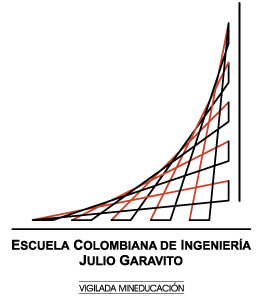
**IMPLEMENTACION DE ALGORITMO DE DEUTSCH Y DECUTSCH-JOZSA**



**IVAN CAMILO RINCON SAAVEDRA**

**IVAN.RINCON-S@MAIL.ESCUELAING.EDU.CO**

**ESCUELA COLOMBIANA DE INGENIERÍA JULIO GARAVITO**

**BOGOTÁ D.C. 08 DE MAYO 2020**

# Tabla de contenidos

[**IMPLEMENTACION DE ALGORITMO DE DEUTSCH Y DECUTSCH-JOZSA** 1](#_Toc39791329)

[Tabla de contenidos 2](#_Toc39791330)

[*1.* *INTRODUCCION* 4](#_Toc39791331)

[*En el presente informe se busca realizar la* Implementación del Algoritmo de Deutsch y Deutsch-Jozsa,para probar su funcionamiento a través de diversos ejercicios propuestos, para cada uno de estos se realizaran diferentes gráficos los cuales se explicaran a lo largo de del informe 4](#_Toc39791332)

[*2. ALGORITMO DE DEUTSCH|* 5](#_Toc39791333)

[*2.1 EXPLICACION* 5](#_Toc39791334)

[*2.2 PROBLEMA* 5](#_Toc39791335)

[2.3 ALGORITMO 5](#_Toc39791336)

[6](#_Toc39791337)

[2.4 IMPLEMENTANDO LAS FUNCIONES EN EL COMPUTADOR CUANTICO 6](#_Toc39791338)

[2.5 PROCEDIMIENTO 6](#_Toc39791339)

[2.5.1 PRIMERA FUNCION 7](#_Toc39791340)

[Circuito que implementa la función 7](#_Toc39791341)

[Prueba: 8](#_Toc39791342)

[2.5.2 Segunda función 12](#_Toc39791343)

[Circuito que implementa la funcion 13](#_Toc39791344)

[Pruebas 13](#_Toc39791345)

[2.5.3 Tercera función 17](#_Toc39791346)

[Circuito que implementa la función 18](#_Toc39791347)

[Pruebas: 18](#_Toc39791348)

[2.5.4 Cuarta Función 21](#_Toc39791349)

[Circuito que implementa la función 22](#_Toc39791350)

[Pruebas 22](#_Toc39791351)

[3. IMPLEMENTANDO EL ALGORITMO DE DEUTSCH EN UN COMPUTADOR CUANTICO 26](#_Toc39791352)

[3.1 PROCEDIMIENTO 26](#_Toc39791353)

[3.1.1 Primera función 27](#_Toc39791354)

[3.1.2 Segunda función 28](#_Toc39791355)

[3.1.3 Tercera función 29](#_Toc39791356)

[3.1.4 Cuarta función 30](#_Toc39791357)

[4. ALGORITMO DE DEUTSCH-JOZSA 31](#_Toc39791358)

[4.1 EXPLICACION 31](#_Toc39791359)

[4.2 Problema 31](#_Toc39791360)

[4.3 ALGORITMO 31](#_Toc39791361)

[5. Implementando las funciones 32](#_Toc39791362)

[**5.1 Primera Función** 32](#_Toc39791363)

[6. Implementando el algoritmo de Deutsch-Josza en un computador cuántico, para la primera función 42](#_Toc39791364)

[6.1 PRIMERA FUNCION 42](#_Toc39791365)

[7. Implementando las funciones 44](#_Toc39791366)

[7.1 SEGUNDA FUNCION 44](#_Toc39791367)

[8. Implementando el algoritmo de Deutsch-Josza en un computador cuántico, para la segunda función 54](#_Toc39791368)

[6.1 SEGUNDA FUNCION 54](#_Toc39791369)

[Bibliografía 57](#_Toc39791370)

# ***INTRODUCCION***

## En el presente informe se busca realizar la ejecuciones del Algoritmo de Deutsch y Deutsch-Jozsa,para probar su funcionamiento a través de diversos ejercicios propuestos, para cada uno de estos se realizaran diferentes gráficos junto a simulaciones en el computador cuántico de IBM los cuales se explicaran a lo largo de este informe, estos algoritmos surgen con el fin de resolver el problema de encontrar o conjeturar una “caja negra” que reciba una función y(Balanceada o constante) sea capaz de decir si esta es balanceada o constante, posteriormente explicaremos como se define el tipo de una función, se debe tener en cuenta que para el desarrollo de estos algoritmos se hacen usos de diversas puertas , como por ejemplo la de hadamard cuya función principal es que los qbits que ingresen entren en lo que se conoce como la superposición cuántica , cabe mencionar que si algunos conceptos no son claros revisar el libro que se encuentra en la bibliografía.

# ***2. ALGORITMO DE DEUTSCH|***

## *2.1 EXPLICACION*

***“***Es un[**algoritmo cuántico**](https://es.wikipedia.org/wiki/Algoritmo_cu%C3%A1ntico)**,** diseñado para ejecutar sobre un computador cuántico**,** en este nos dan una función cuántica (que para nosotros es una caja negra**) *f(x1, x2,)*** que toma *2* bits de entrada *x1, x,* y devuelve un valor binario ***f(x1, x2)*.”**

## *2.2 PROBLEMA*

El problema surge al querer encontrar una especie de "caja negra" que implemente una función f, tal que **f:{0,1} -> {0,1}** que sea capaz de decir si dicha función es **Balanceada o Constante**

**donde:**

* Balanceada si f(1) != f(0)
* Constante si f(1)=f(0)

## 2.3 ALGORITMO

La versión que se muestra a continuación es el algoritmo cuyo dominio será una sola entrada, donde el bloque que se logra apreciar como **H**, es conocida como la compuerta de **HADAMARD**.

## Algoritmo de Deutsch-Jozsa - Wikiwand

## 2.4 IMPLEMENTANDO LAS FUNCIONES EN EL COMPUTADOR CUANTICO

*En esta etapa* implementaremos las 4 funciones posibles de {0,1} a {0,1} usando el computador cuántico de IBM, pero antes de eso realizaremos los siguientes 4 criterios, para que sea más sencillo.

* **Dibujo de función**
* **Matriz correspondiente**
* **Circuito correspondiente**
* **Resultados de las 4 pruebas**

## 2.5 PROCEDIMIENTO

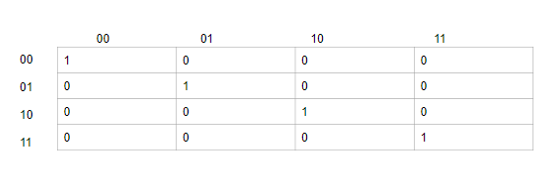
Para cada una de las funciones, es importante realizar su respectivo grafico ya que nos permite abstraer mejor la información presentada y a su vez tener un mejor panorama sobre los mapeos que dará cada una de la diferentes funciones , las 4 funciones posibles de {0,1} a {0,1} que se analizaran en esta sección se nombran a continuación .

### 2.5.1 PRIMERA FUNCION

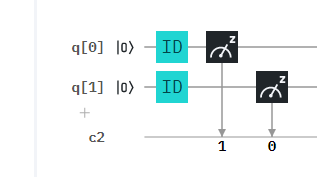
#### Dibujo de función

#### Matriz que implementa la función

Ahora procederemos a crear la matriz correspondiente a la función, para esto debemos tener en cuenta que las columnas **00, 01, 10, 11** representarán las entradas del circuito y las filas serán el resultado de esta entrada, es decir que si la columna es la **10 la** salida será **10,** es justamente en la fila 10, columna 10 , se deberá colocar un 1.



### Circuito que implementa la función

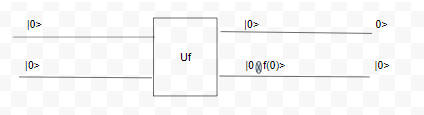
****Una vez que se plantea una matriz que represente la función dada, se debe encontrar un patrón que nos permita crear un circuito en la maquina cuántica que nos presente IBM, como puede observar en la matriz anterior no importa cual sea la entrada, esta será exactamente igual a la salida, esto se debe ya que esta función representa la **“identidad”**.

Cabe recalcar que el circuito se modelo de forma intuitiva, de tal forma que cumpliera y concordara con los mapeos de la función anterior. Lo que haremos ahora es mostrar que el circuito en efecto modela la función propuesta, probando todas las posibles entradas las cuales serán:

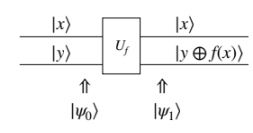
* q[0] = | 0>, q[1] = | 0>
* q[0] = | 0>, q[1] = 1
* q[0] = | 1>, q[1] = | 0>
* q[0] = | 1>, q[1] = | 1>

### Prueba:

##### Entrada| 0>,| 0>



Note que a nuestra función ingresa **| 0> | 0>,** que deforma general se puede ver como:

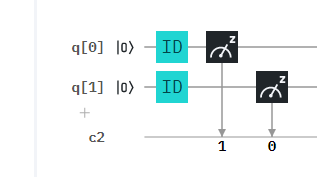


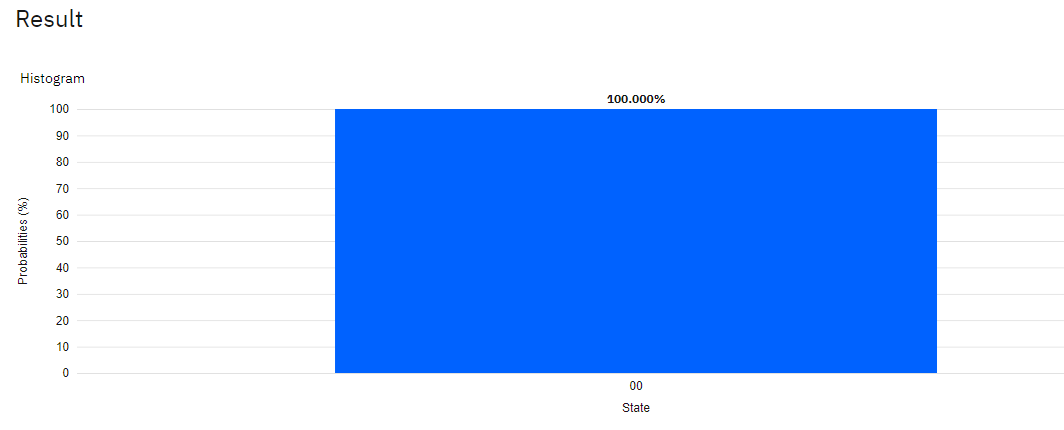
por el grafico anterior podemos apreciar que en el alambre superior no importa que entre siempre su salida será su entrada por lo que en este caso será **|0>**, en el alambre inferior cambia un poco ya que va a ser la entrada del segundo alambre XOR **(O exclusivo)** el mapeo de la función con parámetro del valor del primer alambre.

