Emulación en hardware de circuitos cuánticos basados en compuertas Toffoli

Hardware emulation of quantum circuits based on Toffoli gates

Jorge E. Duarte-Sánchez, Jaime Velasco-Medina[*]

Grupo de Bionanoelectrónica, Universidad del Valle. Edificio 354, Espacio 1011, Ciudad Universitaria Meléndez. Calle 13 No 100-00. C. P. 760033. Santiago de Cali, Colombia.

(Recibido el 21 de junio de 2011. Aceptado el 11 de febrero de 2014)

Resumen

Este trabajo presenta el diseño de una arquitectura hardware para emular circuitos cuánticos basados en compuertas tipo Toffoli con la cual se pueden emular circuitos de más de 50 qubits. El estado del sistema se obtiene procesando independientemente cada estado base mediante las funciones determinadas por las compuertas cuánticas para cada qubit; el tiempo requerido para la emulación crece exponencialmente en función del número de qubits que se usan solamente para generar una superposición de estados, y no en función de la cantidad total de qubits del sistema como ocurre cuando se usa la representación matricial convencional. Adicionalmente, se diseñó un arreglo de unidades de procesamiento para disminuir el tiempo de ejecución. Los resultados de síntesis permiten concluir que se requieren 9,35 segundos para emular la exponenciación modular de 8 bits, la cual utiliza 48 qubits, 155.312 compuertas cuánticas y requiere procesar 131.072 estados base. Estos resultados también permiten estimar que se pueden implementar 256 unidades de procesamiento de 52 qubits en el FPGA EP3C120F780I7.

----------*Palabras clave:* Computación cuántica, compuerta toffoli, procesamiento paralelo, emulación, implementación hardware

Abstract

This work presents the design of a hardware architecture for the emulation of quantum circuits based on Toffoli gates allowing the emulation of more than 50 qubits. The state of the system is obtained processing each basis state by means of the functions determined by the quantum gates for each qubit; the time required to execute the emulation grows exponentially only with the

* Autor de correspondencia: teléfono: + 57 + 2 + 3391780 ext. 117, correo electrónico: jaime.velasco@correounivalle.edu.co (J. Medina)

number of qubits that are used to generate a superposition of states, but not with the total amount of qubits of the system as occurs when the conventional matrix representation is used. Additionally, an array of processing units was designed to decrease the execution time. The synthesis results allow concluding that 9.35 seconds are required to emulate the 8-bit modular exponentiation, which uses 48 qubits, 155,312 quantum gates and requires processing 131,072 basis states. Furthermore these results allow estimating that 256 processing units of 52 qubits can be implemented in the FPGA EP3C120F780I7.

----------*Keywords:* Quantum computation, toffoli gate, parallel processing, emulation, hardware implementation

Introducción

Los circuitos cuánticos son los bloques funcionales que permiten llevar a cabo la computación cuántica. Estos circuitos son usados para implementar algoritmos cuánticos con el fin de resolver problemas de alta complejidad computacional, por ejemplo la factorización de números enteros de gran tamaño, la búsqueda en una lista desorganizada, el cálculo del logaritmo discreto, etc. Sin embargo, hoy en día no existe un procesador cuántico de un gran número de qubits, por lo tanto, muchos de los trabajos relacionados con la computación cuántica se basan es simulaciones o implementaciones no escalables.

Las simulaciones en su mayoría se basan en la representación matricial de los sistemas cuánticos. En esta representación, cuando se considera un sistema de *n* qubits, los estados son vectores de 2^n números complejos, y las compuertas se representan por matrices de 2^n x 2^n números complejos. Por lo tanto, la simulación de sistemas cuánticos es una tarea compleja debido a la gran cantidad de información que se requiere procesar donde el tiempo requerido crece exponencialmente en función del número de qubits simulados. Por esta razón, algunos investigadores trabajan en la emulación de sistemas cuánticos usando FPGAs [1-3] o clusters de computadores [4, 5], los cuales permiten realizar procesamiento paralelo; sin embargo, la cantidad de recursos requeridos para procesar en paralelo un gran número de qubits (mayores a 20) usando la representación matricial es muy grande y los recursos disponibles en un FPGA no son suficientes. Entonces, con el propósito de disponer de un sistema hardware que permita emular circuitos cuánticos de una gran cantidad de qubits, es necesario implementar una representación distinta a la matricial.

En este trabajo se presenta el diseño de un procesador para emular circuitos cuánticos del orden de 50 qubits, en donde los estados base son procesados de manera independiente ejecutando las operaciones de las compuertas cuánticas sobre cada qubit del sistema. En este caso es posible realizar procesamiento paralelo utilizando bloques funcionales cuyo tamaño es linealmente proporcional al número de qubits. Los circuitos cuánticos que pueden ser emulados utilizando esta representación deben tener únicamente compuertas Hadamard en la primera etapa del circuito para generar superposición de estados, y compuertas tipo Toffoli en las demás etapas.

El artículo está organizado de la siguiente manera: inicialmente se presenta una descripción de la estructura general de los circuitos cuánticos; luego, se describe las compuertas cuánticas tipo Toffoli. Posteriormente se describe el principio de funcionamiento y la arquitectura del procesador diseñado seguido por los resultados de síntesis y simulación. Finalmente, se presentan las conclusiones y el trabajo futuro.

Circuitos y compuertas Cuánticas

En la figura 1 se muestra una estructura general de un circuito cuántico de *n* qubits el cual se compone de tres bloques: Hadamard (H), Transformación Unitaria (U) y Medición (M). Cada bloque puede procesar *m* qubits, donde $0 < m \leq n$. Si un algoritmo requiere la aplicación de varios circuitos cuánticos, esta secuencia de bloques se puede repetir, omitiendo en algunos casos el bloque H.

