum computer is the *fourth* example of the *second* chapter in which we describe how to write a quantum program to implement **AND** operation of a classical bit by means of using one ***CCNOT*** gate of three quantum bits. Figure 2.11 is the corresponding quantum circuit of the program in Listing 2.4. The statement "OPENQASM 2.0;" on line one of Listing 2.4 is to point out that the program is written with version 2.0 of Open QASM. Next, the statement "include "qelib1.inc";" on line two of Listing 2.4 is to continue parsing the file "qelib1.inc" as if the contents of the file were pasted at the location of the include statement, where the file "qelib1.inc" is **Quantum Experience (QE) Standard Header** and the path is specified relative to the

current working directory.

```
1.  OPENQASM 2.0;
2.  include "qelib1.inc";
3.  qreg q[5];
4.  creg c[5];
5.  h q[1];
6.  h q[2];
7.  h q[0];
8.  cx q[1],q[0];
9.  tdg q[0];
10. cx q[2],q[0];
11. t q[0];
12. cx q[1],q[0];
13. tdg q[0];
14. cx q[2],q[0];
15. t q[0];
16. t q[1];
17. h q[0];
18. cx q[2],q[1];
19. tdg q[1];
20. t q[2];
21. cx q[2],q[1];
22. measure q[0] -> c[0];
23. measure q[1] -> c[1];
24. measure q[2] -> c[2];
```

Listing 2.4: The program of using one ***CCNOT*** gate to implement **AND** operation.

Then, the statement "qreg q[5];" on line three of Listing 2.4 is to declare that in the program there are five quantum bits. In the left top of Figure 2.11, five quantum bits are subsequently q[0], q[1], q[2], q[3] and q[4]. The initial value of each quantum bit is set to |0>. We use three quantum bits q[2], q[1] and q[0] to respectively encode the first control bit, the second control bit and the target bit. This is to say that we apply quantum bits q[2] and q[1] to encode two inputs of a classical bit in **AND** operation of a *classical* bit and use quantum bit q[0] to store the result of **AND** operation of a *classical* bit. For the convenience of our explanation, $q[k]^0$ for $0 \leq k \leq 4$ is to represent the value of q[k] to be zero (0) and $q[k]^1$ for $0 \leq k \leq 4$ is to represent the value of q[k] to be one (1). Similarly, for the convenience of our explanation, an initial state vector of

implementing **AND** operation of a classical bit is as follows:

$$|A_0\rangle = |q[2]^0\rangle\ |q[1]^0\rangle\ |q[0]^0\rangle = |0\rangle\ |0\rangle\ |0\rangle = |000\rangle.$$

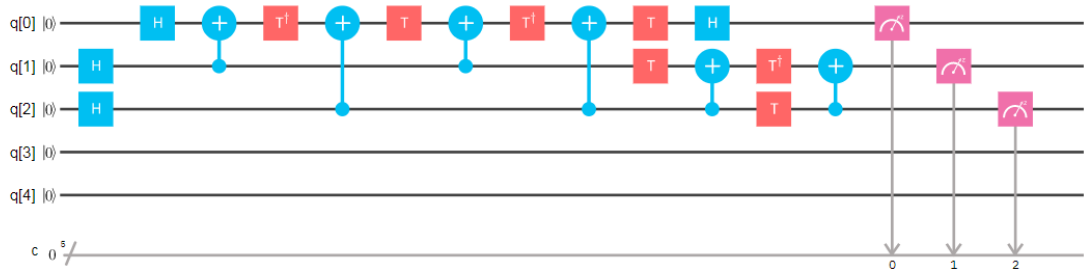

Figure 2.11: The corresponding quantum circuit of the program in Listing 2.4.

Next, the statement "creg c[5];" on line four of Listing 2.4 is to declare that there are five classical bits in the program. In the left bottom of Figure 2.11, five classical bits are subsequently c[0], c[1], c[2], c[3] and c[4]. The initial value of each classical bit is set to 0.

Next, the two statements "h q[1];" and "h q[2];"on line five and line six of Listing 2.4 implement two Hadamard gates of the *first* time slot of the quantum circuit in Figure 2.11. This implies that the statement "h q[1];" converts q[1] from one state |0> to another state $\frac{1}{\sqrt{2}}$ (|0> + |1>) (its superposition) and the statement "h q[2];" converts q[2] from one state |0> to another state $\frac{1}{\sqrt{2}}$ (|0> + |1>) (its superposition). In the *first* time slot of the quantum circuit in Figure 2.11 there is no quantum gate to act on the quantum bit q[0], so its state |0> is not changed. Thus, after two Hadamard gates in the *first* time slot of the quantum circuit in Figure 2.11 are implemented by means of using the two statements "h q[1];" and "h q[2];"on line five and line six of Listing 2.4, the following new state vector is obtained:

$$|A_1\rangle = \frac{1}{2}\ (|0\rangle\ |0\rangle\ |0\rangle + |0\rangle\ |1\rangle\ |0\rangle + |1\rangle\ |0\rangle\ |0\rangle + |1\rangle\ |1\rangle\ |0\rangle)$$

$$= \frac{1}{2}\ (|000\rangle + |010\rangle + |100\rangle + |110\rangle).$$

In the new state vector $|A_1\rangle$, four combinational states of quantum bits q[2] and q[1] with that the amplitude of each combinational state is $\frac{1}{2}$ encode all of the possible

inputs for **AND** operation of a classical bit. The initial state of quantum bit q[0] in four combinational states of quantum bits q[2] and q[1] is |0> and it stores the result for **AND** operation of a classical bit.

The next 12 time slots in the quantum circuit of Figure 2.11 implement **AND** operation of a classical bit by means of implementing one ***CCNOT*** gate. Each quantum gate from the *second* time slot through the *thirteenth* time slot in Figure 2.11 were implemented by the statements "h q[0];", "cx q[1],q[0];", "tdg q[0];", "cx q[2],q[0];", "t q[0];", "cx q[1],q[0];", "tdg q[0];", "cx q[2],q[0];", "t q[0];", "t q[1];", "h q[0];", "cx q[2],q[1];", "tdg q[1];", "t q[2];" and "cx q[2],q[1];" from line seven through line twenty-one in Listing 2.4. They take the new state vector $|A_1\rangle = \frac{1}{2}$ (|000> + |010> + |100> + |110>) as the input in the *second* time slot and complete **AND** operation of a classical bit. After they were implemented, the following new state vector is obtained:

$$|A_{16}\rangle = \frac{1}{2} (|000\rangle + |010\rangle + |100\rangle + |111\rangle).$$

Next, three measurements from the *fourteenth* time slot through the *sixteenth* time slot in Figure 2.11 were implemented by the three statements "measure q[0] -> c[0];", "measure q[1] -> c[1];" and "measure q[2] -> c[2];" on line twenty-two through line twenty-four of Listing 2.4 is to measure the first quantum bit q[0], the second quantum bit q[1] and the third quantum bit q[2] and to record the measurement outcome by overwriting the first classical bit c[0], the second classical bit c[1] and the third classical bit c[2]. In the backend *ibmqx4* with five quantum bits in **IBM**'s quantum computers, we use the command "simulate" to execute the program in Listing 2.4. The measured result is shown in Figure 2.12. From Figure 2.12, we obtain the answer 00010 (c[4] = q[4] = |0>, c[3] = q[3] = |0>, c[2] = q[2] = |0>, c[1] = q[1] = |1> and c[0] = q[0] = |0>) with the probability 0.290. Since the value of the first control bit q[2] is equal to zero and the value of the second control bit q[1] is equal to one, the value of the target bit q[0] is not changed and is equal to zero with the probability 0.290.

Figure 2.12: After the measurement to the program in Listing 2.4 is completed, we obtain the answer 00010 with the probability 0.290, the answer 00111 with the

probability 0.280, the answer 00100 with the probability 0.240 or the answer 00000 with the probability 0.190.

Or we obtain the answer 00111 (c[4] = q[4] = |0>, c[3] = q[3] = |0>, c[2] = q[2] = |1>, c[1] = q[1] = |1> and c[0] = q[0] = |1>) with the probability 0.280. The value of the first control bit q[2] is equal to one and the value of the second control bit q[1] is equal to one, so the value of the target bit q[0] is flipped and is equal to one with the probability 0.280. Or we obtain the answer 00100 (c[4] = q[4] = |0>, c[3] = q[3] = |0>, c[2] = q[2] = |1>, c[1] = q[1] = |0> and c[0] = q[0] = |0>) with the probability 0.240. Because the value of the first control bit q[2] is equal to one and the value of the second control bit q[1] is equal to zero, the value of the target bit q[0] is not changed and is equal to zero with the probability 0.240. Or we obtain the answer 00000 (c[4] = q[4] = |0>, c[3] = q[3] = |0>, c[2] = q[2] = |0>, c[1] = q[1] = |0> and c[0] = q[0] = |0>) with the probability 0.190. The value of the first control bit q[2] is equal to zero and the value of the second control bit q[1] is also equal to zero, so the value of the target bit q[0] is not changed and is equal to zero with the probability 0.190.

2.4 Introduction to NAND Operation

The **NAND** operation of a bit takes two inputs of a bit and generates one single output of a bit. If the value of the first input is 1 (one) and the value of the second input is also 1 (one), then it yields a result (output) of 0 (zero). Otherwise, the result (output) is 1 (one). A symbol "$\overline{\wedge}$" is applied to represent the **NAND** operation. Hence, the **NAND** operation of a bit that has two inputs of a bit is the following four possible combinational results:

$$\begin{aligned} \overline{0 \wedge 0} &= 1 \\ \overline{0 \wedge 1} &= 1 \\ \overline{1 \wedge 0} &= 1 \\ \overline{1 \wedge 1} &= 0 \end{aligned} \tag{2.6}$$

The value of a Boolean variable (a bit) is only 1 (one) or 0 (zero). Hence, **NAND** of two Boolean variables (two inputs of a bit) q[2] and q[1], written as $\overline{q[2] \wedge q[1]}$ is equal to 0 (zero) if and only if q[2] and q[1] are both 1 (one). Similarly, $\overline{q[2] \wedge q[1]}$ is equal to 1 (one) if and only if either q[2] or q[1], or both, are 0 (zero). A truth table is usually applied with logic operation to represent all possible combinations of inputs and the corresponding outputs. Therefore, the rules in (2.6) for the **NAND** operation of a bit that has two inputs of a bit and produces one single output of a bit may also be

expressed in the form of a truth table that is shown in Table 2.5.

Input		Output
q[2]	q[1]	$\overline{q[2] \wedge q[1]}$
0	0	1
0	1	1
1	0	1
1	1	0

Table 2.5: The truth table for the **NAND** operation of a bit that has two inputs of a bit and produces one output of a bit.

2.4.1 Quantum Program of Implementing NAND Operation

We apply one ***CCNOT*** gate that has three quantum input bits and three quantum output bits to implement **NAND** operation of a *classical* bit that has two inputs of a classical bit and produces one output of a classical bit. We make use of the two control bits $\boldsymbol{C}_1$ and $\boldsymbol{C}_2$ of the ***CCNOT*** gate to encode two inputs q[2] and q[1] of a classical bit in **NAND** operation of a *classical* bit and use the target bit $\boldsymbol{T}$ of the ***CCNOT*** gate to store one output $\overline{q[2] \wedge q[1]}$ of a classical bit in **NAND** operation of a *classical* bit. The rule of applying one ***CCNOT*** gate to implement **NAND** operation may also be expressed in the form of a truth table that is shown in Table 2.6. Its graph representation is shown in Figure 2.13. The initial state of the target bit in the ***CCNOT*** gate in Figure 2.13 is set to |1>.

Input			Output		
$\boldsymbol{C}_1$	$\boldsymbol{C}_2$	$\boldsymbol{T}$	$\boldsymbol{C}_1$	$\boldsymbol{C}_2$	$\boldsymbol{T} = \overline{q[2] \wedge q[1]}$
0	0	1	0	0	1
0	1	1	0	1	1
1	0	1	1	0	1
1	1	1	1	1	0

Table 2.6: The truth table of applying one ***CCNOT*** gate to implement **NAND** operation.

