Benemérita Universidad Autónoma de Puebla

FACULTAD DE CIENCIAS DE LA COMPUTACION

TEMA DE TESIS:

APLICACIÓN DE LA COMPUERTA DE FREDKINE EN UN DEMULTIPLEXOR CUÁNTICO

TESISTA:

CECILIA PANI GUTIÉRREZ

ASESOR DE TESIS:
DR. MAURICIO CASTRO CARDONA

PUEBLA, PUE. OCTUBRE 2007

Índice general

Introducción

Durante los pasados cuarenta años han sido asombrosos los avances que se han realizado en la construcción de computadoras. El número de átomos que se necesitan para representar un bit en memoria ha disminuido exponencialmente desde 1950. Del mismo modo, el número de transistores por chip, velocidad de reloj y energía disipada por operación lógica tendrán que seguir mejorando a una dirección exponencial. Para el año 2020 un bit de información requerirá solo de un átomo para representarlo. A ese nivel de miniaturización el comportamiento de los componentes de una computadora serán dominados por los principios de la física cuántica[8].

En la actualidad, uno de los grandes retos de la disciplina computacional es aumentar la eficiencia de los cálculos. El desarrollo de la tecnología actual tiende a miniaturizar los circuitos empleados en la construcción de computadoras, sin embargo se ha establecido que ese proceso de miniaturización tiene límites. Al llegar a niveles comparables con los tamaños de los átomos, las leyes y principios que rigen ya no son clásicos sino cuánticos.

Desde la última década del siglo pasado ha comenzado a tomar auge la computación cuántica. Si se tomara en cuenta otro tipo de hardware diferente al hardware clásico original, es decir, si se consideran nuevos dispositivos computacionales subordinados en su funcionamiento a las leyes y principios de la física cuántica entonces muchos problemas que hasta ahora se han considerado intratables tienen una solución.

La computación cuántica se encarga de la aplicación de la física cuántica a los sistemas de cómputo. Las computadoras cuánticas tienen sus raíces en la teoría cuántica, que trata con partículas atómicas, como electrones, protones y neutrones. De acuerdo a los resultados de la física cuántica, una partícula, por ejemplo un electrón, existe en varios estados, de acuerdo a la dirección del espín y su posición, hasta que se observa y mide, dichas partículas existen en más de un estado a la vez a lo que se llama superposición. Si nosotros consideramos a la dirección del espin (up y down) como los dos estados posibles del electrón, entonces el electrón estará en una superposición de esos estados. Un bit cuántico, llamado qubit sería la superposición de estos dos estados. Un qubit no sólo tendría la capacidad de representar un 1 o un 0, sino *ambos* al mismo tiempo. Una computadora que contenga 100 qubits podría representar hasta 2^{100} estados a la vez, una computadora cuántica podría representar los 2^{100} estados al mismo tiempo. Luego, tras un tick del reloj del sistema de la computadora cuántica, se realizaría un cálculo en los 2^{100} estados maquinales.

Un nuevo modelo matemático de computación es la Lógica Conservativa que estudia los principios fundamentales de la física, tales como la reversibilidad de las leyes dinámicas y la conservación de ciertas cantidades aditivas. La Lógica Conservativa provee indicaciones concernientes a la realización de sistemas de cómputo de alta ejecución, esto es, sistemas que hacen muy eficiente el uso de 'recursos de cómputo' ofrecidos actualmente por naturaleza.

Uno de los elementos de cómputo en que se basa la lógica conservativa es la Compuerta Cuántica de Fredkine, con esta compuerta es posible obtener funciones boleanas fundamentales como OR, NOT y $FAN-OUT$, que se presentarán más adelante. El propósito de esta Tesis es el manejo y

desarrollo matemático de la Compuerta Cuántica de Fredkine para así obtener la simulación de un Demultiplexor Cuántico.

Dentro del Capítulo 1 se estudiará brevemente la funcionalidad de la computación clásica para comprender el formalismo y lo potente que es la computación cuántica, se definirá su elemento principal (bit) y las limitaciones de la computación clásica.

En el Capítulo 2 se analizará el comienzo de la física cuántica, su interpretación matemática mediante los espínes de partículas, el concepto de superposición, los vectores estado así como su aplicación en la computación cuántica.

En el Capítulo 3 se analizará el formalismo de la computación cuántica como son el qubit, registros cuánticos, así como diferentes compuertas unitarias.

En el Capítulo 4 se llevará a cabo el objetivo de la tesis de aplicará la computación cuántica de Fredkine para el desarrollo de un demultiplexor cuántico. Se dará la definición de un demultiplexor clásico y la diferencia entre un demultiplexor clásico y un demultiplexor cuántico, se utilizará la lógica conservativa para poder dar uso a un demultiplexor cuántico mediante la compuerta cuántica de Fredkine, se dará una explicación del funcionamiento e implementación del demultiplexor cuántico; para finalizar este capítulo se darán algunos algoritmos, como son un Algoritmo para construir una Transformación unitaria (mediante el método de Shor), un Algoritmo para construir los productos Tensoriales y otro Algoritmo del Funcionamiento del Demultiplexor Cuántico, en este último algoritmo se da la Aplicación de la Compuerta Cuántica de Fredkine para obtener un Demultiplexor Cuántico.

Se anexan en los Apéndices el código fuente de dichos algoritmos y en el Apéndice A3 se visualizará el orden que tienen las entradas a cada matriz de 64×64, las cuales hacen uso de la compuerta de Fredkine.

Capítulo 1

Computación clásica

El objetivo de este capítulo es mostrar los principios teóricos de la computación clásica, se determinan las condiciones que se deben dar para que un problema sea computable. También se da una clasificación de las clases de complejidad y por último analizaremos las limitaciones de la computación clásica y el bloque fundamental de la computación clásica, el 'bit'.

1.1. Historia de la computación clásica

A través de la historia el ser humano ha usado diversos materiales y utilizado múltiples mecanismos en el diseño, construcción y operación de máquinas que agilicen y automaticen la realización de cálculos y el procesamiento de información. La computadora para llegar a ser tal como la conocemos actualmente, ha pasado por un proceso de evolución iniciado hace aproximadamente 2500 años, algunos consideran que las computadoras no tiene más que unos cientos de años de evolución, y otros sostienen que es un fenómeno iniciado recientemente en el siglo pasado. Algunos hechos que han marcado hitos importantes en este proceso son descritos a continuación. Antiguamente, los primeros modelos fueron manuales, estos se remontan aproximadamente hasta 500 A.C., cuando los egipcios inventaron un artefacto que consistía en una serie de esferas atravesadas por varillas; este artefacto fue cambiado y perfeccionado por los chinos; y posteriormente en el siglo XIII D.C. es cuando toma la forma clásica que conocemos; el ABACO está compuesto por 10 líneas con 7 esferas cada una, una línea corta todas las líneas en dos partes una más grande que la otra, ubicándose 2 esferas en la parte superior y cinco en la parte inferior. Mucho tiempo después, se desarrollaron modelos mecánicos y eléctricos, es así que, Blaise Pascal, en 1649, fabricó la PASCALINA, una máquina que hacía operaciones de 8 dígitos. En 1820, Charles Babbage con la ayuda de la Condesa Ada Byron, construyó dos equipos totalmente mecánicos, usaban ejes, engranajes y poleas para realizar cálculos; Byron fue la primera persona que programó una computadora, tiempo después un lenguaje de programación fue nombrado como Ada en su honor. Herman Hollerith desarrolló unas máquinas que clasificaban, ordenaban y enumeraban tarjetas perforadas. Estas se usaron en el censo realizado en 1890 por el gobierno de los Estados Unidos de Norte América. Konraz Suze, ingeniero alemán, en 1942, construyó la primera computadora digital (electromecánica binaria) programable. Entre 1937 y 1942 Atanasoff y Berry, construyeron un prototipo compuesto de tubos al vacío, capacitores y un tambor de rotatorio para el manejo de los elementos de la memoria, fue usada para resolver ecuaciones matemáticas complejas. En 1941 Turing construyó la COLLOSUS, una computadora que usaba miles de válvulas, 2400 bombas de vidrio al vacío, y un escáner con capacidad de leer 5000 caracteres por cinta de papel. En 1944 IBM (Interna Business Machines) construye la MARK I en cooperación con la Universidad de Harvard, media 15 metros de largo, 2.40 metros de altura y pesaba cinco toneladas. La ENIAC contaba con 17468 tubos de vidrio al vacío similares a los tubos de radio, fue construida en 1946 en la Universidad de Pensylvania. Finalmente se inició la era digital, con modelos electrónicos basados inicialmente en tubos de vacío y luego en transistores. La EDVAC fue la primera computadora electrónica digital, su memoria consistía en líneas de mercurio dentro

de un tubo de vidrio al vacío, donde se podía almacenar ceros y unos. El transistor, es el invento que más ha influenciado en la evolución de las computadoras, este fue concebido en 1948, por tres científicos en los laboratorios de Bell, este contiene un material semiconductor que funciona como un interruptor. En 1958 Kilby y Noycea, de la Texas Instrument, inventaron los circuitos integrados, haciendo que las computadoras fuesen cada vez más pequeñas. En Intel, en 1971, Hoff desarrollo un microprocesador de 4 bits que contenía 23000 transistores que procesaban 108 kHz o 0.06 MIPS, tenía 46 instrucciones y 4 kilobytes de espacio de almacenamiento. En 1974 Intel presentó una CPU compuesto por el microchip 8080, este contenía 4500 transistores y podía almacenar 64 kilobytes de memoria RAM, tenía un bus de datos de 8 bits. Wozniak y Jobs, en 1976, empiezan con Apple, revolucionando el mundo de las computadoras al introducir la interfaz gráfica y el ratón. El microprocesador Intel 8086, se lanzó en 1978, e inició una nueva era en la producción de computadoras personales. A comienzos de la década de los 80 IBM empezó a desarrollar las computadoras personales con PC-DOS como sistema operativo, empezando así una nueva era, donde las computadoras estaban al alcance de todos.

1.2. Máquinas de Turing

La teoría de la Computación ha sido considerada por mucho tiempo como un campo teórico independientemente de la física. Sin embargo, los pioneros de la Computación tales como Turing, Church y más tarde Godel fueron capaces de capturar correctamente el cuadro físico de los procesos computacionales; pero ya que sus trabajos no estaban referidos explícitamente a la física, se a asumido falsamente por mucho tiempo que los fundamentos de la teoría clásica de la Computación son en sí mismas "puras abstracciones".

Turing propuso modelos matemáticos para el cómputo que permitía el estudio de algoritmos, independientemente del hardware de cualquier computadora particular. Esta abstracción ha resultado fundamental en el campo de ciencias de la computación. El modelo que diseñó Turing se le conoce como **la máquina de Turing**.

El diseño de la máquina de Turing es el siguiente:

La máquina consiste de una cinta infinita dividida en celdas, cada celda puede contener un símbolo de una palabra, esta palabra puede ser una sucesión de los números 1 o 0, incluso cada celda puede contener un espacio en blanco. Hay también una cabeza que puede moverse a través de la cinta, leer una celda, escribir a una celda, cambiar su estado interno, o terminar el cálculo. El estado interno de la cabeza es un programa que decide cuál acción debe tomar la cabeza[2].

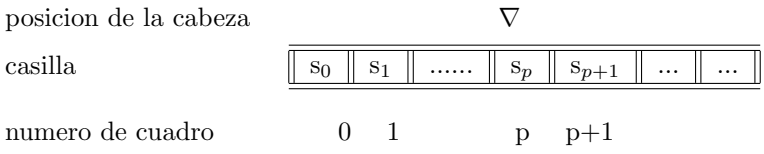
Interpretado en forma matemática una máquina de Turing corresponde a un arreglo de (S, A, Q, q_0, δ) donde

- S es el Alfabeto,
- A es el Alfabeto externo,
- Q es el conjunto de Estados del dispositivo de control,
- q_0 es el Estado inicial,
- δ es la Función de transición.

Los conjuntos S, A y Q son conjuntos finitos y $A \subset S$, q_0 es un elemento de Q y es una función de $Q \times S$ a $Q \times S \times \{-1, 0, 1\}$, que representado matemáticamente es $\delta : Q \times S \to Q \times S \times \{-1, 0, 1\}$[1]

[1]Cuando existe -1,0,1 quiere decir lo siguiente: la cabecera de la mquina de Turing puede quedarse inmóvil (0) o avanzar hacia delante (1)

Ahora veámoslo en una forma gráfica. Sea (σ,p,q) un estado de la máquina de Turing, aquí σ es una palabra del alfabeto S (que puede ser un número 1 o 0), p es un entero no negativo y q pertenece a Q. La palabra σ está escrita sobre la cinta, la cinta está dividida en celdas y en cada celda hay un símbolo de esa palabra. La cinta tiene una cabeza que se coloca sobre la celda numerada con p



La máquina de Turing también tiene un dispositivo de control, cuyo estado es dado por el elemento q del conjunto Q.

En un movimiento del trabajo, el dispositivo de control realiza la siguiente acción con un estado (σ, p, q):

1) Lee el símbolo que se encuentra bajo la cabeza (que es determinar s_p)

2) Calcula el valor de la función de transición δ (q,s_p) = (q' , s,Δp) [2]

3) Escribe en la cinta, en la celda p el símbolo s, se mueve un Δp y pasa al estado q', en otras palabras el nuevo estado de la máquina es dado por la tríada ((s_0, s_1,, ... , s_{p-1}, s_p), p+ Δp, q')

4) Si p+ $\Delta p \prec 0$ entonces la máquina se detiene.

De las acciones que realiza la máquina de Turing tenemos una sucesión de estados de la forma (α_0, p_0, q_0), (α_1, p_1, q_1), (α_2, p_2, q_2), y así sucesivamente hasta llegar al final de la palabra.

Tan simple como parezca este modelo, abarca toda la funcionalidad de las más modernas supercomputadoras actuales. Dado el tiempo y memoria no hay un solo cálculo que pueda ser ejecutado en una supercomputadora moderna que no pueda ser ejecutada en una máquina de Turing. El inmenso valor de la máquina de Turing es que cuando examinamos qué tipo de problemas se pueden resolver por una computadora, sólo necesitaremos examinarlo en una máquina de Turing. Si una máquina de Turing puede ejecutar el cálculo, entonces es computable. Si una máquina de Turing no puede, entonces la función no puede ser calculada por ninguna computadora clásica[9].

[2] El símbolo Δ representa un desplazamiento hacia el siguiente cuadro, ya sea hacia atrás o hacia adelante

1.3. La Tesis de Church

Aunque la lógica no prohíbe el cómputo físico de funciones arbitrarias, parece que la física lo hace. Como se sabe, cuando se diseñan computadoras rápidamente se alcanza un punto donde al adicionar mas hardware no se altera al conjunto de funciones calculables (bajo la idealización de que la capacidad de memoria es limitada). Church (1936) y Turing (1936) supusieron que estas limitaciones del computo no son dependientes del diseño de las computadoras, ni por el ingenio en la construcción de modelos de computadoras, estas limitaciones tienen carácter universal. Es la así llamada (hipótesis de Church Turing); de acuerdo a esto; la Tesis de Church establece:

Cualquier función que se considere naturalmente computable puede ser calculada por la maquina universal de Turing.

Nosotros no estamos acostumbrados a pensar en computación con términos físicos. Vemos a la computación como compuesto de operaciones teóricas, matemáticas; pero, si lo vemos bajo un escrutinio cercano, efectuar un cómputo es esencialmente un proceso físico. Esto puede tomar varias formas radicalmente diversas dependiendo del dispositivo computacional: potenciales del voltaje en las compuertas de un transistor en un microchip del silicio, granos en las barras de un ábaco, impulsos de nervio en el synapse de una neurona, etc. El cómputo mismo consiste en un conjunto de instrucciones (referidas a un algoritmo) realizadas por medio de un proceso físico. La tesis de Church, se refiere fundamentalmente a una relación profunda entre la física y el cálculo. Si una computadora (dispositivo físico) puede calcular una función, entonces puede ser calculada por una máquina de Turing. Cualquier modelo de computo futuro tendrá que subordinarse a los principios de la máquina de Turing. Deutch en su trabajo [2] reformula la tesis de Church de la siguiente manera:
Cualquier sistema físico finitamente realizable puede ser perfectamente simulado por un modelo universal de cómputo operando por medios finitos

Esto lleva a Deutch a contruir la máquina de Turing Cuántica. Una computadora cuántica, por lo tanto, también trabaja con los principios de la máquina de Turing clásica, la diferencia entre ellas es en la lectura de los resultados del cómputo. Mientras que en una maquina clásica los datos de salida son independientes de la forma física de leerlos, en la computadora cuántica los datos de salida tienen una fuerte dependencias de los medios físicos para observar el resultado. En su trabajo Deutsch considera el caso cuando las computadoras se comportan como sistemas cuánticos y pueden tener estados altamente no clásicos. Estas computadoras cuánticas podrían, por ejemplo, existir en una superposición de estados. Cada estado podría seguir coherente un camino de cómputo distintos y así producir una salida final. Este "paralelismo cuántico", alcanzado en una simple pieza de hardware supera en gran medida a cualquier paralelismo que se pueda pensar en las computadoras clásicas, de aquí que se puede prever que las computadoras cuánticas tienen una potencia sin precedente. La computadora cuántica es más potente, en el sentido de que existen problemas tratables en una computadora cuántica pero no en una computadora con arquitectura clásica.

Llevo de hecho otra década para que quedaran claras las potencialidades de las computadoras cuánticas y se exhibieran problemas específicos que se consideran intratables en la Computación clásica, pero que pueden resolverse eficientemente en una computadora cuántica. El más claro ejemplo es el problema de factorizacion. P. Shor (1995) mostró en su trabajo [2] que usando un algoritmo cuántico (o sea un algoritmo que corre en una computadora cuántica) es posible factorizar un numero entero grande. Este problema fue considerado un problema intratable en cualquier computadora clásica y Shor mostró que la clase de problemas accesibles a las computadoras cuánticas incluye problemas que no pueden ser eficientemente resueltos por computadoras clásicas.

Esto significa que hay problemas que son intratables para la computadora clásica puesto que se resuelven en un número de pasos exponencial y para la computadora cuántica sería ese mismo problema como tratable puesto que lo resolvería en un número de pasos polinomial.

Dicha declaración, hace que la Tesis de Church escape de lo real. Si una computadora cuántica puede hacer cosas que una máquina de clásica no puede, Cómo sabremos si no hay alguna computadora (sistema físico)que sea más potente que una máquina clásica?

