Computación Cuántica

El Circuito Cuántico

Índice

Esquema 2

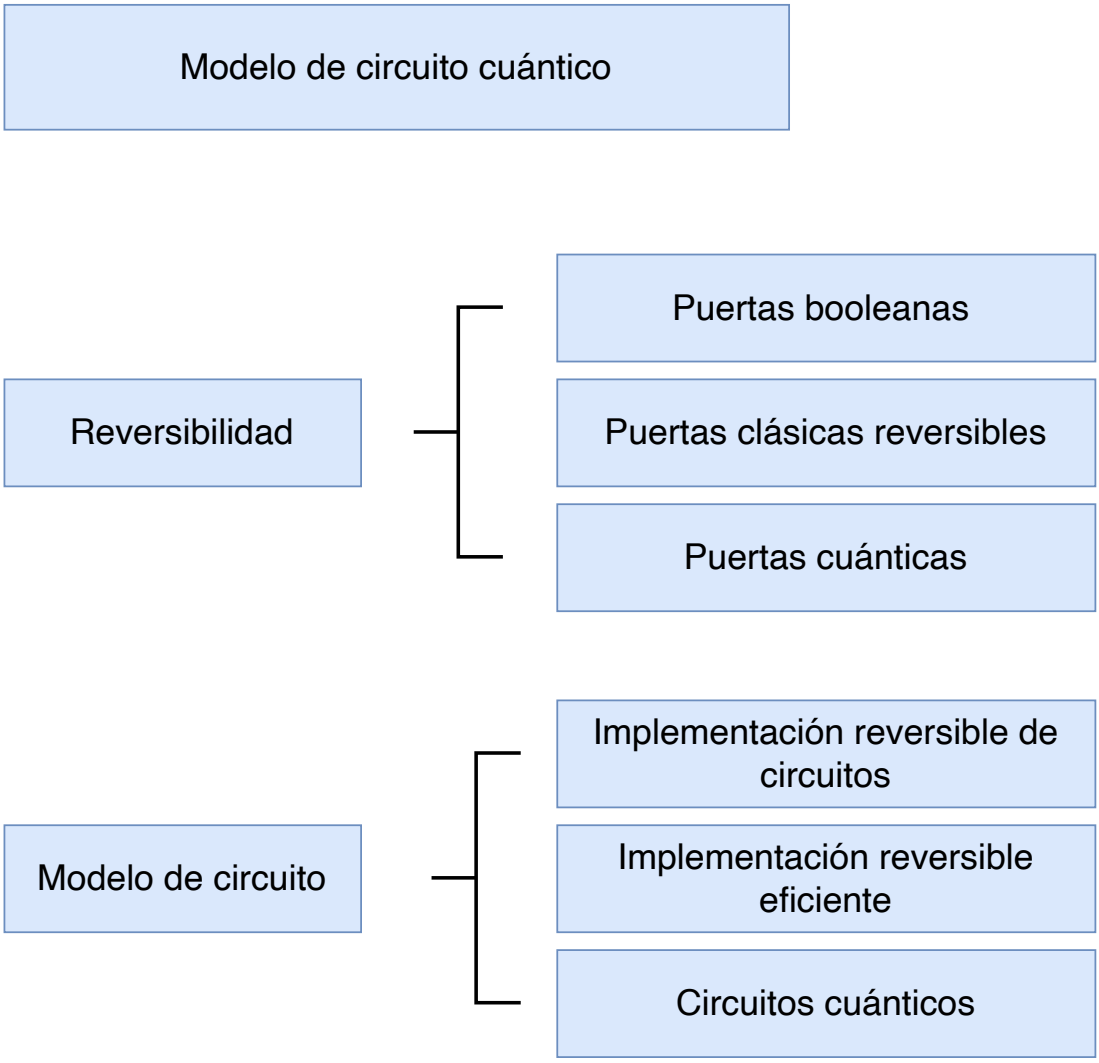
Ideas clave 3

7.1 Introducción y objetivos 3

7.2 Reversibilidad 4

7.3 Referencias bibliográficas 19

Esquema


Modelo de circuito cuántico
Reversibilidad
Puertas booleanas
Puertas clásicas reversibles
Puertas cuánticas
Modelo de circuito
Implementación reversible de circuitos
Implementación reversible eficiente
Circuitos cuánticos

Ideas clave

7.1 Introducción y objetivos

El modelo de circuito estándar en computación cuántica describe el proceso de cómputo mediante una secuencia de puertas simples seguidas de una serie de mediciones.

El conjunto de puertas utilizado en el modelo de circuito estándar, está formado por la puerta CNOT junto con las puertas de un cúbit, para las mediciones se realizan medidas de un cúbit utilizando la base computacional.

Si bien un conjunto finito de puertas sería más adecuado que el conjunto infinito de todas las transformaciones de un solo cúbit, es más fácil trabajar con el conjunto infinito. En cuanto a los cúbits sobre los que se ejecuta el circuito, normalmente se organizan en registros cuánticos o subconjuntos de cúbits que realizan distintas funciones dentro del algoritmo.

Existen otros modelos de computación cuántica, como la computación cuántica adiabática o la topológica. Uno de los puntos fuertes del modelo de circuito estándar es que permite el aprovechamiento directo de circuitos clásicos equivalentes. Encontrar circuitos cuánticos equivalentes a circuitos clásicos reversibles es una tarea sencilla, la dificultad está en convertir un circuito clásico arbitrario en un circuito clásico reversible.

Aunque es posible demostrar que cualquier transformación cuántica es realizable utilizando las puertas básicas del modelo de circuito estándar, ello no asegura su eficiencia. En este tema se verá que no solo se puede encontrar un equivalente cuántico para cualquier cómputo clásico, sino también otro cuántico con eficiencia comparable. Este resultado prueba que la computación cuántica es, al menos, tan potente como la computación clásica. Además, muchos algoritmos cuánticos comienzan creando una superposición de estados antes de usar medios no clásicos para extraer información

de manera eficiente de esta superposición.

La construcción de versiones cuánticas equivalentes a cualquier proceso de computo clásico se basa en la construcción de un circuito clásico, equivalente y reversible.

Se estudiará la relación entre la computación clásica reversible y la computación en general, tanto clásica como cuántica, así como las versiones reversibles de puertas lógicas booleanas y sus equivalentes cuánticas, ya que dado un circuito clásico reversible compuesto de puertas lógicas booleanas reversibles, la simple sustitución de las puertas cuánticas equivalentes por las puertas reversibles, proporciona el circuito cuántico deseado.

La dificultad en la demostración de que cualquier computación clásica tiene una equivalente cuántica de eficiencia comparable, reside en probar que cualquier computación clásica tiene una versión reversible de eficiencia comparable.

- Reversibilidad
- Puertas clásicas simples
- Puertas clásicas simples reversibles
- Puertas cuánticas simples
- Modelo de circuito
- Universalidad
- Aritmética

7.2 Reversibilidad

En función de la energía consumida durante la computación, podemos clasificar el proceso de cómputo en dos tipo, aquel que utiliza operaciones irreversibles y el que utiliza operaciones reversibles.

Una puerta lógica, cuya información de salida es menor que la información de entrada, es irreversible. Esta situación implica una pérdida de energía de algún tipo, sin embargo, una puerta lógica cuya información de salida es la misma que la información de entrada es reversible y, además, la información permanece constante, lo que significa que, al menos de una forma ideal, la energía necesaria para realizar la computación también permanece constante, es decir se puede realizar sin pérdida de energía.

La reversibilidad es fundamental para la computación cuántica ya que el procesador cuántico se comporta bajo las leyes de la física cuántica y, dado que estas leyes son reversibles en el tiempo, las operaciones deberán ser igualmente reversibles.

Cualquier secuencia de transformaciones cuánticas equivale a una transformación unitaria U aplicada al sistema cuántico. Mientras no se realice un proceso de medida, siempre será posible recuperar el estado inicial del sistema, a partir del estado final $|\psi\rangle$, aplicando $U^{-1} = U^{\dagger}$ sobre $|\psi\rangle$.

Cualquier proceso cuántico es reversible antes de realizar la medición y por tanto, la entrada del proceso siempre se puede conocer a partir de la salida.

Figura 1: Reversibilidad. Elaboración propia.

La computación clásica, en general, no es reversible, pues normalmente no es posible calcular la entrada a partir de la salida, así, mientras que la operación clásica NOT es reversible, AND no lo es. Sin embargo, todo cálculo clásico tiene un equivalente reversible clásico que requiere solo unos pocos recursos computacionales más.

En este tema, se mostrará cómo hacer que circuitos booleanos completos sean reversibles de una manera eficiente en recursos, considerando el espacio, el número de bits y el número de puertas requeridos. Esta forma de construir versiones reversibles clásicas eficientes de circuitos booleanos arbitrarios se generaliza fácilmente a una construcción de circuitos cuánticos que implementan eficientemente circuitos clásicos generales.

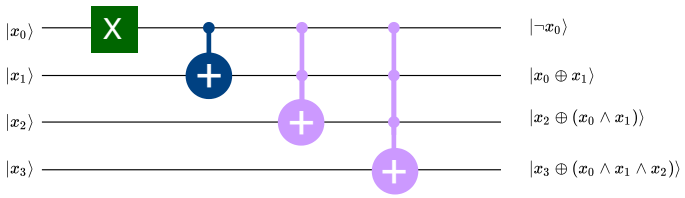


Figura 2: Puerta NOT, CNOT, CCNOT y CCCNOT. Elaboración propia.

Cualquier cálculo reversible clásico con n bits de entrada y n bits de salida simplemente permuta las $N = 2^n$ cadenas de bits y por tanto, para cualquier cómputo reversible clásico de este tipo, existe una permutación $P : Z_N \rightarrow Z_N$ que asigna una cadena de bits de entrada a su correspondiente cadena de bits de salida. Esta permutación se puede utilizar para definir una transformación cuántica que actúa sobre los vectores de la base estándar, vistos como cadenas de bits clásicos, exactamente como lo hace P:

$$U_P = \sum_{x=0}^{N-1} a_x|x\rangle \rightarrow \sum_{x=0}^{N-1} a_x|P(x)\rangle$$

Esta transformación U_P, es unitaria ya que simplemente reordena los elementos de la base estándar.

Cualquier computación clásica con n bits de entrada y m bits de salida define una función:

$$f : Z_N \rightarrow Z_M$$

$$x \rightarrow f(x)$$

asignando las cadenas de $N = 2^n$ bits de entrada a las cadenas de $M = 2^m$ bits de salida. Dicha función puede extenderse de forma canónica a una función reversible P_f que actúa sobre $n + m$ bits repartidos en dos registros: el registro de entrada de n bits y el de salida de m bits:

$$P_f : Z_L \rightarrow Z_L$$

$$(x, y) \rightarrow (x, y \oplus f(x))$$

donde $\oplus$ representa el operador a nivel de bit XOR.

La función P_f actúa sobre las $L = 2^{n+m}$ cadenas de bits, cada una de las cuales formada por una cadena x de n bits y otra cadena y de m bits. Para el caso de $y = 0$ la función P actúa como f, con la diferencia de que la salida aparece en el registro de salida y el registro de entrada conserva la entrada.