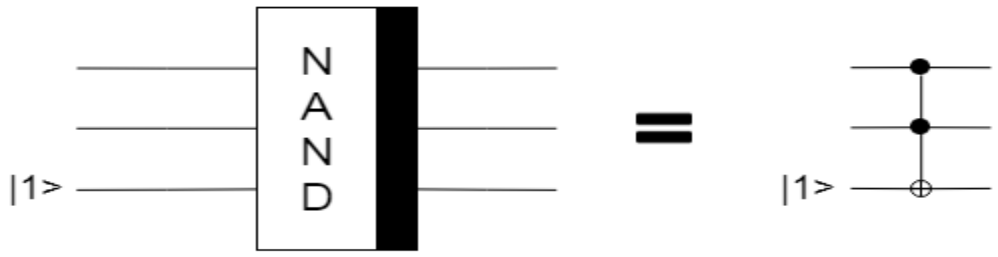

Figure 2.13: The quantum circuit of implementing **NAND** operation of a *classical* bit.

In Listing 2.5, the program in the backend *ibmqx4* with five quantum bits in **IBM**'s quantum computer is the *fifth* example of the *second* chapter in which we illustrate how to write a quantum program to implement **NAND** operation of a classical bit by means of applying one ***CCNOT*** gate of three quantum bits. Figure 2.14 is the corresponding quantum circuit of the program in Listing 2.5. The statement "OPENQASM 2.0;" on line one of Listing 2.5 is to indicate that the program is written with version 2.0 of Open QASM. Then, the statement "include "qelib1.inc";" on line two of Listing 2.5 is to continue parsing the file "qelib1.inc" as if the contents of the file were pasted at the location of the include statement, where the file "qelib1.inc" is **Quantum Experience (QE) Standard Header** and the path is specified relative to the current working directory.

```
1.  OPENQASM 2.0;
2.  include "qelib1.inc";
3.  qreg q[5];
4.  creg c[5];
5.  x q[0];
6.  h q[1];
7.  h q[2];
8.  h q[0];
9.  cx q[1],q[0];
10. tdg q[0];
11. cx q[2],q[0];
12. t q[0];
13. cx q[1],q[0];
14. tdg q[0];
15. cx q[2],q[0];
16. t q[0];
17. t q[1];
18. h q[0];
```

```
19. cx q[2],q[1];
20. tdg q[1];
21. t q[2];
22. cx q[2],q[1];
23. measure q[0] -> c[0];
24. measure q[1] -> c[1];
25. measure q[2] -> c[2];
```

Listing 2.5: The program of applying one ***CCNOT*** gate to implement **NAND** operation.

Next, the statement "qreg q[5];" on line three of Listing 2.5 is to declare that in the program there are five quantum bits. In the left top of Figure 2.14, five quantum bits are subsequently q[0], q[1], q[2], q[3] and q[4]. The initial value of each quantum bit is set to |0>. We make use of three quantum bits q[2], q[1] and q[0] to respectively encode the first control bit, the second control bit and the target bit. This is to say that we use quantum bits q[2] and q[1] to encode two inputs of a classical bit in **NAND** operation of a *classical* bit and apply quantum bit q[0] to store the result of **NAND** operation of a *classical* bit. For the convenience of our explanation, $q[k]^0$ for $0 \le k \le 4$ is to represent the value of q[k] to be zero (0) and $q[k]^1$ for $0 \le k \le 4$ is to represent the value of q[k] to be one (1). Similarly, for the convenience of our explanation, an initial state vector of implementing **NAND** operation of a classical bit is as follows:

$$|B_0\rangle = |q[2]^0\rangle\,|q[1]^0\rangle\,|q[0]^0\rangle = |0\rangle\,|0\rangle\,|0\rangle = |000\rangle.$$

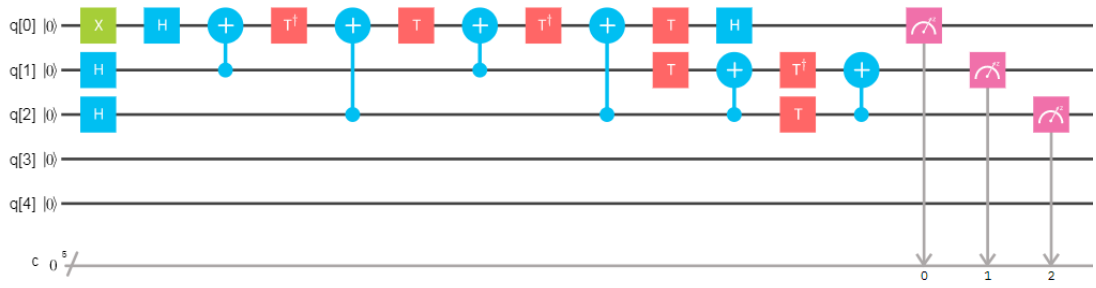

Figure 2.14: The corresponding quantum circuit of the program in Listing 2.5.

Next, the statement "creg c[5];" on line four of Listing 2.5 is to declare that there are five classical bits in the program. In the left bottom of Figure 2.14, five classical bits are subsequently c[0], c[1], c[2], c[3] and c[4]. The initial value of each classical bit is set to 0.

Next, the three statements "x q[0];", "h q[1];" and "h q[2];"on line five through line

seven of Listing 2.5 implement one ***NOT*** gate and two Hadamard gates of the *first* time slot of the quantum circuit in Figure 2.14. This is to say that the statement "x q[0];" converts q[0] from one state |0> to another state |1> (its negation), the statement "h q[1];" converts q[1] from one state |0> to another state $\frac{1}{\sqrt{2}}$ (|0> + |1>) (its superposition) and the statement "h q[2];" converts q[2] from one state |0> to another state $\frac{1}{\sqrt{2}}$ (|0> + |1>) (its superposition). Therefore, after one ***NOT*** gate and two Hadamard gates in the *first* time slot of the quantum circuit in Figure 2.14 are implemented by means of applying the three statements "x q[0];", "h q[1];" and "h q[2];"on line five through line seven of Listing 2.5, the following new state vector is obtained:

$$|B_1\rangle = \frac{1}{2}\ (|0\rangle\,|0\rangle\,|1\rangle + |0\rangle\,|1\rangle\,|1\rangle + |1\rangle\,|0\rangle\,|1\rangle + |1\rangle\,|1\rangle\,|1\rangle)$$

$$= \frac{1}{2}\ (|001\rangle + |011\rangle + |101\rangle + |111\rangle).$$

In the new state vector $|B_1\rangle$, four combinational states of quantum bits q[2] and q[1] with that the amplitude of each combinational state is $\frac{1}{2}$ encode all of the possible inputs for **NAND** operation of a classical bit. The initial state of quantum bit q[0] in four combinational states of quantum bits q[2] and q[1] is |1> and it stores the result for **NAND** operation of a classical bit.

The next 12 time slots in the quantum circuit of Figure 2.14 implement **NAND** operation of a classical bit by means of implementing one ***CCNOT*** gate. Each quantum gate from the *second* time slot through the *thirteenth* time slot in Figure 2.14 were implemented by the statements "h q[0];", "cx q[1],q[0];", "tdg q[0];", "cx q[2],q[0];", "t q[0];", "cx q[1],q[0];", "tdg q[0];", "cx q[2],q[0];", "t q[0];", "t q[1];", "h q[0];", "cx q[2],q[1];", "tdg q[1];", "t q[2];" and "cx q[2],q[1];" from line eight through line twenty-two in Listing 2.5. They take the new state vector $|B_1\rangle = \frac{1}{2}\ (|001\rangle + |011\rangle + |101\rangle + |111\rangle)$ as the input in the *second* time slot and complete **NAND** operation of a classical bit. After they were implemented, the following new state vector is obtained:

$$|B_{16}\rangle = \frac{1}{2}\ (|001\rangle + |011\rangle + |101\rangle + |110\rangle).$$

Then, three measurements from the *fourteenth* time slot through the *sixteenth* time slot in Figure 2.14 were implemented by the three statements “measure q[0] -> c[0];”, “measure q[1] -> c[1];” and “measure q[2] -> c[2];” on line twenty-three through line twenty-five of Listing 2.5 is to measure the first quantum bit q[0], the second quantum bit q[1] and the third quantum bit q[2] and to record the measurement outcome by overwriting the first classical bit c[0], the second classical bit c[1] and the third classical bit c[2]. In the backend *ibmqx4* with five quantum bits in **IBM**’s quantum computers, we make use of the command “simulate” to execute the program in Listing 2.5. The measured result is shown in Figure 2.15. From Figure 2.15, we obtain the answer 00011 (c[4] = q[4] = |0>, c[3] = q[3] = |0>, c[2] = q[2] = |0>, c[1] = q[1] = |1> and c[0] = q[0] = |1>) with the probability 0.340. Because the value of the first control bit q[2] is equal to 0 (zero) and the value of the second control bit q[1] is equal to 1 (one), the value of the target bit q[0] is not changed and is equal to 1 (one) with the probability 0.340.

Figure 2.15: After the measurement to the program in Listing 2.5 is completed, we obtain the answer 00011 with the probability 0.340, the answer 00001 with the probability 0.300, the answer 00101 with the probability 0.200 or the answer 00110 with the probability 0.160.

Or we obtain the answer 00001 (c[4] = q[4] = |0>, c[3] = q[3] = |0>, c[2] = q[2] = |0>, c[1] = q[1] = |0> and c[0] = q[0] = |1>) with the probability 0.300. Since the value of the first control bit q[2] is equal to 0 (zero) and the value of the second control bit q[1] is equal to 0 (zero), the value of the target bit q[0] is not changed and is equal to 1 (one) with the probability 0.300. Or we obtain the answer 00101 (c[4] = q[4] = |0>, c[3] = q[3] = |0>, c[2] = q[2] = |1>, c[1] = q[1] = |0> and c[0] = q[0] = |1>) with the probability 0.200. The value of the first control bit q[2] is equal to 1 (one) and the value of the second control bit q[1] is equal to 0 (zero), so the value of the target bit q[0] is not changed and is equal to 1 (one) with the probability 0.200. Or we obtain the answer 00110 (c[4] = q[4] = |0>, c[3] = q[3] = |0>, c[2] = q[2] = |1>, c[1] = q[1] = |1> and c[0] = q[0] = |0>) with the probability 0.160. Because the value of the first control bit q[2] is equal to 1 (one) and the value of the second control bit q[1] is also equal to 1 (one), the value of the target bit q[0] is flipped and is equal to 0 (zero) with the probability 0.160.

2.5 Introduction to OR Operation

The **OR** operation of a bit acquires two inputs of a bit and yields one single output of a bit. If the value of the first input is 1 (one) or the value of the second input is also 1 (one) or their values are both 1 (one), then it produces a result (output) of 1 (one). Otherwise, the result (output) is 0 (zero). A symbol "$\vee$" is used to represent the **OR** operation. Thus, the **OR** operation of a bit that takes two inputs of a bit is the following four possible combinational results:

$$\begin{aligned} 0 \vee 0 &= 0 \\ 0 \vee 1 &= 1 \\ 1 \vee 0 &= 1 \\ 1 \vee 1 &= 1 \end{aligned} \tag{2.7}$$

The value of a Boolean variable (a bit) is only 1 (one) or 0 (zero). Therefore, **OR** of two Boolean variables (two inputs of a bit) q[2] and q[1], written as q[2] $\vee$ q[1] is equal to 1 (one) if and only if the value of q[2] is 1 (one) or the value of q[1] is 1 (one) or their values are both 1 (one). Similarly, q[2] $\vee$ q[1] is equal to 0 (zero) if and only if the value of q[2] and the value of q[1] are both 0 (zero). A truth table is often used with logic operation to represent all possible combinations of inputs and the corresponding outputs. Hence, the rules in (2.7) for the **OR** operation of a bit that obtains two inputs of a bit and generates one single output of a bit may also be expressed in the form of a truth table that is shown in Table 2.7.

Input		Output
q[2]	q[1]	q[2] $\vee$ q[1]
0	0	0
0	1	1
1	0	1
1	1	1

Table 2.7: The truth table for the **OR** operation of a bit that takes two inputs of a bit and generates one single output of a bit.

