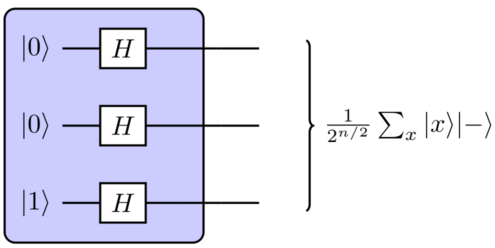
Quantum Computing HW3

Omid Tavakol
(Dated: February 13, 2024)

I. PROBLEM 1

Provide a generalization of Deutsch's problem for the case where x is 2 Qbits and $f(x)$ is 1 Qbit. What can you tell about $f(x)$ with one measurement?

Solution: So let say we have a unitary operator $U_f|x\rangle|y\rangle = |x\rangle|y \oplus f(x)\rangle$. Here x is 2 qubits and y has 1 qubit. The idea is similar to the Deutsch's problem for one qubit case. We start with applying Hadamard gate on all the qubits but the $|y\rangle$ qubit should start from 1.



Now by applying the U_f on the prepared state we will find $\frac{1}{2^{n/2}} \sum_x (-1)^{f(x)}|x\rangle|-\rangle$. Here $|-\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$. Now we apply Hadamard gate on all the input qubits we can write the following:

$$|\psi\rangle = H^{\otimes n} \frac{1}{2^{n/2}} \sum_x (-1)^{f(x)}|x\rangle|-\rangle = \frac{1}{2} \sum_{x,y} (-1)^{f(x)}(-1)^{xy}|y\rangle|-\rangle \tag{1}$$

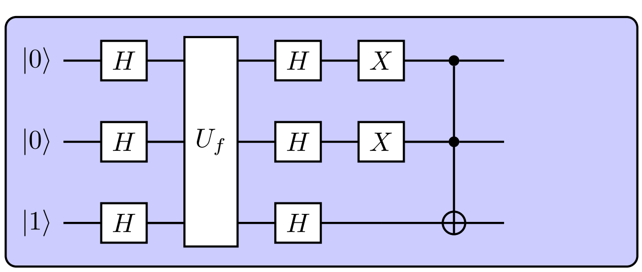
Here we used the following relation:

$$H^{\otimes n}|x\rangle = \frac{1}{2^{n/2}} \sum_y (-1)^{xy}|y\rangle \tag{2}$$

Now let us focus on what happens when $f(x)$ is constant. Then in that case we can write $|\psi\rangle$ as following

$$|\psi\rangle = \frac{1}{2^n}(-1)^{f(0)} \sum_{x,y} (-1)^{xy}|y\rangle|-\rangle = \frac{1}{2^n} \prod_j \sum_{y_j} \sum_{x_j} (-1)^{x_j y_j}|y_j\rangle|-\rangle = \frac{1}{2^n} \prod_j \sum_{y_j} (1 + (-1)^{y_j})|y_j\rangle|-\rangle \tag{3}$$

This means that in the case where $f(x)$ is constant, all the final y_j values have to be 0. Therefore, in this scenario, we can determine if the function is constant or not by measuring all the final $|x\rangle$ qubits. However, this approach does not provide significant quantum supremacy. Since we only need 2 bits of information (whether it is balanced or not, 0 or 1), we can save all this information into the $|y\rangle$ qubit and use it instead of checking all the $|x\rangle$ qubits. To achieve this, we can employ a Toffoli gate. Initially, we need to change the $|00\rangle$ state to $|11\rangle$ for control purposes, and then use the Toffoli gate to transfer that information to the $|y\rangle$ qubit. Currently, the $|y\rangle$ qubit is in the $|-\rangle$ state, so we also need to convert it to the computational basis, which can be accomplished by applying a Hadamard gate. Therefore, the final circuit will be as follows:



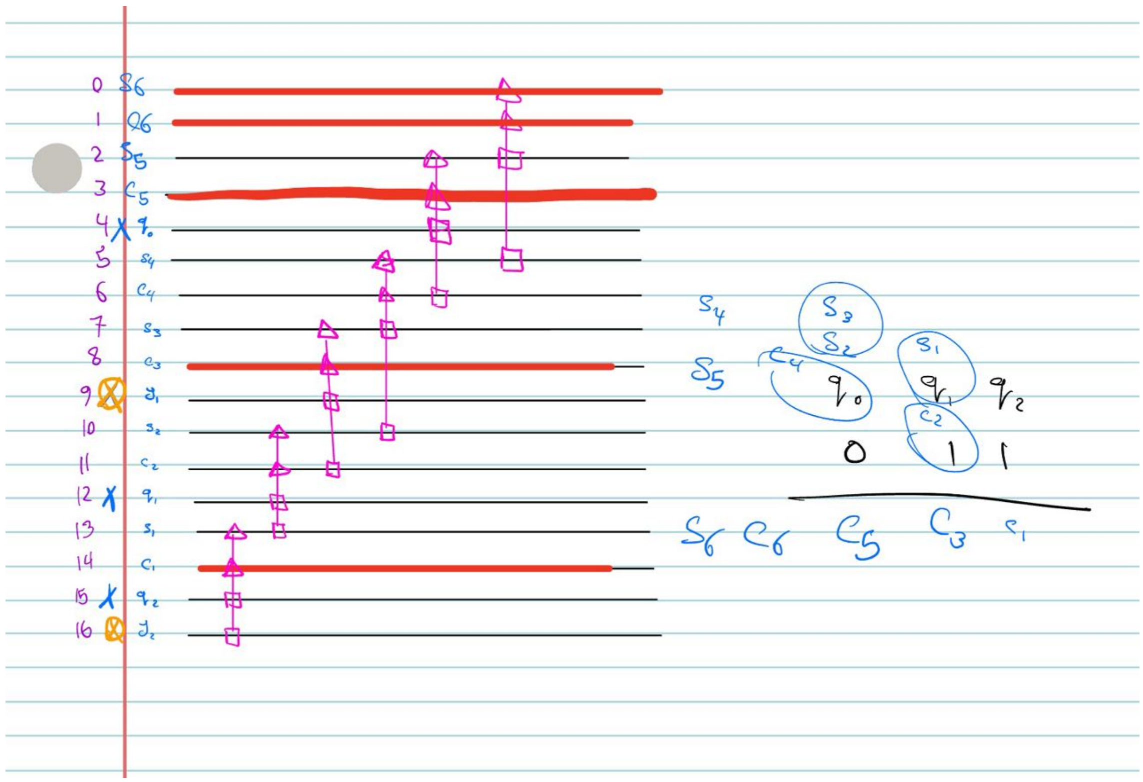


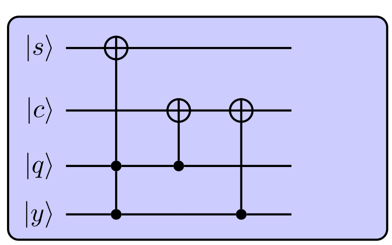
FIG. 1. y_0 and y_1 are chosen to be 11 and $q_0q_1q_2$ is the number that we add. The resulting sum is $s_6c_6c_5c_3c_1$ and correspond to the red line qubits.

Now by only measuring the $|y\rangle$ qubit we can say if the function is constant or not. With this circuit if the measured qubit is 0 then we know that the function is constant and if it is 1 then the function should not be constant.

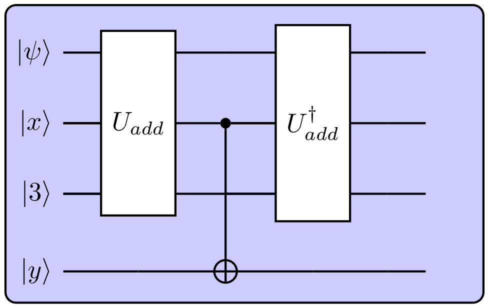
II. PROBLEM 2

Let $|x\rangle$ and $|y\rangle$ be 3 Qbit states. Define $f(x) = (x + 3) \bmod 8$. Draw a quantum circuit for the unitary transformation: $U_f \ : |x\rangle|y\rangle \rightarrow |x\rangle|y \oplus f(x)\rangle$

Solution: First, we attempt to build the adder function using unitary gates that add $x + 3$. This initial circuit employs the CNOT and Toffoli gate. The reason for introducing an additional qubit ψ is the necessity of having more qubits to store the result of the addition. Let's start by constructing the operator that adds two qubits. This operator acts on four-qubit states and performs $q + y = sc$. We can represent this operator as follows:



Let's represent this operator with two squares and two triangles in FIG [1]. We then extend it to calculate $x + 3$, where x is a three-qubit number. Now FIG [1] represents our U_{add}, which we can use in the following circuit to create a unitary operator. We no longer need to worry about the extra qubits that we used since we add the $U^{\dagger}_{add}$.



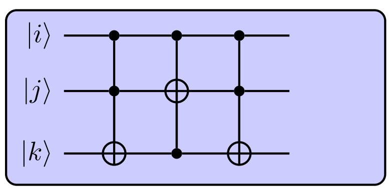
More detailed calculation and the full circuit can be found in the Appendix A. The code is written using Cirq and Qiskit library (just to compare).

III. PROBLEM 3

The Toffoli gate T_{ijk} flips bit k if both bits i and j are 1. Let the c-swap operator S_{ijk} swaps bit j and k if bit i is 1, otherwise leaving them unchanged. Write S_{ijk} in terms of T_{ijk}.

