Facultad de Ciencias Fisíco-Matemáticas

Aritmética Cuántica Eficiente: Sumadores basados en QFT y Métodos Clásicos

Tesis para obtener el título de

Licenciado en Fisíca Aplicada

PRESENTA

Ian Medina Cancela

DIRECTOR (es) DE TESIS

Dr. Luis Manuel Arévalo Aguilar

H. Puebla de Z., Agosot 2025

Agradecimientos

Quiero agradecer a mis padres que siempre me han apoyado, que siempre han creido en mi aun cuando yo no lo hago, que siempre me han motivado a seguir, a mi hermano que siempre esta conmigo que me apoya y me impulsa a que sea yo mismo, a mis amigos que me han hecho disfrutar esta etapa universitaria como nunca, y al Dr. Luis Manuel Arévalo Aguilar por permitirme trabjar con el y desarrollarme en el ambito academico en una area tan interesante y compleja.

Índice general

1. Resumen **1**

2. Introducción **3**
2.1. Antecedentes 3
2.2. Planteamiento del problema 4
2.3. Objetivos 4
2.4. Hipótesis 5
2.5. Introducción a la computación cuántica 5
2.5.1. ¿Qué es la computación cuántica? 5
2.5.2. Fundamentos teóricos 6
2.5.3. Qubit 7
2.5.4. Compuertas lógico cuánticas 9
2.5.5. Rotaciones 15
2.6. Complejidad Computacional Clásica en el Contexto de la Computación Cuántica 16
2.6.1. Fundamentos de la Complejidad Computacional Clásica 17
2.6.2. Notación Asintótica y Clases de Complejidad 17
2.6.3. Importancia para la Computación Cuántica 18
2.6.4. Ejemplo Ilustrativo Clásico 18
2.7. Desigualdad de Helstrom 19
2.7.1. Interpretación física 19
2.8. Aproximación cuadrática de baja fase 20
2.8.1. Planteamiento 20
2.8.2. Sustitución en el límite de Helstrom 20
2.9. Suma binaria 21
2.9.1. Ejemplo suma binaria 21
2.10. Python 21
2.10.1. Qiskit 22

3. Algoritmos Cuánticos **23**
3.1. Transformada de Fourier Cuántica 23
3.1.1. Definición matemática: 23
3.1.2. Implementación en un circuito cuántico: 24
3.1.3. Aplicación en Python 25
3.1.4. Análisis de complejidad 26
3.2. Algoritmo de Grover 27
3.2.1. Planteamiento del problema: 27
3.2.2. Fundamentos matemáticos: 28
3.2.3. Explicación geométrica: 29
3.2.4. Complejidad y ventajas: 29

4. Diseño y Comparación de Sumadores Cuánticos **31**
4.1. Sumador Clásico 31
4.1.1. Operación OR Lógica 31
4.1.2. Sumador Completo Full-Adder 31
4.1.3. Análisis de complejidad de la suma clásica: 33
4.2. Sumador Cuántico Simple 34
4.2.1. Implementación en Qiskit: 36
4.2.2. Análisis de complejidad 39
4.3. Sumador Cuántico con QFT 40
4.3.1. Implementación Qiskit 43
4.3.2. Análisis de complejidad 45
4.4. Suma cuántica con QFT en paralelo 46
4.4.1. Análisis de la estructura de las puertas controladas . . 47
4.4.2. Principio de paralelización 48
4.4.3. Formalización matemática de la paralelización 48
4.4.4. Aplicación en un circuito cuántico 50
4.4.5. Aplicación en Python 51
4.5. Aproximación de rotaciones 52
4.5.1. Fundamento matemático de la aproximación 52
4.5.2. Regla de truncado 52
4.5.3. Suma de ángulos descartados 53
4.5.4. De ángulos a norma operador 54
4.5.5. Probabilidad de error al medir 54
4.5.6. Regla Mnemónica 54
4.5.7. Relación de ϵ con las puertas logico-cuánticas 55
4.5.8. Análisis de complejidad 55

4.6. Comparación de sumadores cuánticos 56
4.7. Aplicación de Sumadores Cuánticos en el Algoritmo de Grover 58
4.7.1. Sumador Cuántico Simple 58
4.7.2. Sumador Cuántico con QFT 59
4.7.3. Suma Cuántica con QFT en Paralelo 60
4.7.4. Sumador con aproximación de rotaciones 60

5. Conclusión 61

A. Librerias utilizadas para las simulaciones 63

B. Programa QFT 65
B.1. Funciones 65
B.2. Llamado de funciones 65

C. Funciones generales de los sumadores 67

D. Sumador Cuántico Simple 69
D.1. Funciones para realizar la suma 69
D.2. Llamado de las funciones para la suma cuántica basica 70

E. Sumador Cuántico con QFT en serie 73
E.1. Funciones para realizar la suma 73
E.2. Llamado de funciones para realizar la suma 75

F. Sumador Cuántico con QFT en paralelo 77

G. Analisis de propiedades del suamdor con rotaciones truncadas 79

H. Bibliografía 83

Capítulo 1

Resumen

Este trabajo ofrece un recorrido completo por la **suma cuántica** y su importancia en la computación cuántica moderna. Comienza con una presentación de los qubits, la notación de Dirac, la esfera de Bloch y las puertas lógicas básicas, estableciendo la diferencia entre bits clásicos y qubits en superposición. Después se analiza la *Transformada de Fourier Cuántica* (QFT), se detalla su circuito, la descomposición paso a paso y su complejidad $O(n^2)$ tanto en número de puertas como en profundidad, señalando que no necesita ancillas, pero sí rotaciones de fase muy precisas.

A partir de este marco se describen cuatro **sumadores cuánticos**:

- **VBE (Vedral-Barenco-Ekert).** Se llama así ya quwe fue propuetso por Vlatko Vedral, Adriano Barenco y Artur Ekert. Utiliza puertas Toffoli y CNOT (Control-Not); presenta profundidad lineal y requiere $3n + 1$ qubits debido al registro de acarreo.

- **QFT secuencial.** Rodea un operando con QFT y QFT^{-1}; emplea únicamente $2n$ qubits, pero incrementa puertas y profundidad a $O(n^2)$.

- **QFT paralelizada.** Reorganiza las rotaciones controladas en capas sin colisiones de control u objetivo, de modo que la profundidad se reduce a $O(n)$ manteniendo el conteo cuadrático de puertas.

- **QFT aproximada (AQFT).** Descarta rotaciones cuyo ángulo sea menor que ε. Con la regla $\varepsilon = \pi/(2n)$, el número de puertas baja a $O(n \log n)$ y la profundidad paralela a $O(\log n)$. La desigualdad de

Helstrom, junto con la expansión cuadrática en fase baja, garantiza que la probabilidad de error resultante queda acotada por $\varepsilon^2/4$.

El presente trabajo compara estos sumadores en cuanto a qubits, puertas y robustez. Destaca que el VBE favorece la simplicidad; el QFT secuencial optimiza memoria a costa de tiempo; la versión paralela minimiza la latencia, y la AQFT ofrece el mejor balance global.

Finalmente, se concluye que la combinación de truncado y paralelización convierte a la QFT en un bloque competitivo para hardware NISQ(Noisy Intermediate-Scale Quantum), define el estado actual de la computación cuántica. Se refiere a los ordenadores cuánticos que poseen un número limitado de qubits (escala intermedia) y están sujetos a altas tasas de error debido a la decoherencia y el ruido (ruidoso), lo que impide la aplicación de corrección de errores a gran escala. Se proponen lineamientos prácticos: utilizar VBE en demostraciones y dispositivos con qubits abundantes, reservar el QFT secuencial para escenarios donde la memoria es crítica y priorizar la AQFT o el esquema paralelo en aplicaciones reales con tiempos de coherencia limitados. Así, se sientan las bases para incorporar aritmética eficiente en algoritmos como Grover, en búsquedas con condiciones aritméticas y en rutinas más complejas, fortaleciendo la viabilidad de escalar operaciones de suma dentro de la computación cuántica actual.

Capítulo 2

Introducción

2.1. Antecedentes

En las últimas décadas, la computación cuántica ha surgido como una alternativa prometedora a la computación clásica, desafiando los límites establecidos por modelos clásicos. Mediante la explotación de fenómenos cuánticos como lo son la superposición, la interferencia y el entrelazamiento.

Uno de los componentes fundamentales de cualquier arquitectura computacional, clásica o cuántica, es la aritmética binaria, y dentro de ella, la operación de la suma representa una base sobre la cual se construyen algoritmos más complejos. En la computación cuántica, diseñar circuitos que realicen la suma binaria de manera más eficiente trae consigo desafíos únicos, ya que las operaciones realizadas en el circuito deben ser reversibles y mantener la coherencia cuántica. Además, no es posible desechar información como en los sitemas clásicos, debido a la naturaleza lineal y unitaria de como evoluciona un sistema cuántico, lo que exige utilizar estructura como las ancillas o técnicas de descomposición que permiten realizar la suma sin violar los principios de la mecánica cuántica. Este requisito implica un diseño cuidadoso del circuito, así como optimización de los qubits utilizados y la profundidad del circuito.

En este contexto, diversas propuestas han surgido para implementar la suma cuántica de manera eficiente, las más destacadas son el **sumador Cuccaro et al** o **sumador VBE** y los esquemas basados en la transformada de Fourier cuántica (QFT). Estos enfoques ofrecen distintos compromisos entre

número de puertas lógico cuánticas, cantidad de qubits y profundidad del circuito. Mientras que el sumador VBE es más directo y se adapta bien a la arquitectura lógica reversible, la versión con QFT aprovecha la diagonalización de la operación de la suma para reducir la profundidad, aunque con una mayor sensibilidad a errores. El estudio detallado de estas variantes es relevante no solo para el diseño de la aritmética cuántica, si no también para comprender cómo construir bloques fundamentales reutilizables en algoritmos cuánticos más complejos.

2.2. Planteamiento del problema

En la computación cuántica, la suma es una operación fundamental. Tradicionalmente, se han utilizado sumadores reversibles como el sumador de *arrastre de acarreo* (ripple-carry adder), a veces conocido por sus siglas en inglés o por el nombre de sus diseñadores como VBE (Vedral, Barenco, Ekert). Aunque son sencillos de diseñar, estos sumadores tienen una limitación importante: su profundidad de circuito crece linealmente con el número de qubits. Esto significa que para sumar números más grandes, el circuito se vuelve progresivamente más profundo, lo que aumenta el tiempo de ejecución y la susceptibilidad a errores (decoherencia), limitando su uso en algoritmos cuánticos complejos.

Por otro lado, la suma basada en la Transformada Cuántica de Fourier (QFT) ofrece una solución más eficiente. En lugar de sumar bit a bit en el dominio convencional, esta técnica traslada la operación al dominio de Fourier. Al operar en este *espacio de Fourier*, la suma se puede realizar con una profundidad de circuito mucho menor, que no depende linealmente del número de qubits. Esta menor profundidad hace que los sumadores basados en QFT sean significativamente más escalables y robustos, convirtiéndolos en la opción preferida para algoritmos cuánticos avanzados que requieren alta precisión y eficiencia.

2.3. Objetivos

El objetivo central es mostrar y validar que la implementación de la suma mediante QFT es más eficiente y escalable en el contexto de algoritmos

cuánticos de gran escala, facilitando así la construcción de bloques aritméticos reutilizables en procesos cuánticos avanzados. De este modo, se busca identificar las ventajas y limitaciones prácticas de la suma con QFT frente a otros sumadores reversibles como el VBE.

2.4. Hipótesis

Si bien existen diversos esquemas para realizar la suma cuántica, entre ellos el sumador cuántico simple (basado en puertas Toffoli/CNOT), el sumador con Transformada de Fourier Cuántica (QFT) secuencial y el sumador con QFT paralelizada son más eficientes que el sumador cuántico simple, y existe un metodo de aproximación que maximiza la eficiencia del sumador.

2.5. Introducción a la computación cuántica

2.5.1. ¿Qué es la computación cuántica?

La computación cuántica es un paradigma emergente en la computación que busca aprovechar los principios de la mecánica cuántica para procesar información de formas más eficientes en comparación con un ordenador clásico en ciertos problemas específicos. La idea de usar los principios de la mecánica cuántica para el desarrollo de nuevos paradigmas de la computación comenzó en la década de 1980, empezando con Richard Feynman (1982) quien, en la conferencia sobre Física de la Computación, propuso que un sistema cuántico podría ser simulado de manera eficiente solo por otro sistema cuántico. Esta idea sentó las bases para el desarrollo de los primeros modelos de computación cuántica, los cuales fueron formalizados gracias a David Deutsch (1985) cuando, en su artículo *Quantum theory, the Church–Turing principle and the universal quantum computer*, afirmó que "al igual que una máquina de Turing, un modelo de computador cuántico Q consta de dos componentes: un procesador finito y una memoria infinita" (p. 103), mostrando evidencia de que los qubits podrían realizar cálculos más allá de lo que podría un bit clásico. Posteriormente, con la creación de algoritmos cuánticos como el algoritmo de Shor (1994) para la factorización de números primos y el algoritmo de Grover (1996) para la búsqueda en bases de datos no estructuradas, se evidenció la ventaja potencial de la computación cuántica sobre la clásica en ciertos problemas específicos.

En 2012 John Perskill acuño el término "Supremacia cuántica"para referirse al momento en que una computadora cuántica pueda realizar una tarea la cual sea prohibitiva en tiempo para la mejor super computadora clásica existente en el momento. Este logro es importante ya que marca la transición de un estudio puramente teórico a uno comprobable experimentalmente.

La primer diferencia entre la computación cuántica y la clásica más alla de los fundamentos de mecánica cuántica es que en lugar de utilizar bits convencionales, que pueden representar los estados binarios 0 o 1, los sistemas cuánticos emplean qubits, que pueden existir en una superposición de ambos estados simultáneamente, lo que habilita el procesamiento paralelo de información.

En este contexto, el presente trabajo se enfoca en el diseño, implementación y optimización de algoritmos cuánticos para la operación de suma, explorando distintas configuraciones de circuitos para mejorar su eficiencia en términos de profundidad de circuito, número de qubits y consumo de compuertas lógicas. Para ello, se empleará el marco de desarrollo Qiskit, que permite la simulación y ejecución de circuitos cuánticos en hardware real.

2.5.2. Fundamentos teóricos

Hay tres principios esenciales de la mecánica cuántica en la que se fundamenta este nuevo paradigma:

- **Superposicion cuántica:** En el contexto de la computación cuántica la superposición nos indica que un qubit puede encontrarse en una combinación lineal de los estados $|0\rangle$ y $|1\rangle$, lo que nos permite que un sistema represente varias posibilidades simultaneamente.

- **Entrelazamiento cuántico:** Dos o más qubits se pueden correlacionar de tal forma que el estado de uno de los qubits determina instantaneamente los demas estados, sin importar la distancia en que se encuentren entre ellos. Esto permite una comunicación y procesamiento más eficiente.

- **Interferencia cuántica:** La naturaleza ondulatoria de la mecánica cuántica permite que se manipulen estados de los qubits para favorecer soluciones y suprimir resultados que contribuyen negativamente al resultado deseado.

2.5.3. Qubit

Comparando un ordenador cuántico y uno clásico, encontramos la primer diferencia, en la unidad de medida básica de información. Ya que en una computadora clásica se manejan los bits que solo pueden tomar valores de 0 o 1 en un momento dado. En cambio, un ordenador cuántico funciona con qubits que son análogos a los bits, con la diferncia de que estos pueden tomar los valores $|0\rangle$, $|1\rangle$ o una superposición de ambos estados:

$$|\psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle\,, \tag{2.1}$$

donde α y β representan la amplitud de probabilidad y deben cumplir la condición de normalización:

$$|\alpha|^2 + |\beta|^2 = 1, \tag{2.2}$$

esto nos indica que la probabilidad de encontrar al qubit con el valor de $|0\rangle$ es de $|\alpha|^2$ y $|1\rangle$ en $|\beta|^2$ y la suma de probabilidades debe ser 1.

