Universidad Nacional de La Plata

Facultad de Informática

Memorias matriciales correlacionadas cuánticas, simples y mejoradas: una propuesta para su estudio y simulación sobre GPGPU.

Trabajo de Tesis para optar al Doctorado en Ciencias Informáticas de la Facultad de Informática de la Universidad Nacional de La Plata

Candidato: *Mario Mastriani*

Director: *Dr. Marcelo Naiouf*

Facultad de Informática, Universidad Nacional de La Plata, Septiembre de 2014.

A ...

Agradecimientos

Mi agradecimiento y disculpas para Paola y Agustín por sacar de nuestro tiempo en común para hacer este trabajo.

Quiero agradecer –muy especialmente– al Lic. Daniel Gibrán Mendoza Vázquez, de la Facultad de Ciencias de la Universidad Nacional Autónoma de México, por su apoyo tan desinteresado a afectos de confeccionar los dos primeros capítulos de la presente tesis, lo cual me ha permitido ahorrar infinito trabajo de compilación bibliográfica.

Memorias matriciales correlacionadas cuánticas, simples y mejoradas: una propuesta para su estudio y simulación sobre GPGPU.

Resumen

En este trabajo se desarrollan - en orden - los fundamentos de la Física Cuántica, y de la Computación Cuántica, una noción completa de las arquitecturas multicapa tolerante a fallos para la implementación física de una computadora cuántica, para completar los primeros cuatro capítulos con las técnicas propias para la simulación de este nuevo paradigma sobre placas multicore del tipo General-Purpose Computing on Graphics Processing Units (GPGPU). La segunda parte de este trabajo consiste en los tres capítulos inmediatamente siguientes, los cuales suman 10 innovaciones en este campo, a saber:

1. *el Proceso de Ortogonalización Booleano (POB) con su inversa,*

2. *el Proceso de Ortogonalización de Gram-Schmidt Mejorado (POGSMe) con su inversa,*

3. *el Proceso de Ortogonalización Cuántico (POCu) con su inversa,*

4. *la Red Ortogonalizadora Booleana Sistólica (ROBS),*

5. *la Red Ortogonalizadora Cuántica Sistólica (ROCS), y*

6. *una métrica que llamamos Tasa Dimensional de Entrada-Salida (TDES) la cual fue creada para monitorear el impacto del mejorador para la estabilidad del Proceso Ortogonalizador de Gram-Schmidt en el costo computacional final.*

7. *una versión mejorada de las ya conocidas Memorias Matriciales Correlacionadas Booleanas (MMCB), es decir, la MMCB mejorada (MMCBMe) en base al innovador Proceso de Ortonormalización Booleano (POB) del Capítulo 5,*

8. *la Memoria Matricial Correlacionada Cuántica (MMCCu), y*

9. *la MMCCu Mejorada (MMCCuMe) en base al Proceso de Ortogonalización Cuántico (POCu) implementado en forma sistólica y conocida como la Red Ortogonalizadora Cuántica Sistólica (ROCS) del Capítulo 5.*

10. *el Capítulo 7, el cual contiene las simulaciones computacionales, las cuales verifican fehacientemente la mejora en la performance de almacenamiento como resultado de aplicar el POCu a las MMCCu, así como una serie de simulaciones relativas a arreglos uni, bi y tridimensionales, los cuales representan señales, imágenes (multimediales, documentales, satelitales, biométricas, etc.) y video o bien imágenes multi e hiper-espectrales satelitales, tomografías o resonancias magnéticas seriadas, respectivamente. Dichas simulaciones tienen por objeto verificar los atributos de ortogonalización de los algoritmos desarrollados. Dado que es la primera vez que en la literatura se realizan este tipo de simulaciones en GPGPU para esta tecnología, el Capítulo 7 representa en si mismo el décimo aporte del presente trabajo a esta área del conocimiento.*

Un último capítulo reservado a conclusiones parciales por capítulo y generales del trabajo como un todo.

Simple and improved quantum correlation matrix memories: a proposal for their study and simulation on GPGPU.

Abstract

In this work we develop - in order - the basics of Quantum Physics and Quantum Computing, a complete notion of fault tolerant multilayer architectures for physical implementation of a quantum computer, to complete the first four chapters with own techniques for simulation of this new paradigm on the multicore card of General-Purpose Computing on Graphics Processing Units (GPGPU) type. The second part of this work consists of the three chapters immediately following, which together add 10 innovations in this field, namely:

1. Boolean Orthogonalization Process (BOP) with its inverse

2. Enhanced Gram-Schmidt Orthogonalization Process (EGSOP) with its inverse

3. Quantum Orthogonalization Process (QOP) with its inverse

4. Systolic Boolean Ortogonalized Network (SBON)

5. Systolic Quantum Ortogonalized Network (SQON), and

6. a metric, we call Dimensional Input-Output Rate (DIOR) which was created to monitor the impact of enhancer for the stability of Gram-Schmidt Ortogonalizador Process in the final computational cost.

7. an improved version of the already known Boolean Correlation Matrix Memories (BCMM), i.e., the enhanced BCMM (EBCMM) based on the innovative Boolean Ortonormalización Process (BOP) of the previous chapter,

8. Quantum Correlation Matrix Memory (QCMM), and

9. the Enhanced QCMM (EQCMM) based on Quantum Orthogonalization Process (QOP) implemented in a systholic way, and known as the Systholic Quantum Ortogonalized Network (SQON) in the previous chapter.

10. Chapter 7, which contains the computer simulations, which reliably verify the performance improvement in storage as a result of applying QOP to QCMM, and a series of simulations on uni, bi, and three-dimensional arrays, representing signal, image (multimedia, documentaries, satellite, biometrics, etc..) and video or multi and hyper-spectral satellite imagery, Computed Tomography (CT) or Magnetic Resonance Imaging (MRI) images, respectively. These simulations are designed to verify the attributes of orthogonalization of the developed algorithms. Since it is the first time in the literature such simulations are performed on GPGPU for this technology, Chapter 7 represents in itself the tenth contribution of this work to this area of knowledge.

A final chapter by chapter reserved for partial and general conclusions of the work as a whole.

Publicaciones previas del tesista en el tema de la tesis:

[1] **M. Mastriani**, *"Enhanced Boolean Correlation Matrix Memory",* (RNL02), X RPIC Reunión de Trabajo en Procesamiento de la Información y Control, 8 al 10 de Octubre 2003, San Nicolás, Buenos Aires, Argentina, vol. 2, pp. 470-473. Disponible en: http://www.frsn.utn.edu/xrpic/trabajos.html
http://www.frsn.utn.edu/xrpic/listado.doc

[2] **M. Mastriani**, "*Systholic Boolean Orthonormalizer Network in Wavelet Domain for Microarray Denoising*," International Journal of Signal Processing, Volume 2, Number 4, pp.273-284, 2005.

[3] **M. Mastriani**, "*Denoising and Compression in Wavelet Domain via Projection onto Approximation Coefficients*," International Journal of Signal Processing, Volume 5, Number 1, pp.20-30, 2008.

Publicaciones del tesista durante la tesis:

Ya publicadas:

[4] **M. Mastriani, M. Naiouf**, "*Enhanced Gram-Schmidt Process for Improving the Stability in Signal and Image Processing"*, International Journal of Mathematical Sciences, WASET, vol. 7, Nro. 4, 2013, pp.7-11.

[5] **M. Mastriani, M. Naiouf**, "*Quantum Enhanced Correlation Matrix Memories via States Orthogonalisation*", International Journal of Electronics Science and Engineering, WASET, vol. 7, Nro. 8, 2013, pp.1131-1136.

En preparación:

[6] **M. Mastriani, M. Naiouf**, "*Systholic Boolean Orthonormalizer Network in Wavelet Domain for SAR Image Despeckling"*, for WSEAS Transactions on Signal Processing.

[7] **M. Mastriani, M. Naiouf**, "*New Tool for Boolean Signal Processing"*, for WSEAS Transactions on Signal Processing.

[8] **M. Mastriani, M. Naiouf**, "*Enhanced Boolean Correlation Matrix Memory"*, for WSEAS Transactions on Signal Processing.

[9] **M. Mastriani, M. Naiouf**, "*Fast Cosine Transform to increase speed-up and efficiency of Karhunen-Loève Transform for lossy compression of SAR imagery"*, for WSEAS Transactions on Signal Processing.

Indice General

Página

Indice de Tablas

Página

Indice de Figuras

Página

Prólogo

*La **computación cuántica** es un paradigma de computación distinto al de la computación clásica. Se basa en el uso de cubits en lugar de bits, y da lugar a nuevas puertas lógicas que hacen posibles nuevos algoritmos.*

Una misma tarea puede tener diferente complejidad en computación clásica y en computación cuántica, lo que ha dado lugar a una gran expectación, ya que algunos problemas intratables pasan a ser tratables. Mientras que un computador clásico equivale a una máquina de Turing, un computador cuántico equivale a una máquina de Turing cuántica.

En este trabajo se presentan 10 (diez) innovaciones a esta nueva disciplina conocida en la actualidad como Computación Cuántica. Dichas innovaciones se encuentran entre los Capítulos 5 y 7. No obstante, y al solo efecto de hacer al presente trabajo autocontenido se han incorporado cuatro capítulos iniciales, los cuales permitirán introducir en forma completa a:

- *Física Cuántica (Capítulo 1),*
- *Computación Cuántica (Capítulo 2),*
- *Implementación Física de la Computación Cuántica a través de arquitecturas de 5 capas tolerante a fallos (Capítulo 3), y*
- *Simulación Computacional de Algoritmos Cuánticos sobre placas multicore del tipo General-Purpose Computing on Graphics Processing Units (GPGPU) de varios cubits (Capítulo 4).*

El trabajo se completa con los capítulos reservados a las innovaciones, es decir,

- *Ortogonalizadores Booleanos y Cuánticos con sus respectivas inversas en su más pura forma algorítmica, o bien, en forma de redes escalonadas para futuras aplicaciones en Procesamiento Cuántico de Señales e Imágenes (Capítulo 5),*
- *Los tres tipos de Memorias Matriciales Correlacionadas Clásicas, Booleanas y Cuánticas, simples y mejoradas (Capítulo 6), estas últimas a través de los ortogonalizadores desarrollados en el capítulo anterior,*
- *Las simulaciones computacionales (Capítulo 7) las cuales constituyen las primeras de su tipo en este campo.*

En concreto, los 10 aportes son:

Capítulo 5:

1. el Proceso de Ortogonalización Booleano (POB) con su inversa,

2. el Proceso de Ortogonalización de Gram-Schmidt Mejorado (POGSMe) con su inversa,

3. el Proceso de Ortogonalización Cuántico (POCu) con su inversa,

4. la Red Ortogonalizadora Booleana Sistólica (ROBS),

5. la Red Ortogonalizadora Cuántica Sistólica (ROCS), y

6. una métrica que llamamos Tasa Dimensional de Entrada-Salida (TDES) la cual fue creada para monitorear el impacto del mejorador de la estabilidad del Proceso Ortogonalizador de Gram-Schmidt en el costo computacional final.

Capítulo 6:

7. una versión mejorada de las ya conocidas Memorias Matriciales Correlacionadas Boolenas (MMCB), es decir, la MMCB mejorada (MMCBMe) en base al innovador Proceso de Ortonormalización Booleano (POB) del capítulo anterior,

8. la Memoria Matricial Correlacionada Cuántica (MMCCu), y

9. la MMCCu Mejorada (MMCCuMe) en base al Proceso de Ortogonalización Cuántico (POCu) implementado en forma escalonada y conocida como la Red Ortogonalizadora Cuántica Sistólica (ROCS) en el capítulo anterior.

Capítulo 7:

10. las simulaciones computacionales, las cuales verifican inequívocamente lo desarrollado en el Capítulo 6, representan en si mismas -por su alcance- otra innovación dado que es la primera vez que se emplea esta tecnología simulada en GPGPU para representar Memorias Matriciales Correlacionadas Cuánticas y aplicaciones concretas en Procesamiento Cuántico de Imágenes.

Finalmente el trabajo se completa con

• las conclusiones (Capítulo 8),

• el glosario con los acrónimos,

• la bibliografía, y

• la sinopsis curricular del doctorando.

Capítulo 1: Física Cuántica

Fundamentos de la Mecánica Cuántica

En esta sección introduciremos las bases matemáticas fundamentales para trabajar con la Teoría de Información Cuántica [1]. Trabajaremos sobre espacios vectoriales discretos, debido a que los sistemas cuánticos que manejaremos (computadora cuántica) son sistemas físicos discretos, no continuos. Se dará una breve introducción a la notación de Dirac y su aplicación en la mecánica cuántica. Una vez establecidas las bases matemáticas se describirán los Postulados de la Mecánica Cuántica: descripción del estado de un sistema, descripción de cantidades físicas, medición de cantidades físicas, reducción del paquete de ondas, evolución temporal, postulado de simetrización y variables de espín. Finalmente se definirá el enredamiento cuántico, su importancia en el cómputo cuántico y los sistemas de dos niveles.

1.1 Antecedentes

Delta de Kronecker

La delta de Kronecker, $\delta_{n,m}$ es un símbolo que representa dos posibles valores, dependiendo de sus índices,

$$\delta_{n,m}\begin{cases}1 & ,\boldsymbol{n}=\boldsymbol{m}\\ 0 & ,\boldsymbol{n}\neq\boldsymbol{m}\end{cases}$$

Dado que el símbolo sólo es diferente de cero cuando sus índices son iguales, las sumas que incluyen la delta de Kronecker pueden ser simplificadas fácilmente

$$\sum_{\boldsymbol{m}}\delta_{\boldsymbol{mn}}\boldsymbol{B}_{\boldsymbol{m}}=0\cdot\boldsymbol{B}_1+0\cdot\boldsymbol{B}_2+\ldots+1\cdot\boldsymbol{B}_{\boldsymbol{n}}+\ldots=\boldsymbol{B}_{\boldsymbol{n}}$$

Notación de Dirac

En 1930 en el libro *Principios de la Mecánica Cuántica*, Paul Dirac introdujo una poderosa notación para poder describir estados cuánticos y funciones lineales, también conocida como notación Bra-Ket. Con la notación de Dirac podemos representar un estado base de n elementos con una cadena binaria de longitud n, mientras que con la representación de vectores columna necesitaríamos 2^n componentes para definir el mismo valor.

Ket

Asociamos a cada función de onda un vector (o ket) en un espacio vectorial lineal ξ.

$$\psi\rightarrow|\psi\rangle$$

tal que la información cuántica del sistema se almacena en sí mismo. La información de este

vector ket puede verse como un vector columna. Un ket nos ayuda a definir el estado de un sistema físico.

Si tenemos dos estados cuánticos distintos $|\alpha_1\rangle$ y $|\alpha_2\rangle$ entonces el ket

$$|\psi\rangle = \boldsymbol{c}_1|\alpha_1\rangle + \boldsymbol{c}_2|\alpha_2\rangle \tag{1.1}$$

donde c_i es un número complejo, es también un posible estado del sistema.

Bra

Dirac definió un vector bra como una función lineal que asocia un número complejo a todo ket de ξ mediante el producto escalar. Asumimos que para cada ket $|\alpha\rangle$ existe un bra, y lo denotamos como $\langle\alpha|$. El bra es llamado el dual del vector ket. Entonces se puede referir al vector bra $\langle\alpha|$ como un vector renglón, permitiendo así el producto de un vector renglón con un vector columna $\langle\alpha|\beta\rangle$.

El bra asociado al ket $|\psi\rangle$ definido en la ecuación (1.1) está dado por

$$\langle\gamma| = \boldsymbol{c}_1^*\langle\alpha_1| + \boldsymbol{c}_2^*\langle\alpha_2| \tag{1.2}$$

La norma del ket es real, no negativo, y se define como: $\||\psi\rangle\| \equiv \sqrt{\langle\psi|\psi\rangle} \geq 0$, donde la norma es igual a cero únicamente si el ket $|\psi\rangle$ es cero.

Producto Interno

El producto interno de un par de vectores $|\alpha\rangle, |\beta\rangle$ es un número complejo denotado por

$$\langle\alpha|\beta\rangle = \langle\beta|\alpha\rangle^* \tag{1.3}$$

donde $\langle\alpha|\alpha\rangle \geq 0$. Si su producto interno es cero entonces los vectores son ortogonales.

Producto Externo

Se define el producto externo como

$|\alpha\rangle\langle\beta|$

que denota un operador que mapea el ket $|\rho\rangle$ a otro ket mediante el bra $\langle\beta|$

$|\alpha\rangle\langle\beta|\rho\rangle = \boldsymbol{c}|\alpha\rangle$

donde $\boldsymbol{c} = \langle\beta|\rho\rangle$. Con este producto externo podemos construir un operador de proyección $\boldsymbol{P} = |\alpha\rangle\langle\alpha|$, con $\langle\alpha|\alpha\rangle = 1$, tal que $\boldsymbol{P}^2 = \boldsymbol{P}\cdot\boldsymbol{P} = |\alpha\rangle\langle\alpha||\alpha\rangle\langle\alpha| = |\alpha\rangle\langle\alpha| = \boldsymbol{P}$.

Sea $|\boldsymbol{a}_i\rangle = \langle\alpha_i|\alpha\rangle$ para $i = 1, \ldots, n$ y $|\alpha\rangle \in \xi$ entonces $\left(\sum_{i=1}^{N}|\alpha_i\rangle\langle\alpha_i|\right)|\alpha\rangle = \sum_{i=1}^{N}|\alpha_i\rangle\langle\alpha_i|\alpha\rangle = \sum_{i=1}^{N}\boldsymbol{a}_i|\alpha\rangle = |\alpha\rangle$

Entonces en una base discreta de dimensión N, el operador identidad puede escribirse

$$\boldsymbol{I} = \sum_{i,j=1}^{N}|\alpha_i\rangle\delta_{i,j}\langle\alpha_i| = \sum_{i=1}^{N}|\alpha_i\rangle\langle\alpha_i|, \tag{1.4}$$

donde $\delta_{i,j} = 0$ para todo $\boldsymbol{i} \neq \boldsymbol{j}$ y $\delta_{i,j} = 1$ para $\boldsymbol{i} = \boldsymbol{j}$.

Espacio de Hilbert

Los espacios de Hilbert [2], creados por el matemático David Hilbert, fueron formalizados por John Von Neumann y se convirtieron en el soporte matemático de la física y mecánica cuántica del primer cuarto del siglo XX. Un espacio de Hilbert es un espacio vectorial *H* completo (lo cual significa que cualquier sucesión de Cauchy de elementos del espacio converge a un elemento en el espacio. Una sucesión de Cauchy es una sucesión tal que la distancia entre dos términos se va reduciendo a medida que se avanza en la sucesión) con respecto a la norma vectorial con producto interno $\langle \boldsymbol{f}|\boldsymbol{g}\rangle$. La norma se encuentra definida por

$$\| \boldsymbol{f} \| \equiv \sqrt{\langle \boldsymbol{f}, \boldsymbol{f} \rangle}. \tag{1.5}$$

Dado un espacio de Hilbert *H* de dimensión 2^n (consideramos un espacio de estas dimensiones debido a que será la dimensión de los espacios que nos interesan en la teoría de la información cuántica), un conjunto de 2^n vectores $\boldsymbol{B} = \{|\boldsymbol{b}_m\rangle\} \subseteq \boldsymbol{H}$ es llamado una base ortonormal en *H* si

$$\langle \boldsymbol{b}_n|\boldsymbol{b}_m\rangle = \delta_{n,m} \qquad \forall\, \boldsymbol{b}_m, \boldsymbol{b}_n \in \boldsymbol{B} \tag{1.6}$$

y todo $|\psi\rangle \in \boldsymbol{H}$ puede ser escrito como

$$|\psi\rangle = \sum_{\boldsymbol{b}_n \in \boldsymbol{B}} \psi_n |\boldsymbol{b}_n\rangle \qquad \text{para algún } \psi_n \in \boldsymbol{C}. \tag{1.7}$$

Los valores de $\psi_n = \langle \boldsymbol{b}_n|\psi\rangle$ y son los coeficientes de $|\psi\rangle$ con respecto a la base $\{|\boldsymbol{b}_m\rangle\}$. Un ejemplo de una base ortonormal para un *H* de dimensión 4 es $\{|00\rangle, |01\rangle, |10\rangle, |11\rangle\}$ que es conocida como base computacional. En estas bases se etiquetan los 2^n vectores bases en la notación de Dirac mediante cadenas binarias de longitud *n*.

Operadores lineales

Un operador lineal *A* en *H* asocia a cada vector (ket) $|\psi\rangle$ otro ket, $|\psi'\rangle = \boldsymbol{A}|\psi\rangle \in \boldsymbol{H}$ en una correspondencia lineal:

$$\boldsymbol{A}\left[\lambda_1|\psi_1\rangle + \lambda_2|\psi_2\rangle\right] = \lambda_1\boldsymbol{A}|\psi_1\rangle + \lambda_2\boldsymbol{A}|\psi_2\rangle \; \forall \lambda_{1,2} \in \boldsymbol{C}$$

Se define la acción de un operador compuesto *AB* sobre un $|\psi\rangle$ cómo $\boldsymbol{A}\left[\boldsymbol{B}|\psi\rangle\right]$. Si los operadores no conmutan entre sí, el orden de la aplicación hace una diferencia. Definimos el conmutador de dos operadores como el operador compuesto $[\boldsymbol{A},\boldsymbol{B}] \equiv \boldsymbol{AB} - \boldsymbol{BA}$. Si $[\boldsymbol{A},\boldsymbol{B}] \neq 0$ entonces $\boldsymbol{AB}|\psi\rangle \neq \boldsymbol{BA}|\psi\rangle$.

Un operador lineal *A* actúa sobre un ket $|\psi\rangle = \sum_i \boldsymbol{a}_i |\boldsymbol{u}_i\rangle$ produciendo otro ket $\boldsymbol{A}|\psi\rangle$ al cual le corresponden componentes $\boldsymbol{b}_i = \langle \boldsymbol{u}_i | \boldsymbol{A}\psi\rangle$ dadas por:

$$\boldsymbol{b}_i = \langle \boldsymbol{u}_i | \boldsymbol{A}\psi\rangle = \langle \boldsymbol{u}_i | \boldsymbol{A}\sum_j \boldsymbol{a}_j |\boldsymbol{u}_j\rangle = \sum_j \boldsymbol{a}_j \langle \boldsymbol{u}|\boldsymbol{A}|\boldsymbol{u}_j\rangle = \sum_j \alpha_{ij}\boldsymbol{a}_j,$$

donde definimos $\alpha_{ij} = \langle \boldsymbol{u}|\boldsymbol{A}|\boldsymbol{u}_j\rangle$ y el conjunto $\{|\boldsymbol{u}_i\rangle, \forall_i\}$ son los estados propios de $\hat{\boldsymbol{A}}$.

Operador Autoadjunto

Dado un operador lineal *A*, se le asocia otro operador $\boldsymbol{A}^\dagger$ (el adjunto de *A*) a través de la relación

$$\langle \Theta|\boldsymbol{A}|\psi\rangle^* = \langle \psi|\boldsymbol{A}^\dagger|\Theta\rangle.$$

$$\boldsymbol{A}^\dagger = \left(\boldsymbol{A}^*\right)^T.$$

Un operador que cumple con la igualdad $\boldsymbol{A} = \boldsymbol{A}^\dagger$ se la llama **hermítico**. Si adicionalmente el dominio de *A* es el mismo que el de $\boldsymbol{A}^\dagger$ entonces el operador es autoadjunto. Esta distinción es únicamente relevante en espacios de dimensión infinita. Estos operadores son importantes en la Teoría de la Información Cuántica porque representan observables físicas. Los valores propios de un operador hermítico son reales.

Operadores Unitarios

Un operador unitario se define por la relación

$$\boldsymbol{U}^\dagger\boldsymbol{U} = \boldsymbol{U}\boldsymbol{U}^\dagger = \boldsymbol{I}$$

Una propiedad importante de los operadores unitarios es que conservan la norma dado que $\langle\psi|\psi\rangle = \langle\psi|\boldsymbol{U}^\dagger\boldsymbol{U}|\psi\rangle$. Por otro lado cabe destacar que los valores propios de un operador unitario tienen módulo 1. Entonces se pueden definir en términos de fases reales, $\varphi \in [0, 2\pi]$ con valor propio $\boldsymbol{e}^{i\varphi}$.

Producto Tensorial

El producto tensorial es una forma de combinar espacios, operadores y vectores. Suponiendo que H_1 y H_2 son espacios de Hilbert de dimensión *n* y *m* respectivamente. Entonces el espacio generado por el producto tensorial $\boldsymbol{H}_1 \otimes \boldsymbol{H}_2$ es un nuevo y más grande espacio de Hilbert de dimen-

sión $\boldsymbol{n}\times\boldsymbol{m}$. El producto tensorial de dos vectores $|\psi_1\rangle$ y $|\psi_2\rangle$ cuyos espacios son H_1 y H_2 respectivamente es un vector en $\boldsymbol{H}_1 \otimes \boldsymbol{H}_2$ denotado como $|\psi_1\rangle \otimes |\psi_2\rangle$. Suponiendo que A y B son dos operadores lineales sobre H_1 y H_2 respectivamente entonces $\boldsymbol{A}\otimes\boldsymbol{B}$ es el operador lineal en $\boldsymbol{H}_1 \otimes \boldsymbol{H}_2$ definido por

$$(\boldsymbol{A}\otimes\boldsymbol{B})(|\psi_1\rangle \otimes |\psi_2\rangle) = \boldsymbol{A}|\psi_1\rangle \otimes \boldsymbol{B}|\psi_2\rangle \tag{1.8}$$

1.2 Postulados

1.2.1 Descripción del estado de un sistema

Un estado cuántico es un objeto matemático que describe completamente un sistema cuántico. Podemos caracterizar el estado cuántico de una partícula por medio de un vector de estado, perteneciente a un espacio vectorial de estados $\xi \in \mathrm{H}$. Un vector de estado indica el estado en que se encuentra un sistema cuántico.

Cualquier elemento o vector del espacio ξ es llamado *vector ket* o *ket* y es representado bajo el símbolo $|\varphi\rangle$ (ver Sección 1.1). El símbolo dentro del vector (en nuestro ejemplo φ) nos sirve para distinguir el *vector ket* de otros vectores. Asociamos a cada función de onda un vector ket en el espacio ξ, de modo que toda la información sobre el estado cuántico del sistema a describir está contenido en el mismo.

Postulado 1: **En un tiempo dado t_0, el estado de un sistema físico está definido especificando el ket $|\varphi(t_0)\rangle$, perteneciente al espacio de estados ξ:**

Como ξ es un espacio vectorial el primer postulado implica el principio de superposición: una combinación lineal de vectores de estado es un vector de estado. Esto tiene fuertes implicaciones en la Teoría de la Información Cuántica ya que clásicamente un sistema de dos estados $(|0\rangle, |1\rangle)$ puede ser usado para representar un bit de información, pero cuánticamente el estado también puede estar "simultáneamente" en una combinación lineal de estados $(|\varphi\rangle = \alpha|0\rangle + \beta|1\rangle)$.

1.2.2 Descripción de cantidades físicas

Postulado 2: Toda cantidad física medible, A está descrita por un operador **A** actuando en el espacio ξ, y este operador es un observable, es decir un operador hermítico cuyos vectores propios son una base de ξ.

Es decir, todo observable físico está representado por operadores hermíticos que operan sobre los vectores ket y sus eigenvectores constituyen una base del espacio. Un operador hermítico definido sobre un espacio de Hilbert es un operador lineal que, sobre un cierto dominio, coincide con su propio operador adjunto (Ver Antecedentes). Una propiedad importante de estos operadores es que sus eigenvalores son siempre números reales y por lo tanto nuestro operador tendrá que ser Hermítico. Otra propiedad importante de los operadores Hermíticos es que sus vectores propios son ortogonales. Algunos operadores comunes en mecánica cuántica son: operador momento, operador posición, operador de momento angular, etc. A diferencia de la

mecánica clásica la mecánica cuántica describe de manera diferente el estado de un sistema y sus cantidades físicas asociadas. Un estado es representado por un vector y una cantidad física por un operador.

