Implementación y análisis de algoritmo de Deutsch y Deutsch-Jozsa

**Juan David Parroquiano Roldán**

**Institución: Escuela colombiana de ingeniería Julio Garavito**

**Correo electrónico: juan.parroquiano@mail.escuelaing.edu.co**

**26/nov/2021**

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

# Tabla de contenidos

[Tabla de contenidos 1](#_Toc39509128)

[1 Introducción 1](#_Toc39509129)

[2 Algoritmo de Deutsch 2](#_Toc39509130)

[2.1 Problema 2](#_Toc39509131)

[2.2 Implementando las funciones en el computador cuántico 2](#_Toc39509132)

[2.3 Implementando el algoritmo de Deutsch en un computador cuántico 2](#_Toc39509133)

[3 Algoritmo de Deutsch-Jozsa 2](#_Toc39509134)

[3.1 Problema 2](#_Toc39509135)

[3.2 Implementando las funciones en el computador cuántico 2](#_Toc39509136)

[3.3 Implementando el algoritmo de Deutsch-Josza en un computador cuántico 2](#_Toc39509137)

[4 Conclusiones 2](#_Toc39509138)

[5 Bibliografía 2](#_Toc39509139)

# Introducción

Computación cuántica

La computación cuántica es la encargada de desarrollar tecnologías informáticas a partir de los principios de teoría cuántica. Según las leyes de la física cuántica, la tremenda

capacidad de procesamiento de los ordenadores cuánticos se deriva de su capacidad de estar en múltiples estados y realizar tareas utilizando todas las permutaciones posibles de manera simultánea.

Hay muchas maneras de entender por qué la mecánica cuántica es difícil de simular. Quizás la más sencilla de entender es la interpretación de la teoría cuántica que dice que la materia, en el nivel cuántico, está en multitud de configuraciones posibles (conocidas como estados). A diferencia de la teoría clásica de las probabilidades, estas numerosas configuraciones del estado de un cuanto, que se podrían llegar a observar, pueden interferir entre sí como las olas en una piscina de mareas. Esta interferencia impide el uso del muestreo estadístico para obtener las configuraciones del estado cuántico. En su lugar, tenemos que hacer un seguimiento de todas las configuraciones posibles en las que un sistema cuántico podría estar si queremos comprender la evolución de los cuantos.

Tomemos un sistema de electrones en el que los electrones pueden estar en cualquiera de 40 posiciones. Por lo tanto, los electrones pueden estar en cualquiera de las configuraciones (porque cada posición puede tener o no un electrón). Para almacenar el estado cuántico de los electrones en la memoria de un equipo convencional, se necesitarían más de 130 GB de memoria. Esto es importante, pero factible para algunos equipos. Si permitimos que las partículas estén en cualquiera de las 41 posiciones, tendríamos el doble de configuraciones en que, a su vez, necesitaría más de 260 GB de memoria para almacenar el estado cuántico. Este juego de aumentar el número de posiciones no se puede reproducir indefinidamente si se desea almacenar el estado convencionalmente, porque superaríamos rápidamente las capacidades de memoria de las máquinas más eficaces del mundo. Con unos cientos de electrones, la memoria necesaria para almacenar el sistema supera el número de partículas del universo; por lo tanto, no hay ninguna esperanza de

que nuestros equipos convencionales simulen su dinámica cuántica. Y, sin embargo, en la naturaleza, estos sistemas evolucionan fácilmente en el tiempo según las leyes de la mecánica cuántica, completamente inconsciente de su incapacidad para diseñar y simular su evolución con la potencia de la informática convencional.

Esta observación llevó a los primeros visionarios de la computación cuántica a formular una pregunta sencilla pero impactante: ¿podemos convertir esta dificultad en una oportunidad? En concreto, si la dinámica cuántica es difícil de simular, ¿qué sucedería si se creara hardware que tuviera efectos cuánticos como operaciones fundamentales? ¿Podríamos simular sistemas de partículas en interacción utilizando un sistema que aprovecha exactamente las mismas leyes que los rigen de forma natural? ¿Podríamos investigar las tareas que no existen en la naturaleza y aún así seguir o aprovechar las leyes de la mecánica cuántica? Estas preguntas condujeron a la aparición de la computación cuántica. La base fundamental de la computación cuántica consiste en almacenar información en los estados cuánticos de la materia y usar operaciones de puertas cuánticas para realizar procesos con dicha información, aprovechando y aprendiendo a "programar" la interferencia cuántica. Uno de los primeros ejemplos de la programación de interferencias para solucionar un problema que se consideraba difícil en nuestros equipos convencionales lo planteó Peter Shor en 1994 para un problema conocido como factorización. La solución de la factorización conlleva la posibilidad de vulnerar muchos de nuestros sistemas de cifrado de claves públicas que subyacen a la seguridad del comercio electrónico actual, incluidas la criptografía de curva elíptica y RSA. A partir de entonces, se han desarrollado algoritmos de computación cuántica eficientes para muchas de las tareas difíciles clásicas: simular sistemas físicos en química, física y ciencia de los materiales, buscar en una base de datos desordenada, resolver sistemas de ecuaciones lineales y aprendizaje automático.

En este reporte:

Este reporte presenta dos algoritmos que pueden ser usados en computación cuántica como lo son el algoritmo de Deutsch y el de Deutsch – Joza, por medio de graficas y pruebas lograremos comprobar el correcto funcionamiento de los mismos identificando si una función es constante o balanceada.

# Algoritmo de Deutsch

El algoritmo de Deutsch es uno de los primeros algoritmos formulados que tiene aplicación a la computación cuántica, su función es básicamente determinar si una función es constante o balanceada. En este capítulo presentamos por medio de gráficas y del entorno de desarrollo pycharm la implementación de este algoritmo y las posibles funciones que se pueden presentar para determinar si son balanceadas o constantes.

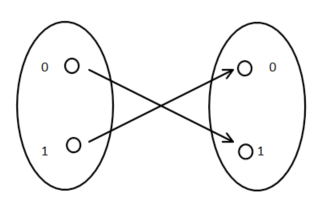
## Problema

Consideremos una función f:{0,1}−→{0,1}. De las posibles funciones f existen cuatro de las cuales dos son constantes es decir f(0) = f(1) y otras dos balanceadas es decir f(0) != f(1) . El objetivo del algoritmo de Deutsch es el de determinar a cuál de las dos clases pertenece una función dada. En computación cuántica es evidente que para resolver el problema debemos evaluar a esta función por lo menos dos veces. Veremos que para la computación cuántica, al menos en teoría, es capaz de resolver el problema “evaluando” la función una sola vez.

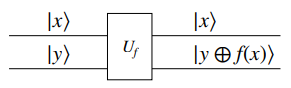
## Implementando las funciones en el computador cuántico

### Función cruzada

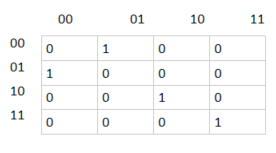
La siguiente grafica representa una función cruzada que indica el hecho que cuando el valor de la entrada en cero su imagen será uno y por el contrario cuando su entrada es uno su imagen será cero.



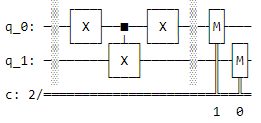
A partir de esta grafica podemos calcular su matriz correspondiente con ayuda del siguiente circuito:



Este circuito nos indica que para la primera entrada no será afectado su valor de salida, pero en cuando la segunda entrada se le aplicara la operación y Ꚛ f(x) en donde el valor de f(x) es posible encontrarlo con el grafico anterior. Como resultado obtenemos la siguiente matriz.

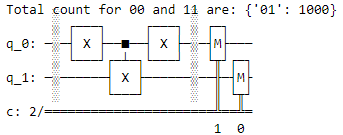


A partir de la matriz encontrada podemos observar un patrón en ella, en donde la segunda entrada será negada siempre y cuando la primera tenga valor cero. Este patrón nos va a ayudar ha plantear un circuito para esta función.

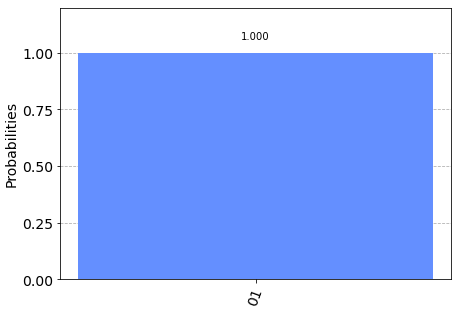


Con este circuito implementado en el IDLE de Pycharm podemos verificar que los valores encontrados en la matriz son los correctos.

Entrada |00>:



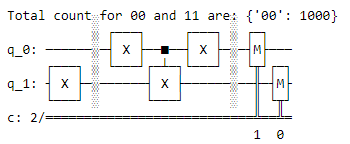
Circuito con entradas por defecto |00>



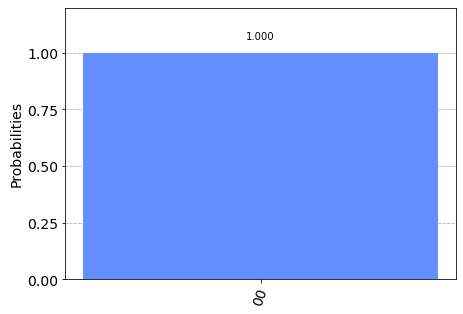
Grafica indicando que el 100%

de las veces el resultado será 01

Entrada |01>

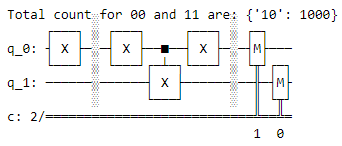


Circuito con entradas |01>

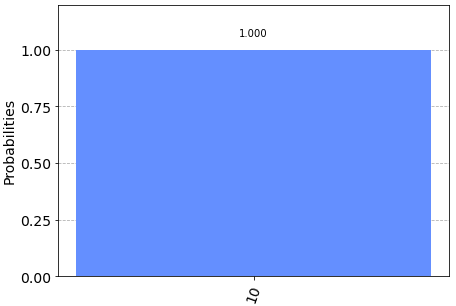


Grafica indicando que el 100% de las veces el resultado será 00

Entrada |01>

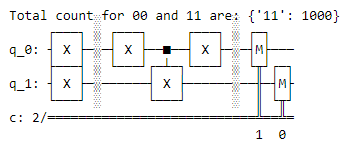


Circuito con entradas |10>

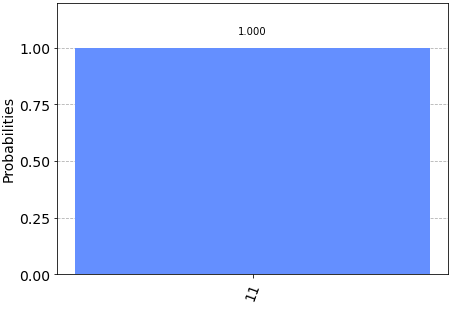


Grafica indicando que el 100% de las veces el resultado será 10

Entrada |11>



Circuito con entradas |11>

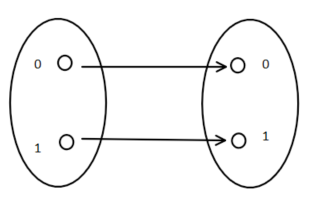


Grafica indicando que el 100% de las veces el resultado será 11

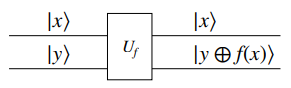
Con los resultados de las diferentes graficas podemos ver que los datos obtenidos en la matriz son correctos.

### Función directa

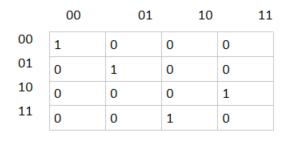
Esta es una función lineal muestra que los valores de las entradas son directamente proporcionales a los valores de las salidas.



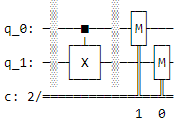
A partir de esta grafica vamos a encontrar su matriz correspondiente.



Con ayuda de esta función explicada anteriormente podemos encontrar los valores de la correspondiente matriz.

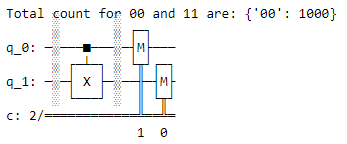


El patrón para esta matriz nos indica que cuando su primera entrada tenga el valor de uno la segunda entrada se va a negar. Al identificar este patrón podemos plantear nuestro circuito.

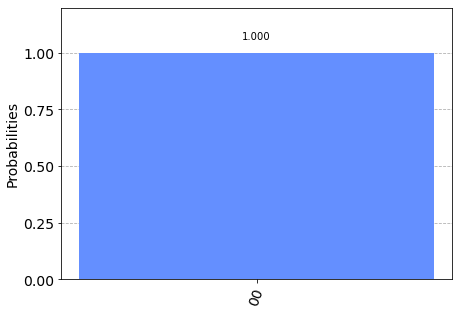


Con este circuito implementado en el IDLE de Pycharm podemos verificar que los valores encontrados en la matriz anterior sean los correctos.

Entrada |00>:

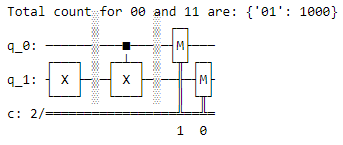


Circuito con entrada estándar |00>

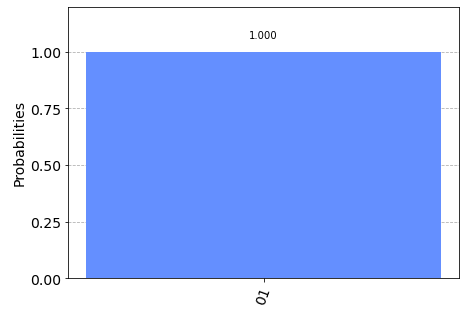


Grafica indicando que el 100% de las veces el resultado será 00

Entrada |01>:

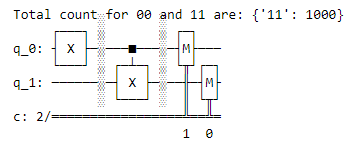


Circuito con entrada |01>

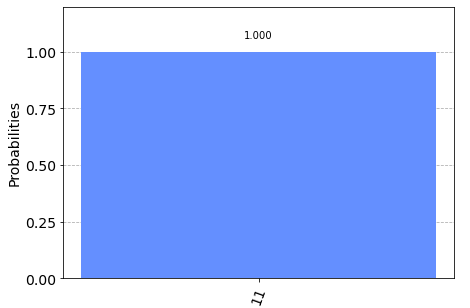


Grafica indicando que el 100% de las veces el resultado será 01

Entrada |10>:

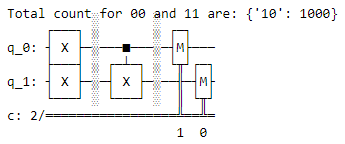


Circuito con entrada |10>

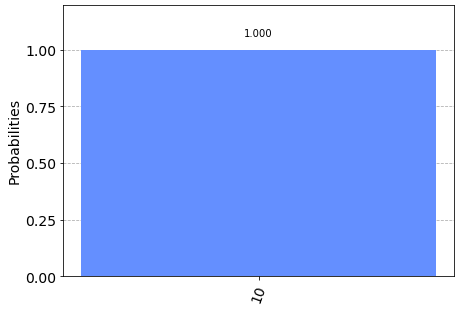


Grafica indicando que el 100% de las veces el resultado será 10

Entrada |11>:



Circuito con entrada |11>

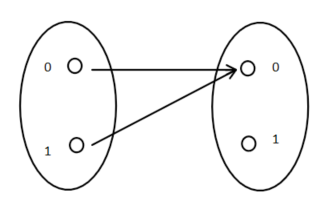


Grafica indicando que el 100% de las veces el resultado será 11

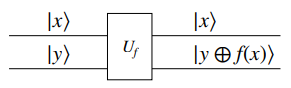
Los datos de las gráficas luego de implementar los circuitos nos indicas que coinciden con los obtenidos anteriormente en la matiz por lo tanto tiene una implementación correcta.

### Función todos a cero

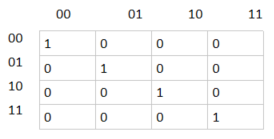
La función todos a cero nos muestra que todo el conjunto entradas cuenta con el mismo valor de salida cero.



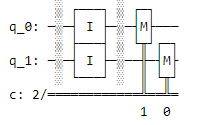
El paso siguiente será armar la matriz correspondiente a esta función con ayuda del siguiente circuito.



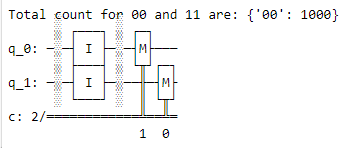
Los valores obtenidos para la matriz correspondiente a la función Todos a cero son :



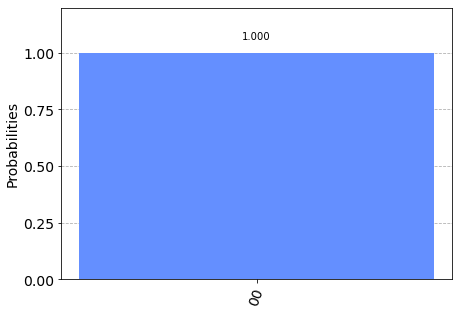
Para la matriz encontrada se puede ver claramente que todos los valores de entrada son los mismos valores de salida. A partir de esto podemos armar un circuito en nuestra computadora cuántica.



Entrada |00>:

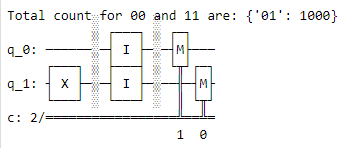


Circuito con entrada estándar |00>

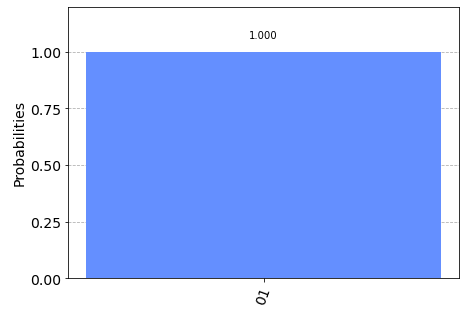


Grafica indicando que el 100% de las veces el resultado será 00

Entrada |01>:

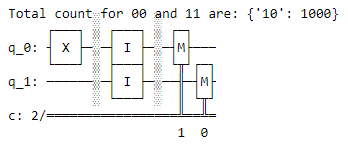


Circuito con entrada |01>

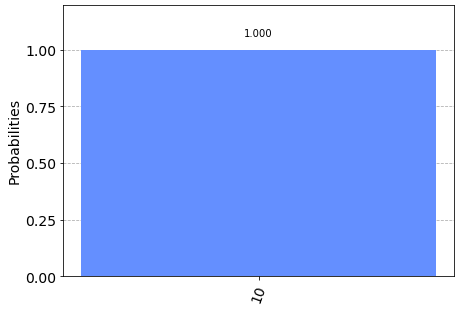


Grafica indicando que el 100% de las veces el resultado será 01

Entrada |10>:

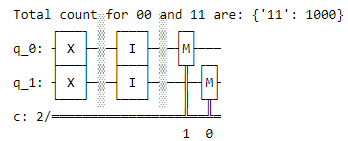


Circuito con entrada |10>

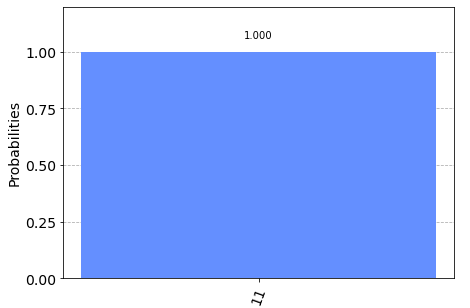


Grafica indicando que el 100% de las veces el resultado será 10

Entrada |11>:



Circuito con entrada |11>

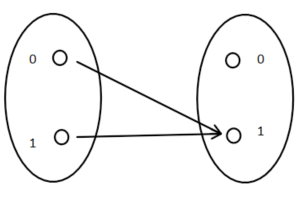


Grafica indicando que el 100% de las veces el resultado será 11

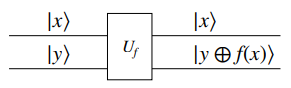
Al verificar los valores por medio de las graficas nos damos cuenta de que coinciden con los obtenidos en la matriz anterior. Por lo tanto la implementación del circuito es correcta

### Función Todos a uno

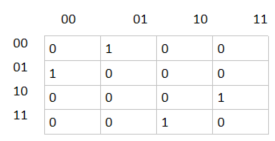
A diferencia de la función anterior esta relaciona todo el conjunto de las entradas con un solo valor de salida “uno”.



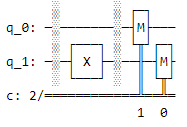
A partir de esta grafica vamos a encontrar su matriz correspondiente.



Con ayuda de este circuito explicado anteriormente podemos encontrar los valores de la correspondiente matriz.

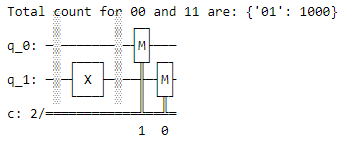


El patrón que podemos observar en esta matriz es el siguiente: para los segundos valores de entradas siempre su salida será negada mientras que para el primer valor no lo afecta. Partiendo de esta idea formulamos un circuito en nuestro computador cuántico.

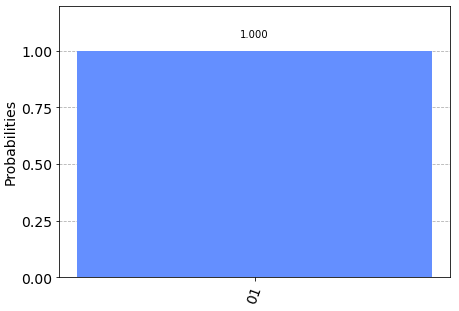


Teniendo este circuito vamos a corroborar que los valores encontrados en la matriz son los correctos.

Entrada |00>:

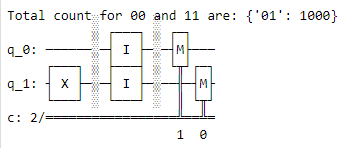


Circuito con entrada estándar |00>

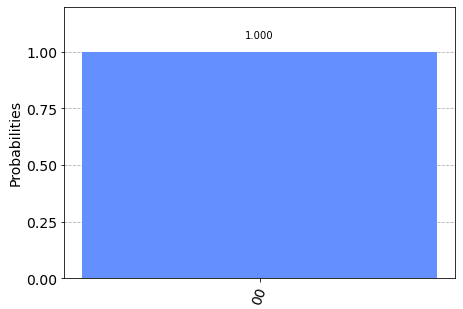


Grafica indicando que el 100% de las veces el resultado será 01

Entrada |01>:

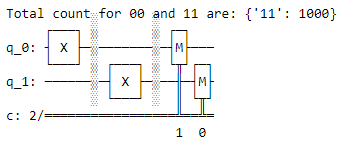


Circuito con entrada |01>

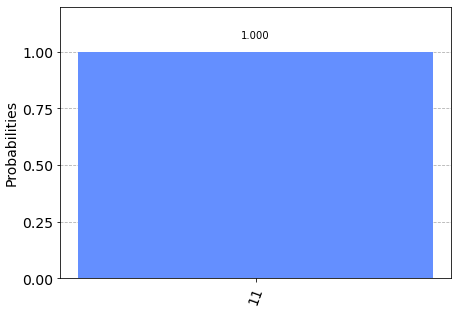


Grafica indicando que el 100% de las veces el resultado será 00

Entrada |10>:

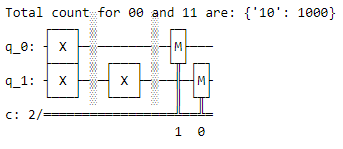


Circuito con entrada |10>

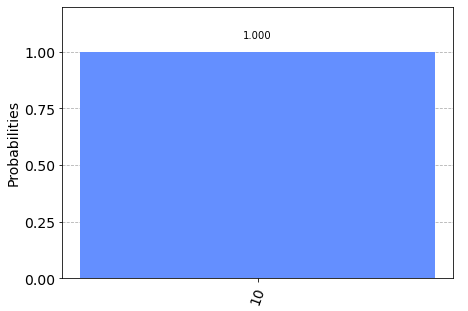


Grafica indicando que el 100% de las veces el resultado será 11

Entrada |11>:



Circuito con entrada |11>



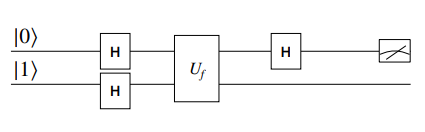
Grafica indicando que el 100% de las veces el resultado será 10

Los valores en las graficas corresponden a los valores de la gráfica para la fusión Todos a uno indicando que la implementación del circuito esta correcta.

## Implementando el algoritmo de Deutsch en un computador cuántico

En este capítulo presentamos la implementación del algoritmo de Deutsch verificando si las funciones presentadas anteriormente son constantes o balanceadas.

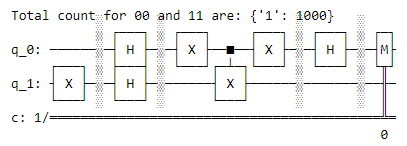
El siguiente es el circuito que representa al algoritmo de Deutsch donde Uf representa cada una de las funciones que deseamos probar. Sabremos si una función es constante cuando el algoritmo nos indique el valor de cero y sabremos si es balanceada si el resultado del circuito es uno.



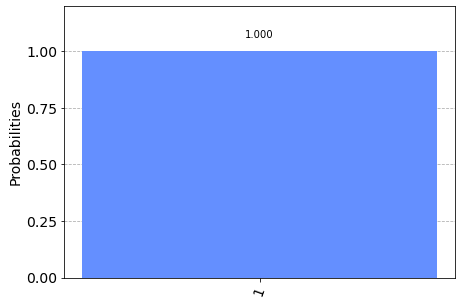
Las entradas iniciales para este circuito son para el primer hilo cero y el segundo hilo uno, se aplican una compuerta de Hadamard por cada uno luego ubicamos la función que deseamos evaluar, añade un ultima compuerta Hadamard al primer hilo y por ultimo tenemos la medición sobre este mismo hilo.

### Función cruzada con Deutsch

Queremos encontrar el tipo para la función cruzada presentada anteriormente. Esta función indica que para la mitad de las imágenes en los elementos de entrada, es diferente a las imagen de la otra mitad



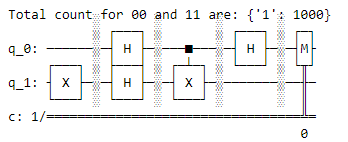
Circuito donde aplicamos la función Cruzada al algoritmo de Deutsch.

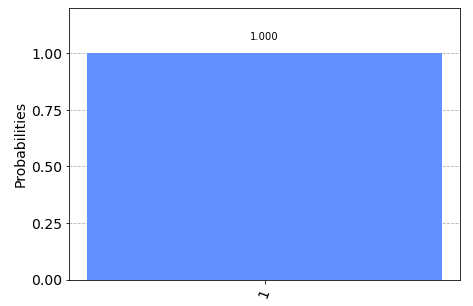


Esta gráfica nos representa el 100 % de las probabilidad de que el valor sea 1 el cual corresponde a ser una función balanceada.

### Función directa

Esta es una función lineal donde muestra que los valores de las entradas son directamente proporcionales a los valores de las salidas. Sabemos que la misma cantidad de entradas que tiene salidas cero son la misma cantidad de entradas que tienen salida uno, por lo tanto podemos decir que esta es una función balanceada y que el resultado de la implementación del circuito será 1.

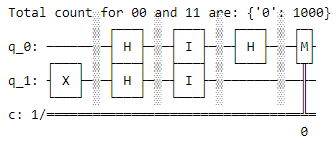


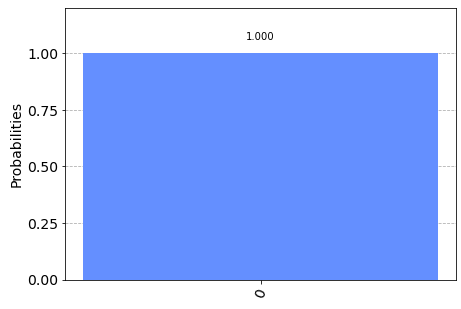


Este gráfico nos muestra el 100% de probabilidad de que el resultado sea 1. Por lo tanto podemos afirmar que corresponde a una función balanceada.

### Función Todos a cero

La función todos a cero nos muestra que todo el conjunto entradas cuenta con el mismo valor de salida a cero. Esta función al tener en todos sus elementos de entrada apuntando al valor cero decimos que es una función constante y que el resultado de la implementación del circuito será cero.

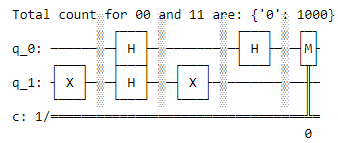


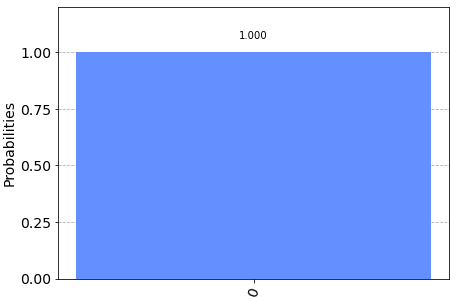


La grafica de este circuito indica que el 100% de las veces el valor será cero confirmando el posible valor que mencionábamos antes para que cumpliera la propiedad de función constante.

### Función todos a uno

Al igual que en la función todos a cero la función Todos a uno muestra que todo el conjunto entradas cuenta con el mismo valor de salida a uno. Para esta función al tener en todos sus elementos de entrada apuntando al valor uno decimos que es una función constante y que el resultado de la implementación del circuito será cero.





En el gráfico observamos que para el 100% de las veces el resultado será cero, indicando que es una función constante.

# Algoritmo de Deutsch-Jozsa

Describa el experimento que va a realizar y lo que presentara en este capítulo. Explique al lector su comprensión del tema.

## Problema

Describa el problema que resuelve el algoritmo. Explique la naturaleza de estas funciones, ¿cuántas son?, ¿como se ven?

## Implementando las funciones en el computador cuántico

Explique la implementación de las funciones en el computador cuántico.

## Implementando el algoritmo de Deutsch-Josza en un computador cuántico

Explique sus gráficas y experimentos

# Conclusiones

Concluya. ¿Qué presentó?, ¿Qué entendió?, ¿Qué aprendió?, ¿Qué proyección le ve a lo aprendido? ¿Qué trabajos futuros propone?

# Bibliografía

¿Qué leyó para construir este reporte?