Berstein-Vazirani_qiskit

December 23, 2020

1 Berstein-Vazirani Algorithm

This implementation of the Berstein-Vazirani model will deal with n-qubit black boxes and is specifically designed as a game. The user will input any binary string and the circuit implementing the algorithm will guess it using only 1 query.

[161]:
```
from qiskit import *
from qiskit.tools.visualization import plot_histogram
from qiskit import IBMQ
from qiskit.tools.monitor import job_monitor
```

1.1 User Input

Please type your number here :

[162]:
```
userNum = 7
binaryConvert = bin(userNum)[2:]
noOfBits = len(binaryConvert)
print(binaryConvert)
```

```
111
```

2 Quantum Variant Implementation

2.1 Defining the Circuit

[163]:
```
qReg = QuantumRegister(noOfBits + 1, name = 'q')
cReg = ClassicalRegister(noOfBits, name = 'cr')
qCirc = QuantumCircuit(qReg, cReg)
```

2.2 Initial Hadamards

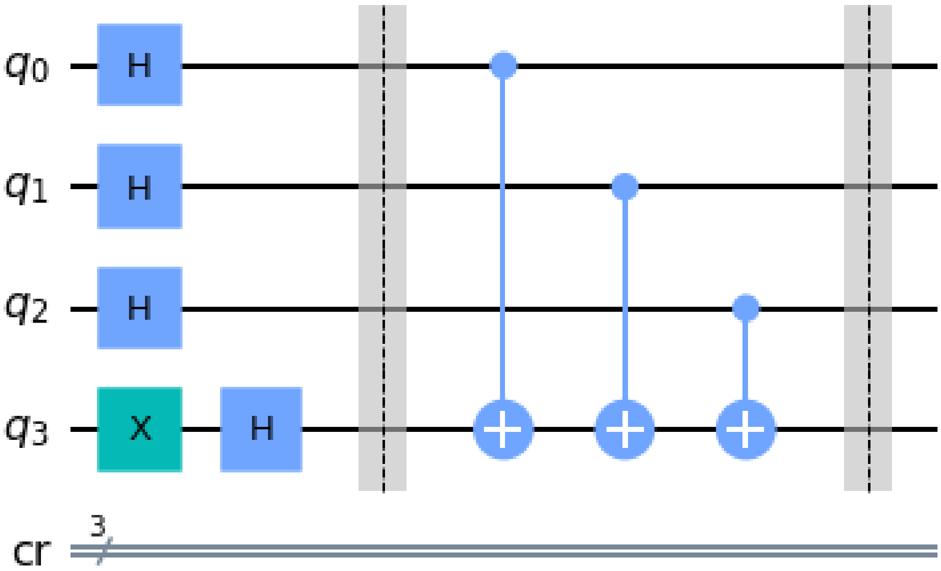
[164]:
```
qCirc.x(noOfBits)
for i in range(0,noOfBits+1) :
    qCirc.h(i)
qCirc.barrier()
```

[164]:
```
<qiskit.circuit.instructionset.InstructionSet at 0x1e8c545dd30>
```

2.3 Querying the Oracle

[165]:
```
for i in range(0,noOfBits) :
    if(binaryConvert[noOfBits - 1 - i] == '1') :
        qCirc.cx(i, noOfBits)
qCirc.barrier()
qCirc.draw('mpl')
```

[165]:



2.4 Final Hadamards

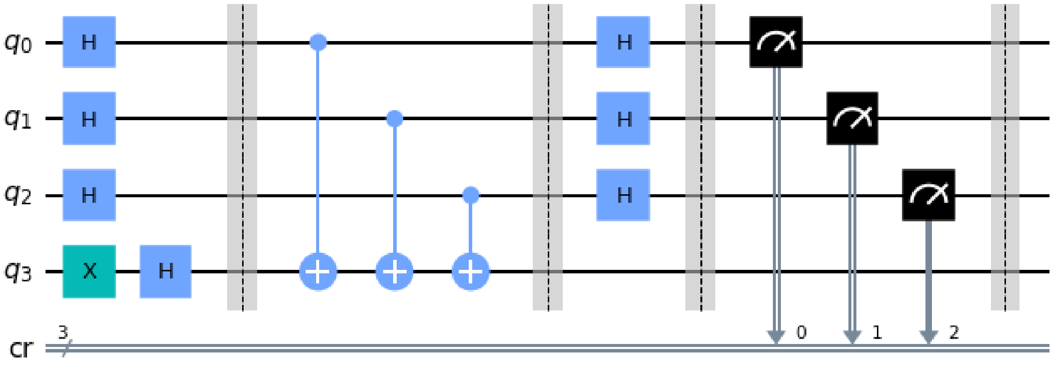
[166]:
```
for i in range(0,noOfBits) :
    qCirc.h(i)
qCirc.barrier()
```

[166]:
```
<qiskit.circuit.instructionset.InstructionSet at 0x1e8c545faf0>
```

2.5 Measuring the Circuit

[167]:
```
for i in range(0,noOfBits) :
    qCirc.measure(qReg[i],cReg[i])
qCirc.barrier()
qCirc.draw('mpl')
```