2.5.1 Quantum Program of Implementing OR Operation

We make use of one ***CCNOT*** gate that has three quantum input bits and three quantum output bits to implement **OR** operation of a *classical* bit that obtains two

inputs of a classical bit and yields one output of a classical bit. We use the two control bits C_1 and C_2 of the ***CCNOT*** gate to encode two inputs q[2] and q[1] of a classical bit in **OR** operation of a *classical* bit and apply the target bit ***T*** of the ***CCNOT*** gate to store one output $q[2] \vee q[1] = \overline{\overline{q[2] \vee q[1]}} = \overline{\overline{q[2]} \wedge \overline{q[1]}}$ of a classical bit in **OR** operation of a *classical* bit. The rule of using one ***CCNOT*** gate to implement **OR** operation may also be expressed in the form of a truth table that is shown in Table 2.8. Its graph representation is shown in Figure 2.16. In Figure 2.16, the first control bit (the top first wire) C_1 and the second control bit (the second wire) C_2 of the ***CCNOT*** gate respectively encode the first input q[2] and the second input q[1]of a classical bit in **OR** operation of a classical bit. In Figure 2.16, the target bit (the bottom wire) T of the ***CCNOT*** gate is to store one output $q[2] \vee q[1] = \overline{\overline{q[2]} \wedge \overline{q[1]}}$ in **OR** operation of a classical bit.

Input			Output		
C_1	C_2	T	C_1	C_2	$T = q[2] \vee q[1] = \overline{\overline{q[2]} \wedge \overline{q[1]}}$
0	0	1	0	0	0
0	1	1	0	1	1
1	0	1	1	0	1
1	1	1	1	1	1

Table 2.8: The truth table of applying one ***CCNOT*** gate to implement **OR** operation.

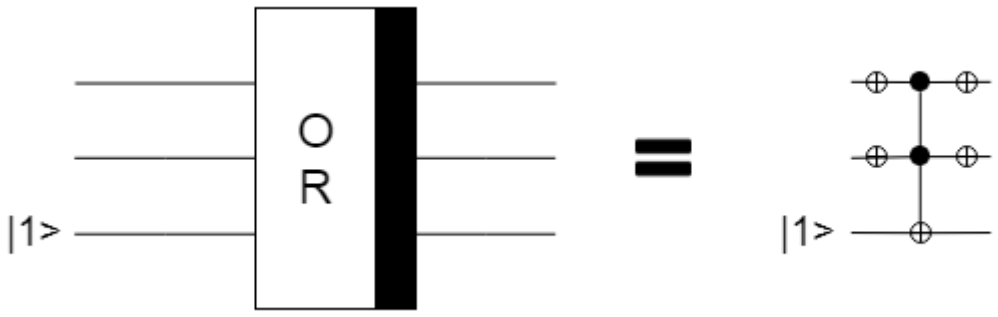

Figure 2.16: The quantum circuit of implementing **OR** operation of a *classical* bit.

The initial state of the target bit T in the ***CCNOT*** gate in Figure 2.16 is set to |1>. Implementing **OR** operation of a *classical* bit that takes two inputs q[2] and q[1] of a classical bit and produces one output $\overline{\overline{q[2]} \wedge \overline{q[1]}}$ of a classical bit is equivalent to

implement **NAND** operation of a classical bit that takes two inputs $\overline{q[2]}$ and $\overline{q[1]}$ of a classical bit and yields one output $\overline{\overline{q[2]} \wedge \overline{q[1]}}$. Therefore, in Figure 2.16, we use two ***NOT*** gates to operate the two control bits C_1 and C_2 of the ***CCNOT*** gate that encode two inputs q[2] and q[1] of a classical bit and to generate their negations $\overline{q[2]}$ and $\overline{q[1]}$. Next, in Figure 2.16, we apply one ***CCNOT*** gate to take their negations $\overline{q[2]}$ and $\overline{q[1]}$ as the input and to complete **NAND** operation of a classical bit. Because from Table 2.8 two inputs q[2] and q[1] of a classical bit in **OR** operation of a *classical* bit that is encoded by the two control bits C_1 and C_2 of the ***CCNOT*** gate in Figure 2.16 are not changed, we again make use of two ***NOT*** gates to operate the two control bits C_1 and C_2 of the ***CCNOT*** gate in Figure 2.16 and to generate the result $\overline{\overline{q[2]}}$ = q[2] and $\overline{\overline{q[1]}}$ = q[1]. This is to say that using ***NOT*** gate twice to the first control bit C_1 and the second control bit C_2 of the ***CCNOT*** gate in Figure 2.16 does nothing to them.

In Listing 2.6, the program in the backend *ibmqx4* with five quantum bits in **IBM**'s quantum computer is the *sixth* example of the *second* chapter in which we introduce how to write a quantum program to implement **OR** operation of a classical bit by means of using one ***CCNOT*** gate of three quantum bits and four ***NOT*** gates of one quantum bits. Figure 2.17 is the corresponding quantum circuit of the program in Listing 2.6. The statement "OPENQASM 2.0;" on line one of Listing 2.6 is to point out that the program is written with version 2.0 of Open QASM. Next, the statement "include "qelib1.inc";" on line two of Listing 2.6 is to continue parsing the file "qelib1.inc" as if the contents of the file were pasted at the location of the include statement, where the file "qelib1.inc" is **Quantum Experience (QE) Standard Header** and the path is specified relative to the current working directory.

```
1.  OPENQASM 2.0;
2.  include "qelib1.inc";
3.  qreg q[5];
4.  creg c[5];
5.  x q[0];
6.  h q[1];
7.  h q[2];
8.  x q[1];
9.  x q[2];
10. h q[0];
11. cx q[1],q[0];
12. tdg q[0];
```

```
13. cx q[2],q[0];
14. t q[0];
15. cx q[1],q[0];
16. tdg q[0];
17. cx q[2],q[0];
18. t q[0];
19. t q[1];
20. h q[0];
21. cx q[2],q[1];
22. tdg q[1];
23. t q[2];
24. cx q[2],q[1];
25. x q[1];
26. x q[2];
27. measure q[0] -> c[0];
28. measure q[1] -> c[1];
29. measure q[2] -> c[2];
```

Listing 2.6: The program of using one ***CCNOT*** gate and four ***NOT*** gates to implement **OR** operation.

Then, the statement "qreg q[5];" on line three of Listing 2.6 is to declare that in the program there are five quantum bits. In the left top of Figure 2.17, five quantum bits are subsequently q[0], q[1], q[2], q[3] and q[4]. The initial value of each quantum bit is set to |0>. We use three quantum bits q[2], q[1] and q[0] to respectively encode the first control bit, the second control bit and the target bit. This indicates that we make use of quantum bits q[2] and q[1] to encode two inputs of a classical bit in **OR** operation of a *classical* bit and apply quantum bit q[0] to store the result of **OR** operation of a *classical* bit. For the convenience of our explanation, $q[k]^0$ for $0 \le k \le 4$ is to represent the value of q[k] to be zero (0) and $q[k]^1$ for $0 \le k \le 4$ is to represent the value of q[k] to be one (1). Similarly, for the convenience of our explanation, an initial state vector of implementing **OR** operation of a classical bit is as follows:

$$|C_0\rangle = |q[2]^0\rangle\ |q[1]^0\rangle\ |q[0]^0\rangle = |0\rangle\ |0\rangle\ |0\rangle = |000\rangle.$$

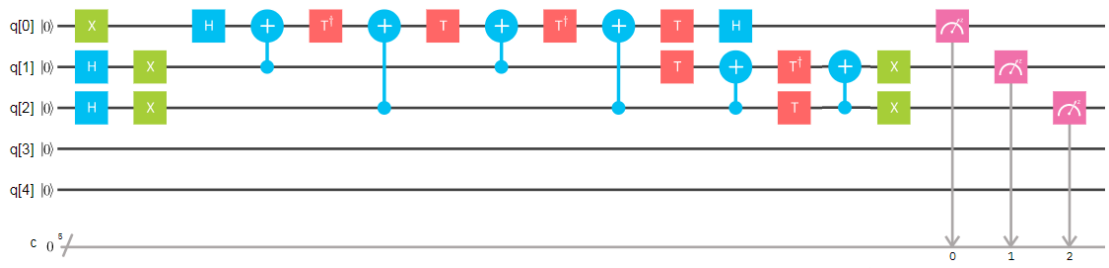

Figure 2.17: The corresponding quantum circuit of the program in Listing 2.6.

Next, the statement "creg c[5];" on line four of Listing 2.6 is to declare that there are five classical bits in the program. In the left bottom of Figure 2.17, five classical bits are subsequently c[0], c[1], c[2], c[3] and c[4]. The initial value of each classical bit is set to 0.

Next, the three statements "x q[0];", "h q[1];" and "h q[2];"on line five through line seven of Listing 2.6 implement one ***NOT*** gate and two Hadamard gates of the *first* time slot of the quantum circuit in Figure 2.17. This implies that the statement "x q[0];" converts q[0] from one state |0> to another state |1> (its negation), the statement "h q[1];" converts q[1] from one state |0> to another state $\frac{1}{\sqrt{2}}$ (|0> + |1>) (its superposition) and the statement "h q[2];" converts q[2] from one state |0> to another state $\frac{1}{\sqrt{2}}$ (|0> + |1>) (its superposition). Hence, after one ***NOT*** gate and two Hadamard gates in the *first* time slot of the quantum circuit in Figure 2.17 are implemented by means of using the three statements "x q[0];", "h q[1];" and "h q[2];"on line five through line seven of Listing 2.6, the following new state vector is obtained:

$$|C_1\rangle = \frac{1}{2}\,(|0\rangle\,|0\rangle\,|1\rangle + |0\rangle\,|1\rangle\,|1\rangle + |1\rangle\,|0\rangle\,|1\rangle + |1\rangle\,|1\rangle\,|1\rangle)$$

$$= \frac{1}{2}\,(|001\rangle + |011\rangle + |101\rangle + |111\rangle).$$

In the new state vector $|C_1\rangle$, four combinational states of quantum bits q[2] and q[1] with that the amplitude of each combinational state is $\frac{1}{2}$ encode all of the possible inputs for **OR** operation of a classical bit. The initial state of quantum bit q[0] in four combinational states of quantum bits q[2] and q[1] is |1> and it stores the result for **OR** operation of a classical bit.

Then, the two statements “x q[1];” and “x q[2];” on line eight through line nine of Listing 2.6 implement two ***NOT*** gates of the *second* time slot of the quantum circuit in Figure 2.17. They take the new state vector $|C_1\rangle = \frac{1}{2}$ (|001> + |011> + |101> + |111>) as the input in the *second* time slot of Figure 2.17. This is to say that in the new state vector $|C_1\rangle$ the state (|0> + |1>) of q[2] is converted into the state (|1> + |0>) and the state (|0> + |1>) of q[1] is converted into the state (|1> + |0>). Because there is no gate to act on q[0], its state is not changed. Therefore, after two ***NOT*** gates in the *second* time slot of the quantum circuit in Figure 2.17 are implemented by means of applying the two statements “x q[1];” and “x q[2];” on line eight through line nine of Listing 2.6, the following new state vector is obtained:

$$|C_2\rangle = \frac{1}{2}\ (|1\rangle\,|1\rangle\,|1\rangle + |1\rangle\,|0\rangle\,|1\rangle + |0\rangle\,|1\rangle\,|1\rangle + |0\rangle\,|0\rangle\,|1\rangle)$$

$$= \frac{1}{2}\ (|111\rangle + |101\rangle + |011\rangle + |001\rangle).$$

The next 12 time slots in the quantum circuit of Figure 2.17 implement **OR** operation ($q[2] \vee q[1] = \overline{\overline{q[2]} \wedge \overline{q[1]}}$) of a classical bit that is equivalent to implement **NAND** operation of a classical bit with two inputs $\overline{q[2]}$ and $\overline{q[1]}$ by means of implementing one ***CCNOT*** gate. Each quantum gate from the *third* time slot through the *fourteenth* time slot in Figure 2.17 was implemented by the statements “h q[0];”, “cx q[1],q[0];”, “tdg q[0];”, “cx q[2],q[0];”, “t q[0];”, “cx q[1],q[0];”, “tdg q[0];”, “cx q[2],q[0];”, “t q[0];”, “t q[1];”, “h q[0];”, “cx q[2],q[1];”, “tdg q[1];”, “t q[2];” and “cx q[2],q[1];” from line ten through line twenty-four in Listing 2.6. They take the new state vector $|C_2\rangle = \frac{1}{2}$ (|111> + |101> + |011> + |001>) as the input in the *third* time slot and complete **OR** operation ($q[2] \vee q[1] = \overline{\overline{q[2]} \wedge \overline{q[1]}}$) of a classical bit. After they were implemented, the following new state vector is obtained:

$$|C_{17}\rangle = \frac{1}{2}\ (|110\rangle + |101\rangle + |011\rangle + |001\rangle).$$

