陳志偉 B06901126 電機五

Electrical Engineering Lab（topics on Communication System）
Lab1 Report

1. e) After apply 'X' gate (NOT gate) and draw it out (Circuit show as *Figure 1*) and plot it on the Bloch sphere (Show as *Figure 2*).

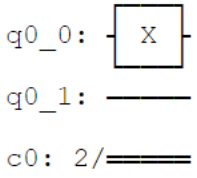


Figure 1

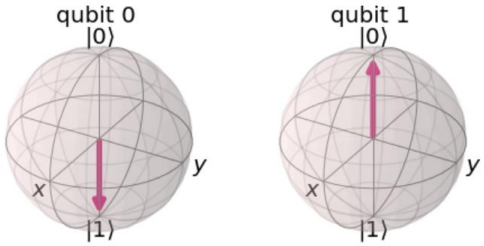


Figure 2

After that we need to verify the 'X' operation is equivalent to 'X ⊗ I'. 'X ⊗ I' operation circuit is show as *Figure 3*. If we plot it on Bloch sphere, we can get the same result as *Figure 2* and plot the measured result {'01': 1024} with plot_histogram() function (*Figure 4*).

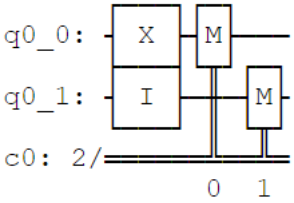


Figure 3

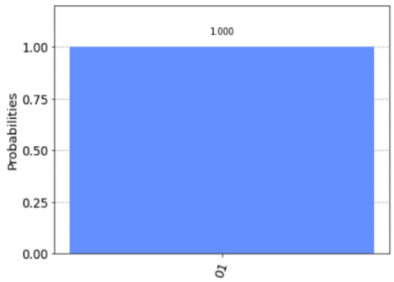


Figure 4

f) Get the result {'01': 517, '00': 507} as *Figure 6* after apply Hadamard gate (circuit show as *Figure 5*).

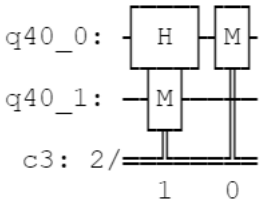


Figure 5

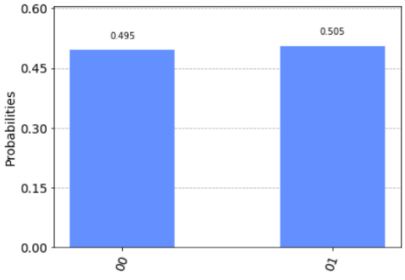


Figure 6

g) After apply 'R_x' gate (circuit show as *Figure 7*) and plot it on Bloch sphere (*Figure 8*). If we print out the statevector of result, we get [0.70710678+0.j 0.70710678+0.j 0. +0.j 0. +0.j].

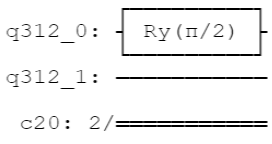


Figure 7

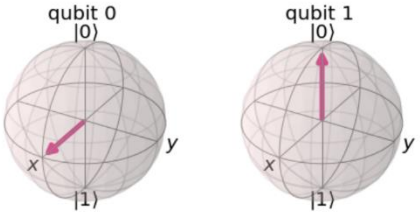


Figure 8

h) *Figure 9* show that apply a 'R_y' gate with -π/2 to a qubit, we can get the a qubit with obtaining $|-\rangle := \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$ and [0.70710678+0.00000000e+00j -0.70710678-8.65956056e-17j].

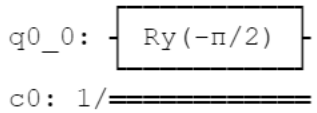


Figure 9

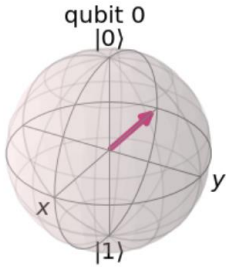


Figure 10

We also can get this by run `circuit.u3(pi/2, 0, 0, q)` and draw circuit as *Figure 11*.

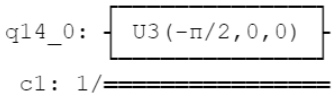


Figure 11

2. a) Get `{'01': 258, '10': 263, '11': 242, '00': 261}` after apply Hadamard gates H ⊗ H on the initial stat es $|0\rangle \otimes |0\rangle$. Circuit show as *Figure 12*.