La palabra clave para resolver ésta pregunta la obtenemos en la Tesis de Church cuando usa la palabra "física"(por ejemplo, la computación clásica utiliza la física clásica y la computación cuántica utiliza la física cuántica). Al tener gran esmero Turing y otros para intentar hacer sus modelos de abstracciones de cómputo matemático, no albergaron ninguna suposición oculta sobre la propia maquinaria de cómputo. De hecho, por muchos años se juzgó que estos modelos no habían sido basados en alguna suposición, tal como la naturaleza física de la computadora, pero esto no fue así. La computadora cuántica es más potente ya que puede resolver problemas que son intratables en cualquier computadora clásica que trabaja bajo el modelo de la máquina clásica de Turing. Cuando la tesis de Church fue escrita todavía no se había pensado en crear una computadora cuántica, cuyo objetivo es realizar operaciones que habían sido físicamente imposibles ejecutar, por ello, la tesis de Church es todavía válida para cualquier máquina de cómputo que funciona de una manera completamente clásica.

1.4. La Complejidad de Clases

Desde que se estableció que todas las computadoras clásicas pueden ser simuladas por una máquina de Turing, quedó una pregunta razonable qué problemas puede resolver una máquina de Turing?

Es interesante determinar si un problema es completamente computable, pero esto no es lo suficientemente aceptable para determinar si un problema puede ser resuelto en forma realista por una computadora física, en una cantidad razonable de tiempo. Si un cálculo tomara billones de años en obtener su resultado, podemos decir que es intratable desde la perspectiva del pragmatista. Un algoritmo se caracteriza por el número de operaciones y cantidad de memoria que se requiere para calcular una respuesta, en dependencia de una entrada de tamaño 'n'. Estas características del algoritmo determinan lo que se llama la complejidad de los algoritmos. Específicamente la complejidad de un algoritmo se determina al observar, dada una entrada 'n', el número de operaciones y memoria usados para completar el programa. El tamaño de la entrada es definido convenientemente para que sea el número de bits necesitados para representar el número.

Tiempo Polinomial: En algunos casos el tiempo de corrida del algoritmo dado está en el peor caso, de acuerdo al tamaño de la entrada. Por ejemplo si un algoritmo con un tamaño de entrada de n = 10 bits que tome 10^4+7*10^2+1001 (que es n^4+7∗n^2+1001) operaciones, para calcular su resultado entonces se considera un algoritmo de tiempo polinomial. Los problemas que se pueden resolver en tiempo polinomial, es decir, el tiempo necesitado para resolver un algoritmo no requiere miles o millones de años para obtener su resultado, son generalmente considerados "tratables". Los problemas que requieren más del tiempo polinomial son usualmente considerados "intratables", por ejemplo, un algoritmo que podría tomar 2^n operaciones para una entrada de tamaño 'n' sería considerado intratable, porque el número de operaciones crece exponencialmente con el tamaño de entrada 'n', de forma no polinomial[9].

Una vez ya dadas estas distinciones para determinar si un algoritmo puede ser resuelto por una computadora clásica, existen dos cuestiones que se deben contestar. Primero, puede una máquina de Turing resolver el problema? Si así es, entonces el problema es llamado computable. Segundo, se debe considerar la complejidad del algoritmo a ser usado. En general, si un algoritmo requiere un tiempo polinomial o menos, entonces para el computo es considerado como tratable y si no, entonces es considerado como intratable.

1.5. Limitaciones de las Computadoras Clásicas

Church y Turing comprendieron que para hacer un cambio físico se tendría que entender el formalismo en que trabaja la física clásica y así diseñar los cambios. La llegada de la física cuántica ha mostrado que el comportamiento de sistemas físicos, en una escala suficientemente pequeña, no se rige por las leyes de la física clásica. La escala donde estas asunciones clásicas ya no son validas es la misma escala a la que nos estamos aproximando rápidamente con la miniaturización de los componentes en nuestras computadoras. A la velocidad actual, el almacenamiento de memoria alcanzará, alrededor del año 2020, el tamaño de aproximadamente un átomo por bit. A este nivel no se espera que el bit se comporte de una manera clásica, el nivel atómico parecería ser absolutamente pequeño comparado con el tamaño de un componente de la computadora clásica. Estos límites no garantizan la terminación eventual en el desarrollo del hardware por otros medios. Los mismos efectos cuánticos que se prevén con la miniaturización continua de nuestras computadoras clásicas, pueden permitir el desarrollo de una computadora cuántica.

1.6. El bit clásico

Para entender las razones por las que una computadora cuántica difiere de una computadora clásica, primero debemos entender los rudimentos de la computadora clásica. El elemento de construcción fundamental para la computadora clásica es el "bit". Un bit es capaz de almacenar una información, puede tener un valor de 0 ó 1. Cualquier cantidad de información puede ser codificada en una lista de bits.

En una computadora clásica un bit es típicamente almacenado en un chip de silicón, o una fuente de unidad de disco duro de metal, o una cinta magnética. Aproximadamente 10^{10} átomos son usados actualmente para representar un bit de información. El más pequeño almacenamiento concebible para un bit comprende una sola partícula elemental de alguna clase. Por ejemplo, para cualquier partícula con un espín de 1/2, puede ser caracterizado por su valor de espín, y cuando se mide difiere entre +1/2 o -1/2.

Así, podemos codificar a 1 como +1/2 y 0 como -1/2, y si asumimos que podemos medir y manipular el espín de dicha partícula, entonces teóricamente se podría usar esta partícula para almacenar un bit de información. Si tratamos de usar el espín de partícula 1/2 como un bit clásico, uno que está siempre en el estado 0 ó 1 (para bit clásico sólo un estado a la vez), entonces fracasaría, porque estaríamos intentando aplicar la física clásica en una escala donde simplemente no se puede aplicar (se necesita que el espín sea capaz de representar ambos, 1 o 0, al mismo tiempo). Este espín de partícula 1/2 actuará, en cambio, en una manera cuántica[8].

El espín de partícula 1/2, cuyo comportamiento es de una manera cuántica, puede ser el bloque fundamental de construcción de una computadora cuántica. Podríamos llamarlo como un qubit, para denotar que es análogo en algunas formas a un bit en una computadora clásica. Justamente como en un registro de memoria en una computadora clásica es un arreglo de bits, por ejemplo,$(b_1, b_2, ..., b_n)$, un registro de memoria cuántica esta compuesta de varios espines de partículas de 1/2, o qubits, por ejemplo, $|q_1, q_2, ..., q_n\rangle$. No es necesario el espín de partícula 1/2, igualmente, bien podríamos usar un átomo de Hidrógeno y designar su electrón, así al medir el estado bajo sería el estado 0, y cuando esté en el primer estado excitado entonces sería el estado 1. Hay una multitud de representaciones posibles para el qubit. Para mas simplicidad sólo se discutirá el espín de partícula 1/2 de aquí en adelante, pero pueden hacerse más argumentos análogos.

Capítulo 2

Física Cuántica

Las computadoras cuánticas tienen sus raíces en la teoría cuántica, que trata con partículas atómicas, como electrones, protones y neutrones. Para considerar la capacidad de una computadora, que hace uso de los resultados de la física cuántica, nos será útil conocer el formalismo de la física cuántica. La física cuántica, también conocida como mecánica cuántica, surgió como resultado del fallo que tuvo la física clásica para controlar el comportamiento de fotones y otras partículas elementales. A continuación se mostrará más detalladamente este asunto.

2.1. Teoría Cuántica

Max Planck realizó una suposición radical al postular que un oscilador molecular sólo puede emitir ondas electromagnéticas en paquetes discretos, que llamó cuantos o fotones. Cada fotón tiene una longitud de onda, una frecuencia característica y una energía E que viene dada por $E = nh\nu$, donde n es un entero, ν es la frecuencia de la onda luminosa y h es la denominada constante de Planck. La longitud de onda λ está relacionada con la frecuencia según la ecuación $\lambda\nu = c$, donde c es la velocidad de la luz. Si la frecuencia se expresa en hercios, o ciclos por segundo, y la energía en julios, la constante de Plank es $6,626 \times 10^{-34}$Js [10], que es extremadamente pequeña. Con su teoría, Planck introdujo una dualidad onda-corpúsculo (onda-partícula) en la naturaleza de la luz.

Cuando una radiación electromagnética de una frecuencia determinada incide sobre un metal, de su superficie se desprenden cargas eléctricas negativas. Esto llevó a Einstein en 1905 a suponer que la luz sólo puede absorberse en cuantos, ó fotones, y que el fotón desaparece por completo en el proceso de absorción y cede toda su energía E a un solo electrón del metal (efecto fotoeléctrico), con esto Einstein amplió la teoría cuántica de Planck a la absorción de radiación electromagnética, lo que concedió una importancia aún mayor a la dualidad onda-corpúsculo de la luz.

Una ruptura radical con la física clásica corrió a cargo del físico danés Niels Born. Según Born, en los átomos existían ciertas órbitas en las que los electrones giran sin emitir radiación electromagnética. Estas órbitas permitidas, los llamados estados estacionarios, están determinadas por la condición de que el momento angular J de la órbita del electrón tiene que ser un múltiplo entero positivo de la constante de Planck dividida entre 2π , es decir, $J = nh/2\pi$, donde el número cuántico n puede tomar cualquier valor entero positivo.

Born propuso la hipótesis de que un electrón 'elevado' por una perturbación suficiente desde la órbita de menor radio y menor energía (el estado fundamental) hasta otra órbita vuelve a 'caer' al estado fundamental al poco tiempo. Esta caída está acompañada de la emisión de un único fotón con energía $E = h\nu$, que corresponde a la diferencia de energía entre las órbitas superior e inferior. Cada transición entre órbitas emite un fotón característico cuya longitud de onda y frecuencia están exactamente definidas.

Entre 1924 y 1930, se desarrolló un nuevo enfoque teórico de la dinámica para explicar el comportamiento subatómico. El nuevo planteamiento, llamado mecánica cuántica comenzó cuando el físico francés Louis de Broglie sugirió en 1924 que no sólo la radiación electromagnética, sino también la materia podía presentar una dualidad onda-corpúsculo. La longitud de onda de las llamadas ondas de materia asociadas con una partícula viene dada por la ecuación $\lambda = h/mv$, donde m es la masa de la partícula y v su velocidad. Posteriormente, los alemanes Werner Heisenberg, Max Born, Ernest Pascual Jordan, y el austriaco Erwin Schrödinger dieron a la idea planteada por de Broglie una forma matemática que podía aplicarse a numerosos fenómenos físicos y a problemas que no podían tratarse con la física clásica. Además de confirmar el postulado de Born sobre la cuantización de los niveles de energía de los átomos, la física cuántica hace que en la actualidad podamos comprender los átomos más complejos.

Desde entonces se han incorporado nuevos conceptos importantes al panorama de la física cuántica, más allá de la idea de Broglie sobre la dualidad onda-corpúsculo de la materia. Uno de estos conceptos es que los electrones deben tener una cierta propiedad magnética, debido a su movimiento rotatorio permanente y por tanto un momento angular intrínseco o espín. Después se comprobó que el espín es una propiedad fundamental de casi todas las partículas elementales.

2.2. Espín de partícula

Cuándo el comportamiento del espín de partícula 1/2, o qubit, difiere del bit clásico?. Un bit clásico puede almacenar un 1 o un 0, y cuando medimos el valor observado siempre será el valor almacenado. La física cuántica establece que cuando medimos el espín de partícula 1/2, determinaremos que está en el estado +1/2 o en el estado -1/2. De esta manera el qubit no difiere del bit clásico, porque se puede medir para que este en +1/2 (estado 1), ó -1/2 (estado 0.) Las diferencias entre el qubit y el bit vienen del tipo de clase de información que un qubit puede almacenar.

Según la física cuántica, se puede describir el estado de espín de partícula 1/2 por un vector estado en un espacio de Hilbert, en la Sección 2.4 se analizará este tema pero antes se describirá el Espacio de Hilbert.

2.3. Espacio de Hilbert

Un espacio de Hilbert es un tipo de espacio especial, tiene la propiedad de que es un espacio vectorial complejo y lineal[8]. Si un espacio vectorial completo tiene dimensión finita 'n' entonces se denota H^n. El espacio de un sistema cuántico se representa por un vector en un espacio vectorial H^n.

El término matemático de espacio se refiere a algo que depende de muchas coordenadas independientes que se pueden definir por un conjunto de ejes perpendiculares, uno por cada variable independiente. Por ejemplo, probablemente nos es familiar el sistema de coordenadas "x, y, z"donde los ejes x, y, y z son líneas mutuamente perpendiculares de números reales, los cuales coinciden en el punto x = 0, y = 0, z = 0.

El hecho de que el espacio de Hilbert es un espacio de vectores complejos significa que la longitud de los vectores dentro del espacio se describen con números complejos. Los números complejos son números que toman la forma $a + i * b$, donde a y b son números reales, e i es la raíz cuadrada de -1.

El hecho de que es un espacio de vectores lineales significa que puede sumar y multiplicar vectores que estén en el espacio de Hilbert y el vector resultante todavía quedará dentro del espacio de Hilbert. Todo lo que necesitamos para representar estados y transformaciones cuánticas son vectores y operadores en un espacio vectorial complejo.

2.4. Vector estado

En el espacio de Hilbert para el vector estado, que describe el estado del espín de partícula 1/2, estos ejes perpendiculares corresponderán a cada estado posible que el sistema pueda medir. Así el espacio de Hilbert para un solo qubit tendrá dos ejes perpendiculares, uno corresponde al espín de partícula 1/2 que está en el estado +1/2, y otra al espín de partícula 1/2 que está en el estado -1/2. Estos estados, cuyo vector se puede medir, son llamados 'estados propios'. El vector que existe en alguna parte de este espacio, que representa el estado de espín de partícula 1/2, se le llama "vector estado."La proyección, del vector estado dentro de uno de los ejes, muestra la contribución de esos ejes de estados propios para el estado entero. Esto significa que en general, el estado del espín de partícula 1/2 puede ser cualquier combinación de los estados base. De esta manera un qubit es totalmente diferente a un bit, porque un bit sólo puede tener el estado 0 ó 1, pero el qubit puede existir, en un principio, en cualquier combinación de los estados 0 (+1/2) y 1(-1/2), y sólo se restringe para que esté en el estado 0 ó 1 cuando nosotros medimos el estado.

2.5. Notación Dirac

Ahora se introduce una notación estándar para vectores estado en la física cuántica. El vector estado se escribe de la manera siguiente $|\psi\rangle$ y se le llama un "vector ket". Donde ψ es un vector en algún espacio vectorial, lo que significa que es una lista de números que contienen información sobre la proyección del vector estado en su estado base. El término ket y su notación vienen del físico Paulo Dirac que deseaba una forma de abreviatura concisa, una escritura de fórmulas, que ocurren en la física cuántica. Estas fórmulas frecuentemente toman la forma del producto de un vector línea con un vector columna. Así, él se refiere al vector línea como vector bra representado como $\langle y|$. El producto de un vector "bra y un ket"se escribe $\langle y|\,|x\rangle$, y se le conoce como "bracket". La notación bracket se utiliza para representar los estados cuánticos. [8]

2.6. Estados Propios y Superposición

Anteriormente, se mencionó que la proyección del vector estado, en uno de los ejes perpendiculares, en el espacio de Hilbert, muestra la contribución de los ejes de los estados propios en el estado entero, esto significa que los estados en un espacio de Hilbert se le llaman estados propios. Deberíamos pensar (con buena razón y por consiguiente en la física clásica) que el espín de partícula 1/2 podría existir por completo solamente en uno de los posibles estados +1/2 y -1/2, y por consiguiente su vector estado podría solo estar completamente ubicado sobre uno de los ejes de las coordenadas. Si los ejes de partículas son conocidos como x y y, entonces la coordenada del vector estado x que denota la contribución del estado -1/2 (0), y la coordenada del vector estado y que denota la contribución del estado +1/2 (1), debe ser (1,0) o (0,1.)

Parece razonable, pero no lo es. De acuerdo a la física cuántica un sistema cuántico puede existir en una mezcla de todos sus estados permitidos simultáneamente. Este es el Principio de Superposición, y es la clave del poder de la computación cuántica.

2.7. Interpretación de Probabilidad

Ahora que conocemos como representar el vector estado, como una superposición de estados, y conocemos que podemos medir el vector estado, para que esté en uno de los estados base, debemos determinar qué pasa cuando medimos el vector estado. Sabemos por la física cuántica que dado el vector estado a una condición inicial, evolucionará con respecto al tiempo, de acuerdo con la ecuación de Schrödenger:

$$\frac{ih\partial\,|X(t)\rangle}{\partial t} = H(t)\,|X(t)\rangle$$

Donde i es la raíz cuadrada de -1, h es $1{,}0545 * 10^{-34}$ Js, y H es el operador de Hamilton, que es determinado por las características físicas del sistema que está evolucionándose

La notación de esta expresión es:

$$\frac{ih\partial w_j(t)}{\partial t} = \sum_j H(t)_{ij} * w_j(t)$$

Parece que esta evolución es continua, pero estas ecuaciones sólo se aplican al sistema mecánico cuántico desarrollado en un ambiente aislado. La única forma para observar el estado del vector estado está en alguna forma causada por el sistema de mecánica cuántica para interactuar con el ambiente. Cuando el vector estado es observado, hace un brinco discontinuo brusco a uno de los estados propios, pero no es una violación a la ecuación de Schrödenger. Cuando el sistema mecánico cuántico interactúa con el ambiente de salida y mide el vector estado, se dice que tiene una decoherencia[8].

Ahora para entender porqué un vector estado fracasa, todavía necesitamos saber la forma en que este fracaso pasa. Para ejecutar cualquier tipo de cálculo útil, primero necesitamos indagar en qué estado base un sistema mecánico cuántico fracasa. La probabilidad de que el vector estado fracase en el j-ésimo estado propio, está dado por $\|w_j\|^2$, el cual se define como $a_j^2 + b_j^2$ si $w_j = a_j + i * b_j$, donde w_j es la proyección compleja del vector estado en el estado propio j-ésimo. En general, la posibilidad de elegir cualquier estado dado es:

$$Prob(j) = \frac{\|w_j\|^2}{\sum_{k=0}^{n-1} \|w_k\|^2}$$

Pero como se mencionó anteriormente insistiremos en tener el vector estado de longitud 1, y en este caso la expresión de probabilidad se simplifica a $Prob(j) = \|w_j\|^2$

Por ejemplo, la medición de un qubit en el estado $\alpha|0\rangle + \beta|1\rangle$ dará el valor clásico 0 con probabilidad $\|\alpha\|^2$ o el valor clásico 1 con probabilidad $\|\beta\|^2$. Si la respuesta 0 se observa, el estado se colapsa (esto es se transforma) a $|0\rangle$; si la respuesta es 1, el estado se transforma a $|1\rangle$.

Cuando medimos un qubit, proyectamos el vector en uno de los vectores base elegidos aleatoriamente dependiendo de sus amplitudes α y β. un qubit debe ser normalizado para que la suma de sus probabilidades sea uno ($\|\alpha\|^2 + \|\beta\|^2 = 1$)

Así, ahora sabemos cuando construir un sistema cuántico de n estados, que puede ser puesto en una superposición arbitraria de estados. También sabemos cuando medir la superposición resultante y obtener un estado base con una cierta probabilidad. Esto es todo lo que se necesita para entender el registro de memoria cuántica para poder simular su comportamiento.

Capítulo 3

Computación Cuántica

La computación cuántica se encarga de la aplicación de la física cuántica a los sistemas de cómputo. Las ciencias de la computación y los mecanismos cuánticos son dos de las teorías más famosas del siglo XX. Estas se unen para formar un nuevo campo: la teoría de computación cuántica, en la que nos preguntamos qué podría hacer una computadora si siguiera las leyes de los mecanismos cuánticos. Lo que necesitamos para representar estados y transformaciones cuánticos son vectores y operadores en un espacio vectorial complejo.

Así como en la computación clásica funciona mediante bits, registros y compuertas clásicas también en la computación cuántica, basado en principios de la física cuántica, se basa en qubits, registros cuánticos de qubits y compuertas unitarias (conocidas también como transformaciones unitarias u operadores unitarios). El objetivo de este capítulo es dar una definición matemática de estos conceptos.

La computación cuántica trabaja de la siguiente manera: prepara un registro cuántico en un estado conocido y se le aplica una compuerta cuántica al registro, por ejemplo, se aplica un operador unitario a algunos qubits en un orden preciso y se mide el estado final del registro (al final) para saber su contenido. Es importante recordar que la computación cuántica es probabilística por naturaleza, ya que aunque la evolución de los estados es determinista, los pasos de medición dan resultados probabilistas[5].

3.1. El Qubit Cuántico

El bit cuántico es un vector normalizado en H^2 [5], varía el espacio de acuerdo al qubit; en la Sección 2.4 se define un vector estado, dicho vector estado es conocido como qubit, se puede escribir como una combinación lineal

$$|\Psi\rangle = \alpha\,|x_0\rangle + \beta\,|x_1\rangle$$

Donde los números complejos α y β son las amplitudes de los estados y $|x_0\rangle, |x_1\rangle$ es una base ortonormal en H^2 .y x_0 , x_1 son los estados propios. Al decir que es normalizado significa que $\|\alpha\|^2 + \|\beta\|^2 = 1$

En la Sección 2.4 analizamos que el qubit puede ser medido para tener un espín de +1/2 o -1/2, éste puede en general ser una superposición de estados cuando no estamos midiéndolos, nos referiremos a su estado de la siguiente manera:

Sea x_1 el estado propio correspondiente al espín de estado +1/2, y sea x_0 el estado propio correspondiente al espín de estado -1/2. Sea ψ el estado total del vector estado, y sea α y β los números complejos que tienen la amplitud de los estados base del estado total, en general tenemos:

$$|\psi\rangle = \alpha\,|x_0\rangle + \beta\,|x_1\rangle \equiv \begin{pmatrix} \alpha \\ \beta \end{pmatrix}$$

A esto debemos recordar que las amplitudes de los estados base α y β son números complejos y que cuando el estado se mide, que tomará el valor del estado

$$0\,|x_0\rangle + \beta\,|x_1\rangle \equiv \begin{pmatrix} 0 \\ \beta \end{pmatrix}$$

ó el estado

$$\alpha\,|x_0\rangle + 0\,|x_1\rangle \equiv \begin{pmatrix} \alpha \\ 0 \end{pmatrix}$$

Esto es análogo a un sistema, que es un vector con las amplitudes de estado reales en un plano de dos dimensiones, en un espacio vectorial de Hilbert en H^2. Sean los vectores unitarios x y y los estados base para este plano bidimensional. En este caso sabemos que el estado de cualquier vector V se puede describir de la siguiente manera:

$$V = x_0 * x + y_0 * y \equiv (x_0, y_0)$$

No es necesario desde una perspectiva física, para el vector estado, que sea un vector unitario pero es más fácil su uso para cálculos futuros, así asumiremos de aquí en adelante que el vector estado tiene longitud de 1.

Con todo esto en mente, hemos definido completamente el bloque de construcción básica de la computadora cuántica, el qubit. Este difiere del bit clásico, fundamentalmente, en que el qubit puede existir en cualquier superposición del estado 0 y 1 cuando no ha sido medido. Podemos medir en una base diferente (ortonormal), por ejemplo $|0\rangle, |1\rangle$ o $|-1/2\rangle, |+1/2\rangle$

3.2. El Registro de Memoria Cuántica

Un registro cuántico de n qubits es un vector normalizado en H^{2^n}. Se puede expresar como una combinación de los vectores base $|0\rangle, ..., |2^n - 1\rangle$:

$$|\psi\rangle = \sum_{i=0}^{2^n-1} \alpha_i\,|i\rangle$$

con restricción $\sum_{i=0}^{2^n-1} \|\alpha_i\|^2 = 1$

Hemos considerado dos estados del sistema cuántico, específicamente el espín de partícula 1/2. No obstante, un sistema cuántico no se restringe para que solo tenga un sistema de dos estados. Muchas de las discusiones anteriores, para un sistema cuántico de dos estados, son aplicables a un sistema cuántico general de n estados.

Un sistema de n estados en el espacio de Hilbert tiene n ejes perpendiculares, o estados propios, que representan los estados posibles en que el sistema puede ser medido. Como con el sistema de dos estados, cuando medimos el sistema cuántico de n estados, encontraremos siempre que va a estar en exactamente uno de los n estados, y no en una superposición de n estados. El sistema permite todavía que una superposición de n estados exista, mientras no esta siendo medido.

Matemáticamente si dos estados del sistema cuántico, con los ejes de coordenadas x_0, x_1, se pueden describir por

$$|X\rangle = w_0 * |x_0\rangle + w_1 * |x_1\rangle \equiv \begin{pmatrix} w_0 \\ w_1 \end{pmatrix}$$

Entonces un sistema cuántico de n estados, con coordenadas x_0, x_1,...,x_{n-1}, se puede describir por

$$|X\rangle = \sum_{k=0}^{n-1} w_k * |x_k\rangle$$

En general, un sistema cuántico con n estados base pueden representarse por n números complejos, de w_0 a w_{n-1}. Cuando así es, entonces el estado se escribirá

$$|x\rangle = \begin{pmatrix} w_0 \\ \vdots \\ w_j \\ \vdots \\ w_{n-1} \end{pmatrix}$$

Donde se entiende que w_k se refiere a la amplitud compleja para el estado propio k-ésimo.

Usando esta información podemos construir un registro de memoria cuántica de salida, para los qubits descritos en la sección previa. Podemos almacenar cualquier número n en el registro de memoria cuántica, tan grande como suficientes qubits tengamos, justamente como podemos almacenar cualquier número n en un registro clásico tan grande como suficientes bits clásicos tengamos para representar ese número. El estado del registro cuántico con n estados está dado por la fórmula anterior. Notemos que en general un registro cuántico compuesto de n qubits requiere 2^n números complejos para describir completamente su estado. un registro de n qubits puede ser medido para que esté en uno de los 2^n estados, y cada estado requiere un número complejo para representar la proyección del estado total dentro de ese estado.

Por ejemplo sea el producto tensorial de los qubits $|\psi\rangle$ y $|\phi\rangle$

$$|\psi\phi\rangle = \alpha\gamma\,|00\rangle + \alpha\delta\,|01\rangle + \beta\gamma\,|10\rangle + \beta\delta\,|11\rangle$$

y su espacio vectorial es H^4, la base estándar de H^4 es $|00\rangle, |01\rangle, |10\rangle, |11\rangle$ y sus estados son 00, 01, 10 y 11.

El registro cuántico $|\psi\phi\rangle$ compuesto de 2 qubits requiere 2^2 números complejos $\alpha\gamma, \alpha\delta, \beta\gamma y \beta\delta$ para así poder describir completamente su estado. Al medir el registro dará uno de los 2^2 estados, por ejemplo podría dar el estado 00 con número complejo $\alpha\gamma$ para representar la proyección del estado total dentro de ese estado.

En contraste, un registro clásico compuesto de n bits requiere sólo de n enteros para describir completamente su estado.

Esto significa que uno puede almacenar una cantidad exponencial de información en un registro cuántico. Aquí vemos algunas de las primeras sugerencias, como el que una computadora cuántica

puede ser exponencialmente más potente que una computadora clásica en algunos aspectos. Recordando la discusión de complejidad, los problemas que pueden ser resueltos en tiempo polinomial, son generalmente tratables y los problemas que pueden ser resueltos en tiempo exponencial son pensados como intratables. Si una computadora cuántica realmente es exponencialmente más poderosa que una computadora clásica, entonces algunos problemas actualmente intratables podrán ser convertidos a tratables!

3.3. Operación de Qubits

Producto tensorial: Para hacer el cómputo útil, necesitaremos más que un qubit. Para representar un grupo de qubits, debemos combinar sus espacios vectoriales usando una herramienta matemática llamada producto tensorial. Si agrupamos unos qubits en H^m y otros qubits en H^n entonces el espacio resultante será $H^{mn} = H^m \otimes H^n$. Las siguientes notaciones son todas equivalentes:

$$|u\rangle \otimes |v\rangle \equiv |u\rangle\,|v\rangle \equiv |uv\rangle$$

Para dar un ejemplo del producto tensorial de dos qubits, supongamos que tenemos dos vectores $|\psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle\,, |\phi\rangle = \gamma\,|0\rangle + \delta\,|1\rangle \in H^2$. El estado del producto es:

$$|\psi\phi\rangle = \alpha\gamma\,|00\rangle + \alpha\delta\,|01\rangle + \beta\gamma\,|10\rangle + \beta\delta\,|11\rangle$$

Y su espacio vectorial es H^4. La base estándar de H^4 es $|00\rangle\,, |01\rangle\,, |10\rangle\,, |11\rangle$. Las leyes de medición se pueden generalizar para muchos qubits. En este caso, midiendo $|\psi\phi\rangle$ resultará en dos clásicos bits dependiendo de sus probabilidades (por ejemplo, dará 00 con probabilidad $\|\alpha\gamma\|^2$).

Representado matemáticamente el producto tensorial del qubit $|00\rangle$ es:

$$|00\rangle = |0\rangle \otimes |0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \otimes \begin{pmatrix} 1 \\ 0 \end{pmatrix}$$

$$|00\rangle = \begin{pmatrix} 1\begin{pmatrix} 1 \\ 0 \end{pmatrix} \\ 0\begin{pmatrix} 1 \\ 0 \end{pmatrix} \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

El producto tensorial de 3 qubit es:

$$|000\rangle = |0\rangle \otimes |0\rangle \otimes |0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \otimes \begin{pmatrix} 1 \\ 0 \end{pmatrix} \otimes \begin{pmatrix} 1 \\ 0 \end{pmatrix}$$

$$|000\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \otimes \begin{pmatrix} 1\begin{pmatrix} 1 \\ 0 \end{pmatrix} \\ 0\begin{pmatrix} 1 \\ 0 \end{pmatrix} \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \otimes \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} 1\begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix} \\ 0\begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix} \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

Producto interno o escalar: Es el producto conocido como "braket" $\langle\bullet|\,|\bullet\rangle$, que es el qubit "bra-$\langle\theta|$, vector de 1 fila y 2^n columnas, con el segundo qubit "ket-$|\psi\rangle$, el cual representará el vector de

2^n filas y 1 columna; realizando las operaciones con los estados $\langle 0| = (10)$ y $|0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$ tenemos que el producto interno de los qubits $\langle 0\,|0\rangle$ es:

$$\langle 0\,|0\rangle = \begin{pmatrix} 1 & 0 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \end{pmatrix} = 1$$

$$\langle 0\,|1\rangle = \begin{pmatrix} 1 & 0 \end{pmatrix} \begin{pmatrix} 0 \\ 1 \end{pmatrix} = 0$$

$$\langle 1\,|0\rangle = \begin{pmatrix} 0 & 1 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \end{pmatrix} = 0$$

$$\langle 1\,|1\rangle = \begin{pmatrix} 0 & 1 \end{pmatrix} \begin{pmatrix} 0 \\ 1 \end{pmatrix} = 1$$

Representación de proyección de qubits (Producto ket-bra): Es el producto de la forma ket-bra de 2 qubits de 1 estado, la obtenemos al multiplicar el primer qubit de la forma "ket"$= |\psi\rangle$, el cual representará el vector de 2^n filas y 1 columna, con el segundo qubit que tendrá la forma "bra"$= \langle 0|$, que es el vector de 1 fila y 2^n columnas.

Por ejemplo el producto de los qubits $|0\rangle\langle 0|$ es:

$$|0\rangle\langle 0| = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \begin{pmatrix} 1 & 0 \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix}$$

El producto de los qubits $|1\rangle\langle 0|$ es:

$$|1\rangle\langle 0| = \begin{pmatrix} 0 \\ 1 \end{pmatrix} \begin{pmatrix} 1 & 0 \end{pmatrix} = \begin{pmatrix} 0 & 0 \\ 1 & 0 \end{pmatrix}$$

El espacio de Hilbert H es un espacio vectorial lineal que está sobre el conjunto escalar C [10]. Sean los qubits $|\psi\rangle, |\phi\rangle, |\gamma\rangle \in H$ y $\alpha, \beta \in C$, tenemos que:

Combinaciń Lineal: $|\psi\rangle + |\phi\rangle \in H$
Multiplicación Escalar: $\alpha\,|\psi\rangle \in H$
Elemento cero: $|\psi\rangle + 0 = |\psi\rangle$
Elemento inverso: $|\psi\rangle + (-|\psi\rangle) = 0$

El producto interno tiene las siguientes condiciones:

$$\langle\psi\,|\phi + \gamma\rangle = \langle\psi\,|\phi\rangle + \langle\psi\,|\gamma\rangle$$

$$\langle\psi\,|\alpha\phi\rangle = \alpha\,\langle\psi\,|\phi\rangle$$

$$\langle\psi\,|\phi\rangle = (\langle\phi\,|\psi\rangle)^*$$

$$\langle\psi\,|\psi\rangle = 0 \Leftrightarrow |\psi\rangle = 0$$

$$\|\psi\| \equiv \sqrt{\langle\psi\,|\psi\rangle} \geq 0$$

3.4. Representación de Qubits

Para poder hacer diferentes operaciones de estados, y así hallar las compuertas, necesitamos conocer la representación de los qubits que emplearemos.

2^1 ESTADOS. Los qubits con un estado, conocidos como 1 qubit, son $|0\rangle$ y $|1\rangle$, su representación vectorial es:

$$|0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \qquad |1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

2^2 ESTADOS. Los qubits con dos estados son $|00\rangle, |01\rangle, |10\rangle$ y $|11\rangle$, estos se obtienen mediante el producto tensorial de los dos qubits, por ejemplo:

$$|10\rangle = |1\rangle \otimes |0\rangle$$

su representación vectorial es:

$$|00\rangle = \begin{pmatrix} 1\begin{pmatrix} 1 \\ 0 \end{pmatrix} \\ 0\begin{pmatrix} 1 \\ 0 \end{pmatrix} \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix} \qquad |01\rangle = \begin{pmatrix} 1\begin{pmatrix} 0 \\ 1 \end{pmatrix} \\ 0\begin{pmatrix} 0 \\ 1 \end{pmatrix} \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix}$$

$$|10\rangle = \begin{pmatrix} 0\begin{pmatrix} 1 \\ 0 \end{pmatrix} \\ 1\begin{pmatrix} 1 \\ 0 \end{pmatrix} \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \end{pmatrix} \qquad |11\rangle = \begin{pmatrix} 0\begin{pmatrix} 0 \\ 1 \end{pmatrix} \\ 1\begin{pmatrix} 0 \\ 1 \end{pmatrix} \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}$$

2^3 ESTADOS. Los qubits con tres estados son $|000\rangle, |001\rangle, |010\rangle, |011\rangle, |100\rangle, |101\rangle, |110\rangle$ y $|111\rangle$, estos se obtienen mediante el producto tensorial de los tres qubits, por ejemplo:

$$|010\rangle = |0\rangle \otimes |10\rangle = |0\rangle \otimes |0\rangle \otimes |1\rangle$$

la representación vectorial es:

$$|000\rangle = \begin{pmatrix} 1\begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix} \\ 0\begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix} \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} \qquad |001\rangle = \begin{pmatrix} 1\begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix} \\ 0\begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix} \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

$$|010\rangle = \begin{pmatrix} 1\begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \end{pmatrix} \\ 0\begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \end{pmatrix} \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} \qquad |011\rangle = \begin{pmatrix} 1\begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix} \\ 0\begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix} \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

$$|100\rangle = \begin{pmatrix} 0 \begin{pmatrix} 1\\0\\0\\0 \end{pmatrix} \\ 1 \begin{pmatrix} 1\\0\\0\\0 \end{pmatrix} \end{pmatrix} = \begin{pmatrix} 0\\0\\0\\0\\1\\0\\0\\0 \end{pmatrix} \qquad |101\rangle = \begin{pmatrix} 0 \begin{pmatrix} 0\\1\\0\\0 \end{pmatrix} \\ 1 \begin{pmatrix} 0\\1\\0\\0 \end{pmatrix} \end{pmatrix} = \begin{pmatrix} 0\\0\\0\\0\\0\\1\\0\\0 \end{pmatrix}$$

$$|110\rangle = \begin{pmatrix} 0 \begin{pmatrix} 0\\0\\1\\0 \end{pmatrix} \\ 1 \begin{pmatrix} 0\\0\\1\\0 \end{pmatrix} \end{pmatrix} = \begin{pmatrix} 0\\0\\0\\0\\0\\0\\1\\0 \end{pmatrix} \qquad |111\rangle = \begin{pmatrix} 0 \begin{pmatrix} 0\\0\\0\\1 \end{pmatrix} \\ 1 \begin{pmatrix} 0\\0\\0\\1 \end{pmatrix} \end{pmatrix} = \begin{pmatrix} 0\\0\\0\\0\\0\\0\\0\\1 \end{pmatrix}$$

3.5. Compuertas unitarias

Una compuerta unitaria también se le conoce como *operador unitario o transformación unitaria.* una transformación unitaria es un operador lineal que actúa en un espacio vectorial H^n.[4] Para que sea unitaria debe satisfacer las siguientes propiedades [5]:

1. u:$|\psi\rangle \rightarrow |\phi\rangle$ $\qquad |\psi\rangle, |\phi\rangle \in H^n$

2. $\langle\phi\,|\psi\rangle = \langle u\phi\,|u\psi\rangle$ $\qquad Paratodo\,|\psi\rangle, |\phi\rangle \in H^n$

3. Existe u^{-1} tal que uu^{-1}=1

De la propiedad 1 tenemos que al aplicar una transformación unitaria a un qubit $|\psi\rangle$, que se encuentra en el espacio de Hilbert H^2, tendremos como resultado otro qubit que también se encontrará dentro del espacio de Hilbert [4], ejemplo:

$$|\psi\rangle = \alpha\,|x_0\rangle + \beta\,|x_1\rangle$$

$$u\,(|\psi\rangle) = u\,(\alpha\,|x_0\rangle + \beta\,|x_1\rangle) = \alpha u\,|x_0\rangle + \beta u\,|x_1\rangle$$

De la propiedad 2 se observa que el producto interno es igual al producto interno de sus transformaciones, es decir, al aplicar una transformación unitaria al producto interno, dicha transformación no perjudica el valor del producto.