Next, the two statements “x q[1];” and “x q[2];” on line twenty-five through line twenty-six of Listing 2.6 implement two ***NOT*** gates of the *fifteenth* time slot of the quantum circuit in Figure 2.17. They take the new state vector $|C_{17}\rangle = \frac{1}{2}$ (|110> + |101>

+ |011> + |001>) as the input in the *fifteenth* time slot of Figure 2.17. This is to say that in the new state vector $|C_{17}>$ the state (|110>) is converted into the state (|000>), the state (|101>) is converted into the state (|011>), the state (|011>) is converted into the state (|101>) and the state (|001>) is converted into the state (|111>). Because there is no gate to act on q[0], its state is not changed. Thus, after two ***NOT*** gates in the *fifteenth* time slot of the quantum circuit in Figure 2.17 were implemented by means of applying the two statements "x q[1];" and "x q[2];" on line twenty-five through line twenty-six of Listing 2.6, the following new state vector is obtained:

$$|C_{18}> = \frac{1}{2} (|000> + |011> + |101> + |111>).$$

Next, three measurements from the *sixteenth* time slot through the *eighteenth* time slot in Figure 2.17 were implemented by the three statements "measure q[0] -> c[0];", "measure q[1] -> c[1];" and "measure q[2] -> c[2];" on line twenty-seven through line twenty-nine of Listing 2.6 is to measure the first quantum bit q[0], the second quantum bit q[1] and the third quantum bit q[2] and to record the measurement outcome by overwriting the first classical bit c[0], the second classical bit c[1] and the third classical bit c[2]. In the backend *ibmqx4* with five quantum bits in **IBM**'s quantum computers, we use the command "simulate" to execute the program in Listing 2.6. The measured result is shown in Figure 2.18. From Figure 2.18, we obtain the answer 00101 (c[4] = q[4] = |0>, c[3] = q[3] = |0>, c[2] = q[2] = |1>, c[1] = q[1] = |0> and c[0] = q[0] = |1>) with the probability 0.290. Since in **OR** operation of a classical bit the value of the first input (the first control bit) q[2] is equal to 1 (one) and the value of the second input (the second control bit) q[1] is equal to 0 (zero), the value of the output (the target bit) q[0] is equal to 1 (one) with the probability 0.290.

Figure 2.18: After the measurement to the program in Listing 2.6 is completed, we obtain the answer 00101 with the probability 0.290, the answer 00011 with the probability 0.270, the answer 00000 with the probability 0.250 or the answer 00111 with the probability 0.190.

Or we obtain the answer 00011 (c[4] = q[4] = |0>, c[3] = q[3] = |0>, c[2] = q[2] = |0>, c[1] = q[1] = |1> and c[0] = q[0] = |1>) with the probability 0.270. Because in **OR**

operation of a classical bit the value of the first input (the first control bit) q[2] is equal to 0 (zero) and the value of the second input (the second control bit) q[1] is equal to 1 (one), the value of the output (the target bit) q[0] is equal to 1 (one) with the probability 0.270. Or we obtain the answer 00000 (c[4] = q[4] = |0>, c[3] = q[3] = |0>, c[2] = q[2] = |0>, c[1] = q[1] = |0> and c[0] = q[0] = |0>) with the probability 0.250. In **OR** operation of a classical bit, the value of the first input (the first control bit) q[2] is equal to 0 (zero) and the value of the second input (the second control bit) q[1] is also equal to 0 (zero), so the value of the output (the target bit) q[0] is equal to 0 (zero) with the probability 0.250. Or we obtain the answer 00111 (c[4] = q[4] = |0>, c[3] = q[3] = |0>, c[2] = q[2] = |1>, c[1] = q[1] = |1> and c[0] = q[0] = |1>) with the probability 0.190. Because in **OR** operation of a classical bit the value of the first input (the first control bit) q[2] is equal to 1 (one) and the value of the second input (the second control bit) q[1] is also equal to 1 (one), the value of the output (the target bit) is equal to 1 (one) with the probability 0.190.

2.6 Introduction of NOR Operation

The **NOR** operation of a bit obtains two inputs of a bit and produces one single output of a bit. If the value of the first input is 0 (zero) and the value of the second input is also 0 (zero), then it produces a result (output) of 1 (one). However, if either the value of the first input or the value of the second input, or both of them, are 1, then it yields a result (output) of 0 (zero). A symbol "$\overline{\vee}$" is applied to represent the **NOR** operation. Therefore, the **NOR** operation of a bit that acquires two inputs of a bit is the following four possible combinational results:

$$\begin{aligned} \overline{0 \vee 0} &= 1 \\ \overline{0 \vee 1} &= 0 \\ \overline{1 \vee 0} &= 0 \\ \overline{1 \vee 1} &= 0 \end{aligned} \tag{2.8}$$

The value of a Boolean variable (a bit) is only 0 (zero) or 1 (one). Hence, **NOR** operation of two Boolean variables (two inputs of a bit) q[2] and q[1], written as $\overline{q[2] \vee q[1]}$ is equal to 1 (one) if and only if the value of q[2] is 0 (zero) and the value of q[1] is 0 (zero). Similarly, $\overline{q[2] \vee q[1]}$ is equal to 0 (zero) if and only if either the value of q[2] is 1 (one) or the value of q[1] is 1 (one) or both of them are 1 (one). A truth table is usually applied with logic operation to represent all possible combinations of inputs and the corresponding outputs. Thus, the rules in (2.8) for the **NOR** operation of a bit that takes two inputs of a bit and generates one single output of a bit may also

be expressed in the form of a truth table that is shown in Table 2.9.

Input		Output
q[2]	q[1]	$\overline{q[2] \vee q[1]}$
0	0	1
0	1	0
1	0	0
1	1	0

Table 2.9: The truth table for the **NOR** operation of a bit that takes two inputs of a bit and generates one single output of a bit.

2.6.1 Quantum Program of Implementing NOR Operation

We use one ***CCNOT*** gate that has three quantum input bits and three quantum output bits to implement **NOR** operation of a *classical* bit that acquires two inputs of a classical bit and generates one output of a classical bit. We apply the two control bits $\boldsymbol{C}_1$ and $\boldsymbol{C}_2$ of the ***CCNOT*** gate to encode two inputs q[2] and q[1] of a classical bit in **NOR** operation of a *classical* bit and make use of the target bit $\boldsymbol{T}$ of the ***CCNOT*** gate to store one output $\overline{q[2] \vee q[1]} = \overline{q[2]} \wedge \overline{q[1]}$ of a classical bit in **NOR** operation of a *classical* bit. The rule of applying one ***CCNOT*** gate to complete **NOR** operation may also be expressed in the form of a truth table that is shown in Table 2.10. Its graph representation is shown in Figure 2.19. In Figure 2.19, the first control bit (the top first wire) C_1 and the second control bit (the second wire) C_2 of the ***CCNOT*** gate respectively encode the first input q[2] and the second input q[1]of a classical bit in **NOR** operation of a classical bit. In Figure 2.19, the target bit (the bottom wire) T of the ***CCNOT*** gate is to store one output $\overline{q[2] \vee q[1]} = \overline{q[2]} \wedge \overline{q[1]}$ in **NOR** operation of a classical bit.

Input			Output		
$\boldsymbol{C}_1$	$\boldsymbol{C}_2$	$\boldsymbol{T}$	$\boldsymbol{C}_1$	$\boldsymbol{C}_2$	$\boldsymbol{T} = \overline{q[2] \vee q[1]} = \overline{q[2]} \wedge \overline{q[1]}$
0	0	0	0	0	1
0	1	0	0	1	0
1	0	0	1	0	0
1	1	0	1	1	0

Table 2.10: The truth table of using one ***CCNOT*** gate to implement **NOR** operation.

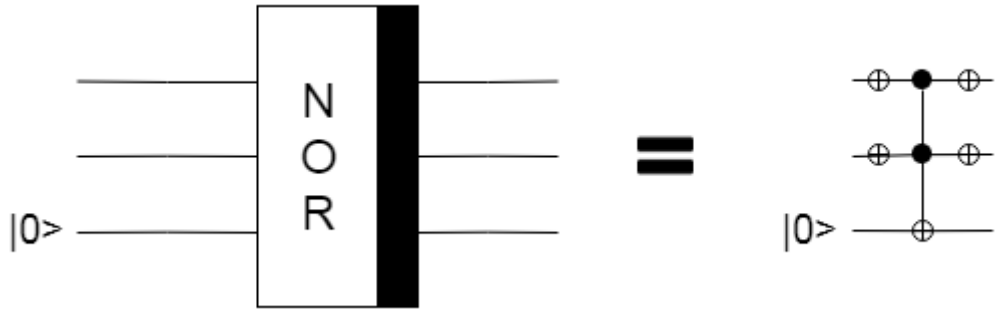

Figure 2.19: The quantum circuit of implementing **NOR** operation of a *classical* bit.

The initial state of the target bit *T* in the ***CCNOT*** gate in Figure 2.19 is set to |0>. Implementing **NOR** operation of a *classical* bit that takes two inputs q[2] and q[1] of a classical bit and generates one output $\overline{q[2] \vee q[1]} = \overline{q[2]} \wedge \overline{q[1]}$ of a classical bit is equivalent to complete **AND** operation of a classical bit that acquires two inputs $\overline{q[2]}$ and $\overline{q[1]}$ of a classical bit and produces one output $\overline{q[2]} \wedge \overline{q[1]}$. Hence, in Figure 2.19, we apply two ***NOT*** gates to operate the two control bits $\boldsymbol{C}_1$ and $\boldsymbol{C}_2$ of the ***CCNOT*** gate that encode two inputs q[2] and q[1] of a classical bit and to yield their negations $\overline{q[2]}$ and $\overline{q[1]}$. Next, in Figure 2.19, we make use of one ***CCNOT*** gate to obtain their negations $\overline{q[2]}$ and $\overline{q[1]}$ as the input and to implement **AND** operation of a classical bit. Since from Table 2.10 two inputs q[2] and q[1] of a classical bit in **NOR** operation of a *classical* bit that is encoded by the two control bits $\boldsymbol{C}_1$ and $\boldsymbol{C}_2$ of the ***CCNOT*** gate in Figure 2.19 are not changed, we again use two ***NOT*** gates to operate the two control bits $\boldsymbol{C}_1$ and $\boldsymbol{C}_2$ of the ***CCNOT*** gate in Figure 2.19 and to produce the result $\overline{\overline{q[2]}}$ = q[2] and $\overline{\overline{q[1]}}$ = q[1]. This implies that using ***NOT*** gate twice to the first control bit $\boldsymbol{C}_1$ and the second control bit $\boldsymbol{C}_2$ of the ***CCNOT*** gate in Figure 2.19 does nothing to them.

In Listing 2.7, the program in the backend *ibmqx4* with five quantum bits in **IBM**'s quantum computer is the *seventh* example of the *second* chapter in which we describe how to write a quantum program to implement **NOR** operation of a classical bit by means of applying one ***CCNOT*** gate of three quantum bits and four ***NOT*** gates of one quantum bits. Figure 2.20 is the corresponding quantum circuit of the program in Listing 2.7. The statement "OPENQASM 2.0;" on line one of Listing 2.7 is to indicate that the program is written with version 2.0 of Open QASM. Next, the statement "include "qelib1.inc";" on line two of Listing 2.7 is to continue parsing the file "qelib1.inc" as if the contents of the file were pasted at the location of the include statement, where the file "qelib1.inc" is **Quantum Experience (QE) Standard Header** and the path is specified relative to the current working directory.

```
1.  OPENQASM 2.0;
2.  include "qelib1.inc";
3.  qreg q[5];
4.  creg c[5];
5.  h q[1];
6.  h q[2];
7.  x q[1];
8.  x q[2];
9.  h q[0];
10. cx q[1],q[0];
11. tdg q[0];
12. cx q[2],q[0];
13. t q[0];
14. cx q[1],q[0];
15. tdg q[0];
16. cx q[2],q[0];
17. t q[0];
18. t q[1];
19. h q[0];
20. cx q[2],q[1];
21. tdg q[1];
22. t q[2];
23. cx q[2],q[1];
24. x q[1];
25. x q[2];
26. measure q[0] -> c[0];
27. measure q[1] -> c[1];
28. measure q[2] -> c[2];
```

Listing 2.7: The program of applying one ***CCNOT*** gate and four ***NOT*** gates to implement **NOR** operation.

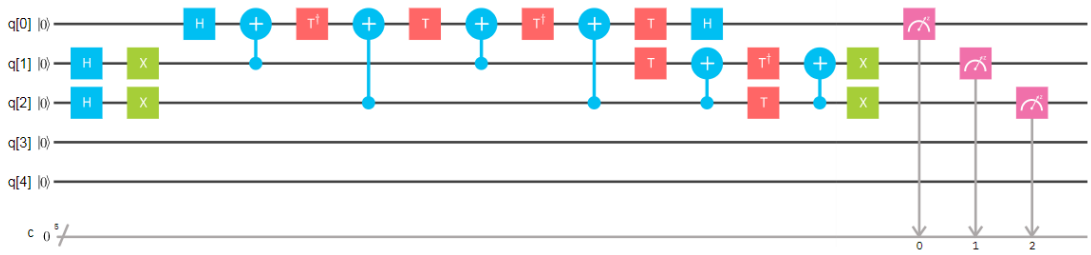

Figure 2.20: The corresponding quantum circuit of the program in Listing 2.7.

Next, the statement “qreg q[5];” on line three of Listing 2.7 is to declare that in the program there are five quantum bits. In the left top of Figure 2.20, five quantum bits are subsequently q[0], q[1], q[2], q[3] and q[4]. The initial value of each quantum bit is set to |0>. We use three quantum bits q[2], q[1] and q[0] to respectively encode the first control bit, the second control bit and the target bit. This is to say that we use quantum bits q[2] and q[1] to encode two inputs of a classical bit in **NOR** operation of a *classical* bit and make use of quantum bit q[0] to store the result of **NOR** operation of a *classical* bit. For the convenience of our explanation, $q[k]^0$ for $0 \le k \le 4$ is to represent the value of q[k] to be zero (0) and $q[k]^1$ for $0 \le k \le 4$ is to represent the value of q[k] to be one (1). Similarly, for the convenience of our explanation, an initial state vector of implementing **NOR** operation of a classical bit is as follows:

$$|D_0\rangle = |q[2]^0\rangle\, |q[1]^0\rangle\, |q[0]^0\rangle = |0\rangle\, |0\rangle\, |0\rangle = |000\rangle.$$

Next, the statement “creg c[5];” on line four of Listing 2.7 is to declare that there are five classical bits in the program. In the left bottom of Figure 2.20, five classical bits are subsequently c[0], c[1], c[2], c[3] and c[4]. The initial value of each classical bit is set to 0.

Next, the two statements “h q[1];” and “h q[2];”on line five through line six of Listing 2.7 implement two Hadamard gates of the *first* time slot of the quantum circuit in Figure 2.20. This is to say that the statement “h q[1];” converts q[1] from one state |0> to another state $\frac{1}{\sqrt{2}}$ (|0> + |1>) (its superposition) and the statement “h q[2];” converts q[2] from one state |0> to another state $\frac{1}{\sqrt{2}}$ (|0> + |1>) (its superposition). Because there is no gate to act on quantum bit q[0], its state is not changed. Therefore, after two Hadamard gates in the *first* time slot of the quantum circuit in Figure 2.20 are implemented by means of applying the two statements “h q[1];” and “h q[2];”on line five through line six of Listing 2.7, the following new state vector is obtained:

$$|D_1\rangle = \frac{1}{2}\,(|0\rangle\,|0\rangle\,|0\rangle + |0\rangle\,|1\rangle\,|0\rangle + |1\rangle\,|0\rangle\,|0\rangle + |1\rangle\,|1\rangle\,|0\rangle)$$

$$= \frac{1}{2}\,(|000\rangle + |010\rangle + |100\rangle + |110\rangle).$$

In the new state vector $|D_1\rangle$, four combinational states of quantum bits q[2] and q[1]

with that the amplitude of each combinational state is $\frac{1}{2}$ encode all of the possible inputs for **NOR** operation of a classical bit. The initial state of quantum bit q[0] in four combinational states of quantum bits q[2] and q[1] is |0> and it stores the result for **NOR** operation of a classical bit.

Next, the two statements "x q[1];" and "x q[2];" on line seven through line eight of Listing 2.7 complete two ***NOT*** gates of the *second* time slot of the quantum circuit in Figure 2.20. They take the new state vector $|D_1\rangle = \frac{1}{2}$ (|000> + |010> + |100> + |110>) as the input in the *second* time slot of Figure 2.20. This indicates that in the new state vector $|D_1\rangle$ the state (|0> + |1>) of q[2] is converted into the state (|1> + |0>) and the state (|0> + |1>) of q[1] is converted into the state (|1> + |0>). There is no gate to act on quantum bit q[0], so its state is not changed. Thus, after two ***NOT*** gates in the *second* time slot of the quantum circuit in Figure 2.20 are implemented by means of using the two statements "x q[1];" and "x q[2];" on line seven through line eight of Listing 2.7, the following new state vector is obtained:

$$|D_2\rangle = \frac{1}{2} (|1\rangle |1\rangle |0\rangle + |1\rangle |0\rangle |0\rangle + |0\rangle |1\rangle |0\rangle + |0\rangle |0\rangle |0\rangle)$$

$$= \frac{1}{2} (|110\rangle + |100\rangle + |010\rangle + |000\rangle).$$

The next 12 time slots in the quantum circuit of Figure 2.20 implement **NOR** operation ($\overline{q[2] \vee q[1]} = \overline{q[2]} \wedge \overline{q[1]}$) of a classical bit that is equivalent to complete **AND** operation of a classical bit with two inputs $\overline{q[2]}$ and $\overline{q[1]}$ by means of implementing one ***CCNOT*** gate. Each quantum gate from the *third* time slot through the *fourteenth* time slot in Figure 2.20 was implemented by the statements "h q[0];", "cx q[1],q[0];", "tdg q[0];", "cx q[2],q[0];", "t q[0];", "cx q[1],q[0];", "tdg q[0];", "cx q[2],q[0];", "t q[0];", "t q[1];", "h q[0];", "cx q[2],q[1];", "tdg q[1];", "t q[2];" and "cx q[2],q[1];" from line nine through line twenty-three in Listing 2.7. They take the new state vector $|D_2\rangle = \frac{1}{2}$ (|110> + |100> + |010> + |000>) as the input in the *third* time slot and complete N**OR** operation ($\overline{q[2] \vee q[1]} = \overline{q[2]} \wedge \overline{q[1]}$) of a classical bit. After they were implemented, the following new state vector is obtained:

$$|D_{17}\rangle = \frac{1}{2} (|111\rangle + |100\rangle + |010\rangle + |000\rangle).$$

Next, the two statements “x q[1];” and “x q[2];” on line twenty-four through line twenty-five of Listing 2.7 implement two ***NOT*** gates of the *fifteenth* time slot of the quantum circuit in Figure 2.20. They take the new state vector $|D_{17}\rangle = \frac{1}{2}\ (|111\rangle + |100\rangle + |010\rangle + |000\rangle)$ as the input in the *fifteenth* time slot of Figure 2.20. This indicates that in the new state vector $|D_{17}\rangle$ the state ($|111\rangle$) is converted into the state ($|001\rangle$), the state ($|100\rangle$) is converted into the state ($|010\rangle$), the state ($|010\rangle$) is converted into the state ($|100\rangle$) and the state ($|000\rangle$) is converted into the state ($|110\rangle$). Because there is no gate to act on quantum bit q[0], its state is not changed. Therefore, after two ***NOT*** gates in the *fifteenth* time slot of the quantum circuit in Figure 2.20 were implemented by means of using the two statements “x q[1];” and “x q[2];” on line twenty-four through line twenty-five of Listing 2.7, the following new state vector is obtained:

$$|D_{18}\rangle = \frac{1}{2}\ (|001\rangle + |010\rangle + |100\rangle + |110\rangle).$$

Next, three measurements from the *sixteenth* time slot through the *eighteenth* time slot in Figure 2.20 were implemented by the three statements “measure q[0] -> c[0];”, “measure q[1] -> c[1];” and “measure q[2] -> c[2];” on line twenty-six through line twenty-eight of Listing 2.7 is to measure the first quantum bit q[0], the second quantum bit q[1] and the third quantum bit q[2] and to record the measurement outcome by overwriting the first classical bit c[0], the second classical bit c[1] and the third classical bit c[2]. In the backend *ibmqx4* with five quantum bits in **IBM**’s quantum computers, we apply the command “simulate” to execute the program in Listing 2.7. The measured result is shown in Figure 2.21. From Figure 2.21, we obtain the answer 00001 (c[4] = q[4] = |0>, c[3] = q[3] = |0>, c[2] = q[2] = |0>, c[1] = q[1] = |0> and c[0] = q[0] = |1>) with the probability 0.270. Since in **NOR** operation of a classical bit the value of the first input (the first control bit) q[2] is equal to 0 (zero) and the value of the second input (the second control bit) q[1] is equal to 0 (zero), the value of the output (the target bit) q[0] is equal to 1 (one) with the probability 0.270.

Figure 2.21: After the measurement to the program in Listing 2.7 is completed, we obtain the answer 00001 with the probability 0.270, the answer 00010 with the

probability 0.270, the answer 00100 with the probability 0.260 or the answer 00110 with the probability 0.200.

Or we obtain the answer 00010 (c[4] = q[4] = |0>, c[3] = q[3] = |0>, c[2] = q[2] = |0>, c[1] = q[1] = |1> and c[0] = q[0] = |0>) with the probability 0.270. Because in **NOR** operation of a classical bit the value of the first input (the first control bit) q[2] is equal to 0 (zero) and the value of the second input (the second control bit) q[1] is equal to 1 (one), the value of the output (the target bit) q[0] is equal to 0 (zero) with the probability 0.270. Or we obtain the answer 00100 (c[4] = q[4] = |0>, c[3] = q[3] = |0>, c[2] = q[2] = |1>, c[1] = q[1] = |0> and c[0] = q[0] = |0>) with the probability 0.260. In **NOR** operation of a classical bit, the value of the first input (the first control bit) q[2] is equal to 1 (one) and the value of the second input (the second control bit) q[1] is equal to 0 (zero), so the value of the output (the target bit) q[0] is equal to 0 (zero) with the probability 0.260. Or we obtain the answer 00110 (c[4] = q[4] = |0>, c[3] = q[3] = |0>, c[2] = q[2] = |1>, c[1] = q[1] = |1> and c[0] = q[0] = |0>) with the probability 0.200. Because in **NOR** operation of a classical bit the value of the first input (the first control bit) q[2] is equal to 1 (one) and the value of the second input (the second control bit) q[1] is also equal to 1 (one), the value of the output (the target bit) q[0] is equal to 0 (zero) with the probability 0.200.

2.7 Introduction for Exclusive-OR Operation

The **Exclusive-OR** (**XOR**) operation of a bit takes two inputs of a bit and generates one single output of a bit. If the value of the first input is the same as that of the second input, then it produces a result (output) of 0 (zero). However, if the value of the first input and the value of the second input are both different, then it generates an output of 1 (one). A symbol "⊕" is used to represent the **XOR** operation. Hence, the **XOR** operation of a bit that gets two inputs of a bit is the following four possible combinational results:

$$\begin{aligned} 0 \oplus 0 &= 0 \\ 0 \oplus 1 &= 1 \\ 1 \oplus 0 &= 1 \\ 1 \oplus 1 &= 0 \end{aligned} \tag{2.9}$$

The value of a Boolean variable (a bit) is only 1 (one) or 0 (zero). Therefore, **XOR** operation of two Boolean variables (two inputs of a bit) q[2] and q[1], written as q[2] ⊕ q[1] is equal to 1 (one) if and only if the value of q[2] and the value of q[1] are different. Similarly, q[2] ⊕ q[1] is equal to 0 (zero) if and only if the value of q[2] and the value of q[1] are the same. A truth table is often used with logic operation to

represent all possible combinations of inputs and the corresponding outputs. Therefore, the rules in (2.9) for the **XOR** operation of a bit that obtains two inputs of a bit and yields one single output of a bit may also be expressed in the form of a truth table that is shown in Table 2.11.