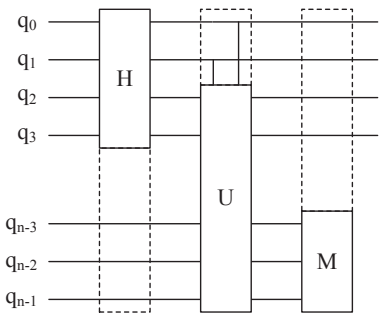

Figura 1 Estructura general de un circuito cuántico

El Bloque Hadamard (H) es implementado usando únicamente compuertas Hadamard y su función es transformar el estado inicial del sistema en una superposición equiprobable de estados base, los cuales serán procesados por el bloque U. El bloque de Transformación Unitaria (U) es implementado usando cualquier tipo de compuerta cuántica que corresponden a transformaciones unitarias; por ejemplo, compuertas de un solo qubit (compuertas de Pauli, compuerta de fase, compuerta $\pi/8$, etc.), o compuertas controladas y no controladas de varios qubits (compuertas de rotación, CNOT, Toffoli, etc.). El Bloque de Medición (M) es un bloque conceptual debido a que no es implementado usando compuertas cuánticas sino que representa una observación realizada sobre los qubits del sistema; sin embargo, este bloque tiene un efecto importante sobre su estado (generalmente una superposición) ya que hace que este colapse a un subconjunto de estados base que dependen del valor observado.

En la figura 2 se muestra uno de los circuitos cuánticos usados para implementar el algoritmo de Shor para la factorización de números enteros muy grandes [6]. La salida de este circuito es una superposición de estados que contiene de forma implícita el periodo de una función de exponenciación modular, la cual permite encontrar los factores de un número entero. El circuito tiene $n+m$ qubits; *n* qubits son inicializados en el estado $|0\rangle$, y *m* qubits en el estado $|1\rangle^{\otimes m}$. En este circuito se pueden identificar claramente los tres bloques mencionados anteriormente: (1) bloque de compuertas Hadamard que transforma el estado de los *n* qubits del estado inicial $|0\rangle$ al estado $|\Psi_1\rangle$, el cual es la superposición lineal de los estados base desde $|0\rangle$ hasta $|2^{n-1}\rangle$. Debido a que en los *m* qubits no se aplican compuertas Hadamard su estado permanece en $|1\rangle^{\otimes m}$; (2) bloque de transformación unitaria que implementa la exponenciación modular, en donde $|\Psi_1\rangle = a$ es el exponente, y cuyo efecto sobre el estado del sistema es asociar a cada estado base de $|\Psi_1\rangle$ su respectiva solución a la exponenciación modular con parámetros X y N (en este caso X = 7 y N = 15). Por ejemplo, para $a = 3$, $7^3 \bmod 15 = 13$, entonces, uno de los estados del sistema en este punto es $|\Psi_2\rangle = |3\rangle|13\rangle$; (3) bloque de medición que actúa sobre los *m* qubits inferiores. Después de la medición, el estado de los *n* qubits $|\Psi_2\rangle$ colapsa a una superposición de estados que es un subconjunto de $|\Psi_1\rangle$. Este subconjunto contiene los estados que anteriormente estaban asociados al valor observado (7), que en el ejemplo son $|1\rangle$, $|5\rangle$, $|9\rangle$, etc.

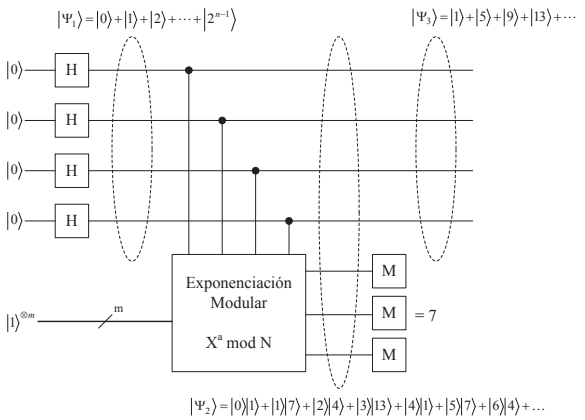

Figura 2 Circuito cuántico usado en el algoritmo de Shor para encontrar el periodo de una función de exponenciación modular

El circuito de exponenciación modular usado como bloque de transformación unitaria en el ejemplo anterior, así como muchos otros circuitos usados en computación cuántica, se implementan utilizado únicamente compuertas tipo Toffoli. Estas compuertas son la base para implementar operaciones booleanas usadas en la implementación de circuitos cuánticos aritméticos convencionales o cualquier tipo de función que pueda implementarse con compuertas lógicas clásicas.

Compuertas tipo Toffoli

La compuerta Toffoli es un operador reversible que permite emular el comportamiento de cualquier compuerta digital clásica. Una compuerta reversible tiene el mismo número de entradas y salidas; en el caso de la Toffoli, dos salidas corresponden a las entradas sin modificación (bits de control), mientras que la tercera salida corresponde a la tercera entrada (bit objetivo), la cual se complementa si los dos bits de control están en 1, de lo contrario, no se modifica. Una implementación de dos qubits de una compuerta Toffoli es la CNOT, la cual tiene un bit de control y un bit objetivo; si el bit de control es 1, el bit objetivo es complementado, y si el bit de control es 0, el bit objetivo permanece en su estado original. Debido a que las compuertas CNOT y Toffoli son compuertas lógicas reversibles, estas pueden ser usadas en computación cuántica para realizar operaciones de cómputo clásico, como por ejemplo la exponenciación modular usada en el algoritmo cuántico de Shor. [6].

Compuertas cuánticas CNOT y Toffoli

La compuerta CNOT puede ser representada en forma matricial o circuital, tal como se muestra en la figura 3. En la representación circuital la línea horizontal con un punto corresponde al qubit de control, y la línea que contiene un círculo representa el qubit objetivo.

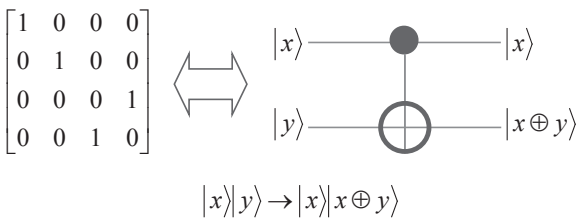

Figura 3 Representación matricial y circuital de la compuerta CNOT

La compuerta Toffoli de tres qubits se representa tal como se muestra en la Figura 4. En este caso, su matriz asociada es de 8x8 elementos lo cual evidencia el crecimiento exponencial de esta representación. En la parte inferior de la figura 4 se muestra en notación de Dirac la transformación que realiza esta compuerta sobre un sistema de tres qubits.

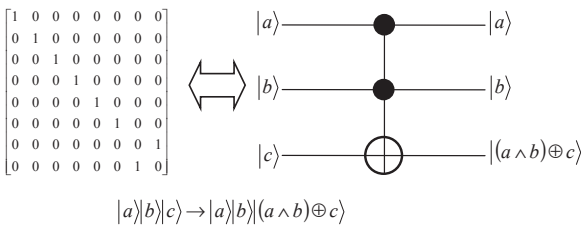

Figura 4 Representación matricial y circuital de la compuerta Toffoli de tres qubits

Compuertas Toffoli de n qubits

En computación cuántica cualquier operación booleana puede ser implementada a partir de compuertas Toffoli y CNOT [7]; sin embargo, para representar un circuito cuántico sencillo, generalmente se requieren muchas compuertas de este tipo. Por lo tanto, para reducir el número de compuertas es conveniente extender la función de la compuerta Toffoli para poder considerar más qubits de control y ubicarlos donde sea necesario. Por ejemplo, en la figura 5 se muestra una compuerta tipo Toffoli, que tiene cuatro qubits de control no contiguos y que hacen parte de un sistema de 6 qubits, en donde el sexto qubit es complementado si los qubits 1, 2, 4 y 5 están en el estado $|1\rangle$.

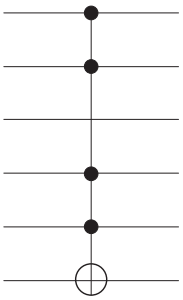

Figura 5 Representación circuital de una compuerta tipo Toffoli con 4 qubits de control, en un sistema de 6 qubits

Diseño del procesador

Para aplicar una compuerta cuántica usando la representación matricial se realiza el producto entre el vector de estado y la matriz asociada a dicha compuerta. Esto implica que cada compuerta del circuito tiene que estar almacenada en la memoria del emulador; sin embargo, como estas compuertas están representadas por matrices de 2^n x 2^n elementos, en donde n es el número de qubits del circuito, la cantidad de memoria requerida para almacenar un circuito cuántico es muy grande.

El *procesador para emular circuitos cuánticos basados en compuertas tipo Toffoli* usa una representación más simple de las compuertas cuánticas. En este caso, las compuertas cuánticas se representan por las operaciones que estas realizan sobre los qubits del sistema, es decir, no se considera el efecto que tienen sobre el vector de estado como en el caso de la representación matricial. Por ejemplo, para representar una compuerta Toffoli solo se necesita establecer la ubicación de los qubits de control y del qubit objetivo. Esto implica que la cantidad de memoria requerida para representar una compuerta cuántica crece linealmente en función del número de qubits del sistema y no exponencialmente.

El principio de funcionamiento del procesador consiste en aplicar las operaciones definidas por las compuertas del bloque de Transformación Unitaria a cada estado base del sistema generado en el bloque Hadamard.

Los estados base que se deben procesar dependen de la cantidad de qubits que son afectados por las compuertas Hadamard y por la ubicación de estas en el circuito. Adicionalmente, el procesador tiene en cuenta el intrincamiento, por lo tanto, al aplicar una operación de medición sobre el sistema, su estado colapsa tal como lo describe la mecánica cuántica. Esto se lleva a cabo mediante una subrutina que emula la medición a través de un proceso selectivo que se ejecuta al final del procesamiento de cada estado base.

El procesador de compuertas tipo Toffoli tiene tres bloques funcionales que corresponden a los mostrados en la figura 1: *H, U* y *M*. Los circuitos que se pueden emular con el procesador generalmente usan los tres bloques, pero pueden prescindir de los bloques H o M si el circuito no tiene compuertas Hadamard o no se requiere hacer una medición. El *Bloque Hadamard (H)* genera secuencialmente los estados base del sistema utilizando un contador, el cual se incrementa cada vez que un estado base se procesa. El número de estados base generados dependen del número de compuertas Hadamard del circuito cuántico. El *Bloque trasformación unitaria (U)* aplica las compuertas tipo Toffoli a cada estado base del sistema. Esto se realiza en 4 operaciones: identificación de los qubits de control, evaluación de los qubits de control, identificación del qubit objetivo y modificación del qubit objetivo.

En la figura 6 se muestra el circuito lógico que representa la ejecución de las dos primeras operaciones en un sistema de cuatro qubits; en la parte izquierda se muestra la compuerta que se desea aplicar, y en la parte derecha, las sub-operaciones requeridas. El estado inicial del sistema de cuatro qubits corresponde a *a*, *b*, *c* y *d*, y la identificación de los qubits de control se hace utilizando una máscara binaria (*máscara 1*).

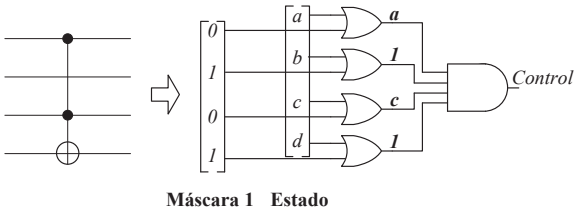

Figura 6 Circuito lógico que representa la ejecución de las dos primeras operaciones para aplicar una compuerta Toffoli

La primera operación consiste en realizar la función OR bit a bit entre la *máscara 1* y el estado inicial. El resultado es un "vector" en donde algunos de sus elementos tienen el mismo valor del estado inicial (aquellos asociados a los qubits de control), y los otros elementos toman el valor '1'.

La segunda operación consiste en realizar la función AND de los elementos del vector resultante. El resultado es una señal de un bit llamada *control*; si *control* es '1', entonces el estado inicial de todos los qubits de control es '1', y por lo tanto el qubit objetivo debe ser complementado. Si *control* es '0', entonces el estado de algunos de los qubits de control es '0' y por lo tanto el qubit objetivo no debe ser complementado.

En la figura 7 se muestra circuito lógico que representa la ejecución de las dos últimas operaciones. En este caso se utiliza la *máscara 2* para identificar cuáles son los qubits objetivos, aunque generalmente solo hay uno por compuerta. La primera operación consiste en realizar la función AND entre *control* y cada uno de los elementos de la *máscara 2*. Si la señal *control* es '1', se genera un vector que tiene un '1' en las posiciones que corresponden a los qubits objetivos, y un '0' en el resto de los elementos. Si la señal *control* es '0', se genera un vector en donde todos sus elementos son '0'. La segunda operación consiste en realizar la función XOR bit a bit entre el vector previamente generado y el vector de estado [*a b c d*]. El resultado es el mismo vector de estado, pero los elementos cuya posición corresponde a los qubits objetivos son complementados si la señal *control* es '1'.

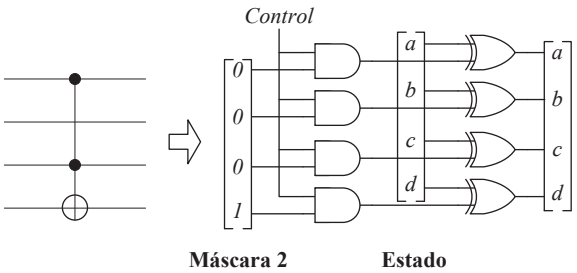

Figura 7 Circuito lógico que representa la ejecución de las dos últimas operaciones para aplicar una compuerta Toffoli

El *Bloque medición (M)* emula el proceso de medición u observación en el sistema cuántico. Cuando un sistema cuántico es medido su estado colapsa a un subconjunto de los estados base en los que el sistema se encontraba en superposición antes de ser observado. Si la medición se realiza sobre todos los qubits del sistema, entonces el sistema colapsa a un solo estado base, el observado; Si la medición es parcial, es decir sobre algunos de los qubits del sistema, el estado de estos qubits también colapsa al estado base observado, pero el sub-sistema conformado por los qubits que no han sido medidos colapsa a una superposición cuyos estados base son determinados por el valor observado.

Para emular la medición se proponen dos alternativas. La primera consiste en procesar todos los estados base y almacenar todos los resultados en memoria; después se comparan los resultados con el valor observado, y solo se siguen procesando los que coincidan. El valor observado se obtiene escogiendo aleatoriamente uno de los resultados almacenados en memoria. Esta alternativa no es práctica debido a que se requiere 2^n posiciones de memoria para almacenar todos los resultados que se generan en el procesamiento de los estados base y por lo tanto esta alternativa no se implementó.

La segunda alternativa consiste en procesar los estados base y almacenar solo los resultados

que coincidan con el valor observado, es decir, solo se almacenan los estados en los que el sistema colapsa. En este caso, se debe establecer previamente el valor observado. Este valor se puede determinar en el "proceso de compilación" de la estructura del circuito cuántico que se desea emular. En este proceso, conociendo el resultado que el circuito debe generar, al menos para un estado base, se calcula uno de los posibles resultados (preferiblemente con parámetros aleatorios), el cual es considerado como valor observado o "Medición". Por ejemplo en el algoritmo de Shor para la factorización, este valor es el resultado de realizar la exponenciación modular con parámetros N y *x* para un exponente seleccionado aleatoriamente.

El circuito lógico que representa el proceso de medición se muestra en la figura 8. El procesador utiliza la *máscara M* para establecer cuáles qubits van a ser observados, y el vector *Medición* para establecer cuál es el valor observado. El vector *Medición* contiene la representación binaria del valor observado en las posiciones equivalentes a los qubits que van a ser observados, y tiene '0's en las demás posiciones, en este caso, el valor observado es "01".

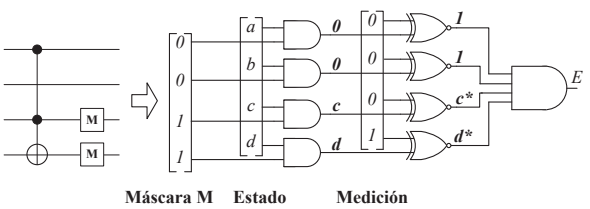

Figura 8 Circuito lógico que representa la ejecución del proceso de medición

En el proceso de medición, para establecer si un estado se debe almacenar en memoria se utilizan dos operaciones: La primera operación es la función AND entre los elementos de la *mascara M* y el vector de estado, esto genera un vector que solo tiene el estado de los qubits que van a ser medidos, el resto de sus elementos son '0'. La segunda operación es una comparación entre los elementos del vector previamente generado y los elementos del vector *Medición*, esta operación se realiza con la función XNOR y una compuerta AND. La XNOR genera '1's en las posiciones en donde exista coincidencia y '0's en el caso contrario. De esta manera, si el estado de los qubits medidos coincide con el valor establecido como medición, se genera un vector de solo unos, lo cual hace que la salida de la AND sea '1' indicando que el estado procesado debe ser almacenado en memoria.

Consideraciones de diseño del procesador

El procesador de compuertas tipo Toffoli utiliza una unidad lógica en donde el número de bits de sus operandos equivale al número de qubits del sistema que se desea emular, lo cual permite que cada estado base sea procesado en un solo ciclo de ejecución. La unidad lógica procesa dos tipos de datos: la representación de los estados base y la representación de las compuertas cuánticas. Los estados son la concatenación de tres campos: la salida del contador del bloque H, un identificador de unidad (ID) que permite distribuir la carga computacional entre varias unidades de procesamiento; y un estado inicial que corresponde al estado inicial de los qubits que no hacen parte de la superposición (los qubits de trabajo). Las máscaras que representan a las compuertas cuánticas son almacenadas en memoria formando una secuencia de datos tal como si fuera una secuencia de instrucciones en un procesador de propósito general. Cada estado corresponde a un dato de entrada para la unidad lógica el cual se procesa aplicando la secuencia de compuertas hasta obtener el estado final, al cual se le aplica la subrutina de medición, de ser necesario

Representación de las compuertas cuánticas

Cada compuerta está representada por dos máscaras; la *máscara 1*, identifica los qubits de control; y la *máscara 2* identifica los qubits objetivos. Las máscaras son registros cuyo tamaño es igual al número de qubits del sistema.

La *Máscara 1* tiene todos los bits en '1' excepto los que corresponden a los qubits de control. La *Máscara 2* tiene todos los bits en '0' excepto los que corresponden a los qubits objetivo.

La figura 9 muestra la representación de una compuerta tipo Toffoli de cuatro qubits, la cual tiene dos qubits de control, un qubit objetivo, y un qubit que no realiza ninguna función. La *Máscara 1* tiene el patrón [0 1 0 1] debido a que el primer y tercer qubit son qubits de control; la *Máscara 2* tiene el patrón [0 0 0 1] debido a que el cuarto qubit es el qubit objetivo.

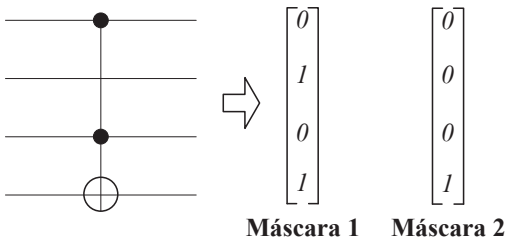

Figura 9 Representación de una compuerta tipo Toffoli mediante dos máscaras

Arquitectura del procesador de compuertas tipo Toffoli

En la figura 10 se muestra el diagrama de bloques del procesador. Los bloques principales son: la unidad de procesamiento, la unidad de control, la memoria de compuertas y la memoria de estados. La unidad de control incluye la maquina de estados (FSM) y dos contadores.

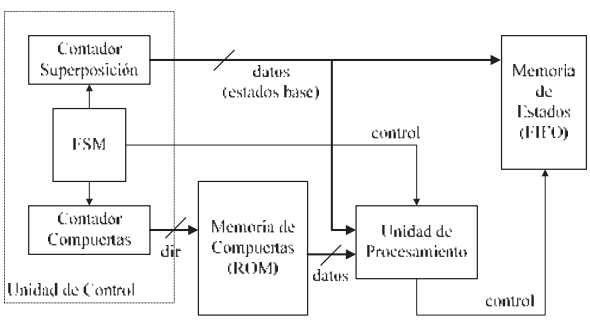

Figura 10 Diagrama de bloques del procesador de compuertas tipo Toffoli

En la figura 11 se muestra un diagrama más detallado de la *Unidad de procesamiento*; su componente principal es la unidad lógica que procesa las compuertas y los estados base del sistema. También contiene los registros de entrada R1 y R2 y los flip-flops C y E usados para almacenar señales de control.

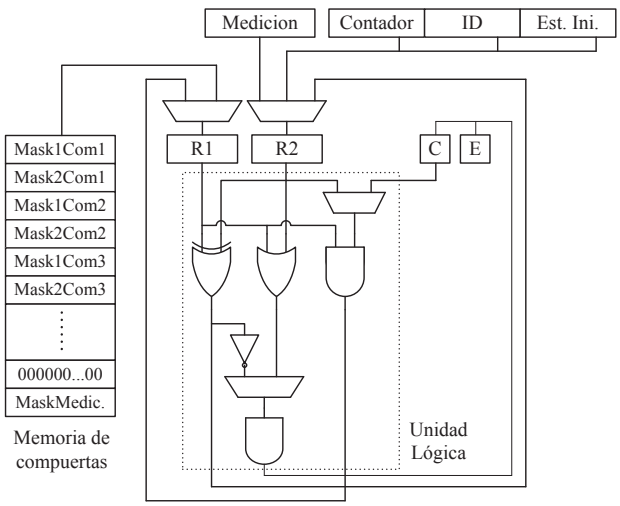

Figura 11 Unidad de procesamiento del procesador de compuertas tipo Toffoli

La unidad lógica ejecuta en un solo ciclo de reloj las operaciones básicas que se realizan entre sus entradas, las mascaras y los estados base, y las operaciones para emular el proceso de medición. Estas operaciones corresponden a las que se muestran en las figuras 6, 7 y 8 (XOR, XNOR, OR y AND). Para lograr esto, la mayoría de compuertas de la unidad lógica tienen dos entradas de *n* bits y una salida de *n* bits y las operaciones se realizan bit a bit entre ambas entradas. La compuerta AND de *n* bits genera un bit que es usado para complementar el qubit objetivo en las compuertas tipo Toffoli (FF-C), o como señal de control para cargar un estado base en la *Memoria de Estados* (FF-E). Cuando la señal es usada para complementar y su estado es '1', indica que todos los qubits de control son '1' y por lo tanto el qubit objetivo debe complementarse. Cuando la señal es usada como señal de control y su estado es '1', indica que el resultado obtenido al procesar un estado base coincide con el valor medido, y por

lo tanto este debe ser almacenado en la memoria que contiene los estados en los que el sistema colapsa después de ser medido.

La *memoria de compuertas* es equivalente a la memoria de programa en un procesador de propósito general; en esta se almacenan las máscaras de la secuencia de compuertas que forman el circuito cuántico, y por lo tanto contiene el "programa" que debe ejecutar el procesador. Cada compuerta requiere dos posiciones de memoria para las *máscaras 1* y *2*. Después de la secuencia de compuertas se almacena una palabra de '0's (para indicar que no hay más compuertas para procesar) seguido de la máscara *Medición*.

La *Memoria de estados* almacena los estados base en los que el sistema colapsa después de emular el proceso de medición. La memoria es tipo FIFO ya que sus datos son extraídos en secuencia sin necesidad de acceder a ellos de forma aleatoria. Estos datos corresponden a estados base usados posteriormente por otro circuito cuántico.

La *unidad de control* se compone de una FSM, un detector de cero y dos contadores. La FSM se encarga de generar las señales de control para cada componente del procesador. El detector de cero está conectado al bus de datos de la memoria de compuertas y sirve para detectar el final de la secuencia de compuertas lo cual permite iniciar la rutina que emula la medición.

El *Contador Compuertas* genera las direcciones para la memoria de compuertas. El *Contador Superposición* genera los estados base que conforman la superposición, y es incrementado al final de cada rutina de procesamiento para generar el siguiente estado base. La concatenación de la salida de este contador, el estado inicial y el identificador de la unidad de procesamiento (ID) forman el estado que se debe procesar.

En la figura 12 se muestra el diagrama de flujo de la FSM. En el estado *S1* se inicializan los registros; desde el estado *S2* hasta el *S5* se procesan todas las compuertas que conforman la secuencia del circuito cuántico; del estado *S6* al *S8* se ejecuta la rutina que emula la medición; si después de finalizar esta rutina todavía hay estados por procesar se retorna a S2 para cargar de nuevo la *máscara 1* de la primera compuerta.

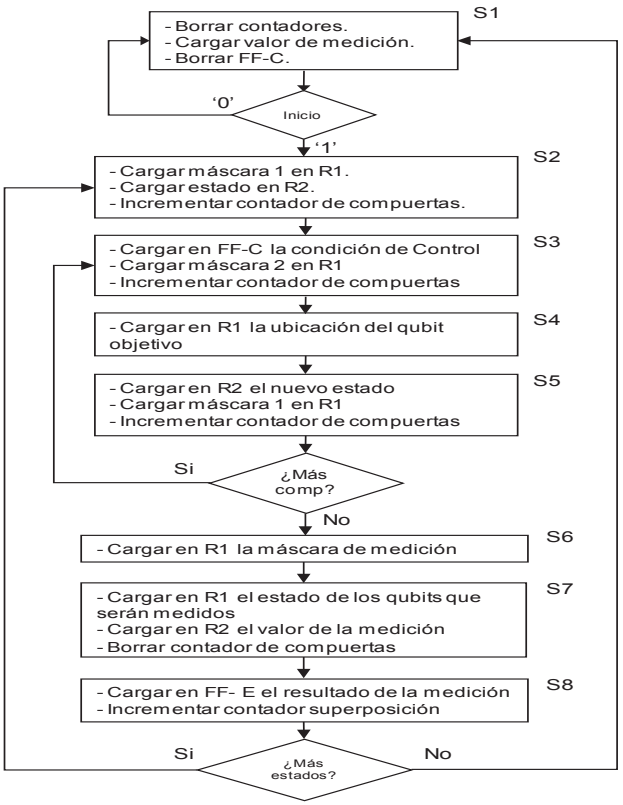

Figura 12 Diagrama de flujo de la rutina implementada por la unidad de control del procesador

Arreglo de unidades de procesamiento

En un circuito cuántico la cantidad de estados base del sistema crece exponencialmente en función del número de qubits, por lo tanto, su emulación requiere gran capacidad de cómputo lo que se logra usando un arreglo de unidades de procesamiento. En este caso es necesario distribuir de manera adecuada los estados base entre las unidades de procesamiento, para lo cual se utiliza un identificador y un contador para cada unidad.

La figura 13 ilustra la representación de un estado base. El estado está formado por la concatenación de tres campos: salida del contador

de superposición (contador), identificador (ID) y estado inicial. El contador genera números enteros desde 0 hasta 2^k-1, en donde $k = n - log_2 P$, n es el número de qubits que hacen parte de la superposición de estados, y P es el número de unidades de procesamiento. Este contador genera secuencialmente los estados base del sistema a medida que se van procesando.

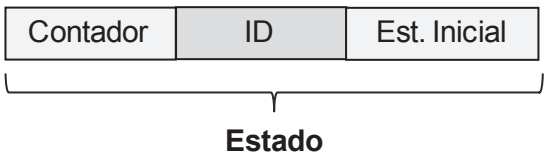

Figura 13 Campos que forman el estado del sistema

A cada unidad se le asigna un ID la cual solo puede procesar los estados que tengan dicho ID, así por ejemplo, si el campo ID es solo de dos bits y la unidad corresponde al ID=3, esta solo procesará los estados del sistema cuya representación binaria sea '11' en el campo ID. Esto evidencia que el rango asignado a cada unidad no es continuo, pero garantiza una distribución adecuada de la información.

El campo *Estado Inicial* siempre es el mismo para cada unidad de procesamiento, su contenido debe reflejar el estado inicial de los qubits del sistema que no fueron superpuestos (qubits de trabajo, a los que no se les aplica compuertas Hadamard). Por ejemplo, en el algoritmo de Shor, para factorizar un número de N bits se deben utilizar al menos N qubits de trabajo que en conjunto forman el estado inicial |000...001>. Estos qubits deben incluirse ya que sobre estos se ejecuta la exponenciación modular requerida en el algoritmo.

Resultados de síntesis y simulación

El procesador propuesto en este trabajo y el arreglo de 64 procesadores son sintetizados en el FPGA EP3C120F780I7. Tanto el procesador como el arreglo de procesadores permiten llevar a cabo la emulación de un sistema de 52 qubits de los cuales 32 se consideran para generar la superposición equiprobable de estados y 20 qubits son de trabajo. Los resultados de la síntesis se presentan en la tabla 1.

Tabla 1 Resultados de síntesis del procesador de compuertas tipo Toffoli y el arreglo de procesadores para un sistema de 52 qubits

Diseño	*Elementos Lógicos*	*Registros*	*% FPGA*	*Frecuencia*
Procesador de compuertas tipo Toffoli	343	153	<1%	138,73 MHz
Arreglo de 64 Procesadores	25.437	11.508	21%	136,04 MHz

Según se observa en la tabla 1, al integrar 64 procesadores, el porcentaje de utilización de recursos del FPGA fue inferior al 25% presentando un crecimiento casi lineal, lo que permite estimar que se podría llegar a integrar hasta 256 unidades de procesamiento para 52 qubits.

Para estimar el tiempo que se requiere para ejecutar un circuito cuántico en el arreglo de procesadores se debe tener en cuenta que la ejecución de cada compuerta cuántica toma 4 ciclos de reloj, y que el proceso de medición toma 3 ciclos de reloj por cada estado. De esta manera, el número de ciclos de reloj requeridos para procesar un circuito completo puede ser calculado usando la ecuación (1).

$$C = \frac{N_E\left(4N_C + 3\right)}{N_P} \qquad (1)$$

En donde N_E es el número de estados base que deben ser procesados, N_C es el número de compuertas del circuito cuántico y N_P es el número de unidades de procesamiento utilizadas, el cual debe ser una potencia de 2.

El funcionamiento del arreglo de procesadores ha sido verificado emulando circuitos cuánticos simples como el *Quantum Full Adder* (QFA), cuya representación circuital se muestra en la figura 14. En este circuito los qubits q3, q2 y q1 corresponden a las entradas del sumador a los cuales se les aplica compuertas Hadamard para generar una superposición de estados base que representan todas las combinaciones posibles de entrada del circuito sumador. El qubit q0 es un qubit de trabajo cuyo estado inicial debe ser $|0\rangle$ para que el circuito funcione correctamente. El resultado de la suma queda en el qubit q_1 y el acarreo en q_0; estos qubits son medidos en la salida (al final del proceso) considerando como valor medido "10", es decir S=1 y C=0.

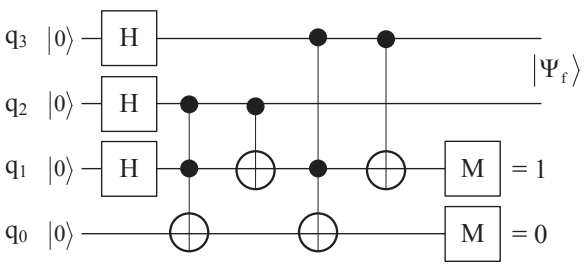

Figura 14 Circuito cuántico de un QFA

El resultado de la emulación debe ser el conjunto de estados base que forman $|\Psi_f\rangle$, el cual representa el estado del sub-sistema q_3q_2 luego de hacer la medición en q_1 y q_0. Teniendo en cuenta la tabla de verdad del QFA mostrada en la tabla 2, el resultado de la emulación debe ser el conjunto de estados base $|\Psi_f\rangle = |00\rangle + |01\rangle + |10\rangle$ que corresponden a $q_3^+q_2^+$ cuando $q_1^+ = 1$ y $q_0^+ = 0$ (S=1 y C=0).

En la figura 15 se muestra el resultado de la emulación del QFA utilizando 4 unidades de procesamiento. En el flanco de subida de la señal *empty* se realiza la transferencia de un estado base desde el vector *estados* hacia la salida del procesador. Como se puede ver, los tres estados transferidos son 2, 0 y 1 ($|10\rangle, |00\rangle, |01\rangle$) tal como se describió anteriormente. En estos flancos también se observa el ID (vector contM) de la unidad de procesamiento que generó cada estado.

Tabla 2 Tabla de verdad del QFA

q_3	q_2	q_1	q_0	q_3^+	q_2^+	q_1^+ (S)	q_0^+ (C)
0	0	0	0	0	0	0	0
0	0	1	0	**0**	**0**	1	0
0	1	0	0	**0**	**1**	1	0
0	1	1	0	0	1	0	1
1	0	0	0	**1**	**0**	1	0
1	0	1	0	1	0	0	0
1	1	0	0	1	1	0	1
1	1	1	0	1	1	1	1

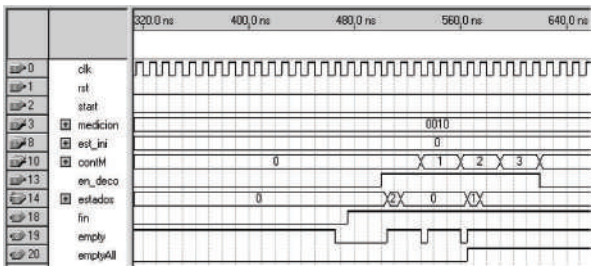

Figura 15 Resultado de la emulación del QFA usando 4 unidades de procesamiento

Para poder emular circuitos de mayor complejidad es necesario desarrollar una herramienta software que permita generar a través de una interfaz de usuario la secuencia de compuertas del circuito cuántico. Por ejemplo, la exponenciación modular para el algoritmo de Shor, cuya transformación unitaria para estados base se representa en la notación de Dirac como se muestra en la ecuación (2).

$$|a\rangle|b\rangle \rightarrow |a\rangle|x^b \bmod N\rangle \qquad (2)$$

donde x y N son parámetros que hacen parte de la estructura del circuito.

Según los resultados presentados en [7], este circuito requiere 5L+8 qubits y aproximadamente

$240L^3+484L^2+182L$ compuertas tipo Toffoli para emular la factorización de un número de L bits. Así por ejemplo, para emular el algoritmo de Shor para factorizar un número de 8 bits, el circuito cuántico para la exponenciación modular requiere 48 qubits, de los cuales 17 (2L+1) deben ponerse en superposición y los 31 restantes son qubits de trabajo. El número de estados que se deben procesar es 2^{17}, es decir, 131.072 estados y se requieren 155.312 compuertas tipo Toffoli; entonces, basándose en la ecuación (1) y en la frecuencia máxima de operación obtenida en la síntesis (Tabla 1), la ejecución de este circuito tardaría aproximadamente 9,35 segundos.

Conclusiones y trabajo futuro

Este trabajo presenta el diseño de un procesador para emular circuitos cuánticos basados en compuertas tipo Toffoli, el cual puede ser usado como un bloque funcional para la simulación de algoritmos cuánticos orientados a resolver problemas de alta complejidad computacional. Este diseño permite procesar un gran número de qubits ya que no usa la representación matricial sino que está basado en el procesamiento de estados base, es decir que cada estado base es procesado de manera independiente ejecutando las operaciones de las compuertas cuánticas sobre cada qubit del sistema. Adicionalmente, la descripción del hardware es parametrizable y permite adaptar el diseño a cualquier cantidad de qubits.

La cantidad de estados base que deben ser procesados crece exponencialmente en función del número de qubits; sin embargo, la representación utilizada, la alta frecuencia de operación y la implementación de un arreglo de unidades de procesamiento hacen que el tiempo de ejecución sea mucho menor con respecto a implementaciones basadas en la representación matricial.

Los resultados de la síntesis de un arreglo de 64 unidades de procesamiento muestran que es posible emular un sistema cuántico de 52 qubits, el cual puede ser usado para la factorización de un número de 8 bits. En este caso se requiere el procesamiento de 155,312 compuertas cuánticas y 131,072 estados con un tiempo de ejecución de 9,35 segundos. Estos resultados también permiten estimar que se pueden implementar hasta 256 unidades de 52 qubits en el FPGA EP3C120F780I7, el cual es de mediana capacidad y desempeño.

Como trabajo futuro se propone la implementación del software que permita generar automáticamente la secuencia de compuertas de un circuito cuántico a través de su representación circuital usando una interfaz grafica. Esto permitirá emular circuitos de mayor cantidad de qubits y complejidad con respecto a los reportados en la literatura.

Referencias

1. U. Khalid, Z. Zilic, K. Radecka. *FPGA Emulation of Quantum Circuits*. ICCD, 2004 IEEE International Conference on Computer Design. San Jose, California, USA. 2004. pp. 310-315.
2. M. Fujishima. *FPGA-based High-speed Emulator of Quantum Computing*. ICFPT, 2003 IEEE International Conference on Field-Programmable Technology. Tokyo, Japan. 2003. pp. 21-26.
3. M. Aminian, M. Saeedi, M. Zamani, M. Sedighi. *FPGA-Based Circuit Model Emulation of Quantum Algorithms*. IEEE Computer Society Annual Symposium on VLSI. Montpellier, Francia. 2008. pp. 399-404.
4. K. Aggour, R. Mattheyses, J. Shultz. *Efficient Quantum Computing Simulation Through Dynamic Matrix Restructuring and Distributed Evaluation*. IEEE International Conference on Cluster Computing (Cluster 2007). Austin, Texas, USA. 2007. pp. 1-10.
5. G. Arnold, T. Lippert, N. Pomplun, M. Richter. *Large Scale Simulation of Ideal Quantum Computers on SMP-Clusters*. Ed. John von Neumann Institute for Computing. Julich, Alemania. 2006. pp. 447-454.
6. M. Nielsen, I. Chuang. *Quantum Computation and Quantum Information*. 7th ed. Ed. Cambridge University Press. Cambridge, United Kingdom. 2010. pp. 216-242.
7. R. Perry. *The Temple Of Quantum Computing Version 1.1*. Available on: http://www.toqc.com/TOQCv1_1.pdf. Accessed: April 27 2011.