**Ejemplo:**

* En el primer alambre ingresa |0> y su salida será |0> que fue lo que entro.
* En el segundo alambre ingresa **|0>** y para su salida será |0> ya que **|0>** **XOR f(0) = |0>**
* Por lo que la salida será **|00>**

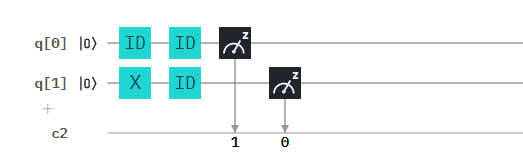
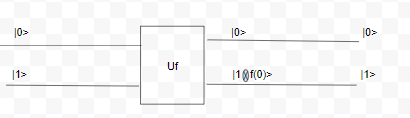
Ya hicimos el calculo de forma teórica ahora lo comprobaremos, con el computador cuántico de IBM; ejecutando el circuito antes visto con las entradas |0>, |0> **(ya que estamos mirando específicamente estas entradas)**

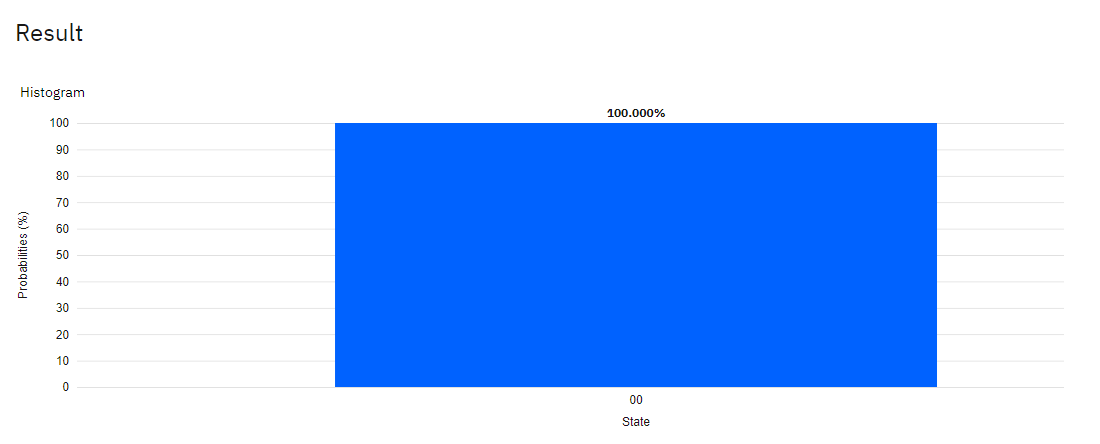




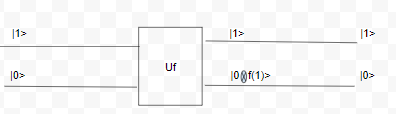
Note que el resultado fue que el 100% de las veces la salida será **|00>,** por lo que desde el punto de vista teórico y práctico está modelando la función dada. **Ahora falta comprobar que las otras 3 entradas también modelan la función, para esto de forma análoga realice el procedimiento ya explicado**.

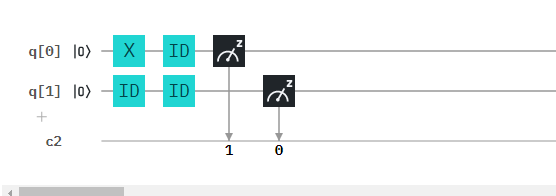
#### Entrada |0>,|1>

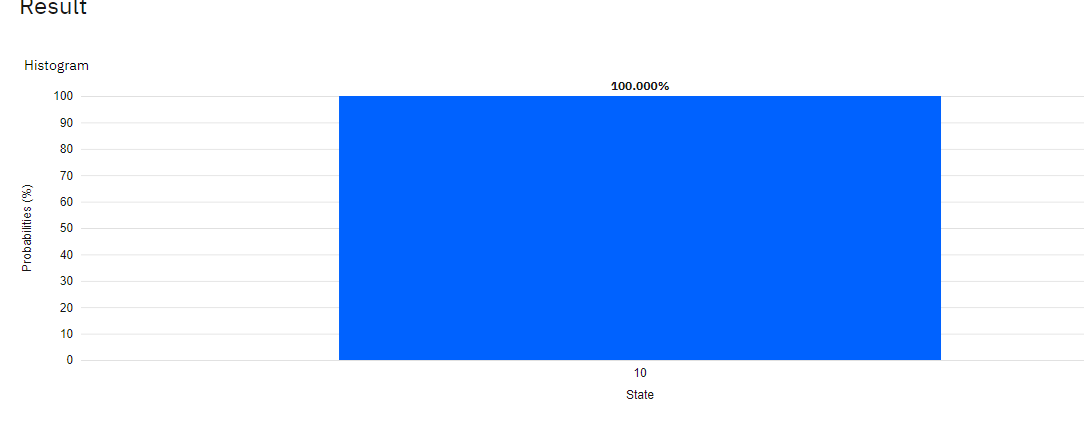




#### Entrada |1>,|0>

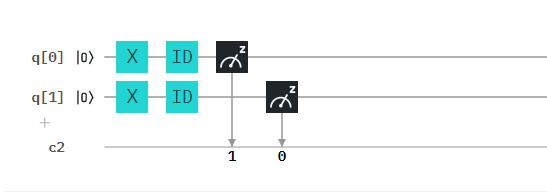


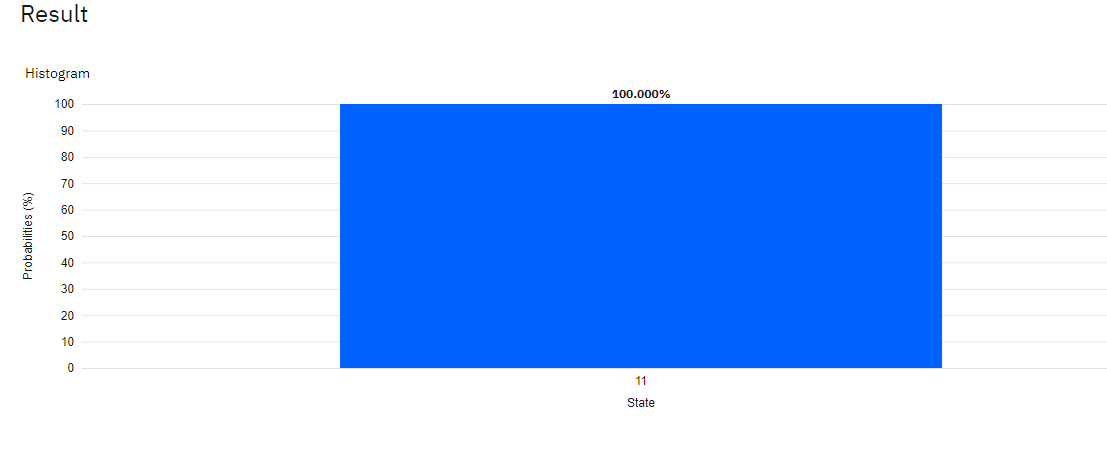




#### Entrada |1>,|1>



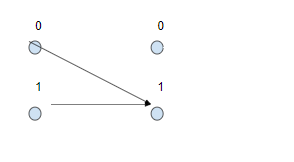




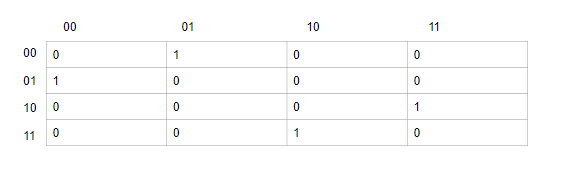
### 2.5.2 Segunda función

El procedimiento que se destaco para la primera función se debe realizar de forma análoga para esta.

#### Dibujo de la función

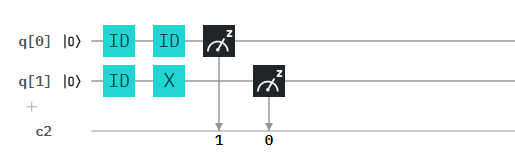


#### Matriz correspondiente



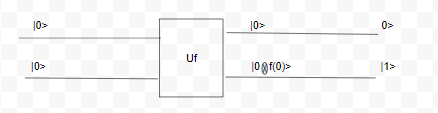
### Circuito que implementa la funcion

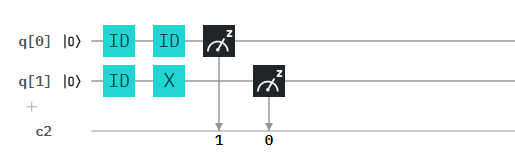
Note que en este caso la entrada del primer alambre siempre será la misma, pero la del segundo siempre será su inversa, esta afirmación se puede modelar como:

