Implementación del algoritmo de Deutsch y Deutsch-Jozsa

Daniel Sebastián Ochoa Urrego
Escuela Colombiana de Ingeniería Julio Garavito
daniel.ochoa-u@mail.escuelaing.edu.co

7 de mayo de 2021

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

Tabla de contenidos

1 Introducción

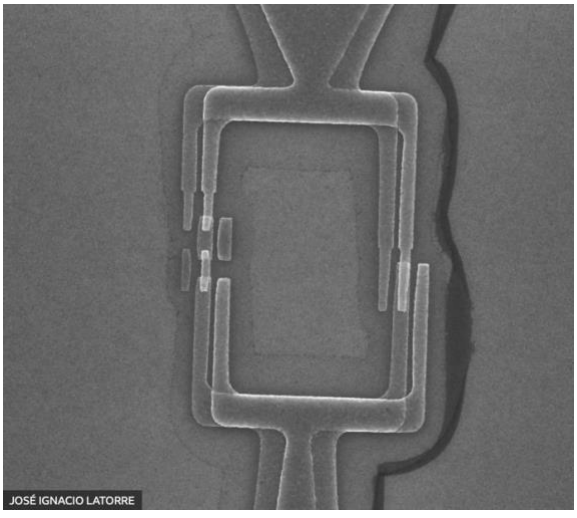
La computación cuántica es una rama totalmente diferente a lo que hemos venido trabajando hasta el momento con los computadores clásicos. Como ya sabemos los computadores clásicos a un nivel muy básico están construidos a partir de 1's y 0's, lo que llamamos bits, pero la gran diferencia con los computadores cuánticos es que estos funcionan a partir de Qubits, los cuales no pueden ser representados simplemente por 1's y 0's, sino que para representarlos debemos irnos al mundo de los numeros complejos, específicamente al campo vectorial de C^2, pues para poder representar un Qubit debemos representarlo como la combinación lineal de los estados 0 y 1, los cuales son representados de la siguiente manera:

$$|0>=[1,0]^T$$
$$|1>=[0,1]^T$$

Por lo que el Qubit será como se ve a continuación, donde c1 y c2 son números complejos:

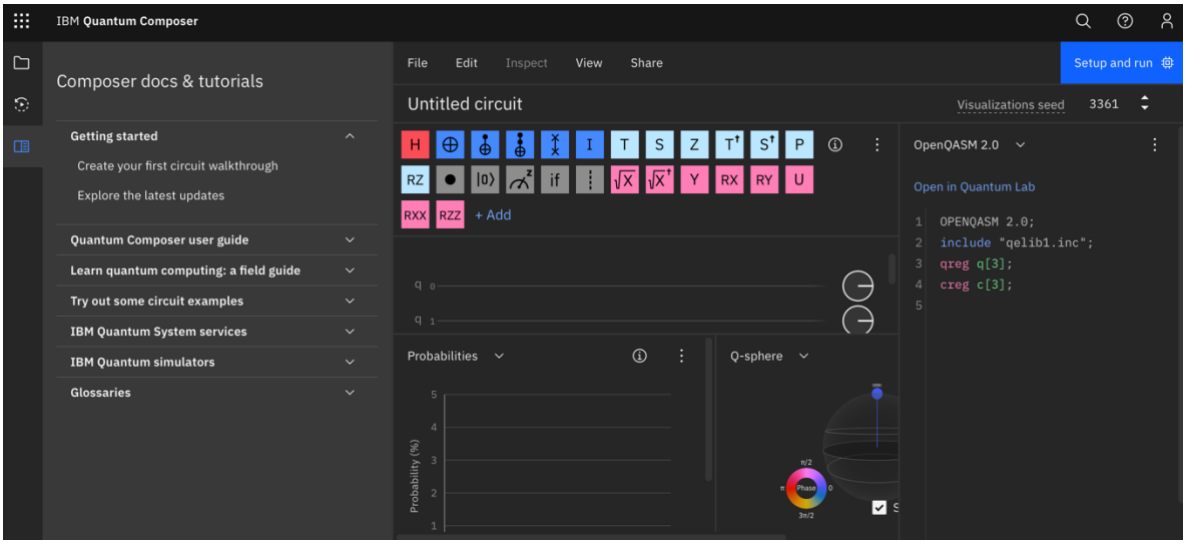
$$|\psi>=c_1|0>+c_2|1>$$

Una imagen real de como se ve un Qubit se puede ver en la siguiente figura:



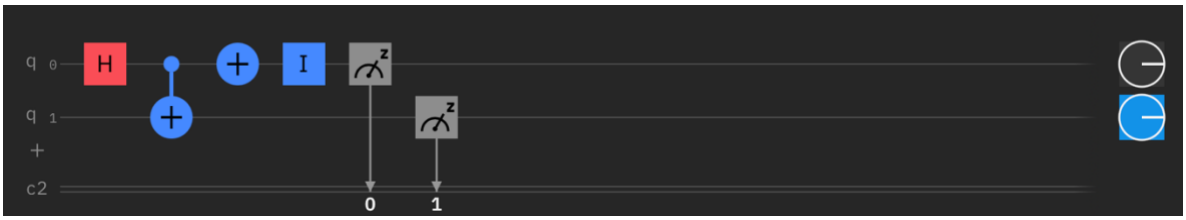
Es a partir de estos que hemos construido lo que llevamos hasta el momento en computación cuántica. Hasta ahora la compañía IBM es la que mas esfuerzo ha hecho a nivel mundial por construir una nueva forma de computación, ya que han construido alrededor de 18 computadores cuánticos que han dispuesto para que cualquiera de

nosotros que estemos interesados podamos explorar la computación cuántica de manera libre y gratuita a través de su pagina web, código fuente o la aplicación gratis que ofrecen.



Actualmente gracias a todas las nuevas posibilidades que nos ofrece la cuántica es que hemos podido crear diferentes artefactos como lo son la fibra óptica, láseres, entre muchos otros, y esto es apenas rascar la superficie, de hecho en un entrevista con BBC el reconocido físico de la Universidad de Barcelona, José Ignacio Latorre, dijo lo siguiente: “El futuro será cuántico o no será. Y el mañana que nos espera es apasionante. La cuántica nos permitirá hacer lo que hasta ahora sólo podíamos soñar.”

En general con un Qubit podemos resolver una gran cantidad de problemas, a través de algoritmos cuánticos, los cuales en este informe son representados por circuitos, que usan compuertas cuánticas, como lo son el CNOT, H, I, NOT, entre otras. Un ejemplo de un circuito sería el siguiente:



En donde se tiene por convención que los valores de q0 y q1 siempre ingresaran con los estados|0> y a partir de ahí utilizando las diferentes compuertas realizaremos operaciones sobre el estado del vector en cuestión para finalmente realizar una medición al estado de salida y tener una respuesta del algoritmo en cuestión.

Para este informe primero se presentarán los tipos de funciones que existen en {0, 1} -> {0, 1} y como se comportan estas, para luego presentar el algoritmo de Deutsch y mostrar

sus resultados con las funciones anteriormente mencionadas, y por ultimo se presentara el algoritmo de Deutsch-Jozsa con algunos ejemplos de sus resultados.

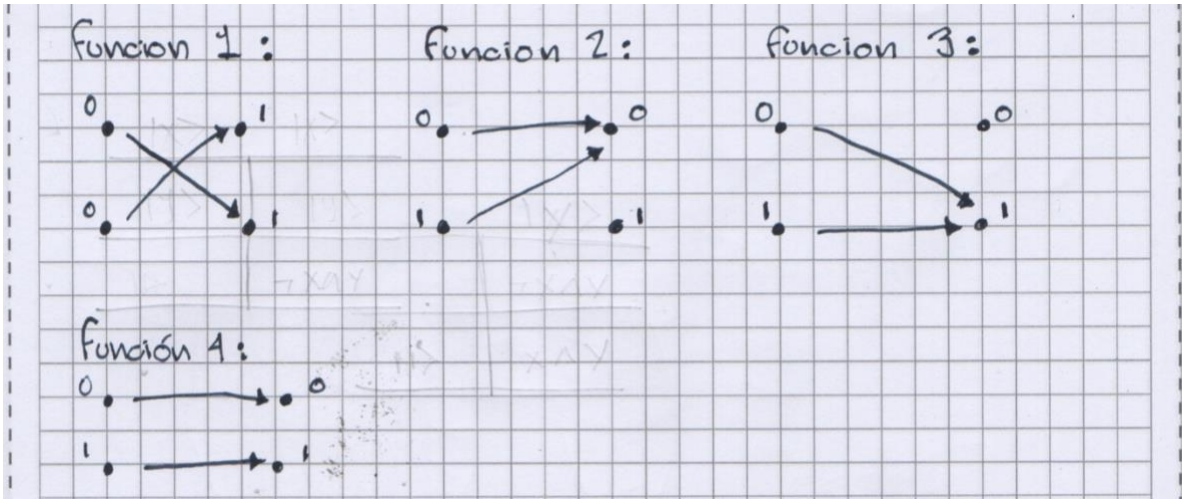
2 Algoritmo de Deutsch

En este capitulo se presentarán las 4 funciones posibles que van de {0, 1} -> {0, 1}, el concepto de funciones balanceadas y constantes, para luego introducir el algoritmo de Deutsch, y como a través de este se calcula cuales de estas funciones son balanceadas y cuales son constantes. Para hacerlo, simularemos el circuito cuántico que implementa el algoritmo y según la salida de este podremos decir con seguridad si la función f(x) es o no balanceada.

2.1 Problema

Para poder entender el algoritmo de Deutsch primero debemos exponer con que tipo de funciones es con el que se trabaja este.

Las funciones que utiliza el algoritmo de Deutsch son tales que tienen su dominio en {0, 1} y su rango en {0, 1}, ya que estas funciones solo tienen 2 posibles entradas con 2 posibles salidas, son solo 4 las funciones posibles de este estilo, que para ponerles algún nombre se les llamara función 1,2,3,4 y su representación en forma de conjuntos.



A las funciones anteriores podemos darles ciertas propiedades, diremos que una función es balanceada si se cumple cada elemento de la entrada tiene un elemento diferente en la salida, (Como lo son la función 1 y 4) o enunciado de manera formal: f(0) /= f(1) (f(0) es diferente de f(1)), y se dice que una función es constante si ambos elementos de la entrada tienen el mismo elemento de la salida, o enunciado de manera formal: f(0) = f(1) (f(0) es igual a f(1)).

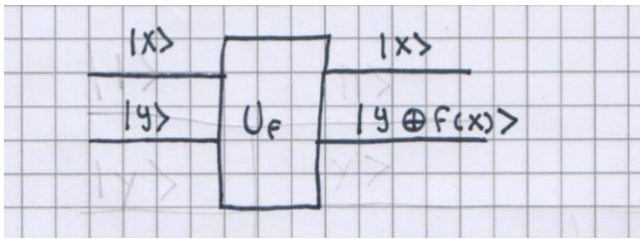
Habiendo dicho lo anterior, podríamos crear un algoritmo que dada una función nos diga si es balanceada o no, y es eso a nivel general a lo que nos ayuda el algoritmo de Deutsch.

Una manera relativamente fácil de implementar el algoritmo de Deutsch en un computador clásico seria lo siguiente:

```
def isBalanced(f):
  return f(0) != f(1)
```

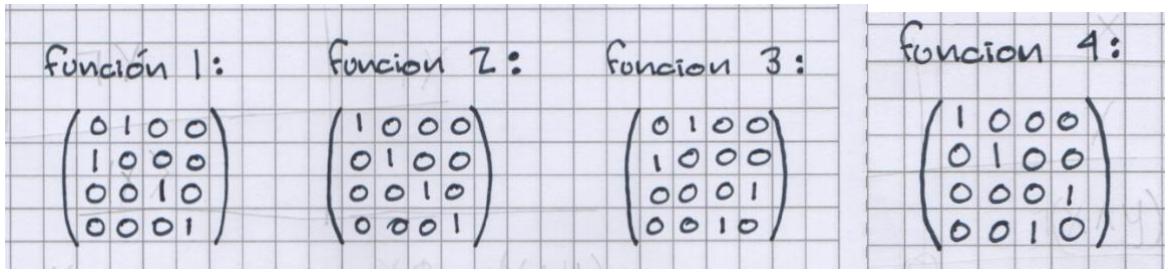
Y aunque es verdad que el algoritmo es relativamente rápido, si nos pasamos a un computador cuántico, podríamos mejorar mucho en términos de tiempo, pues pasaríamos de llamar 2 veces a la función a llamarla solo 1.

Pero antes de poder implementar el algoritmo de Deutsch en un algoritmo cuántico, debemos implementar las cuatro funciones en un sub-algoritmo cuántico, de la siguiente manera:



En la imagen anterior se esta mostrando la manera general de implementar las funciones en un circuito cuántico, lo que estamos haciendo, es crear una compuerta (Uf) que recibe un Qubit y retorna otro según la función que estemos aplicando y unas operaciones lógicas entre estos 2.

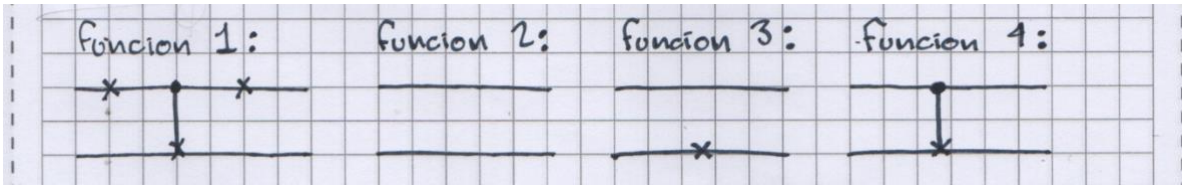
Ya teniendo esta representación grafica y claros los pasos que hay que hacer para obtener la salida, al aplicar la anterior “formula” general a las 4 funciones obtuvimos las siguientes matrices:



Y ahora con estas matrices podemos representar las funciones en un circuito cuántico, lo único que nos quedaría por hacer es encontrar que compuertas ya conocidas podemos usar para obtener el mismo resultado que si se aplicara la compuerta directa de Uf, para

así poder representar en el computador del IBM el circuito cuántico y obtener nuestros resultados.

Para ello simplemente analizamos patrones que siguieran cada estructura de las matrices y obtuvimos que los siguientes circuitos representan cada compuerta de Uf



Y con esto ahora si podemos pasar a implementar los anteriores circuitos en un computador cuántico y ver nuestros resultados.

