BasicQCCircuit

April 17, 2024

[1]:
```python
import numpy as np
from qiskit import QuantumCircuit
```

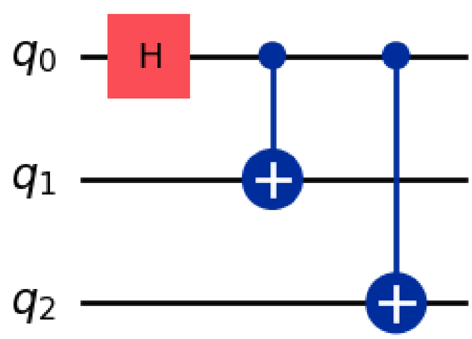
[2]:
```python
# Create a Quantum Circuit acting on a quantum register of three qubits
circ = QuantumCircuit(3)
```

[3]:
```python
# Add a H gate on qubit 0, putting this qubit in superposition.
circ.h(0)
# Add a CX (CNOT) gate on control qubit 0 and target qubit 1, putting
# the qubits in a Bell state.
circ.cx(0, 1)
# Add a CX (CNOT) gate on control qubit 0 and target qubit 2, putting
# the qubits in a GHZ state.
circ.cx(0, 2)
```

[3]:
```
<qiskit.circuit.instructionset.InstructionSet at 0x183abbdb1c0>
```

[4]:
```python
circ.draw('mpl')
```

[4]:



[5]:
```python
from qiskit.quantum_info import Statevector

# Set the initial state of the simulator to the ground state using from_int
```

```
state = Statevector.from_int(0, 2**3)

# Evolve the state by the quantum circuit
state = state.evolve(circ)

#draw using latex
state.draw('latex')
```

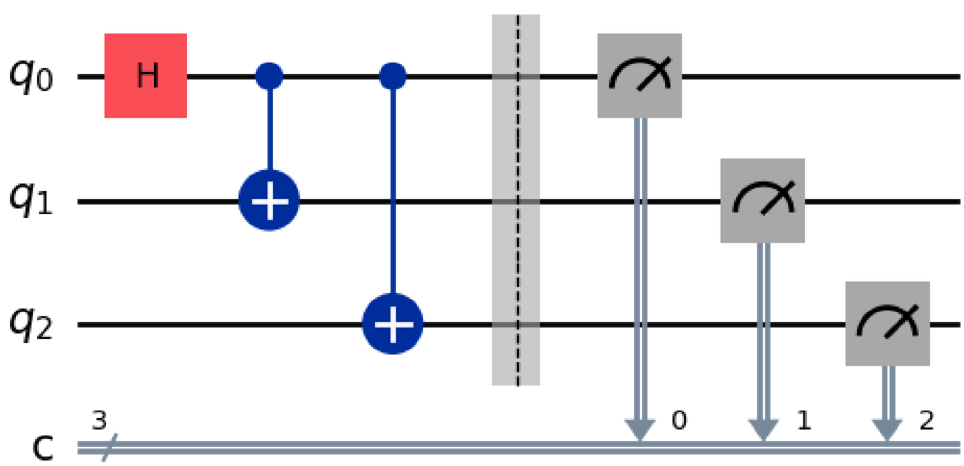
[5]:

$$\frac{\sqrt{2}}{2}|000\rangle + \frac{\sqrt{2}}{2}|111\rangle$$

[6]:

```
# Create a Quantum Circuit
meas = QuantumCircuit(3, 3)
meas.barrier(range(3))
# map the quantum measurement to the classical bits
meas.measure(range(3), range(3))

# The Qiskit circuit object supports composition.
# Here the meas has to be first and front=True (putting it before)
# as compose must put a smaller circuit into a larger one.
qc = meas.compose(circ, range(3), front=True)

#drawing the circuit
qc.draw('mpl')
```

[6]:



[7]:

```
# Adding the transpiler to reduce the circuit to QASM instructions
# supported by the backend
```

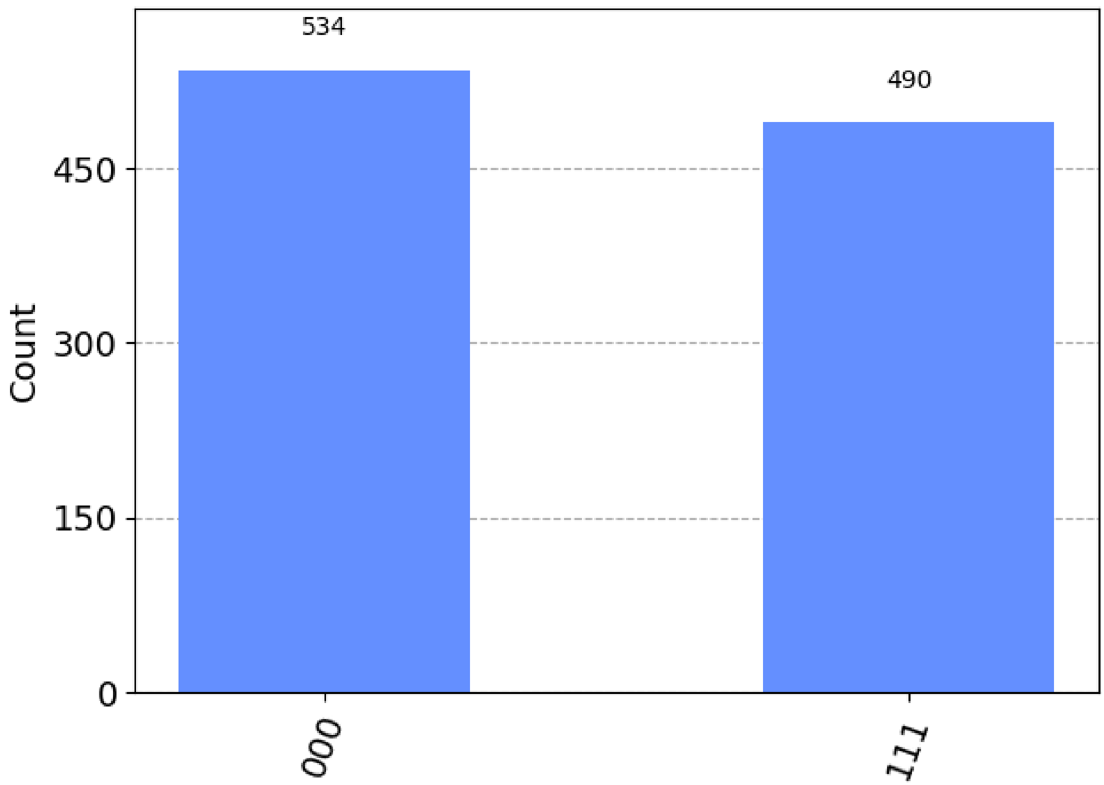
```python
from qiskit import transpile

# Use AerSimulator
from qiskit_aer import AerSimulator

backend = AerSimulator()

# First we have to transpile the quantum circuit
# to the low-level QASM instructions used by the
# backend
qc_compiled = transpile(qc, backend)

# Execute the circuit on the qasm simulator.
# We've set the number of repeats of the circuit
# to be 1024, which is the default.
job_sim = backend.run(qc_compiled, shots=1024)

# Grab the results from the job.
result_sim = job_sim.result()
```

[8]:
```python
counts = result_sim.get_counts(qc_compiled)
print(counts)
```

```
{'111': 490, '000': 534}
```

[9]:
```python
from qiskit.visualization import plot_histogram
plot_histogram(counts)
```

[9]:


534
490
Count
450
300
150
0
000
111

```
[ ]:
```