Para denotar un estado de un sistema cuántico se utiliza la notación de Dirac. Dicha notación emplea la estructura, $|\psi\rangle$, que se llama ket. Esta notación cuando se quiere representar estados como $|0\rangle$, $|1\rangle$, $|2\rangle$ ... $|k\rangle$, donde $k \leq 2^n - 1$, con n siendo el número de qubits. Específicamente en computación empleando la expansión binaria, por ejemplo el número a=2 en binario es 10 y en notación de Dirac $|a\rangle = |10\rangle$.

La notación de Dirac es las forma más común de representar un qubit, sin embargo, también se puede hacer en forma de vector columna.

$$|0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \qquad 1 = \begin{pmatrix} 0 \\ 1 \end{pmatrix}. \tag{2.3}$$

Esto implica que el estado general de un qubit, se puede representar por el siguiente vector columna:

$$\begin{bmatrix} \alpha \\ \beta \end{bmatrix}. \tag{2.4}$$

La forma geométrica de representar un qubit es mediante la esfera de Bloch, la cual nos permite visualizar sus posibles estados como puntos en la superficie de una esfera unitaria en el espacio tridimensional.

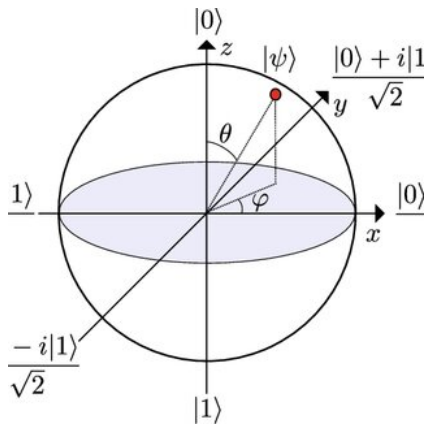


Figura 2.1: Esfera de Bloch

En esta representación en vez de utilizar α y β, se expresa el qubit en coordenadas esféricas con dos ángulos que son θ y ϕ, por lo que un estado cuántico en esta esfera se representa matemáticamente como:

$$|\psi\rangle = \cos\frac{\theta}{2}|0\rangle + e^{i\phi}\sin\frac{\theta}{2}|1\rangle\,, \tag{2.5}$$

donde θ que va desde 0 a π controla la proporción entre los estados $|0\rangle$ y $|1\rangle$, y ϕ va de 0 a 2π representa la fase relativa entre los estados base.

Multiples qubits

Cuando un sistema cuántico esta conformado por multiples qubits, a este conjunto se le denomina registro cuántico, que es utilizado para codificar información más compleja, por ejemplo un sistema conformado por dos qubits, puede tomar los estados $|00\rangle$, $|01\rangle$,$|10\rangle$ y $|11\rangle$ o en una superposición de estos cuatro estados.

$$\alpha_{00}|00\rangle + \alpha_{01}|01\rangle + \alpha_{10}|10\rangle + \alpha_{11}|11\rangle\,, \tag{2.6}$$

donde de manera matricial se representa como:

$$\begin{bmatrix} \alpha_{00} \\ \alpha_{01} \\ \alpha_{10} \\ \alpha_{11} \end{bmatrix}. \tag{2.7}$$

Dichas amplitudes de probabilidad deben cumplir la condición de normalización. De forma general, el estado de un registro con n qubits será la superposición normalizada de los 2^n estados clásicos y se representa con la siguiente ecuación:

$$|\psi\rangle = \sum_{k=0}^{2^n-1} \alpha_k |k\rangle . \tag{2.8}$$

2.5.4. Compuertas lógico cuánticas

Las puertas lógico cuánticas equivalen a las puertas lógicas clásicas, con la diferencia de que las puertas cuánticas actuan sobre qubits utilizando las propiedades de la mecánica cuántica, y tienen tres propiedades principales que son:

- **Reversibilidad:** Ya que todas las puertas cuánticas son unitarias, existe una puerta cuántica inversa, a diferencia de las puertas lógicas clásicas como AND y OR que no son reversibles, y en computación cuántica todas las puertas deben ser reversibles.

- **Matriz unitaria:** Cada una de las puertas cuánticas pueden representarse como una matriz unitaria U que cumple con la siguiente propiedad:

$$U^{\dagger}U = I, \tag{2.9}$$

  con $U^{\dagger}$ es la matriz transpuesta conjugada y I es la matriz identidad.

- **Transformación lineal:** Las puertas cuánticas transforman un estado cuántico inicial en un nuevo estado de manera lineal.

Las puertas cuánticas fundamentales son las siguientes:

Puerta de Pauli-X (X):

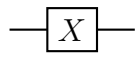
Es el equivalente a la puerta NOT en la computación clásica, intercambia los valores de $|0\rangle$ con $|1\rangle$ y viceversa, su representación matricial es la siguiente:

$$X = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}, \tag{2.10}$$

su efecto sobre los estados base es:

$$X\left|0\right\rangle = \left|1\right\rangle \qquad\qquad X\left|1\right\rangle = \left|0\right\rangle, \tag{2.11}$$

en un circuito cuántico se representa de la siguiente forma:



Puerta de Pauli-Y (Y):

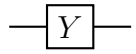
Esta puerta es similar a X pero introduce una fase imaginaria, que es representada por la siguiente matriz:

$$\begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix}, \tag{2.12}$$

su efecto sobre los estados base son los siguientes:

$$Y\left|0\right\rangle = i\left|1\right\rangle \qquad\qquad Y\left|1\right\rangle = -i\left|0\right\rangle, \tag{2.13}$$

la representación de esta puerta en un circuito cuántico es:



Puerta de Pauli-Z (Z):

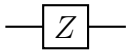
Esta puerta cuántica introduce un cambio de fase al estado $\left|1\right\rangle$, y se representa mediante la siguiente matriz:

$$Z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}, \tag{2.14}$$

su efecto sobre los estados base son:

$$Z\left|0\right\rangle = \left|0\right\rangle \qquad\qquad Z\left|1\right\rangle = -\left|1\right\rangle, \tag{2.15}$$

la representación en un circuito cuántico es:



Puerta de Hadamard (H):

Es de las puertas cuánticas más importantes, ya que al utilizarla crea una superposición de los estados $|0\rangle$ y $|1\rangle$ con la misma probabilidad de obtener uno u otro, y su forma matricial es la siguiente:

$$H = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}, \tag{2.16}$$

su efecto en los estados base son:

$$H\,|0\rangle = \frac{1}{2}(|0\rangle + |1\rangle) \qquad\qquad H\,|1\rangle = \frac{1}{2}(|0\rangle - |1\rangle), \tag{2.17}$$

la representación de esta puerta logico-cuántica en un circuito es:

Puertas multiples qubits

Hasta ahora solo se han mencionado puertas cuánticas que actuan sobre un solo qubit, pero también existen puertas que operan utilizando varios qubits, en este caso mencionaremos solo las utilizadas para los algoritmos que se van a mostrar más adelante.

Puerta C-NOT:

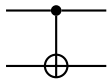
La puerta $C - NOT$ (Controled-NOT) o CX (Controled-X) actúa sobre dos qubits:

- El primer qubit es el qubit control.
- El segundo qubits es el qubit objetivo o target.
- Si el qubit control se encuentra en el estado $|1\rangle$, se aplica la puerta X al qubit objetivo.
- Si el qubit control se encuentra en el estado $|0\rangle$, el qubit objetivo queda intacto.

La matriz de la puerta CX en la base computacional $\{|00\rangle, |01\rangle, |10\rangle, |11\rangle\}$ es:

$$CX = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}, \tag{2.18}$$

en un circuito cuántico se representa de la siguiente manera:



Puerta CC-NOT o Toffoli:

La puerta $CC - NOT$ (también llamada Toffoli) es una extensión de la puerta C-NOT que opera sobre tres qubits:

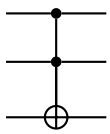
- Dos qubits control.
- Un qubit objetivo.

El qubit objetivo se invierte (X) solo si ambos qubits de control están en el estado $|1\rangle$.

La representación matricial de la puerta Toffoli es:

$$CCX = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}, \tag{2.19}$$

su representación en un circuito cuántico es:



Controled-Z:

La puerta $Controled - Z$ o CZ se usa para cambiar la fase del estado objetivo de $|x\rangle$ a $-|x\rangle$.

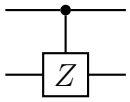
- Hay un qubit control.
- Un qubit objetivo.

En la base computacional $\{|00\rangle, |01\rangle, |10\rangle, |11\rangle\}$, la puerta CZ se representa mediante la siguiente matriz:

$$CZ = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \end{bmatrix}, \tag{2.20}$$

esta matriz muestra que todos los estados base permanecen inalterados, excepto $|11\rangle$, cuya fase se invierte.

En un circuito cuántico se representa como:



Controled-Controled-Z:

Puerta cuántica que funciona con tres qubits, también se puede llamar como CCZ es una extensión de la puerta CZ.

- Hay dos qubits control.
- Un qubit objetivo.

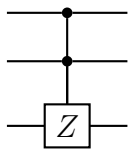
La puerta CCZ aplica una transformación de fase -1 solo cuando los dos qubits de control están en el estado $|1\rangle$, mientras que deja inalterados los demás estados de la base computacional.

En la base computacional $\{|000\rangle, |001\rangle, |010\rangle, |011\rangle, |100\rangle, |101\rangle, |110\rangle, |111\rangle\}$, la puerta CCZ se representa mediante la siguiente matriz:

$$CCZ = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 \end{bmatrix} \tag{2.21}$$

Esta matriz indica que todos los estados base permanecen inalterados, a excepción del estado $|111\rangle$, cuya fase se invierte.

En un circuito cuántico la puerta CCZ se representa como:



Puerta SWAP

La puerta SWAP es una compuerta cuántica de dos qubits que intercambia los estados de ambos qubits. Es una operación completamente reversible y unitariamente implementable. Si los qubits están en los estados $|a\rangle$ y $|b\rangle$, entonces:

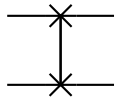
$$\text{SWAP}|a, b\rangle = |b, a\rangle. \tag{2.22}$$

En la base computacional $\{|00\rangle, |01\rangle, |10\rangle, |11\rangle\}$, la puerta SWAP se representa mediante la siguiente matriz unitaria de 4×4:

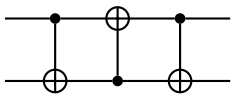
$$\text{SWAP} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1, \end{pmatrix} \tag{2.23}$$

esta matriz intercambia las amplitudes de los estados $|01\rangle$ y $|10\rangle$.

La puerta SWAP se puede representar en circuitos cuánticos de la siguiente manera:



También puede implementarse usando tres puertas CNOT:



Este es un método estándar para descomponer la SWAP en puertas CNOT, útil para plataformas cuánticas donde las puertas CNOT están disponibles pero la SWAP no es nativa, es decir que un computador cuántico no puede ejecutarla directamente una sola operación física.

2.5.5. Rotaciones

En la computación cuántica, las rotaciones son operaciones unitarias que afectan el estado de un qubit al modificar la orientación del mismo en la esfera de Bloch, dichas rotaciones son representadas mediante puertas cuánticas de rotación que son matrices unitarias en el espacio de Hilbert en dos dimensiones.

Las rotaciones se realizan mediante un operador de la forma:

$$R_\theta = e^{-i\theta G}, \tag{2.24}$$

donde:

- G representa un operador de Pauli(X,Y,Z) que define el eje de rotación.
- θ ángulo de rotación.

Las puertas de rotación más comunes son:

Rotación en el eje X($R_x(\theta)$):

$$R_X(\theta) = \cos\left(\frac{\theta}{2}\right) I - i \sin\left(\frac{\theta}{2}\right) X, \tag{2.25}$$

su representación matricial es:

$$\begin{bmatrix} \cos(\theta/2) & -i\sin(\theta/2) \\ -i\sin(\theta/2) & \cos(\theta/2) \end{bmatrix}. \tag{2.26}$$

Rotación en el eje Y($R_Y(\theta)$):

$$R_Y(\theta) = \cos\left(\frac{\theta}{2}\right) I - i\sin\left(\frac{\theta}{2}\right) Y, \tag{2.27}$$

en matriz se representa como:

$$\begin{bmatrix} \cos(\theta/2) & -\sin(\theta/2) \\ \sin(\theta/2) & \cos(\theta/2) \end{bmatrix}. \tag{2.28}$$

Rotación en el eje Z($R_Z(\theta)$):

$$R_Z(\theta) = \cos\left(\frac{\theta}{2}\right) I - i\sin\left(\frac{\theta}{2}\right) Z. \tag{2.29}$$

Matricialmente se representa como:

$$\begin{bmatrix} e^{-i\theta/2} & 0 \\ 0 & e^{i\theta/2} \end{bmatrix}. \tag{2.30}$$

Rotaciones arbitrias:

Cualquier rotación en la esfera de Bloch puede ser expresada en términos de las rotaciones básicas:

$$U(\theta, \phi, \lambda) = R_Z(\phi) R_Y(\theta) R_Z(\lambda). \tag{2.31}$$

Esto es importante hablando de hardware cuántico ya que las puertas para un solo qubit se pueden descomponer en términos de $U(\theta, \phi, \lambda)$.

2.6. Complejidad Computacional Clásica en el Contexto de la Computación Cuántica

La complejidad computacional es un área fundamental de la teoría de la computación que estudia los recursos necesarios para resolver problemas computacionales, particularmente el tiempo y espacio requeridos por los algoritmos. En el contexto de la computación cuántica, es esencial comprender

primero el marco clásico de la complejidad computacional, ya que la comparación con los modelos cuánticos permite identificar las potenciales ventajas y limitaciones de este nuevo paradigma.

2.6.1. Fundamentos de la Complejidad Computacional Clásica

La complejidad computacional clásica evalúa el costo computacional de resolver problemas en función del tamaño de la entrada, denotado como n, utilizando modelos deterministas tradicionales, tales como las máquinas de Turing clásicas. Los recursos computacionales usualmente considerados son:

- **Tiempo de ejecución**: cantidad de pasos computacionales o instrucciones necesarias para completar un algoritmo.
- **Espacio de memoria**: cantidad de memoria utilizada durante la ejecución.

La cuantificación de estos recursos permite clasificar los problemas según su dificultad computacional y la eficiencia de los algoritmos que los resuelven.

2.6.2. Notación Asintótica y Clases de Complejidad

Para caracterizar el crecimiento del consumo de recursos conforme aumenta el tamaño de la entrada, se utiliza la notación Big O, que provee una cota superior asintótica. Por ejemplo:

$$O(1), \quad O(\log n), \quad O(n), \quad O(n \log n), \quad O(n^k), \quad O(2^n), \ldots$$

Dentro del paradigma clásico, las principales clases de complejidad que describen la dificultad de los problemas son:

- **P (Polynomial time)**: problemas que pueden resolverse en tiempo polinomial mediante un algoritmo determinista, es decir el tiempo que tarda un algoritmo en verificar si una solución propuesta es correcta. Son considerados eficientes y prácticos desde la perspectiva clásica.

- **NP (Nondeterministic Polynomial time)**: problemas para los cuales, aunque no se conozca un algoritmo determinista eficiente para su solución, se puede verificar una solución propuesta en tiempo polinomial.

- **NP-completo**: subconjunto de NP que contiene los problemas más difíciles de esta clase, es decir, que si uno de estos problemas se resolviera en tiempo polinomial, todos los problemas en NP también lo serían.

- **NP-hard**: problemas al menos tan difíciles como los NP-completos, sin necesariamente pertenecer a NP.

La relación y posible equivalencia entre P y NP constituye uno de los problemas abiertos más importantes en la ciencia de la computación.

2.6.3. Importancia para la Computación Cuántica

La comprensión de la complejidad computacional clásica es crucial para valorar el impacto y alcance de la computación cuántica. Los modelos cuánticos introducen nuevas clases de complejidad, como **BQP (Bounded-Error Quantum Polynomial time)**, que describen los problemas resolubles eficientemente por computadoras cuánticas. En ciertos casos, la computación cuántica puede ofrecer mejoras significativas en complejidad temporal, como ocurre en algoritmos emblemáticos (por ejemplo, el algoritmo de Shor para factorización).

Por lo tanto, analizar los límites y capacidades de la complejidad clásica es indispensable para contextualizar los avances cuánticos y determinar qué problemas pueden beneficiarse de la computación cuántica y cuáles permanecen intratables.

2.6.4. Ejemplo Ilustrativo Clásico

Considerando el problema de búsqueda en una estructura de datos:

- La búsqueda lineal, que recorre secuencialmente cada elemento, presenta complejidad temporal $O(n)$, es decir que el tiempo de ejecució, crece en proporción directa al tamaño de la entrada.

- La búsqueda binaria, aplicable a listas ordenadas, reduce el espacio de búsqueda a la mitad en cada iteración, logrando una complejidad $O(\log n)$.

Estos ejemplos reflejan cómo el diseño algorítmico influye directamente en la eficiencia, un aspecto que la computación cuántica puede optimizar para ciertos problemas específicos.

2.7. Desigualdad de Helstrom

La desigualdad de Helstrom (también llamada límite de Helstrom o Helstrom bound) es el resultado fundamental que indica cuál es la menor probabilidad posible de error al decidir, mediante una medición cuántica, cuál de dos estados de densidad ρ_0 o ρ_1 ha sido preparado..

Helstrom demostró que, sea cual sea la medición, la **probabilidad mínima de error** está acotada por:

$$P_{\text{err}}^{\text{mín}} \;=\; \frac{1}{2}\Big(1 - \|\eta_1\rho_1 - \eta_0\rho_0\|_1\Big), \tag{2.32}$$

donde $\|X\|_1 = \operatorname{tr}\sqrt{X^\dagger X}$ es la *traza-norma* (o distancia traza) del operador X.

- Cuando los estados son *equiprobables* ($\eta_0 = \eta_1 = 1/2$) la expresión se reduce a:

$$P_{\text{err}}^{\text{mín}} \;=\; \tfrac{1}{2}\big(1 - \tfrac{1}{2}\|\rho_1 - \rho_0\|_1\big). \tag{2.33}$$

La **medición** que alcanza este límite se denomina *medición de Helstrom*. Para dos estados puros $|\psi_0\rangle$ y $|\psi_1\rangle$ equiprobables, consiste sencillamente en proyectar sobre el subespacio generado por la diferencia $|\psi_0\rangle - |\psi_1\rangle$.

2.7.1. Interpretación física

La desigualdad de Helstrom afirma que existe un límite insuperable a la hora de decidir —con una sola copia y una sola medición— si el sistema se preparó en el estado ρ_0 o en ρ_1. Ese límite viene dado por la ecuacion (2.33), donde la norma-traza $\|\cdot\|_1$ mide cuánta "separación estadística" puede

extraerse, bajo *cualquier* experimento, entre los dos posibles estados. En otras palabras: **no importa cuán sofisticada sea tu medición, nunca bajarás de esa probabilidad de error**.

2.8. Aproximación cuadrática de baja fase

2.8.1. Planteamiento

Consideremos dos estados puros de un qubit separados por una fase muy pequeña $\Delta\phi \ll 1$:

$$|\psi_0\rangle = |0\rangle\,, \tag{2.34}$$

$$|\psi_1\rangle = \cos\tfrac{\Delta\phi}{2}\;|0\rangle + \sin\tfrac{\Delta\phi}{2}\;|1\rangle\,. \tag{2.35}$$

Expandiendo *hasta segundo orden* en $\Delta\phi$:

$$\sin\tfrac{\Delta\phi}{2} \approx \tfrac{\Delta\phi}{2}, \qquad\qquad \cos\tfrac{\Delta\phi}{2} \approx 1 - \tfrac{\Delta\phi^2}{8}. \tag{2.36}$$

La diferencia de matrices densidad se aproxima por:

$$\rho_1 - \rho_0 \;\approx\; \tfrac{\Delta\phi}{2}\big(|1\rangle\langle 0| + |0\rangle\langle 1|\big) \;=\; \tfrac{\Delta\phi}{2}\,\sigma_x. \tag{2.37}$$

Por tratarse de un operador puro-trazado, su norma traza es simplemente la suma de los valores absolutos de sus autovalores, de modo que:

$$\|\rho_1 - \rho_0\|_1 \;=\; \Delta\phi + O(\Delta\phi^3). \tag{2.38}$$

2.8.2. Sustitución en el límite de Helstrom

Empleando la ecuación (2.38) en la forma equiprobable del límite (2.33) obtenemos:

$$P_{\text{err}}^{\text{mín}} \;=\; \tfrac{1}{2}\Big(1 - \tfrac{\Delta\phi}{2}\Big) + O(\Delta\phi^3) \;=\; \tfrac{1}{2} - \frac{\Delta\phi}{4} + O(\Delta\phi^3). \tag{2.39}$$

Esto muestra que el exceso de éxito sobre el azar crece linealmente con la fase pequeña.

2.9. Suma binaria

La suma binaria es la operación fundamental en sistemas digitales y de cómputo, donde los números están representados en base 2, es decir, utilizando solo dos dígitos o bits: 0 y 1. Esta operación es la base de la aritmética en computadoras y circuitos digitales, como los sumadores en microprocesadores.

En computación, los números están representados en binario, se siguen reglas similares que la suma decimal con algunas reglas específicas para la base 2.

Las reglas de la suma binaria son:

Bit A	**Bit B**	**Suma (Resultado)**	**Acarreo**
0	0	0	0
1	0	1	0
0	1	1	0
1	1	0	1

Cuadro 2.1: Reglas básicas de la suma binaria de un bit

Si el resultado es mayor a 1, se genera un bit de acarreo que se propaga a los siguientes bits.

2.9.1. Ejemplo suma binaria

Tomemos los números 10 (1010) y 15 (1111) para ver el funcionamiento de la suma:

$$\begin{array}{r} 1010 \\ +1111 \\ \hline 11001 \end{array}$$

2.10. Python

Python es un lenguaje de programación de alto nivel, interpretado y de propósito general que se ha consolidado como una de las herramientas más importantes en la investigación y desarrollo de la computación cuántica. Su

popularidad se debe a varias características que lo hacen particularmente adecuado para esta disciplina emergente.

Para la simulación de algoritmos cuánticos se utilizaron las siguientes librerias:

- *Numpy* para operaciones con matrices y matemáticas.
- *Qiskit* para los circuitos cuánticos.

2.10.1. Qiskit

Qiskit es un framework de código abierto desarrollado por IBM que permite diseñar, simular y ejecutar algoritmos cuánticos sobre hardware clásico y simuladores cuánticos. Su diseño modular y accesible ha posicionado a Qiskit como una herramienta fundamental para la investigación, educación y desarrollo en computación cuántica.

Qiskit esta estructurado en varios módulos, en este caso, para la aplicación y simulación de circuitos cuánticos, se utilizaron los siguientes módulos:

qiskit.QuantumCircuit

El módulo QuantumCircuit permite definir un circuito cuántico, especificando el número de qubits y registros clásicos necesarios.

qiskit aer.primitives.SamplerV2

Este módulo se encarga de la simulación del circuito en un backend sin ruido, donde al final devuelve los resultados medidos.

Capítulo 3

Algoritmos Cuánticos

3.1. Transformada de Fourier Cuántica

La transformada de Foureir cuántica o QFT es la versión cuántica de la transformada discreta de Foureir, es fundamental en varios algoritmos cuánticos.

3.1.1. Definición matemática:

Dado un estado cuántico en el espacio de Hilbert de n qubits, el estado base del sistema es representado por:

$$|x\rangle\,, \quad x \in \{0, 1, \ldots, 2^n - 1\}. \tag{3.1}$$

La QFT actua sobre dicho estado $|x\rangle$ y lo transforma en un nuevo estado de superposición:

$$QFT\,|x\rangle = \frac{1}{\sqrt{N}} \sum_{y=0}^{N-1} e^{2\pi xy/N}\,|y\rangle\,. \tag{3.2}$$

Esto es análogo a la transformada discreta de Fourier(DFT), que transforma una secuencia de números a una combinación de frecuencias, en el caso de la computación $N = 2^n$ ya que en un sistema cuántico de n qubits, cada qubit toma alguno de los estados base $|0\rangle$ o $|1\rangle$, por lo tanto el número de estados posibles es 2^n.

Sustituimos la siguiente expansión binaria en (3.2):

$$y = y_{n-1}y_{n-2}...y_0 = y_{n-1}2^{n-1} + ... + y_0 2^0, \tag{3.3}$$

con $y_i \in \{0, 1\}$, obtenemos:

$$QFT\,|x\rangle = \frac{1}{\sqrt{2^n}} \sum_{y_{n-1}=0}^{1} ... \sum_{y_0=0}^{1} e^{2\pi x \sum_{l=1}^{n} y_{n-l}/2^l} \,|y_{n-1}...y_0\rangle\,. \tag{3.4}$$

Utilizando la propiedad de que la exponencial de una suma es el producto de exponenciales:

$$QFT\,|x\rangle = \frac{1}{\sqrt{2^n}} \bigotimes_{l=1}^{n} [\sum_{y_{n-l}=0}^{1} e^{2\pi i x y_{n-l}/2^l} \,|y_{n-l}\rangle]. \tag{3.5}$$

Ahora si tomamos en cuenta la representación en fracciones binarias y dado que $e^{2\pi ik} = 1$ cuando K es un entero se puede despreciar la parte entera de la representación en fracciones binarias, por lo que se obtiene:

$$QFT\,|x\rangle = \frac{1}{\sqrt{2^n}}(|0\rangle + e^{2\pi i 0.x_0}\,|1\rangle) \otimes (|0\rangle + e^{2\pi i 0.x_1 x_0}\,|1\rangle) \otimes ... \otimes (|0\rangle + e^{2\pi i 0.x_{n-1}...x_0}\,|1\rangle). \tag{3.6}$$

Expresado de manera más compacta:

$$QFT\,|x\rangle = \frac{1}{\sqrt{2^n}} \bigotimes_{l=1}^{n} (|0\rangle + e^{2\pi i 0.x_{l-1}...x_0}\,|0\rangle). \tag{3.7}$$

3.1.2. Implementación en un circuito cuántico:

La QFT puede implementarse mediante puertas cuánticas en un circuito con n qubits. Dicho circuito se descompone en los siguientes pasos:

El primer paso es aplicar la puerta Hadamard al primer qubit, la puerta de Hadamard genera la siguiente superposición cuántica:

$$H\,|x_1\rangle = \frac{1}{\sqrt{2}}(|0\rangle + e^{2\pi i x/2}\,|1\rangle). \tag{3.8}$$

Esa fase $e^{2\pi ix/2}$ aparece porque la puerta H introduce una fase relativa que depende de la posición del bit en su representación binaria.

A continuación se aplica una puerta de rotación controloda al qubit R_k a los otros qubits, introduciendo fases dependientes del estado de los qubits más significativos.

La puerta de rotación R_k esta representada matricialmente de la siguiente forma:

$$R_k = \begin{bmatrix} 1 & 0 \\ 0 & e^{2\pi i/2^k} \end{bmatrix}. \tag{3.9}$$

Se repiten los primeros dos paso a los siguientes qubits, aplicando las rotaciones corresponientes.

El último paso es aplicar puertas SWAP al final para invertir el orden de los qubits, ya que la QFT coloca el resultado en el orden inverso de los bits originales.

El circuito cuántico que realiza la QFT aplicando los pasos anteriormente mencionados es:

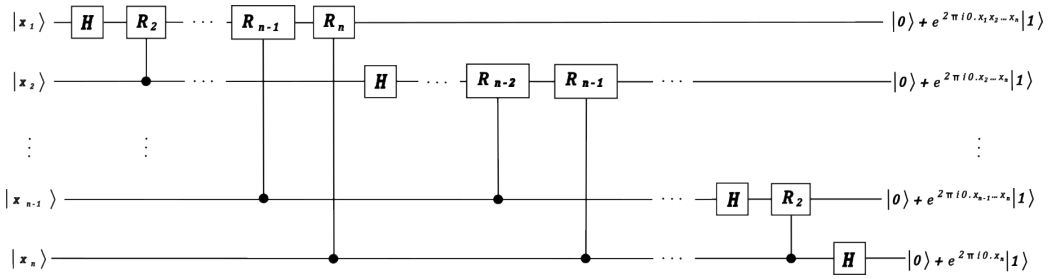


Figura 3.1: Circuito para implementar la QFT

Entonces la QFT transforma estados cuánticos en la base de fourier, lo que va a ser de mucha utilidad para realizar la suma.

3.1.3. Aplicación en Python

A continuación se muestra el diagrama de flujo del funcionamiento de la función QFT que tiene como parámetros de entrada n que es el número de qubits y su salida es la implementación en el circuito cuántico.

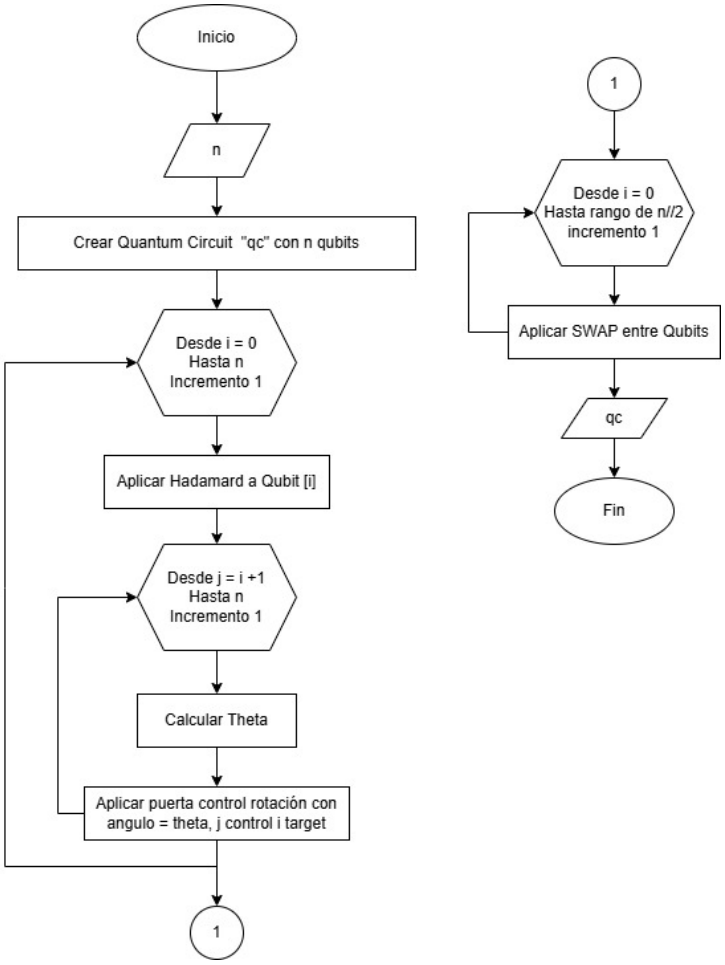


Figura 3.2: Diagrama de flujo de QFT

Esta función será utilizada en algoritmos posteriores.

3.1.4. Análisis de complejidad

Primero analisamos el número de puertas cuánticas requeridas para realizar la QFT a un registro de n qubits, tenemos que se requieren n puertas Hadamard y en el caso de las rotaciones R_k tenemos lo siguiente:

$$\sum_{k=1}^{n-1} \frac{n(n-1)}{2}, \tag{3.10}$$

por lo que en notación $Big - O$ el total de puertas cuánticas requierdas es $O(n^2)$.

En cuestión de profundidad es simple, ya que como todo se ejecuta en serie se tiene una complejidad de $O(n^2)$.

Para el almacenamiento la QFT no requiere de ancillas adicionales ni de ningún qubit extra, por lo que en caso de tener el registro con n qubits solo se requieren n qubits para realizar la transformación.

Métrica	**Valor**
Número de puertas cuánticas	$O(n^2)$
Profundidad del circuito (serial)	$O(n^2)$
Número de qubits requeridos	n
Ancillas requeridas	Ninguna
Precisión de rotaciones	Muy alta

Cuadro 3.1: Resumen de la complejidad de la QFT

3.2. Algoritmo de Grover

El algoritmo de Grover es un algoritmo de busqueda en base de datos no estructuradas implementa una aceleración cuadrática, esto es que el agoritmo de Grover es cuadráticamente más rápido ($\sqrt{N}$) que el mejor algoritmo clásico conocido.

Este algoritmo esta fundamentado en dos operaciones clave:

- Marcado del elemento objetivo mediante un operador cuántico.
- Amplificación de amplitud mediante un operador de difusión.

3.2.1. Planteamiento del problema:

Dado un conjunto $N = 2^n$ elementos, se desea encontrar un elemento marcado x_0 tal que una función $f(x)$ satisfaga las siguientes condiciones:

$$f(x) = \begin{cases} 1 & \text{si } x = x_0 \\ 0 & \text{en otro caso} \end{cases}, \tag{3.11}$$

el objetivo es determinar x_0 evaluando lo menos posible $f(x)$.

3.2.2. Fundamentos matemáticos:

Dado un conjunto N que tiene 2^n elementos, representados en el ket $|x\rangle$ y el objetivo es encontrar $|x_{objetivo}\rangle$ con la menor cantidad de consultas posibles, para ello se define U_f que actua sobre los estados $|x\rangle$ de la siguiente forma:

$$U_f |x\rangle = (-1)^{f(x)} |x\rangle , \tag{3.12}$$

donde:

- $f(x) = 1$ si $x = x_{objetivo}$, es decir que marca al estado objetivo con un signo $-$.
- $f(x) = 0$ en cualquier otro caso.

El algoritmo implementa superposiciones, interferencia cuántica y reflejos geométricos para amplificar las probabilidades de medir los $|x_{objetivo}\rangle$.

En esencia, los reflejos geométricos son transformaciones que voltean una figura a través de una línea, creando una imagen especular. Es como poner un espejo en un punto y ver cómo se ve la figura del otro lado.

Pasos del algoritmo:

El primer paso es inicializar el registro cuántico, se inicia con un registro cuántico de n qubits en estado $|0\rangle$ y se aplica una puerta H a todos los qubits:

$$|s\rangle = H^{\otimes n} |0\rangle = \frac{1}{\sqrt{N}} \sum_{x=0}^{N-1} |x\rangle . \tag{3.13}$$

Esto genera una superposición uniforme en todos los estados.

El siguiente paso es aplicar el oráculo cuántico U_f, este oráculo agrega un $-$ a los $|x_{objetivo}\rangle$:

$$U_f = \begin{cases} 1 & \text{si } x = x_0 \\ 0 & \text{en otro caso} \end{cases} . \tag{3.14}$$

Una vez que están marcados los estados objetivo, se aplica el operador de difusión D que actua como un reflejo sobre las medias de las amplitudes:

$$D = 2\,|s\rangle\,\langle s| - I. \tag{3.15}$$

El último paso es iterar los pasos anteriores, el ciclo de amplificación de Grover ($G = DU_f$) en la aplicación se repite alrededor de:

$$k = \frac{\pi}{4}\sqrt{N}. \tag{3.16}$$

3.2.3. Explicación geométrica:

El algoritmo de Grover de forma geométrica se interpreta en un espacio de dos dimensiones:

1. Un eje corresponde a la superposición uniforme de *kets*
2. Otro eje representa el estado objetivo $|x_{objetivo}\rangle$.

Cada iteración de Grover rota el estado en el subespacio generado por estos dos vectores, acercándolo progresivamente a $|x_{objetivo}\rangle$.

3.2.4. Complejidad y ventajas:

En el caso clásico, si se desea encontrar un elemento $x_0 \in 0, 1, ..., N-1$ tal que $f(x) = 1$, y no se tiene ninguna estructura que se pueda explotar, entonces se requiere en el peor caso revisar N elementos, esto nos da una complejidad de $O(N)$.

Por otro lado, el algoritmo cuántico como vemos en la ecuación (3.16) en el peor caso se busca en $\frac{\pi}{4}\sqrt{N}$ elementos por lo tanto su complejidad es $o(\sqrt{N})$, reduciendo su complejidad gracias a la amplificación de amplitudes.

Capítulo 4

Diseño y Comparación de Sumadores Cuánticos

4.1. Sumador Clásico

La suma lógica es una operación fundamental en álgebra booleana y en el diseño de circuitos digitales, especialmente en los sumadores binarios.

4.1.1. Operación OR Lógica

La operación básica de suma lógica puede definirse como la operación OR entre dos bits A y B, con la siguiente tabla de verdad:

A	B	$A+B$
0	0	0
0	1	1
1	0	1
1	1	1

Sin embargo, la suma binaria más completa requiere la consideración del acarreo entre bits.

4.1.2. Sumador Completo Full-Adder

El sumador completo es un circuito lógico que suma tres bits: dos bits de entrada A y B, y un acarreo de entrada C_{in}. Produce dos salidas:

- El bit suma S.
- El bit acarreo de salida C_{out}.

Las expresiones booleanas para estas salidas son:

$$S = A \oplus B \oplus C_{in}, \tag{4.1}$$

$$C_{out} = (A \wedge B) \vee (B \wedge C_{in}) \vee (A \wedge C_{in}). \tag{4.2}$$

donde $\oplus$ denota la operación XOR, $\wedge$ la operación AND y $\vee$ la operación OR.

Todas las combinaciones posibles de entrada y sus resultados de salida se pueden ver en la siguiente tabla de verdad:

A	B	C_{in}	S	C_{out}
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

A continuación se muestra el diagrama esquemático del circuito lógico para un sumador completo clásico.

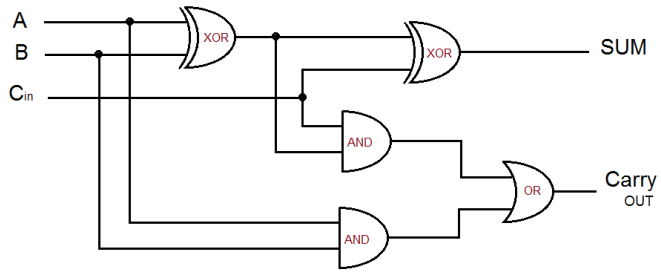


Figura 4.1: Diagrama del sumador completo clásico

En este circuito, las puertas XOR se utilizan para calcular el bit suma S, mientras que las puertas AND y OR se combinan para calcular el acarreo C_{out}. Esta estructura es fundamental para construir sumadores de múltiples bits y realizar operaciones aritméticas en sistemas digitales.

4.1.3. Análisis de complejidad de la suma clásica:

El sumador completo es un bloque fundamental para la suma binaria en sistemas digitales. Su complejidad se analiza en términos de:

- **Complejidad lógica:** número de puertas lógicas necesarias.
- **Complejidad temporal (retardo):** número de etapas secuenciales (niveles de puertas) que determinan la velocidad del circuito.
- **Complejidad espacial:** cantidad de recursos físicos (transistores o componentes) requeridos.

Para calcular la complejidad lógica, se analiza de cuantas puertas lógicas se compone el algoritmo, en este caso tenemos:

- **Puertas XOR:** 2 (para calcular la suma S).
- **Puertas AND:** 3 (para calcular el acarreo C_{out}).
- **Puertas OR:** 1 (para combinar los productos AND y generar el acarreo).

Esto totaliza:

6 puertas lógicas principales.

Ahora la complejidad temporal (retardo), en este caso se mide por la cantidad de niveles de puertas que la infromación debe atravesar para generar la salida correcta.

La suma $S = A \oplus B \oplus C_{in}$ requiere de dos etapas conformadas por puertas XOR en cascada:

- Primera XOR: $A \oplus B$.

- Segunda XOR: resultado anterior $\oplus C_{in}$.

El acarreo $C_{out} = (A \wedge B) \vee (B \wedge C_{in}) \vee (A \wedge C_{in})$ esta confromado por:

- Tres puertas AND en paralelo (retardo igual a 1 etapa).
- Una puerta OR que combina los resultados de los ANDs (retardo adicional de 1 etapa).

Por lo que el retardo total para obtener C_{out} es de 2 fases o etapas.

Por último la complejidad espacial, el sumador completo utiliza un número fijo de puertas sin importar el tamaño de los números a sumar, pero para poder obtener un sumador de multiples bits, es necesario una cadena de sumadores completos en cascada, con complejidad $O(n)$ con n el número de bits.

4.2. Sumador Cuántico Simple

Ahora veremos el sumador cuántico simple que es básicamente el circuito del Full Adder pero implementando puertas lógico cuánticas y aprovechando la reversivilidad de los circuitos cuánticos.

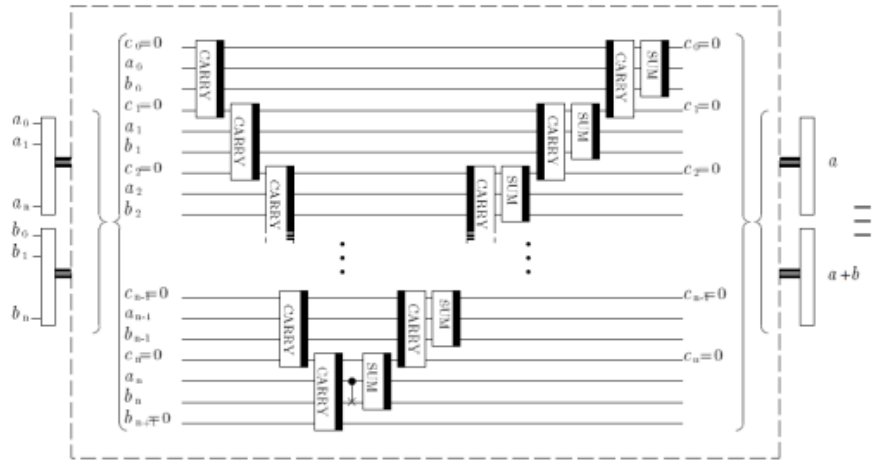


Figura 4.2: Diagrama general del sumador cuántico VBE

La idea es la misma que el Full Adder y es propagar el acarreo, en este caso los circuitos cuánticos van a actuar sobre estados cuánticos representados por n qubits que formaran la entrada del circuito. El espacio de estados será un espacio complejo de Hilbert de dimensión 2^n. Los estados serán representados de la siguiente manera:

$$|x\rangle = |x_1\rangle \otimes |x_2\rangle \otimes ... \otimes |x_n\rangle \,, \tag{4.3}$$

además $x_i = 0, 1$.

El algoritmo opera dos registros de entrada $|a\rangle$ y $|b\rangle$, un registro intermedio $|c\rangle$ para el acarreo que esta integrado al Sumador y dos estados salida $|a\rangle$ y $|a+b\rangle$, gracias a esta información de salida se puede realizar una operación que nos permite recuperar los valores originales.

Entonces para relizar la suma requerimos de los siguinetes estados cuánticos:

- $|a\rangle$ que es un número a en su representación binaria.
- $|b\rangle$ que es un segundo número b en su representación binaria.
- $|c\rangle$ registro de $n+1$ para guardar y propagar el acarreo, este registro esta inicializado en 0.

El circuito se divide primero en el cálculo de los acarreos, para ellos esta el bloque $Carry$, que calcula todos los acarreos hasta que el último bloque calcula el dígito más significativo de la operación, para ello emplea el registro $|c\rangle$ junto con $|a\rangle$ y $|b\rangle$, este bloque esta compuesto por dos puertas $Toffoli$ y una puerta $C-Not$, el esquema de este bloque se muestra en la siguiente figura:

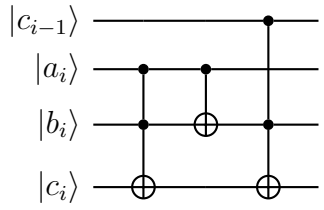


Figura 4.3: Bloque Carry

Posteriormente se deshacen las operaciones de $Carry$ y se realiza la suma bit a bit, aplicando el mismo bloque $Carry$ pero aplicando las puertas que lo conforman de manera inversa, se logra revertir las operaciones regresando al registro $|c\rangle$ a su estado original, excepto el último bloque $Carry$ que se realizo ya que como este último bloque nos calcula el acarreo para la cifra más significativa en caso de que $|c_i\rangle$ se encuentre en el estado 0 no afecta pero si se encuentra en 1 nos indica que hay un acarreo que se propago hasta el final lo cual lo convierte en la cifra más significativa del número $|a+b\rangle$, ahora para realizar la suma se implementa el bloque Sum que se compone de dos puertas $C-Not$. el esquema de este bloque es el siguiente:

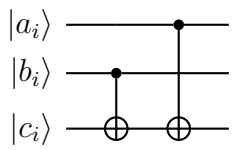


Figura 4.4: Bloque Sum

Entonces el proceso es aplicar $Carry$, luego aplicar Sum al último par de qubits y por último intercalar la Sum con $Carry\ Inv$ para regresar al registro $|c\rangle$ a su estado original y obtener la suma que se almacena en el registro $|b\rangle$

4.2.1. Implementación en Qiskit:

Para su implementación en Python utilizando qiskit se crearon funciones que se describirán a continuación. Para facilitar la comprensión de la estructura se emplean diagramas de flujo que representan visualmente las diferentes funciones que conforman el algoritmo.

La primer función es **decimal to binary equal** que recibe como parámetros de entrada $num1$ y $num2$ y regresa como salida $maxlength$, $bin1$ y $bin2$.

Esta función transforma los números de entrada en decimal a binario y en caso de que alguno en su representación binaria sea más largo que el otro, al número más corto se le agregan ceros a la izquierda para que ambos tengan la misma longitud. La figura (4.5) muestra el diagrama de flujo de la función.

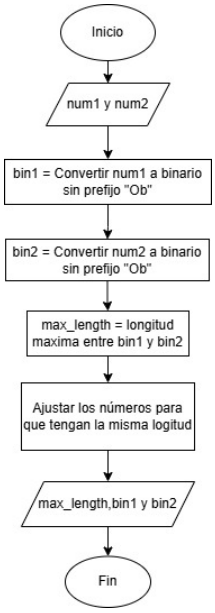


Figura 4.5: Diagrama de flujo de la función "Decimal to binary".

La segunda función llamada **binary to matrix** recibe $bin1$ y $bin2$ de la función anterior y regresa como salida $matrix1$ y $matrix2$. En la figura (4.6) se muestra el diagrama de flujo de dicha función.

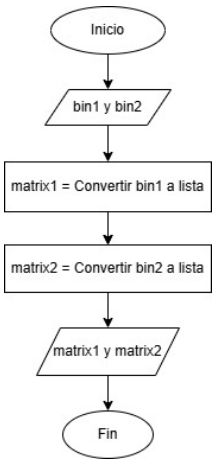


Figura 4.6: Diagrama de flujo de la función "Binary to matrix".

La función mostrada en la figura (4.6) transforma los números en binario

$bin1$ y $bin2$ a listas.

La tercer función **binary to decimal** recibe la variable *listabinaria* y nos arroja como salida *Resultado*, esta función transforma la entrada de lista a cadena y de cadena a decimal. En la figura (4.7) se muestra el diagrama de flujo del funcionamiento.

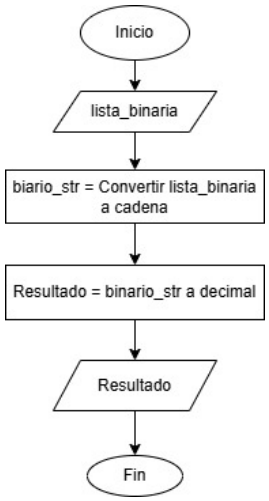


Figura 4.7: Diagrama de flujo de la función "Binary to decimal".

Las funciones **decimal to binary equal**, **binary to matrix** y **binary to decimal** constituyen módulos de procesamiento y preparación de la información, y son utilizadas en otros algoritmos implementados en el presente trabajo.

Éstas funciones son implementadas para realizar la suma como se muestra en la figura (4.8).

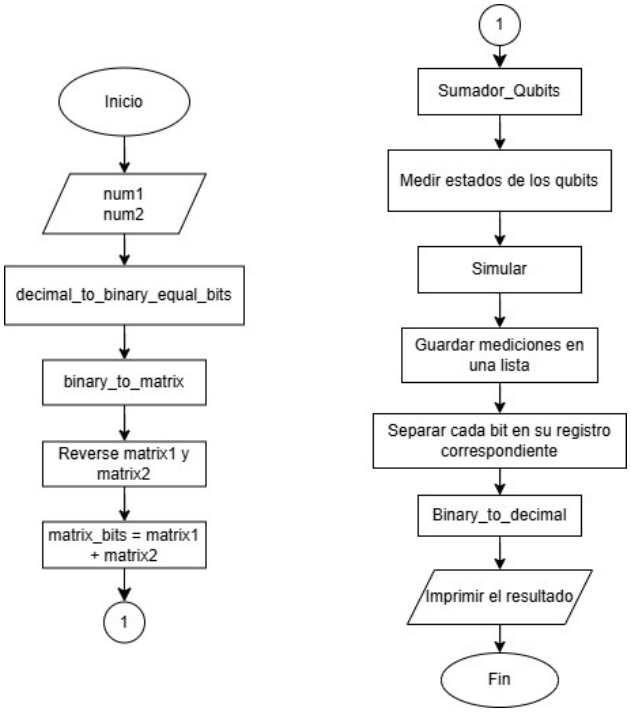


Figura 4.8: Diagrama de flujo del llamado de las funciones.

4.2.2. Análisis de complejidad

El sumador cuántico implementa un sumador en serie reversible, donde el acarreo se propaga bit a bit. Utiliza los siguientes recursos:

- n qubits para el primer número A.
- n qubits para el segundo número B.
- $n + 1$ qubits para el acarreo, con $C_0 = 0$ inicial.

La complejidad de puertas lógicas en este caso son:

- **Puertas Toffoli (CCNOT):** utilizadas para calcular y propagar los acarreos.
- **Puertas C-NOT:** utilizadas para operaciones XOR reversibles (sumas módulo 2).

por lo tanto para la suma de dos números de n bits se requieren:

$$2n - 1 \quad \text{puertas Toffoli}$$

$$5n - 3 \quad \text{puertas CNOT}$$

La profundiad temporal, o número puertas secuenciales es:

$$O(n)$$

Esto debido a la propagación secuencial del bloque $Carry$, lo que impide paralelizar completamente todo el circuito.

En cuestión de qubits tenemos que el circuito requiere de $3n + 1$ qubits, incluyendo los qubist de entrada y acarreo.

4.3. Sumador Cuántico con QFT

Tradicionalmente, los algoritmos para implementar la suma diseñados para ordenadores cuánticos siguen los esquemas que se implementan en la computación clásica, con la extensión que permite a dichos algoritmos realizar cáluculos reversibles. Sin embargo, se pueden diseñar algoritmos cuánticos para realizar la suma sin necesidad de copiar los esquemas clásicos.

En el paper **Adittion on a Quantum Computer** se presenta un sumador cuántico aplicando la transformada de Fourier, con dos configuraciones, este sumador reduce el número de qubits necesarios para el cálculo ya que no requiere qubits de acarreo para realizar la suma.

Primero analizaremos un esquema muy general del sumador, el circuito suma dos números a y b de la siguiente manera:

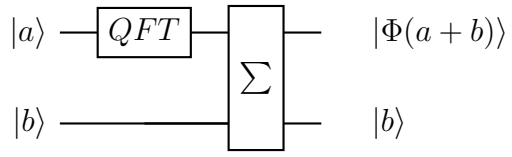


Figura 4.9: Esquema general suma con QFT.

como se puede apreciar en la figura (5), la primera operación que se realiza es la QFT, esta transformada se aplica al $|a\rangle$, la cual realiza la siguiente transformación:

$$|a\rangle \rightarrow \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} e^{\frac{2\pi i a k}{N}} |k\rangle \,, \tag{4.4}$$

donde sabemos que $N = 2^n$ con n el número de bits para representar el número a. El resultado de aplicar la QFT es que los bits que conforman a a se codifican en los desfases entre los términos de $|k\rangle$. El segundo bloque que esta denotado por $\sum$, que es el bloque sumador, dicho bloque esta compuesto por varias puertas de rotación de fase controladas por el estado $|b\rangle$, estas puertas controladas codifican la información relativa a b.

El circuito emplea n qubits para representar a a y otros n para b en la entrada del circuito, en la salida tenemos n qubits para representar a b y otros n para representar a la suma de $a + b$ en el registro de a, esto es:

$$(a, b) \rightarrow (a + b, b)$$

por lo que el algoritmo requiere de $2n$ qubits.

Ahora el bloque denotado por $\sum$ como se menciono anteriormente se conforma por puertas de rotación de fase controlada como se muentras a continuación:

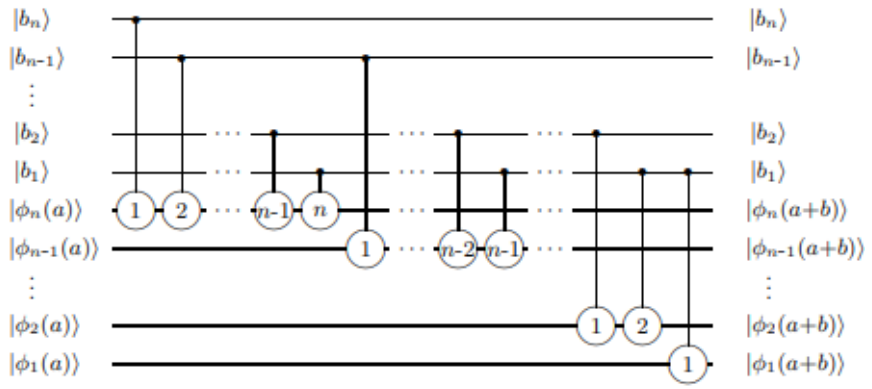


Figura 4.10: Bloque $\sum$

donde al igual que en el QFT estas rotaciones son representadas por la puerta R_k con $k = 1, 2, 3.., n$.

Este bloque $\sum$ añade una fase $e^{\frac{2\pi b}{2}}$ a la $QFT\,|a\rangle$:

$$|a\rangle \rightarrow \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} e^{\frac{2\pi i a k}{N}} e^{\frac{2\pi i b k}{N}} |k\rangle = e^{\frac{2\pi i (a+b) k}{N}} |k\rangle . \tag{4.5}$$

Para lograr esto se requiere trabajar con la expansión de b tal que:

$$b = b_1 2^0 + b_2 2^1 + ... + b_{n-1} 2^{2n-2} + b_n 2^{2n-1}. \tag{4.6}$$

El registro $|b\rangle$ en notación tensorial se puede expresar como:

$$|b\rangle = |b_1\rangle \otimes |b_2\rangle \otimes ... \otimes |b_n\rangle . \tag{4.7}$$

Las puertas de rotación están controladas por los qubits $|b_1\rangle, |b_2\rangle ..., |b_n\rangle$ como se muestra en la figura 4.10 y se construyen de acuerdo a la siguiente matriz:

$$R_k = \begin{bmatrix} 1 & 0 \\ 0 & e^{2\pi i/2^k} \end{bmatrix} . \tag{4.8}$$

Las puertas deben cumplir la siguiente expresión:

$$|a\rangle \rightarrow \frac{1}{\sqrt{2^n}} \sum_{k=0}^{2^n-1} e^{2\pi i \frac{(a+b)\cdot k}{2^n}} |k\rangle = \frac{1}{\sqrt{2^n}} \sum_{k=0}^{2^n-1} e^{2\pi i 2^{n-j} (a_j + b_j) \sum_{s=1}^{n} k_s 2^{n-s}} |k\rangle . \tag{4.9}$$

Esto quiere decir que con cada k_s, que representa $\phi_1, ...\phi_n$, hay que aplicar puertas de rotación de fase para introducir dichas fase.

Para ver cómo actuan las puertas de rotación, analizamos el resultado de aplicar las rotaciones condicionales sobre el estado $|\phi_n(a)\rangle$. Las rotaciones que actuan sobre el qubit son las puertas $R_{1+n-n} = R_1$ controlada por $|b_1\rangle$, $R_{2+n-n} = R_2$ controlada por $|b_2\rangle$ y asi sucesivamente hasta llegar a $R_{n+n-n} = R_n$ controlada por $|b_n\rangle$ como se muestra en la figura (9).

La primer puerta de rotación de fase R_1 que es controlada por $|b_n\rangle$ realiza la siguiente operación:

$$|\phi_n(a)\rangle \rightarrow \frac{1}{\sqrt{2^n}} (|0\rangle + e^{2\pi i (0.a_1 a_2 ... a_n + 0.b_n)} |1\rangle). \tag{4.10}$$

La siguiente puerta es R_2 controlada por $|b_{n-1}\rangle$:

$$|\phi_n(a)\rangle \rightarrow \frac{1}{\sqrt{2^n}}(|0\rangle + e^{2\pi i(0.a_1a_2...a_n+0.b_nb_{n-1})}|1\rangle). \tag{4.11}$$

Se continua aplicando las rotaciones controladas hasta llegar a R_n controlada por $|b_1\rangle$ y se obtiene el siguiente resultado:

$$|\phi_n(a)\rangle \rightarrow \frac{1}{\sqrt{2^n}}(|0\rangle + e^{2\pi i(0.a_1a_2...a_n+0.b_1b_2..b_n)}|1\rangle) = |\phi_n(a+b)\rangle\,. \tag{4.12}$$

Para $|\phi_{n-1}(a)\rangle$ se realiza el mismo proceso, las puertas $R_{1+(n-1)-n} = R_0$ pero no tenemos dicha puerta por lo que seguimos con $R_{2+(n-1)-n} = R_1$ que esta controlada por $|b_{n-1}\rangle$ y repetimos hasta llegar a la puerta $R_{n+(n-1)-n} = R_{n-1}$ controlada por $|b_1\rangle$.

Se repite este procedimiento con cada estado obtenido de aplicar la QFT al registro a, una vez aplicada la QFT y todas las puertas de rotación controladas en el registro inicialmente $|a_1\rangle, |a_2\rangle, ..., |a_n\rangle$ obtenemos el nuevo registro $|\phi_1(a+b)\rangle, |\phi_2(a+b)\rangle, ..., |\phi_n(a+b)\rangle$ por lo que aplicamos de nuevo las puertas que conforman la QFT pero en orden inverso para obtener:

$$|(a+b)_1\rangle, |(a+b)_2\rangle, ..., |(a+b)_n\rangle\,. \tag{4.13}$$

lo que ya es el resultado de realizar la suma aritmética entre a y b.

4.3.1. Implementación Qiskit

Al igual que el **Sumador Cuántico Simple** esta implementación utiliza las funciones mostradas de la figura 4.5 a la 4.7 más la **QFT** que se muestra en la figura 3.2.

Una nueva función es la **QFT Inversa**, la siguiente función que tenemos es **Sumador QFT** que tiene como parametros de entrada *maxlength*, *matrixbits* y *matrix*1, esta función es la encargada de implementar la suma.

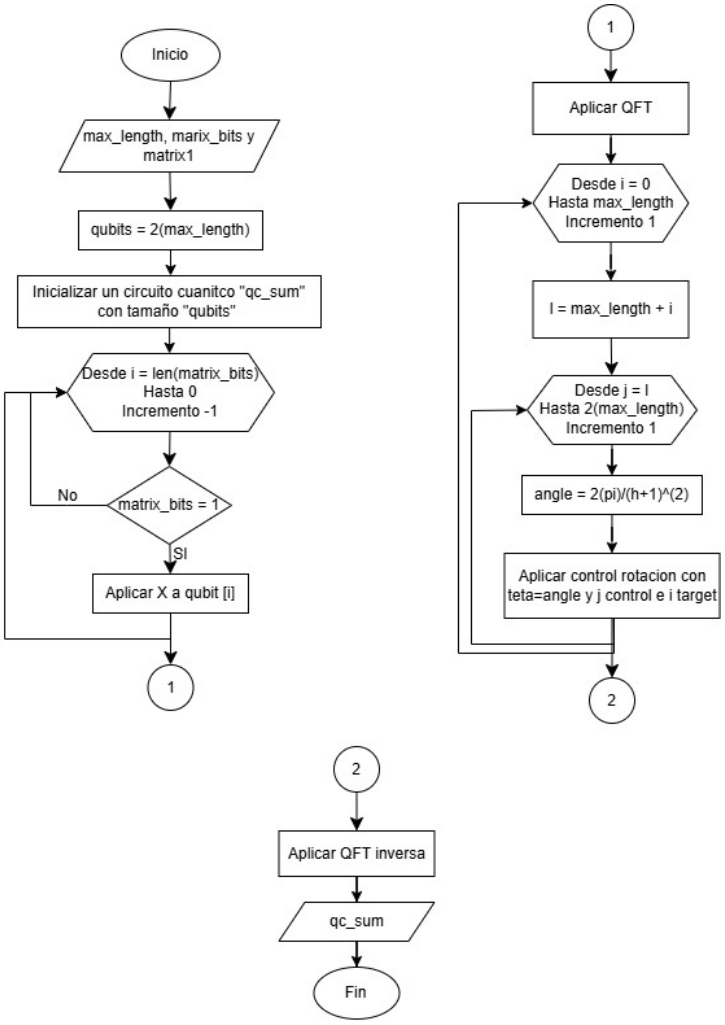


Figura 4.11: Diagrama de flujo de la función sumador QFT.

Dichas funciones son implementadas como se muestra en la figura 4.12 para realizar la suma aplicando la QFT.

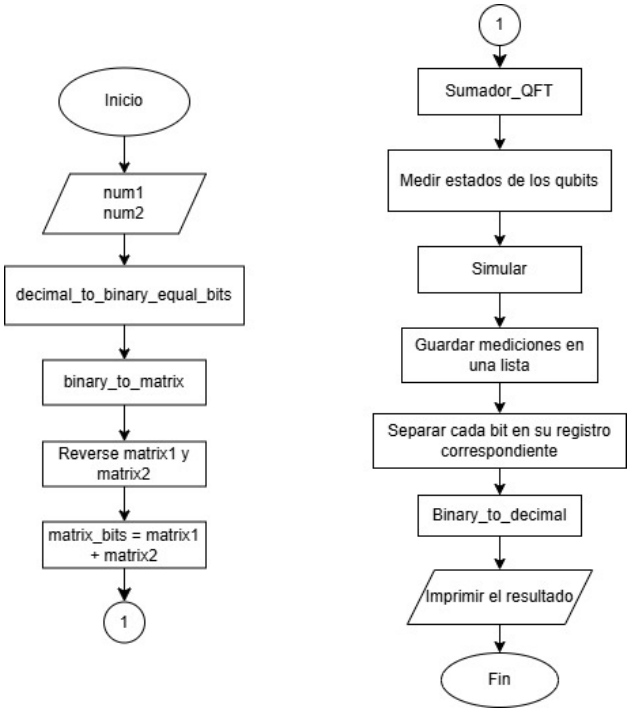


Figura 4.12: Diagrama de flujo de aplicación de las funciones para realizar la suma con QFT.

4.3.2. Análisis de complejidad

Para analizar la complejidad del **Sumador con** QFT hay que analizarlo por bloques, para este caso tenemos:

$$QFT \rightarrow Rotaciones \rightarrow QFT^{-1}$$

Y sabemos por la sección(2.2) la QFT presenta una complejidad $O(n^2)$ en número de puertas cuánticas y en profunidad.

Ahora para las rotaciones controladas que realizan la suma, tenemos que para cada bit b_j del segundo número b, se aplican n puertas de rotación R_k sobre el registro a condicionadas por b_j, lo que añade n^2 rotaciones por lo que la complejidad para el bloque de rotaciones es de $O(n^2)$.

El último bloque de QFT^{-1} tiene exactamente la misma complejidad que QFT, por lo que la complejidad de todo el algortimo en puertas cuánticas y

profundidad es de $O(n^2)$.

En memoria tenemos dos registros de entrada $|a\rangle$ y $|b\rangle$ de n qubits, y no se requieren qubits extras por lo tanto solo se requieren $2n \approx n$.

Recurso	**Valor**
Número de qubits	$n + m$ (para a y b)
Número de puertas	$O(n^2)$
Profundidad	$O(n^2)$
Ancillas	No requiere ancillas

Cuadro 4.1: Recursos del sumador con QFT

4.4. Suma cuántica con QFT en paralelo

En computación cuántica, la implementación eficiente de operaciones aritméticas como la suma es fundamental para el desarrollo de algoritmos avanzados. La suma cuántica utilizando la Transformada de Fourier Cuántica (QFT) ofrece una alternativa que aprovecha las propiedades unitarias y la diagonalización de la suma en el espacio de Fourier, lo que reduce la profundidad total del circuito en comparación con sumadores clásicos reversibles.

Sin embargo, el rendimiento puede optimizarse aún más mediante la paralelización, explotando la independencia estructural de ciertas operaciones dentro del bloque de rotaciones controladas que realizan la suma. Este capítulo presenta un análisis exhaustivo de la paralelización de la suma con QFT, describiendo los principios que la hacen posible, su formalización matemática y las ventajas en términos de eficiencia.

La suma entre dos números binarios a y b, cada uno representado en registros cuánticos de n qubits:

$$|A\rangle = |a_1, a_2, ..., a_n\rangle \quad |B\rangle = |b_1, b_2, ..., b_n\rangle \,, \tag{4.14}$$

se implementa de la siguiente forma:

1. Se aplica la QFT al registro $|A\rangle$, transformando la representación computacional en una superposición con fases codificadas:

$$|A\rangle \xrightarrow{\text{QFT}} \text{QFT}|A\rangle = \frac{1}{\sqrt{2^n}} \sum_{k=0}^{2^n-1} e^{2\pi i a k/2^n} |k\rangle. \tag{4.15}$$

2. A continuación, se aplican las puertas de rotación controladas que añaden fases dependientes de b para realizar la suma en el espacio de Fourier:

$$\prod_{j=1}^{n} \prod_{l=1}^{n-j+1} R_{b_j l}, \tag{4.16}$$

   donde $R_{b_j l}$ es una puerta de rotación de fase controlada por el qubit b_j y que actúa sobre el qubit a_l.

3. Finalmente, se aplica la QFT inversa para transformar el registro A de nuevo al dominio computacional, obteniendo el registro con la suma $|a+b\rangle$.

4.4.1. Análisis de la estructura de las puertas controladas

El bloque fundamental donde se puede aplicar la paralelización es el conjunto de puertas de rotación controladas $R_{b_j l}$. Cada una de estas puertas implementa una rotación de fase de la forma:

$$R_k = \begin{bmatrix} 1 & 0 \\ 0 & e^{2\pi i/2^k} \end{bmatrix}, \tag{4.17}$$

y su acción está condicionada al valor del qubit control b_j.

Observemos la matriz conceptual de las puertas en función de los qubits control b_j (filas) y qubits objetivo a_l (columnas):

$$\begin{array}{cccccc} & a_1 & a_2 & a_3 & \dots & a_{n-j+1} \\ b_1 & R_{b_1 1} & R_{b_1 2} & R_{b_1 3} & \dots & R_{b_1 n} \\ b_2 & R_{b_2 1} & R_{b_2 2} & R_{b_2 3} & \dots & R_{b_2 n-1}. \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ b_n & R_{b_n 1} & R_{b_n 2} & \dots & & \end{array} \tag{4.18}$$

Nótese que las puertas en cada fila actúan sobre qubits objetivo diferentes y las filas están condicionadas por qubits control diferentes.

4.4.2. Principio de paralelización

La paralelización es posible porque las puertas de rotación que actúan sobre distintos qubits objetivo y están controladas por distintos qubits control son operaciones unitarias independientes que conmutan, es decir:

$$R_{b_j l} R_{b_{j'} l'} = R_{b_{j'} l'} R_{b_j l} \quad \text{si } (j \neq j' \text{ y } l \neq l'). \tag{4.19}$$

Esto implica que se pueden aplicar simultáneamente puertas que no comparten ni el qubit control ni el qubit objetivo, pues actúan sobre subespacios independientes del espacio de Hilbert total.

De este modo, agrupando las puertas que cumplen esta condición, el bloque de rotaciones se puede descomponer en capas o etapas paralelas donde todas las puertas dentro de la misma capa se ejecutan simultáneamente.

4.4.3. Formalización matemática de la paralelización

Consideremos la operación de suma cuántica implementada mediante la Transformada de Fourier Cuántica (QFT). Como se mencionó, la suma se realiza aplicando un conjunto de puertas de rotación controladas, indicadas como $R_{b_j l}$, donde:

- b_j es el j-ésimo qubit del registro control $|B\rangle$, con $j = 1, 2, \ldots, n$.
- a_l es el l-ésimo qubit del registro objetivo $|A\rangle$, con $l = 1, 2, \ldots, n-j+1$.
- $R_{b_j l}$ representa la puerta de rotación de fase controlada por el qubit b_j que actúa sobre el qubit a_l.

La operación total que realiza las rotaciones controladas para sumar es:

$$U_{\text{suma}} = \prod_{j=1}^{n} \prod_{l=1}^{n-j+1} R_{b_j l}. \tag{4.20}$$

Para paralelizar esta operación, debemos identificar subconjuntos de puertas que puedan aplicarse simultáneamente sin interferir, es decir, puertas que no comparten ni el qubit control ni el qubit objetivo.

Definamos el conjunto total de puertas:

$$\mathcal{R} = \{R_{b_j l} \mid 1 \leq j \leq n, \quad 1 \leq l \leq n - j + 1\}. \tag{4.21}$$

Dos puertas $R_{b_j l}$ y $R_{b_{j'} l'}$ son compatibles para ejecutarse en paralelo si y sólo si:

$$j \neq j' \quad \text{y} \quad l \neq l'. \tag{4.22}$$

Esto implica que no comparten qubit control ni qubit objetivo.

Con esta definición, podemos particionar el conjunto $\mathcal{R}$ en m subconjuntos disjuntos o *capas* $\{L_k\}_{k=1}^m$, tales que:

$$\bigcup_{k=1}^{m} L_k = \mathcal{R}, \quad L_k \cap L_{k'} = \emptyset \quad \forall k \neq k', \tag{4.23}$$

y todas las puertas dentro de la misma capa L_k pueden aplicarse en paralelo.

Por lo tanto, la operación total se expresa como un producto secuencial de capas paralelas:

$$U_{\text{suma}} = \prod_{k=1}^{m} L_k, \quad \text{donde} \quad L_k = \prod_{R_{b_j l} \in L_k} R_{b_j l}. \tag{4.24}$$

La base física y matemática para esta paralelización radica en la conmutatividad de puertas que actúan sobre qubits distintos. Dado que cada puerta $R_{b_j l}$ actúa únicamente sobre el subespacio asociado a los qubits b_j y a_l, las puertas compatibles conmutan.

Esta conmutatividad garantiza que la aplicación simultánea de puertas dentro de la misma capa no afecta el resultado final, haciendo válida la ejecución paralela. Para visualizar la paralelización, las puertas se organizan en una matriz donde las filas corresponden a los qubits control b_j y las columnas a los qubits objetivo a_l:

$$\begin{array}{cccccc} & a_1 & a_2 & a_3 & \cdots & a_n \\ b_1 & R_{b_1 1} & R_{b_1 2} & R_{b_1 3} & \cdots & R_{b_1 n} \\ b_2 & R_{b_2 1} & R_{b_2 2} & R_{b_2 3} & \cdots & R_{b_2 n-1} \\ b_3 & R_{b_3 1} & R_{b_3 2} & \cdots & & \\ \vdots & \vdots & \vdots & & \ddots & \\ b_n & R_{b_n 1} & & & & \end{array} . \tag{4.25}$$

Cada capa corresponde a seleccionar un conjunto de entradas en esta matriz sin repetir filas ni columnas.

4.4.4. Aplicación en un circuito cuántico

La aplicación de esta paralelización es simple, para entender la paralelización veamos la figura (4.10) que muestra el bloque de suma que esta compuesto por rotaciones R_k y tomando en cuenta que la paralelización es posible siempre que las rotaciones poseen difrentes qubits tanto objetivo como control, podemos notar que todas las puertas R_1 cumplen esta condición, a su vez las R_2 también cumplen y asi sucecivamente hasta n, por lo tanto la paralelización consiste en aplicar todas las puertas R_k con $k = 1, 2, .., n$ en orden de índice, primero todas las $k = 1$, luego $k = 2$ y asi sucesivamente hasta llegar a n.

En un cricuito cuántico la paralelización de la suma sería de la sigueinte forma:

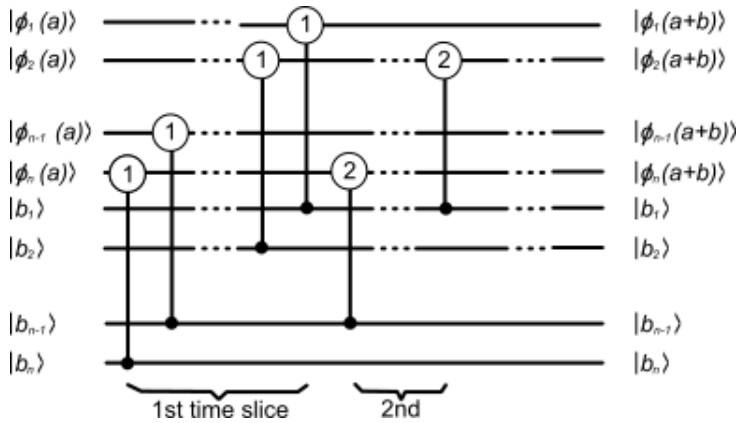


Figura 4.13: Circuito para la suma con paralelización de las rotaciones.

4.4.5. Aplicación en Python

Las funciones requeridas para realizar este algoritmo en python son las mismas que para el algoritmo **Suma con QFT** a excepción de la función **Sumador QFT** donde se tienen los mismos parámetros de entrada pero cambia la manera de aplicar las puertas de rotación como se muestra a continuación:

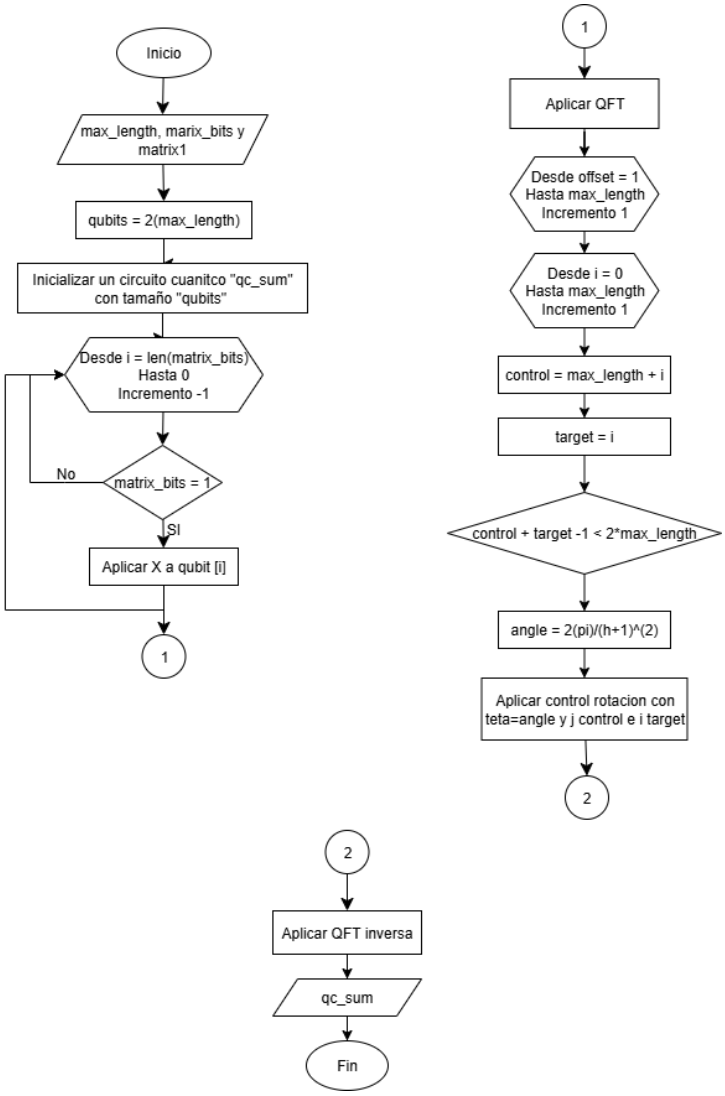


Figura 4.14: Sumador QFT con las rotaciones paralelizadas

A continuación se áplican las funciones de los algoritmos anteriores como se muestra en la figura (4.12).

4.5. Aproximación de rotaciones

El sumador con QFT realiza la operación $|a\rangle |b\rangle \rightarrow |a+b\rangle |b\rangle$ envolviendo $|a\rangle$ entre una QFT y su inversa, e insertando entre ambas una red de rotaciones de fase controladas por el registro $|b\rangle$.

Los ángulos de esas rotaciones se van reduciendo exponencialmente, por lo que los ángulos más pequeños apenas alteran la salida, por tanto, pueden aproximarse para reducir el número de puertas y de profundidad.

4.5.1. Fundamento matemático de la aproximación

Después de la QFT, cada amplitud del registro $|a\rangle$ lleva un factor de fase:

$$e^{\frac{2\pi i a}{2^{k+1}}}. \tag{4.26}$$

Sumar b equivale a añadir fases controladas de la forma:

$$e^{\frac{2\pi i b_j}{2^{k-j+1}}}. \tag{4.27}$$

Cuando $\Delta = k - j$ crece, el denominador $2^{\Delta} = 2^{k-j}$ crece exponencialmente lo que hace que la fase se acerce rápidamente a cero. En la práctica, cualquier efecto menor que el error físico típico de puerta resulta invisible, de modo que esas rotaciones se pueden omitir.

4.5.2. Regla de truncado

Para omitir esas puertas hay que elegir una tolerancia angular ϵ, que es el parámetro para borrar toda $R_k(\theta)$ que cumpla con:

$$\theta < \epsilon, \tag{4.28}$$

con $\theta = \frac{2\pi}{2^{\Delta}}$.

De esta inecuación despejamos 2^{Δ} lo que nos arroja la siguiente inecuación:

$$2^{\Delta} > \frac{2\pi}{\epsilon}. \tag{4.29}$$

Ahora aislamos Δ:

$$\Delta > \log_2(\frac{2\pi}{\epsilon}). \tag{4.30}$$

Sabemos que el desplazamiento Δ solo puede ser un entero, por otra parte llamaremos m al último desplazamiento permitido, de modo que $m+1$ viola la condición.

Para que m sea el último desplazamiento permitido debemos tener lo siguientes:

$$2^{m} \leq \frac{2\pi}{\epsilon}, \qquad 2^{(m+1)} > \frac{2\pi}{\epsilon}. \tag{4.31}$$

Esto se traduce a la siguiente regla:

$$m = \log_2(\frac{2\pi}{\epsilon}), \tag{4.32}$$

o también:

$$m = \log_2(\frac{2\pi}{\epsilon}) - 1. \tag{4.33}$$

4.5.3. Suma de ángulos descartados

Los ángulos ingorados forman la siguiente cola geométrica:

$$S = \sum_{m+1}^{\infty} \theta_{\Delta} = \frac{2\pi}{2^{m+1}} + \frac{2\pi}{2^{m+2}} + ... = \frac{2\pi}{2^{m+1}}(\frac{1}{1-\frac{1}{2}}) = \frac{2\pi}{2^{m}}, \tag{4.34}$$

por definicion de m:

$$\frac{2\pi}{2^{m}} < \epsilon, \tag{4.35}$$

así, la suma de todos los ángulos omitidos es menor que ϵ.

4.5.4. De ángulos a norma operador

Cada puerta suprimida aporta $< \theta_\Delta$ al error, esas puertas suprimidas están en serie; usar la desigualdad triangular da:

$$\| E \|_{op} \leq S. \tag{4.36}$$

Esto junto con el resultado del apartado anterior tenemos la siguiente cota:

$$\| U_{ex} - U_{ap} \|_{op} < \epsilon. \tag{4.37}$$

4.5.5. Probabilidad de error al medir

Como se esta aproximando existe una probabilidad de leer un entero equivocado, dicha probabilidad se obtiene de la siguiente manera.

Sea ρ_{ap} el estado después de aplicar el sumador cuantico aproximado, para cualquier M se cumple la desigualdad de Helstrom:

$$\left| \Pr_{\text{ex}}(M) - \Pr_{\text{ap}}(M) \right| \leq \frac{1}{2} \|\rho_{\text{ex}} - \rho_{\text{ap}}\|_1 \leq \|U_{\text{ex}} - U_{\text{ap}}\|_{\text{op}}. < \epsilon \tag{4.38}$$

Ahora si medimos en la base computacional, la probabilidad de leer un entero equivocado se acota mejor mediante la aproximación cuadrática a baja fase:

$$P_{\text{err}} \leq \frac{\epsilon^2}{4}. \tag{4.39}$$

En resumen el tope ϵ se pone arbitrariamente y tras la medición, la suma saldra mal con una probabilidad de $\leq \frac{\epsilon^2}{4}$.

4.5.6. Regla Mnemónica

A pesar de que ϵ es arbitrario hay una forma de balancear la profundidad del circuito y fidelidad global al momento de aproximar la QFT, para esto primero tenemos que tras aplicar la QFT sin truncar, cada qubit se encuentra en:

$$\frac{|0\rangle + |1\rangle}{\sqrt{2}} \quad \text{o} \quad \frac{|0\rangle - |1\rangle}{\sqrt{2}}, \tag{4.40}$$

la diferencia logica es $0 \leftrightarrow 1$ que en términos geométricos es un giro de π radianes sobre el eje Z.

Un error de fase $\Delta\phi$ solo provoca un error en la lectura si supera la mitad de ese giro, $\frac{\pi}{2}$, al dividir ese margen entre los n qubits obtenemos el siguiente epsilon:

$$\epsilon_{optimo} = \frac{\pi}{2n}. \tag{4.41}$$

De esta forma garantizamos que ni siquiera la suma de los peores errores afecten al qubit mas significativo mas allá del umbral $\frac{\pi}{2}$.

4.5.7. Relación de ϵ con las puertas logico-cuánticas

Como vimos en la seccion **3.5.3** al descartar las rotaciones R_k que cumplen con la desigualdad $k > m$, la fase perdida en cualquier qubit es acotada por la cola geométrica que se muestra en la ecuación (4.28).

Al imponer $\Delta\phi_{max} \leq \epsilon$, donde $\Delta\phi_{max}$ es la cola geométrica mencionada anterioremente, y sustituyendo $\epsilon = \frac{\pi}{2n}$ tenemos:

$$\frac{2\pi}{2^m} \leq \frac{\pi}{2n} \implies 2^m \geq 4n \implies m = \log_2(4n). \tag{4.42}$$

Esto se interpreta como en cuanto más pequeño sea ϵ, mayor será m y, por lo tanto, más capas de rotaciones R_k se deben conservar.

4.5.8. Análisis de complejidad

En este caso el número de puertas gracias a la truncación se reduce a $O(n\log(n))$, la profundidad del circuito si es de forma secuencial tenemos que es $O(n\log(\frac{1}{\epsilon}))$ y si es en paralelo como las rotaciones cuyos controles y objetivos no coinciden conmutan, agrupandose, con ello la profundidad es $O(\log(n))$.

En cuestión de consumo de qubits se mantiene igual que el **Sumador con QFT** tanto en secuenical como en paralelo.

Recurso	**Valor**
Número de qubits	$n + m$ (para a y b)
Número de puertas	$O(n \log(n))$
Profundidad Secuencial	$O(n \log(\frac{1}{\epsilon}))$
Profundidad Paralelo	$O(\log(n))$
Ancillas	No requiere ancillas

Cuadro 4.2: Recursos del sumador con QFT con las rotaciones aproximadas

4.6. Comparación de sumadores cuánticos

En el ámbito de la computación cuántica, los sumadores representan operaciones fundamentales en la aritmética binaria, esencial para la construcción de algoritmos más complejos. En este contexto, se presentan tres implementaciones de sumadores cuánticos, cada uno con características y ventajas distintas.

El **sumador cuántico simple** es una versión cuántica directa del sumador completo clásico, implementado utilizando puertas Toffoli (CCNOT) y CNOT. Este sumador se basa en un enfoque tradicional de propagación de acarreo bit a bit, lo que lo hace conceptualmente sencillo y directo. Sin embargo, presenta varias limitaciones. Para sumar dos números de n bits, requiere $3n + 1$ qubits, ya que necesita qubits adicionales para almacenar y propagar el acarreo. Además, la complejidad temporal es de $O(n)$, lo que implica una dependencia secuencial en el cálculo de los acarreos. Esta característica impide que el sumador sea escalable de manera eficiente para números de mayor longitud, ya que cada operación depende de la anterior y no se puede paralelizar.

Por otro lado, el **sumador cuántico con QFT** mejora la eficiencia de la operación al utilizar la transformada de Fourier cuántica (QFT). En este caso, la QFT se aplica al primer número y las rotaciones de fase controladas son utilizadas para incorporar el segundo número en el proceso de suma. Posteriormente, se aplica la QFT inversa para obtener el resultado final. Este enfoque reduce significativamente el número de qubits necesarios, pues solo requiere $2n$ qubits para los registros de entrada y salida. Además, se elimina la necesidad de qubits de acarreo, lo que representa una ventaja considerable en términos de consumo de recursos. Sin embargo, este método no está

exento de limitaciones. La profundidad del circuito es de $O(n^2)$, debido a que las rotaciones de fase controladas no se ejecutan en paralelo. Esto genera una mayor complejidad temporal, lo que puede afectar su viabilidad en implementaciones prácticas con hardware cuántico, dado que un circuito más profundo implica una mayor susceptibilidad a errores y decoherencia.

Ahora, el **sumador cuántico con QFT en paralelo** toma la estructura del sumador con QFT y la optimiza al aplicar las rotaciones de fase controladas en paralelo, cuando estas no comparten qubits de control ni de destino. Gracias a la independencia de estas operaciones, es posible reorganizar las puertas de rotación en capas que se pueden ejecutar simultáneamente, reduciendo la profundidad del circuito a $O(n)$. Esta mejora en la paralelización es significativa, ya que permite realizar la operación de suma en un tiempo mucho menor que el sumador con QFT, manteniendo la misma cantidad de qubits requeridos.

Finalmente, el **sumador cuántico con QFT con aproximación de las rotaciones**, este método puede estar tanto en serie como en paralelo y en ambos casos el número de puertas es igual, en cambio, en profundiad de circuito cambia en caso de estar en serie, el circuito posee una profunidad de $O(n\log(n))$ y si se encuentra en paralelo la complejidad disminuye a $O(\log(n))$, este cambio aunque mejora la complejidad existe un margen de error en el cual el circuito puede fallar ya que se estan omitiendo algunas rotaciones, sin embargo, como se menciono en la sección **4.5** ya existe una forma de minimizar esa probabildiad de error. En términos de eficiencia, este es el método más adecuado cuando se busca minimizar el tiempo de ejecución, sobre todo en contextos donde la reducción de la profundidad del circuito es crucial para garantizar una alta fidelidad en la operación cuántica.

Sin embargo, en este último algoritmo ocurre algo interesante ya que por su complejidad analizada como función que es $n\log(n)$ tenemos que para el caso de que n este entre $1, 2, 3, 4$ no hay diferencia alguna entre aplicar el sumador QFT con o sin aproximaciones, en cambio entre más aumentemos n se vuelve más eficiente, para más detalle revisar **Apendice G**.

En la siguiente tabla se presenta una comparación resumida de los tres sumadores cuánticos descritos:

Característica	**Simple (VBE)**	**QFT exacto**	**QFT aprox.**	**QFT paralelo**
Qubits	$3n+1$	$2n$	$2n$	$2n$
Puertas	$O(n)$	$O(n^2)$	$O(n \log n)$	$O(n^2)$
Profundidad	$O(n)$	$O(n^2)$	$O(n \log n)$	$O(n)$
Ancillas	1	0	0	0

Cuadro 4.3: Comparación de complejidad de cuatro sumadores cuánticos.

4.7. Aplicación de Sumadores Cuánticos en el Algoritmo de Grover

El algoritmo de Grover representa uno de los desarrollos más relevantes en la computación cuántica, al proporcionar una mejora cuadrática para la búsqueda en bases de datos no estructuradas. Su eficiencia radica en la implementación de un oráculo cuántico capaz de identificar, mediante una inversión de fase, aquellos estados que satisfacen una condición lógica determinada.

Si bien en su forma básica dicho oráculo evalúa una función booleana simple, existen múltiples variantes y extensiones del algoritmo donde el oráculo involucra operaciones aritméticas. En tales escenarios, el diseño de circuitos que implementen estas operaciones de forma reversible es fundamental, siendo los sumadores cuánticos una herramienta clave.

A continuación, se analiza cómo los sumadores cuánticos descritos en las secciones anteriores pueden integrarse como subrutinas en la construcción del oráculo del algoritmo de Grover.

4.7.1. Sumador Cuántico Simple

El sumador cuántico simple implementado en la sección **4.2** utiliza bloques lógicos reversibles basados en puertas Toffoli y CNOT, siguiendo un esquema análogo al sumador completo clásico. Su estructura permite calcular la suma de dos registros de qubits, propagando el acarreo de forma secuencial mediante un registro adicional.

Este tipo de sumador resulta especialmente útil para construir oráculos cuánticos que requieren condiciones como $x + y = z$ o $x = a$, donde es necesario realizar sumas dentro del circuito. El resultado de la operación puede

compararse con un valor constante y, si se cumple la condición, se aplica una compuerta de fase controlada para marcar el estado objetivo. Posteriormente, se deshace la operación aritmética para preservar la reversibilidad del sistema.

Ventajas:

- Construcción robusta y directa.
- Utiliza puertas lógicas comunes en hardware cuántico actual.
- Baja sensibilidad al ruido.

Desventajas:

- Requiere un número considerable de qubits adicionales ($3n + 1$).
- La profundidad del circuito es lineal respecto al número de bits ($O(n)$).

4.7.2. Sumador Cuántico con QFT

El sumador con transformada de Fourier cuántica (QFT) presenta un enfoque alternativo que permite realizar la suma de dos registros sin necesidad de un registro auxiliar para el acarreo. Este sumador transforma el operando a al dominio de Fourier, aplica rotaciones de fase controladas por el operando b, y finalmente realiza la QFT inversa para obtener el resultado $a + b$.

Este método puede integrarse dentro del oráculo de Grover cuando se desea evaluar condiciones numéricas o realizar cálculos que dependan de sumas eficientes en cuanto a recursos cuánticos. Al reducir la cantidad de qubits utilizados, es especialmente adecuado para problemas en los que la memoria cuántica es limitada.

Ventajas:

- Requiere sólo $2n$ qubits.
- Reduce el uso de registros auxiliares.
- Ideal para sumas múltiples en oráculos complejos.

Desventajas:

- Alta sensibilidad a errores debido a las rotaciones de fase.
- Necesita rotaciones con precisión que puede ser difícil de implementar en hardware ruidoso.

4.7.3. Suma Cuántica con QFT en Paralelo

La versión paralelizada del sumador con QFT introduce una optimización clave en términos de profundidad de circuito. Al reorganizar las rotaciones de fase controladas, es posible ejecutar múltiples operaciones de manera simultánea, aprovechando la capacidad de paralelismo del hardware cuántico.

Esta implementación es particularmente valiosa en variantes del algoritmo de Grover que requieren la evaluación de múltiples condiciones en paralelo o la realización de operaciones aritméticas complejas dentro de una única llamada al oráculo. El enfoque paralelizado permite reducir la profundidad total del circuito, mitigando así los efectos de la decoherencia y el ruido.

Ventajas:

- Permite una ejecución más rápida y paralela de las operaciones.
- Reduce la profundidad del circuito en comparación con el sumador QFT tradicional.
- Adecuado para implementaciones sobre arquitecturas paralelas o distribuidas.

Desventajas:

- Requiere control preciso de rotaciones.
- Diseño del circuito más complejo y dependiente del backend cuántico.

4.7.4. Sumador con aproximación de rotaciones

Esta versión posee las mismas ventajas y desventajas que la **Suma con QFT en serie** y **suma con QFT en paralelo**, solo que ya sea en cualqueira de los dos casos se reduce la profundiad del circuito y permite ejecuciones más rápidas, por otro lado añade una desventaja que es que existe una pequeña probabilidad de error al momento de ejecución.

Capítulo 5

Conclusión

Los resultados obtenidos a lo largo de este estudio confirman la hipótesis inicial: la *Transformada de Fourier Cuántica* (QFT) y sus variantes ofrecen un camino más escalable para la aritmética cuántica que los sumadores reversibles tradicionales.

Al comparar el sumador de VBE, cuya profundidad crece linealmente con el número de bits $O(n)$, con el esquema basado en QFT —cuya profundidad y número de puertas son de orden $O(n^2)$ a causa de las rotaciones controladas en serie, se observa una ganancia clara al trasladar la operación al dominio de Fourier.

Ese beneficio se amplifica cuando se explota la conmutatividad de las rotaciones: reagrupándolas en capas independientes se obtiene la *suma con QFT paralela*, cuya profundidad vuelve a ser $O(n)$ sin aumentar el número de qubits.

Finalmente, al aplicar la regla de truncado y descartar las rotaciones cuyo ángulo sea inferior a un umbral ϵ, la complejidad en puertas se reduce a $O\big(n \log n\big)$ y la profundidad puede caer hasta $O(\log n)$ en su versión paralela; todo ello a cambio de una probabilidad de error acotada por $P_{\text{err}} \leq \frac{\epsilon^2}{4}$, mientras que elegir $\epsilon_{\text{opt}} = \frac{\pi}{2n}$ garantiza que la suma coherente de errores de fase nunca supere el umbral que provoque una lectura incorrecta del bit más significativo.

El marco teórico que sustenta esta optimización enlaza la desigualdad de Helstrom, que fija el límite fundamental de discriminación de estados cuánticos, con la aproximación cuadrática a baja fase, proporcionando un

criterio práctico para balancear fidelidad y costo computacional.

En conjunto, este trabajo demuestra que el bloque aritmético puede diseñarse para aprovechar tanto el paralelismo inherente de las puertas que actúan sobre subespacios disjuntos como la tolerancia controlada a errores de fase. Este enfoque reduce recursos críticos, profundidad y puertas controladas, sin requerir qubits auxiliares y habilita la integración de sumadores eficientes en oráculos numéricos, subrutinas de Grover y, en general, algoritmos cuánticos que demandan operaciones aritméticas repetidas. Por lo tanto, la tesis aporta una guía concreta para la construcción de circuitos cuánticos más rápidos y robustos, dejando abiertas líneas de trabajo sobre su mapeo a arquitecturas de conectividad limitada y su combinación con códigos de corrección de errores ligeros.

Apéndice A

Librerias utilizadas para las simulaciones

LIbrerias

```
[ ] from qiskit import QuantumCircuit
    from qiskit_aer.primitives import SamplerV2
    from qiskit.visualization import plot_histogram
    import numpy as np
```

Apéndice B

Programa QFT

B.1. Funciones

A continuación se muestra la función que realiza la QFT

Funciones

```
[ ] def QFT(n):
        """
        Implementa la Transformada de Fourier Cuántica (QFT) en los primeros n
        qubits del circuito.
        """
        qc = QuantumCircuit(n)

        for i in range(n):
            qc.h(i)
            for j in range(i+1, n):
                angle = np.pi / (2**(j-i)) # Calcular angulos para las rotaciones
                qc.cp(angle, j, i)  # Puerta de fase controlada

        # Inversión de bits
        for i in range(n//2):
            qc.swap(i, n-i-1)

        return qc
```

B.2. Llamado de funciones

Ahora se muetsra el llamado de la función QFT

```
[7] n= int(input("Ingrese el numero de qubits: "))

    qc = QFT(n)

    print(qc)

    qc.measure_all()

    #Construir la simulacion
    sampler = SamplerV2()
    sim = sampler.run([qc],shots=40000)

    #Realizar la simulacion
    result_ideal = sim.result()
    counts_ideal = result_ideal[0].data.meas.get_counts() #Se almacenan las mediciones en un diccionario
    print(counts_ideal)
```

Apéndice C

Funciones generales de los sumadores

```python
def decimal_to_binary_equal_bits(num1, num2):

    # Convertir los números a binario sin el prefijo '0b'
    bin1 = bin(num1)[2:]
    bin2 = bin(num2)[2:]

    # Determinar la longitud máxima entre los dos números
    max_length = max(len(bin1), len(bin2))

    # Ajustar ambos números para que tengan la misma cantidad de bits
    bin1 = bin1.zfill(max_length)
    bin2 = bin2.zfill(max_length)

    return bin1, bin2,max_length

def binary_to_decimal(lista_binaria):
    binario_str = "".join(map(str, lista_binaria)) # Convierte la lista a cadena
    Resultado = int(binario_str, 2)
    return Resultado

def binary_to_matrix(bin1, bin2):

    # Convertir las cadenas binarias en listas de enteros
    matrix1 = list(map(int, bin1))
    matrix2 = list(map(int, bin2))
    return matrix1, matrix2
```

Apéndice D

Sumador Cuántico Simple

D.1. Funciones para realizar la suma

```
def Sumador_Qubits(matrix_bits,max_length):

    #Arreglos par aalmacenar las posiciones de las puertas implementadas en Carry
    CCX_1 =[]
    CX = []
    CCX_2 = []

    #Calcular el numero de qubits que hay que tener
    bits_carry = max_length + 1
    qubits = 2*max_length + bits_carry

    #Inicializar el circuito
    qc = QuantumCircuit(qubits,max_length)

    #Inicializar los qubits que esten en el estado ket 1
    for i in range(len(matrix_bits)-1,-1,-1):
        if matrix_bits[i] ==1:
            qc.x(i)

    qc.barrier(range(qubits))
```

```
#Generalizacion para las puertas cuantica Carry
for i in range(max_length):
    #Ejecutar la primer puerta Toffoli para el Carry de manea automatica en la posicion que le corresponde
    qc.ccx(i,i + max_length,2*max_length + 1 + i)
    #Ejecutar C-Not para Carry de manera automatica en la posicion que le coresponde
    qc.cx(i,i + max_length)
    #Ejecutar la segunda puerta Toffoli para el Carry de manea automatica en la posicion que le corresponde
    qc.ccx(max_length + i,2*max_length + i,2*max_length + 1 + i)

    #Guardar las posiciones de la puertas implementadas en Carry para la reversion
    if i < (max_length-1):
        #Almacenar las posicones de la primer Toffoli
        ccx_1 = [i,i + max_length,2*max_length + i + 1]
        CCX_1.append(ccx_1)
        #Almacenar las posiciones de C-Not
        cx = [i,i + max_length]
        CX.append(cx)
        #Almacenar las posicones de la segunda TOffoli
        ccx_2 = [max_length + i,2*max_length + i,2*max_length + 1 + i]
        CCX_2.append(ccx_2)
qc.barrier(range(qubits))

#print(CCX_1,CX,CCX_2)

#C-Not
#print("Puerta C-Not:",max_length-1,2*max_length - 1)
qc.cx(max_length-1,2*max_length-1)
#print()
qc.barrier(range(qubits))
```

```
#Adder y carry inv
for i in range(max_length):
    #En esta parte se ejecuta la Sum  intercalado con Carry inverso para regresar los qubits de acarreo a su estado original
    qc.cx(max_length - i -1,2*max_length - i - 1)
    qc.cx(qubits - i - 2,2*max_length - i - 1)

    #If para intercalar la suma con carry inv
    if i < max_length-1:

        qc.ccx(CCX_2[max_length - 2 - i][0],CCX_2[max_length - 2 - i][1],CCX_2[max_length - 2 - i][2])
        qc.cx(CX[max_length - 2 -i][0],CX[max_length - 2 -i][1])
        qc.ccx(CCX_1[max_length - 2 - i][0],CCX_1[max_length - 2 - i][1],CCX_1[max_length - 2 - i][2])

return qc,bits_carry
```

D.2. Llamado de las funciones para la suma cuántica basica

```
num1 = 2
num2 = 1
bin1, bin2, max_length = decimal_to_binary_equal_bits(num1, num2)
#print(max_length)
print(f"A = {num1} en binario: {bin1}")
print(f"B = {num2} en binario: {bin2}")

matrix1, matrix2 = binary_to_matrix(bin1, bin2)

matrix1.reverse()
matrix2.reverse()
matrix_bits = matrix1 + matrix2

#print(matrix_bits)

qc, bits_carry = Sumador_Qubits(matrix_bits, max_length)
print(qc)

#qc.draw(output="text")

qc.measure_all()

#Construir la simulacion
sampler = SamplerV2()
sim = sampler.run([qc],shots=1)

#Realizar la simulacion
result_ideal = sim.result()
counts_ideal = result_ideal[0].data.meas.get_counts() #Se almacenan las mediciones en un diccionario
print(counts_ideal)
```

```
for clave, valor in counts_ideal.items():
    Bits = list(clave)

A = []
B = []
C = []

for i in range(len(Bits)):

    if i < bits_carry and Bits[i] == "1":
        B.insert(0,Bits[i])
    elif i < bits_carry:
        C.append(Bits[i])
    elif i <= 2*max_length:
        B.append(Bits[i])
    else:
        A.append(Bits[i])

print(A,B,C)

bits_out = [int(bit) for bit in B]

print(bits_out)

Resultado = binary_to_decimal(bits_out)

print(Resultado)
```

Apéndice E

Sumador Cuántico con QFT en serie

E.1. Funciones para realizar la suma

A continuación se muestran las funciones especificas necesarias para realizar la suma con QFT

```
def QFT(circuit,max_length):
    """
    Implementa la Transformada de Fourier Cuántica (QFT)
    en los primeros n qubits del circuito.
    """
    #circuit.h(0)
    for i in range(max_length):
        circuit.h(i)
        for j in range(i+1, max_length):
            angle = np.pi / (2**(j-i))
            circuit.cp(angle, j, i)  # Puerta de fase controlada

    # Inversión de bits
    for i in range(max_length//2):
        circuit.swap(i, max_length-i-1)

    return circuit

def Inv_QFT(matrix1,circuit,max_length):

    # Inversión de bits

    for i in range(max_length//2):
        circuit.swap(max_length-i-1, i)


    circuit.barrier(range(2*max_length))

    for i in reversed(range(max_length)):
        for j in range(i+1, max_length):
            angle = -np.pi / (2**(j-i))
            circuit.cp(angle, j, i)  # Puerta de fase controlada
        circuit.h(i)
    #circuit.h(0)
    return circuit
```

```
def Inv_QFT(matrix1,circuit,max_length):

    # Inversión de bits

    for i in range(max_length//2):
        circuit.swap(max_length-i-1, i)


    circuit.barrier(range(2*max_length))

    for i in reversed(range(max_length)):
        for j in range(i+1, max_length):
            angle = -np.pi / (2**(j-i))
            circuit.cp(angle, j, i)  # Puerta de fase controlada
        circuit.h(i)
    #circuit.h(0)
    return circuit
```

```
def Sumador_QFT(max_length,matrix_bits,matrix1):

    qubits = 2*max_length

    qc_sum = QuantumCircuit(qubits,max_length)

    for i in range(len(matrix_bits)-1,-1,-1):
        if matrix_bits[i] ==1:
            qc_sum.x(i)

    qc_sum.barrier(range(qubits))

    qc_sum = QFT(qc_sum, max_length)

    qc_sum.barrier(range(qubits))

    for i in range(max_length):
        l=max_length+i

        h= 0
        for j in range(l,2*max_length):

            angle = 2*np.pi / (2**(h+1))

            qc_sum.cp(angle,j,i)

            h+=1
    qc_sum.barrier(range(4))

    qc_sum = Inv_QFT(matrix1, qc_sum, max_length)

    return qc_sum
```

E.2. Llamado de funciones para realizar la suma

```
num1 = 2
num2 = 4
bin1, bin2, max_length = decimal_to_binary_equal_bits(num1, num2)
print(max_length)
print(f"A = {num1} en binario: {bin1}")
print(f"B = {num2} en binario: {bin2}")

matrix1, matrix2 = binary_to_matrix(bin1, bin2)

matrix1.reverse()
matrix2.reverse()
matrix_bits = matrix1 + matrix2

#qc=QFT(matrix1, max_length)

qc_sum = Sumador_QFT(max_length, matrix_bits,matrix1)

print(qc_sum)

print(qc_sum.draw(output='mpl'))

qc_sum.measure_all()

#Construir la simulacion
sampler = SamplerV2()
sim = sampler.run([qc_sum],shots=100)

#Realizar la simulacion
result_ideal = sim.result()
counts_ideal = result_ideal[0].data.meas.get_counts() #Se almacenan las mediciones en un diccionario
print(counts_ideal)
```

```
for clave, valor in counts_ideal.items():
    Bits = list(clave)

print(Bits)

A = []
B = []

for i in range(len(Bits)):
    if i < max_length:
        A.append(Bits[i])

    if i >= max_length:
        B.append(Bits[i])


print(A,B)

bits_out = [int(bit) for bit in B]

Resultado = binary_to_decimal(bits_out)

print(Resultado)
```

Apéndice F

Sumador Cuántico con QFT en paralelo

```
def Sumador_QFT(max_length,matrix_bits,matrix1):

    qubits = 2*max_length

    qc_sum = QuantumCircuit(qubits,max_length)

    for i in range(len(matrix_bits)-1,-1,-1):
        if matrix_bits[i] ==1:
            qc_sum.x(i)

    qc_sum.barrier(range(qubits))

    qc_sum = QFT(qc_sum, max_length)

    qc_sum.barrier(range(qubits))

        # Paralelización de rotaciones controladas
    for offset in range(1, max_length + 1):  # Cada capa (basada en distancia entre qubits)
        for i in range(max_length):
            control = max_length + i
            target = i
            if control + offset - 1 < 2 * max_length:
                angle = 2 * np.pi / (2 ** offset)
                qc_sum.cp(angle, control + offset - 1, target)

    qc_sum.barrier(range(4))

    qc_sum = Inv_QFT(matrix1, qc_sum, max_length)

    return qc_sum
```

Apéndice G

Analisis de propiedades del suamdor con rotaciones truncadas

Casos en los que truncar no da beneficios

En el sumador QFT sin truncar pueden aparecer R_k con $k = n, \ldots, 2n$. Con truncado $\frac{\varepsilon}{2n^2}$ se conservan solo las:

$$R_k \quad \text{con} \quad k \leq K_{\max}, \quad K_{\max} = \left\lfloor \log_2 \left(\frac{2\pi}{\varepsilon} \right) \right\rfloor = \lfloor \log_2(4n) \rfloor$$

Para saber en qué casos truncar y no truncar es lo mismo, necesitamos que $K_{\max} \geq n$:

$$\lfloor \log_2(4n) \rfloor \geq n \iff \log_2(4n) \geq n \iff 2^{\log_2(4n)} \geq 2^n \iff 4n \geq 2^n$$

Definimos la función:

$$f(n) = 2 + \log_2(n) - n$$

Buscamos que para $n > 1$ se cumpla que $f(n) \geq 0$.

$$
\begin{aligned}
&f(1) = 2 + 0 - 1 = 1 \geq 0,\\
&f(2) = 2 + 1 - 2 = 1 \geq 0,\\
&f(3) = 2 + \log_2(3) - 3 = 2 + 1{,}585 - 3 = 0{,}585 \geq 0,\\
&f(4) = 2 + 2 - 4 = 0 \geq 0,\\
&f(5) = 2 + \log_2(5) - 5 = 2 + 2{,}322 - 5 = -0{,}678 < 0 \quad \text{(no cumple)}.
\end{aligned}
$$

Por lo tanto:

$$1 \leq n \leq 4$$

Las rotaciones R_k coinciden entre el caso truncado y no truncado cuando $n \in \{1, 2, 3, 4\}$.

Demostración de que la truncación se vuelve más efectiva para n grandes

En la QFT exacta sobre n qubits, el número de rotaciones controladas es

$$T_{\text{exact}} = \sum_{j=1}^{n-1} (n - j) = \frac{n(n-1)}{2} = \Theta(n^2).$$

Si truncamos conservando solo interacciones hasta distancia m (es decir, R_k con $k \leq m$), el número de rotaciones que se conservan es

$$T_{\text{keep}} = \sum_{j=1}^{n-1} \text{mín}(m, n - j) = m(n - m) + \frac{(m-1)m}{2} = mn - \frac{m(m+1)}{2}.$$

Fracción conservada vs. omitida. La fracción conservada es

$$\frac{T_{\text{keep}}}{T_{\text{exact}}} = \frac{mn - \frac{m(m+1)}{2}}{\frac{n(n-1)}{2}} = \frac{2m}{n} - O\left(\frac{m^2}{n^2}\right).$$

Si elegimos $m = O(\log n)$ (caso típico en truncado por precisión), entonces

$$\frac{T_{\text{keep}}}{T_{\text{exact}}} = O\left(\frac{\log n}{n}\right) \xrightarrow[n \to \infty]{} 0.$$

Por tanto, la fracción omitida $1 - \frac{T_{\text{keep}}}{T_{\text{exact}}}$ tiende a 1, es decir, cuanto mayor es n, mayor porcentaje de puertas se ahorra. Además,

$$T_{\text{keep}} = mn - \frac{m(m+1)}{2} = \Theta(n\,m).$$

Con $m = \Theta(\log n)$ se obtiene $T_{\text{keep}} = \Theta(n \log n)$, frente a $T_{\text{exact}} = \Theta(n^2)$.

Error acotado y decreciente. El error de fase total por descartar distancias $> m$ se acota mediante la cola geométrica:

$$\theta_{\text{tail}} \leq \sum_{\ell=m+1}^{\infty} \frac{2\pi}{2^\ell} = \frac{2\pi}{2^m}.$$

Para ángulos pequeños, la probabilidad de error satisface

$$P_{\text{err}} \lesssim \frac{\theta_{\text{tail}}^2}{4} \leq \frac{\pi^2}{2^{2m-2}}.$$

Si fijamos $\varepsilon(n) \sim \pi/(2n)$ para controlar el error global, es suficiente elegir

$$m(n) = \left\lceil \log_2 \frac{2\pi}{\varepsilon(n)} \right\rceil = \lceil \log_2(4n) \rceil = \Theta(\log n),$$

y entonces

$$P_{\text{err}} \leq \frac{\pi^2}{4n^2} = O(n^{-2}) \xrightarrow[n\to\infty]{} 0.$$

Profundidad del circuito. Las rotaciones a igual "distancia" conmutan, por lo que se pueden agrupar en capas. Así:

Profundidad QFT exacta $= \Theta(n)$, Profundidad QFT truncada $= \Theta(\log n)$.

Conclusión. Con $m = \Theta(\log n)$ se cumple

$$\frac{T_{\text{keep}}}{T_{\text{exact}}} = O\left(\frac{\log n}{n}\right) \to 0, \qquad P_{\text{err}} = O\left(\frac{1}{n^2}\right) \to 0,$$

por lo que al crecer n la truncación es cada vez más efectiva: reduce de manera creciente el número de puertas (y la profundidad a $\Theta(\log n)$) manteniendo el error controlado y decreciente.

Apéndice H

Bibliografía

1. Nielsen, M. A., & Chuang, I. L. (2010). *Quantum computation and quantum information* (10th anniversary ed.). Cambridge University Press.

2. Preskill, J. (2018). Quantum computing in the NISQ era and beyond. *Quantum*, *2*, 79.

3. Grover, L. K. (1996). A fast quantum mechanical algorithm for database search. In *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing* (pp. 212–219). ACM.

4. Helstrom, C. W. (1976). *Quantum detection and estimation theory*. Academic Press.

5. Draper, T. G. (2000). Addition on a quantum computer. arXiv.

6. Vedral, V., Barenco, A., & Ekert, A. (1996). Quantum networks for elementary arithmetic operations. Physical Review A, 54(1), 147–153.

7. Gruska, J. (1999). Quantum computing. McGraw-Hill.