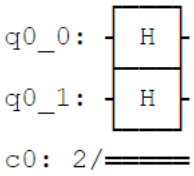


Figure 12

b) If the first classical register is '1', the outcome of the second classical register will be '1'. The pro bability of getting outcome '1' from the first register is almost 50%. Circuit show as *Figure 13*. *Figu er 14* show the result of after measure the circuit with `plot_histogram()` function.

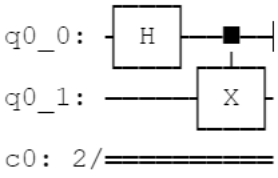


Figure 13

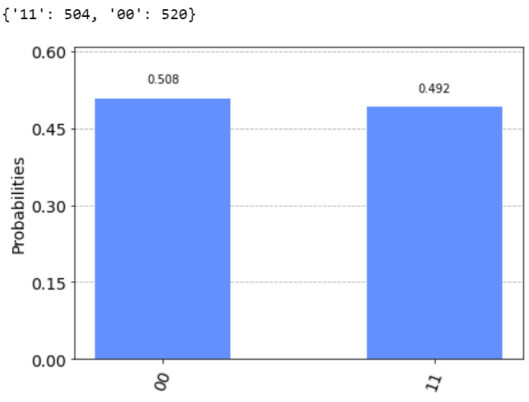


Figure 14

c) We will get the same result as 2. b). Circuit show as *Figure 15*.

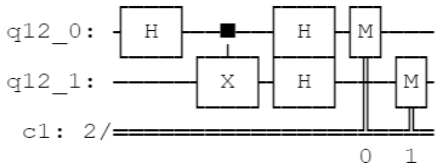


Figure 13

d)

$$|00\rangle \mapsto \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle) = |\Phi^+\rangle; \qquad |01\rangle \mapsto \frac{1}{\sqrt{2}}(|01\rangle + |10\rangle) = |\Psi^+\rangle;$$

$$|10\rangle \mapsto \frac{1}{\sqrt{2}}(|00\rangle - |11\rangle) = |\Phi^-\rangle; \qquad |11\rangle \mapsto \frac{1}{\sqrt{2}}(|01\rangle - |10\rangle) = |\Psi^-\rangle.$$

So we can just add 'X' gate to flip the qubit to get three maximally entangled state.

i. $|\Phi^-\rangle$: Add 'X' gate at first qubit. Circuit show as *Figure 16 a)* and justify by *Figure 16 b)*. Meas ure result show as *Figure 16 c)*
ii. $|\Psi^+\rangle$: Add 'X' gate at second qubit. Circuit show as *Figure 17a)* and justify by *Figure 17 b)*. M easure result show as *Figure 17 c)*
iii. $|\Psi^-\rangle$: Add 'X' gate at both qubits. Circuit show as *Figure 18 a)* and justify by *Figure 18 b)*. Me asure result show as *Figure 18 c)*

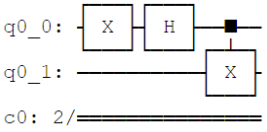


Figure 16 a)

```
[ 0.70710678+0.00000000e+00j  0.        +0.00000000e+00j
  0.        +0.00000000e+00j -0.70710678-8.65956056e-17j]
```

Figure 16 b)

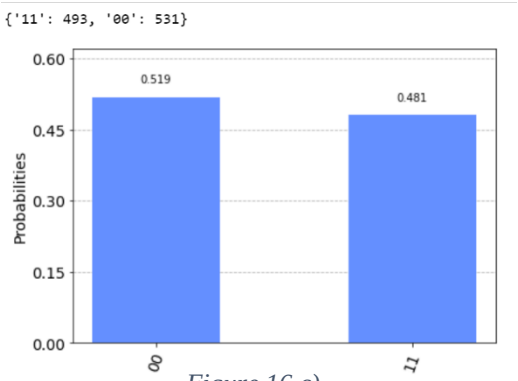


Figure 16 c)

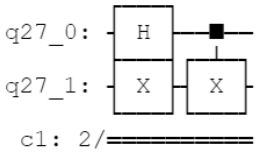


Figure 17 a)

```
[0.        +0.j 0.70710678+0.j 0.70710678+0.j 0.        +0.j]
```