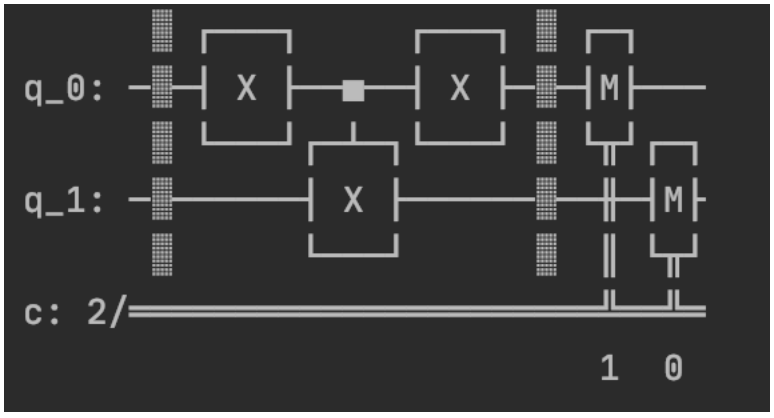
2.2 Implementando las funciones en el computador cuántico

En esta sección se mostrará los resultados obtenidos de ingresar las 4 posibles entradas (las cuales según las matrices serian 00, 01, 10 y 11) en las 4 funciones y comparamos que los resultados fueran los esperados según la matriz construida a partir de Uf.

Para cada función se siguieron los mismos pasos, primero implementamos en código las compuertas propuestas anteriormente que retornan lo mismo que las funciones, segundo realizamos 1000 “disparos” de Qubits y analizamos los resultados en una grafica de barras.

Función 1:

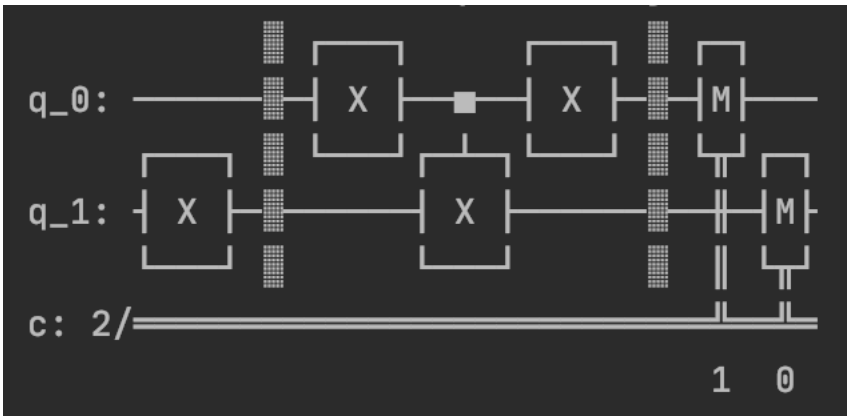
Para esta función el circuito básico que se construyo fue el siguiente:



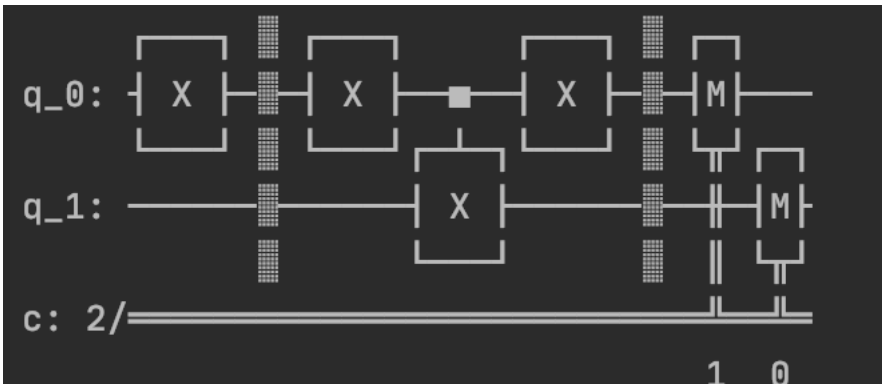
Lo único que cambia relativamente a este es que antes de la primera barrera se agregaran compuertas de NOT para simular las diferentes entradas como lo son 01, 10 y 11 pues como ya se explico en la introducción por convención las entradas siempre son 00 en estos circuitos.

A continuación, se muestran unas las imágenes del mismo algoritmo, pero con las otras 3 entradas:

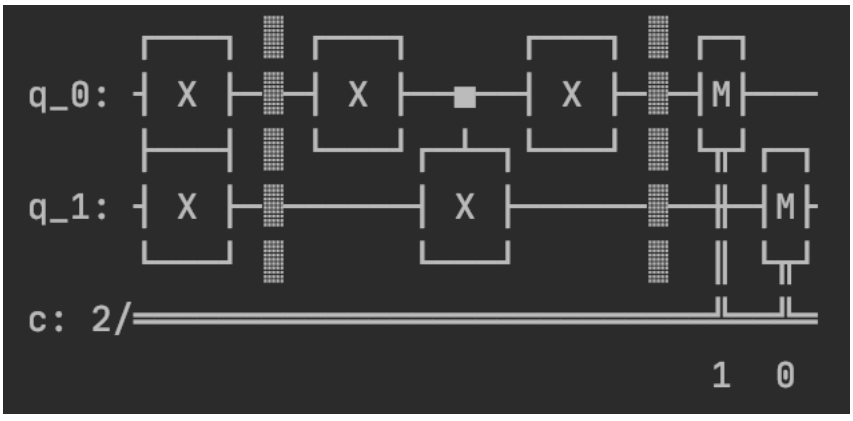
Entrada 01:



Entrada 10:

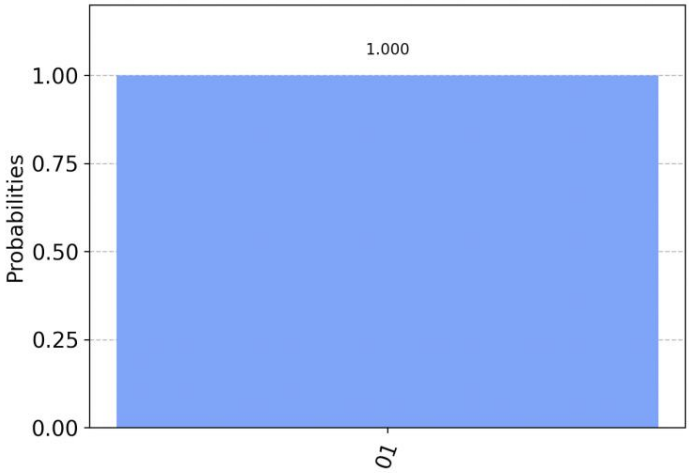


Entrada 11:



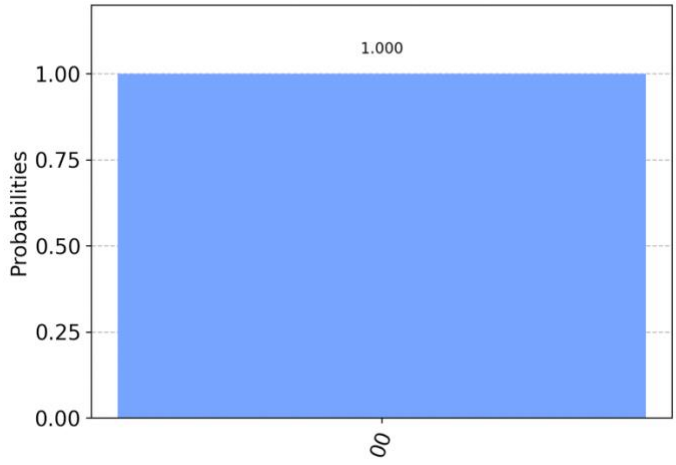
Luego se mostrarán los resultados de los disparos de diferentes Quibits a través del circuito:

- Entrada 00:



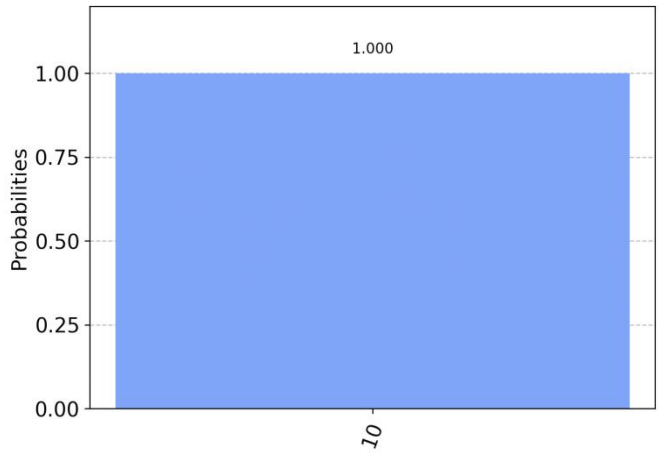
De la grafica anterior podemos ver que para una entrada de 00 en el anterior circuito la salida el 100% de las veces es de 01, resultado que cuadra perfectamente con el esperado por la matriz de Uf de la función 1.

- Entrada 01:



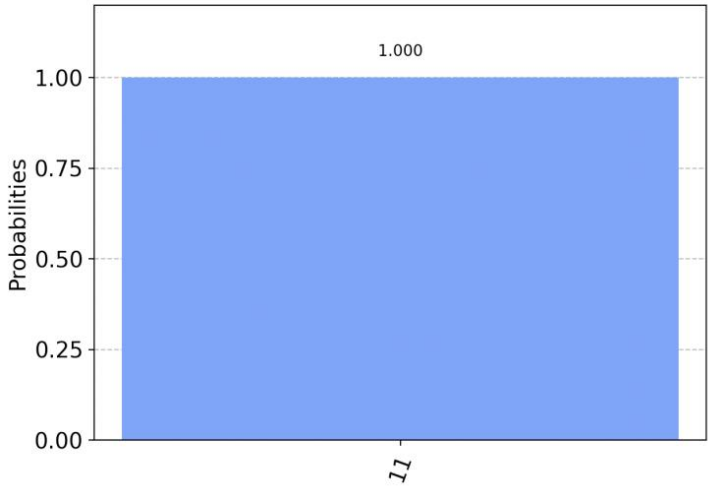
De la grafica anterior podemos ver que para una entrada de 01 en el anterior circuito la salida el 100% de las veces es de 00, resultado que cuadra perfectamente con el esperado por la matriz de Uf de la función 1.

- Entrada 10:



De la grafica anterior podemos ver que para una entrada de 10 en el anterior circuito la salida el 100% de las veces es de 10, resultado que cuadra perfectamente con el esperado por la matriz de Uf de la función 1.

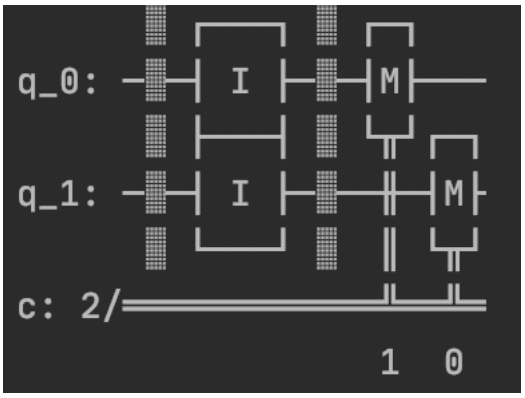
- Entrada 11:



De la grafica anterior podemos ver que para una entrada de 11 en el anterior circuito la salida el 100% de las veces es de 11, resultado que cuadra perfectamente con el esperado por la matriz de Uf de la función 1.

Función 2:

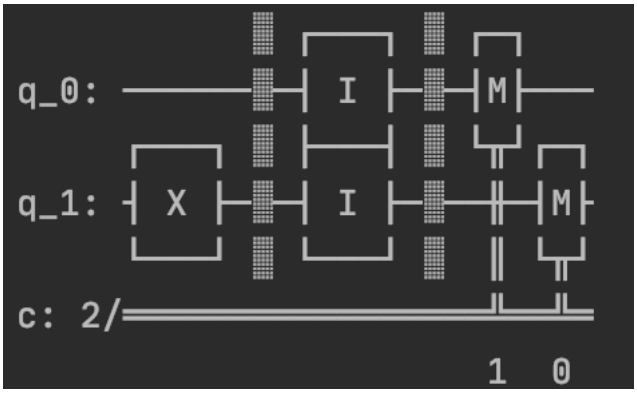
Para esta función el circuito básico que se construyo fue el siguiente:



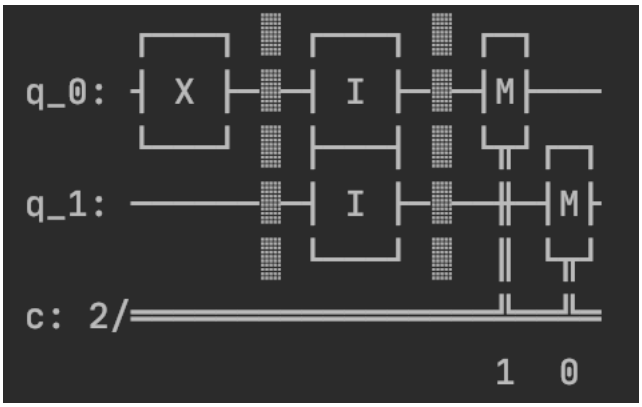
Lo único que cambia relativamente a este es que antes de la primera barrera se agregaran compuertas de NOT para simular las diferentes entradas como lo son 01, 10 y 11 pues como ya se explico en la introducción por convención las entradas siempre son 00 en estos circuitos.

A continuación, se muestran unas las imágenes del mismo algoritmo, pero con las otras 3 entradas:

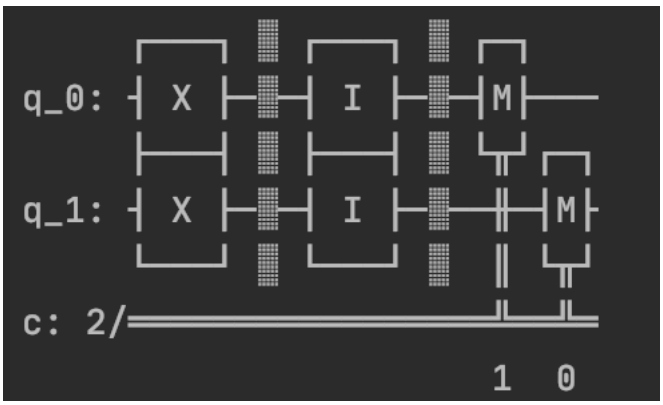
Entrada 01:



Entrada 10:

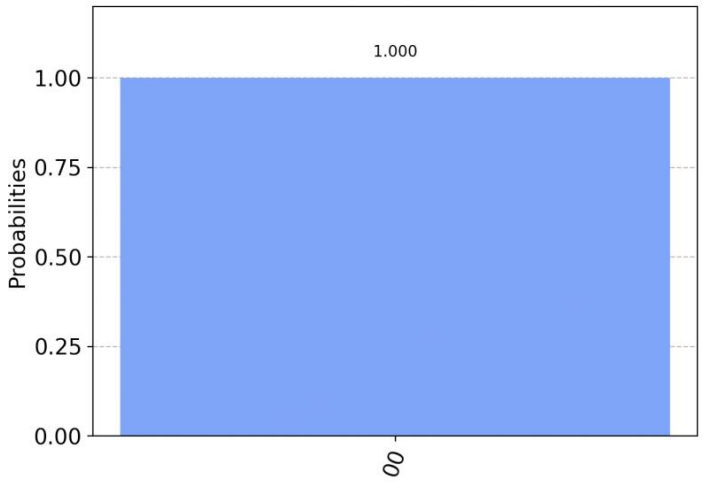


Entrada 11:



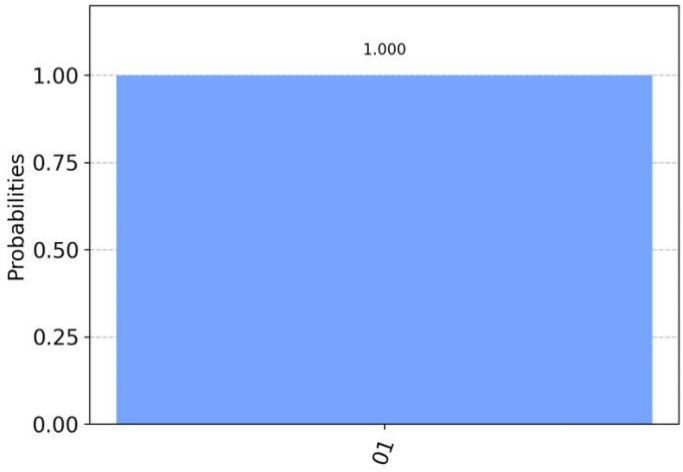
Luego se mostrarán los resultados de los disparos de diferentes Quibits a través del circuito:

- Entrada 00:



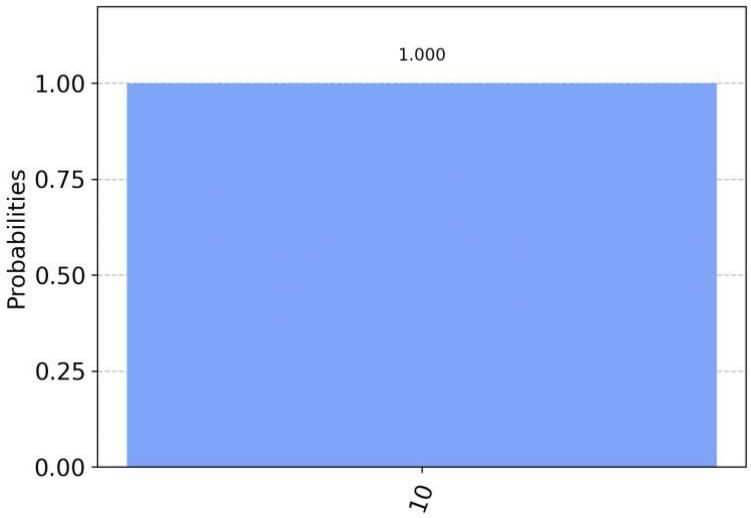
De la grafica anterior podemos ver que para una entrada de 00 en el anterior circuito la salida el 100% de las veces es de 00, resultado que cuadra perfectamente con el esperado por la matriz de Uf de la función 2.

- Entrada 01:



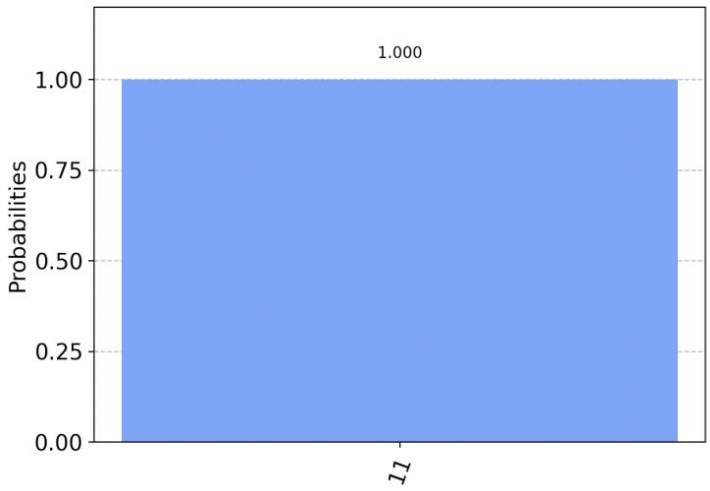
De la grafica anterior podemos ver que para una entrada de 01 en el anterior circuito la salida el 100% de las veces es de 01, resultado que cuadra perfectamente con el esperado por la matriz de Uf de la función 2.

- Entrada 10:



De la grafica anterior podemos ver que para una entrada de 10 en el anterior circuito la salida el 100% de las veces es de 10, resultado que cuadra perfectamente con el esperado por la matriz de Uf de la función 2.

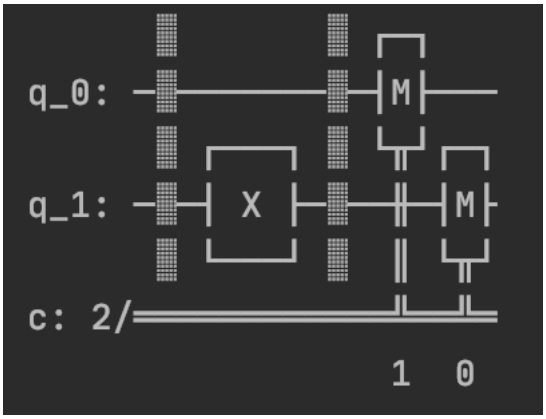
- Entrada 11:



De la grafica anterior podemos ver que para una entrada de 11 en el anterior circuito la salida el 100% de las veces es de 11, resultado que cuadra perfectamente con el esperado por la matriz de Uf de la función 2.

Función 3:

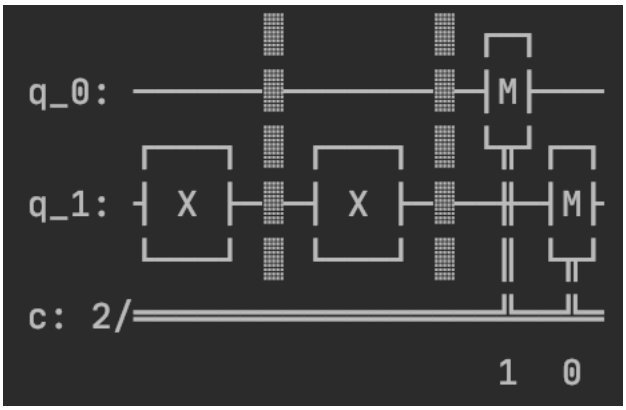
Para esta función el circuito básico que se construyo fue el siguiente:



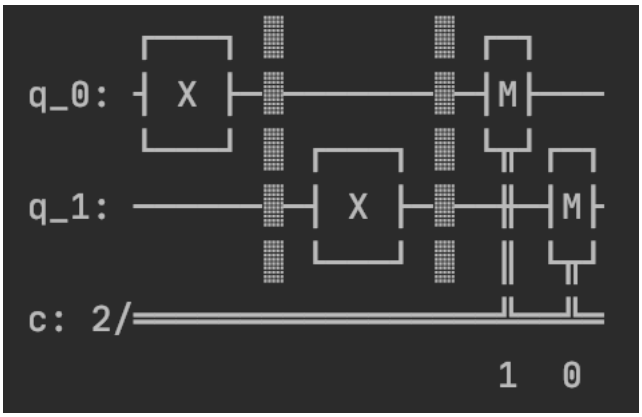
Lo único que cambia relativamente a este es que antes de la primera barrera se agregaran compuertas de NOT para simular las diferentes entradas como lo son 01, 10 y 11 pues como ya se explico en la introducción por convención las entradas siempre son 00 en estos circuitos.

A continuación, se muestran unas las imágenes del mismo algoritmo, pero con las otras 3 entradas:

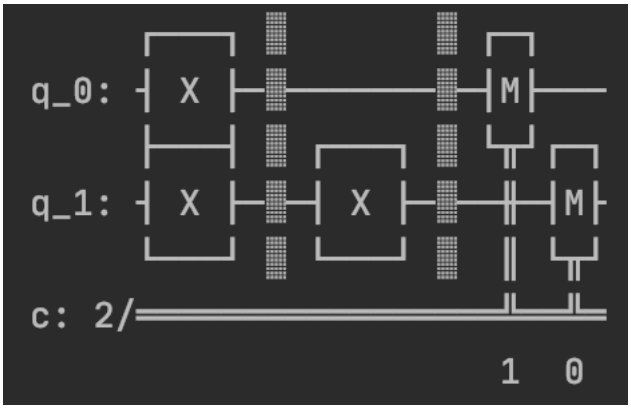
Entrada 01:



Entrada 10:

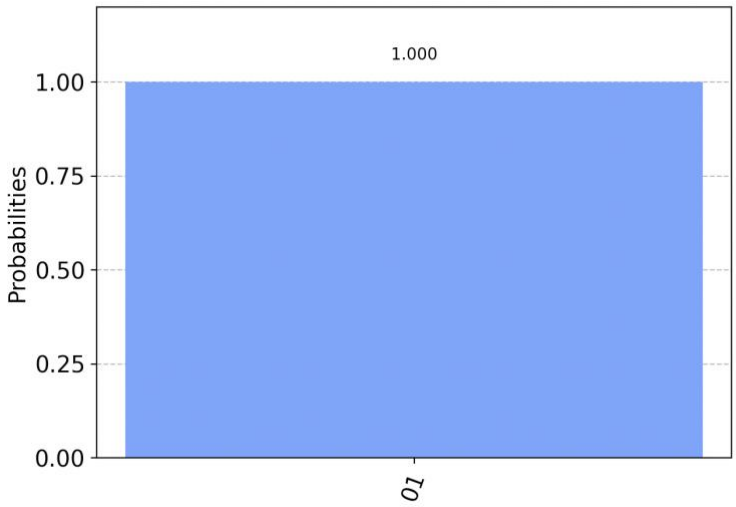


Entrada 11:



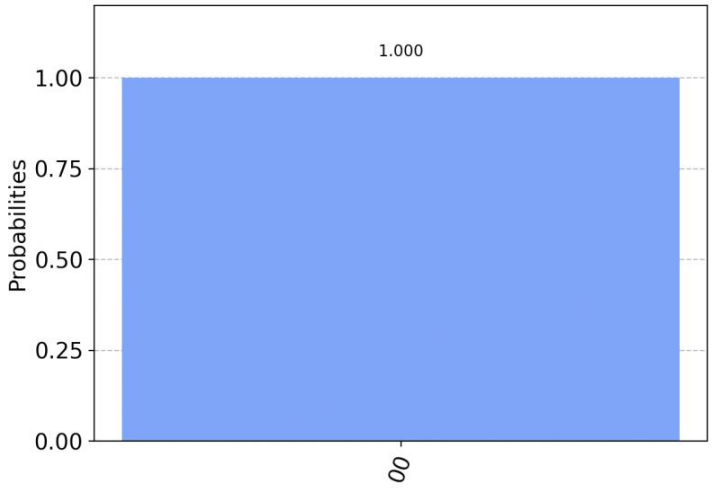
Luego se mostrarán los resultados de los disparos de diferentes Quibits a través del circuito:

- Entrada 00:



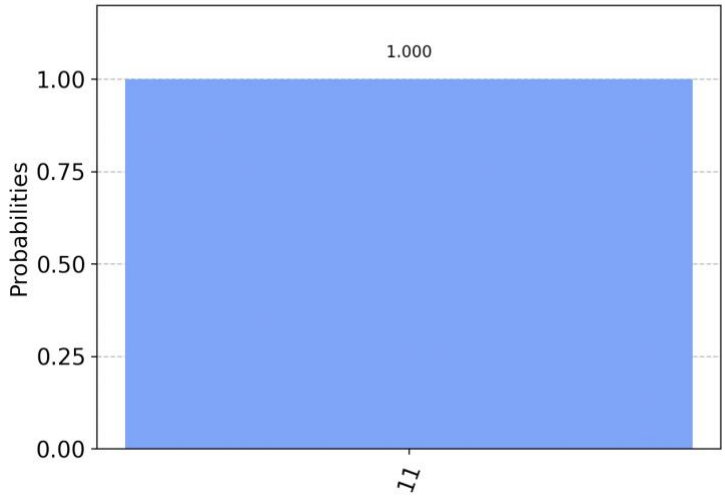
De la grafica anterior podemos ver que para una entrada de 00 en el anterior circuito la salida el 100% de las veces es de 01, resultado que cuadra perfectamente con el esperado por la matriz de Uf de la función 3.

- Entrada 01:



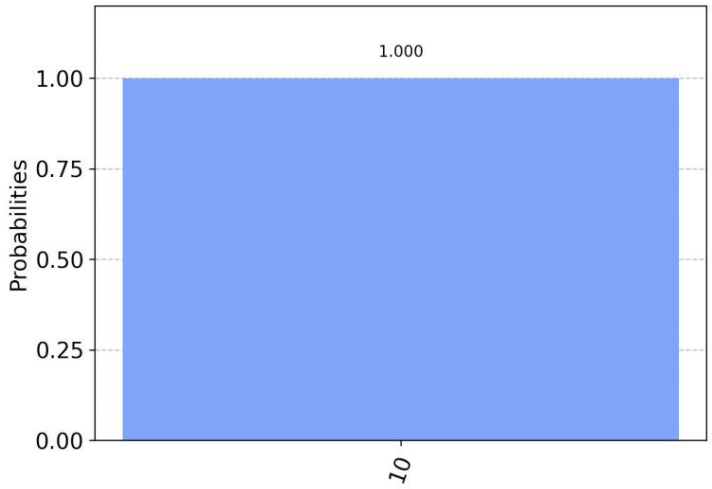
De la grafica anterior podemos ver que para una entrada de 01 en el anterior circuito la salida el 100% de las veces es de 00, resultado que cuadra perfectamente con el esperado por la matriz de Uf de la función 3.

