Assignment 318

Ali Hamza

April 1, 2020

Question 1

If a is an n-bit string then there are 2^n possible values of a. Each value of a has a certain number of b's such that $a \cdot b = 1$ and $a \cdot b = 0$ for the rest.

$$a \cdot b = a_1 \oplus b_1 \oplus a_2 \oplus b_2 \oplus a_3 \oplus b_3 \oplus ... \oplus a_n \oplus b_n$$

According to the definition of $a \cdot b$ the following cases are possible:

- $a \cdot b = 0$ a & b have an even number of 1's in the same positions
- $a \cdot b = 1$ otherwise

Assuming that a string has an k number of 1's then there are 2^k number of possible combinations of those 1's and since there will, resultantly be $n - k$ number of 0's there will be 2^{n-k} number of possible combinations of those 0's. This can be verified as the total number of ways the 1's and 0's can be chosen is $2^{n-k} \cdot 2^k = 2^n$

Out of the 2^k ways, half of combinations will have an even number of 1's therefore the total number of combinations with an odd number of 1's is $2^{k-1} \cdot 2^{n-k}$

Therefore, the probability of choosing an even or odd number of ones equals can be found by divinding the total number of ways to get an even or odd number of 1's by the total number of ways we can choose an n-bit string:

$$\frac{2^{k-1} \cdot 2^{n-k}}{2^n} = \frac{2^{n-1}}{2^n} = \frac{1}{2}$$

Therefore,

$$P(a \cdot b = 1) = P(a \cdot b = 0) = \frac{1}{2}$$

Discussed with: Shehryar Mughal

Question 2

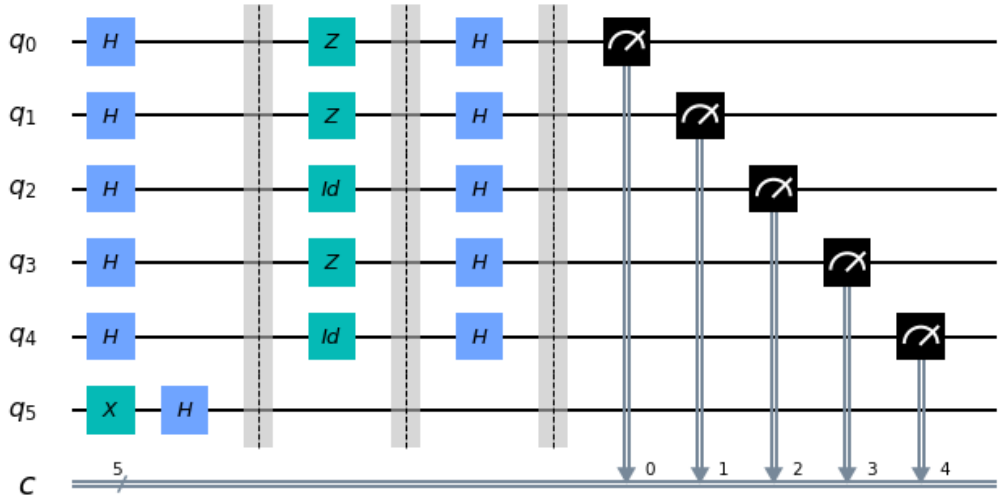
[16]:
```
from qiskit import *
from qiskit.providers.aer import QasmSimulator
# also import a visualization tool from qiskit
from qiskit.tools.visualization import plot_histogram
%matplotlib inline
```

[27]:
```
class BVn():

    def __init__(self, A:str, Id = False):

        self.a = A
        self.s = int(A, 2)
        self.qcirc = QuantumCircuit(len(self.a)+1, len(self.a))
        self.qcirc.h(range(len(self.a)))
        self.qcirc.x(len(self.a))
        self.qcirc.h(len(self.a))
        self.qcirc.barrier()

        if Id: #If Id is true, gate identities are used to build the black box␣
 ↪circuit
            for i in range(len(self.a)):
                if (self.s & (1 << i)):
                    self.qcirc.z(i)
                else:
                    self.qcirc.iden(i)
        else: #If Id is false, cnot gates are used to build the black box circuit
            for i, cx in enumerate(reversed(self.a)):
                if cx == "1":
                    self.qcirc.cx(i, len(self.a))


        self.qcirc.barrier()
        self.qcirc.h(range(len(self.a)))
        self.qcirc.barrier()
        self.qcirc.measure(range(len(self.a)),range(len(self.a)))

    def sim(self):
        self.simulator = Aer.get_backend('qasm_simulator')
        self.job = execute(self.qcirc,self.simulator,shots=1)
        self.result = self.job.result()
        self.counts = self.result.get_counts()
        print("The secret number in decimal is ", self.s, "=" , self.counts)
```

