Multiple Control Constructions

prepared by Maksim Dimitrijev and Abuzer Yakaryilmaz

```
# Because I am running in Android with Google Colab, so I need some cells to be ran every notebook beforehand

!pip install qiskit==1.3.0

import qiskit
print(qiskit.__version__)

!pip install pylatexenc

!pip install qiskit-aer
```

Show hidden output

Remember that when appying CNOT gate, NOT operator is applied to the target qubit if the control qubit is in state $|1\rangle$:

$$CNOT = \left(\begin{array}{cc|cc} \mathbf{1} & 0 & 0 & 0 \\ 0 & \mathbf{1} & 0 & 0 \\ \hline 0 & 0 & 0 & \mathbf{1} \\ 0 & 0 & \mathbf{1} & 0 \end{array} \right) .$$

How can we obtain the following operator, in which the NOT operator is applied to the target qubit if the control qubit is in state $|0\rangle$?

$$C_0NOT = \left(\begin{array}{cc|cc} 0 & \mathbf{1} & 0 & 0 \\ \mathbf{1} & 0 & 0 & 0 \\ \hline 0 & 0 & \mathbf{1} & 0 \\ 0 & 0 & 0 & \mathbf{1} \end{array} \right) .$$

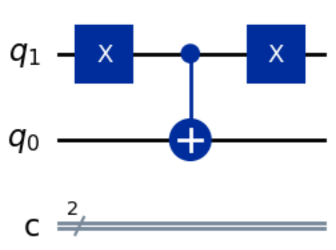
As also mentioned in the notebook Operators on Multiple Bits, we can apply a NOT operator on the control bit before applying $CNOT$ operator so that the NOT operator is applied to the target qubit when the control qubit has been in state $|0\rangle$. To recover the previous value of the control qubit, we apply the NOT operator once more after the $CNOT$ operator. In short:

- apply NOT operator to the control qubit,
- apply $CNOT$ operator, and,
- apply NOT operator to the control qubit.

We can implement this idea in Qiskit as follows.

```
from qiskit import QuantumRegister, ClassicalRegister, QuantumCircuit
from qiskit_aer import UnitarySimulator

q = QuantumRegister(2, "q")
c = ClassicalRegister(2, "c")

qc = QuantumCircuit(q,c)

qc.x(q[1])

qc.cx(q[1],q[0])

# Returning control qubit to the initial state
qc.x(q[1])

job = UnitarySimulator().run(qc, shots = 1)
U=job.result().get_unitary(qc,decimals=3).data

print("CNOT(0) = ")
for row in U:
    s = ""
    for value in row:
        s = s + str(round(value.real,2)) + "  "
    print(s)

qc.draw(output="mpl", reverse_bits=True)
```

```
CNOT(0) =
0.0  1.0  0.0  0.0
1.0  0.0  0.0  0.0
0.0  0.0  1.0  0.0
0.0  0.0  0.0  1.0
```



By using this trick, more complex conditional operators can be implemented.

CCNOT

Now we introduce $CCNOT$ gate: **controlled-controlled-not operator** (Toffoli gate), which is controlled by two qubits. The implementation of $CCNOT$ gate in Qiskit is as follows:

```
circuit.ccx(control-qubit1,control-qubit2,target-qubit)
```

That is, NOT operator is applied to the target qubit when both control qubits are in state $|1\rangle$. Its matrix representation is as follows:

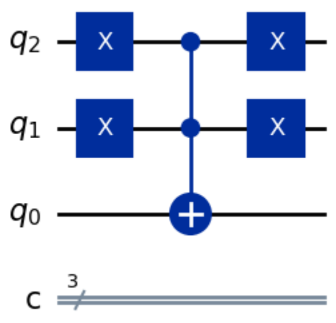
$$CCNOT = \left(\begin{array}{cc|cc|cc|cc} \mathbf{1} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \mathbf{1} & 0 & 0 & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & \mathbf{1} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \mathbf{1} & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & \mathbf{1} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \mathbf{1} & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & 0 & 0 & 0 & \mathbf{1} \\ 0 & 0 & 0 & 0 & 0 & 0 & \mathbf{1} & 0 \end{array}\right).$$

Task 1

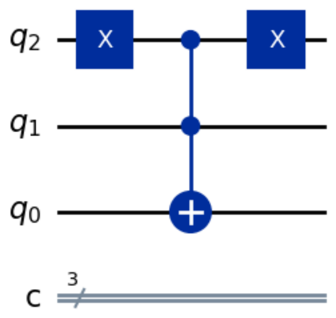
Implement each of the following operators in Qiskit by using three qubits. Verify your implementation by using "unitary_simulator" backend.