- Entrada 10:



De la grafica anterior podemos ver que para una entrada de 10 en el anterior circuito la salida el 100% de las veces es de 11, resultado que cuadra perfectamente con el esperado por la matriz de Uf de la función 3.

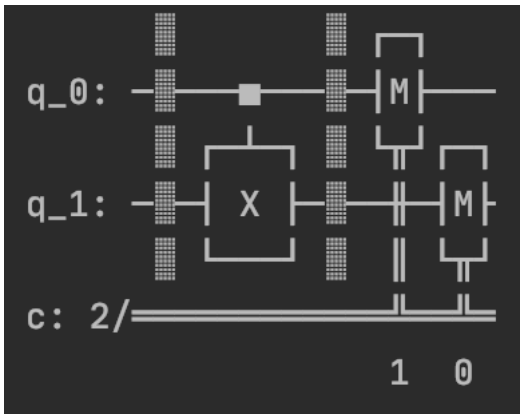
- Entrada 11:



De la grafica anterior podemos ver que para una entrada de 11 en el anterior circuito la salida el 100% de las veces es de 10, resultado que cuadra perfectamente con el esperado por la matriz de Uf de la función 3.

Función 4:

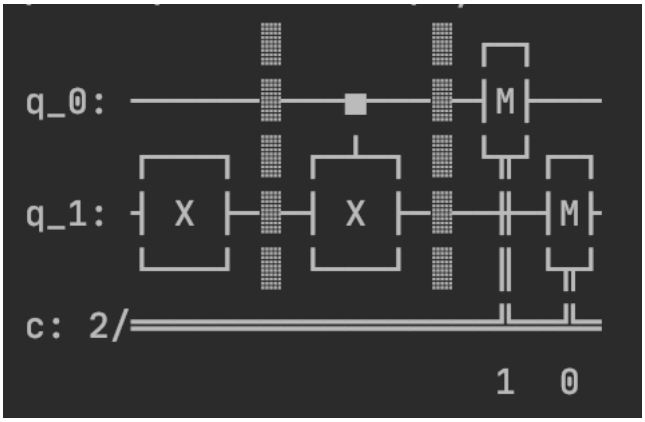
Para esta función el circuito básico que se construyo fue el siguiente:



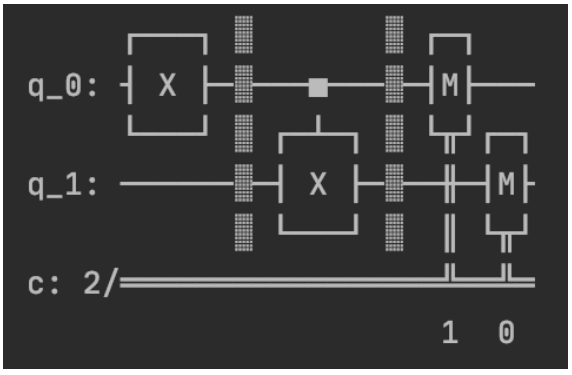
Lo único que cambia relativamente a este es que antes de la primera barrera se agregaran compuertas de NOT para simular las diferentes entradas como lo son 01, 10 y 11 pues como ya se explico en la introducción por convención las entradas siempre son 00 en estos circuitos.

A continuación, se muestran unas las imágenes del mismo algoritmo, pero con las otras 3 entradas:

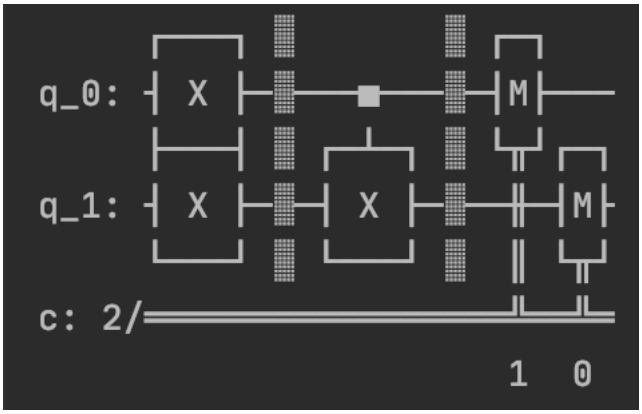
Entrada 01:



Entrada 10:

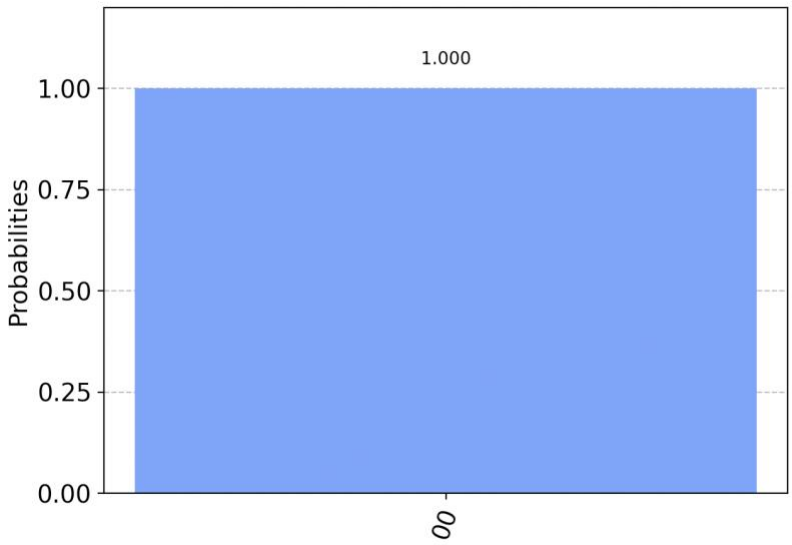


Entrada 11:



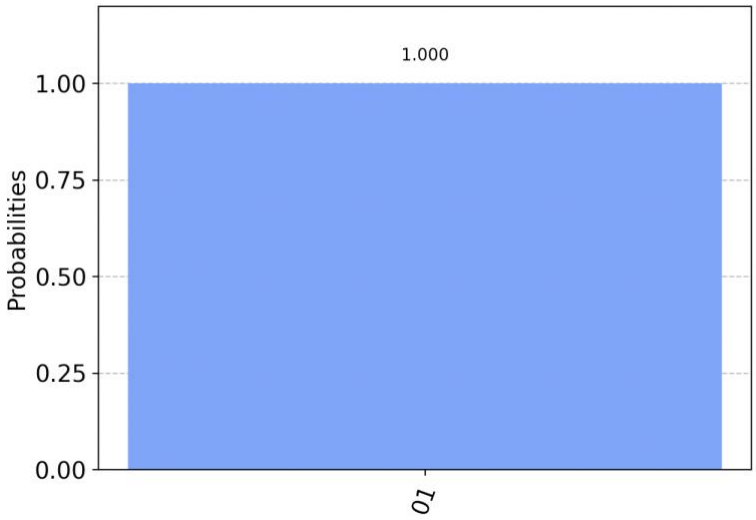
Luego se explicarán los resultados de los disparos de diferentes Quibits a través del circuito:

- Entrada 00:



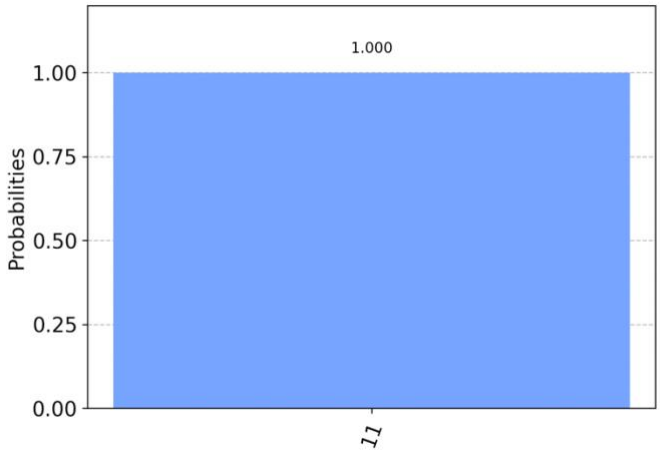
De la grafica anterior podemos ver que para una entrada de 00 en el anterior circuito la salida el 100% de las veces es de 00, resultado que cuadra perfectamente con el esperado por la matriz de Uf de la función 4.

- Entrada 01:



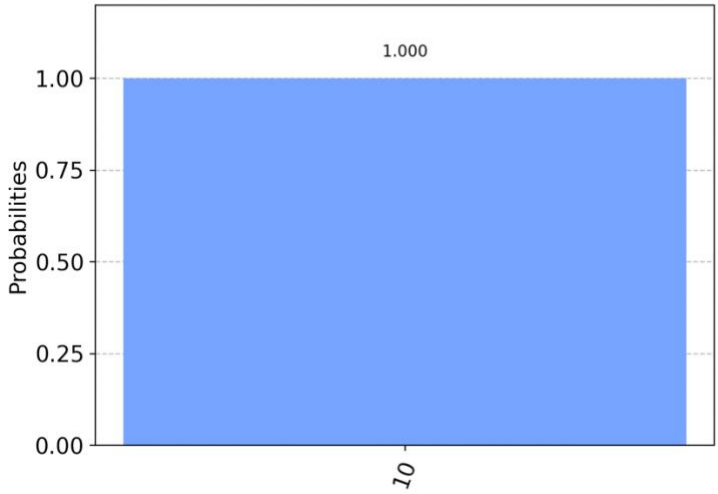
De la grafica anterior podemos ver que para una entrada de 01 en el anterior circuito la salida el 100% de las veces es de 01, resultado que cuadra perfectamente con el esperado por la matriz de Uf de la función 4.

- Entrada 10:



De la grafica anterior podemos ver que para una entrada de 10 en el anterior circuito la salida el 100% de las veces es de 11, resultado que cuadra perfectamente con el esperado por la matriz de Uf de la función 4.

- Entrada 11:



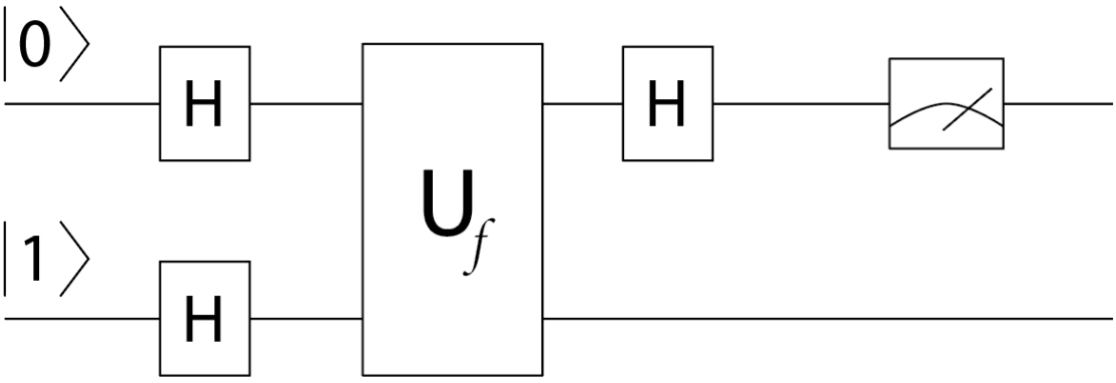
De la grafica anterior podemos ver que para una entrada de 11 en el anterior circuito la salida el 100% de las veces es de 10, resultado que cuadra perfectamente con el esperado por la matriz de Uf de la función 4.

2.3 Implementando el algoritmo de Deutsch en un computador cuántico

En esta sección primero se mostrará, como y porque funciona el algoritmo de Deutsch y luego se exiviran los resultados de implementar el algoritmo de Deutsch para las funciones expuestas anteriormente

Explicando el algoritmo de Deutsch:

Partamos de como luce el algoritmo de Deutsch en su versión mas general:



Iremos paso por paso explicando porque el circuito anterior funciona para determinar si una función f es constante o balanceada.

Primero analizaremos que pasa cuando el Qubit pasa por las primeras compuertas de Hadamard, al hacer los cálculos respectivos multiplicando y haciendo los productos tensores, obtenemos el siguiente resultado

$$|a\rangle = \frac{1}{2}\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{bmatrix}\begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} = \frac{1}{2}\begin{bmatrix} 1 \\ -1 \\ 1 \\ -1 \end{bmatrix}$$

Vector que podemos descomponer en los 4 vectores de la base canónica, teniendo el siguiente resultado

$$|a\rangle = \frac{1}{2}(|00\rangle - |01\rangle + |10\rangle - |11\rangle)$$

Y ahora pasaremos a calcular una formula general de los resultados una vez hallan pasado por la compuerta que representa a la función, para ello usaremos la formula que expusimos antes para encontrar los resultados en general de la función

$$|b\rangle = \frac{1}{2}(|0, 0 \otimes f(0)\rangle - |0, 1 \otimes f(0)\rangle) + |1, 0 \otimes f(1)\rangle - |1, 1 \otimes f(1)\rangle)$$

Y si simplificamos un poco la expresión, tenemos

$$|b\rangle = \frac{1}{2}(|0, f(0)\rangle - |0, \neg f(0)\rangle) + |1, f(1)\rangle - |1, \neg f(1)\rangle) \quad (1)$$

Y es con esta que trabajaremos para ver que sucede si llama a una función constante o una balanceada.

Primero asumiremos que f es constante, si f cumple esta propiedad significa que f(0) = f(1) por lo que para simplificar un poco la función (1) decidimos llamar estos valores “x” y ver que forma toma la ecuación

$$|b\rangle = \frac{1}{2}(|0, x\rangle - |0, \neg x\rangle) + |1, x\rangle - |1, \neg x\rangle)$$

Luego, de la anterior ecuación podemos ver que este vector es separable, por lo que lo representaremos con su versión expresada a través del producto tensor de 2 vectores,

pues así podríamos ver que esta pasando en el cable de arriba y en el cable de abajo por separado, teniendo así la siguiente ecuación

$$|b\rangle = \frac{1}{2}((|0\rangle + |1\rangle) \otimes (|x\rangle - |\neg x\rangle)) \quad (2)$$

Ecuación que llamaremos (2) pues la necesitaremos mas adelante.

