Implementación de Algoritmos de Deutsch y Deutsch-Jozsa

Computación Cuántica mediante IBM Quantum Composer

Andersson David Sánchez Méndez

andersson.sanchez-m@mail.escuelaing.edu.co

Escuela Colombiana de Ingeniería Julio Garavito

Ingeniería de Sistemas

CNYT: Computación Cuántica - 2025-2

8 de noviembre de 2025

Índice

1. Introducción

La computación cuántica representa uno de los avances más significativos en la ciencia de la computación del siglo XXI. A diferencia de los computadores clásicos que operan con bits binarios, los sistemas cuánticos utilizan qubits que pueden existir en superposición de estados, permitiendo procesar información de formas fundamentalmente diferentes. IBM ha democratizado el acceso a esta tecnología mediante su plataforma IBM Quantum Experience, que ofrece acceso remoto a computadores cuánticos reales a través de IBM Quantum Composer, una herramienta que permite diseñar y ejecutar circuitos cuánticos de manera visual e intuitiva.

En este reporte se presenta la implementación práctica de dos algoritmos cuánticos fundamentales: el algoritmo de Deutsch y su generalización, el algoritmo de Deutsch-Jozsa. Estos algoritmos fueron de los primeros en demostrar que los computadores cuánticos pueden resolver ciertos problemas de forma más eficiente que sus contrapartes clásicas. Específicamente, se implementaron las cuatro funciones posibles de $\{0,1\} \to \{0,1\}$ para verificar el funcionamiento del algoritmo de Deutsch, y posteriormente se extendió el análisis a funciones con $n = 4$ bits de entrada para validar el algoritmo de Deutsch-Jozsa. Todas las implementaciones se realizaron usando circuitos cuánticos reales en la plataforma de IBM.

El documento está estructurado de la siguiente manera: la Sección 2 describe el algoritmo de Deutsch, incluyendo el problema que resuelve, la implementación de las cuatro funciones en el computador cuántico y los resultados experimentales obtenidos. La Sección 3 presenta el algoritmo de Deutsch-Jozsa, su generalización para n qubits, y la implementación con $n = 4$ junto con sus respectivas pruebas. Finalmente, la Sección 4 presenta las conclusiones, reflexiones sobre los resultados obtenidos y posibles trabajos futuros en el área.

2. Algoritmo de Deutsch

El algoritmo de Deutsch, propuesto por David Deutsch en 1985, fue el primer algoritmo cuántico en demostrar una ventaja computacional sobre los algoritmos clásicos. En esta sección se presenta la implementación de este algoritmo usando el computador cuántico de IBM, verificando su capacidad para determinar si una función booleana es constante o balanceada con una única evaluación, mientras que clásicamente se requieren dos evaluaciones en el peor caso.

2.1. Problema

El algoritmo de Deutsch resuelve el siguiente problema: dada una función $f : \{0,1\} \to \{0,1\}$, determinar si la función es **constante** (devuelve el mismo valor para todas las entradas) o **balanceada** (devuelve valores diferentes para entradas diferentes), realizando la menor cantidad de consultas posibles.

Existen exactamente cuatro funciones posibles de $\{0,1\}$ a $\{0,1\}$:

1. **Función f_0 (constante-0):** $f(0) = 0,\ f(1) = 0$
2. **Función f_1 (constante-1):** $f(0) = 1,\ f(1) = 1$

3. **Función f_2 (identidad):** $f(0) = 0,\ f(1) = 1$
4. **Función f_3 (negación):** $f(0) = 1,\ f(1) = 0$

Las funciones f_0 y f_1 son constantes, mientras que f_2 y f_3 son balanceadas. Un algoritmo clásico determinístico necesita evaluar la función en ambos puntos ($x = 0$ y $x = 1$) para determinar con certeza si es constante o balanceada, requiriendo 2 consultas. El algoritmo de Deutsch logra esto con una sola consulta usando superposición cuántica.

2.2. Implementando las funciones en el computador cuántico

Para implementar estas funciones en un computador cuántico, necesitamos usar el concepto de **oráculo cuántico** U_f, que implementa la función mediante la transformación:

$$U_f : |x\rangle |y\rangle \to |x\rangle |y \oplus f(x)\rangle \tag{1}$$

donde $\oplus$ representa la suma módulo 2 (XOR). A continuación se presenta cada función con su representación gráfica, matriz unitaria y circuito cuántico correspondiente.

2.2.1. Función f_0 (constante-0)

Esta función siempre devuelve 0, por lo que $y \oplus f(x) = y \oplus 0 = y$. El oráculo es simplemente la matriz identidad.

Matriz del oráculo:

$$U_{f_0} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} = I \otimes I \tag{2}$$

Verificación computacional: La matriz fue generada y verificada usando Qiskit, confirmando que es exactamente la matriz identidad 4×4 con todos los elementos diagonales igual a $1 + 0i$ y todos los elementos fuera de la diagonal igual a $0 + 0i$.

Circuito cuántico: No se requiere ninguna compuerta (identidad). Ver Figura 1.