Existen distintas formas de hacer que un proceso de cómputo clásico sea reversible. Para un cálculo clásico en particular, puede haber una versión reversible que requiera menos bits, pero en cualquier caso, esta construcción siempre funciona. Dado que P_f es reversible, existe una transformación unitaria correspondiente $U_f : |x, y\rangle \rightarrow |x, y \oplus f(x)\rangle$.

Gráficamente la transformación U_f tiene la siguiente representación:

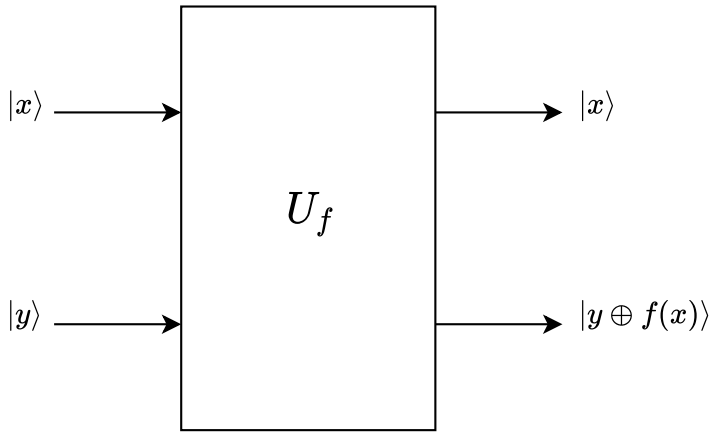


Figura 3: Transformación reversible. Elaboración propia.

En el tema anterior se mostró cómo implementar cualquier operación unitaria en términos de puertas simples. Para la mayoría de las transformaciones unitarias, esa implementación es muy ineficiente, sin embargo, U_f tiene una implementación eficiente, ya que hay un circuito clásico que calcula f de manera eficiente.

El método para construir una implementación eficiente de U_f a partir de un circuito clásico eficiente para f, tiene dos partes, primero se construye un circuito clásico reversible eficiente que calcula f, a continuación se sustituye cada una de las puertas reversibles que forman el circuito clásico reversible por una puerta cuántica.

A continuación se describen las versiones reversibles de algunas puertas lógicas booleanas, como XOR, AND, OR, etc. Sus equivalentes cuánticas son puertas igualmen-

te reversibles que actúan sobre los elementos de la base computacional y su acción sobre otros estados de entrada está prescrita por la linealidad de las operaciones cuánticas, así la acción de una puerta sobre una superposición es la combinación lineal de la acción de la puerta sobre los elementos de la base estándar que componen la superposición:

$$X(\alpha|0\rangle + \beta|1\rangle)) = \alpha X|0\rangle + \beta X|1\rangle = \beta|0\rangle + \alpha|1\rangle$$

De esta manera, el comportamiento de una puerta reversible clásica define completamente el comportamiento de su análogo cuántico y viceversa, esta estrecha conexión permite usar la misma notación para ambas pero recordando que las puertas cuánticas se pueden aplicar a superposiciones arbitrarias, mientras que las puertas reversibles clásicas se aplican a cadenas de bits que corresponden a los elementos de la base estándar.

Se definen las siguientes puertas cuánticas:

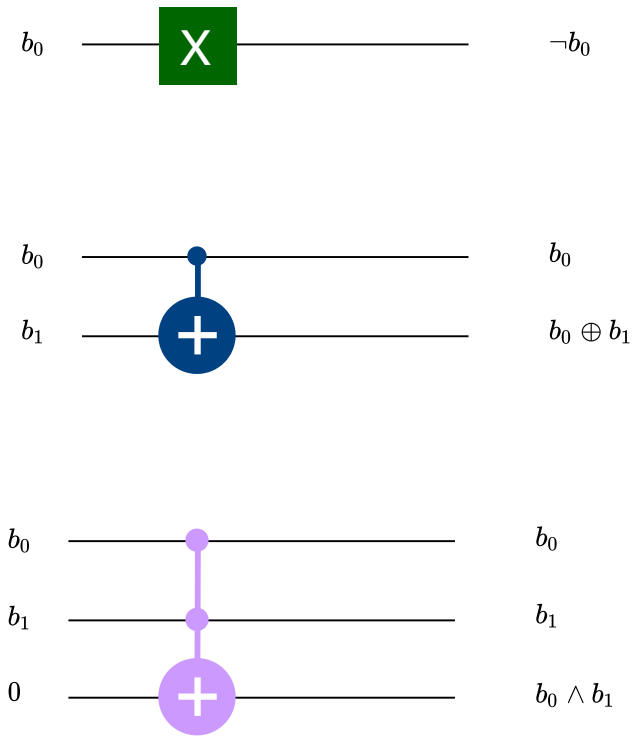


Figura 4: Circuitos cuánticos que implementan las puertas NOT, XOR y AND. Elaboración propia.

NOT. Se trata de una puerta reversible y se utiliza "X" para referirse a ella, tanto para

su versión clásica como para el operador de un solo cúbit: $X = |0\rangle\langle 1| + |1\rangle\langle 0|$, que realiza una operación NOT clásica sobre bits clásicos codificados como elementos de la base estándar.

AND. Para implementar una versión reversible de la operación AND son necesarios tres bits, la puerta de Toffoli or $CCNOT$ permite su implementación.

$NAND$. Utilizando la implementación para la puerta AND y a continuación, negando el resultado.