Solution: We are going to borrow the idea that $S_{ij} = C_{ij}C_{ji}C_{ij}$ and write the CSWAP as following:

$$S_{ijk} = T_{ijk}T_{ikj}T_{ijk} \tag{4}$$



The code that checks the possible input and outputs is in the appendix (problem 3)

Appendix A: Appendix

```python
"""Adder"""
# Import the Cirq library
import cirq
# Get qubits
qubits = cirq.LineQubit.range(17)




def adder_2(a, b, s, c):
  yield cirq.CCNOT(a,b,s)
  yield cirq.CNOT(a,c)
  yield cirq.CNOT(b,c)

circuit = cirq.Circuit()
q0, q1, q2 = [0,0,1]


#The inputs are at qubits 15,12,4. The outputs are 0,1,3,8,14

if q2==1:
  circuit.append(cirq.X(qubits[15]))
if q1 ==1:
  circuit.append(cirq.X(qubits[12]))
if q0 ==1 :
  circuit.append(cirq.X(qubits[4]))

circuit.append(cirq.X(qubits[16]))
circuit.append(cirq.X(qubits[9]))


circuit.append(adder_2(qubits[16],qubits[15],qubits[13],qubits[14]))

circuit.append(adder_2(qubits[12],qubits[13],qubits[10],qubits[11]))
circuit.append(adder_2(qubits[11],qubits[9],qubits[7],qubits[8]))

circuit.append(adder_2(qubits[10],qubits[7],qubits[5],qubits[6]))
circuit.append(adder_2(qubits[6],qubits[4],qubits[2],qubits[3]))

circuit.append(adder_2(qubits[5],qubits[2],qubits[0],qubits[1]))

circuit.append(cirq.measure(qubits[14]))
circuit.append(cirq.measure(qubits[8]))
circuit.append(cirq.measure(qubits[3]))
circuit.append(cirq.measure(qubits[0]))
circuit.append(cirq.measure(qubits[1]))



simulator = cirq.Simulator()
result = simulator.simulate(circuit, initial_state=0)
```

```python
print(f'{0,1,1}+{q0,q1,q2}=',result.measurements['q(0)']
[0],result.measurements['q(1)'][0],result.measurements['q(3)']
[0],result.measurements['q(8)'][0],result.measurements['q(14)'][0])

print()
print()
print('='*20)
print(circuit)
```

```
(0, 1, 1)+(0, 0, 1)= 0 0 1 0 0


====================
                               ┌──┐   ┌──┐           ┌──┐   ┌──┐
0:
────────────────────────────────────────────────────X─────M───────────
──
                                                       │
1:
────────────────────────────────────────────────────┼──────X────X───M─
──
                                                       │      │    │
2:
──────────────────────────────────────────────X─────@──────┼────@─────
──
                                                 │     │      │
3:
──────────────────────────────────────────────┼────X┼─────X┼────M─────
──
                                                 │    ││     ││
4:
──────────────────────────────────────────────@────┼┼─────@┼──────────
──
                                                 │    ││      │
5:
──────────────────────────────X───────────────┼────┼@──────@──────────
──
                                 │               │    │
6:
──────────────────────────────┼──────X────X───@────@──────────────────
──
                                 │      │    │
7:
────────────────────────X─────@──────┼────@───────────────────────────
──
```

```
                           │     │      │
8:
────────────────────────┼────X┼─────X┼────M───────────────────────────
──
                           │    ││     ││
9:
────X───────────────────@────┼┼─────@┼────────────────────────────────
──
                           │    ││      │
10:
───────────X───────────┼────┼@──────@─────────────────────────────────
─
               │           │    │
11:
───────────┼───X───X───@────@─────────────────────────────────────────
─
               │   │   │
12:
───────────@───@───┼──────────────────────────────────────────────────
─
               │       │
13:
───────X───@───────@──────────────────────────────────────────────────
─
           │
14:
───────┼───X───X───M──────────────────────────────────────────────────
─
           │   │   │
15:
───X───@───┼───@──────────────────────────────────────────────────────
─
           │   │
16:
───X───@───@──────────────────────────────────────────────────────────
─
                               └──┘   └──┘            └──┘   └──┘
```

```
"""Adder"""
# Import the Cirq library
import cirq
# Get three qubits -- two data and one target qubit
qubits = cirq.LineQubit.range(20)

q0, q1, q2 = [0,0,1]


def adder_2(a, b, s, c):
  yield cirq.CCNOT(a,b,s)
  yield cirq.CNOT(a,c)
```

```
  yield cirq.CNOT(b,c)

def adder_1(a, b, s, c):
  yield cirq.CNOT(b,c)
  yield cirq.CNOT(a,c)
  yield cirq.CCNOT(a,b,s)

circuit = cirq.Circuit()

#The inputs are at qubits 15,12,4. The outputs are 0,1,3,8,14

if q2==1:
  circuit.append(cirq.X(qubits[15]))
if q1 ==1:
  circuit.append(cirq.X(qubits[12]))
if q0 ==1 :
  circuit.append(cirq.X(qubits[4]))

circuit.append(cirq.X(qubits[16]))
circuit.append(cirq.X(qubits[9]))

def V():
  circuit.append(adder_2(qubits[16],qubits[15],qubits[13],qubits[14]))

  circuit.append(adder_2(qubits[12],qubits[13],qubits[10],qubits[11]))
  circuit.append(adder_2(qubits[11],qubits[9],qubits[7],qubits[8]))

  circuit.append(adder_2(qubits[10],qubits[7],qubits[5],qubits[6]))
  circuit.append(adder_2(qubits[6],qubits[4],qubits[2],qubits[3]))

  circuit.append(adder_2(qubits[5],qubits[2],qubits[0],qubits[1]))

def Cm():
  circuit.append(cirq.CNOT(qubits[14],qubits[17]))
  circuit.append(cirq.CNOT(qubits[8],qubits[18]))
  circuit.append(cirq.CNOT(qubits[3],qubits[19]))

def Vdag():
  circuit.append(adder_1(qubits[5],qubits[2],qubits[0],qubits[1]))
  circuit.append(adder_1(qubits[6],qubits[4],qubits[2],qubits[3]))
  circuit.append(adder_1(qubits[10],qubits[7],qubits[5],qubits[6]))
  circuit.append(adder_1(qubits[11],qubits[9],qubits[7],qubits[8]))
  circuit.append(adder_1(qubits[12],qubits[13],qubits[10],qubits[11]))
  circuit.append(adder_1(qubits[16],qubits[15],qubits[13],qubits[14]))

V()
Cm()
Vdag()

'''
```

```
circuit.append(cirq.measure(qubits[14]))
circuit.append(cirq.measure(qubits[8]))
circuit.append(cirq.measure(qubits[3]))
circuit.append(cirq.measure(qubits[0]))
circuit.append(cirq.measure(qubits[1]))
'''
circuit.append(cirq.measure(qubits[17]))
circuit.append(cirq.measure(qubits[18]))
circuit.append(cirq.measure(qubits[19]))

simulator = cirq.Simulator()
result = simulator.simulate(circuit, initial_state=0)
print('='*40)
print()
print()

print(f'{0}{1}{1} + {q0}{q1}{q2} (mod 8)
=',result.measurements['q(19)'][0],result.measurements['q(18)']
[0],result.measurements['q(17)'][0])

print()
print()
print('='*40)
print(circuit)
```

```
========================================


011 + 001 (mod 8) = 1 0 0


========================================
                                                               ┌──┐     ┌──┐                 ┌──┐     ┌──┐
┌──┐
0:
────────────────────────────────────────────────────────────────────────X────────────────────────
───────────X──────────────────────────────────────────────────
                                                                            │
│
1:
────────────────────────────────────────────────────────────────────────┼────────X──────X────X─
───X───────┼──────────────────────────────────────────────────
                                                                            │        │      │
│     │       │
2:
────────────────────────────────────────────────────────────────X───────@────────┼──────@────@─
```