1.2.3 Medición de cantidades físicas

El operador asociado con la energía de un sistema es conocido como operador Hamiltoniano. La conexión entre el operador Hamiltoniano *H* y la energía total de una partícula se relacionan de la siguiente manera: las únicas energías posibles son los valores propios del operador *H*. Esta relación puede ser extendida a todas las cantidades físicas.

Postulado 3: **El único resultado posible de la medición de *A* es uno de los valores propios correspondientes a la observable.**

Una medición de *A* siempre nos dará un valor real dado que por definición *A* es Hermítico.

Si tenemos un observable A entonces podemos obtener los siguientes espectros (recordemos que un espectro es el conjunto de valores propios de *A*).

- Caso de un espectro discreto con valores propios no-degenerados: A cada valor propio λ_n de *A* se le asocia un único vector propio $|u_n\rangle$:

  $A|u_n\rangle = \lambda_n |u_n\rangle$.

  Como *A* es un observable el conjunto de los $|u_n\rangle$, que en nuestro caso está normalizado, constituyen una base en ξ, y el vector estado $|\psi\rangle$ puede ser escrito como:

  $$|\psi\rangle = \sum_n c_n u_n \ . \tag{1.9}$$

  La probabilidad $P(\lambda_n)$ de encontrar λ_n cuando medimos la observable *A* está dada por:

  $$P(\lambda_n) = |c_n|^2 = |\langle u_n|\psi\rangle|^2 \ . \tag{1.10}$$

  recordemos que $|u_n\rangle$ es el vector propio de *A* asociado con el valor propio a_n:

- Caso de un espectro discreto con valores propios degenerados: Si algunos valores propios de a_n son degenerados entonces varios vectores propios ortonormalizados $|u_n^i\rangle$ no corresponden a

  $$A|u_n^i\rangle = \lambda_n |u_n^i\rangle; \quad i = 1, 2, ..., g_n, \tag{1.11}$$

  donde g_n indica el orden de la degeneración del propio de λ_n. Como los $\{|u_n^i\rangle\}$ constituyen una base en el subespacio propio E_n asociado con el valor propio λ_n de *A* entonces $|\psi\rangle$ puede desarrollarse:

$$|\psi\rangle = \sum_{n}\sum_{i=1}^{g_n} c_n^i \left|u_n^i\right\rangle , \tag{1.12}$$

En este caso la probabilidad $P(\lambda_n)$ está dada por

$$P(a_n) = \sum_{i=1}^{g_n} \left|c_n^i\right|^2 = \sum_{i=1}^{g_n} \left|\left\langle u_n^i \middle| \psi \right\rangle\right|^2 . \tag{1.13}$$

donde g_n es el grado de degeneración de λ_n .

Como el valor propio de λ_n es degenerado entonces la probabilidad $P(\lambda_n)$ será independiente de la base $\left\{\left|u_n^i\right\rangle\right\}$ escogida en E_n.

Notemos que el caso del espectro discreto con valores propios no degenerados se obtiene de la ecuación (1.13) si tomamos $g_n = 1$:

- Caso de un espectro continuo y no degenerado: Sea el sistema de vectores propios ortonormales $|\nu_\alpha\rangle$ de A dado por

$$A|\nu_\alpha\rangle = \lambda|\nu_\alpha\rangle . \tag{1.14}$$

que forma una base continua en E, de manera que $|\psi\rangle$ puede ser expandido como

$$|\psi\rangle = \int d\lambda\, c(\lambda)|\nu_\lambda\rangle \tag{1.15}$$

Como los posibles resultados de una medición de A forman un conjunto continuo, se define una densidad de probabilidad. Entonces la probabilidad $dP(\lambda)$ de obtener un valor entre λ y $d\lambda$ será

$$dP(\lambda) = p(\lambda)\, d\lambda , \tag{1.16}$$

con

$$p(\lambda) = |c_\lambda|^2 = \left|\langle \nu_\lambda | \psi \rangle\right|^2 . \tag{1.17}$$

Un importante resultado de este postulado es que dados dos kets $|\psi\rangle$ y $|\psi'\rangle$ tal que

$$|\psi'\rangle = e^{i\theta}|\psi\rangle , \tag{1.18}$$

donde θ es un número real, si $|\psi\rangle$ está normalizada entonces $|\psi'\rangle$ también lo estará

$$\langle\psi'|\psi'\rangle = \langle\psi|e^{-i\theta}e^{i\theta}|\psi\rangle = \langle\psi|\psi\rangle = 1 . \tag{1.19}$$

Las probabilidades dadas por una medición arbitraria serán las mismas para $|\psi\rangle$ y $|\psi'\rangle$ dado que para cualquier $\left|u_n^i\right\rangle$

$$\left|\left\langle \boldsymbol{u}_n^i \middle| \psi' \right\rangle\right|^2 = \left|\boldsymbol{e}^{i\theta} \left\langle \boldsymbol{u}_n^i \middle| \psi \right\rangle\right|^2 = \left|\left\langle \boldsymbol{u}_n^i \middle| \psi \right\rangle\right|^2 . \tag{1.20}$$

Las fases globales no afectan las predicciones físicas (las fases relativas de los coeficientes de una expansión si lo hacen) [3]. Entonces dos vectores estado proporcionales representan el mismo estado físico. Cabe destacar que este postulado de medición no tiene una versión clásica análoga. La mecánica clásica asume que las mediciones pueden ser realizadas sin afectar al sistema. El postulado de medición de la mecánica cuántica describe un escenario diferente. Si nosotros conocemos todo lo que podemos saber de un estado en el sistema únicamente podemos predecir la probabilidad del resultado de la medición.

1.2.4 Reducción del paquete de ondas

Supongamos que queremos medir en un tiempo dado una cantidad física A. Si el ket $|\psi\rangle$, que representa el estado de un sistema inmediatamente antes de la medición, es conocido, entonces nuestro postulado de medición (en el caso de un espectro continuo no degenerado) nos permite predecir las probabilidades de los distintos resultados posibles. Pero cuando la medición es realmente realizada, únicamente uno de estos posibles resultados es obtenido. Inmediatamente después de la medición ya no podemos hablar de la probabilidad de haber obtenido tal valor sino de la certeza de conocer el valor de la medición. Se obtiene nueva información y por lo tanto el estado del sistema después de la medición es diferente a $|\psi\rangle$. Suponiendo que al medir A obtenemos el valor propio a_n del observable A. Entonces el estado del sistema inmediatamente después de la medición está dado por el vector propio $|\upsilon_n\rangle$ asociado con a_n es

$$|\psi\rangle \xrightarrow{(\lambda_n)} |\upsilon_n\rangle \tag{1.21}$$

Si realizamos una segunda medición de A inmediatamente después de nuestra primera medición (i.e. antes de que el sistema tenga tiempo de evolucionar), entonces siempre encontraremos el mismo resultado λ_n ya que el estado de sistema inmediatamente antes a la segunda medición es $|\upsilon_n\rangle$.

Cuando el valor propio obtenido a_n es degenerado entonces la ecuación (1.21) puede ser generalizada de la siguiente manera. Si la expansión del estado $|\psi\rangle$ inmediatamente antes de la medición la denotamos como en (1.12)

$$|\psi\rangle = \sum_n \sum_{i=1}^{g_n} \boldsymbol{c}_n^i \left| \boldsymbol{u}_n^i \right\rangle \tag{1.22}$$

la modificación del vector estado debido a la medición queda como

$$|\psi\rangle \xrightarrow{(\lambda_n)} \frac{1}{\sqrt{\sum_{i=1}^{g_n} \left|\boldsymbol{c}_n^i\right|^2}} \sum_{i=1}^{g_n} \boldsymbol{c}_n^i \left| \boldsymbol{u}_n^i \right\rangle . \tag{1.23}$$

donde $\sum_{i=1}^{g_n} \boldsymbol{c}_n^i \left| \boldsymbol{u}_n^i \right\rangle$ es el vector $|\psi_n\rangle$, es decir, la proyección de $|\psi\rangle$ en el subespacio asociado a λ_n. Los coeficientes de $\boldsymbol{c}_n^i$ son los mismos obtenidos en la expansión de $|\psi\rangle$

$$c_n^i = \langle u_n^i | \psi \rangle , \tag{1.24}$$

donde

$$\langle \psi_n | \psi_n \rangle = \sum_{i=1}^{g_n} \left| c_n^i \right|^2 . \tag{1.25}$$

Entonces podemos escribir $|\psi_n\rangle$ como

$$|\psi_n\rangle = \sum_{i=1}^{g_n} |u_n^i\rangle \langle u_n^i | \psi \rangle = P_n |\psi\rangle , \tag{1.26}$$

donde $P_n = \sum_{i=1}^{g_n} |u_n^i\rangle \langle u_n^i|$. En la ecuación 1.23 el vector se encuentra normalizado debido a que la probabilidad total debe ser 1.

Entonces el estado de un sistema inmediatamente después de la medición es la proyección normalizada

$$|\psi\rangle \xrightarrow{(\lambda_n)} \frac{P_n |\psi\rangle}{\sqrt{\langle \psi | P_n | \psi \rangle}} . \tag{1.27}$$

Por lo tanto el estado de un sistema inmediatamente después de la medición siempre será un vector propio de *A* con el valor propio λ_n.

1.2.5 Evolución Temporal

La evolución temporal de un vector estado está dada por la ecuación de Schrödinger

$$i\hbar \frac{d|\psi(t)\rangle}{dt} = H(t)\psi(t) . \tag{1.28}$$

donde *H(t)* es el observable asociado con la energía total del sistema. *H* es el operador Hamiltoniano del sistema y se obtiene del Hamiltoniano clásico. Recordemos que si conocemos el Hamiltoniano del sistema entonces podemos conocer su dinámica completamente.

Es decir la evolución dinámica de un sistema cuántico es reversible y determinista. Como la ecuación (1.28) es lineal entonces una combinación lineal de sus soluciones será también solución. Recordemos que $|\psi\rangle$ y $e^{i\theta}|\psi\rangle$ representan el mismo estado.

Se define un operador evolución *U(t0,t)* que al aplicarse $|\psi(t_0)\rangle$ nos da $|\psi(t)\rangle$. Podemos expresar la ecuación (1.28) como

$$\frac{d|\psi(t)\rangle}{\psi(t)} = -i\hbar H(t) dt . \tag{1.29}$$

Si integramos (1.29) obtenemos

$$\left|\psi(t)\right\rangle = \boldsymbol{T}\left\{\boldsymbol{exp}\left[-\frac{\boldsymbol{i}}{\hbar}\int_{t_0}^{t}\boldsymbol{H}(t')dt'\right]\right\}\left|\psi(t_0)\right\rangle = \boldsymbol{U}\left|\psi(t_0)\right\rangle, \tag{1.30}$$

donde el símbolo *T{}* significa integral ordenada temporalmente.

Si el sistema que se está describiendo es conservativo, su energía es una constante del movimiento y el operador *H* no depende del tiempo. Entonces el operador de evolución entre t_1 y t_2 es

$$\boldsymbol{U}(t_1,t_2) = \boldsymbol{e}^{-\frac{i}{\hbar}\boldsymbol{H}(t_2-t_1)}, \tag{1.31}$$

Como el hamiltoniano es hermítico, $\boldsymbol{U}^{\dagger}\boldsymbol{U}=1$. La evolución temporal de un sistema cuántico cerrado está descrita por un operador unitario. Después de la evolución el estado de sistema será

$$\left|\psi(t)\right\rangle = \boldsymbol{U}\left|\psi(t_0)\right\rangle, \tag{1.32}$$

1.2.6 Postulado de simetrización

Dos o más partículas son indistinguibles si ningún valor de expectación de cualquier observable se ve afectado por las permutaciones de las partículas.

En un sistema de *N* partículas indistinguibles, los únicos posibles estados del sistema son aquellos descritos por vectores que son (respecto a las permutaciones de las partículas):

Completamente simétricos: en este caso las partículas son llamadas bosones

$$\left|\psi_{a,b}\right\rangle = \frac{1}{\sqrt{2}}\left\{\left|\varphi\right\rangle_a\left|\chi\right\rangle_b + \left|\chi\right\rangle_a\left|\varphi\right\rangle_b\right\}.$$

Completamente antisimétricos: en este caso las partículas son llamadas fermiones.

$$\left|\psi_{a,b}\right\rangle = \frac{1}{\sqrt{2}}\left\{\left|\varphi\right\rangle_a\left|\chi\right\rangle_b - \left|\chi\right\rangle_a\left|\varphi\right\rangle_b\right\}.$$

1.2.7 Variables de espín

El grado de libertad intrínseco de una partícula es conocido como espín.

Decimos que una partícula tiene espín s si su grado de libertad intrínseco puede ser descrito por una representación del grupo de rotaciones de dimensión 2*s* + 1. Por ejemplo el espín $\frac{1}{2}$ puede ser descrito por una representación de dimensión 2.

Un electrón tiene espín $\frac{1}{2}$, entonces una diferencia de polarización de 180º (arriba y abajo) significa ortogonalidad.

Definimos $\vec{\boldsymbol{S}}$ como un vector de operadores

$$\vec{S} = (S_x, S_y, S_z)\,, \tag{1.33}$$

tal que las componentes de $\vec{S}$ satisface las relaciones de conmutación del momento angular

$$\left[S_x, S_y\right] = i\hbar S_z\,, \tag{1.34}$$
$$\left[S_y, S_z\right] = i\hbar S_x\,, \tag{1.35}$$
$$\left[S_z, S_x\right] = i\hbar S_y\,, \tag{1.36}$$

Estas relaciones de conmutación pueden escribirse de una manera más compacta en términos del producto vectorial

$$\vec{S} \times \vec{S} = i\hbar \vec{S}\,, \tag{1.37}$$

que también puede ser escrito como

$$\left[S_i, S_j\right] = i\hbar \varepsilon_{i,j,k} S_k\,, \tag{1.38}$$

donde

$$\varepsilon_{i,j,k} \begin{cases} +1 & si\,(i,j,k)\,es\,(1,2,3),(3,1,2),(2,3,1) \\ -1 & si\,(i,j,k)\,es\,(3,2,1),(1,3,2),(2,1,3) \\ 0 & i = j; j = k; k = i \end{cases}$$

Como cualquier momento angular, existe un espacio de Hilbert H_s asociado con un espín que soporta valores propios de espín.

Un conjunto de vectores base para este espacio de Hilbert puede ser construido a partir de vectores propios simultáneos de los operadores $S^2 = |S|^2$ y S_z. Los valores propios asociado los denotamos por s y m_s. S^2 y S_z satisfacen las ecuaciones de valores propios siguientes:

$$S^2 \left| s\, m_s \right\rangle = \hbar\, s(s+1) \left| s\, m_s \right\rangle\,, \tag{1.39}$$
$$S_z \left| s\, m_s \right\rangle = \hbar\, m_s \left| s\, m_s \right\rangle\,, \tag{1.40}$$

Notemos que

$$S^2 = S_x^2 + S_y^2 + S_z^2\,, \tag{1.41}$$

y es un operador tal que conmuta con sus componentes

$$\left[S^2, S_i\right] = 0\,, \tag{1.42}$$

El espacio de Hilbert del espín H_s debe unirse al espacio de Hilbert H_r asociado con las variables clásicas de posición y momento $\vec{R}$ y $\vec{P}$. Esto es logrado realizando el producto tensorial de espacios

$$\boldsymbol{H} = \boldsymbol{H}_s \otimes \boldsymbol{H}_r \,, \tag{1.43}$$

por supuesto las variables de espín conmutan con $\vec{\boldsymbol{R}}$ y $\vec{\boldsymbol{P}}$

$$\left[\boldsymbol{S}_i, \boldsymbol{R}_j\right] = \left[\boldsymbol{S}_i, \boldsymbol{P}_j\right] = 0 \,, \tag{1.44}$$

Como el electrón es una partícula cargada, el espín del electrón da lugar a un momento magnético $\vec{\mu}$ intrínseco o de espín. La relación que existe entre el vector momento magnético y el espín es $\vec{\mu} = 2\frac{\mu_B \vec{\boldsymbol{S}}}{\hbar}$; es una cantidad donde

$$\mu_B = \frac{\boldsymbol{e}\hbar}{2\boldsymbol{m}} \qquad \text{Magnetón de Bohr} \tag{1.45}$$

A continuación se discuten experimentos y conceptos muy importantes tanto en el desarrollo de la teoría de la información cuántica como en el establecimiento de los cubits (o bits cuánticos).

1.3 Experimento de la doble rendija

Una manera sencilla de ilustrar algunas peculiaridades de la Mecánica Cuántica es el experimento de la doble rendija de Young [4], que en 1981 se realizó por primera vez con el objetivo de responder la pregunta de si la luz es un haz de partículas o una onda. Durante el transcurso de los años grandes científicos difirieron en referencia a esta pregunta. Maxwell afirmaba que la luz era claramente una onda, posteriormente Einstein le regresó a la luz su descripción como partícula en su teoría del efecto fotoeléctrico. El experimento de Young realiza una completa descripción de este fenómeno que sólo puede ser obtenida si aceptamos la dualidad onda-partícula de la luz. Una fuente *S* emite un haz de luz la cual puede pasar a través de dos rejillas O_1 y O_2 o O_1 u O_2 antes de impactar una pantalla de proyección o una placa fotográfica.

La intensidad de la luz *I(x)* en un punto *x*, es proporcional al cuadrado del módulo del campo eléctrico *E(x)* en el mismo punto. Entonces la intensidad de la luz observada a través de la rejilla O_1 cuando la rejilla O_2 se encuentra cerrada, viene dada por

$$\boldsymbol{I}_1(\boldsymbol{x}) \propto \left|\boldsymbol{E}_1(\boldsymbol{x})\right|^2 , \tag{1.46}$$

donde $E_1(x)$ es el campo eléctrico producido en *x* a través de O_1. Análogamente cuando la rejilla O_2 es cerrada se observa

$$\boldsymbol{I}_2(\boldsymbol{x}) \propto \left|\boldsymbol{E}_2(\boldsymbol{x})\right|^2 , \tag{1.47}$$

Si abrimos ambas rejillas ambos campos se suman algebraicamente

$$\boldsymbol{E}(\boldsymbol{x}) = \boldsymbol{E}_1(\boldsymbol{x}) + \boldsymbol{E}_2(\boldsymbol{x}) \,, \tag{1.48}$$

y por lo tanto la intensidad de la luz resultante es

$$\boldsymbol{I}(\boldsymbol{x}) \propto \left|\boldsymbol{E}(\boldsymbol{x})\right|^2 = \left|\boldsymbol{E}_1(\boldsymbol{x}) + \boldsymbol{E}_2(\boldsymbol{x})\right|^2, \tag{1.49}$$

por lo tanto

$$\boldsymbol{I}(\boldsymbol{x}) \neq \boldsymbol{I}_1(\boldsymbol{x}) + \boldsymbol{I}_2(\boldsymbol{x}), \tag{1.50}$$

Este resultado va de acuerdo a la teoría de que la luz es una onda.

¿Qué pasa si lanzamos fotones uno por uno?

- Si la cantidad de fotones es limitada, entonces se observan puntos de impacto localizados, por lo tanto se comporta como partícula.

- Si la cantidad de fotones es lo suficientemente grande, entonces aparece un patrón de interferencia, lo que implica que su comporta como ondas.

Esta interferencia sólo es plasmada si no existe un observable; esto implica que no podremos conocer su posición sino únicamente *p(x)*, la probabilidad de que un fotón se plasme en el punto *x*.

Entonces la descripción de la luz como onda y como partícula no son mutuamente excluyentes, por lo tanto la luz se comporta simultáneamente como onda y partícula.

1.4 El gato de Schrödinger

Para explicar la ambivalencia existente en el proceso de la medición se describe a continuación el experimento legendario del gato de Schrödinger.

Para observar un sistema físico, el cual se lo hace interactuar con un aparato, el que debe describirse cuánticamente. Pero por otro lado el resultado de la observación es registrada por el aparato en forma clásica, necesitándose entonces una traducción del formalismo del espacio de Hilbert a un lenguaje clásico. Este experimento es el prototipo de ejemplo de la teoría de la información. En donde se establece que los ingredientes esenciales de una medición son el objeto, un aparato y una interacción que produce una correlación entre variables dinámicas del objeto y las variables del aparato. El gato de Schrödinger permite comparar las dos interpretaciones de la mecánica cuántica de un vector estado: en una de ellas se considera que un estado puro $|\psi\rangle$ determine en forma completa un sistema individual y en la otra que únicamente describe las propiedades estadísticas de un ensamble de sistemas preparado en forma similar.

En 1935 Schrödinger atacó el problema de coherencia de la mecánica cuántica, este problema nos ayudará a entender algunas propiedades de la mecánica cuántica. El experimento mental ideado por Schrödinger es el siguiente:

Un gato está encerrado en una cámara de acero (Figura 1.1), junto con el siguiente aparato (que debe ser protegido frente a una posible injerencia por parte del gato): en un contador Geiger hay una minúscula cantidad de una sustancia radioactiva, tan pequeña que tal vez, en el transcurso de una hora, uno de los átomos se desintegre, pero también, con igual probabilidad, ninguno lo haga; si sucede, el tubo del contador Geiger se descarga y, a través de un dispositivo libera un martillo

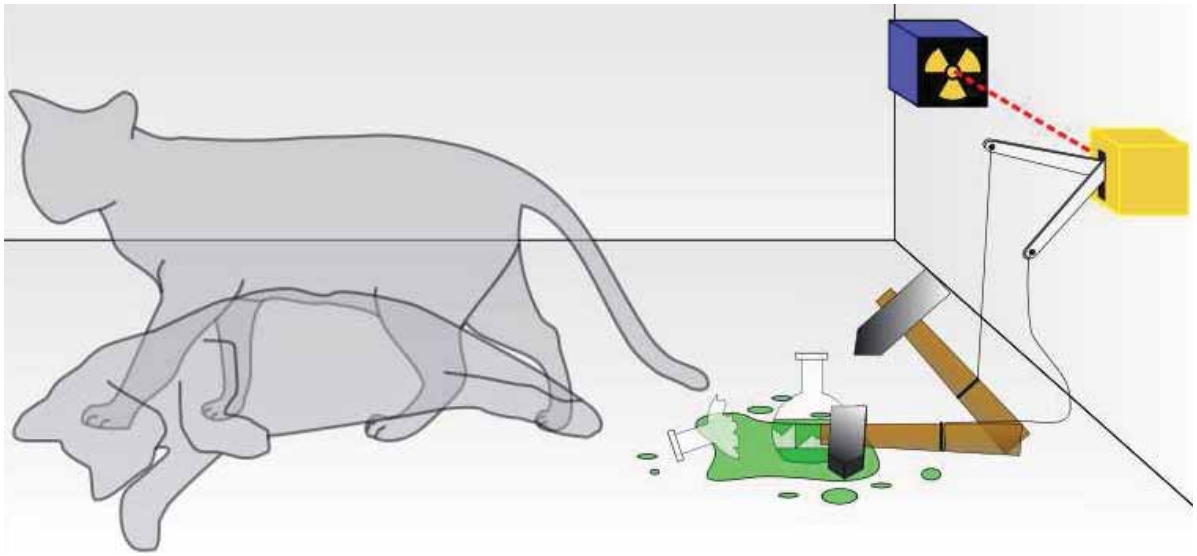


Figura 1.1: Experimento del gato de Schrödinger.

que rompe un pequeño frasco de ácido cianhídrico. Si se deja este sistema aislado durante una hora, podríamos decir entonces que el gato seguirá vivo si ningún átomo se ha desintegrado. La función de onda de este sistema expresaría esto incluyendo el gato vivo y el gato muerto mezclados o esparcidos en partes iguales, junto con el estado del átomo radioactivo.

Entonces podemos describir la función de onda del estado antes de abrir la caja como

$$\frac{1}{\sqrt{2}}|\boldsymbol{vivo}\rangle|\boldsymbol{nd}\rangle+\frac{1}{\sqrt{2}}|\boldsymbol{muerto}\rangle|\boldsymbol{d}\rangle\,, \tag{1.51}$$

donde $|\boldsymbol{nd}\rangle$ significa que el átomo no ha decaído y $|\boldsymbol{d}\rangle$ que el átomo decae.

Este es un estado correlacionado y también es una superposición de estados macroscópicos diferentes que es típico del proceso de medición.

1.5 Entrelazamiento cuántico

A mediados de los 30´s Einstein, Podolsky y Rosen (EPR) junto con Schrödinger reconocieron una interesante propiedad de la mecánica cuántica que marcó a la física del siglo XX.

Einstein, Podolsky y Rosen describieron el entrelazamiento cuántico en su intento de atribuir valores (elementos de la realidad) a las magnitudes físicas antes de efectuar la medición y el mencionar que la teoría cuántica no es completa (Teoría Completa: A todo elemento de la realidad le corresponde una contraparte de la teoría física) excluyendo la posibilidad de acción a distancia (Principio de Localidad). Pero la mecánica cuántica nos arroja resultados contrarios a estos supuestos.

En 1965 Bell formalizó las conclusiones obtenidas por EPR de la siguiente manera: los resultados obtenidos por una medición se encuentran determinados por las propiedades de las partículas y son independientes de la medición (realismo) y los resultados obtenidos en determinada posición son independientes de cualquier acción realizada fuera de su entorno cercano (localidad). Bell

demostró que estos resultados pueden observarse durante experimentos en el laboratorio en forma de las llamadas desigualdades de Bell, de la misma manera también demostró que la mecánica cuántica viola estas desigualdades (Ver 1.6).

En 1935, inspirado por el artículo EPR, Schrödinger analizando algunas consecuencias físicas en el formalismo cuántico descubrió un aspecto fundamental del entrelazamiento: *El mejor conocimiento posible de un todo no incluye el mejor conocimiento posible de sus partes*.

Esta propiedad implica la existencia de estados globales de sistemas compuestos que no pueden ser escritos como un producto de estados de cada uno se los subsistemas.

Este fenómeno, conocido como enredamiento o entrelazamiento subraya un orden intrínseco de correlaciones estadísticas entre subsistemas compuestos de sistemas cuánticos. Una manera sencilla de explicar esto es suponer la existencia de dos sistemas cuánticos A y B. Si estos sistemas están entrelazados, significa que los valores de ciertas propiedades del sistema A se correlacionan con los valores que asumirán las propiedades del sistema B. Estas correlaciones se pueden mantener incluso si ambos sistemas se encuentren espacialmente separados.

Por definición, un estado en un espacio de Hilbert *H* se encuentra entrelazado si no puede ser escrito como un producto tensorial de sus estados

$$|\psi\rangle \neq |\Phi_1\rangle \otimes |\Phi_2\rangle , \tag{1.52}$$

Los estados que no se encuentran entrelazados son llamados estados separables.

Dicho aspecto del entrelazamiento fue formalizado a mediados de los 90´s en términos de desigualdades de entropías (Entropía: Cantidad de información de una variable dada [1]).

La violación de las desigualdades de Bell por estados entrelazados es una característica de los estados cuánticos entrelazados que implica la existencia de información cuántica negativa, aportando un recurso extra en las comunicaciones cuánticas. Posteriormente se descubrió que la capacidad de transmitir información cuántica era posible cuando la entropía de salida del sistema excedía la entropía total del sistema.

Actualmente el entrelazamiento cuántico da origen a varios descubrimientos relacionados con la información y el cómputo cuántico: Criptografía Cuántica mediante el teorema de Bell, el codificado denso [1] y la teleportación cuántica [1]. Todos estos efectos están basados en el éntrelazamiento cuántico y han sido demostrados en el laboratorio. Estos resultados dieron origen a lo que conocemos como Información Cuántica la cual incorpora al entrelazamiento como una noción central.

Desafortunadamente el entrelazamiento cuántico tiene una estructura muy compleja, es muy frágil al ambiente y no puede ser aumentado si los sistemas no se encuentran en contacto directo.

Al parecer la aplicación más interesante del entrelazamiento cuántico se da en la criptografía cuántica. Se puede interpretar dicho entrelazamiento como un equivalente cuántico del significado de privacidad. La herramienta básica de esta privacidad es la llave privada secreta. Si los

sistemas se encuentran en un entrelazamiento puro entonces dichos sistemas están correlacionados y ningún otro sistema puede estar correlacionado a ellos.