XOR. La puerta NOT controlada ($CNOT$) realiza la operación XOR sobre los bits de entrada, conservando el bit de control y proporcionando la operación XOR de los dos valores de entrada en el bit objetivo. La versión cuántica se comporta como la versión reversible sobre los vectores de la base estándar y su comportamiento sobre cualquier otro estado se deriva de la linealidad del operador.

Variando los valores de los bits de entrada, la puerta de Toffoli permite crear un conjunto completo de operadores booleanos, no solo la puerta AND. Así, permite implementar las puertas NOT, AND, XOR y $NAND$ de la forma siguiente:

$$NOT : CCNOT|x, 1, 1\rangle = |\neg x, 1, 1\rangle$$

$$AND : CCNOT|0, y, x\rangle = |x \bigwedge y, y, x\rangle$$

$$XOR : CCNOT|y, x, 1\rangle = |x \oplus y, x, 1\rangle$$

$$NAND : CCNOT|1, y, x\rangle = |\neg(x \bigwedge y), y, x\rangle$$

De la misma forma que CCNOT, la puerta CSWAP o Fredkin puede implementar un conjunto completo de operadores booleanos. La puerta swap actúa de la siguiente forma:

$$S : |xy\rangle \rightarrow |yx\rangle$$

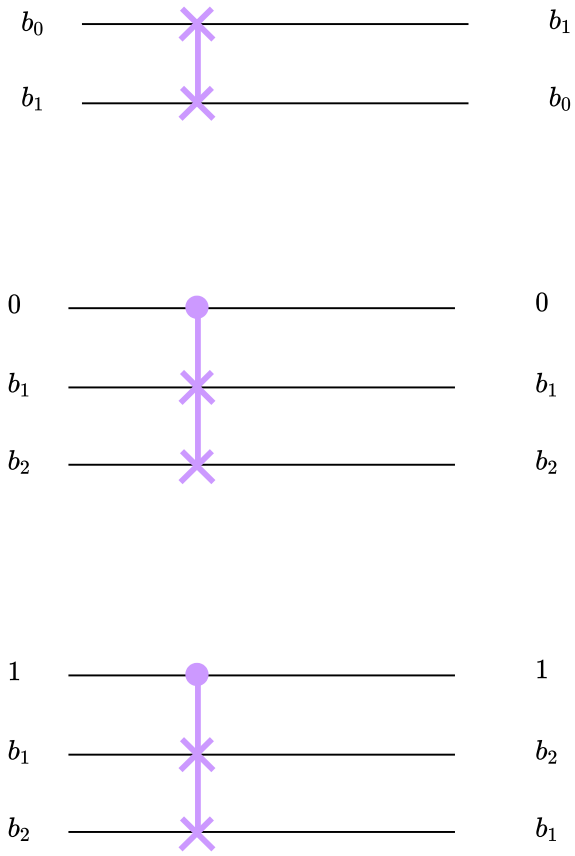


Figura 5: Puerta SWAP y CONTROL SWAP o Fredkin. Elaboración propia.

Puesto que utilizando únicamente CCNOT o CSWAP es posible implementar el conjunto completo de operadores booleanos clásicos, estas puertas se pueden combinar para realizar circuitos booleanos arbitrarios. Por mayor claridad, se utilizan las puertas $CNOT$ y X en los circuitos siguientes, pero todas las construcciones pueden realizarse usando solo puertas Toffoli, ya que podemos reemplazar todos los usos de $CNOT$ y X con puertas Toffoli. El siguiente circuito es un sumador completo que utiliza puertas $CNOT$ y Toffoli, donde $b0$ y $b1$ son los bits de datos, s es su suma (módulo 2), c es el bit de acarreo entrante y c' es el nuevo bit de acarreo. Se pueden unir varios sumadores de un bit para lograr una suma completa de n bits.

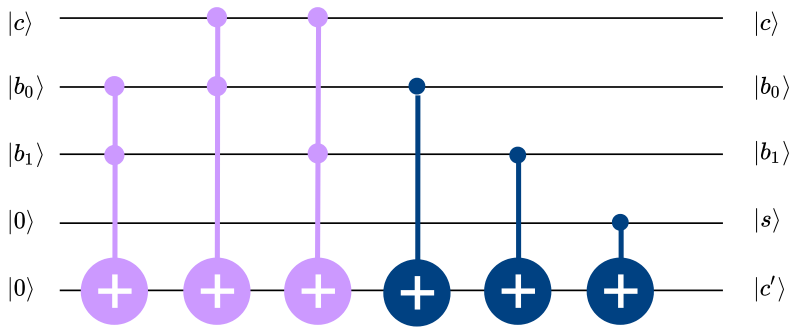


Figura 6: Sumador de dos bits completo. Elaboración propia.

Es posible desarrollar una forma sistemática de convertir circuitos booleanos clásicos arbitrarios en circuitos clásicos reversibles de eficiencia computacional comparable, tanto en el número de bits como en el número de puertas. Los circuitos reversibles resultantes se componen completamente de puertas Toffoli y puertas NOT. Un circuito cuántico, con la misma eficiencia que el circuito reversible clásico, se obtiene mediante la sustitución trivial de puertas cuánticas de Toffoli y X por las correspondientes puertas clásicas de Toffoli y NOT. Por lo tanto, tan pronto como tengamos una versión eficiente de un proceso de cómputo en términos de puertas de Toffoli, inmediatamente sabremos cómo obtener una implementación cuántica de la misma eficiencia.

Consideremos una máquina clásica que consta de un registro de bits y una unidad de procesamiento. La unidad de procesamiento realiza operaciones booleanas simples que actúan sobre uno o dos de los bits y almacena el resultado en uno de los bits. Se supone que, para un tamaño de entrada dado, la secuencia de operaciones y su orden de ejecución son fijos y no dependen de los datos de entrada ni de algún control externo.