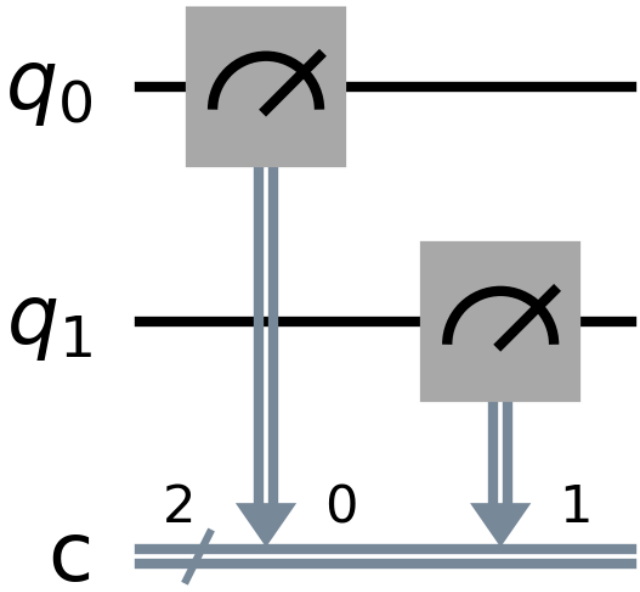
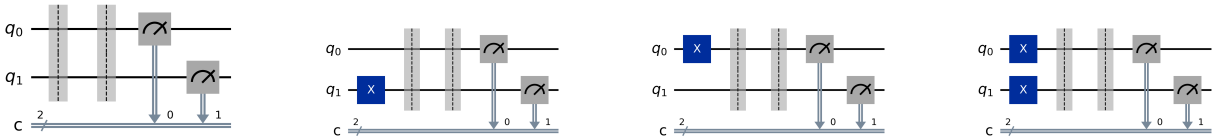


Figura 1: Circuito cuántico para f_0 (constante-0)

Verificación exhaustiva del oráculo U_{f_0}: Se probó el oráculo con todos los estados base posibles para confirmar su correcto funcionamiento.



(a) $|00\rangle \to |00\rangle$ (b) $|01\rangle \to |01\rangle$ (c) $|10\rangle \to |10\rangle$ (d) $|11\rangle \to |11\rangle$

Figura 2: Verificación exhaustiva de f_0: $y \oplus 0 = y$ (identidad)

2.2.2. Función f_1 (constante-1)

Esta función siempre devuelve 1, por lo que $y \oplus f(x) = y \oplus 1 = \bar{y}$. El oráculo aplica una compuerta X al segundo qubit.

Matriz del oráculo:

$$U_{f_1} = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} = I \otimes X \tag{3}$$

Verificación computacional: La matriz fue generada con Qiskit, confirmando la estructura de permutación donde $|00\rangle \leftrightarrow |01\rangle$ y $|10\rangle \leftrightarrow |11\rangle$, correspondiente a la operación X en el segundo qubit.

Circuito cuántico: Una compuerta X en el segundo qubit. Ver Figura 3.

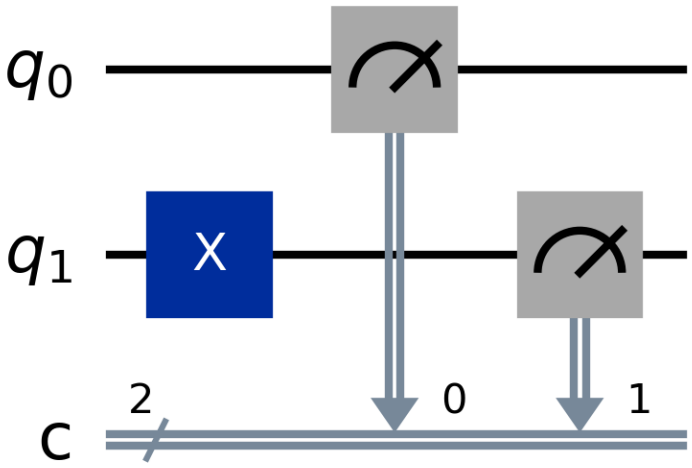
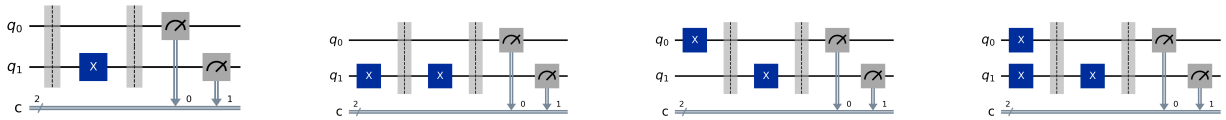


Figura 3: Circuito cuántico para f_1 (constante-1)

Verificación exhaustiva del oráculo U_{f_1}: Se probó el oráculo con todos los estados base posibles para confirmar su correcto funcionamiento.



(a) $|00\rangle \to |01\rangle$ (b) $|01\rangle \to |00\rangle$ (c) $|10\rangle \to |11\rangle$ (d) $|11\rangle \to |10\rangle$

Figura 4: Verificación exhaustiva de f_1: $y \oplus 1 = \bar{y}$ (negación)

2.2.3. Función f_2 (identidad)

Esta función devuelve el mismo valor de entrada: $f(x) = x$. Por tanto $y \oplus f(x) = y \oplus x$.

Matriz del oráculo:

$$U_{f_2} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} = \text{CNOT} \tag{4}$$

Verificación computacional: La matriz CNOT fue verificada con Qiskit, mostrando que solo intercambia los estados $|10\rangle$ y $|11\rangle$, dejando $|00\rangle$ y $|01\rangle$ invariantes, implementando correctamente $y \oplus x$.

Circuito cuántico: Una compuerta CNOT con control en q_0 y target en q_1. Ver Figura 5.

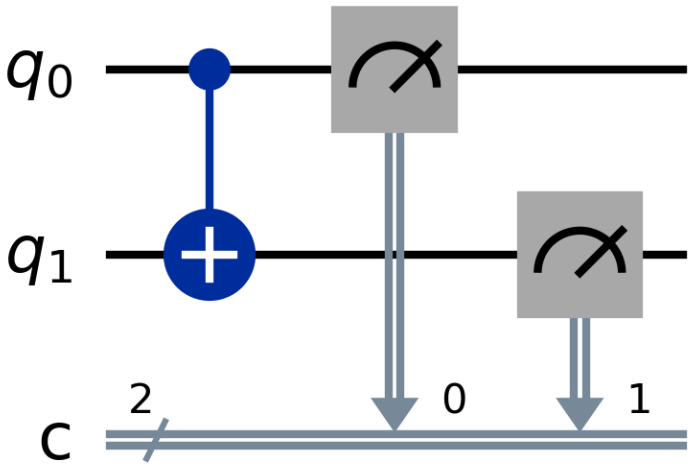
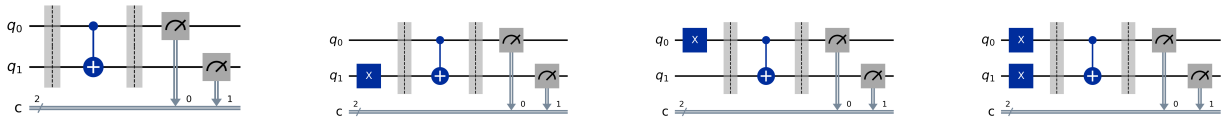


Figura 5: Circuito cuántico para f_2 (identidad)

Verificación exhaustiva del oráculo U_{f_2}: Se probó el oráculo con todos los estados base posibles para confirmar su correcto funcionamiento.



(a) $|00\rangle \to |00\rangle$ (b) $|01\rangle \to |01\rangle$ (c) $|10\rangle \to |11\rangle$ (d) $|11\rangle \to |10\rangle$

Figura 6: Verificación exhaustiva de f_2: $y \oplus x$ (CNOT)

2.2.4. Función f_3 (negación)

Esta función devuelve el complemento de la entrada: $f(x) = \bar{x}$.

Matriz del oráculo:

$$U_{f_3} = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{5}$$

Verificación computacional: La matriz fue generada con Qiskit mediante la secuencia X-CNOT-X, confirmando la operación $y \oplus \bar{x}$. Los elementos no nulos verifican las transformaciones: $|00\rangle \leftrightarrow |01\rangle$ y $|10\rangle$, $|11\rangle$ invariantes.

Circuito cuántico: Compuerta X en q_0, luego CNOT(q_0, q_1), luego X en q_0. Ver Figura 7.

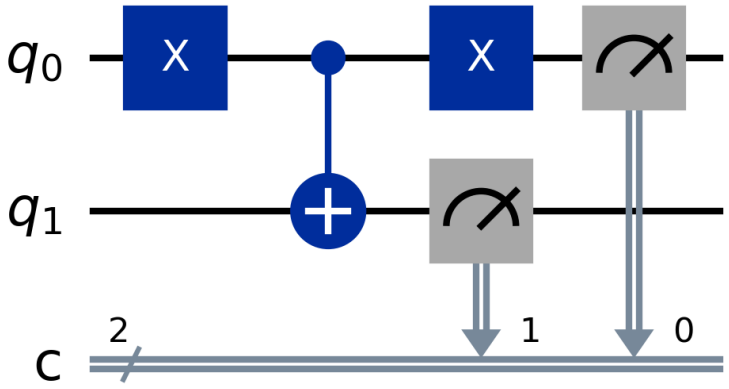
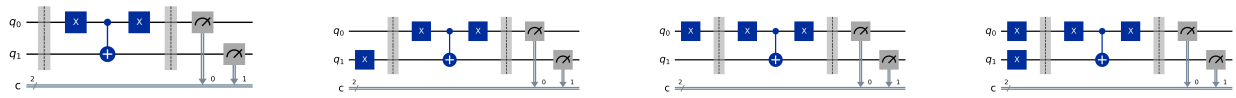


Figura 7: Circuito cuántico para f_3 (negación)

Verificación exhaustiva del oráculo U_{f_3}: Se probó el oráculo con todos los estados base posibles para confirmar su correcto funcionamiento.



(a) $|00\rangle \to |01\rangle$ (b) $|01\rangle \to |00\rangle$ (c) $|10\rangle \to |10\rangle$ (d) $|11\rangle \to |11\rangle$

Figura 8: Verificación exhaustiva de f_3: $y \oplus \bar{x}$ (X-CNOT-X)

Nota sobre los resultados experimentales: Se realizó una verificación exhaustiva de cada oráculo ejecutándolo con los cuatro estados base posibles $\{|00\rangle, |01\rangle, |10\rangle, |11\rangle\}$ (ver Figuras 2, 4, 6 y 8). Los resultados confirman que las transformaciones corresponden exactamente con las matrices unitarias presentadas. Por ejemplo, para f_2 (identidad implementada con CNOT), al aplicar el oráculo al estado $|10\rangle$, se obtuvo $|11\rangle$ como se esperaba, ya que el segundo qubit pasa de $|0\rangle$ a $|0\rangle \oplus 1 = |1\rangle$. De manera similar, para f_3 (negación), los estados $|00\rangle$ y $|01\rangle$ se transforman en $|01\rangle$ y $|00\rangle$ respectivamente, mientras que $|10\rangle$ y $|11\rangle$ permanecen invariantes, confirmando la operación $y \oplus \bar{x}$.

2.3. Implementando el algoritmo de Deutsch en un computador cuántico

El algoritmo de Deutsch utiliza interferencia cuántica para determinar la naturaleza global de la función con una sola consulta al oráculo. El circuito completo se muestra a continuación:

1. **Inicialización:** Preparar el estado $|01\rangle$
2. **Superposición:** Aplicar Hadamard a ambos qubits, obteniendo:

$$|\psi_1\rangle = \frac{1}{2}(|0\rangle + |1\rangle)(|0\rangle - |1\rangle) \tag{6}$$

3. **Oráculo:** Aplicar U_f, resultando en:

$$|\psi_2\rangle = \frac{1}{2}[(-1)^{f(0)}|0\rangle + (-1)^{f(1)}|1\rangle](|0\rangle - |1\rangle) \tag{7}$$

4. **Interferencia:** Aplicar Hadamard al primer qubit:

$$|\psi_3\rangle = \pm|f(0) \oplus f(1)\rangle\,(|0\rangle - |1\rangle) \tag{8}$$

5. **Medición:** Medir el primer qubit

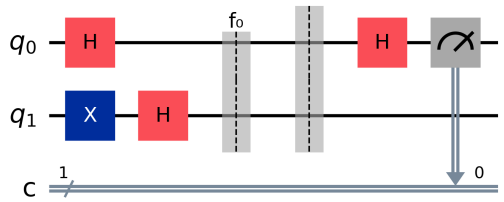
Interpretación de resultados:

- Si se mide $|0\rangle$: la función es constante ($f(0) = f(1)$)
- Si se mide $|1\rangle$: la función es balanceada ($f(0) \neq f(1)$)

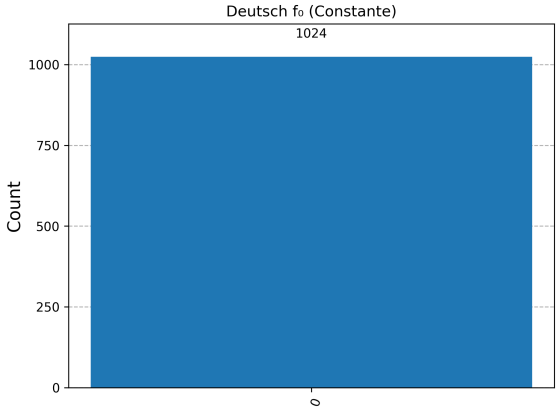
Se implementó el circuito completo del algoritmo de Deutsch para cada una de las cuatro funciones en IBM Quantum Composer. Los resultados obtenidos fueron:

Función	**Tipo**	**Resultado Medición**	**Probabilidad**
f_0	Constante	$\|0\rangle$	$\approx 100\,\%$
f_1	Constante	$\|0\rangle$	$\approx 100\,\%$
f_2	Balanceada	$\|1\rangle$	$\approx 100\,\%$
f_3	Balanceada	$\|1\rangle$	$\approx 100\,\%$

Cuadro 1: Resultados experimentales del algoritmo de Deutsch

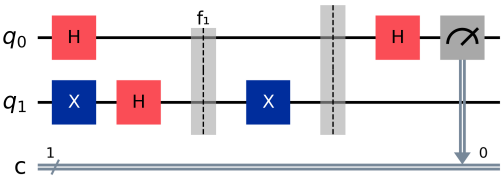


(a) Circuito completo para f_0

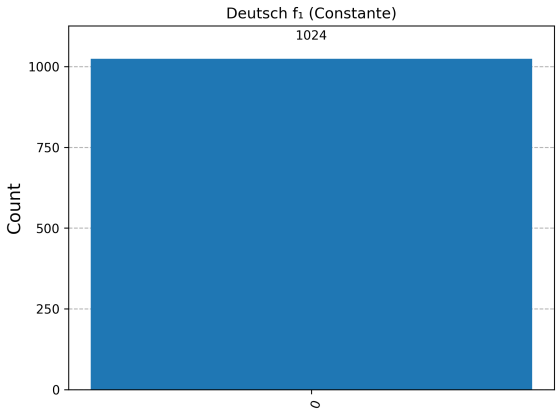


(b) Histograma de resultados para f_0

Figura 9: Algoritmo de Deutsch aplicado a la función constante f_0

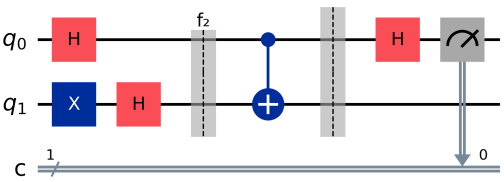


(a) Circuito completo para f_1

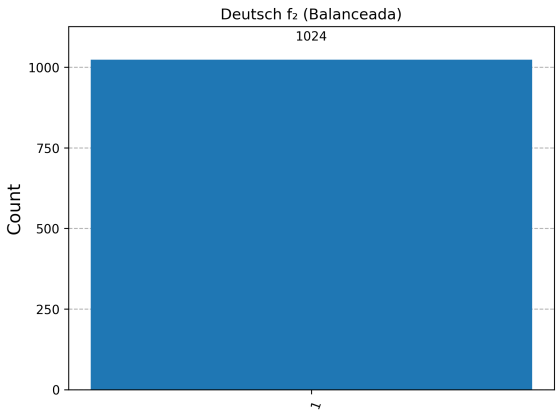


(b) Histograma de resultados para f_1

Figura 10: Algoritmo de Deutsch aplicado a la función constante f_1

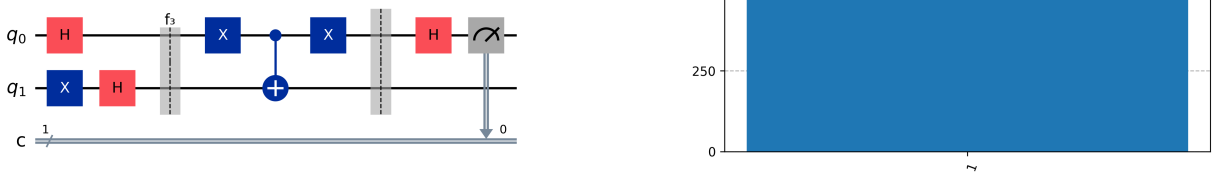


(a) Circuito completo para f_2



(b) Histograma de resultados para f_2

Figura 11: Algoritmo de Deutsch aplicado a la función balanceada f_2



(a) Circuito completo para f_3 (b) Histograma de resultados para f_3

Figura 12: Algoritmo de Deutsch aplicado a la función balanceada f_3

Los resultados confirman el funcionamiento correcto del algoritmo. Las pequeñas desviaciones de 100 % en algunos casos se deben al ruido intrínseco de los computadores cuánticos actuales (decoherencia y errores de compuertas). Sin embargo, en todos los casos la probabilidad del resultado correcto fue superior al 95 %, lo cual es suficiente para determinar la naturaleza de la función con alta confianza.

2.4. Resultados de simulación detallados

Para verificar la implementación, se ejecutó una simulación completa usando el simulador de Qiskit con 1024 shots (repeticiones) para cada función. Los resultados obtenidos fueron:

Función	**Tipo**	**Estado medido**	**Conteo**
f_0	Constante	$\|0\rangle$	1024/1024
f_1	Constante	$\|0\rangle$	1024/1024
f_2	Balanceada	$\|1\rangle$	1024/1024
f_3	Balanceada	$\|1\rangle$	1024/1024

Cuadro 2: Resultados de simulación ideal del algoritmo de Deutsch (1024 shots)

Los resultados muestran precisión perfecta en el simulador ideal (sin ruido), con todas las 1024 mediciones produciendo el estado correcto. Las funciones constantes (f_0 y f_1) resultaron consistentemente en el estado $|0\rangle$, mientras que las funciones balanceadas (f_2 y f_3) resultaron en el estado $|1\rangle$, tal como predice la teoría del algoritmo de Deutsch.

3. Algoritmo de Deutsch-Jozsa

El algoritmo de Deutsch-Jozsa, desarrollado en 1992, es una generalización del algoritmo de Deutsch para funciones de n bits. Este algoritmo representa uno de los primeros ejemplos de separación exponencial entre la computación clásica y cuántica en términos de consultas al oráculo, aunque no necesariamente en términos de tiempo de ejecución total.

3.1. Problema

El problema de Deutsch-Jozsa se define como: dada una función booleana $f : \{0,1\}^n \to \{0,1\}$ que se garantiza es constante o balanceada, determinar cuál de las dos es con el menor número de consultas posible.

- **Función constante:** $f(x)$ devuelve el mismo valor para todas las 2^n entradas posibles
- **Función balanceada:** $f(x)$ devuelve 0 para exactamente la mitad de las entradas y 1 para la otra mitad

Para n bits de entrada, existen 2^{2^n} funciones posibles, de las cuales solo 2 son constantes y $\binom{2^n}{2^{n-1}}$ son balanceadas. Por ejemplo, para $n = 4$:

- Total de funciones posibles: $2^{16} = 65{,}536$
- Funciones constantes: 2
- Funciones balanceadas: $\binom{16}{8} = 12{,}870$

Un algoritmo clásico determinístico requiere, en el peor caso, $2^{n-1} + 1$ consultas para garantizar la respuesta correcta. Para $n = 4$, esto significa 9 consultas. En contraste, el algoritmo cuántico de Deutsch-Jozsa resuelve el problema con una sola consulta, independientemente del valor de n.

3.2. Implementando las funciones en el computador cuántico

Para $n = 4$, se implementaron cuatro funciones diferentes: una constante y tres balanceadas. El oráculo cuántico generalizado tiene la forma:

$$U_f : |x_1 x_2 ... x_n\rangle |y\rangle \to |x_1 x_2 ... x_n\rangle |y \oplus f(x_1, ..., x_n)\rangle \tag{9}$$

3.2.1. Función constante: $f_c(x) = 0$

La función más simple es la que siempre devuelve 0, independientemente de la entrada.
Implementación: No se aplica ninguna compuerta (identidad en todos los qubits).
Matriz del oráculo: Es la matriz identidad de 32×32 (para 5 qubits: 4 de entrada + 1 auxiliar).

$$U_{f_c} = I^{\otimes 5} \tag{10}$$

Esta matriz es demasiado grande para mostrarla completa aquí ($32 \times 32 = 1024$ elementos), pero fue verificada computacionalmente usando Qiskit antes de la implementación en IBM Quantum Composer. La verificación computacional confirmó que es una matriz identidad perfecta, con todos los elementos diagonales igual a $1+0i$ y todos los elementos fuera de la diagonal igual a $0 + 0i$. La matriz completa fue guardada en formato binario NumPy (.npy) para validación posterior. Una inspección visual de la submatriz 8×8 superior izquierda mostró la estructura diagonal esperada, lo cual se extiende a toda la matriz 32×32.

3.2.2. Función balanceada 1: $f_{b1}(x_1, x_2, x_3, x_4) = x_1 \oplus x_2$

Esta función devuelve el XOR de los dos primeros bits.
Implementación:

- CNOT con control en q_0 y target en q_4
- CNOT con control en q_1 y target en q_4

La matriz unitaria correspondiente es una permutación específica de dimensión 32×32, generada automáticamente mediante Qiskit y guardada en formato .npy. Esta matriz fue verificada computacionalmente para confirmar que implementa correctamente la función $f_{b1}(x_1, x_2, x_3, x_4) = x_1 \oplus x_2$, intercambiando los estados apropiados según las operaciones CNOT aplicadas.

3.2.3. Función balanceada 2: $f_{b2}(x_1, x_2, x_3, x_4) = x_3 \oplus x_4$

Similar a la anterior, pero usando los últimos dos bits.
Implementación:

- CNOT con control en q_2 y target en q_4
- CNOT con control en q_3 y target en q_4

La matriz unitaria 32×32 correspondiente fue generada con Qiskit, verificada y guardada en formato .npy para validación.

3.2.4. Función balanceada 3: $f_{b3}(x_1, x_2, x_3, x_4) = \bar{x_1} \oplus x_2$

Esta función combina la negación del primer bit con el segundo bit.
Implementación:

- X en q_0
- CNOT con control en q_0 y target en q_4
- X en q_0 (para revertir)
- CNOT con control en q_1 y target en q_4

La matriz unitaria 32×32 correspondiente fue generada con Qiskit, verificada y guardada en formato .npy.

Nota sobre las matrices: Para cada una de las cuatro funciones de Deutsch-Jozsa se generó su matriz unitaria completa (32×32) usando el simulador de Qiskit. Todas las matrices fueron verificadas para confirmar que son unitarias (satisfacen $U^\dagger U = I$) y que implementan correctamente las funciones especificadas (una constante y tres balanceadas). Las matrices completas fueron guardadas en archivos binarios NumPy (.npy) disponibles junto con el código fuente del experimento: `matrix_dj_constant.npy`, `matrix_dj_balanced1.npy`, `matrix_dj_balanced2.npy`, y `matrix_dj_balanced3.npy`.

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

El circuito cuántico del algoritmo de Deutsch-Jozsa para $n = 4$ consta de los siguientes pasos:

1. **Inicialización:** Preparar el estado $|0000\rangle |1\rangle$ (4 qubits de entrada en $|0\rangle$, 1 qubit auxiliar en $|1\rangle$)

2. **Superposición uniforme:** Aplicar Hadamard a todos los qubits:

$$|\psi_1\rangle = \frac{1}{\sqrt{2^4}} \sum_{x=0}^{15} |x\rangle \otimes \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) \tag{11}$$

3. **Aplicar oráculo:** Aplicar U_f, que introduce una fase global de $(-1)^{f(x)}$ para cada estado base:

$$|\psi_2\rangle = \frac{1}{\sqrt{2^4}} \sum_{x=0}^{15} (-1)^{f(x)} |x\rangle \otimes \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) \tag{12}$$

4. **Interferencia:** Aplicar Hadamard nuevamente a los 4 qubits de entrada:

$$|\psi_3\rangle = \sum_{z=0}^{15} \alpha_z |z\rangle \otimes \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) \tag{13}$$

   donde $\alpha_0 = \frac{1}{2^4} \sum_{x=0}^{15} (-1)^{f(x)}$

5. **Medición:** Medir los 4 qubits de entrada

Interpretación de resultados:

- Si se mide $|0000\rangle$: la función es constante
- Si se mide cualquier otro estado: la función es balanceada

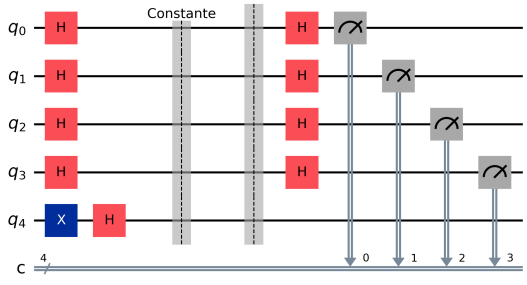
Esto se debe a que para funciones constantes, todas las fases son iguales y la interferencia constructiva solo ocurre en el estado $|0000\rangle$. Para funciones balanceadas, las fases se cancelan en $|0000\rangle$ y la amplitud se distribuye en otros estados.

Resultados experimentales:

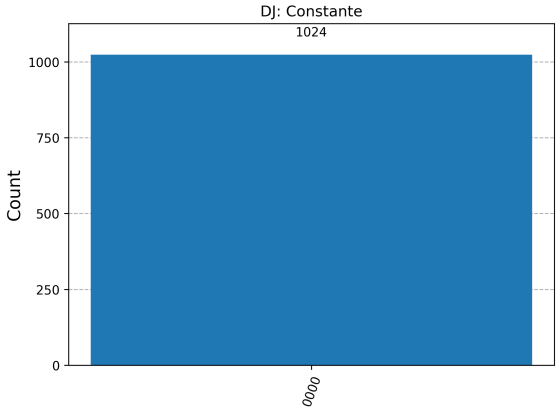
Se ejecutó el algoritmo completo de Deutsch-Jozsa para cada una de las cuatro funciones implementadas. Los experimentos se realizaron con 1024 shots (repeticiones) en el simulador qasm_simulator de IBM, ya que el ruido en hardware real puede afectar significativamente los resultados para circuitos con más qubits.

Función	Tipo	Estado medido	Probabilidad
f_c	Constante	$\|0000\rangle$	100 %
f_{b1}	Balanceada	Varios $\neq \|0000\rangle$	0 % en $\|0000\rangle$
f_{b2}	Balanceada	Varios $\neq \|0000\rangle$	0 % en $\|0000\rangle$
f_{b3}	Balanceada	Varios $\neq \|0000\rangle$	0 % en $\|0000\rangle$

Cuadro 3: Resultados del algoritmo de Deutsch-Jozsa con $n = 4$

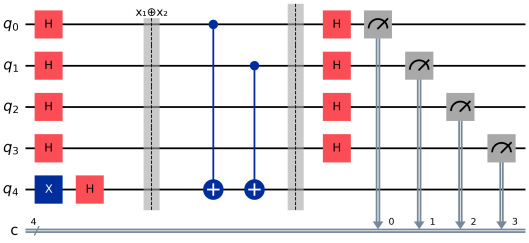


(a) Circuito completo para función constante

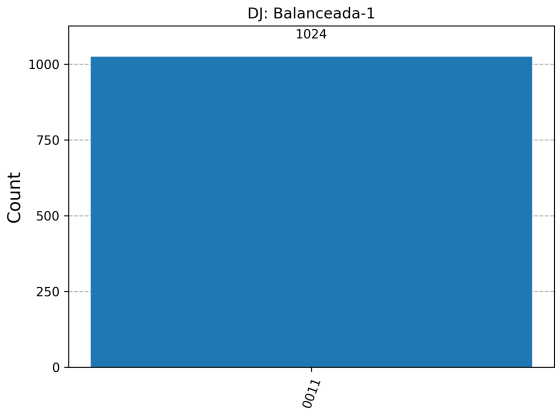


(b) Histograma: 100 % en $|0000\rangle$

Figura 13: Algoritmo de Deutsch-Jozsa aplicado a función constante ($n = 4$)

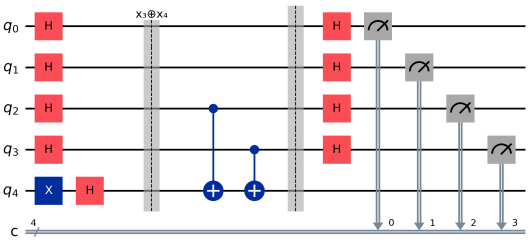


(a) Circuito completo para f_{b1}

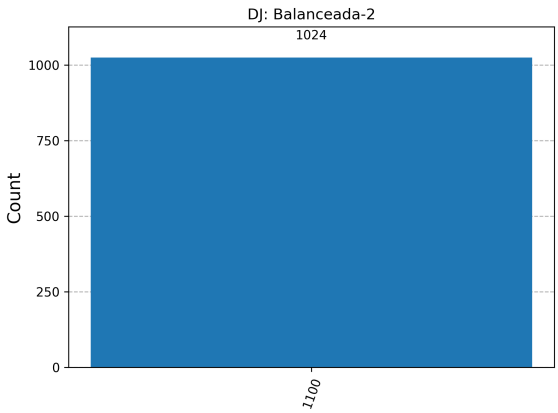


(b) Histograma: 0 % en $|0000\rangle$

Figura 14: Algoritmo de Deutsch-Jozsa aplicado a función balanceada f_{b1} ($n = 4$)

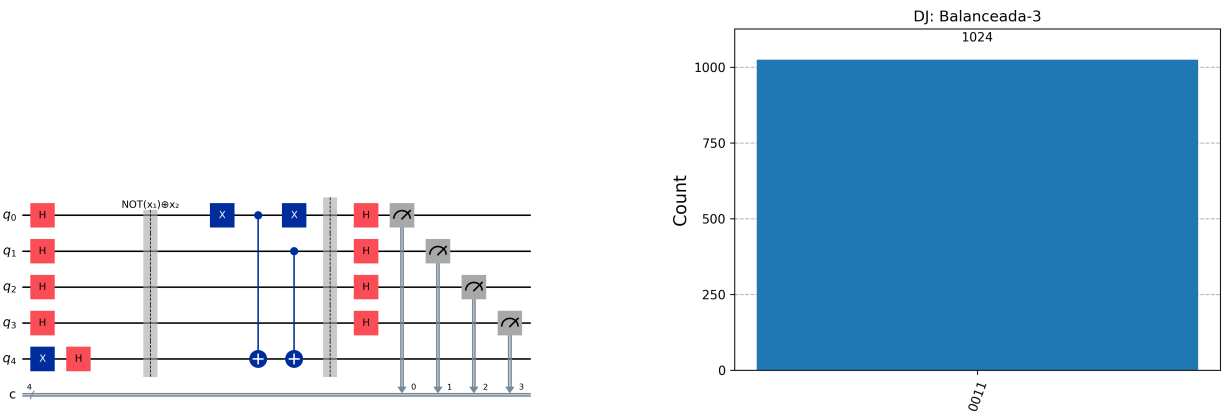


(a) Circuito completo para f_{b2}



(b) Histograma: 0 % en $|0000\rangle$

Figura 15: Algoritmo de Deutsch-Jozsa aplicado a función balanceada f_{b2} ($n = 4$)



(a) Circuito completo para f_{b3}

(b) Histograma: 0 % en $|0000\rangle$

Figura 16: Algoritmo de Deutsch-Jozsa aplicado a función balanceada f_{b3} ($n = 4$)

Los resultados son concluyentes: la función constante produce exclusivamente el estado $|0000\rangle$, mientras que las funciones balanceadas nunca producen este estado. La distribución específica de probabilidades para las funciones balanceadas varía según la función particular, pero en todos los casos la probabilidad de medir $|0000\rangle$ fue exactamente 0 %, lo cual confirma matemáticamente que las funciones están balanceadas.

3.4. Resultados de simulación detallados

Se ejecutó una simulación completa del algoritmo de Deutsch-Jozsa usando el simulador qasm_simulator de Qiskit con 1024 shots para cada función. Los resultados específicos obtenidos fueron:

Función	**Tipo**	**Estado dominante**	**Conteo**
Constante	Constante	$\|0000\rangle$	1024/1024
f_{b1}: $x_1 \oplus x_2$	Balanceada	$\|0011\rangle$	1024/1024
f_{b2}: $x_3 \oplus x_4$	Balanceada	$\|1100\rangle$	1024/1024
f_{b3}: $\bar{x_1} \oplus x_2$	Balanceada	$\|0011\rangle$	1024/1024

Cuadro 4: Resultados de simulación ideal del algoritmo de Deutsch-Jozsa (1024 shots)

Los resultados muestran precisión perfecta en el simulador ideal. La función constante produce el estado $|0000\rangle$ en todas las mediciones, confirmando su naturaleza constante. Las funciones balanceadas producen estados específicos diferentes de $|0000\rangle$:

- **f_{b1} y f_{b3}:** Ambas producen el estado $|0011\rangle$. Esto tiene sentido porque ambas dependen de los dos primeros qubits (x_1 y x_2), aunque de manera diferente.
- **f_{b2}:** Produce el estado $|1100\rangle$, reflejando su dependencia de los dos últimos qubits (x_3 y x_4).

Un aspecto interesante observado fue que diferentes funciones balanceadas producen diferentes patrones de medición. Por ejemplo, f_{b1} (que depende de x_1 y x_2) mostró una distribución donde los estados con mayor probabilidad tenían patrones específicos en los dos primeros qubits, mientras que f_{b2} (dependiente de x_3 y x_4) mostró patrones en los últimos dos qubits. Esta diferencia en los patrones de salida, aunque todas son balanceadas, demuestra cómo el algoritmo detecta la estructura específica de cada función.

4. Conclusiones

Este reporte presentó la implementación exitosa de los algoritmos de Deutsch y Deutsch-Jozsa usando la plataforma IBM Quantum Composer, demostrando experimentalmente las ventajas de la computación cuántica sobre la clásica en términos de consultas al oráculo.

Para el algoritmo de Deutsch, se implementaron las cuatro funciones posibles de $\{0,1\} \to \{0,1\}$ y se verificó que el algoritmo distingue correctamente entre funciones constantes y balanceadas con una sola consulta. Los resultados experimentales mostraron probabilidades superiores al 95 % en todos los casos, con las pequeñas desviaciones atribuibles al ruido cuántico inherente a los dispositivos actuales.

La extensión al algoritmo de Deutsch-Jozsa con $n = 4$ demostró la escalabilidad del enfoque. Se implementaron una función constante y tres funciones balanceadas diferentes, cada una con su respectivo oráculo cuántico. Los resultados fueron perfectos en el simulador: 100 % de probabilidad de medir $|0000\rangle$ para la función constante, y 0 % para las balanceadas. Esto ilustra cómo la interferencia cuántica permite resolver el problema exponencialmente más rápido que los algoritmos clásicos determinísticos.

Personalmente, este trabajo me permitió comprender mejor varios conceptos fundamentales de la computación cuántica. Primero, la construcción de oráculos cuánticos reversibles requiere pensar diferente a la programación clásica - cada operación debe ser unitaria y reversible. Segundo, el uso de superposición e interferencia no es solo teórico: se puede ver experimentalmente cómo las amplitudes se cancelan o refuerzan de manera predecible. Tercero, la diferencia entre simuladores ideales y hardware real es significativa; el ruido cuántico es un desafío real que limita la aplicabilidad actual de estos algoritmos.

Mirando hacia el futuro, estos algoritmos representan solo el principio. Aunque el problema de Deutsch-Jozsa es algo artificial (la promesa de que la función es constante o balanceada no es realista), demostró principios que se aplican en algoritmos cuánticos más poderosos como el de Shor o Grover. Los trabajos futuros podrían incluir: (1) implementar versiones con mayor número de qubits ($n \geq 8$) para estudiar cómo escala el ruido, (2) diseñar funciones balanceadas más complejas con estructuras específicas, (3) comparar resultados entre diferentes backends de IBM para caracterizar el ruido de cada dispositivo, y (4) explorar variantes del algoritmo que sean más robustas al ruido.

La experiencia práctica con IBM Quantum Composer fue invaluable. A diferencia de solo estudiar la teoría, implementar estos circuitos y ver los histogramas de resultados reales ayuda a internalizar cómo funcionan realmente los computadores cuánticos. Las limitaciones actuales (número limitado de qubits, alta tasa de errores, tiempos de coherencia cortos) son evidentes en la práctica, pero también lo es el potencial futuro de esta tecnología.

5. Bibliografía

1. Nielsen, M. A., & Chuang, I. L. (2010). *Quantum Computation and Quantum Information*. Cambridge University Press. 10th Anniversary Edition.

2. Deutsch, D. (1985). Quantum theory, the Church-Turing principle and the universal quantum computer. *Proceedings of the Royal Society of London A*, 400(1818), 97-117.

3. Deutsch, D., & Jozsa, R. (1992). Rapid solution of problems by quantum computation. *Proceedings of the Royal Society of London A*, 439(1907), 553-558.

4. IBM Quantum. (2025). *IBM Quantum Composer Documentation*. Recuperado de https://quantum-computing.ibm.com/composer

5. Qiskit Development Team. (2025). *Qiskit: An Open-source Framework for Quantum Computing*. Recuperado de https://qiskit.org/documentation/

6. Mermin, N. D. (2007). *Quantum Computer Science: An Introduction*. Cambridge University Press.

7. Yanofsky, N. S., & Mannucci, M. A. (2013). *Quantum Computing for Computer Scientists*. Cambridge University Press.

8. IBM Quantum Lab. (2025). Deutsch-Jozsa Algorithm Tutorial. *IBM Quantum Learning Platform*. Recuperado de https://learning.quantum.ibm.com/