[167]:



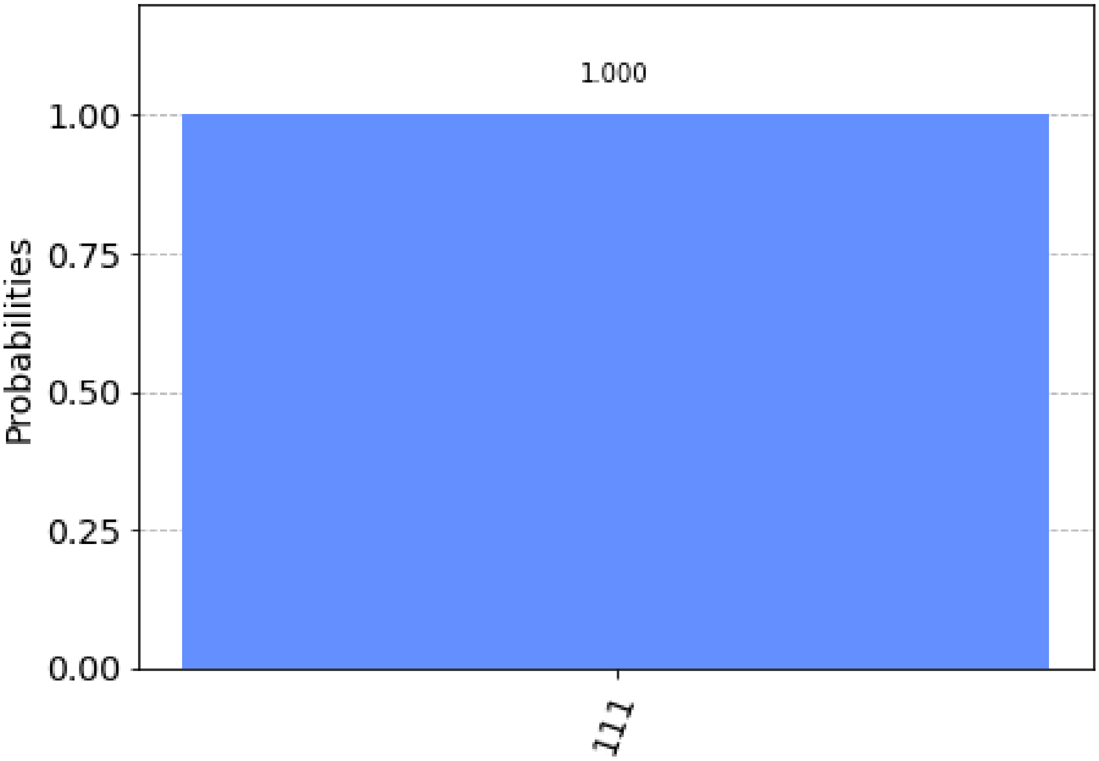
2.6 Simulating Natively

[168]:

```
nativeSim = Aer.get_backend('qasm_simulator')
nativeResult = execute(qCirc, backend = nativeSim, shots = 4096).result()
counts = nativeResult.get_counts()
print(counts)
plot_histogram(counts)
```

```
{'111': 4096}
```

[168]:



SUCCCESSSS!!! We got it done in 1 shot. Yay!

2.7 Simulating on a Real Quantum Computer

```
[169]: IBMQ.load_account()
```

```
ibmqfactory.load_account:WARNING:2020-12-23 18:30:43,903: Credentials are
already in use. The existing account in the session will be replaced.
```

```
[169]: <AccountProvider for IBMQ(hub='ibm-q', group='open', project='main')>
```

```
[175]: qProvider = IBMQ.get_provider()
       qComp = qProvider.get_backend('ibmq_valencia')
       job = execute(qCirc, backend = qComp)
       job_monitor(job)
```

```
Job Status: job incurred error
```

```
[172]: realCounts = job.result().get_counts(qCirc)
       plot_histogram(realCounts)
```

```
---------------------------------------------------------------------------
IBMQJobFailureError                       Traceback (most recent call last)
<ipython-input-172-04e93114b7a3> in <module>
----> 1 realCounts = job.result().get_counts(qCirc)
      2 plot_histogram(realCounts)

~\anaconda3\lib\site-packages\qiskit\providers\ibmq\job\ibmqjob.py in␣
 ↪result(self, timeout, wait, partial, refresh)
    287                 else:
    288                     error_message = ": " + error_message
--> 289                 raise IBMQJobFailureError(

    290                     'Unable to retrieve result for job {}. Job has␣
 ↪failed{}'.format(
    291                         self.job_id(), error_message))

IBMQJobFailureError: 'Unable to retrieve result for job 5fe35497a6f2af001ab8cf3'.
 ↪ Job has failed: Qubit measurement not the final instruction.. Error code:␣
 ↪7006.'
```

3 Classical Variant Implementation

```
[ ]:
```