Input		Output
q[2]	q[1]	q[2] ⊕ q[1]
0	0	0
0	1	1
1	0	1
1	1	0

Table 2.11: The truth table for the **XOR** operation of a bit that acquires two inputs of a bit and produces one single output of a bit.

2.7.1 Quantum Program of Implementing XOR Operation

We apply one ***CNOT*** gate that has two quantum input bits and two quantum output bits to implement **XOR** operation of a *classical* bit that takes two inputs of a classical bit and produces one output of a classical bit. We make use of the control bit $\boldsymbol{C}_1$ and the target bit ***T*** of the ***CNOT*** gate to encode two inputs q[2] and q[1] of a classical bit in **XOR** operation of a *classical* bit and also use the target bit ***T*** of the ***CNOT*** gate to store one output q[2] ⊕ q[1] of a classical bit in **XOR** operation of a *classical* bit. The rule of using one ***CNOT*** gate to implement **XOR** operation may also be expressed in the form of a truth table that is shown in Table 2.12. Its graph representation is shown in Figure 2.22.

Input		Output	
$\boldsymbol{C}_1$	***T***	$\boldsymbol{C}_1$	***T*** = q[2] ⊕ q[1]
0	0	0	0
0	1	0	1
1	0	1	1
1	1	1	0

Table 2.12: The truth table of using one ***CNOT*** gate to implement **XOR** operation.

In Figure 2.22, the first control bit (the top wire) C_1 and the target bit (the bottom wire) ***T*** of the ***CNOT*** gate respectively encode the first input q[2] and the second input q[1]of a classical bit in **XOR** operation of a classical bit in Table 2.11. In Figure 2.22,

the target bit (the bottom wire) *T* of the ***CNOT*** gate also stores one output q[2] ⊕ q[1] of a classical bit in **XOR** operation of a classical bit in Table 2.11.

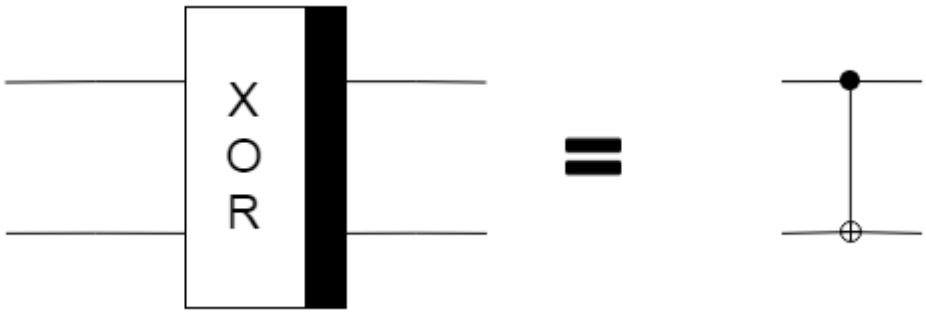

Figure 2.22: The quantum circuit of implementing **XOR** operation of a *classical* bit.

Implementing **XOR** operation of a *classical* bit that acquires two inputs q[2] and q[1] of a classical bit and yields one output q[2] ⊕ q[1] of a classical bit is equivalent to implement one ***CNOT*** gate that its control bit and its target bit encode two inputs q[2] and q[1] of a classical bit and its target bit also stores one output q[2] ⊕ q[1]. Therefore, in Figure 2.22, we use one ***CNOT*** gate to implement **XOR** operation of a classical bit.

In Listing 2.8, the program in the backend *ibmqx4* with five quantum bits in **IBM**'s quantum computer is the *eighth* example of the *second* chapter in which we illustrate how to write a quantum program to complete **XOR** operation of a classical bit by means of using one ***CNOT*** gate of two quantum bits. Figure 2.23 is the corresponding quantum circuit of the program in Listing 2.8. The statement "OPENQASM 2.0;" on line one of Listing 2.8 is to point out that the program is written with version 2.0 of Open QASM. Then, the statement "include "qelib1.inc";" on line two of Listing 2.8 is to continue parsing the file "qelib1.inc" as if the contents of the file were pasted at the location of the include statement, where the file "qelib1.inc" is **Quantum Experience (QE) Standard Header** and the path is specified relative to the current working directory.

```
1. OPENQASM 2.0;
2. include "qelib1.inc";
3. qreg q[5];
4. creg c[5];
5. h q[1];
6. h q[2];
7. cx q[2],q[1];
8. measure q[1] -> c[1];
```

```
9.  measure q[2] -> c[2];
```

Listing 2.8: The program of using one ***CNOT*** gate to implement **XOR** operation.

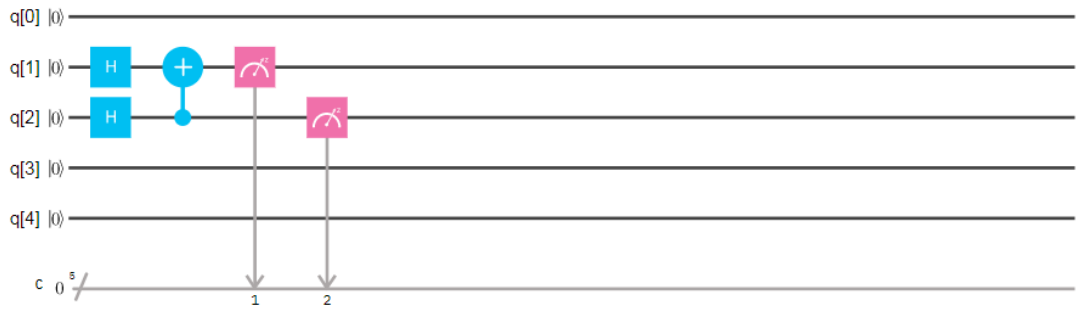

Figure 2.23: The corresponding quantum circuit of the program in Listing 2.8.

Next, the statement "qreg q[5];" on line three of Listing 2.8 is to declare that in the program there are five quantum bits. In the left top of Figure 2.23, five quantum bits are subsequently q[0], q[1], q[2], q[3] and q[4]. The initial value of each quantum bit is set to |0>. We make use of two quantum bits q[2] and q[1] to respectively encode the control bit and the target bit of one ***CNOT*** gate. This implies that we apply quantum bits q[2] and q[1] to encode two inputs of a classical bit in **XOR** operation of a *classical* bit and use quantum bit q[1] to store the result of **XOR** operation of a *classical* bit. For the convenience of our explanation, $q[k]^0$ for $0 \le k \le 4$ is to represent the value of q[k] to be zero (0) and $q[k]^1$ for $0 \le k \le 4$ is to represent the value of q[k] to be one (1). Similarly, for the convenience of our explanation, an initial state vector of implementing **XOR** operation of a classical bit is as follows:

$$|E_0\rangle = |q[2]^0\rangle\,|q[1]^0\rangle = |0\rangle\,|0\rangle = |00\rangle.$$

Next, the statement "creg c[5];" on line four of Listing 2.8 is to declare that there are five classical bits in the program. In the left bottom of Figure 2.23, five classical bits are respectively c[0], c[1], c[2], c[3] and c[4]. The initial value of each classical bit is set to 0.

Then, the two statements "h q[1];" and "h q[2];"on line five through line six of Listing 2.8 implement two Hadamard gates of the *first* time slot of the quantum circuit in Figure 2.23. This indicates that the statement "h q[1];" converts q[1] from one state |0> to another state $\frac{1}{\sqrt{2}}$ (|0> + |1>) (its superposition) and the statement "h q[2];"

converts q[2] from one state |0> to another state $\frac{1}{\sqrt{2}}$ (|0> + |1>) (its superposition). Hence, after two Hadamard gates in the *first* time slot of the quantum circuit in Figure 2.23 are implemented by means of using the two statements “h q[1];” and “h q[2];”on line five through line six of Listing 2.8, the following new state vector is obtained:

$$|E_1\rangle = \frac{1}{2} (|0\rangle |0\rangle + |0\rangle |1\rangle + |1\rangle |0\rangle + |1\rangle |1\rangle)$$

$$= \frac{1}{2} (|00\rangle + |01\rangle + |10\rangle + |11\rangle).$$

In the new state vector $|E_1\rangle$, four combinational states of quantum bits q[2] and q[1] with that the amplitude of each combinational state is $\frac{1}{2}$ encode all of the possible inputs in **XOR** operation of a classical bit in Table 2.11. Quantum bit q[1] stores the result for **XOR** operation of a classical bit in Table 2.11.

Next, the statement “cx q[2],q[1];” on line seven of Listing 2.8 complete one ***CNOT*** gates of the *second* time slot of the quantum circuit in Figure 2.23. They take the new state vector $|E_1\rangle = \frac{1}{2} (|00\rangle + |01\rangle + |10\rangle + |11\rangle)$ as the input in the *second* time slot of Figure 2.23. This is to say that in the new state vector $|E_1\rangle$ the state (|00>) of quantum bits q[2] and q[1] is not changed and the state (|01>) of quantum bits q[2] and q[1] is also not changed because the value of the control bit q[2] is equal to 0 (zero). However, the state (|10>) of quantum bits q[2] and q[1] is converted into the state (|11>) and the state (|11>) of quantum bits q[2] and q[1] is converted into the state (|10>) because the value of the control bit q[2] is equal to 1 (one) and the target bit q[1] is flipped. Therefore, after one ***CNOT*** gate in the *second* time slot of the quantum circuit in Figure 2.23 is implemented by means of applying the statement “cx q[2],q[1];” on line seven of Listing 2.8, the following new state vector is obtained:

$$|E_2\rangle = \frac{1}{2} (|0\rangle |0\rangle + |0\rangle |1\rangle + |1\rangle |1\rangle + |1\rangle |0\rangle)$$

$$= \frac{1}{2} (|00\rangle + |01\rangle + |11\rangle + |10\rangle).$$

Next, two measurements from the *third* time slot through the *fourth* time slot in Figure 2.23 were implemented by the two statements “measure q[1] -> c[1];” and

“measure q[2] -> c[2];” on line eight through line nine of Listing 2.8. They are to measure the second quantum bit q[1] and the third quantum bit q[2] and to record the measurement outcome by overwriting the second classical bit c[1] and the third classical bit c[2]. In the backend *ibmqx4* with five quantum bits in **IBM**’s quantum computers, we use the command “simulate” to run the program in Listing 2.8. The measured result is shown in Figure 2.24. From Figure 2.24, we get the answer 00010 (c[4] = q[4] = |0>, c[3] = q[3] = |0>, c[2] = q[2] = |0>, c[1] = q[1] = |1> and c[0] = q[0] = |0>) with the probability 0.260. Because in **XOR** operation of a classical bit the value of the first input (the control bit) q[2] is equal to 0 (zero) and the value of the second input (the target bit) q[1] is equal to 1 (one), the value of the output (the target bit) q[1] is equal to 1 (one) with the probability 0.260.

Figure 2.24: After the measurement to the program in Listing 2.8 is completed, we obtain the answer 00010 with the probability 0.260, the answer 00110 with the probability 0.260, the answer 00100 with the probability 0.250 or the answer 00000 with the probability 0.230.