1.6 Estados de Bell

Un estado de Bell o también conocido como par EPR, es definido como un estado de máximo entrelazamiento cuántico de dos cubits. Un estado se encuentra máximamente entrelazado si podemos construir a partir de él cualquier otro estado de la base usando operaciones locales y comunicaciones clásicas (LOCC). Una operación local es aquella operación que no actúa en dos cubits simultáneamente. Cualesquiera otros estados pueden ser formados a través de operaciones unitarias actuando en este estado. Los estados de Bell forman una base ortogonal del espacio de estados cuánticos de dos cubits denominada base de Bell.

$$\left|\phi^{+}\right\rangle = \frac{1}{\sqrt{2}}(\left|00\right\rangle + \left|11\right\rangle),$$

$$\left|\phi^{-}\right\rangle = \frac{1}{\sqrt{2}}(\left|00\right\rangle - \left|11\right\rangle),$$

$$\left|\psi^{+}\right\rangle = \frac{1}{\sqrt{2}}(\left|01\right\rangle + \left|10\right\rangle),$$

$$\left|\psi^{-}\right\rangle = \frac{1}{\sqrt{2}}(\left|01\right\rangle - \left|10\right\rangle),$$

1.7 Desigualdades de Bell

En 1965 Bell afirmó que toda teoría de variables ocultas y local tiene necesariamente algunas predicciones incompatibles con la Mecánica Cuántica (Teorema de Bell). De este resultado surgen las desigualdades de Bell que toda teoría realista y local debe de satisfacer.

Cabe destacar que la mecánica cuántica en general no satisface estas desigualdades.

Suponiendo que tenemos un emisor de partículas: Este emisor prepara dos partículas y las reparte una a Alice y otra a Bob. Una vez que Alice recibe su partícula, realiza una medición sobre ella. Supongamos que Alice cuenta con dos aparatos diferentes de medición tal que ella puede escoger que aparato usar. Si Alice usa el primer aparato las propiedades físicas de la medición vienen dadas por P_Q , de la misma manera para el segundo aparato las propiedades físicas vienen dadas por P_R. Alice escoge aleatoriamente cuál de los dos aparatos usar. Al realizar la medición Alice podrá obtener como resultado -1 o +1, suponiendo que la partícula de Alice tiene un valor Q para la propiedad P_Q. De la misma manera se tiene un valor R para P_R . Similarmente Bob es capaz de medir una de las siguientes propiedades P_S o P_T encontrando los valores S o T respectivamente cada uno con valor -1 o +1. Al igual que Alice, Bob escoge aleatoriamente que aparato de medida usar. Una vez que escogieron su aparato de medida, Alice y Bob realizan su medición, tal que la medición realizada por Alice (Bob) no pueda perturbar el resultado de Bob (Alice). El resultado de estas mediciones está regido por el principio de localidad. Calcularemos el valor de

$$\boldsymbol{QS} + \boldsymbol{RS} + \boldsymbol{RT} - \boldsymbol{QT} = (\boldsymbol{Q} + \boldsymbol{R})\boldsymbol{S} + (\boldsymbol{R} - \boldsymbol{Q})\boldsymbol{T}\,, \tag{1.53}$$

Como $\boldsymbol{R}$, $\boldsymbol{Q} = \pm 1$ entonces $(\boldsymbol{Q} + \boldsymbol{R})\boldsymbol{S} = 0$ o $(\boldsymbol{R} - \boldsymbol{Q})\boldsymbol{T} = 0$. En cualquiera de los dos casos obtenemos que $\boldsymbol{QS} + \boldsymbol{RS} + \boldsymbol{RT} - \boldsymbol{QT} = \pm 2$. Se define ahora $p(q,r,s,t)$ como la probabilidad de que el sistema antes

de la medición se encuentra en un estado donde, $Q = q$, $R = r$, $S = s$ y $T = t$. Estas probabilidades dependen en la forma en que el emisor prepara los estados y si encontramos algún ruido externo. Se denota $E(\bullet)$ como el coeficiente de correlación, entonces

$$\begin{aligned} \boldsymbol{E}(\boldsymbol{QS}+\boldsymbol{RS}+\boldsymbol{RT}-\boldsymbol{QT}) &= \sum_{\boldsymbol{q},\boldsymbol{r},\boldsymbol{s},\boldsymbol{t}} \boldsymbol{p}(\boldsymbol{q},\boldsymbol{r},\boldsymbol{s},\boldsymbol{t})(\boldsymbol{qs}+\boldsymbol{rs}+\boldsymbol{rt}-\boldsymbol{qt}) \\ &\le \sum_{\boldsymbol{q},\boldsymbol{r},\boldsymbol{s},\boldsymbol{t}} \boldsymbol{p}(\boldsymbol{q},\boldsymbol{r},\boldsymbol{s},\boldsymbol{t})\times 2 \quad , \\ &= 2 \end{aligned} \tag{1.54}$$

Por otro lado

$$\begin{aligned} \boldsymbol{E}(\boldsymbol{QS}+\boldsymbol{RS}+\boldsymbol{RT}-\boldsymbol{QT}) &= \sum_{\boldsymbol{q},\boldsymbol{r},\boldsymbol{s},\boldsymbol{t}} \boldsymbol{p}(\boldsymbol{q},\boldsymbol{r},\boldsymbol{s},\boldsymbol{t})\boldsymbol{qs} + \sum_{\boldsymbol{q},\boldsymbol{r},\boldsymbol{s},\boldsymbol{t}} \boldsymbol{p}(\boldsymbol{q},\boldsymbol{r},\boldsymbol{s},\boldsymbol{t})\boldsymbol{rs} \\ &+ \sum_{\boldsymbol{q},\boldsymbol{r},\boldsymbol{s},\boldsymbol{t}} \boldsymbol{p}(\boldsymbol{q},\boldsymbol{r},\boldsymbol{s},\boldsymbol{t})\boldsymbol{rt} - \sum_{\boldsymbol{q},\boldsymbol{r},\boldsymbol{s},\boldsymbol{t}} \boldsymbol{p}(\boldsymbol{q},\boldsymbol{r},\boldsymbol{s},\boldsymbol{t})\boldsymbol{qt} \quad , \\ &= \boldsymbol{E}(\boldsymbol{QS})+\boldsymbol{E}(\boldsymbol{RS})+\boldsymbol{E}(\boldsymbol{RT})-\boldsymbol{E}(\boldsymbol{QT}) \end{aligned} \tag{1.55}$$

Por (1.7.2) y (1.7.3) se obtiene la desigualdad de Bell

$$\boldsymbol{E}(\boldsymbol{QS})+\boldsymbol{E}(\boldsymbol{RS})+\boldsymbol{E}(\boldsymbol{RT})-\boldsymbol{E}(\boldsymbol{QT}) \le 2\,, \tag{1.56}$$

esta desigualdad también es conocida como CHSH por Clauser, Horne, Shimony y Holt [5].

Se prepara el mismo experimento pero ahora nuestro emisor prepara sistemas cuánticos de dos cubits tales que se encuentren en el estado

$$|\varphi\rangle = \frac{|01\rangle - |10\rangle}{\sqrt{2}}\,, \tag{1.57}$$

El emisor envía el primer cubit a Alice y el segundo cubit a Bob. Alice y Bob realizan mediciones sobre los siguientes observables:

$$\boldsymbol{Q} = \boldsymbol{Z}_1 \quad , \quad \boldsymbol{S} = \frac{-\boldsymbol{Z}_2 - \boldsymbol{X}_2}{\sqrt{2}} \quad , \quad \boldsymbol{R} = \boldsymbol{X}_1 \quad , \quad \boldsymbol{T} = \frac{\boldsymbol{Z}_2 - \boldsymbol{X}_2}{\sqrt{2}}\,, \tag{1.58}$$

donde X y Z toman la forma

$$\boldsymbol{X} = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \quad , \quad \boldsymbol{Z} = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}. \tag{1.59}$$

Si medimos los valores medios (correlaciones cuánticas) con respecto al estado $|\varphi\rangle$ de estos observables se obtiene

$$\langle \boldsymbol{QS} \rangle = \frac{1}{\sqrt{2}}; \quad \langle \boldsymbol{RS} \rangle = \frac{1}{\sqrt{2}}; \quad \langle \boldsymbol{RT} \rangle = \frac{1}{\sqrt{2}}; \quad \langle \boldsymbol{QT} \rangle = \frac{1}{\sqrt{2}}, \tag{1.60}$$

entonces

$$\langle QS\rangle + \langle RS\rangle + \langle RT\rangle - \langle QT\rangle = 2\sqrt{2}.. \tag{1.61}$$

Por lo tanto la desigualdad de Bell no es satisfecha bajo la teoría de la mecánica cuántica.
Este resultado junto con el entrelazamiento es aprovechado en computación e información cuántica, esencialmente en la teleportación, codificado denso y el protocolo de Ekert de criptografía cuántica [1].

1.8 Cuantificación del entrelazamiento

Definir si un estado se encuentra entrelazado o no es relativamente sencillo pero medir o cuantificar la cantidad de entrelazamiento del estado no lo es. Algunas condiciones necesarias [6] que cualquier tipo de medición de entrelazamiento *E* sobre un estado σ debe de satisfacer las siguientes postulados:

- No-negatividad: $E(\sigma) \geq 0$
- $E(\sigma) = 0$ sí y solo sí σ es separable.
- Operaciones Locales Unitarias no modifican el valor $E(\sigma)$.
- *E* no aumenta su promedio sobre operaciones locales clásicas: $E(\sigma) \geq \sum_i p_i E(\rho_i)$, donde el estado ρ_i bajo operaciones locales es obtenido con probabilidad p_i.
- *E* es continua
- *E* es aditiva (sobre estados puros): $E(|\varphi^{AB}\rangle \otimes |\phi^{AB}\rangle) = E(|\varphi^{AB}\rangle) + E(|\phi^{AB}\rangle)$.

Entropia de Von Neumann: Esta medición del entrelazamiento es más sencilla de calcular.

Dado un estado puro ρ_{AB} (un estado puro es un estado donde su vector estado $|\psi\rangle$ es exactamente conocido) de dos subsistemas A y B, se define el estado $\rho_A = tr_B\{\rho_{AB}\}$ (tr_B es la traza parcial sobre el cubit B) y $\rho_B = tr_A\{\rho_{AB}\}$ es la traza parcial sobre el cubit A.

Entonces la entropía de Von Neumann está dada por [6]

$$S(\rho_A) = -tr(\rho_A \ln \rho_A) = -tr(\rho_B \ln \rho_B). \tag{1.62}$$

En este caso la cantidad de entrelazamiento es cero si el estado es separable y para un sistema compuesto de dos cubits es *ln*2 si el estado se encuentra máximamente entrelazado. Se verá más a fondo esta entropía en el Capitulo 3: Teoría Cuántica de la Información.

Se puede también caracterizar la cantidad de entrelazamiento de un estado calculando la concurrencia *C*, es decir, la cantidad de traslape entre un estado $|\psi\rangle$ y otro estado $|\tilde{\psi}\rangle$

$$C|\psi\rangle = |\langle\psi|\tilde{\psi}\rangle|,$$

donde $|\tilde{\psi}\rangle = Y \otimes Y|\psi^*\rangle$ y $|\psi^*\rangle$ es el complejo conjugado del estado y *Y* un operador de Pauli. La concurrencia también puede ser calculada usando el operador densidad (Ver Capitulo 3), observando los valores propios de la matriz densidad resultante de $\rho(Y \otimes Y)\rho^{\dagger}(Y \otimes Y)$.

La idea inicial de realizar una cuantificación del entrelazamiento fue dada por Bennett en 1996, definida por el costo de entrelazamiento.

Costo de Entrelazamiento: Es una medición de entrelazamiento entre dos sistemas (Alice y Bob). Se define el costo de la creación de entrelazamiento de un estado ρ por:

$$\boldsymbol{E}_C(\rho) = \min \sum_i \boldsymbol{p}_i \, \boldsymbol{S}(\rho_A^i) \,. \tag{1.63}$$

donde $\boldsymbol{S}(\rho_A)$ es la entropía de Von Neumann (ver ecuación 1.62) y el mínimo es tomado sobre todas las posibles realizaciones del estado, p_i denota la probabilidad de que el estado se encuentre en $|\psi_i\rangle$ y $\rho_A^i = \boldsymbol{tr}_B(|\psi_i\rangle\langle\psi_i|)$.

1.9 Dinámica con partículas con Espín

Como partículas de espín 1/2 son el paradigma de un sistema de dos niveles a continuación discutimos la dinámica clásica y cuántica de partículas con espín. Como veremos más adelante los sistemas de dos niveles tienen un papel relevante en el establecimiento de los cubits. La dinámica de espín puede ser descrita usando una teoría cuasi-clásica. El espín de una partícula es análogo a su momento angular intrínseco. Este momento angular intrínseco es un vector, no escalar, y por lo tanto el espín también es un vector este espín intrínseco siempre se encontrará presente. Una propiedad importante para determinar la dinámica del espín en un campo magnético es su momento magnético. La propiedad principal de un electrón o espín nuclear es que su momento magnético $\vec{\mu}$ es paralelo al momento angular de espín $\vec{\boldsymbol{S}}$, y entonces puede escribirse

$$\vec{\mu} = \pm\gamma \vec{\boldsymbol{S}} \,. \tag{1.64}$$

donde γ es la magnitud de la razón giromagnética. El signo positivo corresponde al espín del protón y muchos otros espines nucleares. El signo negativo corresponde al espín del electrón y algunos espines nucleares. Esto significa que la dirección del momento magnético es opuesta a la dirección del espín. Debido a que el momento magnético es proporcional al momento angular de espín, si una partícula no tiene espín entonces dicha partícula tampoco tendrá momento magnético. Es bien conocido que un campo magnético uniforme $\vec{\boldsymbol{B}}$ no produce una fuerza neta sobre un momento magnético. La fuerza actuando sobre el polo Norte positiva es balanceada por la fuerza actuando sobre el polo sur. La torca $\vec{\tau}$ producida por el campo magnético es igual a la razón de cambio de la dirección del espín

$$\vec{\tau} = \frac{\boldsymbol{d}\vec{\boldsymbol{S}}}{\boldsymbol{dt}} = \vec{\mu} \times \vec{\boldsymbol{B}} \,. \tag{1.65}$$

Multiplicando por $-\gamma$ se deriva la ecuación de movimiento cuasi-clásica para el momento magnético, que en componentes cartesianas es

$$\begin{aligned} \dot{\mu}_x &= -\gamma(\mu_y \boldsymbol{B}_z - \mu_z \boldsymbol{B}_y), \\ \dot{\mu}_y &= -\gamma(\mu_z \boldsymbol{B}_x - \mu_x \boldsymbol{B}_z), \\ \dot{\mu}_z &= -\gamma(\mu_x \boldsymbol{B}_y - \mu_y \boldsymbol{B}_x). \end{aligned} \tag{1.66}$$

Si $\vec{\boldsymbol{B}} = \boldsymbol{B}_0\,\hat{\boldsymbol{z}}$, entonces se tiene

$$\begin{aligned}\dot{\mu}_x &= -\gamma \boldsymbol{B}_0 \mu_y,\\ \dot{\mu}_y &= \gamma \boldsymbol{B}_0 \mu_x,\\ \dot{\mu}_z &= 0.\end{aligned} \tag{1.67}$$

con $\vec{\mu}(t) = \mu_x(t)\hat{x} + \mu_y(t)\hat{y} + \mu_z(t)\hat{z}$. Si volvemos a derivar las ecuaciones en (1.67) y escribimos $\omega = -\gamma \boldsymbol{B}$ (ω es conocida como frecuencia de Larmor), obtenemos

$$\begin{aligned}\ddot{\mu}_x &= -\omega^2 \mu_x,\\ \ddot{\mu}_y &= -\omega^2 \mu_y.\end{aligned} \tag{1.68}$$

Las soluciones para $\mu_x(\boldsymbol{t})$ y $\mu_y(\boldsymbol{t})$ son

$$\begin{aligned}\mu_x(\boldsymbol{t}) &= \mu_x(0)\cos\omega \boldsymbol{t} + \mu_y(0)\sin\omega \boldsymbol{t},\\ \mu_y(\boldsymbol{t}) &= \mu_y(0)\cos\omega \boldsymbol{t} - \mu_x(0)\sin\omega \boldsymbol{t},\\ \mu_z(\boldsymbol{t}) &= 0.\end{aligned} \tag{1.69}$$

Otra solución sería la siguiente: se tienen dos direcciones de equilibrio para el vector momento magnético $\vec{\mu}$, las direcciones positivas y negativas del eje *z*. Sea $\mu_z = \mu_0$, y corresponde a una energía magnética mínima:

$$\boldsymbol{U}_{\boldsymbol{m}} = -\boldsymbol{B}_0 \mu_0 . \tag{1.70}$$

Consideremos el caso $\mu_1 = \sqrt{\mu_x^2 + \mu_y^2} \neq 0$, entonces conviene definir

$$\mu_{\pm} = \mu_x \pm \boldsymbol{i}\,\mu_y . \tag{1.71}$$

y entonces las ecuaciones (1.67) pueden escribirse $\dot{\mu}_{\pm} = \pm \boldsymbol{i}\gamma \boldsymbol{B}_0 \mu_{\pm}$ cuyas soluciones son inmediatamente dadas por

$$\mu_{\pm}(\boldsymbol{t}) = \mu_{\pm}(0)\boldsymbol{e}^{\pm \boldsymbol{i}\omega_b \boldsymbol{t}} . \tag{1.72}$$

que describe un movimiento de precesión en contra del movimiento de las manecillas del reloj del momento magnético alrededor de la dirección del campo magnético; a ésta se le llama precesión de Larmor y a $\omega_b = \gamma \boldsymbol{B}_0$ frecuencia de Larmor. Se concluye entonces que las integrales de movimiento son $\mu_z = \mu_0$ y $\mu = \sqrt{\mu_x^2 + \mu_y^2 + \mu_z^2}$.

Sea el movimiento del momento magnético en la presencia de un campo magnético de radio frecuencia $\vec{\boldsymbol{B}}_1$. Sea $\vec{\boldsymbol{B}}_1$ un campo polarizado en el plano XY rotando en contra de las manecillas del reloj, es decir,

$$\begin{aligned}\boldsymbol{B}_{1x} &= \boldsymbol{B}_1 \cos\omega \boldsymbol{t},\\ \boldsymbol{B}_{1y} &= \boldsymbol{B}_1 \sin\omega \boldsymbol{t}.\end{aligned} \tag{1.73}$$

Substituyendo en la ecuación (1.66), $\vec{B} = B_0\hat{k} + \vec{B}_1$, se obtienen las ecuaciones diferenciales

$$\begin{aligned} \dot{\mu}_+ &= i\gamma(B_0\mu_+ - B_1\mu_z e^{i\omega t}), \\ \dot{\mu}_z &= i\gamma/2(B_1\mu_- e^{i\omega t} - B_1\mu_+ e^{i\omega t}). \end{aligned} \quad (1.74)$$

Para simplificar las ecuaciones se escriben en un sistema de coordenadas rotante por un ángulo $\phi = \omega t$ alrededor del eje *z* y entonces las ecuaciones anteriores para el vector de momento magnético toman la forma

$$\begin{aligned} \dot{\mu}'_+ &= i\gamma\left(B_0 - \frac{\omega}{\gamma}\right)\mu'_+ - i\gamma B_1\mu'_z, \\ \dot{\mu}'_- &= -i\gamma\left(B_0 - \frac{\omega}{\gamma}\right)\mu'_- + i\gamma B_1\mu'_z, . \\ \dot{\mu}'_z &= \frac{i\gamma}{2}B_1\left(\mu'_- - \mu'_+\right). \end{aligned} \quad (1.75)$$

$$\frac{d}{dt}\begin{pmatrix}\mu'_+\\ \mu'_z\\ \mu'_-\end{pmatrix} = i\gamma\begin{pmatrix} B_0 - \frac{\omega}{\gamma} & -B_1 & 0 \\ -\frac{B_1}{2} & 0 & \frac{B_1}{2} \\ 0 & B_1 & -\left(B_0 - \frac{\omega}{\gamma}\right)\end{pmatrix}\begin{pmatrix}\mu'_+\\ \mu'_z\\ \mu'_-\end{pmatrix}$$

que describen el movimiento de un momento magnético en un campo magnético efectivo de la forma

$$\vec{B}_{eff} = \left(B_1, 0, B_0 - \frac{\omega}{\gamma}\right). \quad (1.76)$$

De tal manera que la ecuación (1.75) describe el movimiento del vector magnético $\vec{\mu}'$ precesando alrededor del campo magnético efectivo. En el sistema de referencia del laboratorio se tiene un movimiento complicado del vector magnético $\vec{\mu}$: la precesión de Larmor alrededor del campo efectivo $\vec{B}_{eff}$, que al mismo tiempo rota alrededor del eje *z* con frecuencia ω:

1.9.1 Tratamiento mecánico cuántico

El vector de estado un sistema está dado por

$$|\psi(t)\rangle = \alpha(t)|0\rangle + \beta(t)|1\rangle = \begin{pmatrix}\alpha(t)\\ \beta(t)\end{pmatrix}, \quad (1.77)$$

donde $|0\rangle$ y $|1\rangle$ son vectores propios del operador proyección S_z con valores propios $+\hbar/2$ y $-\hbar/2$ respectivamente.

El operador Hamiltoniano $H(t)$ para un electrón en un campo magnético es $\boldsymbol{H}(\boldsymbol{t}) = -\frac{\hbar\gamma}{2}\bar{\sigma}.\beta$, y utilizando la representación matricial de las matrices de Pauli toma la forma

$$\begin{aligned}\boldsymbol{H}(\boldsymbol{t}) &= \hbar\frac{\omega_0}{2}\begin{pmatrix}1 & 0\\ 0 & -1\end{pmatrix} + \hbar\frac{\omega_1}{2}\left[\begin{pmatrix}0 & \cos\omega\boldsymbol{t}\\ \cos\omega\boldsymbol{t} & 0\end{pmatrix} + \begin{pmatrix}0 & -\boldsymbol{i}\sin\omega\boldsymbol{t}\\ \boldsymbol{i}\sin\omega\boldsymbol{t} & 0\end{pmatrix}\right]\\ &= \frac{\hbar}{2}\begin{pmatrix}\omega_0 & \omega_1\boldsymbol{e}^{-i\omega t}\\ \omega_1\boldsymbol{e}^{i\omega t} & -\omega_0\end{pmatrix}\end{aligned} \tag{1.78}$$

donde se usó $\vec{\boldsymbol{B}} = \boldsymbol{B}_0\hat{\boldsymbol{k}} + \vec{\boldsymbol{B}}_1$, $\omega_0 \equiv \gamma\boldsymbol{B}_0$, y $\omega_1 \equiv \gamma\boldsymbol{B}_1$.

Se puede resolver ahora la ecuación de Schrödinger

$$\boldsymbol{i}\hbar\frac{\boldsymbol{d}}{\boldsymbol{dt}}\left|\tilde{\psi}(\boldsymbol{t})\right\rangle = \tilde{\boldsymbol{H}}\left|\tilde{\psi}(\boldsymbol{t})\right\rangle, \tag{1.79}$$

Donde $\left|\psi(\boldsymbol{t})\right\rangle$ está definido en (1.77) y H en (1.78) describe la interacción del espín con un campo magnético variable:

$$\boldsymbol{i}\hbar\frac{\partial}{\partial\boldsymbol{t}}\begin{pmatrix}\alpha(\boldsymbol{t})\\ \beta(\boldsymbol{t})\end{pmatrix} = \frac{\hbar}{2}\begin{pmatrix}\omega_0 & \omega_1\boldsymbol{e}^{-i\omega t}\\ \omega_1\boldsymbol{e}^{i\omega t} & -\omega_0\end{pmatrix}\begin{pmatrix}\alpha(\boldsymbol{t})\\ \beta(\boldsymbol{t})\end{pmatrix}, \tag{1.80}$$

Se tiene entonces el par de ecuaciones diferenciales

$$\begin{aligned}\boldsymbol{i}\frac{\partial\alpha(\boldsymbol{t})}{\partial\boldsymbol{t}} &= \frac{\omega_0}{2}\alpha(\boldsymbol{t}) + \frac{\omega_1}{2}\boldsymbol{e}^{-i\omega t}\beta(\boldsymbol{t}),\\ \boldsymbol{i}\frac{\partial\beta(\boldsymbol{t})}{\partial\boldsymbol{t}} &= \frac{\omega_1}{2}\boldsymbol{e}^{i\omega t}\alpha(\boldsymbol{t}) - \frac{\omega_0}{2}\beta(\boldsymbol{t}).\end{aligned} \tag{1.81}$$

Se propone $\alpha(\boldsymbol{t}) = \boldsymbol{e}^{-i\frac{\omega_0}{2}t}\boldsymbol{g}_+$ y $\beta(\boldsymbol{t}) = \boldsymbol{e}^{i\frac{\omega_0}{2}t}\boldsymbol{g}_-$, obteniéndose las ecuaciones diferenciales

$$\begin{aligned}\dot{\boldsymbol{g}}_+ &= -\boldsymbol{i}\frac{\omega_1}{2}\boldsymbol{e}^{-i(\omega-\omega_0)t}\boldsymbol{g}_-,\\ \dot{\boldsymbol{g}}_- &= -\boldsymbol{i}\frac{\omega_1}{2}\boldsymbol{e}^{-i(\omega-\omega_0)t}\boldsymbol{g}_+,\end{aligned} \tag{1.82}$$

Estas ecuaciones diferenciales, con la condición lineal $\boldsymbol{g}_+(0) = 1$ y $\boldsymbol{g}_-(0) = 0$, pueden resolverse en forma inmediata:

$$\begin{pmatrix}\alpha(\boldsymbol{t})\\ \beta(\boldsymbol{t})\end{pmatrix} = \begin{pmatrix}\boldsymbol{e}^{-i\frac{\omega_0}{2}t}\cos\left(\frac{\omega_1\boldsymbol{t}}{2}\right)\\ \boldsymbol{e}^{i\frac{\omega_0}{2}t}\sin\left(\frac{\omega_1\boldsymbol{t}}{2}\right)\end{pmatrix}, \tag{1.83}$$

donde se tomó $\omega = \omega_0$. Este vector de estado describe la dinámica del espín del electrón en un tiempo arbitrario *t*.

Todos los resultados mencionados en el presente capítulo son aprovechados completamente por la teoría de la información cuántica. En los siguientes capítulos se verá cómo estos recursos pueden ser utilizados para poder realizar procesos computacionales y protocolos de comunicación.

1.10 Conclusiones del capítulo

En este capítulo exploramos los fundamentos de la Mecánica Cuántica, su lógica simbólica en general y la notación de Dirac en particular, enfatizando las nociones de espín, entrelazamiento cuántico y estados y desigualdades de Bell.

Computación cuántica

Un bit cuántico es la unidad mínima y elemental de la información cuántica, es el elemento básico de una computadora cuántica. Definimos un bit clásico como un sistema físico de dos estados distinguibles y extendemos esta definición del bit clásico para introducir el concepto de bit cuántico [1].

Para poder realizar cualquier cómputo cuántico se necesita definir un estado inicial, implementar una serie de transformaciones unitarias sobre el estado inicial y medir el estado de salida. Definimos la medición del estado de un cubit con la ayuda de una esfera de radio unidad en $\Re^3$ definida como Esfera de Bloch, donde cada punto de la esfera representa el estado de un cubit de nuestro sistema.

Se representa y se define una computadora cuántica a través del modelo de un circuito cuántico, que nos facilita el manejo del cómputo y nos ayuda a observar y a distinguir las propiedades que el cómputo cuántico hace uso de la mecánica cuántica. Con la ayuda de estos circuitos se introducen y se definen las compuertas cuánticas básicas de la computación cuántica. Se demuestra la universalidad de ciertas compuertas y se define su importancia para la computación cuántica.

Después de esto nos enfocamos a definir y desarrollar algunos de los más importantes algoritmos del cómputo cuántico. Finalmente realizamos un breve análisis de la Máquina Universal de Turing Cuántica (MUTC).

2.1 El Cubit

Un bit clásico es un sistema de dos estados distinguibles usualmente representados por *0* y *1*. Podemos representar los dos estados de un bit clásico (Cbit) por un par de vectores ortonormales de 2 dimensiones de la siguiente manera [7]:

$|0\rangle \rightarrow \begin{pmatrix} 1 \\ 0 \end{pmatrix}$ Representación del Cbit 0 como vector columna,

$|1\rangle \rightarrow \begin{pmatrix} 0 \\ 1 \end{pmatrix}$ Representación del Cbit 1 como vector columna.

Para representar los 4 estados que podemos representar con 2 Cbits los representaremos como el producto tensorial de dos pares de vectores ortogonales en 4 dimensiones.

$|0\rangle \otimes |0\rangle = |00\rangle = |0\rangle_2$,
$|0\rangle \otimes |1\rangle = |01\rangle = |1\rangle_2$,
$|1\rangle \otimes |0\rangle = |10\rangle = |2\rangle_2$,
$|1\rangle \otimes |1\rangle = |11\rangle = |3\rangle_2$.

El subíndice "2" en la última representación nos ayuda a identificar cuantos Cbits describe el vector.

Entonces podríamos describir los estados de *n* Cbits como los 2^n vectores ortonormales en 2^n dimensiones como:

$$\left|x\right\rangle_n, \quad 0 \le x < 2^n - 1 \tag{2.1}$$

dado por los *n* productos tensoriales de *n* espacios vectoriales de dos dimensiones.

La regla general de la representación de $\left|x\right\rangle_n$ es 1 en la posición X y 0 en el resto de las posiciones, así de la siguiente manera podríamos expresar el vector columna $\left|4\right\rangle_3$:

$$\left|4\right\rangle = \left|100\right\rangle = \left|1\right\rangle \otimes \left|0\right\rangle \otimes \left|0\right\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix}\begin{pmatrix} 1 \\ 0 \end{pmatrix}\begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

Si al sistema de numeración de nuestra base clásica le introducimos el valor $\sqrt{-1}$, podremos entonces lograr grandes simplificaciones entre ciertas relaciones referidas a los números reales. Entonces extendemos el conjunto de los estados de la base clásica hasta los vectores unitarios arbitrarios de todo el espacio consistente de las combinaciones lineales (superposiciones) de los estados básicos clásicos con coeficientes complejos (amplitudes). Llamaremos a esta extensión bit cuántico o cubit, entonces el estado general de un cubit es una superposición de los dos estados de la base clásica:

$$\left|\varphi\right\rangle = \alpha\left|0\right\rangle + \beta\left|1\right\rangle, \quad \alpha, \beta \in C$$
$$\left|\alpha\right|^2 + \left|\beta\right|^2 = 1.$$

Ahora el estado general de n-cubits tiene la siguiente forma:

$$\left|\varphi\right\rangle = \sum_{0 \le x < 2^n - 1} \alpha_x \left|x\right\rangle_n \tag{2.2}$$

con las amplitudes complejas restringidas sólo por la condición de normalización.

Por otro lado como un factor de fase global no afecta al estado físico, podemos escoger a α como real y positivo. Con excepción del estado base $\left|1\right\rangle$ en que $\alpha = 0$ y $\beta = 1$. Entonces el estado genérico del cubit lo podríamos escribir como:

$$|\varphi\rangle = \cos\frac{\theta}{2}|0\rangle + e^{i\phi}\sin\frac{\theta}{2}|1\rangle. \tag{2.3}$$

De tal manera que el cubit reside en un espacio vectorial parametrizado por las variables continuas α y β (θ y ϕ).

Pareciera que esta extensión nos puede llevar a almacenar una cantidad infinita de información en un solo cubit, contrario a lo que teníamos en el caso del Cbit. Sin embargo esto no es cierto ya que la mecánica cuántica nos dice que para extraer dicha información tenemos que realizar un proceso de medición y esta medición únicamente nos podrá regresar un solo bit de información.

Por lo tanto con el resultado de las mediciones sabremos que el respectivo estado cuántico se encuentra en un estado $|0\rangle$ o $|1\rangle$ con probabilidades

$$p_0 = |\alpha|^2 = \left(\cos\frac{\theta}{2}\right)^2, \quad p_1 = |\beta|^2 = \left(\sin\frac{\theta}{2}\right)^2$$

respectivamente, de acuerdo a los postulados de la mecánica cuántica.

2.2 Representación y Medición del estado de un cubit

Se puede realizar la representación geométrica de un cubit y de las transformaciones que pueden operar sobre él a través de la *Esfera de Bloch*, Fig.2.1.

Dicha esfera es representada por una esfera de radio unidad en $\Re^3$ con coordenadas Cartesianas ($x = \cos\phi\sin\theta$, $y = \sin\phi\cos\theta$, $z = \cos\theta$) donde cada punto de la esfera representa un estado puro del espacio de Hilbert de dimensión compleja 2, que en nuestro caso caracteriza a nuestro sistema de bits cuánticos.

Un vector de Bloch corresponde a un vector cuyos componentes (x, y, z) corresponden a un solo punto perteneciente a las esfera de Bloch, satisfaciendo la condición de normalización $x^2 + y^2 + z^2 = 1$.

2.3 Definición de la Esfera de Bloch

La Esfera de Bloch es una representación geométrica del espacio de estados puros de un sistema cuántico de dos niveles, a través de puntos pertenecientes a una esfera unitaria.

Sabemos que un cubit puede ser escrito de la siguiente manera generalizada:

$$|\varphi\rangle = e^{i\gamma}\left(\cos\frac{\theta}{2}|0\rangle + e^{i\phi}\sin\frac{\theta}{2}|1\rangle\right), \tag{2.4}$$

donde θ, ϕ y γ son números reales. θ y ϕ definen un punto en la *Esfera de Bloch*. $e^{i\gamma}$ es un factor que no tiene ningún efecto observable, es decir, que para cualquier valor de γ el estado del

cubit está representado por el mismo punto en la *Esfera de Bloch*.

Entonces podemos escribir el estado del cubit como

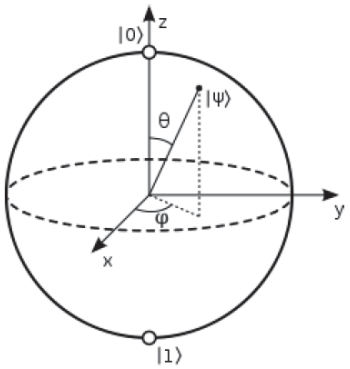


Figura 2.1: Esfera de Bloch.

$$|\varphi\rangle = \cos\frac{\theta}{2}|0\rangle + e^{i\phi}\sin\frac{\theta}{2}|1\rangle, \tag{2.5}$$

2.3.1 Deducción de la Esfera de Bloch

La *Esfera de Bloch* es una generalización de un número complejo z con $|z|^2 = 1$ como un punto en un círculo unitario.

Sea $z = x + iy \;\; | \, x, y$ pertenecen a los reales,

$$|z|^2 = \overline{z} * z = (x - iy) * (x + iy) = x^2 + y^2 \tag{2.6}$$

con $x^2 + y^2 = 1$, la ecuación de un círculo de radio uno, con centro en el origen.

En coordenadas polares para un $z = x + iy$ arbitrario, podemos escribir $x = r\cos\theta, \;\; y = r\sin\theta$, entonces

$$z = r\left(\cos\theta + i\sin\theta\right). \tag{2.7}$$

Por la Fórmula de Euler $e^{i\theta} = \cos\theta + i\sin\theta$

$$z = re^{i\theta} \tag{2.8}$$

y si es un círculo unitario $r = 1$, entonces $z = e^{i\theta}$.

Así dado el estado de un cubit general

$$|\varphi\rangle = \alpha|0\rangle + \beta|1\rangle,$$

podemos expresar dicho estado en coordenadas polares de la siguiente manera:

$$|\varphi\rangle = r_\alpha e^{i\phi_\alpha}|0\rangle + r_\beta e^{i\theta_\beta}|1\rangle. \tag{2.9}$$

Podemos multiplicar el estado por un factor de fase global $e^{i\gamma}$ sin modificar el resultado de medir el observable, esto es,

$$\left|e^{i\gamma}\alpha\right|^2 = \left|e^{i\gamma}\alpha\right| * \left|e^{i\gamma}\alpha\right| = \left(e^{-i\gamma}\alpha\right) * \left(e^{i\gamma}\alpha\right) = \alpha * \alpha = |\alpha|^2.$$

Multiplicando por $e^{-i\phi_\alpha}$ tenemos

$$|\varphi'\rangle = r_\alpha|0\rangle + r_\beta e^{i(\phi_\beta - \phi_\alpha)}|1\rangle = r_\alpha|0\rangle + r_\beta e^{i\phi}|1\rangle. \tag{2.10}$$

donde $\phi = \phi_\beta - \phi_\alpha$, además $\langle\varphi'|\varphi'\rangle = 1.$

$$|\varphi'\rangle = r_\alpha|0\rangle + (x+iy)|1\rangle \tag{2.11}$$

y la constante de normalización se obtiene de

$$|r_\alpha|^2 + |x+iy|^2 = r_\alpha^2 + (x+iy)*(x-iy) = r_\alpha^2 + x^2 + y^2 = 1, \tag{2.12}$$

que es la ecuación de la esfera unitaria en el espacio de coordenadas cartesianas.

Las coordenadas cartesianas y las coordenadas polares están relacionadas por

$$x = r\sin\theta\cos\phi,$$
$$y = r\sin\theta\sin\phi,$$
$$z = r\cos\theta,$$

En la ecuación anterior renombramos r_α como z y como $r = 1$ tenemos

$$\begin{aligned}|\varphi'\rangle &= z|0\rangle + (\sin\theta\cos\phi + i\sin\theta\sin\phi)|1\rangle,\\ &= \cos\theta|0\rangle + \sin\theta(\cos\phi + i\sin\phi)|1\rangle,\\ &= \cos\theta|0\rangle + \sin\theta e^{i\phi}|1\rangle.\end{aligned}$$

Renombramos $|\varphi'\rangle = |\varphi\rangle$ y $\theta = \theta'$

$$|\varphi\rangle = \cos\theta'|0\rangle + \sin\theta' e^{i\phi}|1\rangle. \tag{2.13}$$

Notemos que si $\theta' = 0 \Rightarrow |\varphi\rangle = |0\rangle$ y si $\theta' = \frac{\pi}{2} \Rightarrow |\varphi\rangle = e^{i\phi}|1\rangle$. Entonces si $0 \le \theta' \le \frac{\pi}{2}$ podremos generar todos los puntos de la Esfera de Bloch.

No hay necesidad de considerar ambos hemisferios de la esfera ya que dichos puntos sólo difieren por un factor de fase global -1.

Entonces definimos $\theta = 2\theta'$

$$|\varphi\rangle = \cos\frac{\theta}{2}|0\rangle + \sin\frac{\theta}{2}e^{i\phi}|1\rangle. \tag{2.14}$$

2.4 Medición

Como tenemos un gran número de cubits preparados idénticamente en principio podemos medir el estado de un cubit $|\varphi\rangle$.

Nos ayudaremos de la esfera de Bloch para realizar la medición de nuestro estado, basándonos en que podemos medir las coordenadas x, y, z de un cubit en nuestra esfera de la siguiente manera:

Usaremos las matrices de Pauli $\sigma_x = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$, $\sigma_y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}$, $\sigma_z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$, y los multiplicaremos por el estado $|\varphi\rangle$, entonces

$$\sigma_x|\varphi\rangle = e^{i\phi}\sin\frac{\theta}{2}|0\rangle + \cos\frac{\theta}{2}|1\rangle,$$
$$\sigma_y|\varphi\rangle = -ie^{i\phi}\sin\frac{\theta}{2}|0\rangle + i\cos\frac{\theta}{2}|1\rangle,$$
$$\sigma_z|\varphi\rangle = \cos\frac{\theta}{2}|0\rangle - e^{i\phi}\cos\frac{\theta}{2}|1\rangle,$$

Si calculamos el valor esperado tendremos

$$\langle\varphi|\sigma_x|\varphi\rangle = \sin\theta\cos\phi = x,$$
$$\langle\varphi|\sigma_y|\varphi\rangle = \sin\theta\sin\phi = y,$$
$$\langle\varphi|\sigma_z|\varphi\rangle = \cos\theta = z.$$

Observamos entonces que las coordenadas (x,y,z) pueden ser obtenidas con cierta precisión arbitraria.

Por otro lado tenemos:

$$p_0 - p_1 = \left(\cos\frac{\theta}{2}\right)^2 - \left(\sin\frac{\theta}{2}\right)^2 = \cos\theta = z.$$

Entonces obtenemos z por medio de la diferencia de probabilidades de encontrar 0 o 1 de una medición de σ_z, y si tenemos una gran cantidad N de sistemas idénticos preparados en el estado $|\varphi\rangle$ podremos estimar z con mayor precisión de acuerdo a nuestro valor de N.
Estimamos z como $N_0/N\text{-}N_1/N$ donde N_0 y N_1 nos indican el número de veces que obtuvimos 0 y 1 en nuestra medición.

Análogamente podemos obtener las coordenadas x; y operando mediante transformaciones de rotación apropiadas sobre el cubit.

2.5 Circuito cuántico

Definimos una computadora clásica como una cinta finita dividida en n estados o celdas, cada cinta tiene un estado inicial y un estado final distinguible. En cada celda de la cinta puede ser escrito un símbolo a partir de un alfabeto $\sum$. Una cabeza de escritura-lectura está posicionada en cada paso sobre la celda de nuestra cinta dirigida por una unidad de control. La computadora se detiene cuando la unidad de control alcanza un estado final.

El conjunto de operaciones elementales actúan sobre estados de 1 o 2 bits y pueden ser combinadas de tal manera que se pueden realizar operaciones o funciones más complejas.

Este mismo modelo puede ser representado en computación cuántica. Ahora en vez de cbits tenemos cubits y en vez de una cinta o registro clásico tendremos un registro cuántico de n estados.

El estado de una computadora cuántica de n-cubits es

$$\begin{aligned} |\varphi\rangle &= \sum_{i=0}^{2^n-1} c_i |i\rangle \\ &= \sum_{i_{n-1}=0}^{1} \cdots \sum_{i_1=0}^{1} \sum_{i_0=0}^{1} c_{i_{n-1}\ldots i_1, i_0} |i_{n-1}\rangle \otimes \ldots \otimes |i_1\rangle \otimes |i_0\rangle, \end{aligned} \tag{2.15}$$

donde $\sum |c_i|^2 = 1$ y el ket $|i\rangle$ está definido por el entero $i = i_{n-1}2^{n-1} + \ldots + i_1 2 + i_0$, con i_k denotando dígitos binarios. Entonces el estado de una computadora cuántica de *n*-cubits está construida como el producto tensorial de *n*-espacios de Hilbert de 2 dimensiones.

Se puede observar el principio de superposición en la ecuación anterior denotando que podemos registrar el estado $|i\rangle$ de nuestra base computacional en una superposición de *n*-cubits, mientras que en el caso clásico en el que tenemos *n* cbits podemos registrar o almacenar un solo entero *i*. **Por lo tanto, la cantidad de operaciones que podemos realizar con una computadora cuántica crece exponencialmente gracias a este principio de superposición, lo que aparentemente nos trae un nuevo poder de cómputo.**

Para poder realizar algún cómputo cuántico necesitamos:

1. Definir un estado inicial $|\varphi_i\rangle$ (estado fiducial, lo cual significa "referencia al origen o al cero") y preparar dicho estado en nuestra computadora cuántica.

2. Manipular u operar la función de onda con nuestra computadora cuántica a través de transformaciones unitarias U, tal que $|\varphi_f\rangle = U|\varphi_i\rangle$

3. Al finalizar nuestro algoritmo, realizar una medición estándar en nuestra base computacional, por ejemplo, medir la polarización a lo largo del eje z mediante el operador de Pauli σ_z de cada uno de los cubit.

2.6 Compuertas cuánticas de un solo cubit

Una compuerta cuántica es un circuito cuántico básico operando en un cierto número de cubits, i.e., nos ayudan a procesar uno o más cubits de acuerdo a los axiomas de la mecánica cuántica. Las compuertas cuánticas son reversibles, contrarias a la mayoría de las compuertas clásicas (compuertas lógicas) (Figura 2.2). Dichas compuertas cuánticas las representamos por medio de matrices unitarias, las que preservan la condición de normalización. Las compuertas cuánticas tendrán entonces el mismo número de cubits de entrada que de salida.

Las compuertas cuánticas de un solo cubit están representadas por matrices unitarias de 2 x 2.

$$A = \begin{pmatrix} a & b \\ c & d \end{pmatrix}$$

De manera general podemos expresar una compuerta cuántica A de un solo cubit como

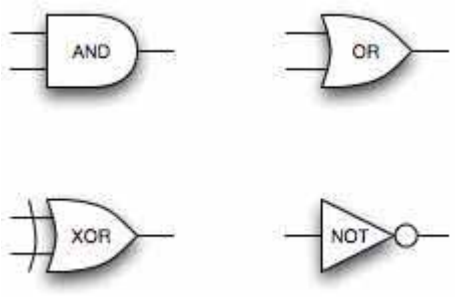


Figura 2.2: Representación clásica de compuertas Booleanas.

$$|\varphi\rangle \underline{A} |\chi\rangle. \tag{2.16}$$

Donde A es un operador lineal unitario invertible.

Entonces $A|\varphi\rangle = |\chi\rangle$

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix}\begin{pmatrix} \alpha \\ \beta \end{pmatrix}=\begin{pmatrix} \gamma \\ \delta \end{pmatrix}$$

Existen 3 maneras generales para representar esquemáticamente una compuerta cuántica en notación de circuitos (Figura 2.3), un operador arbitrario de un cubit U_1, un operador arbitrario de dos cubits U_2 y un operador de dos cubits-controlado donde U es aplicado al cubit *blanco* si el cubit control es inicializado.

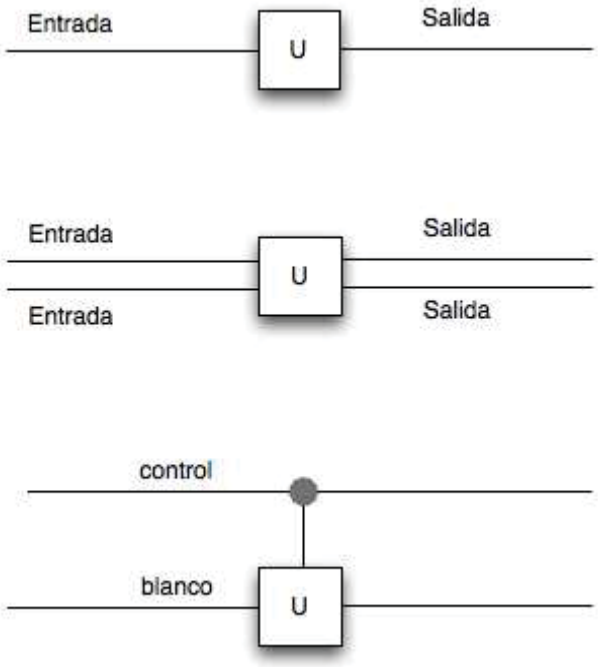


Figura 2.3: Representación de Compuertas Cuánticas.

2.6.1 Compuerta Hadamard

La compuerta de Hadamard convierte nuestra base computacional $\{|0\rangle,|1\rangle\}$ en la nueva base $\{|+\rangle,|-\rangle\}$ que forma una superposición de nuestra base computacional.

La compuerta de Hadamard es una matriz de 2 x 2 definida como

$$H=\frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}.$$

La acción de H está dada entonces por

$$H|0\rangle = \frac{1}{\sqrt{2}}\left(|0\rangle + |1\rangle\right) \equiv |+\rangle,$$
$$H|1\rangle = \frac{1}{\sqrt{2}}\left(|0\rangle - |1\rangle\right) \equiv |-\rangle. \qquad (2.17)$$

Cabe destacar que $H^2 = I$ y $H^{-1} = H$. Como H es hermítica $\Rightarrow \left(H^T\right)^* = H$.

2.6.2 Compuerta de corrimiento de fase

Esta compuerta convierte $|0\rangle$ en $|0\rangle$ y $|1\rangle$ en $e^{i\delta}|1\rangle$. Como una fase global no tiene significado físico, los estados de la base computacional $|0\rangle, |1\rangle$ no cambian.
Dicha compuerta está definida como

$$R_z(\delta) = \begin{bmatrix} 1 & 0 \\ 0 & e^{i\delta} \end{bmatrix}.$$

La acción de dicha compuerta sobre un cubit genérico $|\varphi\rangle$ se comporta de la siguiente manera:

$$R_z(\delta)|\varphi\rangle = \begin{bmatrix} 1 & 0 \\ 0 & e^{i\delta} \end{bmatrix} \begin{bmatrix} \cos\frac{\theta}{2} \\ e^{i\phi}\sin\frac{\theta}{2} \end{bmatrix} = \begin{bmatrix} \cos\frac{\theta}{2} \\ e^{i(\phi+\delta)}\sin\frac{\theta}{2} \end{bmatrix}.$$

Como fases relativas son observables, el estado del cubit ha cambiado debido a la aplicación de la compuerta de corrimiento de fase. Dicha compuerta genera una rotación en dirección contraria a las manecillas del reloj a través de un ángulo δ sobre el eje z de la *Esfera de Bloch*.

Cualquier operación unitaria sobre un solo cubit puede ser construida únicamente a partir de nuestras compuertas de Hadamard y de la compuerta de corrimiento de fase.

Toda transformación unitaria mueve el estado de un cubit de un punto sobre de la Esfera de Bloch a otro punto de la misma.

2.6.3 Rotación de la Esfera de Bloch

Otra clase de transformaciones unitarias útiles son las rotaciones de la Esfera de Bloch sobre un eje arbitrario.

Las matrices de Pauli X,Y,Z cuando son exponenciales dan como resultado operadores de rotación, que rotan el vector de Bloch $(\sin\theta\cos\phi, \sin\theta\sin\phi, \cos\theta)$, sobre el eje $\hat{x}$, $\hat{y}$ o $\hat{z}$:

$R_x(\theta) \equiv e^{-i\theta\frac{X}{2}}$,

$R_y(\theta) \equiv e^{-i\theta\frac{Y}{2}}$,

$R_z(\theta) \equiv e^{-i\theta\frac{Z}{2}}$,

Para una función exponencial tenemos

$$e^A = I + A + \frac{A^2}{2!} + \frac{A^3}{3!} + \ldots \tag{2.18}$$

Ahora consideremos $e^{i\theta A}$

$$e^{i\theta A} = I + i\theta A - \frac{(\theta A)^2}{2!} - i\frac{(\theta A)^3}{3!} + \ldots \tag{2.19}$$

En el caso en que $A^2 = I$,

$$\begin{aligned} e^{i\theta A} &= I + i\theta A - \frac{\theta^2 I}{2!} - i\frac{\theta^3 A}{3!} + \ldots \\ &= \left(1 - \frac{\theta^2}{2!} + \frac{\theta^4}{4!} + \ldots\right) I + i\left(\theta - \frac{\theta^3}{3!} + \frac{\theta^5}{5!} + \ldots\right) A \\ &\therefore\ e^{i\theta A} = \cos(\theta) I + i\sin(\theta) A \end{aligned} \tag{2.20}$$

Las matrices de Pauli tienen la propiedad que $X^2 = Y^2 = Z^2 = I$ entonces podemos evaluar los operadores de rotación con el resultado anterior:

$$\begin{aligned} R_x(\theta) \equiv e^{-i\theta\frac{X}{2}} &= \cos\frac{\theta}{2} I - i\sin\frac{\theta}{2} X = \begin{bmatrix} \cos\frac{\theta}{2} & -i\sin\frac{\theta}{2} \\ -i\sin\frac{\theta}{2} & \cos\frac{\theta}{2} \end{bmatrix}, \\ R_y(\theta) \equiv e^{-i\theta\frac{Y}{2}} &= \cos\frac{\theta}{2} I - i\sin\frac{\theta}{2} Y = \begin{bmatrix} \cos\frac{\theta}{2} & -\sin\frac{\theta}{2} \\ \sin\frac{\theta}{2} & \cos\frac{\theta}{2} \end{bmatrix}, \\ R_z(\theta) \equiv e^{-i\theta\frac{Z}{2}} &= \cos\frac{\theta}{2} I - i\sin\frac{\theta}{2} z = \begin{bmatrix} e^{-i\frac{\theta}{2}} & 0 \\ 0 & e^{i\frac{\theta}{2}} \end{bmatrix}. \end{aligned} \tag{2.21}$$

Si $\hat{n} = \left(n_x, n_y, n_z\right)$ es un vector real unitario, entonces el operador $R_{\hat{n}}(\theta) \equiv e^{-i\theta\hat{n}.\frac{\vec{\sigma}}{2}}$ rota el vector de Bloch por un ángulo θ sobre el eje $\hat{n}$ y $\vec{\sigma}$ denota el vector cuyas componente son las matrices de Pauli $X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, Y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}, Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$. Además $\left(\hat{n}.\vec{\sigma}\right)^2 = I$ entonces podremos escribir

$$\begin{aligned} R_{\hat{n}}(\theta) &= \cos\left(\frac{\theta}{2}\right) I - i \sin\left(\frac{\theta}{2}\right) \hat{n}.\vec{\sigma} \\ &= \cos\left(\frac{\theta}{2}\right) I - i \sin\left(\frac{\theta}{2}\right) \left(n_x X + n_y Y + n_z Z\right) \end{aligned} \tag{2.22}$$

De manera particular como cualquier operador es una rotación sobre nuestro cubit, entonces un cubit puede ser escrito como una combinación lineal de los operadores *I*, *X*, *Y*, *Z*.

2.7 Compuertas de control y generación de entrelazamiento

Una de las propiedades más conocidas de la mecánica cuántica, es el fenómeno de entrelazamiento cuántico, observado en sistemas cuánticos compuestos, propiedad predicha en 1935 por Einstein, Podolsky y Rosen en la formulación de la ahora conocida como paradoja EPR. Este fenómeno cuántico, sin equivalente clásico, nos dice que los estados cuánticos de dos o más objetos se deben describir haciendo referencia a los estados cuánticos de todos los objetos del sistema, es decir, los objetos están ligados de tal manera que uno no puede ser descrito adecuadamente sin una mención total del resto de los objetos, sin importar que los objetos estén separados espacialmente (Capítulo 1).

El espacio de Hilbert *H* asociado con un sistema compuesto es el producto tensorial de los espacios de Hibert H_i asociados con los componentes *i*. En el caso de un sistema cuántico bipartito tenemos

$$H = H_1 \otimes H_2.$$

Podemos construir una base del espacio de Hilbert *H* a partir del producto tensorial de los vectores base de H_1 y H_2, si nuestros espacios de Hilbert H_1 y H_2 son de dos dimensiones y tienen como vectores base, respectivamente,

$$\left\{|0\rangle_1, |1\rangle_1\right\}, \qquad \left\{|0\rangle_2, |1\rangle_2\right\},$$

entonces una base del espacio de Hilbert definido por *H* está dado por los 4 vectores:

$|0\rangle_1, |0\rangle_2,$
$|0\rangle_1, |1\rangle_2,$
$|1\rangle_1, |0\rangle_2,$
$|1\rangle_1, |1\rangle_2.$

El principio de superposición nos dice que el estado más general en el espacio de Hilbert *H* no es el producto tensorial de los estados pertenecientes a H_1 y H_2, sino una superposición arbitraria de dichos estados:

$$|\varphi\rangle = \sum_{i,j=0}^{1} c_{ij} |i\rangle_1 \otimes |j\rangle_2 = \sum_{i,j} c_{ij} |i,j\rangle. \tag{2.23}$$

donde en el estado $|i,j\rangle$, *i* se refiere al primer cubit y *j* al segundo cubit.

Por definición, un estado en *H* está entrelazado si NO puede ser escrito como un simple producto tensorial de un estado $|\alpha\rangle_1$ perteneciendo a H_1 y de un estado $|\beta\rangle_2$ perteneciente a H_2.
Si podemos escribir

$$|\varphi\rangle = |\alpha\rangle_1 \otimes |\beta\rangle_1, \tag{2.24}$$

decimos que el estado $|\varphi\rangle$ es separable.

Ejemplos:

Sea $|\varphi_1\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$. Decimos que está entrelazado, ya que no puede ser descompuesto como producto tensorial de los estados del sistema.
Sea $|\varphi_2\rangle = \frac{1}{\sqrt{2}}(|01\rangle + |11\rangle)$. Decimos que es separable, ya que puede ser escrito de la siguiente manera: $|\varphi_2\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \otimes |1\rangle$.

En general un estado separable de *n*-cubits tiene únicamente 2*n* parámetros reales mientras que el estado más general enredado tiene $2(2^n\text{-}1)$ estados.

Cabe destacar que con las compuertas cuánticas de un solo cubit no es posible generar entrelazamiento en un sistema de *n*-cubits. Además, si iniciamos desde un estado separable podemos mover a nuestro deseo cualquier cubit en la Esfera de Bloch. Cualquier estado del tipo $|\psi_i\rangle$ puede ser transformado por compuertas actuando sobre el cubit *i* en cualquier superposición de estados de nuestra base, pero el estado de nuestro sistema de *n*-cubits seguirá siendo separable.

Para preparar un estado entrelazado necesitaremos interacciones entre cubits y lo haremos con una compuerta de 2-cubits.

2.7.1 CNOT (NO-controlada)

La compuerta que es capaz de generar entrelazamiento será la compuerta CNOT (NO-controlada). Esta compuerta actúa en los estados de la base computacional, $\{|i_1\, i_0\rangle = |00\rangle, |01\rangle, |10\rangle, |11\rangle\}$.

$$CNOT\left(|x\rangle|y\rangle\right)=|x\rangle|x\otimes y\rangle,$$

con x, y = 0, 1 y $\otimes$ indicando adición módulo 2. El primer cubit en la compuerta CNOT actúa como *cubit de control* y el segundo como *cubit blanco*. La compuerta cambia el estado de nuestro *cubit blanco* si el *cubit de control* está en el estado $|1\rangle$, y no hace nada si el cubit de control se encuentra en el estado $|0\rangle$.

La compuerta CNOT puede ser descrita por el siguiente operador:

$$CNOT=|00\rangle\langle 00|+|01\rangle\langle 01|+|10\rangle\langle 11|+|11\rangle\langle 10|.$$

La representación matricial de la compuerta CNOT está dada por

$$CNOT=\begin{bmatrix}1&0&0&0\\0&1&0&0\\0&0&0&1\\0&0&1&0\end{bmatrix},$$

donde las componentes $(CNOT)_{ij}$ de esta matriz están dadas por $(CNOT)_{ij}=\langle i|CNOT|j\rangle$.

La compuerta CNOT puede ser aplicada a cualquier superposición de los estados de la base computacional.

La compuerta CNOT puede generar estados entrelazados.

Ejemplo:

$$CNOT\left(\alpha|0\rangle+\beta|1\rangle\right)|0\rangle=\alpha|00\rangle+\beta|11\rangle,$$

que es un estado no separable $(\alpha,\beta\neq 0)$.

$$A=\begin{bmatrix}1&0&0&0\\0&1&0&0\\0&0&0&1\\0&0&1&0\end{bmatrix},$$

$$B=\begin{bmatrix}0&1&0&0\\1&0&0&0\\0&0&1&0\\0&0&0&1\end{bmatrix},$$

$$C=\begin{bmatrix}1&0&0&0\\0&0&0&1\\0&0&1&0\\0&1&0&0\end{bmatrix},$$

$$D=\begin{bmatrix}0&0&1&0\\0&1&0&0\\1&0&0&0\\0&0&0&1\end{bmatrix}.$$

También podemos definir compuertas CNOT generalizadas, que dependen de la posición del cubit de control, ya sea el primero o segundo cubit.

A cambia el segundo cubit si el primero es $|1\rangle$ (CNOT estándar),

B cambia el segundo cubit si el primero es $|0\rangle$,

C cambia el primer cubit si el segundo cubit es $|1\rangle$,

D cambia el primer cubit si el segundo cubit es $|0\rangle$.

Otras compuertas a destacar son:

CPHASE: Aplica un corrimiento de fase al cubit objetivo sólo cuando el cubit de control se encuentra en el estado $|1\rangle$.

$$CPHASE(\delta)=\begin{bmatrix}1&0&0&0\\0&1&0&0\\0&0&1&0\\0&0&0&e^{i\delta}\end{bmatrix},$$

cuando $\delta=\pi$ se tiene la compuerta CMINUS.

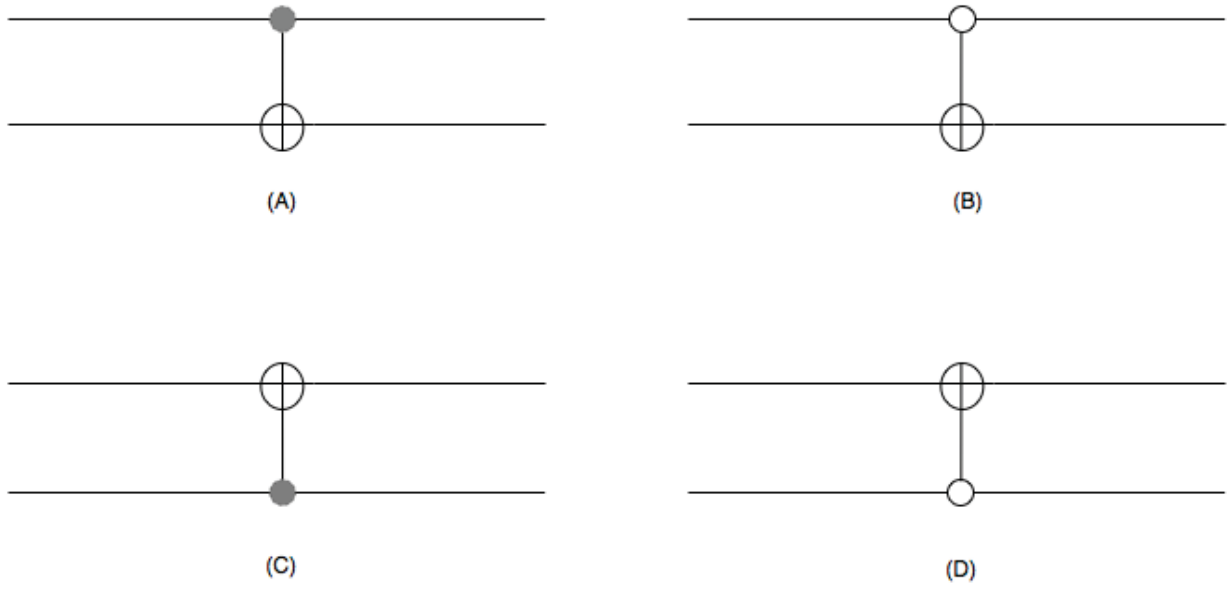


Figura 2.4: Representación de las compuertas cuánticas CNOT a través de circuitos. El cubit control es dibujado con un círculo lleno si el cubit blanco es cambiado cuando el bit control está en $|1\rangle$, un círculo vacío es dibujado cuando el blanco cambia si el bit control es $|0\rangle$.

2.7.2 Bases de Bell

Definimos los estados entrelazados Bases de Bell de la siguiente manera:

$$|\phi^+\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle),$$

$$|\phi^-\rangle = \frac{1}{\sqrt{2}}(|00\rangle - |11\rangle),$$

$$|\psi^+\rangle = \frac{1}{\sqrt{2}}(|01\rangle + |10\rangle),$$

$$|\psi^-\rangle = \frac{1}{\sqrt{2}}(|01\rangle - |10\rangle),$$

que pueden ser obtenidas a partir de la base computacional, mediante el circuito de la Figura 2.5 donde se tienen dos cubits al primero se le aplica una transformación de Hadarmard y posteriormente se utiliza la compuerta CNOT(A).

Dicho circuito produce las siguientes transformaciones:

$$AH|00\rangle = |\phi^+\rangle, \qquad AH|01\rangle = |\psi^+\rangle,$$

$$AH|10\rangle = |\phi^-\rangle, \qquad AH|11\rangle = |\psi^-\rangle,$$

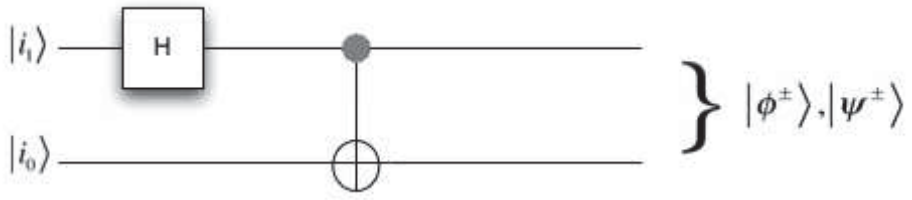


Figura 2.5: Circuito que transforma los estados de la base computacional a los estados de Bell.

2.8 Compuertas Cuánticas Universales

Una computadora clásica es capaz de generar cualquier cómputo de gran complejidad a partir únicamente de la combinación de operaciones elementales (NAND y COPY), es decir, a partir de estas operaciones podemos implementar cualquier función lógica, en esta característica radica la gran utilidad de modelo de circuito del cómputo clásico. La compuerta NAND produce salida cero si y sólo si ambas entradas son uno. La compuerta COPY convierte un bit en dos bits iguales.

En Computación Cuántica existe un resultado similar, cualquier operación unitaria en el espacio de Hilbert *H* de *n*-cubits puede descomponerse en una secuencia de compuertas de 1-qubit y de 2-qubits CNOT.

Sea *U* una transformación arbitraria sobre 1-qubit, definimos *control-U* como la operación donde *U* actúa sobre el *cubit blanco* sólo si el *cubit de control* se encuentra en $|1\rangle$:

$$|i_1\rangle|i_0\rangle \rightarrow |i_1\rangle U^{i_1}|i_0\rangle. \tag{2.25}$$

Como una matriz U es unitaria si y sólo si (*sii*) sus renglones y columnas son ortonormales, entonces cualquier matriz unitaria de 2 x 2 puede ser escrita

$$\begin{aligned} U &= \begin{bmatrix} e^{i(\alpha+\beta)/2}\cos\frac{\theta}{2} & -e^{i(\alpha-\beta)/2}\sin\frac{\theta}{2} \\ e^{i(\beta-\alpha)/2}\sin\frac{\theta}{2} & e^{-i(\alpha+\beta)/2}\cos\frac{\theta}{2} \end{bmatrix} \\ &= \begin{pmatrix} e^{i\alpha/2} & 0 \\ 0 & e^{-i\alpha/2} \end{pmatrix} \begin{pmatrix} \cos\frac{\theta}{2} & -\sin\frac{\theta}{2} \\ \sin\frac{\theta}{2} & \cos\frac{\theta}{2} \end{pmatrix} \begin{pmatrix} e^{i\beta/2} & 0 \\ 0 & e^{-i\beta/2} \end{pmatrix} \\ &= R_z(\alpha)R_y(\theta)R_z(\beta), \end{aligned}$$

donde α, β y θ son parámetros reales, y se utilizan los resultados (2.21).

Es inmediato que las matrices *R* satisfacen la propiedad de grupo, entonces $R(\theta+\phi) = R(\theta)R(\phi)$, además $R(0) = I \Rightarrow R(\theta)^{-1} = R(-\theta)$

Sea $X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$ la matriz de Pauli en la dirección del eje X que tiene la propiedad $X.R(\theta) = R(-\theta).X$.

Definimos las matrices siguientes:

$$\begin{aligned} AXBXC &= R_z(\alpha)R_y\left(\frac{\theta}{2}\right)X\,R_y\left(-\frac{\theta}{2}\right)R_z\left(-\frac{\beta+\alpha}{2}\right)X\,R_z\left(\frac{\beta-\alpha}{2}\right) \\ &= R_z(\alpha)R_y\left(\frac{\theta}{2}\right)X\,R_y\left(\frac{\theta}{2}\right)X\,R_z\left(-\frac{\beta+\alpha}{2}\right)X\,R_z\left(\frac{\beta-\alpha}{2}\right) \\ &= R_z(\alpha)R_y\left(\frac{\theta}{2}\right)R_y\left(\frac{\theta}{2}\right)R_z\left(\frac{\beta+\alpha}{2}\right)XX\,R_z\left(\frac{\beta-\alpha}{2}\right) \\ &= R_z(\alpha)R_y\left(\theta\right)R_z\left(\beta\right) \\ &= U. \end{aligned}$$

Se concluye únicamente que puede realizarse la compuerta control-U mediante el producto de transformaciones de un solo cubit (A, B, C) y compuertas de dos cubits CNOT.

2.8.1 Compuertas Toffoli

Las compuertas cuánticas, como ya hemos mencionado, operan sobre cubits de manera reversible, mientras que en la manera clásica las compuertas lógicas operan sobre bits en la mayoría de los casos de manera irreversible. Es posible construir compuertas clásicas que son reversibles; la idea general es el copiar algunos de los bits de entrada a la salida para que los bits de entrada sean recuperados a partir del resultado y de los bits de salida extras. Representaremos esta idea a través de la compuerta Toffoli.

La compuerta Toffoli tiene 3 bits de entrada y 3 bits de salida, que llamaremos $a,b,c \in \{0,1\}$, con la propiedad que aplica una operación NOT al bit blanco únicamente si los dos bits de control son uno.

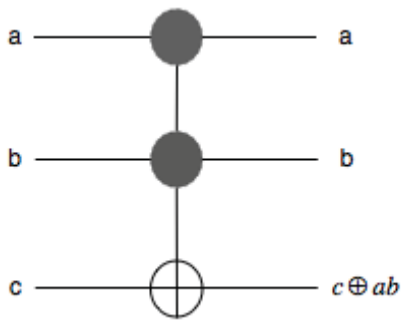


Figura 2.6: Compuerta clásica Toffoli.

En la Fig. 2.6 el término $c \oplus ab$, la operación $\oplus$ es la adición módulo 2 y ab es la multiplicación usual, i.e., aplica una operación NOT al bit objetivo sólo si los dos bits de control son 1.

Podemos escribir la compuerta Toffoli como una tabla de entrada-salida de la siguiente manera:

$(0,0,0) \mapsto (0,0,0)$,
$(0,0,1) \mapsto (0,0,1)$,
$(0,1,0) \mapsto (0,1,0)$,
$(0,1,1) \mapsto (0,1,1)$,
$(1,0,0) \mapsto (1,0,0)$,
$(1,0,1) \mapsto (1,0,1)$,
$(1,1,0) \mapsto (1,1,1)$,
$(1,1,1) \mapsto (1,1,0)$.

Es inmediato encontrar que si aplicamos 2 veces la compuerta Toffoli nos da la identidad. Entonces la compuerta Toffoli es invertible, siendo igual a su inversa. Por lo tanto la compuerta Toffoli es reversible.

Es importante notar que la redundancia en la salida de la compuerta Toffoli que reproduce idénticamente los bits a y b es la manera de evitar el borrado de información, que es una condición necesaria para permitir la reversibilidad (En 1961 Rolf Landauer descubrió que el único proceso irreversible en computación es el borrado de información).

Para demostrar que toda compuerta clásica puede ser obtenida a partir de la compuerta de Toffoli basta con demostrar que la operación NAND (la negación de la compuerta AND) puede ser construida en dicho modo, esto es, [8]

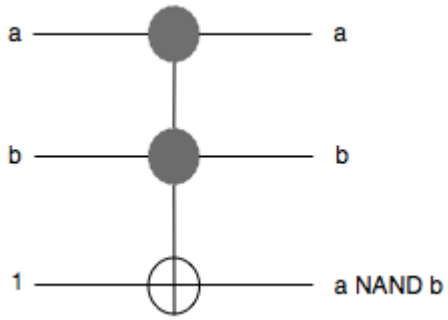


Figura 2.7: Obtención de la operación NAND a partir de la compuerta Toffoli.

La compuerta clásica Toffoli tiene también su versión cuántica. Está definida a través de la tripleta de estados $|abc\rangle$. Hablando cuánticamente la compuerta Toffoli tiene la siguiente acción

sobre la cadena de bits cuánticos: $|abc\rangle \rightarrow |ab(c \oplus ab)\rangle$. De la misma manera que la compuerta Toffoli clásica, podemos escribir una tabla de entrada-salida:

$(0,0,0) \mapsto (0,0,0)$,
$(0,0,1) \mapsto (0,0,1)$,
$(0,1,0) \mapsto (0,1,0)$,
$(0,1,1) \mapsto (0,1,1)$,
$(1,0,0) \mapsto (1,0,0)$,
$(1,0,1) \mapsto (1,0,1)$,
$(1,1,0) \mapsto (1,1,1)$,
$(1,1,1) \mapsto (1,1,0)$.

La representación por medio de circuitos de la compuerta cuántica está dada por:

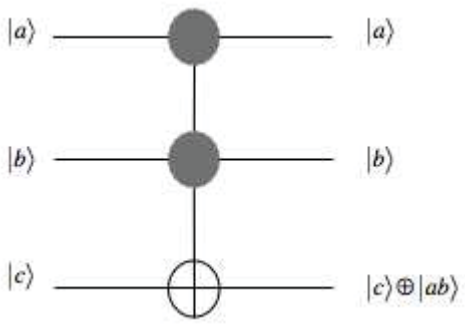


Figura 2.8: Compuerta Cuántica Toffoli.

La compuerta Toffoli es un componente básico en la implementación de casi todo algoritmo cuántico. Entonces una computadora cuántica construida de esta manera nos permite hacer cualquier cosa que una computadora clásica puede hacer. A continuación se presenta una generalización de la compuerta Toffoli (C^2 - *NOT*).

2.8.2 Compuerta C^k-U

Aplica una transformación arbitraria *U* al cubit blanco si todos los cubits de control toman el valor uno. Estas compuertas pueden ser implementadas por medio de compuertas elementales, particularmente por compuertas de un solo cubit y compuertas CNOT.

La compuerta Toffoli es un caso particular de dicha compuerta para k=2.

Además cualquier operador unitario genérico U^n actuando en un espacio de Hilbert de n-cubits puede ser descompuesto por la compuerta C^k-*U*, donde U^n puede ser descompuesto como (ver Barenco *et. al.*, 1995)

$$U^n = \prod_{i=1}^{2^n-1} \prod_{j=0}^{i-1} V_{ij},$$

donde V_{ij} induce una rotación de los estados $|i\rangle, |j\rangle$ de acuerdo a una matriz unitaria de 2x2. La idea de implementar V_{ij} en una computadora cuántica es el reducir las rotaciones de los ejes $|i\rangle$ y $|j\rangle$ a una rotación controlada en un solo cubit. [9] Las compuertas Toffoli pueden implementarse usando compuertas CNOT, control-U y Hadamard. Las compuertas C^k-U pueden descomponerse en Toffoli y compuertas control-U. Finalmente una compuerta de n-cubits puede descomponerse en compuertas C^k-U. Por lo tanto las compuertas de un solo cubit y la CNOT son compuertas universales de la computación cuántica.

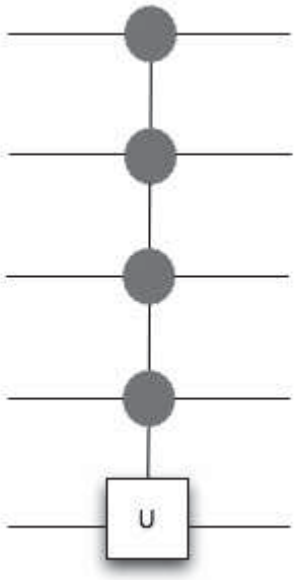


Figura 2.9: Compuerta C^k-U para $k = 4$.

2.8.3 Preparación del estado inicial

La preparación, en general, de un estado en cómputo cuántico requiere un número de compuertas que es exponencial en el número de cubits. La computadora cuántica tiene una ventaja exponencial en los requerimientos de memoria. Un vector de onda de n-cubits es determinado por 2^n números complejos y los coeficientes de su expansión definidos sobre la base computacional. Una computadora clásica necesita $m2^n$ bits para almacenar 2^n números complejos, donde m es el número de bits requerido para almacenar un número complejo con una precisión dada. La gran capacidad de memoria del cómputo cuántico permite manejar este mismo problema con solo n cubits.

A modo de ilustración se considera la construcción de un estado genérico de tres cubits del tipo $|\psi\rangle = \sum_{i=0}^{7} a_i |i\rangle$ mediante la acción de 7 rotaciones controladas sobre el estado fiducial $|000\rangle$, i.e.,

$$|\psi\rangle = \prod_{i=1}^{7} R_y(-2\Theta_i)|000\rangle.$$

donde $R_y(-2\Theta_i) = e^{-i\Theta_i\sigma_i}$. En forma más precisa se requiere una rotación en el eje *y* del primer cubit, dos rotaciones controladas por el primer cubit y de blanco el segundo cubit y finalmente 4 rotaciones del tipo $C^2 - R_y$ sobre los tres cubits. De esta manera se obtienen las amplitudes $|a_i\rangle$ de los coeficientes del desarrollo del estado de tres cubits. Para determinar las fases se aplican 4 componentes de un solo cubit de la forma

$$\Gamma_0 = \begin{bmatrix} e^{i\gamma_0} & 0 \\ 0 & e^{i\gamma_1} \end{bmatrix},$$
$$\Gamma_1 = \begin{bmatrix} e^{i\gamma_2} & 0 \\ 0 & e^{i\gamma_3} \end{bmatrix},$$
$$\Gamma_2 = \begin{bmatrix} e^{i\gamma_4} & 0 \\ 0 & e^{i\gamma_5} \end{bmatrix},$$
$$\Gamma_3 = \begin{bmatrix} e^{i\gamma_6} & 0 \\ 0 & e^{i\gamma_7} \end{bmatrix}.$$

donde $a_i = |a_i| e^{i\gamma_i}$.

Una operación en una computadora cuántica se dice que es implementada eficientemente si se requiere un número de compuertas elementales polinomiales en el número de cubits. Por ejemplo la superposición no sesgada de todos los estados de la base computacional,

$$|\psi\rangle = \sum_{i=0}^{2^n-1} \frac{1}{\sqrt{2^n}} |i\rangle, \tag{2.26}$$

es obtenida después de la aplicación de n compuertas de Hadamard (una para cada cubit) al estado $|0\rangle$.

2.8.4 Errores Unitarios

Cualquier cómputo cuántico está dado por una secuencia de compuertas cuánticas aplicadas a un estado inicial:

$$|\psi_n\rangle = \prod_{i=1}^{n} U_i |\psi_0\rangle. \tag{2.27}$$

Como los operadores unitarios forman un conjunto continuo, cualquier implementación podrá tener algún error. Supóngase que los errores son también unitarios, por lo que en vez de los operadores U_i se aplican ahora los operadores unitarios imperfectos V_i.

Sea $|\psi_i\rangle$ el estado obtenido después de *i* pasos (transformaciones unitarias V_i) se tiene que

$$|\psi_i\rangle = U_i |\psi_{i-1}\rangle$$

Si aplicamos el operador imperfecto V_i obtenemos
$V_i |\psi_{i-1}\rangle = |\psi_i\rangle + |E_i\rangle,$

donde definimos

$$|E_i\rangle = (V_i - U_i)|\psi_{i-1}\rangle.$$

Si $|\tilde{\psi}_i\rangle$ denota la función de onda del cómputo cuántico después de la aplicación de *i* transformaciones unitarias imperfectas se obtiene

$$|\tilde{\psi}_1\rangle = |\psi_1\rangle + |E_1\rangle,$$
$$|\tilde{\psi}_2\rangle = V_2 |\tilde{\psi}_1\rangle = |\psi_2\rangle + |E_2\rangle + V_2 |E_1\rangle.$$

Por lo tanto después de *n* iteraciones se obtendrá la expresión

$$|\tilde{\psi}_n\rangle = |\psi_n\rangle + |E_n\rangle + V_n |E_{n-1}\rangle + V_n V_{n-1} |E_{n-2}\rangle + \ldots + V_n V_{n-1} \ldots V_2 |E_1\rangle.$$

En el peor de los casos los errores son lineales con respecto a la longitud del cómputo cuántico. Esto permite obtener, como consecuencia de la desigualdad del triángulo, que

$$\| |\tilde{\psi}_n\rangle - |\psi_n\rangle \| \leq \sum_{k=1}^{n} \| |E_k\rangle \|.$$

Donde además hemos utilizado el hecho de que la evolución es unitaria
$\| V_i |E_{i-1}\rangle \| = \| |E_{i-1}\rangle \|.$

Podemos acotar la norma Euclideana del vector de error $|E_i\rangle$ de la siguiente manera

$$\| |E_i\rangle \| = \| (V_i - U_i)|\psi_{i-1}\rangle \| \leq \| V_i - U_i \|_{\sup}, \tag{2.28}$$

donde $\| V_i - U_i \|_{\sup}$ denota la norma superior del operador $V_i - U_i$, es decir, el eigenvalor de módulo máximo. Suponiendo que el error está acotado uniformemente en cada paso:

$$\| V_i - U_i \|_{\sup} < \varepsilon, \tag{2.29}$$

se obtiene después de la aplicación de n operadores imperfectos

$$\left\| \left| \tilde{\psi}_n \right\rangle - \left| \psi_n \right\rangle \right\| < n\varepsilon. \tag{2.30}$$

Por lo tanto los errores unitarios se acumulan en el peor de los casos en forma lineal con respecto a la longitud de cómputo. Este error toma lugar en los errores sistemáticos que se alinean en la misma dirección, mientras que errores estocásticos están aleatoriamente direccionados y tienen un crecimiento del orden de $\sqrt{n}$.

Es importante destacar que cualquier cómputo cuántico termina con una medición proyectiva sobre la base computacional, dando como salida *i* con probabilidad $p_i = \left| \langle i | \psi_n \rangle \right|^2$. En la presencia de errores unitarios, la probabilidad real se vuelve $\tilde{p}_i = \left| \langle i | \tilde{\psi}_n \rangle \right|^2$. De esta forma se relaciona la precisión de la función de onda del cómputo cuántico con la precisión del resultado de la computación cuántica.

2.9 Algoritmos Cuánticos

La compuerta de Hadamard (ver 2.6.1) es una herramienta importante para el desarrollo de algoritmos cuánticos; recordemos que esta compuerta ayuda a crear una superposición de estados. Una característica interesante de dicha compuerta es que si la aplicamos en serie actúa de manera reversible, regresándonos el estado original ($H^2 = I$).

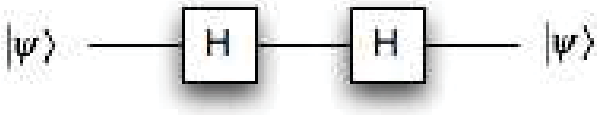


Figura 2.10: Dos compuertas de Hadamard aplicadas en serie.

Ahora, si aplicamos dicha compuerta de manera paralela (actuando sobre cada cubit) obtenemos el producto de dos estados superpuestos

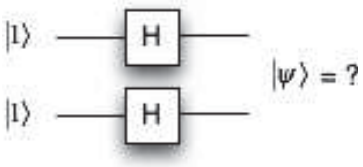


Figura 2.11: Dos compuertas de Hadamard aplicadas de manera paralela.

Por ejemplo, suponiendo el estado inicial $|1\rangle |1\rangle$, y aplicamos estas compuertas paralelamente obtenemos

$$(H\otimes H)|1\rangle|1\rangle=\big(H|1\rangle\big)\big(H|1\rangle\big)=\left(\frac{|0\rangle-|1\rangle}{\sqrt{2}}\right)\left(\frac{|0\rangle-|1\rangle}{\sqrt{2}}\right)$$
$$=\big(|00\rangle-|01\rangle-|10\rangle+|11\rangle\big) \tag{2.31}$$

Se llama *transformada de Hadamard* a la aplicación de *n* compuertas de Hadamard en paralelo sobre *n* cubits: $H^{\otimes n}$. Entonces la operación (2.31) la podríamos escribir en forma abreviada como $H^{\otimes 2}$.

Si aplicamos $H^{\otimes 3}$ al estado $|0\rangle\,|0\rangle\,|0\rangle$ obtenemos,

$$(H\otimes H\otimes H)|000\rangle=\big(H|0\rangle\big)\big(H|0\rangle\big)\big(H|0\rangle\big)$$
$$=\left(\frac{|0\rangle+|1\rangle}{\sqrt{2}}\right)\left(\frac{|0\rangle+|1\rangle}{\sqrt{2}}\right)\left(\frac{|0\rangle+|1\rangle}{\sqrt{2}}\right)$$
$$=\frac{1}{\sqrt{2^3}}\big(|000\rangle+|001\rangle+|010\rangle+|011\rangle+|100\rangle+|101\rangle+|110\rangle+|111\rangle\big).$$

Los resultados anteriores se pueden escribir de forma más compacta. Sea $|x\rangle$ un estado general donde $x=0,\ldots,2^n-1$, es decir, $|x\rangle$ es alguno de los estados de dos cubits que representamos así: $|0\rangle=|00\rangle,|1\rangle=|01\rangle,|2\rangle=|10\rangle,|3\rangle=|11\rangle$. De la misma manera si escribimos $x\in\{0,1\}$ entonces $|x\rangle$ es alguno de los estados: $|000\rangle,|001\rangle,|010\rangle,|011\rangle,|100\rangle,|101\rangle,|110\rangle,|111\rangle$. Por lo tanto si sumamos sobre la variable $|x\rangle$, se obtiene

$$(H\otimes H)|0\rangle|0\rangle=H^{\otimes 2}|0\rangle^{\otimes 2}=\frac{1}{\sqrt{2^2}}\sum_{x=0}^{3}|x\rangle. \tag{2.32}$$

$$(H\otimes H\otimes H)|0\rangle|0\rangle|0\rangle=H^{\otimes 3}|0\rangle^{\otimes 3}=\frac{1}{\sqrt{2^3}}\sum_{x=0}^{7}|x\rangle.$$

De esta manera la aplicación $H^{\otimes n}$ a un estado con n copias de $|0\rangle$ puede escribirse como

$$H^{\otimes n}|0\rangle^{\otimes n}=\frac{1}{\sqrt{2^n}}\sum_{x=0}^{2^n-1}|x\rangle. \tag{2.33}$$

De igual manera si aplicamos $H\otimes H$ al producto de estados $|0\rangle|1\rangle$ obtendremos

$$(H\otimes H)|0\rangle|1\rangle=\left(\frac{|0\rangle+|1\rangle}{\sqrt{2}}\right)\left(\frac{|0\rangle-|1\rangle}{\sqrt{2}}\right)$$
$$=\frac{1}{2}\big(|00\rangle-|01\rangle+|10\rangle-|11\rangle\big). \tag{2.34}$$

Si $x \equiv 1,2y3$ para denotar los estados respectivamente, entonces se puede escribir la ecuación de manera más compacta:

$$H^{\otimes 2}\left|01\right\rangle = \frac{1}{2}\sum_{x=0}^{3}(-1)^{x}\left|x\right\rangle.$$

Es directo escribir la acción de la compuerta de Hadamard en la base computacional para el i-ésimo cubit

$$H\left|x_i\right\rangle = \frac{1}{\sqrt{2}}\sum_{y_i=1}^{i}(-1)^{x_i y_i}\left|y_i\right\rangle.$$

Entonces la acción en paralelo de *n* compuertas de Hadamard sobre un estado de *n* cubits

$$\left|x\right\rangle = \left|x_{n-1}x_{n-2},\ldots,x_0\right\rangle,$$

puede escribirse como

$$\begin{aligned} H^{\otimes n}\left|x\right\rangle &= \prod_{i=0}^{n-1}\left\{\frac{1}{\sqrt{2}}\sum_{y_i=0}^{1}(-1)^{x_i y_i}\left|y_i\right\rangle\right\} \\ &= \frac{1}{\sqrt{2^n}}\sum_{y=0}^{2^n-1}(-1)^{x.y}\left|y\right\rangle, \end{aligned} \tag{2.35}$$

donde $x.y = \left(x_{n-1}y_{n-1} \oplus x_{n-2}y_{n-2} \oplus \ldots \oplus x_0 y_0\right)$ denota el producto escalar de *x* y *y* en base 2.

Es inmediato probar que (2.35) es una generalización de los resultados obtenidos en las expresiones (2.32), (2.33) y (2.34).

2.9.1 Interferencia Cuántica

La aplicación de una compuerta de Hadamard a un cubit arbitrario es un buen ejemplo para ilustrar el fenómeno de interferencia cuántica.

Si calculamos $H\left|\psi\right\rangle$ para $\left|\psi\right\rangle = \alpha\left|0\right\rangle + \beta\left|1\right\rangle$ obtenemos

$$H\left|\psi\right\rangle = \left(\frac{\alpha+\beta}{\sqrt{2}}\right)\left|0\right\rangle + \left(\frac{\alpha-\beta}{\sqrt{2}}\right)\left|1\right\rangle. \tag{2.36}$$

Notemos que la amplitud de probabilidad de obtener $\left|0\right\rangle$ después de una medición ha cambiado como sigue:

$$\alpha \rightarrow \frac{\alpha+\beta}{\sqrt{2}},$$

y la amplitud de probabilidad de obtener $|1\rangle$ sufre la transformación:

$$\beta \to \frac{\alpha - \beta}{\sqrt{2}}.$$

Existen 2 tipos de interferencia: *interferencia positiva*, en donde las amplitudes de las probabilidades se suman constructivamente para aumentarse o *interferencia negativa* donde las amplitudes de las probabilidades decrecen.

Por ejemplo si aplicamos H a $|\psi\rangle = \left(|0\rangle + (-1)^x |1\rangle\right)/\sqrt{2}$, donde $x \in \{0,1\}$ obtenemos

$$H|\psi\rangle = \left(\frac{1+(-1)^x}{2}\right)|0\rangle + \left(\frac{1-(-1)^x}{2}\right)|1\rangle.$$

Por lo tanto se concluye que

$$H|\psi\rangle = |0\rangle \qquad si\ x = 0,$$
$$H|\psi\rangle = |1\rangle \qquad si\ x = 1.$$

Entonces observamos lo siguiente en (2.36) para $x = 0$:

- *Interferencia positiva* sobre el estado $|0\rangle$, las dos amplitudes se suman para aumentar la probabilidad de encontrar el estado $|0\rangle$ durante la medición. En este caso la probabilidad se vuelve unitaria.
- *Interferencia negativa* sobre $|1\rangle$, vamos de un estado donde teníamos 50% de probabilidad de encontrar el estado $|1\rangle$ a otro donde la probabilidad de encontrar el estado $|1\rangle$ es nula.

La interferencia cuántica como veremos permite obtener información sobre las propiedades globales de una función *f(x)*, donde *f(x)* es una función lógica binaria que tiene una entrada de *n*-cubits (*x*) y una salida de un cubit, $\{0,1\}$.

2.9.2 Algoritmo de Deutsch

Permite determinar si una función lógica de 2 cubits es constante o balanceada.

Una función básica de una computadora clásica es la evaluación de una función lógica con *n* bits de entrada y un bit de salida, esto es

$$f : \{0,1\}^n \to \{0,1\}.$$

Definimos una *función balanceada* como aquella cuyos valores de salida pueden ser opuestas para la mitad de sus entradas. Como ejemplo se tiene la *función swap y la función identidad*.

Una función de un solo bit la definimos como función constante, si ocurre que *f(x)* = 0 o *f(x)* = 1 . Si una función es balanceada o constante es una propiedad global. El algoritmo de Deustsch ayudará a saber si una función de un cubit es una función constante o una función balanceada.

El primer paso es denotar un operador unitario U_f que actúe sobre dos cubits, con la propiedad de que es la función identidad en el primer cubit y produzca una compuerta OR exclusiva del segundo qubit con una función *f* que usa al primer cubit como argumento,

$$U_f\left|x_i, y_i\right\rangle = \left|x_i, y_i \oplus f(x_i)\right\rangle, \quad x_i, y_i \in \{0,1\}. \tag{2.37}$$

Como $\left|x_i\right\rangle$ es un cubit, entonces puede estar en un estado superpuesto.

El algoritmo de Deutsch utiliza los resultados anteriores para explotar el que un estado se encuentre en una superposición para obtener información sobre la propiedad global de una función. El procedimiento es el siguiente:

$$\left|\psi\right\rangle_{salida} = (H \otimes I) U_f \, (H \otimes H)\left|0\right\rangle\left|1\right\rangle.$$

Descripción del algoritmo de Deutsch

1. Aplicar las compuertas Hadamard al estado inicial $\left|0\right\rangle\left|1\right\rangle$ para producir el producto de estados de dos superposiciones.
2. Aplicar U_f al estado obtenido.
3. Aplicar una compuerta Hadamard al primer qubit dejando libre el segundo cubit.

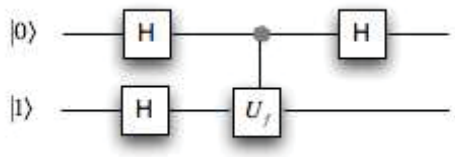


Figura 2.12: Circuito Cuántico del Algoritmo de Deutsch.

Usando (2.34) sabemos el resultado del paso 1

$$(H \otimes H)\left|0\right\rangle\left|1\right\rangle = \frac{1}{2}\left(\left|00\right\rangle - \left|01\right\rangle + \left|10\right\rangle - \left|11\right\rangle\right),$$

obteniéndose entonces una superposición de estados.

Ahora apliquemos U_f a cada término del resultado obtenido en el paso 1.

Para el primer término tenemos

$$U_f\left|00\right\rangle = \left|0, 0 \oplus f(0)\right\rangle = (1 - f(0))\left|00\right\rangle + f(0)\left|01\right\rangle.$$

Este resultado toma en cuenta las posibilidades $0 \oplus f(0) = 0$ y $0 \oplus f(0) = 1$. Notemos que si $f(0) = 0$, entonces $0 \oplus f(0) = 0 + 0 = 0$.

Por otro lado, si $f(0) = 1$, entonces $0 \oplus f(0) = 0 + 1 = 1$.

Similarmente para el resto de los términos, se obtiene

$$U_f|01\rangle = |0,1 \oplus f(0)\rangle = f(0)|00\rangle + (1 - f(0))|01\rangle,$$
$$U_f|10\rangle = |1,0 \oplus f(1)\rangle = f(1)|11\rangle + (1 - f(1))|10\rangle,$$
$$U_f|11\rangle = |1,1 \oplus f(1)\rangle = f(1)|10\rangle + (1 - f(1))|11\rangle.$$

Utilizando los resultados anteriores se tiene que

$$|\psi'\rangle = U_f (H \otimes H)|0\rangle|1\rangle$$
$$= \frac{1}{\sqrt{2}}\left\{\left(\frac{1}{2} - f(0)\right)|0\rangle + \left(\frac{1}{2} - f(1)\right)|1\rangle\right\}\left\{\frac{|0\rangle - |1\rangle}{\sqrt{2}}\right\}.$$

Para obtener la salida final del algoritmo de Deutsch aplicamos $H \otimes I$ a $|\psi'\rangle$, esto es la compuerta de Hadamard es aplicada al primer cubit, y el segundo cubit es dejado libre.

Por lo tanto aplicando $H \otimes I$ a los términos de $|\psi'\rangle$ obtenemos el estado final del algoritmo de Deutsch:

$$|\psi_{salida}\rangle = (1 - f(0) - f(1))|0\rangle\left(\frac{|0\rangle - |1\rangle}{\sqrt{2}}\right) + (f(1) - f(0))|1\rangle\left(\frac{|0\rangle - |1\rangle}{\sqrt{2}}\right) \quad (2.38)$$

Ahora, suponiendo que la función es constante, tal que *f(0) = f(1)*, entonces usando (2.38) obtenemos dos posibilidades para el estado final de salida

$$|\psi_{salida}\rangle = \pm|0\rangle\left(\frac{|0\rangle - |1\rangle}{\sqrt{2}}\right).$$

Ahora, si $f(0) \neq f(1)$ se tiene que $(1 - f(0) - f(1)) = 0$ y el estado final está dado por las expresiones

$$|\psi_{salida}\rangle = \pm|1\rangle\left(\frac{|0\rangle - |1\rangle}{\sqrt{2}}\right).$$

Podemos observar cómo la aplicación de interferencia cuántica nos ayuda a distinguir entre los dos casos de salidas de la función. De tal manera que si se mide el primer cubit y obtenemos 0 se tiene una función constante y si se obtiene 1 es una función balanceada.

2.9.3 Algoritmo Deutsch-Jozsa

El algoritmo de Deutsch-Jozsa es una generalización del algoritmo de Deutsch. Nos permite deducir si una función es constante o balanceada pero para una función con múltiples valores de entrada, esto es una función de *n* cubits.

Si *f(x)* es constante entonces los valores de salida son los mismos para todas las *x*. Si *f(x)* es balanceada entonces *f(x)* = 0 para la mitad de las entradas y *f(x)* = 1 para la otra mitad de las entradas.

Fase de reinicio

Consideremos la compuerta U_f de 2 cubits que definimos en el algoritmo de Deutsch (2.37):

$$U_f|x,y\rangle = |x, y \oplus f(x)\rangle, \quad x,y \in \{0,1\}.$$

Cambiemos nuestro registro blanco $|y\rangle$ por $\left(\frac{|0\rangle - |1\rangle}{\sqrt{2}}\right)$, y analicemos la acción de nuestro operador sobre un estado base en el cubit control:

$$\begin{aligned} U_f|x\rangle\left(\frac{|0\rangle - |1\rangle}{\sqrt{2}}\right) &= \left(\frac{|x\rangle|0 \oplus f(x)\rangle - |x\rangle|1 \oplus f(x)\rangle}{\sqrt{2}}\right), \\ &= |x\rangle\left(\frac{|0 \oplus f(x)\rangle - |1 \oplus f(x)\rangle}{\sqrt{2}}\right). \end{aligned} \tag{2.39}$$

Evaluemos la expresión $\left(\frac{|0 \oplus f(x)\rangle - |1 \oplus f(x)\rangle}{\sqrt{2}}\right)$ en los casos donde *f*(*x*) = 0 y *f*(*x*) = 1:

$$f(x) = 0: \qquad \left(\frac{|0\rangle - |1\rangle}{\sqrt{2}}\right),$$

$$f(x) = 1: \qquad \left(\frac{|1\rangle - |0\rangle}{\sqrt{2}}\right) = (-1)\left(\frac{|0\rangle - |1\rangle}{\sqrt{2}}\right).$$

Estos dos resultados difieren por un factor (-1) que depende únicamente por el valor de *f*(*x*): Tenemos entonces

$$\left(\frac{|0 \oplus f(x)\rangle - |1 \oplus f(x)\rangle}{\sqrt{2}}\right) = (-1)^{f(x)}\left(\frac{|0\rangle - |1\rangle}{\sqrt{2}}\right).$$

Asociando el factor $(-1)^{f(x)}$ con el primer cubit el estado (2.39) puede reescribirse como

$$U_f|x\rangle\left(\frac{|0\rangle-|1\rangle}{\sqrt{2}}\right)=(-1)^{f(x)}|x\rangle\left(\frac{|0\rangle-|1\rangle}{\sqrt{2}}\right).$$

Cuando el cubit de control se encuentra en una superposición de $|0\rangle$ y $|1\rangle$ tenemos

$$U_f\left(\alpha_0|0\rangle+\alpha_1|1\rangle\right)\left(\frac{|0\rangle-|1\rangle}{\sqrt{2}}\right)=\left((-1)^{f(0)}\alpha_0|0\rangle+(-1)^{f(1)}\alpha_1|1\rangle\right)\left(\frac{|0\rangle-|1\rangle}{\sqrt{2}}\right)$$
$$=\sum_{x=0}^{1}(-1)^{f(x)}\alpha_x|x\rangle\left(\frac{|0\rangle-|1\rangle}{\sqrt{2}}\right).$$

De la misma manera, si tenemos una superposición de estados dada por
$H^{\otimes n}|0\rangle^{\otimes n}=1/\sqrt{2^n}\sum_{x=0}^{2^n-1}|x\rangle$ obtenemos

$$U_f\frac{1}{\sqrt{2^n}}\sum_{x=0}^{2^n-1}|x\rangle\left(\frac{|0\rangle-|1\rangle}{\sqrt{2}}\right)=\frac{1}{\sqrt{2^n}}\sum_{x=0}^{2^n-1}(-1)^{f(x)}|x\rangle\left(\frac{|0\rangle-|1\rangle}{\sqrt{2}}\right).$$

Algoritmo

Iniciamos con un estado inicial con n cubits en el estado $|0\rangle$ y un solo cubit en el estado $|1\rangle$. Aplicamos las *n* compuertas de Hadamard a todos los cubits.

$$|\psi'\rangle=H^{\otimes n}|0\rangle^{\otimes n}\otimes\left(H|1\rangle\right).$$

De (2.9.3) sabemos que

$$|\psi'\rangle=\frac{1}{\sqrt{2^n}}\sum_{x\in\{0,1\}^n}|x\rangle\left(\frac{|0\rangle-|1\rangle}{\sqrt{2}}\right).$$

Ahora aplicamos U_f para evaluar la función. Los primeros *n* cubits son los valores de *x* y el último cubit es el valor de *y*. La salida de la compuerta U_f nos da

$$|\psi''\rangle=\frac{1}{\sqrt{2^n}}\sum_{x}(-1)^{f(x)}|x\rangle\left(\frac{|0\rangle-|1\rangle}{\sqrt{2}}\right).$$

Aplicando la compuerta de Hadamard en un estado de *n* cubits se obtiene el resultado (2.9.5)

$$H^{\otimes n}|x\rangle=\frac{1}{\sqrt{2^n}}\sum_{y}(-1)^{x.y}|y\rangle,$$

donde $x.y$ denota el producto interno de *x* y de *y mod* 2 i.e., $x.y=x_{n-1}y_{n-1}+\ldots+x_0y_0$.
El estado final da

$$|\psi_{salida}\rangle = \frac{1}{2^n}\sum_y\sum_x (-1)^{x.y+f(x)}|y\rangle\left(\frac{|0\rangle-|1\rangle}{\sqrt{2}}\right).$$

Finalmente se miden las *n* entradas y existen dos posibles resultados de mediciones sobre $|y\rangle$, que es el estado de *n* entradas en este momento. Los posibles resultados son los siguientes:

- Si la medición de los primeros *n* cubits da el estado $|000\ldots0\rangle$ con probabilidad uno, entonces la función es constante mientras que si la probabilidad es cero se tiene una función balanceada.
- Lo anterior se debe a que $\sum_{x=0}^{2^n-1}(-1)^{f(x)} = \pm 1$ si *f* es constante y 0 si *f* es balanceada.

Sea $|\psi_{inicial}\rangle = |001\rangle$ y apliquemos directamente la función de salida obtenida, entonces:

$$|\psi_s\rangle = \frac{1}{2^2}\sum_y\left((-1)^{f(0)}+(-1)^{f(1)+y_0}+(-1)^{f(2)+y_1}+(-1)^{f(3)+y_1+y_0}\right)|y\rangle\left(\frac{|0\rangle-|1\rangle}{\sqrt{2}}\right)$$

$$= \frac{1}{4}\left\{\left((-1)^{f(0)}\left(|0\rangle+|1\rangle+|2\rangle+|3\rangle\right)+(-1)^{f(1)}\left(|0\rangle-|1\rangle+|2\rangle-|3\rangle\right)\right.\right.$$

$$\left.\left.+(-1)^{f(2)}\left(|0\rangle+|1\rangle-|2\rangle-|3\rangle\right)+(-1)^{f(3)}\left(|0\rangle-|1\rangle-|2\rangle+|3\rangle\right)\right).\left(\frac{|0\rangle-|1\rangle}{\sqrt{2}}\right)\right\}$$

$$= \left\{\frac{1}{4}\left[(-1)^{f(0)}+(-1)^{f(1)}+(-1)^{f(2)}+(-1)^{f(3)}\right]|0\rangle\right.$$

$$+\frac{1}{4}\left[(-1)^{f(0)}-(-1)^{f(1)}+(-1)^{f(2)}-(-1)^{f(3)}\right]|1\rangle$$

$$+\frac{1}{4}\left[(-1)^{f(0)}+(-1)^{f(1)}-(-1)^{f(2)}-(-1)^{f(3)}\right]|2\rangle$$

$$\left.+\frac{1}{4}\left[(-1)^{f(0)}-(-1)^{f(1)}-(-1)^{f(2)}+(-1)^{f(3)}\right]|3\rangle\right\}\frac{1}{\sqrt{2}}\left(|0\rangle-|1\rangle\right).$$

Si *f* es constante entonces se obtiene $|0\rangle$; si no es constante, uno de los cubits restantes da el valor 1.

Ejemplo: Sea *f*(*x*) = 1 y $|\psi_{inicial}\rangle = |001\rangle$. Demostraremos directamente como se utiliza el algoritmo de Deutsch-Jozsa.

Aplicamos a cada uno de los cubits del estado inicial la compuerta Hadamard, obteniendo

$$|\psi'\rangle = \frac{1}{2\sqrt{2}}\left(|000\rangle-|001\rangle+|010\rangle-|011\rangle+|100\rangle-|101\rangle+|110\rangle-|111\rangle\right).$$

Aplicamos U_f:

$$|\psi''\rangle = \frac{1}{2\sqrt{2}}\left(|001\rangle - |000\rangle + |011\rangle - |010\rangle + |101\rangle - |100\rangle + |111\rangle - |110\rangle\right).$$

Finalmente aplicamos H^2 a los primeros 2 cubits:

$$\begin{aligned}|\psi_{salida}\rangle = \frac{1}{2\sqrt{2}}\Bigg(&\left(\frac{|0\rangle+|1\rangle}{\sqrt{2}}\right)\left(\frac{|0\rangle+|1\rangle}{\sqrt{2}}\right)|1\rangle - \left(\frac{|0\rangle+|1\rangle}{\sqrt{2}}\right)\left(\frac{|0\rangle+|1\rangle}{\sqrt{2}}\right)|0\rangle \\ &+\left(\frac{|0\rangle+|1\rangle}{\sqrt{2}}\right)\left(\frac{|0\rangle-|1\rangle}{\sqrt{2}}\right)|1\rangle - \left(\frac{|0\rangle+|1\rangle}{\sqrt{2}}\right)\left(\frac{|0\rangle-|1\rangle}{\sqrt{2}}\right)|0\rangle \\ &+\left(\frac{|0\rangle-|1\rangle}{\sqrt{2}}\right)\left(\frac{|0\rangle+|1\rangle}{\sqrt{2}}\right)|1\rangle - \left(\frac{|0\rangle-|1\rangle}{\sqrt{2}}\right)\left(\frac{|0\rangle+|1\rangle}{\sqrt{2}}\right)|0\rangle \\ &+\left(\frac{|0\rangle-|1\rangle}{\sqrt{2}}\right)\left(\frac{|0\rangle-|1\rangle}{\sqrt{2}}\right)|1\rangle - \left(\frac{|0\rangle-|1\rangle}{\sqrt{2}}\right)\left(\frac{|0\rangle-|1\rangle}{\sqrt{2}}\right)|0\rangle\Bigg).\end{aligned}$$

Expandiendo los términos obtenemos

$$\begin{aligned}|\psi_{salida}\rangle = \frac{1}{4\sqrt{2}}\Big(&\left(|00\rangle+|01\rangle+|10\rangle+|11\rangle\right)|1\rangle - \left(|00\rangle+|01\rangle+|10\rangle+|11\rangle\right)|0\rangle \\ &+\left(|00\rangle-|01\rangle+|10\rangle-|11\rangle\right)|1\rangle - \left(|00\rangle-|01\rangle+|10\rangle-|11\rangle\right)|0\rangle \\ &+\left(|00\rangle+|01\rangle-|10\rangle-|11\rangle\right)|1\rangle - \left(|00\rangle+|01\rangle-|10\rangle-|11\rangle\right)|0\rangle \\ &+\left(|00\rangle-|01\rangle-|10\rangle+|11\rangle\right)|1\rangle - \left(|00\rangle-|01\rangle-|10\rangle+|11\rangle\right)|0\rangle\Big).\end{aligned}$$

Factorizamos estas funciones para ponerlas en la forma del tercer cubit $\left(|0\rangle - |1\rangle\right)/\sqrt{2}$

$$\begin{aligned}|\psi_{salida}\rangle = \frac{1}{4}\Big(&-\left(|00\rangle+|01\rangle+|10\rangle+|11\rangle\right)\left(|0\rangle-|1\rangle\right)/\sqrt{2} \\ &-\left(|00\rangle-|01\rangle+|10\rangle-|11\rangle\right)\left(|0\rangle-|1\rangle\right)/\sqrt{2} \\ &-\left(|00\rangle+|01\rangle-|10\rangle-|11\rangle\right)\left(|0\rangle-|1\rangle\right)/\sqrt{2} \\ &-\left(|00\rangle-|01\rangle-|10\rangle+|11\rangle\right)\left(|0\rangle-|1\rangle\right)/\sqrt{2}\Big).\end{aligned}$$

Por lo tanto

$$|\psi_{salida}\rangle = -|00\rangle\left(\frac{|0\rangle-|1\rangle}{\sqrt{2}}\right).$$

La medida sobre los primeros dos cubits nos da el estado $|00\rangle$ con probabilidad uno, confirmando que nuestra función es una función constante.

2.9.4 Transformada de Fourier Cuántica

La Transformada de Fourier Discreta (TFD) es un caso particular de la Transformada de Fourier para secuencias de longitud finita en que se evalúa el espectro (Un espectro de frecuencias es el gráfico que muestra cómo es la descomposición de una señal ondulatoria - sonora, luminosa, electromagnética,... - en el dominio frecuencial) solamente en unas frecuencias concretas, obteniendo un espectro discreto. La TFD tiene aplicaciones en la física, la teoría de los números, la combinatoria, el procesamiento de señales (electrónica), la teoría de la probabilidad, la estadística, la óptica, la propagación de ondas y otras áreas.

La Transformada de Fourier Discreta de una función discreta $f_0,\ldots,f_{N-1}$ está dada por

$$\tilde{f}_k \equiv \frac{1}{\sqrt{N}}\sum_{j'=0}^{N-1} e^{2\pi i j' k/N} f_j.$$

La transformada inversa

$$f_k \equiv \frac{1}{\sqrt{N}}\sum_{j=0}^{N-1} e^{-2\pi i j k/N} \tilde{f}_j.$$

Se puede verificar que substituyendo la expresión anterior en $\tilde{f}_k$ se obtiene una identidad, esto es $\tilde{f}_k \equiv 1/\sqrt{N}\sum_{j=0}^{N-1} e^{2\pi i j k/N} f_j \equiv \tilde{f}_k$. Demostración:

$$\begin{aligned}\tilde{f}_k &\equiv \frac{1}{\sqrt{N}}\sum_{j'=0}^{N-1} e^{\frac{2\pi i}{N} j' k}\left(\frac{1}{\sqrt{N}}\sum_{j=0}^{N-1} e^{-\frac{2\pi i}{N} j.j'}\tilde{f}_j\right)\\ &= \frac{1}{N}\sum_{j.j'}^{N-1} e^{\frac{2\pi i}{N} j'(k-j)}\tilde{f}_j,\end{aligned}$$

donde del resultado $\sum_j e^{\frac{2\pi i}{N} j'(k-j)} = N\delta_{kj}$, se obtiene

$$\tilde{f}_k = \frac{1}{N}\left(N\sum_{j=0}^{N-1}\delta_{kj}\tilde{f}_j\right) = \tilde{f}_k,$$

que es lo que se quería demostrar.

Puede definirse la TFD cuántica como un operador lineal que actúa sobre las amplitudes del sistema cuántico, esto es,

$$\sum_j \alpha_j |j\rangle \rightarrow \sum_k \tilde{\alpha}_k |k\rangle,$$

donde

$$\tilde{\alpha}_k \equiv \frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} e^{2\pi ijk/N} \alpha_j .$$

Denotando la TDF por el operador $\hat{F}$, tenemos que el estado cuántico transformado está dado por:

$$|\tilde{\psi}\rangle = \hat{F}|\psi\rangle .$$

Observamos que las amplitudes $\tilde{\alpha}_j$ son lineales en el α_j original. Por lo tanto existe un operador lineal $\hat{F}$, que implementa la transformada, y podemos escribirla

$$\hat{F} \equiv \sum_{j,k=0}^{N-1} \frac{e^{2\pi ijk/N}}{\sqrt{N}} |k\rangle\langle j| .$$

A continuación mostramos que

$$|\psi\rangle \to |\tilde{\psi}\rangle ,$$

$$\begin{aligned} F|\psi\rangle &= \sum_{j,k=0}^{N-1} e^{2\pi ijk/N} |k\rangle\langle j| \left(\sum_{j'=0}^{N-1} \alpha_{j'i} |j'\rangle \right) \\ &= \sum_{k=0}^{N-1} \left(\sum_{j=0}^{N-1} e^{2\pi ijk/N} \alpha_i \right) |k\rangle \\ &= \sum_{k=0}^{N-1} \tilde{\alpha}_k |k\rangle = |\tilde{\psi}\rangle . \end{aligned}$$

Falta chequear que $\hat{F}$ es unitaria. Tomando el hermítico conjugado de $\hat{I}$ se tiene

$$\hat{F}^{\dagger} \equiv \sum_{j,k=0}^{N-1} \frac{e^{-2\pi ijk/N}}{\sqrt{N}} |j\rangle\langle k| ,$$

y ahora efectuando el producto con $\hat{I}$ se tiene

$$\begin{aligned} \hat{F}^{\dagger}\hat{F} &= \frac{1}{N} \sum_{j,kj'} e^{2\pi i(j'-j)k/N} |j\rangle\langle j'| , \\ &= \frac{1}{N} \sum_{j,j} |j\rangle\langle j'| \delta_{jj'} N , \\ &= \sum_{j} |j\rangle\langle j| = \hat{I} . \end{aligned}$$

De la misma forma puede probarse que $\hat{F}\hat{F}^{\dagger} = I$. Esta transformación es unitaria, por lo tanto, puede ser implementada por una computadora cuántica.

Se puede construir ahora el circuito cuántico de la Transformada de Fourier Cuántica, por medio de productos de estados.

Se ha encontrado que al actuar la Transformada de Fourier sobre un estado de *n* cubits da

$$\widehat{F}\left(\left|j\right\rangle\right)=\frac{1}{\sqrt{2^n}}\sum_{k=0}^{2^n-1}e^{\frac{2\pi i}{2^n}jk}\left|k\right\rangle.$$

Substituyendo en la expresión anterior $k=k_{n-1}\ldots k_0=k_{n-1}2^{n-1}+\ldots+k_0 2^0$ donde $k_l\in\{0,1\}$ con $l=0,1,\ldots,n-1$, se obtiene que

$$\widehat{F}\left(\left|j\right\rangle\right)=\frac{1}{\sqrt{2^n}}\sum_{k_{n-1}=0}^{1}\ldots\sum_{k_0=0}^{1}e^{2\pi ij\sum_{l=1}^{n}\frac{k_{n-l}}{2^l}}\left|k_{n-l}\right\rangle.$$

Utilizando que la exponencial de una suma es el producto de las exponenciales resulta

$$\begin{aligned}\widehat{F}\left(\left|j\right\rangle\right)&=\frac{1}{\sqrt{2^n}}\sum_{k_{n-1}=0}^{1}\ldots\sum_{k_0=0}^{1}\prod_{l=1}^{n}e^{2\pi ij\frac{k_{n-l}}{2^l}}\left|k_{n-l}\right\rangle\\&=\frac{1}{\sqrt{2^n}}\prod_{l=1}^{n}\left[\sum_{k_{n-1}=0}^{1}e^{2\pi ij\frac{k_{n-l}}{2^l}}\left|k_{n-l}\right\rangle\right].\end{aligned}$$

donde usamos que $\sum_{k_{n-1}}\sum_{k_0}\prod_{l=1}^{n}\rightarrow\prod_{l=1}^{n}\sum_{k_{n-l=0}}^{1}$

Finalmente se define la representación de una fracción binaria, esto es,

$$0.j_l j_{l+1}\ldots j_m=\frac{1}{2}j_l+\frac{1}{2^2}j_{l+1}+\ldots+\frac{1}{2^{m-l+1}}j_m.$$

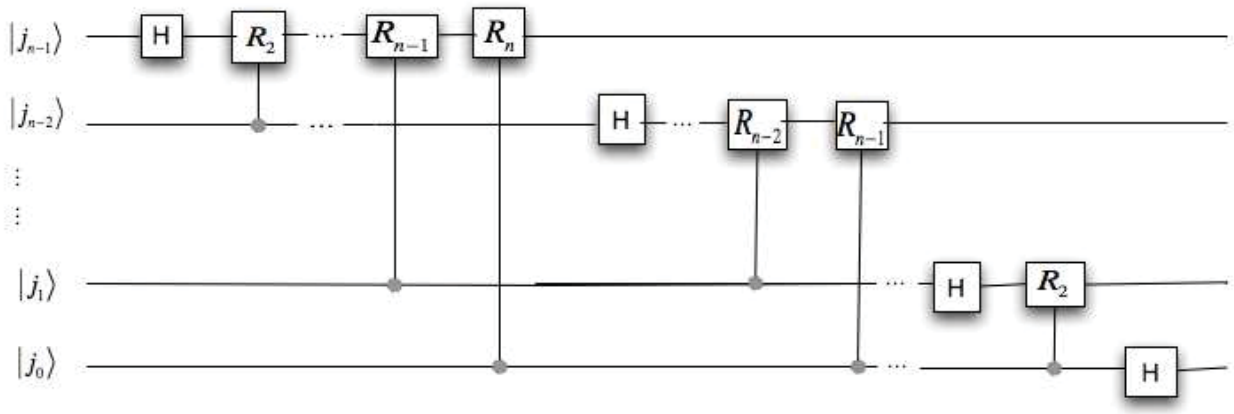


Figura 2.13: Circuito de la Transformada de Fourier Cuántica.

obteniéndose

$$F\left(\left|j\right\rangle\right)=\frac{1}{\sqrt{2^n}}\left(\left|0\right\rangle+e^{2\pi i 0.j_0}\left|1\right\rangle\right)\left(\left|0\right\rangle+e^{2\pi i 0.j_1 j_0}\left|1\right\rangle\right)\ldots\left(\left|0\right\rangle+e^{2\pi i 0.j_{n-1}j_{n-2}\ldots j_0}\left|1\right\rangle\right). \quad (2.40)$$

Se puede notar que la representación que se obtuvo se encuentra factorizada, esto demuestra que el estado cuántico no está entrelazado. Esta representación permite construir un circuito cuántico para la Transformada de Fourier Cuántica de forma más eficiente (Ver Figura 2.13). En la figura el operador está definido por la expresión

$$R_k=\begin{bmatrix}1 & 0\\ 0 & e^{\left(\frac{2\pi i}{2^k}\right)}\end{bmatrix}.$$

Se actúa con la compuerta de Hadamard sobre el cubit más significativo, esto es,

$$H\left|j\right\rangle=\left(H\left|j_{n-1}\right\rangle\right)\left|j_{n-2}j_{n-3}\cdots j_0\right\rangle$$

$$Si\ j_{n-1}=0\ entonces\ H\left|j_{n-1}\right\rangle=\frac{1}{\sqrt{2}}\left\{\left|0\right\rangle+\left|1\right\rangle\right\},$$

$$Si\ j_{n-1}=1\ entonces\ H\left|j_{n-1}\right\rangle=\frac{1}{\sqrt{2}}\left\{\left|0\right\rangle-\left|1\right\rangle\right\}.$$

Observamos que los resultados anteriores pueden escribirse en una sola expresión como

$$H\left|j_{n-1}\right\rangle=\frac{1}{\sqrt{2}}\left(\left|0\right\rangle+e^{2\pi i 0.j_{n-1}}\left|1\right\rangle\right),$$

donde $0.j_{n-1}=\frac{1}{2}j_{n-1}$, de tal manera que si $j_{n-1}=0$ vale cero, y si $j_{n-1}=1$ vale $\frac{1}{2}$; por lo tanto $e^{2\pi i 0.j_{n-1}}=-1$.

Por lo tanto, la primera compuerta de Hadamard actúa en el cubit más significativo y genera el estado

$$\frac{1}{\sqrt{2}}\left(\left|0\right\rangle+e^{2\pi i 0.j_{n-1}}\left|1\right\rangle\right)\left|j_{n-2}\cdots j_0\right\rangle.$$

Las subsecuentes *n*-1 compuertas de fase de rotación, *R_2-controlada* hasta *R_n-controlada* agregan fases desde $\frac{\pi}{2}$ hasta $\frac{\pi}{2^n-1}$ si el correspondiente cubit de control es uno. Después de estas *n*-1 compuertas la función se encuentra en el estado

$$\frac{1}{\sqrt{2}}\left(\left|0\right\rangle+e^{2\pi i 0.j_{n-1}j_{n-2}\ldots j_0}\left|1\right\rangle\right)\left|j_{n-2}\cdots j_0\right\rangle.$$

De manera similar se realizan las subsecuentes *n* - 2, *n* - 1, …, 1 rotaciones controladas con su correspondiente transformación de Hadamard para el resto de los cubits obteniendo así el estado de salida

$$\frac{1}{\sqrt{2}}\left(|0\rangle + e^{2\pi i 0.j_{n-1}j_{n-2}\cdots j_0}|1\rangle\right)\left(|0\rangle + e^{2\pi i 0.j_{n-2}\cdots j_0}|1\rangle\right)\cdots\left(|0\rangle + e^{2\pi i 0.j_0}|1\rangle\right).$$

Comparando el resultado con (2.40), lo único que queda por realizar son *n* SWAPS al estado de salida para obtener el orden correcto del estado transformado.

Estados periódicos

Suponiendo que estamos en N dimensiones y tenemos un estado de la forma

$$|\Theta\rangle = \sum_{n=0}^{N/r-1} c\,|a_0 + nr\rangle,$$

donde $c = \sqrt{r/N}$. Este es un estado periódico con periodo *r* y un elemento compensatorio a_0.

Aplicando la TDF cuántica al estado $|\Theta\rangle$ se obtiene

$$|\tilde{\Theta}\rangle = \hat{F}|\hat{\Theta}\rangle = \sum_{n=0}^{N/r-1} c \sum_{k=0}^{N-1} \frac{1}{\sqrt{n}} e^{2\pi i \frac{(a_0+nr)}{N}} |k\rangle,$$

donde utilizamos $\langle j|a_0 + nr\rangle = \delta_{j,a_0+nr}$. Intercambiando las sumas tenemos

$$|\tilde{\Theta}\rangle = \frac{c}{\sqrt{n}} \sum_{k=0}^{N-1} e^{2\pi i \frac{a_0 k}{N}} \left(\sum_{n=0}^{N/r-1} e^{2\pi i \frac{nrk}{N}} \right) |k\rangle.$$

La suma entre los paréntesis redondos puede simplificarse utilizando la progresión geométrica

$$\sum_{n=0}^{N/r-1} a^n = \begin{cases} N/r & si\ a = 1, \\ \dfrac{1-a^{N/r}}{1-a} & si\ a \neq 1, \end{cases}$$

con $a = e^{\left\{\frac{2\pi ikr}{N}\right\}}$. Como $a^{N/r} = e^{\{2\pi ikr\}} = 1$ entonces se concluye que para tener un resultado diferente de cero debe de ocurrir que *a* = 1 y entonces *k* es un múltiplo entero de *N/r*. Sea por lo tanto $k = m\frac{N}{r}$ y tenemos finalmente

$$|\tilde{\Theta}\rangle = \sum_{m=0}^{r-1} \alpha_m \left| m\frac{N}{r} \right\rangle,$$

donde $\alpha_m = c\frac{N}{r}\frac{1}{\sqrt{N}}e^{2\pi i a_0 m/r} = \frac{e^{2\pi i a_0 m/r}}{\sqrt{r}}$ y $|\alpha_m| = \sqrt{1/r}$ para toda *m*.

Este estado también es periódico y nuestro elemento compensatorio es ahora cero. Podemos explotar este hecho para encontrar el periodo del estado. Si medimos en este momento nuestro registro obtendremos un valor *mN/r* para alguna m entre 0 y *r*-1. Esto por si solo no nos dice mucho de quien es *N/r* y por lo tanto *r.* Pero si corremos nuestro algoritmo *d* veces obtendremos una secuencia de enteros $m_1 N/r, \ldots, m_d N/r$ que son todos múltiplos de *N/r* . Con un número de iteraciones *d* que crece moderadamente sobre *N*, podemos decir con alta probabilidad que *N/r* es el único factor común de todos los números obteniendo así *r*.

2.9.5 Algoritmo de Factorización de Shor

En 1994 Peter Shor publicó el artículo "*Algorithms for quantum computation, discrete logarithms and factorig*" en donde mostró un nuevo enfoque al algoritmo de factorización, combinando principios de la mecánica cuántica con la teoría de números. Este algoritmo ha creado gran interés en computación cuántica debido a que los sistemas criptográficos basan su seguridad en la dificultad de factorizar números muy grandes.

El problema consiste en escribir un número entero positivo impar-no primo como un producto de números primos, N = fac1 x fac2. (Ej. 154,729 = 359 x 431).

Por el Teorema Fundamental de la Aritmética sabemos que todo entero positivo puede representarse de forma única como producto de factores primos.

No es complicado resolver este problema para factores primos pequeños, pero si nos encontramos con números enteros más grandes no existe clásicamente un algoritmo que pueda de manera rápida factorizar dicho número. El mejor algoritmo clásico de factorización (Criba Numérica de Campo) requiere $exp(O(n^{1/3}(\log n)^{2/3}))$ de operaciones donde *n* es el tamaño de entrada.

Algoritmo Clásico de Factorización

Dado un número *N* impar - no primo, que sea producto de dos primos, describiremos el algoritmo de la siguiente manera:

1. Seleccionar un número $y < N$, tal que *y* sea coprimo de *N*, i.e., *mcd*(*y, N*) = 1.

2. Calcular el orden *r* de *y mod N*. El orden se define como el período de repetición de la congruencia $y^r \equiv 1 \bmod N$.

3. Si *r* es par y $y^{r/2} \neq -1 \bmod N$, entonces $x = y^{r/2}$, caso contrario volver a (1).

4. Calcular los dos factores primos: fac1 = *mcd*(*x*+1,*N*), fac2 = *mcd*(*x*-1,*N*).

Ejemplo: Sea N = 55.

Notemos que N es impar y no primo.

1. Los valores de y coprimos a N son: {1, 2, 3, 4, 6, 7, 8, 9, 12, 13,…,54} y tomamos unos al azar, sea este número el 9.

2. Debemos hallar el orden r de 9 *mod* 55,

   Por Teoría de Números sabemos que: "Suponiendo que el *mcd*(y, N) = 1, entonces el orden r de y *mod* N es la menor potencia de y congruente a *1 mod N,* i.e., $y^r \equiv 1 \bmod N$." Como $y = 9$ sus potencias son: {9, 81, 729, 6561,…, 3486784401, …,}
   Sus valores de congruencia están dadas por
   $y^i \bmod 55$; i = 1, 2, 3,…,: {9, 26, 14, 16, 34, 31, 4, 36, 49, 1, 9, 26,…}, podemos notar que el orden es r =10.

3. Dado que r es par y $9^{10/2}$ no es congruente con -1 *mod* 55, $x = 9^{10/2} = 59094$.
   x = 59049

4. *fact*1 = *mcd*(59050, 55) = 5,
   fact2 = *mcd*(59048, 55) = 11.

   De donde obtenemos que 5 y 11 son los dos factores primos de 55.

Algoritmo Cuántico de Shor

Shor usa elementos del algoritmo clásico (teoría de números) para resolver el problema de factorización de manera cuántica. Halla el orden r de y *mod* N en tiempo polinomial, descomponiendo en factores un número N en tiempo O((*log* N)^3).

Hacemos uso de 2 registros, uno de L cubits que permitirá determinar el orden r de y *mod* N y otro de L´ cubits de longitud que servirá como auxiliar:

$\psi = |L\rangle|L'\rangle$.

1. Determinar L y L'.
   Elegimos $q = 2^L$ tal que $N^2 \le q < 2N^2$. L´ la obtenemos para garantizar que forman un número de longitud de N-1 en forma binaria.

2. Una vez obtenidos L y L' pondremos nuestra máquina en una superposición de estados cuánticos.
   Entonces preparamos nuestros registros L y L' en el estado $|0\rangle$ y le aplicamos la transformada discreta de Fourier al primer registro, obteniendo

   $$\frac{1}{\sqrt{q}} \sum_{a=0}^{q-1} |a\rangle|0\rangle, \qquad (2.41)$$

   donde a representa los números binarios de 0 a q -1, es decir, de 0 a 2^L -1. La acción de la Transformada de Fourier Discreta y la Transformación de Hadamard son iguales al actuar sobre un estado $|0\rangle$ de q cubits.

3. Calcular la función $y^a \ mod \ N$ para cada valor de a entre 0 y q - 1. Almacenamos el resultado en el segundo registro.

$$\frac{1}{\sqrt{q}}\sum_{a=0}^{q-1}\left|a\right\rangle\left|y^a \bmod N\right\rangle. \tag{2.42}$$

Cabe destacar que este mismo paso lo realizamos en el algoritmo clásico pero de manera secuencial, el algoritmo de Shor aprovecha las propiedades del cómputo cuántico para realizar los cálculos en una misma iteración.

Los valores obtenidos en el segundo registro son los mismos que obtuvimos en el paso 2 del algoritmo clásico. Ahora, sabemos que por los principios de la mecánica cuántica si realizamos en este momento una observación del estado, el estado colapsará en un nuevo estado donde la información del orden r se encontrará dentro de él.

4. Se realiza una medición en la base computacional para determinar los valores de los bits en el segundo registro, suponiendo que el resultado es $k = y^{a_0} \bmod N$ para algún valor mínimo a a_0. Si r es el orden de $y \ mod \ N$, entonces $y^{a_0} \equiv y^{dr+a_0} \bmod N$ para todas las d. Entonces una medición selecciona n valores de $a = a_0, a_0 + r, a_0 + 2r, \ldots, a_0 + (Ar)$ donde A es el entero más grande menor que $\frac{q - a_0}{r}$ y $a_0 \le r$. Notemos que $A \approx \frac{q}{r}$.

Por lo tanto el nuevo estado colapsado está dado por [10]

$$\psi = \frac{1}{\sqrt{A+1}}\sum_{d=0}^{A}\left|a_0 + dr, k\right\rangle. \tag{2.43}$$

Sea M = A+1:

$$\left|\Phi\right\rangle = \frac{1}{\sqrt{M}}\sum_{d=0}^{M-1}\left|a_0 + dr, k\right\rangle, \tag{2.44}$$

5. Aplicaremos la TFD al estado (2.44) para determinar, en general, el orden r:

$$TDF : \left|\Phi\right\rangle \rightarrow \frac{1}{\sqrt{q}}\sum_{c=0}^{q-1}\frac{1}{\sqrt{M}}\sum_{d=0}^{M-1} e^{2\pi i(a_0+dr)c/q}\left|c, k\right\rangle$$

$$\left|\Phi\right\rangle = \sum_{c=0}^{q-1} e^{2\pi i a_0 \frac{c}{q}}\tilde{f}(c)\left|c, k\right\rangle,$$

donde

$$\tilde{f}(c) = \frac{1}{\sqrt{qM}}\sum_{d=0}^{M-1} e^{2\pi i d\left(\frac{rc}{q}\right)}.$$

De acuerdo al resultado anterior para una función periódica, su TFD cuántica será diferente de cero si *c* es un múltiplo de *q/r*. En este caso es aproximadamente periódica, y entonces $M \approx q / r$.

6. El nuevo estado obtenido estará gobernado por una distribución de probabilidades, la cual está dada por [10]

$$P(c) = \frac{1}{qM} \left| \sum_{d=0}^{M-1} e^{\left(2\pi i d (rc \bmod q)\right)/q} \right|^2 ,$$

donde $P(c)$ es la probabilidad de obtener cualquier valor de *c* entre 0 y q – 1.

Como existen ciertos valores de *c* que tienen mayor probabilidad de ser observados, estos son los cercanos a los múltiplos de *q/r* y cumplen con la relación [11]

$$-\frac{r}{2} \le rc \bmod q \le \frac{r}{2}. \qquad (2.45)$$

Existen precisamente *r* valores de *c mod q* que satisfacen la ecuación y la probabilidad de ver un estado *c* será de al menos $\frac{1}{3} r^2$ [11].

7. Una vez obtenidos los valores de *c*, se escoge uno aleatoriamente. Sea *d* su valor que debe satisfacer la relación:

$$-\frac{1}{2q} \le \frac{c}{q} - \frac{d}{r} \le \frac{1}{2q},$$

para algún valor entre $0 \le d \le r-1$.

La fracción *d/r* puede ser hallada mediante la expansión de fracciones continuas de *c/q*, donde uno de los convergentes del desarrollo nos dará *d/r*. Los convergentes son las aproximaciones racionales generadas por la expansión de fracciones continuas.

Obtenemos así el orden *r* el cual nos permitirá obtener los dos factores primos de *N*.

Ejemplo del algoritmo Cuántico de Shor

Consideremos N = 55 y $y = 9$.

Para obtener *L*, tomemos $q = 2^{12} = 4096$ donde $55^2 \le q < 2.55^2$. Dado que $q = 2^{12}$, $L = 12$. El valor de *L´* debe ser capaz de almacenar de 0 a 54 en binario, entonces como 54 = 110110, $L' = 6$ (Para fines prácticos obtenemos *L* de la relación: $L = \frac{\ln\left(2N^2\right)}{\ln 2}$, considerando el rango al que pertenece *q*).

Consideremos un registro con 2 cubits en el estado $\left|0^2\right\rangle$. Recordando la acción de la transformada de Hadamard.

$$H\left|0\right\rangle \rightarrow \frac{1}{\sqrt{2}}\left(\left|0\right\rangle+\left|1\right\rangle\right).$$

Entonces si aplicamos Hadamard a los dos cubits obtenemos

$$H\left|0^2\right\rangle \rightarrow \frac{1}{\sqrt{2^2}}\sum_{a=0}^{2^2-1}\left|a\right\rangle,$$

donde 00 es el binario de 0, 01 el binario de 1, 10 de 2 y 11 de 3 (para el ejemplo usaremos números enteros).

De la misma manera, si aplicamos *H* a los 12 cubits del primer registro del ejemplo 1:

$$\psi=\left|000000000000\right\rangle\left|000000\right\rangle,$$

donde L = 12 cubits y L' = 6 cubits. Obtenemos entonces la superposición deseada con 2^{12} - 1 términos, esto es

$$\frac{1}{\sqrt{2^{12}}}\left(\left|0,0\right\rangle+\left|1,0\right\rangle+\left|2,0\right\rangle+\ldots+\left|4095,0\right\rangle\right)=\frac{1}{\sqrt{2^{12}}}\sum_{a=0}^{2^{12}-1}\left|a\right\rangle\left|0\right\rangle\equiv\left|\chi\right\rangle.$$

Ahora calculamos la función 9^a *mod* 55 para cada valor de a desde 0 hasta,

$$\left|\chi\right\rangle=\frac{1}{\sqrt{2^{12}}}\left(\left|0,9^0 \bmod 55\right\rangle+\left|1,9^1 \bmod 55\right\rangle+\ldots+\left|4095,9^{4095} \bmod 55\right\rangle\right),$$

que al evaluar las expresiones del segundo registro toma la forma

$$\left|\chi\right\rangle=\frac{1}{\sqrt{4096}}\left(\left|0,1\right\rangle+\left|1,9\right\rangle+\left|2,26\right\rangle+\ldots+\left|4,16\right\rangle+\ldots+\left|10,1\right\rangle+\ldots+\left|4095,34\right\rangle\right).$$

Suponiendo que al efectuar una medición se obtiene k = 16, que implica un valor de a_0 = 4. Entonces el estado después de la medición está determinado por la expresión $\left|\psi\right\rangle$ con M = 410,

$$\begin{aligned}\left|\Phi\right\rangle&=\frac{1}{\sqrt{410}}\left(\left|4,16\right\rangle+\left|14,16\right\rangle+\left|24,16\right\rangle+\ldots+\left|104,16\right\rangle+\ldots+\left|4096,16\right\rangle\right)\\&=\frac{1}{\sqrt{410}}\sum_{d=0}^{410-1}\left|4+d.10,16\right\rangle.\end{aligned}$$

Para verificar que r = 10 y M = 410, hay que demostrar que $y^{a_0} \equiv y^{a_0+d.r} \bmod N$, i.e., verificar que para un r dado se cumple $\mathrm{mod}\left(y^{a_0}, N\right) = \mathrm{mod}\left(y^{a_0+d.r}, N\right)$ para todo d entre 0 y $M - 1$. En nuestro caso tenemos

$mod(9^4, 55) = 16$.

Entonces debemos comprobar que para un r dado se cumple: $\mathrm{mod}(9^4, 55) = \mathrm{mod}(9^{4+d.r}, 55)$. En la tabla se observa la igualdad para todo valor de d. Por lo que el valor de r es 10.

Tabla 2.1 Igualdad para todo valor de d con $r = 10$.

$mod(9^{(4+(d\cdot r))}, 55)$	d=1	d=2	d=3	d=4	d=5	⋯	d=409
r=1	34	31	4	36	49	-	14
r=2	31	36	1	26	16	-	25
⋯	-	-	-	-	-	-	-
r=8	26	1	36	31	16	-	31
r=9	14	26	9	1	49	-	34
r=10	16	16	16	16	16	-	16
r=11	34	31	4	36	49	-	14
⋯	-	-	-	-	-	-	-

Entonces, puede calcularse inmediatamente el valor de M, esto es,

$$M = \frac{q}{r} = \frac{4096}{10} \approx 410.$$

Consideremos la aplicación de la TFD al estado obtenido:

$$\left|\Phi\right\rangle = \frac{1}{\sqrt{410}} \sum_{d=0}^{410-1} \left|4 + d.10, 16\right\rangle,$$

de tal manera que

$$TDF\ \left|\Phi\right\rangle = \sum_{c=0}^{4095} \frac{e^{2\pi i(4)c/4096}}{\sqrt{4096.410}} \left(\sum_{d=0}^{409} \zeta^d \right) \left|c, 1\right\rangle,$$

con $\zeta = e^{2\pi i(10)c/4096}$.

A continuación se calcula la distribución de probabilidades para el caso $N = 55$ con $q = 4096$ y $r = 10$, y los valores de c siguientes:

{0, 410, 819, 1229, 1638, 2048, 2458, 2867, 3277, 3686}, obteniéndose

$P(c)$ = {.100, .057, .087, .087, .057, .100, .057, .087, .087, .057}, respectivamente.

Podemos checar la desigualdad (2.9.15) para $c = 2458$:

$$-\frac{10}{2} \le 10.2458 \bmod 4096 \le \frac{10}{2}$$

$$-5 \le 4 \le 5$$

Entonces para $c = 2458$, hallaremos $\frac{d}{r}$ mediante la expansión la fracción continua siguiente

$$\frac{c}{q} = \frac{2458}{4096} = \cfrac{1}{1+\cfrac{1}{1+\cfrac{1}{1+\cfrac{1}{1+\cfrac{1}{409}}}}}$$

cuyos convergentes son

$$\frac{1}{1} = 1$$

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

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

$$\cfrac{1}{1+\cfrac{1}{1+\cfrac{1}{1+\frac{1}{1}}}} = \frac{3}{5}$$

$$\cfrac{1}{1+\cfrac{1}{1+\cfrac{1}{1+\cfrac{1}{1+\frac{1}{409}}}}} = \frac{1229}{2048}$$

De aquí obtenemos que $\frac{d}{r} = \frac{6}{10} = \frac{3}{5}$, ya que el denominador no excede a 55 (N=55).

El orden *r* de mod N es un múltiplo de *r* =5. La siguiente tabla muestra la función $y^a \equiv 1$.

Tabla 2.2 Función $y^a \equiv 1$

a	$y^a mod\ N\ = 9^a mod\ 55$
5	34
10	1
15	34

donde *a* es un múltiplo de 55, obteniendo así el orden *r* = 10. Una vez obtenido el período se continúan los pasos del algoritmo clásico de factorización para obtener los dos factores primos de 55, esto es

$$x = y^{\frac{r}{2}} = 9^{\frac{10}{2}} = 59094,$$

$$fac1 = mcd(59050, 55) = 5,$$
$$fac2 = mcd(59048, 55) = 11.$$

2.9.6 Algoritmo de Grover

Un algoritmo de búsqueda es aquel que está diseñado para encontrar un elemento *x* en un conjunto posible de soluciones (estructura de datos) tal que *P*(*x*) sea verdadero. Una gran clase de problemas dentro de las Ciencias de la Computación implican un proceso de búsqueda.