Figure 17 b)

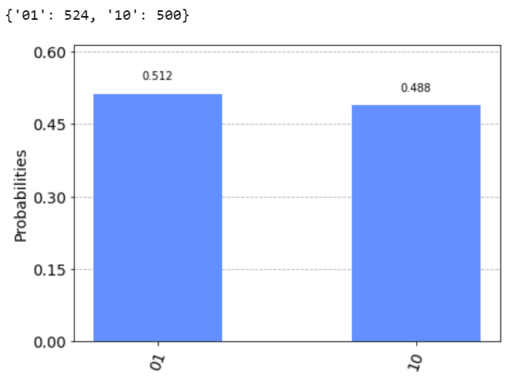


Figure 17 c)

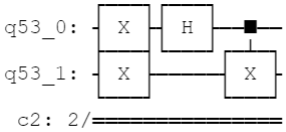


Figure 18 a)

```
[ 0.         +0.00000000e+00j -0.70710678-8.65956056e-17j
  0.70710678+0.00000000e+00j  0.         +0.00000000e+00j]
```

Figure 18 b)

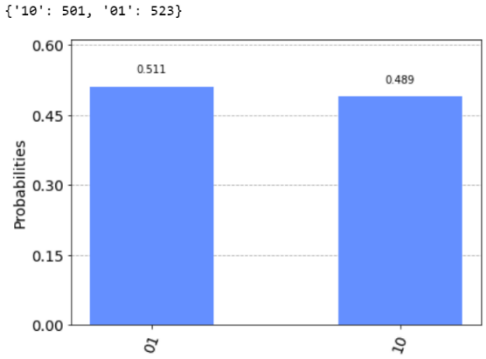


Figure 18 c)

e) ‘Swap’ gate circuit and result show as *Figure 19.*

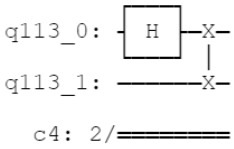


Figure 19 a)

```
[0.70710678+0.j 0.         +0.j 0.70710678+0.j 0.         +0.j]
```

Figure 19 b)

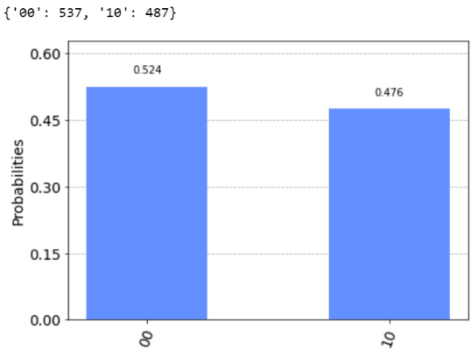
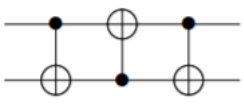


Figure 19 c)

Otherwise, *Figure 20* show the circuit of  by apply 3 ‘CX’ gate.

The circuit design by circuit.cx(q[0], q[1]), circuit.cx(q[1], q[0]), circuit.cx(q[0], q[1]).

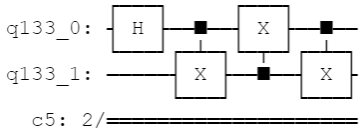


Figure 20 a)

```
[0.70710678+0.j 0.         +0.j 0.70710678+0.j 0.         +0.j]
```

Figure 20 b)

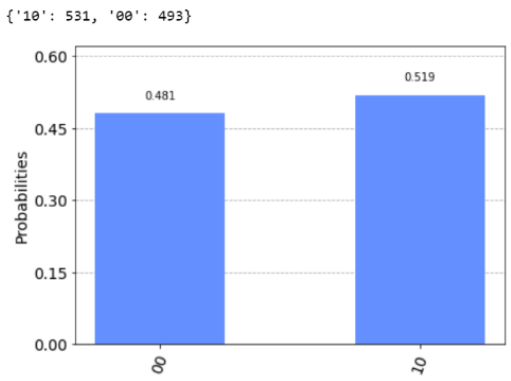


Figure 20 c)

3. a) We can exactly get the similar probability after the measure of this two circuit, but difference stat e vector. *Figure 21* show the simulation result of the first circuit. *Figure 22* show the result of secon d circuit.