Or we obtain the answer 00110 (c[4] = q[4] = |0>, c[3] = q[3] = |0>, c[2] = q[2] = |1>, c[1] = q[1] = |1> and c[0] = q[0] = |0>) with the probability 0.260. Since in **XOR** operation of a classical bit the value of the first input (the control bit) q[2] is equal to 1 (one) and the value of the second input (the target bit) q[1] is equal to 0 (zero), the value of the output (the target bit) q[1] is equal to 1 (one) with the probability 0.260. Or we acquire the answer 00100 (c[4] = q[4] = |0>, c[3] = q[3] = |0>, c[2] = q[2] = |1>, c[1] = q[1] = |0> and c[0] = q[0] = |0>) with the probability 0.250. Since in **XOR** operation of a classical bit the value of the first input (the control bit) q[2] is equal to 1 (one) and the value of the second input (the target bit) q[1] is equal to 1 (one), the value of the output (the target bit) q[1] is equal to 0 (zero) with the probability 0.250. Or we get the answer 00000 (c[4] = q[4] = |0>, c[3] = q[3] = |0>, c[2] = q[2] = |0>, c[1] = q[1] = |0> and c[0] = q[0] = |0>) with the probability 0.230. In **XOR** operation of a classical bit the value of the first input (the control bit) q[2] is equal to 0 (zero) and the value of the second input (the target bit) q[1] is also equal to 0 (zero), so the value of the output (the target bit) q[1] is equal to 0 (zero) with the probability 0.230.

2.8 Introduction of Exclusive-NOR Operation

The one's complement of the **Exclusive-OR** (**XOR**) operation of a bit that acquires two inputs of a bit and yields one single output of a bit is known as the **Exclusive-NOR** (**XNOR**) operation of a bit. The **Exclusive-NOR** (**XNOR**) operation of a bit obtains two inputs of a bit and produces one single output of a bit. If the value of the first input is the same as that of the second input, then it generates a result (output) of 1 (one). However, if the value of the first input and the value of the second input are both different, then it produces an output of 0 (zero). A symbol "$\overline{\oplus}$" is applied to represent the **XNOR** operation of a bit. Therefore, the **XNOR** operation of a bit that takes two inputs of a bit is the following four possible combinational results:

$$\begin{aligned} \overline{0 \oplus 0} &= 1 \\ \overline{0 \oplus 1} &= 0 \\ \overline{1 \oplus 0} &= 0 \\ \overline{1 \oplus 1} &= 1 \end{aligned} \tag{2.10}$$

The value of a Boolean variable (a bit) is just 0 (zero) or 1 (one). Hence, **XNOR** operation of two Boolean variables (two inputs of a bit) q[2] and q[1], written as $\overline{q[2] \oplus q[1]}$ is equal to 1 (one) if and only if the value of q[2] and the value of q[1] are the same. Similarly, $\overline{q[2] \oplus q[1]}$ is equal to 0 (zero) if and only if the value of q[2] and the value of q[1] are different. A truth table is usually applied with logic operation to represent all possible combinations of inputs and the corresponding outputs. Hence, the rules in (2.10) for the **XNOR** operation of a bit that gets two inputs of a bit and produces one single output of a bit may also be expressed in the form of a truth table that is shown in Table 2.13.

Input		Output
q[2]	q[1]	$\overline{q[2] \oplus q[1]}$
0	0	1
0	1	0
1	0	0
1	1	1

Table 2.13: The truth table for the **XNOR** operation of a bit that acquires two inputs of a bit and generates one single output of a bit.

2.8.1 Quantum Program of Implementing XNOR Operation

We make use of one ***CNOT*** gate and one ***NOT*** gate to implement **XNOR** operation of a *classical* bit that acquires two inputs of a classical bit and generates one output of a classical bit. We use the control bit $\boldsymbol{C}_1$ and the target bit ***T*** of the ***CNOT*** gate to encode two inputs q[2] and q[1] of a classical bit in **XNOR** operation of a *classical* bit and also use the target bit ***T*** of the ***CNOT*** gate to store one output $\overline{\mathrm{q}[2] \oplus \mathrm{q}[1]}$ of a classical bit in **XNOR** operation of a *classical* bit. The rule of applying one ***CNOT*** gate and one ***NOT*** gate to complete **XNOR** operation may also be expressed in the form of a truth table that is shown in Table 2.14. Its graph representation is shown in Figure 2.25.

Input		Output	
$\boldsymbol{C}_1$	$\boldsymbol{T}$	$\boldsymbol{C}_1$	$\boldsymbol{T} = \overline{\mathrm{q}[2] \oplus \mathrm{q}[1]}$
0	0	0	1
0	1	0	0
1	0	1	0
1	1	1	1

Table 2.14: The truth table of using one ***CNOT*** gate and one ***NOT*** gate to implement **XNOR** operation.

In Figure 2.25, the first control bit (the top wire) C_1 and the target bit (the bottom wire) ***T*** of the ***CNOT*** gate respectively encode the first input q[2] and the second input q[1]of a classical bit in **XNOR** operation of a classical bit in Table 2.13. In Figure 2.25, the target bit (the bottom wire) *T* of the ***CNOT*** gate also stores one output $\overline{\mathrm{q}[2] \oplus \mathrm{q}[1]}$ of a classical bit in **XNOR** operation of a classical bit in Table 2.13.

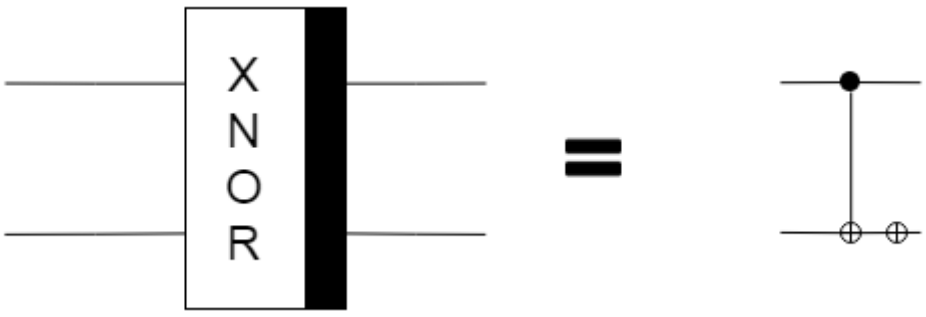

Figure 2.25: The quantum circuit of implementing **XNOR** operation of a *classical* bit.

Implementing **XNOR** operation of a *classical* bit that takes two inputs q[2] and q[1] of a classical bit and produces one output $\overline{\mathrm{q}[2] \oplus \mathrm{q}[1]}$ of a classical bit is equivalent to implement one ***CNOT*** gate and one ***NOT*** gate in which the control bit and the target

bit encode two inputs q[2] and q[1] of a classical bit and its target bit also stores one output $\overline{q[2] \oplus q[1]}$. Hence, in Figure 2.25, we first apply one ***CNOT*** gate to generate an output $q[2] \oplus q[1]$ of **XOR** operation that is stored in the target bit (the bottom wire) ***T***. Next, we use one ***NOT*** gate to produce the negation of **XOR** operation $(q[2] \oplus q[1])$ that is to complete **XNOR** operation $\overline{q[2] \oplus q[1]}$ that is stored in the target bit ***T***.

In Listing 2.9, the program in the backend *ibmqx4* with five quantum bits in **IBM**'s quantum computer is the *ninth* example of the *second* chapter in which we describe how to write a quantum program to complete **XNOR** operation of a classical bit by means of applying one ***CNOT*** gate and one ***NOT*** gate. Figure 2.26 is the corresponding quantum circuit of the program in Listing 2.9. The statement "OPENQASM 2.0;" on line one of Listing 2.9 is to indicate that the program is written with version 2.0 of Open QASM. Next, the statement "include "qelib1.inc";" on line two of Listing 2.9 is to continue parsing the file "qelib1.inc" as if the contents of the file were pasted at the location of the include statement, where the file "qelib1.inc" is **Quantum Experience (QE) Standard Header** and the path is specified relative to the current working directory.

```
1. OPENQASM 2.0;
2. include "qelib1.inc";
3. qreg q[5];
4. creg c[5];
5. h q[1];
6. h q[2];
7. cx q[2],q[1];
8. x q[1];
9. measure q[1] -> c[1];
10. measure q[2] -> c[2];
```

Listing 2.9: The program of using one ***CNOT*** gate and one ***NOT*** gate to implement **XNOR** operation.

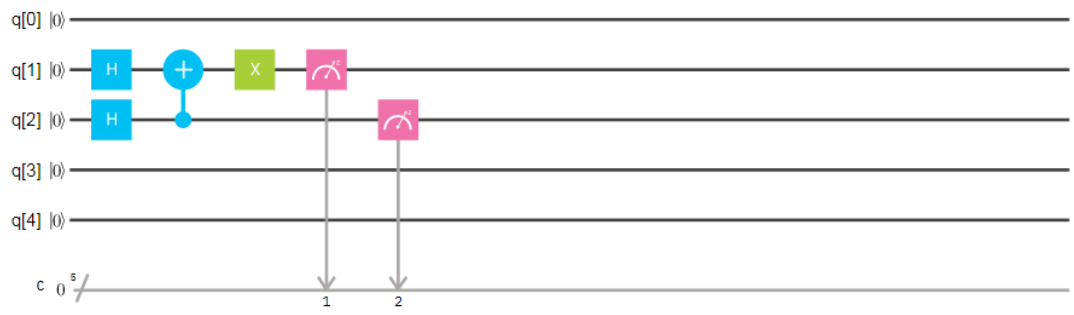

Figure 2.26: The corresponding quantum circuit of the program in Listing 2.9.

Next, the statement "qreg q[5];" on line three of Listing 2.9 is to declare that in the program there are five quantum bits. In the left top of Figure 2.26, five quantum bits are subsequently q[0], q[1], q[2], q[3] and q[4]. The initial value of each quantum bit is set to |0>. We use two quantum bits q[2] and q[1] to respectively encode the control bit and the target bit of one ***CNOT*** gate. This implies that we apply quantum bits q[2] and q[1] to encode two inputs of a classical bit in **XNOR** operation of a *classical* bit and use quantum bit q[1] to store the result of **XNOR** operation of a *classical* bit. For the convenience of our explanation, $q[k]^0$ for $0 \le k \le 4$ is to represent the value of q[k] to be zero (0) and $q[k]^1$ for $0 \le k \le 4$ is to represent the value of q[k] to be one (1). Similarly, for the convenience of our explanation, an initial state vector of implementing **XNOR** operation of a classical bit is as follows:

$$|F_0\rangle = |q[2]^0\rangle\ |q[1]^0\rangle = |0\rangle\ |0\rangle = |00\rangle.$$

Then, the statement "creg c[5];" on line four of Listing 2.9 is to declare that there are five classical bits in the program. In the left bottom of Figure 2.26, five classical bits are respectively c[0], c[1], c[2], c[3] and c[4]. The initial value of each classical bit is set to 0.

Next, the two statements "h q[1];" and "h q[2];"on line five through line six of Listing 2.9 implement two Hadamard gates of the *first* time slot of the quantum circuit in Figure 2.26. This is to say that the statement "h q[1];" converts q[1] from one state |0> to another state $\frac{1}{\sqrt{2}}$ (|0> + |1>) (its superposition) and the statement "h q[2];" converts q[2] from one state |0> to another state $\frac{1}{\sqrt{2}}$ (|0> + |1>) (its superposition). Hence, after two Hadamard gates in the *first* time slot of the quantum circuit in Figure 2.26 are implemented by means of applying the two statements "h q[1];" and "h

q[2];”on line five through line six of Listing 2.9, the following new state vector is obtained:

$$|F_1\rangle = \frac{1}{2}\ (|0\rangle\,|0\rangle + |0\rangle\,|1\rangle + |1\rangle\,|0\rangle + |1\rangle\,|1\rangle)$$

$$= \frac{1}{2}\ (|00\rangle + |01\rangle + |10\rangle + |11\rangle).$$

In the new state vector $|F_1\rangle$, four combinational states of quantum bits q[2] and q[1] with that the amplitude of each combinational state is $\frac{1}{2}$ encode all of the possible inputs in **XNOR** operation of a classical bit in Table 2.13. Quantum bit q[1] stores the result for **XNOR** operation of a classical bit in Table 2.13.

Next, the statement “cx q[2],q[1];” on line seven of Listing 2.9 implements one ***CNOT*** gates of the *second* time slot of the quantum circuit in Figure 2.26. They take the new state vector $|F_1\rangle = \frac{1}{2}\ (|00\rangle + |01\rangle + |10\rangle + |11\rangle)$ as the input in the *second* time slot of Figure 2.26. This implies that in the new state vector $|F_1\rangle$ the state ($|00\rangle$) of quantum bits q[2] and q[1] is not changed and the state ($|01\rangle$) of quantum bits q[2] and q[1] is also not changed because the value of the control bit q[2] is equal to 0 (zero) and the value of the target bit q[1] is not changed. However, the state ($|10\rangle$) of quantum bits q[2] and q[1] is converted into the state ($|11\rangle$) and the state ($|11\rangle$) of quantum bits q[2] and q[1] is converted into the state ($|10\rangle$) because the value of the control bit q[2] is equal to 1 (one) and the value of the target bit q[1] is flipped. Hence, after one ***CNOT*** gate in the *second* time slot of the quantum circuit in Figure 2.26 is implemented by means of using the statement “cx q[2],q[1];” on line seven of Listing 2.9, the following new state vector is obtained:

$$|F_2\rangle = \frac{1}{2}\ (|0\rangle\,|0\rangle + |0\rangle\,|1\rangle + |1\rangle\,|1\rangle + |1\rangle\,|0\rangle)$$

$$= \frac{1}{2}\ (|00\rangle + |01\rangle + |11\rangle + |10\rangle).$$

Next, the statement “x q[1];” on line eight of Listing 2.9 implements one ***NOT*** gate in the *third* time slot of the quantum circuit in Figure 2.26. It takes the new state vector $|F_2\rangle = \frac{1}{2}\ (|00\rangle + |01\rangle + |11\rangle + |10\rangle)$ as the input in the *third* time slot of the quantum

circuit in Figure 2.26. This is to say that in the new state vector $|F_2\rangle$ the states (|00>), (|01>), (|11>) and (|10>) of quantum bits q[2] and q[1] are subsequently converted into the new states (|01>), (|00>), (|10>) and (|11>) because the value of quantum bit q[1] is flipped, there is no quantum gate to act on quantum bit q[2] and the value of quantum bit q[2] is not changed. Therefore, after one ***NOT*** gate in the *third* time slot of the quantum circuit in Figure 2.26 is implemented by means of applying the statement "x q[1];" on line eight of Listing 2.9, the following new state vector is obtained:

$$|F_3\rangle = \frac{1}{2}\ (|0\rangle\,|1\rangle + |0\rangle\,|0\rangle + |1\rangle\,|0\rangle + |1\rangle\,|1\rangle)$$

$$= \frac{1}{2}\ (|01\rangle + |00\rangle + |10\rangle + |11\rangle).$$

Next, two measurements from the *fourth* time slot through the *fifth* time slot of the quantum circuit in Figure 2.26 were implemented by the two statements "measure q[1] -> c[1];" and "measure q[2] -> c[2];" on line nine through line ten of Listing 2.9. They are to measure the second quantum bit q[1] and the third quantum bit q[2] and to record the measurement outcome by overwriting the second classical bit c[1] and the third classical bit c[2]. In the backend *ibmqx4* with five quantum bits in **IBM**'s quantum computers, we make use of the command "simulate" to execute the program in Listing 2.9. The measured result is shown in Figure 2.27. From Figure 2.27, we obtain the answer 00000 (c[4] = q[4] = |0>, c[3] = q[3] = |0>, c[2] = q[2] = |0>, c[1] = q[1] = |0> and c[0] = q[0] = |0>) with the probability 0.370. Since in **XNOR** operation of a classical bit the value of the first input (the control bit) q[2] is equal to 0 (zero) and the value of the second input (the target bit) q[1] is equal to 1 (one), the value of the output (the target bit) q[1] is equal to 0 (zero) with the probability 0.370. Or we get the answer 00100 (c[4] = q[4] = |0>, c[3] = q[3] = |0>, c[2] = q[2] = |1>, c[1] = q[1] = |0> and c[0] = q[0] = |0>) with the probability 0.260. Because in **XNOR** operation of a classical bit the value of the first input (the control bit) q[2] is equal to 1 (one) and the value of the second input (the target bit) q[1] is equal to 0 (zero), the value of the output (the target bit) q[1] is equal to 0 (zero) with the probability 0.260.

Figure 2.27: After the measurement to the program in Listing 2.9 is completed, we obtain the answer 00000 with the probability 0.370, the answer 00100 with the

probability 0.260, the answer 00010 with the probability 0.230 or the answer 00110 with the probability 0.140.

Or we acquire the answer 00010 (c[4] = q[4] = |0>, c[3] = q[3] = |0>, c[2] = q[2] = |0>, c[1] = q[1] = |1> and c[0] = q[0] = |0>) with the probability 0.230. Because in **XNOR** operation of a classical bit the value of the first input (the control bit) q[2] is equal to 0 (zero) and the value of the second input (the target bit) q[1] is equal to 0 (zero), the value of the output (the target bit) q[1] is equal to 1 (one) with the probability 0.230. Or we obtain the answer 00110 (c[4] = q[4] = |0>, c[3] = q[3] = |0>, c[2] = q[2] = |1>, c[1] = q[1] = |1> and c[0] = q[0] = |0>) with the probability 0.140. In **XNOR** operation of a classical bit, the value of the first input (the control bit) q[2] is equal to 1 (one) and the value of the second input (the target bit) q[1] is also equal to 1 (one), so the value of the output (the target bit) q[1] is equal to 1 (one) with the probability 0.140.

2.9 Summary

In this chapter we offered an illustration to how logic operations consisting of **NOT**, **AND**, **NAND, OR**, **NOR**, **Exclusive-OR** (**XOR**) and **Exclusive-NOR** (**XNOR**) on bits were implemented by means of using quantum bits and quantum gates in **IBM**'s quantum computers. We introduced the first program in Listing 2.1 and the second program in Listing 2.2 to explain how the one's complement (the **NOT** operation) of a bit and the one's complement (the **NOT** operation) of two bits were implemented by means of using quantum bits and the ***X*** gates (the ***NOT*** gates) in **IBM**'s quantum computers. Next, we described the third program in Listing 2.3 to show how one ***CCNOT*** gate was implemented by means of decomposing **CCNOT** gate into six **CNOT** gates and nine gates of one bit in **IBM**'s quantum computers.

Then, we introduced the fourth program in Listing 2.4 to reveal how the **AND** operation of a bit was implemented by means of using one ***CCNOT*** gate and three quantum bits in **IBM**'s quantum computers. We also illustrated the fifth program in Listing 2.5 to explain how the **NAND** operation of a bit was implemented by means of applying one ***CCNOT*** gate and three quantum bits in **IBM**'s quantum computers. Next, we described the sixth program in Listing 2.6 to show how the **OR** operation of a bit was implemented by means of using one ***CCNOT*** gate, four ***NOT*** gates (four ***X*** gates) and three quantum bits in **IBM**'s quantum computers.

We then illustrated the seventh program in Listing 2.7 to reveal how the **NOR** operation of a bit was implemented by means of applying one ***CCNOT*** gate, four ***NOT***

gates (four ***X*** gates) and three quantum bits in **IBM**'s quantum computers. We also introduced the eighth program in Listing 2.8 to explain how the **XOR** operation of a bit was implemented by means of using one ***CNOT*** gate and two quantum bits in **IBM**'s quantum computers. Next, we described the ninth program in Listing 2.9 to show how the **XNOR** operation of a bit was implemented by means of applying one ***CNOT*** gate, one ***NOT*** gate (one ***X*** gate) and two quantum bits in **IBM**'s quantum computers.

2.10 Bibliographical Notes

The textbooks written by these authors in [Mano 1979; Mano 1993; Chang and Vasilakos 2014] is a good illustration to logic operations including **NOT**, **AND**, **NAND**, **OR**, **NOR**, **Exclusive-OR (XOR)** and **Exclusive-NOR (XNOR)** on bits. A good introduction of decomposing ***CCNOT*** gate into six ***CNOT*** gates and nine gates of one bit can be found in the textbook [Nielsen and Chuang 2000] and in the famous article [Shende and Markov 2009]. A good guide of writing nine quantum programs from Listing 2.1 to Listing 2.9 can also be found from the famous menu in [**IBM Q** 2016]. A good illustration to Boolean's functions discussed in exercises in Section 2.11 is [Mano 1979; Mano 1993; Brown and Vranesic 2007; Chang and Vasilakos 2014].

2.11 Exercises

2.1 The unary operator "$\bar{}$" denotes logical operation **NOT** and the binary operator "$\vee$" denotes logical operation **OR**. For a logical operation, $\bar{x} \vee y$, x and y are Boolean variables that are subsequently the first input and the second input. Its truth table is shown in Table 2.15. Please write a quantum program to implement the function of the logical operation, $\bar{x} \vee y$.

The first input (x)	The second input (y)	$\bar{x} \vee y$
0	0	1
0	1	1
1	0	0
1	1	1

Table 2.15: The truth table to a logical operation $\bar{x} \vee y$.

2.2 The unary operator "$\bar{}$" denotes logical operation **NOT** and the binary operator "$\vee$" denotes logical operation **OR**. For a logical operation, $x \vee \bar{y}$, x and y are Boolean variables that are respectively its first input and its second input. Its truth table is shown in Table 2.16. Please write a quantum program to implement the function of

the logical operation, $x \vee \bar{y}$.

The first input (x)	The second input (y)	$x \vee \bar{y}$
0	0	1
0	1	0
1	0	1
1	1	1

Table 2.16: The truth table to a logical operation $x \vee \bar{y}$.

2.3 The unary operator "$\bar{}$" denotes logical operation **NOT** and the binary operator "$\wedge$" denotes logical operation **AND**. For a logical operation, $\bar{y} \wedge (x \vee \bar{x}) = \bar{y} \wedge 1 = \bar{y}$, x and y are Boolean variables that are subsequently the first input and the second input. Its truth table is shown in Table 2.17. Please write a quantum program to implement the function of the logical operation, $\bar{y} \wedge (x \vee \bar{x}) = \bar{y} \wedge 1 = \bar{y}$.

The first input (x)	The second input (y)	$\bar{y} \wedge (x \vee \bar{x}) = \bar{y} \wedge 1 = \bar{y}$
0	0	1
0	1	0
1	0	1
1	1	0

Table 2.17: The truth table to a logical operation $\bar{y} \wedge (x \vee \bar{x}) = \bar{y} \wedge 1 = \bar{y}$.

2.4 The unary operator "$\bar{}$" denotes logical operation **NOT** and the binary operator "$\wedge$" denotes logical operation **AND**. For a logical operation, $\bar{x} \wedge (y \vee \bar{y}) = \bar{x} \wedge 1 = \bar{x}$, x and y are Boolean variables that are respectively the first input and the second input. Its truth table is shown in Table 2.18. Please write a quantum program to implement the function of the logical operation, $\bar{x} \wedge (y \vee \bar{y}) = \bar{x} \wedge 1 = \bar{x}$.

The first input (x)	The second input (y)	$\bar{x} \wedge (y \vee \bar{y}) = \bar{x} \wedge 1 = \bar{x}$
0	0	1
0	1	1
1	0	0
1	1	0

Table 2.18: The truth table to a logical operation $\bar{x} \wedge (y \vee \bar{y}) = \bar{x} \wedge 1 = \bar{x}$.

2.5 The unary operator "$\bar{}$" denotes logical operation **NOT** and the binary operator "$\wedge$" denotes logical operation **AND**. For a logical operation, $\bar{x} \wedge y$, x and y are Boolean variables that are subsequently the first input and the second input. Its truth table

is shown in Table 2.19. Please write a quantum program to implement the function of the logical operation, $\bar{x} \wedge y$.

The first input (x)	The second input (y)	$\bar{x} \wedge y$
0	0	0
0	1	1
1	0	0
1	1	0

Table 2.19: The truth table to a logical operation $\bar{x} \wedge y$.

2.6 The unary operator "$\bar{}$" denotes logical operation **NOT** and the binary operator "$\wedge$" denotes logical operation **AND**. For a logical operation, $x \wedge \bar{y}$, x and y are Boolean variables that are respectively its first input and its second input. Its truth table is shown in Table 2.20. Please write a quantum program to implement the function of the logical operation, $x \wedge \bar{y}$.

The first input (x)	The second input (y)	$x \wedge \bar{y}$
0	0	0
0	1	0
1	0	1
1	1	0

Table 2.20: The truth table to a logical operation $x \wedge \bar{y}$.