Sea $\langle\phi| = \langle 1|$ y $|\psi\rangle = |0\rangle$

Su producto interno es:

$$\langle 1\,|0\rangle = \begin{pmatrix} 0 & 1 \end{pmatrix} \begin{pmatrix} 1\\0 \end{pmatrix} = 0$$

Sea u la transformación unitaria NOT, de aquí $\langle u\phi\,|u\psi\rangle =$

$$\langle 0\,|1\rangle = \begin{pmatrix} 1 & 0 \end{pmatrix} \begin{pmatrix} 0\\1 \end{pmatrix} = 0$$

De la propiedad 3 tenemos que los operadores son llamados unitarios cuando $u^\dagger = u^{-1}$[4]. Desde que la evolución temporal de un sistema cuántico se describe por un operador unitario y $u^\dagger(t) = u^{-1}(t)$, muestra que el comportamiento temporal de un sistema cuántico es reversible.

Un operador unitario u de n qubit describe una transformación u: $C^{2^n} \to C^{2^n}$ y por consiguiente corresponde a una matriz de dimensión $2^n \times 2^n$ en los números complejos C.

Un operador unitario es una compuerta cuántica. una secuencia de operadores unitarios actuando sobre un vector produce un resultado que puede ser interpretado como una operación lógica

3.5.1. Compuerta IDENTIDAD

La compuerta IDENTIDAD, representada por la letra I, trabaja para 1 o más qubits, en esta parte se aplicará a 1 qubit. La propiedad de la compuerta identidad es que si se le aplica la compuerta identidad a un qubit dará como resultado el mismo qubit. [1]

Para obtener esta compuerta utilizaremos algunas operaciones aplicadas a los qubits. En este caso será.

$$|0\rangle\langle 0| + |1\rangle\langle 1|$$

y matemáticamente expresado tenemos:

$$I = |0\rangle\langle 0| + |1\rangle\langle 1| = \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix} + \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

Si aplicamos esta compuerta a los qubits $|0\rangle$ y $|1\rangle$ entonces tendremos como resultado $|0\rangle$ y $|1\rangle$ respectivamente. Lo que significa:

$$I|1\rangle = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}\begin{pmatrix} 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \end{pmatrix} = |1\rangle \qquad I|0\rangle = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}\begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \end{pmatrix} = |0\rangle$$

Esta compuerta nos será útil para obtener otras compuertas, como la compuerta XOR y la compuerta TOFFOLI.

3.5.2. Compuerta NOT (X)

La compuerta cuántica NOT la aplicaremos para 1 qubit, y la representaremos con X. Si aplicamos esta compuerta al qubit $|0\rangle$ obtendremos como resultado $|1\rangle$, y viceversa. Para obtener esta compuerta es mediante el siguiente grupo de qubits. [1]

$$|0\rangle\langle 1| + |1\rangle\langle 0|$$

lo que significa:

$$X = |0\rangle\langle 1| + |1\rangle\langle 0| = \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix} + \begin{pmatrix} 0 & 0 \\ 1 & 0 \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$$

Si aplicamos un NOT al qubit $|0\rangle$ obtendremos como resultado $|1\rangle$, y si aplicamos un NOT al qubit $|1\rangle$ obtendremos $|0\rangle$. A continuación veremos la demostración:

$$X|1\rangle = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}\begin{pmatrix} 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \end{pmatrix} = |0\rangle \qquad X|0\rangle = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}\begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \end{pmatrix} = |1\rangle$$

Esta compuerta nos será útil para obtener la compuerta XOR.

3.5.3. Compuerta P(π)=Z

La compuerta P(π) trabaja para 1 qubit [1]. La denotaremos con Z, el efecto de esta compuerta es:

$$Z\left|0\right\rangle = \left|0\right\rangle \qquad Z\left|1\right\rangle = -\left|1\right\rangle$$

Para obtener la compuerta P(π) necesitamos aplicar

$$\left|0\right\rangle\left\langle 0\right| + exp.(i\pi)\left|1\right\rangle\left\langle 1\right|$$

En el que sustituiremos exp.(iπ) por -1

$$Z = \left|0\right\rangle\left\langle 0\right| + exp.(i\pi)\left|1\right\rangle\left\langle 1\right| = \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix} + (-1)\begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$$

Aplicando esta compuerta a los qubits $\left|0\right\rangle$ y $\left|1\right\rangle$ tendremos lo siguiente:

$$Z\left|1\right\rangle = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}\begin{pmatrix} 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ -1 \end{pmatrix} = -\left|1\right\rangle \qquad Z\left|0\right\rangle = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}\begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \left|0\right\rangle$$

3.5.4. Compuerta XZ=Y

Esta compuerta también actuará para un solo qubit. El efecto que tiene esta compuerta para los qubits $\left|0\right\rangle$ y $\left|1\right\rangle$ es la siguiente:

$$Y\left|0\right\rangle = \left|1\right\rangle \qquad Y\left|1\right\rangle = -\left|0\right\rangle$$

Para obtener esta compuerta cuántica [1] usaremos la combinación de las compuertas NOT, que lo representaremos con X, y P(π), que lo representaremos con Z; sus matrices son las siguientes:

$$X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \qquad Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$$

Así la compuerta Y es igual a XZ como se muestra a continuación:

$$Y = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}\begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} = \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix}$$

Aplicando esta compuerta a los qubits $\left|0\right\rangle$ y $\left|1\right\rangle$ obtendremos:

$$Y\left|1\right\rangle = \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix}\begin{pmatrix} 0 \\ 1 \end{pmatrix} = \begin{pmatrix} -1 \\ 0 \end{pmatrix} = -\left|0\right\rangle \qquad Y\left|0\right\rangle = \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix}\begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \end{pmatrix} = \left|1\right\rangle$$

3.5.5. Compuerta HADAMARD

La compuerta Hadamard [1], representada por H, trabaja para un qubit. La construcción de la compuerta Hadamard es la siguiente:

$$H = \frac{1}{\sqrt[2]{2}} \left[(|0\rangle + |1\rangle) \langle 0| + (|0\rangle - |1\rangle) \langle 1| \right]$$

$$H = \frac{1}{\sqrt[2]{2}} \left[(|0\rangle \langle 0| + |1\rangle \langle 0|) + (|0\rangle \langle 1| - |1\rangle \langle 1|) \right]$$

se sustituyen los valores:

$$H = \frac{1}{\sqrt[2]{2}} \left[\left(\begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix} + \begin{pmatrix} 0 & 0 \\ 1 & 0 \end{pmatrix} \right) + \left(\begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix} - \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix} \right) \right] = \frac{1}{\sqrt[2]{2}} \left[\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \right]$$

Esta compuerta la aplicaremos a los qubits de una dimensión:

$$H|0\rangle = \frac{1}{\sqrt[2]{2}} \left[\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \end{pmatrix} \right] = \frac{1}{\sqrt[2]{2}} \left[\begin{pmatrix} 1 \\ 1 \end{pmatrix} \right] = \frac{1}{\sqrt[2]{2}} \left[\begin{pmatrix} 1 \\ 0 \end{pmatrix} + \begin{pmatrix} 0 \\ 1 \end{pmatrix} \right] = \frac{1}{\sqrt[2]{2}} \left[|0\rangle + |1\rangle \right]$$

$$H|1\rangle = \frac{1}{\sqrt[2]{2}} \left[\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \begin{pmatrix} 0 \\ 1 \end{pmatrix} \right] = \frac{1}{\sqrt[2]{2}} \left[\begin{pmatrix} 1 \\ -1 \end{pmatrix} \right] = \frac{1}{\sqrt[2]{2}} \left[\begin{pmatrix} 1 \\ 0 \end{pmatrix} + \begin{pmatrix} 0 \\ -1 \end{pmatrix} \right] = \frac{1}{\sqrt[2]{2}} \left[|0\rangle - |1\rangle \right]$$

3.5.6. Compuerta XOR

Como se ha notado las compuertas anteriores son útiles para un qubit, ahora veremos las compuertas para dos y tres qubits.

La construccin del OR EXCLUSIVO es:

$$XOR = |0\rangle \langle 0| \otimes I + |1\rangle \langle 1| \otimes NOT$$

XOR: Si el primer qubit es 0 entonces el segundo qubits pasa en forma paralela. Si el primer qubit es 1 entonces aplica un NOT al último qubit.

$$XOR|00\rangle = |00\rangle$$
$$XOR|01\rangle = |01\rangle$$
$$XOR|10\rangle = |11\rangle$$
$$XOR|11\rangle = |10\rangle$$

La compuerta XOR en su forma matricial es:

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

Utilizando la siguiente regla

$$A \otimes B \left|q_1 q_2\right\rangle = A \otimes B \left|q_1\right\rangle \otimes \left|q_2\right\rangle = A \left|q_1\right\rangle \otimes B \left|q_2\right\rangle$$

se realizará la siguiente operación:

$$\begin{aligned} XOR \left|10\right\rangle &= \left(\left|0\right\rangle \left\langle 0\right| \otimes I + \left|1\right\rangle \left\langle 1\right| \otimes NOT\right) \left|10\right\rangle \\ &= \left(\left|0\right\rangle \left\langle 0\right| \otimes I + \left|1\right\rangle \left\langle 1\right| \otimes NOT\right) \left|1\right\rangle \otimes \left|0\right\rangle \\ &= \left(\left|0\right\rangle \left\langle 0\right| \otimes I\right) \left|1\right\rangle \otimes \left|0\right\rangle + \left(\left|1\right\rangle \left\langle 1\right| \otimes NOT\right) \left|1\right\rangle \otimes \left|0\right\rangle \\ &= \left(\left|0\right\rangle \left\langle 0\right|\right) \left|1\right\rangle \otimes I \left|0\right\rangle + \left(\left|1\right\rangle \left\langle 1\right|\right) \left|1\right\rangle \otimes NOT \left|0\right\rangle \end{aligned}$$

Dado que

$$\left\langle 0\right| \left|1\right\rangle = 0, \;\; 0 \otimes I \left|0\right\rangle = 0, \;\; \left\langle 1\right| \left|1\right\rangle = 1 \;\; y \;\; NOT \left|0\right\rangle = \left|1\right\rangle$$

tenemos

$$\begin{aligned} &= \left|1\right\rangle \otimes \left|1\right\rangle \\ &= \left|11\right\rangle \end{aligned}$$

3.5.7. Compuerta CCNOT

Su forma matemática es:

$$CCNOT = \left|0\right\rangle \left\langle 0\right| \otimes I \otimes I + \left|1\right\rangle \left\langle 1\right| \otimes NOT \otimes NOT$$

CCNOT: Si el primer qubit es 0 entonces los dos últimos qubits pasan paralelos. Si el primer qubit es 1 entonces se aplica un NOT al segundo qubit y también al tercer qubit.

$$\begin{aligned} CCNOT \left|000\right\rangle &= \left|000\right\rangle \\ CCNOT \left|001\right\rangle &= \left|001\right\rangle \\ CCNOT \left|010\right\rangle &= \left|010\right\rangle \\ CCNOT \left|011\right\rangle &= \left|011\right\rangle \\ CCNOT \left|100\right\rangle &= \left|111\right\rangle \\ CCNOT \left|101\right\rangle &= \left|110\right\rangle \\ CCNOT \left|110\right\rangle &= \left|101\right\rangle \\ CCNOT \left|111\right\rangle &= \left|100\right\rangle \end{aligned}$$

Compuerta CCNOT representada en matrices:

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \end{pmatrix}$$

3.5.8. Compuerta TOFFOLI

Su construcción es la siguiente:

$$TOFFOLI = |0\rangle\langle 0| \otimes I \otimes I + |1\rangle\langle 1| \otimes XOR$$

TOFFOLI: Si el primer qubit es 0 entonces los dos últimos qubits pasan paralelos. Si el primer qubit es 1 entonces se aplica un XOR a los dos últimos qubits.

$$\begin{aligned}
T|000\rangle &= |000\rangle \\
T|001\rangle &= |001\rangle \\
T|010\rangle &= |010\rangle \\
T|011\rangle &= |011\rangle \\
T|100\rangle &= |100\rangle \\
T|101\rangle &= |101\rangle \\
T|110\rangle &= |111\rangle \\
T|111\rangle &= |110\rangle
\end{aligned}$$

Compuerta TOFFOLI representada en matrices:

$$\begin{pmatrix}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0
\end{pmatrix}$$

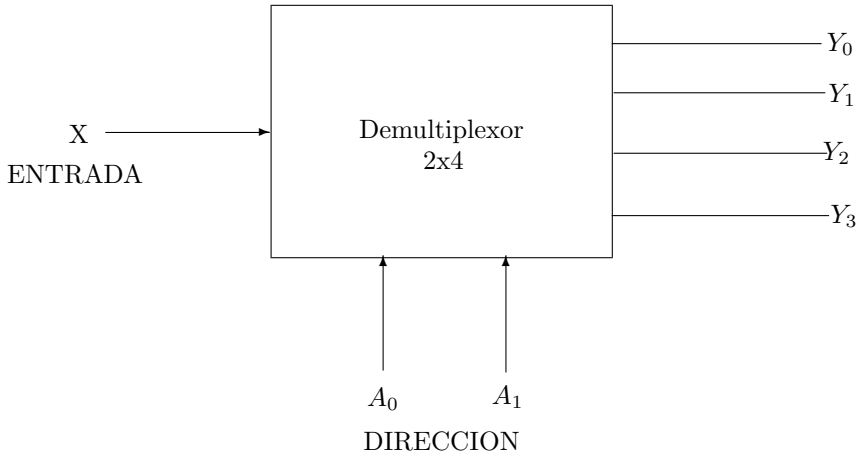
Capítulo 4

Aplicación de Fredkine en un Demultiplexor Cuántico

En esta parte se describirá mediante la Lógica Conservativa como un demultiplexor cuántico se puede obtener mediante la compuerta cuántica de Fredkine, gracias a que esta compuerta es reversible y conservativa. También se obtendrá la compuerta cuántica de Fredkine mediante el proceso de Shor y por último se desarrollará el demultiplexor cuántico y algunos algoritmos para su implementación del Lenguaje de Programación C++.

4.1. Demultiplexor Clásico

Un demultiplexor clásico es un circuito que recibe información por una sola línea y transmite esta información en una de las 2n líneas posibles de salida [6]. La selección de una línea de salida específica se controla por los valores de los n bits de dirección, como a continuación se muestra en la figura:



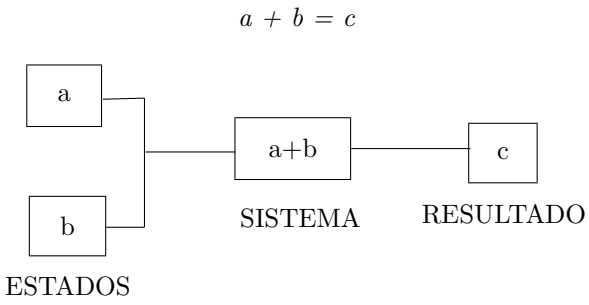
Notemos que n=2 corresponde a los bits de dirección A_0 y A_1, estos bits controlarán en que salida Y_0, Y_1, Y_2 y Y_3 se transmitirá el valor de X; como vemos las salidas Y_0, Y_1, Y_2 y Y_3 corresponden al valor $2^n = 2^2 = 4$.

La diferencia entre un demultiplexor clásico a un cuántico es que el último se obtiene basado en los principios de la física cuántica y para su implementación hacemos uso de la compuerta cuántica de Fredkine[3].

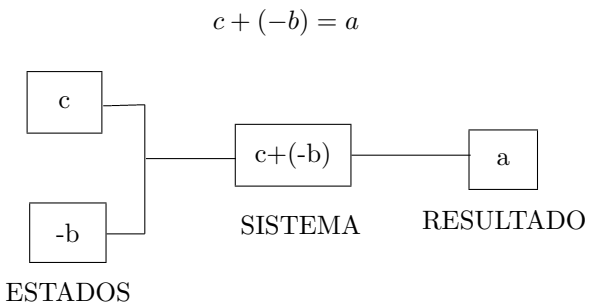
4.2. Reversibilidad

Desde un nivel microscópicamente determinista las leyes dinámicas, como son las leyes aplicadas a las físicas clásica y cuántica, son reversibles (Ver Fredkine y Toffoli [3]).

Un sistema reversible es aquel que puede ser determinísticamente dirigido hacia atrás. Por ejemplo, si se aplica un sistema a los estados a y b nos da como resultado el estado c, el estado b se almacenará, y puesto que es un solo dato ocupa una cantidad mínima de memoria, la reversibilidad se observa al aplicar el sistema al resultado c con el dato almacenado b y nos dará el estado original a:



Note que $a+b=c$ equivale a $a-(-b)=c$, por lo que:



La reversibilidad permite regresar a los estados originales a y b utilizando el mínimo almacenamiento de memoria y como se observa en el esquema anterior distintos estados iniciales siempre conducen a distintos estados finales.

Como podemos ver la reversibilidad de un sistema permite que sea invertible. Cada operador unitario que, por definición, le corresponde la condición $U^{\dagger}U = I$, para cada transformación U existe una transformación inversa $U^{\dagger}$. Como consecuencia el cálculo cuántico es reversible[10].

Cada operación reversible puede ser descrito por un operador unitario U el cual corresponde a la condición $U^{-1} = U^{\dagger}$. Las composiciones de operadores unitarios también son unitarias desde que $(UV)^{-1} = V^{\dagger}U^{\dagger}$[4].