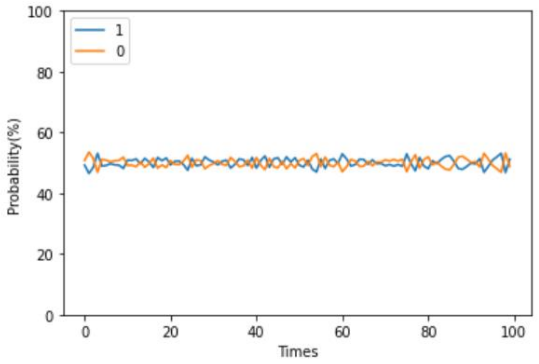
Figure 21

Figure 22

b) In this problem, I create a 100 time for loop with random of θ from 0 to pi. θ will be the first inpu t of circuit.u3() function. As the result show as *Figure 23*, we get a similar probability for every mea surement with random θ.

```
theta_arr = []
count_1 = []
count_0 = []

for i in range(100):
    q = QuantumRegister(1)
    c = ClassicalRegister(1)
    circuit = QuantumCircuit(q,c)

    circuit.h(q)
    theta = random.uniform(0, pi)
    circuit.u3(theta, -pi/2, pi/2,q)

    simulator = Aer.get_backend('statevector_simulator')
    job = execute(circuit, simulator)
    result = job.result()
    statevector = result.get_statevector()
    theta_arr.append(theta)


    circuit.measure(q, c)

    simulator = Aer.get_backend('qasm_simulator')
    job = execute(circuit, simulator, shots = 1024)
    result = job.result()
    counts = result.get_counts()
#     print(counts)

    count_1.append(counts['1']*100/1024)
    count_0.append(counts['0']*100/1024)
```

```
plt.plot(count_1, label="1")
plt.plot(count_0, label="0")

plt.ylabel('Probability(%)')
plt.xlabel('Times')
plt.ylim([0, 100])
plt.legend(loc="upper left")
plt.show()
```




```
plt.plot(theta_arr)
plt.show()
```

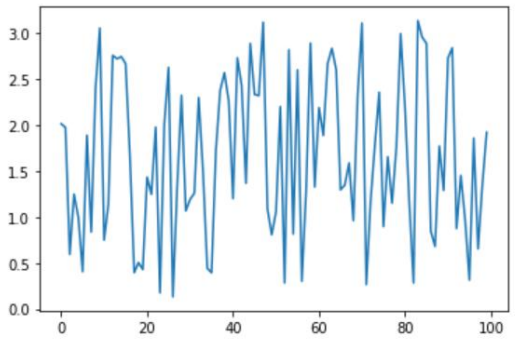


Figure 23

c) The measurement outcome as similar probability to measure ‘0’ and ‘1’. So we can consider them to equivalent, because we cannot distinguish them in the physical world.

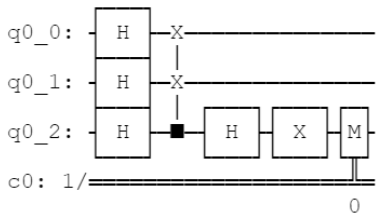
4. If both states are totally the same, the Swap test will measure only ‘1’. If not, the Swap test will have the probability to measure ‘0’ and ‘1’. *Figure 24* and *Figure 25* show the result of simulation .

In [1]:
```
import qiskit
import numpy as np
from qiskit import *
from qiskit.visualization import plot_histogram, plot_bloch_multivector
from math import sqrt, pi
```

In [2]:
```
q = QuantumRegister(3)
c = ClassicalRegister(1)
circuit = QuantumCircuit(q,c)
```

In [3]:
```
# circuit.x(q[0])
# circuit.x(q[1])

circuit.h(q[0])
circuit.h(q[1])

# circuit.rx(pi/4, q[0])
# circuit.rx(pi/4, q[1])

circuit.h(q[2])
circuit.cswap(q[2], q[1], q[0])
circuit.h(q[2])
circuit.x(q[2])

circuit.measure(q[2], c)
circuit.draw()
```

Out[3]:



In [4]:
```
simulator = Aer.get_backend('qasm_simulator')
job = execute(circuit, simulator, shots = 1024)
result = job.result()
counts = result.get_counts()
print(counts)
plot_histogram(counts)
```

```
{'1': 1024}
```

Out[4]:

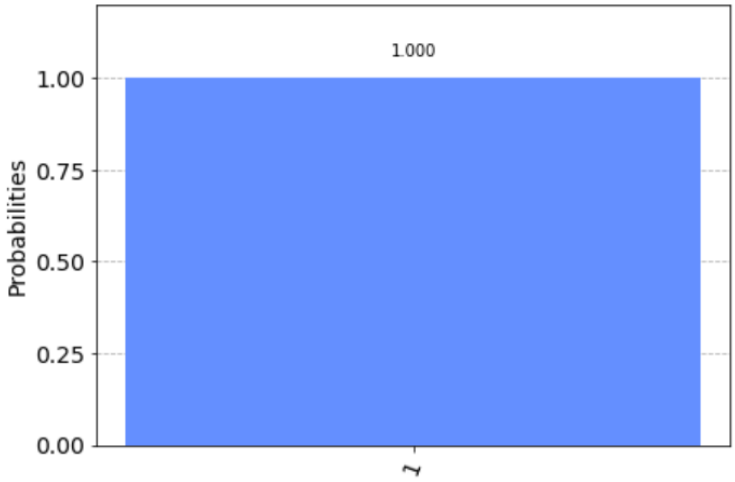


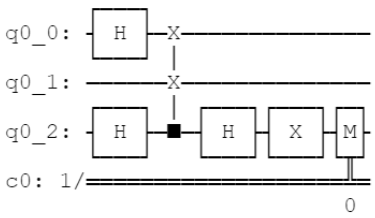
Figure 24

In [1]:
```
import qiskit
import numpy as np
from qiskit import *
from qiskit.visualization import plot_histogram, plot_bloch_multivector
from math import sqrt, pi
```

In [2]:
```
q = QuantumRegister(3)
c = ClassicalRegister(1)
circuit = QuantumCircuit(q,c)
```

In [3]:
```
# circuit.x(q[0])
# circuit.x(q[1])

circuit.h(q[0])
# circuit.h(q[1])

# circuit.rx(pi/4, q[0])
# circuit.rx(pi/4, q[1])

circuit.h(q[2])
circuit.cswap(q[2], q[1], q[0])
circuit.h(q[2])
circuit.x(q[2])

circuit.measure(q[2], c)
circuit.draw()
```

Out[3]:



In [4]:
```
simulator = Aer.get_backend('qasm_simulator')
job = execute(circuit, simulator, shots = 1024)
result = job.result()
counts = result.get_counts()
print(counts)
plot_histogram(counts)
```

```
{'0': 260, '1': 764}
```

Out[4]:

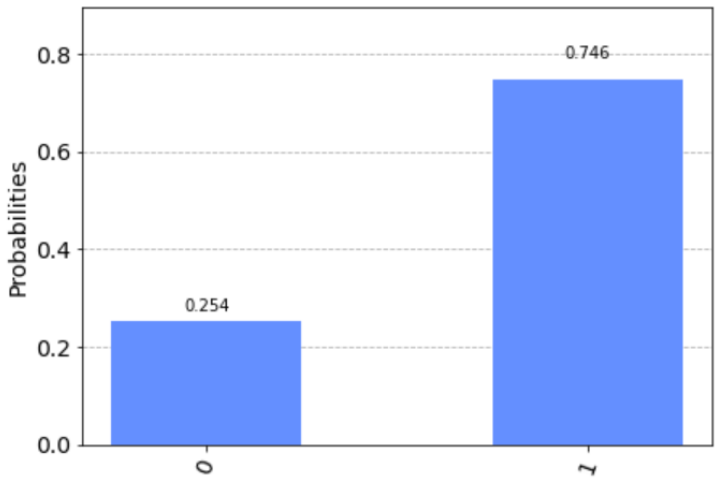


Figure 25

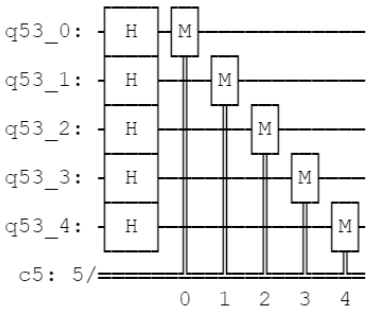
5. a) We can apply Hadamard gate to every qubit, then we can get the outcomes from 0 to 2^n-1 with similar probability, where n is the number of qubits. Figure 26 show the simulation by using 'qasm_simulator'. I think they are not really random. I set the ‘shots’ (arguments of the exexute() function) with 1024 and 8192, they come out with similar probability. So they are probability, not random. That means if you do it a large number of times, you will always get the result you want.

