Centro Universitario de la Defensa en la Escuela Naval Militar

TRABAJO FIN DE MÁSTER

El problema de la factorización de números enteros de gran tamaño y su resolución mediante computación cuántica

Máster Universitario en Dirección TIC para la Defensa

ALUMNO: Rafael Romero Margaritti

DIRECTORES: Javier Vales Alonso

Milagros Fernández Gavilanes

CURSO ACADÉMICO: 2022-2023

Universidade Vigo

Centro Universitario de la Defensa
en la Escuela Naval Militar

TRABAJO FIN DE MÁSTER

El problema de la factorización de números enteros de gran tamaño y su resolución mediante computación cuántica

Máster Universitario en Dirección TIC para la Defensa

Especialidad de Sistemas y Tecnologías de la Información

Universidade Vigo

RESUMEN

El objetivo final de este Trabajo Fin de Master es plasmar con un ejemplo cómo estarían amenazados los Sistemas de Encriptado actuales por la Computación Cuántica, exponiendo las ventajas y desventajas de la Computación Cuántica respecto a la Computación Binaria. Para lograr este objetivo se ha escogido el problema de la Factorización de Números Enteros de gran tamaño en la que se basan muchos algoritmos de encriptación actuales.

En la primera parte del Trabajo se introduce los conceptos básicos de la Computación Binaria (codificación, puertas lógicas, algebra de Boole, etc.) Posteriormente se detalla quizás el más famoso y utilizado de los Algoritmos de Encriptación y Autenticación de Claves pública y privada que se basa en la dificultad de factorizar enteros grandes, este Algoritmo es conocido por RSA. Se expone también el Algoritmo Asimétrico Diffie-Hellman utilizado para intercambio de claves simétricas. Este Algoritmo se basa en la dificultad de calcular logaritmos discretos.

A continuación se explica de forma somera los algoritmos de factorización de enteros grandes con computación clásica que existen en la actualidad. Seguidamente se introducen los aspectos más importantes de la Computación Cuántica (puertas lógicas, qubits, entrelazamiento, superposición, Qutrit, paralelismo cuántico, etc.)

Finalmente se describe el Algoritmo de Shor que es la gran esperanza de mejora, si se llega a crear alguna vez un ordenador cuántico totalmente operativo, para la factorización de números enteros grandes. Para la explicación del Algoritmo también es necesario explicar la Transformada Cuántica de Fourier, su inversa y el Algoritmo de Estimación de Fase. A modo resumen se presentará un ejemplo de un circuito cuántico del Algoritmo de Shor.

PALABRAS CLAVE

Algoritmo, Factorización, Binario, Cuántico, RSA, Shor, Qubit, Transformada Cuántica Fourier, Estimación de Fase y Exponenciación Modular

CONTENIDO

Contenido 1
Índice de Figuras 3
Índice de Tablas 4
Índice de Ecuaciones 5
1 Introducción 8
2 Plataformas Binarias 9
2.1 Codificación 9
2.2 Puertas Lógicas Computación Clásica 11
2.2.1 Puertas Unarias 11
2.2.2 Puertas Binarias 13
2.3 Operaciones (Algebra de Boole) 17
2.3.1 Suma 17
2.3.2 Comparador 18
2.3.3 Resta 19
2.3.4 Multiplicación 19
2.3.5 División 20
2.3.6 Potencia 20
2.3.7 Rotación o Permutación 20
3 Algoritmo RSA 21
3.1 Algoritmo de Euclides 25
3.2 Algoritmo de Euclides Extendido 26
3.3 Exponenciación por cuadrados o exponenciación binaria 26
3.4 Exponenciación Modular Binaria 27
4 Algoritmo Diffie-Hellman 28
5 Algoritmos de Factorización 29
6 Universo Cuántico 31
6.1 Introducción 31
6.2 Codificación 37
6.3 Puertas Lógicas Cuánticas 38
6.3.1 Puertas Lógicas Cuánticas Unarias 38
6.3.2 Puerta Lógicas Cuánticas Binarias 40
6.3.3 Puerta Lógicas Cuánticas Terciarias 43
6.4 El Paralelismo Cuántico 45

6.4.1 Producto Tensorial 45
6.4.2 Estados de Bell (2 qubits entrelazados) 46
6.4.3 El controlador controlado 48
6.4.4 Algoritmo de Deutsch-Jozsa 49
6.5 Notación 53
7 Algoritmo de Shor 56
7.1 Algoritmo de Shor 56
7.1.1 Fundamentos del Algoritmo de Shor 56
7.2 Transformada Cuántica de Fourier (QFT) 57
7.3 Transformada Inversa de Fourier Cuántica 58
7.4 Estimación Cuántica de Fase 59
7.5 Periodo de la función $[m^k]_n$ 62
7.6 Parte Cuántica del Algoritmo de Shor 63
7.7 Mejoras al Algoritmo de Shor 69
7.8 REFLEXIONES 72
8 Ejemplo de Circuito Cuántico 77
9 Conclusiones 79
10 Bibliografía 80

ÍNDICE DE FIGURAS

Figura 2-1, *Código Morse [Wikipedia]* 9
Figura 2-2, Código ASCII [Google Sites].....10
Figura 2-3, *Implementación puerta OR de 3 entradas con puertas binarias [Google Sites]* 11
Figura 2-4, *Puerta NOT [Mi Electrónica Fácil]* 12
Figura 2-5, *Puertas Binarias [WordPress]* 13
Figura 2-6, *Puertas NOT, AND y OR implementadas con NAND y NOR [WordPress]*.....16
Figura 2-7, *Puerta XOR implementada con NAND [Wikipedia]* 17
Figura 2-8, *Puerta XOR implementada con NOR [Wikipedia]* 17
Figura 2-9, *Suma de 2 bits con acarreo de salida sin acarreo de entrada [lc.fie.umich.mx]* 18
Figura 2-10, *Suma de 2 bits con Acarreo de entrada y de salida [lc.fie.umich.mx]* 18
Figura 2-11, *Comparador de 2 bits, M => x>y, I => x=y, m => x<y [Wikipedia]* 19
Figura 2-12, *Resta de 2 bits [Electrónica y la Web]* 19
Figura 5-1 *Órdenes de Complejidad [Wikipedia]* 30
Figura 6-1, *Tren de juguete* 32
Figura 6-2, *Esfera de Bloch [Wikipedia]* 34
Figura 6-3, *Representación gráfica Número Complejo [Sangakoo]*.....35
Figura 6-4, X = NOT [DocIRS] 53
Figura 6-5, HADAMARD [DocIRS] 54
Figura 6-6, CAMBIO DE FASE Z [DocIRS] 54
Figu*ra 6-7, CX= CNOT Controlada* [DocIRS].....54
Figura 6-8, Swap [DocIRS].....54
Figura 6-9, C*CX=Toffoli* [DocIRS].....54
Figura *6-10, U de un qubit controlada por un qubit* [DocIRS] 55
Fig*ura 6-11, Medición de un qubit* [DocIRS] 55
Figura 7-1, Ruedas Dentadas.....76
Figura 8-1, *Esquema Circuito Cuántico del Algoritmo de Shor [Wikipedia]* 77
Figura 8-2, *Esquema Circuito Cuántico del algoritmo de Kitaev [Naukas]* 78

ÍNDICE DE TABLAS

Tabla 2-1: Salida siempre “0” 12
Tabla 2-2: Salida siempre “1” 12
Tabla 2-3: Salida igual a la Entrada 12
Tabla 2-4: Salida siempre “0” 13
Tabla 2-5: Salida siempre “1” 14
Tabla 2-6: Salida = entrada A 14
Tabla 2-7: Salida = entrada B 14
Tabla 2-8: Salida = NOT A 14
Tabla 2-9: Salida = NOT B 15
Tabla 2-10: OR con la entrada A negada 15
Tabla 2-11: AND con la entrada A negada 15
Tabla 2-12: OR con la entrada B negada 15
Tabla 2-13: AND con la entrada B negada 16
Tabla 2-14: Ejemplo multiplicación 20
Tabla 6-1: NAND 44
Tabla 6-2: Funciones de 1 bit 49

ÍNDICE DE ECUACIONES

Ecuación 3-1 21
Ecuación 3-2 21
Ecuación 3-3 21
Ecuación 3-4 21
Ecuación 3-5 22
Ecuación 3-6 22
Ecuación 3-7 22
Ecuación 3-8 22
Ecuación 3-9 23
Ecuación 3-10 23
Ecuación 3-11 23
Ecuación 3-12 23
Ecuación 3-13 24
Ecuación 3-14 24
Ecuación 3-15 24
Ecuación 3-16 24
Ecuación 3-17 24
Ecuación 3-18 25
Ecuación 4-1 28
Ecuación 6-1 31
Ecuación 6-2 31
Ecuación 6-3 33
Ecuación 6-4 33
Ecuación 6-5 33
Ecuación 6-6 35
Ecuación 6-7 35
Ecuación 6-8 36
Ecuación 6-9 36
Ecuación 6-10 38
Ecuación 6-11 39
Ecuación 6-12 39
Ecuación 6-13 39
Ecuación 6-14 40

Ecuación 6-15 40
Ecuación 6-16 40
Ecuación 6-17 41
Ecuación 6-18 42
Ecuación 6-19 43
Ecuación 6-20 44
Ecuación 6-21 44
Ecuación 6-22 45
Ecuación 6-23 45
Ecuación 6-24 46
Ecuación 6-25 46
Ecuación 6-26 46
Ecuación 6-27 47
Ecuación 6-28 47
Ecuación 6-29 48
Ecuación 6-30 48
Ecuación 6-31 49
Ecuación 6-32 50
Ecuación 6-33 50
Ecuación 6-34 50
Ecuación 6-35 51
Ecuación 6-36 51
Ecuación 6-37 51
Ecuación 6-38 51
Ecuación 6-39 52
Ecuación 6-40 53
Ecuación 7-1 56
Ecuación 7-2 56
Ecuación 7-3 59
Ecuación 7-4 59
Ecuación 7-5 59
Ecuación 7-6 60
Ecuación 7-7 60
Ecuación 7-8 61
Ecuación 7-9 61
Ecuación 7-10 62

Ecuación 7-11 62
Ecuación 7-12 63
Ecuación 7-13 66
Ecuación 7-14 69
Ecuación 7-15 69
Ecuación 7-16 71
Ecuación 7-17 72
Ecuación 7-18 72
Ecuación 7-19 72
Ecuación 7-20 72
Ecuación 7-21 72
Ecuación 7-22 73
Ecuación 7-23 73
Ecuación 7-24 73
Ecuación 7-25 74

1 INTRODUCCIÓN

En la actualidad se está especulando bastante si la Computación Cuántica va a suponer un antes y un después en la Computación. Al igual que lo que supuso la computación clásica que consiguió resolver problemas que nunca se habían pensado que fuésemos capaces de resolver, por ejemplo, el descifrado del genoma humano, la computación cuántica se supone que va a resolver problemas que hoy por hoy son irresolubles o irresolubles en un tiempo razonable (de modo más preciso, el tiempo de resolución no crece exponencialmente con el tamaño del problema, sino de modo polinómico).

Un ejemplo de los problemas que son actualmente irresolubles en un tiempo razonable (polinómico) es el problema de la Factorización de números enteros grandes. Hay varios Algoritmos de Encriptación que se basan en la complejidad del cálculo de Factorización de enteros y es un hecho que no podrá seguir usándose tal cual si se consigue resolver este problema con los ordenadores cuánticos. No obstante, la Computación Cuántica puede traer también mejores Algoritmos de Encriptación.

Por ser un tema de interés indiscutible se va a intentar clarificar conceptos no sólo de la Computación Cuántica sino también de la Computación Clásica o Binaria, para poder comparar las dos filosofías y tener claro sus ventajas e inconvenientes.

Como no puede ser de otro modo los dos tipos trabajan con Información. La Información puede ser palabras, números, fotos, videos, etc. Existen tres funciones esenciales para el manejo de la Información (almacenamiento, transmisión y operación). Las dos primeras son dependientes y no se pueden dar por separado. Para tratar con información tenemos que poder almacenarla (escribir, recibir, etc.) y poder leerla (transmitir, reproducir, etc.) Hay infinidad de formas de representar Información pero todas ellas se resumen en tener **Símbolos** y una **Codificación** para poder Almacenarla, Transmitirla y Operarla. La Información puede ser operada de múltiples formas dependiendo sobre todo del tipo que sea. Las operaciones más básicas son Sumar, Restar, Comparar, Multiplicar, Dividir, Ordenar, Permutar, Rotar, etc.

2 PLATAFORMAS BINARIAS

2.1 Codificación

Como la palabra *binario* indica sólo tenemos 2 símbolos para representar la Información, existen infinidad de formas de representarlos en la naturaleza, estos 2 símbolos pueden ser: blanco-negro, lleno-vacío, quemado-no quemado, magnetizado- no magnetizado, perforado-no perforado, verdadero-falso, 5 voltios-0 voltios, cambio de estado-no cambio de estado, etc. Se suelen representar como "0" y "1" que no tienen que suponer cantidades pero la realidad es que representarlos con números ayuda a los seres humanos a la hora de comprender las operaciones que se realizan. El nombre que se le dio al ítem individual fue ***Bit***. Con un solo bit se pueden representar dos estados. Existen múltiplos del bit: Byte (8 bits), Kilobyte (1024 bytes), etc. Lo habitual es trabajar siempre con potencias de 2.

Se puede codificar todo, mayor sea el número de elementos a codificar mayor será el número de bits necesarios. Por ejemplo, para representar/codificar los días de la semana (7) necesitamos 3 bits porque dispondremos de 8 combinaciones posibles (2^3). Podemos decidir que "000" será el domingo, "001" el lunes, "010" el martes, "011" el miércoles, "100" jueves, "101" viernes y "110" sábado. Nos quedaría libre el "111".

Uno de los más conocidos ejemplos clásicos de alfabeto binario que se utilizó para comunicar y almacenar la información fue el Código Morse (Punto y Raya) (Siglo XIX) que consta de 5 caracteres como máximo y 1 como mínimo. No distingue mayúsculas de minúsculas. Entre sus propiedades destacan que los símbolos más frecuentes poseen codificaciones más cortas, lo que minimiza el tiempo de transmisión de un mensaje.

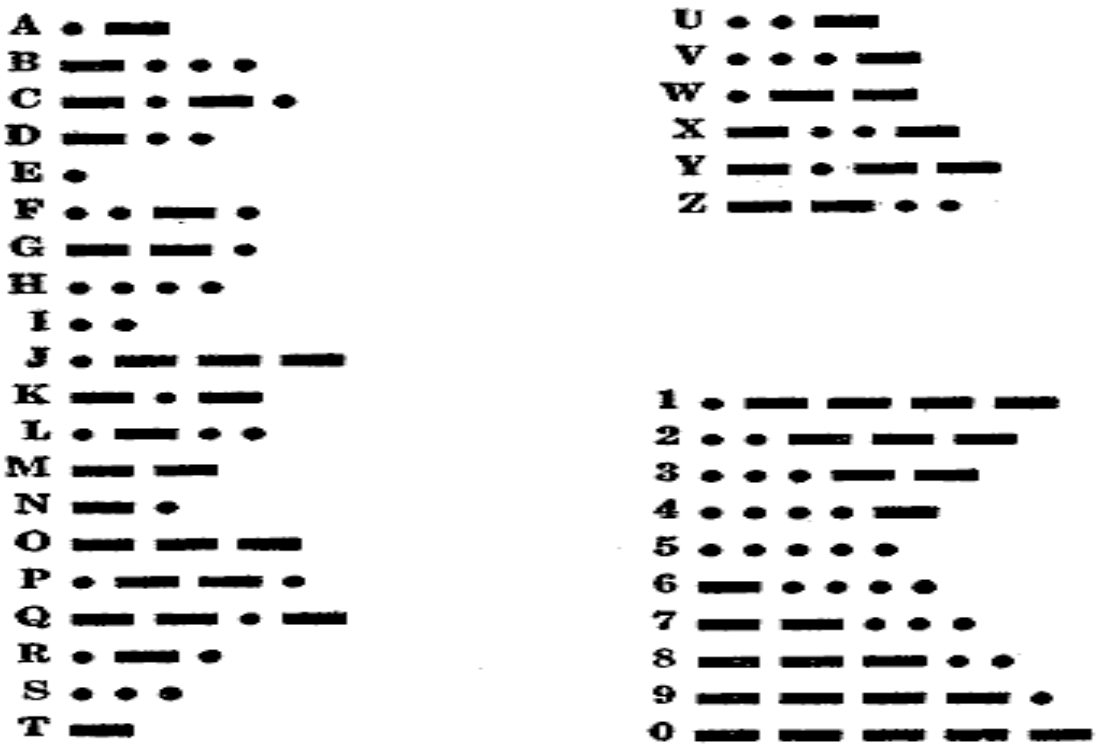


Figura 2-1, ***Código Morse [Wikipedia]***

En el ámbito de los ordenadores el más conocido y todavía utilizado es el ASCII (American Standard Code for Information Interchange) (1963). Es un código para representar caracteres y comandos, en un teclado latino, consta de 8 bits (256 valores, 2^8).

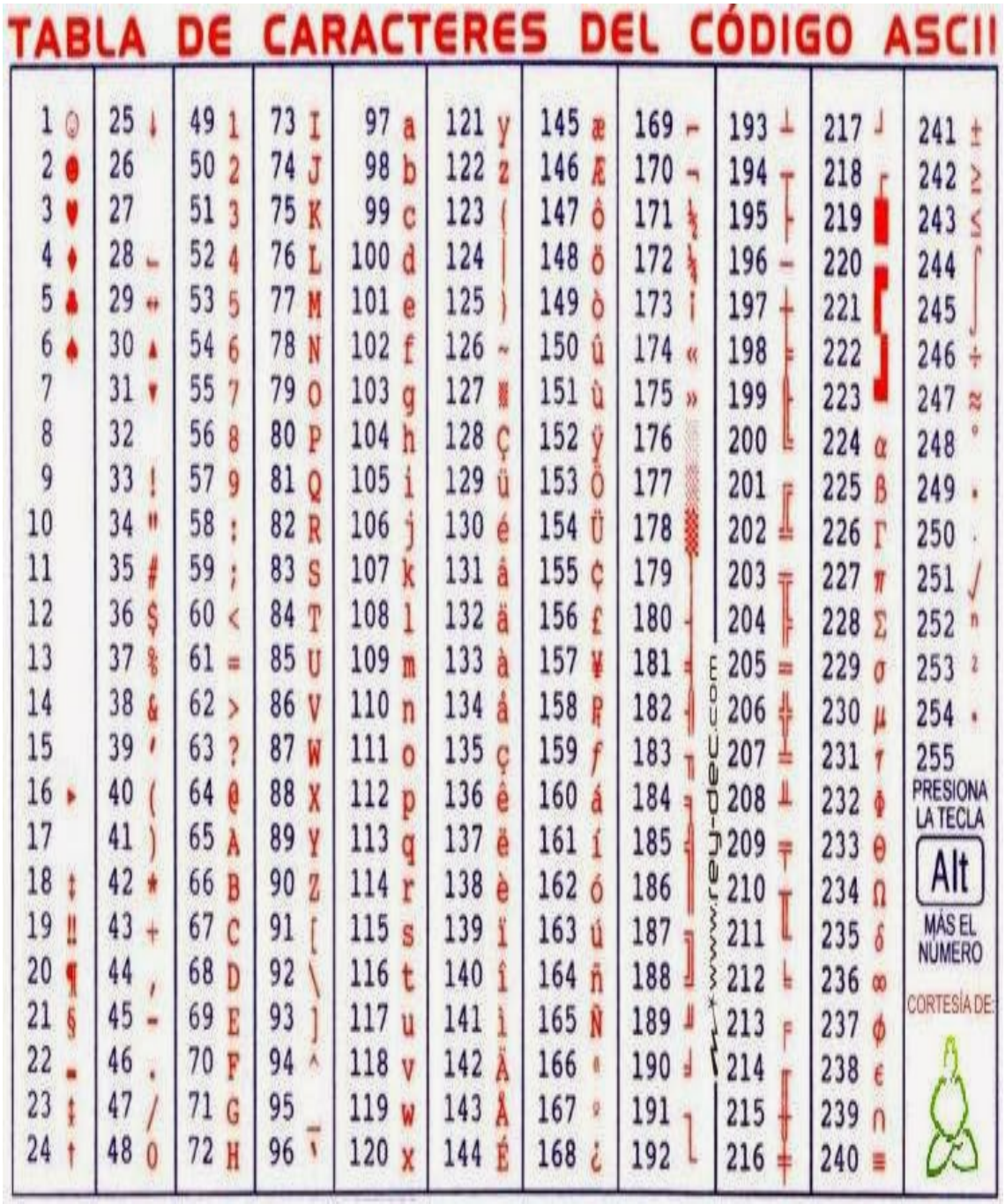


Figura 2-2, Código ASCII [Google Sites]

Existen multitud de codificaciones jpg, bmp, pdf, mp4, mp3, etc. A la hora de representar números (decimales) podríamos utilizar, por ejemplo, ASCII pero perderíamos mucho espacio y tiempo de computación en las operaciones matemáticas. En vez de eso se codifican los números digitales en binario

se opera en binario y después se transforma el número binario (potencias de 2) a un número digital (potencias de 10) para representar el resultado a los humanos. Aproximadamente, un número de tres cifras digitales (1000 valores, 10^3) parte entera o parte fraccionaria equivale aproximadamente a 10 bits (1024 valores, 2^{10}).

2.2 Puertas Lógicas Computación Clásica

Se catalogan las puertas lógicas dependiendo del número de entradas que tengan, todas las puertas lógicas en la computación binaria tienen una sola salida. Las puertas lógicas de 1 entrada son las **unarias**, si tienen 2 entradas son las **binarias** y si tienen 3 terciarias, etc. Pero en realidad sólo existen unarias y binarias ya que con ellas se pueden representar las demás. Por ejemplo una terciaria se forma con 2 binarias, se coge la salida de la primera puerta (2 entradas ya utilizadas) que dará como salida una entrada de la segunda puerta y la otra entrada que queda será la otra entrada de la segunda puerta.

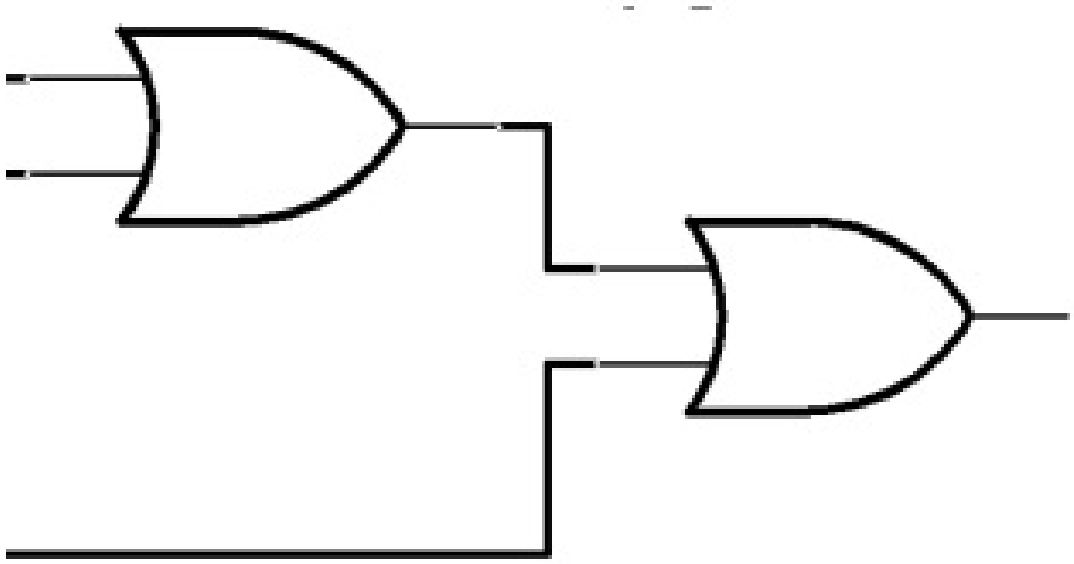


Figura 2-3, ***Implementación puerta OR de 3 entradas con puertas binarias [Google Sites]***

2.2.1 Puertas Unarias

Sólo existe una puerta unaria no trivial, que se llama **NOT**, y cuyo cometido de esta puerta es dar por la salida lo contrario de la entrada. Algunas veces se simplifica con un círculo vacío en la entrada que corresponda o en la salida de una puerta.