Ahora, si asumimos que la función es balanceada, podríamos representar cada valor de la función de la siguiente manera

$$f(0) = x \rightarrow f(1) = \neg x$$

Y si reemplazamos con las anteriores ecuaciones en la ecuación (1), tendremos lo siguiente

$$|b\rangle = \frac{1}{2}((|0, x\rangle - |0, \neg x\rangle + |1, \neg x\rangle - |1, x\rangle)$$

Y de igual manera que antes separaremos el resultado anterior para poder ver que esta pasando independientemente en cada cable, obteniendo lo siguiente

$$|b\rangle = \frac{1}{2}((|0\rangle - |1\rangle) \otimes (|x\rangle - |\neg x\rangle)) \quad (3)$$

Y en donde podemos ver que la única diferencia que ocurre cuando una función es balanceada o constante es el signo que relaciona los vectores base en el primer cable, por lo que la intención en este momento seria poder diferenciar que signo esta entre estos, y eso es exactamente lo que la ultima matriz de Hadamard del circuito hace, como lo mostramos a continuación, primero con la función constante y luego con la función balanceada, para ello repartiremos el ½ como 1/sqrt(2) en cada vector del producto tensor, y tomaremos solo la parte que representa el cable de arriba para ver el resultado

$$|c\rangle = \frac{1}{\sqrt{2}}\begin{bmatrix}1 & 1\\ 1 & -1\end{bmatrix} * \frac{1}{\sqrt{2}}\begin{bmatrix}1\\ 1\end{bmatrix} = \frac{1}{2}\begin{bmatrix}2\\ 0\end{bmatrix} = \begin{bmatrix}1\\ 0\end{bmatrix} = |0\rangle$$

En donde podemos ver que el resultado al medir el cable de arriba siempre que una función sea constante seria de 0

Luego si hacemos los mismos cálculos, pero con el resultado que obtuvimos cuando la función es balanceada, tendríamos lo siguiente

$$|c\rangle = \frac{1}{\sqrt{2}}\begin{bmatrix}1 & 1\\ 1 & -1\end{bmatrix} * \frac{1}{\sqrt{2}}\begin{bmatrix}1\\ -1\end{bmatrix} = \frac{1}{2}\begin{bmatrix}0\\ 2\end{bmatrix} = \begin{bmatrix}0\\ 1\end{bmatrix} = |1\rangle$$

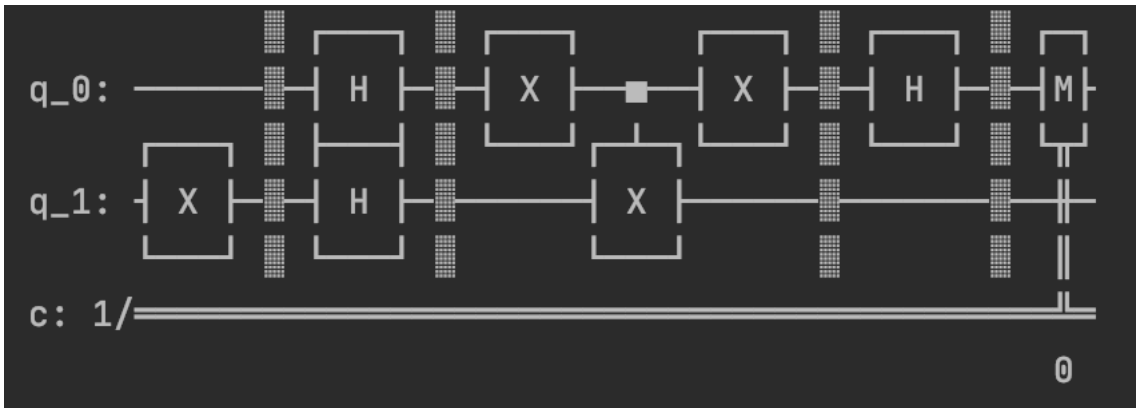
Donde podemos ver que, si la función es balanceada, siempre que midiéramos el cable de arriba el resultado seria de 1.

Teniendo así un algoritmo cuántico en donde podemos verificar si una función es balanceada o constante, pues si es constante el algoritmo de Deutsch medirá el cable de arriba con un valor de 0, y viceversa si es balanceada.

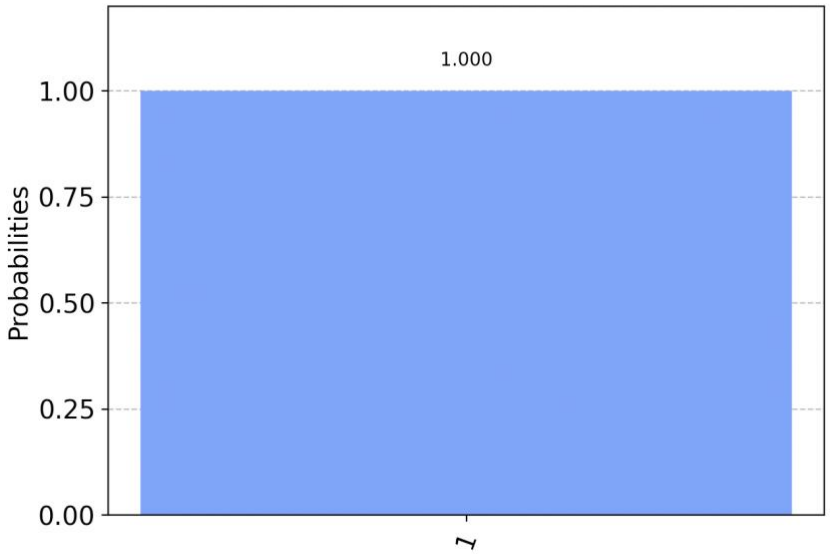
Resultados de la implementación del algoritmo de Deutsch con las 4 funciones:

- Función 1:

El circuito que obtuvimos al aplicar el algoritmo de Deutsch en la función 1 fue el siguiente:



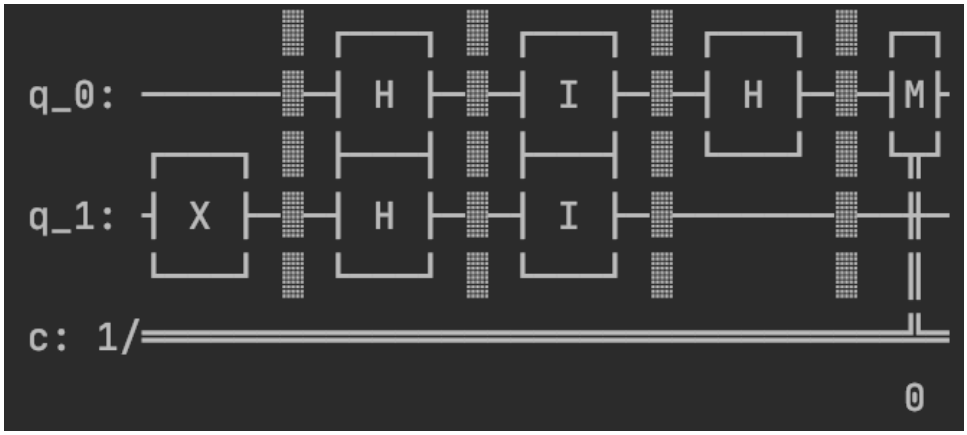
Y los resultados luego de lanzar 1000 Qubits al circuito fueron:



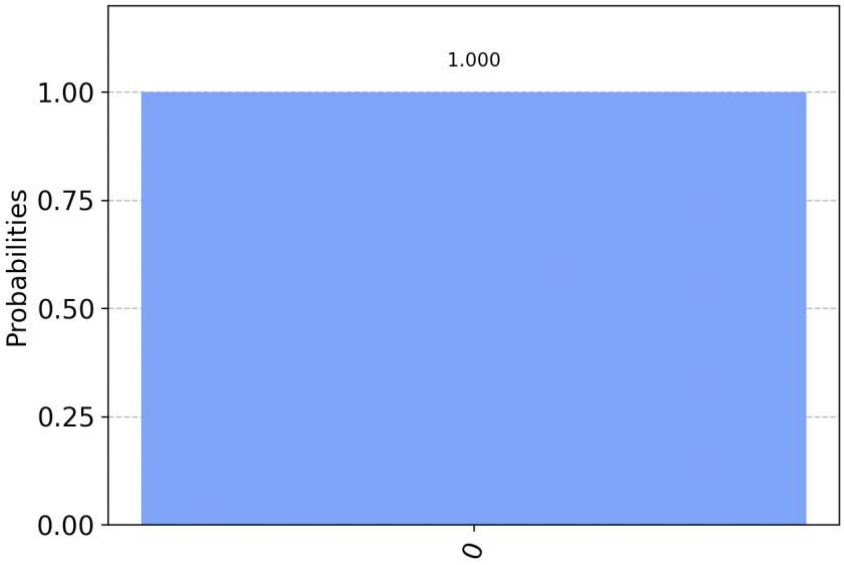
En donde podemos ver que el algoritmo nos dijo que esta función en concreto es balanceada, dato que sabemos que es verdadero por su representación grafica.

- Función 2:

El circuito que obtuvimos al aplicar el algoritmo de Deutsch en la función 2 fue el siguiente:



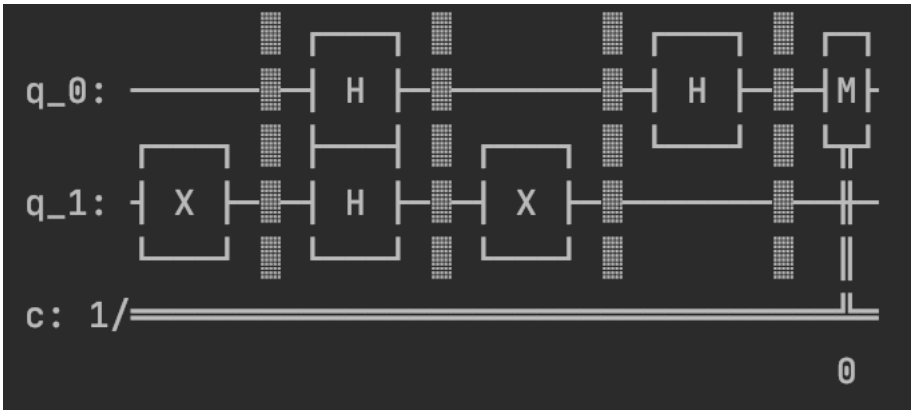
Y los resultados luego de lanzar 1000 Qubits al circuito fueron:



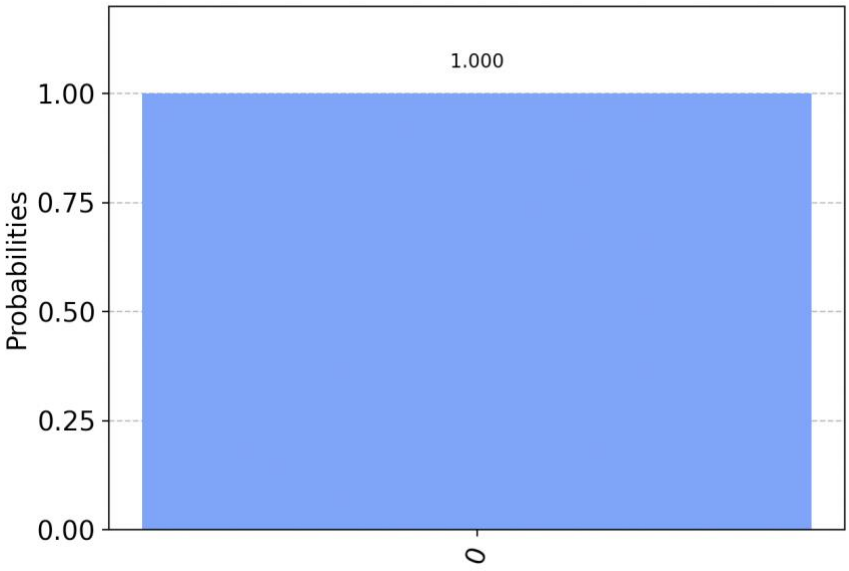
En donde podemos ver que el algoritmo nos dijo que esta función en concreto es constante, dato que sabemos que es verdadero por su representación grafica.

- Función 3:

El circuito que obtuvimos al aplicar el algoritmo de Deutsch en la función 3 fue el siguiente:



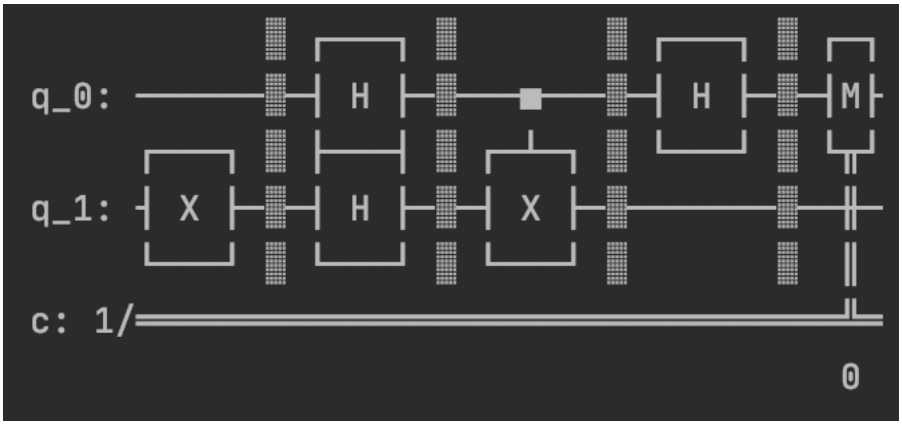
Y los resultados luego de lanzar 1000 Qubits al circuito fueron:



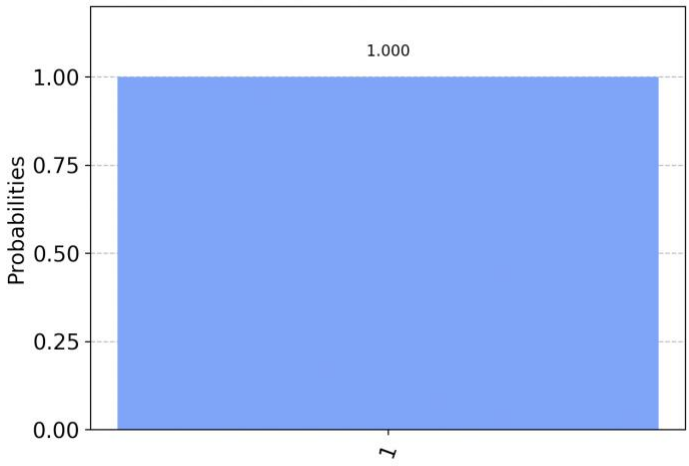
En donde podemos ver que el algoritmo nos dijo que esta función en concreto es constante, dato que sabemos que es verdadero por su representación grafica.

- Función 4:

El circuito que obtuvimos al aplicar el algoritmo de Deutsch en la función 4 fue el siguiente:



Y los resultados luego de lanzar 1000 Qubits al circuito fueron:



En donde podemos ver que el algoritmo nos dijo que esta función en concreto es balanceada, dato que sabemos que es verdadero por su representación grafica.

3 Algoritmo de Deutsch-Jozsa

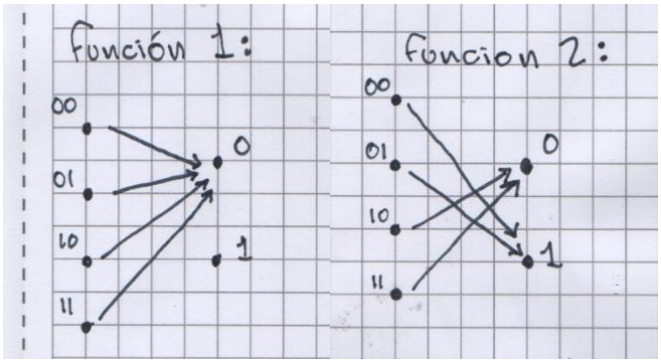
En el siguiente capitulo se explicará como el algoritmo de Deutsch-Jozsa, (el cual es solo una generalización de el algoritmo de Deutsch presentado en el capitulo anterior) funciona. Se explicará, porque funciona, como funciona y que tipos de funciones trabaja este.

3.1 Problema

Para poder entender el algoritmo de Deutsch-Jozsa primero debemos exponer con que tipo de funciones es con el que se trabaja este.

Las funciones que utiliza el algoritmo de Deutsch-Jozsa son tales que tienen su dominio en {0, 1}^n y su rango en {0, 1}.

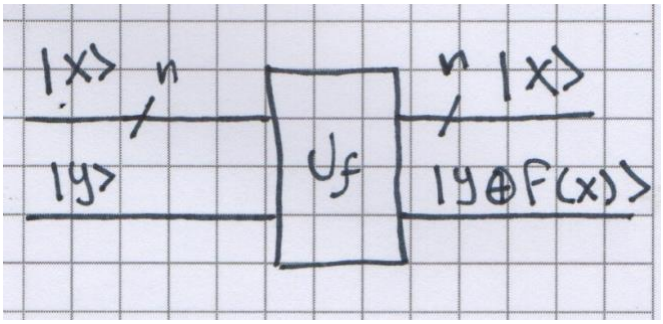
Unos ejemplos, y las que se van a usar en este informe podrían ser las 2 siguientes:



Ahora extenderemos la definición de función constante y balanceada de la sección anterior, definiremos una función constante, si a todos los elementos de su dominio, les corresponde la misma salida, y balanceada si exactamente a la mitad de los elementos del dominio les corresponde una salida, y a la otra mitad la otra salida.

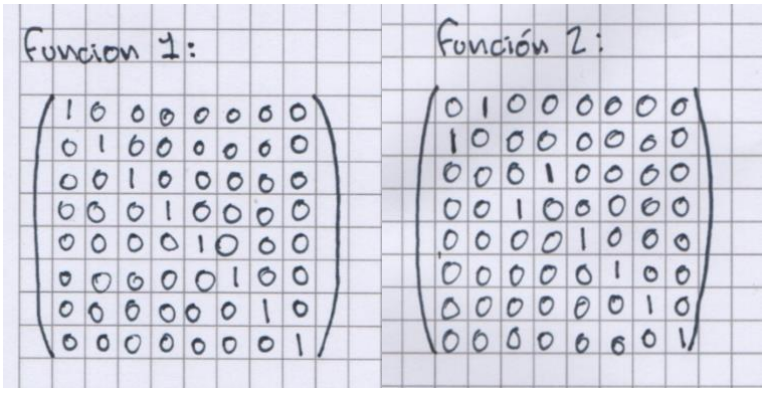
De modo que podemos ver que, según las funciones presentadas, la primera seria constante y la segunda balanceada.

Luego extenderemos también la "formula" general que nos permite hallar la matriz que representa a la función en un circuito cuántico de la siguiente manera:



En donde lo único que cambia es que el ket 'x' ya no representa un solo estado 0 o 1 sino una combinación de esos estados, pues con este representamos la entrada a la funcion que trabajamos, por lo que en realidad el cable de arriba no es un solo cable, por eso se le añade la barra con la n pues así denotamos que este cable representa n cables que entran a la función para poder arrojar otro valor.

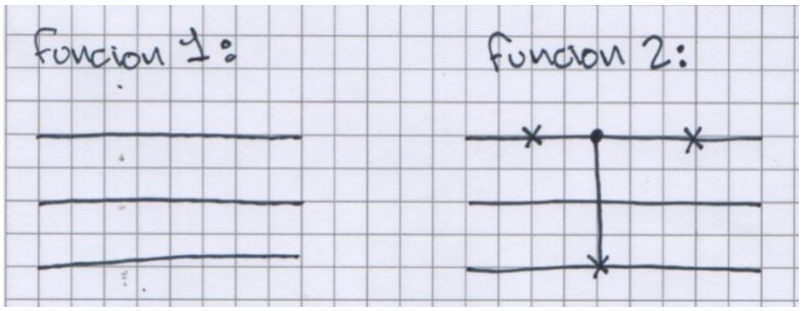
De esta manera y si se sigue el circuito anterior, podemos representar las funciones presentadas anteriormente, por las siguientes matrices



Y ahora con estas matrices podemos representar las funciones en un circuito cuántico, lo único que nos quedaría por hacer es encontrar que compuertas ya conocidas podemos usar para obtener el mismo resultado que si se aplicara la compuerta directa de Uf, para

así poder representar en el computador del IBM el circuito cuántico y obtener nuestros resultados.

Para ello simplemente analizamos patrones que siguieran cada estructura de las matrices y obtuvimos que los siguientes circuitos representan cada compuerta de Uf

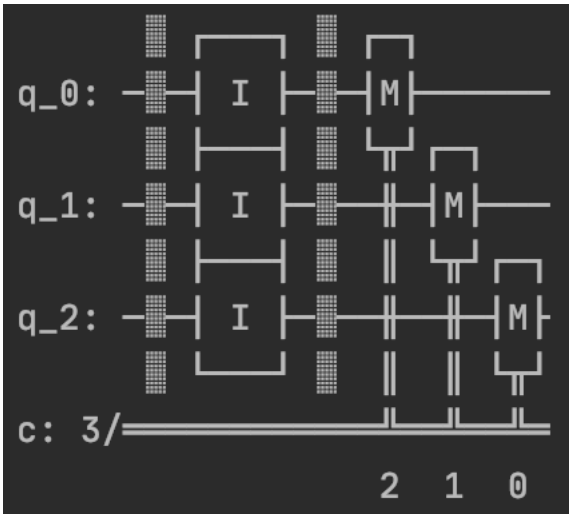


Y con esto ahora si podemos pasar a implementar los anteriores circuitos en un computador cuántico y ver nuestros resultados.

3.2 Implementando las funciones en el computador cuántico

- **Función 1:**

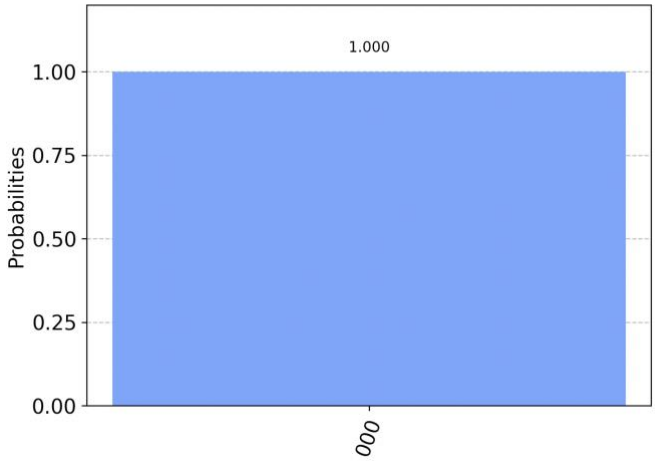
Para esta función el circuito básico que se construyo fue el siguiente:



Lo único que cambia relativamente a este es que antes de la primera barrera se agregaran compuertas de NOT para simular las diferentes entradas, pues como ya se explico en la introducción por convención las entradas siempre son 000 en estos circuitos. (No se mostrará pues con las anteriores funciones ya mostramos como van cambiando los circuitos según la entrada)

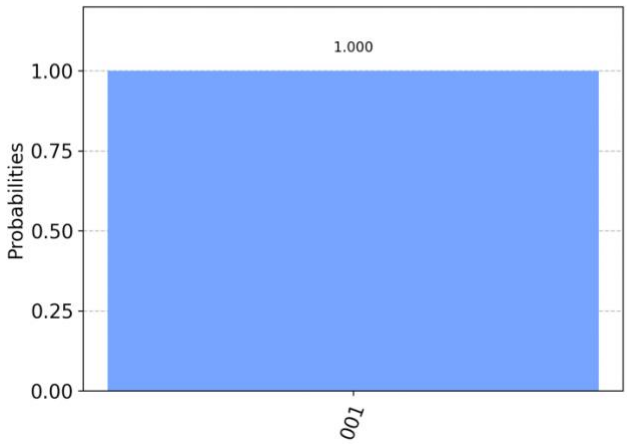
Luego se mostrarán los resultados de los disparos de diferentes Quibits a través del circuito:

- Entrada 000:



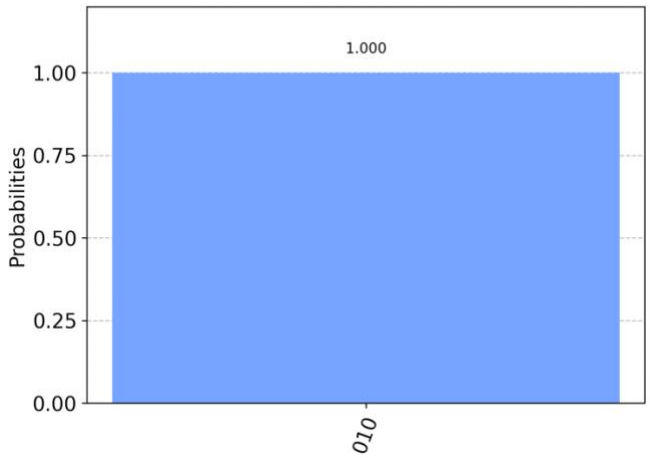
De la grafica anterior podemos ver que para una entrada de 000 en el anterior circuito la salida el 100% de las veces es de 000, resultado que cuadra perfectamente con el esperado por la matriz de Uf de la función 1.

- Entrada 001:



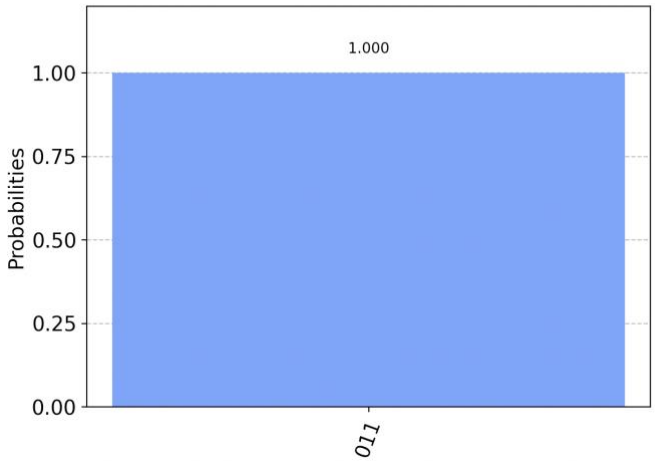
De la grafica anterior podemos ver que para una entrada de 001 en el anterior circuito la salida el 100% de las veces es de 001, resultado que cuadra perfectamente con el esperado por la matriz de Uf de la función 1.

- Entrada 010:



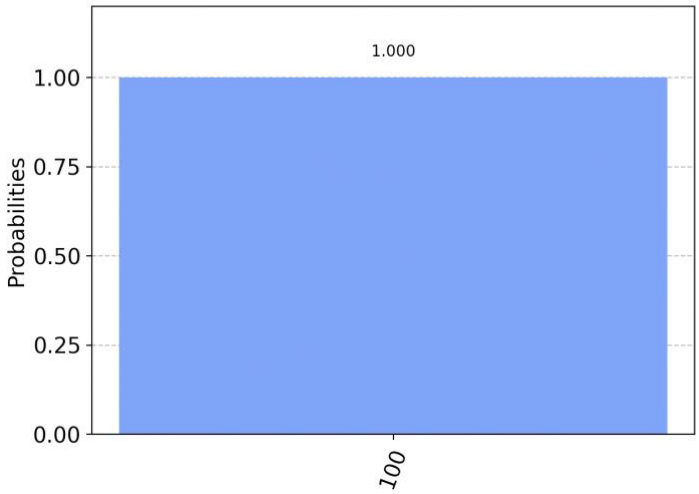
De la grafica anterior podemos ver que para una entrada de 010 en el anterior circuito la salida el 100% de las veces es de 010, resultado que cuadra perfectamente con el esperado por la matriz de Uf de la función 1.

- Entrada 011:



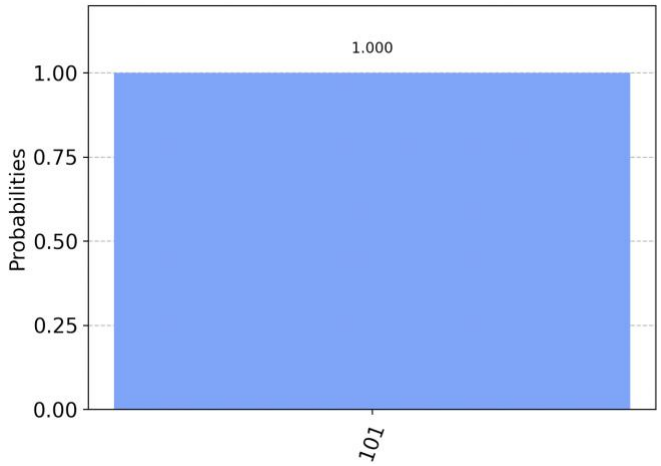
De la grafica anterior podemos ver que para una entrada de 011 en el anterior circuito la salida el 100% de las veces es de 011, resultado que cuadra perfectamente con el esperado por la matriz de Uf de la función 1.