Un circuito booleano arbitrario se puede transformar en una secuencia de operaciones en un registro lo suficientemente grande como para contener bits de entrada, de salida e intermedios. La complejidad espacial de un circuito es el tamaño del registro. Los cálculos realizados por esta máquina no son reversibles en general. Al reutilizar bits del registro, se pierde información que no se puede recuperar más tarde.

Una solución a este problema trivial pero muy ineficiente espacialmente (tamaño del

registro), es no reutilizar los bits durante todo el proceso de computación. La operación que calcula reversiblemente la puerta booleana AND y deja el resultado en un bit inicialmente en estado 0 es, por supuesto, la puerta de Toffoli, como se ha visto. Dado que la puerta NOT es reversible, y NOT junto con AND forman un conjunto completo de operaciones booleanas, esta construcción se puede generalizar para convertir cualquier cómputo que utilice operaciones lógicas booleanas en uno que utilice sólo puertas reversibles. Esta implementación, sin embargo, necesita un bit adicional para cada operación AND que se realiza, por lo que si el circuito original necesita un número de pasos P, entonces, su versión reversible, construida de esta forma trivial, requiere hasta P bits adicionales de espacio. Además, este espacio adicional ya no está en el estado 0 y no se puede reutilizar.

La reutilización de los bits temporales es fundamental para mantener una complejidad espacial parecida a la del circuito original no reversible.

Restablecer un bit a cero no es trivial. Una transformación que restablece un bit a 0, independientemente de si antes era 0 o 1, no es reversible (se pierde información), por lo que no se puede utilizar como parte de un cálculo reversible. Los cálculos reversibles no pueden recuperar espacio mediante una simple operación de reinicio. Sin embargo, pueden revertir el estado de cualquier conjunto de bits durante el proceso de cómputo reversible invirtiendo la parte correspondiente de la computación.

Se puede reducir el número de cúbits necesarios invirtiendo la computación y reutilizándolos en el transcurso del algoritmo. El método de invertir la computación al realizar todos los pasos en orden inverso, excepto aquellos que proporcionan el resultado, funciona para cualquier circuito booleano clásico. Para un subcircuito booleano clásico de n puertas que operan en un registro de m bits, la construcción trivial requiere hasta n bits adicionales en el registro.

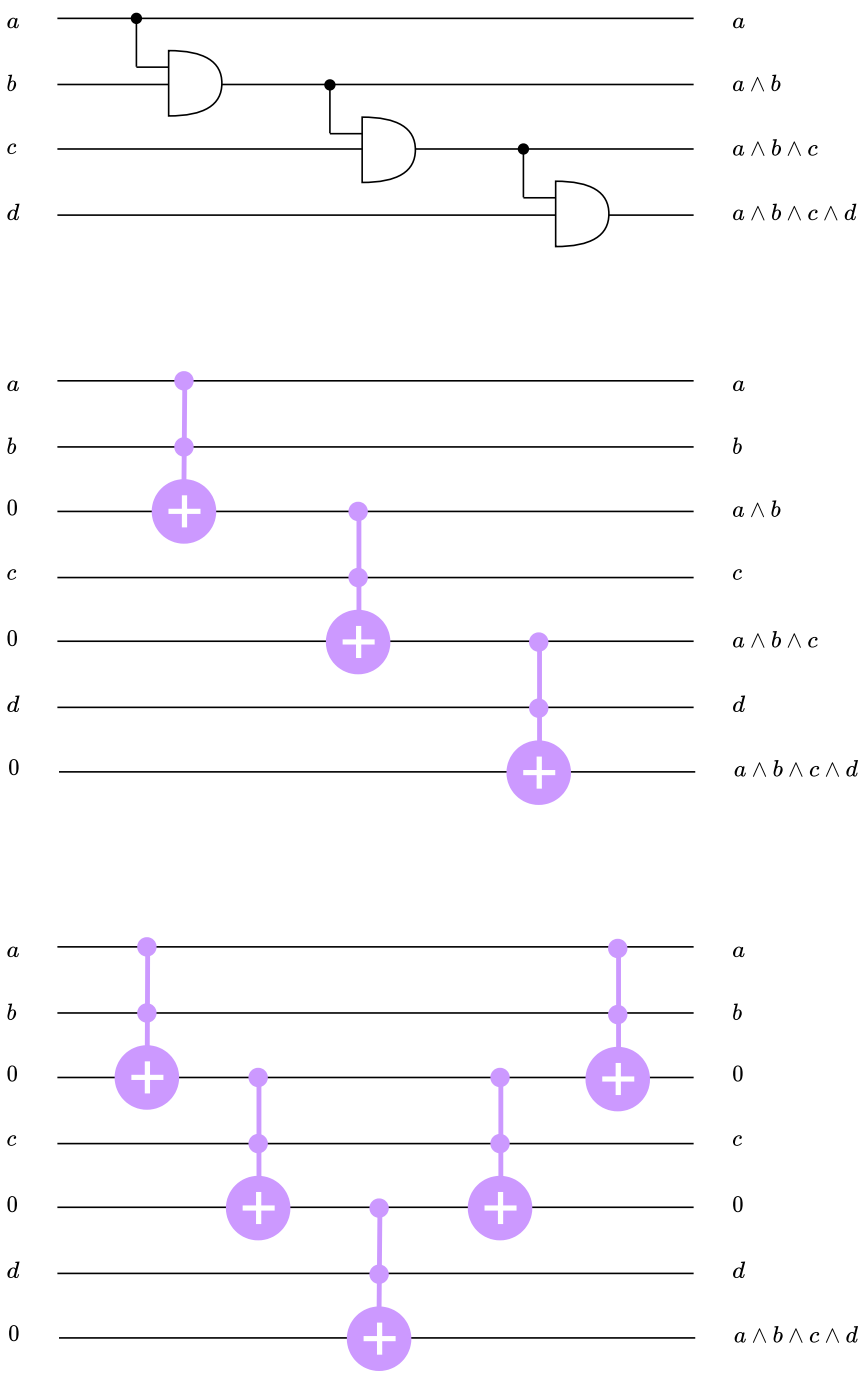


Figura 7: Circuito irreversible, reversible y reutilización de bits realizando la computación inversa. Elaboración propia.