In [1]:
```
import qiskit
import numpy as np
from qiskit import *
from qiskit.visualization import plot_histogram, plot_bloch_multivector
from math import sqrt, pi
```

In [18]:
```
n = 5
q = QuantumRegister(n)
c = ClassicalRegister(n)
circuit = QuantumCircuit(q,c)
```

In [19]:
```
circuit.h(q)
circuit.measure(q, c)
circuit.draw()
```

Out[19]:



In [20]:
```
simulator = Aer.get_backend('qasm_simulator')
job = execute(circuit, simulator, shots = 8192)
result = job.result()
counts = result.get_counts()
plot_histogram(counts)
```

Out[20]:

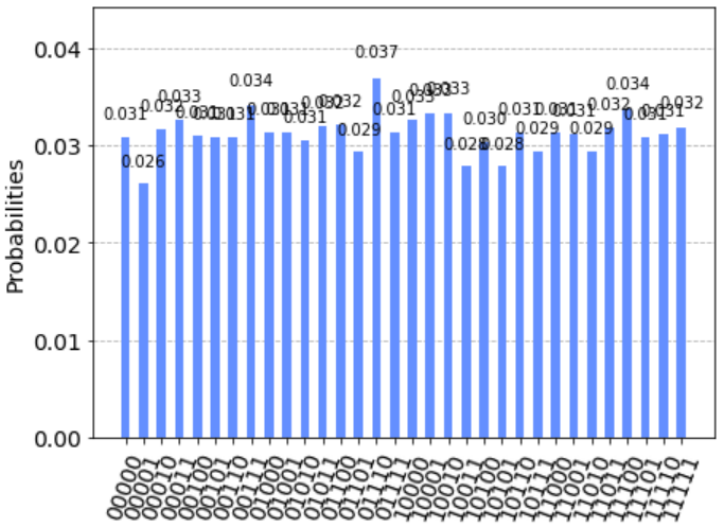


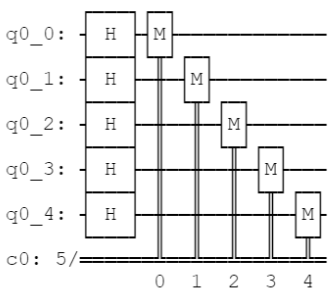
Figure 26

b) The result of run by real device of IBM Quantum Experience show as *Figure 27*. I think I have the same opinion as 5a).

In [2]:
```
n = 5
q = QuantumRegister(n)
c = ClassicalRegister(n)
circuit = QuantumCircuit(q,c)
```

In [3]:
```
circuit.h(q)
circuit.measure(q, c)
circuit.draw()
```

Out[3]:



In [4]:
```
IBMQ.load_account()
provider = IBMQ.get_provider(hub = 'ibm-q', group = 'open', project = 'main')
qcomp = provider.get_backend('ibmq_santiago')
job = execute(circuit, backend = qcomp, shots = 1000)

from qiskit.tools.monitor import job_monitor
job_monitor(job)

result = job.result()
counts = result.get_counts(circuit)

plot_histogram(counts)
```

```
Job Status: job has successfully run
```

Out[4]:

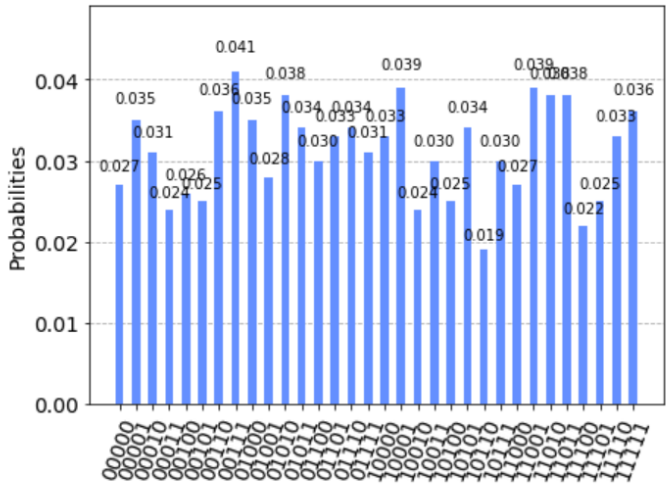


Figure 27