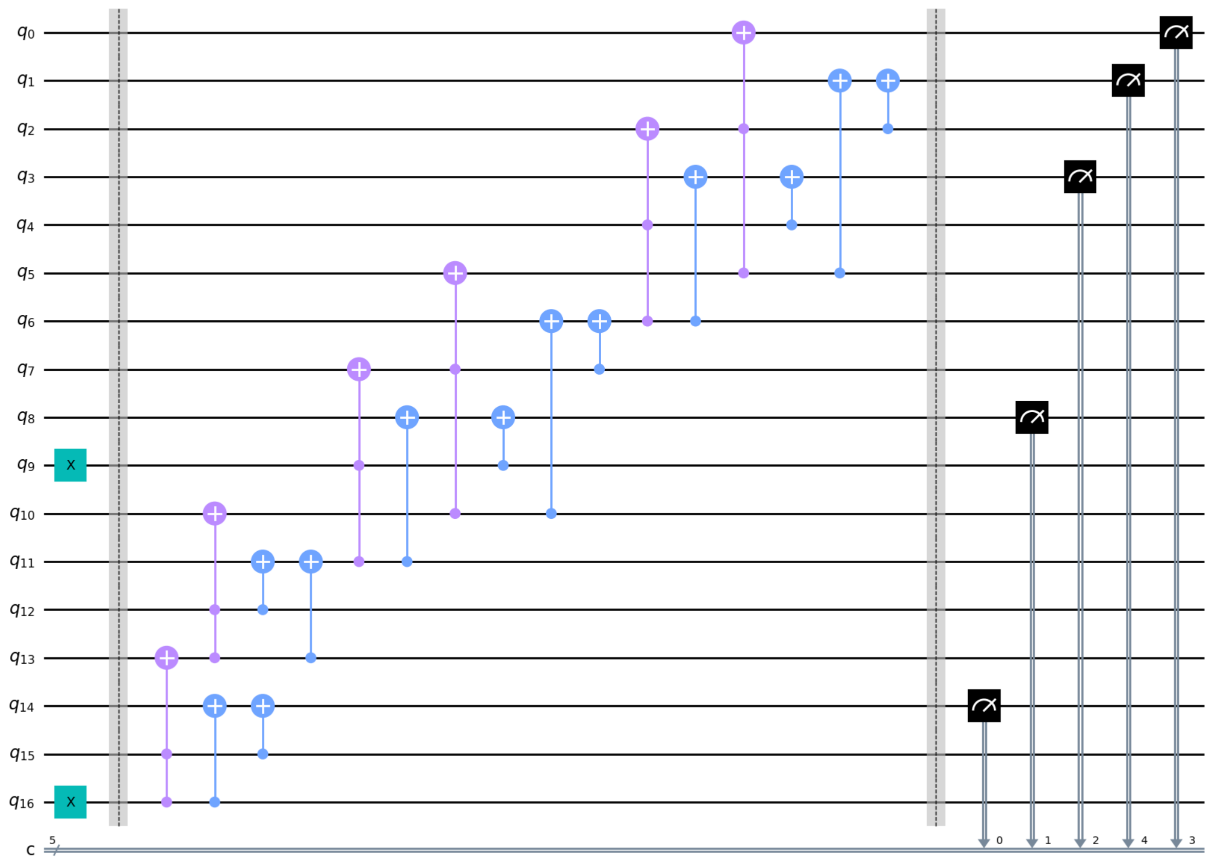
```
───┼─────@───X───────────────────────────────────
                                                 |     |      |
|     |   |
3:
──────────────────────────────────────────────┼────X┼─────X┼────@───X─
───┼X────┼───┼───────────────────────────────────
                                                 |    ||     ||    |
|    ||    |   |
4:
──────────────────────────────────────────────@────┼┼─────@┼────┼───@─
───┼┼────┼───@───────────────────────────────────
                                                 |    ||      |    |
||    |   |
5:
──────────────────────────────X───────────────┼────┼@──────@────┼─────
───@┼────@───┼───────────X───────────────────────
                                 |               |    |            |
|        |           |
6:
──────────────────────────────┼──────X────X───@────@────────────┼─────
────@────────@───X───X───┼───────────────────────
                                 |      |    |                     |
|   |   |
7:
────────────────────────X─────@──────┼────@─────────────────────┼─────
─────────────────@───┼───@───X───────────────────
                           |     |      |                          |
|   |   |
8:
────────────────────────┼────X┼─────X┼────@───X────X────────────┼─────
─────────────────────┼───┼───┼───────────────────
                           |    ||     ||    |   |    |            |
|   |   |
9:
────X───────────────────@────┼┼─────@┼────┼───@────┼────────────┼─────
─────────────────────┼───┼───@───────────────────
                           |    ||      |    |        |            |
|   |   |
10:
───────────X───────────┼────┼@──────@────┼────────┼────────────┼──────
────────────────────@───@───┼───────────X───────
               |           |    |            |        |            |
|           |
11:
───────────┼───X───X───@────@────────────┼────────@────────────┼──────
────────────────────────────@───X───X───┼───────
               |   |   |                     |                     |
|   |   |
12:
```

```
───────────@───@───┼─────────────────────┼─────────────────────┼──────
────────────────────────────────┼───@───@───────
               │       │                     │                     │
│       │
13:
───────X───@───────@─────────────────────┼─────────────────────┼──────
────────────────────────────────@───────@───X───
           │                                 │                     │
│
14:
───────┼───X───X───@───X────X────────────┼─────────────────────┼──────
────────────────────────────────────────────┼───
           │   │   │   │   │    │            │                     │
│
15:
───X───@───┼───@───┼───@────┼────────────┼─────────────────────┼──────
────────────────────────────────────────────@───
           │   │       │        │            │                     │
│
16:
───X───@───@───────┼────────@────────────┼─────────────────────┼──────
────────────────────────────────────────────@───
                       │                     │                     │
17:
───────────────────X───M─────────────────┼─────────────────────┼──────
────────────────────────────────────────────────
                                             │                     │
18:
─────────────────────────────────────────X───M─────────────────┼──────
────────────────────────────────────────────────
                                                                   │
19:
───────────────────────────────────────────────────────────────X───M──
────────────────────────────────────────────────
                               └──┘   └──┘           └──┘   └──┘
└──┘
```

#Problem 2 (Qiskit)

```python
from qiskit import QuantumCircuit, Aer, execute

qc = QuantumCircuit(17,5)

def adder_2(a, b, s, c):
    """Constructs a quantum circuit for a 2-qubit adder."""
    qc.ccx(a, b, s)
    qc.cx(a, c)
    qc.cx(b, c)
```

```
q0, q1, q2 = [0,0,0]

if q2==1:
  qc.x(15)
if q1 ==1:
  qc.x(12)
if q0 ==1 :
  qc.x(4)

qc.x(16)
qc.x(9)
qc.barrier()
adder_2(16, 15, 13, 14)
adder_2(12, 13, 10, 11)
adder_2(11,9,7,8)
adder_2(10,7,5,6)
adder_2(6,4,2,3)
adder_2(5,2,0,1)
qc.barrier()


qc.measure(14,0)
qc.measure(8,1)
qc.measure(3,2)
qc.measure(1,4)
qc.measure(0,3)


qc.draw('mpl')
```



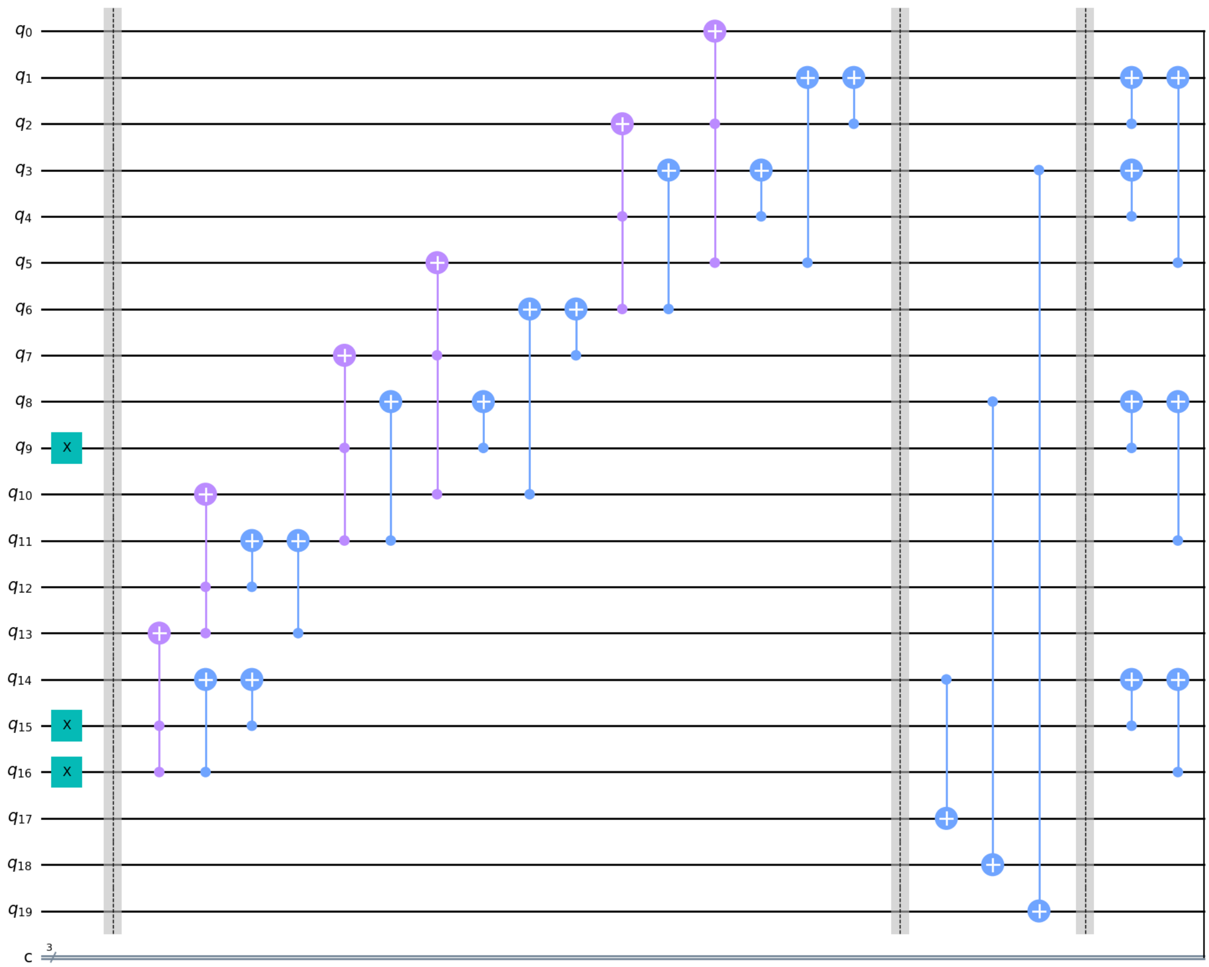

```
# Choose the Aer simulator backend
backend = Aer.get_backend('qasm_simulator')

# Execute the circuit on the backend
job = execute(qc, backend, shots=1000)

# Get the result
result = job.result()

# Get the counts
counts = result.get_counts(qc)

# Print the counts
sum = next((key for key, value in counts.items() if value == 1000),
None)
print("="*20)
print(f'{0,1,1} + {q0,q1,q2} mod8 =',sum[2:5])
print("="*20)

====================
(0, 1, 1) + (0, 0, 1) mod8 = 100
====================
```

