UNIVERSIDAD TECNOLÓGICA CENTROAMERICANA

FACULTAD DE INGENIERÍA

PROYECTO DE INVESTIGACIÓN

IMPLEMENTACIÓN DE ALGORITMOS CUÁNTICOS EN IBM QUANTUM EXPERIENCE

PREVIO A LA OBTENCIÓN DEL TÍTULO:

INGENIERO EN MECATRÓNICA

PRESENTADO POR:

21241154 JOSÉ DAVID PERDOMO MARADIAGA

ASESOR: JOSÉ LUIS ORDOÑEZ ÁVILA

CAMPUS: SAN PEDRO SULA; JULIO, 2020

DEDICATORIA

La presente tesis está dedicada a las personas que me inspiraron y motivaron a salir adelante. A Dios que nos da la sabiduría, y siempre me acompaño en esta etapa. A mis padres, que con esfuerzo y dedicación me brindaron mi educación. A mi esposa, que con paciencia y con su apoyo incondicional me ayudo a sobreponerme a los momentos más difíciles en este largo camino. A mi hijo primogénito, que me dio una razón más para esforzarme aún más en esta etapa final.

AGRADECIMIENTO

Primeramente, mi agradecimiento eterno con Dios, por darme fuerzas y sabiduría para lograr culminar mis estudios. A mis padres por siempre apoyarme a lo largo de este arduo camino sin importar las circunstancias. A mi esposa por siempre creer en mí, apoyarme en los momentos más difíciles y motivarme cuando los fracasos me desmotivaban. A mi familia en general por apoyarme siempre. A todos, ¡Muchas gracias!

RESUMEN EJECUTIVO

La computación cuántica ha sido investigada y desarrollada desde 1960 aunque aún es desconocida para muchas personas. Los sistemas de computación clásicos están llegando a los límites físicos de la construcción de transistores, dispositivo clave para el funcionamiento de los mismos. Los fabricantes se ven obligados a reducir el tamaño de estos dispositivos para lograr incluir más en sus chips y así alcanzar potencias mayores. No parece existir una solución viable para esta problemática por lo que las apuestas están en la computación cuántica. Esta tecnología aprovecha los fenómenos de la física cuántica para potenciar sus capacidades de procesamiento, como ser la superposición, entrelazamiento e incluso el túnel cuántico, que es responsable de hacer fallar los chips clásicos a escalas muy pequeñas. Al ser una tecnología en desarrollo, aún tiene espacio de mejora en corrección de errores y velocidad de procesamiento. Se comprobaron los algoritmos cuánticos más prominentes en la plataforma cuántica IBM Quantum Experience. Los resultados obtenidos en estas implementaciones fueron satisfactorios, pudiendo comprobar con éxito el funcionamiento de estos algoritmos. De igual manera el presente documento hace una comparación entre los diferentes procesadores cuánticos de IBM y analiza las diferencias en resultados de los mismos algoritmos basándose en las calibraciones para cada procesador. Se ha logrado identificar que cada procesador obtendrá una tasa de éxito mayor dependiendo del tipo de compuerta utilizada, y la tasa de error para la misma obtenida de su hoja de calibración. Los efectos del ruido y la decoherencia son notables, afectando negativamente algunos resultados obtenidos específicamente en algoritmos de mayor tamaño.

Palabras clave: computación cuántica, algoritmo de Deutsch, algoritmo de Grover, algoritmo de Shor, IBM Quantum experience, algoritmos cuánticos.

ABSTRACT

Quantum computing has been researched and developed since 1960, even though it's still unknown to most people. Classic computer systems are reaching the physical limits of the construction size of transistors, a key device for their operation. Manufacturers are forced to reduce the size of these devices to include a bigger amount of transistors in their chips and thus achieve higher processing speeds. Up to date, there isn't a viable solution to this problem so the stakes are on quantum computing. This technology takes advantage of the properties of quantum physics to enhance its processing capabilities, such as superposition, entanglement, and even quantum tunneling, which is responsible for causing classic chips to fail at very small scales. As a technology under development, it still has room for improvement in error correction and processing speeds. The most prominent quantum algorithms were tested on the IBM Quantum Experience platform. The results obtained in these implementations were satisfactory, running the algorithms on a real quantum processor. Similarly, this document compares the IBM quantum processors and analyzes the differences in results obtained on the same algorithms based on the calibrations for each processor. It has been proven that each processor will obtain a higher success rate depending on the type of gate used, and the error rate for each gate. This error rates can be obtained from the processors calibration sheets. The effects of noise and decoherence are present, negatively affecting the results obtained specifically in larger algorithms.

Keywords: Quantum computing, quantum algorithms, Deutsch algorithm, Grover algorithm, Shor algorithm, superposition, entanglement, quantum tunneling, IBM Quantum experience, quantum processor.

Índice de contenido

Índice de Ilustraciones

Índice de Ecuaciones

Índice de tablas

I. Introducción

Capitulo II

En esta sección se estudiara la problemática que motivo la presente investigación. Las limitantes físicas para la construcción de hardware en computación clásica, y la falta de información para la ejecución de algoritmos cuánticos han sido un tema que aún permanece relativamente desconocido.

Capitulo III

En este capítulo se sientan las bases necesarias para comprender el funcionamiento de la computación clásica y cuántica. De igual manera, se expone la problemática abordada en el capítulo II con más profundidad, y se define porque la computación cuántica parece ser el método de computación del futuro.

Capitulo IV

Se identifica la metodología a utilizar, las variables de investigación y toda la información pertinente para llevar a cabo la investigación. Se determinan los ciclos de la espiral a implementar y se realiza un cronograma de actividades para garantizar el cumplimiento en tiempo y forma de los objetivos.

Capitulo V

En esta sección se analizan los resultados obtenidos en la presente investigación. Se analizan los datos de las simulaciones y ejecuciones en procesadores cuánticos, y se comparan con investigaciones científicas anteriores con información similar. Se estudia la factibilidad de implementación de cada algoritmo, y la eficiencia con la que se ejecuta en un procesador real en comparación con un entorno simulado.

Capitulo VI

Se vierten las conclusiones obtenidas de los resultados de la presente investigación, y se responden las preguntas de investigación iniciales.

Capitulo VII

Esta es la última sección de la presente investigación, y se realizan recomendaciones para futuras investigaciones en la misma área, en base a los conocimientos adquiridos en el presente documento

II. Planteamiento del problema

En el presente capitulo se abordara la problemática de las limitaciones a las que está llegando la computación clásica. La computación ha tenido un crecimiento exponencial, teniendo prototipos más potentes y capaces cada año. Sin embargo, aun hoy en día existe una limitación en cuanto a las operaciones que pueden realizar, y la rapidez con la que lo hacen. Adicionalmente, año con año está llegando a los límites físicos de construcción de hardware.

2.1 Precedentes del problema

Desde la aparición de las primeras computadoras rudimentarias en 1642 de las manos del científico francés Blaise Pascal, la humanidad comenzó una nueva era tecnológica. Estas computadoras ocupaban enormes tamaños, tenían capacidades limitadas y no estaban a la altura de las necesidades humanas. Con el avance de la tecnología y la revolución industrial, las computadoras se volvieron más potentes reduciendo su tamaño significativamente (Cedano Olvera, Cedano Rodriguez, Rubio Gonzalez, & Vega Gutierrez, 2014). En 1965, el co-fundador de Intel Gordon Moore afirmo que el número de transistores por centímetro cuadrado en un circuito integrado se duplicaba cada año, aunque posteriormente modifico su afirmación indicando que se duplicaría cada 18 meses y dicha tendencia continuaría durante las próximas dos décadas (Cheang, 2005). Si el incremento en la potencia de los procesadores continuara como lo ha venido haciendo, se estima que para el 2050 los chips de computadora deberán tener 10^{16} compuertas lógicas y operar a una velocidad de 10^{14} Hz, lo que significaría un total de 10^{30} operaciones lógicas por segundo. Este crecimiento es improbable con la tecnología que poseemos actualmente y proyecta el final de la computación clásica (Gruska, 2000).

2.2 Definición del problema

Para poder duplicar los circuitos contenidos por centímetro cuadrado como lo afirmaba la ley de Moore, es necesario reducir el tamaño de los transistores usados para las operaciones lógicas, sin embargo están llegando a los límites físicos de unos cuantos nanómetros. Este límite tergiversa las leyes de la física clásica y fuerza a los chips a regirse bajo las reglas de la física cuántica, causando así fallos generalizados en los mismos al permitir el paso de información a través de compuertas lógicas que deberían de permanecer cerradas (Cedano Olvera, Cedano Rodriguez, Rubio Gonzalez, & Vega Gutierrez, 2014). La computación cuántica ha surgido como una posible solución, sin embargo a la fecha su implementación es bastante limitada. Existen muy pocos procesadores cuánticos accesibles al público y sus parámetros de ejecución son desconocidos. De igual forma, el número de qubits necesarios para para la ejecución de algunos algoritmos sobrepasa las capacidades de los procesadores cuánticos actuales.

2.3 Justificación

Existen un sinnúmero de resultados convincentes que afirman que la computación cuántica es exponencialmente más potente en comparación a las clásica (Gruska, 2000). La plataforma IBM Quantum Experience ofrece una plataforma en línea para el desarrollo y ejecución de algoritmos cuánticos en tiempo real, en un entorno simulado y un procesador superconductor real. Esta interfaz aporta los datos técnicos de ejecución de los algoritmos para su posterior análisis. Es posible simplificar un circuito cuántico para permitirle ser ejecutado en un número inferior de qubits a los que teóricamente requeriría.

2.4 Preguntas de investigación

1. ¿Qué es la computación cuántica y cómo funciona?
2. ¿Podemos ejecutar algoritmos clásicos en un procesador cuántico, y viceversa?
3. ¿Los ordenadores cuánticos presentan realmente una ventaja tangible?
4. ¿Es posible alcanzar velocidades de procesamiento más altas en algoritmos cuánticos, en relación a los clásicos?
5. ¿El inicio de la computación cuántica presenta un problema con los sistemas de seguridad actual?

2.5 Objetivos

2.5.1 Objetivo General

Demostrar la factibilidad de implementación de los algoritmos cuánticos de mayor renombre, ejecutando los mismos en la plataforma cuántica de IBM Quantum Experience.

2.5.2 Objetivos específicos

1. Definir los principios de funcionamiento de la computación cuántica y compararlos con la clásica.
2. Determinar el rol de la reversibilidad en los algoritmos.
3. Identificar la factibilidad de implementación de los algoritmos cuánticos.
4. Analizar los tiempos de procesamiento de un algoritmo cuántico.
5. Comprobar el algoritmo de Shor y analizar si realmente es una amenaza tangible a la criptografía actual.

III. Marco Teórico

Si se analizaran los pilares de la sociedad actual, sin duda se definiría a los avances tecnológicos como uno de los más grandes aliados de la humanidad. Las computadoras digitales han hecho posible un sin fin de avances científicos. Actualmente, las computadoras digitales son responsables de procesamiento de datos de negocios, control de tráfico aéreo, e incluso son las pioneras de la carrera espacial (Mano, Logica Digital y Diseño de Computadores, 1989) . Si bien esta ha sido la realidad que ha conocido siempre la mayoría de las personas, se requirieron de muchos años y la colaboración de grandes mentes científicas para llegar al punto en el que se encuentra la tecnología ahora.

3.1 Antecedentes de la computación clásica

En la antigüedad se usaban métodos rudimentarios para cálculos. Se conoce que los antepasados usaban piedras, nudos e incluso sus dedos para llevar a cabo cálculos de suma y resta sencillos. Posteriormente, aparecería en Egipto y China el ábaco, una herramienta mecánica para la suma que simplifica hasta cierto punto algunas operaciones matemáticas (Cedano Olvera, Cedano Rodriguez, Rubio Gonzalez, & Vega Gutierrez, 2014). Si bien las matemáticas de la antigüedad eran relativamente más sencillas en comparación a las actuales, el volumen de cálculos realizados siempre causaba el inherente error humano.

Fue hasta 1642 que aparecería la primera máquina mecánica para sumar, inventada por el científico francés Blaise Pascal. Este método novedoso consistía en una serie de engranes y ruedas dentadas que llevarían a cabo las sumas requeridas. Es muy común encontrarlas en las primeras cajas registradoras e incluso en los odómetros de ciertos modelos vehiculares (Cedano Olvera, Cedano Rodriguez, Rubio Gonzalez, & Vega Gutierrez, 2014). Si bien esta máquina era un considerable avance tecnológico para su época, aun carecía de una funcionalidad universal para ser considerada como una computadora, al igual que poseía una limitación en cuanto a las operaciones aritméticas que podía realizar.

A mediados del siglo XIX, la sociedad se vio en la necesidad de realizar cálculos de manera más precisas y rápidas, todo esto a raíz de la revolución industrial. Esto motivo al científico inglés Charles Babbage a diseñar una maquina diferencial y analítica. Estas máquinas eran capaces de realizar hasta 60 operaciones por segundo y con una asombrosa precisión. También era posible

realizar operaciones trigonométricas y logarítmicas en las mismas, y ofrecían la posibilidad de hacer cálculos repetitivos con el uso de tarjetas perforadas, que serían consideradas en la actualidad como el primer avance humano hacia la programación de una computadora. Estas máquinas requerían miles de engranes y transmisiones mecánicas, y ocupaban el área de un campo de futbol (Cedano Olvera, Cedano Rodriguez, Rubio Gonzalez, & Vega Gutierrez, 2014).

Si bien Charles Babbage logro un enorme salto en cuanto a las capacidades aritméticas de los prototipos hasta la fecha, el enorme tamaño de su máquina y la falta de capacidad de programación aun dejaba mucho espacio para mejora. Dicha problemática se resolvería con los avances tecnológicos en electrónica que se habrían de dar en los años subsiguientes. En 1937 inspirado en la idea de las máquinas de Babbage, el investigador Howard Aiken y un grupo de ingenieros de IBM diseñarían la primera computadora electromecánica. La misma poseía capacidades mucho más avanzadas a las máquinas de Babbage y utilizaba entrada de datos de tarjetas y cintas perforadas. Solo 4 años más tarde, John W. Mauchly y John Presper, construirían la primera computadora electrónica a petición del ministerio de defensa de Estados Unidos. Su capacidad de procesamiento era tan avanzada para su época, que fue utilizado en la carrera armamentística de USA, calculando la trayectoria de proyectiles lanzados por los mismos. Era universal y podía ser usada para cualquier tipo de cálculos (Cedano Olvera, Cedano Rodriguez, Rubio Gonzalez, & Vega Gutierrez, 2014).

Si bien se han abordado asombrosas máquinas para resolver problemas matemáticos, aun no estaban cerca de su prototipo final. Las primeras computadoras "modernas" se remontan a los años 50s y desde entonces se ha abierto paso con una gran popularidad en la sociedad actual volviéndose un arma de trabajo indispensable. Con una considerable ventaja en el manejo de información y cálculos matemáticos, rápidamente desplazaron a los humanos en el desarrollo de ciertas tareas empíricas aumentando la productividad en muchas industrias. (Cedano Olvera, Cedano Rodriguez, Rubio Gonzalez, & Vega Gutierrez, 2014)

Hoy en día, es posible realizar diferentes tareas al mismo tiempo debito a la alta tecnología de los microchips empleados que permiten cierto grado de paralelismo. Se conoce que la velocidad de procesamiento de una computadora actual es de $1\,x\,10^{-9}$ nanosegundos, siendo capaces de realizar complejos cálculos matemáticos en fracciones de segundos. (Cedano Olvera, Cedano Rodriguez, Rubio Gonzalez, & Vega Gutierrez, 2014)

En este punto, se ha podido estudiar los avances más importantes que llevaron a la informática moderna. Actualmente, la ciencia computacional ha avanzado de una manera impresionante, reduciendo exponencialmente su tamaño y proporcionalmente aumentando su capacidad de procesamiento. En 1965, el co-fundador de Intel Gordon Moore afirmo que el número de transistores por centímetro cuadrado en un circuito integrado se duplicaba cada año, aunque posteriormente modifico su afirmación indicando que se duplicaría cada 18 meses y esta tendencia continuaría durante las próximas dos décadas (Cheang, 2005). Para cumplir con la Ley de Moore, los transistores deben ser cada vez más pequeños, tema que será de particular interés posteriormente en nuestra investigación.

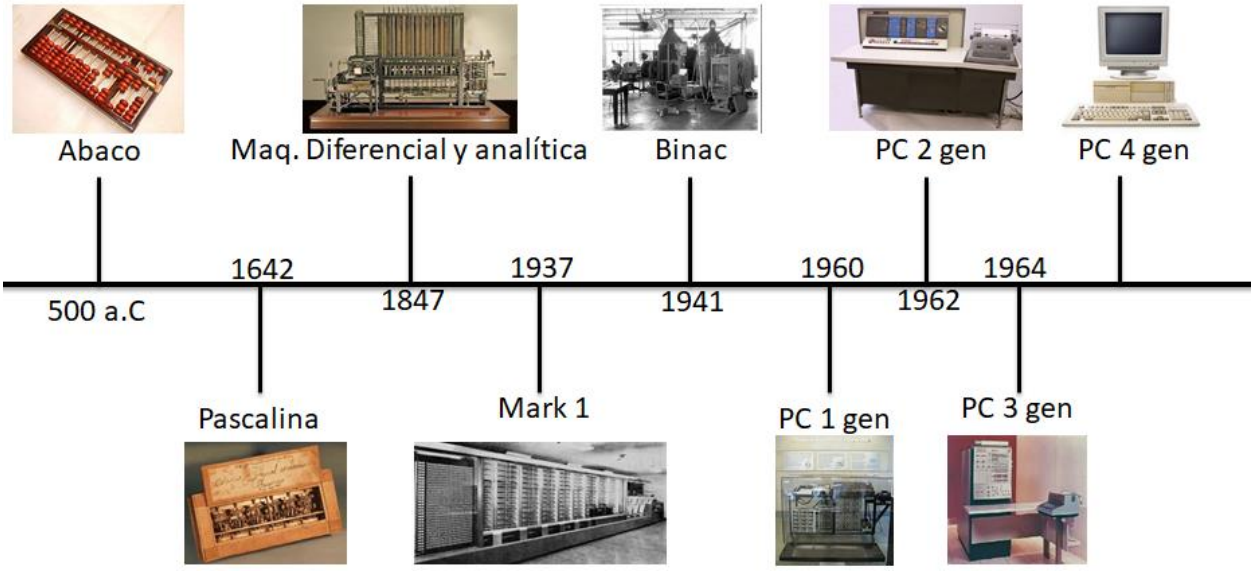

Ilustración 1. Evolución de la computación clásica

Fuente: Propia

3.2 Sistemas de numeración computacionales

Aunque existen un gran número de sistemas de numeración, el presente documento se centrara exclusivamente en los sistemas de numeración binarios. Según Cedano Olvera, Cedano Rodríguez, Rubio González, & Vega Gutiérrez (2014) el sistema de numeración binario es un "Sistema de numeración de base 2, el cual se representa por los números 0 y 1. Los números binarios son el sistema común interno de la computación digital debido a la relativa simplicidad de registrar, almacenar y reconocer variables de solo 2 valores." (p.31)

Existe un mito común que dicta que es imposible usar un sistema de numeración diferente al binario para un sistema computacional. Si bien hasta cierto punto es cierto, no siempre fue así. Según Mano, Lógica Digital y Diseño de Computadores (1989) “el diseñador de sistemas digitales está restringido al uso de señales binarias debido a la baja confiabilidad de los circuitos electrónicos de muchos valores” (p.2). Esto indica que es posible aplicar un valor determinado a cada variación de voltaje/corriente, sin embargo el sistema se vuelve ineficiente y poco confiable. Es por esta razón que el sistema binario se ha vuelto el lenguaje exclusivo para los procesamientos de computadoras (Mano, Logica Digital y Diseño de Computadores, 1989).

Ahora que se ha definido el sistema de numeración a utilizar para un sistema computacional, es necesario definir: ¿Cómo se utilizara este sistema para comunicarse con la maquina? La respuesta a esta gran interrogante son los bits. Según Cedano Olvera, Cedano Rodríguez, Rubio González, & Vega Gutiérrez (2014) un bit “es la mínima pulsación electrónica que lee nuestro computador” (p.72). Es decir la mínima cantidad de datos que se pueden transmitir, y los mismos son transmitidos en una numeración binaria, teniendo como opción únicamente un 1 y 0. Se podría simplificar esta definición a dos exclusivos estados, el estado 1 – encendido y el estado 0 – apagado. Todo esto es transmisible por una simple señal electrónica en forma de voltaje, en donde un 0 es la ausencia total del mismo (Cedano Olvera, Cedano Rodriguez, Rubio Gonzalez, & Vega Gutierrez, 2014).

Es imposible resolver un problema complejo mediante un solo bit, como ser una función logarítmica o visualizar una fotografía en una computadora. Para esto se utilizan patrones de bits, que son una cadena de hasta 16 bits agrupados en un orden específico. La memoria de una computadora lee exclusivamente patrones de bits, y ya dependerá de los periféricos de entrada y salida la representación de los mismos como números, textos o fotografías (Cedano Olvera, Cedano Rodriguez, Rubio Gonzalez, & Vega Gutierrez, 2014).

3.3 Lógica binaria

La lógica binaria se encarga de variables con dos valores discretos y que en conjunto tienen un significado lógico. Estas operaciones se llevan a cabo a un nivel de bits, en donde 2 o más bits serán entrada y se poseerá una sola salida. Dichas operaciones obedecen los principios del algebra booleana. Se podría definir al algebra booleana como un grupo de elementos y axiomas

para resolver un determinado problema, bastante similar al algebra convencional. (Mano, Diseño Digital, 2003).

3.3.1 Lógica binaria positiva y negativa

Por definición, un sistema binario solo posee dos posibles estados un 1 y 0. A simple vista se podría pensar que si se tuviera que definir estos valores en estados de "Alto" o "Bajo", probablemente se escogería al 1 como el estado más alto. Esto solo es verdad si se usa la lógica binaria positiva. La lógica positiva determina que si se tienen dos valores, el valor alto siempre será el más positivo, mientras que la lógica negativa implica que el valor alto será el valor más negativo (Maloney, 1983). En el presente documento se utilizara una lógica positiva, por lo que un valor alto será considerado como un 1 y un valor bajo como un 0.

3.3.2 Compuertas lógicas

Mano, Diseño Digital (2003) afirma que "Las compuertas lógicas son circuitos electrónicos que operan con una o más señales de entrada para producir una señal de salida" (p.28). Como se había indicado anteriormente, estas señales de entrada son magnitudes físicas comúnmente en formas de voltajes o corrientes. Es importante resaltar que estas magnitudes deben estar en un estándar permisible. Toda información deseada ya sea informativa o de control manipulada por una computadora, se puede manejar haciendo uso de las compuertas lógicas. (Mano, Diseño Digital, 2003) Las compuertas lógicas más utilizadas son:

Compuerta AND: Esta compuerta realiza una operación booleana de producto lógico. Se puede definir que la formula general de esta compuerta es:

$$F = A \times B$$

Ecuación 1. Compuerta AND

Donde A y B son dos entradas binarias. Partiendo de esta fórmula general, es posible deducir que para obtener una salida binaria de "1", se deben tener dos entradas binarias de "1". Caso contrario, el resultado seria "0". (Mano, Diseño Digital, 2003) Se pueden comprobar estas deducciones con la tabla de verdad para esta compuerta lógica.

Tabla 1. Tabla de verdad compuerta AND

Entradas		Salida
A	B	F
0	0	0
0	1	0
1	0	0
1	1	1

Fuente: Propia

Compuerta OR: La compuerta OR es una operación booleana aditiva. Esta compuerta está definida por la ecuación:

$$F = A + B + \cdots$$

Ecuación 2. Compuerta OR

Donde A, B Y C son los bits de entradas y F es el único bit de salida. Es importante resaltar que al igual que en la compuerta anterior se pueden tener más de 2 bits de entradas, sin embargo sin importar el número de entradas siempre se tendrá una sola salida. Partiendo de la ecuación anterior, es posible definir que si un bit de entradas es igual a "1", la salida siempre será "1". Si todas las entradas son "0", la salida será "0". (Mano, Diseño Digital, 2003) Esto es demostrable con la siguiente tabla de verdad de dos bits:

Tabla 2. Tabla de verdad OR

Entradas		Salida
A	B	F
0	0	0
0	1	1
1	0	1
1	1	1

Fuente: Propia

Compuerta NOT: Esta operación se basa en el principio de negación, por consiguiente su fórmula específica es:

$$F = \overline{A}$$

Ecuación 3. Compuerta NOT

Esta operación comúnmente se llama operación de complemento, puesto que toma la entrada y la cambia a su valor contrario. Si se tiene una entrada A = "1", usando la operación de complemento se obtendría una salida de "0". Es importante resaltar que esta operación se lleva a cabo a nivel de un solo bit. (Mano, Diseño Digital, 2003) La tabla de verdad para esta compuerta es la siguiente:

Tabla 3. Tabla de verdad NOT

Entradas	Salida
A	F
0	1
1	0

Fuente: Propia

Compuerta NAND: La compuerta NAND, al igual que la compuerta AND es una compuerta de producto lógico. La variación con respecto a la compuerta AND, es que la salida será un nivel bajo cuando todas sus entradas estén en un nivel alto. La ecuación para describir dicha compuerta es:

$$F = \overline{(A \times B)}$$

Ecuación 4. Compuerta NAND

Es posible simplificar la definición de esta compuerta, si se utiliza una compuerta AND seguida de un NOT. Es decir, se niega el resultado de la compuerta AND. (Maloney, 1983) Es posible comprobar esta deducción mediante la siguiente tabla de verdad:

Tabla 4. Tabla de verdad NAND

Entradas		Salida
A	B	F
0	0	1
0	1	1

1	0	1
1	1	0

Fuente: Propia

Compuerta NOR: Al igual que la compuerta NAND, es posible modelar esta compuerta negando la salida de una compuerta OR. La ecuación característica para esta función es:

$$F = \overline{(A + B)}$$

Ecuación 5. Compuerta NOR

Si se definen las entradas y se calcula el resultado, es posible encontrar que la salida se encontrara en un nivel bajo si cualquiera de las entradas se encuentra en un nivel alto. Se obtendrá un nivel alto de salida exclusivamente cuando todas las entradas estén en un nivel bajo. (Maloney, 1983) Todo es demostrable en la siguiente tabla de verdad:

Tabla 5. Tabla de verdad NOR

Entradas		Salida
A	B	F
0	0	1
0	1	0
1	0	0
1	1	0

Fuente: Propia

3.4 Algoritmos Clásicos

Hasta ahora se tiene una idea clara de los principios de la computación, la manera de transmitir y computar lógicamente los bits. Pero surge una interrogante, ¿Qué hacer con toda esa información recabada? La respuesta a esta pregunta sería: Los algoritmos. Los métodos que se aprendieron en la escuela como sumar, multiplicar y dividir son considerados algoritmos. Brassard & Bratley (1997) afirma que “la ejecución de un algoritmo no debe implicar, normalmente, ninguna decisión subjetiva, ni tampoco debe hacer preciso el uso de la intuición ni de la creatividad” (p.2).

Los algoritmos son una serie de operaciones definidas para resolver un problema, y bastara con seguir las instrucciones para poder utilizarlos. Podrían ser considerados como una receta, en la cual tenemos los ingredientes y la cantidad exacta que se utilizara (Brassard & Bratley, 1997). La presente sección se centrara específicamente en el algoritmo de la transformada rápida de Fourier.

3.4.1 Transformada rápida de Fourier

Antes de definir el algoritmo de la transformada rápida de Fourier, se discutirá un poco en la vida de su creador, Jean Baptiste Joseph Fourier (1768-1830). Fourier nació el 21 de Marzo de 1768 en la ciudad de Auxerre, Francia. Siempre tuvo un don natural para las matemáticas y en 1798 serviría al ejército Francés bajo el mando de Napoleón Bonaparte como oficial de artillería (calculaba trayectorias matemáticamente). Fungió como secretario del instituto de Matemáticas del el Cairo y en 1801 abandono su carrera militar y se dedicó de lleno a la ciencia. La carrera matemática de Fourier comenzó cuando afirmaba que la transferencia de calor era modelable por una onda sinusoidal que se atenuaba gradualmente. Tuvo numerosas críticas de aclamados científicos como LaPlace y Poisson. En 1822 completaría su investigación sobre el flujo del calor y lo publicaría en un libro llamado “Teoría analítica del Calor”, que serviría de inspiración a Ohm en sus investigaciones. Posteriormente, Fourier propondría lo que hoy se conoce como el Teorema de Fourier, que afirma que toda oscilación periódica sin importar su complejidad se repite exactamente una y otra vez, pudiendo descomponer la misma en movimientos ondulatorios simples y regulares, que sumados dan la onda compleja original. El teorema de Fourier tiene una gran variedad de aplicaciones, utilizado desde sistemas de sonidos, luz y básicamente cualquier fenómeno ondulatorio (De Vito & Suarez, 1999).

La transformada rápida de Fourier, o “FFT” por sus siglas en ingles ha revolucionado la eficiencia para cálculos de polinomios. La transformada de Fourier ha sido utilizada para obtener muestras de una señal análoga. Es sabido que estos sistemas se digitalizan mediante muestras en el tiempo siguiendo los lineamientos propuestos por Nyquist, en donde afirmaba que la frecuencia de muestreo debía ser el doble de la muestra. La FFT logra acelerar este proceso por el hecho que los cálculos de coeficientes de la transformada de Fourier pueden ser realizados iterativamente, lo que resulta en una disminución de tiempo de procesamiento (Cochran, et al., 1967). Antes de entrar de lleno en la definición de la FFT, es necesario sentar algunas bases de

algoritmos de multiplicación de polinomios. Un dato importante para la multiplicación de los mismos es que cualquier polinomio de grado d es caracterizado de manera única por sus valores en cualquier punto en la ecuación $d + 1$. (Dasgupta, Papadimitiou, & Vazirani, 2006)

Podríamos asociar esto con la idea de que solo se ocupan 2 puntos definidos para trazar una línea, la multiplicación de polinomios sigue un principio similar. Se supondrá que se tienen los siguientes polinomios: $A(x) \times B(x)$; ambos polinomios son de grado d. Para resolver este problema por medio de un algoritmo, primero será necesario escoger puntos para $x_0, x_1, \ldots\ldots, x_{n-1}$ donde $n \geq 2d + 1$. Posteriormente se evalúa $A(x_0), A(x_1), \ldots, A(x_{n-1})$ y $B(x_0), B(x_1), \ldots, B(x_{n-1})$. Se procede a realizar el producto de ambos polinomios y se interpola de la forma $C(x) = c_0 + c_1 x + \cdots + c_{2d} x^{2d}$ (Dasgupta, Papadimitiou, & Vazirani, 2006). Esta solución de polinomio tiene un tiempo de respuesta de

$$t = O(n^2) ,$$

Ecuación 6. Tiempo de respuesta factorización polinomica convencional.

Mientras que usando la transformada rápida de Fourier se podría resolver el mismo problema en solo:

$$t = O(n \log n).$$

Ecuación 7. Tiempo de respuesta factorización transformada rápida de Fourier

La transformada de Fourier toma como entrada un vector de la forma $\alpha = (\alpha_0, \ldots , \alpha_{n-1})$ y un número complejo en el cual sus exponentes son n raíces complejas de unidad. Multiplica el vector α por la matriz $n \times n$ (también denominada $M_n(w)$) que tiene una entrada (j, k) de la forma w^{jk}. La matriz $M_n(w)$ es posteriormente separada en términos pares e impares, y podemos obtener la siguiente matriz simplificada:

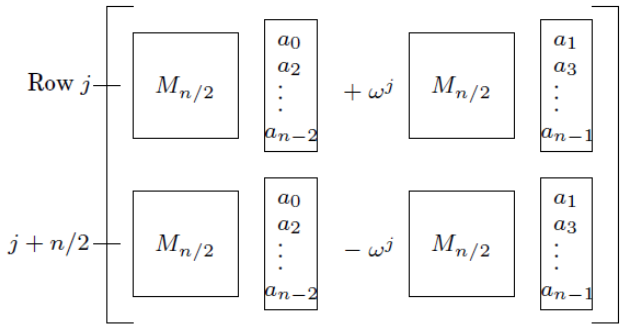

Ilustración 2. Forma Matricial de la Transformada rápida de Fourier

Fuente: (Dasgupta, Papadimitiou, & Vazirani, 2006)

Esta matriz reduce el tiempo de procesamiento a solo $O(n \log n)$, puesto que simplifica las operaciones realizadas. El circuito para la transformada rápida de Fourier es:

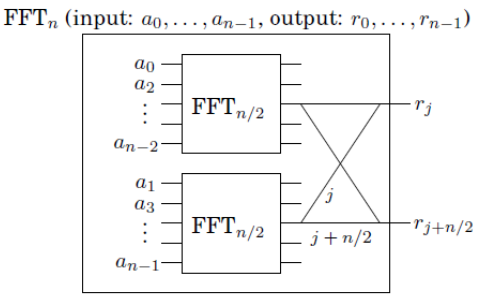

Ilustración 3. Circuito equivalente de la Transformada rápida de Fourier

Fuente: (Dasgupta, Papadimitiou, & Vazirani, 2006)

3.5 Computación Molecular

Se ha definido completamente los principios de funcionamiento de la computación clásica, sin embargo además de la computación clásica y cuántica que tienen un renombre pronunciado, es posible encontrar a la computación molecular. La computación molecular se basa en el principio físico que dicta que biopolímeros como el ácido nucleico codifican información que pueden ser vistos como letras químicas. A esta tecnología también se le conoce como computación biomolecular, o de DNA. Un claro ejemplo de una computadora molecular es el cerebro humano,

mediante la interacción química de neurotransmisores, es capaz de controlar con gran precisión el funcionamiento del cuerpo humano. Se ha definido que un autómata molecular, una maquina basada en computación molecular, podría ser el futuro de la exploración espacial. Un autómata es capaz de adaptarse a un entorno mediante la recopilación de información, y usar esta para reproducirse y adaptarse. Un virus informático que se adapta a un computador y se oculta y reproduce para no ser detectado es un ejemplo de un sistema autómata. Si bien la computación molecular presenta una gran escalabilidad, no será de mayor interés para el presente documento (Benenson & Shapiro, 2004).

3.6 Computación Cuántica

Alrededor de los 60s, el físico alemán Rolf Landauer se interesó en las limitaciones que poseía la computación clásica, y en el origen del calor generado por los procesadores. El científico sugería que dicho calor era generado por la ineficiencia del procesador, siguiendo las leyes de la física clásica. Esta ineficiencia es inherente de todos los sistemas computacionales actuales, y aunque ha reducido no se ha podido eliminar. Con los avances actuales de la tecnología, los microchips son cada vez más pequeños y eficientes, alcanzando ya a la escala de unos pocos nanómetros. Al alcanzar tamaños tan pequeños, la física clásica se ve comprometida y los chips no funcionan como deberían a causa del efecto túnel, un fenómeno que solo es apreciable en un entorno cuántico. Para ejemplificar, si se posee una partícula que obedece la física clásica, al encontrar un obstáculo la misma rebotaría y sería incapaz de continuar su camino. Sin embargo, esta situación cambia radicalmente cuando se tiene solamente un electrón, ya que al ser su naturaleza ondulatoria, existe una pequeña posibilidad de que la misma logre cruzar la barrera debido al efecto túnel, causando que el chip no funcione correctamente. Esto supone una importante limitante y nos indica que la computación clásica está llegando cada vez más cerca de sus límites físicos. (Bonillo, 2013)

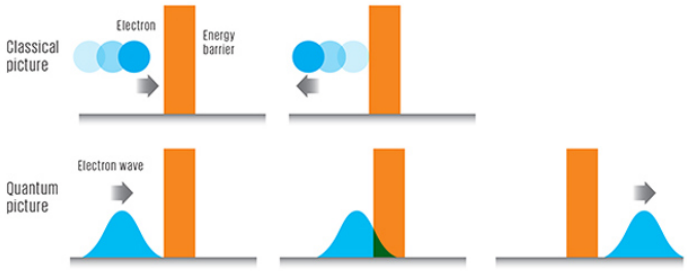

Ilustración 4. Efecto túnel cuántico

Fuente: (physicsopenlab.org, 2017)

Las primeras ideas de la computación cuántica se remontan a los años 80, en donde el físico estadounidense Paul Benniof trabajaba con máquinas de Turing**,** las que funcionaban con algunos principios fundamentales de la mecánica cuántica. En el año 1982 el físico estadounidense Richard Feynman afirmaba que el uso de los fenómenos cuánticos en la computación, permitiría realizar cálculos de gran complejidad de manera más rápida**.** En los años 90 la computación cuántica finalmente comenzaba a moverse de un campo teórico a uno práctico. Surgieron los primeros algoritmos cuánticos y las primeras máquinas capaces de resolverlos. Entre 1994 y 1995, el científico estadounidense Peter Shor definió un algoritmo cuántico que permitía encontrar los factores primos de cualquier número a una velocidad mucho mayor en relación a una computadora tradicional. De la misma manera, en 1996 el científico informático Lov Grover propone un algoritmo de búsqueda de datos, aunque la aceleración no era tan drástica como en la aplicación de Shor, su rango de aplicaciones es mucho más grande y posee un alto índice de acierto. (Bonillo, 2013)

Entre 1997 y 1999 surgieron importantes avances en la computación cuántica. En este periodo se logró establecer una comunicación encriptada usando tecnología cuántica a una distancia de 23km. Asimismo, se logra la propagación de un bit cuántico, denominado qubit, en una disolución de aminoácidos. Esta sería la base para el estudio de transporte de información mediante qubits. En 1998 aparecería la primera máquina de 2 qubits en California. Posteriormente, IBM diseñaría una máquina de 3 qubits, que ejecutaría por primera vez el algoritmo de búsqueda de Grover. En los años 2000, se fabricaría la primera máquina de 5 qubits, nuevamente por IBM. Esta máquina era capaz de ejecutar el algoritmo de Shor en una

simple iteración, mientras que una computadora clásica requería muchas más iteraciones para ejecutar el mismo algoritmo. (Bonillo, 2013)

Esta sería la primera prueba contundente de la gran velocidad de las maquinas cuánticas. Durante los siguientes años continuaría el avance progresivo de computadoras cuánticas capaces de procesar más qubits. En el año 2009 investigadores de la Universidad de Yale, crearon el primer procesador cuántico de estado sólido. Este procesador se asemejaba bastante a un procesador convencional, aunque con la capacidad de realizar tareas muy simples como operaciones aritméticas o búsquedas de datos. Finalmente en el año 2011, la primera computadora cuántica comercial es vendida por 10 millones de dólares, e IBM anuncia la creación de un chip lo suficientemente estable como para ser comercializado en los próximos años. (Bonillo, 2013)

3.7 Principios Cuánticos

Ahora que se ha definido la historia de la computación cuántica, se dará un salto hacia los principios de funcionamiento. La computación cuántica explora fenómenos que solo son apreciables en la física cuántica para potenciar su capacidad de procesamiento. Se comenzara definiendo a lo que se le denomina un estado cuántico. Es posible definir al estado cuántico como un descripción completa para modelar un sistema (Gruska, 2000). Si se toma en consideración cualquier sistema atómico, compuesto de partículas interactuando entre sí a través de las leyes de la física. Estas partículas tendrán determinadas cantidades de movimiento por las interacciones entre sí, cada uno de estos movimientos seria el estado cuántico del sistema. Este estado cuántico es imposible de modelar usando valores numéricos como comúnmente hacemos con estados clásicos (Dirac, 1930).

3.7.1 Qubits

En concordancia a su homologo clásico, la computación cuántica se rige bajo alguna de las mismas reglas como ser la transmisión de señales eléctricas o físicas que posteriormente son decodificadas en un formato comprensible para el ordenador. Se había establecido anteriormente que en computación clásica, cada una de estas señales eran denominados “Bits” (Bonillo, 2013). Este es un método de transmisión de datos, pero se podría transmitir información de una manera distinta, como ser el estado de un átomo. Tal y como se haría con un sistema clásico, para el procesamiento clásico se elige un sistema de dos niveles. Esto hace favorable el uso de partículas

de spin medio, tal y como lo son los electrones (Miranda, 2014). Se asumirá que el estado estable de este electrón es un "0" y el estado excitado es un "1", posteriormente se definirá la notación de este valor, siendo el estado estable de un electrón como $|0\rangle$ y el estado excitado como $|1\rangle$, a esta notación se le conoce como notación de Dirac. Luego de definir el sistema de notación, se puede afirmar que se posee un sistema de transmisión bastante similar a la de un sistema cuántico (Dasgupta, Papadimitiou, & Vazirani, 2006). El término utilizado para definir a un bit cuántico es "Qubit", y seria usado por primera vez por el físico teórico Benjamin Schumacher a finales del siglo XX cuando logro interpretar los estados cuánticos de la información, y comprimir la misma en estados más pequeños (Bonillo, 2013).

La palabra "Qubit" proviene del termino en ingles "Quantum Bit". Bonillo (2013) afirma que un qubit "es un sistema con dos estados propios y que puede ser manipulado arbitrariamente" (p. 15). Puesto de otra manera un qubit es un vector en un espacio complejo vectorial bidimensional (Nielsen & Chuan, 2010). Ahora que se comprende la mínima unidad de medida de la computación cuántica, es importante adentrarse más en sus propiedades específicas en donde yace la verdadera capacidad de procesamiento de estos sistemas.

3.7.2 Superposición

Ahora que se ha comprendido la definición de un estado cuántico y los qubits que lo conforman, es necesario centrarse en las peculiaridades de los mismos. El principio general de la superposición cuántica es aplicable a los estados de cualquier sistema determinado. La idea de la superposición, obliga a considerar que entre los posibles estados de una partícula existe una relación, tal que cuando un electrón se encuentra en un estado, también podría considerarse parcialmente en otro estado (Dirac, 1930). Posiblemente, en alguna ocasión se ha escuchado acerca de la paradoja de Schrödinger. Este relato no es más que una forma sencilla de explicar el principio de superposición.

Latorre (2017) afirma que el relato de Schrödinger consistía en lo siguiente:

Un gato es encerrado en una caja donde hay una botella con veneno —para ser precisos, Schrödinger escogió el ácido cianhídrico—. En la caja hay un átomo radioactivo. Si este átomo decae y emite radiación, esa misma radiación dispara un contador Geiger. Este, a su vez, dispara

un mecanismo con un martillo que cae sobre la botella, la rompe, esparce el veneno y mata al gato. Si el átomo no decae, todo sigue igual y el gato deambula felizmente por la caja. (p.32)

Al estar en una caja completamente cerrada, a simple vista no es posible saber el estado del gato puesto que no es observable. Se podría modelar matemáticamente esta situación con la ecuación: Estado del gato = Gato vivo + Gato muerto. Esto es básicamente una ecuación de un sistema cuántico en superposición. Latorre (2017) afirma que "el gato no está vivo y muerto a la vez, como a veces se lee en textos divulgativos. Lo que describimos es la información sobre el gato" (p.33). Si bien el gato jamás se comportaría como un electrón en un entorno cuántico, la paradoja de Schrödinger sirvió como base para un concepto de superposición.

Uno de los objetivos principales de la mecánica cuántica es predecir si un evento sucederá o no, y si lo hará ¿Cuál es la probabilidad de que suceda? La probabilidad de que un evento suceda, será dada por la ecuación:

$$p = |\alpha|^2$$

Ecuación 8. Probabilidad de un evento cuántico

Donde α es un número complejo denominado amplitud de un evento (Gruska, 2000).

Si se considera el caso específico de un electrón. El principio de superposición cuántico indica que un electrón puede estar en un estado estable de energía, en un estado excitado, o en una superposición linear de ambos estados. Por ejemplo, el estado de un electrón podría ser $\frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle$, o podría tener una infinita cantidad de combinaciones que obedezcan a la forma:

$$\alpha_0|0\rangle + \alpha_1|1\rangle$$

Ecuación 9. Estado de superposición cuántica.

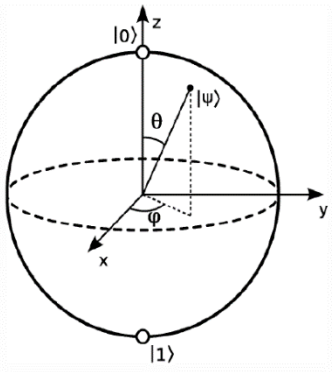

Ilustración 5. Esfera de Bloch

Fuente: (Bonillo, 2013)

Este vector de posición de un electrón, puede ser visualizado a través de la esfera de bloch. Esta fórmula general de estado, puede contener combinaciones de números complejos siempre y cuando se rijan bajo el principio matemático que afirma que:

$$|\alpha_0|^2 + |\alpha_1|^2 = 1$$

Ecuación 10. Magnitud de un vector cuántico

Visto de esta forma, podemos se puede definir que el estado $\frac{1}{\sqrt{5}}|0\rangle + \frac{2i}{\sqrt{5}}|1\rangle$ aun teniendo números complejos es un estado cuantico perfectamente valido, puesto que la magnitud del vector da como resultado 1 (Dasgupta, Papadimitiou, & Vazirani, 2006).

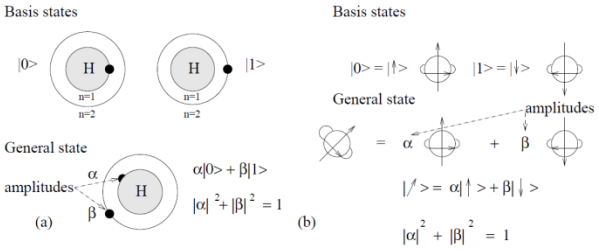

Ilustración 6. Estados de un Qubit

Fuente: (Gruska, 2000)

Se ha estudiado la complejidad detrás de la superposición de un qubit, pero ¿Qué sucedería si agregáramos un qubit adicional? Si agregáramos un qubit extra tendríamos un estado en superposición mucho más grande. Consideremos un sistema de 2 bits clásicos, los posibles resultados de este sistema serian 00, 01, 10 y 11. En un sistema de 2 qubits tenemos un sistema en superposición que obedece a la forma:

$$\alpha_{00}|00\rangle + \alpha_{01}|01\rangle + \alpha_{10}|10\rangle + \alpha_{11}|11\rangle$$

Ecuación 11. Estado de superposición de un sistema de 2 Qubits

3.7.3 Entrelazamiento Cuántico

Si se pensara en la palabra "reacción", posiblemente una de las primeras ideas en surgir sería un compuesto químico o incluso la tercera ley de Newton que estipula que toda acción tiene una reacción de igual magnitud en dirección contraria. Si bien ambas ideas son acertadas, todas tienen algo en común, dichas reacciones se llevan a cabo en superficies o sustancias en contacto entre sí. Entonces surge una gran interrogante, ¿Puede una acción desencadenar una reacción en un objeto lejano? La respuesta dependerá del entorno.

En 1840 el científico alemán Julius Robert Mayer dedujo que la energía se conservaba, años más tarde, esta teoría seria comprobada experimentalmente por Joule, Kelvin y Carnot. Basados en estas deducciones, se formuló la ley de conservación de la energía que afirma que la energía no se crea ni se destruye, solo se transforma. En todo proceso físico, la sumatoria total del momento de entrada es igual al de salida, y como ya es conocido, este principio es regido bajo la ley de la conservación de la energía. Cuando se aplica este principio a un entorno cuántico el resultado trasciende las deducciones de conservación de energía clásicos. En mecánica cuántica, cuando dos partículas interactúan en el mismo punto y son separadas, permanecerán enlazadas entre sí sin importar la distancia, es decir si algo le sucede a una, repercutirá directamente en la otra no importa que tan lejos este. (Aczel, 2004)

Para comprender un poco más acerca del entrelazamiento, se debe indicar la intrínseca relación entre el entrelazamiento y la superposición. Según Aczel (2004) "El entrelazamiento es una aplicación del principio de superposición a un sistema compuesto consistente en dos o más subsistemas" (p.41). En 1935 mediante un conjunto de experimentos llevados a cabo por Einstein, Podolski y Rosen formularon la paradoja EPR. Esta paradoja afirmaba que si se tienen

dos cantidades físicas denominadas A y B, y ambas eran no conmutables, es decir $AB \neq BA$, estas estaban en un estado entrelazado, y cualquier esfuerzo de determinar alguna de las dos influiría irreparablemente en el estado de la otra. Estos experimentos concluyeron en que la descripción de la realidad de la mecánica cuántica no estaba completa. Y que debía haber variables ocultas en la física cuántica pues era imposible predecir su comportamiento sin disturbarlo mediante una medición. (Einstein, Podolsky, & Rosen, 1935)

El entrelazamiento cuántico es responsable de la gran potencia de los métodos de cálculo en la computación cuántica, es aprovechado por las operaciones de factorización y búsqueda de datos (Miranda, 2014). Para comprender de una manera más sencilla lo que es el entrelazamiento cuántico, se pueden considerar a dos átomos de Hidrogeno que interactúan entre si y luego son separados. Estos átomos se consideran entrelazados y con obtener información de uno se puede conocer el estado del otro. Si se midieran uno de estos electrones y se observara que su espín es para la izquierda, se sabrá inmediatamente que el electrón entrelazado estará girando en sentido contrario, es decir a la derecha.

3.7.4 Reversibilidad

Al igual que las compuertas lógicas clásicas, las compuertas cuánticas llevan a cabo operaciones lógicas a nivel de qubits. La diferencia más importante entre una compuerta lógica clásica y una cuántica es la reversibilidad o conservación de información. En los procesos de información cuántica, todas las operaciones son unitarias, esto significa que los grados de libertad de entrada serán igual a los de salida. En otras palabras, las operaciones unitarias garantizan que no hay perdida de información de la superposición inicial (Scheel, Pachos, Hinds., & Knight, 2004).

La reversibilidad es uno de los pilares de la computación cuántica. Se considera que una operación es reversible cuando partiendo de su salida es posible recuperar la entrada, como ser en una compuerta NOT. Para comprender un poco más acerca de la reversibilidad se analizara el resultado de la compuerta clásica NAND. En dicha compuerta se tienen dos entradas y 1 salida, esto significa que una de las entradas ha sido perdida irreversiblemente en el proceso. El cambio en la entropía por la pérdida de 1 bit es $\ln 2$, termodinámicamente esto podría ser expresado como:

$$kT \ln 2$$

Ecuación 12. Ecuación de Landauer

Donde k es la constante de Bozeman y equivale a $1.38\,x\,10^{-23}$ J/K, y T es la temperatura. A esta ecuación se le denomina ecuación de Landauer y equivale a una pérdida de energía. El calor disipado durante un proceso es tomado como un signo de irreversibilidad (Muthukrishnan, 1999).

3.7.5 Compuertas lógicas cuánticas

Las compuertas lógicas cuánticas funcionan bajo los mismos principios de funcionamiento de las compuertas clásicas. Como se mencionaba anteriormente, la única diferencia entre ellas es la reversibilidad. Esta sección se centrara en un grupo específico de puertas que son consideradas universales, es decir, partiendo de estas puertas es posible representar todas las demás puertas existentes.

3.6.5.1 Compuerta NOT

Anteriormente se habían definido las compuertas de la computación clásica, como se pudo observar se tenían compuertas que solo requerían de un bit de entrada, como ser la compuerta NOT. ¿Es posible tener una compuerta NOT en un entorno cuántico? Si se analizara esta situación con lo que se sabe de la compuerta clásica, una compuerta NOT cuántica tomaría un bit de entrada de forma $|0\rangle$ y lo cambiaria a un $|1\rangle$. Este proceso dejaría ignoraría completamente los principios de superposición que caracterizan a los qubits, aunque la definición no es tan alejada de la realidad. Una compuerta NOT cuántica invierte la amplitud de los vectores de superposición de la siguiente manera (Nielsen & Chuan, 2010):

$$\alpha|0\rangle + \beta|1\rangle \rightarrow \alpha|1\rangle + \beta|0\rangle$$

Ecuación 13. Compuerta NOT Cuántica

Increíblemente, aun con todas las propiedades de la mecánica cuántica una compuerta NOT se comporta linealmente. Existe una forma conveniente de representar una compuerta NOT cuántica de forma matricial de la siguiente manera (Nielsen & Chuan, 2010):

$$X = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$

Ecuación 14. Compuerta NOT cuántica en forma matricial

Partiendo de esta matriz, si tuviéramos un estado cuántico de la forma $\alpha|0\rangle + \beta|1\rangle$ la matriz de la compuerta NOT se comportaría en la forma (Nielsen & Chuan, 2010):

$$X \begin{bmatrix} \alpha \\ \beta \end{bmatrix} = \begin{bmatrix} \beta \\ \alpha \end{bmatrix}$$

Ecuación 15. Cambio de amplitud de una compuerta NOT cuántica

La compuerta NOT cuántica aparenta cierta simplicidad, pero ¿Existe algún limite en cuanto a las matrices y valores de $\alpha\ y\ \beta$? Al igual que: $\alpha^2 + \beta^2 = 1$ las amplitudes resultantes de la compuerta NOT deben obedecer el mismo principio, de manera que $\alpha'^2 + \beta'^2 = 1$ (Nielsen & Chuan, 2010).

3.6.5.2 Compuerta Z

A diferencia de la computación clásica, en un sistema cuántico la compuerta NOT no es la única que trabaja a nivel de un solo qubit. Es posible encontrar la compuerta Z que deja a la amplitud de $|0\rangle$ intacta y cambia el signo de $|1\rangle$ a $-|1\rangle$ (Nielsen & Chuan, 2010). Adicionalmente, al igual que la compuerta Z que es solo una rotación de un eje, existen otras compuertas encargadas de rotar en los ejes restantes. Estos cálculos se hacen en base a los ángulos de los vectores resultantes apreciables en la esfera de Bloch (Gruska, 2000).

$$R_x(\theta) = \begin{pmatrix} \cos\theta & i\sin\theta \\ i\sin\theta & \cos\theta \end{pmatrix}, \quad R_y(\theta) = \begin{pmatrix} i\cos\theta & \sin\theta \\ \sin\theta & i\cos\theta \end{pmatrix}, \quad R_z(\theta) = \begin{pmatrix} e^{i\theta} & 0 \\ 0 & e^{-i\theta} \end{pmatrix},$$

Ilustración 7. Compuertas de rotación axial.

Fuente: (Gruska, 2000)

3.6.5.3 Compuerta Hadamard

La compuerta Hadamard al igual que las dos anteriores, trabaja a nivel de un solo qubit. Usualmente es conocida como la raíz cuadrada de la compuerta NOT. La compuerta Hadamard cambia el vector de entrada a:

$$\frac{|0\rangle + |1\rangle}{\sqrt{2}}|0\rangle \ o \ \frac{|0\rangle - |1\rangle}{\sqrt{1}}|1\rangle$$

Ecuación 16. Compuerta Hadamard

Como es observable, la compuerta Hadamard cambia el $|0\rangle$ a una posición "en medio", entre $|0\rangle$ y $|1\rangle$, y realiza la misma operación con el $|1\rangle$. En forma matricial esto seria apreciable en la forma:

$$H = \frac{1}{\sqrt{2}}\begin{bmatrix}1 & 1\\ 1 & -1\end{bmatrix}$$

Ecuación 17. Compuerta Hadamard en forma matricial

La compuerta Hadamard es una de las compuertas cuánticas más útiles, y se podría modelar el comportamiento de la misma mediante la esfera de Bloch. La compuerta Hadamard rota el eje $\hat{y}$ en 90° y el eje $\hat{x}$ en 180° (Nielsen & Chuan, 2010).

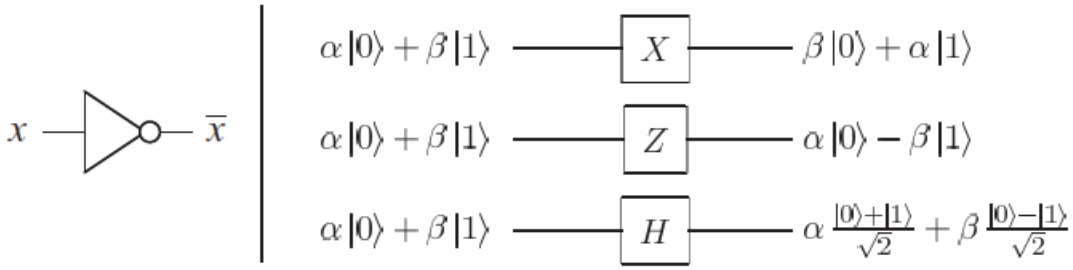

Figure 1.5. Single bit (left) and qubit (right) logic gates.

Ilustración 8. Compuertas de 1 Qubit

Fuente: (Nielsen & Chuan, 2010)

3.6.5.4 Compuerta CNOT

La compuerta CNOT, o Controlled NOT por sus siglas en ingles es una compuerta de 2 qubits; 1 de los qubits es considerado como qubit de control y el siguiente como qubit de proceso. La compuerta CNOT funciona invirtiendo el qubit de proceso si el de control es un 1, de otra

manera lo deja intacto. Puesto en notación de Dirac, el funcionamiento de esta compuerta es el siguiente (Nielsen & Chuan, 2010):

$$|00\rangle \rightarrow |00\rangle; |01\rangle \rightarrow |01\rangle; |10\rangle \rightarrow |11\rangle; |11\rangle \rightarrow |10\rangle;$$

Es posible describir el funcionamiento de una compuerta CNOT como una generalización de la compuerta clásica XOR, sin embargo a diferencia del resultado de un XOR, la compuerta CNOT es completamente reversible pues el qubit de control no se pierde, por lo que se puede saber cuál era el qubit de proceso viendo el resultado y el de control. (Bonillo, 2013)

También es posible modelar el CNOT mediante la siguiente matriz:

$$UCN = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

Ecuación 18. Forma matricial de una compuerta CNOT

3.7.6 Circuitos Cuánticos

Para comprender mejor acerca de los circuitos cuánticos y como entenderlos, es necesario comprender su nomenclatura y como están compuestos. Un circuito cuántico es una colección de puertas cuánticas conectadas aciclicamente por "cables cuánticos". El tamaño y profundidad del circuito dependerá del número de nodos en el esquema de conexión (Gruska, 2000).

Todos los esquemas de circuitos cuánticos se leen de izquierda a derecha, y los "cables" no son necesariamente elementos físicos, pueden ser algo tan complejo como una partícula de luz moviéndose de un lugar a otro del espacio.

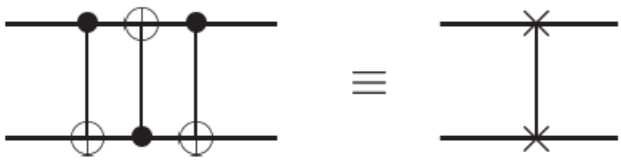

Ilustración 9. Intercambio de 2 qubits

Fuente: (Nielsen & Chuan, 2010)

En la figura podemos ver un circuito clásico para el cambio de valores de 2 qubits. Este circuito es usado tan comúnmente que tiene su propio símbolo de abreviación. La relación entre los circuitos clásicos y los cuánticos es bastante similar, son pocas las cosas que son permitidas en los clásicos y no en los cuánticos. La retroalimentación es una peculiaridad meramente clásica y no es permitida en un entorno cuántico, por tal razón a los circuitos cuánticos se les denomina "aciclicos". En una manera más simple, cualquier característica que posea irreversibilidad no podrá ser utilizado en circuitos cuánticos (Nielsen & Chuan, 2010).

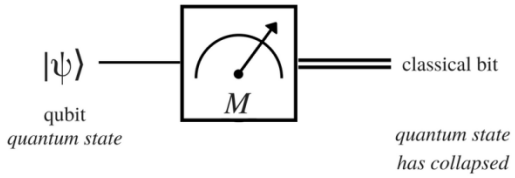

Ilustración 10. Compuerta de medición

Fuente: (Mishra, 2019)

La medición es una notación muy importante para comprender los circuitos cuánticos. Como se mencionaba anteriormente, cuando se tiene un qubit y se somete a una medición, el qubit colapsa a un estado de $|0\rangle$ o $|1\rangle$. Por tal razón se aprecia que el qubit sale de su proceso de medición como un bit clásico.

3.7.7 Algoritmos Cuánticos

Todos han experimentado la frustración de necesitar una computadora más potente para poder resolver algún problema computacional. De hecho, se puede afirmar que muchos problemas interesantes son imposibles de resolver mediante la computación clásica pues se necesitarían recursos astronómicos para poder llevar a cabo su procesamiento. Una de las promesas más grandes de la computación cuántica es la ejecución de algoritmos que aumentan sustancialmente la velocidad de procesamiento. Los 2 algoritmos clásicos más famosos son: El algoritmo de Shor basado en la transformada cuántica de Fourier, usado para factorizar y resolver logaritmos logrando un aumento exponencial en velocidad de procesamiento, y el algoritmo de Grover que lleva a cabo búsquedas cuánticas aunque no con una ventaja de velocidad tan considerable como el de Shor. De igual manera la transformada cuántica de Fourier tiene muchas aplicaciones

interesantes, como resolver logaritmos discretos y problemas de factorización. Esto conllevara al fracaso total de los criptosistemas de seguridad, puesto que podrían ser descifrados sin problema alguno por la transformada de Fourier (Nielsen & Chuan, 2010).

3.7.7.1 La transformada cuántica de Fourier

La transformada discreta de Fourier o DFT por sus siglas en ingles es una de las transformadas que posee un aumento exponencial en cuanto a velocidad de procesamiento en un entorno cuántico. Esta transformada toma una entrada de un vector compuesto por números complejos $x_0, \dots, x_{N-1}$ donde la longitud N del vector es un parámetro fijo, y lo transforma a un vector de números complejos $y_0, \dots, y_{N-1}$. La DFT puede ser definida como:

$$y_k = \frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} x_j e^{2\pi i j k / N}$$

Ecuación 19. Transformada Discreta de Fourier

Si bien esta es la representación clásica, la transformada cuántica de Fourier es igual solo cambiando la notación.

$$\sum_{j=0}^{N-1} x_j |j\rangle \rightarrow \sum_{k=0}^{N-1} y_k |k\rangle$$

Ecuación 20. Transformada cuántica de Fourier

Donde la amplitud y_k es la DFT de la amplitud x_j. A simple vista no resulta obvio pero esta es una operación unitaria, lo que permite su procesamiento en computadoras cuánticas (Nielsen & Chuan, 2010).

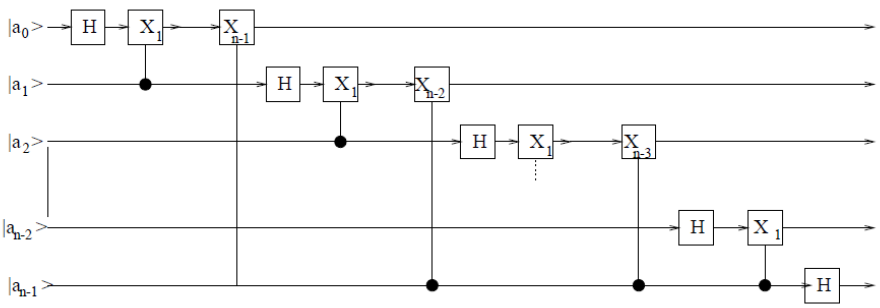

Ilustración 11. Circuito equivalente de la Transformada Cuántica de Fourier de 3 qubits

Fuente: (Gruska, 2000)

3.7.7.2 Estimación de Fase

La transformada de Fourier es la llave para poder llevar a cabo el algoritmo cuántico conocido como estimación de fase, el cual también es importante para la ejecución de otra variedad de algoritmos. Si se supone que un operador unitario tiene un auto vector $|u\rangle$ con un valor propio de $e^{2\pi i\varphi}$ y se desconoce el valor de φ. La función de la estimación de fase atraves de la transformada de Fourier es encontrar este valor que posteriormente será utilizado por otro algoritmo. La estimación de fase usa 2 registros, el primer registro contiene una cantidad de t qubits en el estado $|0\rangle$. La cantidad de qubits que se escojan dependerá del grado de precisión que se desee alcanzar. El segundo registro empezara en el estado $|u\rangle$ y constara de los qubits necesarios para definir este vector.

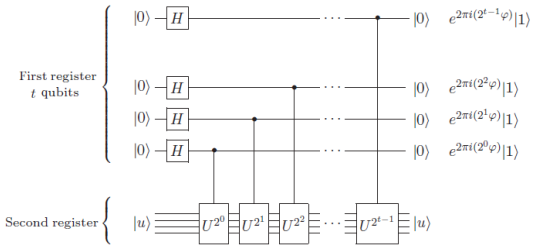

Ilustración 12. Circuito equivalente de estimación de fase

Fuente: (Nielsen & Chuan, 2010)

El circuito cuántico para estimación de fase comienza aplicando transformadas de hadamard al primer registro y posteriormente se aplican operaciones de U-controladas (CNOT) en el segundo registro. El último paso para la estimación, es aplicar la transformada inversa de Fourier. En pocas palabras, bastara con invertir el circuito cuántico de la transformada de Fourier. (Nielsen & Chuan, 2010)

3.7.7.3 Algoritmo de Deutsch

El algoritmo de Deutsch está diseñado para poder identificar una función oculta de la cual no es posible obtener información, y definir si la misma es balanceada o constante. Se analizara el caso de una moneda. Se desea identificar si la moneda es real y contiene cara y escudo en ambos

lados, o si es falsa y solo contiene uno de los dos. En la computación clásica se tendría que ver la moneda 2 veces para poder conocer la información completa de la misma (Gruska, 2000). El algoritmo de Deutsch combina el paralelismo y la interferencia cuántica para poder realizar este cómputo en solo 1 iteración. Este algoritmo será de particular interés posteriormente en la investigación.

3.7.7.4 Algoritmo de Grover

El algoritmo de Grover fue uno de los primeros algoritmos en probar una ventaja de velocidad en comparación con los algoritmos clásicos en 1995. Ha sido considerado como uno de las pruebas tangibles de la potencia de procesamiento cuántica. Este algoritmo se basa en principios cuánticos para realizar búsquedas en bases de datos desordenadas. Grover baso el desarrollo de este algoritmo en el uso de superposición y entrelazamiento para conseguir una ventaja sobre la computación clásica (Scully & Zubairy, 2001). Este algoritmo será estudiado con más profundidad posteriormente.

$$t = O(\sqrt{N})$$

Ecuación 21. Operaciones de búsqueda de un registro con el algoritmo de Grover

$$t = O(N)$$

Ecuación 22. Operaciones de búsqueda de un registro con un algoritmo clásico

3.7.7.5 Algoritmo de Shor

Como se mencionaba anteriormente, la factorización de números grandes ha sido el reto más difícil para la computación clásica, aumentando exponencialmente su dificultad con el aumento de la cantidad de dígitos a factorizar. Este aumento intratable de la dificultad al factorizar ha sido utilizado como el principio de funcionamiento para las criptologia. Al igual que el algoritmo de Grover, el algoritmo de Shor fue catalogado como la prueba más contundente de la velocidad impresionante de los algoritmos cuánticos. Este algoritmo utiliza la potencia de una computadora cuántica para encontrar el periodo de una función:

$$f(x) = a^x mod\ N$$

Ecuación 23. Periodo de una función

Este algoritmo está compuesto por dos secciones, la primera parte es la encargada de la aritmética modular descrita en la ecuación 24, mientras que la segunda está compuesta por la transformada inversa de Fourier. El proceso para encontrar los factores de un número de l digitos utilizando este algoritmo toma un tiempo de:

$$t = O(l^3)$$

Ecuación 24. Operaciones de procesamiento del algoritmo de Shor

$$t = O(2^{l^{\frac{1}{3}}})$$

Ecuación 25. Operaciones de procesamiento de factorización clásica

3.7.8 Procesadores Cuánticos

Hasta ahora se han explorado todas las características de la computación cuántica y los fenómenos que la definen. Ahora se procederá a definir el funcionamiento de un procesador cuántico y su asombrosa capacidad para el manejo de qubits. Para considerar un sistema cuántico como funcional, este debe tener una estructura tolerante a fallos eventuales y puede ser proyectado como una serie de procesos que deberán cumplirse antes de considerarse factibles. La parte más baja de esta estructura estaría compuesta por un procesador corrector de errores cuánticos o QEC por sus siglas en ingles. En esencia, el QEC es un procesador clásico que lee las mediciones de los qubits para realizar correcciones. La parte superior de la estructura se conoce como la etapa lógica. En esta sección los qubits son manejados luego de ser corregidos y son codificados para realizar los complejos algoritmos cuánticos (Gambetta, Chow, & Matthias, 2017).

El procesador cuántico codifica la información de los qubits en superposición y entrelazamiento. Esta información es manejada mediante interferencias en las complejas superposiciones utilizando compuertas cuánticas. La medición de los datos procedentes de estos qubits son definidos por una combinación de estados que obedece la forma:

$$M = 2^N$$

Ecuación 26. Cantidad de estados para N qubits

Es decir, si se tuviera una cantidad de 400 qubits, la información recabada tendría un valor de 2^{400}, lo que es más que todas las partículas fundamentales del universo. La capacidad de ejecución paralela de la computación cuántica permite el procesamiento de todos estos qubits, sin embargo se requiere de la menor cantidad de ruido exterior posible para garantizar funcionamientos sin errores (Monroe & Kim, 2013).

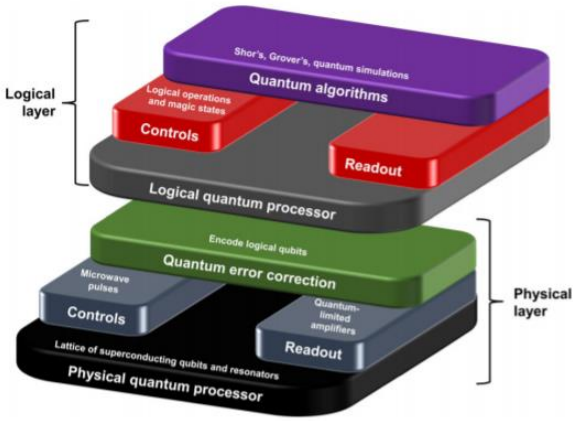

Ilustración 13. Esquema básico para computación cuántica libre de errores

Fuente: (Gambetta, Chow, & Matthias, 2017)

3.7.8.1 Trampa Iónica

Los átomos individuales son un transporte natural para la información cuántica puesto que son universales, es decir permanecen invariables sin importar en que parte del mundo se encuentren. Estos átomos se confinan en un entorno mediante el uso de trampas magnéticas, aislándolos e impidiendo que interactúen incontrolablemente con otros átomos (Monroe & Kim, 2013).

Las primeras propuestas para el uso de trampas iónicas para los procesadores cuánticos, involucraba el aislamiento de una serie de átomos en una sola trampa y posteriormente usar sus estados electrónicos como niveles lógicos de un qubit para transmitir información cuántica mediante sus interacciones de Coulomb. Todos los elementos básicos para la computación cuántica como preparación, manipulación y lectura han podido ser demostradas con estas trampas iónicas, sin embargo cuando se aumentan el número de iones la dificultad de manejo aumenta significativamente. Para solucionar esta problemática se propuso el uso el dispositivo de

carga acoplada cuántica o QCCD por sus siglas en ingles. Este dispositivo es básicamente una serie de trampas iónicas interconectadas en donde es posible almacenar los iones en cualquiera de las trampas, y moverlas en el dispositivo mediante la aplicación de distintos voltajes. Los iones que contienen información cuántica son guardados en la región de memoria, y para realizar una compuerta lógica con los mismos son movidos a la región de interacción (Kielpinski, Monroe, & Wineland, 2002).

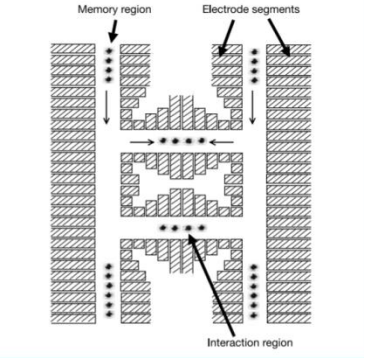

Ilustración 14. Diagrama de un QCCD

Fuente: (Kielpinski, Monroe, & Wineland, 2002)

3.7.8.2 Circuitos superconductores

A diferencia de los circuitos de trampa iónica que son a escala microscópica, los circuitos superconductores están basados en un oscilador eléctrico, que son sistemas macroscópicos de átomos acoplados en la forma de cables y placas metálicas. Como su nombre lo indica, estos dispositivos funcionan mediante el principio de superconducción, que se basa en el flujo eléctrico sin fricción a temperaturas muy bajas. Estos circuitos poseen un dispositivo denominado túnel de unión superconductor, que hace pasar una corriente eléctrica mediante el efecto de efecto túnel cuántico, que permite convertir este circuito en un átomo artificial que posteriormente podrá ser excitado para ser representado como un qubit. Cuando estos osciladores no-lineales son acoplados a un oscilador verdadero, se obtiene un multiqubit de gran fidelidad. Los qubits en estos circuitos tardan mucho más tiempo en decaer en comparación a una trampa iónica lo que permite una corrección de errores más eficiente (Devoret & Schoelkopf, 2013).

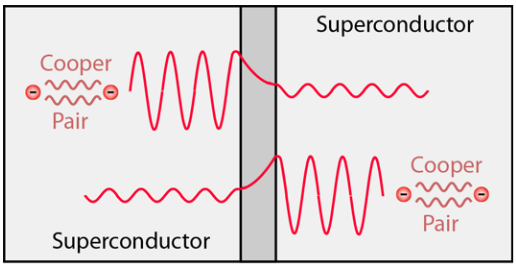

Ilustración 15. Efecto túnel cuántico en un superconductor

Fuente: (Hyperphysics)

3.7.8.3 Decoherencia

La decoherencia es un fenómeno que afecta exclusivamente a los estados de información cuánticos. Este suceso afecta negativamente los resultados obtenidos en un algoritmo, y en conjunto con el ruido externo a los que es sometido un procesador son responsables de datos probabilísticos erróneos. El principio de la decoherencia se basa en el entrelazamiento y la superposición cuántica. Si se supone un sistema de x cantidad de electrones en estado de superposición, estos inherentemente entraran en un estado de entrelazamiento con los átomos circundantes del entorno, lo que cambiara radicalmente el espín cuántico del electrón de interés. Es por esta razón, que los procesadores cuánticos deben estar completamente aislados del entorno, o en cualquier caso en un entorno controlado (Yanofsky & Mannucci, 2008).

3.7.9 IBM Quantum Experience

El 04 de Mayo del 2016, la compañía estadounidense IBM lanzo su campaña IBM Quantum Experience. Esta campaña consiste en una interfaz que puede ser accesada por el público en la página oficial de IBM, en la cual es posible ejecutar algoritmos cuánticos y experimentar la verdadera potencia de los procesadores cuánticos. La compañía pone a disposición de sus usuarios procesadores que varían desde 1 qubit hasta 32 qubits, pudiendo alcanzar velocidades de procesamiento mucho más altas que un ordenador convencional (IBM, 2016)

Esta plataforma pone a disposición un compositor de circuitos cuánticos, proveyendo al usuario las compuertas más comunes, y un glosario en el cual está representado en la esfera de Bloch el comportamiento resultante de un qubit después de pasar por determinada compuerta.

Ilustración 16. Compositor de circuitos cuánticos.

Fuente: (IBM, 2020)

Esta herramienta proporcionada por IBM también ofrece una interfaz para programación en open QASM, que es el lenguaje cuántico basado en Python utilizado por las computadoras cuánticas de IBM. El usuario podrá escribir las líneas de código o usar la interfaz visual en el editor de circuito pudiendo obtener los mismos resultados. Cuando el usuario finaliza un circuito o programa en QASM, es posible simular su comportamiento en el simulador integrado e inspeccionar las amplitudes resultantes; también es posible ejecutar las instrucciones directamente en los procesadores cuánticos de IBM para obtener una medición más enfocada a un ámbito real.

Tabla 6. Lista de procesadores cuánticos de IBM Quantum Experience

Procesador	Qubits
Simulador	32
Melbourne	15
London	5
Burlington	5
Essex	5
Ourense	5
Vigo	5
Yorktown	5

Fuente: (IBM, 2020)

IV. Metodología

En este capítulo abordaremos la metodología y procesos a realizar para llevar a cabo la investigación. En la presente sección se implementaran métodos de comprobación de algoritmos basados en los conocimientos adquiridos anteriormente. Esto será realizado mediante simulaciones en computadoras cuánticas y se analizara la factibilidad de los algoritmos cuánticos. Para poder realizar todas las comprobaciones correspondientes y analizar la los resultados de implementación de los algoritmos cuánticos, se utilizara una metodología en espiral, en donde cada espiral realizada nos dará un resultado concreto de la investigación.

4.1 Enfoque

Uno de los principales objetivos de la investigación es el análisis de tiempos de respuesta, y tomar en consideración las posibles dificultades para implementar algoritmos cuánticos en la actualidad para definir qué tan factible es su implementación en un futuro cercano. Por esta razón se ha optado por tomar un enfoque cuantitativo, en donde se comparara la velocidad de respuesta de estos circuitos ejecutados en procesadores cuánticos reales y los tiempos de respuesta en un entorno simulado.

4.2 Variables de investigación

Una vez determinado el enfoque de la investigación, se pueden establecer las variables que se involucran en la misma. Como su nombre lo indica, es posible definir a una variable como una magnitud no constante, que puede tomar diferentes valores en el tiempo por circunstancias externas.

4.2.1 Variables dependientes

Las variables dependientes serán el resultado de interacciones de otros elementos. Se definirá como variable dependiente a los Qubits. Como se mencionaba anteriormente, un qubit cambiara su comportamiento dependiendo las compuertas que se apliquen en el mismo provocando rotaciones axiales, superposiciones y entrelazamientos. Esta variabilidad de un qubit por el manejo del mismo es de vital importancia para el manejo de la información, y por consiguiente se le considera una variable dependiente.

4.2.2 Variables independientes

Las compuertas cuánticas son responsables del cambio de estado de los qubits, sin embargo estas podrán ser colocadas arbitrariamente según las exigencias del algoritmo de interés. Se consideraran a las compuertas cuánticas como variables independientes a lo largo de este documento puesto que las mismas no serán afectadas por ningún evento en las simulaciones.

4.3 Técnicas e instrumentos aplicados

En la presente tesis se tomaron encuentra investigaciones científicas y libros profesionales con conocimiento comprobado. Se analizaron los avances tecnológicos a la fecha y se escogió un simulador cuántico exclusivo de la compañía IBM, misma que ha sido pionera de los avances tecnológicos en la computación clásica. Se definieron los algoritmos que serían comprobados para sus posteriores análisis, mismos que han sido catalogados por la comunidad científica como las grandes promesas de la computación cuántica.

4.4 Materiales

La computación cuántica ha tenido avances sustanciales en las últimas décadas, sin embargo aún tiene mucho espacio para mejora. Por las limitaciones actuales en cuanto a hardware de procesadores cuánticos disponibles, la lista de materiales es escasa. Para los efectos de la presente investigación solo se utilizara el simulador de circuitos cuánticos de IBM, mismo que se encargara de ejecutar los algoritmos presentados en un procesador cuántico físico, e investigaciones científicas que serán base para el conocimiento comprobado.

4.5 Metodología de estudio

En esta sección se abordaran los métodos de estudio para abordar la problemática anteriormente presentada. Para este fin, se implementara un método en espiral que constara de 3 iteraciones, en el que tan pronto finaliza una de las iteraciones, empieza otra. Las espirales de estudio son las siguientes:

4.5.1 Ciclo 1. Implementación y comprobación del algoritmo de Deutsch

Ilustración 17. Espiral de implementación de algoritmo de Deutsch

Fuente: Propia

Etapa I. Objetivos

Implementar el circuito cuántico correspondiente al algoritmo de Deutsch, partiendo de su forma básica y modificándolo acorde a la estructura de IBMQX. Comprobar si la función en la caja negra del algoritmo es constante o balanceada.

Etapa II. Análisis de riesgos

El algoritmo de Deutsch al igual que todos los algoritmos cuánticos fueron diseñados hace décadas, el riesgo principal que presenta esta espiral es la incorrecta traducción del circuito original a la estructura IBMQX, lo que provocaría resultados incorrectos. Adicionalmente, el ruido presente en un procesador cuántico físico podría alterar los resultados.

Etapa III. Desarrollar, Verificar y validar

En esta sección se desarrollará el circuito de Deutsch en el entorno IBMQX. El algoritmo a implementar es el siguiente:

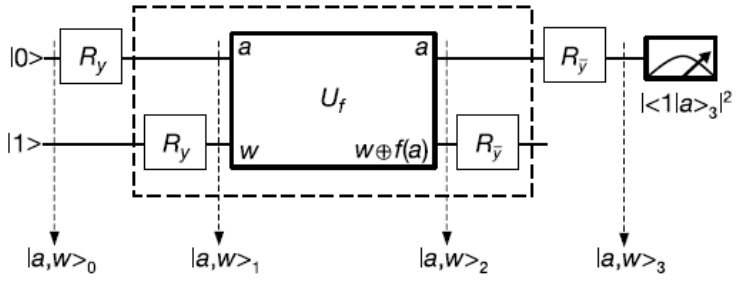

Ilustración 18. Algoritmo de Deutsch

Fuente: (Gulde, et al., 2003)

Como es apreciable, el algoritmo en cuestión no presenta compuertas inexistentes en la plataforma IBMQX. Las funciones de rotación en el eje Y son compuertas Hadamard. La función oráculo U (f) deberá ser modelada específicamente para cada situación, es decir, dependiendo del tipo de función ya sea constante o balanceada el circuito cambiara. No se podría considerar a este circuito como universal, puesto que la función oráculo será variable dependiendo de la aplicación.

Etapa IV. Planificar

Pudiendo considerar esta espiral como un éxito, se comienza a planificar para el algoritmo de Grover. Este algoritmo es sustancialmente más grande en comparación al de Deutsch y requerirá de un número mayor de compuertas.

4.5.2 Ciclo 2. Implementación y comprobación del algoritmo de Grover

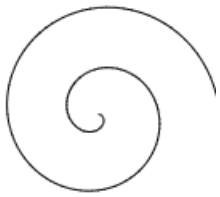

Ilustración 19. Espiral de implementación de algoritmo de Grover

Fuente: Propia

Etapa I. Objetivos

Implementar el circuito cuántico creado por Lov Grover, interpretando el esquema original y traduciéndolo a la plataforma IBMQX. Encontrar el valor de entrada del circuito utilizando la lógica de funcionamiento del algoritmo, y simularlo en un procesador cuántico real.

Etapa II. Análisis de riesgos

Como en el algoritmo anterior, el mayor riesgo presentado en esta implementación es una incorrecta interpretación del circuito traducido lo que ocasionaría un funcionamiento incorrecto. Adicionalmente, la cantidad superior de compuertas utilizadas en contraste al algoritmo de Deutsch, podría ocasionar el inherente error humano y ruidos indeseados en la ejecución.

Etapa III. Desarrollar, verificar y validar

Se procederá a desarrollar el algoritmo de Grover en la plataforma IBMQX. El algoritmo en su forma general es el siguiente:

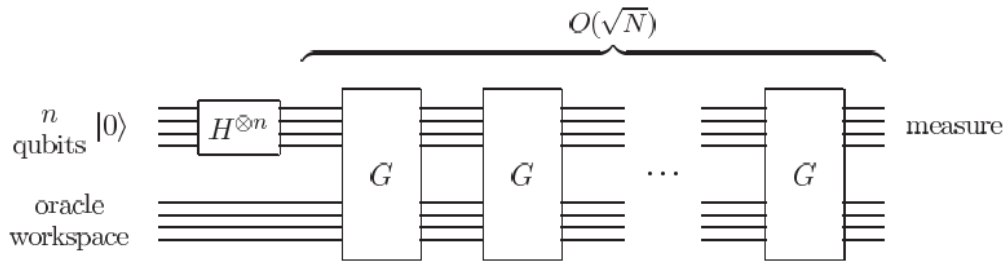

Ilustración 20. Algoritmo de Grover

Fuente: (Nielsen & Chuan, 2010)

Este algoritmo como es apreciable en su forma general, involucra un número mayor de compuertas en comparación al de Deutsch. Al igual que en el algoritmo anterior este circuito dependerá enteramente de su función oráculo, denominada por "G" en el circuito de referencia.

Etapa IV. Planificar

Una vez se ha comprobado el algoritmo de Grover con éxito, se comienza a planificar el desarrollo del algoritmo final de investigación, al algoritmo de Shor. Una vez más, el próximo algoritmo posee una complejidad mayor al de la presente espiral e incorpora una combinación especial entre lo ideado por Shor y la transformada de Fourier.

4.5.3 Ciclo 3. Implementación y comprobación del algoritmo de Shor

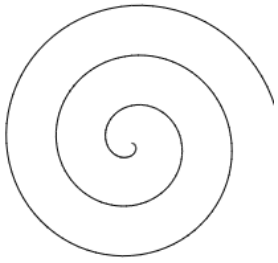

Ilustración 21. Espiral de implementación de algoritmo de Shor

Fuente: Propia

Etapa I. Objetivos

Implementar el algoritmo de Shor para encontrar el periodo de un número propuesto a factorizar. Transpilar el circuito original creado por Shor a la estructura de programación IBMQX. Complementar el algoritmo de Shor con la correcta aplicación de la transformada inversa de Fourier.

Etapa II. Análisis de riesgos

El algoritmo de Shor presenta la mayor complejidad de los algoritmos estudiados en la presente investigación, dicho algoritmo implementa un subalgoritmo, la transformada cuántica inversa de Fourier que es esencial para el cálculo correcto de los periodos. Al tener un número bastante grande de compuertas y el complemento de la transformada inversa de Fourier el error humano será el mayor riesgo. Adicionalmente, el algoritmo de Shor implementa compuertas inexistentes en la plataforma IBMQX, por lo que un equivalente erróneo dará resultados inesperados.

Etapa III. Desarrollar, verificar y validar

En esta sección se desarrollará el último algoritmo de la presente investigación, y el que es considerado como el más complicado. El circuito para el algoritmo de Shor se muestra a continuación:

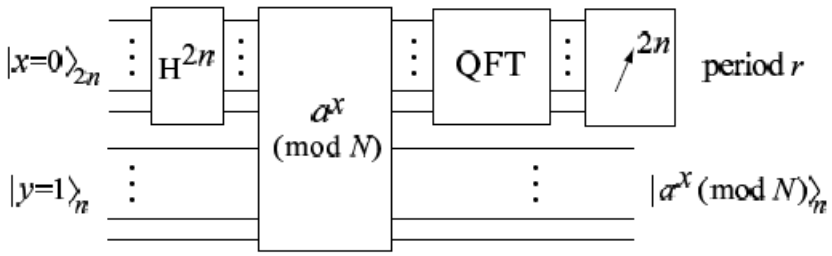

Ilustración 22. Algoritmo de Shor

Fuente: (Vartiainen, Niskanen, Nakahara, & Salomaa, 2004)

A simple vista el circuito aparenta una relativa simplicidad, sin embargo la función de algebra modular representa un reto importante pues cambiara el circuito completamente dependiendo de los valores que se escojan. Adicionalmente, la implementación de la transformada inversa de

Fourier vuelve al circuito más largo y complejo incorporando rotaciones axiales para encontrar el periodo de la función.

Etapa IV. Planificar

En este punto se puede afirmar que todos los algoritmos cuánticos de mayor renombre han sido comprobados con éxito y se avanzara a la siguiente etapa de la investigación.

4.6 Metodología de validación

La presente investigación será validable en base a investigaciones y proyectos científicos realizados en diferentes plataformas con respecto al tema. Los resultados de la presente investigación pretenden comprobar y complementar las investigaciones realizadas anteriormente entorno a la computación cuántica y estudiar la evolución de la misma, para definir qué tan factible es su implementación en los próximos años.

4.7 Cronograma de actividades

Se realizó un cronograma de actividades a realizar a lo largo del periodo para poder completar el presente proyecto de investigación. El mismo se diseñó en base a 10 semanas, teniendo entregables y avances considerables a lo largo de este margen de tiempo.

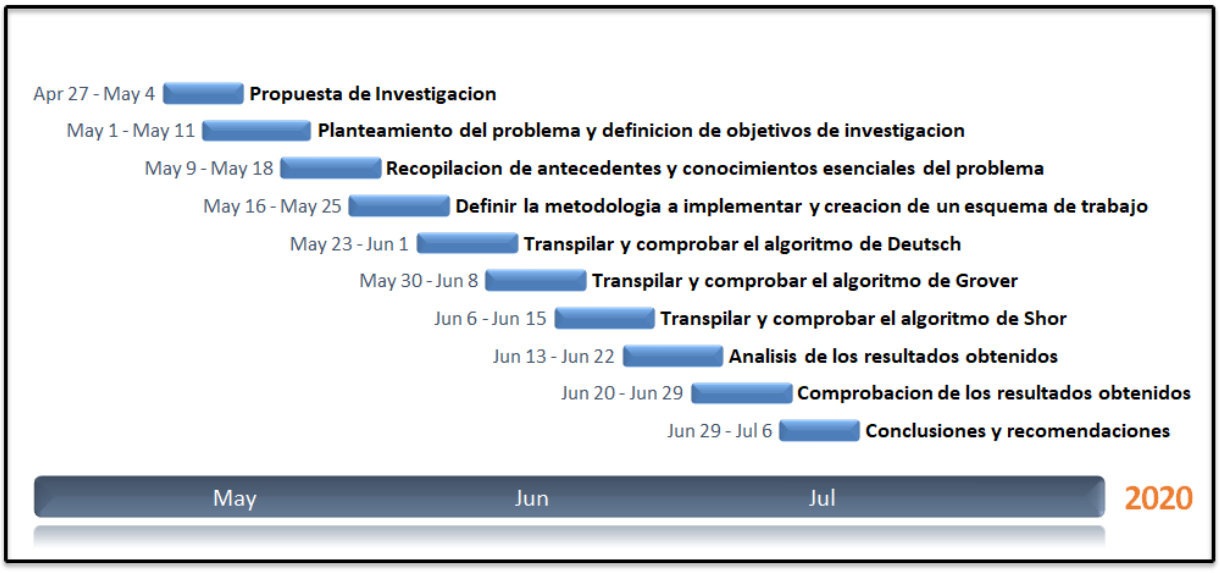

Ilustración 23. Cronograma de actividades

Fuente: Propia

V. Análisis de resultados

En la presente sección se analizaran los soportes brindados por las teorías de sustento expuestas anteriormente y se expondrán los resultados obtenidos por cada espiral de la metodología de investigación. Adicionalmente, se compararan estos resultados con los resultados simulados, y se analizara su factibilidad de implementación basándonos en una relación de dificultad de diseño y ventaja de procesamiento.

5.1 Análisis de teorías de sustento

Si bien a lo largo de este documento se ha estudiado la complejidad inherente de la computación cuántica en comparación con la computación clásica, ambas poseen una lógica bastante similar. Las compuertas lógicas son utilizables en ambos entornos, teniendo diferencias considerables en su construcción y su reversibilidad, pero siempre manteniendo la lógica de ejecución. Como fundamento para esta afirmación, se usara el algoritmo clásico de la transformada de Fourier. Este algoritmo fue creado en un entorno clásico y ha sido una importante herramienta para la digitalización de señales análogas, sin embargo este algoritmo también tiene una forma cuántica, realizando la misma operación en un entorno de programación muy diferente. Es por estas razones que podemos afirmar que si se conocen los principios básicos de funcionamiento de la computación clásica, de cierta manera, se podría facilitar el comprender la computación cuántica.

Es estrictamente necesario poder comprender algunos principios de la mecánica cuántica como la superposición y el entrelazamiento para poder comprender los sistemas computacionales cuánticos. De igual manera, el principio de reversibilidad es uno de los fundamentos básicos de estos sistemas, y clave esencial para las compuertas lógicas. Una vez se ha logrado comprender estos principios básicos, bastara con mezclar todos los conocimientos para poder comprender los circuitos cuánticos. Si bien existen un gran número de algoritmos cuánticos a la fecha, la computación cuántica siempre hace referencia a tres algoritmos específicos para resaltar su capacidad superior de procesamiento: El algoritmo de Deutsch, el algoritmo de Shor y el algoritmo de Grover. Estos algoritmos han sido señalados como una muestra tangible de la superioridad de estas máquinas, y han servido de inspiración para continuar con el desarrollo de

estas tecnologías con la esperanza de poder alcanzar velocidades de procesamiento nunca antes vistas.

5.2 Pruebas realizadas

Se realizaron las pruebas correspondientes a cada espiral de la investigación para comprobar los algoritmos cuánticos propuestos. En esta sección se analizara de una manera más profunda los resultados obtenidos, se estudiara la lógica de ejecución de estos algoritmos y se comprobaran los datos obtenidos mediante comprobaciones matemáticas si el caso lo amerita.

Como método de apoyo para esta sección, se implementara una tabla de descripción de pruebas en donde se recapitulara la definición y objetivos de cada espiral y se integrara esta información con los resultados obtenidos en las pruebas. El formato a seguir es el siguiente:

Tabla 7. Formato de presentación de pruebas realizadas

Objetivos	
Análisis de riesgos	
Desarrollar, verificar y validar	
Resultados	
Planificación	

Fuente: Propia

5.2.1 Ciclo 1 – Algoritmo de Deutsch

Como se mencionaba anteriormente, el algoritmo cuántico de Deutsch está diseñado para encontrar una función $f(x)$ oculta, a la que se le denomina función oráculo en una sola iteración, a diferencia de las dos o más iteraciones necesarias para resolver el mismo problema con un acercamiento clásico.

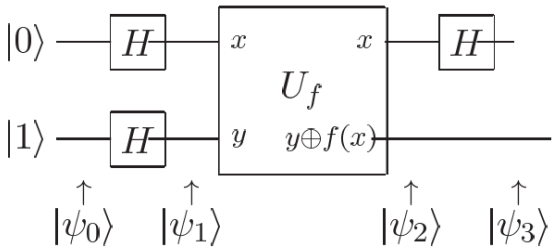

Ilustración 24. Algoritmo de Deutsch y sus etapas.

Fuente: (Nielsen & Chuan, 2010)

En la ilustración 24 se puede observar el algoritmo propuesto por Deutsch y sus etapas. Esta ilustración será de vital importancia para comprender el funcionamiento de este algoritmo. Se analizara este circuito por etapas matemáticamente para comprender el principio de funcionamiento. Es necesario comprender que la función oráculo de este algoritmo es definida directamente por el usuario, y los cálculos matemáticos que se requieran para comprender este funcionamiento partirán de la suposición de una función de oráculo específica. De igual manera, el circuito equivalente de este algoritmo cambiara dependiendo de la función oráculo escogida, por lo que no habrá una forma universal de implementación de este circuito.

Para analizar este algoritmo se afirma que el oráculo posee una función balanceada, es decir, las mediciones finales de los qubits serán un 0 y un 1 el 50% de las veces. Se puede definir esta función como:

$$f(0) = 1$$
$$f(1) = 0$$

Ecuación 27. Función oráculo balanceada

Partiendo de esta premisa, la primera etapa del circuito denominada φ_0 se define por:

$$\varphi_0 = |0,1\rangle$$

Posteriormente las entradas pasan por la compuerta Hadamard, obteniendo:

$$\varphi_1 = \frac{|0\rangle + |1\rangle}{\sqrt{2}} \times \frac{|0\rangle - |1\rangle}{\sqrt{2}}$$

En el siguiente paso, el circuito realiza la función oráculo. Es importante resaltar que la función oráculo es aplicada a un solo qubit, en este caso al primer qubit por lo que la ecuación para este punto es:

$$\varphi_2 = -1^{f(x)}|x\rangle \times \frac{|0\rangle - |1\rangle}{\sqrt{2}}$$

Ecuación 28. Formula general de la función oráculo

$$\varphi_2 = \frac{-1^{f(0)}|0\rangle + (-1^{f(1)})|1\rangle}{\sqrt{2}} \times \frac{|0\rangle - |1\rangle}{\sqrt{2}}$$

Partiendo de la ecuación 28, el valor resultante es:

$$\varphi_2 = (-1)\frac{|0\rangle - |1\rangle}{\sqrt{2}} \times \frac{|0\rangle - |1\rangle}{\sqrt{2}}$$

Una vez ha sido aplicada la función oráculo, se tiene como último paso la aplicación de otra compuerta Hadamard al primer qubit, por lo que nuestro punto final sería definido por la ecuación:

$$\varphi_3 = -1|1\rangle \times \frac{|0\rangle - |1\rangle}{\sqrt{2}}$$

En este punto se puede afirmar que teniendo una función oráculo balanceada, la medición será de un 1 en el qubit de salida, y esto será aplicable a cualquier configuración de función balanceada. Pero, ¿Qué sucedería si se tuviera una función constante? ¿Sería el resultado el opuesto del encontrado, es decir 1? Se supondrá que la función oráculo es:

$$f(0) = 0$$
$$f(1) = 0$$

Ecuación 29. Función oráculo constante

Si se cambia únicamente la función oráculo del algoritmo, el resultado variaría solamente desde φ_2 por lo que en este punto, con una función constante se tiene:

$$\varphi_2 = \frac{-1^{f(0)}|0\rangle + -1^{f(1)}|1\rangle}{\sqrt{2}} \times \frac{|0\rangle - |1\rangle}{\sqrt{2}}$$

Partiendo de la ecuación 29, el resultante sería:

$$\varphi_2 = \frac{|0\rangle + |1\rangle}{\sqrt{2}} \times \frac{|0\rangle - |1\rangle}{\sqrt{2}}$$

Posteriormente, se procede a aplicar una última compuerta Hadamard antes de la medición, obteniendo:

$$\varphi_3 = |0\rangle \times \frac{|0\rangle - |1\rangle}{\sqrt{2}}$$

Es apreciable que efectivamente, el resultado con una función constante fue el complemento del resultado con una función balanceada. Bajo este análisis, se puede definir que en el algoritmo de Deutsch siempre obtendrá una medición 1 para una función oráculo balanceada, y una medición de 0 para una constante. Habiendo analizado el funcionamiento de este algoritmo se aplicara la tabla de presentación para este ciclo.

Tabla 8. Ciclo 1 – Algoritmo de Deutsch

Objetivos	Implementar el circuito cuántico correspondiente al algoritmo de Deutsch, partiendo de su forma básica y modificándolo acorde a la estructura de IBMQX. Comprobar si la función en la caja negra del algoritmo es constante o balanceada.
Análisis de riesgos	El algoritmo de Deutsch al igual que todos los algoritmos cuánticos fueron diseñados hace décadas, el riesgo principal que presenta esta espiral es la incorrecta traducción del circuito original a la estructura IBMQX, lo que provocaría resultados incorrectos. Adicionalmente, el ruido presente en un procesador cuántico físico podría alterar los

	resultados.
Desarrollar, verificar y validar	El algoritmo en cuestión no presenta compuertas inexistentes en la plataforma IBMQX. Las funciones de rotación en el eje Y son compuertas Hadamard. La función oráculo U (f) deberá ser modelada específicamente para cada situación, es decir, dependiendo del tipo de función ya sea constante o balanceada el circuito cambiara. No se podría considerar a este circuito como universal, puesto que la función oráculo será variable dependiendo de la aplicación.
Resultados	Se procedió a realizar el diseño del algoritmo cuántico de Deutsch partiendo de las partes esenciales del mismo. Dicho circuito empieza con dos compuertas Hadamard encargadas de colocar a los bits de entradas en un estado de superposición, el cual será responsable de realizar la operación completa en un solo paso. Posteriormente, se diseñó una función oráculo para cada posible escenario, una función balanceada y una constante. En la ilustración 25 e ilustración 26 se encuentra el circuito cuántico diseñado para cada función. Este circuito se ejecutó en el procesador cuántico de 5 qubits de IBM en Burlington. El circuito y los resultados fueron los esperados, obteniendo una medición de un 1 para un circuito balanceado, y un 0 para un constante.

Planificación	Pudiendo considerar esta espiral como un éxito, se comienza a planificar para el algoritmo de Grover. Este algoritmo es sustancialmente más grande en comparación al de Deutsch y requerirá de un número mayor de compuertas.

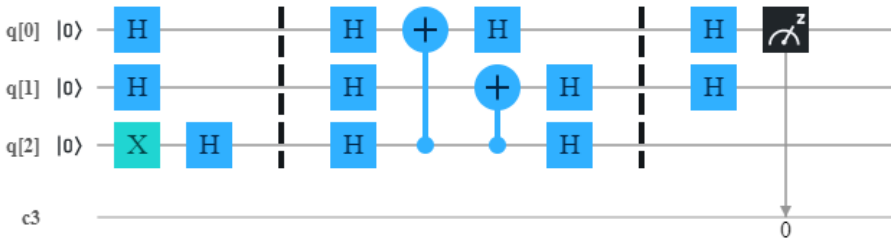

Ilustración 25. Circuito de Deutsch Balanceado en IBMQX

Fuente: Propia

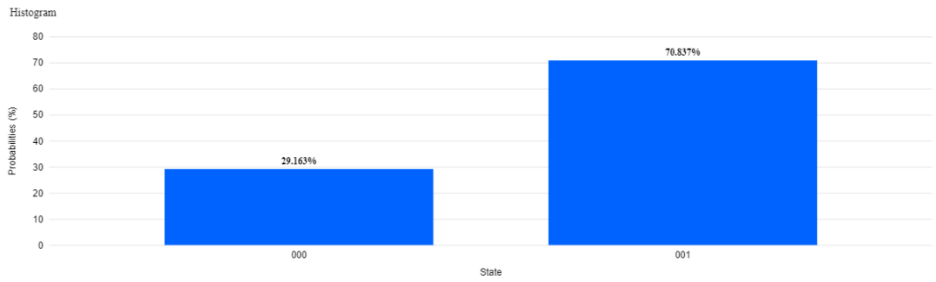

Ilustración 26. Resultados de circuito balanceado de Deutsch en el procesador cuántico de Burlington

Fuente: Propia

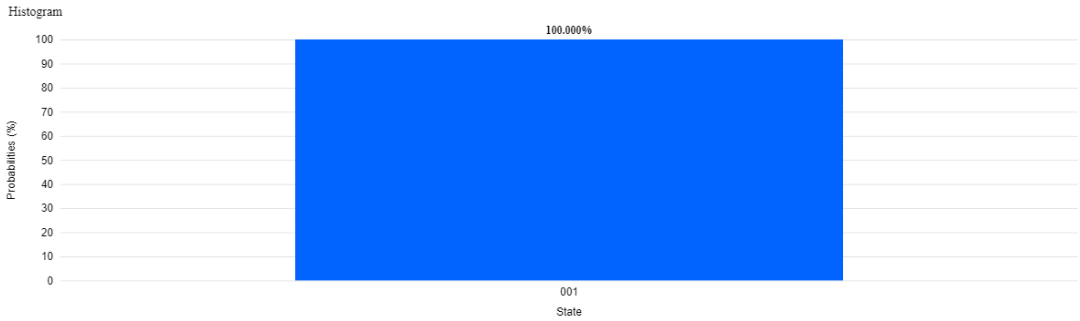

Ilustración 27. Resultados del circuito balanceado de Deutsch en un entorno simulado

Fuente: Propia

La ilustración 25, 26 y 27 muestran el circuito y los resultados obtenidos para una función oráculo balanceada. Es importante recordar que se está midiendo el qubit menos significativo, por lo que en este caso la lectura de 001 denota un 1 en el qubit menos significativo, obteniendo el resultado esperado en la comprobación anterior. De igual manera, la ilustración 27 muestra el resultado obtenido en el simulador, el cual no obtiene el porcentaje de ruido para el valor de 0 obtenido en el procesador cuántico real.

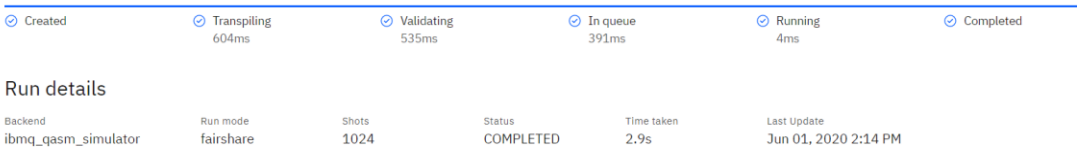

Ilustración 28. Tiempo de procesamiento para el algoritmo de Deutsch balanceado en un entorno simulado

Fuente: Propia

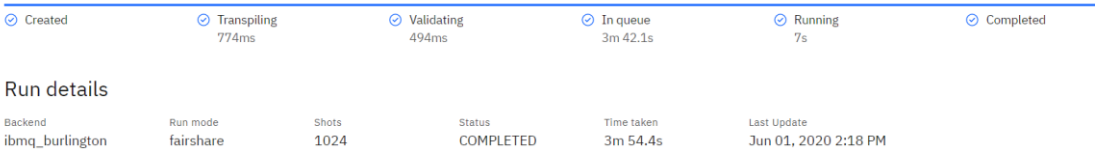

Ilustración 29. Tiempo de procesamiento para el algoritmo de Deutsch balanceado en el procesador cuántico de Burlington

Fuente: Propia

Se ha demostrado que los resultados obtenidos en el procesador real tienen un porcentaje de datos erróneos producto del ruido en comparación con los resultados simulados. Las ilustraciones 28 y 29 muestran los tiempos de procesamiento de un entorno simulado y el procesador cuántico de Burlington. Es apreciable que el tiempo de ejecución tomado por el procesador cuántico real fue sustancialmente mayor al entorno simulado.

Tabla 9. Resultados del algoritmo de Deutsch balanceado

Resultados obtenidos en procesadores cuánticos				
Resultado	Procesador			
	Burlington	Melbourne	Ourense	Yorktown

0	29.12%	34.18%	11.33%	3.32%
1	70.88%	65.82%	88.67%	96.68%
Tiempo de ejecución	7s	6.4s	6.4s	4.9s

Para obtener un resultado más fidedigno, se ejecutó el mismo circuito cuántico en 4 procesadores diferentes para analizar las diferencias obtenidas en cuanto a precisión y tiempos de procesamiento. Los resultados obtenidos pueden ser observados en la tabla 7. El tiempo promedio de ejecución fue de 6.17 s y un porcentaje de precisión del 80.51%. Existe una marcada diferencia entre la precisión del procesador de Burlington y el procesador de Yorktown, sin embargo ambos procesadores contienen la misma cantidad de qubits. Entonces, ¿Por qué existe una diferencia tan marcada si son iguales? Aunque la cantidad de qubits contenidos en ambos procesadores son iguales, la construcción de los mismos no lo es. Ambos procesadores contienen una topología distinta de interacción de qubits y poseen márgenes de errores diferentes. La ilustración 30 y 31 muestran la topología de ambos procesadores y la calibración proporcionada por IBM.

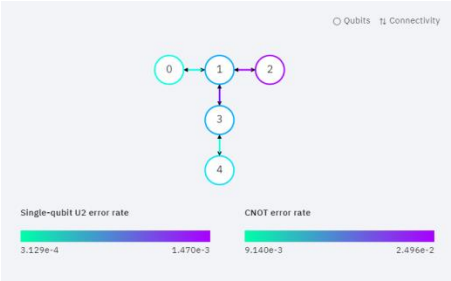

Ilustración 30. Topología y calibración de procesador cuántico de Burlington.

Fuente: (IBM, 2020)

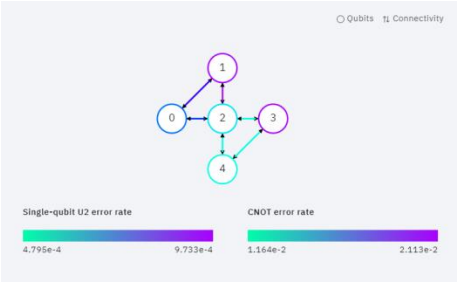

Ilustración 31. Topología y calibración de procesador cuántico de Yorktown.

Fuente: (IBM, 2020)

El circuito cuántico implementado posee en su mayoría compuertas de 1 solo qubit, como ser compuertas Hadamard. Es apreciable que el error máximo para compuertas de un solo qubit en el procesador de Yorktown es de $9.733\ x\ 10^{-4}$, mientras que en el de Burlington es $1.470\ x\ 10^{-3}$. Esto significa, que si se tiene un número mayor de compuertas de un solo qubit, el procesador de Yorktown tendrá una ventaja considerable. De igual manera, la topología de interacción de qubits le permite alcanzar velocidades de ejecución ligeramente superiores.

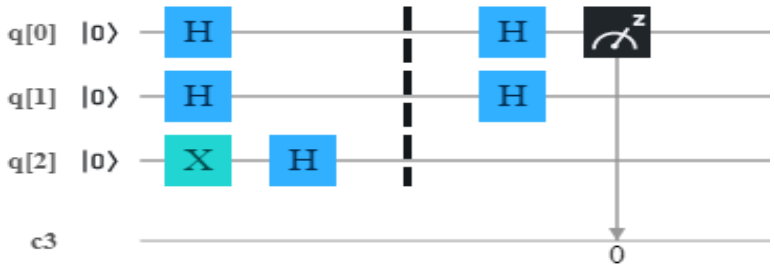

Ilustración 32. Circuito de Deutsch constante en IBMQX

Fuente: Propia

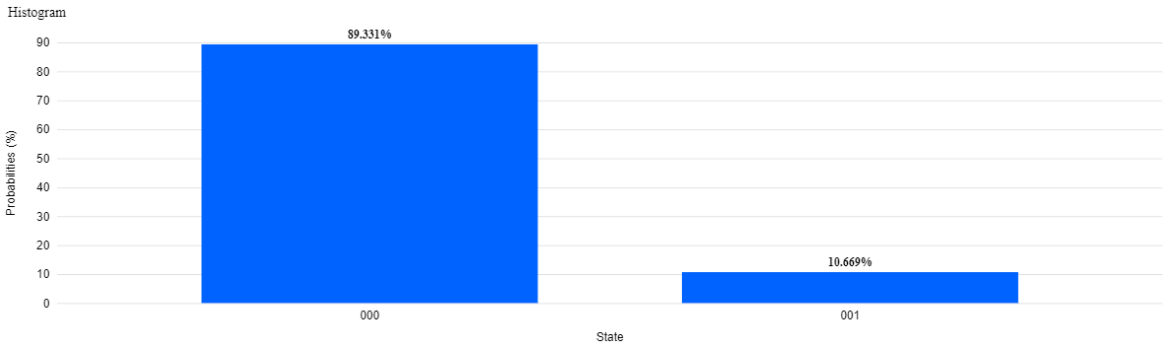

Ilustración 33. Resultados de circuito constante de Deutsch en el procesador cuántico de Burlington.

Fuente: Propia

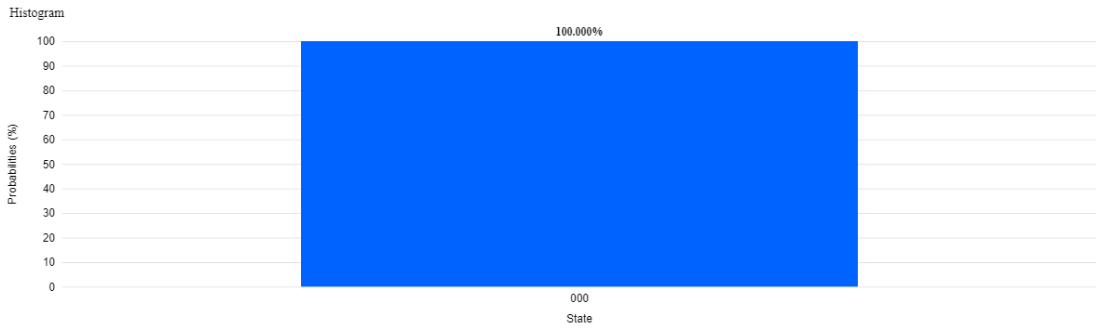

Ilustración 34. Resultados del circuito constante de Deutsch en un entorno simulado

Fuente: Propia

La ilustración 32, muestra el circuito implementado para una función constante. Este algoritmo se ejecutó con 2 configuraciones diferentes, una constante y otra balanceada para obtener resultados sólidos en cuanto a la fiabilidad de los datos obtenidos. Las ilustraciones 33 y 34 muestran los resultados obtenidos en el simulador y en el procesador de Burlington. Ambos resultados fueron los esperados, obteniendo un 0 como resultado en el bit menos significativo puesto que la función es constante. El procesador real obtuvo una pequeña probabilidad de resultado erróneo, esto es debido al ruido.

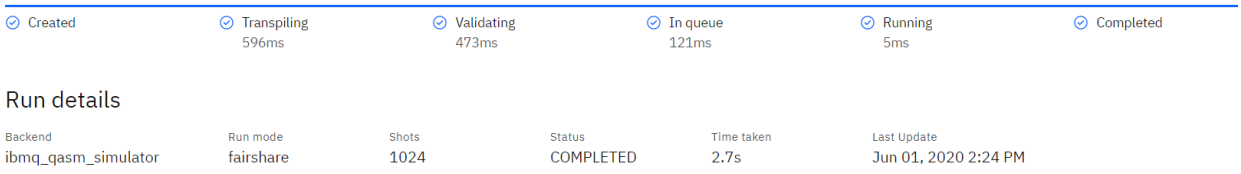

Ilustración 35. Tiempo de procesamiento del algoritmo de Deutsch en un entorno simulado.

Fuente: Propia

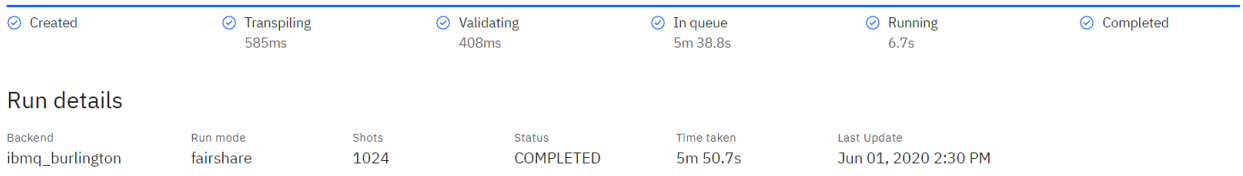

Ilustración 36. Tiempo de procesamiento del algoritmo de Deutsch en el procesador cuántico de Burlington

Fuente: Propia

Los tiempos de procesamiento obtenidos para esta sección del algoritmo están en un rango similar con los datos obtenidos para la función balanceada. La tabla 9 muestra los resultados obtenidos en otros procesadores cuánticos. El tiempo de ejecución promedio para este algoritmo fue de 6.05s y tuvo una precisión de 93%. En esta ocasión el procesador de Yorktown no fue el más preciso como en el algoritmo balanceado, esto es causado por la eliminación de las compuertas CNOT que eliminan ese margen de error del resto de los procesadores.

Tabla 10. Resultados del algoritmo de Deutsch Constante

Resultados obtenidos en procesadores cuánticos				
Resultado	Procesador			
	Burlington	Melbourne	Ourense	Yorktown
000	89.33 %	93.16%	97.56%	91.41%
001	10.67%	0.29%	0.78%	0.49%
010	0%	6.45%	1.66%	7.91%
011	0%	0.10%	0.00%	0.20%
Tiempo de ejecución	6.7s	6.1s	6.7s	4.7s

El algoritmo cuántico de Deutsch fue implementado en el 2003 por un grupo de investigadores en un procesador cuántico de trampa iónica. Se llevaron a cabo pruebas para una función constante y una balanceada, y los resultados obtenidos tienen un alto porcentaje de fidelidad. Estos resultados afirman tener sobre un 90% de precisión en la respuesta correcta, en un número indeterminado de repeticiones. (Gulde, et al., 2003). La comparación de los resultados obtenidos en el procesador de trampa iónica y los obtenidos en el procesador superconductor de IBM poseen una relativa similitud, aunque no está claro el tiempo de procesamiento de dicho procesador. En ambos casos es apreciable una cantidad aproximada de 10-20% de error, y esto es debido a la alta sensibilidad del sistema al ruido.

Posteriormente, en 2018 un grupo de investigadores implemento una variante del algoritmo de Deutsch-Josza en un procesador cuántico de IBM de 5 qubits. Si bien igual al artículo anterior, no se proporcionan tiempos de procesamiento, en esta implementación es apreciable la aparición de ruido. Este algoritmo fue implementado en el procesador cuántico de IBM ibmqx4 de Tenerife, que a la fecha no está disponible, por lo que es imposible una ejecución del algoritmo

en el mismo procesador para efectos de comparación. Sin embargo, la interferencia obtenida para esta implementación está en un rango similar a las obtenidas en el presente documento, por lo que es muy probable que los tiempos de procesamiento sean bastante cercanos (Gangopadhyay, Manabputra, Behera, & Panigrahi, 2018).

Basado en los resultados, se puede afirmar que el algoritmo de Deutsch efectivamente comprueba la gran capacidad de paralelismo de una computadora cuántica, pudiendo obtener el resultado en una sola iteración, sin embargo el algoritmo por si solo carece de una funcionalidad concreta y puede ser considerado como una simple demostración de las capacidades cuánticas. En este punto la investigación avanza a la siguiente etapa, el algoritmo de Grover.

5.2.2 Ciclo 2 – Algoritmo de Grover

Como se mencionaba anteriormente, el algoritmo de Grover consiste en un conjunto de operaciones para encontrar un valor determinado en una base de datos desordenada. La ecuación 22 y 23 contienen los tiempos de procesamiento para estas operaciones en un procesador cuántico y uno clásico. Es apreciable la ventaja de velocidad en el algoritmo cuántico, y como ya sabemos esto será por el alto nivel de paralelismo que se puede alcanzar mediante las superposiciones. El algoritmo de Grover a desarrollar es el siguiente:

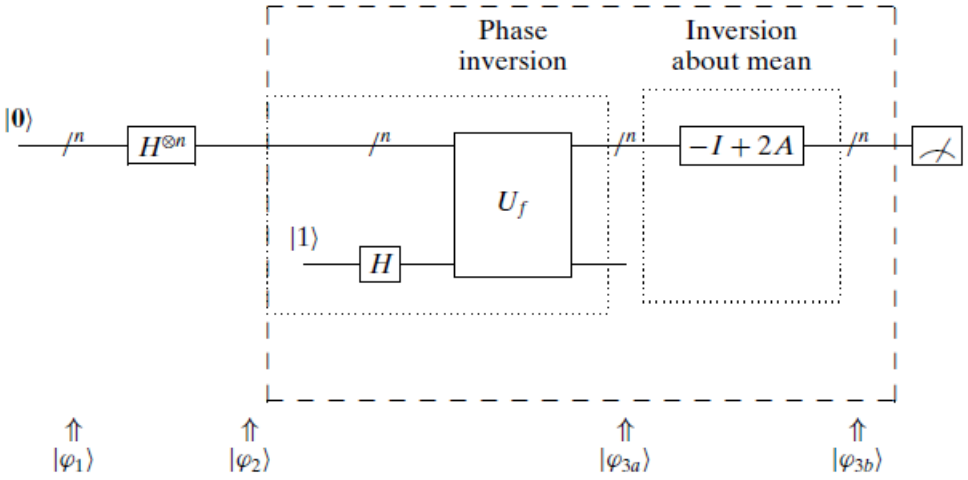

Ilustración 37. Formula general del algoritmo de Grover

Fuente: (Yanofsky & Mannucci, 2008)

Si se observa el circuito a desarrollar y se compara con la ilustración 24, que contiene el algoritmo de Deutsch, es identificable que ambos circuitos son bastante similares, aplicando una compuerta Hadamard en las entradas, y pasando por una función oráculo desconocida. El circuito de Grover constara de dos etapas: la primera será la responsable de identificar el valor buscado, cambiando el signo del valor de interés a negativo. Posteriormente, la inversión de fase con respecto al promedio hará resaltar aún más el valor buscado. Se puede definir los parámetros de búsqueda como:

$$f(x) = \begin{cases} 0 \; si \; x \; \neq u \\ 1 \; si \; x = u \end{cases}$$

Ecuación 30. Parámetros de búsqueda de algoritmo de Grover

Al ser gran parte de este circuito idéntico al algoritmo de Deutsch, tendrá el mismo comportamiento matemático hasta la función oráculo, por lo que se define que en φ_3 el circuito tendría la forma obtenida en la ecuación 30, por lo que:

$$\varphi_3 = \; -1^{f(x)}|x\rangle \; \times \left[\frac{|0\rangle - \; |1\rangle}{\sqrt{2}}\right]$$

Por consiguiente, si $x \; \neq u$:

$$\varphi_3 = \; \frac{-1^0|0\rangle + (-1^0)|1\rangle}{\sqrt{2}} \; \times \left[\frac{|0\rangle - \; |1\rangle}{\sqrt{2}}\right]$$

Es observable que el signo del primer Qubit será positivo si el registro en cuestión no es el indicado, y será negativo si ha llegado al registro de interés, por lo que de manera general se definirá que:

$$\varphi_3 = \begin{cases} -1|x\rangle \; \times \left[\frac{|0\rangle - \; |1\rangle}{\sqrt{2}}\right] \; si \; x = u \\ 1|x\rangle \; \times \left[\frac{|0\rangle - \; |1\rangle}{\sqrt{2}}\right] \; si \; x \; \neq u \end{cases}$$

Ecuación 31. Inversión de signo de algoritmo de Grover

En este punto se puede considerar que es bastante identificable el valor que estamos buscando pues el signo será diferente. Si se supone que el qubit $|x\rangle$ comienza en una superposición

uniforme de la forma:$\left[\frac{1}{2} \quad \frac{1}{2} \quad \frac{1}{2}\right]$, se cambia el signo de alguno de estos registros y se realiza una medición, la magnitud de todos sin importar el signo siempre seria $\frac{1}{4}$. Es por esta razón que el algoritmo de Grover implementa una inversión del promedio, para resaltar aún mas cual es el valor buscado. Esta inversión, realiza una operación que obedece a la forma (Yanofsky & Mannucci, 2008):

$$v' = -v + 2a$$

Ecuación 32. Inversión del promedio

Donde v es el valor numérico, y a es el promedio. Para ejemplificar mejor esta situación, se supondrá que se poseen 4 números: 10,20, 50 y 100. Se está buscando el número 50, por lo que en la primera fase del circuito se obtiene: 10, 20, -50, 100. Aplicando la inversión del promedio el resultado sería:

$$v' = -10 + 2(45) = 80$$
$$v' = -20 + 2(45) = 70$$
$$v' = -(-50) + 2(45) = 140$$
$$v' = -100 + 2(45) = -10$$

Es apreciable que la combinación del cambio de signo con la inversión del promedio ha hecho resaltar aún más el número de interés, pudiendo señalar con una alta precisión el número buscado. Ahora que se ha comprendido a fondo el funcionamiento de este algoritmo, se procederá a detallar las pruebas.

Tabla 11. Ciclo 2 – Algoritmo de Grover

Objetivos	Implementar el circuito cuántico creado por Lov Grover, interpretando el esquema original y traduciéndolo a la plataforma IBMQX. Encontrar el valor de entrada del circuito utilizando la lógica de funcionamiento del algoritmo, y simularlo en un procesador cuántico real.
Análisis de riesgos	Como en el algoritmo anterior, el mayor riesgo

	presentado en esta implementación es una incorrecta interpretación del circuito traducido lo que ocasionaría un funcionamiento incorrecto. Adicionalmente, la cantidad superior de compuertas utilizadas en contraste al algoritmo de Deutsch, podría ocasionar el inherente error humano y ruidos indeseados en la ejecución.
Desarrollar, verificar y validar	Este algoritmo como es apreciable en su forma general, involucra un número mayor de compuertas en comparación al de Deutsch. Al igual que en el algoritmo anterior, este circuito dependerá enteramente de su función oráculo, denominada por "G" en el circuito de referencia.
Resultados	Se procedió a realizar el diseño del algoritmo cuántico de Grover para buscar el numero binario 110. Se implementó el circuito con sus 3 partes específicas, la preparación de las superposiciones, la función oráculo y la inversión del promedio. Se puede observar el circuito implementado en la ilustración 38. El algoritmo fue ejecutado en el procesador cuántico de 5 qubits de IBM de Burlington. El tiempo de procesamiento fue de 7.7s y se obtuvieron resultados con una cantidad considerable de ruido. El circuito funciono como se esperaba pudiendo encontrar el número indicado. Es importante resaltar que este circuito depende exclusivamente de su función oráculo, y esta deberá ser cambiada

	dependiendo del número a buscar.
Planificación	Una vez se ha comprobado el algoritmo de Grover con éxito, se empezara a planificar el desarrollo del algoritmo final de investigación, al algoritmo de Shor. Una vez más, el próximo algoritmo posee una complejidad mayor al de la presente espiral e incorpora una combinación especial entre lo ideado por Shor y la transformada de Fourier.

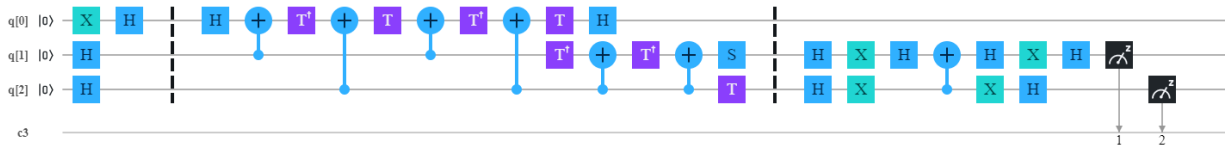

Ilustración 38. Circuito de Grover para buscar 110

Fuente: Propia

En la ilustración 38 se puede apreciar el circuito utilizado en este ciclo. La primera parte del circuito es la preparación de las superposiciones, la parte central corresponde a la función oráculo para la inversión del signo, y la parte final consiste en la inversión de media.

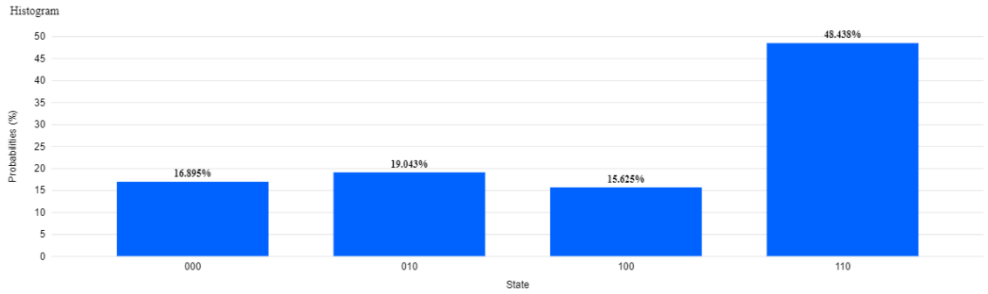

Ilustración 39. Resultados obtenidos para el algoritmo de Grover en el procesador cuántico de Burlington.

Fuente: Propia

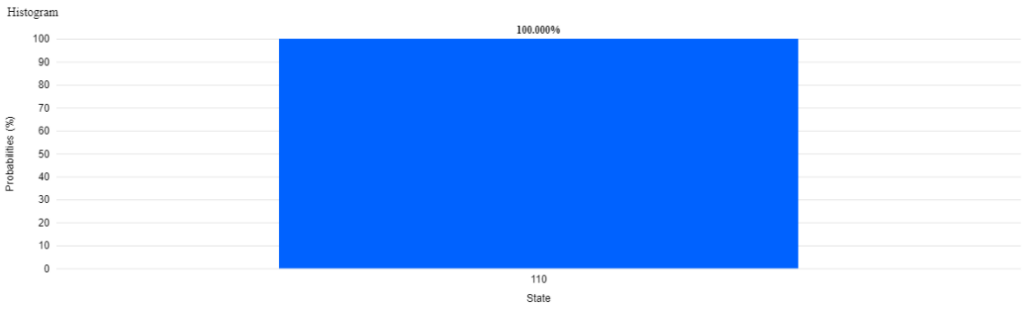

Ilustración 40. Resultados del algoritmo de Grover en un entorno simulado.

Fuente: Propia

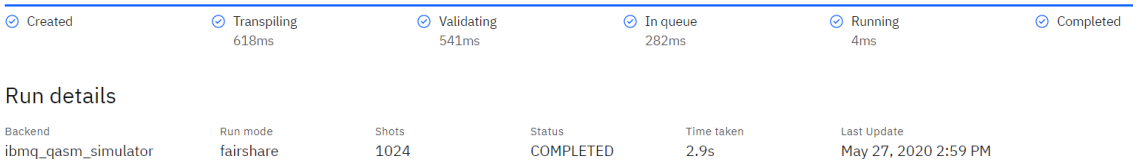

Ilustración 41. Tiempo de ejecución del algoritmo de Grover simulado

Fuente: Propia

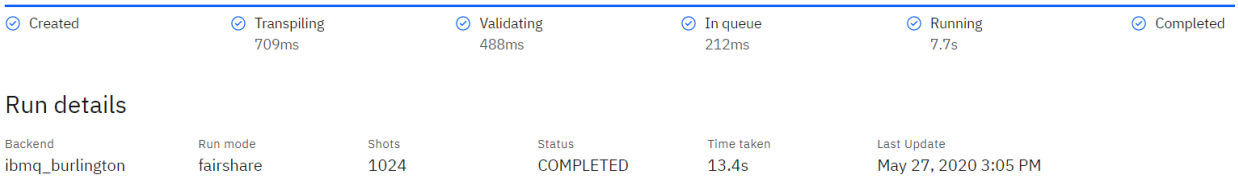

Ilustración 42. Tiempo de ejecución del algoritmo de Grover en procesador cuántico de Burlington.

Fuente: Propia

Los resultados obtenidos fueron los esperados, sin embargo el comportamiento real del procesador es bastante alejado al comportamiento del circuito en un entorno simulado, puesto que en las mediciones hubo bastante ruido. Es importante resaltar que este circuito depende excesivamente en su función oráculo, y esta debe ser modificada en función del registro a buscar. De cierta manera, se podría considerar que este algoritmo recibe un “tip” de la función oráculo con respecto a donde se encuentra la respuesta. Es por esta razón, que se considera que si bien el algoritmo ha podido ser probado y funciono exitosamente, carece de una universalidad para ser

implementado a corto plazo de una manera general. Es decir, es imposible cambiar las entradas del circuito para poder buscar un número diferente, puesto que la función oráculo no funcionaria de la manera deseada y daría una respuesta completamente errada.

Tabla 12. Resultados del algoritmo de Grover

Resultados obtenidos en procesadores cuánticos				
Resultados	Procesador			
	Burlington	Melbourne	Ourense	Yorktown
000	16.90%	8.20%	2.73%	6.06%
010	19.04%	16.80%	7.42%	13.57%
100	15.63%	21.38%	8.98%	6.83%
110	48.44%	53.61%	81%	73.54%
Tiempo de ejecución	7.7s	6.4s	6.1	4.9s

Fuente: Propia

La tabla 11 muestra los resultados obtenidos en los demás procesadores cuánticos. En este caso el procesador más preciso fue el de Ourense, y esto es debido a que el algoritmo de Grover contiene bastantes compuertas CNOT, y el procesador de Ourense es el que tiene el menor margen de error en este tipo de compuertas. La precisión media para este algoritmo fue de 64% y un tiempo promedio de procesamiento de 6.275s.

En 2008 el investigador Pedro J. Salas llevo a cabo un estudio para analizar los efectos del ruido y decoherencia sobre el algoritmo de Grover. El estudio concluyo en que el algoritmo de Grover solo podría ser útil para cálculos que involucren 11 qubits y 2048 registros o menos. Esto no significa que en estos parámetros el ruido será inexistente, sin embargo experimentalmente utilizar valores superiores a este rango seria completamente inaccesible por el ruido ocasionado por la decoherencia (Salas, 2008).

Adicionalmente, este algoritmo fue implementado de una manera similar en 2018 en la plataforma de IBMQX por un grupo de investigadores. Diseñaron un circuito cuántico para encontrar el número 111 y fue ejecutado al igual que el algoritmo de Deutsch en el procesador cuántico de IBM en Tenerife. Para esta implementación se usaron 8192 intentos para maximizar la precisión, obteniendo resultados correctos para un 59.69% de las ocasiones y un tiempo de procesamiento de 84.33s (Mandviwalla, Ohshiro, & Ji, 2018).

En el mismo año, otro grupo de investigadores de Estados Unidos implementaría un circuito para buscar el 110 en la plataforma de IBMQX en el procesador de Tenerife. Los resultados obtenidos en ese estudio fueron de un 65% de precisión, un resultado mayor al obtenido en la presente investigación, lo que da un claro panorama de que todos los procesadores tienen un distinto grado de precisión y tolerancia al ruido (Coles, et al., 2018).

En este punto se ha concluido con la implementación del algoritmo de Grover con éxito, y la investigación avanza al siguiente ciclo, la implementación del algoritmo de Shor.

5.2.3 Ciclo 3 – Algoritmo de Shor

Anteriormente se había hablado del algoritmo de Shor, y como era capaz de realizar la factorización de un número en un lapso de tiempo mucho menor en comparación con el método clásico computacional de factorización. La ecuación 25 y 26 hace referencia a los tiempos de procesamiento para encontrar los factores de un número en ambos métodos. El algoritmo a implementar es el siguiente:

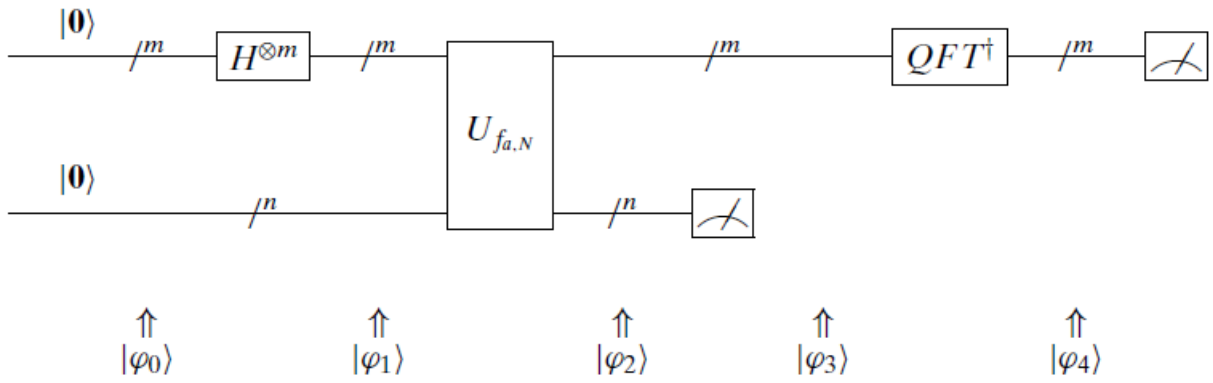

Ilustración 43. Algoritmo de Shor

Fuente: (Yanofsky & Mannucci, 2008)

La ilustración 43 muestra el circuito generalizado con sus respectivas fases de cálculo. La línea superior de nuestro circuito representa a los términos m, que serán qubits de control encargados de llevar a cabo las operaciones del algoritmo, mientras que la línea inferior denominada como n son todos los qubits necesarios para representar el número trivial "a" escogido para el cálculo. Se abordara el proceso matemático involucrado en cada etapa de este circuito, y posteriormente se demostrara de una manera simplificada. En la primera etapa se tiene que:

$$\varphi_0 = |0_m, 0_n\rangle$$

Después de haber inicializado los qubits en 0, se procederá a aplicar compuertas Hadamard a todos los qubits de control. De forma que:

$$\varphi_1 = \frac{\sum_{x\in\{0,1\}^m}|x, 0_n\rangle}{\sqrt{2^m}}$$

Después de haber aplicado las compuertas Hadamard a los qubit de control, se pasara por la función oráculo, que aplicara la función de algebra modular, por lo que en la segunda etapa tenemos que:

$$\varphi_2 = \frac{\sum_{x\in\{0,1\}^m}|x, a^x ModN\rangle}{\sqrt{2^m}}$$

La etapa 3 y 4 son las mediciones de los datos obtenidos, siendo la 4 la responsable de encontrar el periodo de la función haciendo rotaciones axiales de 45 y 90 grados a los qubits de control. Si bien se ha estudiado un circuito general y sus respectivas fases, la complejidad de este algoritmo presenta un verdadero reto. A continuación se abordara el proceso matemático involucrado en este algoritmo más a profundidad, basándose en las pruebas a realizar durante este ciclo. El circuito a implementar es el siguiente:

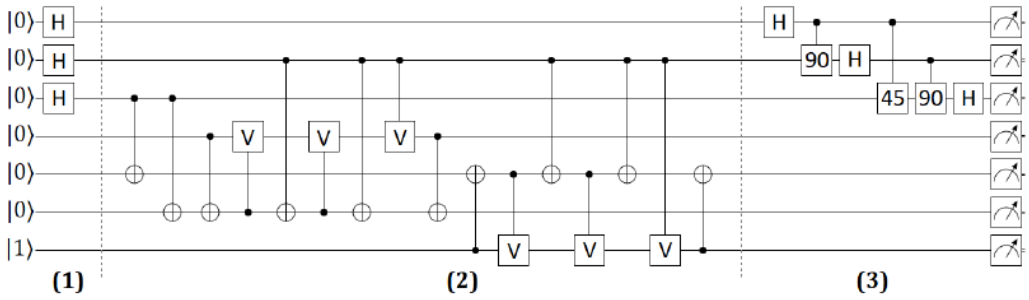

Ilustración 44. Algoritmo de Shor implementado para factorizar el número 15.

Fuente: (Carare, Carillo de Albornoz, & Taylor, 2018)

De manera general, se ocupa factorizar un numero N. como criterio especial para escoger el factor N, se debe tomar en cuenta que no debe ser un número par ni una potencia de un número más pequeño, puesto que el uso del algoritmo seria innecesario. Posteriormente, se escoge un número trivial a, que cumpla la limitante $1 < a < N$. Como siguiente paso, se debe encontrar el máximo común divisor del factor trivial a, y el número a factorizar N. Para poder continuar con el problema, ambos deben ser divisibles solamente por el número 1, caso contrario el uso del algoritmo será innecesario puesto que ya se conoce el factor. Si efectivamente solo son divisibles por 1, se debe escoger el número más pequeño x, que sea mayor que 1 y que cumpla la limitante $a^p \equiv 1 Mod N$. Si p es impar, o cumple la condición de $a^{p/2} \equiv -1 Mod N$, se deberá volver a empezar y escoger un nuevo factor trivial a. Una vez se ha encontrado el valor p, los factores del número N serán:

$$f = \begin{cases} gcd(a^{\frac{p}{2}} + 1, C) \\ gcd(a^{\frac{p}{2}} - 1, C) \end{cases}$$

Ecuación 33. Factores de un numero C

El algoritmo de Shor solamente se ha ejecutado bajo dos configuraciones posibles en entornos completamente cuánticos, por lo que en la demostración se usara una de estas configuraciones. Se procederá a factorizar el número 15, y se escogió el factor trivial a = 7. A continuación, se presentara la tabla de resultados para este ciclo.

Tabla 13. Ciclo 3 – Algoritmo de Shor

Objetivos	Implementar el algoritmo de Shor para encontrar el periodo de un número propuesto a factorizar. Transpilar el circuito original creado por Shor a la estructura de programación IBMQX. Complementar el algoritmo de Shor con la correcta aplicación de la transformada inversa de Fourier.
Análisis de riesgos	El algoritmo de Shor presenta la mayor complejidad de los algoritmos estudiados en la presente investigación, dicho algoritmo implementa un subalgoritmo, la transformada cuántica inversa de Fourier que es esencial para el cálculo correcto de los periodos. Al tener un número bastante grande de compuertas y el complemento de la transformada inversa de Fourier el error humano será el mayor riesgo. Adicionalmente, el algoritmo de Shor implementa compuertas inexistentes en la plataforma IBMQX, por lo que un equivalente erróneo dará resultados inesperados.
Desarrollar, verificar y validar	A simple vista el circuito aparenta una relativa simplicidad, sin embargo la función de algebra modular representa un reto importante pues cambiara el circuito completamente dependiendo de los valores que se escojan. Adicionalmente, la implementación de la transformada inversa de Fourier vuelve al

	circuito más largo y complejo incorporando rotaciones axiales para encontrar el periodo de la función.
Resultados	Se procedió a implementar el algoritmo de Shor para factorizar el número 15. Se construyeron compuertas equivalentes para modelar las compuertas de V-controlada, ya que las mismas son inexistentes en la plataforma de IBMQX. Se diseñó el circuito partiendo de la ilustración 36, teniendo como resultado el circuito traducido a la plataforma IBMQX en la ilustración 37. A diferencia de los otros dos algoritmos, este deberá ser ejecutado en el procesador de Melbourne, puesto que el de Burlington es solo de 5 qubits, mientras que el algoritmo de Shor requiere 7. Se obtuvieron los resultados esperados con cierta cantidad de ruido, y los resultados en velocidad de procesamiento son bastante alejados en el entorno real si los comparamos con los resultados obtenidos en el entorno simulado. Si bien el algoritmo de Shor funciono como se esperaba para encontrar el periodo, se debieron hacer cálculos matemáticos para comprobar la veracidad de este dato. Al igual que los algoritmos anteriores, este algoritmo no presenta universalidad y su dificultad de implementación es bastante elevada. Se requiere una computadora clásica para finalizar el proceso de factorización y poder

Planificación	obtener los factores. En este punto se puede afirmar que todos los algoritmos cuánticos de mayor renombre han sido comprobados con éxito y se avanzara a la siguiente etapa de la investigación.

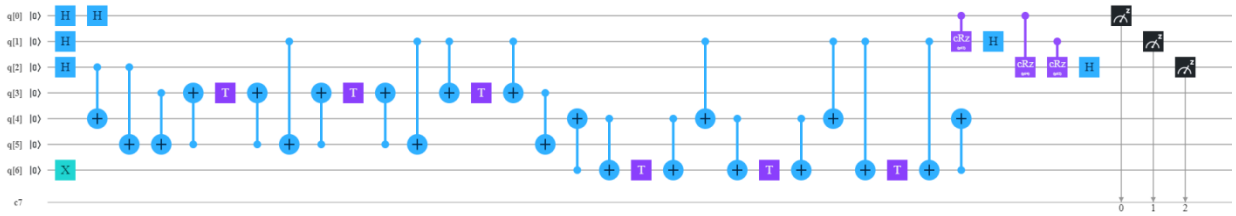

Ilustración 45. Algoritmo de Shor implementado en IBMQX

Fuente: Propia

Como es apreciable en la ilustración 45, para poder implementar el algoritmo de Shor se ocuparon 3 qubits de control y 4 qubits para representar los números procesados por el álgebra modular. Adicionalmente, las compuertas de V-controlada fueron modeladas con sus equivalencias en la plataforma IBMQX, que consisten en dos compuertas CNOT sobre el qubit de control, y un una rotación en el eje z de $\frac{\pi}{4}$ en medio de los CNOT.

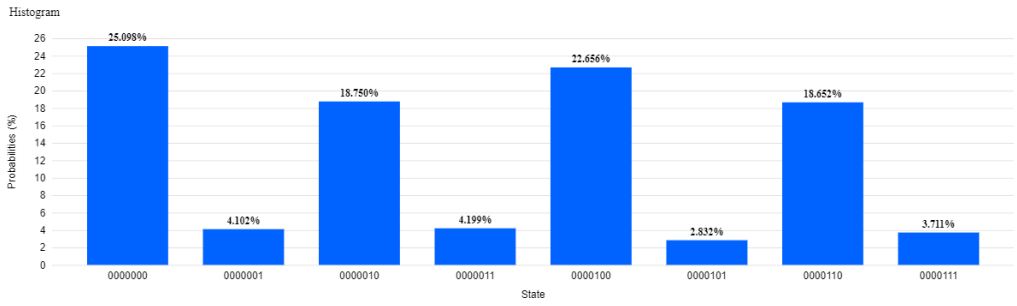

Ilustración 46. Resultados de algoritmo de Shor en procesador cuántico de Melbourne

Fuente: Propia

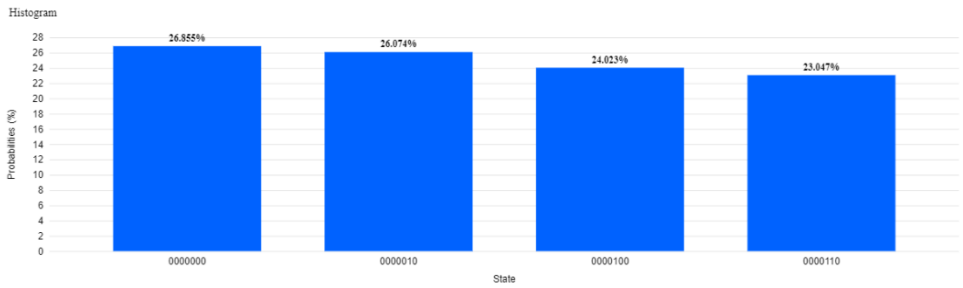

Ilustración 47. Resultados del algoritmo de Shor en un entorno simulado.

Fuente: Propia

La ilustración 46 y 47 contienen los resultados obtenidos para este algoritmo, una vez mas es apreciable que el resultado del procesador cuántico contiene cierta cantidad de ruido, pero las probabilidades erróneas son relativamente bajas en comparación a las probabilidades correctas, por lo que se considera que el algoritmo funciono exitosamente. Se definió que este algoritmo serviría como base para factorizar el número 15, sin embargo en números decimales los resultados obtenidos fueron: 0, 2, 4 y 6. Estos números no corresponden a los factores del 15, si no a los valores medios muestreados por la transformada cuántica de Fourier. Se considera que la obtención de estos valores son los que presentan realmente un reto a la computación clásica, es por eso que esta es la única parte ejecutada en un procesador cuántico, el resto se hará en un computador clásico. Una vez obtenidos estos datos, se puede comprobar la veracidad de los mismos con la siguiente ecuación:

$$p = \frac{\widetilde{x}}{2^L}$$

Ecuación 34. Frecuencia de una muestra tomada por la transformada de Fourier

Si se ingresan los valores obtenidos en los resultados en la ecuación 35, se obtiene que:

$$\frac{0}{2^3}, \frac{2}{2^3}, \frac{4}{2^3}, \frac{6}{2^3}$$

Si se simplifica esta ecuación, se obtiene:

$$0, \frac{1}{4}, \frac{2}{4}, \frac{3}{4}$$

Se puede apreciar que el denominador común para todos estos valores es 4, por lo que ese será el valor p que cumpla $a^p \equiv 1 Mod N$. Con este valor, es posible remitirse a lo expuesto en la ecuación 34, y finalmente obtener los factores del número 15, por lo que se tendría:

$$f = \begin{cases} gcd(50, 15) \\ gcd(48, 15) \end{cases}$$

Al obtener el máximo común divisor de estos números, se obtiene como respuesta el 3 y 5, que efectivamente son los factores del número 15, por lo que se puede afirmar que los resultados obtenidos por el algoritmo son correctos.

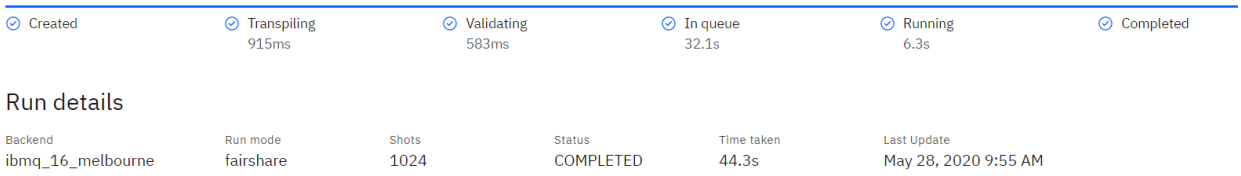

Ilustración 48. Tiempos de procesamiento del algoritmo de Shor en el procesador de Melbourne

Fuente: Propia

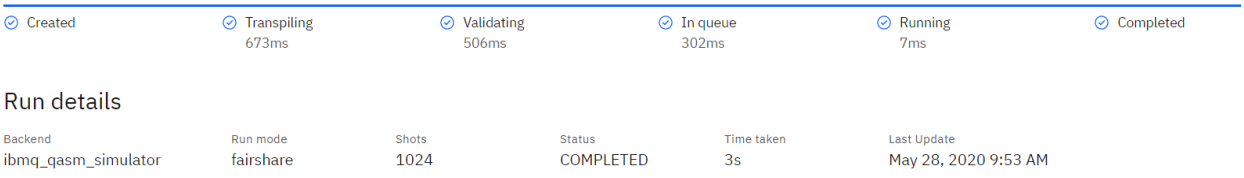

Ilustración 49. Tiempo de procesamiento del algoritmo de Shor en un entorno simulado

Fuente: Propia

La ilustración 48 y 49, con tiene los tiempos de procesamiento para este ciclo. Es apreciable que al igual en todos los demás algoritmos objetos de esta investigación, el tiempo de procesamiento en un procesador real fue 900 veces más lento en comparación con el entorno simulado.

El algoritmo de Shor presenta un grado de dificultad bastante elevado, a tal punto que solo ha sido implementado en 2 ocasiones en procesadores completamente cuánticos, para factorizar el número 15 y 21. Existen variadas opiniones acerca de los resultados obtenidos por este algoritmo, puesto que algunos investigadores afirman que este algoritmo sobre simplifica su función oráculo para un número *a* especifico y así lograr resolver la factorización con una mayor rapidez (Smolin, Smith, & Vargo, 2013). El algoritmo de Shor para factorizar el numero 15 fue implementado en el 2015 en un entorno clásico por medio de simuladores. Los resultados obtenidos para esta implementación concuerdan con los resultados obtenidos en la presente sección, comprobando la veracidad de los resultados experimentales (Candela, 2015).

VI. Conclusiones

1. Si bien aún no es completamente comprobable la superioridad de la computación cuántica, los fenómenos de superposición y entrelazamiento permiten realizar operaciones más simplificadas en comparación con su homólogo clasico. Los ordenadores cuánticos pueden trabajar paralelamente, como es comprobable con el algoritmo de Deutsch, y lograr resolver problemas en una sola iteración, a diferencia de la computación clásica, que simula el paralelismo mediante la multiplexacion.
2. La reversibilidad es una propiedad exclusivamente cuántica, por lo que un algoritmo clásico no podrá ser implementado en un entorno cuántico directamente, pues carece de reversibilidad. Existen algoritmos que se ejecutan en ambos entornos, como la transformada de Fourier, sin embargo se debe diseñar una nueva función para cada entorno.
3. Actualmente, la computación cuántica carece de una factibilidad de implementación inmediata. La misma ha demostrado ser superior a la computación clásica en capacidades de procesamiento a través de los fenómenos cuánticos como la superposición y entrelazamiento, pero los tiempos de ejecución en los procesadores cuánticos actuales aún son más lentos que una computadora clásica. Se deben diseñar chips con mayor estabilidad para así poder aprovechar al máximo las capacidades ofrecidas por la computación cuántica.

4. Aun no existe un procesador cuántico estable y universal, capaz de resolver todo tipo de compuertas con la mayor precisión posible. Es necesario escoger un procesador cuántico basándose en la cantidad y tipo de compuertas a utilizar, y compararlo con la calibración del mismo para analizar su factibilidad. Una cantidad mayor de compuertas significara una cantidad de ruido mayor, por lo que los circuitos se deben simplificar a su menor expresión posible para evitar interferencias y resultados no deseados. Las velocidades de procesamiento en un entorno simulado y un procesador cuántico real son drásticamente diferentes, lo que indica que el ruido y la interpretación software-hardware ha causado mucha más interferencia de la aceptable.
5. La computación cuántica está muy lejos aún de poner en riesgo la criptografía actual, puesto que a pesar que se han logrado factorizar números de 768 bits, esto ha sido logrado simplificando en exceso el circuito a utilizar, y para poder llevarse a cabo en un ambiente completamente cuántico, se requeriría un procesador de 30,002 qubits, siendo el máximo disponible a la fecha de 72 qubits. (Smolin, Smith, & Vargo, 2013).

VII. Recomendaciones

La computación cuántica es un tema que en general aun ha sido estudiado muy poco. El presente documento sienta las bases para la ejecución y comprobación de algoritmos cuánticos en procesadores reales, y será de apoyo para futuras investigaciones en la materia. La implementación y diseño para funciones oráculos especificas deberá ser estudiado a más profundidad en futuras investigaciones. De la misma manera se debe realizar un análisis del impacto de la decoherencia, el ruido, los tiempos de relajación y desfase para identificar la manera que estos factores afectan el tiempo de procesamiento en un procesador real

VIII. Bibliografía

physicsopenlab.org. (2017). Obtenido de http://physicsopenlab.org/2017/05/30/tunnel-effect/

Aczel, A. D. (2004). *Entrelazamiento.*

Bonillo, V. M. (2013). *Principios fundamentales de computacion cuantica.*

Brassard, G., & Bratley, P. (1997). *Fundamentos de algoritmia.*

Candela, D. (2015). Undergraduate computational physics projects on quantum computing.

Carare, V., Carillo de Albornoz, A., & Taylor, J. (2018). Expected optimal time for the NMR implementation of Shor´s algorithm for factorising 15.

Cedano Olvera, M. A., Cedano Rodriguez, A., Rubio Gonzalez, J. A., & Vega Gutierrez, A. C. (2014). *Fundamentos de computacion para ingenieros.*

Cheang, J. (2005). Ley de Moore, Nanotecnologia y Nanociencias: Sintesis y modificacion de nanoparticulas mediante la implantacion de iones. *Revista digital universitaria*.

Cochran, W., Cooley, J., Favin, D., Helms, H., Kaenel, R., Lang, W., y otros. (1967). What is the fast fourier transform?

Coles, P., Eidenbenz, S., Pakin, S., Adedoyin, A., Ambrosiano, J., Anisimov, P., y otros. (2018). Quantum Algorithm Implementations for Beginners.

Dasgupta, S., Papadimitiou, C., & Vazirani, U. (2006). *Algorithms.*

De Vito, E., & Suarez, A. (1999). Jean Baptiste Joseph Fourier: la inesperada armonia del azar.

Devoret, M., & Schoelkopf, R. (2013). *Superconducting Circuits for Quantum Information: An Outlook.*

Dirac, P. (1930). *Principles of Quantum mechanics.*

Einstein, A., Podolsky, B., & Rosen, N. (1935). Can Quantum-Mechanical Description of Physical Reality Be Considered Complete?

Gambetta, J., Chow, J., & Matthias, S. (2017). *Building logical qubits in a superconducting quantum computing system.*

Gangopadhyay, S., Manabputra, Behera, B., & Panigrahi, P. (2018). Generalization and Partial Demonstration of an Entanglement Based Deutsch-Jozsa Like Algorithm Using a 5-Qubit Quantum Computer.

Gruska, J. (2000). *Quantum Computing.*

Gulde, S., Riebe, M., Lancaster, G., Becher, C., Eschner, J., Haffner, H., y otros. (2003). Implementation of the Deutsch-Jozsa algorithm on an ion-trap quantum computer.

Hyperphysics. (s.f.). *Hyperphysics*. Obtenido de http://hyperphysics.phy-astr.gsu.edu/hbase/Solids/Squid.html

IBM. (2016). *IBM*. Obtenido de https://www-03.ibm.com/press/us/en/pressrelease/49661.wss

IBM. (2020). *IBM Quantum Experience*. Obtenido de https://quantum-computing.ibm.com/composer/new-experiment

Kielpinski, D., Monroe, C., & Wineland, D. (2002). *Architecture for a large-scale ion-trap quantum computer.*

Latorre, J. I. (2017). *Cuantica tu futuro en juego.*

Maloney, T. (1983). *Electronica Industrial.*

Mandviwalla, A., Ohshiro, K., & Ji, B. (2018). Implementing Grover's Algorithm on the IBM Quantum Computers.

Mano, M. (1989). *Logica Digital y Diseño de Computadores.*

Mano, M. (2003). *Diseño Digital.*

Miranda, N. D. (2014). *Computacion Cuantica.*

Mishra, N. (2019). *Towards Data Science*. Obtenido de https://towardsdatascience.com/understanding-basics-of-measurements-in-quantum-computation-4c885879eba0

Monroe, C., & Kim, J. (2013). *Scaling the Ion Trap Quantum Processor.*

Muthukrishnan, A. (1999). Classical and quantum gates: An introduction to quantum computing.

Nielsen, M., & Chuan, I. (2010). *Quantum Computation and Quantum Information.*

Salas, P. J. (2008). Noise Effect on Grover algorithm.

Scheel, S., Pachos, J., Hinds., E., & Knight, P. (2004). Quantum Gates and Decoherence.

Scully, M., & Zubairy, M. (2001). Quantum optical implementation of Grover´s algorithm.

Smolin, J., Smith, G., & Vargo, A. (2013). Oversimplifying quantum factoring.

Vartiainen, J., Niskanen, A., Nakahara, M., & Salomaa, M. (2004). Implementig Shor's algorithm on Josephson Charge Qubits.

Yanofsky, N., & Mannucci, M. (2008). *Quantum Computing for Computer Scientists.* Cambridge.