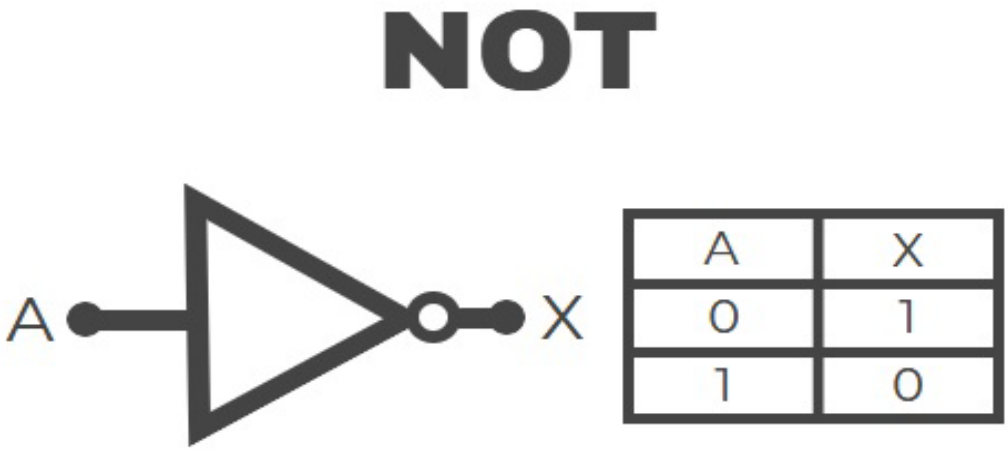


Figura 2-4, *Puerta NOT [Mi Electrónica Fácil]*

Las demás combinaciones no son puertas lógicas ya que:

ENTRADA	SALIDA
0	0
1	0

Tabla 2-1: Salida siempre "0"

ENTRADA	SALIDA
0	1
1	1

Tabla 2-2: Salida siempre "1"

ENTRADA	SALIDA
0	0
1	1

Tabla 2-3: Salida igual a la Entrada

2.2.2 Puertas Binarias

Existen 4 combinaciones posibles en las entradas (“00”, “01”, “10”, “11”) y una salida para cada entrada. Por lo que tenemos 16 posible combinaciones:

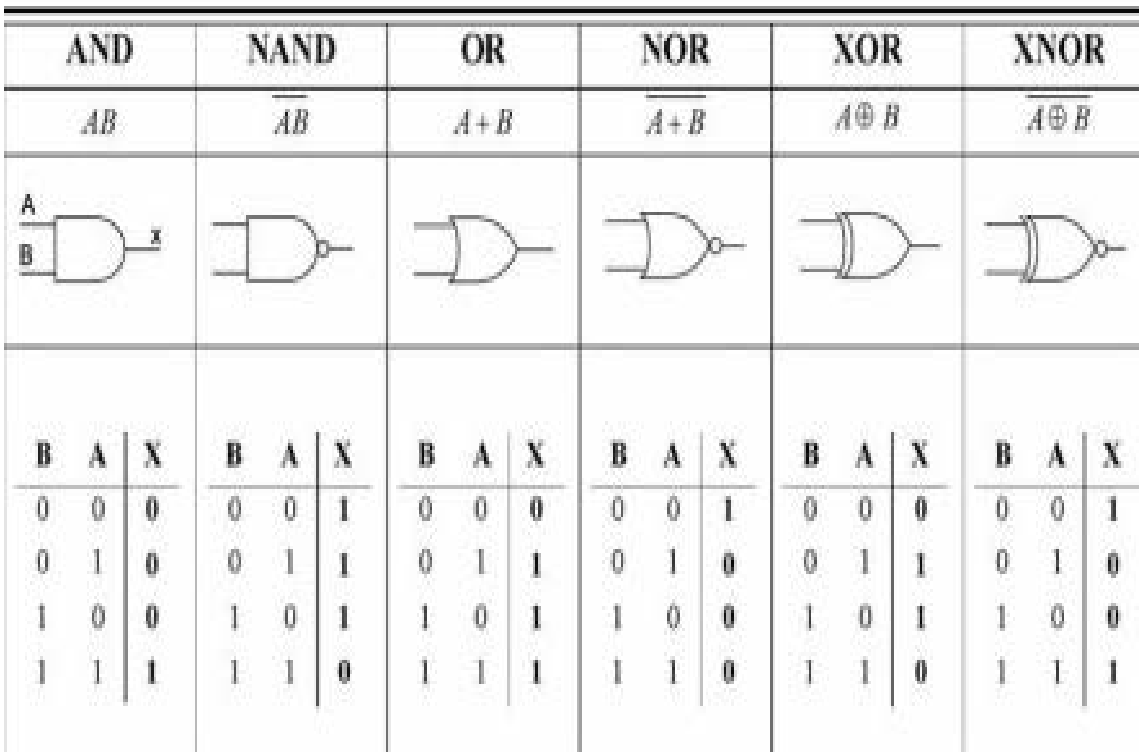


Figura 2-5, *Puertas Binarias [WordPress]*

Con esta figura hemos cubierto 6 combinaciones, las otras combinaciones posibles son:

ENTRADAS	SALIDA
00	**0**
01	**0**
10	**0**
11	**0**

Tabla 2-4: Salida siempre “0”

ENTRADAS	SALIDA
00	**1**
01	**1**
10	**1**
11	**1**

Tabla 2-5: Salida siempre "1"

ENTRADAS	SALIDA
00	**0**
01	**0**
10	**1**
11	**1**

Tabla 2-6: Salida = entrada A

ENTRADAS	SALIDA
00	**0**
01	**1**
10	**0**
11	**1**

Tabla 2-7: Salida = entrada B

ENTRADAS	SALIDA
00	**1**
01	**1**
10	**0**
11	**0**

Tabla 2-8: Salida = NOT A

ENTRADAS	SALIDA

00	**1**
01	**0**
10	**1**
11	**0**

Tabla 2-9: Salida = NOT B

ENTRADAS	SALIDA
00	**1**
01	**1**
10	**0**
11	**1**

Tabla 2-10: OR con la entrada A negada

ENTRADAS	SALIDA
00	**0**
01	**1**
10	**0**
11	**0**

Tabla 2-11: AND con la entrada A negada

ENTRADAS	SALIDA
00	**1**
01	**0**
10	**1**
11	**1**

Tabla 2-12: OR con la entrada B negada

ENTRADAS	SALIDA
00	**0**
01	**0**
10	**1**
11	**0**

Tabla 2-13: AND con la entrada B negada

Por lo que tenemos sólo tres tipos de puertas lógicas binarias puras (**AND, OR y XOR)** y sólo una puerta lógica unaria (**NOT**). Estas cuatro puertas se pueden representar con **NAND o NOR (Puertas Universales)**. NAND (AND seguida de una puerta NOT) y NOR (OR seguida de una puerta NOT).

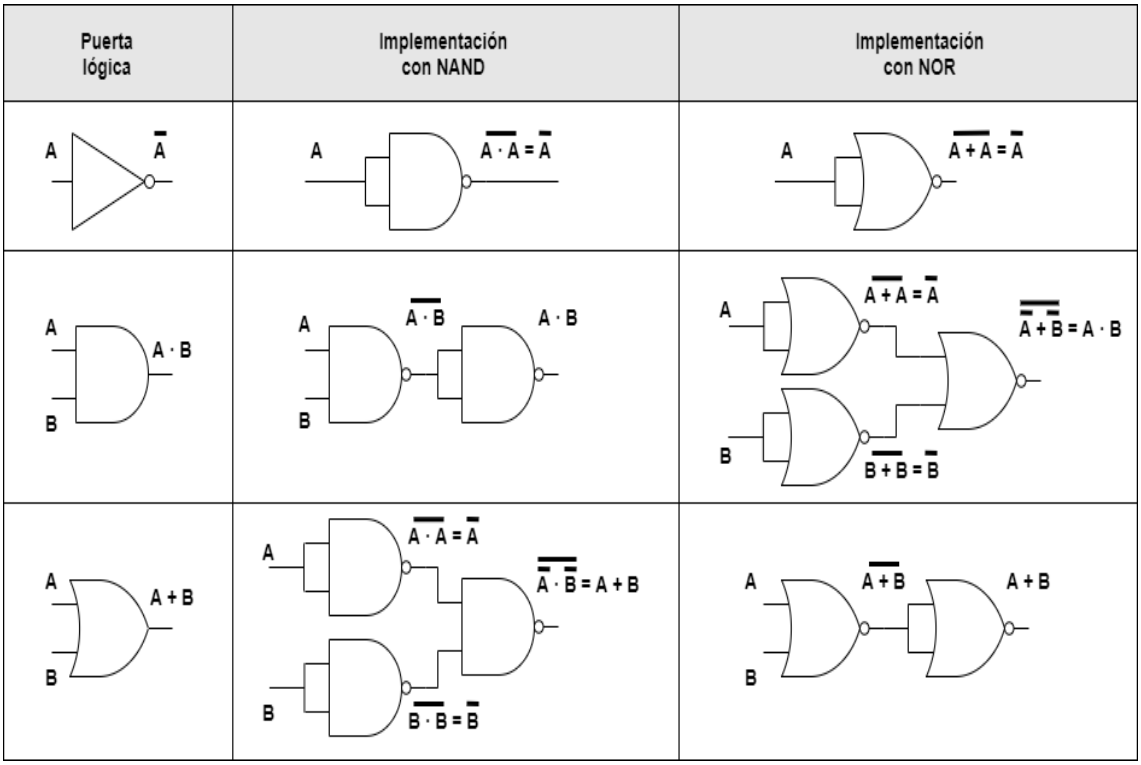


Figura 2-6, ***Puertas NOT, AND y OR implementadas con NAND y NOR [WordPress]***

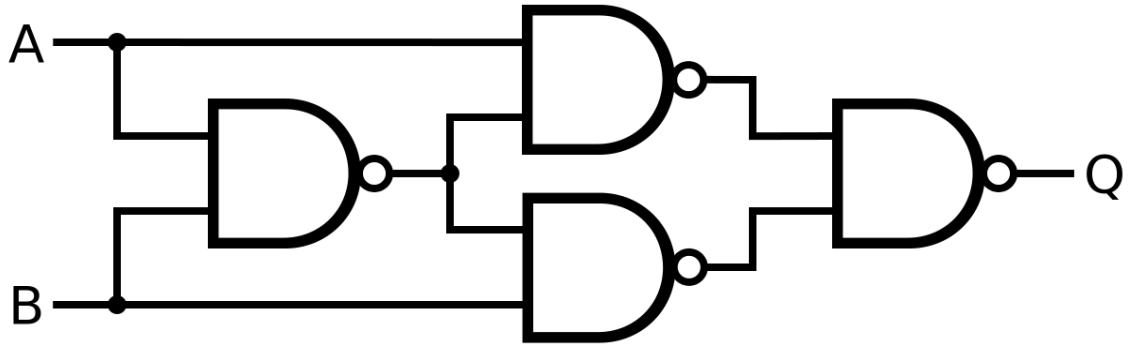


Figura 2-7, *Puerta XOR implementada con NAND [Wikipedia]*

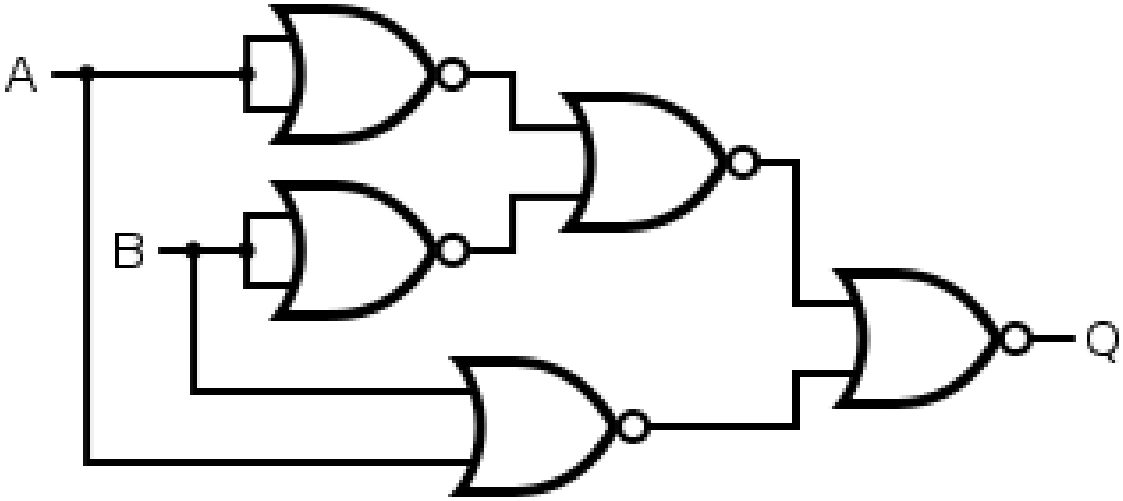


Figura 2-8, *Puerta XOR implementada con NOR [Wikipedia]*

2.3 Operaciones (Algebra de Boole)

Se hace constar que con las puertas binarias y unaria antes descritas los ordenadores actuales realizan todos los cálculos y operaciones. A continuación, se van a explicar sólo las operaciones esenciales, necesarias en la factorización.

2.3.1 Suma

El primer bit menos significativo de cada sumando se hace con la primera figura y los sucesivos con la segunda figura y si los últimos bits los más significativos tienen acarreo se pone directamente el acarreo como bit más significativo de la suma.

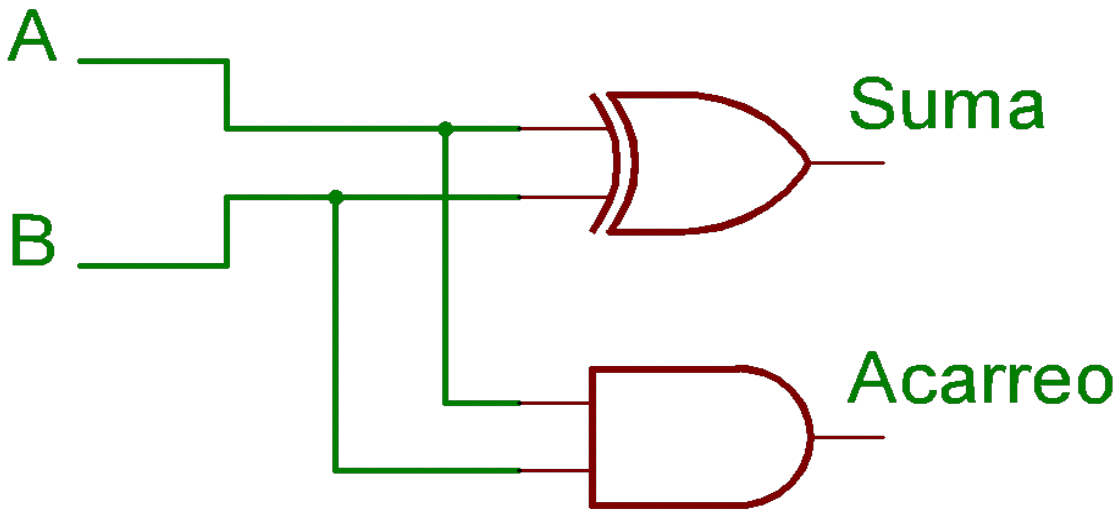


Figura 2-9, ***Suma de 2 bits con acarreo de salida sin acarreo de entrada [lc.fie.umich.mx]***

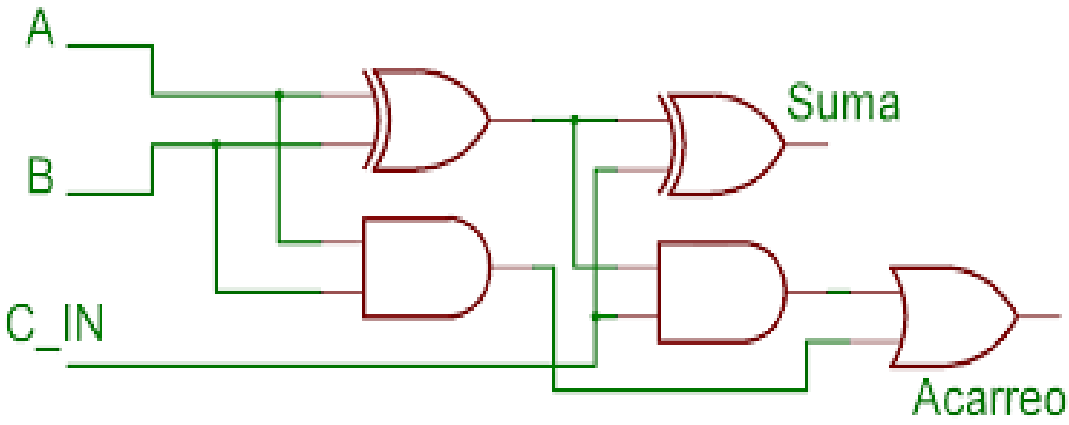


Figura 2-10, ***Suma de 2 bits con Acarreo de entrada y de salida [lc.fie.umich.mx]***

2.3.2 Comparador

Para comparar dos números binarios, se van comparando bit a bit empezando por los bits más significativos (para esta operación también sería posible comenzar por los menos significativos) de cada número. Si no son iguales ya hemos acabado, si son iguales continuamos con el siguiente así hasta que lleguemos al final. Si el último resultado coincide, eso quiere decir que son iguales, ya que tienen todos los bits iguales.

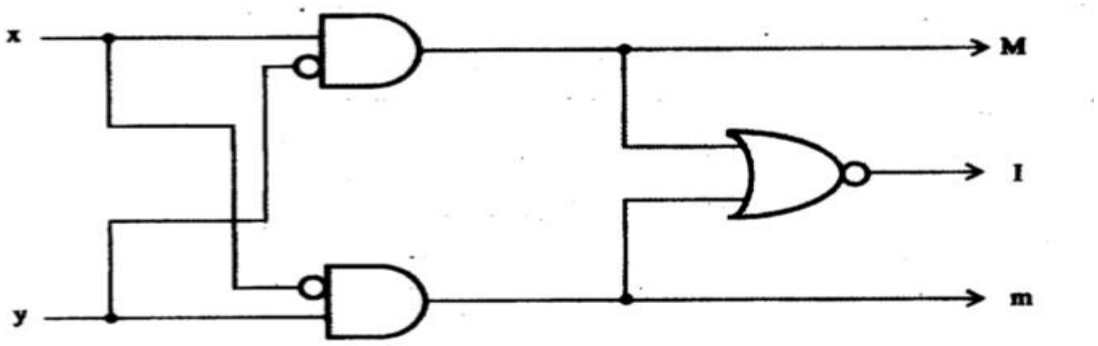


Figura 2-11, ***Comparador de 2 bits, M => x>y, I => x=y, m => x<y [Wikipedia]***

2.3.3 Resta

Para realizar la resta de dos números binarios, lo primero que se hace es compararlos, si son iguales la resta es 0, si no se coge el mayor y se le resta el menor. El signo se guarda si se ha cambiado el orden de entrada. Y a continuación se resta de derecha a izquierda, al igual que con el sumador la primera resta no tiene prestado , Bin = 0, los otros si lo pueden tener. Como hacemos el mayor menos el menor, la última operación no va a tener prestado.

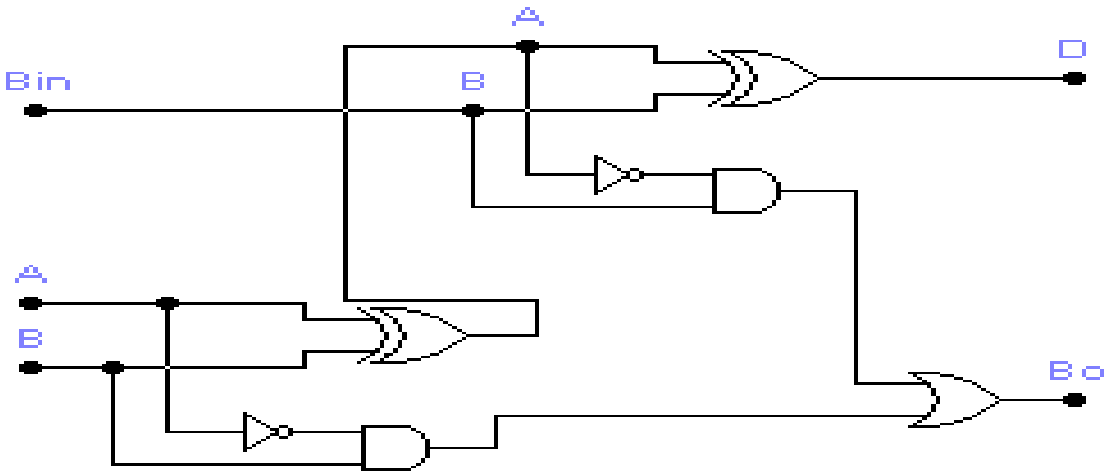


Figura 2-12, ***Resta de 2 bits [Electrónica y la Web]***

2.3.4 Multiplicación

Una forma de resumirlo fácilmente es separar la multiplicación como sumas parciales. Tenemos el primer operando que es el que vamos a sumar dependiendo del segundo. Recorremos el segundo operando de derecha a izquierda es decir desde el bit menos significativo al más significativo. Si el bit que cogemos del segundo operando es un 0 no se suma nada y si es 1 se suma el primer operando añadiéndole tantos ceros a la derecha como la posición que ocupe el bit del segundo operando. Se

empieza a contar en cero las posiciones. Por ejemplo, al multiplicar 101 (5 decimal) por 1101 (13 decimal) obtendríamos 1000001 (65 decimal)

Bit del Segundo Operando	Cantidades a Sumar
Bit del segundo Operando “1” posición 0	101
Bit del segundo Operando “0” posición 1	0
Bit del segundo Operando “1” posición 2	10100
Bit del segundo Operando “1” posición 3	101000
Resultado	1000001

Tabla 2-14: Ejemplo multiplicación

Una forma de implementarlo sería poniendo una puerta “and” por cada bit del primer operando donde una entrada es el bit del primer operando y la otra entrada es el bit del segundo operando que toque. Se recuerda que la puerta “and” sólo saca “1” por la salida si las dos entradas son “1”. Dependiendo de la posición se añadirían “0”s a la derecha, mientras se realiza la suma de los resultados.

2.3.5 División

Esquematizándolo también, y como es necesario para calcular el resto entero de una división de 2 enteros positivos, se puede simplificar diciendo que al dividir dos números enteros positivos $\frac{m}{n}$ es restar **n** repetidamente hasta que el resultado sea menor que **n**. En definitiva la concatenación de operaciones básicas de comparación, resta, comparación. En el resultado final (resto) tendremos **n** posibles valores: 0 y los **n-1** números menores que **n**. Por otra parte, el número de veces que hayamos restado **n** será el cociente. El signo se puede calcular por separado (depende solamente de los signos particulares de **m** y **n**).

2.3.6 Potencia

Como ya sabemos, una potencia entera es multiplicar el mismo número varias veces. Son multiplicaciones sucesivas dónde el factor siempre es el mismo. Ese factor se denomina base y el exponente es el número de veces que lo multiplicamos.

2.3.7 Rotación o Permutación

Esta operación es muy sencilla, consiste en leer el primer bit y almacenarlo en la posición auxiliar, a continuación leer el siguiente y escribirlo donde estaba el primer bit y así sucesivamente. Al finalizar el bit guardado en la posición auxiliar se escribe en la última posición. La rotación también puede realizarse en el orden inverso (rotación a izquierdas o a derechas).

3 ALGORITMO RSA

Este algoritmo toma su nombre en honor a sus creadores Rivest, Shamir y Adleman (1979) y es muy utilizado en la actualidad. Es un algoritmo de cifrado autenticado (permite tanto cifrar como autenticar un mensaje) y se basa en el uso de clave asimétrica, es decir existen dos claves (una pública y otra privada), una para encriptar/autenticar y otra para el proceso inverso de desencriptar/verificar.

Este algoritmo, al igual que otros, tiene como uno de sus pilares la aritmética modular o discreta y sus propiedades. El resto-módulo de la división entera de un número **p** por otro **n**, siendo **p** y **n** enteros positivos y coprimos (primos entre sí), nos da una función **periódica** (respecto al valor de p), con valor máximo **n-1**. Por ejemplo si **n = 5**, los restos posibles son 1, 2, 3, 4 el único resto que falta es el 0, pero eso implicaría que **p = k n** por lo que serían divisibles (no coprimos). Es decir si $\frac{p}{n}$ tiene resto 4 quiere decir que:

$$p = k\,n + 4$$

Ecuación 3-1

p = k n + x donde x sólo puede ser uno de estos valores (1, 2, 3, 4)

En lo sucesivo, en este documento representaremos el resto de la división entera de $\frac{p}{n}$ como:

$$[p]_n$$

Ecuación 3-2

El RSA está basado en que es fácil multiplicar números enteros, aunque sean muy grandes, pero es muy difícil computacionalmente descomponer un número entero positivo grande en factores. Por ello, en RSA tenemos un número entero positivo grande que llamaremos **n**, conocido por todos, que se sabe que es el resultado de la multiplicación de dos números primos distintos enteros grandes (**p** y **q**), supuestos desconocidos y de tamaños parecidos para complicarlo más.