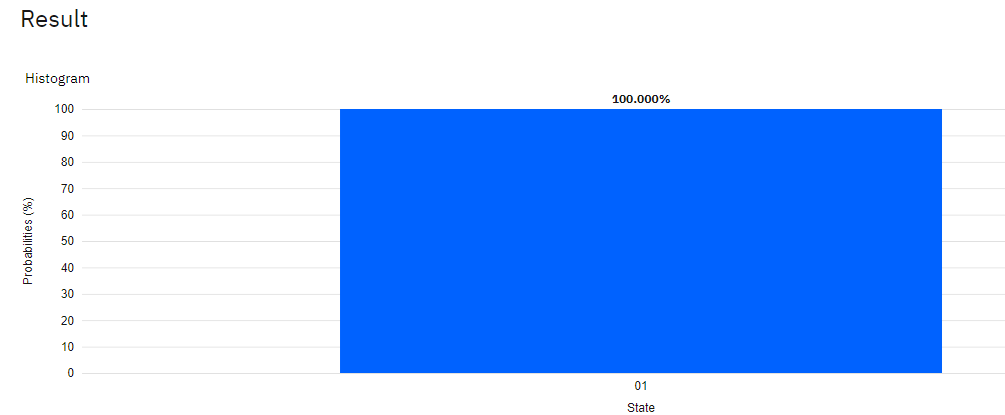


### Pruebas

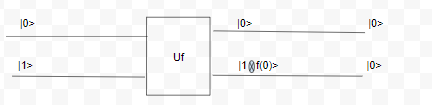
#### Entrada |0>,|0>



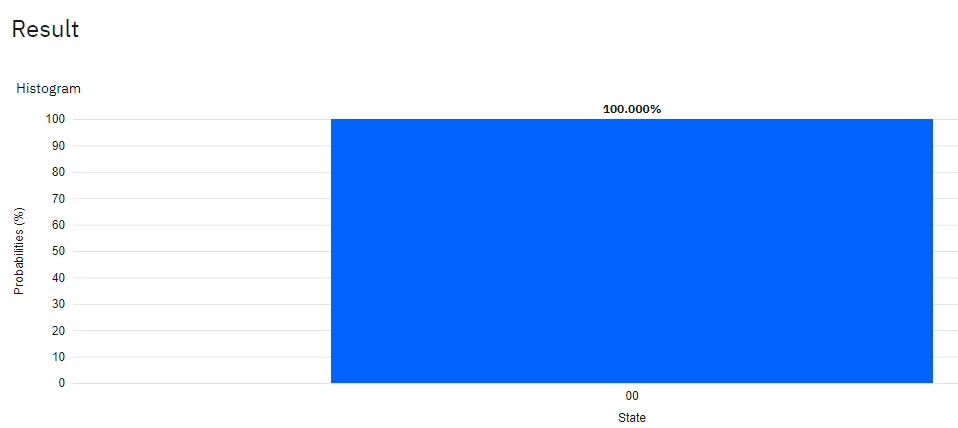




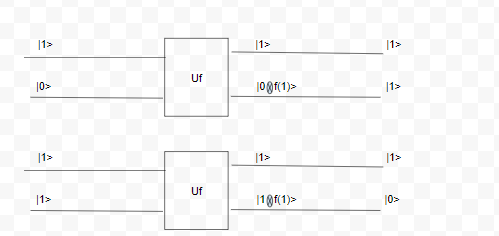
#### Entrada |0>,|1>

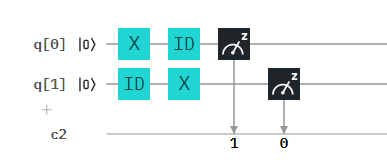


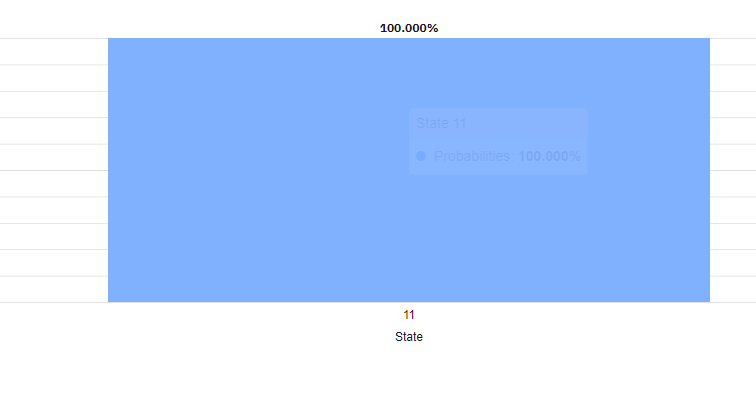




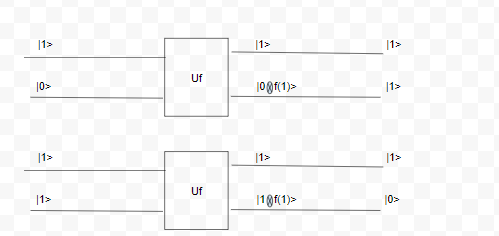
#### Entrada |1>,|0>

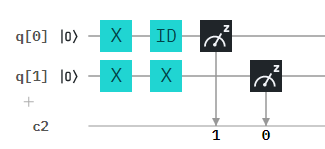


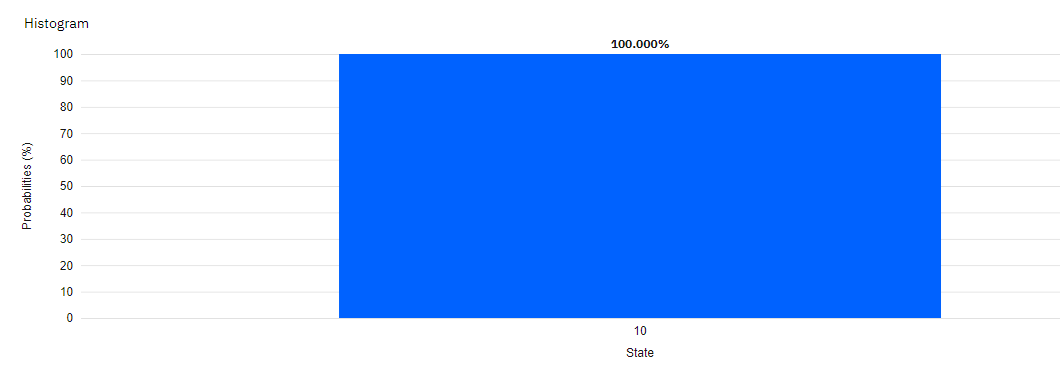




#### Entrada |1>,|1>



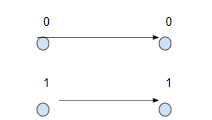




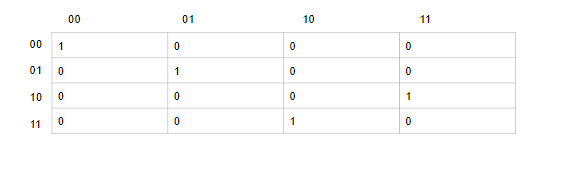
### 2.5.3 Tercera función

El procedimiento que se destacó para la primera función se debe realizar de forma análoga para esta.

#### Dibujo de función

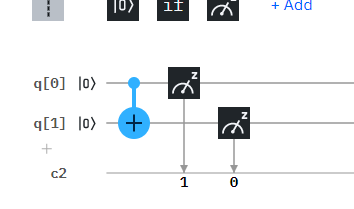


#### Matriz



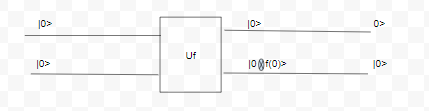
### Circuito que implementa la función

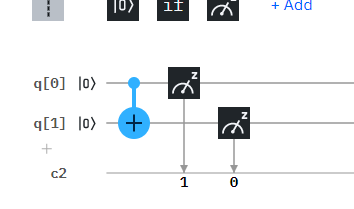
En este caso el patrón que podemos observar es que el segundo alambre solo cambia si el primero es un uno, este patrón es equivalente al que se presenta con la compuerta CNOT controlado.

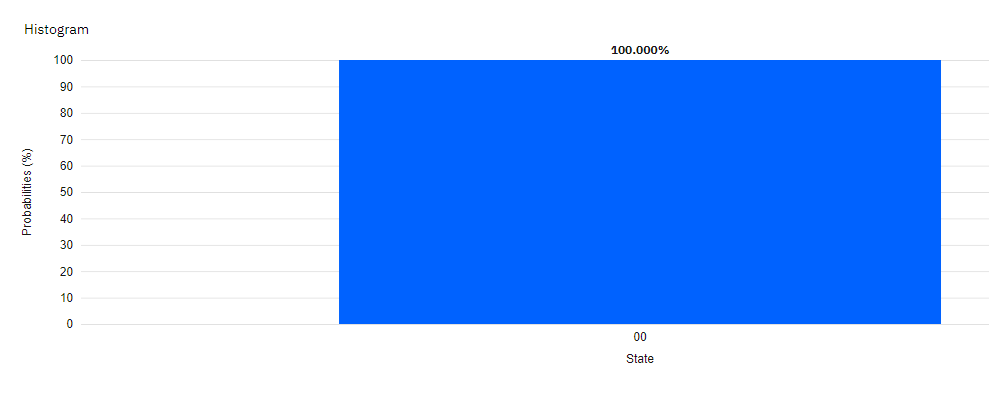


### Pruebas:

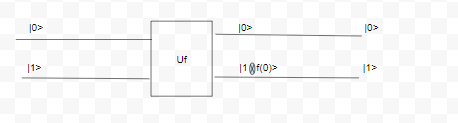
#### Entrada |0>,|0>

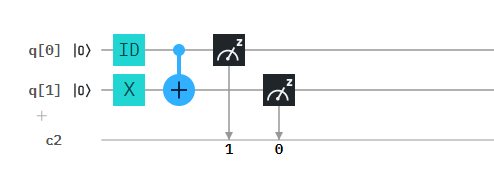


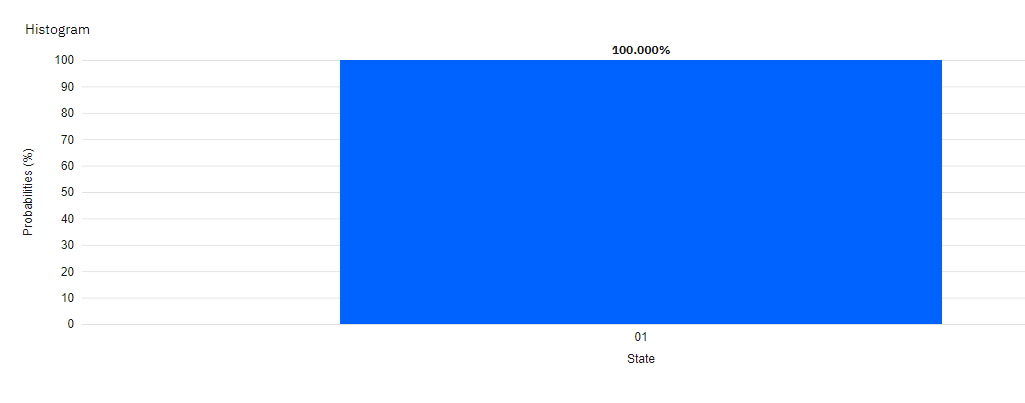




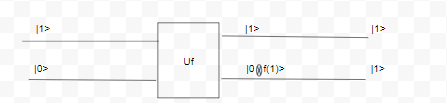
#### Entrada |0>,|1>

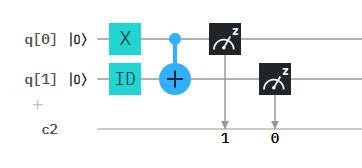


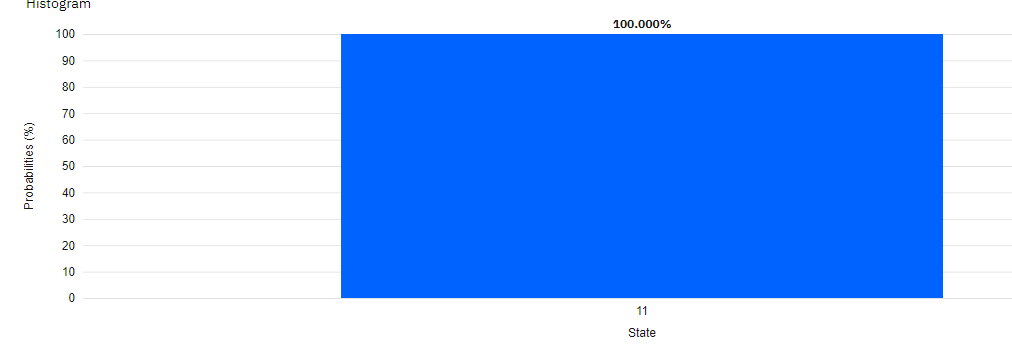




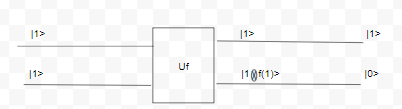
#### Entrada |1>,|0>

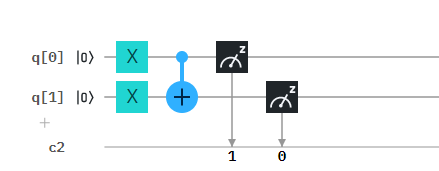


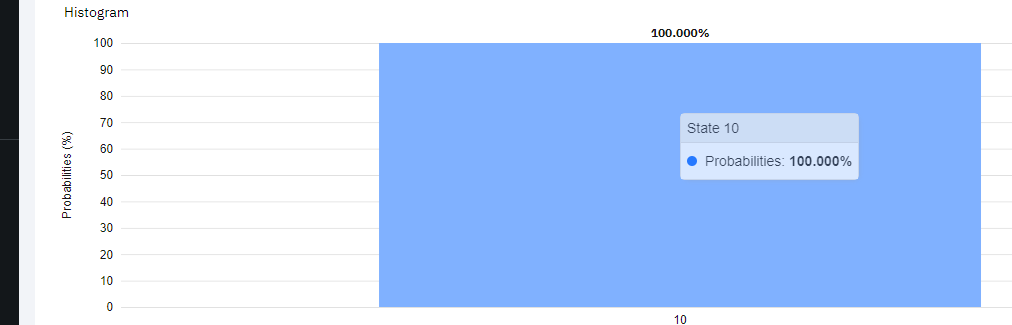




#### Entrada |1>,|1>

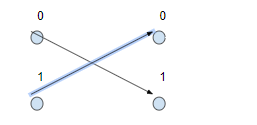




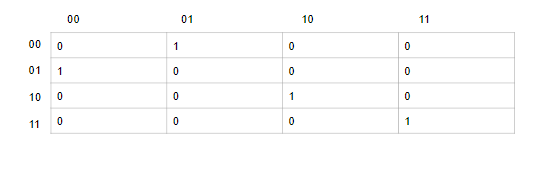


### 2.5.4 Cuarta Función

#### Dibujo de función

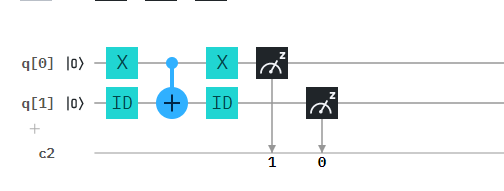


#### Matriz



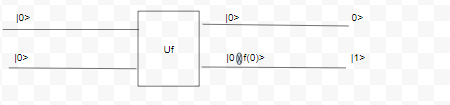
### Circuito que implementa la función

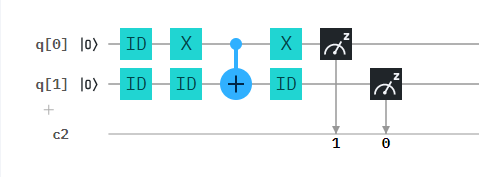
En esta función al igual que la anterior en ciertos casos el primer alambre se mantiene igual y en el segundo es la negación de la segunda entrada, pero note que en este caso solo se niega el segundo alambre si en el primero le ingreso un cero, por lo que el circuito se puede modelar como un CNOT controlado , pero negando su bit controlador antes y después de su lectura.