Eligiendo cuidadosamente para qué bits realizar la computación inversa y en que momento, se puede lograr una versión reversible de cualquier cálculo clásico con pequeños incrementos en el número de puertas y bits. Se ha demostrado que cualquier circuito clásico que usa t puertas y s bits, tiene una versión reversible que tan solo re-

quiere un número de puertas del orden de $O(t^{1+\varepsilon})$ y un número de bits del orden de $O(s\ log\ t)$. Cuando el número de puertas es muy superior al número de bits, este tipo de construcción requiere un espacio considerablemente menor que el espacio $s + t$ para la construcción trivial descrita anteriormente con solo un pequeño incremento en el número de puertas.

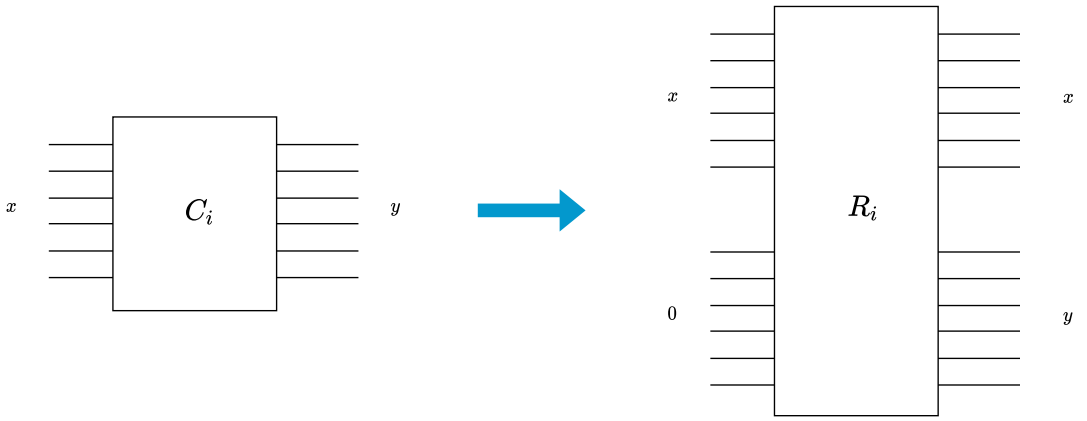


Figura 8: Construcción de un circuito reversible a partir de uno irreversible. Elaboración propia.

Para entender cómo se deducen estos límites, se debe considerar cuidadosamente cuántos bits se están utilizando y de qué forma. Sea C un circuito clásico, compuesto por puertas AND y NOT, que utiliza t puertas y s bits. El circuito C se puede dividir en el tiempo, en subcircuitos r, cada uno de los cuales contiene t o menos puertas consecutivas $C = C_1C_2...C_r$. Cada subcircuito C_i tiene s bits de entrada y s de salida, algunos de los cuales pueden permanecer sin cambios.

Como se ha visto anteriormente, cada circuito C_i puede ser reemplazado por un circuito reversible CR que usa como máximo s bits adicionales, como se muestra en la figura anterior. El circuito reversible R_i devuelve tanto los valores de entrada como los s bits de salida que serán utilizados en una computación posterior. Los valores de entrada se utilizarán para realizar la computación inversa y volver a computar R_i con el fin de ahorrar espacio.

Es posible que se requieran más de t puertas para construir R_i. En general, R_i se puede construir utilizando como máximo el triple de puertas ($3t$). Si bien son posibles otras construcciones más eficientes, aplicando los siguientes pasos se consigue una versión reversible para cualquier circuito.

En primer lugar se deberán realizar todos los cómputos parciales de forma reversible. Así, por cada puerta AND o NOT en el circuito original C_i, el circuito R_i tendrá una puerta Toffoli o una puerta NOT. Este paso usa el mismo número de puertas, t, que C_i, y no usa más de s bits adicionales.

En segundo lugar se copiarán todos los valores de salida, que serán utilizados en las partes posteriores del cómputo, en el registro de salida, un conjunto no superior a s bits adicionales.

Finalmente se realizará la secuencia de puertas utilizada para realizar el primer paso, pero en orden inverso. De esta forma, todos los bits, excepto los del registro de salida, se restablecen a sus valores originales. Específicamente, todos los bits temporales se reinicializan a 0 y se han recuperado todos los valores de entrada.

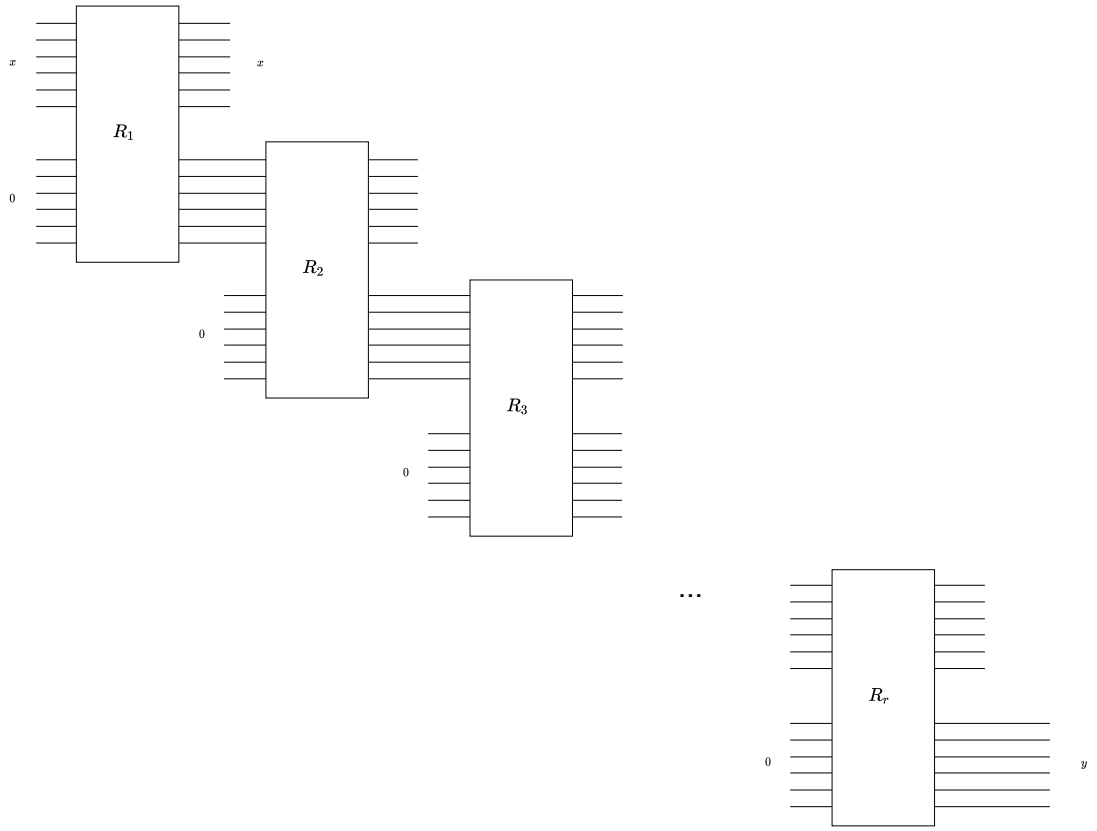


Figura 9: Composición de circuitos R_i para conseguir una versión reversible pero ineficiente del circuito C. Elaboración propia.

Los circuitos $R_1 ... R_r$, cuando se combinan como en la figura 7, realiza la computación de forma reversible pero ineficiente, los subcircuitos R_i se pueden combinar de forma que aprovechen el espacio de forma más eficiente realizando la computación inversa y

reutilizando algunos de los bits. La computación inversa requiere puertas adicionales, y por ello se debe elegir cuidadosamente cuándo realizarla para reducir el espacio sin incrementar excesivamente el número de puertas.

A continuación se muestra cómo obtener una versión reversible usando un número de puertas del orden de $O(t^{log_2 3})$ y un número de bits del orden de $O(s\ log\ t)$, y luego mejoramos este método para conseguir un orden de $O(t^{1+\varepsilon})$ puertas y un orden de $O(s\ log\ t)$ bits.

El principio básico para combinar los circuitos es realizar una computación inversa y volver a computar partes del estado, de forma selectiva, para reutilizar el espacio. Se modifica sistemáticamente la computación $R_1 R_2 ... R_r$ para reducir, tanto la cantidad total de espacio utilizado, como para reiniciar todos los bits temporales a cero al final de la computación.

Para simplificar el análisis, se considera *r* una potencia de 2, $r = 2^k$. Para $1 \leq i \leq k$, sea $r_i = 2^i$. Realizamos la siguiente transformación recursiva B que divide una secuencia en dos partes de igual tamaño, transforma de forma recursiva las partes y luego las compone de la manera que se muestra:

$$B(R_1, ..., R_{r_{i+1}}) = B(R_1, ..., R_{r_i}) B(R_{1+r_i}, ..., R_{r_{i+1}}) (B(R1, ..., R_{r_i}))^{-1}$$

$$B(R) = R,$$

donde $(B(R_1, ..., R_{t_i}))^{-1}$ actúa exactamente en los mismos bits que $B(R_1, ..., R_{t_i})$ y, por lo tanto, no requiere espacio adicional.

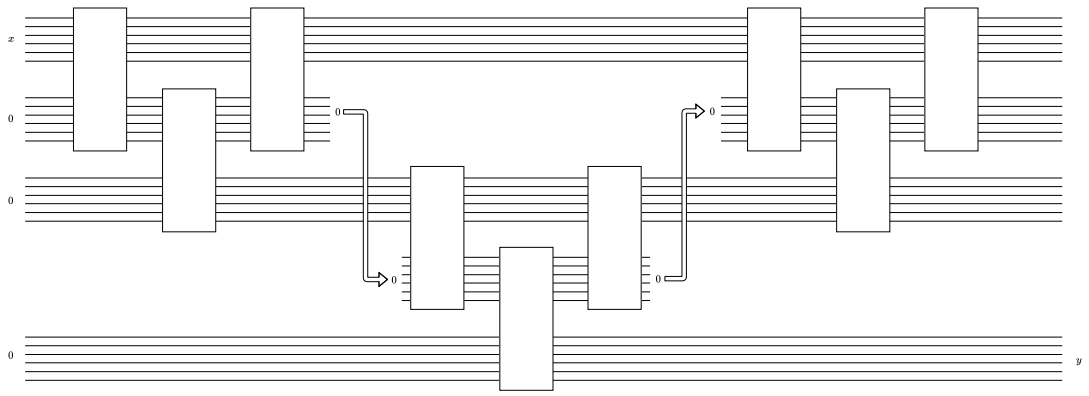


Figura 10: Composición de circuitos R_i para conseguir una versión reversible que reutiliza almacenamiento. Elaboración propia.