El pequeño teorema de Fermat dice que siendo ***a*** un número natural distinto de 0, ***p*** primo y asumiendo que ***a*** y ***p*** son primos entre sí, entonces se cumple que:

$$[a^p]_p = [a]_p$$

Ecuación 3-3

Si dividimos la ecuación por ***a*** en los 2 lados:

$$[a^{p-1}]_p = [1]_p = 1$$

Ecuación 3-4

Una posible demostración es por inducción: primero se comprueba que para **a= 1** es cierta $\boldsymbol{([1^p]_p = [1]_p = 1)}$ después se supone cierto para **n =>** $\boldsymbol{([n^p]_p = [n]_p)}$ y finalmente se comprueba que es válido para **n+1**:

$$[(n+1)^p]_p = [(n+1)]_p => [(n+1)^p - (n+1)]_p = 0$$

Ecuación 3-5

El primer término en la parte derecha de la ecuación anterior es un binomio de newton, donde el coeficiente k-ésimo (desde $\boldsymbol{k=0}$ hasta $\boldsymbol{k=p}$) de su descomposición resultaría: $\frac{\boldsymbol{p!}}{\boldsymbol{k!(p-k)!}}$. A continuación hemos de notar que como ***p*** es primo y todos los coeficientes diferentes del primero y el último son enteros y múltiplos de ***p***, entonces tendrán módulo 0. Con lo cual,

$$[n^p + 1 - n - 1]_p = 0 => \boldsymbol{[n^p]_p = [n]_p}$$

Ecuación 3-6

qed.

La función de Euler, **φ(n)**, siendo **n** un número natural, se define como la cantidad de enteros positivos menores a **n** y coprimos con **n**. Por ejemplo 6 y 15 no son coprimos ya que el 3 es un divisor común. Algunas de las propiedades de la función de Euler son:

φ(p) = p-1, si **p** es primo. Esto se demuestra fácilmente ya que al ser primo no puede haber ningún número menor que tenga un factor que divida a los 2.

$\boldsymbol{\varphi(n)= \varphi(pq) = \varphi(p)\,\varphi(q) = (p-1)\,(q-1)}$ para **p** y **q** primos. Ya que los coprimos de **n** serán **n-1** menos los múltiplos de p=> **(q-1) p** y los múltiplos de q => **(p-1) q** al ser **p** y **q** primos:

$$\varphi(n)= (pq-1) -(p-1) - (q-1) = pq -p -q +1 = (p-1)(q-1)$$

Ecuación 3-7

En el caso que **p** y **q** sean el mismo número:

$$\varphi(n)=pp-1-(p-1) = p(p-1)$$

Ecuación 3-8

A continuación se mostrará la Ecuación más importante en la que se basa el Algoritmo RSA:

Tomamos **t** un número natural que cumple:

$$[\boldsymbol{t}]_{\boldsymbol{\varphi}\,(\boldsymbol{n})} = \mathbf{1}$$

Ecuación 3-9

Siendo **m** un número menor que **n** y que no es múltiplo ni de **p** ni de **q** se cumple:

$$[\boldsymbol{m}^{\boldsymbol{t}}]_{\boldsymbol{n}} = [\boldsymbol{m}]_{\boldsymbol{n}}$$

Ecuación 3-10

Por lo tanto se cumple:

$$[t^{p}]_{\varphi\,(n)} = [1]_{\varphi\,(n)}[t^{p-1}]_{\varphi\,(n)} = 1 \Rightarrow \; [m^{t^{p}}]_{n} = [m]_{n}$$

$$[\mathrm{t} + \mathrm{k}\varphi\,(\mathrm{n})]_{\varphi\,(\mathrm{n})} = 1 \Rightarrow \; [m^{\mathrm{t}+\mathrm{k}\varphi\,(\mathrm{n})}]_{n} = [m]_{n}$$

Ecuación 3-11

De aquí se puede sacar que: $\mathbf{[([m^{e}]_{n})^{d}]_{n} = m}$ donde $\boldsymbol{t = d\ e}$. A partir de esta expresión, podemos presentar un modelo criptográfico donde **d** y **e** serán las claves. Una de ellas será pública y la otra privada. Ya se dijo que el número ***n*** también es conocido por todos. Como se puede ver, para una misma ***n*** tenemos muchas posibles claves. La menor **t** que tenemos, sin contar el 1 es: $\boldsymbol{\varphi\,(n) + 1}$.

Encriptación: $\mathbf{[m^{e}]_{n} = c}$

Desencriptación: $\mathbf{[c^{d}]_{n} = m}$

Como se puede deducir se basa en la propiedad de los módulos: $\mathbf{[m^{ed}]_{n} = [([m^{e}]_{n})^{d}]_{n}}$

Para poder sacar la clave privada necesitaríamos saber **t (t = d e)** sabemos que:

$$[\mathrm{t}]_{\varphi\,(\mathrm{n})} = 1 \Rightarrow [\mathrm{d\ e}]_{\varphi\,(\mathrm{n})} = 1$$

Ecuación 3-12

De aquí deducimos que necesitamos saber **φ (n) => φ(n) = φ(p q) = φ(p) φ(q) = (p-1) (q-1)** para lo cual es patente que necesitamos saber **p** y **q**.

Como en el juego del gato y el ratón, para que el algoritmo sea eficiente y no se pueda romper es necesario que **p** y **q** sean suficientemente grandes y que produzcan una **n** muy grande. Por otro lado, para poder romperlo por fuerza bruta necesitamos un algoritmo de factorización eficiente que nos calcule **p** y **q** u otro Algoritmo que calcule **φ (n)**.

Demostración de:

$$[\mathbf{t}]_{\boldsymbol{\varphi}\,(\mathbf{n})} = \mathbf{1} \Rightarrow [\boldsymbol{m}^{\boldsymbol{t}}]_{\boldsymbol{n}} = [\boldsymbol{m}]_{\boldsymbol{n}}$$

Ecuación 3-13

Como ya se ha dicho la función módulo es periódica y el periodo cumple **n > periodo ≥ 0** por este motivo la función $[\boldsymbol{m}^{\boldsymbol{x}}]_{\boldsymbol{n}}$ es también periódica, es decir se repite cíclicamente, y además está acotada. Si suponemos que el periodo es ***e*** entonces:

$$[m^{e+1}]_n = [m]_n \Rightarrow [m^e]_n = [1]_n \Rightarrow [m^{e-1}]_n[m]_n = 1 \Rightarrow [m^{e-2}]_n[m^2]_n = 1 \Rightarrow \dots$$

Ecuación 3-14

Como puede verse todos los valores van a tener inversa (que multiplicados dan 1) y $[\boldsymbol{m}^{\boldsymbol{e}}]_{\boldsymbol{n}}$ tiene que valer 1. Además los módulos múltiplos de **p** y **q** no se van a poder dar porque:

$$p\,[m^x]_n = 1 \Rightarrow k\,p = s\,n + 1 \Rightarrow k\,p = s\,p\,q + 1 \Rightarrow p(k - s\,q) \neq 1$$

Ecuación 3-15

Esto no se puede dar ya que todos los valores son enteros positivos y $\mathbf{p} \neq \mathbf{1}$, no hay un número entero positivo que al multiplicarlo por **p** de 1, la conclusión es la misma para **q**.

Por este motivo tenemos que la cantidad de valores posibles para la función $[\boldsymbol{m}^{\boldsymbol{x}}]_{\boldsymbol{n}}$ queda reducida a: $\boldsymbol{\varphi}\,(\mathbf{n})$. Donde $[\boldsymbol{m}^{\boldsymbol{\varphi}\,(\mathbf{n})}]_{\boldsymbol{n}} = \mathbf{1}$ ya que tienen que ser ciclos completos y tenemos que $[\boldsymbol{m}^{\mathbf{0}}]_{\boldsymbol{n}} = \mathbf{1}$ por lo que tenemos:

$$\varphi\,(n) = k\,e$$

$$t = k\,\varphi\,(n) + 1 \Rightarrow [m^t]_n = [m^{k\,\varphi\,(n)+1}]_n = [m]_n\,[([m^{\varphi\,(n)}]_n)^k]_n = [m]_n$$

Ecuación 3-16

Con lo que queda demostrado lo que queríamos.

Tenemos que **p** y **q** son primos e impares y por tanto **n** es impar y $\boldsymbol{\varphi}\,(\mathbf{n})$ = **(p-1) (q-1)** es par y múltiplo de 4, además tenemos que si ***m*** y ***n*** son coprimos entonces $[\boldsymbol{m}^{\boldsymbol{\varphi}\,(\mathbf{n})}]_{\boldsymbol{n}} = \mathbf{1}$ y se deduce que el periodo u orden **(e)** de la función $\mathbf{[m^x]_n}$ tiene que ser un divisor de $\boldsymbol{\varphi}\,(\mathbf{n})$. Tenemos entonces que:

$$1 < e < \varphi\,(n)$$

Ecuación 3-17

El periodo **(e)** tiene que ser mayor que 1, ya que como **m** es menor que **n** el valor de $[\boldsymbol{m^1}]_{\boldsymbol{n}} = \boldsymbol{m}$ y $[\boldsymbol{m^e}]_{\boldsymbol{n}} = \mathbf{1}$ donde se desprende que el valor mínimo de **e** es 2.

El Periodo **e** de $[m^x]_n$ es $\varphi(n) = k\ e$, siendo k un número natural mayor que 0

Ecuación 3-18

Por la *Ecuación 3-18* y sabiendo que φ(n) es par y múltiplo de 4 hay muchas más posibilidades que el periodo u orden "e" de $[m^x]_n$ sea par que impar.

3.1 Algoritmo de Euclides

Encuentra el Máximo Común Divisor de 2 números. Se exponen dos ejemplos ilustrativos:

Tenemos n = 823 y k = 113 se divide de forma entera el mayor entre el menor

$823 = \mathbf{113} * 7 + \mathbf{32}$ (Tenemos como resto 32, se divide el divisor por el resto)

$113 = \mathbf{32} * 3 + \mathbf{17}$ (Se vuelve hacer lo mismo, dividir el divisor por el resto)

$32 = \mathbf{17} * 1 + \mathbf{15}$ (Ídem)

$17 = \mathbf{15} * 1 + \mathbf{2}$ (Ídem)

$15 = \mathbf{2} * 7 + \mathbf{1}$ (Ídem)

$2 = \mathbf{1} * 2 + \mathbf{0}$ (Hemos llegado al final con un 0 de resto)

Se coge el último divisor en este caso es 1 por lo que no tienen divisores en común.

Ahora tenemos n = 1547 y k = 1001 se divide de forma entera el mayor entre el menor

$1547 = \mathbf{1001} * 1 + \mathbf{546}$ (Tenemos como resto 546, se divide el divisor por el resto)

$1001 = \mathbf{546} * 1 + \mathbf{455}$ (Se vuelve hacer lo mismo, dividir divisor por el resto)

$546 = \mathbf{455} * 1 + \mathbf{91}$ (Ídem)

$455 = \mathbf{91} * 5 + \mathbf{0}$ (Hemos llegado al final con un 0 de resto)

Se coge el último divisor en este caso 91 que es el máximo común divisor de 1547 y 1001.

3.2 Algoritmo de Euclides Extendido

Este algoritmo nos serviría para encontrar la clave privada **(e)** si supiéramos la clave pública **(d)** y **φ (n)** pongamos un ejemplo n = 35 (7 * 5) por tanto $\varphi(n) = 6 * 4 = 24$ y tenemos d = 31.

$$[d\ e]_{\varphi(n)} = 1 \Rightarrow 31e = 24k + 1 \Rightarrow 31e - 24k = 1$$

Aplicamos el algoritmo de Euclides a 24 y 31

31 = 24+7 => 7 = 31-24

24= 3*7 +3 => 3 = 24 - 3*7

7 = 2*3 +1 => 1 = 7 – 2*3

$3 = 1 * 3 + 0$

Tenemos que 1= 7–2*3 = 7-2*(24 - 3*7) = 7*7 – 2*24 = 7(31-24) -2*24 = 7*31-9*24 por lo que:

k= -9 y **e= 7**

$$[d\ e]_{\varphi(n)} = 1 \Rightarrow [31*7]_{24} = 1$$

3.3 Exponenciación por cuadrados o exponenciación binaria

Este Algoritmo sirve para calcular fácilmente las potencias. Se pone un ejemplo: 7^{25}

El exponente se pasa a binario $(25)_d = (11001)_b$

El primer bit del exponente empezando por la izquierda (más significativo) siempre será “1”, el resultado se va a guardar en x, inicializamos x:= 7^2, la base al cuadrado.

Si el siguiente bit es “1” se hace x:= $(x * 7)^2 = (7^2 * 7)^2$

Si el siguiente bit es “0” se hace x:= $(x)^2 = ((7^2 * 7)^2)^2$

Si el siguiente bit es “0” se hace x:= $(x)^2 = (((7^2 * 7)^2)^2)^2$

Si el último bit es “1” se hace x:= $(x * 7)^2 = (((7^2 * 7)^2)^2)^2 * 7$

Otro ejemplo: $7^{10}{}_d = 7^{1010}{}_b$

El primer bit empezando por la izquierda (más significativo) siempre será “1”, el resultado se va a guardar en x, inicializamos x:= 7^2, la base al cuadrado

Si el siguiente bit es “0” se hace x:= $(x)^2 = (7^2)^2$

Si el siguiente bit es “1” se hace x:= $(x * 7)^2 = ((7^2)^2 * 7)^2$

Si el último bit es “0” no se hace nada x:= $((7^2)^2 * 7)^2$

3.4 Exponenciación Modular Binaria

Este Algoritmo sirve para calcular fácilmente los módulos de grandes potencias. Se basa en el algoritmo anterior. Pongamos un ejemplo: $[7^9]_{11}$

El primer paso es pasar el exponente a binario $(9)_d = (1001)_b$ y crear una variable x que se inicializa x:=7 es decir x:= base

Se coge el primer bit por la izquierda del exponente, el más significativo, siempre "1" por lo que $x:=[x]_{11}= 7$

Se sigue con el siguiente bit del exponente si es "0" entonces $x:= [x^2]_{11} = [49]_{11}=5$

Se sigue con el siguiente bit del exponente si es "0" entonces $x:= [x^2]_{11} = [25]_{11}=3$

Se sigue con el siguiente bit del exponente si es "1" entonces $[x^2\ 7]_{11} = [9*7]_{11}=8$, resultado final.

Ahora tenemos $= [8^{10}]_5$

El primer paso es pasar el exponente a binario $(10)_d = (1010)_b$ y se crea e inicializa x:=8

Se coge el primer bit por la izquierda del exponente, el más significativo, "1" se hace $x:=[x]_5= 3$

Se sigue con el siguiente bit del exponente si es "0" entonces $x:= [x^2]_5 = [9]_5=4$

Se sigue con el siguiente bit del exponente si es "1" entonces $x:= [x^2\ 8]_5 = [128]_5=3$

Se sigue con el siguiente bit del exponente si es "0" entonces $[x^2]_5 = [9]_5=4$, resultado final.

4 ALGORITMO DIFFIE-HELLMAN

Recibe este nombre porque son los apellidos de sus inventores (1976). Es un protocolo de establecimiento de claves entre partes que no han tenido contacto previo, utilizando un canal inseguro y de manera anónima (no autenticada).

El Algoritmo se basa en esta ecuación, dónde **n** y **g** son números enteros positivos, **g** es menor que **n** y mayor que 1 y no tiene ningún factor en común con **n** para que sus potencias no puedan dar resto 0 al dividirlas por **n**. El número **n** no tiene por qué ser primo pero se suele coger para tener total libertad a la hora de elegir g:

$$[g^{ab}]_n = [([g^a]_n)^b]_n = [([g^b]_n)^a]_n$$

Ecuación 4-1

Tenemos **n** y **g** conocidos por todos. A elige **a**, entero positivo y realiza $[\boldsymbol{g^a}]_{\boldsymbol{n}}$ y lo manda a B. B elige **b**, también entero positivo y realiza $[\boldsymbol{g^b}]_{\boldsymbol{n}}$ y lo manda a A. Tenemos que sólo A conoce **a** y sólo B conoce **b**. Ahora, con lo que le manda B, el nodo A calcula $\mathbf{[B^a]_n}$ y de modo paralelo B calcula $\mathbf{[A^b]_n}$, como hemos visto en la ecuación anterior tanto A como B obtienen el mismo resultado que sería la clave común a ambos. Si no conocen ni **a** ni **b** no se puede obtener la clave.

Para obtener de otro modo la clave, dados $[\boldsymbol{g^a}]_{\boldsymbol{n}}$ o $[\boldsymbol{g^b}]_{\boldsymbol{n}}$ habría que calcular el logaritmo discreto de **g** y **n**. Es decir, calcular a que número tengo que elevar **g** para obtener un número que su módulo de **n** sea **x**. Por fuerza bruta es ir haciendo potencias de **g** y calculando su módulo respecto de **n** hasta que obtengamos el valor buscado.

Este algoritmo se basa en que no existe en computación clásica un algoritmo polinomial que calcule los logaritmos discretos. Sin embargo, existe también un algoritmo cuántico que lo hace en tiempo polinomial desarrollado por Shor.

5 ALGORITMOS DE FACTORIZACIÓN

La factorización de un número entero positivo consiste en descomponerlo en una multiplicación de dos enteros o más **(n = p q)**.

La complejidad de un Algoritmo se mide en cómo crece el tiempo de computación respecto al crecimiento de **n**. Se suele medir de esta forma cuando **n** crece en un digito decimal (10 => 100) como crece el tiempo de computación. El crecimiento de un digito de **n** es como decir que ha tenido un incremento de 1 en el $\log n$. Por ahora no existe un algoritmo que en computación binaria (clásica) sea capaz de factorizar enteros positivos en la que su tiempo de computación se represente con una función polinómica que dependa del valor de $\log n$. En definitiva aún no se ha descubierto un Algoritmo que sea capaz de factorizar **n** en un tiempo que resulte de un polinomio de grado $(\log n)^k$, siendo **k** cualquier constante.

Existen algoritmos de propósito general que factorizan cualquier número y de propósito específico que está diseñado para números que cumplen ciertas propiedades. *[Información obtenida de la Wikipedia- Algoritmos de Factorización de Enteros]*

Algoritmos de propósito general:

- Algoritmo de Dixon
- Factorización con fracciones continuas
- Criba cuadrática
- Criba racional
- Factorización de formas cuadradas de Shanks
- Algoritmo general de criba del cuerpo de números

Algoritmos de propósito específico:

- División por tentativa
- Algoritmo rho de Pollard
- Algoritmo p-1 de Pollard
- Algoritmo p+1 de Williams
- Factorización de curva elíptica de Lenstra
- Método de factorización de Fermat
- Método de factorización de Euler
- Algoritmo especial de criba del cuerpo de números

El algoritmo de propósito general que tiene mejor tiempo de computación es el Algoritmo general de criba del cuerpo de números (General Number Field Sieve) que tiene un orden de magnitud que está entre las 2 últimas de la figura de abajo. Es decir entre polinómica y exponencial (subexponencial) respecto al $\log n$.

Figura 5-1 ***Órdenes de Complejidad [Wikipedia]***

6 UNIVERSO CUÁNTICO

6.1 Introducción

A diferencia de la computación binaria ahora en vez bits tenemos **qubits**.

Para ir introduciendo conceptos de forma más simple vamos a ir de menos a más. Supongamos que tenemos un tren de juguete que realiza una circunferencia de longitud 2 unidades (ver figura página siguiente). Se supone que la posición inicial, reposo, cuando está abajo es la estación A. Está claro que la posición más alejada de la posición inicial es justo cuando está arriba. En esta posición estará alejado 1 unidad, a esta posición la llamaremos estación B. Por tanto la distancia de media circunferencia equivale a 1.

Es obvio que el tren puede estar en A, en B, o en las demás posiciones de la circunferencia. Si tomamos como ***d*** la suma de la distancias sobre la circunferencia de la posición del tren a A y a B, siempre tendremos que en valor absoluto de ***d*** es 1 (media circunferencia). A la posición del tren la llamaremos el estado del **qubit** que puede ser:

$$\alpha(A) + \beta(B) \Rightarrow |\alpha|+|\beta| = 1 \Rightarrow \alpha \text{ distancia a A, } \beta \text{ distancia a B}$$

Ecuación 6-1

Para diferenciar si la posición del tren está a la izquierda o a la derecha, cuando no está en A o en B, la *Ecuación 6-1* sería para cuando este a la derecha y si el tren está por la izquierda la ecuación sería:

$$\alpha(A) - \beta(B) \Rightarrow |\alpha| + |\beta| = 1 \Rightarrow \boldsymbol{\alpha} \text{ distancia a A, } \boldsymbol{\beta} \text{ distancia a B}$$

Ecuación 6-2

La posiciones A será: **1A + 0B** y la posición B será: **0A + 1B** ó **(0A -1B)**

Superposición o **Coherencia Cuántica**, el tren puede estar en cualquier punto intermedio o en las estaciones. En la computación binaria el bit sólo podía estar en A o en B. Cuando está entre las dos estaciones se dice que en está en un estado de superposición.

La **Decoherencia** es que pasado un tiempo y/o bajo unas condiciones adversas el tren tiende a quedarse parado en la estación A. Esta estación también es el punto de partida inicial. Por lo que a medida que pasa el tiempo el tren se acerca más a A. Dicho de otro modo, para que el tren se mueva es necesario aplicar una "fuerza".

Colapsar / Medir: se supone que para que los hipotéticos pasajeros bajen o suban sólo pueda ser en las estaciones. Por lo cual para medir (que bajen o suban los pasajeros) obligamos que el tren se vaya a una estación. Lo más probable es que vaya a la estación más cercana pero no siempre se tiene que cumplir. De aquí también sacamos que la probabilidad de que vaya a A y la probabilidad que vaya a B tienen que sumar siempre 1. Pues sólo son posibles esas 2 opciones. Cuando estamos en un estado de superposición tenemos una probabilidad de que colapse a A (**|α|**) y otra probabilidad de que colapse a B

(|**β**|). Como hemos visto con la Decoherencia pasado un tiempo o en condiciones adversas la probabilidad de que vaya a A va aumentando hasta que llega a ser 1.

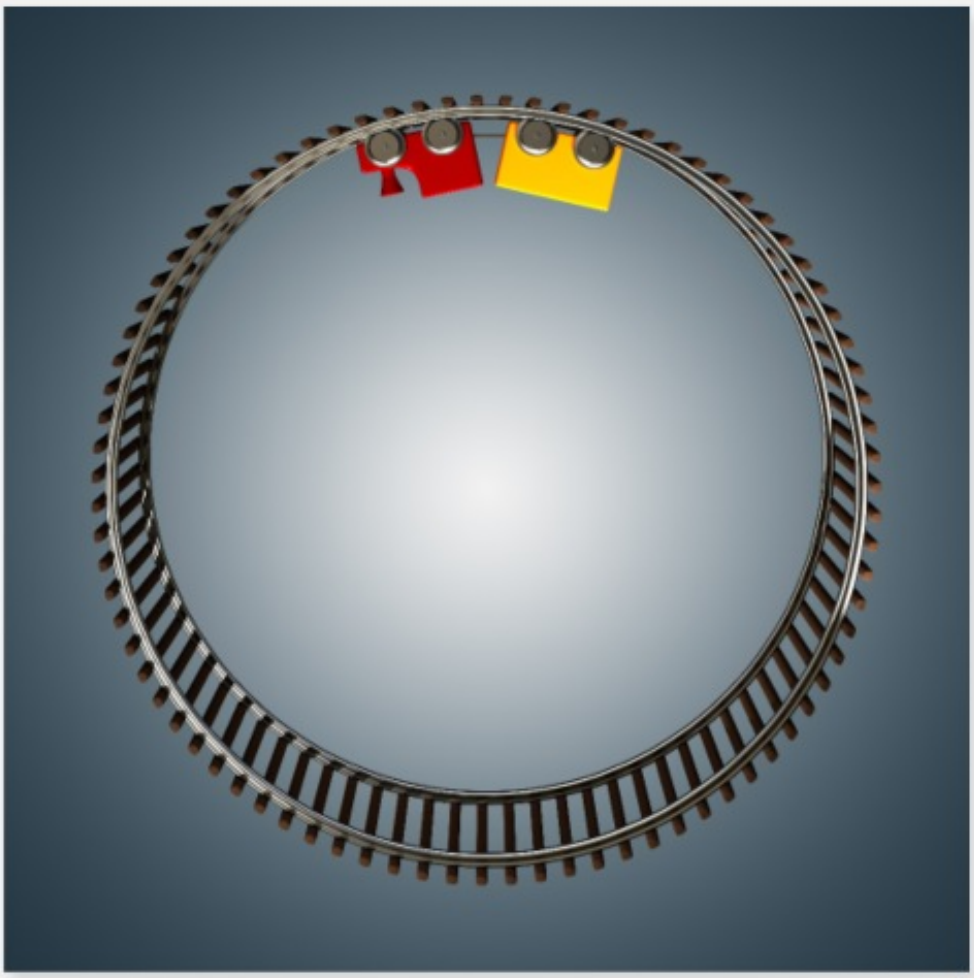


Figura 6-1, ***Tren de juguete***

Si se utiliza un espacio vectorial de 2 dimensiones. A = $\binom{1}{0}$ y B = $\binom{0}{1}$ entonces tenemos que un estado (qubit) vendría dado por la matriz $\binom{\alpha}{\beta}$ que es una combinación lineal de los 2 valores base (estaciones), dónde **α** y **β** toman valores reales: $\alpha\binom{1}{0}+\beta\binom{0}{1}$, como hemos dicho lo normal es que el **β** sea el único que pueda ser negativo.

Si ahora definimos una puerta lógica cuántica será la matriz que multiplicándola por un estado lo cambia a otro estado válido:

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix}\begin{pmatrix} \alpha_0 \\ \beta_0 \end{pmatrix} = \begin{pmatrix} \alpha_1 \\ \beta_1 \end{pmatrix} = \begin{pmatrix} a\alpha_0 + b\beta_0 \\ c\alpha_0 + d\beta_0 \end{pmatrix}$$

Ecuación 6-3

Los valores de **a, b, c, d** tienen que llevar el tren a un estado permitido es decir:

$$|\alpha_1| + |\beta_1| = 1$$

Ecuación 6-4

En la siguiente suposición vamos a pasar de 1 dimensión a 2 dimensiones. Ahora vamos a tener el mismo tren pero ahora la circunferencia va a ser de radio uno. Se supone que la Estación A, la inicial y de reposo, son las 2 posiciones de la línea horizontal y la Estación B las 2 posiciones de la línea vertical. El ángulo $\boldsymbol{\theta}$ va a ser el ángulo que forma con la horizontal (A). El qubit quedaría representado como:

$$\cos\theta\ (punto\ A) + \sin\theta\ (punto\ B) => \ (\cos\theta)^2 + (\sin\theta)^2 = 1$$

Ecuación 6-5

Las posiciones vienen dadas por los signos de los senos y cosenos en los 4 cuadrantes. La única salvedad que para los ángulos 180^0 y 270^0 tomamos su valor positivo, por lo cual equivalen a 0^0 y 90^0 respectivamente.

Ahora pasamos de 2 dimensiones a 3 dimensiones en este caso vamos a suponer que tenemos una esfera de radio 1 unidad y el objeto (qubit) puede estar en cualquier punto superficial de la esfera en este caso los estados básicos o estaciones van a ser el polo norte “0” y el polo sur “1”. Esta esfera recibe el nombre de **Bloch**, sirve para representar gráficamente un qubit.

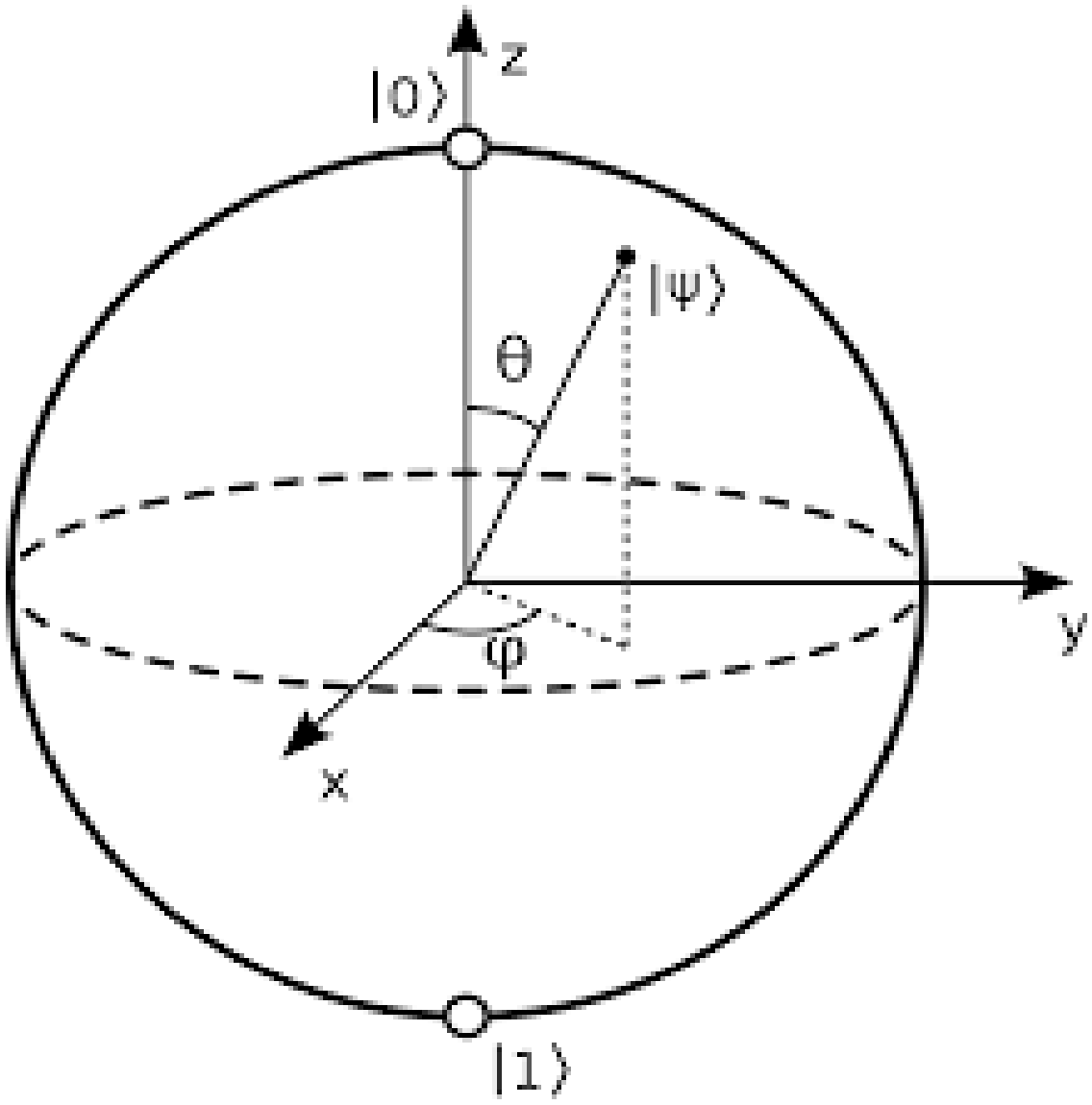


Figura 6-2, ***Esfera de Bloch [Wikipedia]***

Seguimos teniendo un espacio vectorial de 2 dimensiones $\binom{\boldsymbol{\alpha}}{\boldsymbol{\beta}}$ y para poder representar la tercera dimensión ahora **α** y **β** son números complejos no reales como antes.

Definimos el Eje **Z**, el que va de $\binom{1}{0}$ parte positiva de **Z** al centro y de ahí a $\binom{0}{1}$ parte negativa de **Z**.

Definimos el Eje **X**, el que va de $\frac{1}{\sqrt{2}}\binom{1}{1}$ (real derecha) parte positiva de **X**, al centro y de ahí $\frac{1}{\sqrt{2}}\binom{1}{-1}$ (real izquierda) parte negativa de **X**.

Definimos el Eje **Y**, el que va de $\frac{1}{\sqrt{2}}\binom{1}{i}$ (imaginario derecha) parte positiva de **Y**, al centro y de ahí a $\frac{1}{\sqrt{2}}\binom{1}{-i}$ (imaginario izquierda) parte negativa de **Y**.

Los ángulos (θ, φ, Ψ) se denominan **fase** respecto a cada eje.

La ecuación del qubit queda:

$\alpha \binom{1}{0} + \beta \binom{0}{1} = \binom{\alpha}{\beta} \Rightarrow \alpha^2 + \beta^2 = 1$ y donde α y β son números **complejos**.

$$\cos\frac{\theta}{2}\binom{1}{0} + e^{i\varphi}\sin\frac{\theta}{2}\binom{0}{1} \Rightarrow \left(\cos\frac{\theta}{2}\right)^2 + \left(e^{i\varphi}\sin\frac{\theta}{2}\right)^2 = 1$$

Ecuación 6-6

Matemáticamente un qubit puede describirse como un vector de módulo unidad en un espacio vectorial complejo bidimensional.

Por lo que tenemos que un cambio de estado de un qubit debido a una Puerta Lógica Cuántica será:

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix}\begin{pmatrix} \alpha_0 \\ \beta_0 \end{pmatrix} = \begin{pmatrix} \alpha_1 \\ \beta_1 \end{pmatrix} = \begin{pmatrix} a\alpha_0 + b\beta_0 \\ c\alpha_0 + d\beta_0 \end{pmatrix}$$

Ecuación 6-7

Donde $\boldsymbol{\alpha_0}$y $\boldsymbol{\beta_0}$ es el estado del qubit inicial y $\boldsymbol{\alpha_1}$ y $\boldsymbol{\beta_1}$ es el estado final después de haberle aplicado la puerta lógica. Todas las variables son complejas.

Se recuerda las 3 formas de representar un número complejo:

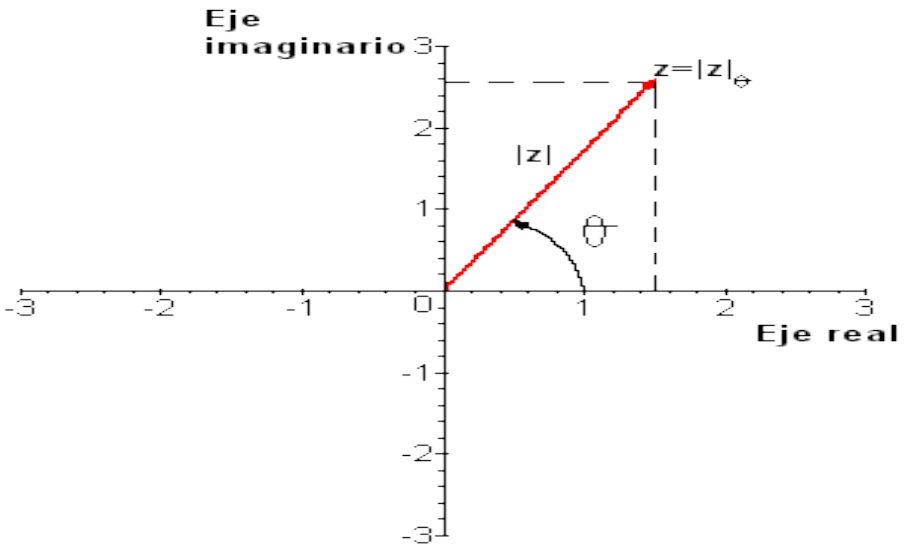


Figura 6-3, ***Representación gráfica Número Complejo [Sangakoo]***

$$a + bi$$
$$r(\cos\theta + \sin\theta\, i)$$
$$re^{\theta i}$$

Ecuación 6-8

Entrelazamiento Cuántico: Es lo más difícil de comprender del universo cuántico. Para una primera aproximación supongamos que representamos un qubit como: **cos θ** $\binom{1}{0}$ **+ sin θ** $\binom{0}{1}$ y otro qubit como: **cos** $\boldsymbol{\gamma}$ $\binom{1}{0}$ **+ sin** $\boldsymbol{\gamma}$ $\binom{0}{1}$**,** se establece la condición de que: $\boldsymbol{\theta + \gamma = 90}$. Si le hacemos un cambio al primer qubit el segundo se vería afectado por este cambio. Y si medimos el primer qubit y nos da $\binom{0}{1}$ por lo que $\theta = 90$, la medida del otro tiene que ser $\binom{1}{0}$ ya que $\gamma = 0$. Estos 2 qubits estaban entrelazados por lo que un cambio en uno afectaba al otro. Se pueden entrelazar n qubits. Un cambio en un qubit afecta a los demás independientemente de lo separados que estén, a este efecto se le ha llamado **teletrasportación**.

La representación genérica de 2 qubits sería:

$$\alpha_1\begin{pmatrix}1\\0\\0\\0\end{pmatrix} + \alpha_2\begin{pmatrix}0\\1\\0\\0\end{pmatrix} + \alpha_3\begin{pmatrix}0\\0\\1\\0\end{pmatrix} + \alpha_4\begin{pmatrix}0\\0\\0\\1\end{pmatrix} \Rightarrow (\alpha_1)^2 + (\alpha_2)^2 + (\alpha_3)^2 + (\alpha_4)^2 = 1$$

$$|00\rangle = \begin{pmatrix}1\\0\\0\\0\end{pmatrix},\ |01\rangle = \begin{pmatrix}0\\1\\0\\0\end{pmatrix},\ |10\rangle = \begin{pmatrix}0\\0\\1\\0\end{pmatrix},\ |11\rangle = \begin{pmatrix}0\\0\\0\\1\end{pmatrix}$$

Ecuación 6-9

Ahora tenemos un espacio vectorial complejo de cuatro dimensiones. Siendo $\boldsymbol{\alpha_1, \alpha_2, \alpha_3, \alpha_4}$ números complejos que indican las probabilidades de cada combinación.

Debido a la complejidad de representar matrices, se creó otra notación **Dirac (<bra|ket>)** con la cual se esquematiza el estado de los qubit. Si el estado de los qubit se quiere representar con una matriz de una fila, entonces se puede representar también con **bra** "<00|" y si se quiere representar con una matriz de una columna se puede representar con **ket** "|01>". En la *Ecuación 6-9* se han representado las 4 bases en la notación (matriz columna) y en la de ket. También se toma como notación reconocida: |+> cuando el qubit está en la parte positiva del eje real (eje x), |-> cuando el qubit está en la parte negativa del eje real (eje x), |i> cuando el qubit está en la parte positiva del eje imaginario (eje y), |-i> cuando el qubit está en la parte negativa del eje imaginario (eje y).

Como ejemplo se exponen los estados básicos de los ejes de la esfera de Bloch, en notacíón columna y en |Ket>:

Polo norte eje z, estado 0: $\begin{pmatrix} 1 \\ 0 \end{pmatrix}$ => |0> + 0|1> = |0>

Polo sur eje z, estado 1: $\begin{pmatrix} 0 \\ 1 \end{pmatrix}$ => 0|0> + |1> = |1>

Eje real positivo, eje x: $\frac{1}{\sqrt{2}}\begin{pmatrix} 1 \\ 1 \end{pmatrix}$ => $\frac{1}{\sqrt{2}}$|0> + $\frac{1}{\sqrt{2}}$|1> = |+>

Eje real negativo, eje x: $\frac{1}{\sqrt{2}}\begin{pmatrix} 1 \\ -1 \end{pmatrix}$ => $\frac{1}{\sqrt{2}}$|0> -$\frac{1}{\sqrt{2}}$|1> = |->

Eje imaginario positivo, eje y: $\frac{1}{\sqrt{2}}\begin{pmatrix} 1 \\ i \end{pmatrix}$ => $\frac{1}{\sqrt{2}}$|0> + $\frac{i}{\sqrt{2}}$|1> = |i>

Eje imaginario positivo, eje y: $\frac{1}{\sqrt{2}}\begin{pmatrix} 1 \\ -i \end{pmatrix}$ => $\frac{1}{\sqrt{2}}$|0> - $\frac{i}{\sqrt{2}}$|1> = |-i>

Para dos qubits:

$$\begin{pmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 \\ \alpha_4 \end{pmatrix} => \alpha_1|00> + \alpha_2|01> + \alpha_3|10> + \alpha_4|11>$$

Independientemente del número de qubits que tengamos siempre la suma de probabilidades de los estados que tengamos tiene que sumar 1.

$$(\alpha_1)^2 + (\alpha_2)^2 + \cdots + (\alpha_n)^2 = 1$$

Qutrit: Suponiendo el caso del tren, ahora tendríamos tres estaciones en vez de dos. No se descarta que surjan más estaciones en el futuro. El Qutrit aparte de cambiar el pensamiento binario a terciario promete que tiene menor Decoherencia por lo tanto menor error, actualmente está en estudio. *[Wikipedia Qutrit]*

6.2 Codificación

La computación basada en qubits al tener dos estados bases al igual que la computación binaria utiliza la misma codificación. El almacenamiento es un gran problema debido a la Decoherencia cuántica. En las comunicaciones tiene grandes expectativas con la teletrasportación porque felicitaría las comunicaciones seguras.

De todas formas no se espera tener un ordenador cuántico puro sino un ordenador hibrido que aproveche las ventajas de los 2.

6.3 Puertas Lógicas Cuánticas

Los ordenadores binarios están compuestos por puertas lógicas binarias AND, OR, XOR y NOT. Estas puertas están conectadas por cables/semiconductores que van de la salida de una puerta a la entrada de otra puerta. En cambio, los ordenadores cuánticos están compuestos por qubits y por puertas lógicas cuánticas, que realizan cambios en los qubits es decir en las probabilidades de obtener los estados base (estaciones). Además, las puertas cuánticas **son reversibles** a diferencia de la mayoría de las puertas lógicas clásicas. Quitando la puerta unaria NOT, en las demás, binarias, no existe una correspondencia única de entradas y salidas, por lo cual no son reversibles.

Las puertas cuánticas son representadas por matrices cuadradas, ya que tienen que ser reversibles, tienen que tener las mismas entradas que salidas. Además, estas matrices son **unitarias**, es decir, una matriz cuya inversa es igual a su conjugada transpuesta. Las puertas cuánticas tienen que llevar a los qubits a un estado permitido, es decir que su módulo sea 1.

La matriz inversa **(A^{-1})** es la que cumple: $\mathbf{A\ A^{-1} = I}$, siendo **I** la matriz identidad.

La matriz traspuesta **(A^t)** es cambiar las filas por las columnas. Por lo que en las matrices cuadradas la diagonal principal queda igual.

Un conjugado de un número complejo **(a + bi)** es cambiarle el signo a la parte imaginaría **(a-bi)**. Por lo que al sumar un número complejo más su conjugado obtenemos un real que es 2 veces la parte real del número que teníamos **(2a)**.

Como todas las puertas cuánticas son matrices unitarias y reversibles podemos determinar que su inversa que es su matriz traspuesta conjugada, lleva a los qubits al estado anterior.

Como ya se ha dicho anteriormente el aplicar una puerta lógica cuántica equivale a la multiplicación matricial del estado inicial por la matriz de la puerta.

6.3.1 Puertas Lógicas Cuánticas Unarias

Para este tipo de puertas sólo vamos a tener un qubit, por lo que tendremos una entrada y una salida. La matriz que tenemos es de 2X2. A diferencia de la computación clásica que sólo teníamos una puerta lógica unaria aquí tenemos infinitas puertas. Se van a exponer las más importantes:

6.3.1.1 Puerta Identidad

$$\begin{pmatrix}1 & 0\\ 0 & 1\end{pmatrix}\begin{pmatrix}\alpha\\ \beta\end{pmatrix} = \begin{pmatrix}\alpha\\ \beta\end{pmatrix}$$

Ecuación 6-10

6.3.1.2 Puerta Hadamard

$$\frac{\mathbf{1}}{\sqrt{\mathbf{2}}}\begin{pmatrix}\mathbf{1} & \mathbf{1}\\ \mathbf{1} & \mathbf{-1}\end{pmatrix}\begin{pmatrix}\alpha\\ \beta\end{pmatrix} = \frac{1}{\sqrt{2}}\begin{pmatrix}\alpha+\beta\\ \alpha-\beta\end{pmatrix}$$

$$\frac{\mathbf{1}}{\sqrt{\mathbf{2}}}\begin{pmatrix}\mathbf{1} & \mathbf{1}\\ \mathbf{1} & \mathbf{-1}\end{pmatrix}\begin{pmatrix}1\\ 0\end{pmatrix} = \frac{1}{\sqrt{2}}\begin{pmatrix}1\\ 1\end{pmatrix}\ \text{Estado } |+>\ \text{(Eje X)}$$

$$\frac{1}{\sqrt{2}}\begin{pmatrix}\mathbf{1} & \mathbf{1}\\ \mathbf{1} & \mathbf{-1}\end{pmatrix}\begin{pmatrix}0\\1\end{pmatrix} = \frac{1}{\sqrt{2}}\begin{pmatrix}1\\-1\end{pmatrix} \text{ Estado |-> (Eje X)}$$

Ecuación 6-11

La inversa de la puerta Hadamard es ella misma:

$$\frac{\mathbf{1}}{\sqrt{\mathbf{2}}}\begin{pmatrix}\mathbf{1} & \mathbf{1}\\ \mathbf{1} & \mathbf{-1}\end{pmatrix}\frac{\mathbf{1}}{\sqrt{\mathbf{2}}}\begin{pmatrix}1\\1\end{pmatrix} = \frac{1}{2}\begin{pmatrix}2\\0\end{pmatrix} = \begin{pmatrix}1\\0\end{pmatrix}$$

$$\frac{\mathbf{1}}{\sqrt{\mathbf{2}}}\begin{pmatrix}\mathbf{1} & \mathbf{1}\\ \mathbf{1} & \mathbf{-1}\end{pmatrix}\frac{\mathbf{1}}{\sqrt{\mathbf{2}}}\begin{pmatrix}1\\-1\end{pmatrix} = \frac{1}{2}\begin{pmatrix}0\\2\end{pmatrix} = \begin{pmatrix}0\\1\end{pmatrix}$$

Ecuación 6-12

Como se puede ver esta puerta pasa los estados básicos a una probabilidad de 50% de cada uno.

6.3.1.3 Puerta NOT o puerta X

$$\begin{pmatrix}\mathbf{0} & \mathbf{1}\\ \mathbf{1} & \mathbf{0}\end{pmatrix}\begin{pmatrix}\alpha\\\beta\end{pmatrix} = \begin{pmatrix}\beta\\\alpha\end{pmatrix}$$

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

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

Ecuación 6-13

Esta puerta intercambia las probabilidades sobre un qubit. Rotación de π radianes alrededor del eje X.

6.3.1.4 Puerta Y

$$\begin{pmatrix}\mathbf{0} & \mathbf{-i}\\ \mathbf{i} & \mathbf{0}\end{pmatrix}\begin{pmatrix}\alpha\\\beta\end{pmatrix} = \begin{pmatrix}-\beta i\\\alpha i\end{pmatrix}$$

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

$$\begin{pmatrix} \mathbf{0} & -\boldsymbol{i} \\ \boldsymbol{i} & \mathbf{0} \end{pmatrix} \begin{pmatrix} 0 \\ 1 \end{pmatrix} = \begin{pmatrix} -i \\ 0 \end{pmatrix}$$

Ecuación 6-14

Rotación de π radianes alrededor del eje Y

6.3.1.5 Puerta Z

$$\begin{pmatrix} \mathbf{1} & \mathbf{0} \\ \mathbf{0} & -\mathbf{1} \end{pmatrix} \begin{pmatrix} \alpha \\ \beta \end{pmatrix} = \begin{pmatrix} \alpha \\ -\beta \end{pmatrix}$$

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

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

Ecuación 6-15

Rotación de π radianes alrededor del eje Z.

6.3.1.6 Puertas de Rotación del Eje Z (rotación $\boldsymbol{\theta}$) Rθ

$$\begin{pmatrix} \mathbf{1} & \mathbf{0} \\ \mathbf{0} & \mathbf{cos}\,\boldsymbol{\theta} + \mathbf{sin}\,\boldsymbol{\theta}\ \boldsymbol{i} \end{pmatrix} \begin{pmatrix} \alpha \\ \beta \end{pmatrix} = \begin{pmatrix} \mathbf{1} & \mathbf{0} \\ \mathbf{0} & \boldsymbol{e^{i\theta}} \end{pmatrix} \begin{pmatrix} \alpha \\ \beta \end{pmatrix} = \begin{pmatrix} \alpha \\ \beta \boldsymbol{e^{i\theta}} \end{pmatrix}$$

Ecuación 6-16

La puerta Z pertenece a este tipo. Estas puertas al girar respecto al eje Z no cambian las distancias absolutas a los polos. Por tanto las probabilidades en valor absoluto se mantienen.

6.3.2 Puerta Lógicas Cuánticas Binarias

Las puertas binarias (2 entradas y 2 salidas) cuánticas se utilizan para cuando tenemos 2 qubits. Al igual que las unarias tenemos infinitas puertas. La matriz que tenemos es de 4x4. Detallaremos las más importantes.

6.3.2.1 Puerta de CNOT o NOT controlada

$$\begin{pmatrix} \mathbf{1} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{1} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{1} \\ \mathbf{0} & \mathbf{0} & \mathbf{1} & \mathbf{0} \end{pmatrix} \begin{pmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 \\ \alpha_4 \end{pmatrix} = \begin{pmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_4 \\ \alpha_3 \end{pmatrix}$$

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

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

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

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

Ecuación 6-17

Las puertas controladas son un tipo de puertas que se caracterizan por tener uno o varios qubits de control y otro/s de objetivo. Si no son 1 todos los qubits de control, los qubits objetivo no se modifican y si los qubits de control son todos 1 se cambia el estado los qubits objetivo. En este caso cuando el primer qubit vale 1, cambia de valor en segundo qubit, qubit objetivo, aplicando el Not.