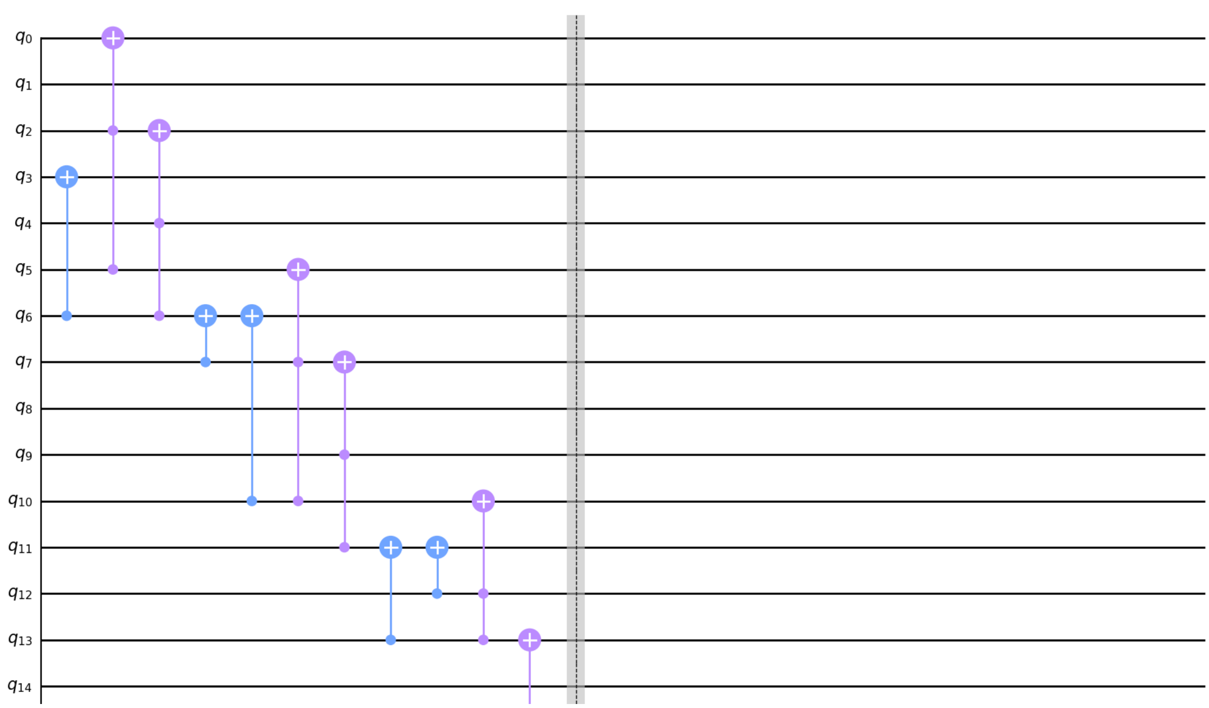
```
<ipython-input-7-6330d06fab74>:2: DeprecationWarning: The 'qiskit.Aer'
entry point is deprecated and will be removed in Qiskit 1.0. You
should use 'qiskit_aer.Aer' directly instead.
  backend = Aer.get_backend('qasm_simulator')
<ipython-input-7-6330d06fab74>:5: DeprecationWarning: The function
``qiskit.execute_function.execute()`` is deprecated as of qiskit
0.46.0. It will be removed in the Qiskit 1.0 release. This function
combines ``transpile`` and ``backend.run``, which is covered by
``Sampler`` :mod:`~qiskit.primitives`. Alternatively, you can also run
:func:`.transpile` followed by ``backend.run()``.
  job = execute(qc, backend, shots=1000)
```

```
from qiskit import QuantumCircuit, Aer, execute

qc = QuantumCircuit(20,3)

def adder_2(a, b, s, c):
    """Constructs a quantum circuit for a 2-qubit adder."""
    qc.ccx(a, b, s)
    qc.cx(a, c)
    qc.cx(b, c)

def adder_1(a, b, s, c):
    qc.cx(b, c)
    qc.cx(a, c)
    qc.ccx(a, b, s)

q0, q1, q2 = [0,0,1]

if q2==1:
  qc.x(15)
if q1 ==1:
  qc.x(12)
if q0 ==1 :
  qc.x(4)

qc.x(16)
qc.x(9)
qc.barrier()

adder_2(16, 15, 13, 14)
adder_2(12, 13, 10, 11)
adder_2(11,9,7,8)
adder_2(10,7,5,6)
adder_2(6,4,2,3)
adder_2(5,2,0,1)
qc.barrier()


qc.cx(14,17)
```

```python
qc.cx(8,18)
qc.cx(3,19)
qc.barrier()

adder_1(5,2,0,1)
adder_1(6,4,2,3)
adder_1(10,7,5,6)
adder_1(11,9,7,8)
adder_1(12, 13, 10, 11)
adder_1(16, 15, 13, 14)
qc.barrier()

qc.measure(17,0)
qc.measure(18,1)
qc.measure(19,2)

qc.draw('mpl')
```

```
/usr/local/lib/python3.10/dist-packages/qiskit/visualization/circuit/
matplotlib.py:266: FutureWarning: The default matplotlib drawer scheme
will be changed to "iqp" in a following release. To silence this
warning, specify the current default explicitly as style="clifford",
or the new default as style="iqp".
  self._style, def_font_ratio = load_style(self._style)
```