4.3. Conservación

La conservación es una propiedad fundamental de los sistemas físicos. En todos los sistemas reversibles existe un número de cantidades que se conservan, esto es, existen determinadas funciones del estado de un sistema que permanecen constantes en el tiempo, en cualquier trayectoria u órbita. En muchos sistemas físicos, las leyes dinámicas poseen cierta simetría y en correspondencia con estas simetrías uno puede identificar un número de cantidades conservadas cuyo comportamiento es mejor debido a que son analíticos y lo que es más importante aditivos, es decir, estas cantidades son definidas por porciones individuales de un sistema y se adhieren contribuciones de partes diferentes, de ahí que son linealmente independientes. Un ejemplo de ello es el comportamiento de un vector.

De las simetrías antedichas, unas se derivan de la uniformidad del tiempo-espacio, y se expresan por la conservación de principios tales como la conservación de energía (homogeneidad de tiempo), el momento (homogeneidad de espacio), y el momento angular (isotopía de espacio.) Otras simetrías que no tienen un correlativo clásico se encuentran en la dinámica cuántica de partículas elementales[3].

4.4. Funcionalidad de la Compuerta Cuántica de Fredkine

La Compuerta de Fredkine es una Compuerta Lógico Conservativa. Una compuerta lógico conservativa es cualquier función Booleana que es reversible y conservativa. Se conoce bien que bajo las reglas ordinarias de composición de funciones (donde se permite el copiado), las dos entradas de la compuerta NAND constituyen una primitiva universal para el conjunto de todas las funciones Booleanas. En lógica conservativa, un papel análogo es ejecutado por una sola primitiva de señal de proceso, a saber, la compuerta de Fredkine, definida por la siguiente tabla [3]:

U	X_1	X_2		V	Y_1	Y_2
0	0	0		0	0	0
0	0	1		0	1	0
0	1	0		0	0	1
0	1	1	$\rightarrow$	0	1	1
1	0	0		1	0	0
1	0	1		1	0	1
1	1	0		1	1	0
1	1	1		1	1	1

Y gráficamente representado por la Fig.1(a). Este elemento de cálculo puede ser visualizado como un dispositivo que ejecute la intersección condicional de dos señales de datos de acuerdo al valor de la señal de control Fig.1(b). Cuando este valor es 1 dos señales de datos siguen caminos paralelos; cuando es 0, se cambian. De acuerdo al gráfico, observemos que la compuerta de Fredkine no tiene una salida lineal y coincide con su inverso.

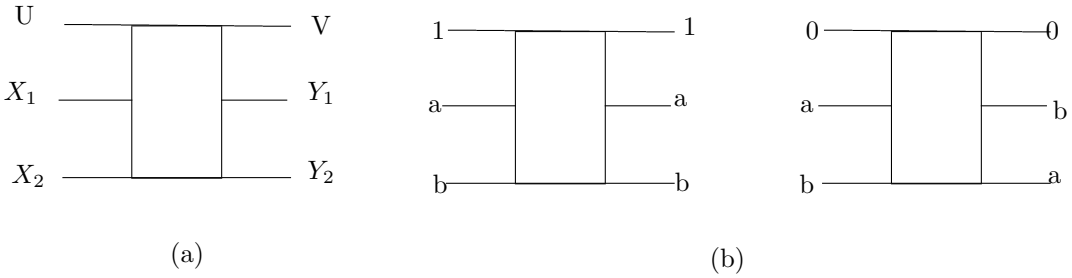
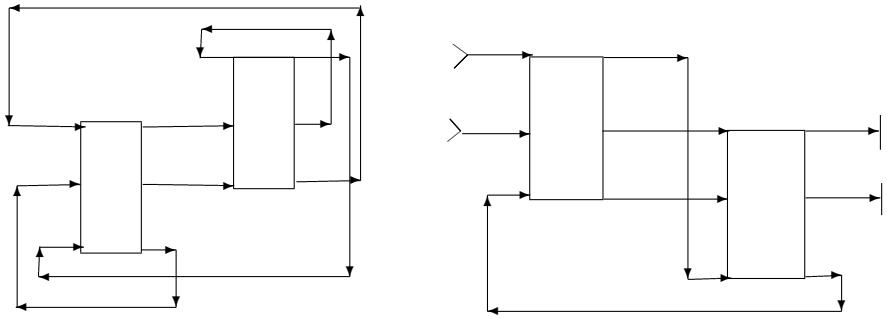


Fig.1. (a) Símbolo de la compuerta de Fredkine (b) operación de la compuerta de Fredkine.

En lógica conservativa, toda la señal del proceso se reduce por último para un encaminamiento condicional de señales. Más o menos hablando, las señales son tratadas como objetos inalterables que pueden ser movidos alrededor del curso de cálculo, pero nunca se crean o destruyen.

Circuitos de lógica conservativa. Finalmente, presentaremos un esquema para la conexión de señales, representados por unidades de alambre, con eventos, representados por las compuertas de lógica conservativa.

La representación de un circuito lógico conservativo es un gráfico dirigido cuyos nodos son compuertas lógico conservativas y cuyos arcos son los alambres de cualquier longitud. (Fig. 2.)



(a) (b)

Fig. 2. (a) Cerrado. (b) Abertura del circuito lógico conservativo.

Cualquier salida de una compuerta sólo se puede conectar a la entrada de un alambre, y de modo similar, cualquier entrada de una compuerta se puede conectar sólo a la salida de un alambre. La interpretación de tal circuito en términos de secuencia convencional de cálculo es inmediata, puesto que la compuerta juega el papel de un elemento combinacional 'instantáneo' y el alambre, el de un elemento de retraso integrado en línea de interconexión. En un circuito cerrado lógico conservativo, todas las entradas y las salidas de cualesquier elemento es conectado dentro del circuito (Fig.2(a)). Tal circuito corresponde a lo que en física se llaman un sistema cerrado (o aislado.) Un circuito abierto lógico conservativo posee un número de puertos de entrada y salida externos (Fig.2(b).) En el aislamiento, tal circuito se conocería como un transductor (típicamente con memoria) el cual, según su estado inicial, responderá con una secuencia de salida particular a cualquier secuencia de entrada particular. Sin embargo, por lo general tal circuito será una porción de una parte del largo circuito; de ahí la notación para puertos de entrada y salida (Fig.2(b)), que son respectivamente el rastreo y el borde principal de un alambre. Observe que en circuitos de lógica conservativa el número de puertos de salida siempre es igual que las entradas. La unión entre dos unidades de alambres adyacentes formalmente se puede tratar como un nodo que consiste de una compuerta trivialmente lógico conservativa, a saber, la compuerta identidad. En lo que sigue, siempre que hablemos de la posibilidad de una función en términos de cierto conjunto de primitivas de lógica conservativa, asumiremos tácitamente que la unidad de alambre y la compuerta identidad serán incluidas dentro de este conjunto.

Un circuito de lógica conservativa es un sistema discreto de tiempo dinámico. Las unidades de alambre representan las variables de estado individuales del sistema, mientras las compuertas (incluyendo, desde luego, cualquier presencia de la compuerta identidad) en conjunto representan la función de transición del sistema. El número N de las unidades de alambre que están presentes en el circuito pueden ser vistos como el número de grados de libertad del sistema. De estas N unidades, cualquier momento N_1 estarán en el estado 1, y las restantes $N_0(= N - N_1)$ estará en el estado 0. La cantidad N_1 es una función aditiva del estado del sistema, esto es, es definida para cualquier parte del circuito y su valor para el circuito entero es la suma de las contribuciones individuales de todas las porciones. Además, desde que ambas, la unidad de alambre y las compuertas, regresen en sus salidas tanto como 1's que están presentes en sus entradas, la cantidad N_1 es una integral del

movimiento del sistema, esto es, está constante a lo largo de cualquier trayectoria. (Consideraciones Análogas se aplican a la cantidad N_0, pero, desde luego, N_0 y N_1 no son integrales independientes del movimiento.) De este "principio de conservación"para las cantidades, en cuyas señales son codificadas, se deriva el nombre de Lógica Conservativa.

Computación en circuitos de lógica conservativa; constantes y basura: En la Fig.3(a) se han expresado las variables de salida de la compuerta de Fredkine como las funciones explícitas de las variables de entrada. La relación total funcional entre la entrada y la salida es, como hemos visto, reversible. Por otra parte, las funciones que uno desea calcular a menudo son no reversibles. Así, provisiones especiales deben ser hechas en el empleo de la compuerta de Fredkine (o, en realidad, de cualquier función reversible que se proponga para que sea una primitiva de uso general de procesado de señales) para obtener el poder adecuado de cálculo.

Supongamos, por un instante, que uno desea calcular la función AND, que es no reversible. En la Fig.3(b) sólo las entradas u y x_1 son alimentados con los valores arbitrarios a y b, mientras x_2 es alimentado con el valor constante 0. En este caso, la salida y_1 proporcionará el valor deseado ab ('a AND b'), mientras las otras dos salidas v y y_2 cederán los valores 'no solicitados a y $\ddot{a}b$'. Así, intuitivamente, la función AND se puede realizar mediante la compuerta de Fredkine mientras uno este dispuesto en suministrar 'constantes' a esta compuerta al lado del argumento, y aceptar la 'basura' de este lado con el resultado.

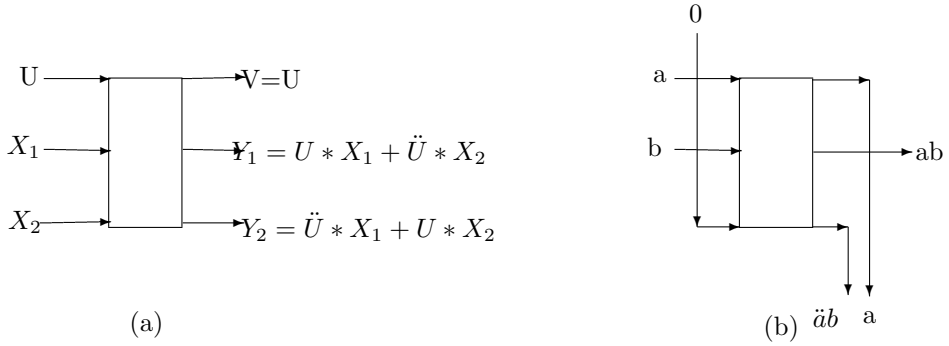


Fig. 3. Comportamiento de la compuerta de Fredkine (a) con entradas libres, y (b) con X_2 sea el valor 0, así comprendiendo la función AND.

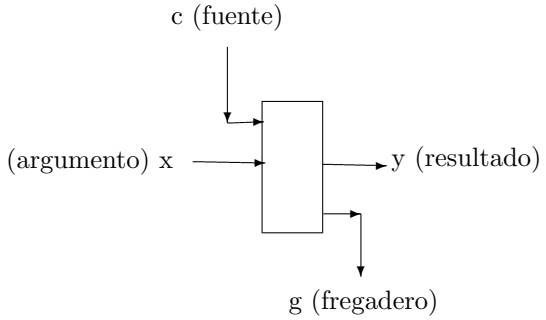


Fig. 4. Realización de f por ϕ usando fuentes 'c' y caño o fregadero 'g'. La función $\phi : \langle c, x\rangle \mapsto \langle y, g\rangle$ se escoge para un valor particular de c, y= f(x)

Terminología: fuente, caño, constantes (o argumentos), resultado. Dada cualquier función finita fi, obtenemos una nueva función f 'encajada' a este por la asignación de valores específicos a ciertas líneas de entrada distinguidas, en conjunto conocido como fuente, y desatendiendo ciertas líneas de salida distinguidas, en conjunto conocidas como desecho de basura. Las líneas de entrada constituirán los argumentos, y las líneas de salida restantes, el resultado. A esta construcción (Fig.4) se le llama una realización de f mediante ϕ usando la fuente y el caño. Para realizar f mediante ϕ, en la que ϕ es una función finita como OR, NOT y FAN-OUT, las líneas de fuente se alimentarán con valores constantes, esto es, con valores que no dependen del argumento. De otra forma, las líneas del caño en general cederán los valores, que dependen del argumento, y así no pueden ser usados como constantes de entrada para un cálculo nuevo. Tales valores serán conocidos como basura[2].

Por la selección apropiada de líneas de fuentes y caño es posible obtener de la compuerta de Fredkine otras funciones elementales Booleanas, como OR, NOT, y FAN-OUT (Fig.5.) Para sintetizar funciones más complejas necesitamos circuitos que contengan varias ocurrencias de la compuerta de Fredkine. Por ejemplo, en el siguiente capítulo, en la Figura 6 se ilustra un demultiplexor de 1 línea a 4 líneas. Por las tardanzas representadas por los alambres, esto es formalmente una red secuencial. Sin embargo, ya que ninguna regeneración está presente y todos los caminos del argumento al resultado atraviesan el mismo número de unidades de alambre, el análisis de este circuito es considerablemente idéntico a la red combinacional.

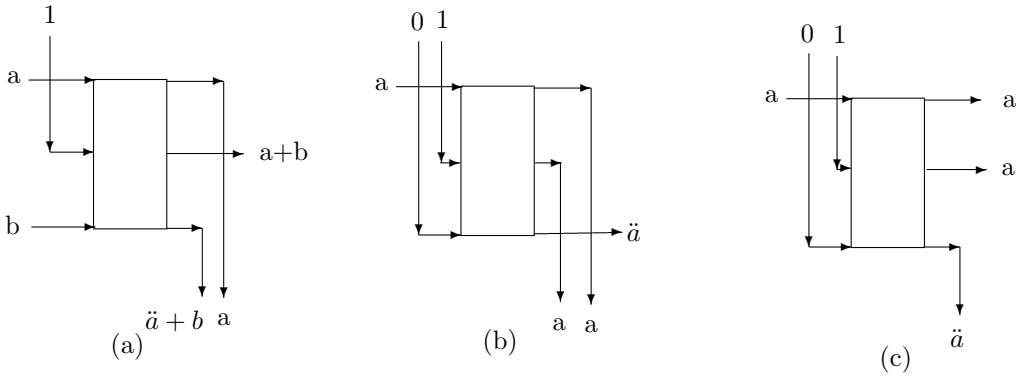


Fig. 5 Realización de las funciones (a) OR , (b) NOT y (c) FAN-OUT mediante la compuerta de Fredkine.

4.5. Desarrollo de la Compuerta Cuántica de Fredkine

Construcción matemática de Fredkine

$$fredkine = |0\rangle\langle 0| \otimes |0\rangle\langle 0| \otimes |0\rangle\langle 0| + |0\rangle\langle 0| \otimes |0\rangle\langle 1| \otimes |1\rangle\langle 0| +$$

$$|0\rangle\langle 0| \otimes |1\rangle\langle 0| \otimes |0\rangle\langle 1| + |0\rangle\langle 0| \otimes |1\rangle\langle 1| \otimes |1\rangle\langle 1| + |1\rangle\langle 1| \otimes I \otimes I$$

La compuerta cuántica de Fredkine opera sobre los qubits $|000\rangle$, $|001\rangle$, $|010\rangle$, $|011\rangle$, $|100\rangle$, $|101\rangle$, $|110\rangle$ y $|111\rangle$. La propiedad de esta compuerta es que si el primer qubit es 0 entonces los dos últimos qubits se intercambian. Si el primer qubit es 1 entonces no sufren alteración los dos últimos qubits, el efecto de la compuerta de Fredkine es la siguiente:

$$\begin{array}{ccc} a & & b \\ |000\rangle & & |000\rangle \\ |001\rangle & & |010\rangle \\ |010\rangle & & |001\rangle \\ |011\rangle & \rightarrow & |011\rangle \\ |100\rangle & & |100\rangle \\ |101\rangle & & |101\rangle \\ |110\rangle & & |110\rangle \\ |111\rangle & & |111\rangle \end{array}$$

Para ver la aplicación de FREDKINE sobre el qubit $|001\rangle$ tenemos:

$= (|0\rangle\langle 0| \otimes |0\rangle\langle 0| \otimes |0\rangle\langle 0|)|001\rangle + (|0\rangle\langle 0| \otimes |0\rangle\langle 1| \otimes |1\rangle\langle 0|)|001\rangle + (|0\rangle\langle 0| \otimes |1\rangle\langle 0| \otimes |0\rangle\langle 1|)|001\rangle + (|0\rangle\langle 0| \otimes |1\rangle\langle 1| \otimes |1\rangle\langle 1|)|001\rangle + (|1\rangle\langle 1| \otimes I \otimes I)|001\rangle$

$= |0\rangle\langle 0|\,|0\rangle \otimes |0\rangle\langle 0|\,|0\rangle \otimes |0\rangle\langle 0|\,|1\rangle + |0\rangle\langle 0|\,|0\rangle \otimes |0\rangle\langle 1|\,|0\rangle \otimes |1\rangle\langle 0|\,|1\rangle + |0\rangle\langle 0|\,|0\rangle \otimes |1\rangle\langle 0|\,|0\rangle \otimes |0\rangle\langle 1|\,|1\rangle + |0\rangle\langle 0|\,|0\rangle \otimes |1\rangle\langle 1|\,|0\rangle \otimes |1\rangle\langle 1|\,|1\rangle + |1\rangle\langle 1|\,|0\rangle \otimes I\,|0\rangle \otimes I\,|1\rangle$

Dado que $\langle 0|\,|0\rangle = 1$, $\langle 0|\,|1\rangle = 0$, $\langle 1|\,|0\rangle = 0$, $\langle 1|\,|1\rangle = 1$, $I\,|0\rangle = |0\rangle$ y $I\,|1\rangle = |1\rangle$ entonces

$$= |0\rangle \otimes |1\rangle \otimes |0\rangle$$

$$= |010\rangle$$

Para desarrollar la compuerta se utiliza el método de Shor el cual establece que en la intersección de entrada a, que son los qubits de la fila, y la salida b, que son los qubits de la columna, se coloca el valor 1 como se muestra a continuación:

FREDKINE	$\|000\rangle$	$\|001\rangle$	$\|010\rangle$	$\|011\rangle$	$\|100\rangle$	$\|101\rangle$	$\|110\rangle$	$\|111\rangle$
$\|000\rangle$	1	0	0	0	0	0	0	0
$\|001\rangle$	0	0	1	0	0	0	0	0
$\|010\rangle$	0	1	0	0	0	0	0	0
$\|011\rangle$	0	0	0	1	0	0	0	0
$\|100\rangle$	0	0	0	0	1	0	0	0
$\|101\rangle$	0	0	0	0	0	1	0	0
$\|110\rangle$	0	0	0	0	0	0	1	0
$\|111\rangle$	0	0	0	0	0	0	0	1

Su transformación unitaria es:

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

4.6. Demultiplexor Cuántico

Esta es una función más compleja debido a que este demultiplexor cuántico está compuesto por 3 compuertas cuánticas de Fredkine; también recibe información por una sola línea de entrada (valor X) y transmite esta información en una de las 2^2 líneas posibles de salida (valores Y_0, Y_1, Y_2, Y_3.), ya que está controlado por dos valores de dirección.

Para la implementación del Demultiplexor Cuántico [3], utilizando Compuertas Cuánticas de Fredkine, tendremos 2 direcciones de selección A_0 y A_1, con una entrada X y salidas Y_0, Y_1, Y_2 y Y_3 como se muestra a continuación:

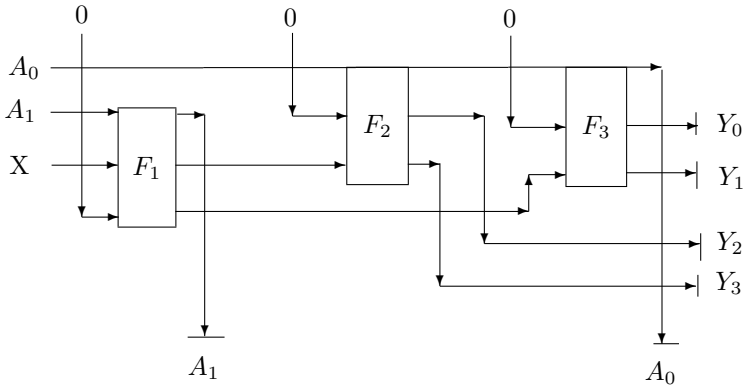
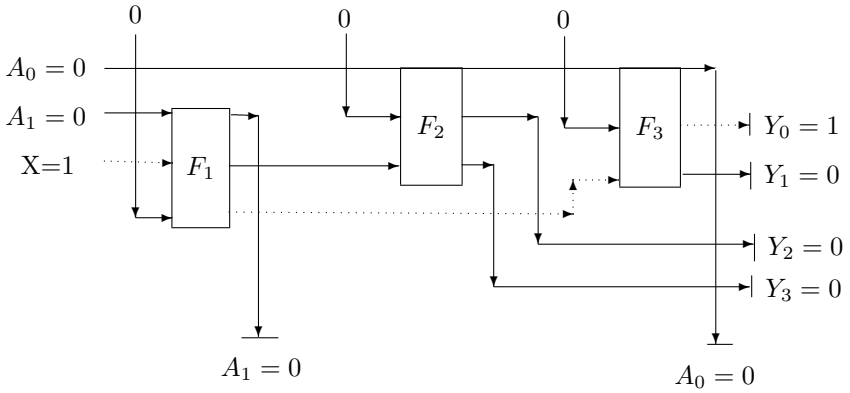


Fig.6 Demultiplexor de 1 línea a 4 líneas. Las líneas de "dirección" A_0, A_1 especifican cuál de los cuatro datos de salida $Y_0, ..., Y_3$ señalan donde se rotará la señal de dato X. (Note que aquí las líneas de fregadero pasan de repetir las líneas de dirección.)

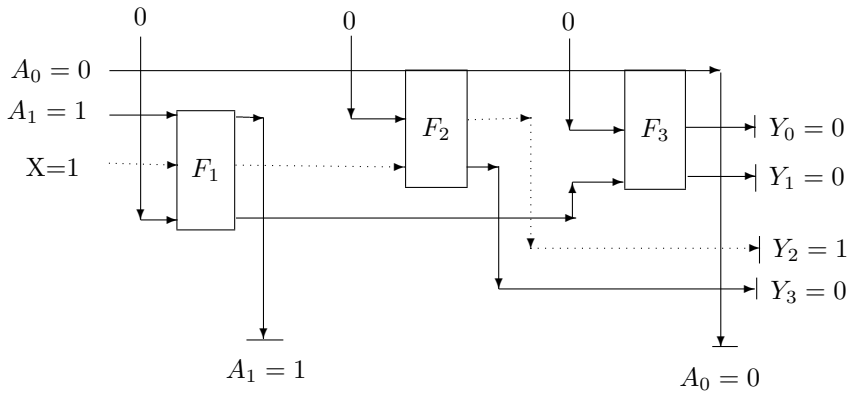
Veamos la funcionalidad del demultiplexor cuántico asignando las siguientes direcciones y con una entrada X = 1.

A_0	A_1
0	0
0	1
1	0
1	1

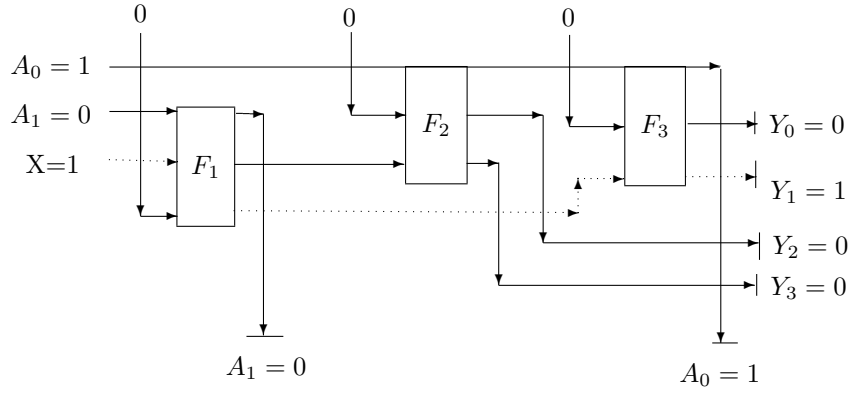
Con la dirección $A_0 = 0$ y $A_1 = 0$ nos manda el valor X = 1 a la salida Y_0



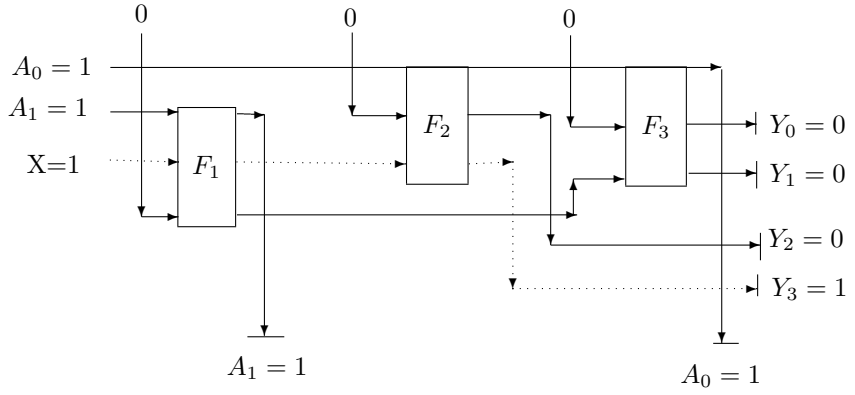
Con la dirección $A_0 = 0$ y $A_1 = 1$ nos manda el valor X = 1 a la salida Y_2



Con la dirección $A_0 = 1$ y $A_1 = 0$ nos manda el valor X = 1 a la salida Y_1



Con la dirección $A_0 = 1$ y $A_1 = 1$ nos manda el valor X = 1 a la salida Y_3

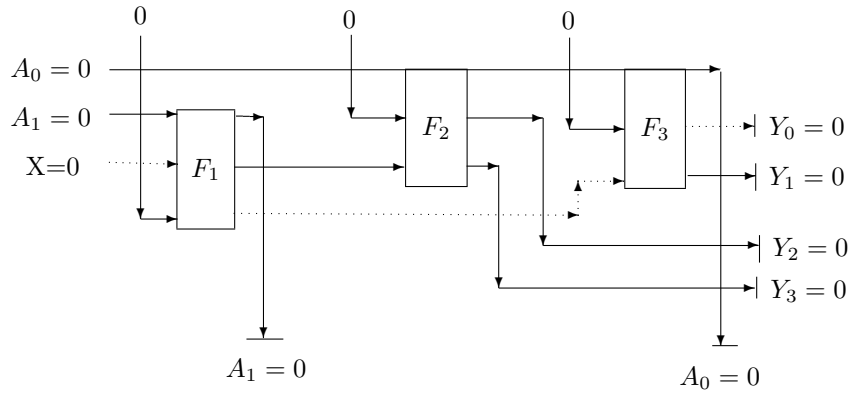


Colocando en una tabla las direcciones A_0 y A_1 para el dato de entrada X = 1 tendremos los siguientes resultados del demultiplexor:

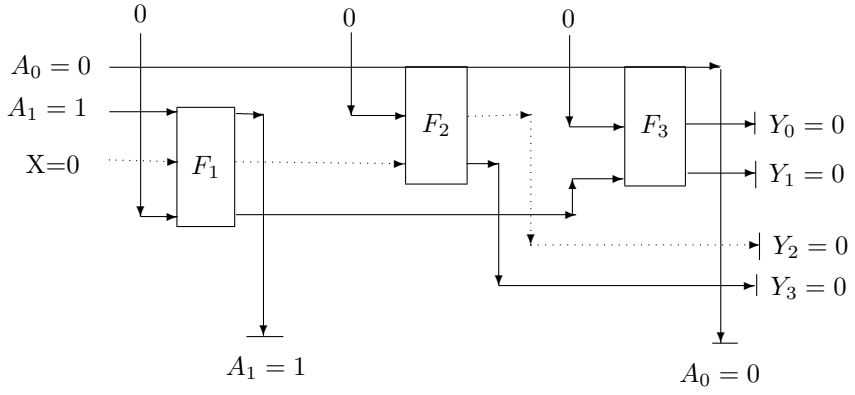
A_0	A_1	Y_0	Y_1	Y_2	Y_3
0	0	1	0	0	0
0	1	0	0	1	0
1	0	0	1	0	0
1	1	0	0	0	1

Para el valor X = 0 sólo seguiremos su trayectoria:

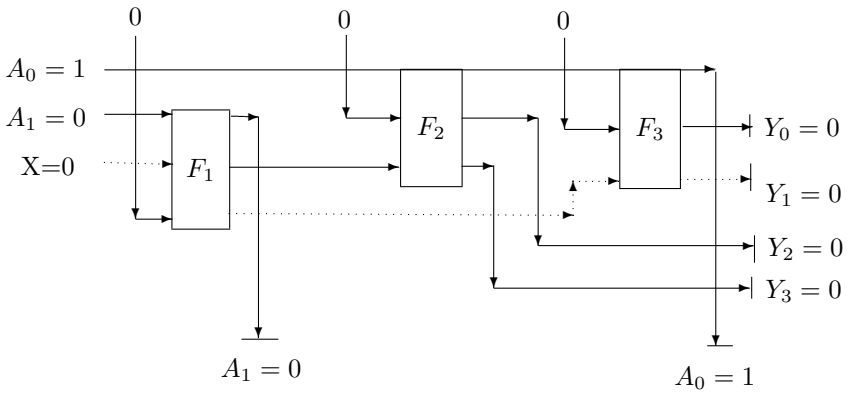
Con la dirección $A_0 = 0$ y $A_1 = 0$ nos manda el valor X = 0 a la salida Y_0



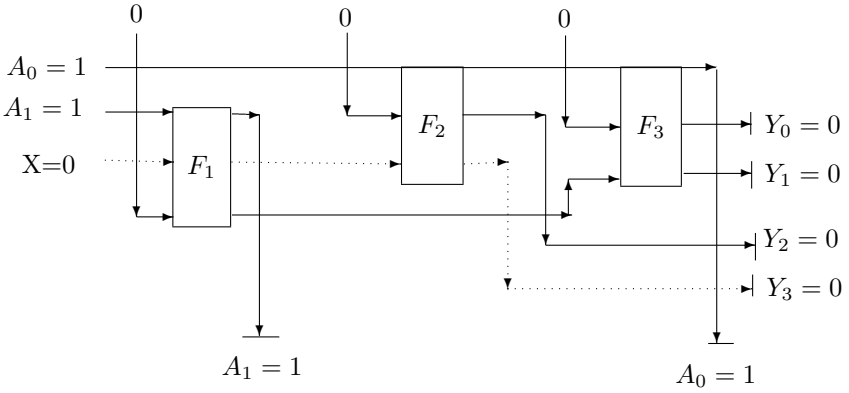
Con la dirección $A_0 = 0$ y $A_1 = 1$ nos manda el valor X = 0 a la salida Y_2



Con la dirección $A_0 = 1$ y $A_1 = 0$ nos manda el valor X = 0 a la salida Y_1



Con la dirección $A_0 = 1$ y $A_1 = 1$ nos manda el valor X = 0 a la salida Y_3



4.6.1. Desarrollo del Demultiplexor Cuántico con Transformaciones Unitarias

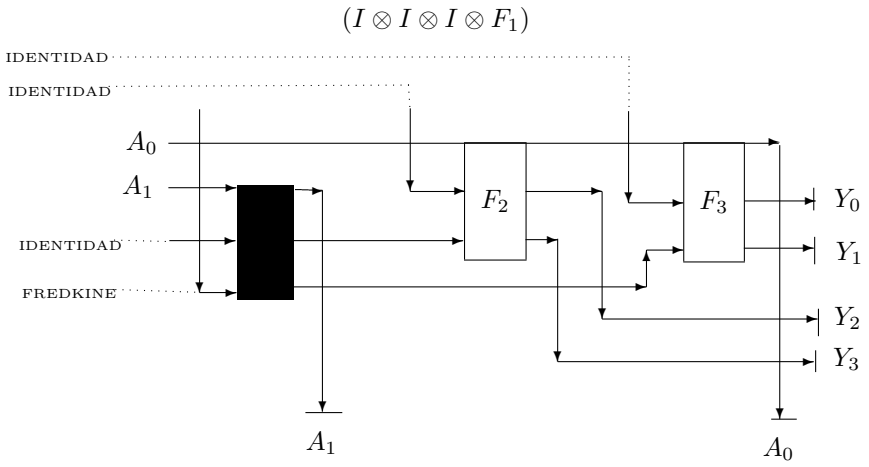
El demultiplexor cuántico se obtiene mediante tres matrices de dimensión 64×64, cada matriz tiene los siguientes productos tensoriales $(I \otimes I \otimes I \otimes F_1), (I \otimes F_2 \otimes I \otimes I)$ y $(F_3 \otimes I \otimes I \otimes I)$; notemos que en cada matriz de 64×64 tenemos una compuerta cuántica de Fredkine y 3 compuertas cuánticas Identidad, a continuación analizaremos como obtener cada una de las matrices.

El demultiplexor cuántico tiene 3 constantes $|0\rangle, |0\rangle$ y $|0\rangle$ que entran en su correspondiente compuerta de Fredkine. Como se muestran en las ilustraciones anteriores, la primera constante permite el acceso a la compuerta de Fredkine F_1 y así se obtiene la primera matriz $(I \otimes I \otimes I \otimes F_1)$, la segunda constante permite la entrada de la segunda compuerta de Fredkine F_2 que con las identidades correspondientes tenemos la matriz $(I \otimes F_2 \otimes I \otimes I)$ y por último la constante $|0\rangle$ entra en la compuerta de Fredkine F_3 con la cual obtenemos la última matriz $(F_3 \otimes I \otimes I \otimes I)$.

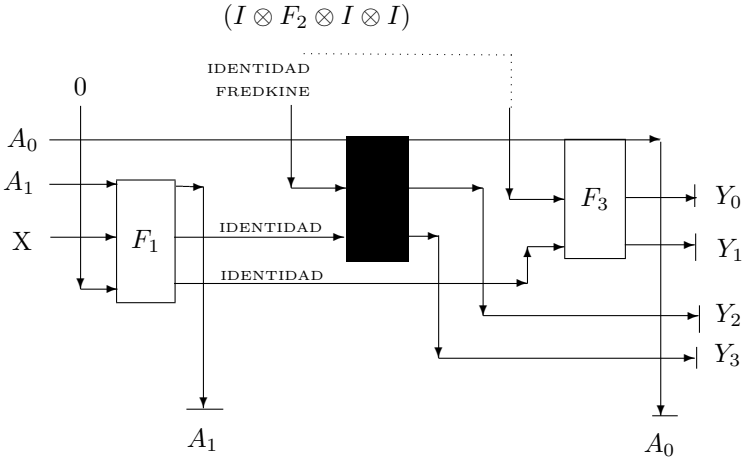
Como se puede ver las constantes de entrada $|0\rangle, |0\rangle$ y $|0\rangle$ permiten obtener las matrices $(I \otimes I \otimes I \otimes F_1), (I \otimes F_2 \otimes I \otimes I)$ y $(F_3 \otimes I \otimes I \otimes I)$ de dimensión 64×64.

Pata obtener las compuertas identidad y fredkine no se toman en cuenta las entradas de las direcciones A_0 Y A_1, sólo las constantes $|0\rangle, |0\rangle$ y $|0\rangle$ y la entrada X. En las siguientes líneas se analizará e ilustrará cada una de ellas.

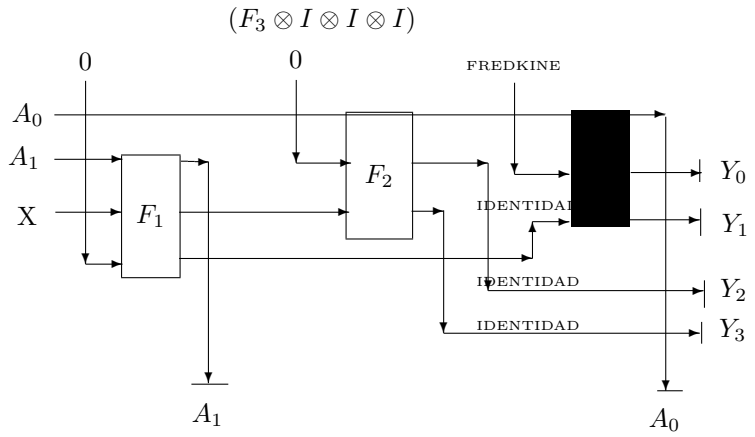
El producto tensorial $(I \otimes I \otimes I \otimes F_1)$ se obtiene de la siguiente forma: la línea de la constante $|0\rangle$ que entra en la compuerta F_3 se sustituye por compuerta identidad de dimensión 2×2, la línea de la constante $|0\rangle$ que entra en la compuerta F_2 se sustituye por compuerta identidad 2×2, la línea del valor de X que entra en la compuerta F_1 se sustituye por compuerta identidad 2×2 y la línea que representa a la primera constante $|0\rangle$ que entra en la primera compuerta de Fredkine F_1 se sustituye por la primera compuerta de Fredkine de dimensión 8×8. Los productos tensoriales darán como resultado una compuerta de dimensión 64×64, que es el producto de las dimensiones de todas las matrices $2 * 2 * 2 * 8$, veamos la siguiente ilustración:



El producto tensorial $(I \otimes F_2 \otimes I \otimes I)$ se obtiene de la siguiente forma: la línea que representa la segunda constante $|0\rangle$ que pasa a la compuerta F_3 se sustituirá por una compuerta identidad de dimensión 2×2, la línea que representa a la constante $|0\rangle$ que entra a la segunda compuerta F_2 se le sustituirá por la compuerta de Fredkine de dimensión 8×8, las dos últimas líneas que entran a las compuertas F_2 y F_3 se sustituyen por compuertas identidad de dimensión 2×2. Los productos tensoriales darán como resultado una compuerta de dimensión 64×64, que es el producto de las dimensiones de todas las matrices $2 * 8 * 2 * 2$, veamos la siguiente ilustración:



El producto tensorial $(F_3 \otimes I \otimes I \otimes I)$ se obtiene de la siguiente forma: la línea que representa la tercera constante $|0\rangle$ está conectada a la tercera compuerta F_3, se le sustituirá por la compuerta de Fredkine de dimensión 8×8, las siguientes 3 líneas de las salidas Y_1, Y_2 y Y_3 se les sustituirá por compuertas identidad de dimensión 2×2. Los productos tensoriales darán como resultado una compuerta de dimensión 64×64, que es el producto de las dimensiones de todas las matrices $8*2*2*2$, como se muestra a continuación:



4.7. Método para construir una Compuerta Unitaria (Método de Shor)

Al aplicarse una transformación unitaria a un qubit (entrada) nos da una salida. Cada transformación unitaria tiene sus propiedades, por ejemplo, la compuerta cuántica NOT aplicada a 1 qubit, que pueden ser $|0\rangle$ o $|1\rangle$, tiene la propiedad de que si se aplica al qubit $|0\rangle$ obtendremos como resultado $|1\rangle$, y viceversa.

