

## Experiment : 2

### CASE : 1

```
from qiskit import QuantumCircuit, transpile
from qiskit_aer import AerSimulator
from qiskit.visualization import plot_histogram

qc = QuantumCircuit(1, 1)
qc.x(0)
qc.z(0)
qc.h(0)
qc.y(0)
qc.s(0)
qc.measure(0, 0)

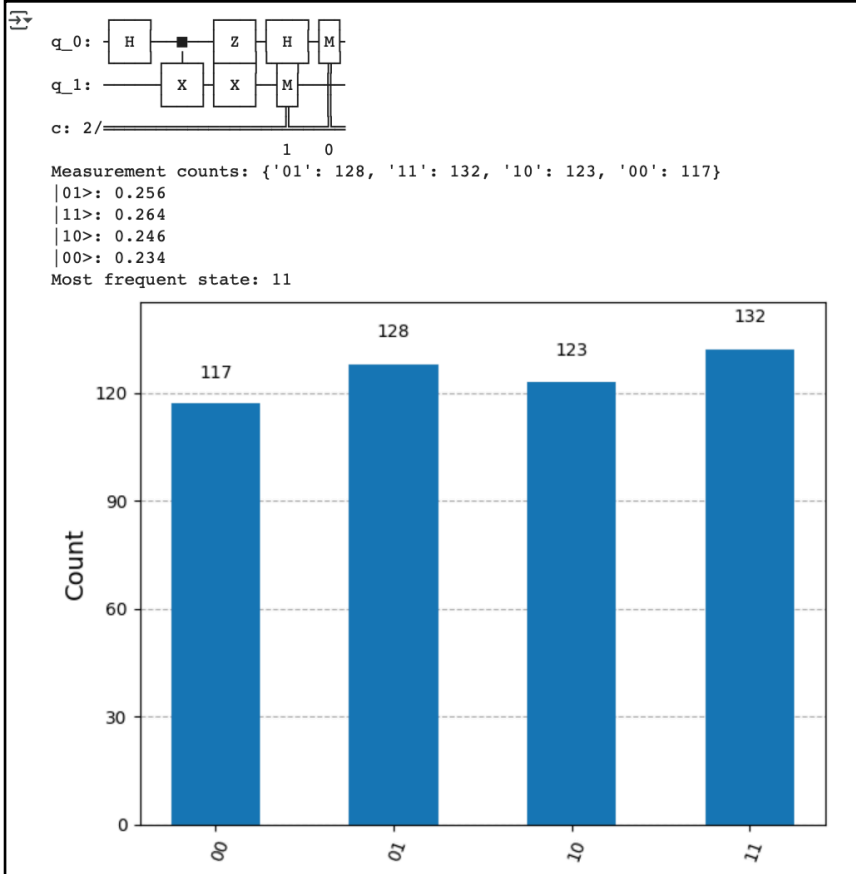
simulator = AerSimulator()
compiled_circuit = transpile(qc, simulator)
job = simulator.run(compiled_circuit, shots=100)
result = job.result()
counts = result.get_counts()

print(qc.draw())
print("Measurement counts:", counts)

total_shots = sum(counts.values())
print("Probabilities:")
for state, count in counts.items():
    print(f"State |{state}>: {count / total_shots:.3f}")

max_state = max(counts, key=counts.get)
print(f"Most frequent collapsed state after measurement: |{max_state}>")

plot_histogram(counts)
```



## CASE : 2

```
from qiskit import QuantumCircuit, transpile
from qiskit_aer import AerSimulator
from qiskit.visualization import plot_histogram

qc = QuantumCircuit(1, 1)
qc.x(0)
qc.z(0)
qc.h(0)
qc.y(0)
qc.s(0)
qc.measure(0, 0)

simulator = AerSimulator()
compiled_circuit = transpile(qc, simulator)
job = simulator.run(compiled_circuit, shots=100)
result = job.result()
counts = result.get_counts()

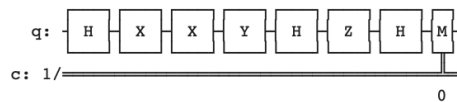
print(qc.draw())
print("Measurement counts:", counts)

total_shots = sum(counts.values())
print("Probabilities:")
for state, count in counts.items():
    print(f"State |{state}>: {count / total_shots:.3f}")

max_state = max(counts, key=counts.get)
print(f"Most frequent collapsed state after measurement: |{max_state}>")

plot_histogram(counts)
```

⇒



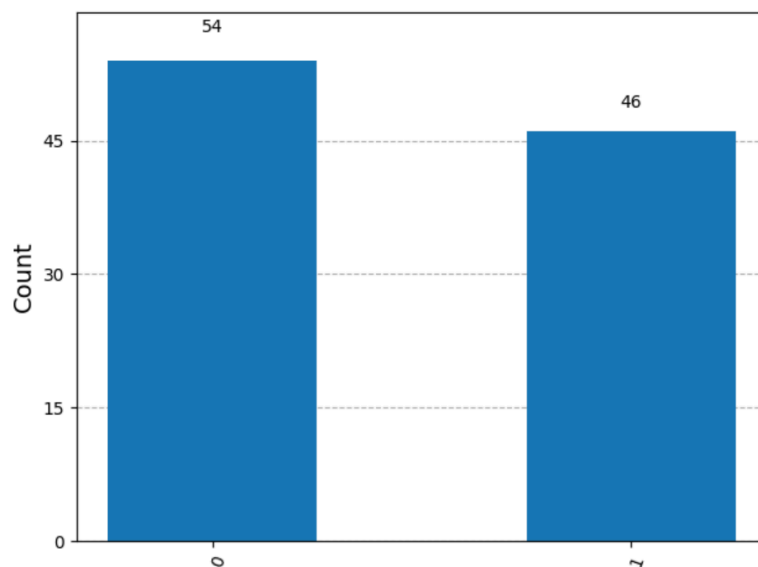
Measurement counts: {'1': 46, '0': 54}

Probabilities:

State |1>: 0.460

State |0>: 0.540

Most frequent collapsed state after measurement: |0>



## CASE 3 :

```
from qiskit import QuantumCircuit, transpile
from qiskit_aer import AerSimulator
from qiskit.visualization import plot_histogram

qc = QuantumCircuit(1, 1)
qc.h(0)
qc.x(0)
qc.x(0)
qc.h(0)
qc.z(0)
qc.h(0)
qc.y(0)
qc.h(0)
qc.measure(0, 0)

simulator = AerSimulator()
compiled_circuit = transpile(qc, simulator)
job = simulator.run(compiled_circuit, shots=100)

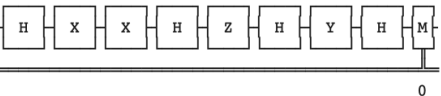
result = job.result()
counts = result.get_counts()

print(qc.draw())
print("Measurement counts:", counts)

total_shots = sum(counts.values())
print("Probabilities:")
for state, count in counts.items():
    print(f"State |{state}>: {count / total_shots:.3f}")

max_state = max(counts, key=count)
print(f"Most frequent collapsed state after measurement: |{max_state}>")

plot_histogram(counts)
```

q: 

Measurement counts: {'1': 100}

Probabilities:

State |1>: 1.000

Most frequent collapsed state after measurement: |1>