q0
q1
q2
q3
q4
q5
q6
q7
q8
q9
q10
q11
q12
q13
q14
q15
q16
q17
q18
q19
c
3
X
X
X


q0
q1
q2
q3
q4
q5
q6
q7
q8
q9
q10
q11
q12
q13
q14

```
# Choose the Aer simulator backend
backend = Aer.get_backend('qasm_simulator')

# Execute the circuit on the backend
job = execute(qc, backend, shots=1000)

# Get the result
result = job.result()

# Get the counts
counts = result.get_counts(qc)

# Print the counts
sum = next((key for key, value in counts.items() if value == 1000),
None)
print("="*40)
print(f'{0}{1}{1} + {q0}{q1}{q2} mod8 =',sum)
print("="*40)

========================================
011 + 001 mod8 = 100
========================================

<ipython-input-24-25af10de9fd2>:5: DeprecationWarning: The function
``qiskit.execute_function.execute()`` is deprecated as of qiskit
0.46.0. It will be removed in the Qiskit 1.0 release. This function
combines ``transpile`` and ``backend.run``, which is covered by
``Sampler`` :mod:`~qiskit.primitives`. Alternatively, you can also run
:func:`.transpile` followed by ``backend.run()``.
  job = execute(qc, backend, shots=1000)
```

#Problem 3

```
import cirq


def CSWAP(x0, x1, x2):
  q0, q1, q2 =cirq.LineQubit.range(3)

  qc = cirq.Circuit()

  if x2==1:
    qc.append(cirq.X(q2))
  if x1 ==1:
    qc.append(cirq.X(q1))
  if x0 ==1 :
    qc.append(cirq.X(q0))
```

```
  qc.append([cirq.CCNOT(q0,q1,q2),
cirq.CCNOT(q0,q2,q1),cirq.CCNOT(q0,q1,q2)])

  qc.append(cirq.measure(q0))
  qc.append(cirq.measure(q1))
  qc.append(cirq.measure(q2))


  simulator = cirq.Simulator()
  result = simulator.simulate(qc, initial_state=0)
  print('='*40)
  print()
  print()

  print(f'{x0} {x1} {x2} --->',result.measurements['q(0)']
[0],result.measurements['q(1)'][0],result.measurements['q(2)'][0])

  print()
  print()
  print('='*40)

print("checking possible inputs and outputs: ")
for x0 in range(2):
  for x1 in range(2):
    for x2 in range(2):
      CSWAP(x0, x1, x2)
```

```
checking possible inputs and outputs:
========================================


0 0 0 ---> 0 0 0


========================================
========================================


0 0 1 ---> 0 0 1


========================================
========================================


0 1 0 ---> 0 1 0


========================================
```

```
========================================

0 1 1 ---> 0 1 1

========================================
========================================

1 0 0 ---> 1 0 0

========================================
========================================

1 0 1 ---> 1 1 0

========================================
========================================

1 1 0 ---> 1 0 1

========================================
========================================

1 1 1 ---> 1 1 1

========================================
```