QuantumTeleportation_qiskit

December 30, 2020

[228]:
```
%matplotlib inline
# Importing standard Qiskit libraries
import numpy as np
from qiskit import *
from qiskit.compiler import transpile, assemble
from qiskit.tools.jupyter import *
from qiskit.extensions import Initialize
from qiskit.visualization import *
from qiskit.tools.monitor import job_monitor
from iqx import *
from qiskit_textbook.tools import random_state, array_to_latex
#from numpy import random_state
# Loading your IBM Q account(s)
provider = IBMQ.load_account()
```

```
ibmqfactory.load_account:WARNING:2020-12-30 08:04:37,835: Credentials are
already in use. The existing account in the session will be replaced.
```

1 Defining the Circuit

We will first define the circuit in the following block of code

1.1 Make a Random State

In the following bit of code, we will initialize q_0 with a random state and then transport that onto q_2

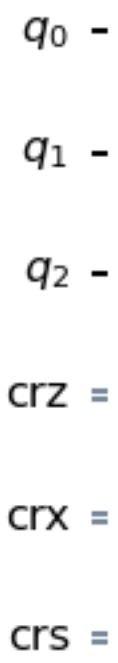
[229]:
```
# Create random 1-qubit state
psi = random_state(1)

# Display it nicely
array_to_latex(psi, pretext="|\\psi\\rangle =")
# Show it on a Bloch sphere
plot_bloch_multivector(psi)
init_gate = Initialize(psi)
init_gate.label = "init"
```

$$|\psi\rangle = \begin{bmatrix} -0.78485 + 0.29156i \\ -0.44767 + 0.314i \end{bmatrix}$$

[230]:
```
quantReg = QuantumRegister(3, name = 'q')
crz = ClassicalRegister(1, name = 'crz')
crx = ClassicalRegister(1, name = 'crx')
cRes = ClassicalRegister(1, name = 'crs')
qCirc = QuantumCircuit(quantReg, crz, crx, cRes)
qCirc.draw()
```

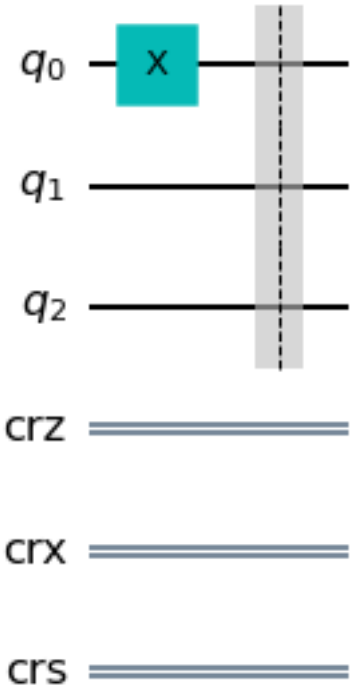
[230]:



Here, we will set the state of $ q_0 $ to 1 and then teleport that bit to $ q_2 $

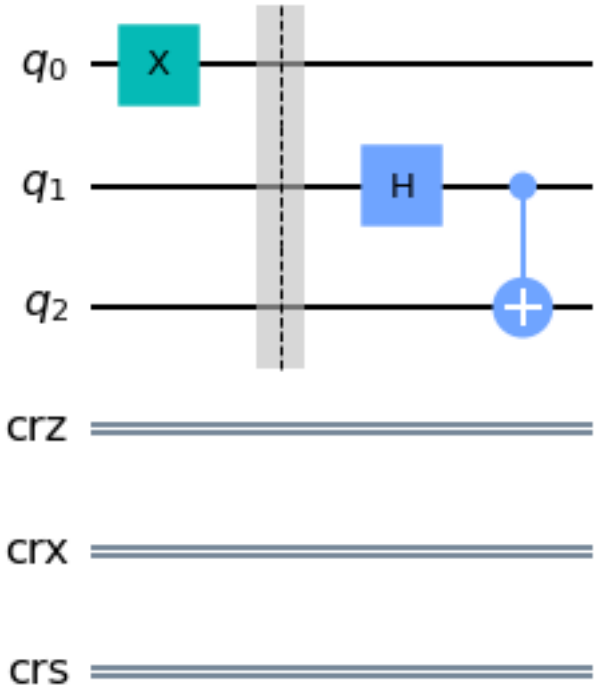
[231]:
```
#qCirc.append(init_gate, [0])
qCirc.x(0)
qCirc.barrier()
qCirc.draw()
```

[231]:



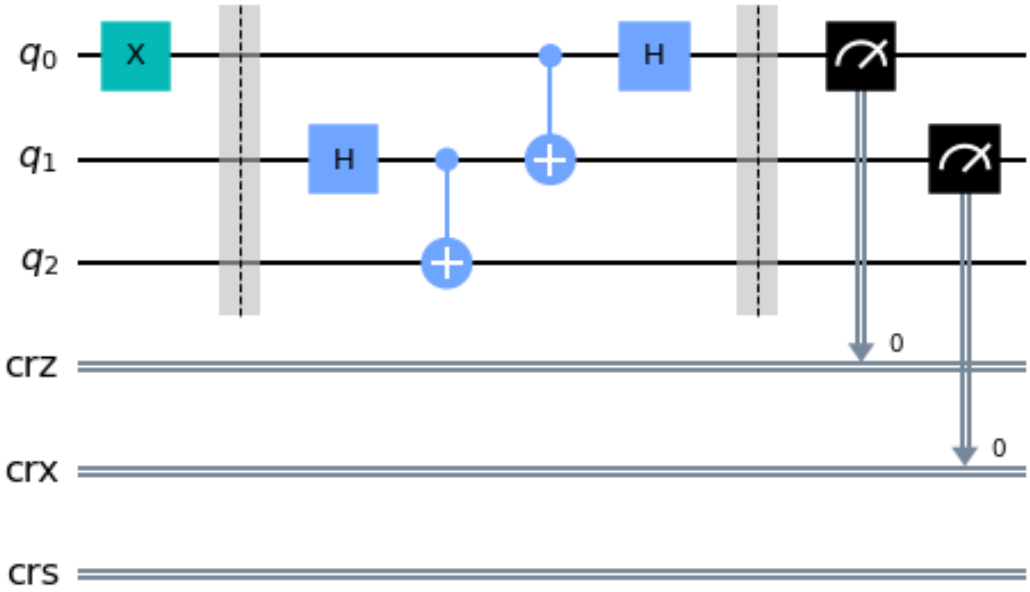
[232]:
```
qCirc.h(1)
qCirc.cx(1,2)
qCirc.draw()
```

[232]:



[233]:
```
qCirc.cx(0,1)
qCirc.h(0)
qCirc.barrier()
qCirc.measure([0,1],[0,1])
qCirc.draw()
```

[233]:



[234]:
```
qCirc.barrier()
```

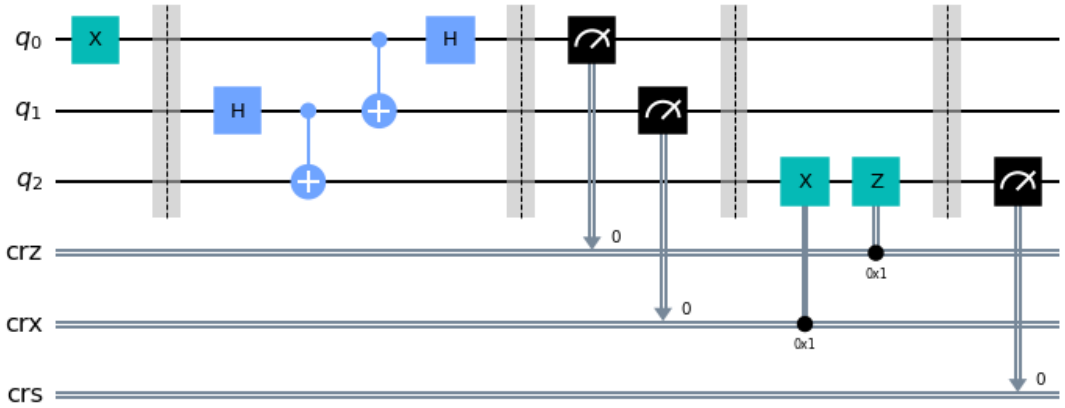
[234]: <qiskit.circuit.instructionset.InstructionSet at 0x7f174e2ca610>

This function is a funtion on the `intermediate' $ q_1 $ qubit so that we can transport the state to $ q_2 $.

[235]:
```
def measurementAction(qCirc, qubit, crz, crx) :
    qCirc.x(qubit).c_if(crx, 1)
    qCirc.z(qubit).c_if(crz, 1)
```

[236]:
```
measurementAction(qCirc, quantReg[2], crz, crx)
qCirc.barrier()
qCirc.measure(2,cRes)
qCirc.draw()
```

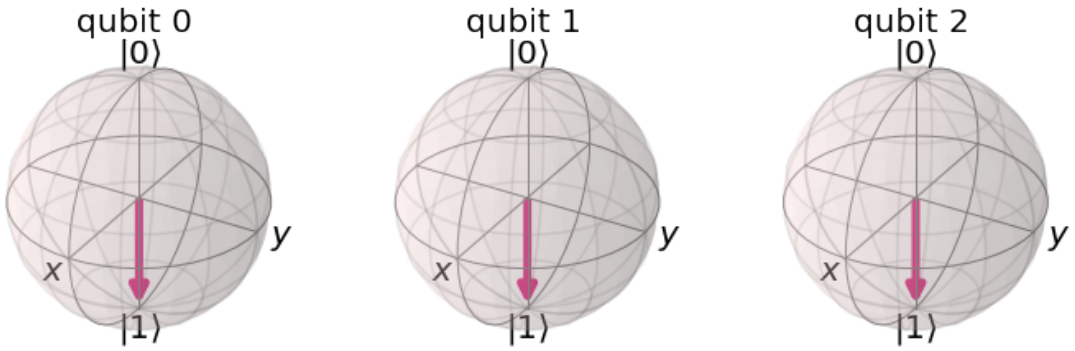
[236]:



[237]:
```
svBackend = Aer.get_backend('statevector_simulator')
countsBackend = Aer.get_backend('qasm_simulator')
resultSV = execute(qCirc, svBackend).result()
resultCounts = execute(qCirc, countsBackend).result()
counts = resultCounts.get_counts()
stateVector = resultSV.get_statevector()
print(stateVector)
plot_bloch_multivector(stateVector)
```

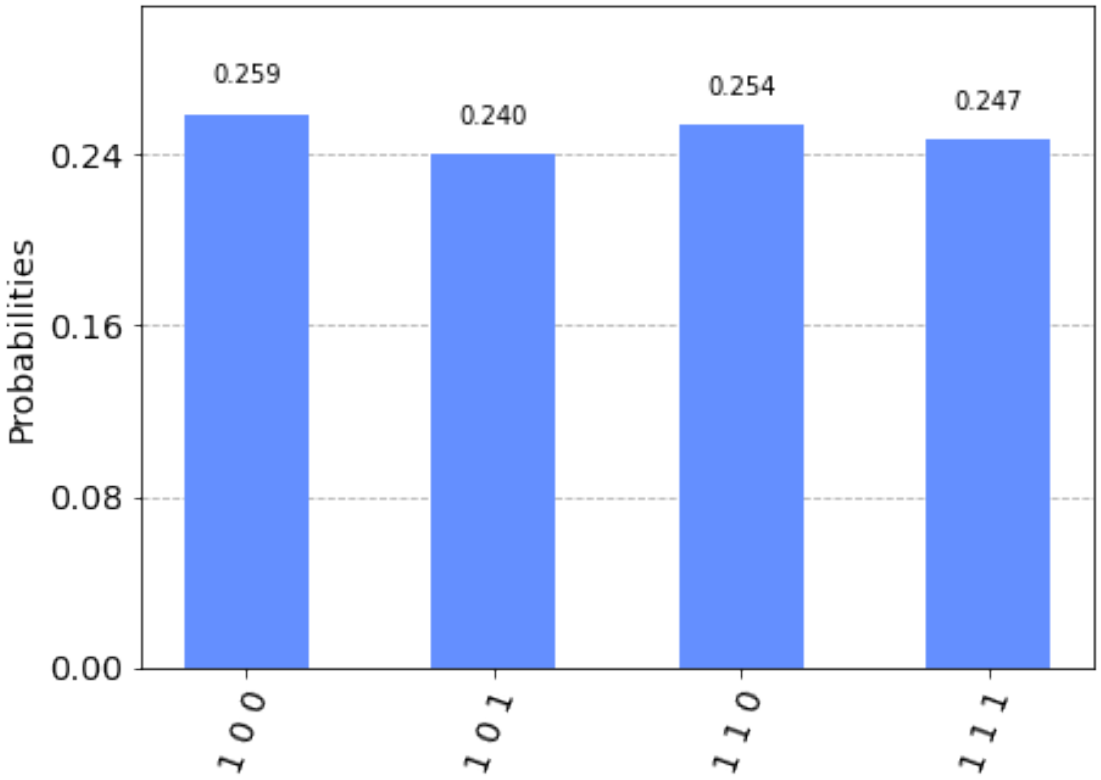
```
[ 0.+0.0000000e+00j  0.+0.0000000e+00j  0.+0.0000000e+00j
  0.+0.0000000e+00j -0.+0.0000000e+00j -0.+0.0000000e+00j
 -0.+0.0000000e+00j  1.-1.2246468e-16j]
```

[237]:



[238]:
```
plot_histogram(counts)
```

[238]:



This is a successfull run since the state of q_2 is now 1 in all the cases.

1.2 Running on an IBM Quantum Computer

[239]:
```
IBMQ.load_account()
provider = IBMQ.get_provider(hub='ibm-q')
```

```
/opt/conda/lib/python3.7/site-packages/qiskit/providers/ibmq/ibmqfactory.py:192:
UserWarning: Timestamps in IBMQ backend properties, jobs, and job results are
all now in local time instead of UTC.
  warnings.warn('Timestamps in IBMQ backend properties, jobs, and job results '
ibmqfactory.load_account:WARNING:2020-12-30 08:04:43,206: Credentials are
already in use. The existing account in the session will be replaced.
```

[240]:
```
# First, see what devices we are allowed to use by loading our saved accounts

# get the least-busy backend at IBM and run the quantum circuit there
from qiskit.providers.ibmq import least_busy
backend = least_busy(provider.backends(filters=lambda b: b.configuration().
 ↪n_qubits >= 3 and
```

```
                                    not b.configuration().simulator and b.
 ↪status().operational==True))
job_exp = execute(qc, backend=backend, shots=256)
job_monitor(job_exp)
```

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

[241]:
```
exp_result = job_exp.result()
exp_measurement_result = exp_result.get_counts()
print(exp_measurement_result)
plot_histogram(exp_measurement_result)
```

```
{'0 0': 256}
```

[241]: