# Implementación de Algoritmo de Deutsch y Deutsch-Jozsa

**JUAN ALEJANDRO SOLARTE MONCADA**

Escuela Colombiana De Ingeniería Julio Garavito
Ciencias Naturales y Tecnología

**Juan.solarte-m@mail.escuelaing.edu.co**

**27/11/2022**

*Este reporte se entrega para cumplir con los requisitos parciales del curso CNYT: Computación Cuántica- 2020-1*

## Introducción

En este informe veremos como la computación cuántica puede dar pasos por una computación clásica desde el programa de Python. Viendo cómo se comportan los siguientes algoritmos de Deutsch y Deutsch-Jozsa, siendo estos de los primeros diseños algorítmicos para la computación cuántica y viendo cómo se superponen en los estados cuánticos

## Algoritmo Deutsch

En esta parte se implementa el código de Deutsch para las 4 funciones que se pueden formar, en cada uno se mostrara su función, su matriz, la lógica del circuito y si es una función balanceada o constante contruyendolo en Python.

Función #1:

- Dibujo de la función:

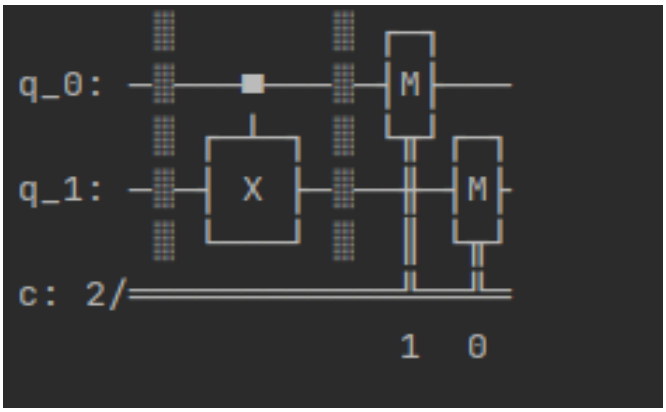


- Matríz: $Uf =$

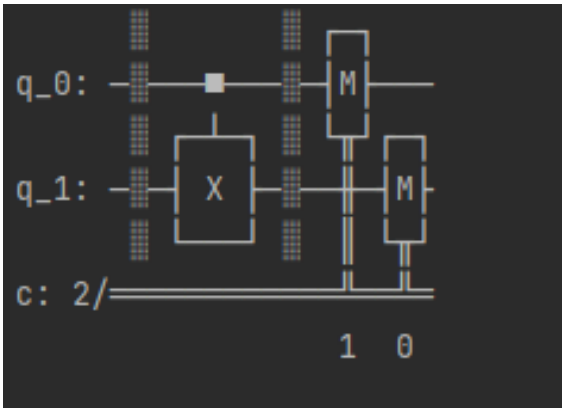
| | | | | |
|---|---|---|---|---|
| 00 | 1 | 0 | 0 | 0 |
| 01 | 0 | 1 | 0 | 0 |
| 10 | 0 | 0 | 0 | 1 |

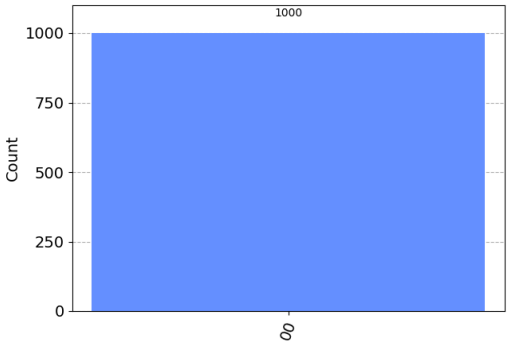
| | | | | |
|---|---|---|---|---|
| 11 | 0 | 0 | 1 | 0 |

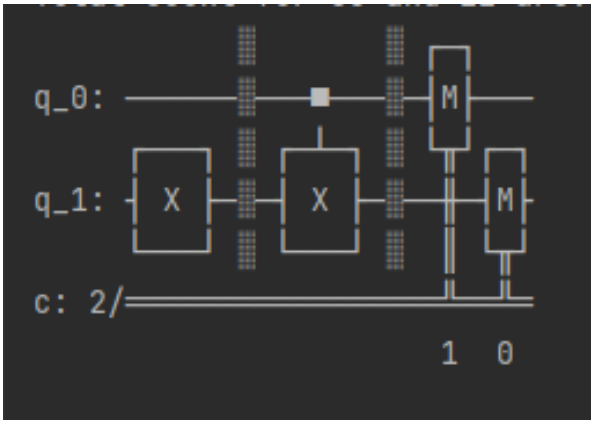
- Circuito:

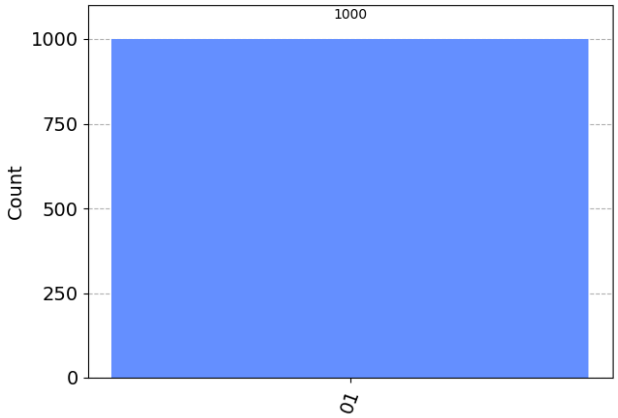



- Pruebas:
    a. Con 00 y 01

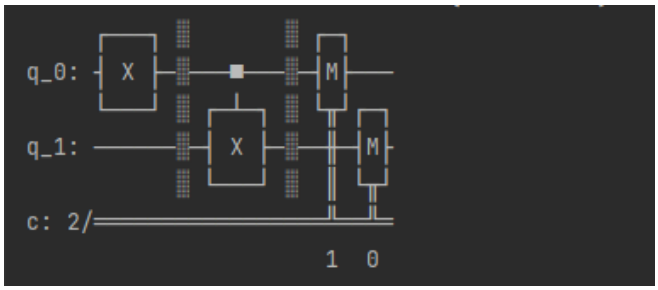


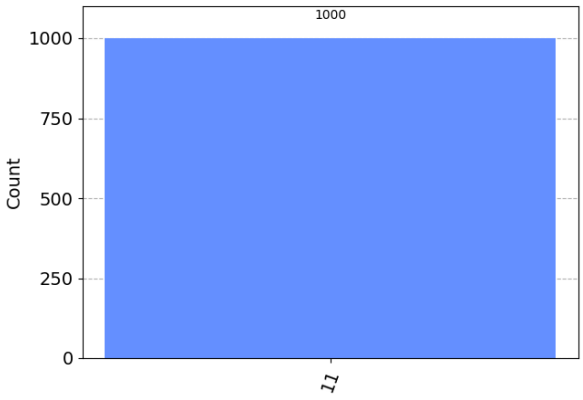



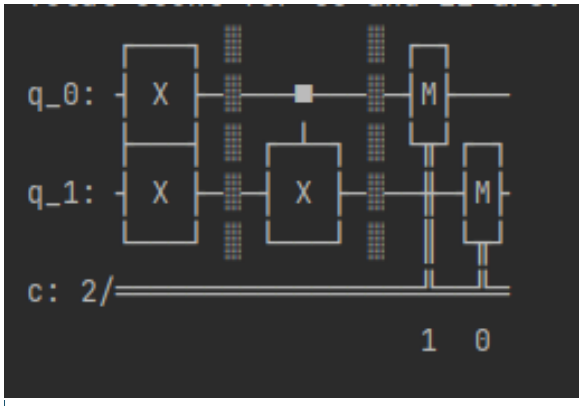

q_0:
q_1:
c: 2/
X
X
M
M
1 0

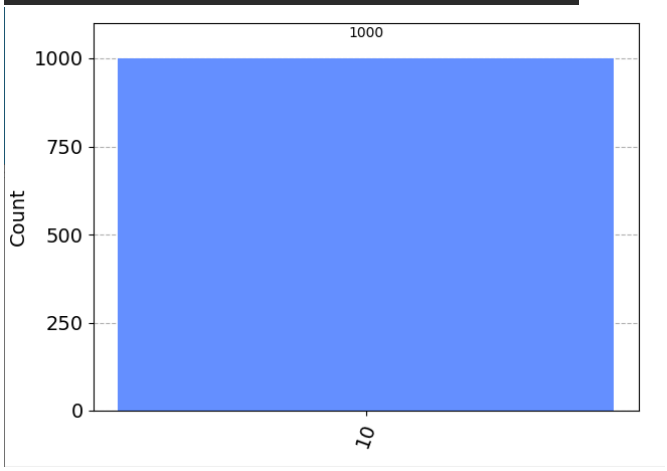

1000
1000
750
500
250
0
Count
01

b. Con 10 y 11.

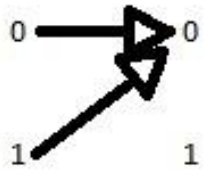

q_0:
q_1:
c: 2/
X
X
M
M
1 0


1000
1000
750
500
250
0
Count
11



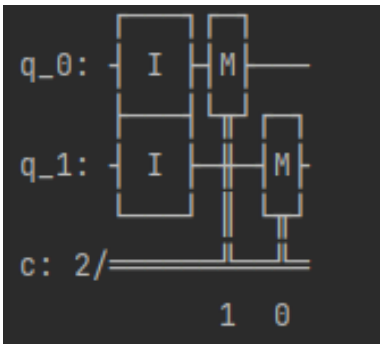

2. Función #2:
    - Dibujo de la función:




    - Matríz: $Uf =$

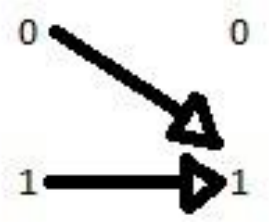
| | | | | |
|---|---|---|---|---|
| 00 | 1 | 0 | 0 | 0 |
| 01 | 0 | 1 | 0 | 0 |
| 10 | 0 | 0 | 1 | 0 |
| 11 | 0 | 0 | 0 | 1 |

- Circuito:




- Pruebas: Al ser la identidad no hay necesidad de realizar pruebas.

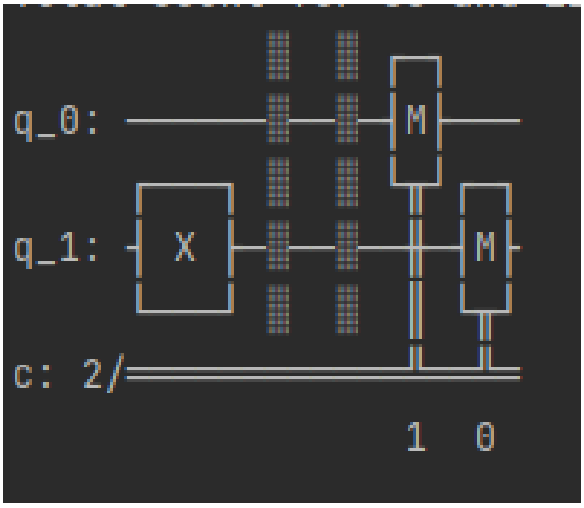
3. Función #3:
   - Dibujo de la función:



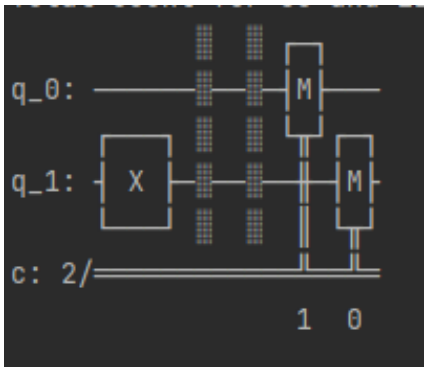

   - Matríz: $Uf =$

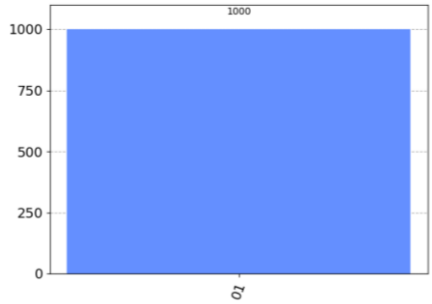
| | | | | |
|---|---|---|---|---|
| 00 | 0 | 1 | 0 | 0 |
| 01 | 1 | 0 | 0 | 0 |
| 10 | 0 | 0 | 0 | 1 |
| 11 | 0 | 0 | 1 | 0 |

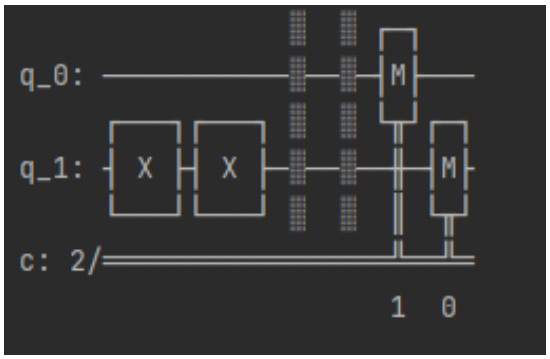
   - Circuito:

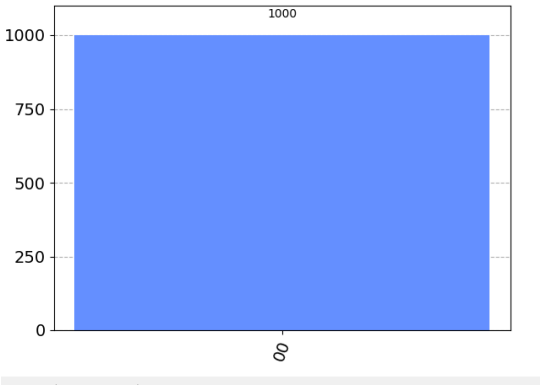



- Pruebas:
    a. Caso 00 y 01.

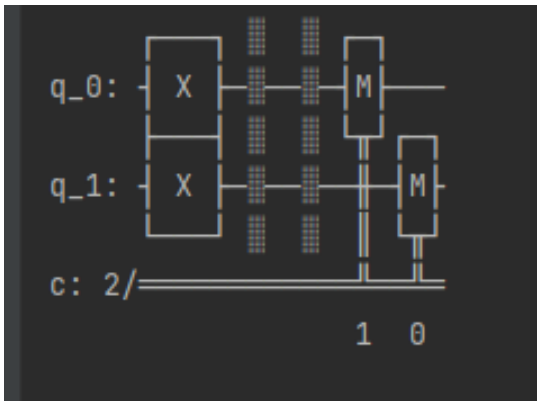


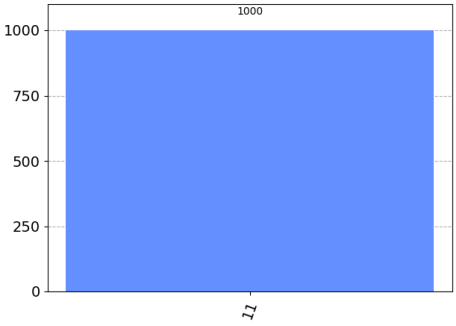



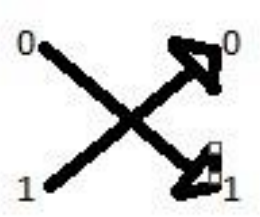



b. Caso 10 y 11.




4. Función #4:
    - Dibujo de la función:

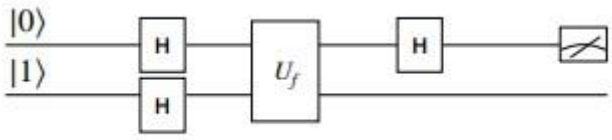



• Matríz: $Uf =$

| | | | | |
|---|---|---|---|---|
| 00 | 0 | 1 | 0 | 0 |
| 01 | 1 | 0 | 0 | 0 |
| 10 | 0 | 0 | 1 | 0 |
| 11 | 0 | 0 | 0 | 1 |

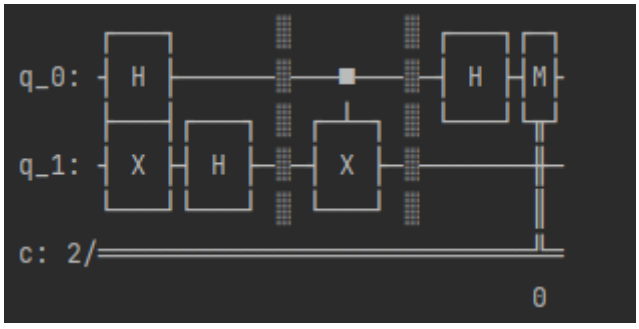
**<u>Verificar que el algoritmo de Deutsch funciona para comprobar cuáles de estas funciones son balanceadas o constantes.</u>**

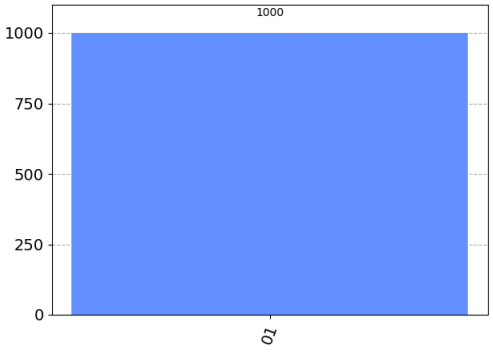
Recordemos que verificar si es constante o no, el circuito que se usará será el siguiente:



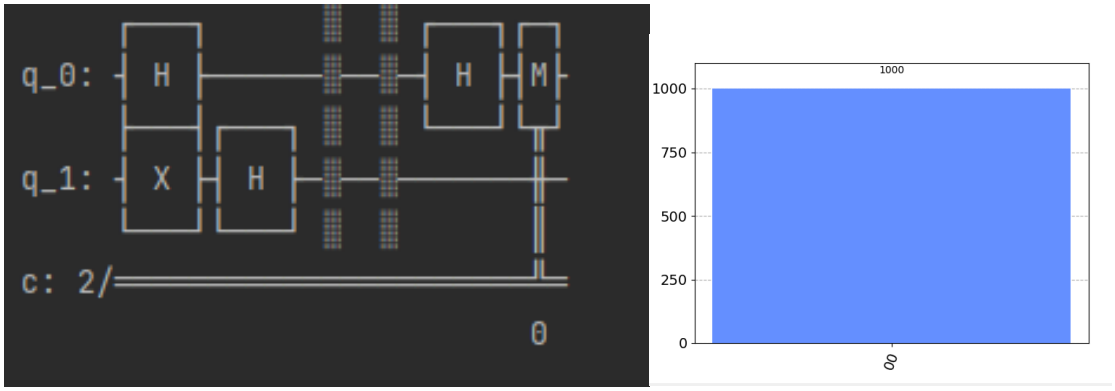

1. Primer circuito:

   Como construimos la función y como es su estructura se puede saber que la función es balanceada, viendo el resultado que es 1 nos da por asegurado que es balanceada.

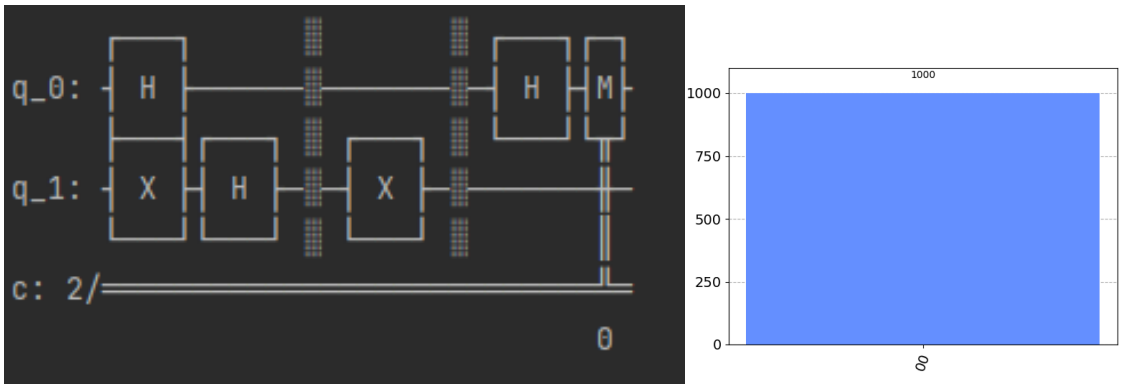


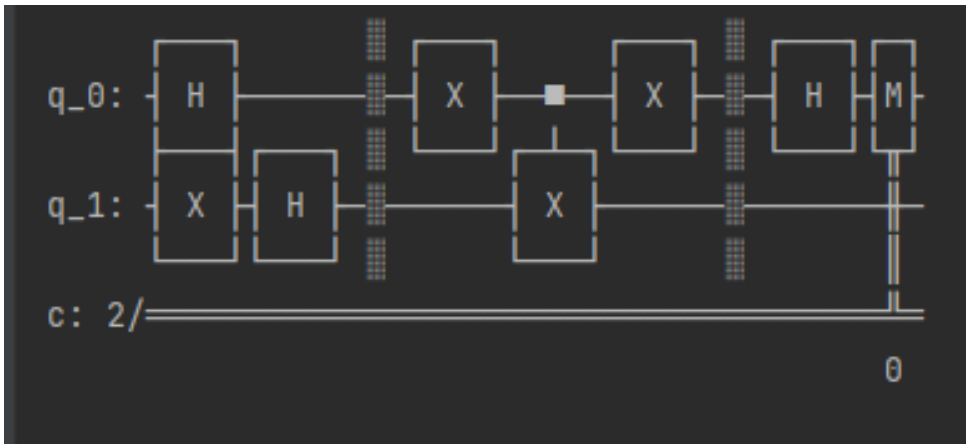
2. Segundo cirucito:
   Como construimos la función y como es su estructura se puede saber que la función es contante, viendo el resultado que es 0 nos da por asegurado que es constante.

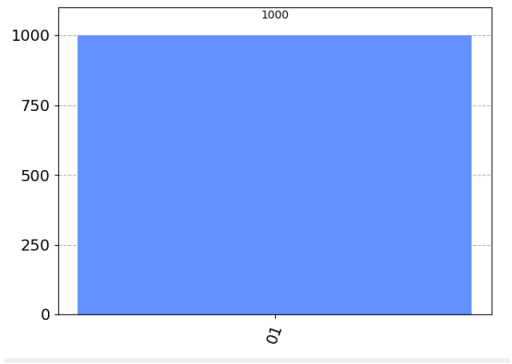



3. Tercer cirucito:
   Como construimos la función y como es su estructura se puede saber que la función es contante, viendo el resultado que es 0 nos da por asegurado que es constante.



4. Cuarto circuito:
   Como construimos la función y como es su estructura se puede saber que la función es balanceada, viendo el resultado que es 1 nos da por asegurado que es balanceada.

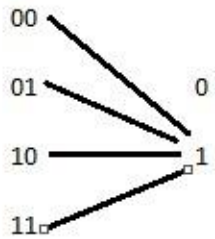



## Algoritmo Deutsch-Jozsa

A continuación, vamos a implementar el algoritmo de Deutsch-Joza, para 2 funciones.

1. Función #1:
   - Dibujo de la función:

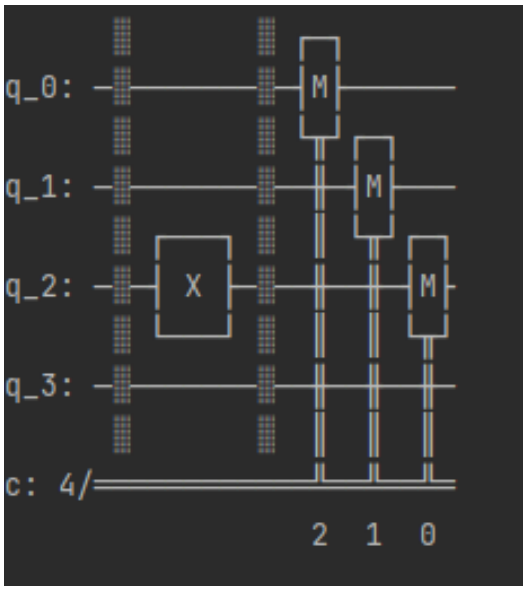



   - Matríz:
     $Uf =$

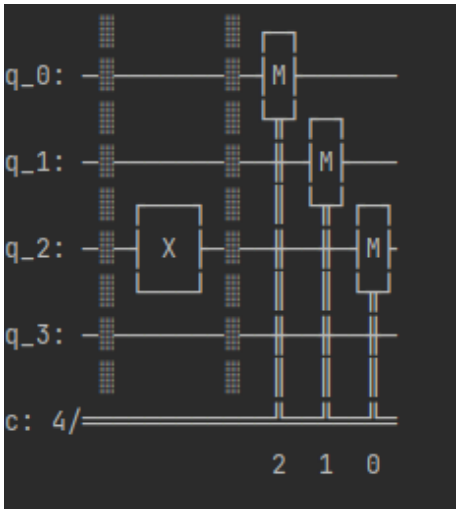
|

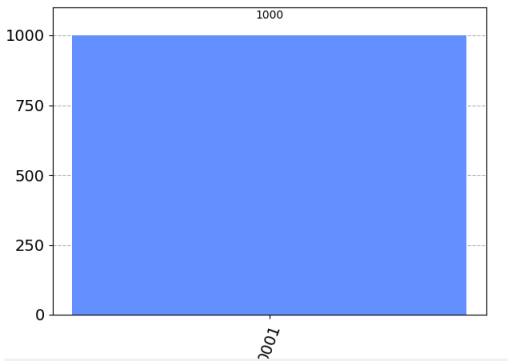
| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 000 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 001 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 010 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| 011 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| 100 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| 101 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| 110 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| 111 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |

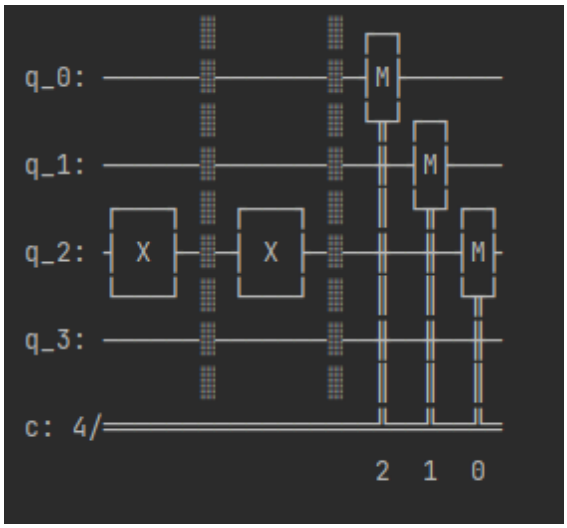
- Circuito:

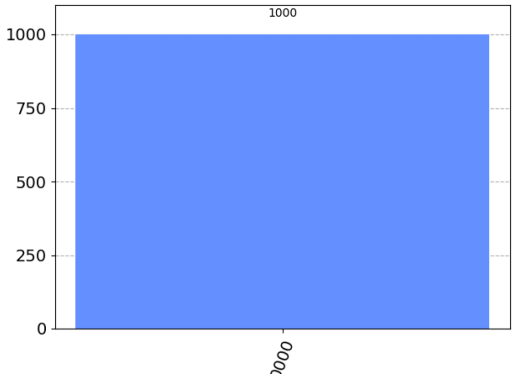


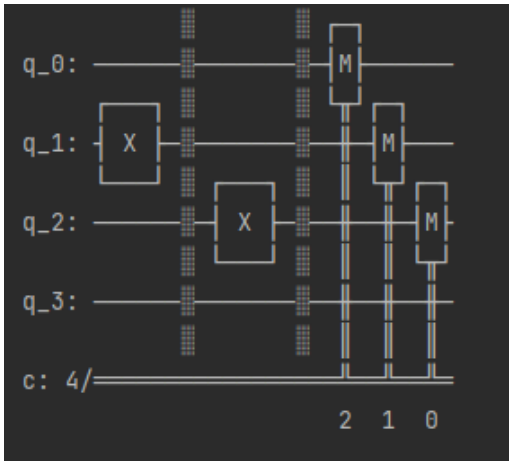

- Pruebas:
    a. Caso 000, 001, 010, 011:

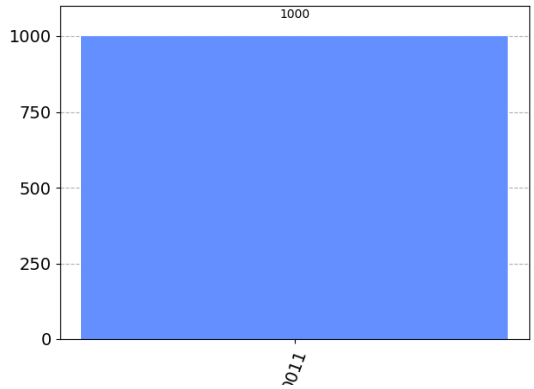

q_0:
q_1:
q_2:
q_3:
c: 4/
M
M
X
M
2 1 0

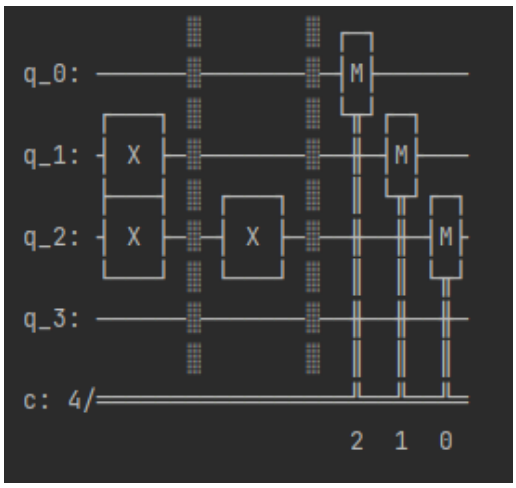

1000
1000
750
500
250
0

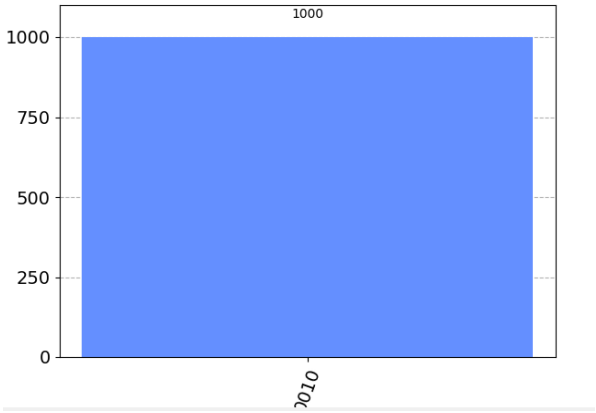

q_0:
q_1:
q_2:
q_3:
c: 4/
M
M
X
X
M
2 1 0


1000
1000
750
500
250
0
1000

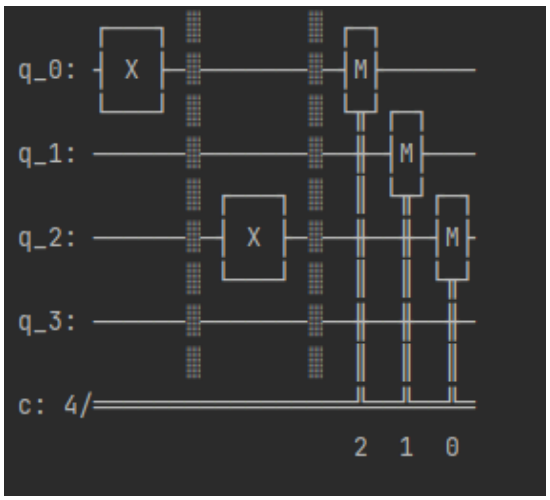

q_0:
q_1:
q_2:
q_3:
c: 4/
X
X
M
M
M
2 1 0

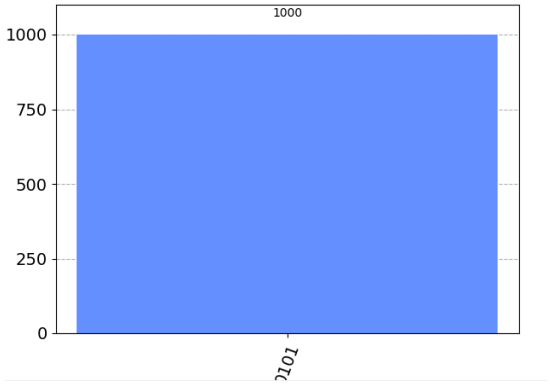

1000
1000
750
500
250
0
1011

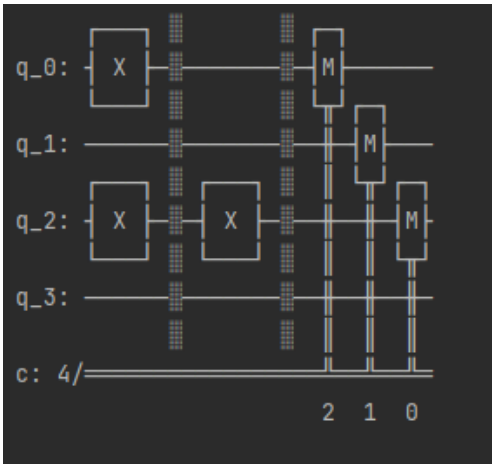

q_0:
q_1:
q_2:
q_3:
c: 4/
X
X
X
M
M
M
2 1 0

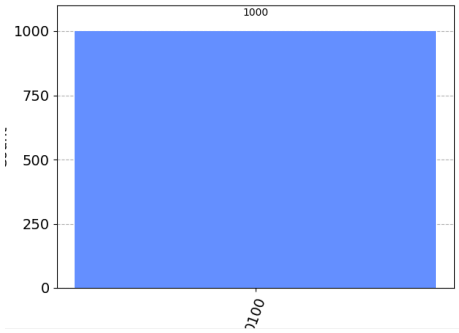


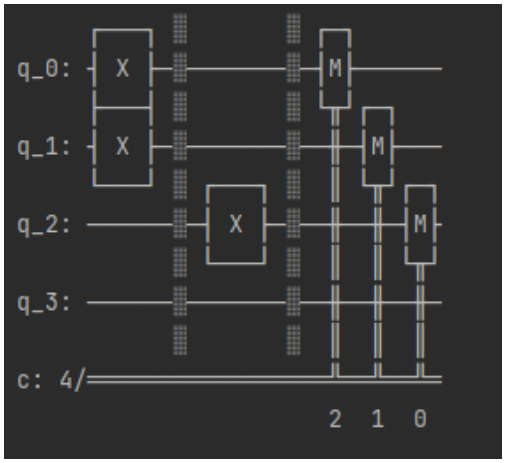

b. Caso 100, 101, 110, 111

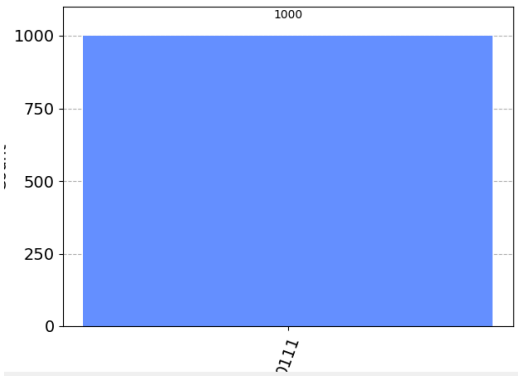


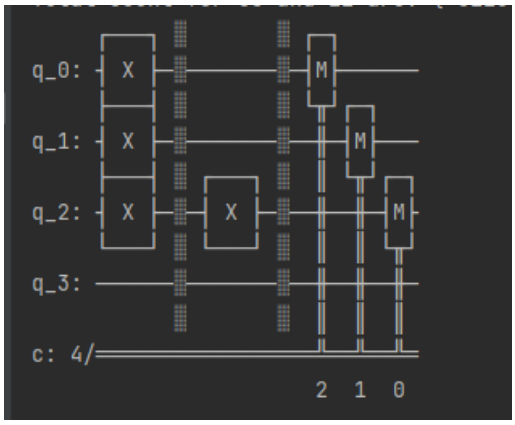



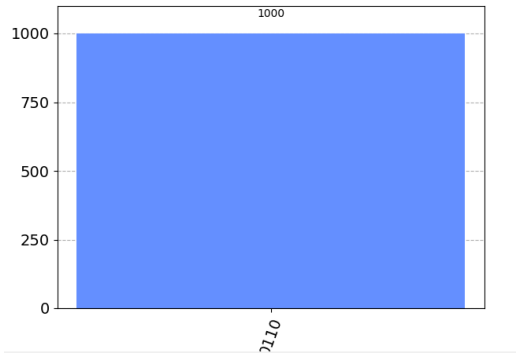

q_0:
q_1:
q_2:
q_3:
c: 4/
X
X
X
M
M
M
2 1 0


1000
1000
750
500
250
0

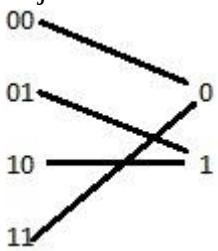

q_0:
q_1:
q_2:
q_3:
c: 4/
X
X
X
M
M
M
2 1 0


1000
1000
750
500
250
0



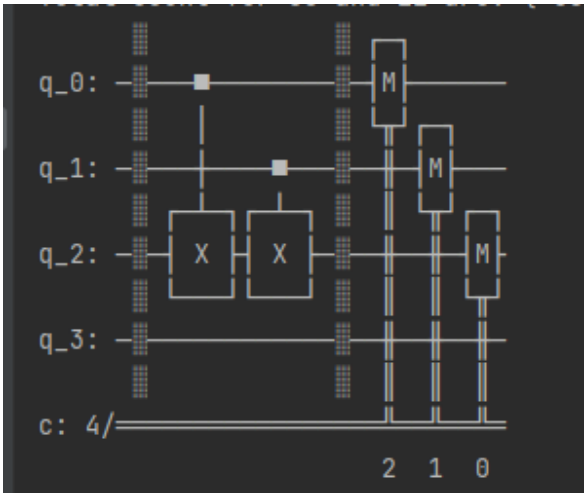

2. Función #2:
    - Dibujo de la función:



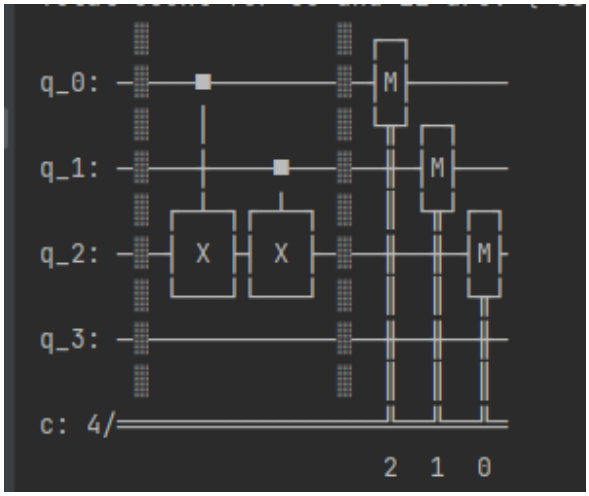

    - Matríz: $Uf =$

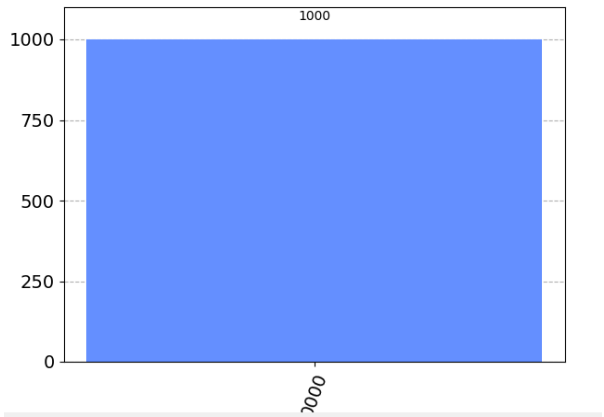
| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 000 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 001 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 010 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| 011 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| 100 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| 101 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| 110 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| 111 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |

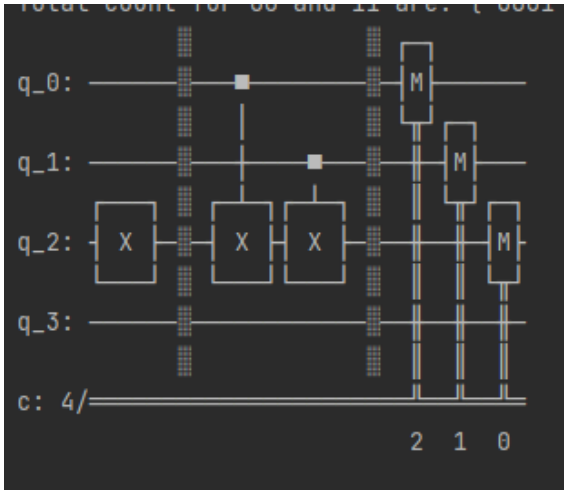
    - Circuito:

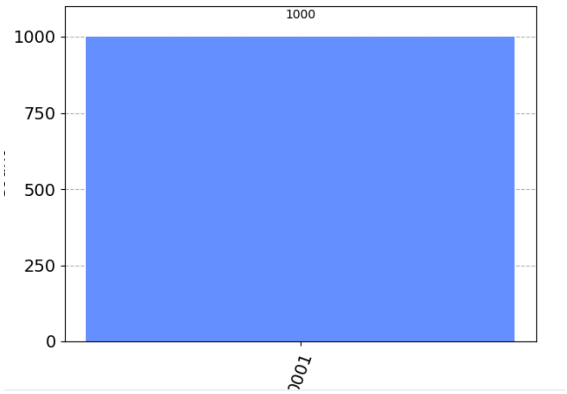


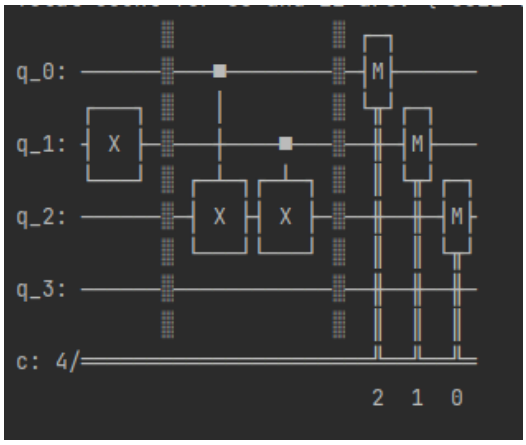

- Pruebas:
    a. Caso 000, 001, 010, 011.

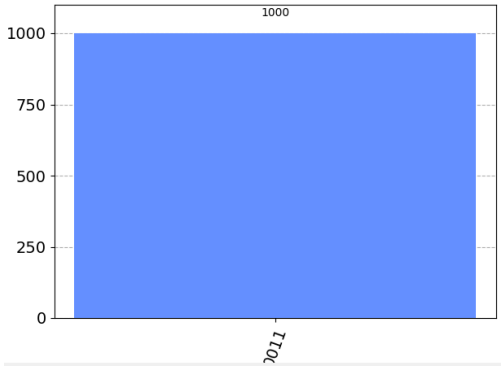


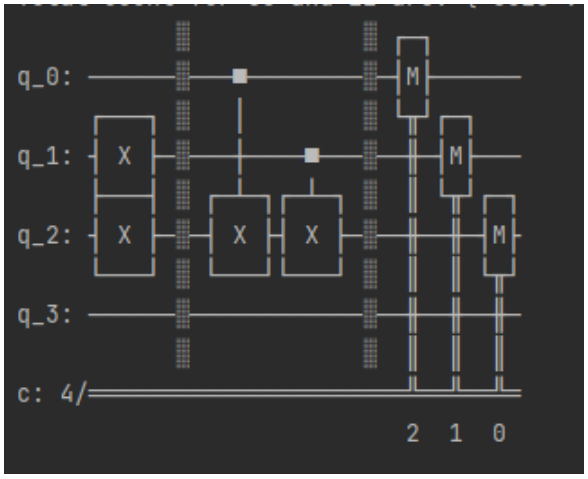



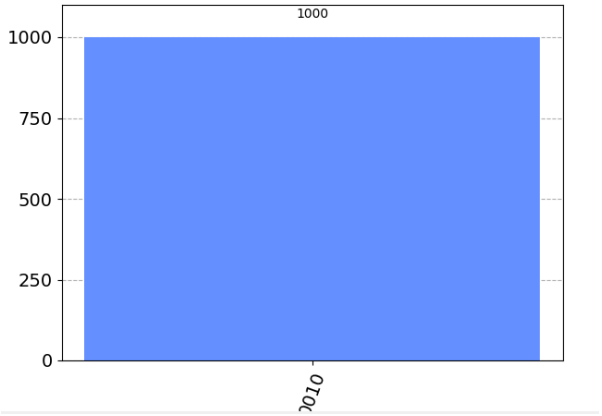

q_0:
q_1:
q_2:
q_3:
c: 4/
X
X
X
M
M
M
2 1 0


1000
1000
750
500
250
0

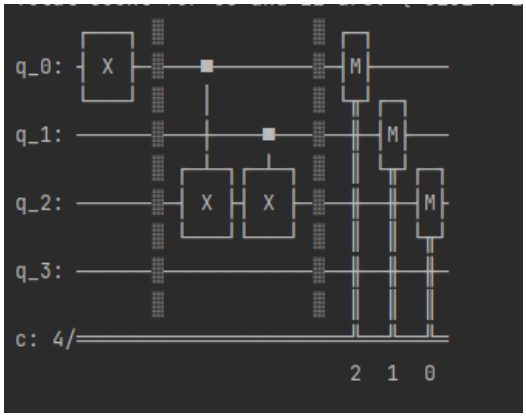

q_0:
q_1:
q_2:
q_3:
c: 4/
X
X
X
M
M
M
2 1 0

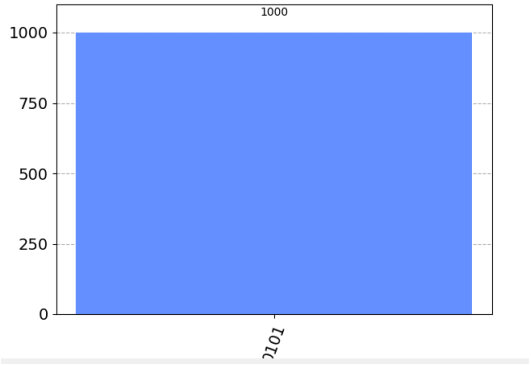

1000
1000
750
500
250
0

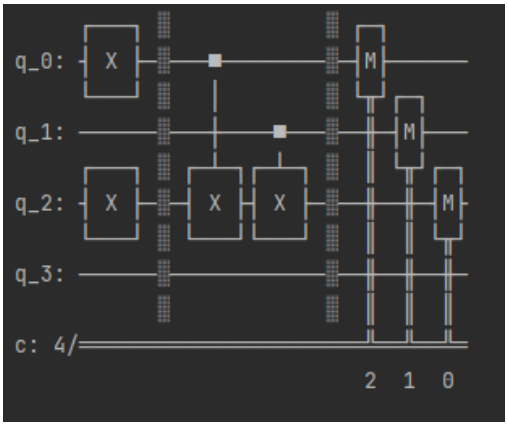


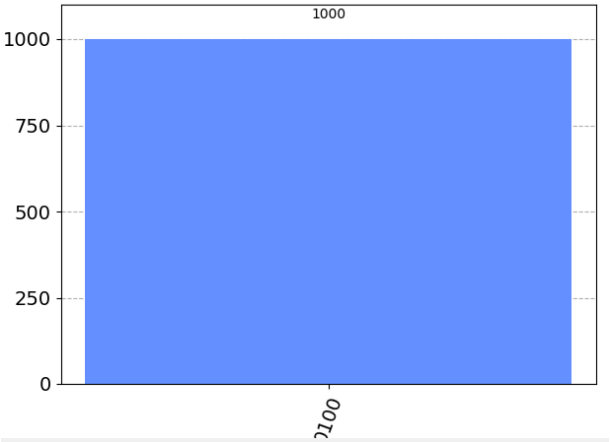



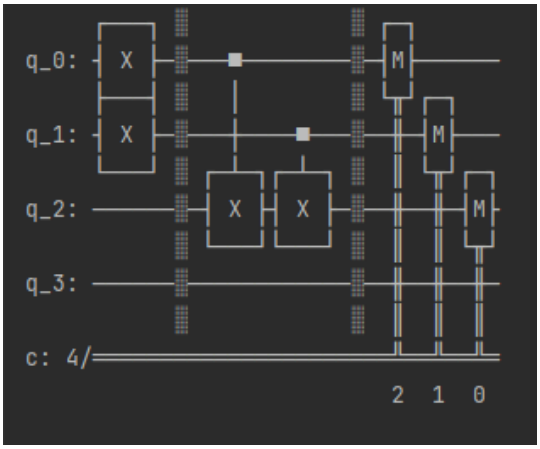

b. Caso 100, 101, 110, 111

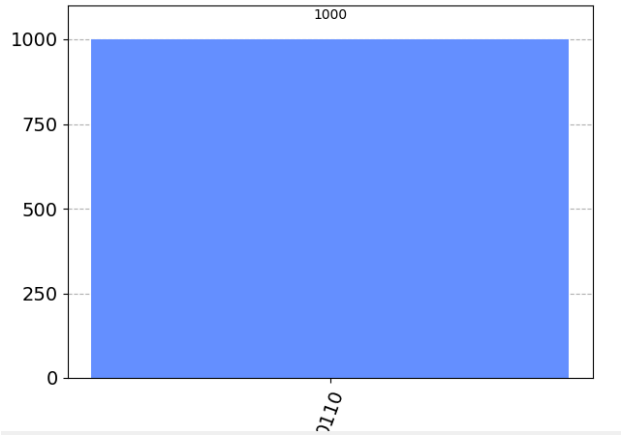


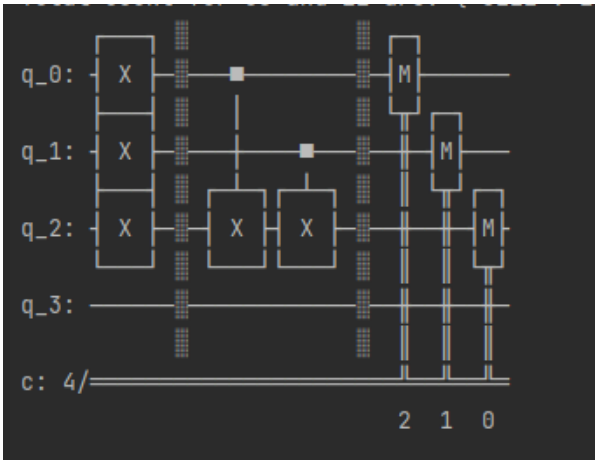

1000
1000
750
500
250
0
1101

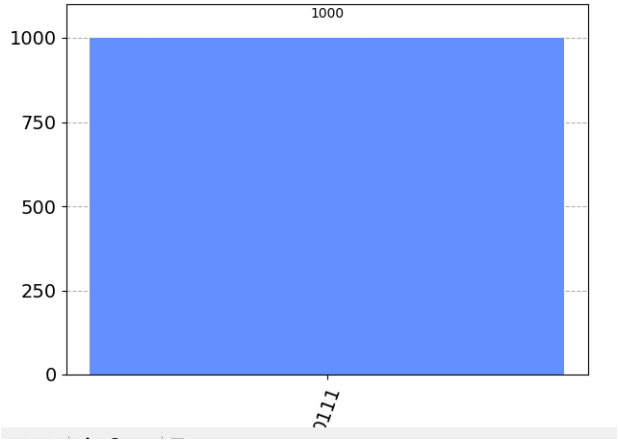

q_0:
q_1:
q_2:
q_3:
c: 4/
X
M
2 1 0


1000
1000
750
500
250
0
0100


q_0:
q_1:
q_2:
q_3:
c: 4/
X
M
2 1 0

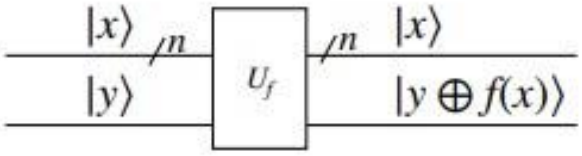


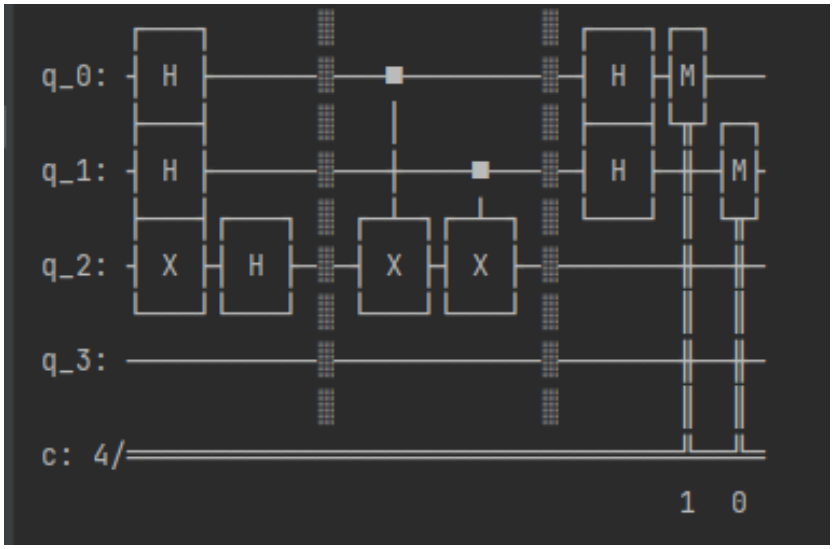



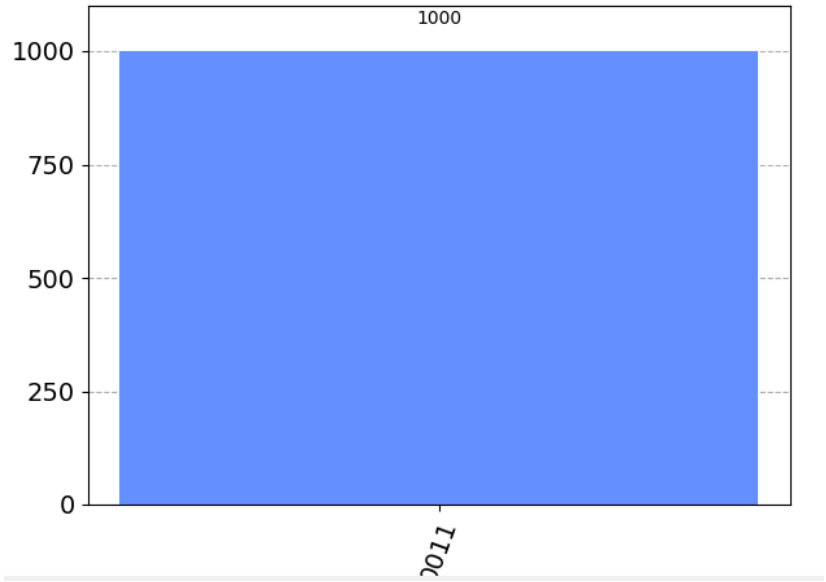

**Verificar que el algoritmo de Deutsch funciona para comprobar cuáles de estas funciones son balanceadas o constantes.**

Recordemos que verificar si es constante o no, el circuito que se usará será el siguiente:




Nuestro circuito y su resultado será como se ver a continuación.




De acuerdo con la función que tenemos, se sabía gracias a su estructura y como se manejaba la matriz que la función iba a ser balanceada, realizando el algoritmo nos damos cuenta que es balanceada.

## Conclusiones

1. Con la implementación de los algoritmos nos facilita el camino para deducir si una función es balanceada o constante haciéndolo de forma eficiente nos reduce bastante pasos a como se hacía en la computación clásica de antaño.
2. Reducción de complejidad, la complejidad en un computador clásico es de $O(2^n)$ y un computador cuántico es de $O(1)$.

## BIBLIOGRAFIA:

**http://scielo.iics.una.py/scielo.php?script=sci_arttext&pid=S2222-145X2021000200083#f2**