A continuación veremos la compuerta cuántica NOT aplicada a una entrada y nos da como resultado la salida correspondiente.

$$\begin{array}{ccc} ENTRADA & & SALIDA \\ |0\rangle & \rightarrow & |1\rangle \\ |1\rangle & & |0\rangle \end{array}$$

El método de Shor consiste en hallar una matriz de $n \times n$, colocar un valor en la intersección de las entradas (filas) y salidas (columnas), dicho valor es de acuerdo a la propiedad de la transformación unitaria. La dimensión de la transformación unitaria se obtendrá evaluando $n = 2^m$ en el que m es la dimensión del qubit, por ejemplo, en la compuerta cuántica NOT aplicada a un qubit de dimensión $m = 1$, el valor de n será de dimensión $n = 2^1 = 2$, por lo que la matriz unitaria será de dimensión 2×2.

El número de los elementos fila y columna de la matriz son de acuerdo a la dimensión del qubit, por ejemplo, si el qubit es de dimensión 1 entonces los elementos de la matriz son los estados $|0\rangle$ y $|1\rangle$ puesto que la dimensión de la matriz es $n = 2^1 = 2$. De acuerdo a la propiedad de la compuerta unitaria se coloca el valor 1 en la intersección de la columna y fila correspondiente quedando de la siguiente forma:

NOT	$\|0\rangle$	$\|1\rangle$
$\|0\rangle$	0	1
$\|1\rangle$	1	0

Otro ejemplo es la Compuerta Unitaria XOR de 2 qubit, cuya dimensión será $n = 2^2 = 4$, dicha compuerta tiene la propiedad de que si el primer qubit es 0 entonces el segundo qubit pasa en forma paralela. Si el primer qubit es 1 encontes aplica un NOT al segundo qubit, ejemplo:

$$\begin{array}{ccc} ENTRADA & & SALIDA \\ |00\rangle & & |00\rangle \\ |01\rangle & \rightarrow & |01\rangle \\ |10\rangle & & |11\rangle \\ |11\rangle & & |10\rangle \end{array}$$

Como podremos ver los elementos de la matriz son $|00\rangle, |01\rangle, |10\rangle$ y $|11\rangle$. Por último, la matriz unitaria queda de la siguiente forma:

XOR	$\|00\rangle$	$\|01\rangle$	$\|10\rangle$	$\|11\rangle$
$\|00\rangle$	1	0	0	0
$\|01\rangle$	0	1	0	0
$\|10\rangle$	0	0	0	1
$\|11\rangle$	0	0	1	0

La dimensión de la transformación unitaria variará de acuerdo a la dimensión del qubit como es el caso de la compuerta de Fredkine que se aplica a un qubit de dimensión m = 3, por lo que la matriz será de dimensión $n = 2^3 = 8$. Los elementos de las filas serán $|000\rangle$, $|001\rangle$, $|010\rangle$, $|011\rangle$, $|100\rangle$, $|101\rangle$, $|110\rangle$, $|111\rangle$

La propiedad que se aplica a esta compuerta es que si el primer qubit es 0 entonces los dos últimos qubits se intercambian. Si el primer qubit es 1 entonces no sufren alteración los dos últimos qubits, su efecto es el siguiente:

$$\begin{array}{ccc} ENTRADA & & SALIDA \\ |000\rangle & & |000\rangle \\ |001\rangle & & |010\rangle \\ |010\rangle & & |001\rangle \\ |011\rangle & \rightarrow & |011\rangle \\ |100\rangle & & |100\rangle \\ |101\rangle & & |101\rangle \\ |110\rangle & & |110\rangle \\ |111\rangle & & |111\rangle \end{array}$$

Su representación mediante el método de Shor es el siguiente:

FREDKINE	$\|000\rangle$	$\|001\rangle$	$\|010\rangle$	$\|011\rangle$	$\|100\rangle$	$\|101\rangle$	$\|110\rangle$	$\|111\rangle$
$\|000\rangle$	1	0	0	0	0	0	0	0
$\|001\rangle$	0	0	1	0	0	0	0	0
$\|010\rangle$	0	1	0	0	0	0	0	0
$\|011\rangle$	0	0	0	1	0	0	0	0
$\|100\rangle$	0	0	0	0	1	0	0	0
$\|101\rangle$	0	0	0	0	0	1	0	0
$\|110\rangle$	0	0	0	0	0	0	1	0
$\|111\rangle$	0	0	0	0	0	0	0	1

Con estos datos tenemos el algoritmo siguiente:
1.- Se obtiene la dimensión m del qubit entrada.
2.- Inicializar la matriz $n \times n$ con ceros.
3.- Se pedirá los n elementos de las filas (entradas)
4.- Dar la propiedad de la transformación unitaria(en el qubit entrada, dar la salida.)
5.- De la primera fila hasta la fila 2^m, hacer lo siguiente:
Si el qubit cumple la propiedad entonces: Realizar las asignaciones necesarias. Se obtienen los índices de binario a decimal, de la fila y columna. El algoritmo de conversión se realiza de la siguiente forma:

0	1	1
2^2	2^1	2^0
0	2	1
Q[0]	Q[1]	Q[2]

De acuerdo a los índices en decimal de todas las filas y las columnas poner el valor 1 en la matriz, esto es, obtener la compuerta unitaria de acuerdo a sus propiedades.

Aplicando este algoritmo a la compuerta cuántica de Fredkine tenemos:
1.- Obtener la dimensión del qubit entrada, m = 3
2.- Inicializar la matriz de $2^m \times 2^m$ con ceros
De $i \leftarrow 0$ hasta $2^m - 1$
    De $j \leftarrow 0$ hasta $2^m - 1$
        $fredkine[i][j] \leftarrow 0$
3.- Los elementos de las filas serán:

$qfil[0][m] = |000\rangle, qfil[1][m] = |001\rangle, qfil[2][m] = |010\rangle, qfil[3][m] = |011\rangle, qfil[4][m] = |100\rangle, qfil[5][m] = |101$

4,5.- De $i \leftarrow 0$ hasta $2^m - 1$ filas realizar lo siguiente:

Si qfil[i][0] = 0 entonces //si el primer elemento es cero
    $qcol[0] \leftarrow qfil[i][0]$
    $qcol[1] \leftarrow qfil[i][2]$
    $qcol[2] \leftarrow qfil[i][1]$
Si no entonces //si el primer elemento es uno
    $qcol[0] \leftarrow qfil[i][0]$
    $qcol[1] \leftarrow qfil[i][1]$
    $qcol[2] \leftarrow qfil[i][2]$
FinSi
Obtener índices de binario a decimal de la fila (qfil) y la columna (qcol)
ind = 0
De j = 0 hasta que $j \prec= (m-1)$
    Si q[j] = 1 entonces
        $ind = 2^{((m-1)-j)} + ind$
    FinSi
    Incrementar j
FinDe
Obtener la compuerta unitaria mandando los índices en decimal de la fila y la columna:
fredkine[indfil][indcol] = 1
Incrementar i
FinDe

A continuación se mostrará una tabla con el contenido de los índices y los valores correspondientes, esto para entender mejor el algoritmo.

FREDKINE	$\|000\rangle$	$\|001\rangle$	$\|010\rangle$	$\|011\rangle$	$\|100\rangle$	$\|101\rangle$	$\|110\rangle$	$\|111\rangle$
$\|000\rangle$	F[0][0]=1	F[0][1]=0	F[0][2]=0	F[0][3]=0	F[0][4]=0	F[0][5]=0	F[0][6]=0	F[0][7]=0
$\|001\rangle$	F[1][0]=0	F[1][1]=0	F[1][2]=1	F[1][3]=0	F[1][4]=0	F[1][5]=0	F[1][6]=0	F[1][7]=0
$\|010\rangle$	F[2][0]=0	F[2][1]=1	F[2][2]=0	F[2][3]=0	F[2][4]=0	F[2][5]=0	F[2][6]=0	F[2][7]=0
$\|011\rangle$	F[3][0]=0	F[3][1]=0	F[3][2]=0	F[3][3]=1	F[3][4]=0	F[3][5]=0	F[3][6]=0	F[3][7]=0
$\|100\rangle$	F[4][0]=0	F[4][1]=0	F[4][2]=0	F[4][3]=0	F[4][4]=1	F[4][5]=0	F[4][6]=0	F[4][7]=0
$\|101\rangle$	F[5][0]=0	F[5][1]=0	F[5][2]=0	F[5][3]=0	F[5][4]=0	F[5][5]=1	F[5][6]=0	F[5][7]=0
$\|110\rangle$	F[6][0]=0	F[6][1]=0	F[6][2]=0	F[6][3]=0	F[6][4]=0	F[6][5]=0	F[6][6]=1	F[6][7]=0
$\|111\rangle$	F[7][0]=0	F[7][1]=0	F[7][2]=0	F[7][3]=0	F[7][4]=0	F[7][5]=0	F[7][6]=0	F[7][7]=1

4.8. Algoritmo para construir productos tensoriales

Producto tensorial: Es el producto de cada elemento de la matriz A por toda la matriz B, también pueden ser vectores.

Para ilustrarlo tenemos:

$$A = \begin{pmatrix} a_{00} & a_{01} \\ a_{10} & a_{11} \end{pmatrix} \qquad B = \begin{pmatrix} b_{00} & b_{01} \\ b_{10} & b_{11} \end{pmatrix}$$

$$A \otimes B = \begin{pmatrix} a_{00}B & a_{01}B \\ a_{10}B & a_{11}B \end{pmatrix}$$

El producto tensorial tiene las siguientes propiedades:

- $A \otimes B \neq B \otimes A$ No es conmutativa
- $(A \otimes B) \otimes C = A \otimes (B \otimes C)$ Si es asociativa

Si A , B y C son vectores qubit entonces se obtendrá como resultado un vector qubit.

Si A, B y C son matrices cuadradas entonces se obtendrá como resultado una matriz cuadrada, no importando la dimensión de la matriz, por ejemplo, si la matriz A es de dimensión 2×2 y la matriz B es de dimensión 3×3, entonces $A \otimes B$ será de dimensión 6×6 como resultado del producto de $2 * 3$. En el caso del producto tensorial de dos vectores, si la dimensión del vector a es 4×1 y la dimensión del vector b es 2×1 entonces el vector resultante será de dimensión 8×1 que se obtiene multiplicando la dimensión de las filas de los vectores $(4 * 2)$ y también la dimensión de las columnas$(1 * 1)$.

El producto de 2 o más qubits se puede representar de la siguiente forma:

$$|uv\rangle \equiv |u\rangle\,|v\rangle \equiv |u\rangle \otimes |v\rangle$$

por ejemplo, el producto tensorial del qubit de dimensiones 3 es el siguiente:

$$|010\rangle = |0\rangle \otimes |1\rangle \otimes |0\rangle$$

con

$$|0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \qquad y \qquad |1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

tenemos

$$|010\rangle = \begin{pmatrix} 1\begin{pmatrix} 0 \\ 1 \end{pmatrix} \\ 0\begin{pmatrix} 1 \\ 0 \end{pmatrix} \end{pmatrix} \otimes |0\rangle = \begin{pmatrix} 0\begin{pmatrix} 1 \\ 0 \end{pmatrix} \\ 1\begin{pmatrix} 1 \\ 0 \end{pmatrix} \\ 0\begin{pmatrix} 1 \\ 0 \end{pmatrix} \\ 0\begin{pmatrix} 1 \\ 0 \end{pmatrix} \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

Algoritmo que transforma un qubit de dimensión n a vector mediante productos tensoriales:
1.- Leer el qubit
2.- Calcular su dimensión
3.- De k=0 hasta $k \prec dim - 1$ //dim es la dimensión del qubit
$dim2 \leftarrow dim2 * 2$ //dim2 evalúa el tamaño del vector resultante
Si k=0 entonces //asignación para el primer elemento del qubit
Si vector[k]=0 entonces
$a \leftarrow |0\rangle$
si no entonces
$a \leftarrow |1\rangle$
FinSi
FinSi
Si $k \succ= 1$ entonces //para un producto ya hecho se asigna a $a \leftarrow aux$
$a \leftarrow aux$
FinSi
Si vector[k+1]=0 entonces $b = |0\rangle$ //asignación para el siguiente elemento del qubit
Si no entonces $b = |1\rangle$
FinSi
De $i \leftarrow 0$ hasta $i \prec dim2$ //producto tensorial
$elem \leftarrow a[i]$
De $j \leftarrow 0$ hasta $j \prec 2$
$aux[j + (i * 2))] = elem * b[j]$
Incrementa j
Incrementa i
Incrementa k

Explicación: Se leerá el qubit $|q_0 q_1 q_2 ... q_n\rangle$ del dispositivo de entrada, a continuación se obtendra la dimensión n, con este valor se hallará la dimensión del producto tensorial mediante 2^n.

Se tomarán los elemento q_i y q_{i+1} del vector qubit, si q_i es 0 entonces $a = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$, si es 1 entonces $a = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$, igualmente ocurre con q_{i+1} pero en este caso se almacena el resultado en el vector b.

Una vez asignados los vectores se toma el primer valor del vector a y se multiplica por todo el vector b, el resultado se asigna a los primeros lugares del vector aux , después se toma el siguiente elemento del vector a y se multiplica por todo el vector b y se almacena en el lugar correspondiente del vector aux, se repite la operación hasta que se llegue al último elemento del vector a.

La variable dim2 corresponde a la dimensión del producto tensorial, como es el caso $|00\rangle$ que da un vector de dimensión $2^2 = 4$, $|001\rangle$ que da un vector de dimensión $2^3 = 8$, y así sucesivamente $|a_1, a_2, ..., a_n\rangle$ da un vector de dimensión 2^n.

La operación $j + (i * 2)$ corresponde al lugar correcto, del producto tensorial, donde se colocará el resultado del producto $elem * b[j]$ en el vector aux, i se multiplica por 2 como consecuencia de la dimensión del qubit $0 = |0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$ y qubit $1 = |1\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$.

Ahora para el producto tensorial de 2 matrices o transformaciones A=IDENTIDAD y B=NOT tenemos:

$$A = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \qquad y \qquad B = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$$

$$A \otimes B = \begin{pmatrix} 1\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} & 0\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \\ 0\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} & 1\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \end{pmatrix} = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

El algoritmo para obtener el producto tensorial de dos matrices es el siguiente:
1.- Obtener las dimensiones de las matrices A (dimA) y B (dimB)
2.- Obtener las matrices A y B.
3.- De $i \leftarrow 0$ hasta $i \prec dimA$ //producto tensorial
De $j \leftarrow 0$ hasta $j \prec dimA$
$elem \leftarrow A[i][j]$
De $i2 \leftarrow 0$ hasta $i2 \prec dimB$
De $j2 \leftarrow 0$ hasta $j2 \prec dimB$
$aux[i2 + (i * dimB))][j2 + (j * dimB))] = elem * b[i2][j2]$
Incrementa j2
Incrementa i2
Incrementa j Incrementa i

En este algoritmo es necesario obtener las dimensiones de las matrices, para así primeramente tomar un elemento de la matriz A y multiplicarlo por toda la matriz B, después se toma el siguiente elemento de la columna de la matriz A y se multiplica por toda la matriz B, así sucesivamente hasta abarcar todos los elementos de la columna de la matriz A y siguiendo con las filas de la matriz A.

Las expresiones $[i2 + (i * dimB))]$ y $[j2 + (j * dimB))]$ nos permiten colocar el resultado del producto tensorial en el lugar adecuado de la matriz aux que es donde se almacena el resultado del producto tensorial de dos matrices, por ejemplo, en el producto del primer elemento con índice i=1 de la matriz A por toda la matriz B el resultado se almacenará en la matriz aux, el cual ocupará los primeros ik=k lugares de la matriz aux, para cuando se toma el siguiente elemento con índice i=2 de la matriz A y se multiplica por toda la matriz B el resultado se tiene que empezar a almacenar en el lugar 1+k y terminará en ik=2k de la matriz aux, si se toma el siguiente elemento con índice i=3 de la matriz A y se multiplica por toda la matriz B entonces se tiene que empezar a almacenar en el lugar 1+2k y terminará en ik=3k de la matriz aux, y así sucesivamente tendría que empezarse a almacenar el en lugar 1+ik.

Como podremos notar $[i2 + (i * dimB))] = 1 + ik$, cuyo primer elemento de la fila de la matriz B tiene como índice i2=1 y dimB = k. El valor de k será de acuerdo a la dimensión de la matriz B

4.9. Algoritmo de funcionamiento del Demultiplexor Cuántico

En el tema Desarrollo del Demultiplexor Cuántico con Transformaciones Unitarias de la Sección 4.6.1 se analizó como se obtenía el demultiplexor cuántico mediante el producto de tres matrices de dimensión 64×64, que son: $(I \otimes I \otimes I \otimes F_1), (I \otimes F_2 \otimes I \otimes I)$ y $(F_3 \otimes I \otimes I \otimes I)$; en la Sección 4.8 se analizó el algoritmo para obtener productos tensoriales, cabe notar que las compuertas F_1, F_2 y F_3 son compuertas cuánticas de Fredkine,.