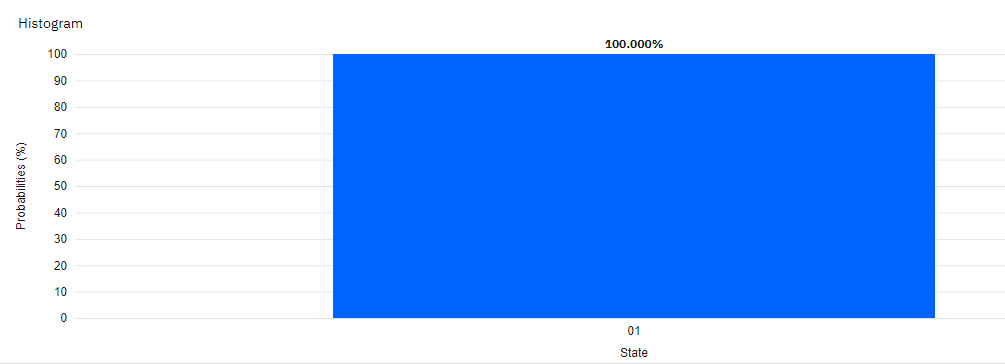


### Pruebas

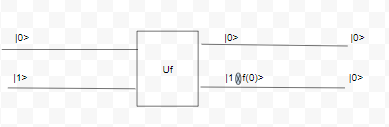
#### Entrada |0>,|0>

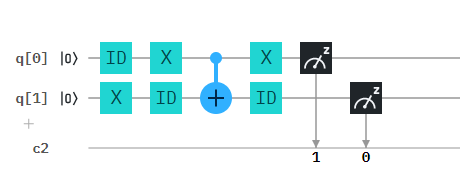


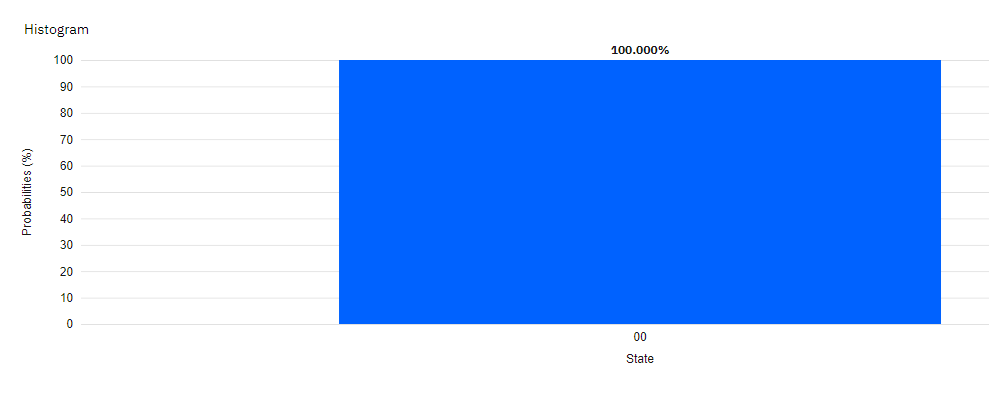




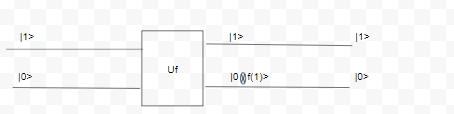
#### Entrada |0>,|1>

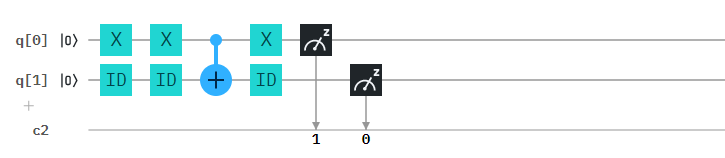


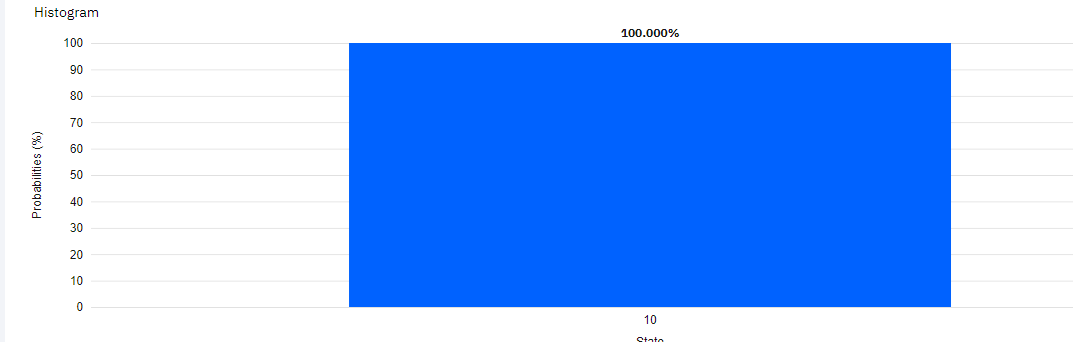




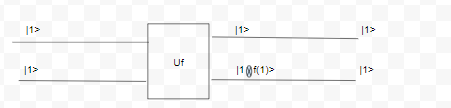
#### Entrada |1>,|0>

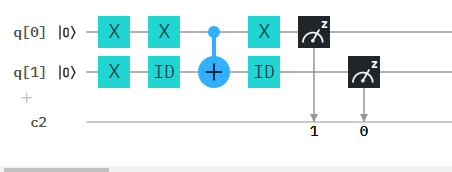


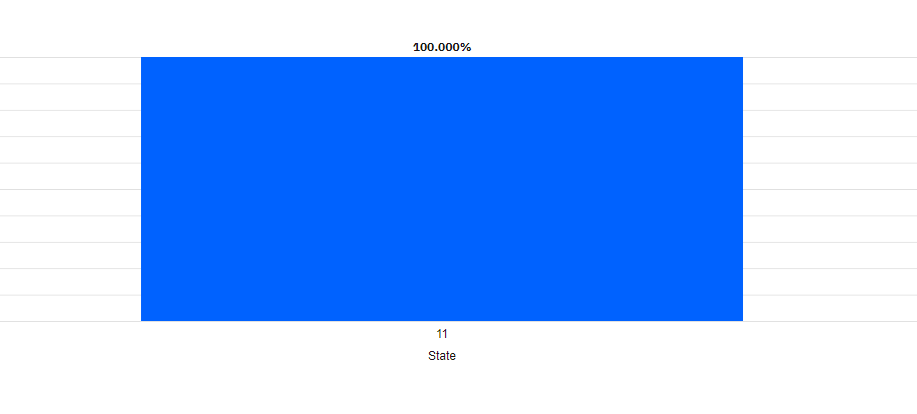




#### Entrada |1>,|1>







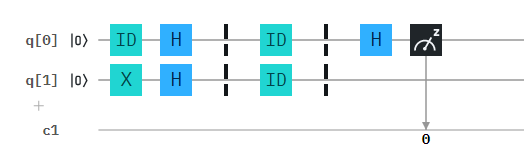
# **3. IMPLEMENTANDO EL ALGORITMO DE DEUTSCH EN UN COMPUTADOR CUANTICO**

## 3.1 PROCEDIMIENTO

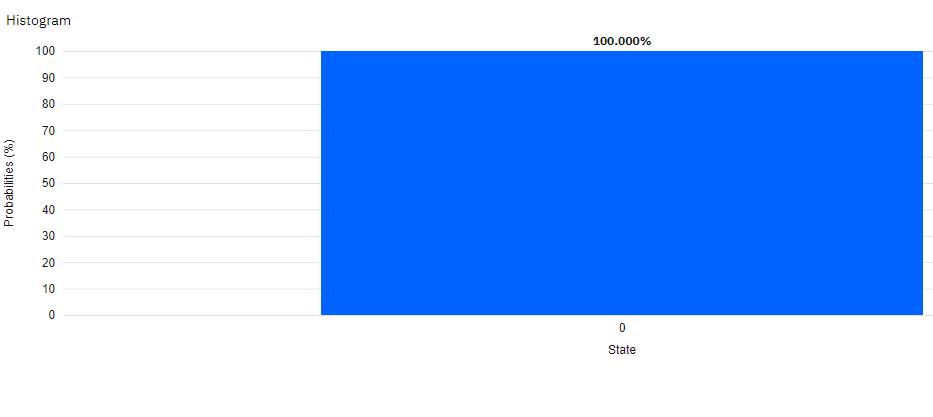
En este punto lo que haremos será verificar que el algoritmo de Deutsch funciona, comprobando si las funciones establecidas anteriormente son balanceadas o son constantes.

### 3.1.1 Primera función

#### Circuito en el algoritmo

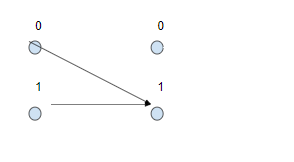


#### Resultados

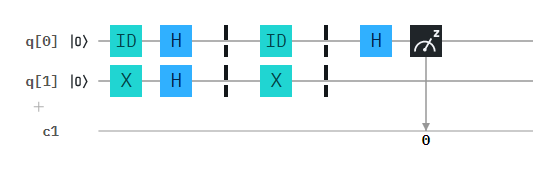


Noteque el 100% de las veces el resultado es **|0>,** por lo que podemos interpretarlo como que el algoritmo nos está diciendo que la función que le ingreso era una función **CTE,** revisando la función anterior podemos apreciar que f (0) = f (1), la cual es la definición formal de una función **CTE.**

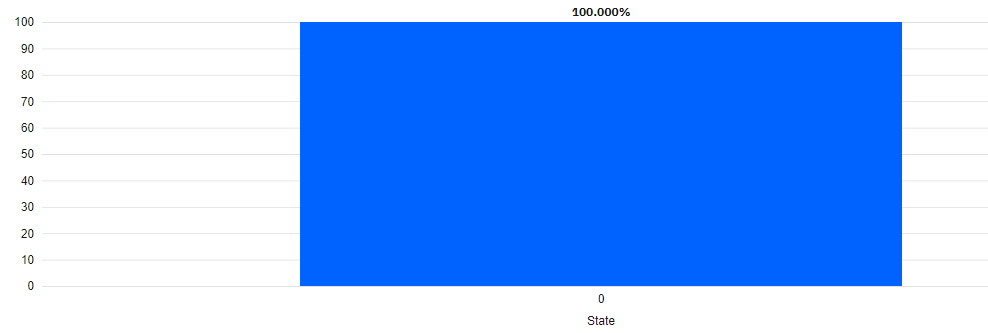
### 3.1.2 Segunda función



#### Circuito en el algoritmo



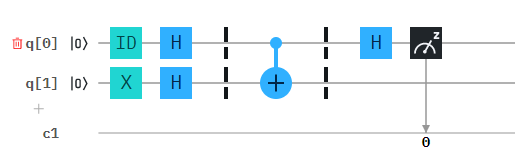
#### Resultados



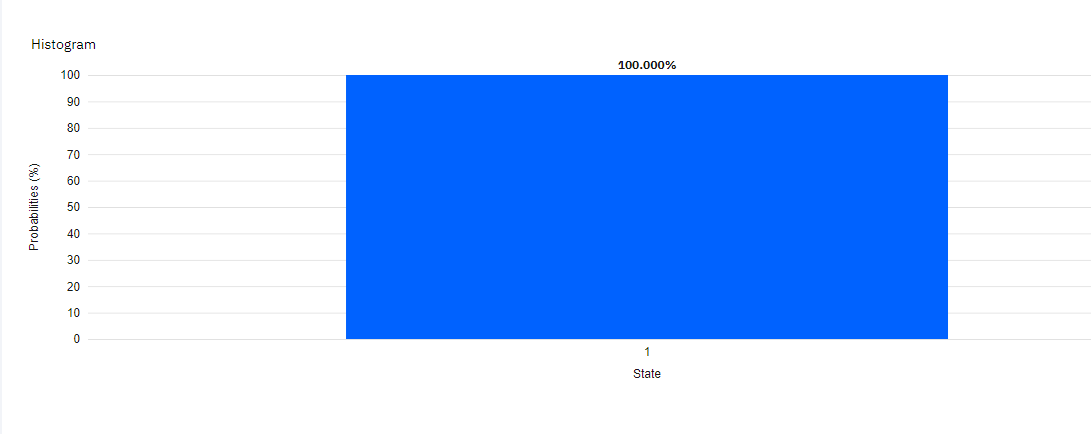
Noteque el 100% de las veces el resultado es **|0>,** por lo que podemos interpretarlo como que el algoritmo nos está diciendo que la función que le ingreso era una función **CTE,** revisando la función anterior podemos apreciar que f (0) = f (1), la cual es la definición formal de una función **CTE.**

### 3.1.3 Tercera función

#### Circuito en el algoritmo



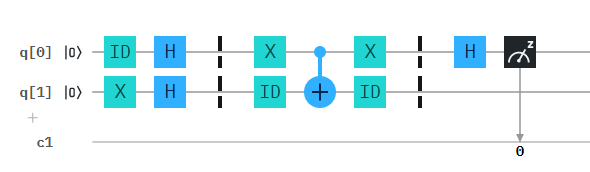
#### Resultados



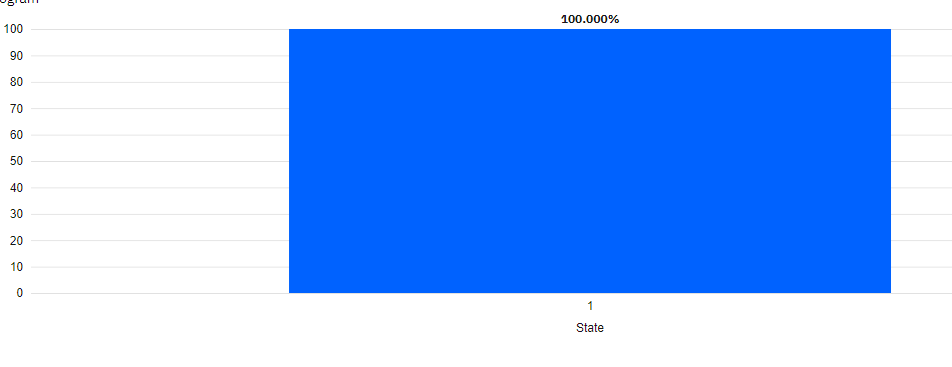
Noteque el 100% de las veces el resultado es **|1>,** por lo que podemos interpretarlo como que el algoritmo nos está diciendo que la función que le ingreso era una función **BALANCEADA,** revisando la función anterior podemos apreciar que f (0) != f (1), la cual es la definición formal de una función **BALANCEADA.**

### 3.1.4 Cuarta función

#### Circuito en el algoritmo



#### Resultados



Noteque el 100% de las veces el resultado es **|1>,** por lo que podemos interpretarlo como que el algoritmo nos está diciendo que la función que le ingreso era una función **BALANCEADA,** revisando la función anterior podemos apreciar que f (0) != f (1), la cual es la definición formal de una función **BALANCEADA.**

# **4. ALGORITMO DE DEUTSCH-JOZSA**

## 4.1 EXPLICACION

***“***Es un[**algoritmo cuántico**](https://es.wikipedia.org/wiki/Algoritmo_cu%C3%A1ntico)**,** propuesto por[**David Deutsch**](https://es.wikipedia.org/wiki/David_Deutsch)**y**[**Richard Jozsa**](https://es.wikipedia.org/w/index.php?title=Richard_Jozsa&action=edit&redlink=1)en 1992. Fue uno de los primeros algoritmos diseñados para ejecutar sobre un computador cuántico**,** este nos dan una función cuántica (que para nosotros es una caja negra**) *f(x1, x2,..., xn)*** que toma *n* bits de entrada *x1, x2,..., xn* y devuelve un valor binario ***f(x1, x2,..., xn)*.”**

## 4.2 Problema

El problema surge al querer encontrar una especie de "caja negra" que implemente una función f, tal que **f:{0,1} -> {0,1}** que sea capaz de calcular cualquier pregunta en el universo cuya respuesta sea **si/no**, pero específicamente este algoritmo busca decir si dicha función es **Balanceada o Constante**

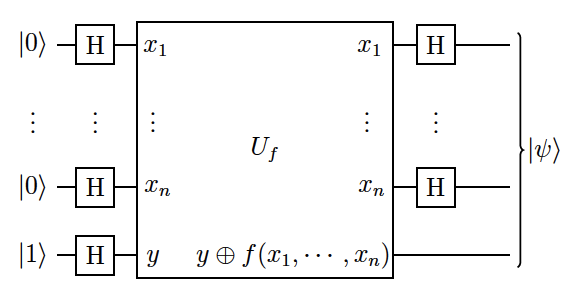
**donde:**

* ***Constante si la salida es 0^n***
* ***Balanceada en cualquier otro caso***

## 4.3 ALGORITMO

Esta es la versión general del algoritmo para funciones *f(x)* de *n* entradas.

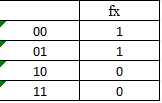
En este caso, la entrada al circuito es:



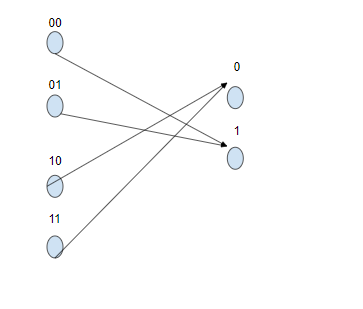
# **5. Implementando las funciones**

En este punto Implementaremos 2 funciones con n= 2 para probar el funcionamiento del algoritmo Deustch-Jozsa.

### **5.1 Primera Función**



#### Gráfico de la función

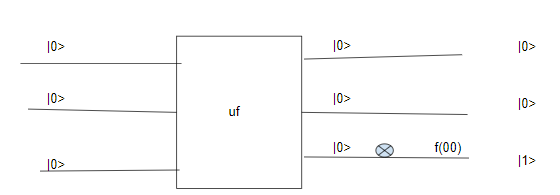


A continuación, debemos completar la matriz, según la función dada.

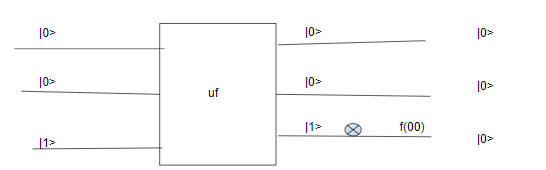
El procedimiento es similar al de **Deutch-Jozca**, únicamente se deben añadir más alambres y todos a excepción del ultimo su entrada será su salida**.**

#### PROCEDIMIENTO

**F (00) = |1>**

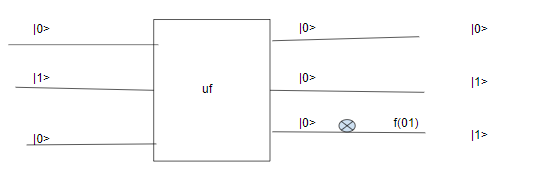


**Para |001> .**

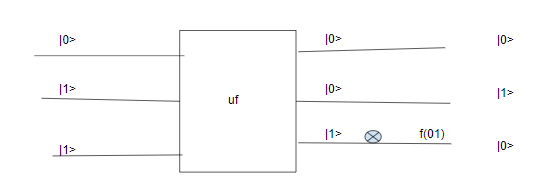


**Para |010> .**

**F(01)=1**

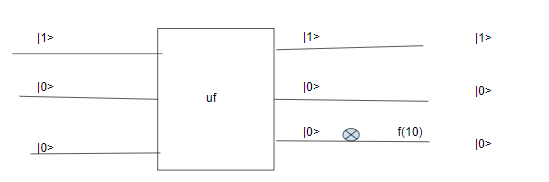


**Para |011> .**

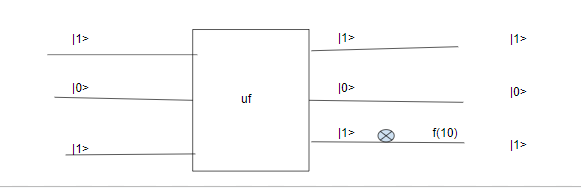


**Para |100> .**

**F(10)=0**

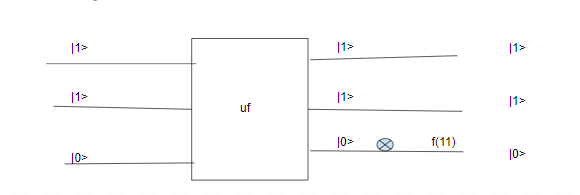


**Para |101> .**

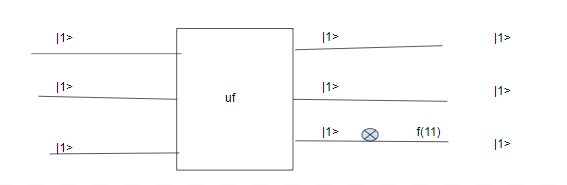


**Para |110> .**

**F(11)=0**

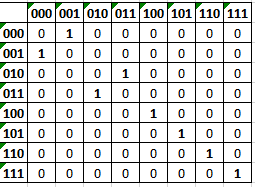


**Para |111> .**

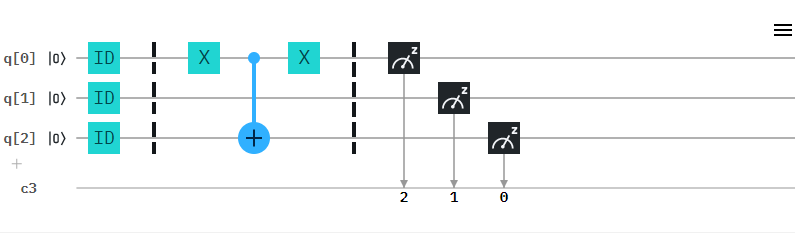


Luego de realizar los anteriores cálculos podemos calcular la matriz, colocando los resultados según corresponda, la matriz correspondiente será:

#### MATRIZ CORRESPONDIENTE



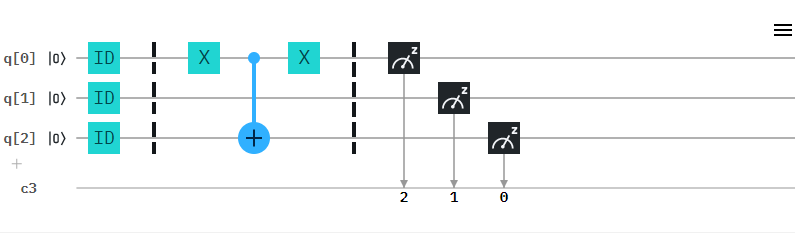
#### el circuito asociado a la función es

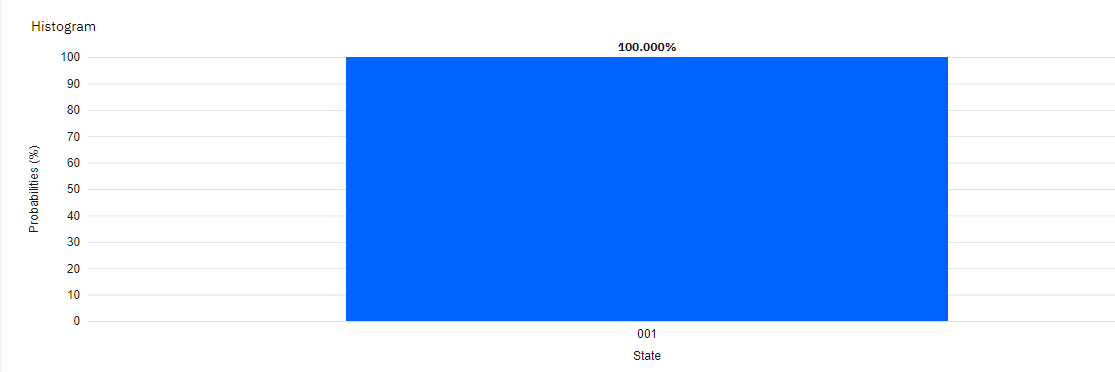


#### **VERIFICACION DEL CIRCUITO**

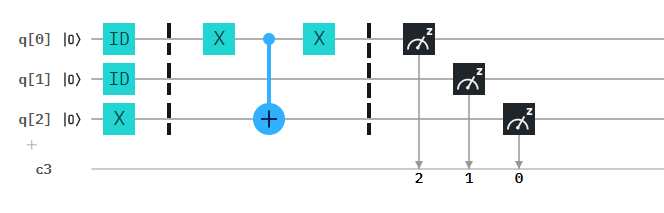
**3) verificaremos que el circuito cumple la función correspondiente**

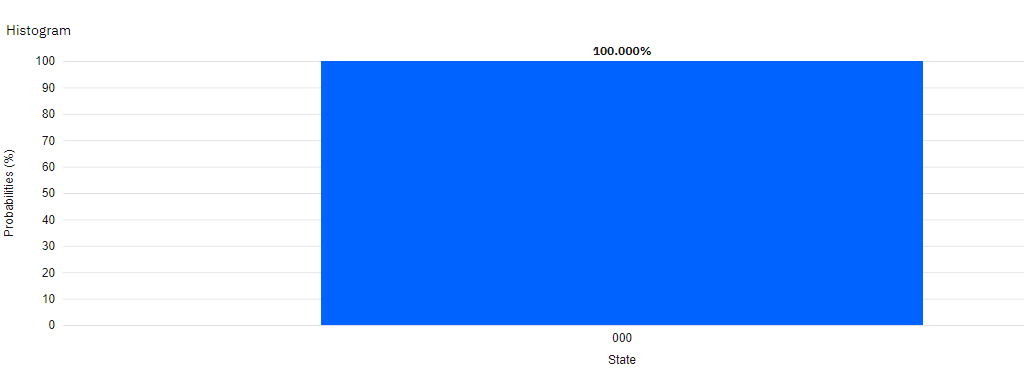
**para |000>**



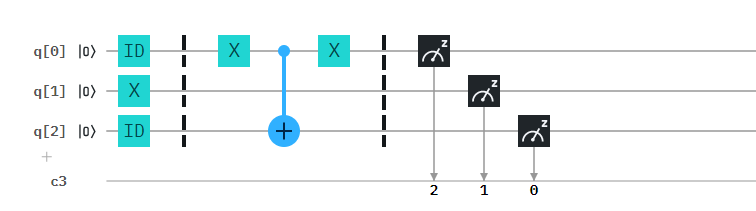


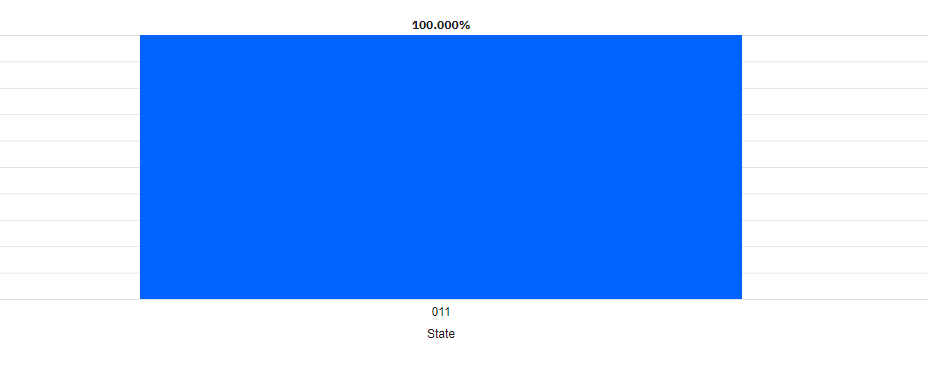
**para |001>**



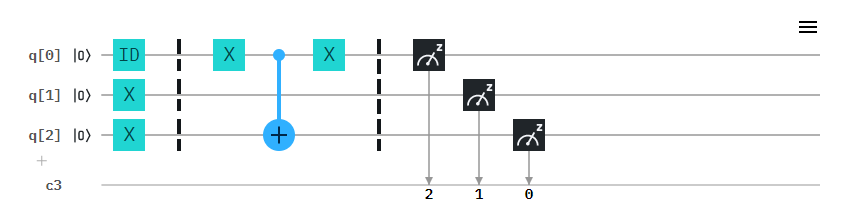


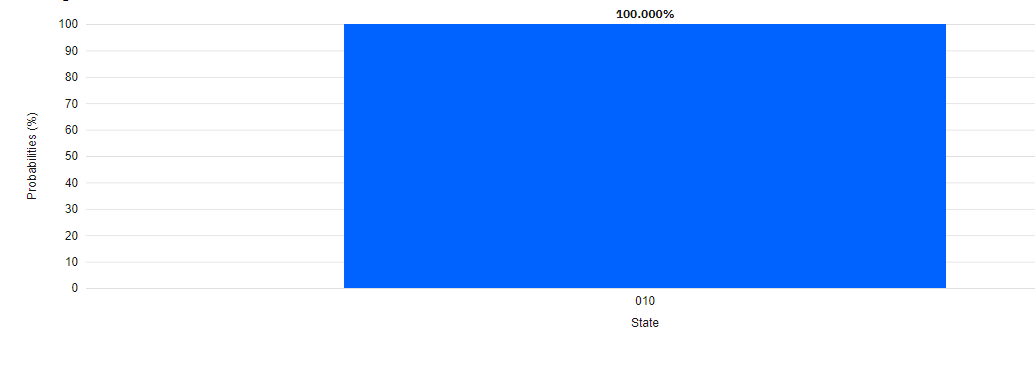
**para |010>**



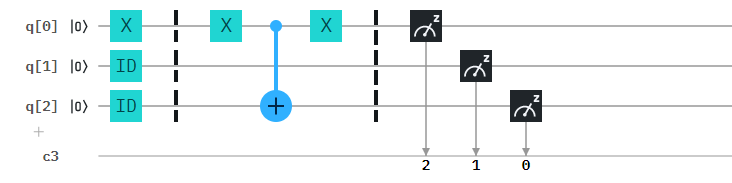


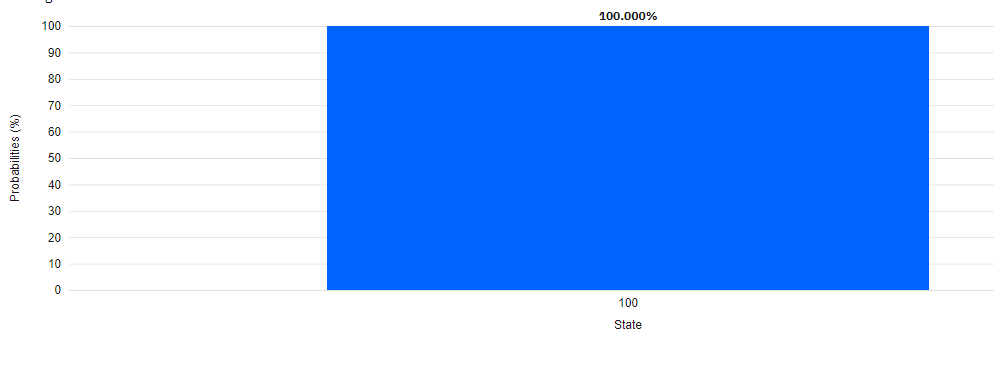
**para |011>**



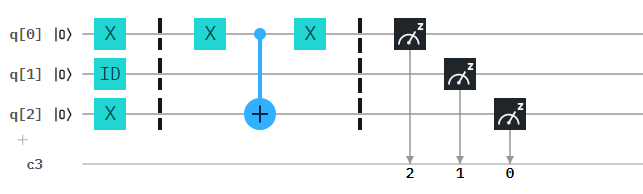


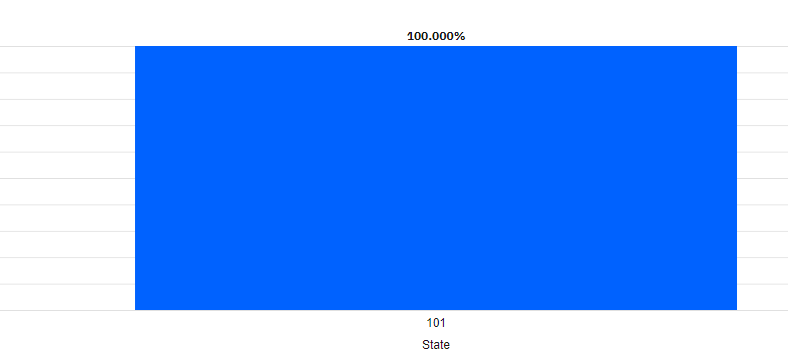
**para |100>**



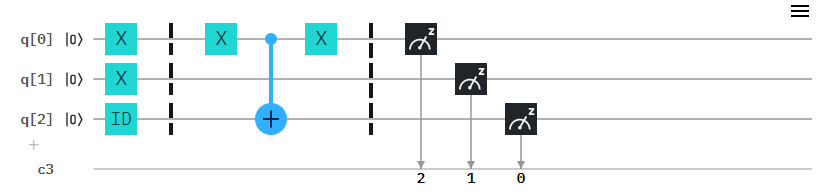


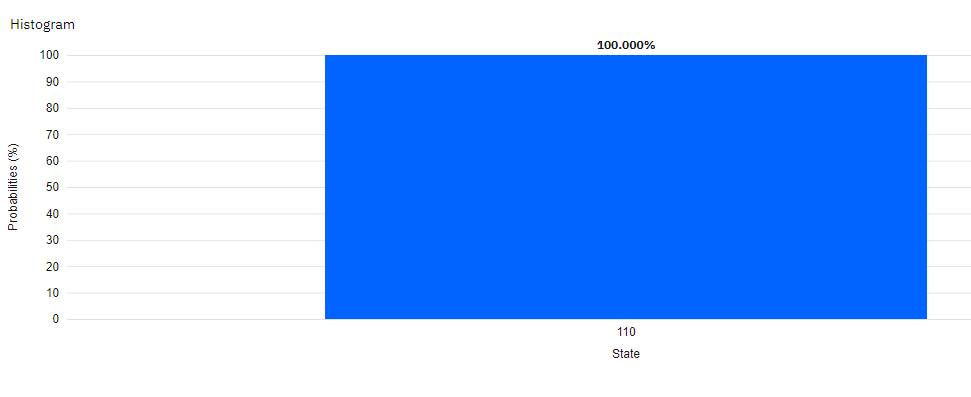
**para |101>**



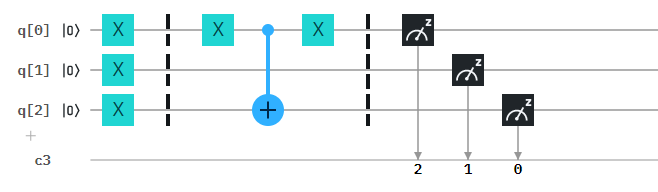


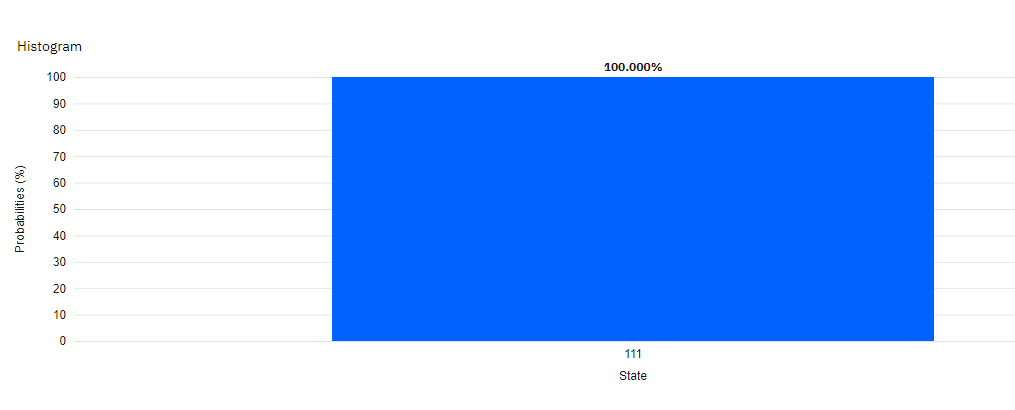
**para |110>**





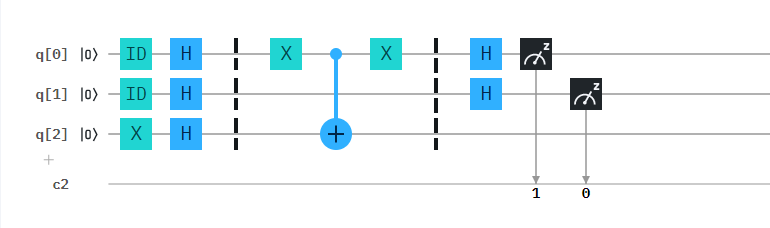
**para |111>**



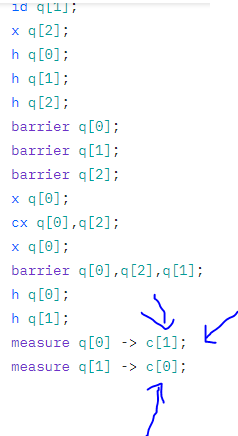


# **6. Implementando el algoritmo de Deutsch-Josza en un computador cuántico, para la primera función**

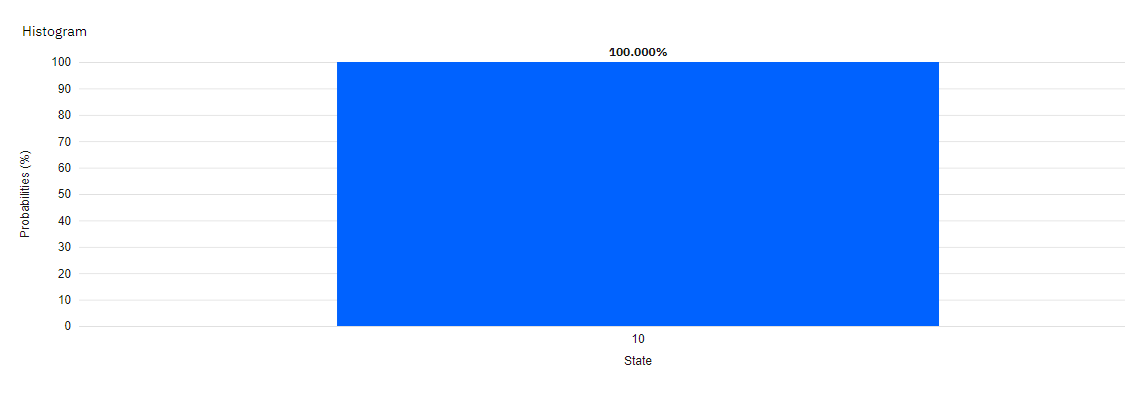
## 6.1 PRIMERA FUNCION



Para la lectura de los datos primero se debe cambiar el mapeo como se muestra en la siguiente imagen



Ya que IBM siempre pone ese mapeo de forma inversa, de no hacerlo puede que el circuito sea correcto pero al mostrar la respuesta la muestre invertida, una vez realizado este cambio corremos el algoritmo, el cual nos dará el siguiente resultado **(ya que solo miramos los dos primeros alambres )**:

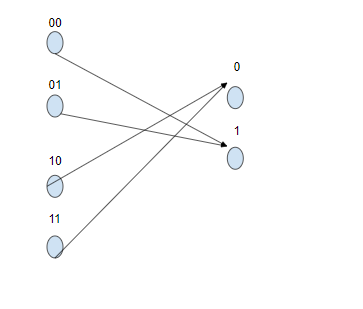


Se debe recordar que si el resultado:

* da **00** se dice que la función es constante
* en cualquier otro caso se dice que es balanceada

cómo podemos observar 10 es diferente a 00, luego f es una función balanceada.

Este lo podemos comprobar ya que si revisamos

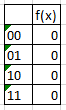


Podemos observar que el 50% de las veces se hace un mapeo a 0 y el otro 50% se hace un mapeo a 1.

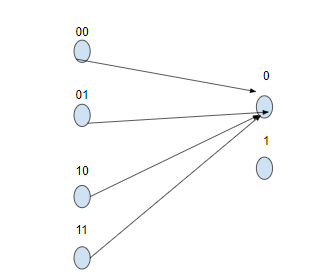
# **7. Implementando las funciones**

## 7.1 SEGUNDA FUNCION

#### función



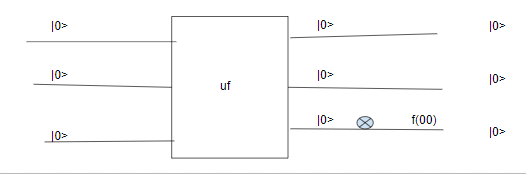
#### Gráfico de la función:



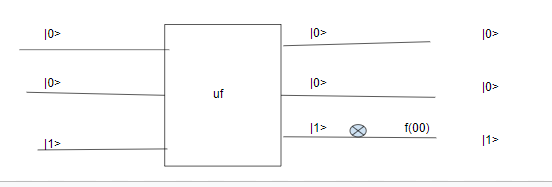
A continuación, debemos completar la matriz, según la función dada.

Mostraremos el procedimiento para **|000> .**

**F(00) = |0>**

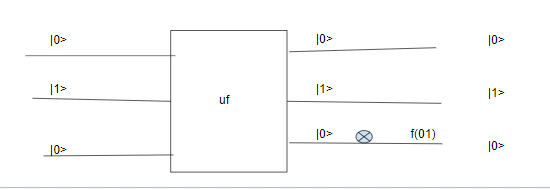


**Para |001> .**

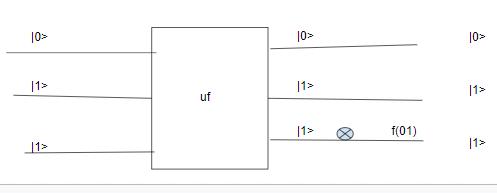


**Para |010> .**

**F(01)=0**



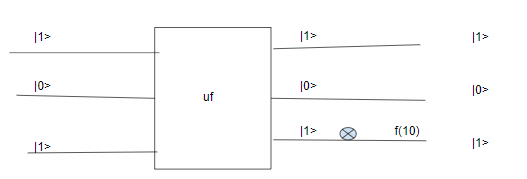
**Para |011> .**



**Para |100> .**

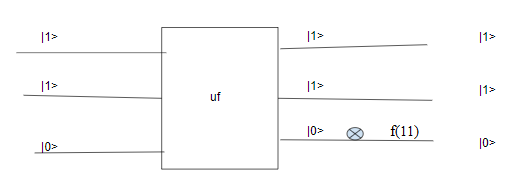
**F(10)=0**

**Para |101> .**

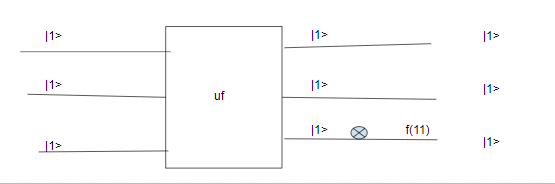


**Para |110> .**

**F(11)=0**

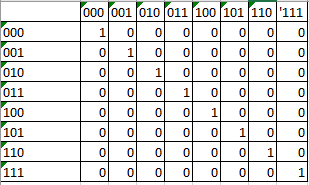
****

**Para |111> .**

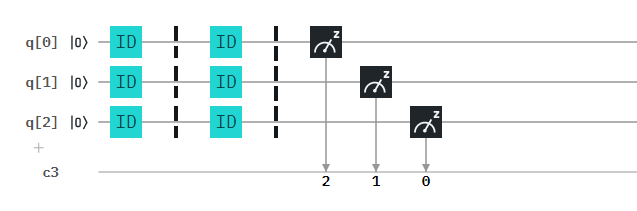


Luego de realizar los anteriores cálculos podemos calcular la matriz, colocando los resultados según corresponda, la matriz correspondiente será:

#### **MATRIZ**



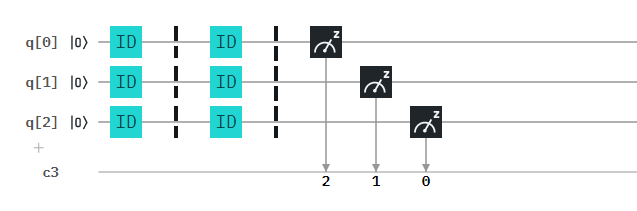
#### **CIRCUITO QUE REPRESENTEA A LA FUNCION**

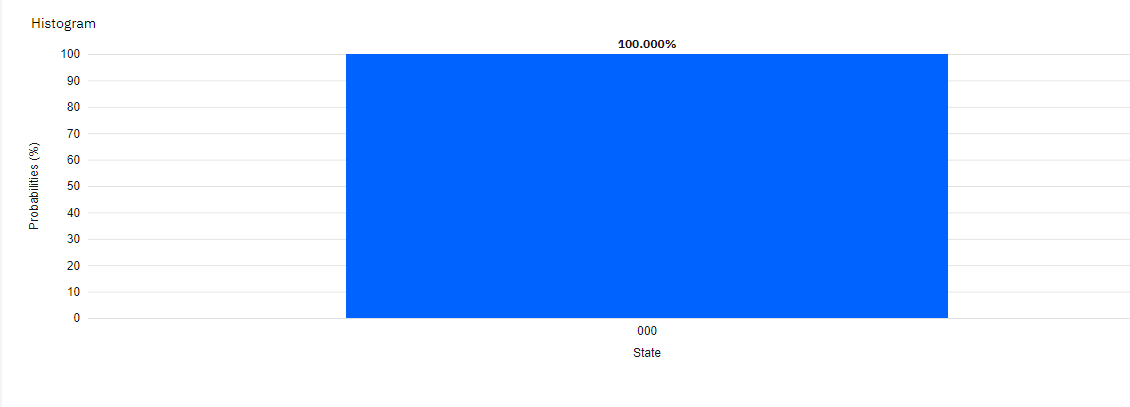


#### **PRUEBAS**

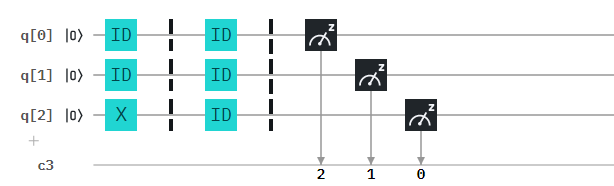
**3) verificaremos que el circuito cumple la función correspondiente**