$$C_0C_0NOT = \left(\begin{array}{cc|cc|cc|cc} 0 & \mathbf{1} & 0 & 0 & 0 & 0 & 0 & 0 \\ \mathbf{1} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & \mathbf{1} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \mathbf{1} & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & \mathbf{1} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \mathbf{1} & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & 0 & 0 & \mathbf{1} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & \mathbf{1} \end{array}\right), \quad C_0C_1NOT = \left(\begin{array}{cc|cc|cc|cc} \mathbf{1} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \mathbf{1} & 0 & 0 & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & \mathbf{1} & 0 & 0 & 0 & 0 \\ 0 & 0 & \mathbf{1} & 0 & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & \mathbf{1} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \mathbf{1} & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & 0 & 0 & \mathbf{1} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & \mathbf{1} \end{array}\right), \text{ and } C_1C_0NOT = \left(\begin{array}{cc|cc|cc|cc} \mathbf{1} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \mathbf{1} & 0 & 0 & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & \mathbf{1} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \mathbf{1} & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & 0 & \mathbf{1} & 0 & 0 \\ 0 & 0 & 0 & 0 & \mathbf{1} & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & 0 & 0 & \mathbf{1} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & \mathbf{1} \end{array}\right).$$

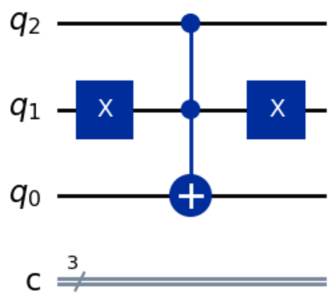
```
Start coding or generate with AI.
```



```
CNOT( 00 ) =
0.0  1.0  0.0  0.0  0.0  0.0  0.0  0.0
1.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0
0.0  0.0  1.0  0.0  0.0  0.0  0.0  0.0
0.0  0.0  0.0  1.0  0.0  0.0  0.0  0.0
0.0  0.0  0.0  0.0  1.0  0.0  0.0  0.0
0.0  0.0  0.0  0.0  0.0  1.0  0.0  0.0
0.0  0.0  0.0  0.0  0.0  0.0  1.0  0.0
0.0  0.0  0.0  0.0  0.0  0.0  0.0  1.0
```



```
CNOT( 01 ) =
1.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0
0.0  1.0  0.0  0.0  0.0  0.0  0.0  0.0
0.0  0.0  0.0  1.0  0.0  0.0  0.0  0.0
0.0  0.0  1.0  0.0  0.0  0.0  0.0  0.0
0.0  0.0  0.0  0.0  1.0  0.0  0.0  0.0
0.0  0.0  0.0  0.0  0.0  1.0  0.0  0.0
0.0  0.0  0.0  0.0  0.0  0.0  1.0  0.0
0.0  0.0  0.0  0.0  0.0  0.0  0.0  1.0
```



```
CNOT( 10 ) =
1.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0
0.0  1.0  0.0  0.0  0.0  0.0  0.0  0.0
0.0  0.0  1.0  0.0  0.0  0.0  0.0  0.0
0.0  0.0  0.0  1.0  0.0  0.0  0.0  0.0
0.0  0.0  0.0  0.0  0.0  1.0  0.0  0.0
0.0  0.0  0.0  0.0  1.0  0.0  0.0  0.0
0.0  0.0  0.0  0.0  0.0  0.0  1.0  0.0
0.0  0.0  0.0  0.0  0.0  0.0  0.0  1.0
```

[click for our solution]

Double-click (or enter) to edit

```
Start coding or generate with AI.
```

More controls

Here we present basic methods on how to implement NOT gates controlled by more than two qubits by using $CNOT$, $CCNOT$, and some ancilla (auxiliary) qubits.

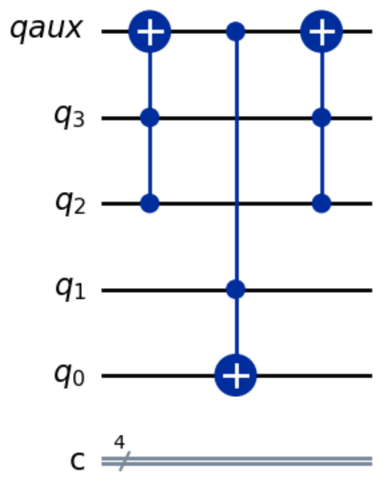
(Note that Qiskit has a method called "mct" to implement such gates. Another multiple-controlled operator in Qiskit is "mcrz".)

Implementation of CCCNOT gate

We give the implementation of $CCCNOT$ gate: NOT operator is applied to target qubit when the control qubits are in state $|111\rangle$. This gate requires 4 qubits. We also use an auxiliary qubit.

Our qubits are $q_{aux}, q_3, q_2, q_1, q_0$, and the auxiliary qubit q_{aux} should be in state $|0\rangle$ after each use. The implementation of the $CCCNOT$ gate in Qiskit is given below. The short explanations are given as comments.

```
from qiskit import QuantumRegister, ClassicalRegister, QuantumCircuit
from qiskit_aer import AerSimulator

qaux = QuantumRegister(1,"qaux")
q = QuantumRegister(4,"q")
c = ClassicalRegister(4,"c")

qc = QuantumCircuit(q,qaux,c)

# step 1: set qaux to |1> if both q3 and q2 are in |1>
qc.ccx(q[3],q[2],qaux[0])

# step 2: apply NOT gate to q0 if both qaux and q1 are in |1>
qc.ccx(qaux[0],q[1],q[0])

# step 3: set qaux to |0> if both q3 and q2 are in |1> by reversing the affect of step 1
qc.ccx(q[3],q[2],qaux[0])

qc.draw(output="mpl",reverse_bits=True)
```



Now, we execute this circuit on every possible inputs and verify the correctness of the implementation experimentally.

```
from qiskit import QuantumRegister, ClassicalRegister, QuantumCircuit
from qiskit_aer import AerSimulator

all_inputs=[]
for q3 in ['0','1']:
    for q2 in ['0','1']:
        for q1 in ['0','1']:
            for q0 in ['0','1']:
                all_inputs.append(q3+q2+q1+q0)
#print(all_inputs)

```

```
print("input --> output")
for the_input in all_inputs:
    # create the circuit
    qaux = QuantumRegister(1,"qaux")
    q = QuantumRegister(4,"q")
    c = ClassicalRegister(4,"c")
    qc = QuantumCircuit(q,qaux,c)
    # set the initial value of the circuit w.r.t. the input
    if the_input[0] =='1': qc.x(q[3])
    if the_input[1] =='1': qc.x(q[2])
    if the_input[2] =='1': qc.x(q[1])
    if the_input[3] =='1': qc.x(q[0])
    # implement the CCNOT gates
    qc.ccx(q[3],q[2],qaux[0])
    qc.ccx(qaux[0],q[1],q[0])
    qc.ccx(q[3],q[2],qaux[0])
    # measure the main quantum register
    qc.measure(q,c)
    # execute the circuit
    job = AerSimulator().run(qc,shots=1)
    counts = job.result().get_counts(qc)
    for key in counts: the_output = key
    printed_str = the_input[0:3]+" "+the_input[3]+" --> "+the_output[0:3]+" "+the_output[3]
    if (the_input!=the_output): printed_str = printed_str + " the output is different than the input"
    print(printed_str)
```

```
['0000', '0001', '0010', '0011', '0100', '0101', '0110', '0111', '1000', '1001', '1010', '1011', '1100', '1101', '1110', '1111']
input --> output
000 0 --> 000 0
000 1 --> 000 1
001 0 --> 001 0
001 1 --> 001 1
010 0 --> 010 0
010 1 --> 010 1
011 0 --> 011 0
011 1 --> 011 1
100 0 --> 100 0
100 1 --> 100 1
101 0 --> 101 0
101 1 --> 101 1
110 0 --> 110 0
110 1 --> 110 1
111 0 --> 111 1 the output is different than the input
111 1 --> 111 0 the output is different than the input
```

Task 2

Provide an implementation of the NOT operator controlled by 4 qubits ($CCCCNOT$) in Qiskit. Verify its correctness by executing your solution on all possible inputs. (See the above example)

You may use two auxiliary qubits.

click for our solution

Task 3

Repeat Task 2 for the operator $C_1C_0C_1C_0NOT$: NOT operator is applied to the target qubit if the four control qubits are in state $|1010\rangle$.

click for our solution

Task 4 (extra)

Write a function taking a binary string "$b_1b_2b_3b_4$ that repeats Task 2 for the operator $C_{b_1}C_{b_2}C_{b_3}C_{b_4}NOT$ gate, where $b_1,\ldots,b_4$ are bits and NOT operator is applied to target qubit if the control qubits are in state $|b_1b_2b_3b_4\rangle$.

click for our solution