6.3.2.2 Puerta SWAP

$$\begin{pmatrix} \mathbf{1} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{1} & \mathbf{0} \\ \mathbf{0} & \mathbf{1} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{1} \end{pmatrix} \begin{pmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 \\ \alpha_4 \end{pmatrix} = \begin{pmatrix} \alpha_1 \\ \alpha_3 \\ \alpha_2 \\ \alpha_4 \end{pmatrix}$$

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

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

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

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

Ecuación 6-18

Intercambia el |01> por |10> y viceversa.

6.3.2.3 Puerta de cambio de fase (eje Z) controlada

$$\begin{pmatrix} \mathbf{1} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{1} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{1} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \boldsymbol{e^{i\theta}} \end{pmatrix} \begin{pmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 \\ \alpha_4 \end{pmatrix} = \begin{pmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 \\ \alpha_4 e^{i\theta} \end{pmatrix}$$

$$\begin{pmatrix} \mathbf{1} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{1} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{1} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \boldsymbol{e^{i\theta}} \end{pmatrix} \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

$$\begin{pmatrix} \mathbf{1} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{1} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{1} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \boldsymbol{e^{i\theta}} \end{pmatrix} \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix}$$

$$\begin{pmatrix} \mathbf{1} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{1} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{1} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \boldsymbol{e^{i\theta}} \end{pmatrix} \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \end{pmatrix}$$

$$\begin{pmatrix} \mathbf{1} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{1} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{1} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \boldsymbol{e^{i\theta}} \end{pmatrix} \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ e^{i\theta} \end{pmatrix}$$

Ecuación 6-19

Como se puede ver sólo para |11> se aplica el cambio de fase *[ver página 35]*

6.3.3 Puerta Lógicas Cuánticas Terciarias

6.3.3.1 Puerta Toffoli o CCNOT

Con esta puerta se consigue implementar una puerta NAND, como se vio anteriormente con una puerta NAND se podía implementar el restos de puertas clásicas. Para ello es necesario 3 qubits. Actúa de forma similar a la puerta CNOT, sólo que en este caso hay dos controles, ambos han de tener el valor 1 para que se aplique la operación NOT sobre el qubit objetivo. **Con esta puerta podemos implementar las puertas clásicas por lo que se podría simular toda la lógica clásica.** Repasamos los estados básicos cuando tenemos 3 qubits. (Ket y matriz columna)

$$|000\rangle = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}, \quad |001\rangle = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}, \quad |010\rangle = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}, \quad |011\rangle = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}, \quad |100\rangle = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \end{pmatrix}, \quad |101\rangle = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \end{pmatrix},$$

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

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 \\ \alpha_4 \\ \alpha_5 \\ \alpha_6 \\ \alpha_7 \\ \alpha_8 \end{pmatrix} = \begin{pmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 \\ \alpha_4 \\ \alpha_5 \\ \alpha_6 \\ \alpha_8 \\ \alpha_7 \end{pmatrix}$$

|000> => |000>, |001> => |001>, |010> => |010>, |011> => |011>

|100> => |100>, |101> => |101>, |110> => |111>, |111> => |110>

Ecuación 6-20

AB	S
00	1
01	1
10	1
11	0

Tabla 6-1: NAND

Si el qubit más a la derecha en la entrada que va a ser la salida del NAND vale 1, se consigue una NAND binaria. Las otras entradas van a ser las entradas de la NAND.

Entrada Salida

|001> => |001> (00 => 1)

|011> => |011> (01 => 1)

|101> => |101> (10 => 1)

|111> => |110> (11 => 0)

Ecuación 6-21

6.4 El Paralelismo Cuántico

El paralelismo cuántico es la posibilidad de representar simultáneamente los valores 0 y 1. Los algoritmos cuánticos operan sobre estados de superposición y realizan simultáneamente las operaciones sobre todas las combinaciones de las entradas. En este "paralelismo cuántico" reside la potencia del cómputo cuántico. El paralelismo cuántico se describe como la capacidad para evaluar una función **f(x)** sobre varios valores de **x** de forma simultánea gracias a la superposición.

Si tenemos varios qubits separados y los queremos juntar en un mismo espacio vectorial se consigue con el **producto tensorial (⊕)**:

6.4.1 Producto Tensorial

6.4.1.1 Dos qubits:

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

$$\begin{pmatrix}\alpha_0\\ \beta_0\end{pmatrix}\oplus\begin{pmatrix}\alpha_1\\ \beta_1\end{pmatrix}=\begin{pmatrix}\alpha_0\alpha_1\\ \alpha_0\beta_1\\ \beta_0\alpha_1\\ \beta_0\beta_1\end{pmatrix}=>\alpha_0\alpha_1|00>+\alpha_0\beta_1|01>+\beta_0\alpha_1|10>+\beta_0\beta_1|11>$$

Ecuación 6-22

Tenemos que recordar que la suma de las probabilidades al cuadrado siempre tiene que sumar 1. [Página 43]

$$(\alpha_0\alpha_1)^2+(\alpha_0\beta_1)^2+(\beta_0\alpha_1)^2+(\beta_0\beta_1)^2=1$$

Ecuación 6-23

6.4.1.2 Tres qubits:

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

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

Ecuación 6-24

Sería lo mismo para n qubits.

6.4.1.3 Puertas

En el caso de las puertas es igual se pone de ejemplo la puerta Hadamard para 2 qubits. Para n qubits sería lo mismo que antes.

$$\frac{1}{\sqrt{2}}\begin{pmatrix}1 & 1\\ 1 & -1\end{pmatrix} \oplus \frac{1}{\sqrt{2}}\begin{pmatrix}1 & 1\\ 1 & -1\end{pmatrix} = \frac{1}{2}\begin{pmatrix} 1\begin{pmatrix}1 & 1\\ 1 & -1\end{pmatrix} & 1\begin{pmatrix}1 & 1\\ 1 & -1\end{pmatrix} \\ 1\begin{pmatrix}1 & 1\\ 1 & -1\end{pmatrix} & -1\begin{pmatrix}1 & 1\\ 1 & -1\end{pmatrix} \end{pmatrix} = \frac{1}{2}\begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{pmatrix}$$

Ecuación 6-25

6.4.2 Estados de Bell (2 qubits entrelazados)

$$|\psi_1> = \frac{1}{\sqrt{2}}\,(\,|00> + |11>) \Rightarrow (\frac{1}{\sqrt{2}})^2 + (\frac{1}{\sqrt{2}})^2 = 1$$

$$|\psi_1> = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$|\psi_2> = \frac{1}{\sqrt{2}}\,(\,|01> + |10>) \Rightarrow (\frac{1}{\sqrt{2}})^2 + (\frac{1}{\sqrt{2}})^2 = 1$$

$$|\psi_2> = \frac{1}{\sqrt{2}}\begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

Ecuación 6-26

Están entrelazados porque no se puede obtener como resultado de ningún producto tensorial de 2 matrices 2x2.

Se pone un ejemplo que no es un estado entrelazado y por tanto si se puede:

$$|\psi> = \frac{1}{\sqrt{2}}\,(\,|00> + |01>) \Rightarrow \frac{1}{\sqrt{2}}\begin{pmatrix}1\\1\\0\\0\end{pmatrix}$$

$$|\psi> = |0> \oplus \frac{1}{\sqrt{2}}(|0> + |1>) \Rightarrow \begin{pmatrix}1\\0\end{pmatrix} \oplus \begin{pmatrix}\frac{1}{\sqrt{2}}\\\frac{1}{\sqrt{2}}\end{pmatrix} = \frac{1}{\sqrt{2}}\begin{pmatrix}1\\1\\0\\0\end{pmatrix}$$

Ecuación 6-27

Los estados de Bell tienen la peculiaridad que si medimos un qubit dependiendo del resultado se sabe el resultado que va a tener el otro qubit sin necesidad de medirlo y perder su superposición.

Vamos a explicar una forma de llegar a los estados de Bell. Se empieza con el estado |00> estado inicial de los 2 qubits. Aplicamos una puerta Hadamard al primer qubit y después una puerta CNOT donde el qubit objetivo es el segundo qubit y el de control es el primero.

$$|\psi_0> = \begin{pmatrix}1\\0\end{pmatrix} \oplus \begin{pmatrix}1\\0\end{pmatrix}$$

$$|\psi_1> = \frac{1}{\sqrt{2}}\begin{pmatrix}1 & 1\\1 & -1\end{pmatrix}\begin{pmatrix}1\\0\end{pmatrix} \oplus \begin{pmatrix}1\\0\end{pmatrix} = \frac{1}{\sqrt{2}}\begin{pmatrix}1\\1\end{pmatrix} \oplus \begin{pmatrix}1\\0\end{pmatrix} = \frac{1}{\sqrt{2}}\begin{pmatrix}1\\0\\1\\0\end{pmatrix}$$

$$|\psi_2> \begin{pmatrix}1 & 0 & 0 & 0\\0 & 1 & 0 & 0\\0 & 0 & 0 & 1\\0 & 0 & 1 & 0\end{pmatrix}\frac{1}{\sqrt{2}}\begin{pmatrix}1\\0\\1\\0\end{pmatrix} = \frac{1}{\sqrt{2}}\begin{pmatrix}1\\0\\0\\1\end{pmatrix} = \frac{1}{\sqrt{2}}\,(\,|00> + |11>)$$

Ecuación 6-28

Para obtener el otro estado de Bell aplicamos una puerta X al segundo qubit antes de aplicar la CNOT.

$$|\psi_0> = \binom{1}{0} \oplus \binom{1}{0}$$

$$|\psi_1> = \frac{1}{\sqrt{2}}\begin{pmatrix}1 & 1\\ 1 & -1\end{pmatrix}\binom{1}{0} \oplus \begin{pmatrix}0 & 1\\ 1 & 0\end{pmatrix}\binom{1}{0} = \frac{1}{\sqrt{2}}\binom{1}{1} \oplus \binom{0}{1} = \frac{1}{\sqrt{2}}\begin{pmatrix}0\\1\\0\\1\end{pmatrix}$$

$$|\psi_2> \begin{pmatrix}1 & 0 & 0 & 0\\ 0 & 1 & 0 & 0\\ 0 & 0 & 0 & 1\\ 0 & 0 & 1 & 0\end{pmatrix}\frac{1}{\sqrt{2}}\begin{pmatrix}0\\1\\0\\1\end{pmatrix} = \frac{1}{\sqrt{2}}\begin{pmatrix}0\\1\\1\\0\end{pmatrix} = \frac{1}{\sqrt{2}}\ (|01> + |10>)$$

Ecuación 6-29

6.4.3 El controlador controlado

Como se ha visto anteriormente en los estados de Bell aunque en las puertas controladas hay unos qubits que controlan y otros que son controlados en el caso especial que los qubits controlados y controladores se entrelazan pasan a depender uno de los otros. Es decir los controladores dependen de los controlados y viceversa. Lo que ocurre en este ejemplo es una de las bases del paralelismo cuántico, el entrelazamiento.

Pongamos otro ejemplo vamos a trabajar con la puerta de cambio de fase controlada (Z) de $\emptyset$**.** La puerta normal de cambio de fase como se puede ver sólo afecta al estado |1> y su autovalor es $\boldsymbol{e^{\emptyset i}}$**.**

$$\begin{pmatrix}\mathbf{1} & \mathbf{0}\\ \mathbf{0} & \boldsymbol{e^{\emptyset i}}\end{pmatrix}\binom{1}{0} = |0>$$

$$\begin{pmatrix}\mathbf{1} & \mathbf{0}\\ \mathbf{0} & \boldsymbol{e^{\emptyset i}}\end{pmatrix}\binom{0}{1} = \boldsymbol{e^{\emptyset i}}\ |1>$$

Ecuación 6-30

Con la puerta controlada tenemos, que si aplicamos una puerta Hadamard al primer qubit (controlador) que parte de |0> lo convertimos a |+> y después aplicamos la puerta de cambio de fase controlada obtenemos:

$$\frac{1}{\sqrt{2}}\begin{pmatrix}1\\1\end{pmatrix} \oplus \begin{pmatrix}1\\0\end{pmatrix} = \frac{1}{\sqrt{2}}\begin{pmatrix}1\\0\\1\\0\end{pmatrix} => \frac{1}{\sqrt{2}}\begin{pmatrix}1&0&0&0\\0&1&0&0\\0&0&1&0\\0&0&0&e^{\emptyset i}\end{pmatrix}\begin{pmatrix}1\\0\\1\\0\end{pmatrix} = \frac{1}{\sqrt{2}}\begin{pmatrix}1\\0\\1\\0\end{pmatrix} => No\ hace\ nada$$

$$\frac{1}{\sqrt{2}}\begin{pmatrix}1\\1\end{pmatrix} \oplus \begin{pmatrix}0\\1\end{pmatrix} = \frac{1}{\sqrt{2}}\begin{pmatrix}0\\1\\0\\1\end{pmatrix} => \frac{1}{\sqrt{2}}\left(|01> + |11>\right) => \frac{1}{\sqrt{2}}\begin{pmatrix}1&0&0&0\\0&1&0&0\\0&0&1&0\\0&0&0&e^{\emptyset i}\end{pmatrix}\begin{pmatrix}0\\1\\0\\1\end{pmatrix} = \frac{1}{\sqrt{2}}\begin{pmatrix}0\\1\\0\\e^{\emptyset i}\end{pmatrix} =>$$

$$\frac{1}{\sqrt{2}}\left(|01> + e^{\emptyset i}|11>\right)$$

Ecuación 6-31

Como se puede ver la puerta sólo realiza el cambio de fase ($\boldsymbol{e^{\emptyset i}}$) cuando el qubit objetivo es |1> y el qubit de control también es |1>. Es decir aplica el cambio de fase sólo a la componente $\boldsymbol{\beta}$ de un qubit que está en estado de superposición.

6.4.4 Algoritmo de Deutsch-Jozsa

Para explicar mejor lo que es el paralelismo cuántico utilizaremos como ejemplo el Algoritmo de Deutsch-Jozsa. Tenemos que adivinar si una función es constante devuelve siempre “0” o “1” o es variable devuelve la mitad de las veces “0” y la otra mitad “1”.

Para empezar cogeremos una función de un solo bit de entrada. Como ya vimos, existen cuatro posibles funciones. La que devuelve siempre “0”, la que devuelve siempre “1” también constante, la que devuelve la entrada a la salida, función identidad, que es variable y la que devuelve lo contrario de la entrada función Not, también variable. Por lo que tenemos dos constante y dos variables. Si tengo como entrada “0” y me devuelve “0” no sabré si es constante o variable, necesitaré meter la entrada “1” para saber cuál de las 2 posibilidades tengo. Esto ocurre para cualquier entrada y cualquier salida siempre necesitaré ejecutar la función con las 2 entradas posibles. Es decir, ejecutarla 2 veces. Pero con un circuito cuántico sólo necesitamos ejecutar la función una vez.

Entrada	Zero	Uno	I	NOT
0	0	1	0	1
1	0	1	1	0

Tabla 6-2: Funciones de 1 bit

Tenemos 2 qubit inicializados a **x** = |0>, **y** = |0> al bit **y** le aplicamos una puerta **X(Not)** con lo que tenemos **y** = |1>. A los 2 qubits le aplicamos una puerta Hadamard con lo que conseguimos que se superpongan, se detallan los cambios de estados:

$$|x_0, y_0> = \binom{1}{0} \oplus \binom{1}{0}$$

$$|x_1, y_1> = \binom{1}{0} \oplus \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \binom{1}{0} = \binom{1}{0} \oplus \binom{0}{1}$$

$$|x_2, y_2> = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \binom{1}{0} \oplus \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \binom{0}{1} = \frac{1}{2} \binom{1}{1} \oplus \binom{1}{-1} = \frac{1}{2} \begin{pmatrix} 1 \\ -1 \\ 1 \\ -1 \end{pmatrix}$$

$$= \frac{1}{2} (|00> - |01> + |10> - |11>)$$

Ecuación 6-32

Ahora pasamos los 2 qubits por una puerta que saca por la primera salida **"x"** sin modificar y por la segunda **"y XOR f(x)"**:

$$|x_2, y_2 \; XOR \; f(x_2)> = \begin{pmatrix} f(0) & 0 & 0 & 0 \\ 0 & f(0) & 0 & 0 \\ 0 & 0 & f(1) & 0 \\ 0 & 0 & 0 & f(1) \end{pmatrix} \frac{1}{2} \begin{pmatrix} 1 \\ -1 \\ 1 \\ -1 \end{pmatrix} = \begin{pmatrix} f(0) \\ -f(0) \\ f(1) \\ -f(1) \end{pmatrix} =$$

$$= \frac{1}{2} (|0,0 \; XOR \; f(0)> - |0,1 XOR \; f(0)> + |1,0 XOR \; f(1)> - |1,1 XOR f(1)>)$$

Ecuación 6-33

Para el primer sumando tenemos que el XOR va a dar lo que sea **f(0)** para el segundo lo contrario de **f(0)**, el tercero lo que sea **f(1)** y el cuarto lo contrario que sea **f(1)**. La ecuación se queda:

$$|x_2, y_2 \; XOR \; f(x_2)> = \frac{1}{2} (|0, f(0)> - |0, NOT f(0)> + |1, f(1)> - |1, NOT f(1)>)$$

$$|x_2, y_2 \; XOR \; f(x_2)> = \frac{1}{2} (|0> \oplus (|\, f(0)> - |\, NOT f(0)>) + |1> \oplus (\, |f(1)> - |NOT f(1)>))$$

Ecuación 6-34

- Si es constante **f(0) = f(1)**

$$|x_2, y_2 \; XOR \; f(x_2)> = \frac{1}{2} (|0> \oplus (|\, f(0)> - |\, NOT f(0)>) + |1> \oplus (\, |f(0)> - |NOT f(0)>))$$

$$|x_2, y_2\ XOR\ f(x_2)> = \frac{\mathbf{1}}{\sqrt{\mathbf{2}}}\ (|\mathbf{0}>\ +|\mathbf{1}>) \oplus \frac{1}{\sqrt{2}}(|\ f(0)>\ -|\ NOTf(0)>) =$$

$$= \frac{\mathbf{1}}{\sqrt{\mathbf{2}}}\begin{pmatrix}1\\1\end{pmatrix} \oplus \frac{1}{\sqrt{2}}\begin{pmatrix}f(0)\\-notf(0)\end{pmatrix}$$

Ecuación 6-35

- Si es variable **f(0) ≠ f(1)**

$$|x_2, y_2\ XOR\ f(x_2)> = \frac{1}{2}\ (|0> \oplus (|\ f(0)> -|\ NOTf(0)>) + |1> \oplus (\ |NOT\ f(0)> -|f(0)>))$$

$$|x_2, y_2\ XOR\ f(x_2)> = \frac{\mathbf{1}}{\sqrt{\mathbf{2}}}\ (|\mathbf{0}>\ -|\mathbf{1}>) \oplus \frac{1}{\sqrt{2}}(|\ f(0)>\ -|\ NOTf(0)>)=$$

$$= \frac{\mathbf{1}}{\sqrt{\mathbf{2}}}\begin{pmatrix}1\\-1\end{pmatrix} \oplus \frac{1}{\sqrt{2}}\begin{pmatrix}f(0)\\-notf(0)\end{pmatrix}$$

Ecuación 6-36

Si ahora al primer qubit le aplicamos una puerta Hadamard obtenemos:

- Si es constante:

$$\frac{\mathbf{1}}{\sqrt{\mathbf{2}}}\begin{pmatrix}\mathbf{1} & \mathbf{1}\\ \mathbf{1} & \mathbf{-1}\end{pmatrix} X \frac{\mathbf{1}}{\sqrt{\mathbf{2}}}\begin{pmatrix}1\\1\end{pmatrix} = \frac{1}{2}\begin{pmatrix}2\\0\end{pmatrix} = \begin{pmatrix}1\\0\end{pmatrix}\ Estado\ |0>$$

Ecuación 6-37

- Si es variable

$$\frac{\mathbf{1}}{\sqrt{\mathbf{2}}}\begin{pmatrix}\mathbf{1} & \mathbf{1}\\ \mathbf{1} & \mathbf{-1}\end{pmatrix} X \frac{\mathbf{1}}{\sqrt{\mathbf{2}}}\begin{pmatrix}1\\-1\end{pmatrix} = \frac{1}{2}\begin{pmatrix}0\\2\end{pmatrix} = \begin{pmatrix}0\\1\end{pmatrix}\ Estado\ |1>$$

Ecuación 6-38

Hemos demostrado que si medimos el qubit **x**, si nos da "0" la función que tenemos es constante y si nos da "1" la función que tenemos es variable. En este ejemplo el beneficio es menor, pero si en vez de tener una función con una entrada la tenemos de **n** entradas, entonces sí cobraría una gran relevancia. Se necesitaría en el peor de los casos con la computación binaria $\frac{\boldsymbol{n}}{\mathbf{2}} + \mathbf{1}$ resultados de **f(x)** para saber si es variable o constante y con la computación cuántica con ejecutar sólo una vez la función lo sabríamos.

En el caso de n entradas necesitaríamos **n+1** qubits, al qubit que se añade se le aplica una puerta NOT y se le aplica a todos los qubits una puerta Hadamard que los superponga. Posteriormente se le aplicaría la puerta de antes que saca por el qubit que se ha añadido **{y XOR f(x)}** y por las otras salidas las entradas sin modificar. Posteriormente se pasan los primeros **n** qubits por una puerta Hadamard. Si el resultado es todo 0 entonces es constante, sino es variable.

Ejemplo para una función de 2 bits. Ahora necesitamos 3 qubits:

$$|x_0, y_0, z_0> = \binom{1}{0} \oplus \binom{1}{0} \oplus \binom{1}{0}$$

$$|x_1, y_1, z_1> = \binom{1}{0} \oplus \binom{1}{0} \oplus \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}\binom{1}{0} = \binom{1}{0} \oplus \binom{1}{0} \oplus \binom{0}{1}$$

$$|x_2, y_2, z_2> = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}\binom{1}{0} \oplus \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}\binom{1}{0} \oplus \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}\binom{0}{1} =$$

$$= \frac{1}{2\sqrt{2}}\binom{1}{1} \oplus \binom{1}{1} \oplus \binom{1}{-1} = \frac{1}{2\sqrt{2}}\begin{pmatrix} 1 \\ -1 \\ 1 \\ -1 \\ 1 \\ -1 \\ 1 \\ -1 \end{pmatrix}$$

$$= \frac{1}{2\sqrt{2}}(|000> -|001> +|010> -|011> +|100> -|101> +|110> -|111>)$$

$$|x_2, y_2, z_2\ XOR\ f(x_2, y_2)> = \frac{1}{2\sqrt{2}}\begin{pmatrix} f(0,0) \\ -notf(0,0) \\ f(0,1) \\ -notf(0,1) \\ f(1,0) \\ -notf(1,0) \\ f(1,1) \\ -notf(1,1) \end{pmatrix}$$

Ecuación 6-39

- Si es Constante **f(0,0) = f(0,1)= f(1,0) = f(1,1)**

$$|x_2, y_2, z_2\ XOR\ f(x_2, y_2) >= \frac{1}{2\sqrt{2}} \begin{pmatrix} f(0,0) \\ -notf(0,0) \\ f(0,0) \\ -notf(0,0) \\ f(0,0) \\ -notf(0,0) \\ f(0,0) \\ -notf(0,0) \end{pmatrix} =$$

$$= \frac{1}{\sqrt{2}}\begin{pmatrix}1\\1\end{pmatrix} \oplus \frac{1}{\sqrt{2}}\begin{pmatrix}1\\1\end{pmatrix} \oplus \frac{1}{\sqrt{2}}\begin{pmatrix} f(0,0) \\ -notf(0,0) \end{pmatrix}$$

Ecuación 6-40

Si le aplicamos una puerta Hadamard a los 2 primeros qubits y los medimos obtenemos: |00> por lo que se puede determinar que es constante. En caso contrario sería variable.

6.5 Notación

A continuación se resume la notación típica de los circuitos cuánticos. En la izquierda se nombra los qubits, de abajo a arriba, del menos significativo al más significativo. Se separan por bloques, por ejemplos qubits objetivo y qubits de control, se indica el valor inicial. Si es un |1> se supone que se ha aplicado una puerta X. Un qubit se representa con una línea hasta el final, parte derecha. Si una puerta afecta a sólo un qubit es un rectángulo que sólo afecta a ese 1ubit y si el rectángulo engloba a más de un qubit significa que afecta a todos los qubits. En el interior de la caja se nombra el tipo de puerta que es, por ejemplo si tiene 90, indica que es una puerta de cambio de fase de ángulo 90. Si es una H, que es una puerta Hadamard.