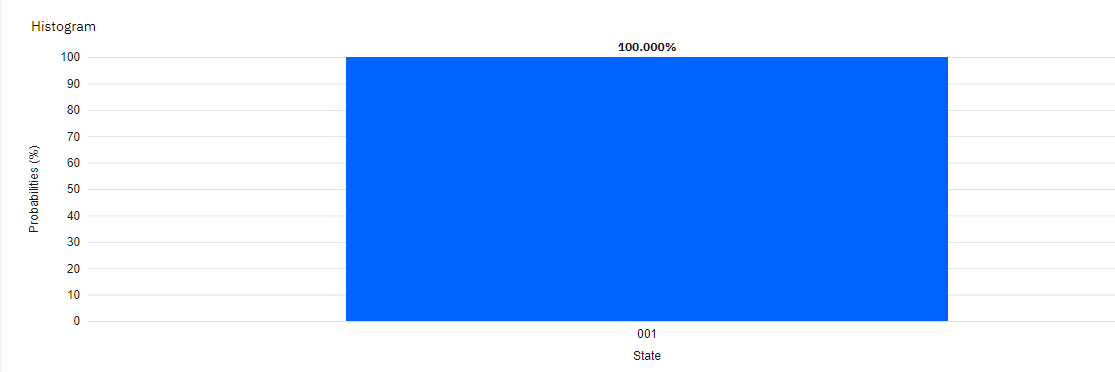
**para |000>**



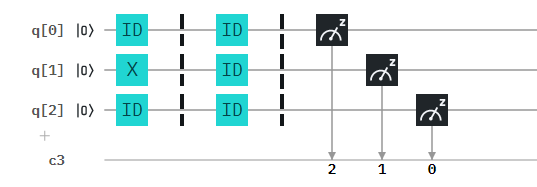


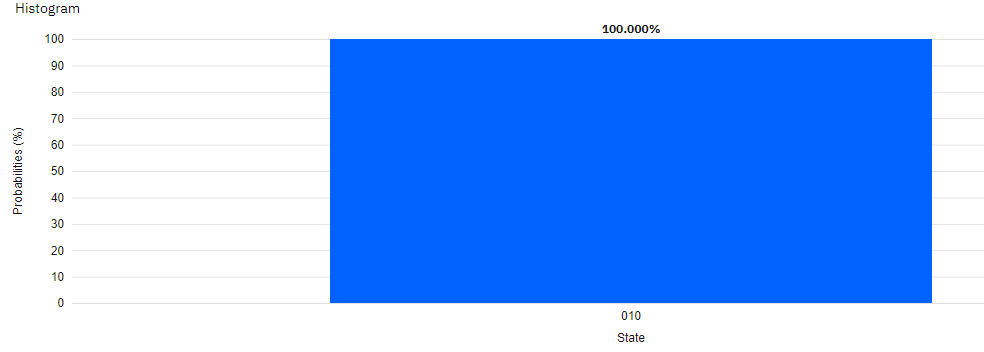
**para |001>**



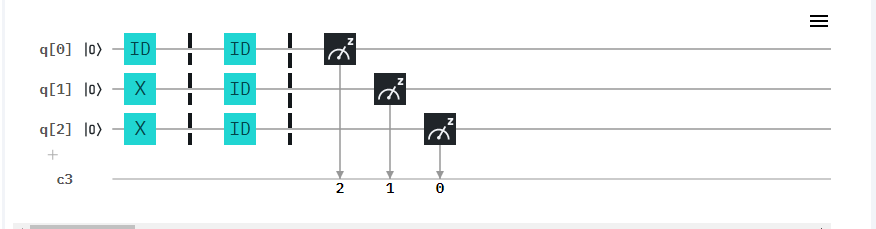


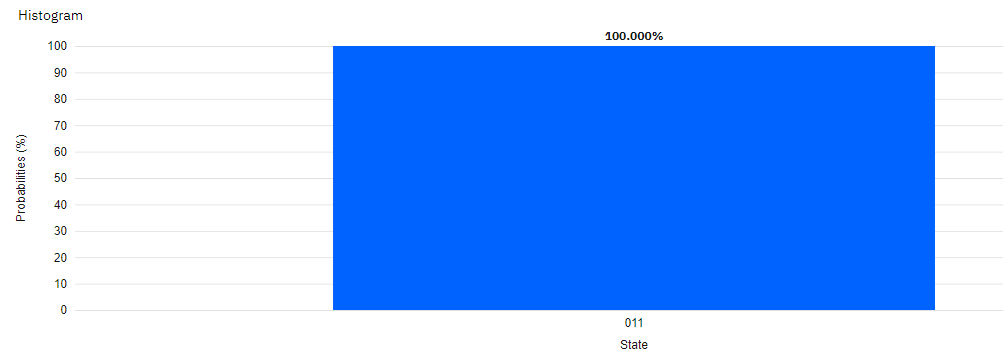
**para |010>**



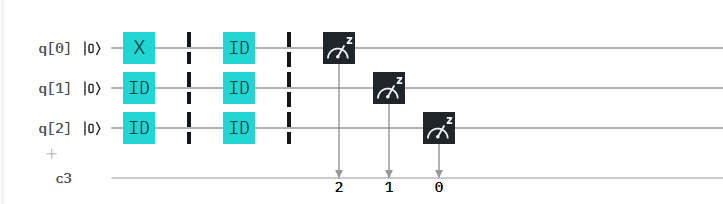


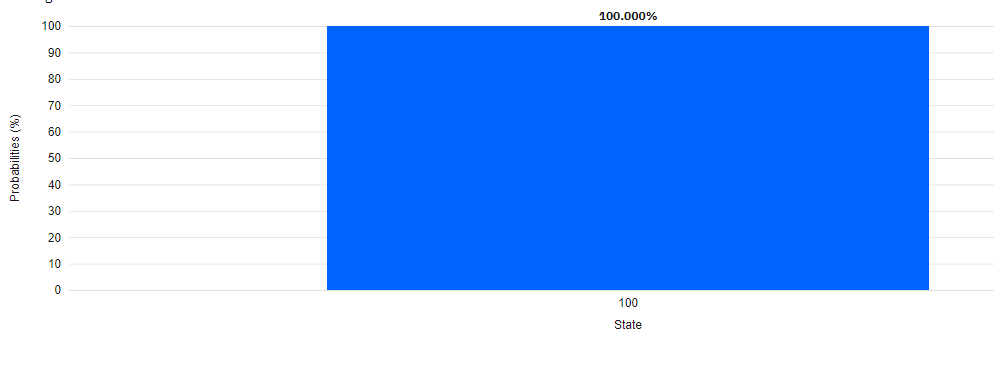
**para |011>**



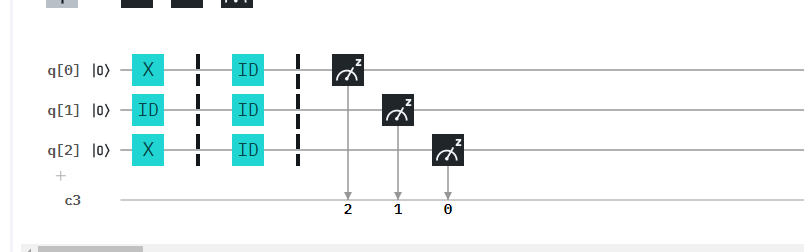


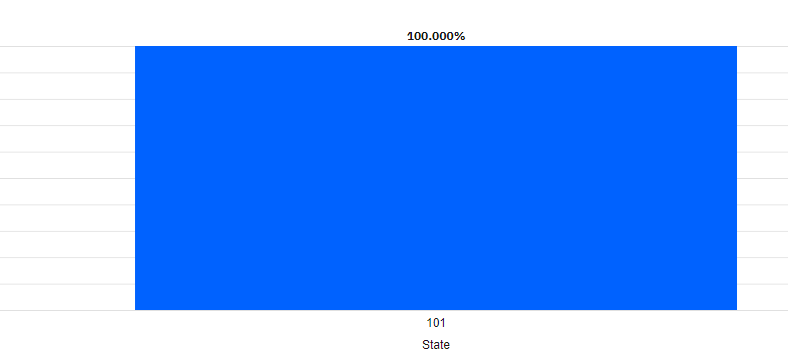
**para |100>**



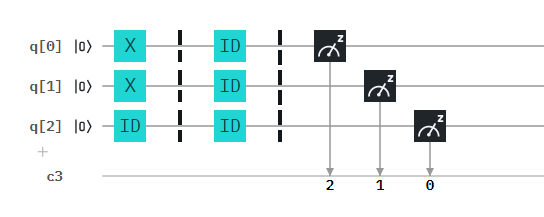


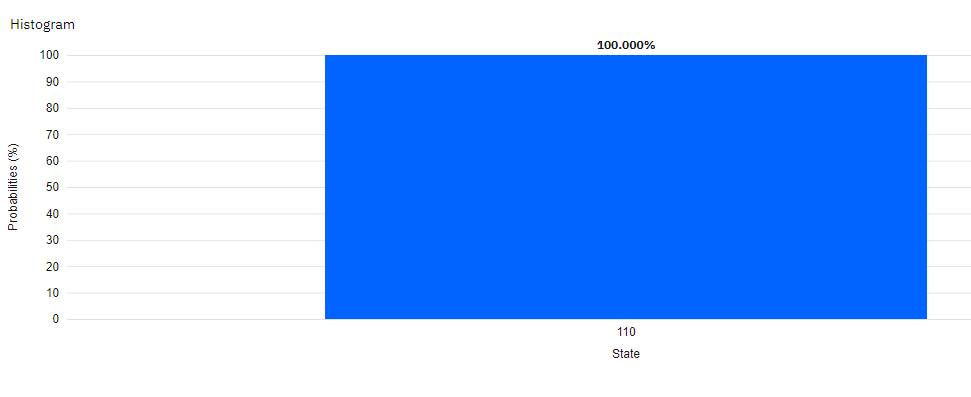
**para |101>**



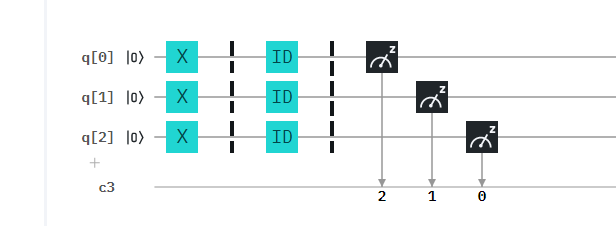


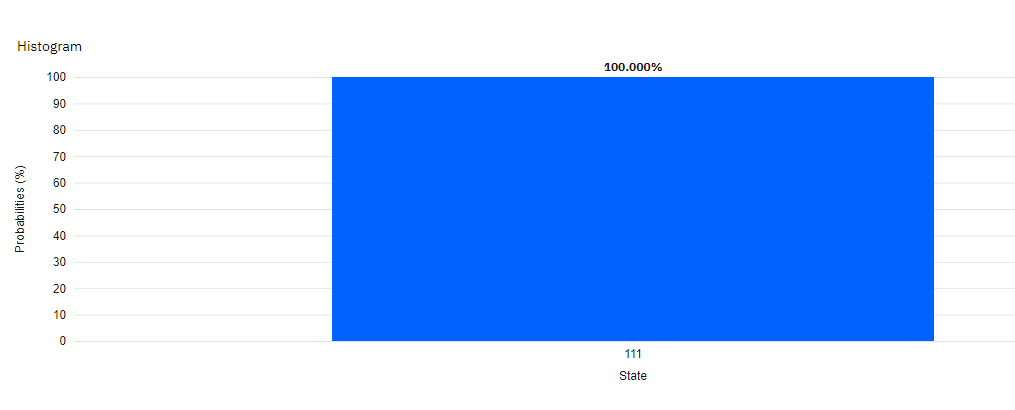
**para |110>**





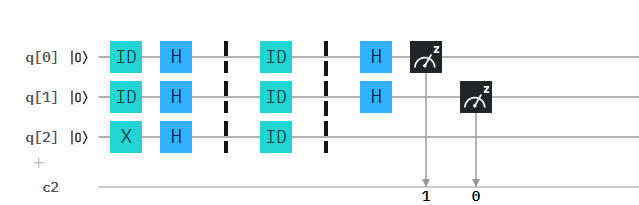
**para |111>**



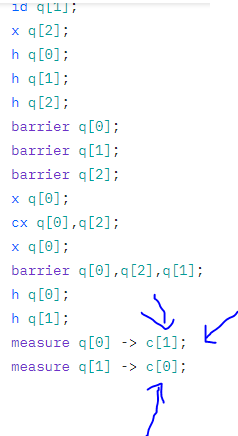


# **8. Implementando el algoritmo de Deutsch-Josza en un computador cuántico, para la segunda función**

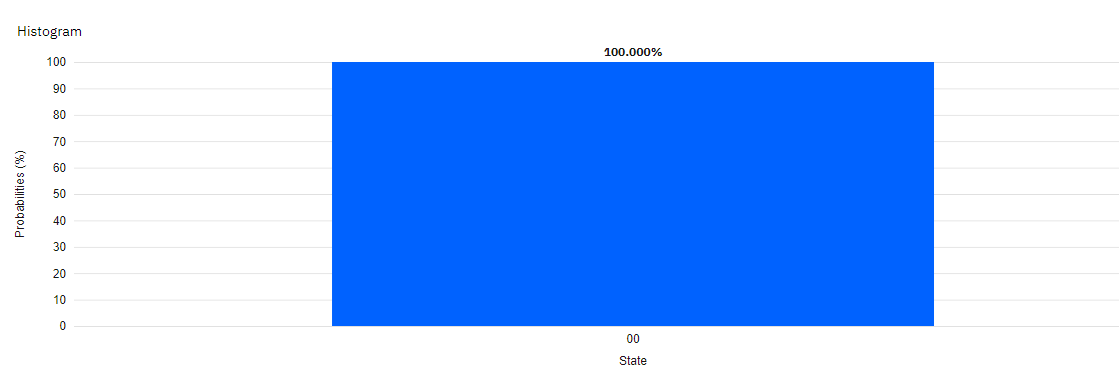
## 6.1 SEGUNDA FUNCION



Para la lectura de los datos primero se debe cambiar el mapeo como se muestra en la siguiente imagen



Ya que IBM siempre pone ese mapeo de forma inversa, de no hacerlo puede que el circuito sea correcto pero al mostrar la respuesta la muestre invertida, una vez realizado este cambio corremos el algoritmo, el cual nos dará el siguiente resultado **(ya que solo miramos los dos primeros alambres )**:

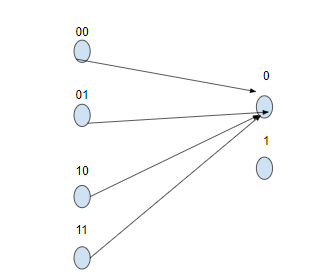


Se debe recordar que si el resultado:

* da **00** se dice que la función es constante
* en cualquier otro caso se dice que es balanceada

cómo podemos observar es igual a 00, luego f es una función constante.

Este lo podemos comprobar ya que si revisamos



Podemos observar que el 100% de las veces se hace un mapeo a 0.

# Bibliografía

* **Google Books. 2020. *Quantum Computing For Computer Scientists*. [online] Available at:** **<**https://books.google.com.co/books?id=U1chAwAAQBAJ&printsec=frontcover&dq=quantum+computing+for+computer+scientists&hl=es-419&sa=X&ved=0ahUKEwi-2ZGQ-ZzpAhVCneAKHfv-AhkQ6AEIKDAA#v=onepage&q=quantum%20computing%20for%20computer%20scientists&f=false**> [Accessed 5 May 2020].**
* **Campusvirtual.escuelaing.edu.co. 2020. *COURSE:CNYT*. [online] Available at:** **<**http://campusvirtual.escuelaing.edu.co/moodle/course/view.php?id=1741**> [Accessed 5 May 2020].**
* **Es.wikipedia.org. 2020. *Algoritmo De Deutsch-Jozsa*. [online] Available at:** **<**https://es.wikipedia.org/wiki/Algoritmo\_de\_Deutsch-Jozsa**> [Accessed 5 May 2020].**