- Entrada 100:



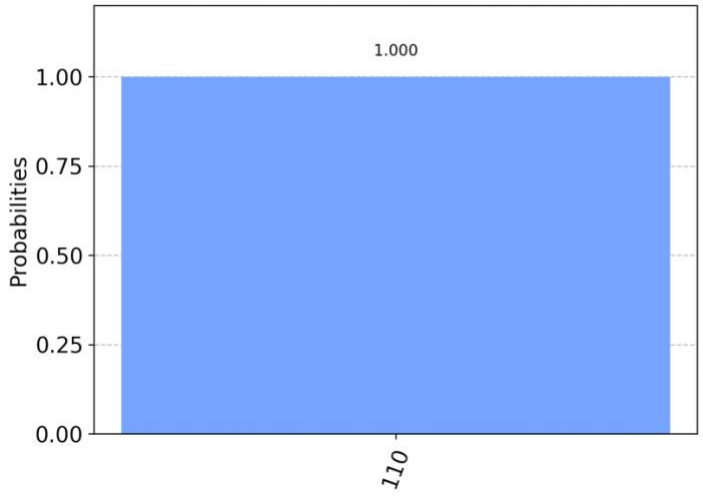
De la grafica anterior podemos ver que para una entrada de 100 en el anterior circuito la salida el 100% de las veces es de 100, resultado que cuadra perfectamente con el esperado por la matriz de Uf de la función 1.

- Entrada 101:



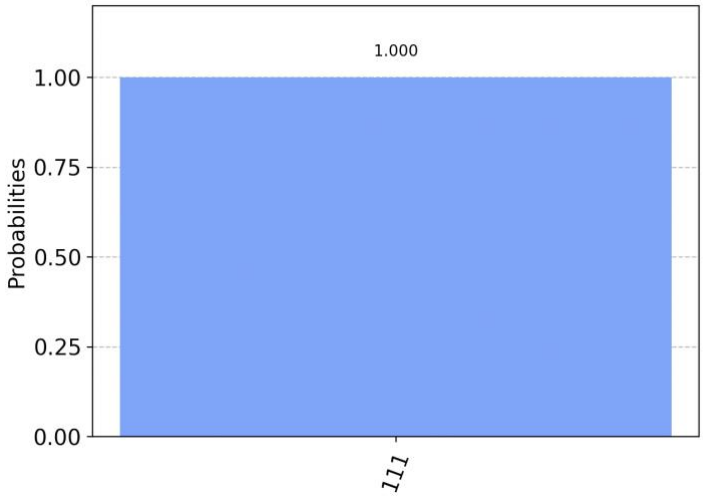
De la grafica anterior podemos ver que para una entrada de 101 en el anterior circuito la salida el 100% de las veces es de 101, resultado que cuadra perfectamente con el esperado por la matriz de Uf de la función 1.

- Entrada 110:



De la grafica anterior podemos ver que para una entrada de 110 en el anterior circuito la salida el 100% de las veces es de 110, resultado que cuadra perfectamente con el esperado por la matriz de Uf de la función 1.

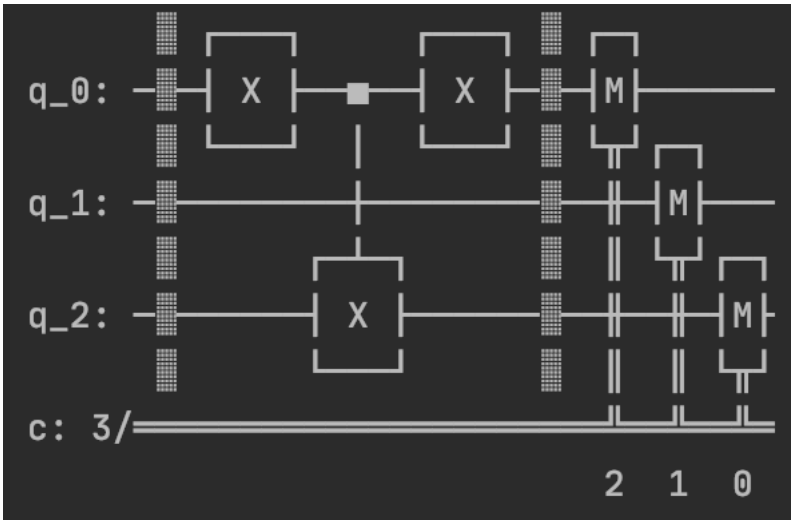
- Entrada 111:



De la grafica anterior podemos ver que para una entrada de 111 en el anterior circuito la salida el 100% de las veces es de 111, resultado que cuadra perfectamente con el esperado por la matriz de Uf de la función 1.

- **Función 2:**

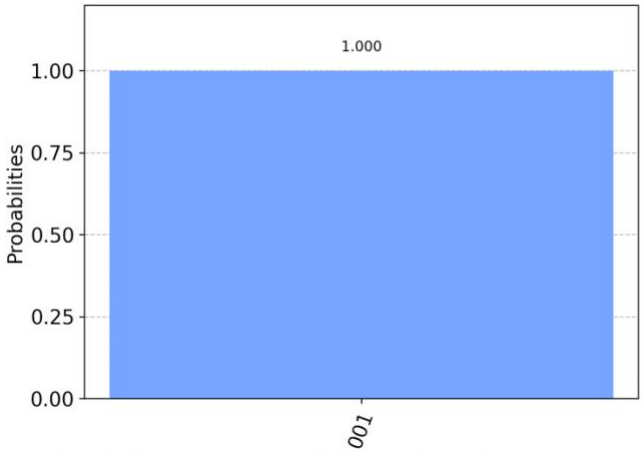
Para esta función el circuito básico que se construyo fue el siguiente:



Lo único que cambia relativamente a este es que antes de la primera barrera se agregaran compuertas de NOT para simular las diferentes entradas, pues como ya se explico en la introducción por convención las entradas siempre son 000 en estos circuitos. (No se mostrará pues con las anteriores funciones ya mostramos como van cambiando los circuitos según la entrada)

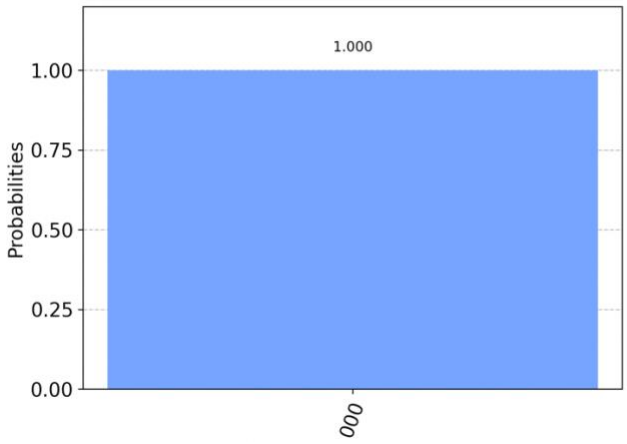
Luego se mostrarán los resultados de los disparos de diferentes Quibits a través del circuito:

- Entrada 000:



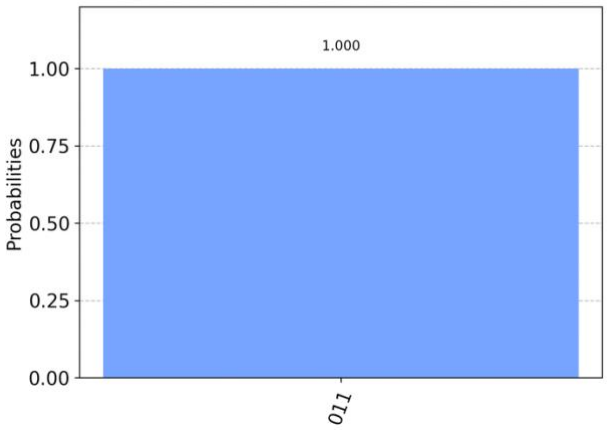
De la grafica anterior podemos ver que para una entrada de 000 en el anterior circuito la salida el 100% de las veces es de 001, resultado que cuadra perfectamente con el esperado por la matriz de Uf de la función 2.

- Entrada 001:



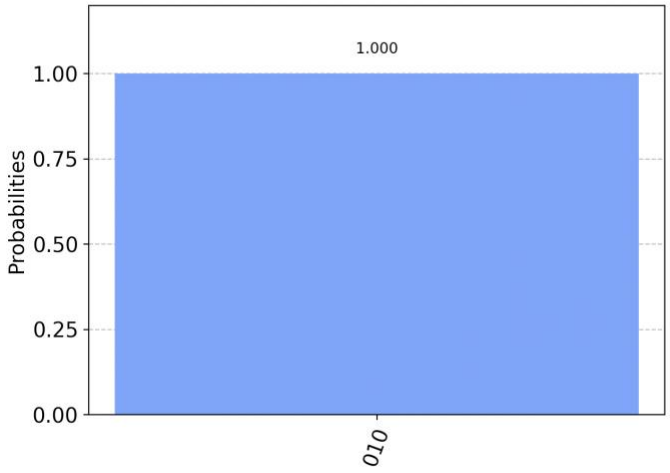
De la grafica anterior podemos ver que para una entrada de 001 en el anterior circuito la salida el 100% de las veces es de 000, resultado que cuadra perfectamente con el esperado por la matriz de Uf de la función 2.

- Entrada 010:



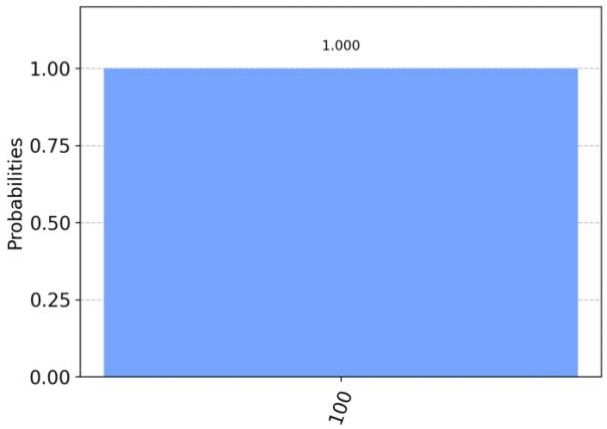
De la grafica anterior podemos ver que para una entrada de 010 en el anterior circuito la salida el 100% de las veces es de 011, resultado que cuadra perfectamente con el esperado por la matriz de Uf de la función 2.

- Entrada 011:



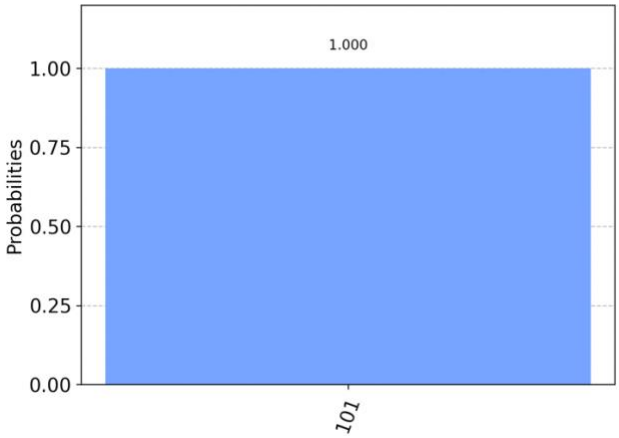
De la grafica anterior podemos ver que para una entrada de 011 en el anterior circuito la salida el 100% de las veces es de 010, resultado que cuadra perfectamente con el esperado por la matriz de Uf de la función 2.

- Entrada 100



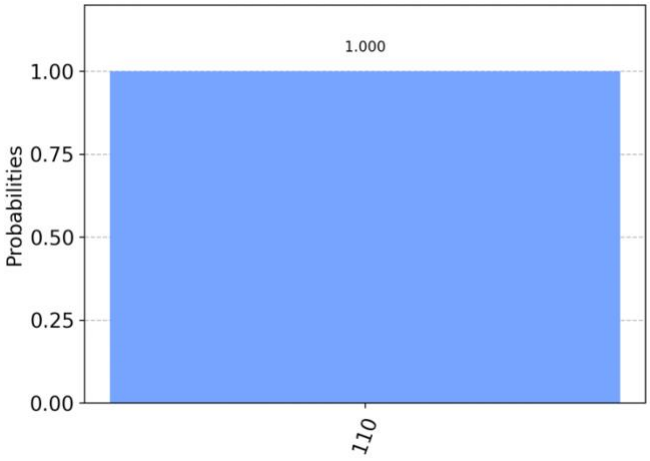
De la grafica anterior podemos ver que para una entrada de 100 en el anterior circuito la salida el 100% de las veces es de 100, resultado que cuadra perfectamente con el esperado por la matriz de Uf de la función 2.

- Entrada 101:



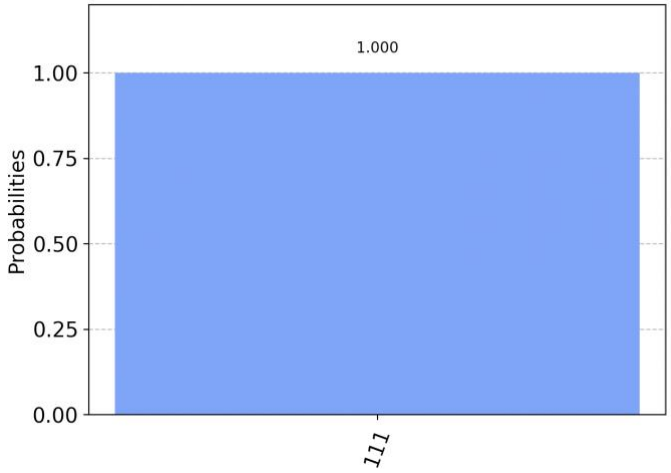
De la grafica anterior podemos ver que para una entrada de 101 en el anterior circuito la salida el 100% de las veces es de 101, resultado que cuadra perfectamente con el esperado por la matriz de Uf de la función 2.

- Entrada 110:



De la grafica anterior podemos ver que para una entrada de 110 en el anterior circuito la salida el 100% de las veces es de 110, resultado que cuadra perfectamente con el esperado por la matriz de Uf de la función 2.

- Entrada 111:



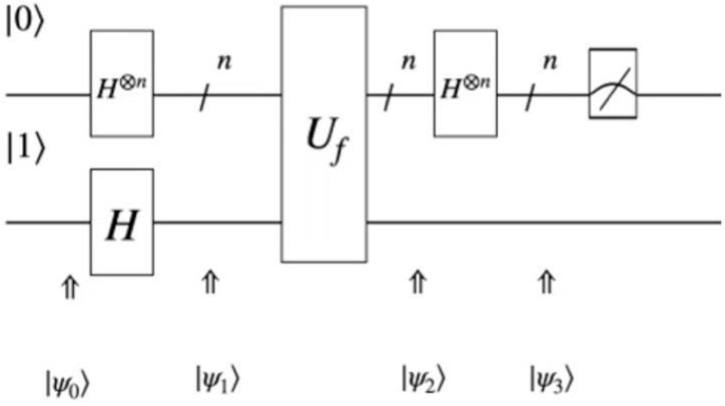
De la grafica anterior podemos ver que para una entrada de 111 en el anterior circuito la salida el 100% de las veces es de 111, resultado que cuadra perfectamente con el esperado por la matriz de Uf de la función 2.

3.3 Implementando el algoritmo de Deutsch-Jozsa en un computador cuántico

En esta sección primero se mostrará, como y porque funciona el algoritmo de Deutsch y luego se exiviran los resultados de implementar el algoritmo de Deutsch para las funciones expuestas anteriormente

Explicando el algoritmo de Deutsch-Jozsa:

En su forma mas general el algoritmo de Deutsch-Jozsa se ve de la siguiente manera:



Analizaremos este algoritmo que de misma manera que con el algoritmo de Deutsch miraremos como y porque funciona este, aunque los cálculos de para hallar cada estado

son un poco complejos, por lo que no se explicaran muy a fondo, y solo se enunciaran en la siguiente figura:

$$|\psi_0\rangle = |\mathbf{0}\rangle \otimes |1\rangle = |\mathbf{0},1\rangle = |00000...000,1\rangle$$

$$|\psi_1\rangle = H^{\otimes n} * |\mathbf{0}\rangle \otimes H * |1\rangle = [\frac{1}{\sqrt{2}^n} \sum_{x\in\{0,1\}^n} |x\rangle] \otimes [\frac{|0\rangle - |1\rangle}{\sqrt{2}}]$$

$$|\psi_2\rangle = U_f * (H^{\otimes n} * |\mathbf{0}\rangle \otimes H * |1\rangle) = [\frac{1}{\sqrt{2}^n} \sum_{x\in\{0,1\}^n} (-1)^{f(x)}|x\rangle] \otimes [\frac{|0\rangle - |1\rangle}{\sqrt{2}}]$$

$$|\psi_3\rangle = [\frac{1}{2^n} \sum_{x\in\{0,1\}^n} \sum_{z\in\{0,1\}^n} (-1)^{f(x)\oplus\langle z,x\rangle}|z\rangle] \otimes [\frac{|0\rangle - |1\rangle}{\sqrt{2}}]$$

En donde la parte que nos interesa es la parte izquierda del producto tensor del estado psi 3 pues esta representa el resultado de los vectores que aparecen en los primeros n cables del algoritmo que son los que se miden.

Luego antes de pasar a pensar considerar si f es balanceada o constante, queremos primero ver que forma toma la parte de la ecuación de los cables de arriba si calculamos la probabilidad que tiene este de colapsar a **0** una vez sean medidos los resultados, para ver si al asumir que f es constante se tiene algún resultado especial con el que se pueda diferenciar las funciones, teniendo asi lo siguiente:

Cuál es la probabilidad de $|\psi_3\rangle$ colapse a $|0\rangle$? Es decir $z = |0\rangle$, entonces $\langle z,x\rangle = \langle 0,x\rangle = 0$

$$|\psi_3\rangle = [\frac{1}{2^n} \sum_{x\in\{0,1\}^n} (-1)^{f(x)}|0\rangle] \otimes [\frac{|0\rangle - |1\rangle}{\sqrt{2}}]$$

Y ahora si asumimos que f(x) es constante, en ambos casos (solo se puede tener 2 posibles casos, que la función sea igual a 0 o a 1) tendríamos los siguientes resultados

Si $f(x) = 1$ (constante en 1), entonces $[\frac{1}{2^n} \sum_{x\in\{0,1\}^n} (-1)|0\rangle] = \frac{-(2^n)|0\rangle}{2^n} = -1|0\rangle$

Si $f(x) = 0$ (constante en 0), entonces $[\frac{1}{2^n} \sum_{x\in\{0,1\}^n} (1)|0\rangle] = \frac{(2^n)|0\rangle}{2^n} = 1|0\rangle$

Y ya que en ambos casos obtuvimos que si la función es constante el resultado seria un múltiplo del vector **0**, si al asumir que la función es balanceada los resultados dieran diferente, tendríamos una manera perfecta de diferenciar las funciones balanceadas y constantes.

Si $f(x)$ es balanceada, la sumatoria se cancela entonces $[\frac{1}{2^n}\sum_{x\in\{0,1\}^n}(-1)^{f(x)}|0\rangle] = 0|0\rangle$

Teniendo así que si la función es balanceada el resultado nunca será el vector 0 (pues ya que la mitad de las veces el resultado es 1 y la otra mitad es 0 el -1 elevado a estos valores se cancelarían los unos con otros), por lo que encontramos una manera de diferenciar las funciones a través del algoritmo de Deutsch-Jozsa.

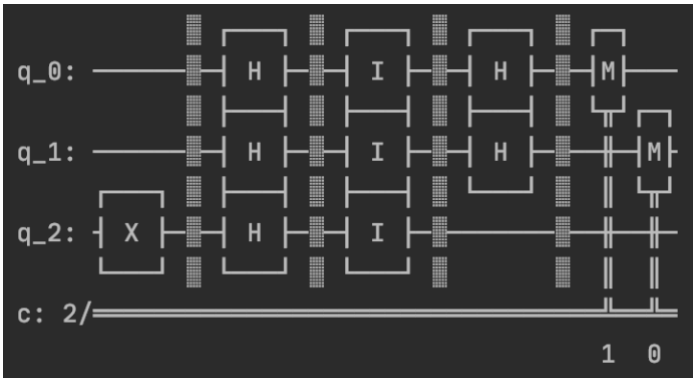
Por lo que en resumidas cuentas tenemos que el algoritmo de Deutsch-Josza retornaría lo siguiente en cada caso:

$$\begin{cases} |0\rangle \otimes [\frac{|0\rangle - |1\rangle}{\sqrt{2}}] & \text{si } f(x) \text{ es constante} \\ |\phi\rangle \otimes [\frac{|0\rangle - |1\rangle}{\sqrt{2}}] & \text{, con } \phi \neq |0\rangle \text{ y si } f(x) \text{ es balanceada} \end{cases}$$

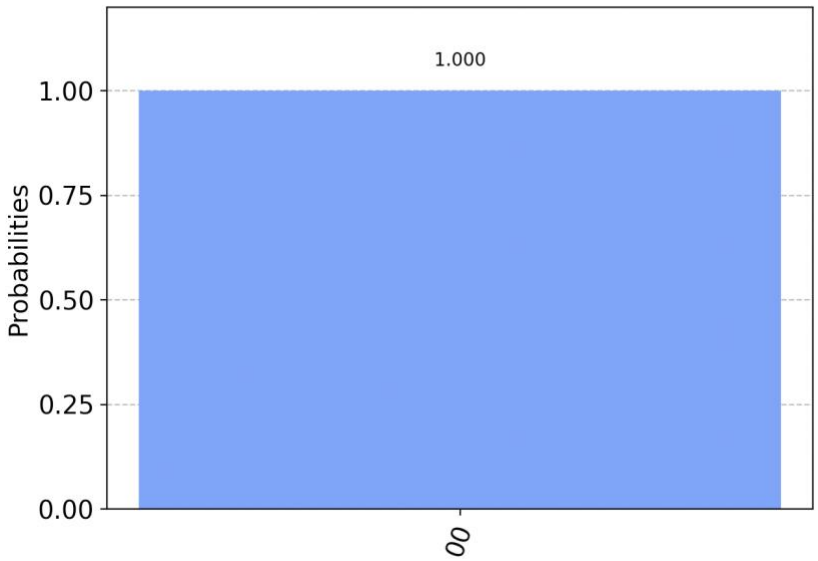
Resultados de la implementación del algoritmo de Deutsch con las 4 funciones:

- Función 1:

El circuito que obtuvimos al aplicar el algoritmo de Deutsch en la función 1 fue el siguiente:



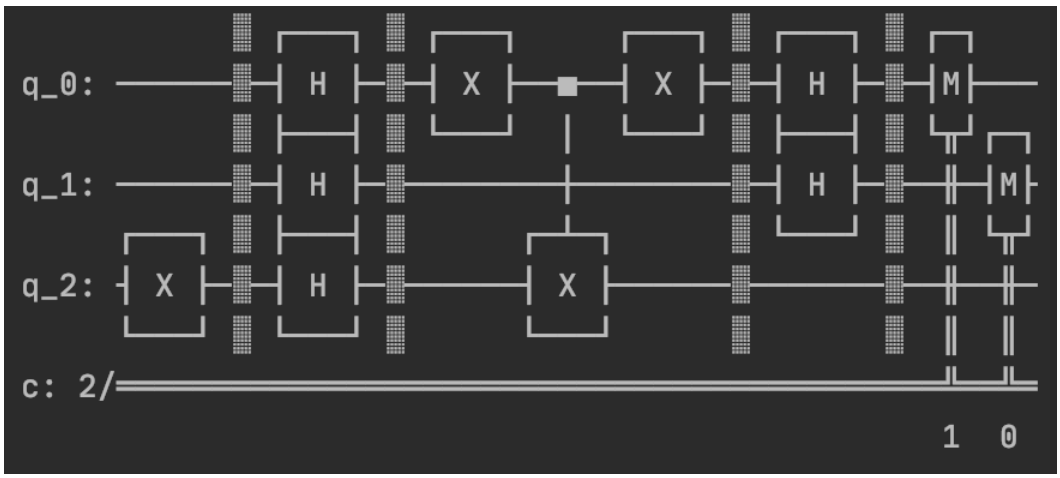
Y los resultados luego de lanzar 1000 Qubits al circuito fueron:



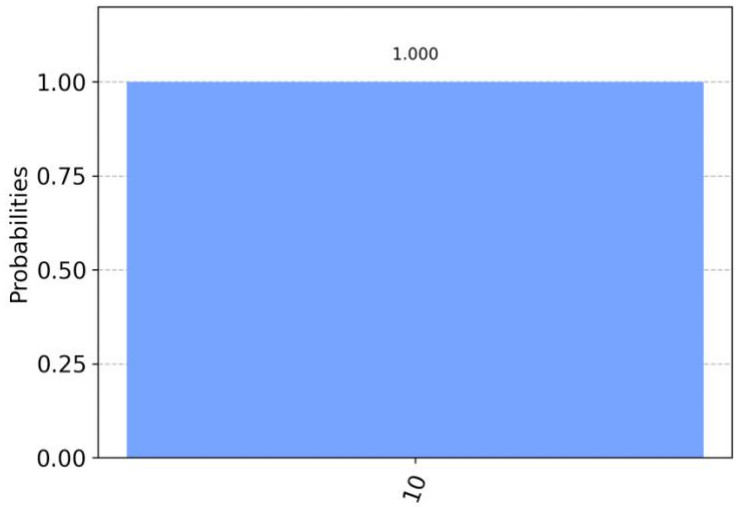
En donde podemos ver que el algoritmo nos dijo que esta función en concreto es constante, dato que sabemos que es verdadero por su representación grafica.

- Función 2:

El circuito que obtuvimos al aplicar el algoritmo de Deutsch en la función 2 fue el siguiente:



Y los resultados luego de lanzar 1000 Qubits al circuito fueron:



En donde podemos ver que el algoritmo nos dijo que esta función en concreto es balanceada, dato que sabemos que es verdadero por su representación grafica.

4 Conclusiones

En este informe pudimos ver primero como funcionan las funciones de {0, 1} a {0, 1} y como representarlas a través de compuertas cuánticas, para poder explicar como funciona algoritmo de Deutsch e implementarlo para poder comprobar sus resultados, para luego explicar las funciones de {0, 1}^n a {0, 1} para poder representarlas después por compuertas cuánticas las cuales pudiéramos utilizar para luego implementar el algoritmo de Deutsch-Jozsa y comprobar que funciones son balanceadas y cuales no.
En este informe pudimos ver como los algoritmos cuánticos pueden llegar a ser muchas veces mas eficientes que los algoritmos clásicos, pues como vimos en el caso del algoritmo de Deutsch-Jozsa, si quisiéramos implementarlo en un computador clásico, en el peor de los casos debíamos llamar la función un mínimo de 2^(n - 1) veces, mientras que para un algoritmo cuántico solo debemos llamar función una sola vez cuando los Qubits estén pasando por la compuerta de Uf, haciendo extremadamente mas eficiente el algoritmo de Deutsch-Jozsa en un computador cuántico que en uno clásico.

Como explicamos antes, con el algoritmo de Deutsch-Jozsa, los computadores cuánticos tienen la capacidad de mejorar el tiempo de ejecución de este de O(2^n) a O(1) usando el multiverso para realizar nuestras calculaciones, y quien sabe en cuantos mas algoritmos el tiempo de ejecución baje tan drásticamente, o pueda que problemas que antes eran intratables en un computador clásico, sean posibles de resolver en uno cuántico, es tan poderosa que esta podría ser hasta un camino para resolver el gran problema matemático y de computación de P-NP solo por dar un ejemplo. El caso es que en el momento en el que estamos no tenemos idea de que limites pueda tener la computación cuántica, lo único que si sabemos es que es infinitamente mas poderosa y eficiente que la manera de computación actual, la computación clásica.

5 Bibliografía

- Noson S. Yanofsky, Mirco A. Mannucci. (2008) Quantum Computing for Computer Scientist
- https://www.cnet.com/news/ibm-now-has-18-quantum-computers-in-its-fleet-of-weird-machines/
- https://www.bbc.com/mundo/noticias-46833112