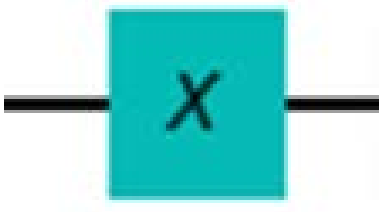


Figura 6-4, X = NOT [DocIRS]

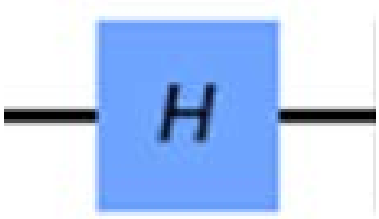


Figura 6-5, HADAMARD [DocIRS]

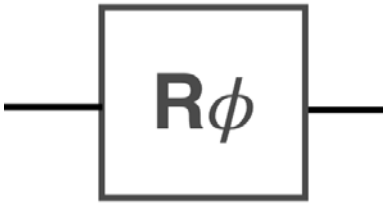
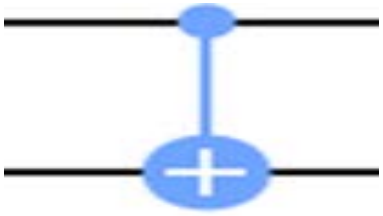


***Figura 6-6, CAMBIO DE FASE Z* [DocIRS]**



Figu*ra 6-7, CX= CNOT Controlada* [DocIRS]

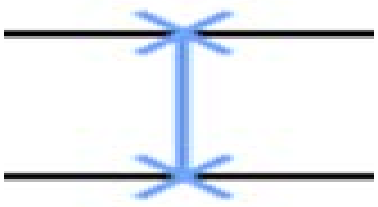


Figura 6-8, Swap [DocIRS]

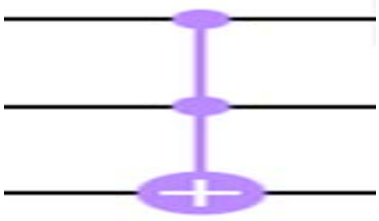


Figura 6-9, C*CX=Toffoli* [DocIRS]

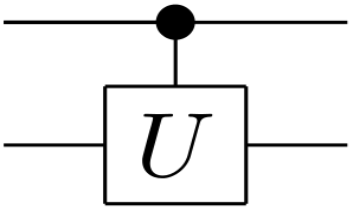
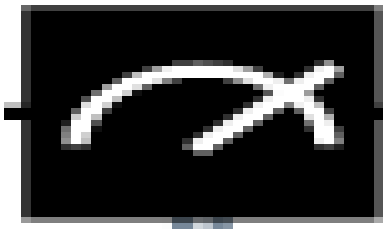


Figura *6-10, U de un qubit controlada por un qubit* [DocIRS]



Fig*ura 6-11, Medición de un qubit* [DocIRS]

7 ALGORITMO DE SHOR

7.1 Algoritmo de Shor

Este algoritmo encuentra dos números **p** y **q** que dividen **n**. Todos ellos enteros positivos. Para lo cual realiza varios pasos:

1. Se elige un número entero pseudo-aleatorio ***k*** menor que ***n-1*** y mayor que **1**.
2. Miramos si ***k*** y ***n*** tienen divisores en común distintos de **1** utilizando el algoritmo de Euclides *[ver página 32]* , si los tiene ya hemos encontrado un divisor de ***n*** (es decir, a ***p*** o ***q*)** en caso contrario continuamos.
3. **Ejecutar parte cuántica.** Se busca el periodo (***e***) de la función $[k^e]_n$. Encontrar el periodo de esta función cuando ***n*** es muy grande es un proceso que conlleva muchísimo tiempo de computación con los computadores actuales.
4. Si con esa ***k*** no encontramos **p** y **q** volvemos al paso 1 cogemos otro número que no hayamos cogido antes.

Este algoritmo resuelve el problema de factorización de números enteros en tiempo polinomial.

$$\mathbf{O(\log(n)^3)}$$

Ecuación 7-1

7.1.1 Fundamentos del Algoritmo de Shor

Tenemos un número **n** que es resultado de multiplicar **p** y **q**. Si **m** cumple:

$$m^2 = kn + 1 \implies m^2 - 1 = kn \implies (m-1)(m+1) = kn$$

Ecuación 7-2

Si **(m-1)** y **(m+1)** no son múltiplos de **n**. Entonces **(m-1)** es múltiplo de **p** ó **q** y **(m+1)** será múltiplo del otro que no lo sea de **(m-1)**. Ya que si **m** no tiene factores en común con **n** no puede ser múltiplo de **p** y **q** a la vez. Se buscarían **p** y **q** aplicando el algoritmo de **Euclides** *[ver página 32]*. Si directamente **m-1** y **m+1** son menores que n hemos encontrado la solución ya que no se puede dar otra situación.

Pongamos ejemplos:

n = 15 y m = 7, como se sabe no tienen divisores en común:

$$7^4 = 49^2 = 2401 = 15 * 160 + 1 \implies (48)\,(50) = 160*15$$

Por Euclides se puede sacar que **3** es el máximo común divisor de 48 y 15 y que **5** es el máximo común divisor de 50 y 15. En este ejemplo hemos tenido un final exitoso, hemos descubierto que 15 = 3·5.

Veamos otros ejemplos no exitosos.

n = 57 y m = 7, no tienen divisores en común:

$$7^3 = 343 = 57 * 6 + 1$$

En este ejemplo la potencia que hemos obtenido es impar y no podemos seguir.

Otro ejemplo: n = 33 y m = 32, no tienen divisores en común:

$$32^2 = 1024 = 33*31 + 1 \Rightarrow (32-1) * (32+1) = (31) * (33)$$

En este ejemplo no tenemos éxito ya que **(m+1)** es múltiplo de **n**. De aquí se deduce por qué m tiene que ser menor que **n-1**.

Otro ejemplo: n = 342 y m = 7, no tienen divisores en común:

$$7^6 = 343^2 = 117649 = 342 * 344 + 1 \Rightarrow (342) * (344)$$

En este ejemplo no tenemos éxito ya que **$(m^3-1) = 342$** es múltiplo de **n**.

7.2 Transformada Cuántica de Fourier (QFT)

La transformada cuántica de Fourier es la implementación cuántica de la transformada discreta/rápida de Fourier. Forma parte de muchos algoritmos cuánticos y también del algoritmo de Shor. Se trata de una transformación lineal que codifica una entrada en binario (generalmente representando un número) en rotaciones del eje Z sin cambiar las probabilidades de los estados de los qubits. Se codifica el número como un desplazamiento de fase en la circunferencia ecuador de la esfera de Bloch *[ver página 40]* (rotaciones del eje Z). Se necesitan los mismos qubits que bits para representar el resultado para que sea reversible el cambio. Es decir **n** qubits para representar un número igual o menor que 2^n.

Vamos a poner un ejemplo para representar números de 3 bits, necesitamos 3 qubits. Inicializamos los 3 qubits a |0> y después pasamos los 3 qubits cada uno por puertas Hadamard independientes. Por lo que se nos quedan los 3 qubits en el estado |+>. La representación de los 8 números sería:

0 => 000 => |+,+,+>

1 => 001 => $|\frac{\pi}{4}+,\frac{\pi}{2}+,\pi>$

2 => 010 => $|\frac{2\pi}{4}+,\pi+,2\pi>$

3 => 011 => $|\frac{3\pi}{4}+,\frac{3\pi}{2}+,3\pi>$

4 => 100 => $|\frac{4\pi}{4}+,2\pi,4\pi>$

5 => 101 => $|\frac{5\pi}{4}+,\frac{5\pi}{2}+,5\pi>$

6 => 110 => $|\frac{6\pi}{4}+,3\pi+,6\pi>$

7 => 111 => $|\frac{7\pi}{4}+,\frac{7\pi}{2}+,7\pi>$

Como se puede ver el qubit más significativo tiene guardada toda la información dependiendo de lo que ha girado respecto a +, ni siquiera gira una vuelta completa para el máximo valor, los demás han girado más de una vuelta siendo el menos significativo el que más gira. Si tuviéramos un número de 4 bits necesitaríamos 4 qubits y el máximo de giro del qubit más significativo sería $\frac{15}{8}\pi$ y así sucesivamente para n (bits/qubits) $\frac{2^n-1}{2^n}\pi$. Dependiendo de lo deseado puede ser necesario cambiar el orden, es decir que el qubit menos significativo sea el que menos gire y a medida que sean más significativos giren más. Independientemente del primer qubit que elijamos, el qubit con mayor fase siempre girará en múltiplos de π $(media\ vuelta)$.

Se recuerda que la puerta de giro en Z era: $\begin{pmatrix}\mathbf{1} & \mathbf{0}\\ \mathbf{0} & \boldsymbol{e^{i\theta}}\end{pmatrix}$ siendo $\boldsymbol{\theta}$ el ángulo de giro en radianes.

7.3 Transformada Inversa de Fourier Cuántica

Sirve para obtener el valor (número) guardado en n qubits. Se pone un ejemplo con 3 qubits para indicar como sería, se supone que es el qubit más significativo el que posee menor fase por lo que el menos significativo es el que ha girado más, es decir vamos a ir de derecha a izquierda, también se supone que el punto de partida viene de aplicar una Transformada Cuántica de Fourier o similar.

Se aplica una puerta Hadamard al primer qubit, el que gira múltiplos de π, será |+> ó |-> y se mide:

- Si el resultado es "0" es que hay un número par de giros en el primer qubit (000,010,100,110) como es un número par de giros el segundo qubit será |+> ó |-> por lo que se le aplica una puerta Hadamard y se mide:
  - Si el resultado es "0" es que hay un número par de giros en el segundo qubit (000,100), el tercer qubit será |+> ó |-> por lo que se le aplica una puerta Hadamard y se mide:
    - Si el resultado es "0" es que el número es "000", **0**.
    - Si el resultado es "1" es que el número es "100", **4**.
  - Si el resultado es "1" es que hay un número impar de giros en el segundo qubit (010,110), el tercer qubit será $|\frac{\pi}{2}+>$ ó $|\frac{3\pi}{2}+>$ por lo que se le aplica una puerta de rotación de $(\frac{\pi}{2})$ y posteriormente una Hadamard y se mide:
    - Si el resultado es "1" es que el número es "010", **2**.
    - Si el resultado es "0" es que el número es "110", **6**.

- Si el resultado es "1" es que hay un número impar de giros (001,011,101,111) como es un número impar de giros el segundo qubit necesita $\frac{\pi}{2}$ para ser |+> ó |-> por lo que se le aplica una puerta de rotación de ($\frac{\pi}{2}$) y posteriormente una puerta Hadamard y se mide:
    - Si el resultado es "1" es que hay un número par de giros en el segundo qubit (001,101), el tercer qubit necesita $\frac{3\pi}{4}$ para ser |+> ó |-> por lo que se le aplica una puerta de rotación de ($\frac{3\pi}{2}$) y posteriormente una puerta Hadamard y se mide:
        - Si el resultado es "1" es que el número es "001", **1**.
        - Si el resultado es "0" es que el número es "101", **5**.
    - Si el resultado es "0" es que hay un número impar de giros en el segundo qubit (011,111), el tercer qubit necesita $\frac{\pi}{4}$ para ser |+> ó |-> por lo que se le aplica una puerta de rotación de ($\frac{\pi}{4}$) y posteriormente una puerta Hadamard y se mide:
        - Si el resultado es "1" es que el número es "011", **3**.
        - Si el resultado es "0" es que el número es "111", **7**.

Si tuviéramos más qubits se seguiría con el mismo procedimiento.

7.4 Estimación Cuántica de Fase

Al igual que la Transformada Cuántica de Fourier y su inversa, la Estimación Cuántica de Fase se utiliza en muchos algoritmos cuánticos y por supuesto en el algoritmo de Shor. Dado una puerta cuántica (U) de varios qubits **(*m*)** que cambia el estado de los ***m*** qubits a otro estado, la Estimación Cuántica de Fase estima el **autovalor (fase)** de ese cambio. Es decir estima que cambio de fase se produce el aplicar la puerta U a los ***m*** qubits.

$$\mathbf{U|\gamma\rangle = e^{2\pi\emptyset i}|\gamma\rangle = (\cos 2\pi\emptyset + i \sin 2\pi\emptyset)\,|\gamma\rangle}$$

Ecuación 7-3

Donde $\emptyset$ indicará el tanto por uno de una vuelta completa (2π) que produce U y será el valor a estimar entonces:

$$1>\emptyset\geq 0$$

Ecuación 7-4

El valor 1 y 0 significa lo mismo, ya que supone vueltas completas. Por eso sólo se toma el 0, es decir, múltiplos de 2π.

Para poder realizarlo añadimos **n** qubits, qubits controladores, dependiendo de la aproximación que esperemos conseguir, teniendo en cuenta que se consiguen 2^n valores posibles de $\emptyset$. Es decir:

$$\emptyset = x_0 2^{-1} + x_1 2^{-2} + \cdots + x_{n-1}\, 2^{-n}$$

Ecuación 7-5

Como se puede observar el valor 1 no se alcanza pero como ya hemos visto ese valor estaría recogido con el valor 0. Luego transformamos U en una U controlada por un qubit, qubit controlador, que como vimos anteriormente se verá afectado cuando se aplica la U controlada y en el componente $\boldsymbol{\beta}$. Está afectación va a ser proporcional a la fase que aplica U. Dependiendo de la aproximación que queramos utilizamos un número **n** de qubits controladores. Por supuesto es necesario aplicarle puertas Hadamard a los **n** qubits controladores para que estén en estado de superposición. Por cada qubit controlador se aplica un número de veces (potencia de 2) la puerta U controlada dependiendo de su posición, es decir para el primer qubit controlador se aplica una vez la U controlada, para el segundo 2 veces, para el tercero 4 veces, para el cuarto 8 veces, … :

$$\frac{1}{\sqrt{2^n}} \sum_{k=0}^{n-1} \left(\prod_{t=0}^{n-1} \boldsymbol{e}^{\frac{\boldsymbol{\pi i 2^t}}{2^k}} \right) \ |k_{|1>}>$$

Ecuación 7-6

La Ecuación 7-6 refleja el sumatorio de cambios de fase que hemos hecho. La **k** indica la posición del qubit controlador, siendo 0 para el menos significativo y **n-1** para el más significativo. Y $|k_{|1>}>$ indica que la parte afectada es el estado |1> de ese qubit. Como se puede ver es aplicar la Transformada Cuántica de Fourier. Por lo que se deduce que para estimar la fase ($\emptyset$) será necesario realizar después la Transformada Cuántica Inversa de Fourier y medir para obtener el resultado (fase).

Pongamos un ejemplo con una puerta U de cambio de fase $\begin{pmatrix} 1 & 0 \\ 0 & e^{\frac{\pi i}{4}} \end{pmatrix}$ aplicada a un solo qubit por lo que m=1 y tomamos para la estimación n = 3 qubits (control) para el cálculo, entonces:

$$\frac{1}{\sqrt{2^3}} \sum_{k=0}^{2} \left(\prod_{t=0}^{2} \boldsymbol{e}^{\frac{\boldsymbol{\pi i 2^t}}{2^k}} \right) \ |k_{|1>}>$$

Ecuación 7-7

Ya sabemos de antemano el resultado de la fase $\emptyset = \frac{1}{8}$.

La U controlada: $\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & e^{\frac{\pi i}{4}} \end{pmatrix}$

A los 3 qubits de control inicializados a |0> le aplicamos puertas Hadamard y después se le aplica al qubit objetivo puertas controladas U de esta forma:

- Al qubit controlador menos significativo (k = 0) se le aplican (2^0) U controladas y al qubit objetivo, inicializado a |1> , con una puerta X:

$$\frac{1}{\sqrt{2}}\binom{1}{1}\oplus\binom{0}{1}=\frac{1}{\sqrt{2}}\begin{pmatrix}0\\1\\0\\1\end{pmatrix}=>\frac{1}{\sqrt{2}}\begin{pmatrix}1&0&0&0\\0&1&0&0\\0&0&1&0\\0&0&0&e^{\frac{\pi i}{4}}\end{pmatrix}\begin{pmatrix}0\\1\\0\\1\end{pmatrix}=\frac{1}{\sqrt{2}}\begin{pmatrix}0\\1\\0\\e^{\frac{\pi i}{4}}\end{pmatrix}=>\frac{1}{\sqrt{2}}\left(\left|01>+e^{\frac{\pi i}{4}}\right|11>\right)$$

$$|k_0>=\frac{1}{\sqrt{2}}\left(\left|0>+e^{\frac{\pi i}{4}}\right|1>\right)=\frac{1}{\sqrt{2}}\begin{pmatrix}1\\e^{\frac{\pi i}{4}}\end{pmatrix}$$

Ecuación 7-8

- Al qubit controlador siguiente (k=1) se le aplican (2^1) U controladas y al qubit objetivo, inicializado a |1> otra vez.

$$\frac{1}{\sqrt{2}}\binom{1}{1}\oplus\binom{0}{1}=\frac{1}{\sqrt{2}}\begin{pmatrix}0\\1\\0\\1\end{pmatrix}=>\frac{1}{\sqrt{2}}\begin{pmatrix}1&0&0&0\\0&1&0&0\\0&0&1&0\\0&0&0&e^{\frac{\pi i}{4}}\end{pmatrix}\begin{pmatrix}0\\1\\0\\1\end{pmatrix}=\frac{1}{\sqrt{2}}\begin{pmatrix}0\\1\\0\\e^{\frac{\pi i}{4}}\end{pmatrix}$$

$$\frac{1}{\sqrt{2}}\begin{pmatrix}1&0&0&0\\0&1&0&0\\0&0&1&0\\0&0&0&e^{\frac{\pi i}{4}}\end{pmatrix}\begin{pmatrix}0\\1\\0\\e^{\frac{\pi i}{4}}\end{pmatrix}=\frac{1}{\sqrt{2}}\begin{pmatrix}0\\1\\0\\e^{\frac{\pi i}{2}}\end{pmatrix}$$

$$|k_1>=\frac{1}{\sqrt{2}}\left(\left|0>+e^{\frac{\pi i}{2}}\right|1>\right)=\frac{1}{\sqrt{2}}\begin{pmatrix}1\\e^{\frac{\pi i}{2}}\end{pmatrix}$$

Ecuación 7-9

- Al qubit más significativo (k=2) se le aplican (2^2) U controladas y al qubit objetivo inicializado a |1> otra vez.

$$\frac{1}{\sqrt{2}}\binom{1}{1}\oplus\binom{0}{1}=\frac{1}{\sqrt{2}}\begin{pmatrix}0\\1\\0\\1\end{pmatrix}=>\frac{1}{\sqrt{2}}\begin{pmatrix}1&0&0&0\\0&1&0&0\\0&0&1&0\\0&0&0&e^{\frac{\pi i}{4}}\end{pmatrix}\begin{pmatrix}0\\1\\0\\1\end{pmatrix}=\frac{1}{\sqrt{2}}\begin{pmatrix}0\\1\\0\\e^{\frac{\pi i}{4}}\end{pmatrix}$$

$$\frac{1}{\sqrt{2}}\begin{pmatrix}1&0&0&0\\0&1&0&0\\0&0&1&0\\0&0&0&e^{\frac{\pi i}{4}}\end{pmatrix}\begin{pmatrix}0\\1\\0\\e^{\frac{\pi i}{4}}\end{pmatrix}=\frac{1}{\sqrt{2}}\begin{pmatrix}0\\1\\0\\e^{\frac{\pi i}{2}}\end{pmatrix}$$

$$\frac{1}{\sqrt{2}}\begin{pmatrix}1&0&0&0\\0&1&0&0\\0&0&1&0\\0&0&0&e^{\frac{\pi i}{4}}\end{pmatrix}\begin{pmatrix}0\\1\\0\\e^{\frac{\pi i}{2}}\end{pmatrix}=\frac{1}{\sqrt{2}}\begin{pmatrix}0\\1\\0\\e^{\frac{3\pi i}{4}}\end{pmatrix}$$

$$\frac{1}{\sqrt{2}}\begin{pmatrix}1&0&0&0\\0&1&0&0\\0&0&1&0\\0&0&0&e^{\frac{\pi i}{4}}\end{pmatrix}\begin{pmatrix}0\\1\\0\\e^{\frac{3\pi i}{4}}\end{pmatrix}=\frac{1}{\sqrt{2}}\begin{pmatrix}0\\1\\0\\e^{\pi i}\end{pmatrix}$$

$$|k_2>=\frac{1}{\sqrt{2}}\left(|0>+e^{\pi i}|1>\right)=\frac{1}{\sqrt{2}}\begin{pmatrix}1\\e^{\pi i}\end{pmatrix}$$

Ecuación 7-10

A los 3 qubits de control se le aplica la Transformada Cuántica Inversa de Fourier, como se puede ver en este caso el qubit que menos ha girado ha sido el menos significativo y el que más el más significativo. Por lo que empezamos a decodificar por el más significativo. Se le aplica una puerta Hadamard y se mide. El resultado que obtenemos es "1" que equivale a $k_2 = 1$. Como hemos obtenido "1" al segundo le aplicamos una puerta rotación de $(\frac{\pi}{2})$ y posteriormente una puerta Hadamard. Al medir obtenemos "1" que equivale a $k_1 = 0$. Ahora debemos aplicarle una puerta de rotación de $(\frac{3\pi}{4})$ al último qubit tras una puerta Hadamard, y obtenemos "1" que equivale a $k_0 = 0$. El resultado final obtenido es $0.k_0k_1k_2 = (0.001)_b = \frac{1}{8}$, el valor que esperábamos.

En este caso el bit/qubit más significativo es el que tiene más precisión por lo tanto el de más a la derecha.

Es importante resaltar que no siempre vamos a obtener el mismo valor, hay que recordar que cuando medimos obtenemos un valor con cierta probabilidad. Si no tenemos probabilidad 1 como pasaba en este ejemplo, que está preparado, los resultados pueden cambiar en cada iteración del circuito cuántico y hay que realizar varias medidas y estudiar que valores se observan más.

7.5 Periodo de la función $[m^k]_n$

Como ya se dijo la función de $\mathbf{[m^k]_n}$ es periódica con periodo máximo **φ(n)**. Para el algoritmo de Shor tenemos:

$$\boldsymbol{f(e) = [m^e]_n = f(ke) = [m^{ke}]_n = 1 => ke = \varphi(n)}$$

Ecuación 7-11

La finalidad del Algoritmo de Shor es encontrar el valor de **e** que hace el módulo 1. Que esa **e** sea par y que después $\left(\boldsymbol{m^{\frac{e}{2}}-1}\right)$ y $\left(\boldsymbol{m^{\frac{e}{2}}+1}\right)$ no sean múltiplos de n.

El número **m** se escoge de forma aleatoria y como ya vimos tiene que cumplir **1<m<n-1** y por supuesto que no tengan divisores en común con **n**. Si lo tienen ya hemos encontrado lo que buscábamos.

Como ya vimos la probabilidad de encontrar potencias pares (e) es muy alta.

Por la propiedad de los módulos/congruencias el cálculo se simplifica multiplicando el modulo obtenido por **m** para la siguiente iteración. Ejemplo: n = 15, m = 7, φ(n) = 8

$$7^0 => [1]_{15} = 1, \quad 1 * 7 => [7]_{15} = 7, \quad 7 * 7 => [49]_{15} = 4, \quad 7 * 4 => [28]_{15} = 13,$$
$$13 * 7 => [91]_{15} = 1, \; 1 * 7 => [7]_{15} = 7, \quad 7 * 7 => [49]_{15} = 4, \quad 7 * 4 => [28]_{15} = 13$$

Como se puede ver el periodo es: 4 => $[7^4]_{15} = [2401]_{15} = 1$

Ecuación 7-12

7.6 Parte Cuántica del Algoritmo de Shor

Como la mayoría de los algoritmos cuánticos, el algoritmo de Shor es probabilístico. Entrega una respuesta con una alta probabilidad de acierto y el error decrece a medida que aumentan las repeticiones del algoritmo. De todas formas para este algoritmo se puede comprobar si ha tenido éxito comprobando el resultado. El algoritmo de Shor puede resolver en horas lo que a una computadora clásica le llevaría años o décadas. La parte cuántica del Algoritmo es encontrar el periodo o fase (**e**) de $\mathbf{[m^k]_n}$. Una vez hecho el cálculo si la búsqueda no ha tenido éxito o el resultado no es el deseado se elegiría otra **m** y así sucesivamente hasta encontrar el resultado deseado (e) o hasta no tener más números **m** disponibles.

Hallar el periodo $\mathbf{[m^k]_n}$ con la computación clásica es igual de complejo que el hallar la factorización de **n**, por ese motivo recurrimos a la computación cuántica.

Para hallar el periodo se aprovecha la capacidad de una computadora cuántica de estar en muchos estados simultáneamente (superposición y entrelazamiento) y el algoritmo de estimación cuántica de fase, con su posterior Transformada Cuántica Inversa de Fourier.

El gran problema que presenta el algoritmo y los actuales ordenadores cuánticos reales es que hay que construir el circuito expresamente según la **n** que elijamos.

Para implementar el algoritmo necesitamos tener **t** qubits objetivo, siendo **t** el número menor que cumple $\mathbf{n-1 \leq 2^t}$, **n-1** es el valor máximo del módulo de **n**. Y necesitamos **s** qubits de control, el mínimo número de qubits de control que necesitamos es el que cumpla $\boldsymbol{\varphi(n) \leq 2^s}$, esto es debido a que el máximo periodo de $\mathbf{[m^k]_n}$ es $\boldsymbol{\varphi(n)}$, como ya vimos los periodos **(e)** que podemos obtener tienen que cumplir $\boldsymbol{\varphi(n) = de}$, siendo **d** un número natural no nulo. Y $\boldsymbol{\varphi(n)}$ como ya vimos es múltiplo de 4. Como una de las incógnitas que tenemos es $\boldsymbol{\varphi(n)}$, no sabemos cuánto es **s** por lo que se suele tomar **t = s**. Tendremos más posibilidad de éxito si aumentamos **s**, es decir el número de qubits de control.

Se le aplican U transformaciones controladas a los qubits objetivo y a un qubit de control cada vez. En total, tantas veces como qubits de control tengamos. Se supone que con la afectación de las puertas U controladas a los **s** qubits podemos estimar el periodo buscado. Las puertas U realizan la exponenciación modular a los qubits objetivo afectando a un qubit de control cada vez.

En resumen, de los **φ(n)** posibles valores del periodo de la función $\mathbf{[m^k]_n}$, que mayorandolo por arriba serían $\mathbf{2^t}$, posibles restos/módulos de **n** sólo se realizan t cálculos de la función $\mathbf{[m^k]_n}$, igual al número de qubits de control, con esos **t** cálculos se estima cual es el periodo o fase de la función. Como

se puede ver a medida que t crece la diferencia $(\boldsymbol{2^t - t})$ se hace mucho más grande y por tanto la mejora del algoritmo cuántico es mayor. Ya que a mayor $\boldsymbol{n}$ mayor $\boldsymbol{\varphi(n)}$ y por tanto mayor $\boldsymbol{e}$ (periodo) posible para la función $\boldsymbol{[m^k]_n}$ y las operaciones que realizamos con el algoritmo de Shor son cada vez menores en proporción a $\boldsymbol{\varphi(n)}$ que es el mayor $\boldsymbol{e}$ (periodo) posible. Los $\boldsymbol{2^t}$ cálculos máximos, valores de $\boldsymbol{[m^k]_n}$, serían el cálculo por fuerza bruta en la computación clásica para calcular el periodo y **t** cálculos para la computación cuántica. En vez de tener **t** qubits de control se pueden coger más incrementando la probabilidad de que el proceso tenga éxito.

Se expone un ejemplo ilustrativo: n= 15, p=3, q=5 y $\varphi(n) = 2*4 = 8$. Tenemos $t = 4 \Rightarrow 14 \leq 2^4 = 16$ suficiente para representar 14 que va a ser el módulo/resto mayor de 15. Sólo vamos a coger para el ejemplo **3** qubits de control ya que con 3 bits podemos representar 8 valores, el periodo máximo de la función $[m^k]_{15}$ y **4** (t) qubits de objetivo. Por lo que la diferencia de iteraciones necesarias en la computación cuántica respecto a la computación clásica en el peor de los casos será $\varphi(n) - s \Rightarrow 8 - 3 = 5$ que en verdad es muy poco, pero hay que tener en cuenta que es un ejemplo. Es decir, sólo vamos a realizar 3 iteraciones de la función $[m^k]_{15}$ en el ejemplo se supone que realizaríamos las 3 exponenciaciones binarias $[m^1]_{15}$, $[m^2]_{15}$, $[m^3]_{15}$ pero podrían ser otras. Las que quedan que no se realizan son: $[m^4]_{15}$, $[m^5]_{15}$, $[m^6]_{15}$, $[m^7]_{15}$, $[m^8]_{15}$

Los qubits de control (C_0, C_1, C_2) se inicializan a |0> y después se le aplican puertas Hadamard para superponerlos.

En el ejemplo vamos a coger m = 7. Con puertas Toffoli *[ver página 49]* construimos una puerta U que haga 7*(entrada) y le calcule el módulo de 15 sacándolo por la salida. En definitiva que haga las exponenciaciones modular binaría que habíamos dicho ($[m^1]_{15}$, $[m^2]_{15}$, $[m^3]_{15}$).

(O_3, O_2, O_1, O_0)

1. $[U|0001\rangle\ (*7)]_{15} = |0111\rangle$ (7)
2. $[U|0111\rangle\ (*7)]_{15} = |0100\rangle$ (4)
3. $[U|0100\rangle\ (*7)]_{15} = |1101\rangle$ (13)

Esta puerta U la convertimos en una U controlada para controlar (C_0, C_1, C_2).

1. $U_c|\ C_0 0001\rangle = |C_{0c}0111\rangle$
2. $U_c|\ C_1 0111\rangle = |C_{1c}0100\rangle$
3. $U_c|\ C_2 0100\rangle = |C_{2c}1101\rangle$

Se denota como (C_{0c}, C_{1c}, C_{2c}) a los qubits de control después de aplicarles las puertas U controladas (U_c). Cada vez que apliquemos una puerta U controlada medimos los qubits objetivos, que estarán afectados al estar entrelazados con un qubit de control. Se reinician los qubits objetivo con el valor que toque y se le vuelve aplicar la U controlada afectando a otro qubit de control y se vuelven a medir los qubits objetivos. Esto se realiza tantas veces como qubits de control tengamos.

Al terminar se realiza la Transformada Cuántica Inversa de Fourier a los qubits de control. Con esta medida se estima la fase $\boldsymbol{\phi}$ (tanto por 1 de vuelta) que hemos realizado con la función $[m^k]_n$.

Obtenemos un resultado que será $\phi = \frac{k}{e}$, sabemos k que en el ejemplo es k = 1 + (2-1) + (3-2) = 1 + 1 + 1 = 3 saltos menos que un periodo (vuelta). Ahora para calcular **e** sólo tenemos que utilizar una regla de tres:

$$k \ -----\ \phi$$
$$e \ -----\ 1$$

$$e = \frac{k}{\phi} => \frac{3}{\phi}$$

Los valores posibles de $\boldsymbol{\phi}$ que podemos obtener en el ejemplo son:

$$\frac{0}{8}(0.000), \frac{1}{8}\ (0.001), \frac{2}{8}(0.010), \frac{3}{8}(0.011), \frac{4}{8}(0.100), \frac{5}{8}(0.101), \frac{6}{8}(0.110), \frac{7}{8}(0.111)$$

Como ya se dijo estos valores posibles no tienen por qué ser exactos a $\boldsymbol{\phi}$**.** Si ejecutamos el circuito un número **x** de veces obtendremos con más probabilidad uno o varios resultados. Por lo que es de suponer que los valores que más obtendremos para este ejemplo sean $\frac{6}{8}\ (0.110) = \frac{3}{4}$ que equivale a **e**= $\frac{3}{\frac{3}{4}} = 4$.

Se obtiene un valor exacto de **e**, ya que **e** es potencia de 2. Estudiemos ahora un ejemplo dónde **e** no lo es: n= 13 y m = 3, para este ejemplo **e** = 3 ya que $[3^3]_{13} = 1$, se cumple que φ(n) = 13 -1 =12 y 12 es múltiplo de **e** = 3. Si hiciéramos un desplazamiento de fase que fuese menor de 3 obtendríamos que la fase real podría ser: $\frac{1}{3}\ para\ k = 1$ ó $\frac{2}{3}\ para\ k = 2$, estos valores no se pueden representar exactamente con potencias negativas de 2. Si tenemos el circuito de antes, es decir los mismos qubits (3):

$$\frac{2}{8} < \frac{1}{3} = 0.33 \ldots < \frac{3}{8}$$

$$\frac{5}{8} < \frac{2}{3} = 0.66 \ldots < \frac{6}{8}$$

Si funciona bien la estimación de fase obtendremos el resultado que más se acerque por arriba o por abajo es decir para $\frac{1}{3}$ => $\frac{3}{8} = 0.375$ y para $\frac{2}{3}$ => $\frac{5}{8} = 0.625$ ahora con estos valores tenemos que determinar qué **e** = 3. Tenemos que $\frac{8}{3} * 1 = 2.66..$ y $\frac{8}{5} * 2 = 3.2$, como se puede ver con redondear al entero más próximo se obtiene el resultado. Si queremos estar más seguros se puede probar con los 2 enteros, por arriba y por abajo.

Por lo tanto tenemos que después de ejecutar el circuito un número de veces obtenemos resultados estimados de $\boldsymbol{\phi}$ cada uno con una probabilidad. Una limitación por ahora es que las **k** resultantes que cojamos deben ser menor que **e** pero como hemos dicho al ser las **n** tan grandes esto es lo más probable. Se cogen los valores con más probabilidad y se cogen los enteros por arriba y por debajo de las fracciones y se calcula si cumplen que $\boldsymbol{[m^e]_n = 1}$. Si ninguno lo cumple es que hemos fallado la estimación y probamos con otro $\boldsymbol{m}$. Y si se cumple se comprueba que es $\boldsymbol{e}$ par, si fuese impar probaríamos con otra $\boldsymbol{m}$ y si es $\boldsymbol{e}$ par se comprueba que $\boldsymbol{m^{\frac{e}{2}} - 1}$ y $\boldsymbol{m^{\frac{e}{2}} + 1}$ no son múltiplos de $\boldsymbol{n}$ si esto no ocurre se coge otro $\boldsymbol{m}$. Si no el $\boldsymbol{e}$ encontrado si cumple lo que estábamos buscando y con el podemos sacar $\boldsymbol{p}$ y $\boldsymbol{q}$.

Para realizar un ejemplo completo cogemos n=3, φ(n) = 2. Vamos a coger para el ejemplo 1 qubit de control (C_0) y 2 (O_0, O_1) qubits de objetivo. El único ***m*** posible es m= 2 que es coprimo con n=3 y el periodo a calcular será e = 2. (2^1, 2^2). Aplicaríamos dos U controladas. Inicializamos C_0 a |0> y le aplicamos una puerta Hadamard y a (O_1 ,O_0) lo inicializamos a |0> y le aplicamos una puerta X a O_0 = |1>.

La U que tendríamos que construir sería:

$$\begin{pmatrix}\alpha_0\\ \beta_0\end{pmatrix}\oplus\begin{pmatrix}\alpha_1\\ \beta_1\end{pmatrix}=\begin{pmatrix}\alpha_0\alpha_1\\ \alpha_0\beta_1\\ \beta_0\alpha_1\\ \beta_0\beta_1\end{pmatrix}=>\ \begin{pmatrix}1\\ 0\end{pmatrix}\oplus\begin{pmatrix}1\\ 0\end{pmatrix}=\ \begin{pmatrix}1\\ 0\\ 0\\ 0\end{pmatrix}=>|00>$$

$$\begin{pmatrix}1\\ 0\end{pmatrix}\oplus\begin{pmatrix}0\\ 1\end{pmatrix}=\begin{pmatrix}0\\ 1\\ 0\\ 0\end{pmatrix}=>|01>=>\begin{pmatrix}0\\ 1\end{pmatrix}\oplus\begin{pmatrix}1\\ 0\end{pmatrix}=\begin{pmatrix}0\\ 0\\ 1\\ 0\end{pmatrix}=>|10>=>$$

$$\begin{pmatrix}0\\ 1\end{pmatrix}\oplus\begin{pmatrix}0\\ 1\end{pmatrix}=\begin{pmatrix}0\\ 0\\ 0\\ 1\end{pmatrix}=>|11>$$

$$\begin{pmatrix}a_{00} & a_{01} & a_{02} & a_{03}\\ a_{10} & a_{11} & a_{12} & a_{13}\\ a_{20} & a_{21} & a_{22} & a_{23}\\ a_{30} & a_{31} & a_{32} & a_{33}\end{pmatrix}\begin{pmatrix}\alpha_0\alpha_1\\ \alpha_0\beta_1\\ \beta_0\alpha_1\\ \beta_0\beta_1\end{pmatrix}$$

Ecuación 7-13

La puerta U que estamos buscando tiene que cambiar el estado U|01> = |10> y el estado U|10> = |01>; mientras que para los estados |00> y |11> no importa lo que haga. Vamos a suponer que deja los 2 estados igual.

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

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

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

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

Nos queda entonces U:

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

Ahora comprobamos si U es una matriz unitaria. Para ello su inversa tenía que ser igual a su conjugada transpuesta ($U * U_c^t = I \; y \; U_c^t * U = I$) . Como todos los números son reales no tenemos que fijarnos en su conjugada. La transpuesta era cambiar filas por columnas. Se ve que su traspuesta coincide con ella misma por lo que no sólo hay que hacer una comprobación ($U * U_c^t = U_c^t * U$).

$$U^t = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

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

Como se puede ver U cumple con todo lo requerido, ahora construimos la U_c controlada de un qubit de control. Que realiza U sólo para la parte imaginaria β del qubit de control. La parte real α no realiza ningún cambio.

$$\binom{\alpha}{\beta} \oplus \binom{\alpha_0}{\beta_0} \oplus \binom{\alpha_1}{\beta_1} = \begin{pmatrix} \alpha\alpha_0\alpha_1 \\ \alpha\alpha_0\beta_1 \\ \alpha\beta_0\alpha_1 \\ \alpha\beta_0\beta_1 \\ \beta\alpha_0\alpha_1 \\ \beta\alpha_0\beta_1 \\ \beta\beta_0\alpha_1 \\ \beta\beta_0\beta_1 \end{pmatrix}$$

$$U_C = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \alpha\alpha_0\alpha_1 \\ \alpha\alpha_0\beta_1 \\ \alpha\beta_0\alpha_1 \\ \alpha\beta_0\beta_1 \\ \beta\alpha_0\alpha_1 \\ \beta\alpha_0\beta_1 \\ \beta\beta_0\alpha_1 \\ \beta\beta_0\beta_1 \end{pmatrix} = \begin{pmatrix} \alpha\alpha_0\alpha_1 \\ \alpha\alpha_0\beta_1 \\ \alpha\beta_0\alpha_1 \\ \alpha\beta_0\beta_1 \\ \beta\alpha_0\alpha_1 \\ \boldsymbol{\beta\beta_0\alpha_1} \\ \boldsymbol{\beta\alpha_0\beta_1} \\ \beta\beta_0\beta_1 \end{pmatrix}$$

Como se puede ver U_C cumple con lo que se había dicho.

Para estar seguros al 100% comprobamos si es Unitaria, la matriz transpuesta conjugada coincide con ella misma por lo que comprobamos si:

$$U * U_c^t = U_c^t * U = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} =$$

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

Una vez comprobado pasamos a aplicarla:

1. $U_c|C_0 01> = \frac{1}{\sqrt{2}} U_c(|001> + |101>) = \frac{1}{\sqrt{2}} Uc \begin{pmatrix} 1 \\ 1 \end{pmatrix} \oplus \begin{pmatrix} 1 \\ 0 \end{pmatrix} \oplus \begin{pmatrix} 0 \\ 1 \end{pmatrix} =$

$$= \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \end{pmatrix} = \frac{1}{\sqrt{2}} \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \end{pmatrix} = \frac{1}{\sqrt{2}} (|001> + |110>)$$

Al aplicar la primera U_c se entrelaza C_0 con O_1O_0.

Para medir hay que realizar la Transformada Inversa Cuántica de Fourier y medimos.

$$\frac{0}{2}(0.0), \frac{1}{2}\ (0.1)$$

De los 2 valores posibles que tenemos se obtendrá $\frac{1}{2}$, como k = 1. Tenemos que e= 2. Que es lo que esperábamos.

El cuello de botella del tiempo de ejecución del algoritmo de Shor es la exponenciación modular cuántica, que es mucho más lenta que la transformada inversa cuántica de Fourier y el pre y post procesamiento clásico. Hay varios enfoques para construir y optimizar circuitos para potenciación modular. El enfoque más simple y (actualmente) más práctico es imitar circuitos aritméticos convencionales con puertas reversibles (puertas de Toffoli).

7.7 Mejoras al Algoritmo de Shor

Para introducir la primera mejora vamos a introducir el concepto de módulo negativo de **n** que equivale al número que falta para completar una cantidad exacta de n. Se deduce que lo que falta más lo que sobra siempre suma **n**. Por lo que el módulo se puede representar con un número positivo o con un número negativo. Entonces tenemos:

$$[m]_n = k = -(n-k) => 1 = -(n-1) => (n-1) = -1$$

Ecuación 7-14

Si tomamos n impar tenemos que $\boldsymbol{r = \frac{n-1}{2}}$, podemos representar los módulos **k** como:

$$-r \leq k \geq +r$$

Ecuación 7-15

Ponemos un ejemplo con n=15.

$$[0]_{15} = 0 => [15]_{15} = -0$$

$$[1]_{15} = 1 => [14]_{15} = -1 => [-1]_{15}$$

$$[2]_{15} = 2 \Rightarrow [13]_{15} = -2 \Rightarrow [-2]_{15}$$

$$[3]_{15} = 3 \Rightarrow [12]_{15} = -3 \Rightarrow [-3]_{15}$$

$$[4]_{15} = 4 \Rightarrow [11]_{15} = -4 \Rightarrow [-4]_{15}$$

$$[5]_{15} = 5 \Rightarrow [10]_{15} = -5 \Rightarrow [-5]_{15}$$

$$[6]_{15} = 6 \Rightarrow [9]_{15} = -6 \Rightarrow [-6]_{15}$$

$$[7]_{15} = 7 \Rightarrow [8]_{15} = -7 \Rightarrow [-7]_{15}$$

Si **k** es mayor que **r** lo representamos por su homologo negativo. Ahora tomamos la función $\mathbf{[m^k]_n}$, si **m** es mayor que **r** vamos a representar los módulos con negativos. En el ejemplo anterior si cogemos m = 9, tenemos que:

$$[13]_{15} = [-2]_{15} = -2$$

$$[13^2]_{15} = [[13]_{15}[13]_{15}]_{15} = [(-2) * (-2)]_{15} = [4]_{15} = 4$$

$$[13^3]_{15} = [[13^2]_{15}[13]_{15}]_{15} = [4 * (-2)]_{15} = [-8]_{15} = -8$$

$$[13^4]_{15} = [[13^2]_{15}[13^2]_{15}]_{15} = [4 * 4]_{15} = [16]_{15} = 1$$

Como se puede ver las potencias con exponentes pares tienen los módulos positivos y los que tienen exponentes impares pueden tener módulos negativos.

$$[2]_{15} = 2$$

$$[2^2]_{15} = [4]_{15} = 4$$

$$[2^3]_{15} = [8]_{15} = 8$$

$$[2^4]_{15} = [[2^2]_{15}[2^2]_{15}]_{15} = [4 * 4]_{15} = [16]_{15} = 1$$

Como se puede ver la función $\mathbf{[m^k]_n}$ es simétrica en valor absoluto respecto a $\boldsymbol{r = \frac{n-1}{2}}$ y para los exponentes pares son todos positivos. Si una **m** menor que **r** tiene un periodo impar había que desecharlo porque no nos valía para el algoritmo de Shor, sólo nos quedábamos con las **m** que tenían periodos pares. Para el caso de los **m** que tengan su periodo impar, sus homólogos tendrán -1 de resto para ese exponente y el periodo será justo el doble [(-1) (-1) = 1], haciendo más notable que los periodos pares se dan mucho más. Pero este valor también hay que desecharlo ya que si el módulo es -1 quiere decir que si le sumamos uno será múltiplo de **n** y por lo tanto ya no podremos seguir con la factorización.

Con esta mejora se pasa a coger un número aleatorio m que cumpla:

$$1 > m \geq \frac{n-1}{2}$$

Ecuación 7-16

Tenemos que hemos pasado de tener n-2 posibles m para elegir a $\frac{n-1}{2}$.

Una segunda mejora consiste en calcular **φ(n)**, si conseguimos calcular este valor no necesitamos calcular **p** y **q**. Con este valor ya podríamos calcular la clave privada sabiendo la clave pública. De **φ(n)** sabemos que es un entero positivo, es múltiplo de 4 y menor que **n**. Y que tiene que ser múltiplo de todos los periodos de la función $\mathbf{[m^k]_n}$. Tenemos acotado por arriba **φ(n)**, mayor múltiplo de 4 menor de $\boldsymbol{n}$ y por abajo como mínimo tiene que ser 4.

Creamos una variable que llamaremos por ejemplo **x**, la inicializamos con 4. Para cada periodo que encontremos de la función $\mathbf{[m^k]_n}$ independientemente de que sea par o impar, comprobamos si tiene algún factor no incluido en **x**, si lo tiene multiplicamos esos factores a **x** y guardamos el resultado en **x**. Si el periodo calculado no divide exactamente a **x** o es mayor que **x**, tiene que tener un factor o factores diferentes y calcularíamos el máximo común divisor por Euclides. Por ejemplo, si tenemos x=4 y calculamos un periodo e=30. Se calcula que 15 no está en x por lo que **x** pasa ahora a valer x=60. Si **x** llegara a valer su cota superior entonces tendríamos ya calculado φ(n).

Antes de ejecutar la parte cuántica del algoritmo de Shor se calcula $\mathbf{[m^x]_n}$; si diera 1 quiere decir que el periodo de $\mathbf{[m^e]_n}$ es un múltiplo de x. Se calcula $[\boldsymbol{m^{\frac{x}{2}}}]_{\boldsymbol{n}}$ y se vuelve a calcular si vuelve a dar 1 se vuelve a dividir el exponente por 2 hasta que no de 1 o no sea divisible por 2. Si da 1 y no es divisible por 2 es que el periodo es impar y por tanto cogemos otra **m**. Si ya no da 1 quiere decir que el periodo era par y era el doble del exponente que tenemos ahora y por tanto seguimos con la parte clásica del Algoritmo de Shor para comprobar si este valor nos sirve para obtener **p** y **q**.

A medida que se va cogiendo más valores de **m**, mayor será **x** por lo que tendremos mucha más probabilidad de que los periodos estén incluidos en **x**. Para los valores de **x** también son útiles los periodos impares ya que si no están ya incluido en **x** se introducen y aumentan a **x**.

Se pone un ejemplo aclaratorio:

Tomamos n = 35, el mayor múltiplo de 4 menor de 35 es 32. Tendríamos entonces que 4>φ(35)<32. Como conocemos p y q se sabe que φ(35) = 24. Los periodos (e) posibles que podemos obtener en la función $[m^x]_{35}$ son: 2, **3**, 4, 6, 8, 12 y 24. Y las x que se pueden ir acumulando son: 4, 8, 12 y 24. Como se puede ver el periodo impar 3 también es útil para el acumulado de x.

7.8 REFLEXIONES

En definitiva la parte más complicada y más delicada del Algoritmo de Shor es encontrar la **e** mínima que satisfaga la fórmula:

$$[m^e]_n = 1$$

Ecuación 7-17

Antes de evaluar las **e** que cumplen esta ecuación recordemos una propiedad de los módulos:

$$[m^{a+b}]_n = [[m^a]_n[m^b]_n]_n$$

Ecuación 7-18

Si **e** es impar como ya se ha dicho esa **m** se descarta y se elige otra. Tenemos que si **e** es impar cumple que **e = 2p+1**, tomamos $\boldsymbol{t_x}$ como un resto/módulo de **n**:

$$[[m^{2p}]_n[m^1]_n]_n = 1$$

$$[t_a t_b]_n = 1$$

Ecuación 7-19

Se tiene que cumplir que: $\boldsymbol{t_a \neq t_b}$, ya que **1 <m < n-1** y por lo tanto $[\boldsymbol{m^1}]_{\boldsymbol{n}} \neq \mathbf{1}$ **y** $[\boldsymbol{m^1}]_{\boldsymbol{n}} \neq -\mathbf{1}$ entonces se tiene que cumplir que $[\boldsymbol{m^{2p}}]_{\boldsymbol{n}} \neq [\boldsymbol{m^1}]_{\boldsymbol{n}}$ ya que si $[\boldsymbol{m^{2p}}]_{\boldsymbol{n}} = [\boldsymbol{m^1}]_{\boldsymbol{n}}$ entonces se podría representar la fórmula de arriba $[[\boldsymbol{m^{2p}}]_{\boldsymbol{n}}[\boldsymbol{m^{2p}}]_{\boldsymbol{n}}]_{\boldsymbol{n}} = \mathbf{1}$, que va en contra de que **e** era impar.

Si trabajamos con módulos negativos, representamos el inverso de $\boldsymbol{t_x}$ como $\boldsymbol{t_{-x}}$ que como habíamos dicho cumplían que $\boldsymbol{t_x + t_{-x} = n}$. Entonces:

$$[t_a t_b]_n = 1 \Rightarrow [t_a t_{-b}]_n = -1 \Rightarrow [t_{-a} t_{-b}]_n = 1$$

Ecuación 7-20

Si **e** es par tenemos entonces que **e = p + p**:

$$[[m^p]_n[m^p]_n]_n = 1$$

$$[t_a t_a]_n = [t_a^2]_n = 1$$

Ecuación 7-21

Y si tenemos en cuenta los módulos negativos también se cumple:

$$[(t_{-a})^2]_n = 1$$

Ecuación 7-22

Se simplifica ya que puede haber dos tipos de módulos que su cuadrado obtenga el resto 1. Los que nos sirven para obtener la factorización y los que no, en estos casos no nos sirven:

$$[t_a^2]_n = 1 => (-1)(-1) = 1 => [(n-1)(n-1)]_n$$

$$[t_a^2]_n = 1 => (1)(1) = 1 => [(1)(1)]_n$$

Ecuación 7-23

Con esto se deduce que tiene que haber 2 **m** por lo menos que cumplan que tengan su periodo **e**=2 (el 1 y n-1 ya estaban excluidos).

Se deduce que sólo hay 3 posibilidades de multiplicación con un exponente par: La 1*1 que indica que tenemos que probar con la mitad del exponente (e). La de (-1)(-1) que implica que hay que descartar esa m y probar con otra. Y la de x * x, siendo x distinto de 1 y -1. Esto implica (x+1)(x-1) es múltiplo de n, que es lo que estábamos buscando.

Tenemos entonces por lo menos 2 $\mathbf{t_x}$ que cumplen que son módulos 1 de **n**, que ni son múltiplos de **p** ni de **q** y que uno de ellos $\sqrt{\boldsymbol{n}} < \boldsymbol{t_{x1}} \leq \frac{\boldsymbol{n-1}}{\boldsymbol{2}}$ que cumple $[\boldsymbol{t_x^2}]_{\boldsymbol{n}} = \mathbf{1}$ y el otro tiene que cumplir $\mathbf{n} = \boldsymbol{t_{x1}} + \boldsymbol{t_{x2}}$ por lo que $\boldsymbol{t_{x2}} = \boldsymbol{t_{-x1}}$, se deduce que $-\boldsymbol{t_{x1}} < -\sqrt{\boldsymbol{n}}$.

Tenemos entonces que se cumple que: $\mathbf{2}\boldsymbol{t_{x1}}$ **+ y = n**, donde **y** tiene que ser impar y el valor menor que puede tener **y** es y=1.

Se pone como ejemplo n=33. Tenemos que $[10 * 10]_{33} = 1$ y $[23 * 23]_{33} = 1$ y que es igual a $[(-10) * (-10)]_{33} = 1$, tenemos entonces que 2*10 +y = 33 que nos da y = 13. Para n = 15 tenemos que $[4 * 4]_{15} = 1$ y $[11 * 11]_{15} = 1$ y que es igual a $[(-4) * (-4)]_{15} = 1$, tenemos entonces que 2*4+y = 15 que nos da y = 7.

Se cumple también que el módulo **n** de **n+1**, cumple:

$$[(n+1)]_n = 1$$

Ecuación 7-24

Como n es impar (n+1) es par por lo que (n+1) = 2 ($\frac{n+1}{2}$) por lo tanto:

$$\left[2\left(\frac{n+1}{2}\right)\right]_n = 1$$

$$\left[(\mathrm{n}-2)\left(\frac{n+1}{2}\right)\right]_n = -1$$

Ecuación 7-25

Se deduce que descomponiendo en factores se llegaría a encontrar todos los módulos que tengan el periodo **e = 2**. Pongamos un ejemplo ilustrativo: **n= 33**, tenemos entonces que n-1 = 32 = 2 *16 = 4*8 y n+1 = 34 = 2*17:

$$[2 * 16]_{33} = [4 * 8]_{33} = -1$$

$$[31 * 16]_{33} = 1 \Rightarrow [2 * 17]_{33} = 1 \Rightarrow [29 * 8]_{33} = 1 \Rightarrow [\mathbf{4 * 25}]_{\mathbf{33}} = \mathbf{1}$$

$$[31 * 17]_{33} = -1 \Rightarrow [29 * 25]_{33} = -1$$

$$[4 * 25]_{33} = 1 \Rightarrow [20 * 5]_{33} = 1 \Rightarrow [\mathbf{10 * 10}]_{\mathbf{33}} = \mathbf{1}$$

$$[20 * 28]_{33} = -1 \Rightarrow [13 * 5]_{33} = -1$$

$$[13 * 28]_{33} = 1$$

$$[\mathbf{10 * 10}]_{\mathbf{33}} = [\mathbf{23 * 23}]_{\mathbf{33}} = \mathbf{1}$$

Por lo que tenemos que:

$$[\mathbf{10^2}]_{\mathbf{33}} = [\mathbf{23^2}]_{\mathbf{33}} = \mathbf{1}$$

Tenemos entonces que:

$$9 * 11 = 33t \; y \; 22 * 24 = 33p$$

Aplicando el Algoritmo de Euclides sacamos que 33 = 3 * 11

Un segundo ejemplo con n = 91 = 13 * 7, tenemos entonces que n -1 = 90 = 2 *45 = 18 * 5 = 10 * 9 = 30 * 3 = 6 *15 y n+1 = 92 = 2 *46 = 4*23

$$[2 * 45]_{91} = [18 * 5]_{91} = [10 * 9]_{91} = [3 * 30]_{91} = [6 * 15]_{91} = -1$$

$$[89 * 45]_{91} = -[\mathbf{2 * 45}]_{\mathbf{91}} = -[18 * 5]_{91} = [18 * 86]_{91} = [\mathbf{9 * 81}]_{\mathbf{91}} = [82 * 10]_{91} = 1$$

$$[88 * 30]_{91} = [3 * 61]_{91} = [85 * 15]_{91} = [6 * 76]_{91} = 1$$

$$[2 * 46]_{91} \Rightarrow [4 * 23]_{91} \Rightarrow -[16 * 17]_{91} \Rightarrow [32 * 37]_{91} \Rightarrow -[64 * 27]_{91} \Rightarrow [\mathbf{64 * 64}]_{\mathbf{91}}$$

$$[\mathbf{27 * 27}]_{\mathbf{91}} = [\mathbf{64 * 64}]_{\mathbf{91}} = \mathbf{1}$$

Por lo que tenemos que:

$$[\mathbf{27^2}]_{\mathbf{91}} = [\mathbf{64^2}]_{\mathbf{91}} = \mathbf{1}$$

Tenemos entonces que:

$$26 * 28 = 91t \; y \;\; 63 * 65 = 91p$$

Aplicando el Algoritmo de Euclides sacamos que 91 = 7 * 13

Se deduce que vamos a tener siempre $|[\mathbf{4 * x}]_{\boldsymbol{n}}| = \mathbf{1}$ por lo que los números buscados serán par e impar, respectivamente.

Tenemos entonces que existe la posibilidad de que se coja al azar una **m** que su periodo **(e)** sea pequeño incluso tan pequeño como 2. Por este motivo al aplicar la parte cuántica tenemos la duda de que el **k** que suponíamos menor que una vuelta **(e)** se pueda dar el caso que implique más de una vuelta, $\boldsymbol{\phi = \frac{k}{e}}$, al poder ser **k** mayor que **e**, pasamos a reescribir la fórmula como $\boldsymbol{\phi = \frac{[k]_e}{e}}$, ahora se contempla de que al aplicar la transformada inversa cuántica de Fourier existe la posibilidad de que $\boldsymbol{k \neq [k]_e}$ se ponen ejemplos de cómo intentar solucionarlo.

Se coge el circuito que teníamos para n = 15, ahora nuestra puerta U hace estos cálculos: $[m^2]_{15}$, $[m^4]_{15}$, $[m^8]_{15}$, sabemos que e = 4, por lo que k = 8 es mayor que e y tenemos que $[k]_e = [8]_4 = 0$. Para esta implementación unos de los valores que obtendríamos con más probabilidad sería el 0, que indica que **k** es múltiplo de **e**. Para esta implementación no tendríamos problema a la hora de encontrar **e**.

Ahora se supone que la puerta U hace: $[m^1]_{15}$, $[m^3]_{15}$, $[m^5]_{15}$, k = 5 es mayor que **e** y tenemos que $[k]_e = [5]_4 = 1$. Uno de los resultados que tendremos que obtener con mayor probabilidad es $\frac{2}{8}(0.010) = \frac{1}{4}$, ahora tomamos como posible **e**, el divisor (4). Los valores que se podrían obtener sin conocer la **k**, serían: $0, \frac{1}{4}, \frac{2}{4}, \frac{3}{4}$ en lo que siempre se resume en coger el divisor como posible **e**. En este caso es sencillo al ser **e** una potencia de 2.

Cojamos ahora un ejemplo dónde no lo sea, si el periodo es 3 tenemos estos posibles valores de fase: $0, \frac{1}{3}, \frac{2}{3}$. Como ya se dijo antes para $\frac{1}{3} \approx \frac{3}{8}$ y para $\frac{2}{3} \approx \frac{5}{8}$ serán los valores que obtendríamos con la transformada inversa cuántica de Fourier, como no tenemos seguro si **k** es mayor que **e** o no. Se puede empezar suponiendo que no lo es y probar y si no obtenemos lo deseado seguir de esta otra forma, descomponiendo las fracciones obtenidas con mayor probabilidad en fracciones continuas:

$$\frac{3}{8} = \frac{1}{\frac{8}{3}} = \frac{1}{2+\frac{2}{3}} = \frac{1}{2+\frac{1}{\frac{3}{2}}} = \frac{1}{2+\frac{1}{1+\frac{1}{2}}}$$

Se prueba con todos los divisores distintos de 1: 3,2. Se puede observar que uno de ellos es el 3, que es el que buscamos.

$$\frac{5}{8} = \frac{1}{\frac{8}{5}} = \frac{1}{1+\frac{3}{5}} = \frac{1}{1+\frac{1}{\frac{5}{3}}} = \frac{1}{1+\frac{1}{1+\frac{2}{3}}} = \frac{1}{1+\frac{1}{1+\frac{1}{\frac{3}{2}}}} = \frac{1}{1+\frac{1}{1+\frac{1}{1+\frac{1}{2}}}}$$

Se prueba con todos los divisores distintos de 1: 5,3, 2. Se puede observar que uno de ellos es el 3, que es el que buscamos. El algoritmo de descomposición de fracciones en fracciones continuas se realiza fácilmente con computación clásica.

Para finalizar podemos resumir la parte cuántica del Algoritmo de Shor con esta imagen: tenemos 2 ruedas dentadas, de una rueda dentada se conoce el número de dientes y se desconoce el número de dientes de la otra. Gracias al paralelismo cuántico se puede calcular con una gran probabilidad el número de dientes de la segunda aplicando un giro de una vuelta a la primera.

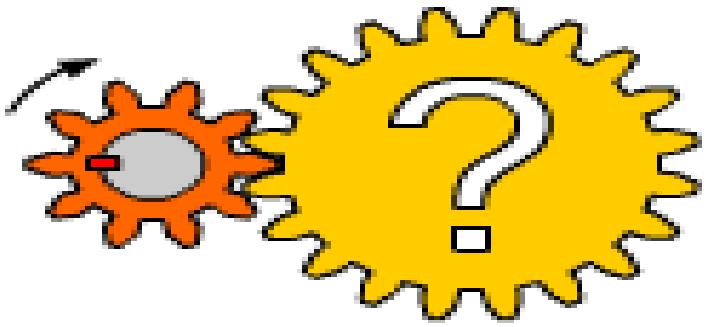


Figura 7-1, Ruedas Dentadas

8 EJEMPLO DE CIRCUITO CUÁNTICO

En 1997 se implementó el algoritmo de Shor con 12 qubits para factorizar el número 15 en un computador cuántico real, constituyendo un logro histórico. Desde entonces se ha repetido este hito muchas veces. La mayoría de las veces se usa información a priori (que ya sabemos los factores del número) para cablear el algoritmo de forma eficiente. En 2012 se consiguió factorizar el 21.

Para factorizar el número 15 = 5 × 3 el algoritmo de Shor (1994) recomienda 12 qubits, pero bastan 5 qubits con el algoritmo de Kitaev (1995). Se publica en Science *[Referencia 1]* una implementación del algoritmo de Kitaev usando 5 átomos atrapados. Los autores afirman que es un algoritmo escalable. Por supuesto, la implementación del algoritmo de Kitaev usando átomos atrapados es un gran avance en el campo de los ordenadores cuánticos. Aunque sólo se hayan atrapado 5 átomos, en las próximas décadas se podrán atrapar muchos más (quizás cientos de átomos). Además, hay técnicas de teletransporte cuántico que permite entrelazar los estados de los átomos de diferentes trampas, con lo que es previsible que en las próximas décadas se puedan usar cientos de qubits. Sin embargo, el nuevo artículo no usa ninguna técnica de corrección de errores, lo que penaliza el tamaño del número a factorizar conforme crece el número de qubits. Aunque el título del artículo afirme que la nueva implementación es escalable, hoy por hoy es tecnológicamente imposible lograrlo.

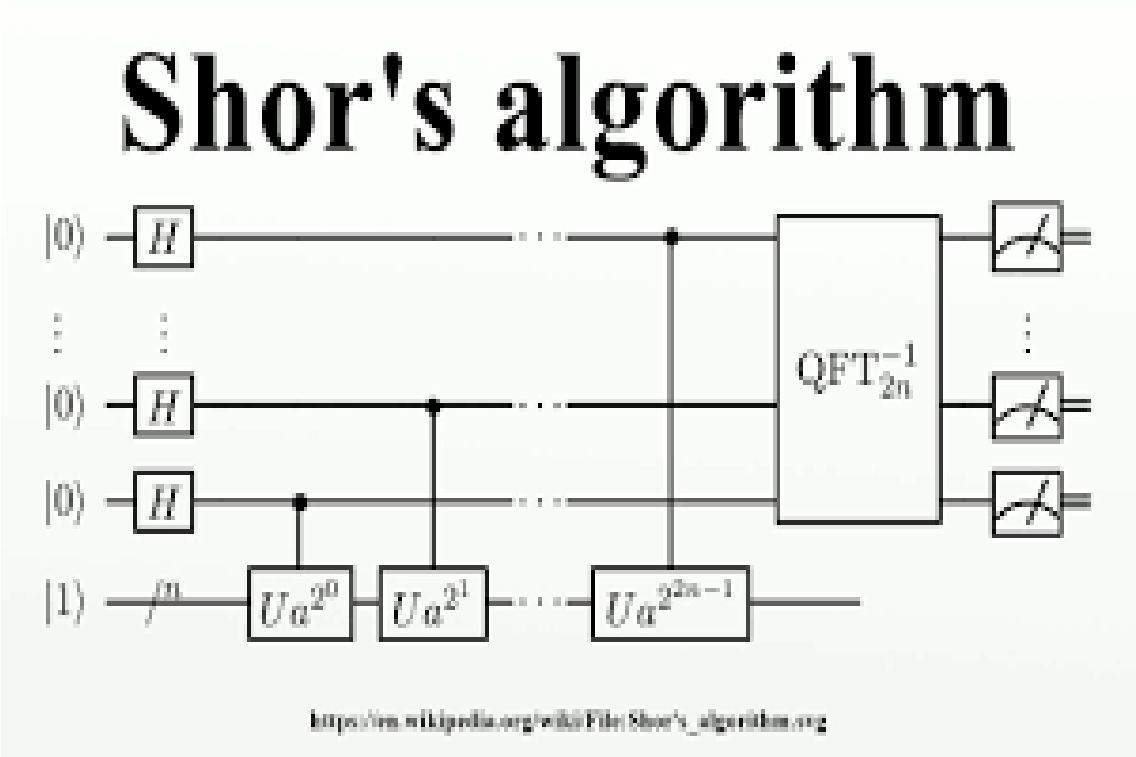


Figura 8-1, ***Esquema Circuito Cuántico del Algoritmo de Shor [Wikipedia]***

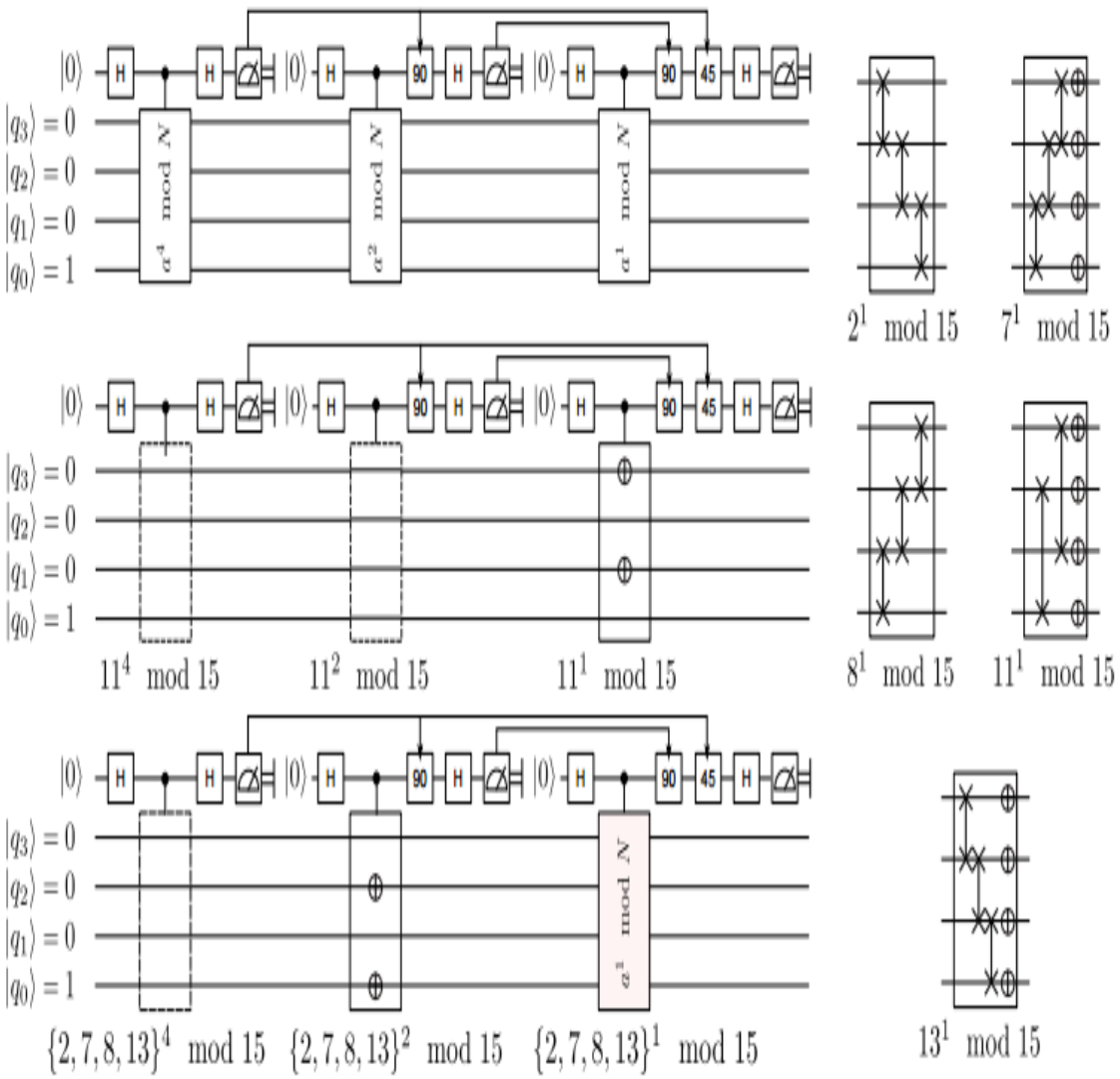


Figura 8-2, ***Esquema Circuito Cuántico del algoritmo de Kitaev [Naukas]***

En este algoritmo sólo hace falta un qubit de control ya que se va midiendo la fase (transformada inversa) independientemente para cada U controlada que se aplica. Es un ejemplo de implementación para factorizar el número 15.

9 CONCLUSIONES

En la computación cuántica al igual que con la clásica hay una parte matemática y otra física. Como hemos visto durante todo el trabajo la parte matemática de la computación cuántica se basa en las matrices y en los números complejos, al igual que la computación clásica se basaba en el Algebra de Boole. La parte matemática de la computación cuántica no presenta ningún problema. El gran reto es la mecánica cuántica es decir conseguir un ordenador cuántico funcional con el número de qubits necesarios para implementar el Algoritmo de Shor u otro cualquiera. Otro gran problema es la Decoherencia *[Página 37]*, los Qubits de los ordenadores actuales son inestables y necesitan unas condiciones de uso muy estrictas (vacío, temperatura, ambiente limpio, etc.) Y para finalizar la unión o arquitectura de los ordenadores es compleja ya que los qubits no se pueden unir uno con cualquiera, por lo cual hay hacer encajes de bolillos para implementar algunos algoritmos e incluso crear el ordenador para el algoritmo. Es decir no se dispone de un ordenador universal en el que se puedan implementar cualquier algoritmo.

Aunque con ordenadores cuánticos híbridos (parte cuántica y parte clásica) el record es mayor (e hace un Mezcla de algoritmos). El record actual con un computador cuántico puro es la factorización de 21. Hay que construir el circuito expresamente según el n que queramos factorizar. Para un número RSA de los que tenemos ahora se requeriría un ordenador cuántico de cientos de miles de qubits.

Lo que sí es una realidad es que si se llega alguna vez a desarrollar un ordenador cuántico en el que se pueda implementar el Algoritmo de Shor con el número de Qubits necesarios se conseguiría factorizar los números RSA actuales. Aunque siempre queda el incrementar el tamaño de los números pero está claro que estaría en duda la seguridad de RSA y de otros algoritmos de encriptación actuales. Pero como ya se ha dicho la cuántica también abre el campo para que se desarrollen nuevos algoritmos de encriptación.

Otro algoritmo cuántico importante es el Algoritmo de Grover que mejora los algoritmos de búsqueda actuales en una lista desordenada.

En definitiva aunque matemáticamente y teóricamente sea factible el Algoritmo de Shor en la actualidad estamos muy lejos de disponer un ordenador cuántico totalmente operativo, los pocos que existen tienen muy pocos qubits y son muy inestables. Además hay que recordar que la mayoría de los cálculos cuánticos son probabilísticos y no se tiene certeza 100 % de conseguir el resultado esperado.

10 BIBLIOGRAFÍA

[1]	Factorizan el número 15 usando el algoritmo de Kitaev con 5 átomos atrapados - La Ciencia de la Mula Francis (naukas.com)
[2]	https://www.google.es/url?sa=t&rct=j&q=&esrc=s&source=web&cd=&cad=rja&uact=8&ved=2ahUKEwiz6raPrqX6AhUU5hoKHTjmADkQFnoECBIQAQ&url=http%3A%2F%2Fusers.df.uba.ar%2Fpaz%2Fpag_comp_cuant%2Fresumenes%2Fclase12.pdf&usg=AOvVaw2lo0-prWfB42attmcceyhV&cshid=1663745329614844
[3]	https://www.google.es/url?sa=t&rct=j&q=&esrc=s&source=web&cd=&cad=rja&uact=8&ved=2ahUKEwjAzcbpsKX6AhUO4oUKHQ8-DzsQFnoECBQQAQ&url=https%3A%2F%2Febuah.uah.es%2Fdspace%2Fbitstream%2Fhandle%2F10017%2F30585%2FTFG-Fuentes-Izquierdo-2017.pdf%3Fsequence%3D1%26isAllowed%3Dy&usg=AOvVaw04jmHvxgFXHjsmhizuxPd0

Para los Algoritmos no Cuánticos se ha consultado sobre todo la Wikipedia y para la parte cuántica ha habido una gran parte de investigación que ha consistido en ver videos de Internet sobre el tema en especial se ha seguido el canal de "ket.g". Casi todo lo contemplado sobre la cuántica en el trabajo tiene un capítulo que lo trata. Por ejemplo un capítulo para la Transformada Inversa Cuántica de Fourier, otro para el Algoritmo de Shor, el Algoritmo de Deutsch-Jozsa, el Algoritmo de Estimación de Fase, etc.

Otros videos que se han visualizado son:

- "Computación Cuántica Estado de Desarrollo" Instituto de la Ingeniería de España (09/05/22)
- "Introducción a la Computación Cuántica (sin la física) Eduardo Sáenz de Cabezón. Universidad de la Rioja