El algoritmo del demultiplexor cuántico es el siguiente:
1.- Obtener el producto tensorial de las siguientes compuertas: $aux1 = I \otimes I \otimes I \otimes F_1$
2.- Obtener el producto tensorial de las siguientes compuertas: $aux2 = I \otimes F_2 \otimes I \otimes I$
3.- Obtener el producto tensorial de las siguientes compuertas: $aux3 = F_3 \otimes I \otimes I \otimes I$
4.- Se multiplican las matrices aux1 y aux2
De i=0 hasta $i \prec 64$

De j=0 hasta $j \prec 64$
mat[i][j]=0
De k=0 hasta $k \prec 64$
$mat[i][j] = mat[i][j] + aux1[i][k] * aux2[k][j]$
Incrementa k
Incrementa j
Incrementa i

5.- Se multiplican las matrices mat y aux3 De i=0 hasta $i \prec 64$ De j=0 hasta $j \prec 64$
demul[i][j]=0
De k=0 hasta $k \prec 64$
$demul[i][j] = demul[i][j] + mat[i][k] * aux3[k][j]$
Incrementa k
Incrementa j
Incrementa i

Para el funcionamiento del demultiplexor cuántico tendremos que dar como entrada un qubit de dimensión 6, cuyos valores son los siguientes $|A_0A_1x000\rangle$, se le aplica el demultiplexor cuántico a esta entrada y nos dará como salida $|A_0Y_0A_1Y_1Y_2Y_3\rangle$. A continuación se verá su efecto:

$$DEMUL * |A_0A_1x000\rangle = |A_0Y_0A_1Y_1Y_2Y_3\rangle$$

Conclusiones

Todos los circuitos que se han desarrollado en la Computación Clásica se rigen bajo leyes de la física clásica y cada uno tiene su función, por ejemplo, tenemos circuitos como AND, OR, NOT, NAND, etc., y no solo eso sino que también se han diseñado sus dispositivos electrónicos. Aunque la Computación Cuántica no ha llegado tan lejos, como es diseñar los dispositivos electrónicos, sí ha dado un gran avance al analizar teóricamente estos circuitos, pero dentro de la física cuántica. La tesis se centró en desarrollar teóricamente un Demultiplexor Cuántico, regido por las leyes de la Física Cuántica, y mostrar su simulación mediante un lenguaje visual de programación.

Para ello se dio un breve análisis de la Computación Clásica y también se comprendió los límites de esta, en forma teórica, en comparación con la Computación Cuántica.

En el Capítulo 2 se analizó los comienzos de la Física Cuántica y que mediante un espín de partícula se puede tener varios estados del espín por lo cual se definen los estados propios y superposición de estados.

En el Capítulo 3 se comprendió mejor el Capítulo 2 pero matemáticamente y se describió la diferencia entre un bit clásico y un qubit, así como comprender los registros, algunas operaciones como el producto tensorial, la descripción de estados de un qubit y la definición de algunas compuertas unitarias.

Y por último, para cumplir el objetivo de la Tesis de desarrollar el demultiplexor cuántico se utilizó la Lógica Conservativa, la cual utiliza la compuerta cuántica de Fredkine. Mediante la implementación de 3 compuertas cuánticas de Fredkine en un circuito de lógica conservativa, que es el demultiplexor cuántico, se analizó como mediante una dato de entrada X se transmite en las salidas Y_0, Y_1, Y_2 y Y_3, gracias a los datos de direccionamiento A_0 y A_1; para comprender todo ello se elaboró un análisis paso por paso, pero también se simuló su funcionamiento mediante el Lenguaje Orientado a Objetos que es Visual C++, definiendo asimismo un algoritmo para obtener productos tensoriales de vectores y compuertas unitarias, un algoritmo que obtiene la matriz de una transformación unitaria y otro algoritmo que obtiene el demultiplexor cuántico.

Con el desarrollo de esta Tesis se tiene la perspectiva de que en un futuro se diseñe un demultiplexor cuántico pero no de una sola entrada, es decir, podrán ser x=2,3,...,n entradas con 2^x direcciones.

Debido a que la Computación Cuántica es una área de la computación reciente, no se ha diseñ ado un dispositivo electrónico, pero sí se ve su funcionalidad al programarlo en un lenguaje de programación orientado a objetos, por ello se incluyó un Apéndice con los códigos correspondientes a dichos algoritmos en el Lenguaje de Programación Visual C++.

Apéndice A

Código de la Compuerta de Fredkine

El Método de Shor se utiliza para construir la compuerta cuántica de Fredkine.

- En qubit[i][j] tenemos los 8 estados qubit $|000\rangle, |001\rangle, |010\rangle, |011\rangle, |100\rangle, |101\rangle, |110\rangle$ y $|111\rangle$.

La siguiente función obtiene dicha compuerta:

```
void compuertafredkine()
    int q1[3],q2[3],i,j,ind1,ind2;
    for(i = 0; i ≺ 8; i + +)          //8 estados qubit que da la matriz(shor) DE 8 × 8

        for (j = 0; j ≺ 3; j + +)          //el qubit tiene dimensión 3
            q1[j]=qubit[i][j];      //q1 son las entradas, filas.
        //propiedades de la compuerta de fredkine
        if (qubit[i][0]==0)
                          //q2 son las salidas, columnas.
            q2[0]=q1[0];          //si primer elemento qubit es cero
            q2[1]=q1[2];          //entonces se realiza un intercambio
            q2[2]=q1[1];          //de los dos últimos qubits

            else

            q2[0]=q1[0];          //si primer elemento qubit es uno
            q2[1]=q1[1];          //entonces queda igual
            q2[2]=q1[2];

        ind1=indice(q1);          //la función indice( )
        ind2=indice(q2);          // convierte un qubit a forma decimal
        fredkine[ind1][ind2]=1;     //Método de Shor

// CONVIERTIR UN QUBIT (BINARIO) A FORMA DECIMAL
int indice(int q[3])
    double ind=0.0;
    int j, dec, sign, ent, ndig = 2;
    char *str, string[3];
```

```
    for (j = 0; j <= 2; j++)
        if (q[j]==1) ind=pow(2,(2-j))+ind;
    str = ecvt(ind, ndig, dec, sign);
    strncpy(string, str,dec);
    string[dec] = final de array;
    ent=atoi(string);
return ent;
```

Apéndice B

Código de Productos tensoriales de matrices

Las funciones que obtienen el producto tensorial de matrices son tensorial() y opera().

- La función **opera()** multiplica al elemento de la matriz A por toda la matriz B
- La función **tensorial()** toma un elemento de la matriz A y llama a la función opera(), también respeta el lugar donde se colocarán los nuevos elementos del producto tensorial.
- La variable **elem** es un elemento de la matriz A.
- Las variables **incI y incJ** nos permitirán que el resultado del producto se coloque en el lugar adecuado de la nueva matriz debido a que es un producto tensorial.
- El array **vB[][]** es la matriz B.
- El array **vA[][]** es la matriz A.
- La variable **dimA** es la dimensión de la matriz A.
- La variable **dimB** es la dimensión de la matriz B.
- El array **aux[][]** es la matriz donde se guardará el producto.

A continuación se muestra el código:

```
void opera(int elem,int incI,int incJ,int
vB[sub6][sub6],int dimB,int aux[sub6][sub6])
    int i,j;
    for(i = 0; i ≺ dimB; i + +)          //esta matriz depende de la dimensión de vB
        for (j = 0; j ≺ dimB; j + +)
            aux[i + incI][j + incI] = elem ∗ vB[i][j];
void tensorial(int vA[sub6][sub6],int dimA, int vB[sub6][sub6], int dimB,int aux[sub6][sub6])
    int i, j, elem;
    for (i = 0; i ≺ dimA; i + +)          //la dimension depende de la dimensión de vA
        for (j = 0; j ≺ dimA; j + +)
            elem=vA[i][j];
            opera(elem, i ∗ dimB, j ∗ dimB, vB, dimB, aux);
```

NOTA: Para productos tensoriales de vectores sólo se utiliza una fila, como se muestra a continuación.

```
void operavector(int elem,int incI,int vB[2][1])
    int i;
    for (i = 0; i ≺ 2; i + +)
        aux[i + incI][0] = elem ∗ vB[i][0];
void tensorialvector(int vA[sub6][sub6],int vB[2][1])
    int i,elem;
    for (i = 0; i ≺ dim2; i + +)
        elem=vA[i][0];
        operavector(elem, i ∗ 2, vB);
```

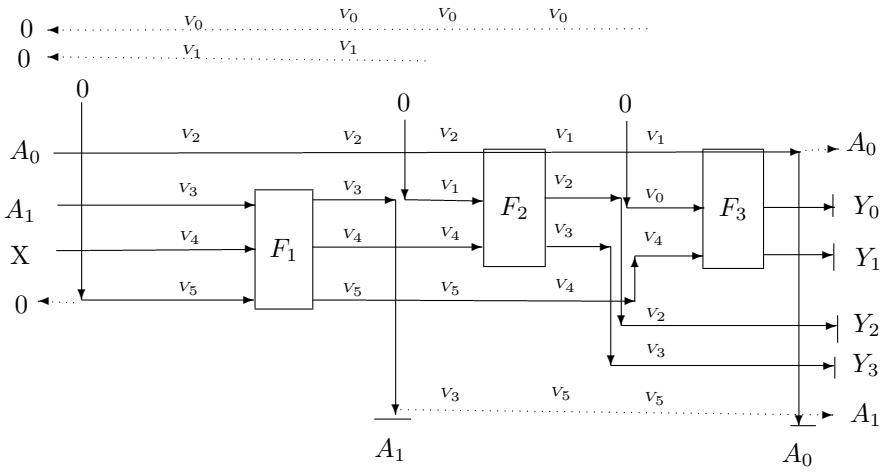
Apéndice C

Implementación del Demultiplexor Cuántico

Para obtener el Demultiplexor Cuántico se utiliza la Compuerta Cuántica de Fredkine y la Compuerta Identidad y se realizan los correspondientes productos tensoriales.

Se obtendrán 3 matrices, los cuales son $F_1 = (I \otimes I \otimes I \otimes F), F_2 = (I \otimes F \otimes I \otimes I)$ y $F_3 = (F \otimes I \otimes I \otimes I)$ Se obtiene un vector el cual tiene 3 valores constantes ceros, el valor de entrada X=1 y dos direcciones A_0 y A_1 dados por el usuario. El orden del vector de entrada es de acuerdo al orden de entrada de los datos en el demultiplexor cuántico, como se mostrará más adelante, y es $|0\rangle |0\rangle |A_0\rangle |A_1\rangle |1\rangle |0\rangle$ Este vector $|0\rangle |0\rangle |A_0\rangle |A_1\rangle |1\rangle |0\rangle$, los datos de este vector pasarán, por así decirlo, por la matriz de la compuerta cuántica de Fredkine F_1 y como resultado dará otro vector identificado como $|v_0\rangle |v_1\rangle |v_2\rangle |v_3\rangle |v_4\rangle |v_5\rangle$, este vector tiene que pasar por la matriz de la compuerta de Fredkine F_2 pero no en el mismo orden de los elementos, al observar la gráfica del demultiplexor cuántico en la Sección 4.6 se observa que el orden del vector es $|v_0\rangle |v_2\rangle |v_1\rangle |v_4\rangle |v_5\rangle |v_3\rangle$, los datos de este vector pasará por la compuerta de Fredkine F_2 y como resultado dará otro vector identificado como $|v_0\rangle |v_1\rangle |v_2\rangle |v_3\rangle |v_4\rangle |v_5\rangle$, este vector tiene que pasar por la matriz de la compuerta de Fredkine F_3 pero no en el mismo orden de los elementos, de acuerdo al demultiplexor cuántico se observa que el orden del vector es $|v_1\rangle |v_0\rangle |v_4\rangle |v_2\rangle |v_3\rangle |v_5\rangle$, este vector pasará por la matriz F_3 y dará como salida $|A_0\rangle |Y_0\rangle |Y_1\rangle |Y_2\rangle |Y_3\rangle |A_1\rangle$.

Su funcionamiento es el siguiente:



A continuación se presentará parte del programa de un Demultiplexor Cuántico que utiliza la Compuerta Cuántica de Fredkine.

- La función compuertaFredkine( ), está en el Apéndice A_1
- Las funciones tensorial(), opera(), tensorialvector() y operavector( ) están el apéndice A_2

```
void main()
    compuertafredkine();                //Obtiene la compuerta de Fredkine
    identidad();                        //Obtiene la compuerta Identidad
    //Matriz F1: Producto tensorial (I ⊗ I ⊗ I ⊗ F) de dimensión 64 × 64
    tensorial(iden,8,fredkine,8,aux);
    //Matriz F2: Producto tensorial (I ⊗ F ⊗ I ⊗ I) de dimensión 64 × 64
    tensorial(iden,2,fredkine,8,aux2);      // (I ⊗ F)      2 * 8 = 16
    asigna(mat,aux2);                   //asigna lo de una matriz a otra
    tensorial(mat,16,iden,4,aux2);      //(I ⊗ F ⊗ I ⊗ I)      16 * 4 = 64
    //Matriz F3: Producto tensorial (F ⊗ I ⊗ I ⊗ I) de dimensión 64 × 64
    tensorial(fredkine,8,iden,8,aux3);      //(F ⊗ I ⊗ I ⊗ I)      8 * 8 = 64
    asigna(mat,aux);                    //asigna lo de una matriz a otra
    //Entradas del vector
    v[0]=0; v[1]=0; v[2]=mValor1edit; //mEntrada0edit es la dirección A0
    v[3]=mValor2edit; v[4]=1; v[5]=0; //mEntrada1edit es la dirección A1
    qVector(v);
    //Pasa a la matriz F1
    producto2(aux,mat);
    conversion(direccion);
    //Pasa a la matriz F2
    asigInd(vq[0],vq[2],vq[1],vq[4],vq[5],vq[3],vq);
    qVector(vq);
    producto2(aux,aux2);
    conversion(direccion);
    //Pasa a la matriz F3
    asigInd(vq[1],vq[0],vq[4],vq[2],vq[3],vq[5],vq);
    qVector(vq);
    producto2(aux,aux3);
```

```
    conversion(direccion);
    if (vq[1]=1) yn= 1;
    if (vq[2]=1) yn= 2;
    if (vq[3]=1) yn= 3;
    if (vq[4]=1) yn= 4;
return yn;
//MATRIZ IDENTIDAD
void identidad()
int i,j;
    for (i = 0; i ≺ sub6; i + +)
        for(j = 0; j ≺ sub6; j + +)
            if (i==j)
                iden[i][j]=1;
//OBTIENE VECTOR DE DIMENSIóN 6 A VECTOR DE DIMENSIóN 64
void qVector(int vector[6])
int i,i2;
    //PRODUCTO TENSORIAL DEL VECTOR
    dim2=1;
    for(i=0;i¡dim-1;i++)
        dim2=2*dim2;
        if (i==0)
            if (vector[i]==0)
                for(i2 = 0; i2 ≺ dim; i2 + +)      aux2[i2][0]=v0[i2][0];
            else      for (i2 = 0; i2 ≺ dim; i2 + +)      aux2[i2][0]=v1[i2][0];
        if (i¿=1)
            for (i2 = 0; i2 ≺ dim2; i2 + +)      aux2[i2][0]=aux[i2][0];
        if (vector[i+1]==0) tensorialVector(aux2,v0);
        else tensorialVector(aux2,v1);

//FUNCION DEL PRODUCTO DEMUL ∗ VECTOR QUBIT
void producto2(int vect[sub6][sub6],int mat[sub6][sub6])
int i,j=0,k;
    for (i = 0; i ≺ 64; i + +)
        direccion[i][j]=0;
        for(k = 0; k ≺ 64; k + +)
            direccion[i][j] = direccion[i][j] + mat[i][k] ∗ vect[k][j];

//OBTIENE VECTOR DE DIMENSIóN 64 A VECTOR DE DIMENSIóN 6
void conversion(int vector[sub6][1])
    int i,n=64,cont=0,ban=0,a[2];
    while((n/2) ≻= 1)

        for (i = 0; i ≺ (n/2); i + +) //chequeo de la primera mitad

            if (vector[i][0]==1)      a[0]=1; ban=1;
            if ((ban==0)AND(i==((n/2)-1)))      a[0]=0;

    ban=0;
    for (i = (n/2); i ≺ n; i + +) //chequeo de la segunda mitad
        if (vector[i][0]==1)      a[1]=1; ban=1;
        if ((ban==0)AND(i==n-1)) a[1]=0;

    if (a[0]==0 AND a[1]==1) vq[cont]=1;
```

```
    if (a[0]==1 AND a[1]==0) vq[cont]=0;
    if (a[1]==1)
        for (i = (n/2); i ≺ n; i + +)        vector[i-(n/2)][0]=vector[i][0];
    cont++;
    n=n/2;

void asigInd(int a,int b,int c, int d, int e, int f, int vect[6])
    vect[0]=a;
    vect[1]=b;
    vect[2]=c;
    vect[3]=d;
    vect[4]=e;
    vect[5]=f;
```

Bibliografía

[1] Andrew Steane, "Quantum computing", Department of Atomic and Laser Physics, University of Oxford, England, July 1997.

[2] Andrew Steane, "Quantum Computing", Reports on Progress in Physics, Vol. 61 (1998), pág. 117-173.

[3] Edward Fredkin and Tommaso Toffoli, " Conservative Logic", International Journal of Theoretical Physics, Vol. 21, No 3 / 4, Massachusetts, 1982.

[4] Bernhard *Ömer*, " A Procedural Formalism for Quantum Computing,"Department of Theoretical Physics, Technical University of Vienna, July 1998.

[5] Christian Paquin, " Computing in the quantum world," Université de Montréal, July 1998.

[6] M. Morris Mano, "Lógica Digital y Diseño de Computadores", California State University, Los Angeles, Prentice Hall, 1982.

[7] Stanley I. Grossman, " Álgebra Lineal-Segunda Edición," University of Montana, Grupo Editorial Iberoamérica, 1988.

[8] Scott H. Williams, Colin P. y Clearwater, " Explorations in Quantum Computing", New York: Springer-Verlag, 1998

[9] Peter Shor, " Algorithms for Quantum Computation: Discrete Logarithms and Factoring", Proceedings 35th Annual Symposium on Foundations of Computer Science (1994), pág. 124-134.

[10] Bernhard *Ömer*, "Quantum Programming in QCL", Institute of Information Systems, Technical University of Vienna, 20th January 2000.