[28]:
```
bv_1 = BVn("01011", True)
bv_1.sim()
bv_1.qcirc.draw(output = 'mpl')
```

```
The secret number in decimal is  11 = {'01011': 1}
```

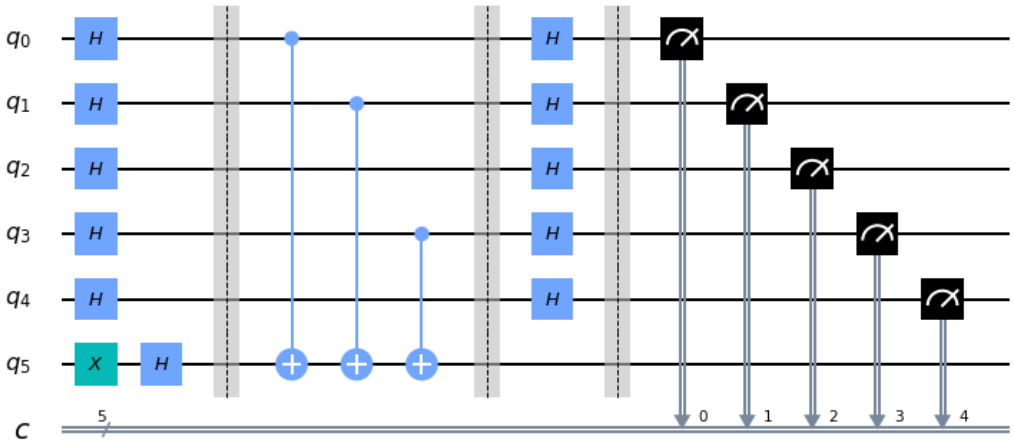
[28]:



[29]:
```
bv_1 = BVn("01011")
bv_1.sim()
bv_1.qcirc.draw(output= 'mpl')
```

```
The secret number in decimal is  11 = {'01011': 1}
```

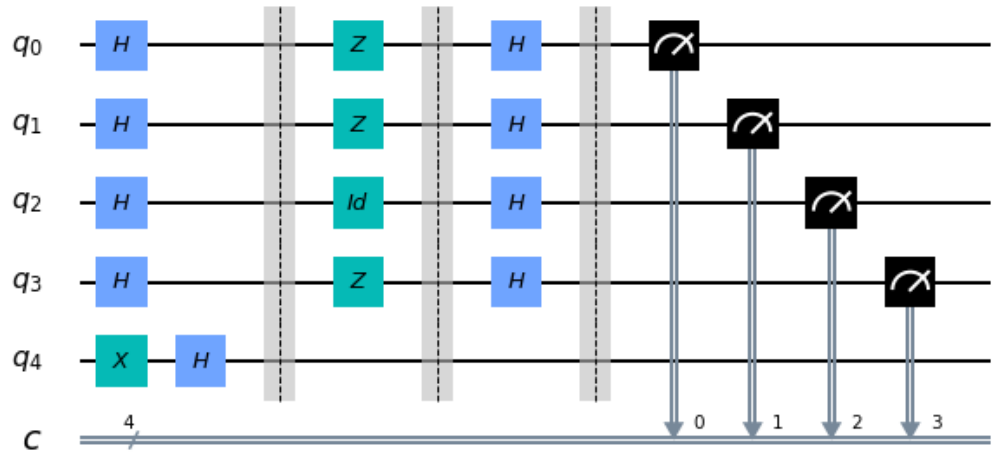
[29]:



[30]:
```
bv_2 = BVn("1011", True)
bv_2.sim()
bv_2.qcirc.draw(output = 'mpl')
```

```
The secret number in decimal is  11 = {'1011': 1}
```

[30]:



[31]:
```
bv_2 = BVn("1011")
bv_2.sim()
bv_2.qcirc.draw(output = 'mpl')
```

```
The secret number in decimal is  11 = {'1011': 1}
```

[31]: