Universidad Autónoma de Madrid
Escuela politécnica superior

Grado Universitario en
Ingeniería Informática

Trabajo Fin de Grado

ALGORITMO DE DISEÑO MATRICIAL DE CIRCUITOS CUÁNTICOS

Javier París Uhryn
Dra. Estrella Pulido Cañabate

21 de junio de 2018

ALGORITMO DE DISEÑO MATRICIAL DE CIRCUITOS CUÁNTICOS

Javier París Uhryn
Dra. Estrella Pulido Cañabate

Escuela Politécnica Superior
Universidad Autónoma de Madrid
21 de junio de 2018

Resumen

Resumen

Este Trabajo de Fin de Grado tiene como objetivo el estudio de la computación cuántica y el desarrollo de herramientas y algoritmos basados en ella.

La computación cuántica es un paradigma computacional en la actualidad muy poco desarrollado, que basa su funcionamiento en fenómenos cuánticos, lo que permite la implementación de nuevos algoritmos con potencia de cálculo paralelizable mucho mayores que los obtenibles por ordenadores actuales.

El trabajo desarrollado comienza por el estudio y entendimiento del funcionamiento de un ordenador cuántico y sus posibilidades de uso. Este tipo de ordenadores y este nuevo paradigma conllevan un alto conocimiento matemático, así como conocimientos básicos sobre física cuántica y computación.

Una vez entendidas las bases de la computación cuántica, se estudia en profundidad ciertos algoritmos cuánticos, en particular, el algoritmo de Grover. Este es un algoritmo de búsqueda sobre un conjunto finito de elementos, con una complejidad computacional no alcanzable por un ordenador lógico, pero con dificultades a la hora de ser programado en un ordenador cuántico.

Como enfoque principal de este trabajo se encuentra la creación de un algoritmo lógico generalizable que automatiza la creación de circuitos cuánticos.

Se ha diseñado un nuevo algoritmo lógico que genera una concatenación de transformaciones realizables por un ordenador cuántico para generar circuitos de más alto nivel que los actualmente existentes. En concreto, el algoritmo propuesto es capaz de generar un circuito cuántico cuyo resultado equivalga a una matriz cualquiera dada.

Este algoritmo se ha demostrado matemáticamente a lo largo del trabajo y se ha implementado en un ordenador lógico, mediante el cual se ha probado su funcionalidad ejecutándolo para diversos casos de prueba sobre un simulador cuántico y sobre un ordenador cuántico real.

Palabras clave

Ordenador cuántico, computación cuántica, *qubit*, puerta cuántica, algoritmo cuántico, algoritmo de Grover, circuito cuántico.

Abstract

This End-of-Degree Project aims to study quantum computing and the development of tools and algorithms in it.

Quantum computing is a computational paradigm that is currently little developed, which bases its operation on quantum phenomena. This allows the implementation of new algorithms with parallelizable computing power greater than any current computer.

The work developed starts with the study of quantum computing and its functionality and usability. This kind of computers and this new paradigm require a high knowledge of maths, or basic knowledge about quantum physics and computer science.

Once knowing the basis of quantum computing, the study focuses on quantum algorithms and Grover's algorithm in particular. This is a search algorithm over a non sorted set of elements with lower coplexity than any possible logical program. But it has its difficulties when it needs to be implemented in a real quantum computer.

The main focus of this work is the creation of a generalizable logical algorithm that automates the creation of quantum circuits. A new logical algorithm has been designed that generates a concatenation of quantum transformations reliazable by a quantum computer to generate circuits of higher level than those currently existing. In particular, the proposed algorithm is capable of generating a quantum circuit which results to be equivalent to any given matrix.

This algorithm has been demonstrated mathematically throughout the work and has been implemented in a logical computer, through which its efficiency has been proven with a quantum simulator and with a real quantum computer.

Key words

Quantum computer, quantum computing, qubit, quantum gate, quantum algorithm, Grover's algorithm, quantum circuit.

Agradecimientos

Finalizado el que ha sido el proyecto más importante de mi carrera (por ahora) hay mucha gente a la que me gustaría agradecer el haberme ayudado y haberme apoyado para conseguirlo.

Gracias a IBM, al equipo de desarrollo de QISKit y a la comunidad Q-Network, en especial a Juan Gómez, por toda la ayuda y el esfuerzo dedicado y por hacer posible este y muchos otros proyectos.

Muchas gracias a Gonzalo y a muchos otros profesores y alumnos de la UAM, que han luchado e indagado en pos de ayudarme. Y a mi padre, que se devanó los sesos como el que más para poder echarme una mano.

Un agradecimiento especial para Francisco y Estrella, por haberme abierto la posibilidad de realizar este proyecto, y por ejercer de mentores y ayuda desde su comienzo.

Y muchas gracias a mis compañeros de la *cátedra UAM-IBM* por su ayuda, a mis queridos *yatekomo* por aguantar mis divagaciones cuánticas, a mis *serious math stuff* por ayudarme con cuentas matemáticas más allá de mi entendimiento, a la *familia feliz* por todos esos buenos momentos, a mi *pupi* por motivarme en cada paso para conseguirlo, y en especial a Antonio, quien considero un amigo y figura referente. Y por supuesto a toda mi familia, por vuestro cariño y apoyo constante y por aguantarme día a día.

Índice general

Índice de figuras

Índice de tablas

Definiciones

- **Algoritmo cuántico**: Conjunto teórico de transformaciones ordenadas sobre uno o varios qubits para obtener un resultado concreto.
- **Circuito cuántico**: Concatenación de puertas cuánticas que afectan a uno o varios qubits.
- **Esfera de Bloch**: Representación geométrica de una esfera de radio 1, en la cual puede representarse el comportamiento teórico de un *qubit*.
- **Estado cuántico**: Estado en el que se encuentra un qubit (es único en tiempo fijo) y del que depende el valor real del qubit al medirlo.
- **Estado inicial básico**: Estado de menor energía de los qubits en un ordenador cuántico o estado de los qubits al iniciar un programa cuántico. Se representa mediante el estado $|0..., 0\rangle$.
- **Ordenador cuántico**: Máquina capaz de llevar entrelazamientos y transformaciones sobre *qubits* y capaz de medir los mismos.
- **Profundidad**: Número de puertas cuánticas básicas que contiene un circuito cuántico.
- **Programa cuántico**: Conjunto ordenado de transformaciones sobre uno o varios qubits (similar a un circuito cuántico).
- **Puerta cuántica**: Transformación sobre uno o varios qubits.
- **Puerta cuántica básica**: Puerta que se da por hecho estará implementada físicamente sobre un ordenador cuántico.
- **QCMD**: *Quantum Circuit Matricial Design algorithm* o Algoritmo de Diseño Matricial de Circuitos Cuánticos.
- **Qubit**: Unidad elemental de la computación cuántica con un valor probabilístico medible entre 0 o 1. En castellano se denomina Cúbit, aunque se considera aceptado Qubit y es el término utilizado en este documento.

Glosario matemático

- $\langle\phi|\psi\rangle$: Notación de Dirac. $\langle\phi|$ representa una transformación y $|\psi\rangle$ un estado cuántico.
- $\epsilon(\alpha)$: $\epsilon(\alpha) = e^{i\alpha}$.
- $O()$: Notación O grande o de *cota superior asintótica*.
- P_x: Representa la profundidad de la puerta x.
- $U(\alpha)$: Giro de un qubit sobre el eje U de ángulo α.
- **Vectores y matrices**: Los vectores se ha representado contenidos en paréntesis () y las matrices contenidas en corchetes [].

1 Introducción

1.1. Motivación

La computación cuántica podría considerarse a día de hoy una de las más plausibles aplicaciones de las teorías físicas más modernas y un campo con un esfuerzo en investigación que crece día a día. Este paradigma de computación casi recién nacido se encuentra en un estado muy básico, donde la programación y la física se enfrentan a problemas fuera de nuestro entendimiento, en una carrera por conseguir el primer ordenador cuántico funcional, capaz de realizar cálculos en cuestión de segundos que antes se creían imposibles por el tiempo que necesitaban.

La motivación principal que llevó a elegir este campo de estudio fue la posibilidad de contribuir en un paradigma de computación, hasta la fecha casi únicamente estudiado por los físicos. Parece una rama de estudio con infinidad de posibilidades, ya que todo es nuevo y en constante crecimiento; así como un reto personal ante la idea de enfrentarse a fenómenos físicos que nos rodean y que escapan a nuestra razón.

La idea que persigue la computación cuántica es la *paralelización perfecta* de cálculos o algoritmos muy específicos, los cuales son aplicables a la resolución de problemas tan particulares como importantes, como la famosa **factorización de números primos**, lo que conllevaría un colapso de la criptografía actual usada en internet. Pero esta tecnología aún no se ha desarrollado lo suficiente como para permitir este tipo de cálculos, debido a limitaciones del *hardware* actual, así como del *software* y la teoría y algoritmia que la rodea.

La intención de este trabajo es ayudar a resolver ciertos aspectos relativos al *software* de este paradigma de computación desde un punto de vista teórico pero aplicable, para intentar dar un uso más simple y extendido a dicha tecnología.

1.2. Objetivos

La idea inicial de este Trabajo de Fin de Grado es, partiendo del estudio del **algoritmo de Grover**, algoritmo de búsqueda en computación cuántica, buscar aplicaciones para dicho algoritmo o diseñar nuevos algoritmos basados en el ordenador cuántico.

El descubrimiento de que la mayor parte de la teoría relativa a este algoritmo ya estaba desarrollada y era muy difícil innovar sobre ello, hizo reconducir la investigación a la implementación desde un ordenador real de un algoritmo genérico para construir cualquier algoritmo cuántico.

Análogamente a la computación lógica, los ordenadores cuánticos funcionan mediante qubits (en español *cúbit*, es el análogo a un bit lógico. En este documento se usará la terminología inglesa) y puertas cuánticas (en vez de puertas lógicas). El objetivo principal de este trabajo es el diseño de un nuevo algoritmo **lógico** que sea capaz de generar automáticamente las *puertas cuánticas* necesarias para crear cualquier *programa* o *circuito* cuántico, que implemente un determinado algoritmo o cálculo sobre un computador cuántico.

1.3. Contenido

Esta memoria está dividida en los siguientes capítulos:

- **Estado del arte**: En este capítulo se hará una breve introducción al estado en el que actualmente se encuentra la computación cuántica, así como a algoritmos e investigaciones sobre el campo.
- **Introducción a la computación cuántica**: En este capítulo se hace una breve introducción a los conocimientos básicos matemáticos, físicos y computacionales necesarios para poder entender la computación cuántica, su utilidad, importancia y estado actual.
- **Algoritmo de Diseño Matricial de Circuitos Cuánticos**: En este capítulo se explicará paso a paso el funcionamiento del algoritmo diseñado, y se incluirán demostraciones matemáticas de su funcionamiento.
- **Implementación empírica del algoritmo**: En este capítulo se explica de forma resumida la implementación propia de este algoritmo y las pruebas realizadas sobre el simulador y el ordenador cuántico de IBM.
- **Conclusiones y trabajo futuro**: Por último, en este capítulo se recogen las diferentes conclusiones extraídas y se identifican posibles puntos de mejora y el trabajo futuro relativo a este proyecto.

2
Estado del arte

2.1. Ordenador cuántico

La computación cuántica es un paradigma de programación que basa su funcionamiento en la idea abstracta de *Máquina de Turing Cuántica*, la cual es capaz de realizar cierto tipo de cálculos de forma mucho más veloz que un ordenador convencional basado en fundamentos electromagnéticos.
Para llevar a cabo este tipo de cálculos es necesario un tipo de máquina denominada *ordenador cuántico*, el cual explota los fenómenos relativos a la física cuántica. Actualmente estos efectos solo son perceptibles en partículas subatómicas y bajo condiciones extremas como altos gastos de energía o temperaturas cercanas al 0 absoluto (en la actualidad se usan entornos a menos temperatura que el espacio abierto).
Todo esto ha generado la conocida como *carrera por la supremacía cuántica*. Esto es una *competición* por parte de diversas empresas o instituciones por conseguir un ordenador cuántico de un *tamaño* (en número de qubits y tasa de error) capaz de realizar cálculos que escapan a cualquier intento de simulación por parte de un ordenador actual[2].

En esta carrera se pueden encontrar grandes gigantes informáticos como *Intel*, que anunció a principios del año 2018 su intención de construir un ordenador cuántico de 49 qubits, o *Google*, que alrededor de las mismas fechas, anunció su intención de disponer durante este año de un ordenador cuántico operativo de 72 qubits[3]. Existen también empresas como *D-Wave*, centrada en el diseño y construcción de computadoras cuánticas[4], con colaboraciones con la *Nasa*, *Google* y *USRA* entre otros. Cabe destacar entre estas empresas a *IBM*, la cual cuenta con un servicio web mediante el cual se puede hacer uso de sus ordenadores cuánticos de manera pública y gratuita. Este servicio web se ha utilizado en este trabajo para probar empíricamente el resultado del algoritmo implementado.

2.2. Algoritmos cuánticos

Los algoritmos cuánticos son algoritmos diseñados para ser ejecutados en un ordenador cuántico, y de esta forma aprovechar los fenómenos cuánticos para realizar cálculos en segundos que en ordenadores convencionales llevarían millones de años. Existen gran cantidad de algoritmos teóricos [5], como pueden ser el **algoritmo de Grover**, un algoritmo de búsqueda sobre

una secuencia desordenada, o el **algoritmo de Shor** que es capaz de factorizar un número en factores primos.

En general, los algoritmos cuánticos se usan en campos orientados a la *inteligencia artificial*[6] y a la simulación de fenómenos cuánticos. Por ejemplo, en la actualidad son usados para la simulación de comportamientos de ciertos compuestos químicos.

2.3. Diseño de circuitos

Este *Trabajo de Fin de Grado* se ha centrado en el diseño de un algoritmo orientado a la computación cuántica mediante el cual se genera automáticamente un circuito de puertas cuánticas que implementa cualquier algoritmo cuántico con matriz conocida. Esto en la actualidad se lleva a cabo de manera manual por los programadores cuánticos.
Se ha investigado de forma exhaustiva para encontrar un algoritmo o proceso que realizase esta función. Se ha investigado en libros [7] y artículos y se ha consultado mediante internet en distintas páginas y foros, como por ejemplo en *Slack - QISKit*[8], foro online formado por profesionales y estudiantes del campo de la computación cuántica que usan la *API* de *Python QISKit* desarrollada por *IBM* para el uso de su ordenador cuántico de forma online. Tras estas investigaciones no se ha encontrado un algoritmo genérico (para cualquier número de qubits) que implemente esta función.

Lo más cercano a este algoritmo que se ha podido encontrar ha sido un artículo de investigación donde se explica cómo implementar ciertos algoritmos para una máquina de 5 qubits[9] y que hace referencia a la descomposición de Schmidt[10], operación matemática que sirve para descomponer una matriz generada por un producto tensorial. Pero este método encuentra una limitación a la hora de resolver grandes sistemas de ecuaciones cuando se aumenta el número de qubits del ordenador.

2.4. Herramientas

Para la implementación del algoritmo diseñado, se ha usado el lenguaje de programación *Python 3.5 :: Anaconda 4.2* y librerías estándar del mismo, sobre un entorno *Ubuntu 16.04*.

Para llevar a cabo las pruebas, se ha usado la *API* de *Python QISKit*[11] desarrollada por *IBM* para usar simuladores cuánticos y un servicio web para lanzar pruebas contra el ordenador cuántico *IBM-Q* de 5 qubits.

Para llevar a cabo la realización de esta memoria, se ha utilizado el lenguaje LaTeXy ciertas librerías como *blochsphere*[12] para poder dibujar *esferas de Bloch*, o librerías externas como *qpic*[13] para crear las imágenes de los circuitos cuánticos mostrados.

3

Introducción a la computación cuántica

En esta sección se hace una introducción a los conceptos técnicos necesarios para entender la motivación y el diseño del algoritmo. Se abarcarán tanto los fenómenos físicos que se producen en el ordenador cuántico como los principios matemáticos necesarios para entender todos los teoremas y demostraciones relativos a la computación cuántica de este documento. El lector debería tener una base de conocimiento científico en los campos de matemáticas, física e informática para entender toda la argumentación, y sería imposible abarcar todos los temas necesarios, por lo que se aconseja que el lector esté familiarizado con **números complejos**, operaciones con **matrices** y con ideas básicas de **algoritmia** y **programación**.
La física cuántica es un campo muy extendido y en actual desarrollo, y con una complejidad matemática y física muy alta, por lo que la siguiente sección intentará no ser excesivamente formal, centrándose sobre todo en las ideas claves mínimas para entender la computación cuántica.

3.1. Mecánica cuántica

La mecánica cuántica es una de las ramas más modernas de la física actual. Nacida en los principios del siglo XX, y eclipsada parcialmente en sus inicios por otra teoría que surgió a la par: la teoría de la relatividad, esta teoría intenta explicar el comportamiento de las partículas subatómicas tales como quarks, electrones, etc.
Esta teoría surgió como intento de explicar el problema de la radiación de cuerpo negro 1900, por Max Plank, aunque las primeras formulaciones matemáticas no llegaron hasta el año 1920.

Fenómenos cuánticos A continuación enumeramos los fenómenos cuánticos más relevantes e importantes a la hora de trabajar con un ordenador cuántico[14].

- **Principio de superposición**: Según este principio, un elemento puede poseer **simultáneamente** dos o más valores (infinitos).
- **Principio de entrelazamiento**: Este fenómeno cuántico introducido por Erwin Schrödinger dice que dos o más partículas entrelazadas no pueden considerarse como partículas individuales, sino como un sistema con una *función de onda* única para todo el sistema.
- **Colapso de la función de onda**: Esta teoría implica que la función de onda de un sistema *colapsa* o *toma* un valor concreto una vez se interactúe con ella, por ejemplo, se haga una medida. Esto hace que el sistema pierda su *estado de superposición*.

- **Principio de decaimiento o emisión espontánea**: Según este fenómeno, un átomo, molécula, etc. excitado tiende a pasar a un estado de energía más bajo, cumpliendo así el *principio de conservación de la energía*. Aunque este principio no afecta directamente al uso teórico del ordenador, es muy importante a la hora de usar el ordenador real.

3.2. Ordenador cuántico

La computación cuántica es una rama dentro de la computación que basa su funcionamiento físico en fenómenos cuánticos, en vez de en fenómenos electromagnéticos, como la computación lógica (también denominada computación binaria, clásica, electrónica o discreta).
La teoría cuántica es algo nuevo y complejo que surgió a principios del siglo XX (aproximadamente 1920). En la década de los 80 se pensó en una aplicación de la mecánica cuántica para crear una máquina de cálculo, lo cual se atribuye a Paul Benioff. En 1985, David Deutsch presentó el diseño de la llamada **Máquina Cuántica de Turing**, que sería el equivalente cuántico a la *Máquina de Turing*, y el pilar de la computación cuántica.

Qubit El elemento básico de la computación cuántica es el **qubit**. El *bit* lógico puede almacenar un valor 0 o 1. Análogamente, el qubit almacena **todos** los valores entre 0 y 1 **a la vez**. Un qubit se encuentra en un mismo momento en **un solo** estado cuántico, y este estado es el que determina la probabilidad de que el qubit tenga uno u otro valor al ser medido.
Al igual que en la computación lógica, existen una serie de *puertas* que transforman el estado cuántico de un qubit.

Ordenador cuántico Es una máquina compuesta de qubits, la cual basa su funcionamiento en medir el estado de un qubit en un momento dado una vez aplicadas las transformaciones (puertas) pertinentes, y pasa el resultado de esta medida a un ordenador lógico ligado. Esta medición es un bit con valor 0 o 1, ya que pierde su estado de superposición al ser medido (colapsar). Este conjunto de transformaciones es lo que se denomina algoritmo cuántico.

Uso en paralelización El hecho de que un qubit contenga más de un valor a la vez (y no sólo uno, sino todos) es relacionable con la idea de la *paralelización computacional*.
Cualquier *valor*, ya sea número, carácter o texto (finito), puede ser codificado en una cadena finita de bits. Si imaginamos que cada uno de nuestros bits puede contener un 0 y un 1 a la vez, podemos deducir que tenemos en una cadena finita de bits **todos** los valores posibles de dicho número, carácter o texto.
Hasta ahora, los sistemas de paralelización computacional se basan en usar varios procesadores, los cuales llevan a cabo el mismo cálculo (o aproximado), y en un sistema que sea capaz de organizar los cálculos de cada uno y de recoger y reorganizar sus resultados. El hecho de poder contar con un elemento que tenga varios valores a la vez permite lo que se conocería como *paralelización perfecta*.

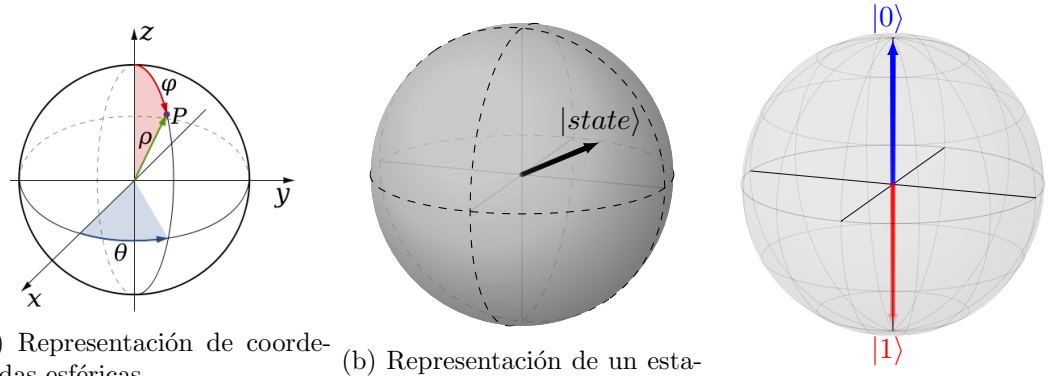
Resultado probabilístico El qubit contiene todos los valores a la vez, pero una vez se tome una medida sobre él estos valores colapsan a un único valor. En nuestro caso los denotaremos como 0 y 1 para seguir la analogía con la computación lógica. La idea principal es que podemos modificar nuestros qubits para que se acerquen más al estado 0 o al estado 1. Esto implica que en el momento de nuestra medición, el resultado es, (con un cierto sesgo), **aleatorio**. Esto choca con la idea básica de la computación lógica y la teórica *Máquina de Turing*, pues no podemos anticipar el resultado de un cálculo antes de haberlo realizado.

Utilidad Existen gran variedad de algoritmos y cálculos realizables por un ordenador cuántico que serían impensables para un ordenador lógico debido al tiempo que requeriría ejecutarlos. Entre ellos se encuentra el *Algoritmo de Grover* (capítulo A), o uno de los más famosos algoritmos debido a su importancia en la criptografía actual, el *Algoritmo de Shor*.
(Para mas información sobre algoritmos cuánticos, consulte la bibliografía [5]).

Limitaciones Existen varias limitaciones a la hora de trabajar con computación cuántica. Entre las limitaciones debidas al específico y complejo hardware que necesita una máquina de estas características, encontramos el del decaimiento cuántico, lo que hace muy difícil mantener el valor del estado de los qubits por mucho tiempo. En este momento se consigue mediante el uso de superconductores y temperaturas cercanas al 0 absoluto.
También existen problemas a la hora de realizar mediciones sobre los qubits. Estas mediciones de estados cuánticos se llevan a cabo sobre partículas de tamaños subatómicos.
Existen también problemas debidos al software, como puede ser el hecho de que el resultado sea aleatorio, lo que obliga a repetir un mismo cálculo muchas veces para asegurarse (nunca al 100 %) de que el resultado es fiable.
También existe la limitación de que el resultado de una operación cuántica debe ser siempre **reversible**, contrariamente a lo que ocurre en una operación lógica, como podría ser la operación *AND*. Esto implica que en computación cuántica no se puede copiar información. Por ejemplo, no se puede copiar el estado de un qubit a otro.

3.3. Esfera de Bloch

La Esfera de Bloch es un concepto matemático utilizado en la mecánica cuántica para describir el comportamiento de un estado cuántico. Esta esfera ayuda a representar de una forma visual y cómoda un estado cuántico, como puede ser un qubit[15].



(a) Representación de coordenadas esféricas.

(b) Representación de un estado cualquiera.

(c) Estados $|0\rangle$ Norte y $|1\rangle$ Sur.

Figura 3.1: Representación de una Esfera de Bloch.

Qubit Un qubit puede representarse como un vector unitario (módulo 1) con origen en el origen de coordenadas. De esta manera vemos que cualquier qubit puede representarse como un punto en la superficie de una esfera de radio 1 (figura 3.1b), o una **Esfera de Bloch**. Este qubit está representado unívocamente mediante dos valores, una *elevación* sobre el ecuador, representado como φ y un *acimut* o *fase* representado como θ (figura 3.1a).
Se podrían usar 3 valores para representar un estado, ya sea mediante coordenadas cartesianas o mediante coordenadas esféricas, pero uno de esos tres valores siempre va a ser dependiente de los otros dos (debido a que el radio de la esfera es constante), por lo que es innecesario.
Al trabajar matemáticamente con un qubit, por sencillez a la hora de aplicar transformaciones,

este se representa mediante un vector complejo, y no mediante estos dos ángulos.
Vamos a entender un qubit como un punto en la *Esfera de Bloch*. Si este punto se encuentra en el punto *Norte* de la Esfera (figura 3.1c, azul) este qubit tendrá valor *teórico* 0 (100 % de probabilidad de ser 0) y se representa con el nombre $|0\rangle$ (Este valor es puramente teórico. Debido a errores de medición y al fenómeno de decaimiento no se puede asegurar 100 % el valor de un qubit antes de medirlo). Análogamente, si el qubit está en el punto *Sur* (figura 3.1c, rojo) tendrá valor 1 y se llamará $|1\rangle$.

Para cualquier otro punto en la esfera, el qubit se puede representar como una combinación de los estados $|0\rangle$ y $|1\rangle$ de forma que cada estado se representa mediante un vector $\begin{pmatrix} a \\ b \end{pmatrix}$, que representa la combinación lineal $a\,|0\rangle + b\,|1\rangle$. Estos dos valores representan un punto en la esfera de Bloch de forma que $a = cos(\frac{\varphi}{2})$ y $b = sin(\frac{\varphi}{2})e^{i\theta}$, por lo que se cumple que $|a|^2 + |b|^2 = 1$ (necesario para que el vector sea unitario).

Transformaciones Las transformaciones básicas que existirían en un ordenador cuántico son representables con giros sobre los ejes de la esfera. De esta forma, un giro de 180 grados sobre el eje X convertiría un qubit $|0\rangle$ en $|1\rangle$ y viceversa, permitiéndonos modificar el valor de nuestro qubit. Nótese que un giro sobre el eje Z no modifica el valor del qubit, pero sí su estado.

Más dimensiones Al igual que en la computación lógica, para dar sentido a un programa o algoritmo es necesario más de un valor. En computación cuántica se pueden representar los qubits como *Esferas de Bloch* individuales mientras no exista conexión entre estos, pero en el momento en que un qubit entra en relación con otro esta representación deja de tener validez; necesitaríamos una 5-esfera (esfera de 5 dimensiones), para usar su superficie de 4 dimensiones. Por eso, en cuanto se trabaja sobre más de un qubit entrelazado, se usa notación matemática y se deja aparte la intuición espacial.

3.4. Notación matemática e interpretación matricial

Bra-Ket En esta sección introduciremos la notación matemática necesaria para entender los razonamientos matemáticos del algoritmo. Tanto en mecánica cuántica como en computación cuántica se usa la notación bra-ket $\langle\phi|\psi\rangle$ o **notación de Dirac** para representar valores cuánticos. La notación *bra* $\langle\phi|$ se usa para representar transformaciones o aplicaciones sobre un estado (matriz), y la notación *ket* $|\psi\rangle$ para representar estados cuánticos (vector).

Ket-Estado Un estado cuántico de un solo qubit se representa mediante dos ángulos *acimut-elevación* o mediante un vector complejo unitario de dimensión 2. Es difícil establecer una relación concreta entre los valores de este vector y la representación del qubit, pero aproximadamente puede entenderse que la primera dimensión representa una proporción entre 0 y 1 la cercanía del estado al norte de la esfera, y la segunda dimensión (compleja) representa la *fase* o giro sobre el eje Z del qubit.
En la tabla 3.I se representan los estados básicos de la esfera (estos son los más representativos, formados por el corte de los ejes con la esfera) y en la figura 3.2 se muestran estos estados localizados en la esfera.

El estado $|x\rangle$ en la tabla 3.I es un ejemplo general sobre un estado cuántico cualquiera, y los valores que se esperarían de él. El vector debe ser unitario, es decir, $cos^2(\varphi/2) + sin^2(\varphi/2) = 1$. Por ejemplo, si el estado $|x\rangle$ representase al estado $|i\rangle$, φ tendría valor $\pi/2$ y por lo tanto su probabilidad de ser 0 sería $cos^2(\pi/4) = 0{,}5$.

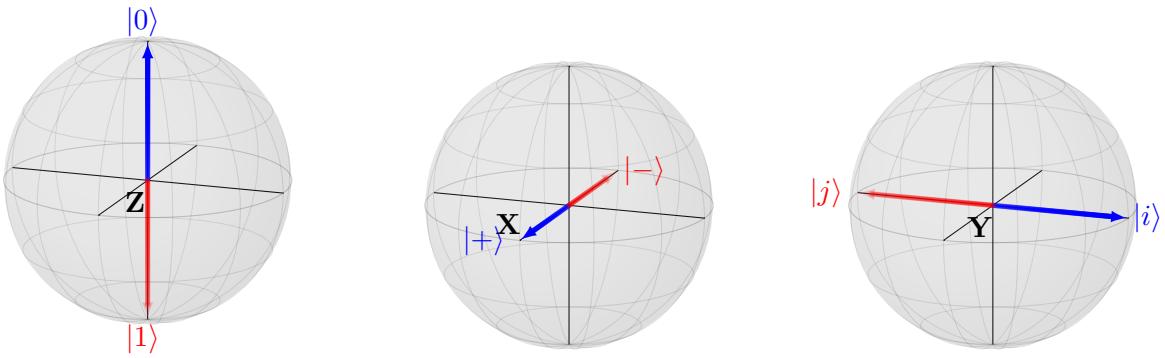
Se puede ver que los estados que se encuentran sobre el ecuador tienen las mismas probabilidades de ser 1 como de ser 0. Para denotar un estado se puede también usar la suma de distintos estados. Por ejemplo, el estado $|+\rangle = |0\rangle \frac{1}{\sqrt{2}} + |1\rangle \frac{1}{\sqrt{2}}$.

Cuando se aumenta la dimensión (el número de qubits), el vector que representa a un estado se duplica. De este modo, el estado $|00\rangle$ que representaría dos qubits en el *Norte* de la esfera, tendría una representación como vector $\begin{pmatrix} 1 & 0 & 0 & 0 \end{pmatrix}^t$ (^{t} indica transpuesto, se usa esta notación para evitar escribir vectores en vertical), y el estado $|11\rangle$ se representaría con el vector $\begin{pmatrix} 0 & 0 & 0 & 1 \end{pmatrix}^t$. Esto genera un problema a la hora de trabajar teóricamente con la computación cuántica, ya que, teniendo n qubits, trabajamos con vectores de 2^n dimensiones, lo que complica la simulación en ordenadores lógicos.

nombre	$\|0\rangle$	$\|1\rangle$	$\|+\rangle$	$\|-\rangle$
vector	$\begin{pmatrix} 1 \\ 0 \end{pmatrix}$	$\begin{pmatrix} 0 \\ 1 \end{pmatrix}$	$\begin{pmatrix} 1 \\ 1 \end{pmatrix} \frac{1}{\sqrt{2}}$	$\begin{pmatrix} 1 \\ -1 \end{pmatrix} \frac{1}{\sqrt{2}}$
Probabilidad de tener valor "0"	100 %	0 %	50 %	50 %
Probabilidad de tener valor "1"	0 %	100 %	50 %	50 %

nombre	$\|i\rangle$	$\|j\rangle$	$\|x\rangle$
vector	$\begin{pmatrix} 1 \\ i \end{pmatrix} \frac{1}{\sqrt{2}}$	$\begin{pmatrix} 1 \\ -i \end{pmatrix} \frac{1}{\sqrt{2}}$	$\begin{pmatrix} cos(\varphi/2) \\ sin(\varphi/2)e^{i\theta} \end{pmatrix}$
Probabilidad de tener valor "0"	50 %	50 %	$cos^2(\varphi/2)$
Probabilidad de tener valor "1"	50 %	50 %	$sin^2(\varphi/2)$

Tabla 3.I: Tabla con los estados cuánticos estándar



(a) Estados $|0\rangle$ y $|1\rangle$. (b) Estados $|+\rangle$ y $|-\rangle$. (c) Estados $|i\rangle$ y $|j\rangle$.

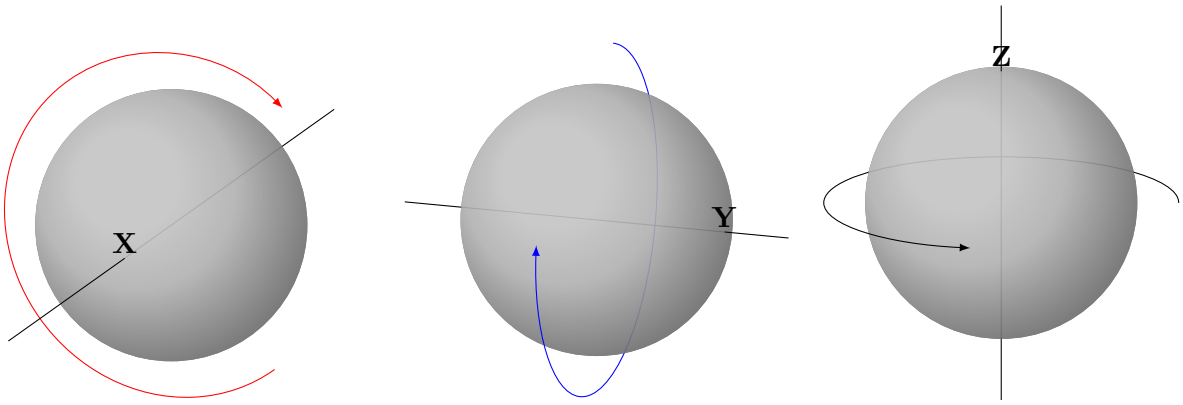
Figura 3.2: Representación de los estados básicos de un qubit.

Bra-Transformación Las transformaciones que se pueden aplicar sobre un estado cuántico son giros sobre uno de los ejes de la *Esfera de Bloch*, o una concatenación de varios giros, lo que equivale también a un giro.
Estos giros se representan mediante una matriz cuadrada de 2×2. Estas son matrices unitarias, matrices cuyo módulo es 1 y que cumplen $AA^* = A^*A = I$, donde A^* representa la matriz conjugada transpuesta.
La implementación a nivel físico de estas transformaciones puede variar dependiendo del hardware utilizado, pero se puede generalizar el uso de 3 transformaciones básicas (o como mínimo dos de ellas, puesto que aplicando giros sobre 2 ejes distintos se puede obtener cualquier giro en la esfera) que son los giros sobre los ejes principales de la Esfera. En la tabla 3.II se muestran

las distintas representaciones matriciales para los giros que se puede realizar, y en la figura 3.3 se muestran dichos giros. La demostración de que estas matrices representan giros se puede ver en el anexo B.[16]

X(α)	Y(β)	Z(γ)
$\begin{bmatrix} cos(\frac{\alpha}{2}) & sin(\frac{\alpha}{2})i \\ sin(\frac{\alpha}{2})i & cos(\frac{\alpha}{2}) \end{bmatrix}$	$\begin{bmatrix} cos(\frac{\beta}{2}) & -sin(\frac{\beta}{2}) \\ sin(\frac{\beta}{2}) & cos(\frac{\beta}{2}) \end{bmatrix}$	$\begin{bmatrix} 1 & 0 \\ 0 & e^{i\gamma} \end{bmatrix}$

Tabla 3.II: Giros básicos. *X(α) denota un giro de α grados sobre el eje X*



(a) Giro X. (b) Giro Y. (c) Giro Z.

Figura 3.3: Representación de los giros básicos.

3.5. Puertas cuánticas

Ya se ha explicado que las transformaciones de estados son giros sobre los ejes de la Esfera. Dos giros sobre distinto eje concatenados (uno detrás de otro) generan una transformación sobreyectiva, esto es, pueden transformar cualquier estado en cualquier otro dentro de la *Esfera de Bloch* (demostración en B). Pero al hablar sobre computación cuántica, es más común referirnos a esas transformaciones como **puertas cuánticas**, que harían las veces de *puertas lógicas* en la computación lógica. Hay dos tipos de puertas: aquellas que afectan a un solo qubit, y una puerta especial que afecta a dos qubits.

Puertas Unitarias Las puertas unitarias son aquellas que afectan a un solo qubit, como pueden ser los giros representados en la figura 3.3, aunque comúnmente en la computación cuántica se tiende a usar **puertas estándar** ya existentes en vez de los giros parametrizados sobre los distintos ejes.
Cuando se habla de *puerta cuántica* y no de *transformación* se suele nombrar con una o varias letras mayúsculas, por ejemplo X, aunque la representación correcta sería $\langle X|$.
En la tabla 3.III se representan los giros sobre los ejes principales(puertas de **Pauli**[14]), así como la puerta **Hadamard** o también llamada puerta H. Para cada uno de ellos se muestra los estados estándar y a qué estados se transforman tras aplicar cada puerta.
En la sección 3.6 se describirá cómo modelar estas transformaciones de manera matemática.

Las tres puertas XYZ representan giros de ángulo π con respecto al eje que nombran. La puerta *Hadamard* o puerta H es una transformación no trivial que surge de la concatenación de un giro Y de 90° y un giro Z de 180°. Esta puerta tiene una gran trascendencia porque transforma el estado inicial $|0\rangle$ en un estado *superpuesto* $|+\rangle$, es decir, un estado con la misma probabilidad de ser 0 y 1. Este giro se puede ver representado en el anexo D.1.

nombre	X	Y	Z	H
matriz	$\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$	$\begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}$	$\begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$	$\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \frac{1}{\sqrt{2}}$
giro	180° sobre eje X	180° sobre eje Y	180° sobre eje Z	$\sqrt{Y} \cdot Z$
$\|0\rangle \longrightarrow$	$\|1\rangle$	$\|1\rangle$	$\|0\rangle$	$\|+\rangle$
$\|1\rangle \longrightarrow$	$\|0\rangle$	$\|0\rangle$	$\|1\rangle$	$\|-\rangle$
$\|+\rangle \longrightarrow$	$\|+\rangle$	$\|-\rangle$	$\|-\rangle$	$\|0\rangle$
$\|-\rangle \longrightarrow$	$\|-\rangle$	$\|+\rangle$	$\|+\rangle$	$\|1\rangle$
$\|i\rangle \longrightarrow$	$\|j\rangle$	$\|i\rangle$	$\|j\rangle$	$\|j\rangle$
$\|j\rangle \longrightarrow$	$\|i\rangle$	$\|j\rangle$	$\|i\rangle$	$\|i\rangle$

Tabla 3.III: Puertas cuánticas básicas. Cada fila $|x\rangle$ representa el estado origen y el estado al que transforma cada puerta.

Como podemos apreciar en la tabla 3.III las puertas básicas transforman los estados estándar entre ellos. Podemos ver también que estas operaciones son a su vez sus propias inversas, ya que $\forall\phi$ se cumple que $\langle XX|\phi\rangle = |\phi\rangle$, e igualmente se puede comprobar para cualquiera de las puertas estándar.
Una duda razonable que puede surgir sería el hecho de que el giro X no se corresponde con la matriz solución de sustituir el ángulo. Esto no es una errata, pero tampoco tiene una explicación trivial. Se explicará más adelante en esta misma sección.

Puertas de Fase Existen también las puertas de fase, incluidas en las **puertas estándar**. Son aquellas que se corresponden con giros en el eje Z. Estas puertas, no modifican el *valor* final del qubit, pero afectan a futuras operaciones sobre el mismo.
Todas las operaciones tienen una operación inversa, representada mediante el símbolo † la cual cumple $\forall U \Rightarrow UU^\dagger = U^\dagger U = I$. Si una misma puerta es su inversa (como se ha explicado antes) no se usa la notación †. En la tabla 3.IV se muestran las puertas de fase junto a sus inversas y algunos ejemplos de estados estándar y los estados finales en los que se ven transformados.

nombre	S	$S^\dagger$	T	$T^\dagger$
matriz	$\begin{bmatrix} 1 & 0 \\ 0 & i \end{bmatrix}$	$\begin{bmatrix} 1 & 0 \\ 0 & -i \end{bmatrix}$	$\begin{bmatrix} 1 & 0 \\ 0 & \frac{1+i}{\sqrt{2}} \end{bmatrix}$	$\begin{bmatrix} 1 & 0 \\ 0 & \frac{1-i}{\sqrt{2}} \end{bmatrix}$
ángulo sobre Z	$\frac{\pi}{2}$	$\frac{-\pi}{2}$	$\frac{\pi}{4}$	$\frac{-\pi}{4}$
$\|0\rangle \longrightarrow$	$\|0\rangle$	$\|0\rangle$	$\|0\rangle$	$\|0\rangle$
$\|+\rangle \longrightarrow$	$\|i\rangle$	$\|j\rangle$	$(\|0\rangle + \|1\rangle\, e^{i\pi/4})\frac{1}{\sqrt{2}}$	$(\|0\rangle + \|1\rangle\, e^{-i\pi/4})\frac{1}{\sqrt{2}}$
$\|i\rangle \longrightarrow$	$\|-\rangle$	$\|+\rangle$	$(\|0\rangle + \|1\rangle\, e^{i3\pi/4})\frac{1}{\sqrt{2}}$	$(\|0\rangle + \|1\rangle\, e^{i\pi/4})\frac{1}{\sqrt{2}}$

Tabla 3.IV: Puertas cuánticas de cambio de fase

En la tabla 3.IV se puede ver que dos puertas T generan una puerta S, y que dos puertas S generan una puerta Z.

Teorema de Universalidad El teorema descrito en [7] implica que solo mediante las puertas H y T se genera un sistema *universal*. Esto es, mediante la concatenación de operaciones de estos dos tipos se puede llegar a cualquier punto de la Esfera. Esto tiene importancia en los casos donde estas sean las operaciones básicas del ordenador. En nuestro caso vamos a usar los giros parametrizados con los ángulos deseados, por lo que la universalidad la da el tener un set infinito de transformaciones.

Puerta XNOT La puerta $XNOT$ es la única puerta física que afecta a dos qubits y mediante la cual se puede generar un sistema de mayores dimensiones.
Esta puerta se denomina *X-NOT* o $XNOT$ y funciona como una $CNOT$ lógica aplicada a uno de los dos qubits. El qubit que modifica su valor se denomina **target** y el qubit del cual depende esa modificación se denomina **source**.

$$\begin{aligned}|00\rangle &\rightarrow |00\rangle\\ |01\rangle &\rightarrow |01\rangle\\ |10\rangle &\rightarrow |11\rangle\\ |11\rangle &\rightarrow |10\rangle\end{aligned}$$

Tabla 3.V: *Tabla de verdad* para los estados estándar siendo el primer qubit *source* y el segundo *target*

Como vemos, esta puerta funciona como una puerta X sobre el segundo qubit, siempre que el primer qubit tenga un valor de "1". Este efecto se consigue mediante el fenómeno del *entrelazamiento cuántico*. El primer qubit no tiene un valor concreto, por lo que la transformación sobre el segundo no será concreta tampoco hasta que la onda no colapse.
Esta puerta se representa mediante una matriz cuadrada de 4×4 mostrada en la ecuación 3.1.

$$\begin{array}{c} |00\rangle \\ |01\rangle \\ |10\rangle \\ |11\rangle \end{array} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \tag{3.1}$$

Las puertas estándar y giros parametrizados y la puerta $XNOT$ crean el conjunto de las **Puertas básicas** o puertas *implementadas* dentro de un ordenador cuántico. Este conjunto genera un *sistema universal*, es decir, es suficiente para generar cualquier *programa* o *circuito*.

Puerta Identidad Sobre la Esfera de Bloch existen ciertas operaciones que se consideran *identidad*, es decir: $\forall U \rightarrow UI = IU = U \Leftrightarrow I = identidad$. Aunque no existe la puerta *identidad*, existe un conjunto de matrices que equivalen a ella.
Para empezar, vemos que cualquiera de los giros, si los parametrizamos con un ángulo de valor 0, forman la *matriz identidad* $\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$ que devuelve a cualquier qubit afectado a él mismo.
Pero no es la única operación identidad que existe. Debido a la notación matemática escogida para representar matricialmente los estados y las operaciones, cualquier operación que de como resultado una matriz como esta: $\begin{bmatrix} \epsilon(\alpha) & 0 \\ 0 & \epsilon(\alpha) \end{bmatrix} \forall \alpha$ (se usa la notación $\epsilon(\alpha) = e^{i\alpha}$) genera una operación identidad. Esto complica las operaciones a la hora de calcular los vectores que representan el mismo estado. Por ejemplo el vector $\begin{pmatrix} -i \\ 1 \end{pmatrix}$ y el vector $\begin{pmatrix} 1 \\ i \end{pmatrix}$ representa el mismo estado, y existen infinitas representaciones del mismo estado. Debido a esto, los estados se representan de forma que la primera dimensión no tenga valores complejos, y de esta forma solo existe una representación posible de cada estado. Es importante no confundir esto con el hecho de que las transformaciones de fase no modifican el *valor* del qubit.

Puertas Compuestas En la siguiente sección veremos como se calcula matemáticamente la concatenación de estas puertas básicas para dar lugar a *circuitos* o *puertas compuestas*, pero para hacer una pequeña introducción, diremos que las *puertas compuestas* nacen de la unión de una o más puertas concatenadas modificando uno o varios qubits al mismo tiempo, y generando puertas de *más alto nivel* computacional.

3.6. Circuitos y simuladores

Un *programa* cuántico es una concatenación de puertas básicas. Vamos a explicar las operaciones matemáticas que modelan la concatenación de dichas puertas.

Circuitos Se llama circuito a una concatenación de puertas cuánticas básicas o complejas (formadas por puertas básicas) y cada puerta se representa matemáticamente con una matriz. Esta matriz se puede obtener a base de operaciones matemáticas.
Estos circuitos se representan mediante gráficos como el mostrado en la figura ejemplo 3.4. Cada una de las líneas horizontales representa un qubit, y las transformaciones que va sufriendo (se lee de izquierda a derecha). El cuadrado representa una puerta básica, en este caso una puerta de *Hadamard*. El círculo con una cruz representa que ese qubit será el *target* de una puerta $XNOT$ y el punto negro unido a este círculo representa que ese qubit será *source* para dicha puerta.

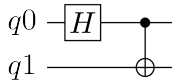


Figura 3.4: Ejemplo de circuito cuántico. Representación del *Primer Estado de Bell*

Los circuitos se pueden dividir en **pasos**, donde cada paso equivale, o bien a una puerta cuántica que afecta a varios qubits o a un conjunto de puertas unitarias sobre diferentes qubits, es decir, que las operaciones son individuales y no se afectan entre ellas.
Cuando un paso está formado por puertas unitarias, se aplica un producto tensorial[17] sobre las matrices para obtener la matriz esperada. En la tabla 3.VI se muestra tres circuitos formados por puertas unitarias que afectan a un qubit u otro y sus matrices representativas, formadas mediante el producto tensorial de la matriz correspondiente al primer qubit con la matriz correspondiente al segundo qubit.

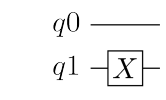
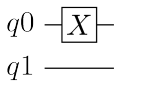
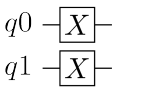
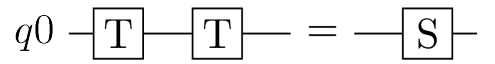
circuito			
matriz qubit solución	$\begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$	$\begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$	$\begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}$
$\|00\rangle \longrightarrow$	$\|01\rangle$	$\|10\rangle$	$\|11\rangle$

Tabla 3.VI: Resultados de la concatenación de matrices en un mismo paso

A la hora de concatenar varias puertas o pasos seguidos, el funcionamiento matemático equivale a multiplicar las matrices soluciones de cada paso. Un ejemplo sencillo se ilustra en la figura 3.5 donde se ve la concatenación de dos puertas T.
Como sabemos, la puerta T equivale a un giro de $\frac{\pi}{4}$ con respecto al eje Z, y la puerta S equivale a un giro de $\frac{\pi}{2}$. Por lo tanto vemos que una concatenación sobre un circuito de un solo qubit de dos puertas T debería dar una matriz igual a S:

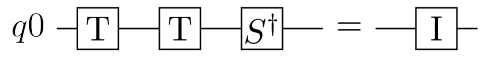
Así vemos como la multiplicación de matrices nos da el resultado esperado. Si seguimos con este ejemplo, vemos que al concatenar las dos puertas T y una puerta $S^\dagger$, la solución debería ser una matriz identidad. Esto se muestra en la figura 3.6.

Es importante tener en cuenta que la multiplicación de las matrices se lleva a cabo de forma



$$\begin{bmatrix} 1 & 0 \\ 0 & \epsilon(\frac{\pi}{4}) \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & \epsilon(\frac{\pi}{4}) \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & \epsilon(\frac{\pi}{2}) \end{bmatrix} \tag{3.2}$$

Figura 3.5: Circuito TT.



$$\begin{bmatrix} 1 & 0 \\ 0 & \epsilon(\frac{-\pi}{2}) \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & \epsilon(\frac{\pi}{4}) \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & \epsilon(\frac{\pi}{4}) \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \tag{3.3}$$

Figura 3.6: Circuito $TTS^\dagger$.

inversa al orden lógico de las puertas. De esta manera, una puerta X concatenada detrás con una puerta S daría una matriz $\langle XS| = \begin{bmatrix} 0 & 1 \\ i & 0 \end{bmatrix}$ y no $\langle SX| = \begin{bmatrix} 0 & i \\ 1 & 0 \end{bmatrix}$.

Simuladores Un **Simulador Cuántico** *simula* el funcionamiento de un ordenador cuántico mediante la transformación de puertas a matrices, y realizando las operaciones necesarias para generar la matriz final. De esta forma, dado un estado inicial y un circuito, se puede conocer la matriz solución de dicho circuito, y multiplicando la matriz por el vector del estado, se obtiene el vector del **estado resultante**.

Los simuladores trabajan siempre con un estado inicial $|0..,00\rangle$, que es el estado inicial por defecto, o estado de *menor energía*. Esto es debido a que para partir de un estado que no fuese el básico, debería aplicarse al estado básico un circuito previo, lo que equivaldría a concatenar ambos circuitos (recuérdese que en la computación cuántica no es posible copiar información ni guardar un estado de una ejecución a otra).

4

Algoritmo de Diseño Matricial de Circuitos Cuánticos

Este capítulo constituye el fuerte de la investigación llevada a cabo durante el proceso de este *Trabajo de Fin de Grado*. En este capítulo se recoge toda la información relativa al Algoritmo de Diseño de Circuitos Cuánticos (a aprtir de ahora QCMD - Quantum Circuit Matricial Design algorithm).

4.1. Introducción al algoritmo

El algoritmo QCMD es un nuevo algoritmo diseñado para el ámbito de la Computación Cuántica, pero pensado para ejecutarse en un ordenador o un autómata Lógico.

4.1.1. Idea principal

El propósito de este algoritmo es la posibilidad de abstracción a la hora de trabajar con *programas* dentro de la computación cuántica. Como ya sabemos, el ordenador cuántico funciona mediante un circuito que concatena transformaciones sobre qubits para llegar a un resultado deseado. Este resultado se puede representar matemáticamente mediante una matriz.
La idea principal de este algoritmo es llevar a cabo el proceso contrario, es decir, dada una **matriz solución** a la que se quiere llegar, construir **automáticamente** el **circuito cuántico** necesario para llegar a ella. Análogamente a un ordenador electrónico, sería crear un *lenguaje* auxiliar mediante el cual construir un camino *hardware* para tener un procesador que, con una entrada específica (estado inicial básico) consiga una salida esperada, como sería el caso del lenguaje **HDL** (Hardware Description Language).

4.1.2. Motivación

Abstracción Uno de los principales problemas a la hora de enfrentarse a la creación de un programa cuántico, es el hecho de que se trabaja a muy bajo nivel, a nivel de puertas básicas sobre los qubits. El algoritmo propuesto permitiría poder usar un ordenador cuántico sin tener que conocer la física y el *hardware* interno, y de una forma más cómoda y rápida.

Uso en algoritmos Este algoritmo nunca se usará a la hora de crear programas enteros, ya que, si se conoce de antemano la solución no es necesaria la implementación del circuito. Pero existen varios algoritmos cuánticos basados en la concatenación de matrices conocidas, de forma que se conoce el resultado matemático esperado del mismo, pero no se conoce el circuito interior que generaría esa salida esperada.
Este es el caso del **Algoritmo de Búsqueda de Grover** descrito en el anexo A. Este es uno de los algoritmos más importantes en la actualidad en el mundo de la computación cuántica ya que permite una búsqueda en un conjunto no ordenado con una complejidad no alcanzable por un ordenador lógico.
También se incluiría entre los usos del algoritmo QCMD la generación de un circuito previo a un algoritmo con circuito conocido, mediante el cual podríamos parametrizar el estado inicial de entrada a dicho algoritmo simplemente con generar la matriz que transformaría el *estado inicial básico* en el estado inicial deseado.
De esta forma, el algoritmo reduciría el trabajo de un *programador cuántico* al estudio teórico y matemático del programa que quiere realizar, y no a la construcción del circuito de bajo nivel que dará los resultados esperados.

4.1.3. Primera aproximación

La primera aproximación sobre cómo afrontar el problema fue mediante una búsqueda exhaustiva en árbol a través de las puertas básicas conocidas (X, H, T, etc.) hasta llegar a alcanzar la matriz deseada.
El límite de puertas necesarias para alcanzar dicho objetivo se describe en el artículo de la bibliografía [18]. Se sabe que existe una solución (H y T generan un conjunto universal) pero el número de puertas necesario es muy alto.
Como es fácil de imaginar, esta aproximación tenía una complejidad computacional extremadamente alta, y solo resultaba efectiva para circuitos de uno o dos qubits, y matrices con giros comprendidos entre las puertas básicas. Los problemas que se encontraron, entre otros, fueron:

- Multiplicación a cada paso de las matrices solución $O(N^3), N = 2^n$ siendo n el número de qubits y N el tamaño de la matriz.
- Crecimiento exponencial a la hora de buscar en anchura en árbol.
- La multiplicación de matrices es difícilmente rastreable, y da lugar a resultados que a la vista podrían considerarse aleatorios, por lo que impide la poda trivial de ramas en la búsqueda.

Por esto, esta primera aproximación se rechazó rápidamente.

Finalmente, la aproximación elegida para el diseño del algoritmo es la división de la matriz principal en submatrices.

4.2. Descripción del algoritmo

En esta sección se describe el algoritmo, es decir, el fundamento teórico en el que se basa su funcionamiento, las razones por las que se ha decidido tomar esta vía, las demostraciones matemáticas necesarias para comprobar que el algoritmo ciertamente genera lo esperado y el cálculo de tiempo y espacio necesario para llevar a cabo la ejecución del mismo.

La idea principal de este algoritmo es generar un circuito para una matriz dada, pero los pasos intermedios que se han diseñado (puertas compuestas) también suponen en sí mismas una abstracción del problema de la programación a bajo nivel, y su uso también es generalizable fuera del ámbito de este algoritmo.

Nótese que la idea principal del algoritmo es tratar de encontrar un circuito que genere una matriz, y esta matriz es de tamaño $N \times N, N = 2^n$ donde n es el número de qubits. Por tanto, sea cual sea la implementación, será necesario un espacio mínimo de $O(2^{2n})$. Esta limitación existe para cualquier algoritmo que intente generar un circuito mediante la construcción de una matriz.

Este diseño tiene el objetivo de encontrar de una forma **sencilla** y con una **cota superior** en número de puertas, el circuito relacionado con una matriz. No intenta ser eficiente en número de pasos del circuito, sino que basa más su funcionamiento en encontrar una solución entendible y de fácil trazabilidad en un tiempo acotado superiormente.

4.2.1. Fundamento teórico

El teorema sobre el que se basa este algoritmo, demostrado en la bibliografía[7] indica que cualquier matriz **unitaria** es divisible en un número **finito** de matrices **unitarias** que *afecten* únicamente a **dos filas y columnas**, o lo que sería lo mismo, una matriz 2×2 *encajada* en una matriz diagonal de mayor dimensión.
Existe también un teorema en la documentación de la *Quantum Experience IBM* [19] en el que se recoge la posibilidad de obtener una puerta cuántica condicionada para cualquier transformación necesaria. La única puerta cuántica condicionada básica es la $XNOT$, la cual es una transformación X sobre un qubit. Esto se generalizará a cualquier transformación básica.

Se ha creado un sistema completo cimentado en estos teoremas donde se trabaja con puertas cuánticas condicionadas con más de un qubit que crean giros muy específicos, los cuales tienen una matriz muy acotada y controlable que se puede usar más adelante para dividir la matriz principal.

4.3. Diseño

4.3.1. Resumen de diseño

Análogamente a un ordenador clásico, las puertas básicas de un ordenador cuántico (aquellas implementadas a nivel *hardware*) se pueden concatenar para generar circuitos más complejos y con distintas funcionalidades.
Este algoritmo se ha diseñado para generar distintos circuitos con complejidades ascendentes de forma que mediante puertas básicas (lo que serían puertas lógicas NOT o AND entre otras) poder llegar a ciertas funcionalidades específicas (análogo a generar *multiplexores* o circuitos combinacionales de mayor complejidad), hasta el punto de generar un algoritmo cuyo parámetro de entrada sea una matriz, y que sea capaz de generar la concatenación interna de puertas para crear un circuito cuántico que genere dicha matriz.

Esto quiere decir que se han ido creando puertas o circuitos cada vez de más alto nivel, hasta llegar a un circuito capaz de generar la matriz deseada. Pero las puertas intermedias también suponen una solución en sí mismas ante ciertos problemas actuales en la computación cuántica, y tienen valor aún aisladas del resto del algoritmo.

4.3.2. Puertas básicas

Las puertas básicas, como se explicó en el capítulo 3.5, son puertas que se encuentran implementadas a nivel *hardware* en un ordenador cuántico (o se puede suponer que están implementadas).
La **profundidad** (número de transformaciones a nivel hardware necesarias para alcanzar dicha puerta) de dichas puertas es 1.

Las puertas que se van a usar para el diseño de este algoritmo son las puertas *estándar* X y H, las puertas de giro parametrizadas sobre los ejes $X(\alpha)$, $Y(\beta)$ y $Z(\gamma)$, y la puerta $XNOT$.

4.3.3. Puertas compuestas

A partir de aquí se describen los circuitos diseñados para el algoritmo. Aunque la concatenación de puertas básicas genere un circuito cuántico, estos circuitos se representan mediante una matriz y suponen una transformación sobre los qubits de igual manera que hace una puerta cuántica, por lo que se nombrarán también como puertas *compuestas*, que son el resultado de concatenar otras puertas inferiores.

Varias de estas puertas tenían ya formas de implementarse, como la puerta *Toffoli*, y algunas otras existen solo de forma teórica (se usa matemáticamente su matriz pero sin la necesidad de conocer la implementación a nivel físico), como las puertas de giros condicionados, mientras que otras son de diseño propio. En este capítulo se incluyen todas las puertas y los diseños elegidos para generarlas, así como el cálculo de la complejidad/profundidad de las mismas y la matriz solución de cada una de ellas.

4.3.3.1. Puerta de giro condicionado

Esta es la primera puerta **compuesta** con la que vamos a trabajar, que es, a su vez, la base de toda la estructura superior.
Esta puerta consta de un qubit **source** y un qubit **target** y de un parámetro **ángulo** que indica el ángulo de giro sobre el eje destino. Su funcionamiento se basa en realizar el giro sobre uno de los tres ejes principales sobre el qubit *target* siempre que el qubit *source* se encuentre en un valor positivo (cuando su onda colapse a un valor positivo).

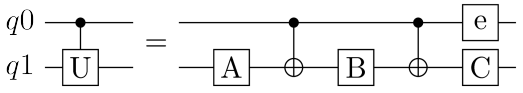
La puerta básica $XNOT$ se consideraría como un giro condicionado de 180° sobre el eje X. Existen algunos circuitos simples para los giros más utilizados, como el giro de 180° sobre los ejes Y (figura 4.1a) y Z (figura 4.1b), que solo se componen de puertas estándar.

(a) Puerta Y condicionada. (b) Puerta Z condicionada.

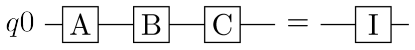
Figura 4.1: Puertas condicionadas estándar.

Según el teorema descrito en el libro de la bibliografía[7], cualquier puerta condicionada puede generarse mediante la concatenación de puertas estándar y el uso únicamente de dos

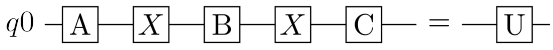
puertas $XNOT$, como se muestra en la figura 4.2a. Esto quiere decir que para cualquier giro U (no tiene que ser necesariamente en uno de los ejes principales) existen tres concatenaciones de puertas A, B, C y una puerta de fase e tales que $eCXBXA = U$ (figura 4.2c) y a su vez $CBA = I$ (figura 4.2b)



(a) Teorema de puertas condicionadas.



(b) Generación de la puerta identidad.

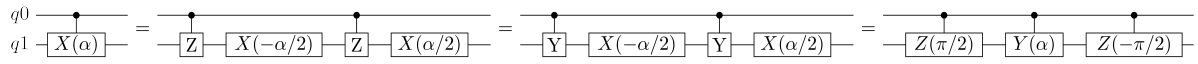


(c) Generación de la puerta U.

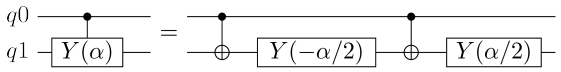
Figura 4.2: Teorema de construcción de una puerta condicionada.

En lo que a nuestro algoritmo respecta, solo vamos a usar giros sobre los ejes principales, debido a su sencillez y a que son suficientes para generar el resto de las puertas. Recordemos que dos giros de distinto eje se asimilan a un giro en otro eje. Por lo tanto, con giros en dos de los ejes principales distintos podemos crear cualquier giro.
Para el caso del eje Z sabemos que la creación de este giro es bastante sencilla debido a que, concatenando un giro de un ángulo α junto con otro $-\alpha$ conseguimos una transformación identidad. Pero si introducimos un giro π sobre el eje X (también podría ser sobre el Y) tras el primer giro sobre Z, lo que conseguimos es que los dos giros sobre Z se concatenen, generando un giro de ángulo 2α (demostración en C.1). Un giro en el eje Y tiene las mismas características (demostración en C.2). Se puede ver una representación gráfica de un giro condicionado $Y(\alpha)$ usando puertas condicionadas Z en las figuras D.2 (sin activar la puerta Z) y D.3 (con puerta Z activa).

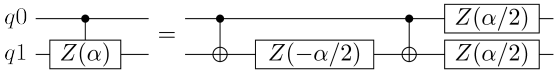
En el caso del eje X existe la diferencia de que los giros condicionados entre los dos giros X debe ser en cualquiera de los otros dos ejes. Esto nos permite concluir con uno de los circuitos más básicos que vamos a usar a partir de ahora, que se divide entre los tres giros (figura 4.3).



(a) Giro X condicionado.



(b) Giro Y condicionado.



(c) Giro Z condicionado.

Figura 4.3: Construcción de los circuitos para generar un giro concatenado.

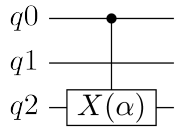
Podemos observar que hay tres formas distintas de crear un giro X. Para las dos primeras se usa una puerta condicionada de ángulo π entre los ejes principales, y después se concatenan los giros sobre el eje X. La última forma transforma la matriz del giro Y en una matriz de giro X, como se demustra en la ecuación C.4.
La puerta en el qubit $q0$ del giro del eje Z tiene su explicación, pero no es una demostración trivial. Es la puerta de fase e necesaria en el teorema descrito anteriormente.

Matriz La matriz de esta puerta está compuesta por una diagonal de elementos 1 menos en los casos donde la puerta esté *activada*, donde se encuentra la matriz del giro. De esta forma, en circuitos de solo dos qubits obtenemos una matriz como la representada en 4.1 para el caso en que el qubit *target* sea el último qubit, o una matriz como 4.2 para el caso donde el primer qubit sea *target*. Las variables a, b, c, d representan los valores relativos al giro en 1 qubit.

$$\begin{matrix}|00\rangle\\|01\rangle\\|10\rangle\\|11\rangle\end{matrix}\begin{bmatrix}1&0&0&0\\0&1&0&0\\0&0&a&b\\0&0&c&d\end{bmatrix} \quad (4.1) \qquad \begin{matrix}|00\rangle\\|01\rangle\\|10\rangle\\|11\rangle\end{matrix}\begin{bmatrix}1&0&0&0\\0&a&0&b\\0&0&1&0\\0&c&0&d\end{bmatrix} \quad (4.2)$$

Figura 4.4: Matrices que representan un giro condicionado.

Al aumentar el número de qubits, el producto tensorial se lleva a cabo de igual manera que con las puertas básicas. Por ejemplo, en la figura 4.5 se muestra como sería un giro sobre el eje X de ángulo α en un circuito de tres qubits, desde el primer al último qubit.



(a) Circuito representativo.

$$\begin{matrix}|000\rangle\\|001\rangle\\|010\rangle\\|011\rangle\\|100\rangle\\|101\rangle\\|110\rangle\\|111\rangle\end{matrix}\begin{bmatrix}1&0&0&0&0&0&0&0\\0&1&0&0&0&0&0&0\\0&0&1&0&0&0&0&0\\0&0&0&1&0&0&0&0\\0&0&0&0&cos(\frac{\alpha}{2})&sin(\frac{\alpha}{2})i&0&0\\0&0&0&0&sin(\frac{\alpha}{2})i&cos(\frac{\alpha}{2})&0&0\\0&0&0&0&0&0&cos(\frac{\alpha}{2})&sin(\frac{\alpha}{2})i\\0&0&0&0&0&0&sin(\frac{\alpha}{2})i&cos(\frac{\alpha}{2})\end{bmatrix}$$

(b) Matriz solución del circuito.

Figura 4.5: Giro condicionado X en un circuito de 3 qubits.

Inversión Como ocurre con los giros básicos, los giros condicionados generan su inversa con el mismo giro condicionado pero con ángulo inverso. Se puede ver la demostración en las ecuaciones C.3 y C.5.

Complejidad (P_g) Esta ya es una puerta compuesta, lo que quiere decir que necesita de una concatenación de instrucciones para llevarse a cabo. Como se puede ver fácilmente, el uso del giro condicionado X no es eficiente, ya que para generar las puertas de giro Y o Z condicionadas que harían de auxiliares, se necesitan más puertas internas, por lo que el algoritmo no usará puertas X condicionadas, y cuando sean necesarias se crearán mediante concatenación de las otras 2.
Para tomar una medida concreta en cuanto a esta puerta, se tomará como cota superior el número de puertas que usa el giro $Z = 5$, ya que es mayor que el giro $Y = 4$. Por lo tanto podemos concluir que la complejidad de esta puerta. $P_g = \mathbf{5} = \mathbf{O(1)}$ (P_x indica la profundidad de la puerta x. En este caso g representa a la puerta de giro condicionado).

4.3.3.2. Puerta de giro condicionado múltiple

Esta puerta corresponde a un giro igual al de una puerta de giro condicionada (ver 4.3.3.1) pero donde existe más de un qubit *source* (Figura 4.6). Esto es, para que se lleve a cabo el giro, todos los qubits *source* de la puerta deben tener valor 1.

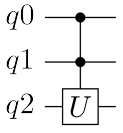


Figura 4.6: Puerta condicionada múltiple.

Como vemos a continuación, existen dos diseños diferentes para ella, cada uno con sus ventajas e inconvenientes.

Diseño principal En este diseño, la puerta con 2 qubits *sources* viene representada por la figura 4.7[7] mientras que las demás puertas se generan de forma recursiva (ver figuras 4.8 y 4.9), de manera que se necesita una puerta condicionada con un *source* menos en cada paso, acabando el último paso en una puerta condicionada simple. La lógica tras este diseño y los diseños de niveles superiores (figura 4.8 y 4.9) no es trivial y se recoge en el anexo C.2. El diseño de la puerta con 2 *sources* se ha recogido de la bibliografía, mientras que el diseño recursivo de las puertas de más nivel es propio.

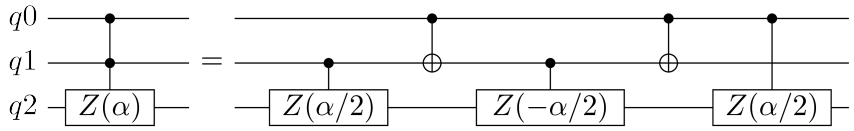


Figura 4.7: Giro Z condicionado por 2 qubits.

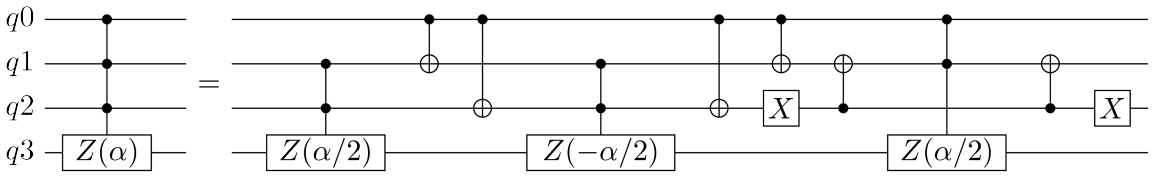


Figura 4.8: Giro Z condicionado por 3 qubits.

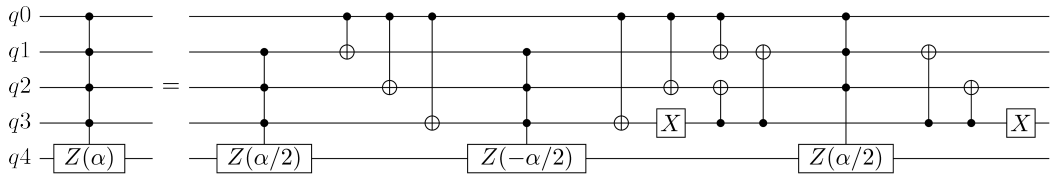
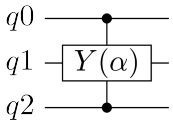


Figura 4.9: Giro Z condicionado por 4 qubits.

Matriz Esta matriz se construye de igual manera que la matriz de giro condicionado simple, pero teniendo en cuenta que los valores que representan el giro solo deben aparecer en los puntos correctos donde los *sources* estén activos. En la figura 4.10 se muestra un ejemplo.



$$
\begin{array}{c}
|000\rangle \\ |001\rangle \\ |010\rangle \\ |011\rangle \\ |100\rangle \\ |101\rangle \\ |110\rangle \\ |111\rangle
\end{array}
\begin{bmatrix}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & cos(\frac{\alpha}{2}) & 0 & -sin(\frac{\alpha}{2}) \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & sin(\frac{\alpha}{2}) & 0 & cos(\frac{\alpha}{2})
\end{bmatrix}
$$

Figura 4.10: Giro condicionado Y en un circuito de 3 qubits.

Inversión Al igual que los giros condicionados simples, los giros condicionados múltiples generan su inversa mediante la inversión del ángulo del giro.

Complejidad ($P_{m(d1)}$) La profundidad de esta puerta viene dada por la suma recursiva de la complejidad de las puertas más pequeñas (con menos *sources*).
Para calcular la complejidad usaremos el parámetro s para referirnos al número de qubits *source* de la puerta. El número de puertas básicas X es 2 para cualquier número de qubits. El número de puertas $XNOT$ es $2(s-1)+2(s-2)$. El número de puertas necesario para las puertas condicionadas múltiples inferiores es $3(P_m(s-1))$ donde $P_m(x)$ es la profundidad de una puerta condicionada múltiple para x qubits *source*, y donde $P_m(2) = 3P_g + 2 = 17$. Por lo tanto, el cálculo de profundidad de esta puerta será $P_m(s) = 2+2(s-1)+2(s-2)+P_m(s-1)$ de donde obtenemos que $P_m(s) = 22 \cdot 3^{s-2} - 2s - 1 = \mathbf{O(3^s)}$. La demostración se puede ver en el anexo C.2.

Diseño secundario En este segundo diseño se usan las puertas Toffoli (se verá con más detalle en la siguiente sección 4.3.3.3) que funcionan como una puerta lógica *AND*. Es decir, se puede generar un único qubit auxiliar que se encuentre a 1 si todos los demás qubits implicados tienen valor 1, o a 0 en cualquier otro caso (figura 4.11), y de esta forma reducir el diseño a una única puerta condicionada simple[7]. La demostración de este diseño se encuentra en la ecuación C.12.
El problema principal de esta puerta es que necesita qubits extra para poder utilizarse, y estos aumentan en 2^x el tamaño actual de la matriz. También supondrían un problema dependiendo del número de qubits disponibles en el ordenador cuántico. A cambio se ahorra muchas puertas al evitar el carácter recursivo de la puerta (la puerta Toffoli sigue siendo una puerta condicionada múltiple de dos *sources* que requiere construirse implícitamente).

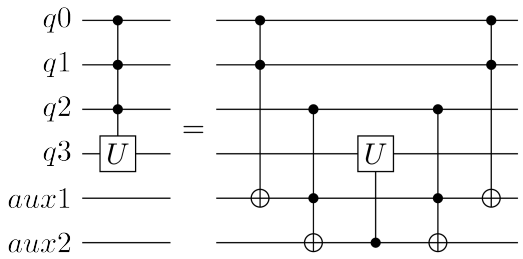


Figura 4.11: Puerta condicionada múltiple con qubits auxiliares.

La matriz y la forma de inversión de la puerta es independiente del diseño. Por lo tanto, en este segundo diseño solo se modifica la complejidad.

Complejidad ($P_{m(d2)}$) Vemos que la profundidad para s qubits *sources* será de $P_{(d2)2}(s) = 2(s-1)(P_t(2)) + P_g$ donde $P_t(2)$ es la complejidad de una puerta Toffoli de 2 *sources* y $P_g = 5$ es la profundidad de la puerta de giro condicionada simple.

Podemos comprobar que, con este segundo diseño, la profundidad es mucho menor que con el primer diseño, ya que en este caso se tiene una profundidad de $\mathbf{O(s)}$ mientras que con el primer diseño teníamos una profundidad de $\mathbf{O(3^s)}$. Se ha elegido el primer diseño como principal debido a que una de las grandes limitaciones de un ordenador cuántico actual es el número de qubits, por lo que usar puertas que necesiten qubits auxiliares no siempre es una posibilidad. En el resto del documento el uso del primer diseño se denotará mediante $P_{(d1)}$ y el segundo diseño mediante $P_{(d2)}$.
El segundo diseño tiene otra limitación, que se debe a que construir las puertas Toffoli necesarias conlleva una gran cantidad de puertas básicas. Como se demuestra en el anexo C.9 el uso del segundo diseño será útil con circuitos mayores de 6 qubits.

4.3.3.3. Puerta Toffoli

Esta es una puerta condicionada múltiple particular. Se toma como puerta independiente debido a su gran uso, pero en realidad se podría sustituir por una puerta condicionada múltiple de giro X y ángulo π. Esto hace que se comporte como una puerta $XNOT$ (figura 4.12a) de varios *sources*, o lo que es lo mismo, una puerta AND sobre su qubit *target*.
La construcción de esta puerta se puede realizar de la misma forma que las puertas condicionadas múltiples, aunque esto conlleva usar la puerta condicionada X, que es menos eficiente. Por eso el diseño elegido es el que se muestra en la figura 4.12b. La demostración de que esta concatenación genera este giro se encuentra en la ecuación C.13.
También se puede generar un diseño de esta puerta igual al **diseño secundario** de la puerta condicionada múltiple, teniendo en cuenta que la puerta Toffoli de 2 *sources* debe implementarse explícitamente, ya que es la base de dicho diseño.

(a) Puerta Toffoli.

(b) Diseño de la puerta Toffoli.

Figura 4.12: Diseño de puerta Toffoli.

Matriz La matriz es igual que las matrices de las puertas condicionadas múltiples. El ejemplo de la matriz para la puerta del ejemplo 4.12a se representa en la matriz 4.3.

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix} \tag{4.3}$$

A pesar de la relativa simplicidad de la matriz solución, el diseño interno para llegar a dicho cálculo es muy complejo.

Inversión Cualquier puerta Toffoli (para cualquier número de *source*) es su propia inversa, debido a que se trata de una puerta condicionada múltiple de ángulo π.

Complejidad (P_t) La complejidad de esta puerta depende del número de qubits *source* s. Podemos ver en la figura 4.12b que según este diseño, $P_t(s) = 2P_m(s) = \mathbf{O(3^s)}_{(d1)} \vee \mathbf{O(s)}_{(d2)}$ (en función de la opción de diseño elegida).

4.3.3.4. Puerta de giro especial

A partir de aquí las puertas son de diseño propio y no existentes en la actualidad.

Esta puerta representa el paso más importante a la hora de construir una matriz, que se parametriza mediante dos parámetros complejos, que denominaremos a y b. El uso general de

esta puerta es generar un giro condicionado múltiple, donde todos los qubits salvo el último sean *source* y el último qubit es el *target*.

Se quiere que este giro represente la matriz $\begin{pmatrix} a^* & b^* \\ b & -a \end{pmatrix} \frac{1}{\sqrt{|a|^2+|b|^2}}$. Este giro se puede dividir a su vez en una matriz parametrizada con tres valores, que es $\begin{pmatrix} cos(\theta)\epsilon(-x) & sin(\theta)\epsilon(-y) \\ sin(\theta)\epsilon(y) & -cos(\theta)\epsilon(x) \end{pmatrix}$ donde $x = fase(a), y = fase(b), \theta = sin^{-1}(\frac{|b|}{\sqrt{|a|^2+|b|^2}}) = cos^{-1}(\frac{|a|}{\sqrt{|a|^2+|b|^2}})$. De esta manera sabemos qué puertas debemos usar para llegar a crear este giro. El circuito para llegar a esta puerta se muestra en la figura 4.13. Este circuito es generalizable a cualquier número de qubits *source* (incluído 0 para circuitos con un solo qubit). La demostración de este diseño se encuentra en el anexo C.4.

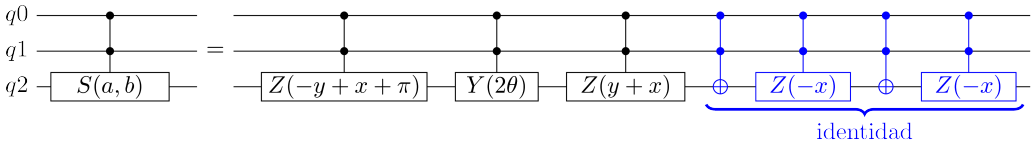


Figura 4.13: Diseño para la puerta de giro especial.

La parte del circuito que aparece en azul en la figura 4.13 son unas puertas que generan una transformación identidad (ver sección anterior 3.5), pero es necesario incluirlas en el diseño de la puerta. Se usan para forzar a la matriz a coincidir con la matriz esperada, ya que existen infinitas matrices que representan una misma transformación.

Matriz La matriz final de este giro con los parámetros a y b para dos qubits como la mostrada en la ecuación 4.4.

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \frac{a^*}{\sqrt{|a|^2+|b|^2}} & \frac{b^*}{\sqrt{|a|^2+|b|^2}} \\ 0 & 0 & \frac{b}{\sqrt{|a|^2+|b|^2}} & \frac{-a}{\sqrt{|a|^2+|b|^2}} \end{bmatrix} \tag{4.4}$$

Para cualquier número de qubits, la matriz será diagonal exceptuando las dos últimas filas y columnas, donde se observará el mismo caso que el mostrado anteriormente.

Inversión Para invertir esta puerta basta con sustituir el parámetro a por a^*, o lo que sería igual, sustituir el parámetro x por $-x$. De esta forma, conseguiremos generar una matriz 4.5 o la matriz equivalente 4.6.

$$\begin{bmatrix} \frac{a}{\sqrt{|a|^2+|b|^2}} & \frac{b^*}{\sqrt{|a|^2+|b|^2}} \\ \frac{b}{\sqrt{|a|^2+|b|^2}} & \frac{-a^*}{\sqrt{|a|^2+|b|^2}} \end{bmatrix} \tag{4.5}$$

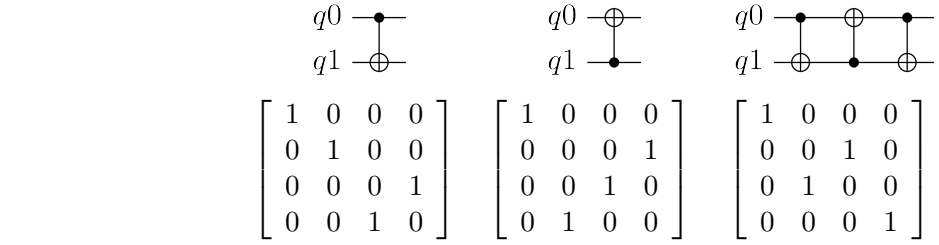
$$\begin{bmatrix} cos(\theta)\epsilon(x) & sin(\theta)\epsilon(-y) \\ sin(\theta)\epsilon(y) & -cos(\theta)\epsilon(-x) \end{bmatrix} \tag{4.6}$$

De esta manera es fácil demostrar que $S(a,b)S(a^*,b) = I \Rightarrow S(a,b)^\dagger = S(a^*,b)$. Se demuestra en la ecuación C.15.

Complejidad (P_s) La complejidad de esta puerta depende de si se implementa de forma que se genere el giro identidad. Para hacer nuestro cálculo de complejidad no vamos a añadir esa parte del diseño, por lo que vemos que $P_s(s) = 7P_m(s) = \mathbf{O(3^s)}_{(d1)} \vee \mathbf{O(s)}_{(d2)}$ siendo s el número de qubits *source* (normalmente todos menos el último qubit).

4.3.3.5. Puerta de intercambio

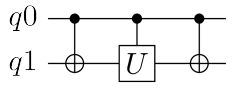
Antes de explicar esta puerta, explicaremos primero una cualidad interesante que vamos a explotar sobre la puerta *XNOT*, o en general, una puerta Toffoli con cualquier número de *sources*. Las matrices de este tipo de puertas son matrices donde los elementos de su diagonal son 1, excepto en dos puntos de la diagonal, donde el elemento diagonal es 0 y el elemento 1 de la misma se ha *movido* hasta la siguiente fila donde la diagonal no es 1. También cuentan con la propiedad de que, concatenando esas puertas entre ellas, generan matrices con la misma peculiaridad. Todo ello se puede ver mejor en la figura 4.15.



$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \quad \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \quad \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

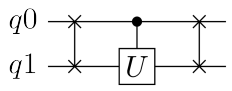
Figura 4.15: Circuitos de puertas *XNOT* y matrices asociadas

Estas puertas cuya matriz cumple esta peculiaridad (matriz identidad salvo 2 filas/columnas) cuentan con la característica que, si se implementan rodeando a cualquier otro circuito, consiguen que su matriz **invierta** esas filas y columnas entre ellas. Un ejemplo práctico se muestra en las figuras 4.16 y 4.17. Para ver la demostración de este efecto véase el anexo C.5.
En la figura 4.17 las puertas $\times$ representan una puerta *SWAP* que intercambia ambos qubits, que se genera con una concatenación de *XNOT* como el tercer circuito de la figura 4.15.



$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & a & b \\ 0 & 0 & c & d \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & d & c \\ 0 & 0 & b & a \end{bmatrix}$$

Figura 4.16: Característica matricial de las puertas *XNOT*.



$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & a & b \\ 0 & 0 & c & d \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & a & 0 & b \\ 0 & 0 & 1 & 0 \\ 0 & c & 0 & d \end{bmatrix}$$

Figura 4.17: Característica matricial de las puertas *SWAP*.

Esta característica nos permite mover las filas y columnas de una matriz desde la penúltima y última columna (es donde los giros condicionados múltiples con el último qubit como *target* tienen la *submatriz* de giro) hasta las filas y columnas que se deseen.
Para ello se usa una concatenación de puertas Toffoli, de modo que todos los qubits sean *target* de alguna de ellas, y se usen todos los demás qubits como *source*, invertidos en algunos casos para hacer que cada una de las puertas Toffoli convierta uno de los qubits desde el estado inicial al estado final esperado.

En la figura 4.18 se muestra como se construiría la puerta intercambio que cambiaría la penúltima fila ($|110\rangle$) por la primera fila ($|000\rangle$). Explicado en detalle, el primer paso es modificar el primer qubit de 0 a 1, por lo que queremos que la puerta Toffoli se *active*. Modificamos los qubits que sean 0 por defecto para que *activen* dicha Toffoli, y tras esto los devolvemos a su estado original. Así tendremos el estado $|100\rangle$. Al llegar al segundo qubit, necesitamos que también se invierta, por lo que *activamos* la Toffoli (hay que tener cuidado, porque para el primer qubit hay que tomar su valor ya cambiado y no el original). Al final, vemos que el último qubit no necesita ser modificado, por lo que esa puerta Toffoli no se incluiría en el circuito.

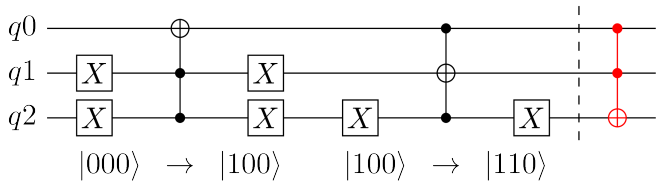


Figura 4.18: Diseño para la puerta de intercambio entre los estados *lógicos* $|000\rangle \rightarrow |110\rangle$.

La concatenación del circuito 4.18 antes y después de otro circuito hará que se inviertan las filas primera y penúltima. Su matriz se puede ver en la matriz 4.7

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.7}$$

Esta puerta tiene un funcionamiento complejo y no existe una forma estándar de representarla, por lo que se utilizará una puerta multiqubit, como se muestra en la figura 4.19. R se usa como abreviatura de *reverse*.

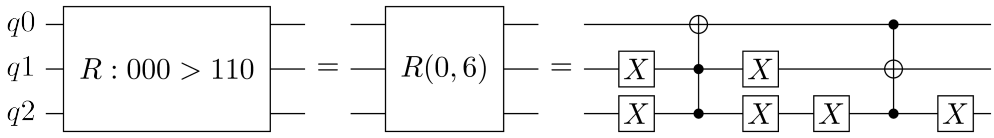


Figura 4.19: Representación de la puerta de intercambio entre los estados *lógicos* $|000\rangle \leftrightarrow |110\rangle$.

Matriz La matriz solución de esta puerta es la ya mostrada anteriormente. Todos los elementos de la diagonal son 1 menos dos de ellos, que son 0, y el valor 1 aparece en la intersección de la fila y columna que se están intercambiando.

Inversión Las puertas de intercambio son su propia puerta inversa.

Complejidad (P_i) La complejidad de esta puerta proviene de la complejidad para generar las puertas toffoli, y depende de la transformación que se quiera llevar a cabo, ya que cuantos más *bits* se transformen, más puertas Toffoli necesitará. Por ello vamos a calcular una cota superior de profundidad para esta puerta, sabiendo que el diseño nunca alcanzará este límite. De esta manera $P_i(n) = n(P_t(n-1) + 2(n-1)) = \mathbf{O(n3^n)}_{(d1)} \vee \mathbf{O(n^2)}_{(d2)}$.

4.3.3.6. Puerta de ajuste matricial simple

La puerta de ajuste matricial simple (a partir de ahora AMS) es la transición entre las puertas compuestas vistas hasta ahora, y la creación de un circuito que genere una matriz unitaria dada. Esta puerta modifica un valor dentro de la matriz para generar una nueva matriz donde ese valor sea 0 (se explicará en la subsección 4.3.4.2).
Esta puerta consta de los parámetros *columna*, *fila*, *a*, *b*. Los parámetros *columna* y *fila* representan la columna y fila de la matriz *input* en la que queremos modificar el elemento, el valor a representa al elemento que se encuentra en esa misma columna en la diagonal de la matriz, y el valor b representa el valor del elemento de la matriz sobre el que estamos trabajando. Por ejemplo, en la matriz $\begin{bmatrix} v_{00} & v_{01} \\ v_{10} & v_{11} \end{bmatrix}$, si quisiésemos trabajar sobre el elemento v_{10}, los parámetros serían: $\begin{bmatrix} columna = 0 & fila = 1 & a = v_{00} & b = v_{10} \end{bmatrix}$.
La puerta que vamos a diseñar a continuación crea un giro especial con parámetros a y b, y traslada la primera columna de este giro (penúltima de la matriz) a la columna *columna* y la segunda columna (la última de la matriz) a la columna *fila*. Puesto que tanto las filas como las columnas son simétricas, si movemos una columna a otra, se mueve también las filas. Esto aparece representado en la figura 4.20. El orden de las puertas de intercambio es importante, ya que podría darse el caso en que *columna* fuese una de las columnas modificadas por la puerta de giro especial, y por lo tanto esa operación alteraría la operación *fila*. Por eso siempre se debe hacer la operación de intercambio de fila externamente para que no interfiera (no importa que en el circuito esta puerta se ejecute antes, la que se encuentre más interna modificará antes a la matriz de la puerta de giro especial).

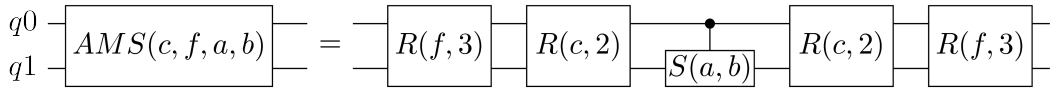


Figura 4.20: Diseño de una puerta matricial simple.

Matriz La matriz de esta puerta se asemeja a la matriz de la puerta de giro especial, salvo que las filas y columnas se han intercambiado para llegar a un punto determinado. Por ejemplo, la matriz correspondiente a la figura 4.20 siendo $c = 0, f = 2$ sería como la matriz representada en la matriz 4.8

$$\begin{bmatrix} \frac{a^*}{\sqrt{|a|^2+|b|^2}} & 0 & \frac{b^*}{\sqrt{|a|^2+|b|^2}} & 0 \\ 0 & 1 & 0 & 0 \\ \frac{b}{\sqrt{|a|^2+|b|^2}} & 0 & \frac{-a}{\sqrt{|a|^2+|b|^2}} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.8}$$

Inversión La inversión de esta puerta se obtendría invirtiendo la puerta de giro especial que contiene, como se demuestra en la ecuación C.17.

Complejidad (P_{ams}) La profundidad de esta puerta es $P_{ams}(n) = 4(P_i(n)) + P_s(n-1) = \mathbf{O(n3^n)}_{(d1)} \vee \mathbf{O(n^2)}_{(d2)}$ siendo n el número de qubits.

4.3.4. Circuitos de ajuste a una matriz

Estos circuitos o puertas compuestas trabajan ya sobre una matriz inicial dada y generan el circuito interno que es capaz de llegar a dicha matriz.

4.3.4.1. Puerta de ajuste diagonal

Esta es la primera puerta que trabaja sobre una matriz *input* y la transforma en un circuito capaz de generar dicha matriz. En este caso solo resolveremos el problema para una matriz diagonal.
En este caso, el diseño elegido para esta puerta es la concatenación de puertas de giro Z condicionadas múltiples donde el *target* sea el último qubit y el resto de qubits sean *source*, cuya solución sería una matriz diagonal en la que el último elemento se ha sustituido el valor con el que se haya parametrizado el giro. Este giro se rodea con puertas de intercambio para que el valor del giro se coloque en la posición de la matriz que queremos modificar. Este paso se repite para cada uno de los elementos de la matriz diagonal y de este modo se genera el circuito cuyo funcionamiento teórico representa la matriz *input*. La demostración matemática de esta puerta se puede ver en el anexo C.7.
En la figura 4.21 ($R : x$ representa $R(3, x)$) se hace un ejemplo muy simple con 2 qubits que corresponda al circuito que genera la matriz 4.9.

$$\begin{bmatrix} \epsilon(\alpha) & 0 & 0 & 0 \\ 0 & \epsilon(\beta) & 0 & 0 \\ 0 & 0 & \epsilon(\gamma) & 0 \\ 0 & 0 & 0 & \epsilon(\delta) \end{bmatrix} \tag{4.9}$$

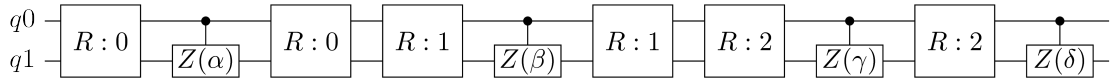


Figura 4.21: Diseño de puerta de ajuste diagonal para 2 qubits.

La matriz de este tipo de puertas que trabajan generando matrices, ya sean diagonales o completas, será la matriz que se está buscando.

No tendría sentido hablar de puerta inversa, ya que este circuito es en sí un constructor de una matriz, y no se necesita conocer su inversa (esta se daría invirtiendo las puertas de giro).

Complejidad (P_d) La complejidad de este circuito para n qubits será de $P_d(n) = N(P_m(n-1) + 2P_i(n)) + P_m(n-1) = \mathbf{O(Nn3^n)}_{(d1)} \vee \mathbf{O(Nn^3)}_{(d2)}$ siendo $N = 2^n$.

4.3.4.2. Puerta de ajuste matricial

Esta es la última puerta necesaria para la generalización del problema. Con ella conseguimos generar un circuito con una profundidad acotada superiormente y cuya funcionalidad se ajusta a una matriz dada como parámetro de la puerta.
Esta puerta concatena puertas AMS para cada uno de los valores distintos de 0 que se encuentren debajo de la diagonal de la matriz *input*, ordenando dichas puertas para ir recorriendo la matriz por columnas de izquierda a derecha y de arriba a abajo. Esto va generando circuitos que al concatenarse, generan la matriz final a la que se quiere llegar[7].
El objetivo es, para una matriz M dada, encontrar U_i tales que $\forall i \rightarrow U_i$ sea una matriz unitaria y diagonal exceptuando dos filas y dos columnas. Esto quiere decir que cada puerta U_i es una matriz generable por una puerta AMS. Entonces, partiendo de M, podemos calcular $U_0 M = V_0$, donde V_0 es una matriz similar a M pero donde uno de sus valores se ha transformado en 0. Este procedimiento se repite hasta m veces, siendo m el número de valores distintos de 0 debajo de la diagonal. A partir de la ecuación 4.10 vemos la concatenación de matrices que debemos realizar para obtener M, con un límite de m calculado en la ecuación 4.11.

$$\prod_{i=m}^{0}(U_i)M = I \Longrightarrow M = \prod_{i=0}^{m} U_i^{\dagger} \tag{4.10}$$

$$m < \sum_{i=1}^{N-1}(N-i) = \frac{N^2-N}{2} \tag{4.11}$$

De esta manera, concatenando las puertas AMS $U_i^{\dagger}$ obtendremos un circuito cuya matriz solución sea la dada.
Con este diseño, en una matriz donde una columna entera bajo la diagonal tenga valores 0, el valor de dicha columna en la diagonal no será obligatoriamente 1. Para evitar usar una puerta AMS para modificar un único valor en la diagonal, usaremos la puerta de ajuste diagonal. De esta manera, ahorraremos dos puertas de intercambio en cada punto de la diagonal que haya que modificar, y usaremos una puerta condicionada múltiple en vez de una puerta AMS (mucho más costosa).

El ejemplo general para cualquier circuito de 2 qubits se puede ver explicado matemáticamente en C.8, mientras que un ejemplo de uso se puede ver en el anexo E.

Complejidad (P_M) El límite superior de complejidad de este circuito para n qubits será de la mostrada en la ecuación 4.12 siendo $N = 2^n$.

$$P_M(n) = \frac{N^2-N}{2}(P_{ams}(n)) + P_d(n) \Rightarrow O(N^2 n 3^{n-1}) = \mathbf{O(n6^n)}_{(d1)} \vee O(N^2n^2) = \mathbf{O(2^{2n}n^3)}_{(d2)} \tag{4.12}$$

4.3.5. Resultado del algoritmo

El resultado del algoritmo QCMD es obtener un circuito de puertas básicas para generar una matriz dada. Por lo tanto, la puerta de **ajuste matricial** es la solución de dicho algoritmo. Esto quiere decir que, para una matriz dada, podemos generar una puerta de ajuste matricial que se adapte a dicha matriz, y el circuito necesario para implementar dicha puerta será el circuito solución buscado.

4.4. Complejidad total

El valor real de la cota superior del diseño se calcula despejando el número real de puertas máximo que se usará en cada caso, teniendo en cuenta que las puertas básicas tienen profundidad 1. Por lo tanto, para n qubits podemos despejar la tablas 4.I para el diseño 1. Hay que tener en cuenta la diferencia entre s, número de qubits source y n, número de qubits total. Las puertas con $s = 3$ serán en realidad puertas sobre 4 qubits.

Para el diseño 2 se ha tomado como diseño de la puerta Toffoli de dos *sources* como la representada en la figura 4.12b, con complejidad $P_t(2) = 2(P_m(2)) = 2(17) = 34$, y en la tabla 4.II se ve el cálculo completo de complejidad.

Vemos como el segundo diseño empieza a ser efectivo con puertas de giro condicionadas múltiples de más de 5 qubits, pero como la puerta Toffoli reduce su complejidad, el circuito final será más eficiente con el diseño 2 por encima de 4 quibts. Cálculo recogido en el anexo C.9(cálculo según la complejidad, no según el número real de puertas).

puerta	complejidad ($d1$)	profundidad máxima($d1$)	3 qubits	4 qubits	5 qubits	6 qubits
básica	$O(1)$	1	1	1	1	1
giro condicionado	$O(1)$	5	5	5	5	5
giro condicionado múltiple (s)	$O(3^s)$	$22 \cdot 3^{s-2}$	66	198	594	1782
toffoli (s)	$O(3^s)$	$44 \cdot 3^{s-2}$	132	396	1188	3564
giro especial (s)	$O(3^s)$	$154 \cdot 3^{s-2}$	462	1386	4158	12474
intercambio (n)	$O(n3^n)$	$44n \cdot 3^{n-3}$	132	528	1980	7128
AMS (n)	$O(n3^n)$	$176n \cdot 3^{n-3}$	528	2112	7920	28512
diagonal (n)	$O(n6^n)$	$2^n 88n \cdot 3^{n-3}$	2112	16896	126720	912384
matricial (n)	$O(n12^n)$	$2^{2n}176n \cdot 3^{n-3}$	33792	540672	8e(6)	1e(8)

Tabla 4.I: Cálculo de complejidad y profundidad máxima (simplificada al máximo componente). para el *diseño 1*

puerta	complejidad ($d2$)	profundidad máxima($d2$)	3 qubits	4 qubits	5 qubits	6 qubits
básica	$O(1)$	1	1	1	1	1
giro condicionado	$O(1)$	5	5	5	5	5
giro condicionado múltiple (s)	$O(s)$	$68s$	204	272	340	408
toffoli (s)	$O(s)$	$68s$	204	272	340	408
giro especial (s)	$O(s)$	$476s$	1428	1904	2380	2856
intercambio (n)	$O(n^2)$	$n68(n-1)$	408	816	1360	2856
AMS (n)	$O(n^2)$	$n272(n-1)$	1632	3264	5440	11424
diagonal (n)	$O(2^n n^2)$	$2^n n136(n-1)$	6528	26112	87040	731136
matricial (n)	$O(2^{2n} n^2)$	$2^{2n} n272(n-1)$	104448	835584	5e(6)	3e(7)

Tabla 4.II: Cálculo de complejidad y profundidad máxima (simplificada al máximo componente) para el *diseño 2*

Un dato que debemos tener en cuenta es que la construcción de la puerta Toffoli puede reducirse en gran medida a un diseño más eficiente, variando altamente los valores obtenidos. Y los valores obtenidos variarían más aún si se pudiese contar con una puerta Toffoli implementada físicamente. En la tabla 4.III se resume los valores de profundidad que resultarían de este algoritmo para el diseño 2 si la construcción de la puerta Toffoli fuese constante. Se puede ver como los valores y las complejidades se reducen vertiginosamente, y a esto se suma el hecho de que solo sería necesario 1 qubit auxiliar.

4.5. Mejoras

La complejidad total de todo el diseño del circuito tiene un carácter exponencial. Este crecimiento es insalvable siempre que el diseño de construcción se base en crear uno por uno los valores de la matriz, ya que el tamaño de la matriz crece de manera exponencial con el número de qubits.
Por lo tanto, las mejoras referidas a este algoritmo **QCDA** se podrían aplicar a mejorar ciertas puertas (al igual que el segundo diseño de la puerta condicionada múltiple) y a acotar y hacer más eficientes cada uno de los pasos del algoritmo. Pero nunca se reduciría el nivel exponencial del mismo $O(2^{2n})$. Existen otras vías para llegar a construir un circuito desde su matriz, como

puerta	complejidad ($d2$)	profundidad máxima($d2$)	3 qubits	4 qubits	5 qubits	6 qubits
básica	$O(1)$	1	1	1	1	1
giro condicionado	$O(1)$	5	5	5	5	5
giro condicionado múltiple (s)	$O(1)$	7	7	7	7	7
toffoli (s)	$O(1)$	1	1	1	1	1
giro especial (s)	$O(1)$	37	37	37	37	37
intercambio (n)	$O(n)$	$3n$	9	12	15	21
AMS (n)	$O(n)$	$12n$	36	48	60	84
diagonal (n)	$O(2^n n)$	$2^n 3n$	72	192	480	2688
matricial (n)	$O(2^{2n} n)$	$4^{n+1} 3n$	2304	12288	61440	1e(6)

Tabla 4.III: Cálculo de complejidad y profundidad máxima para el *diseño 2* con puerta Toffoli constante

sería de **Descomposición de Schmidt**[10], pero estas se referirían a un cambio en la idea principal de este algoritmo y a su base de funcionamiento, por lo que se podría considerar como algoritmos diferentes.

5

Implementación empírica del algoritmo

En este capítulo se incluye un resumen de un programa implementado para la demostración empírica del algoritmo *QCMD* y las pruebas realizadas sobre el mismo.

5.1. Implementación y simulador propio

En esta sección se va a explicar un programa implementado para generar el algoritmo **QCMD**. Este programa se ha escrito en lenguaje *Python* y simula el funcionamiento del algoritmo, generalizado a cualquier matriz unitaria y cualquier número de qubits de entrada. La salida generada es un fichero *.qasm* (extensión del archivo escrito en *ensamblador cuántico* para el ordenador de IBM).
Este programa no ha sido diseñado para presentar una implementación eficiente del algoritmo, sino que se ha pensado para ser una demostración empírica del funcionamiento del mismo. Más adelante podría contemplarse la idea de crear una librería para hacer aplicable dicho programa como un ejecutable o un sistema interno para desarrollar un escritor automático de código *quantum assembler* o **qasm**.

Este programa está publicado en un repositorio de *GitHub* y es público y accesible para cualquier persona que quiera ver su contenido. Este repositorio cuenta con varios scripts autónomos que ejecutan dicho algoritmo automáticamente (según varios parámetros) y demuestran empíricamente su funcionamiento mediante pruebas contra un simulador externo o contra un ordenador cuántico real (IBM-Q).

La dirección para clonar el repositorio es:
`https://github.com/jpUhryn/Quantum-Circuit-Matricial-Design-Algorithm.git`

(Este código se ha escrito y comentado en inglés.)

5.1.1. Diseño

El diseño que se ha elegido basa su funcionamiento en dividir el circuito en las mismas puertas que se han visto con anterioridad en la sección 4.3[1].
Todas estas puertas se guardan en un *diccionario*, mapeadas por un **id** único que se va asignando según se van generando dichas puertas. Cada puerta cuenta con una matriz que la representa y un *path* que guarda los ids de las puertas que se han concatenado hasta llegar a ella. Cada vez que se crea una nueva puerta, el programa genera el *path* necesario para construirla, y recursivamente va generando las puertas inferiores necesarias que no hayan sido creadas aún. Una vez creadas todas las puertas que se necesiten, se genera la matriz de la puerta, multiplicando las matrices de las puertas inferiores (las cuales a su vez se han generado multiplicando las anteriores). Las únicas matrices que se generan *automáticamente* son las que representan las puertas básicas (aquellas que no se consiguen por concatenación), por lo que el resultado final de implementar un circuito desde una matriz es realmente la matriz del circuito. De esta manera, en el caso de la puerta de ajuste matricial, sabemos que la matriz de dicha puerta es la generada por el circuito y no la dada como parámetro de entrada.

Para ahorrar tiempo de ejecución, se ha diseñado un sistema de **programación dinámica** donde, a la hora de buscar si una puerta está ya creada o no, utiliza un sistema de *diccionarios*, donde el primer diccionario contiene como clave cada uno de los distintos tipos de puerta, y cada elemento del diccionario es a su vez un diccionario que se mapea mediante un parámetro de la puerta. Estos diccionarios contienen en su último atributo el *id* de una de las puertas que responde a todos los parámetros. Si uno de los parámetros aún no ha sido incluido en el diccionario indica que esta puerta aún no existe, y se auto genera esa rama para incluir dicho *id*. De esta manera se puede saber rápidamente si una puerta existe ya, y si no crearla e instanciarla en el diccionario automáticamente. De esta forma se reduce el tamaño del diccionario (si contuviese todos los parámetros desde el principio tendría una dimensión demasiado grande, y si no se guardasen las puertas se harían demasiadas llamadas recursivas a creación de puertas simples), reducir el tiempo de búsqueda y evitar generar una puerta que ya existe. El único punto importante a la hora de trabajar con este *diccionario de diccionarios* es saber que el orden de los parámetros para cada tipo de puerta siempre debe ser el mismo.

El pseudocódigo 5.1 resume el funcionamiento de la creación de una puerta en el programa.

```
1  crearPuerta(tipo, parametros, diccionario):
2
3    #puerta a buscar
4    nuevaPuerta
5
6    if nuevaPuerta en diccionario:
7      #si ya existe se devuelve la existente
8      return nuevaPuerta.id
9
10   #si no existe se genera
11   else:
12
13     # lista de puertas necesarias para crear la puerta actual
14     circuitoNecesario = generarCircuito(parametros)
15
16     # puertas que generaran la puerta buscada
17     circuito = listaVacia
18
19     for puertaNecesaria en circuitoNecesario: #para cada puerta necesaria
20
```

[1]Esta implementación tiene algunas diferencias debido a que el algoritmo ha ido evolucionando desde que se implementó. Por ejemplo, no existe la *puerta diagonal*. Además la implementación sigue únicamente el **diseño 1**.

```
21        # si existe ya devuelve el id, si no crea la puerta y devuelve el nuevo id
22        circuito.annadir(crearPuerta(puertaNecesaria, parametros, diccionario))
23
24    #crea la matriz mediante las matrices inferiores
25    nuevaMatriz = crearMatriz(circuito.matriz)
26
27    #annade la nueva puerta al diccionario
28    nuevoId = diccionario.annadir(nuevaPuerta, parametros, nuevaMatriz,
      diccionario)
29
30    #devuelve el id de la nueva puerta
31    return nuevoId
```

Listing 5.1: Pseudocódigo de funcionamiento

5.1.2. Complejidad

El cálculo más pesado en un programa que tenga funcionalidad de *simulador cuántico* es a la hora de multiplicar las matrices, ya que esto equivale a una complejidad de $O(N^3)$ siendo N el tamaño de la matriz.
Mediante el uso del *diccionario* (reusando puertas) se ahorra tiempo evitando generar varias veces la misma puerta, lo cual puede llegar a hacerse pesado en aquellas con mucha profundidad; y también evita la multiplicación excesiva de matrices, ya que una vez se haya generado una puerta no es necesario volver a recalcularla. Con este diseño también nos aseguramos que cada puerta está generada correctamente, ya que su matriz no se toma de su definición si no que se calcula (exceptuando el caso de las puertas básicas).
Por lo tanto, se tiene un programa que funciona a la vez como implementación del algoritmo **QCMD** y como simulador de un ordenador cuántico, cuya complejidad total depende del número de puertas inferiores que necesite la puerta en *construcción*. Esto da una complejidad de

$$O(2^{3n} \cdot 2^n n 3^{n-1}) = O(2^{4n} n 3^{n-1})$$

siendo n el número de qubits.

Sería fácil eliminar de este diseño el apartado de multiplicación de matrices, usado como forma de depurar la salida obtenida, y de esta manera el programa seguiría funcionando reduciendo su complejidad, pero sin opción a conocer la solución del circuito creado.

5.1.3. Interfaz

El programa basa su funcionamiento en recibir el tipo y los parámetros de un tipo de puerta en concreto, y un *diccionario* sobre el que trabajar. Como salida devuelve el *id* de la puerta solución, y en el diccionario se encuentran todas las puertas inferiores que se han utilizado. A través de este *id* se puede visualizar la matriz, el *path* de los ids de las puertas directamente inferiores y escribir dicha puerta en código *qasm*, el cual se genera de manera automática y recursiva, entrando desde el *diccionario* en cada puerta hasta llegar a una puerta básica, la cual escribe su instrucción o instrucciones ensamblador.

```
1  # inicializa un sistema de diccionarios para 4 qubits
2  dic = DictionarySearcher(nQubits=4)
3
4  # genera una puerta de giro condicionado multiple y las subpuertas necesarias
5  gateId = AQP.generateMultipleTurn(sources=[0,1,2], target=3, angle=math.pi/2,
     turnType=TurnType.Z, dic=dic)
6
7  # genera el fichero .qasm para crear esta puerta
```

```
8   WriterAsm.writeAsm(gateId, dic, "multiConditionalGate.qasm", reset=True)
```

Listing 5.2: Ejemplo básico de uso (faltan librerías e inicializaciones)

```
1  // Assembler file: multiConditionalGate.qasm for id: 35 //
2
3  OPENQASM 2.0;
4  include "qelib1.inc";
5  qreg q[4];
6  creg c[4];
7
8  // X-NOT - source: 2 target: 3 //
9  cx q[1], q[0];
10 // X-NOT - source: 2 target: 3 //
11
12 // Basic Turn - Z target: 3 angle: 5.890486225480862 //
13 u1(5.890486225480862) q[0];
14 // Basic Turn - Z target: 3 angle: 5.890486225480862 //
```

Listing 5.3: Estracto del código *.qasm* obtenido

En el código 5.2 se puede observar un uso simple de dicho programa (se han omitido *imports* y demás configuraciones anteriores) para generar una puerta de giro condicionado múltiple con 3 qubits *sources*, y un ángulo de $\pi/2$ sobre el eje Z. En el código mostrado en 5.3 se puede apreciar por los comentarios que los qubits del circuito se invierten para concordar con los estándares tomados para el compilado de *.qasm*. Las matrices y resultados no se ven afectados, solo se modifica el hecho de elegir el qubit más significativo.

5.1.4. Otros Posibles Diseños

- **Prescindir de las matrices**: Se puede evitar guardar cada matriz de cada puerta ya que se conoce la solución esperada para cada puerta. De esta manera se ahorraría una gran cantidad de memoria y procesamiento al evitar la multiplicación. Este diseño no permitiría comprobar de forma eficiente la funcionalidad del algoritmo.

- **Usar funcionalidad *qasm***: El lenguaje *qasm* cuenta con un método de creación de *funciones* para evitar repeticiones de puertas. Esto podría disminuir el tiempo a la hora de escribir el fichero *.qasm*. Esto no variaría el número de puertas final a ejecutar sobre el ordenador.

- **Automatizar diseño de nuevas puertas**: Se podría implementar el *diccionario* para que permitiese incluir nuevos tipos de puertas en tiempo de ejecución, tomando los parámetros como un *array* o *diccionario*.

- **Implementar el *diseño 2***: Implementar el *diseño 2* del algoritmo permitiría una eficiencia en profundidad mucho mayor. Este nuevo diseño requeriría aumentar el número de qubits de la matriz de forma exponencial, y por lo tanto sería ineficiente a la hora de tener que aguantar el cálculo de la multiplicación de matrices mayores. Esto podría implementarse solo si se usase también el nuevo diseño comentado *prescindir de las matrices*.

- **Revisar eficiencia de las puertas**: Las puertas se han generado mediante un diseño que obtuviese la solución esperada, pero no de la manera más eficiente posible. Estos tipos de puerta se podrían rediseñar para encontrar una solución más rápida y eficiente. Por ejemplo, en la puerta de giro especial, está integrado el uso de la operación identidad para que la matriz sea la esperada.

5.2. Pruebas con IBM-Q

Todas las pruebas aquí mencionadas se pueden llevar a cabo descargándose el código del repositorio y lanzando los scripts siguiendo las instrucciones de los mismos.

5.2.1. Simulador

Se ha diseñado un programa ejecutable en *Python* donde, dando una matriz deseada, calcula el circuito asociado. Tras esto, genera el fichero *.qasm* donde recoge el circuito de dicha puerta, y lo ejecuta en un *simulador local* incluido en la *API* **QISKit**, librería *OpenSource* desarrollada por *IBM* para el uso del computador cuántico desde la *nube*. De esta ejecución obtenemos la matriz que representa a dicho circuito, y así podemos contrastar que la solución es la esperada, y con esto podemos demostar que **el algoritmo funciona de forma teórica**.

El programa relativo a esta ejecución y sus distintos tipos de prueba se puede ver en el anexo F.1. Este código compara las matrices generadas por el programa propio, o en el caso de las puertas de ajuste matricial, las matrices que se desean generar, con las matrices que genera dicho código *.qasm* en el simulador de *IBM*. Se usa la función *allclose* de la librería *numpy* para comparar dichas matrices y evitar el error de las operaciones en coma flotante con un error relativo de *e-8* y un error absoluto de *e-5* (valores por defecto de la función *allclose* de numpy). La salida de dicho programa es una recopilación de todas las pruebas, donde se muestra si las matrices generadas y las simuladas son las mismas.

5.2.2. Ordenador

Para demostrar empíricamente el funcionamiento del algoritmo, se ha diseñado un programa que prueba la eficiencia del algoritmo sobre el ordenador real. Este programa prueba la puerta de intercambio y la puerta de ajuste matricial para una matriz determinada para cualquier número de qubits. Se han elegido estas dos puertas debido a que el estado inicial del ordenador es el estado $|00..., 0\rangle$ de menor energía, por lo tanto cualquiera de las otras puertas no afectarían a dicho estado.

Las gráficas 5.1 y 5.2 representa una ejecución de dicho programa, en el cual se puede ver una simulación exacta del resultado teórico de un ordenador cuántico (azul), una simulación aleatoria más semejante al comportamiento esperado del ordenador (amarillo) y las mediciones reales llevadas a cabo en el ordenador cuántico de *IBM*. En la Gráfica 5.1 el circuito que se ha simulado es un circuito generado por el algoritmo *QCMD* para la construcción de un estado inicial superpuesto para los valores $|00\rangle$ y $|11\rangle$, y se ha generado mediante una puerta de ajuste matricial con *input* la matriz 5.1.

$$\begin{bmatrix} \frac{1}{\sqrt{2}} & 0 & 0 & \frac{1}{\sqrt{2}} \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ \frac{1}{\sqrt{2}} & 0 & 0 & -\frac{1}{\sqrt{2}} \end{bmatrix} \tag{5.1}$$

En la Gráfica 5.2 el circuito que se ha simulado es un circuito de intercambio de estados en 3 qubits, del estado $|000\rangle$ al estado $|101\rangle$, que equivale a la matriz 5.2.

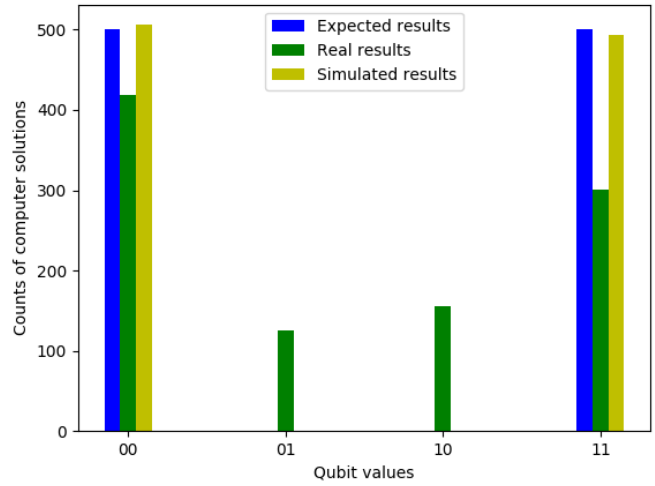


Figura 5.1: Resultados del ordenador cuántico para un circuito de 2 qubits.

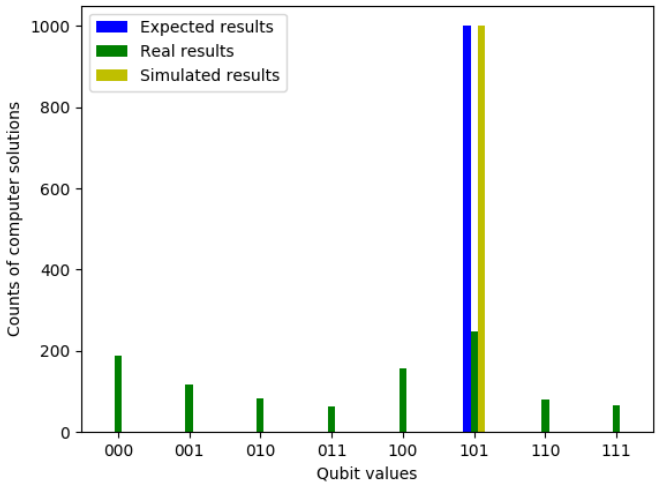


Figura 5.2: Resultados del ordenador cuántico para un circuito de 3 qubits.

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{5.2}$$

Se puede ver fácilmente como el resultado dado por el ordenador cuántico tiene una tasa de error muy alta, aunque se puede apreciar que los estados solución cuentan con una pequeña probabilidad superior al resto de estados.
También se puede ver que con mayor número de qubits, el error de la medición aumenta.

6

Conclusiones y trabajo futuro

6.1. Conclusiones

La computación cuántica es una rama de investigación aún en sus inicios, con una complejidad matemática, física y computacional muy alta, y con un gran horizonte de posibilidades aún por abrirse.
Una de las conclusiones de este trabajo de fin de grado ha sido conocer su funcionamiento, y conocer sus diferencias con respecto a un ordenador lógico. Las leyes físicas que rigen su funcionamiento, que son aquellas que ofrecen tan alta potencia de cálculo a este paradigma computacional, al mismo tiempo son las mismas que le impiden comportarse como una computadora lógica usual. Los principios de no almacenamiento y no copia de información limitan en cierta medida las posibilidades de estas máquinas.
Pero a pesar de estas limitaciones, se trata de máquinas que son capaces de realizar cálculos en segundos donde una máquina lógica necesitaría más tiempo que la edad misma del universo. Por lo tanto, puede que no sean las sustitutas de los ordenadores actuales, pero serán igual de necesarias en el mundo moderno.

El algoritmo diseñado y explicado en este trabajo no es más que una primera aproximación a un algoritmo posiblemente necesario en un futuro para todos aquellos que quieran hacer uso de la computación cuántica. Es una idea inicial de como abstraer el problema de generar un programa cuántico, desde el estado actual de la programación cuántica, que es usando directamente instrucciones sobre el *hardware*, a una idea más generalizable y puramente teórica de nivel *software*. El algoritmo diseñado abstrae por completo el problema de la programación o implementación real del algoritmo, reduciendo el problema de la algoritmia al campo puramente matemático.

El estado actual de la computación cuántica limita en gran medida el uso de este o cualquier otro algoritmo cuántico. Pero cualquier avance *software* sobre ella servirá para acortar el futuro camino que se deberá recorrer para encontrarse a la par con el desarrollo y mejora del *hardware* de dicha tecnología.

6.2. Trabajo futuro

El algoritmo propuesto en este trabajo no está optimizado para ser eficiente, y también cuenta con la desventaja de su complejidad exponencial. Pero a pesar de estas limitaciones, es una demostración empírica de que cualquier resultado matricial puede ser programable en un ordenador cuántico con un límite máximo de operaciones requeridas; y es una primera aproximación necesaria para la abstracción del uso de dicho ordenador.

Por lo tanto, el trabajo futuro de este algoritmo podría dividirse en tres puntos importantes.

En primer lugar, se podría ajustar la eficiencia de cada una de las puertas intentando reducir su cota máxima, lo que resultaría en un algoritmo de similar complejidad máxima, pero disminuiría mucho el número de puertas en un diseño real.

En segundo lugar, se podría mejorar dicho algoritmo buscando patrones de matrices conocidas para evitar su construcción si ya existe una implementación eficiente de la misma.

En tercer lugar, se podrían buscar nuevas formas de dividir una matriz unitaria para su creación, haciendo así que disminuya la complejidad total del algoritmo.

En conclusión, este algoritmo (al igual que la computación cuántica) se encuentra en un estado inicial, que requiere de más estudio e innovación. Pero es un importante primer paso para generalizar y abstraer el uso de la programación cuántica.

Bibliografía

[1] Danski14. Own work, cc by-sa 3.0 https://commons.wikimedia.org/w/index.php?curid=18415805.

[2] Varios autores. Quantum computing news https://quantumcomputingreport.com/news/.

[3] Brad Jones. Three things you need to know about google's new quantum processor https://futurism.com/googles-new-quantum-processor/.

[4] Elizabeth Gibney. Nature. d-wave upgrade how scientist are using the worlds most controversial quantum computer.

[5] Stephan Jordan. Algebraic and number theoretic algorithms https://math.nist.gov/quantum/zoo/.

[6] U. Alvarez-Rodriguez, M. Sanz, L. Lamata, and E. Solano. Quantum Artificial Life in an IBM Quantum Computer. *ArXiv e-prints*, November 2017.

[7] Michael A. Nielsen and Isaac L. Chuang. *Quantum Computation and Quantum Information: 10th Anniversary Edition*. Cambridge University Press, New York, NY, USA, 10th edition, 2011.

[8] Profesionales y estudiantes de la computación cuántica. Foro online https://qiskit.slack.com/.

[9] Patrick J. Coles, Stephan Eidenbenz, Scott Pakin, Adetokunbo Adedoyin, John Ambrosiano, Petr M. Anisimov, William Casper, Gopinath Chennupati, Carleton Coffrin, Hristo Djidjev, David Gunter, Satish Karra, Nathan Lemons, Shizeng Lin, Andrey Y. Lokhov, Alexander Malyzhenkov, David Mascarenas, Susan M. Mniszewski, Balu Nadiga, Dan O'Malley, Diane Oyen, Lakshman Prasad, Randy Roberts, Philip Romero, Nandakishore Santhi, Nikolai Sinitsyn, Pieter Swart, Marc Vuffray, Jim Wendelberger, Boram Yoon, Richard J. Zamora, and Wei Zhu. Quantum algorithm implementations for beginners. *CoRR*, abs/1804.03719, 2018.

[10] *Quantum Computation and Quantum Information*. Cambridge University Press, New York, NY, USA, 2000.

[11] Varios autores. Qiskit - quantum information science kit https://qiskit.org/.

[12] Matthew Wardrop. latex-blochsphere https://github.com/matthewwardrop/latex-blochsphere.

[13] Thomas Draper Sandy Kutin. qpic - free software https://github.com/qpic/qpic.

[14] Giuliano Benenti, Giulio Casati, and Giuliano Strini. *Principles of Quantum Computation And Information: Basic Tools And Special Topics*. World Scientific Publishing Co., Inc., River Edge, NJ, USA, 2007.

[15] Diederik Aerts and Massimiliano Sassoli de Bianchi. The extended bloch representation of quantum mechanics and the hidden-measurement solution to the measurement problem. *Annals of Physics*, 351:975 – 1025, 2014.

[16] V. V. Shende, S. S. Bullock, and I. L. Markov. Synthesis of quantum-logic circuits. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 25(6):1000–1010, June 2006.

[17] C. Lavor, L. R. U. Manssur, and R. Portugal. Grover's Algorithm: Quantum Database Search. *eprint arXiv:quant-ph/0301079*, January 2003.

[18] C. M. Dawson and M. A. Nielsen. The Solovay-Kitaev algorithm. *eprint arXiv:quant-ph/0505030*, May 2005.

[19] Varios autores. Ibm quantum experience - user guide `https://quantumexperience.ng.bluemix.net/qx/tutorial?sectionId=full-user-guide&page=introduction`.

[20] K. Srinivasan, B. K. Behera, and P. K. Panigrahi. Solving Linear Systems of Equations by Gaussian Elimination Method Using Grover's Search Algorithm: An IBM Quantum Experience. *ArXiv e-prints*, December 2018.

[21] A. Wichert. *Principles of Quantum Artificial Intelligence*. World Scientific, 2013.

[22] Varios autores. Ibm quantum experience - advanced single-qubit gates `https://quantumexperience.ng.bluemix.net/qx/tutorial?sectionId=full-user-guide&page=002-The_Weird_and_Wonderful_World_of_the_Qubit~2F004-advanced_qubit_gates`.

Anexos

A

Algoritmo de Grover

El **Algoritmo de Grover** es un algoritmo cuántico para la búsqueda de un elemento en una secuencia no ordenada de datos.

Este algoritmo, junto con el **Algoritmo de Shor** para la factorización de números, son los algoritmos cuánticos más conocidos, debido a su utilidad a la hora de agilizar cálculos actualmente complejos para la computación lógica.
Este algoritmo recibe su nombre del científico de la computación Lov Kumar Grover (nacido en 1961) a quien se atribuye su invención en 1996.

Este algoritmo es famoso debido a que consigue una complejidad $O(\sqrt{N})$ al buscar un elemento en una cadena de N elementos **no ordenados** cuando un algoritmo de computación lógica no puede bajar de una complejidad $O(N)$[17].

A.1. Flujo

El algoritmo de Grover es un algoritmo iterativo generalizable a cualquier número de qubits, que basa su funcionamiento en tres pasos. La idea principal es partir de un estado de **superposición** donde todas las cadenas de *bits* tengan la misma probabilidad de ser medidas, y aplicar iterativamente un **oráculo**, que es una función que **invierte** aquella cadena que se está buscando; y un **amplificador de amplitud**, que aumenta la probabilidad de medir aquellos valores que se encuentran en negativo.

A.1.1. Estado de superposición

El primer paso es inicializar el estado principal a un **estado de superposición**. Esto es, un estado donde todos los qubits se encuentren superpuestos (igual probabilidad de ser 0 y 1). Para esto, se utiliza una puerta *Hadamard* para cada uno de los qubits, convirtiendo el estado de mínima energía $|00..,0\rangle$ en un estado de *superposición* $|++..,+\rangle$.

A.1.2. Oráculo

Esta parte de la iteración es particular para cada elemento a buscar, y representa la función de búsqueda. Esta función debe tener la peculiaridad de que $f(x) = 1$ para todos aquellos valores que se están buscando, y $f(x) = 0$ en otro caso.
Lo que se busca es usar una función $\langle U|$ sobre un estado cualquiera $|x\rangle$ de tal forma que $\langle U_\omega|x\rangle = (-1)^{f(x)}|x\rangle$. De esta forma, aquellos estados que *activen* o devuelvan un 1 en la función, sufrirán una inversión de signo (recordar que esto no afecta a su amplitud, ni, por lo tanto, a su probabilidad).

A.1.3. Amplificación de Amplitud

Esta parte del algoritmo es generalizable para cualquier búsqueda para el mismo número de elementos. Lo que se busca con esta amplificación es devolver los estados negativos a un valor positivo, y a la vez aumentar su amplitud, disminuyendo así la de las demás cadenas.

A.1.4. Iteración de Grover

Debido al carácter probabilístico de la computación cuántica, no podemos estar seguros de que vayamos a llegar al resultado esperado (un estado donde la función tenga valor 1). Por eso el algoritmo aplica de manera recursiva al estado de superposición la transformación **oráculo** y **amplificación de amplitud**. De esta manera, con $O(\sqrt{N})$ iteraciones, podemos estar seguros de que al medir encontraremos el elemento buscado.

A.2. Variantes

Variante 1 Existe una variante de Grover que es menos conocida que el algoritmo general, que se basa en **la búsqueda sobre más de un elemento**. El algoritmo se usa generalmente con funciones cuyo valor es 1 para un único valor. Esta variante incluye la variable k que representa el número de valores en el dominio de la función que tienen como resultado 1.
Esto hace que aumente la probabilidad de encontrar uno de entre todos estos valores, más específicamente, con repetir la iteración $O(\sqrt{\frac{N}{k}})$ veces obtenemos la misma probabilidad que al buscar un solo elemento.
Esta variante tiene un inconveniente, que es que la mayoría de las veces no conoceremos el valor k (porque no conocemos la función a priori).

Variante 2 Existe otra variante de Grover que no se ha encontrado en la bibliografía, que se basa en **la búsqueda sobre un subconjunto**.
Esto se consigue modificando la función de amplificación de amplitud para que no *modifique* aquellos estados que no intervienen en la búsqueda.
Pero esto genera que aquellos estados sobre los que no se busque tendrán una probabilidad *alta* de obtenerse como resultado, ya que no se aumenta su amplitud pero tampoco se reduce.

Esto se solventa usando un estado inicial de superposición donde las cadenas sobre las que no se busque no se introduzcan como índice. Esto no se puede conseguir de una forma trivial, ya que conseguir un estado de superposición es fácil usando puertas *Hadamard*, pero para

conseguir un subconjunto de estados que tengan la misma amplitud solo se puede conseguir mediante la creación de un circuito anterior que convierta el estado inicial en el estado deseado. Si el subconjunto sobre el que se trabaja es fácil de *dividir* por sus qubits (por ejemplo solo los valores pares o los impares, ya que solo se diferencian en el último qubit) entonces esta transformación es trivial, bastaría con no superponer dicho qubit. Pero en cualquier otro caso, el estado de superposición se alcanza mediante entrelazamiento de distintos qubits, difícil de alcanzar de forma manual, y conseguido de forma automática con un algoritmo de diseño de circuitos mediante matriz.

A.3. Motivación

Este algoritmo se puede usar como un algoritmo de búsqueda sobre un conjunto desordenado con una complejidad inalcanzable en un ordenador lógico (no paralelizado). Otra aplicación del algoritmo es encontrar la **función inversa** a una función dada, ya que el algoritmo nos permite averiguar para que valores la función tiene resultado 1 sin tener que recorrer todos los posibles valores.
Existe el problema de que el oráculo es difícilmente implementable, ya que su utilidad reside en que la función es desconocida. Por eso, en muchas ocasiones, si se quiere construir esta función es necesario conocer previamente su resultado para todos los valores, lo que hace que el algoritmo pierda su utilidad. Por eso su uso por ahora es muy limitado, aunque existen estudios sobre la obtención de un oráculo sin necesidad de recorrer todos los posibles valores.

Este algoritmo ha motivado la idea principal de este Trabajo de Fin de Grado. Se conoce matemáticamente la implementación y las matrices que dan su funcionalidad al algoritmo de Grover (la matriz del oráculo y la matriz de la amplificación de amplitud) pero hasta ahora no existía (o no se ha encontrado tras una exhaustiva investigación, ver capítulo 2) una manera de conseguir un circuito que implementase una matriz concreta.
Es cierto que existían medios más eficientes que el algoritmo propuesto en este trabajo para generar los oráculos y la amplificación de amplitud, pero no existe la forma de generalizarlo para cualquier matriz, lo cual es muy interesante a la hora de construir las variantes del algoritmo descritas previamente, o cualquier otro algoritmo cuya matriz o matrices sean conocidas.

A.4. Demostraciones matemáticas sobre el algoritmo de Grover

Para llevar a cabo estas demostraciones, entre las cuales se encuentran demostraciones encontradas y cálculos propios, se ha buscado mucha información en diferentes puntos de la bibliografía. [20][21]

A.4.1. Modelo básico

Este modelo representa una búsqueda de un elemento sobre una cadena de N elementos no ordenados, siendo $N = 2^n$ y n el número de qubits del circuito.

La cadena a buscar se representa mediante el estado $|\omega\rangle = (0, 0.., 1.., 0, 0)^t$.

El estado $|s\rangle = (1, 1.., 1, 1)^t \frac{1}{\sqrt{N}}$ representa el estado de superposición de todos los qubits.

La transformación $\langle U_\omega|$ representa el oráculo de Grover para el estado ω, representado en la ecuación A.1.

La transformación $\langle U_a|$ representa la amplifiación de amplitud, representado en la ecuación A.2.

$$\langle U_\omega| = \langle I| - 2\,|\omega\rangle\,\langle\omega| \tag{A.1}$$

$$\langle U_a| = 2\,|s\rangle\,\langle s| - \langle I| \tag{A.2}$$

El estado $|s'\rangle$ representa el vector perpendicular a $|\omega\rangle$.

$$\big|s'\big\rangle = (|s\rangle\,\sqrt{N} - |\omega\rangle)\frac{1}{\sqrt{N-1}} = (1, 1.., 0.., 1, 1)\frac{1}{\sqrt{N-1}} \tag{A.3}$$

A continuación se recogen los productos vectoriales entre estos estados. el producto $\langle\omega|s\rangle$ representa la amplitud del estado $|\omega\rangle$ sobre $|s\rangle$, lo que implica que $\langle\omega|s\rangle^2$ es la probabilidad de medir $|\omega\rangle$.

$$\langle\omega|s\rangle = \frac{1}{\sqrt{N}} \tag{A.4}$$

$$\big\langle\omega\big|s'\big\rangle = 0 \tag{A.5}$$

$$\big\langle s\big|s'\big\rangle = \frac{\sqrt{N-1}}{\sqrt{N}} \tag{A.6}$$

Estas ecuaciones representan las transformaciones que se realizan sobre cada uno de los estados, para más tarde realizar un cálculo de una primera iteración de Grover.

$$\langle U_\omega|\omega\rangle = \langle I|\omega\rangle - 2\,|\omega\rangle\,\langle\omega|\omega\rangle = |\omega\rangle - 2\,|\omega\rangle = -\,|\omega\rangle \tag{A.7}$$

$$\langle U_\omega | s' \rangle = \langle I | s' \rangle - 2 |\omega\rangle \langle \omega | s' \rangle = |s'\rangle - 0 = |s'\rangle \tag{A.8}$$

$$\langle U_\omega | s \rangle = \langle I | s \rangle - 2 |\omega\rangle \langle \omega | s \rangle = |s\rangle - 2 |\omega\rangle \frac{1}{\sqrt{N}} = |s\rangle - \frac{2}{\sqrt{N}} |\omega\rangle = \\ = (1, 1.., -1.., 1, 1)\frac{1}{\sqrt{N}} = |x_0\rangle \tag{A.9}$$

$$\langle U_a | \omega \rangle = 2 |s\rangle \langle s | \omega \rangle - \langle I | \omega \rangle = \frac{2}{\sqrt{N}} |s\rangle - |\omega\rangle \tag{A.10}$$

$$\langle U_a | s \rangle = 2 |s\rangle \langle s | s \rangle - \langle I | s \rangle = |s\rangle \tag{A.11}$$

A continuación se muestra como resultaría la primera iteración del algoritmo (oráculo más amplificador de amplitud) sobre el estado $|s\rangle$.

$$\langle U_a U_\omega | s \rangle = \langle U_a | x_0 \rangle = 2 |s\rangle \langle s | x_0 \rangle - \langle I | x_0 \rangle = 2 |s\rangle \langle s | s \rangle - \frac{4}{\sqrt{denN}} |s\rangle \langle s | \omega \rangle - \langle I | s \rangle + \frac{2}{\sqrt{N}} \langle I | \omega \rangle = \\ = \left(1 - \frac{4}{N}\right) |s\rangle + \frac{2}{\sqrt{N}} |\omega\rangle = |y_0\rangle \tag{A.12}$$

Como se calcula en A.13 según el resultado de A.12 podemos ver que la probabilidad de obtener ω tras la primera iteración de Grover se ha aumentado de $\frac{1}{N}$ a $\frac{3N-4}{N\sqrt{N}}$.

$$\langle y_0 | s \rangle = \frac{2}{N} \tag{A.13}$$

Se puede demostrar el funcionamiento del algoritmo de Grover mediante una visión geométrica del mismo. En la figura A.1 se ve representados los estados $|\omega\rangle$ y $|s'\rangle$ perpendiculares, y el estado $|s\rangle$ entre medias. Sabemos que $|s\rangle$ tiene que encontrarse en el mismo plano que generan los otros dos estados, así como cualquier punto intermedio $|y_i\rangle$ resultado de una iteración del algoritmo, ya que $|s\rangle$ puede representarse mediante una combinación lineal como se muestra en la ecuación A.14, y las transformaciones que se llevan a cabo modifican a y b, lo que mantiene $|y_i\rangle$ en el mismo plano.

$$|s\rangle = a |\omega\rangle + b |s'\rangle = \frac{1}{\sqrt{N}} |\omega\rangle + \frac{\sqrt{N-1}}{\sqrt{N}} |s'\rangle \tag{A.14}$$

Como vemos en la figura A.1, el oráculo de la función genera una inversión del estado al que se aplique con respecto al eje $|s'\rangle$, y la amplificación de amplitud genera una inversión con respecto al eje $|s\rangle$. De esta forma, vemos que en cada iteración de grover, la amplitud del estado $|\omega\rangle$ aumenta en un ángulo θ, cuyo valor se calcula en la ecuación A.15.

$$cos(\frac{\theta}{2}) = \langle s | s' \rangle = \frac{\sqrt{N-1}}{\sqrt{N}} \Rightarrow sin(\frac{\theta}{2}) = \sqrt{1 - \frac{N-1}{N}} \Rightarrow \theta = 2sin^{-1}(\frac{1}{\sqrt{N}}) \tag{A.15}$$

Sabemos que para cada iteración el ángulo aumenta en θ y sabemos que la probabilidad de medir $|\omega\rangle$ es 1 menos la probabilidad de medir $|s'\rangle$, que sabemos que es $cos^2(\phi)$ siendo

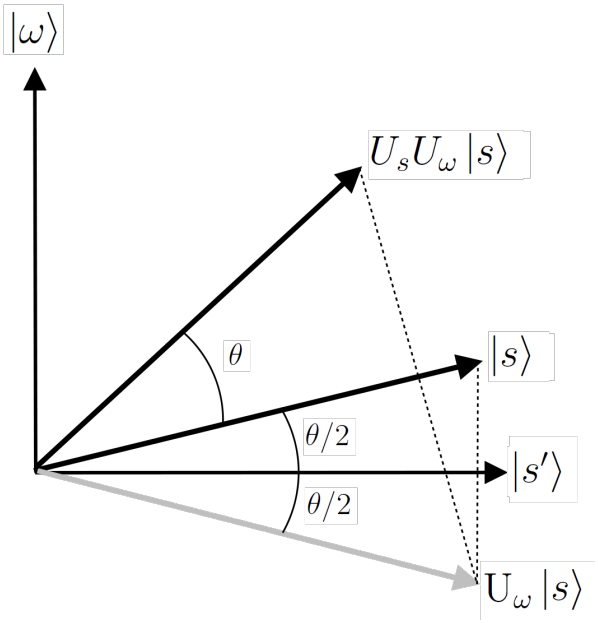


Figura A.1: Representación geométrica de una iteración de Grover[1].

$\phi = t\frac{1}{2}\theta$ siendo t el número de iteraciones de Grover aplicadas. De esta forma sabemos que la probabilidad de medir $|\omega\rangle$ es $sin^2((t+\frac{1}{2})\theta)$.

Calculando la derivada de esta función con respecto a t (número de iteraciones) podemos encontrar los máximos de dicha función, y el primero de ellos será aquel que nos de el mejor resultado de Grover en el menor número de iteraciones. Como se calcula en la ecuación A.16

$$(sin^2(t\frac{1}{2}\theta))' = 0 \Leftrightarrow 2sin((t+\frac{1}{2})\theta)cos((t+\frac{1}{2})\theta)\theta = 0 \Leftrightarrow$$
$$\Leftrightarrow (t+\frac{1}{2})\theta = 0 \vee (t+\frac{1}{2})\theta = \frac{\pi}{2} \Leftrightarrow t = \frac{-1}{2} \vee t = \frac{\pi}{2\theta} - \frac{1}{2} \quad \text{(A.16)}$$

Como t debe ser un número positivo, sabemos que $t = \frac{\pi}{2\theta} - \frac{1}{2}$, y despejando la ecuación en A.18, de la que sacamos que la complejidad del algoritmo de Grover es $O(\sqrt{N})$.

$$lim_{N\to\infty}(sin^{-1}(\frac{1}{\sqrt{N}})) = \frac{1}{\sqrt{N}} \quad \text{(A.17)}$$

$$t = \frac{\pi}{2\theta} - \frac{1}{2} \Rightarrow t = \frac{\pi}{4sin^{-1}(\frac{1}{\sqrt{N}})} - \frac{1}{2} \Rightarrow t \approx \frac{\pi\sqrt{N}-2}{4} \quad \text{(A.18)}$$

A.4.2. Búsqueda de varios elementos

En este modelo del algoritmo, el estado ω es la suma de más de un estado buscado, de forma que su probabilidad de ser resultado sobre el estado $|s\rangle$ es $\frac{k}{\sqrt{N}}$ donde k es el número de estados buscados.

Realizando los cálculos análogos al *modelo básico* hayamos que $\theta = 2sin^{-1}(\frac{\sqrt{k}}{\sqrt{N}})$, y por tanto el número de iteraciones óptimo para encontrar una solución es $O(\frac{\sqrt{N}}{\sqrt{k}})$.

El principal problema de este modelo es que comúnmente no se conocerá k de antemano, y el algoritmo empeora la probabilidad de encontrar una solución si se realizan excesivas iteraciones.

A.4.3. Búsqueda sobre un subgrupo

En este modelo del algoritmo se utiliza una búsqueda sobre n qubits y sobre N elementos, pero en este caso N no es necesariamente 2^n. Eso quiere decir que el estado $|\omega\rangle$ (buscado) y el estado $|s'\rangle$ (estados no buscados) no generan (no se encuentra en su plano) el estado de superposición $|s\rangle$ sobre todos los qubits.

Si no modificamos el oráculo ni el amplificador de amplitud, obtenemos que debemos realizar el algoritmo un número de iteraciones $O(\sqrt{2^n})$ en vez de $O(\sqrt{N}); N < 2^n$, lo que no es una solución eficiente.

Si hallamos el vector superposición $|s_s\rangle$ solo para los elementos del subgrupo donde queremos buscar, podremos generar un amplificador de amplitud de la forma $\langle U_{as}| = |s_s\rangle \langle s_s| - I$ y usamos como estado de entrada al algoritmo el estado $|s_s\rangle$ tendremos una complejidad para este algoritmo de $O(\sqrt{N})$.
La nueva amplificación de amplitud $\langle U_{as}|$ y el estado $|s_s\rangle$ se pueden alcanzar gracias al algoritmo QCMD.

A.5. Matrices relativas al algoritmo de Grover

En esta sección vamos a ver las matrices relativas a las transformaciones relativas al algoritmo. Estas matrices pueden generarse mediante circuitos descritos en el capítulo **??**.

A.5.1. Oráculo

La matriz oráculo de un algoritmo de Grover tendrá una forma como A.19 donde $a_i = -1$ si en la posición i se encuentra un estado a buscar, o $a_i = 1$ en otro caso.

$$\begin{bmatrix} a_0 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & a_{n-1} \end{bmatrix} \tag{A.19}$$

A.5.2. Amplificación de amplitud

Esta matriz se genera con la multiplicación de los estados $|s\rangle \langle s|$, que genera una matriz con todos sus valores $\frac{1}{\sqrt{N}}$. Y a esta matriz se le resta la matriz identidad, como se ve en la ecuación A.20.

$$\begin{bmatrix} \frac{1}{\sqrt{N}} & \cdots & \frac{1}{\sqrt{N}} \\ \vdots & \ddots & \vdots \\ \frac{1}{\sqrt{N}} & \cdots & \frac{1}{\sqrt{N}} \end{bmatrix} - \begin{bmatrix} 1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & 1 \end{bmatrix} = \begin{bmatrix} \frac{1}{\sqrt{N}} - 1 & \cdots & \frac{1}{\sqrt{N}} \\ \vdots & \ddots & \vdots \\ \frac{1}{\sqrt{N}} & \cdots & \frac{1}{\sqrt{N}} - 1 \end{bmatrix} \tag{A.20}$$

A.5.3. Variante de amplificación de amplitud

Para trabajar sobre un subgrupo de elementos, debemos modificar la matriz de amplificación de estado, de forma que aquellos estados con los que no trabajemos queden a 0 sus filas y columnas salvo el elemento de la diagonal. La diagonal solución vendría dada por la matriz mostrada en A.21.

Para conseguir el estado de superposición sobre solo algunos elementos también se puede usar el algoritmo QCMD para generar un circuito previo a la entrada del algoritmo.

$$\begin{bmatrix} \frac{1}{\sqrt{N}} & \cdots & 0 & \cdots & \frac{1}{\sqrt{N}} \\ \vdots & \ddots & 0 & \ddots & \vdots \\ 0 & 0 & 0 & 0 & 0 \\ \vdots & \ddots & 0 & \ddots & \vdots \\ \frac{1}{\sqrt{N}} & \cdots & 0 & \cdots & \frac{1}{\sqrt{N}} \end{bmatrix} - \begin{bmatrix} 1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & 1 \end{bmatrix} = \begin{bmatrix} \frac{1}{\sqrt{N}} - 1 & \cdots & 0 & \cdots & \frac{1}{\sqrt{N}} \\ \vdots & \ddots & 0 & \ddots & \vdots \\ 0 & 0 & -1 & 0 & 0 \\ \vdots & \ddots & 0 & \ddots & \vdots \\ \frac{1}{\sqrt{N}} & \cdots & 0 & \cdots & \frac{1}{\sqrt{N}} - 1 \end{bmatrix} \tag{A.21}$$

B

Giros básicos

En este anexo se recogen las demostraciones relativas a los giros básicos sobre la *esfera de Bloch*. La computación cuántica aún no está estandarizada, por lo que los giros elegidos en este trabajo no tienen porqué ser los existentes a nivel hardware en los distintos ordenadores cuánticos.
Se ha elegido trabajar sobre estos giros debido a que son los más simples y más fáciles para calcular con ellos.

B.1. Demostración de matrices de giro

En esta sección se demuestra que las matrices tomadas para estos giros son, en verdad, giros sobre la esfera de bloch. En la tabla 3.II se muestran las matrices de los giros utilizados en el trabajo, que representan giros sobre cada uno de los ejes de la esfera.

X(α)	Y(β)	Z(γ)
$\begin{bmatrix} cos(\frac{\alpha}{2}) & sin(\frac{\alpha}{2})i \\ sin(\frac{\alpha}{2})i & cos(\frac{\alpha}{2}) \end{bmatrix}$	$\begin{bmatrix} cos(\frac{\beta}{2}) & -sin(\frac{\beta}{2}) \\ sin(\frac{\beta}{2}) & cos(\frac{\beta}{2}) \end{bmatrix}$	$\begin{bmatrix} 1 & 0 \\ 0 & e^{i\gamma} \end{bmatrix}$

Tabla B.I: Representación matricial de los giros básicos.

Giro Z

Puntos fijos En las ecuaciones B.1 y B.2 se ve demostrado cómo un giro Z no modifica los estados $|0\rangle$ y $|1\rangle$, que se encuentran en los puntos donde el eje Z corta la esfera.

$$\langle Z(\alpha)|0\rangle = \begin{bmatrix} 1 & 0 \\ 0 & \epsilon(\alpha) \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \end{bmatrix} = |0\rangle \tag{B.1}$$

$$\langle Z(\alpha)|1\rangle = \begin{bmatrix} 1 & 0 \\ 0 & \epsilon(\alpha) \end{bmatrix} \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ \epsilon(\alpha) \end{bmatrix} = \begin{bmatrix} \epsilon(-\alpha) & 0 \\ 0 & \epsilon(-\alpha) \end{bmatrix} \begin{bmatrix} 0 \\ \epsilon(\alpha) \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \end{bmatrix} = |1\rangle \tag{B.2}$$

Giros ejemplo En las ecuaciones B.3 y B.4 se muestran los estados a los que se modifican el estado estándar $|+\rangle$. Si se contrasta en una esfera de Bloch se ve que el resultado es el esperado para un giro en el eje Z.

$$\langle Z(\pi)|+\rangle = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \end{bmatrix} \frac{1}{\sqrt{2}} = \begin{bmatrix} 1 \\ -1 \end{bmatrix} \frac{1}{\sqrt{2}} = |-\rangle \tag{B.3}$$

$$\langle Z(\pi/2)|+\rangle = \begin{bmatrix} 1 & 0 \\ 0 & i \end{bmatrix} \begin{bmatrix} 1 \\ 1 \end{bmatrix} \frac{1}{\sqrt{2}} = \begin{bmatrix} 1 \\ i \end{bmatrix} \frac{1}{\sqrt{2}} = |i\rangle \tag{B.4}$$

Demostración general En la ecuación B.5 se demuestra de forma generalizada que esta matriz representa un giro en el eje Z.

$$\left\langle Z(\alpha) \middle| \begin{pmatrix} cos(\varphi/2) \\ sin(\varphi/2)\epsilon(\theta) \end{pmatrix} \right\rangle = \begin{bmatrix} 1 & 0 \\ 0 & \epsilon(\alpha) \end{bmatrix} \begin{bmatrix} cos(\varphi/2) \\ sin(\varphi/2)\epsilon(\theta) \end{bmatrix} = \begin{bmatrix} cos(\varphi/2) \\ sin(\varphi/2)\epsilon(\theta+\alpha) \end{bmatrix} \tag{B.5}$$

Giro X

Puntos fijos En las ecuaciones B.6 y B.7 se ve demostrado cómo un giro X no modifica los estados $|+\rangle$ y $|-\rangle$, que se encuentran en los puntos donde el eje X corta la esfera.

$$\langle X(\alpha)|+\rangle = \begin{bmatrix} cos(\alpha/2) & sin(\alpha/2)i \\ sin(\alpha/2)i & cos(\alpha/2) \end{bmatrix} \begin{bmatrix} 1 \\ 1 \end{bmatrix} \frac{1}{\sqrt{2}} = \begin{bmatrix} cos(\alpha/2)+sin(\alpha/2)i \\ cos(\alpha/2)+sin(\alpha/2)i \end{bmatrix} \frac{1}{\sqrt{2}} = \begin{bmatrix} \epsilon(\alpha/2) \\ \epsilon(\alpha/2) \end{bmatrix} \frac{1}{\sqrt{2}} = |+\rangle \tag{B.6}$$

$$\langle X(\alpha)|-\rangle = \begin{bmatrix} cos(\alpha/2) & sin(\alpha/2)i \\ sin(\alpha/2)i & cos(\alpha/2) \end{bmatrix} \begin{bmatrix} 1 \\ -1 \end{bmatrix} \frac{1}{\sqrt{2}} = \begin{bmatrix} cos(\alpha/2)-sin(\alpha/2)i \\ cos(\alpha/2)-sin(\alpha/2)i \end{bmatrix} \frac{1}{\sqrt{2}} = \begin{bmatrix} \epsilon(-\alpha/2) \\ \epsilon(-\alpha/2) \end{bmatrix} \frac{1}{\sqrt{2}} = |-\rangle \tag{B.7}$$

Giros ejemplo En las ecuaciones B.8 y B.9 se muestran los estados a los que se modifican el estado estándar $|0\rangle$. Si se contrasta en una esfera de Bloch se ve que el resultado es el esperado para un giro en el eje X.

$$\langle X(\pi)|0\rangle = \begin{bmatrix} cos(\pi/2) & sin(\pi/2)i \\ sin(\pi/2)i & cos(\pi/2) \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} cos(\pi/2) \\ sin(\pi/2)i \end{bmatrix} = \begin{bmatrix} 0 \\ i \end{bmatrix} = |1\rangle \tag{B.8}$$

$$\langle X(\pi/2)|0\rangle = \begin{bmatrix} cos(\pi/4) & sin(\pi/4)i \\ sin(\pi/4)i & cos(\pi/4) \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} cos(\pi/4) \\ sin(\pi/4)i \end{bmatrix} = \begin{bmatrix} 1 \\ i \end{bmatrix} \frac{1}{\sqrt{2}} = |i\rangle \tag{B.9}$$

Demostración general En la ecuación B.10 se demuestra representa un giro en el eje X para cualquier estado en el plano YZ (para cualquier otro estado, la modificación de los ángulos no sigue una combinación lineal o patrón razonable).

$$\left\langle X(\alpha) \middle| \begin{pmatrix} cos(\varphi/2) \\ sin(\varphi/2)i \end{pmatrix} \right\rangle = \begin{bmatrix} cos(\alpha/2) & sin(\alpha/2)i \\ sin(\alpha/2)i & cos(\alpha/2) \end{bmatrix} \begin{bmatrix} cos(\varphi/2) \\ sin(\varphi/2)i \end{bmatrix} =$$
$$= \begin{bmatrix} cos(\alpha/2)cos(\varphi/2) - sin(\alpha/2)sin(\varphi/2) \\ sin(\alpha/2)cos(\varphi/2)i + cos(\alpha/2)sin(\varphi/2)i \end{bmatrix} = \begin{bmatrix} cos((\alpha+\varphi)/2) \\ cos((\alpha+\varphi)/2)i \end{bmatrix} = \tag{B.10}$$

Giro Y

Puntos fijos En las ecuaciones B.11 y B.12 se ve demostrado cómo un giro Y no modifica los estados $|i\rangle$ y $|j\rangle$, que se encuentran en los puntos donde el eje Y corta la esfera.

$$\langle Y(\alpha)|i\rangle = \begin{bmatrix} cos(\alpha/2) & -sin(\alpha/2) \\ sin(\alpha/2) & cos(\alpha/2) \end{bmatrix} \begin{bmatrix} 1 \\ i \end{bmatrix} \frac{1}{\sqrt{2}} = \begin{bmatrix} cos(\alpha/2) - sin(\alpha/2)i \\ sin(\alpha/2) + cos(\alpha/2)i \end{bmatrix} \frac{1}{\sqrt{2}} = \begin{bmatrix} \epsilon(-\alpha/2) \\ i\epsilon(-\alpha/2) \end{bmatrix} \frac{1}{\sqrt{2}} = |i\rangle \tag{B.11}$$

$$\langle Y(\alpha)|j\rangle = \begin{bmatrix} cos(\alpha/2) & -sin(\alpha/2) \\ sin(\alpha/2) & cos(\alpha/2) \end{bmatrix} \begin{bmatrix} 1 \\ -i \end{bmatrix} \frac{1}{\sqrt{2}} = \begin{bmatrix} cos(\alpha/2) + sin(\alpha/2)i \\ sin(\alpha/2) - cos(\alpha/2)i \end{bmatrix} \frac{1}{\sqrt{2}} = \begin{bmatrix} \epsilon(\alpha/2) \\ -i\epsilon(\alpha/2) \end{bmatrix} \frac{1}{\sqrt{2}} = |j\rangle \tag{B.12}$$

Giros ejemplo En las ecuaciones B.13 y B.14 se muestran los estados a los que se modifican el estado estándar $|0\rangle$. Si se contrasta en una esfera de Bloch se ve que el resultado es el esperado para un giro en el eje Y.

$$\langle Y(\pi)|0\rangle = \begin{bmatrix} cos(\pi/2) & -sin(\pi/2) \\ sin(\pi/2) & cos(\pi/2) \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} cos(\pi/2) \\ sin(\pi/2) \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \end{bmatrix} = |1\rangle \tag{B.13}$$

$$\langle Y(\pi/2)|0\rangle = \begin{bmatrix} cos(\pi/4) & -sin(\pi/4) \\ sin(\pi/4) & cos(\pi/4) \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} cos(\pi/4) \\ sin(\pi/4) \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \end{bmatrix} \frac{1}{\sqrt{2}} = |+\rangle \tag{B.14}$$

Demostración general En la ecuación B.15 se demuestra representa un giro en el eje Y para cualquier estado en el plano XZ (para cualquier otro estado, la modificación de los ángulos no sigue una combinación lineal o patrón razonable).

$$\left\langle Y(\alpha) \middle| \begin{pmatrix} cos(\varphi/2) \\ sin(\varphi/2) \end{pmatrix} \right\rangle = \begin{bmatrix} cos(\alpha/2) & -sin(\alpha/2) \\ sin(\alpha/2) & cos(\alpha/2) \end{bmatrix} \begin{bmatrix} cos(\varphi/2) \\ sin(\varphi/2) \end{bmatrix} = \\ = \begin{bmatrix} cos(\alpha/2)cos(\varphi/2) - sin(\alpha/2)sin(\varphi/2) \\ sin(\alpha/2)cos(\varphi/2) + cos(\alpha/2)sin(\varphi/2) \end{bmatrix} = \begin{bmatrix} cos((\alpha+\varphi)/2) \\ sin((\alpha+\varphi)/2) \end{bmatrix} \tag{B.15}$$

B.2. Demostración de sobreyectividad

En esta sección vamos a demostrar que el uso de estos giros generan cualquier transformación de giro sobre una esfera de Bloch.

Sobreyectividad sobre un estado En la ecuación B.17 se demuestra como, para $\forall$ estados $|x\rangle, |y\rangle$, $\exists(Z(\alpha), Y(\beta)$ tal que $\langle Z(\alpha)Y(\beta)|x\rangle = |y\rangle$. En este caso se han usado los giros Z e Y pero la demostración se puede realizar también para cualquier concatenación.
El cálculo de β y ω se resuelve con complejas ecuaciones trigonométricas de grado 2 que se deja para el lector.

$$|x\rangle = \begin{pmatrix} cos(\varphi/2) \\ sin(\varphi/2)\epsilon(\theta) \end{pmatrix}; |y\rangle = \begin{pmatrix} cos(\phi/2) \\ sin(\phi/2)\epsilon(\gamma) \end{pmatrix} \tag{B.16}$$

$$\left\langle Y(\beta) \middle| \begin{pmatrix} cos(\varphi/2) \\ sin(\varphi/2)\epsilon(\theta) \end{pmatrix} \right\rangle = \begin{bmatrix} cos(\phi/2) \\ sin(\phi/2)\epsilon(\omega) \end{bmatrix}$$
$$\left\langle Z(\alpha) \middle| \begin{pmatrix} cos(\phi/2) \\ sin(\phi/2)\epsilon(\omega) \end{pmatrix} \right\rangle = \begin{bmatrix} cos(\phi/2) \\ sin(\phi/2)\epsilon(\gamma) \end{bmatrix} \tag{B.17}$$

Sobreyectividad sobre un giro La ecuación B.18 demuestra que para $\forall U$ transformación de la esfera de Bloch, $\exists A, B, C$ tal que $ABC = U$. Para esta demostración se usa la matriz de giro B.19 que representa cualquie posible giro de la esfera de Bloch[22].

$$\langle Z(y)Y(x)Z(z)| = \begin{bmatrix} 1 & 0 \\ 0 & \epsilon(z) \end{bmatrix} \begin{bmatrix} cos(x/2) & -sin(x/2) \\ sin(x/2) & cos(x/2) \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & \epsilon(y) \end{bmatrix} =$$
$$= \begin{bmatrix} cos(x/2) & -sin(x/2)\epsilon(y) \\ sin(x/2)\epsilon(z) & cos(x/2)\epsilon(y+z) \end{bmatrix} \tag{B.18}$$

$$\begin{bmatrix} cos(x/2) & -sin(x/2)\epsilon(y) \\ sin(x/2)\epsilon(z) & cos(x/2)\epsilon(y+z) \end{bmatrix} \tag{B.19}$$

B.3. Demostración de suma de ángulos con concatenación de giros

En esta sección se demuestra como la concatenación de dos giros sobre el mismo eje resultan en un mismo giro cuyo ángulo es la suma de los ángulos anteriores.

$$\langle Z(\alpha)Z(\beta)| = \begin{bmatrix} 1 & 0 \\ 0 & \epsilon(\alpha) \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & \epsilon(\beta) \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & \epsilon(\alpha+\beta) \end{bmatrix} \tag{B.20}$$

$$\langle X(\alpha)X(\beta)| = \begin{bmatrix} cos(\alpha/2) & sin(\alpha/2)i \\ sin(\alpha/2)i & cos(\alpha/2) \end{bmatrix} \begin{bmatrix} cos(\beta/2) & sin(\beta/2)i \\ sin(\beta/2)i & cos(\beta/2) \end{bmatrix} =$$
$$= \begin{bmatrix} cos(\alpha/2)cos(\beta/2) - sin(\alpha/2)sin(\beta/2) & cos(\alpha/2)sin(\beta/2)i + sin(\alpha/2)cos(\beta/2)i \\ sin(\alpha/2)cos(\beta/2)i + cos(\alpha/2sin(\beta/2))i & cos(\alpha/2)cos(\beta/2) - sin(\alpha/2)sin(\beta/2) \end{bmatrix} =$$
$$= \begin{bmatrix} cos((\alpha+\beta)/2) & sin((\alpha+\beta)/2)i \\ sin((\alpha+\beta)/2)i & cos((\alpha+\beta)/2) \end{bmatrix} \tag{B.21}$$

$$\langle Y(\alpha)Y(\beta)| = \begin{bmatrix} cos(\alpha/2) & -sin(\alpha/2) \\ sin(\alpha/2) & cos(\alpha/2) \end{bmatrix} \begin{bmatrix} cos(\beta/2) & -sin(\beta/2) \\ sin(\beta/2) & cos(\beta/2) \end{bmatrix} =$$

$$= \begin{bmatrix} cos(\alpha/2)cos(\beta/2) - sin(\alpha/2)sin(\beta/2) & -cos(\alpha/2)sin(\beta/2) - sin(\alpha/2)cos(\beta/2) \\ sin(\alpha/2)cos(\beta/2) + cos(\alpha/2sin(\beta/2)) & cos(\alpha/2)cos(\beta/2) - sin(\alpha/2)sin(\beta/2) \end{bmatrix} =$$

$$= \begin{bmatrix} cos((\alpha+\beta)/2) & -sin((\alpha+\beta)/2) \\ sin((\alpha+\beta)/2) & cos((\alpha+\beta)/2) \end{bmatrix} \quad \text{(B.22)}$$

C

Demostraciones matemáticas del diseño de QCMD

En este anexo se recogen las demostraciones relativas a los pasos de creación de las nuevas puertas/circuitos. Se demuestran tanto la matriz solución de cada puerta, como la puerta inversión, y en algunos casos se demuestra explícitamente la complejidad. Se ha prescindido de las demostraciones triviales.

C.1. Giro condicionado

Demostramos que los diseños de cada una de las puertas propuestas generan la matriz deseada, y la demostración de validez de las puertas inversas a cada giro.

Giro Z

Diseño Con respecto a la figura 4.3c demostramos la matriz solución.

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \epsilon(\alpha/2) & 0 \\ 0 & 0 & 0 & \epsilon(\alpha/2) \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \epsilon(\alpha/2) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & \epsilon(\alpha/2) \end{bmatrix} *$$

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \epsilon(-\alpha/2) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & \epsilon(-\alpha/2) \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} =$$

$$= \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \epsilon(\alpha/2) & 0 & 0 \\ 0 & 0 & \epsilon(\alpha/2) & 0 \\ 0 & 0 & 0 & \epsilon(\alpha) \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \epsilon(-\alpha/2) & 0 & 0 \\ 0 & 0 & \epsilon(-\alpha/2) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} =$$

$$= \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & \epsilon(\alpha) \end{bmatrix} \quad \text{(C.1)}$$

Inversa La demostración de la inversa de esta puerta es trivial. $Z(\alpha)Z(-\alpha) = I$

Giro Y Con respecto a la figura 4.3b demostramos la matriz solución.

Diseño

$$\begin{bmatrix} cos(\alpha/2) & -sin(\alpha/2) & 0 & 0 \\ sin(\alpha/2) & cos(\alpha/2) & 0 & 0 \\ 0 & 0 & cos(\alpha/2) & -sin(\alpha/2) \\ 0 & 0 & sin(\alpha/2) & cos(\alpha/2) \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} *$$

$$\begin{bmatrix} cos(-\alpha/2) & -sin(-\alpha/2) & 0 & 0 \\ sin(-\alpha/2) & cos(-\alpha/2) & 0 & 0 \\ 0 & 0 & cos(-\alpha/2) & -sin(-\alpha/2) \\ 0 & 0 & sin(-\alpha/2) & cos(-\alpha/2) \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} =$$

$$= \begin{bmatrix} cos(\alpha/2) & -sin(\alpha/2) & 0 & 0 \\ sin(\alpha/2) & cos(\alpha/2) & 0 & 0 \\ 0 & 0 & cos(\alpha/2) & -sin(\alpha/2) \\ 0 & 0 & sin(\alpha/2) & cos(\alpha/2) \end{bmatrix} *$$

$$\begin{bmatrix} cos(\alpha/2) & sin(\alpha/2) & 0 & 0 \\ -sin(\alpha/2) & cos(\alpha/2) & 0 & 0 \\ 0 & 0 & cos(\alpha/2) & -sin(\alpha/2) \\ 0 & 0 & sin(\alpha/2) & cos(\alpha/2) \end{bmatrix} =$$

$$= \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & cos(\alpha) & -sin(\alpha) \\ 0 & 0 & sin(\alpha) & cos(\alpha) \end{bmatrix} \quad \text{(C.2)}$$

Inversa

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & cos(\alpha) & -sin(\alpha) \\ 0 & 0 & sin(\alpha) & cos(\alpha) \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & cos(-\alpha) & -sin(-\alpha) \\ 0 & 0 & sin(-\alpha) & cos(-\alpha) \end{bmatrix} =$$

$$= \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & cos(\alpha) & -sin(\alpha) \\ 0 & 0 & sin(\alpha) & cos(\alpha) \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & cos(\alpha) & sin(\alpha) \\ 0 & 0 & -sin(\alpha) & cos(\alpha) \end{bmatrix} =$$

$$= \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & cos(\alpha)^2 + sin(\alpha)^2 & cos(\alpha)sin(\alpha) - cos(\alpha)sin(\alpha) \\ 0 & 0 & cos(\alpha)sin(\alpha) - cos(\alpha)sin(\alpha) & cos(\alpha)^2 + sin(\alpha)^2 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \text{(C.3)}$$

Giro X Con respecto a la última figura de 4.3a demostramos la matriz solución.

Diseño

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & \epsilon(\pi/2) \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & cos(\alpha) & -sin(\alpha) \\ 0 & 0 & sin(\alpha) & cos(\alpha) \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & \epsilon(-\pi/2) \end{bmatrix} =$$

$$= \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & cos(\alpha) & -sin(\alpha) \\ 0 & 0 & sin(\alpha)i & cos(\alpha)i \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -i \end{bmatrix} =$$

$$= \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & cos(\alpha) & sin(\alpha)i \\ 0 & 0 & sin(\alpha)i & cos(\alpha) \end{bmatrix} \quad \text{(C.4)}$$

Inversa

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & cos(\alpha) & sin(\alpha)i \\ 0 & 0 & sin(\alpha)i & cos(\alpha) \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & cos(-\alpha) & -sin(-\alpha)i \\ 0 & 0 & sin(-\alpha)i & cos(-\alpha) \end{bmatrix} =$$

$$= \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & cos(\alpha) & sin(\alpha)i \\ 0 & 0 & sin(\alpha)i & cos(\alpha) \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & cos(\alpha) & -sin(\alpha)i \\ 0 & 0 & -sin(\alpha)i & cos(\alpha) \end{bmatrix} =$$

$$= \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & cos(\alpha)^2 - sin(\alpha)^2 i^2 & cos(\alpha)sin(\alpha)i - cos(\alpha)sin(\alpha)i \\ 0 & 0 & cos(\alpha)sin(\alpha)i - cos(\alpha)sin(\alpha)i & cos(\alpha)^2 - sin(\alpha)^2 i^2 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \text{(C.5)}$$

C.2. Giro condicionado múltiple

Giro de 2 *sources* En la tabla C.I se ve una tabla con el resultado del circuito de la figura 4.7 para cada uno de los inputs *lógicos*.

qubit	$\|00\rangle$	$\|01\rangle$	$\|10\rangle$	$\|11\rangle$
$q0$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$
$q1$	$\varnothing$	$\varnothing$	$XX = \varnothing$	$XX = \varnothing$
$q2 = target$	$\varnothing$	$U(\alpha/2)U(-\alpha/2) = \varnothing$	$U(-\alpha/2)U(\alpha/2) = \varnothing$	$U(\alpha/2)U(\alpha/2) = U(\alpha)$

Tabla C.I: Tabla de verdad para giro condicionado múltiple de 2 qubits

Giro de 3 *sources* En la tabla C.I se ve una tabla con el resultado del circuito de la figura 4.8 para cada uno de los inputs *lógicos*.

qubit	$\|000\rangle$	$\|001\rangle$	$\|010\rangle$	$\|011\rangle$
$q0$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$
$q1$	$XX = \varnothing$	$\varnothing$	$XX = \varnothing$	$\varnothing$
$q2$	$XX = \varnothing$	$XX = \varnothing$	$XX = \varnothing$	$XX = \varnothing$
$q3 = target$	$\varnothing$	$\varnothing$	$\varnothing$	$U(\alpha/2)U(-\alpha/2) = \varnothing$
qubit	$\|100\rangle$	$\|101\rangle$	$\|110\rangle$	$\|111\rangle$
$q0$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$
$q1$	$XXXX = \varnothing$	$XX = \varnothing$	$XXXX = \varnothing$	$XX = \varnothing$
$q2$	$XXXX = \varnothing$	$XXXX = \varnothing$	$XXXX = \varnothing$	$XXXX = \varnothing$
$q3 = target$	$U(-\alpha/2)U(\alpha/2) = \varnothing$	$\varnothing$	$\varnothing$	$U(\alpha/2)U(\alpha/2) = U(\alpha)$

Tabla C.II: Tabla de verdad para giro condicionado múltiple de 3 qubits

Giro de n *sources* Por inducción se puede demostrar que cualquier puerta condicionada múltiple con más de 3 *sources*, en verdad genera el giro esperado.
Esto se puede demostrar dividiendo el diseño de esta puerta en 3 apartados, uno por cada puerta de menor número de *sources*.
En el primer apartado, el giro solo se activará si $\forall i = 2...n\ q_i = 1$.
En el segundo apartado, solo se activará si $q_1 = 0$ y $\forall i = 2...n\ q_i = 1$, lo que hará que se deshaga el giro anterior; o si $q_1 = 1$ y $\forall i = 2...n\ q_i = 0$.
En el tercer apartado, la puerta solo se activará si $q_n = 0$ y $\forall i = 2...n-1\ q_i = 0$ y $q_1 = 1$ lo que desharía el giro del apartado 2 en el caso de que todos los qubits esten no activados; o si $\forall i = 1...n\ q_i = 1$ lo que generaría un giro, que sumado al del apartado 1 sumarían el giro solución.

Complejidad En la ecuación C.11 se demuestra la complejidad total de la puerta de giro múltiple dependiendo del número del número de qubits *source*.

$$P_m(s) = 2 + 2(s-1) + 2(s-2) + 3(P_m(s-1)) = 4s - 4 + 3(P_m(s-1)) \tag{C.6}$$

$$P_m(1) = 5 \tag{C.7}$$

$$P_m(2) = 2 + 3(P_m(1)) = 17 \tag{C.8}$$

$$\sum_{i=0}^{n} 3^i = \frac{1}{2}(3^{n+1} - 1) \tag{C.9}$$

$$\sum_{i=0}^{n} i3^i = \frac{3}{4}(2n3^n - 3^n + 1) \tag{C.10}$$

$$\begin{aligned} P_m(s) = 4 + 2(s-1) + 2(s-2) + 3(P_m(s-1)) = \\ = 4(s-1) + 3 \cdot 4(s-2) + 3^2 P_m(s-2) = \\ = 4\sum_{i=0}^{n} 3^i s - 1 - i = \\ = 2s3^{s-1} - 2s - 2 \cdot 3^{s-2} + 2 - 2s3^{s-2} + 6 \cdot 3^{s-2} + 3^{s-2} - 3 + 3^{s-2} P_m(2) = \\ = 22 \cdot 3^{s-2} - 2s - 1 = \\ = O(3^s) \end{aligned} \tag{C.11}$$

Diseño 2 Demostramos en C.12 la validez de la puerta según el diseño de la figura 4.11 para n qubits *source* y $n-1$ qubits auxiliares.

$$\begin{aligned} aux_1 = s_1 AND s_2; aux_2 = aux_1 AND s_3; aux_i = aux_{i-1} AND s_{i+1} \forall i = 2, 3.., n-1 \Rightarrow \\ \Rightarrow aux_{n-1} = AND_{i=1}^{n}(s_i) \end{aligned} \tag{C.12}$$

De esta forma podemos saber que la puerta con *target* aux_{n-1} solo se activará cuando todos los *sources* estén activados.

C.3. Toffoli

Demostración del diseño Por la ecuación C.13 demostramos la validez del diseño de la puerta Toffoli vista en la figura 4.12b.

$$\begin{bmatrix} 1&0&0&0&0&0&0&0\\ 0&1&0&0&0&0&0&0\\ 0&0&1&0&0&0&0&0\\ 0&0&0&1&0&0&0&0\\ 0&0&0&0&1&0&0&0\\ 0&0&0&0&0&1&0&0\\ 0&0&0&0&0&0&cos(\pi/2)&-sin(\pi/2)\\ 0&0&0&0&0&0&sin(\pi/2)&cos(\pi/2) \end{bmatrix} \begin{bmatrix} 1&0&0&0&0&0&0&0\\ 0&1&0&0&0&0&0&0\\ 0&0&1&0&0&0&0&0\\ 0&0&0&1&0&0&0&0\\ 0&0&0&0&1&0&0&0\\ 0&0&0&0&0&1&0&0\\ 0&0&0&0&0&0&1&0\\ 0&0&0&0&0&0&0&\epsilon(\pi) \end{bmatrix} =$$

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

$$= \begin{bmatrix} 1&0&0&0&0&0&0&0\\ 0&1&0&0&0&0&0&0\\ 0&0&1&0&0&0&0&0\\ 0&0&0&1&0&0&0&0\\ 0&0&0&0&1&0&0&0\\ 0&0&0&0&0&1&0&0\\ 0&0&0&0&0&0&1&0\\ 0&0&0&0&0&0&0&1 \end{bmatrix} \quad \text{(C.13)}$$

Inversa Demostrar que la puerta Toffoli es su propia inversa es un cálculo trivial.

C.4. Puerta de giro especial

Demostración del diseño Demostramos en C.14 que mediante el diseño figura 4.13 llegamos a la matriz deseada (usando la puerta identidad condicionada).

$$\text{Para } x = fase(a),\ y = fase(b),\ \theta = sin^{-1}\left(\frac{|b|}{\sqrt{|a|^2+|b|^2}}\right) = cos^{-1}\left(\frac{|a|}{\sqrt{|a|^2+|b|^2}}\right)$$

$$\begin{bmatrix} \epsilon(-x)&0\\ 0&\epsilon(-x) \end{bmatrix} \begin{bmatrix} 1&0\\ 0&\epsilon(y+x) \end{bmatrix} \begin{bmatrix} cos(\theta)&-sin(\theta)\\ sin(\theta)&cos(\theta) \end{bmatrix} \begin{bmatrix} 1&0\\ 0&\epsilon(-y+x+\pi) \end{bmatrix} =$$

$$= \begin{bmatrix} \epsilon(-x)&0\\ 0&\epsilon(y) \end{bmatrix} \begin{bmatrix} cos(\theta)&sin(\theta)\epsilon(-y+x)\\ sin(\theta)&-cos(\theta)\epsilon(-y+x) \end{bmatrix} =$$

$$= \begin{bmatrix} cos(\theta)\epsilon(-x)&sin(\theta)\epsilon(-y)\\ sin(\theta)\epsilon(y)&-cos(\theta)\epsilon(x) \end{bmatrix} = \begin{bmatrix} a^*&b^*\\ b&-a \end{bmatrix} \frac{1}{\sqrt{|a|^2+|b|^2}} \quad \text{(C.14)}$$

Inversa Demostramos la validez de la inversión de esta puerta según la figura 4.5 en la ecuación C.15

$$\begin{bmatrix} a^* & b^* \\ b & -a \end{bmatrix} \begin{bmatrix} a & b^* \\ b & -a^* \end{bmatrix} \frac{1}{|a|^2+|b|^2} = \begin{bmatrix} |a|^2+|b|^2 & a^*b^*-a^*b^* \\ ab-ab & |b|^2+|a|^2 \end{bmatrix} \frac{1}{|a|^2+|b|^2} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \tag{C.15}$$

C.5. Puerta de intercambio

Se demuestra el efecto de los circuitos mostrados en la figura 4.16 y para cualquier puerta con una matriz como la mencionada en la sección 4.3, en las ecuación C.16.

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} a_{00} & a_{01} & a_{02} & a_{03} \\ a_{10} & a_{11} & a_{12} & a_{13} \\ a_{20} & a_{21} & a_{22} & a_{23} \\ a_{30} & a_{31} & a_{32} & a_{33} \end{bmatrix} = \begin{bmatrix} a_{00} & a_{01} & a_{02} & a_{03} \\ a_{30} & a_{31} & a_{32} & a_{33} \\ a_{20} & a_{21} & a_{22} & a_{23} \\ a_{10} & a_{11} & a_{12} & a_{13} \end{bmatrix}$$

$$\begin{bmatrix} a_{00} & a_{01} & a_{02} & a_{03} \\ a_{10} & a_{11} & a_{12} & a_{13} \\ a_{20} & a_{21} & a_{22} & a_{23} \\ a_{30} & a_{31} & a_{32} & a_{33} \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} = \begin{bmatrix} a_{00} & a_{03} & a_{02} & a_{01} \\ a_{10} & a_{13} & a_{12} & a_{11} \\ a_{20} & a_{23} & a_{22} & a_{21} \\ a_{30} & a_{33} & a_{32} & a_{31} \end{bmatrix}$$

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} a_{00} & a_{01} & a_{02} & a_{03} \\ a_{10} & a_{11} & a_{12} & a_{13} \\ a_{20} & a_{21} & a_{22} & a_{23} \\ a_{30} & a_{31} & a_{32} & a_{33} \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} = \begin{bmatrix} a_{00} & a_{03} & a_{02} & a_{01} \\ a_{30} & a_{33} & a_{32} & a_{31} \\ a_{20} & a_{23} & a_{22} & a_{21} \\ a_{10} & a_{13} & a_{12} & a_{11} \end{bmatrix}$$

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & a & b \\ 0 & 0 & c & d \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & d & c & 0 \\ 0 & b & a & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{C.16}$$

C.6. Puerta AMS

La matriz solución a esta puerta se consigue de forma trivial.

Inversa Demostramos por la ecuacion C.17 la validez de la puerta inversión usando la puerta inversión del giro especial.

$$\begin{aligned} R(f,3)R(c,2)S(a,b)R(c,2)R(f,3)R(f,3)R(c,2)S(a^*,b)R(c,2)R(f,3) = \\ = R(f,3)R(c,2)S(a,b)R(c,2)IR(c,2)S(a^*,b)R(c,2)R(f,3) = \\ = R(f,3)R(c,2)S(a,b)IS(a^*,b)R(c,2)R(f,3) = \\ = R(f,3)R(c,2)IR(c,2)R(f,3) = R(f,3)IR(f,3) = I \end{aligned} \tag{C.17}$$

C.7. Puerta diagonal

En las ecuaciones C.18, C.19 y C.20 se muestra la generación de cada paso de la puerta diagonal, y en la ecuación C.21 se muestra la matriz solución de la concatenación de estas.

$$\begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & \epsilon(\alpha) \end{bmatrix} \begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} = \begin{bmatrix} \epsilon(\alpha) & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{C.18}$$

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & \epsilon(\beta) \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \epsilon(\beta) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{C.19}$$

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & \epsilon(\gamma) \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \epsilon(\gamma) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{C.20}$$

$$\begin{bmatrix} \epsilon(\alpha) & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \epsilon(\beta) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \epsilon(\gamma) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & \epsilon(\delta) \end{bmatrix} =$$

$$= \begin{bmatrix} \epsilon(\alpha) & 0 & 0 & 0 \\ 0 & \epsilon(\beta) & 0 & 0 \\ 0 & 0 & \epsilon(\gamma) & 0 \\ 0 & 0 & 0 & \epsilon(\delta) \end{bmatrix} \tag{C.21}$$

C.8. Puerta matricial

Las ecuaciones de C.22 hasta C.28 representan la división de una matriz en submatrices unitarias de tamaño 2×2, generables mediante una puerta *AMS*. De esta forma, sabemos que concatenando en orden inverso cada una de las inversas de estas puertas obtendremos un circuito cuya matriz solución sea la matriz principal.

$$\begin{bmatrix} \frac{a_{00}^*}{\sqrt{|a_{00}|^2+|a_{10}|^2}} & \frac{a_{10}^*}{\sqrt{|a_{00}|^2+|a_{10}|^2}} & 0 & 0 \\ \frac{a_{10}}{\sqrt{|a_{00}|^2+|a_{10}|^2}} & \frac{-a_{00}}{\sqrt{|a_{00}|^2+|a_{10}|^2}} & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} a_{00} & a_{01} & a_{02} & a_{03} \\ a_{10} & a_{11} & a_{12} & a_{12} \\ a_{20} & a_{21} & a_{22} & a_{23} \\ a_{30} & a_{31} & a_{32} & a_{33} \end{bmatrix} = \begin{bmatrix} a'_{00} & a'_{01} & a'_{02} & a'_{03} \\ 0 & a'_{11} & a'_{12} & a'_{12} \\ a_{20} & a_{21} & a_{22} & a_{23} \\ a_{30} & a_{31} & a_{32} & a_{33} \end{bmatrix} \tag{C.22}$$

$$\begin{bmatrix} \frac{a'^*_{00}}{\sqrt{|a'_{00}|^2+|a_{20}|^2}} & 0 & \frac{a^*_{20}}{\sqrt{|a'_{00}|^2+|a_{20}|^2}} & 0 \\ 0 & 1 & 0 & 0 \\ \frac{a_{20}}{\sqrt{|a'_{00}|^2+|a_{20}|^2}} & 0 & \frac{-a'_{00}}{\sqrt{|a'_{00}|^2+|a_{20}|^2}} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} a'_{00} & a'_{01} & a'_{02} & a'_{03} \\ 0 & a'_{11} & a'_{12} & a'_{12} \\ a_{20} & a_{21} & a_{22} & a_{23} \\ a_{30} & a_{31} & a_{32} & a_{33} \end{bmatrix} = \begin{bmatrix} a''_{00} & a''_{01} & a''_{02} & a''_{03} \\ 0 & a'_{11} & a'_{12} & a'_{12} \\ 0 & a'_{21} & a'_{22} & a'_{23} \\ a_{30} & a_{31} & a_{32} & a_{33} \end{bmatrix} \tag{C.23}$$

$$\begin{bmatrix} \frac{a''^{*}_{00}}{\sqrt{|a''_{00}|^2+|a_{30}|^2}} & 0 & 0 & \frac{a^{*}_{30}}{\sqrt{|a''_{00}|^2+|a_{30}|^2}} \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ \frac{a_{30}}{\sqrt{|a''_{00}|^2+|a_{30}|^2}} & 0 & 0 & \frac{-a''_{00}}{\sqrt{|a''_{00}|^2+|a_{30}|^2}} \end{bmatrix} \begin{bmatrix} a''_{00} & a''_{01} & a''_{02} & a''_{03} \\ 0 & a'_{11} & a'_{12} & a'_{12} \\ 0 & a'_{21} & a'_{22} & a'_{23} \\ a_{30} & a_{31} & a_{32} & a_{33} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & a'_{11} & a'_{12} & a'_{12} \\ 0 & a'_{21} & a'_{22} & a'_{23} \\ 0 & a'_{31} & a'_{32} & a'_{33} \end{bmatrix} \tag{C.24}$$

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \frac{a'^{*}_{11}}{\sqrt{|a'_{11}|^2+|a'_{21}|^2}} & \frac{a'^{*}_{21}}{\sqrt{|a'_{11}|^2+|a'_{21}|^2}} & 0 \\ 0 & \frac{a'_{21}}{\sqrt{|a'_{11}|^2+|a'_{21}|^2}} & \frac{-a'_{11}}{\sqrt{|a'_{11}|^2+|a'_{21}|^2}} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & a'_{11} & a'_{12} & a'_{12} \\ 0 & a'_{21} & a'_{22} & a'_{23} \\ 0 & a'_{31} & a'_{32} & a'_{33} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & a''_{11} & a''_{12} & a''_{12} \\ 0 & 0 & a''_{22} & a''_{23} \\ 0 & a'_{31} & a'_{32} & a'_{33} \end{bmatrix} \tag{C.25}$$

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \frac{a''^{*}_{11}}{\sqrt{|a''_{11}|'^2+|a'_{31}|^2}} & 0 & \frac{a'^{*}_{31}}{\sqrt{|a''_{11}|^2+|a'_{21}|^2}} \\ 0 & 0 & 1 & 0 \\ 0 & \frac{a'_{31}}{\sqrt{|a''_{11}|^2+|a'_{31}|^2}} & 0 & \frac{-a''_{11}}{\sqrt{|a''_{11}|^2+|a'_{31}|^2}} \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & a''_{11} & a''_{12} & a''_{12} \\ 0 & 0 & a''_{22} & a''_{23} \\ 0 & a'_{31} & a'_{32} & a'_{33} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & a''_{22} & a''_{23} \\ 0 & 0 & a''_{32} & a''_{33} \end{bmatrix} \tag{C.26}$$

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \frac{a''^{*}_{22}}{\sqrt{|a''_{22}|'^2+|a''_{32}|^2}} & \frac{a''^{*}_{32}}{\sqrt{|a''_{22}|^2+|a''_{32}|^2}} \\ 0 & 0 & \frac{a''_{32}}{\sqrt{|a''_{22}|^2+|a''_{32}|^2}} & \frac{-a''_{22}}{\sqrt{|a''_{22}|^2+|a''_{32}|^2}} \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & a''_{22} & a''_{23} \\ 0 & 0 & a''_{32} & a''_{33} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & \epsilon(\alpha) \end{bmatrix} \tag{C.27}$$

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & \epsilon(\alpha) \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & \epsilon(-\alpha) \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{C.28}$$

C.9. Uso del diseño 2

Se demuestra en la ecuación C.31 que el uso del diseño 2 solo es eficiente a partir de un número de qubits, debido al alto coste de usar puertas Toffoli.

$$P_{m(d1)}(s) = 4(s-1) + 3P_m(s-1) \simeq 22 \cdot 3^{s-2} \tag{C.29}$$

$$P_{m(d2)}(s) = 2sP_t(2) = 4sP_{m(d1)}(2) \simeq 68s \tag{C.30}$$

$$P_{m(d2)}(s) < P_{m(d1)}(s) \Leftrightarrow 68s < 22 \cdot 3^{s-2} \Leftrightarrow s > 4{,}369 \tag{C.31}$$

Demostramos así que, sin un diseño más óptimo de una puerta Toffoli, por debajo de 5 qubits es más eficiente el diseño 1 para una puerta de giro condicionada múltiple, y por lo tanto, para cualquiera de las puertas superiores.

D

Representaciones de giros en esfera de Bloch

D.1. Representación de un giro H

En la figura D.1 se ve representado un giro H. Cada estado representado se transforma en el estado con su mismo color.

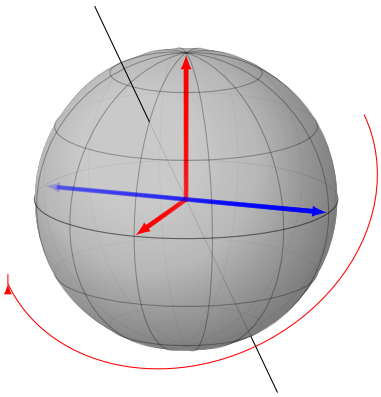


Figura D.1: Giro H.

D.2. Representación de un giro condicionado

D.2.1. Giro con *target* inactivo

Aquí se representa el caso de un giro condicionado sobre el eje Y de ángulo α en el caso en que el *source* no esté activo, por lo que se generará un giro en una dirección, y el mismo en dirección contraria. Se puede ver como esta concatenación deja todos los estados sin modificar.

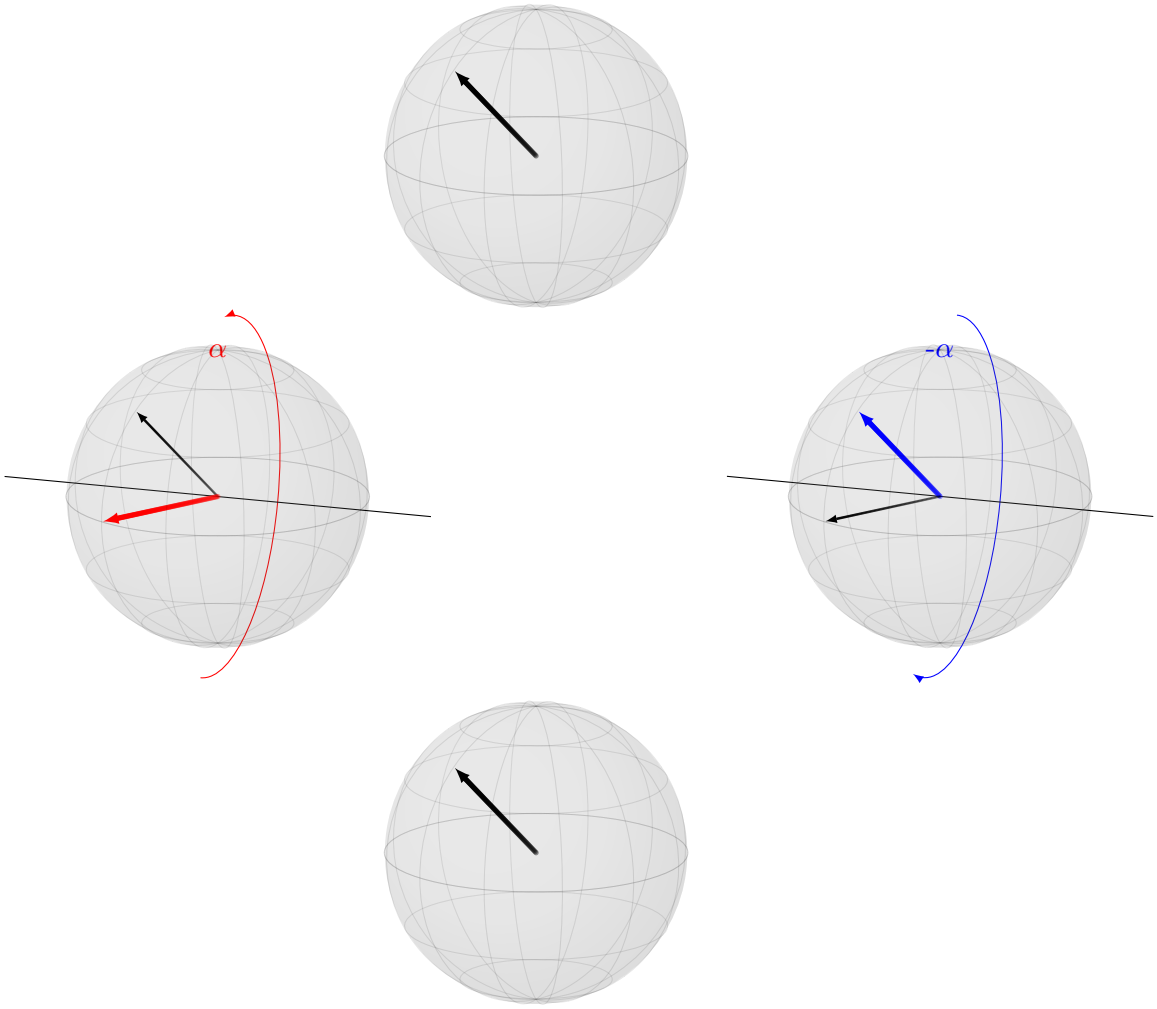


Figura D.2: Representación de concatenación de giros dos giros en el eje Y opuestos.

D.2.2. Giro con *target* activo

Aquí se representa el caso de un giro condicionado sobre el eje Y de ángulo α en el caso en que el *source* esté activo, por lo que se generará un giro α sobre Y, un giro de ángulo π sobre el eje Z, un giro sobre Y de ángulo $-\alpha$ y por último otro giro de π sobre Z. Se puede ver como el estado se ha modificado un ángulo α sobre el eje Y.

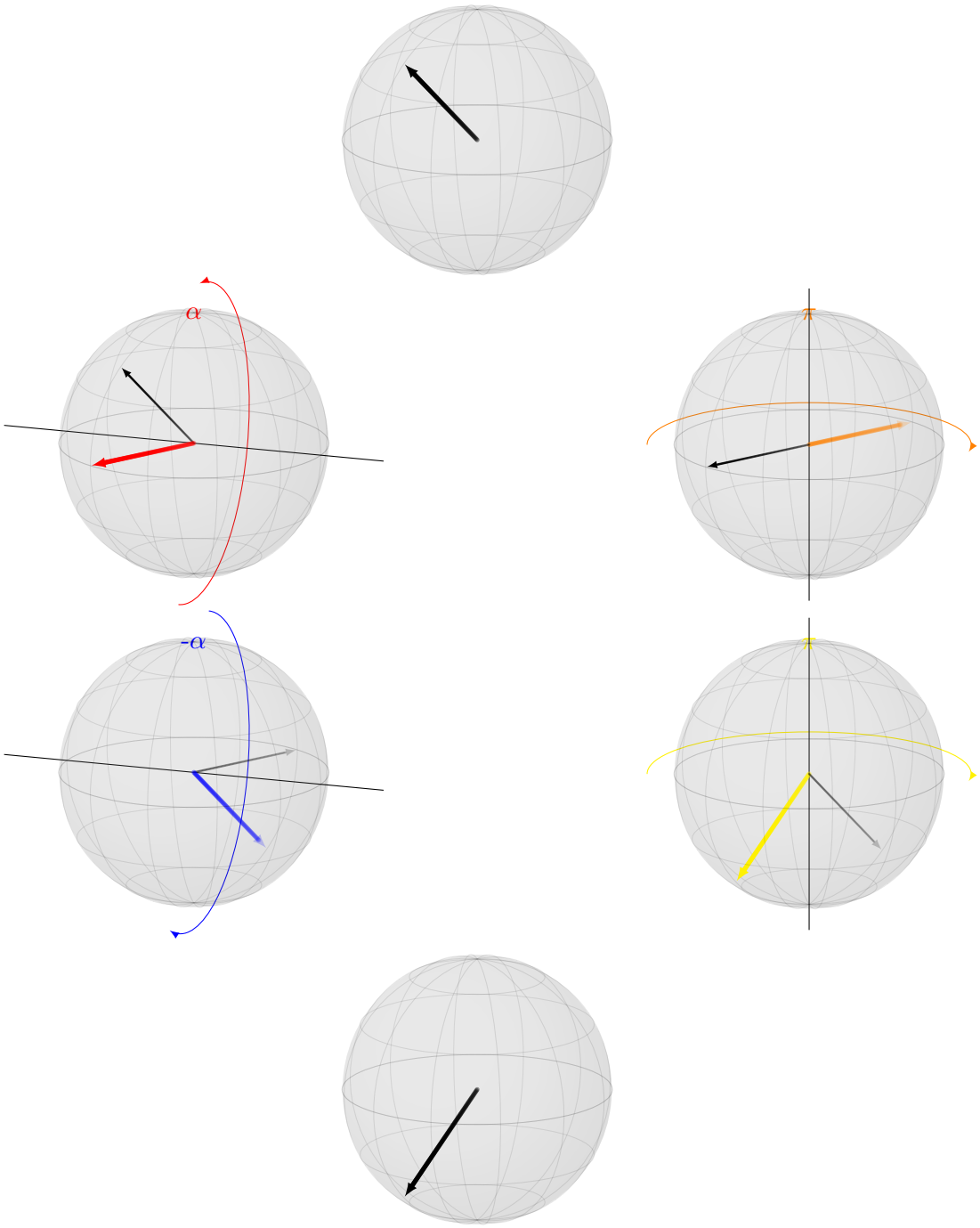


Figura D.3: Representación de concatenación de dos giros en el eje Y opuestos con giros Z intercalados.

E

Ejemplo de funcionamiento del algoritmo QCMD

En este anexo se explica el funcionamiento de esta puerta mediante un pequeño ejemplo de un circuito de 2 qubits. En la ecuación E.1 se ve un ejemplo de matriz que queremos construir M. La demostración de que dicha matriz es unitaria (construible por un ordenador cuántico) se puede ver en la ecuación E.2.

Al dividir esta matriz en submatrices, empezamos por el elemento más *arriba* a la *izquierda* de la matriz, que se ve en negrita en la figura E.1. La submatriz que queremos generar se muestra en la ecuación E.3, denominada V_0. Vemos en la ecuación E.4 como esta matriz *transforma* nuestra matriz destino en una matriz similar pero con un 0 en el valor que estabamos modificando.

Siguiendo con el mismo procedimiento, buscamos la siguiente matriz, que será la mostrada en la ecuación E.5, denominada V_1. Vemos en la ecuación E.6 como hemos llegado a una matriz identidad.

Por último vemos que queda un valor dentro de la diagonal que no se ha modificado. Para esto usamos la matriz V_2 E.7 y obtenemos la matriz identidad según la ecuación E.8.

De esta forma, sabemos que la concatenación de las matrices $V_0^* V_1^* V_2^*$ dará lugar a la matriz M. Esto se demuestra en la ecuación E.9.

Generando las puertas *AMS* con los parámetros necesarios y concatenándolas podemos crear un circuito que se ajuste a la matriz M. De esta forma $\langle U_0|$ será una puerta *AMS* con parámetros: fila=1, columna=0, a=$\frac{-i}{\sqrt{2}}$, b=$\frac{1}{\sqrt{2}}$; cuya matrix es $V_0^\dagger$. Y la puerta $\langle U_1|$ será una puerta *AMS* con parámetros: fila=1, columna=0, a=$\frac{\sqrt{2}}{\sqrt{3}}$, b=$\frac{1}{\sqrt{3}}$; cuya matrix es $V_1^\dagger$.

Pueden existir valores en la diagonal que no se hayan modificado por estas puertas, como por ejemplo el i de *abajo derecha* de M. Para esto se genera al final una matriz diagonal que transforma estos valores a una matriz identidad. En el caso de la matriz M, la única puerta necesaria para esta matriz diagonal será la referenete al último valor, que se puede hacer me-

diante un giro condicionado múltiple sobre Z de ángulo π, o la puerta $\langle U_2|$ con matriz $V_2^\dagger$.

Por lo tanto concluimos que el circuito $\langle U_0U_1U_2|$ mostrado en la figura E.1 ($a_0 = \frac{i}{\sqrt{2}}$, $b_0 = \frac{1}{\sqrt{2}}$, $a_1 = \frac{\sqrt{2}}{\sqrt{3}}$, $b_1 = \frac{1}{\sqrt{3}}$) generan la matriz M buscada. **Este proceso se puede generalizar a cualquier matriz unitaria**.

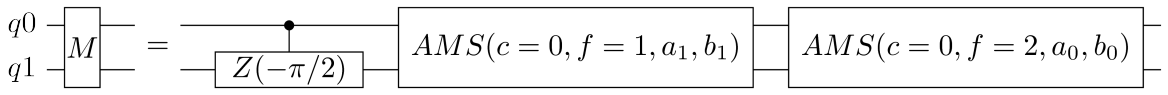


Figura E.1: Diseño de puerta de ajuste matricial para 2 qubits.

$$M = \begin{bmatrix} \frac{i}{\sqrt{3}} & \frac{1}{\sqrt{2}} & \frac{i}{\sqrt{6}} & 0 \\ \frac{\mathbf{1}}{\sqrt{\mathbf{3}}} & \frac{i}{\sqrt{2}} & \frac{1}{\sqrt{6}} & 0 \\ \frac{1}{\sqrt{3}} & 0 & \frac{-\sqrt{2}}{\sqrt{3}} & 0 \\ 0 & 0 & 0 & i \end{bmatrix} \tag{E.1}$$

$$MM^* = \begin{bmatrix} \frac{i}{\sqrt{3}} & \frac{1}{\sqrt{2}} & \frac{i}{\sqrt{6}} & 0 \\ \frac{\mathbf{1}}{\sqrt{\mathbf{3}}} & \frac{i}{\sqrt{2}} & \frac{1}{\sqrt{6}} & 0 \\ \frac{1}{\sqrt{3}} & 0 & \frac{-\sqrt{2}}{\sqrt{3}} & 0 \\ 0 & 0 & 0 & i \end{bmatrix} \begin{bmatrix} \frac{-i}{\sqrt{3}} & \frac{1}{\sqrt{3}} & \frac{1}{\sqrt{3}} & 0 \\ \frac{1}{\sqrt{2}} & \frac{-i}{\sqrt{2}} & 0 & 0 \\ \frac{-i}{\sqrt{6}} & \frac{1}{\sqrt{6}} & \frac{-\sqrt{2}}{\sqrt{3}} & 0 \\ 0 & 0 & 0 & i \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & i \end{bmatrix} = I \tag{E.2}$$

$$V_0 = \begin{bmatrix} \frac{-i}{\sqrt{3}} & \frac{1}{\sqrt{3}} & 0 & 0 \\ \frac{1}{\sqrt{2}} & \frac{-i}{\sqrt{2}} & 0 & 0 \\ \frac{1}{\sqrt{3}} & \frac{-i}{\sqrt{3}} & 0 & 0 \\ \frac{1}{\sqrt{2}} & \frac{-i}{\sqrt{2}} & 0 & 0 \\ \frac{1}{\sqrt{3}} & \frac{-i}{\sqrt{3}} & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} \frac{-i}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 & 0 \\ \frac{1}{\sqrt{2}} & \frac{-i}{\sqrt{2}} & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{E.3}$$

$$V_0M = \begin{bmatrix} \frac{-i}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 & 0 \\ \frac{1}{\sqrt{2}} & \frac{-i}{\sqrt{2}} & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \frac{i}{\sqrt{3}} & \frac{1}{\sqrt{2}} & \frac{i}{\sqrt{6}} & 0 \\ \frac{1}{\sqrt{3}} & \frac{i}{\sqrt{2}} & \frac{1}{\sqrt{6}} & 0 \\ \frac{1}{\sqrt{3}} & 0 & \frac{-\sqrt{2}}{\sqrt{3}} & 0 \\ 0 & 0 & 0 & i \end{bmatrix} = \begin{bmatrix} \frac{2}{\sqrt{6}} & 0 & \frac{1}{\sqrt{3}} & 0 \\ 0 & 1 & 0 & 0 \\ \frac{1}{\sqrt{3}} & 0 & \frac{-2}{\sqrt{6}} & 0 \\ 0 & 0 & 0 & i \end{bmatrix} = M_1 \tag{E.4}$$

$$V_1 = \begin{bmatrix} \frac{\frac{2}{\sqrt{6}}}{1} & 0 & \frac{\frac{1}{\sqrt{3}}}{1} & 0 \\ 0 & 1 & 0 & 0 \\ \frac{\frac{1}{\sqrt{3}}}{1} & 0 & \frac{-\frac{2}{\sqrt{6}}}{1} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} \frac{\sqrt{2}}{\sqrt{3}} & 0 & \frac{1}{\sqrt{3}} & 0 \\ 0 & 1 & 0 & 0 \\ \frac{1}{\sqrt{3}} & 0 & \frac{-\sqrt{2}}{\sqrt{3}} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{E.5}$$

$$V_1 M_1 = \begin{bmatrix} \frac{\sqrt{2}}{\sqrt{3}} & 0 & \frac{1}{\sqrt{3}} & 0 \\ 0 & 1 & 0 & 0 \\ \frac{1}{\sqrt{3}} & 0 & \frac{-\sqrt{2}}{\sqrt{3}} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \frac{2}{\sqrt{6}} & 0 & \frac{-i}{\sqrt{3}} & 0 \\ 0 & 1 & 0 & 0 \\ \frac{i}{\sqrt{3}} & 0 & \frac{2}{\sqrt{6}} & 0 \\ 0 & 0 & 0 & i \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & i \end{bmatrix} = M_2 \tag{E.6}$$

$$V_2 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & \epsilon(-\pi/2) \end{bmatrix} \tag{E.7}$$

$$V_2 M_2 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & \epsilon(-\pi/2) \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & i \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = M_3 = I \tag{E.8}$$

$$V_0^* V_1^* V_2^* = \begin{bmatrix} \frac{i}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 & 0 \\ \frac{1}{\sqrt{2}} & \frac{i}{\sqrt{2}} & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \frac{\sqrt{2}}{\sqrt{3}} & 0 & \frac{1}{\sqrt{3}} & 0 \\ 0 & 1 & 0 & 0 \\ \frac{1}{\sqrt{3}} & 0 & \frac{-\sqrt{2}}{\sqrt{3}} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & \epsilon(\pi/2) \end{bmatrix} \begin{bmatrix} \frac{1}{\sqrt{3}} & \frac{1}{\sqrt{2}} & \frac{i}{\sqrt{6}} & 0 \\ \frac{1}{\sqrt{3}} & \frac{-1}{\sqrt{2}} & \frac{i}{\sqrt{6}} & 0 \\ \frac{i}{\sqrt{3}} & 0 & \frac{-\sqrt{2}}{\sqrt{3}} & 0 \\ 0 & 0 & 0 & i \end{bmatrix} = M \tag{E.9}$$

F

Pruebas del algoritmo QCMD

F.1. Pruebas en el simulador

Aquí se recoge el código de salida del programa explicado en la sección 5.2 donde se ha usado el simulador de la API de Python QISKit para comprobar el funcionamiento de las distintas puertas implementadas. En el código F.1 se recoge el comando utilizado para lanzar dichos test, y en el código F.2 se recoge la salida por pantalla de dicho script, donde se ven los pasos del test y las soluciones obtenidas y simuladas. Recodar que el número de puertas calculado en la sección 4.4 era para un diseño en concreto, y esta implementación sigue determinadas decisiones de diseño que pueden hacer que estos valores cambien.

```
python aqp_matrix_test.py -n 3 -i 9 -v -a > test.out
```

Listing F.1: Comando de ejecución

```

Assembler written with 9777 gates

Generated matrix
[[-0.75+0.j  0.25-0.j  0.25-0.j  0.25-0.j  0.25-0.j  0.25-0.j  0.25-0.j  0.25-0.j
  ]
 [ 0.25-0.j -0.75+0.j  0.25-0.j  0.25-0.j  0.25-0.j  0.25-0.j  0.25-0.j  0.25-0.j]
 [ 0.25-0.j  0.25-0.j -0.75+0.j  0.25-0.j  0.25-0.j  0.25-0.j  0.25-0.j  0.25-0.j]
 [ 0.25-0.j  0.25-0.j  0.25-0.j -0.75+0.j  0.25-0.j  0.25-0.j  0.25-0.j  0.25-0.j]
 [ 0.25-0.j  0.25-0.j  0.25-0.j  0.25-0.j -0.75+0.j  0.25-0.j  0.25-0.j  0.25-0.j]
 [ 0.25-0.j  0.25-0.j  0.25-0.j  0.25-0.j  0.25-0.j -0.75+0.j  0.25-0.j  0.25-0.j]
 [ 0.25-0.j  0.25-0.j  0.25-0.j  0.25-0.j  0.25-0.j  0.25-0.j -0.75+0.j  0.25-0.j]
 [ 0.25-0.j  0.25-0.j  0.25-0.j  0.25-0.j  0.25-0.j  0.25-0.j  0.25-0.j -0.75+0.j
  ]]

Simulated matrix
[[-0.75+0.j  0.25-0.j  0.25-0.j  0.25-0.j  0.25-0.j  0.25-0.j  0.25-0.j  0.25-0.j
  ]
 [ 0.25-0.j -0.75+0.j  0.25-0.j  0.25-0.j  0.25-0.j  0.25-0.j  0.25-0.j  0.25-0.j]
 [ 0.25-0.j  0.25-0.j -0.75+0.j  0.25-0.j  0.25-0.j  0.25-0.j  0.25-0.j  0.25-0.j]
 [ 0.25-0.j  0.25-0.j  0.25-0.j -0.75+0.j  0.25-0.j  0.25-0.j  0.25-0.j  0.25-0.j]
 [ 0.25-0.j  0.25-0.j  0.25-0.j  0.25-0.j -0.75+0.j  0.25-0.j  0.25-0.j  0.25-0.j]
 [ 0.25-0.j  0.25-0.j  0.25-0.j  0.25-0.j  0.25-0.j -0.75+0.j  0.25-0.j  0.25-0.j]
 [ 0.25-0.j  0.25-0.j  0.25-0.j  0.25-0.j  0.25-0.j  0.25-0.j -0.75+0.j  0.25-0.j]
 [ 0.25-0.j  0.25-0.j  0.25-0.j  0.25-0.j  0.25-0.j  0.25-0.j  0.25-0.j -0.75+0.j
  ]]
```

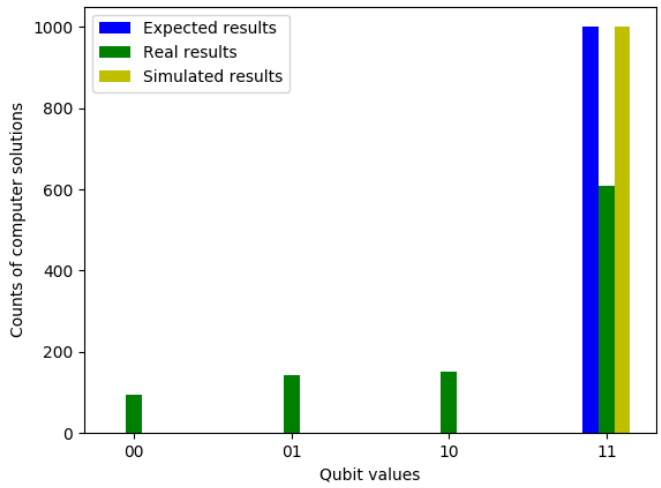
```
23
24 CLOSE
```

Listing F.2: Salida por pantalla

F.2. Pruebas en el ordenador

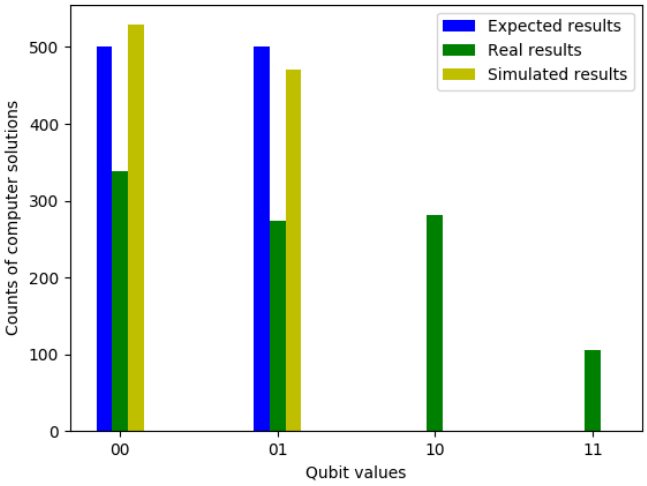
Aquí se recogen las distintas pruebas que se han realizado contra el ordenador cuántico real *IBM-Q* a través del servicio web proporrcionado por *QISKit*.

Podemos apreciar, como los circuitos con 2 qubits generan mejores resultados que aquellos con 3 qubits. Esto en general se debe a que una puerta para 3 qubits requiere de una profundidad mucho mayor que para 2 qubits. También observamos que para las puertas de intercambio, aquellas que modifican menos qubits, y por tanto requieren de menos puertas toffoli F.6 obtiene mejores resultados que aquellas que modifican más de un qubit F.9.



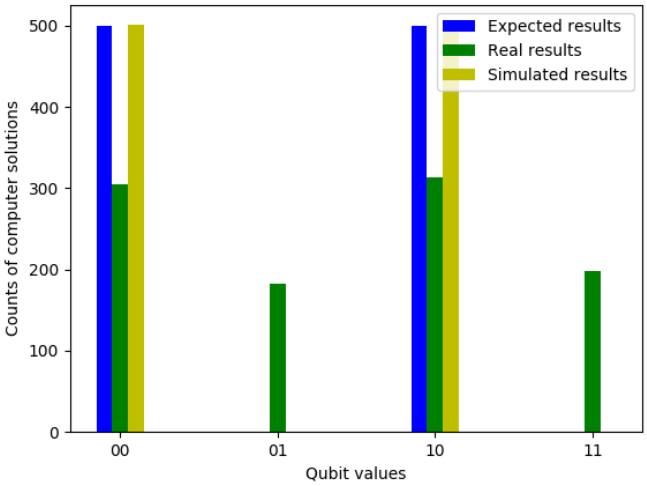
$$\begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \quad \text{(F.1)}$$

Figura F.1: Resultados del ordenador cuántico para un circuito de 2 qubits I.



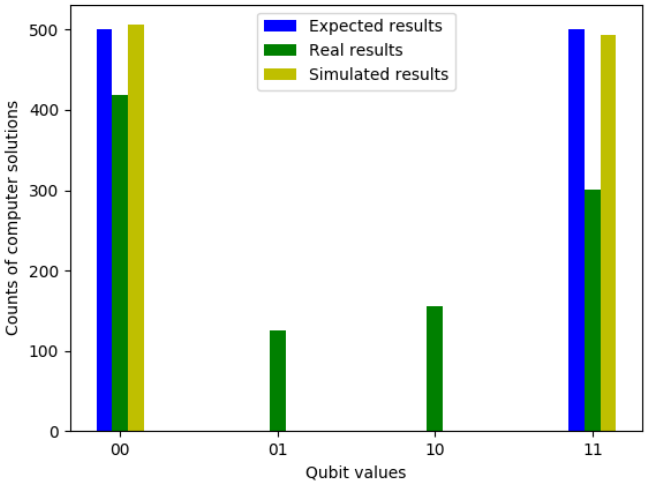
$$\begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 & 0 \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{F.2}$$

Figura F.2: Resultados del ordenador cuántico para un circuito de 2 qubits II.



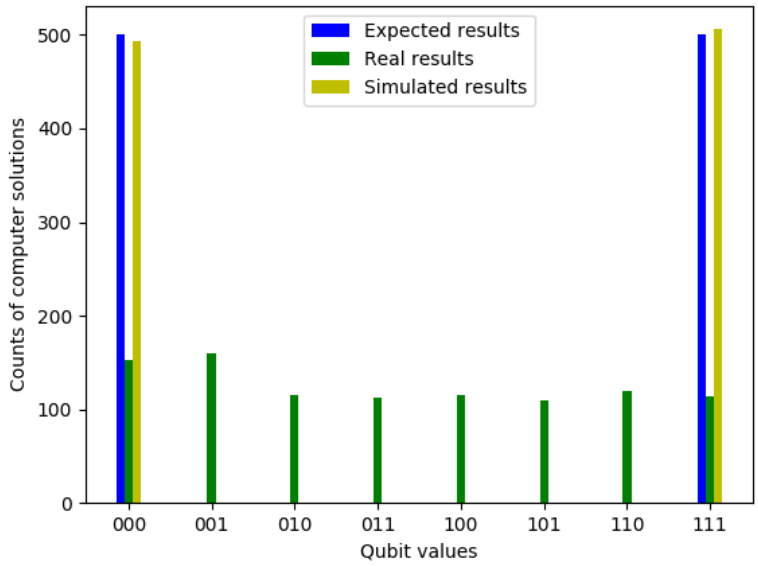
$$\begin{bmatrix} \frac{1}{\sqrt{2}} & 0 & \frac{1}{\sqrt{2}} & 0 \\ 0 & 1 & 0 & 0 \\ \frac{1}{\sqrt{2}} & 0 & -\frac{1}{\sqrt{2}} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{F.3}$$

Figura F.3: Resultados del ordenador cuántico para un circuito de 2 qubits III.



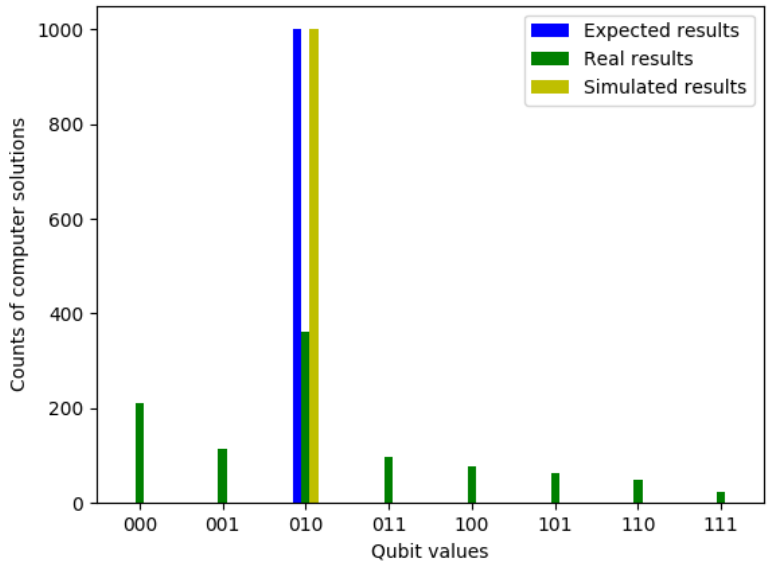
$$\begin{bmatrix} \frac{1}{\sqrt{2}} & 0 & 0 & \frac{1}{\sqrt{2}} \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ \frac{1}{\sqrt{2}} & 0 & 0 & -\frac{1}{\sqrt{2}} \end{bmatrix} \tag{F.4}$$

Figura F.4: Resultados del ordenador cuántico para un circuito de 2 qubits IV.



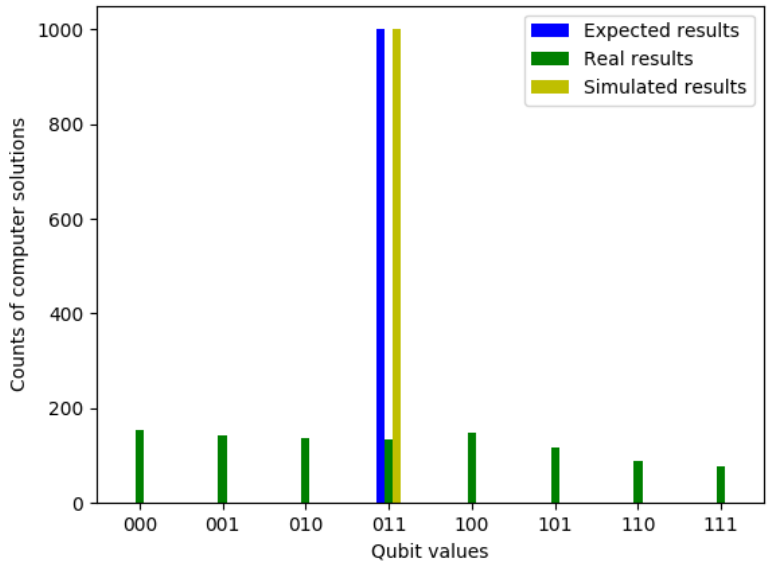
$$\begin{bmatrix} \frac{1}{\sqrt{2}} & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{\sqrt{2}} \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ \frac{1}{\sqrt{2}} & 0 & 0 & 0 & 0 & 0 & 0 & -\frac{1}{\sqrt{2}} \end{bmatrix} \quad \text{(F.5)}$$

Figura F.5: Resultados del ordenador cuántico para un circuito de 3 qubits I.



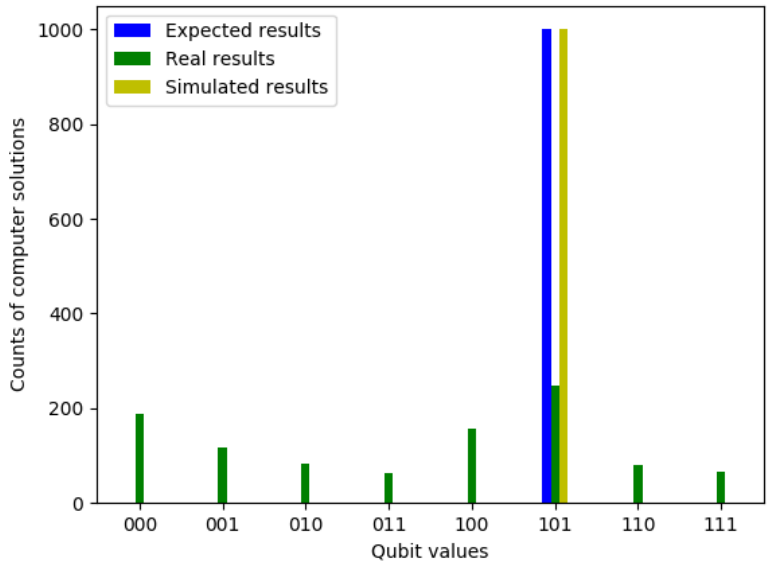
$$\begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \quad \text{(F.6)}$$

Figura F.6: Resultados del ordenador cuántico para un circuito de 3 qubits II.



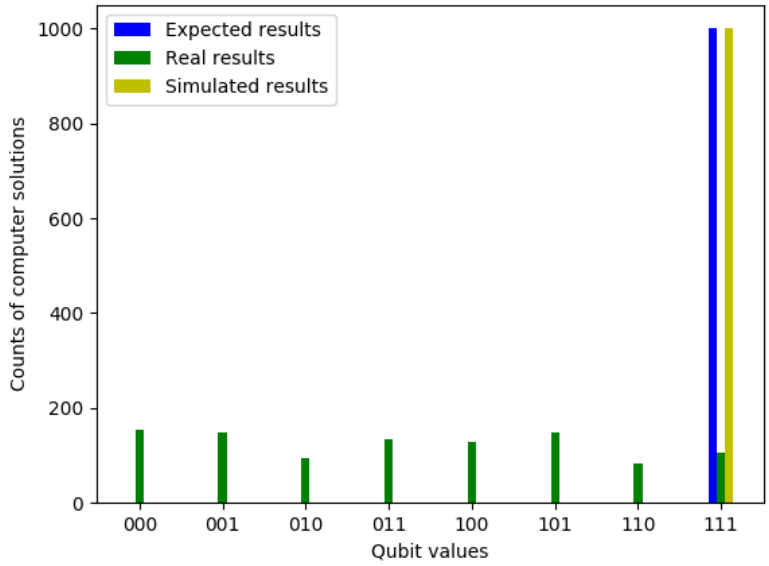
$$\begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \quad \text{(F.7)}$$

Figura F.7: Resultados del ordenador cuántico para un circuito de 3 qubits III.



$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \quad \text{(F.8)}$$

Figura F.8: Resultados del ordenador cuántico para un circuito de 3 qubits IV.



$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{F.9}$$

Figura F.9: Resultados del ordenador cuántico para un circuito de 3 qubits V.