Esta computación resultado de la transformación, realiza la computación inversa de todo el espacio excepto del resultado del último paso, por lo que la necesidad de espacio adicional está limitado por s. Por lo tanto, $B(R_1, ..., R_{r_i})$ requiere, como máximo, s más espacio que $B(R_1, ..., R_{r_{i-1}})$. Se puede escribir el espacio $S(i)$ necesario para cada uno de los $k = log_2 r$ pasos i en la recursión en términos de las necesidades del paso anterior: $S(i) \leq s + S(i-1)$ con $S(1) \leq 2s$. La recursión finaliza después de $k = log_2 r$ pasos, por lo que el cómputo final $B(R_1, ..., R_r)$ requiere como máximo $S(r) \leq (k+1)s = s(log_2 r + 1)$ espacio. De la definición de B, se sigue inmediatamente que $T(i)$, el número de circuitos R_j ejecutados por el cálculo $B(R_1, ..., R_{r_i})$, es $T(i) = 3T(i-1)$ con $T(1) = 1$. Suponiendo que $r = 2^k$, la versión reversible de C construida utiliza $T(2^k) = 3T(2^{k-1}) = 3^k = 3^{log_2 r} = r^{log_2 3}$ circuitos reversibles R_i, cada uno de los cuales requiere menos de 3 puertas. Por tanto, cualquier cálculo clásico de t pasos y s bits se puede realizar de forma reversible en un orden de $O(t^{log_2 3})$ pasos y $O(s\ log_2 t)$ bits.

Para obtener el límite de $O(t^{1+\epsilon})$, en lugar de utilizar una descomposición binaria, se considera la siguiente descomposición: para simplificar el análisis, se supone que r es una potencia de m, $r = m^k$. Para $1 \leq i \leq k$ sea $r_i = m^i$. Abreviando $R_{1+(x-1)r_i}, ..., R_{xr_i}$ como $\vec{R}_{x,i}$, entonces:

$$B(\vec{R}_{1,i+1}) = B(\vec{R}_{1,i}, \vec{R}_{2,i} ... \vec{R}_{m,i}) =$$

$B(\vec{R}_{1,i}), B(\vec{R}_{2,i}), ...B(\vec{R}_{m-1,i}),$

$B(\vec{R}_{m,i}),$

$B(\vec{R}_{m-1,i})^{-1}, ...B(\vec{R}_{2,i})^{-1}, B(\vec{R}_{1,i})^{-1}$

$B(R) = R.$

En cada paso de la recursión, cada bloque se divide en m piezas y se reemplaza con $2m-1$ bloques. En este punto $r = m^k$ subcircuitos han sido reemplazados por $(2m-1)^k$ circuitos reversibles R_i, de forma que el número total de circuitos R_i para la computación final es de $(2m-1)^k$.

El número de de puertas de R_i está limitado a $3s$. El número total de puertas de un circuito reversible de t puertas viene dado por:

$T(t) \approx 3s(\frac{t}{s})^{1+\frac{1}{log_2 m}} < 3t^{1+\frac{1}{log_2 m}}$

Por lo tanto, para cualquier $\varepsilon > 0$, es posible elegir m lo suficientemente grande como para que el número de puertas necesarias para la computación reversible sea del orden de $O(t^{1+\varepsilon})$ siendo el límite de espacio el mismo que antes, del orden de $O(s log_2 t)$.

Las versiones reversibles de los circuitos booleanos clásicos construidos de esta manera se pueden convertir directamente en circuitos cuánticos que constan completamente de puertas Toffoli y puertas X. Charles Bennett demostró, aprovechando el mismo argumento que el seguido anteriormente, que cualquier máquina de Turing clásica puede convertirse en una máquina de Turing reversible. A partir de cualquier circuito clásico para f, se puede construir una implementación de U_f con un número de puertas y bits semejante.

El cuidado necesario para realizar la computación inversa y reutilizar los bits se generaliza a los cúbits donde la necesidad de realizar la computación inversa es todavía mayor pues se debe asegurar que los cúbits temporales no se entrelacen con los cúbits de salida. Esta necesidad de eliminar el entrelazamiento con los cúbits auxiliares al final del proceso de cómputo es una de las diferencias entre la implementación clásica y cuántica.

Las transformaciones cuánticas, al ser reversibles, no pueden simplemente reiniciar los cúbits. Se podría pensar que los cúbits auxiliares se pueden reiniciar realizando un proceso de medida y luego, dependiendo del resultado, aplicar una transformación para reiniciarlos al estado $|0\rangle$. Pero si los cúbits auxiliares estaban entrelazados con cúbits que contienen el resultado del algoritmo, la medición de los cúbits auxiliares podría alterar los resultados. La computación inversa elimina el entrelazamiento entre los cúbits auxiliares y el resto del sistema sin afectar el estado del resto del sistema.

7.3 Referencias bibliográficas

Nielsen and Chuang (2011) Quantum Computation and Quantum Information

Aaronson (2013), Quantum Computing Since Democritus

Eric R. Johnston, Nic Harrigan y Mercedes Gimeno-Segovia (2019), Programming Quantum Computers

Eleanor Rieffel and Wolfgang Polak (2011), Quantum Computing

Robert Sutor (2019), Dancing with cúbits