Como ejemplo podríamos ver la búsqueda de un elemento en una base de datos, el ordenamiento de una lista o el coloreado de una gráfica. El coloreado de una gráfica puede ser vista como una búsqueda para encontrar y asignar el color correcto a los vértices de la gráfica tal que el enunciado "todos los vértices adyacentes tienen diferentes colores" sea verdadero. El problema de coloreado es uno de los problemas más conocidos de la teoría de gráficas, el problema consiste en asignar colores diferentes a los vértices de una gráfica de modo que ningún par de vértices adyacentes tengan el mismo color (Figura 2.14).

Un problema de búsqueda sobre una base de datos estructurada es aquella donde la información de nuestro espacio de búsqueda y nuestro enunciado *P* puede ser explotado para construir un algoritmo eficiente. Realizar una búsqueda sobre una lista ordenada alfabéticamente puede ser explotada para encontrar una solución eficiente.

Un problema de búsqueda sobre una base de datos no estructurada es aquella donde no "sabemos" nada acerca de la estructura del espacio de soluciones y de nuestro enunciado *P*. De manera general en un problema de búsqueda no estructurada, probando aleatoriamente la veracidad de $P(x_i)$ elemento por elemento es lo mejor que podemos hacer clásicamente. Para un problema de búsqueda sobre un espacio no estructurado de tamaño *N* requiere *O*(*N*) evaluaciones de *P*. En una computadora cuántica Grover demostró que este mismo problema puede ser resuelto con una probabilidad acotada en $O\left(\sqrt{N}\right)$. Cabe destacar que el algoritmo de Grover hace la búsqueda *más eficiente* que un algoritmo clásico, no la hace más sencilla.

El algoritmo de Grover busca en una lista no estructurada de tamaño N alguna x tal que $P(x)$ sea verdadero. Sea n tal que $2^n \geq N$ y sea U_p la compuerta cuántica que implementa la función clásica $P(x)$ que prueba la veracidad del enunciado, donde denotaremos que el enunciado es verdadero con un 1. Recordemos que $P(x)$ es una función binaria de $\{0,1\}^n \rightarrow \{0,1\}$ tal que $P(x) = 1$ si $x = x_0$ y $P(x) = 0$ de otra manera

$$U_p : |x,0\rangle \rightarrow |x,P(x)\rangle.$$

El primer paso es calcular P para todas las posibles entradas x_i, aplicando U_p a un registro que contiene la superposición $\frac{1}{\sqrt{2^n}}\sum_{x=0}^{n-1}|x\rangle$ de todas las 2^n posibles entradas x junto con un registro $P(x)$ iniciado en 0, obteniendo el registro

$$\frac{1}{\sqrt{2^n}}\sum_{x=0}^{n-1}|x\rangle|P(x)\rangle. \qquad (2.46)$$

Es difícil obtener un resultado útil a partir de esta superposición.

Para cualquier x_0 tal que $P(x_0)$ es verdadero, $|x_0,1\rangle$ formará parte de la superposición descrita en (2.46). Como la amplitud de dicho estado es $\frac{1}{\sqrt{2^n}}$, la probabilidad de que una medición aleatoria produzca x_0 es solo 2^{-n}. El truco reside en lograr incrementar las amplitudes de los vectores $|x_0,1\rangle$ para los cuales P es verdadero y disminuir las amplitudes de los vectores $|x_0,0\rangle$ donde P sea falso en la ecuación (2.46).

Una vez que dicha transformación fue realizada sobre el estado cuántico, solo se requiere medir el último cubit del estado cuántico que representa $P(x)$. Debido al cambio de amplitudes que se realizó, existe una alta probabilidad de que el resultado sea 1. En este caso la medición proyecta el estado (2.46) sobre el subespacio $\frac{1}{\sqrt{2^n}}\sum_{i=1}^{k}|x,1\rangle$ donde k es el número de soluciones. Si la medición da 0 entonces el proceso debe iniciarse de nuevo y la superposición de la ecuación (2.46) deberá calcularse nuevamente.

Descripción del Algoritmo de Grover

1. Preparar la función de onda de la computadora en el estado $|00\ldots0\rangle|1\rangle$ donde utilizamos un cubit auxiliar

2. Preparar un registro que contenga una superposición de todos los posibles valores del tipo $x_i \in \left[0,\ldots,2^n-1\right]$ y el estado $\frac{1}{\sqrt{2}}\{|0\rangle-|1\rangle\}$ del cubit auxiliar. Esto se realiza mediante la acción de $n + 1$ compuertas de Hadamard.

3. Calcular $P(x_i)$ sobre el registro.

4. Cambiar las amplitudes α_i a $-\alpha_i$ para x_j tal que $P(x_j) = 1$ (ver subsección sobre cambio de Signo).

5. Aplicar inversión sobre el promedio (ver Inversión sobre el promedio) para incrementar las amplitudes de x_j con $P(x_j) = 1$. Las amplitudes resultantes donde $P(x_i) = 0$ han disminuido en forma considerable.

6. Repetir $\left[\frac{\pi}{4}\sqrt{N}\right]$ veces los pasos del 2 al 4.

7. Leer el resultado.

Cambio de signo

El objetivo es implementar la transformación:

$$U|x\rangle = (-1)^{f(x)}|x\rangle$$

que no modifica los $|x\rangle$ si ocurre que $f(x) = 0$ y le agrega un coeficiente -1 en los que verifica que $f(x) = 1$.

La transformación U_f (2.9.7) implementa la evaluación de la función booleana f. Cuando sólo queremos evaluar f sobre un estado $|x\rangle$ se aplica U_f con $b = 0$, en este caso se escoge $|b\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$. Nótese que si $f(x) = 0$, entonces $|b \oplus f(x)\rangle = b$, mientras que si $f(x) = 1$ es $|b \oplus f(x)\rangle = \frac{1}{\sqrt{2}}(|1\rangle - |0\rangle) = -|b\rangle$. Luego

$$U_f(|x,b\rangle) = (-1)^{f(x)}|x,b\rangle.$$

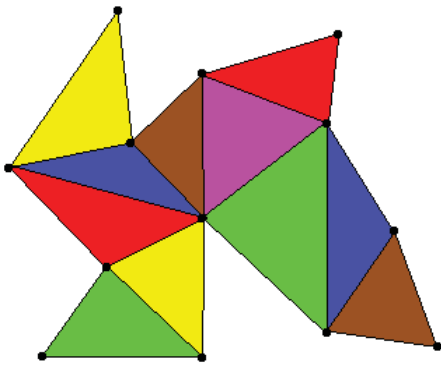


Figura 2.14: Problema de coloreado de gráficas.

Con esta transformación realizamos la evaluación de f y el cambio de signo de las amplitudes $|x_j\rangle$ que satisfacen la propiedad.

Al aplicar U_f sobre un estado cualquiera $|\phi\rangle = \sum_{j=0}^{N-1} a_j |x_j\rangle$.

Sea $X_0 = \{x \mid f(x) = 0\}$ y $X_1 = \{x \mid f(x) = 1\}$

$$U_f\left(|\phi, b\rangle\right) = U_f\left(\sum_{x_j \in X_0} a_j |x_j, b\rangle + \sum_{x_j \in X_1} a_j |x_j, b\rangle\right)$$
$$= \sum_{x_j \in X_0} a_j |x_j, b\rangle - \sum_{x_j \in X_1} a_j |x_j, b\rangle$$
$$= \left(\sum_{x_j \in X_0} a_j |x_j\rangle - \sum_{x_j \in X_1} a_j |x_j\rangle\right) \otimes |b\rangle.$$

Inversión sobre el promedio

Para realizar la operación de inversión sobre el promedio en una computadora cuántica tiene que usarse una transformación unitaria. Se puede observar que la transformación

$$\sum_{i=0}^{N-1} a_i |x_i\rangle \rightarrow \sum_{i=0}^{N-1} (2A - a_i) |x_i\rangle,$$

donde A denota el promedio de las a_i, es realizada por la matriz de N x N, de la forma $D_{ij} = -\delta_{ij} + \frac{2}{N}$, esto es

$$D = \begin{pmatrix} \frac{2}{N} - 1 & \frac{2}{N} & \cdots & \frac{2}{N} \\ \frac{2}{N} & \frac{2}{N} - 1 & \cdots & \frac{2}{N} \\ \cdots & \cdots & \cdots & \cdots \\ \frac{2}{N} & \frac{2}{N} & \cdots & \frac{2}{N} - 1 \end{pmatrix}$$

Como $DD^*=I$, D es unitaria y entonces puede ser implementada por una computadora cuántica (Grover propuso una implementación eficiente de esta transformación con $O(\log(N))$ puertas elementales $D = H\sigma_x^2 (I \otimes H) CNOT (I \otimes H)\sigma_x^2$ donde $(I \otimes H) CNOT (I \otimes H)\sigma_x^2 = CPHASE(\pi)$. Grover, L (1996) "*A fast quantum mechanical algortihm for database search*").

Ejemplo Algoritmo de Grover

Consideremos una búsqueda en un espacio de 4 objetos que pueden ser representados por dos cubits. Inicialmente los dos cubits se preparan en el estado $|00\rangle$ y el cubit auxiliar se encuentra en el estado $|y\rangle = |1\rangle$. Cada uno de ellos sufre una transformación de Hadarmard obteniéndose

$$|\psi\rangle = H^3|001\rangle = \frac{1}{\sqrt{2^2}}\sum_{x=0}^{2^2-1}|x\rangle\frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$$
$$= \{|000\rangle + |010\rangle + |100\rangle + |110\rangle - |001\rangle - |011\rangle - |101\rangle - |111\rangle\},$$

donde en el primer renglón usamos la base computacional. Ahora se evalúa la función *f*(*x*), las preguntas o pregunta se hacen por medio de un operador unitario que para hacer el proceso reversible utiliza un cubit auxiliar y produce el resultado

$$|x\rangle|y\rangle \underline{O}|x\rangle|y \oplus f(x)\rangle$$

Como $|y\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$ por discusiones anteriores se tiene que

$$|y \oplus f(x)\rangle = \begin{cases} +|y\rangle & f(x) = 0 \\ -|y\rangle & f(x) = 1 \end{cases}$$

Sea *f*(10)=1 y *f*(*x*) para *x* = 00, 01, y 11, entonces después de la indagación el estado queda

$$\frac{1}{2}\{|00\rangle + |01\rangle - |10\rangle + |11\rangle\}\frac{1}{\sqrt{2}}\{|0\rangle - |1\rangle\}$$

que difiere del anterior en el signo del coeficiente del estado favorable. Como el registro auxiliar no ha cambiado ya no se considera. El paso siguiente es transformar la diferencia de fase que aparece en $|01\rangle$ en una diferencia de amplitud. Esto se logra mediante la transformación unitaria *D* con *N* = 2 que toma la forma

$$\frac{1}{2}\begin{bmatrix} -1 & 1 & 1 & 1 \\ 1 & -1 & 1 & 1 \\ 1 & 1 & -1 & 1 \\ 1 & 1 & 1 & -1 \end{bmatrix}$$

de tal manera que

$$\frac{1}{2}D\begin{bmatrix} 1 \\ 1 \\ -1 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}$$

Por lo tanto una medición estándar de los dos cubits da el resultado $|10\rangle$ con certeza. Concluyéndose que el problema ha sido resuelto con una sola indagación de la función *f*, mientras que la computadora clásica requiere en promedio $N_I = \frac{1}{4}*1+\frac{1}{4}*2+\frac{1}{2}*3 = 2.25$ indagaciones. Para una búsqueda de un objeto entre 8 posibles en la primera iteración se tiene una probabilidad de $\frac{25}{32}$ de obtener el item buscado. Para aumentar éste se tiene que repetir el procedimiento indicado del algoritmo de Grover.

2.10 Máquina Universal de Turing Cuántica

La teoría de la computación ha tenido un gran avance durante los últimos años, ha ayudado a estudiar, definir y entender el cómputo de mejor manera. Nos ayuda a comprender la importancia de la teoría matemática en el cómputo, a definir lo computable y lo no computable (i.e posibilidades y limitaciones del cómputo) y a realizar una clasificación de los problemas computacionales, de manera más general la Teoría de la Computación son los cimientos de las ciencias de la computación. En 1985 Deutsch [12] definió una computadora como cualquier sistema físico cuya evolución dinámica lo lleva de uno de un conjunto de estados de entrada a uno de un conjunto de estados de salida. Los estados están etiquetados en alguna forma canónica, la máquina es preparada en un estado con una etiqueta inicial dada y después, seguida por unos movimientos (cada movimiento es un paso en el programa), el estado de salida es medido. Para un sistema clásico determinístico la medición de la etiqueta de salida es una función definida *f* de la etiqueta de entrada; además el valor de esa etiqueta de salida en principio puede ser medida por un observador externo (el usuario) y entonces se puede decir que la máquina calcula la función *f*.
En este sentido dos computadoras son computacionalmente equivalentes, sobre etiquetas dadas, si en cualquier posible experimento o secuencia de experimentos en donde sus etiquetas fueron preparadas equivalentemente sobre sus etiquetas de entradas y los observables correspondientes a cada una de las etiquetas de entrada fueron medidos, los valores de medición de estos observables para estas dos máquinas serán estadísticamente indistinguibles. Es decir las funciones de distribución de probabilidad para las salidas de las dos máquinas serán idénticas.

En 1936 Alan Turing describió un modelo abstracto de una máquina conocida como máquina de Turing que sigue un conjunto finito de reglas bien definidas que actúan sobre cadenas finitas de entrada y las convierten en cadenas finitas de salida. La máquina de Turing es un dispositivo que corre sobre una cinta infinita bidireccional dividida en celdas discretas donde en cada celda está contenido el símbolo 0,1 o blanco. Además de un conjunto finito de posibles estados internos y un cabezal que puede leer los contenidos de las celdas de las cintas que están inmediatamente sobre ella. El cabezal en cada paso puede escribir un símbolo sobre la celda en la que se encuentre. Existen dos estados internos especiales: un estado inicial q_0 y un estado de detención q_H. Una Máquina de Turing (MT) contiene una lista de reglas de transición describiendo su operación, existe a lo más una regla de transición para cada probable contenido de la celda y el estado interno. Si el estado interno es q_i y el cabezal se encuentra sobre la celda con contenido S_j entonces la máquina busca la regla de transición (q_i, S_j). Si ninguna regla es encontrada la máquina entra a un estado de detención inmediatamente. Un cómputo consiste en inicializar la MT con el cabezal sobre la primera celda no vacía del lado izquierdo de la cinta y la máquina en el estado interno q_0. Posteriormente las reglas de transición son simplemente aplicadas hasta que la máquina llega al estado de detención q_H, en este momento el contenido de la celda será la salida del cómputo.

El modelo matemático para una máquina de estado finita es conocido como autómata. El modelo más sencillo de un autómata puede considerarse como la computadora más simple una máquina *M* computa a lo sumo una función. No debe de haber una diferencia fundamental entre alterar el estado de entrada en la que *M* es preparada y alterar sistemáticamente la constitución de *M* para que se convierta en una máquina diferente *M´*, que compute una función diferente. Para realizar estas operaciones es necesario considerar una computadora con dos entradas y la preparación de un programa que determine cuál de las funciones será computada. A cada máquina *M* le corresponde un conjunto *C*(*M*) de *M* funciones computables. Una función *f* es *M-computable* si *M* puede computar *f* con un programa preparado. Dadas dos máquinas *M* y *M´* es posible construir una nueva máquina cuyo conjunto de funciones computables contenga la unión de *C*(*M*) y *C*(*M´*) y así consecutivamente. Una computadora es un autómata gobernado por un programa, tal que diferentes programas harán trabajar a la computadora de manera distinta.

En 1936 Church y Turing estabecen que "*toda función que es intuitivamente computable puede ser computada por la máquina universal de Turing*". Esta tesis nos indica que todas las máquinas de cómputo finitas pueden ser simuladas por una sola máquina llamada Máquina Universal de Turing.

Definimos *C*(*T*) como el conjunto de las funciones recursivas que es menor al total de las funciones que van de Z a Z donde *T* es la máquina universal de Turing. Las entradas de estas funciones pueden ser números naturales, números binarios, hexadecimales o cadenas de algún lenguaje formal. Para las funciones que van de Z a Z el conjunto *C*(*M*) siempre está contenido en *C*(*T*). Esto quiere decir que existen problemas que las computadoras clásicas pueden no resolver (problemas no decidibles) y otros tantos que son difícil de resolver, es decir que el tiempo requerido para encontrar su solución es demasiado grande (problemas intratables). Un ejemplo de un problema no decidible es el problema de detención. Dada la descripción de un programa y una entrada finita, el problema consiste en decidir si el programa llega a un estado de detención o nunca se detiene. Un problema intratable es el problema de factorización de números primos (Sec. 2.7.5).

La tesis de Church Turing acota el espacio de funciones computables mediante la descripción de un subconjunto de las matemáticas que puede ser calculado. Si no existe algoritmo que solucione el problema la función no será computable, permitiendo como ya se mencionó que existan sistemas físicos finitos que no puedan ser simulados por una máquina de Turing.

Deutsch reinterpreta esta tesis de Church-Turing. Define una función naturalmente computable como las funciones que en principio pueden ser computadas por algún sistema físico real en un número finito de pasos. Introduce el concepto de simulación perfecta. Una máquina o computadora *M* es capaz de simular perfectamente un sistema físico *S* sobre un etiquetado dado en sus entradas y en su salida si existe un programa $\pi(s)$ para *M* que exprese a un programa *M* computacionalmente equivalente a *S* sobre estas etiquetas. En otras palabras $\pi(s)$ convierte a *M* en una caja negra funcional e indistinguible de *S*. La versión física del principio de Church-Turing descrito por Deutsch dice que *todo sistema físico realizable puede ser simulado perfectamente por un modelo universal de máquina operando por medios finitos* [12]. *Todo sistema físico realizable* se refiere a cualquier objeto físico donde la experimentación sea posible.

Esta definición de Deutsch es más fuerte que la tesis de Chusrch-Turing. Dada la continuidad de la dinámica clásica, los posibles estados de un sistema clásico necesariamente forman un con-

tinuo. Por otro lado sólo hay una cantidad contable de maneras de preparar una entrada finita en *T*. Entonces *T* no puede simular perfectamente un sistema dinámico, *T* puede simular un sistema continuo únicamente mediante aproximaciones discretas sucesivas. La Teoría Cuántica es compatible con la reinterpretación de Deutsch de la tesis de Church-Turing [12].

Deutsch afirma que no existe razón alguna para pensar que las leyes de la física deben respetar las limitaciones de la hipótesis de Church-Turing y de los procesos matemáticos llamados algoritmos, aunque existan funciones fuera del conjunto de funciones computables de cada máquina físicamente posible, es decir no existe alguna inconsistencia en postular sistemas físicos que computen funciones fuera de *C*(*T*), afirmando que la razón por la que podemos construir computadoras aritméticas es gracias a que las leyes de la física permiten la existencia de modelos para los operadores de la aritmética tales como la suma, la resta y la diferencia. Deutsch propuso un modelo de una Máquina Universal de Turing Cuántica (MUTC) para la cual siempre existe una Máquina de Turing Cuántica (MTC) con un programa como parte del estado de entrada que realiza una transformación unitaria sobre un número arbitrario de cubits arbitrariamente cercanos a cualesquiera cubits deseados.

La MUTC de Deutsch no es el único modelo universal de una computadora cuántica. Nielsen y Chuang [13] propusieron un arreglo de compuertas cuánticas programables. Bernstein y Vazirani [14] se basaron en el modelo de Deutsch y propusieron un dispositivo cuántico y demostraron que existe una máquina de Turing *U* capaz de simular otra MT *M* con precisión ε.

2.10.1 Máquina de Turing Cuántica

A partir de la generalización de una máquina de Turing clásica, una MTC consiste en un procesador finito de *N* cubits $n=\{n_1\}(i=0,\ldots,N-1)$ y una cinta infinita consistente de una secuencia de cubits $m=\{m_1\}(i=\ldots,-1,0,1,\ldots)$, en donde sólo una porción finita de la cinta es usada. El cómputo es realizado en pasos fijos con duración *T* y durante cada paso solo el procesador y una parte finita de la memoria interactúan, el resto de la memoria se mantiene estática. La dirección actual de la cinta, es decir la posición de la cabeza esta descrita por el observable *x*, la cual contiene a todo *Z* como su espectro. Se define el estado de una MTC como un vector unitario en el espacio de Hilbert desarrollada sobre los estados base

$$|x\rangle|n\rangle|m\rangle,$$

donde $|n\rangle \equiv |n_0, n_1, \ldots, n_{N-1}\rangle, \quad |m\rangle \equiv |\ldots, m_{-1}, m_0, m_1, \ldots\rangle.$

Si *U* es el operador unitario que describe una aplicación de la regla de transición de la máquina, los elementos no ceros de la matriz están determinados por

$$\langle x\pm 1; n'; m_x', m_{y\neq x} | U | x; n; m_x, m_{y\neq x}\rangle,$$

donde cada elección de *U* se define una MTC diferente. La evolución de la máquina durante *s* pasos se encuentra descrita por

$$|\Psi(sT)\rangle = U^* |\Psi(0)\rangle$$

donde $|\Psi(0)\rangle$ es el estado inicial y T es el tiempo de duración de cada paso. Si la medición ocurre después de n_1 pasos, y la medición es descrita por un operador J_1 entonces la evolución de la máquina para los primeros $n_1 + j$ pasos se encuentra descrita por $U^j J_1 U^{n_1}$, la cual ha dejado de ser unitaria dado que el operador J_1 es una medición sobre la base computacional.

La salida de la máquina se encuentra en la cinta como una superposición de los estados base y deberá ser leída después de haber realizado la medición del contenido del cubit de detención y haberla encontrado en el estado uno. El operador podrá medir en cualquier momento el bit de detención en orden para decidir cuándo leer el contenido de la cinta (y colapsar el estado de la máquina). La intención del bit de detención es dar al operador de la máquina una indicación de cuándo la salida deberá de ser leída de la cinta sin interferir excesivamente en el cómputo. La salida de una máquina de Turing cuántica para alguna entrada x, que puede ser una superposición de los estados clásicos de entrada, es una distribución de probabilidad P_x sobre todos los posibles contenidos de la cinta en el momento de observar el bit de detención que ha sido activado. Dada la unitariedad, la dinámica de la MTC, así como la de cualquier sistema cuántico cerrado, es necesariamente reversible.

2.10.2 Máquina Universal de Turing Cuántica

Deutsch afirmó que existe una MUTC (basada en la MTC y la MUT con 8 operaciones adicionales [12]) para la cual existe un programa que realiza una transformación unitaria, arbitrariamente cercana a cualquier transformación unitaria sobre un número finito de cubits.
Para la MUTC escribimos su estado como

$$|Q_x, n\rangle |n_h\rangle |D\rangle |P\rangle |\Sigma\rangle,$$

donde $|Q_x, n\rangle$ es el estado del procesador, incluyendo la posición de la cabeza, $|n_h\rangle$ es el cubit de detención, $|D\rangle$ es el estado de los datos de registro y $|P\rangle$ es el estado del programa. $|D\rangle$ y $|P\rangle$ son ambos parte de la cinta y $|\Sigma\rangle$ es el resto de la cinta, no afectada durante el cómputo.

Deutsch afirmó que para una U, con alguna transformación arbitraria Υ y una precisión arbitraria ε, existe siempre un estado de programa $|P(D, \Upsilon, \varepsilon)\rangle$ y un número entero $s(D, \Upsilon, \varepsilon)$ tal que

$$U^{s(D,\Upsilon,\varepsilon)} |Q_x, n\rangle |D\rangle |P(D, \Upsilon, \varepsilon)\rangle |\Sigma\rangle = |Q_x', n\rangle |D'\rangle |P'(D, \Upsilon, \varepsilon)\rangle |\Sigma\rangle,$$

donde $\| |D'\rangle - |U\, D\rangle |^2 < \varepsilon$, P' es el estado de programa después de s pasos, Q y Q' son los estados del procesador en el tiempo inicial y después de s pasos respectivamente.

Deutsch realiza la demostración de la MUTC mediante un esquema de concatenación. La concatenación de dos programas es un programa cuyo efecto es el seguimiento del segundo programa inmediatamente después del primero. Se dio por sentado que si estos dos programas eran válidos entonces su concatenación existe pero la validez de esta no fue probada [15] por Deutsch. Entonces la MUTC de Deutsch fue definida más no realmente probada. Por otro lado el concepto de

universalidad en las MTC no es la misma que tenemos para las máquinas de Turing clásicas donde las simulaciones son exactas, la simulación es claramente solo una aproximación [16].

La MUT nos ha dado los instrumentos necesarios para encontrar en un número finito de pasos soluciones a problemas en distintas áreas de las ciencias, pudiendo ejecutar todo tipo de cálculo que sea realizable. No podríamos entender el concepto de la computadora digital sin la MUT. Grandes avances en la ciencia se han dado gracias al desarrollo y las capacidades logradas por las computadoras, una MUT es el modelo abstracto de nuestras computadoras hoy en día.

De la misma manera el lograr la construcción de una MUTC ampliaría las capacidades ya logradas por los algoritmos cuánticos, pudiendo calcular cualquier función cuánticamente computable que le sea introducida.

2.11 Conclusiones del capítulo

En este capítulo presentamos los conceptos fundamentales de la Computación Cuántica, desde la noción de cubit hasta la Máquina de Turing Cuántica pasando por la noción de medición, compuertas y algoritmos cuánticos.

Capítulo 3: Arquitectura de computadora cuántica tolerante a fallos

3.1 Introducción

La Computación Cuántica como una disciplina de la Ingeniería está todavía en su infancia [17]. Aunque la Física se comprende bien, el desarrollo de dispositivos que calculan a partir de la Mecánica Cuántica es tecnológicamente desalentador. Mientras que los experimentos hasta la fecha manipulan solo un puñado de bits cuánticos (cubits) [18], consideramos aquí el esfuerzo requerido para construir una computadora cuántica a gran escala. Este esfuerzo exige más que una estimación somera de la cantidad de cubits y compuertas requeridas para un algoritmo dado. Uno debe considerar las fallas del hardware cuántico, con errores causados tanto por el ambientes y el error deliberado de las operaciones, el procesamiento clásico requerido cuando la corrección de errores es invocada, el tratamiento especial necesario para la construcción de secuencias de compuertas arbitrarias de un conjunto tolerante a fallos limitado, y así sucesivamente. Este capítulo provee un marco para afrontar el reto completo de diseñar una computadora cuántica. Muchos investigadores han presentado y examinado componentes de computación cuántica de gran escala. En este capítulo se estudia como estos componentes pueden ser combinados en un diceño eficiente, e introducimos nuevos métodos que mejoran la computadora cuántica que se propone. Esta búsqueda de Ingeniería es la arquitectura de computadora cuántica, la cual se desarrolla aquí en capas. Una arquitectura descompone los comportamientos de un sistema complejo en un conjunto manejable de operaciones. Una arquitectura multicapa lo hace a través de capas de abstracción donde cada capa encarna un conjunto crítico de funciones relacionadas. Para los propósitos de este trabajo, cada capa ascendente trae al sistema más cercano a un entorno ideal de computación cuántica.

Este capítulo está organizado como sigue. El resto de la Sección 3.1 provee una visión global de una arquitectura de computadora cuántica multicapa, indicando cómo los tópicos que se examinan están vinculados unos con otros. La Sección 3.2 enumera los componentes esenciales de una computadora cuántica para examinar una nueva plataforma de hardware basado en el control óptico de los puntos cuánticos. En la Sección 3.3 se discuten las técnicas de control para la supresión de errores de hardware antes de utilizar la corrección de errores activa. En la Sección 3.4 se demuestra cómo implementar y explicar los recursos de corrección de error cuántico, con particular énfasis sobre el código de superficie [19]. En la Sección 3.5 se analizan las técnicas necesarias para construir compuertas cuánticas universales de un conjunto limitado de operaciones previstas por la corrección del error. La Sección 3.6 calcula los recursos computacionales necesarios para implementar dos algoritmos cuánticos prominentes: la factorización de enteros y la simulación cuántica. En la Sección 3.7 se discuten los problemas de tiempo que afectan a la forma en que las capas en la arquitectura interactúan con otra. Finalmente, en la Sección 3.8 se discute la arquitectura de computadora cuántica tolerante a fallos, así como el diseño multicapa.

3.1.1 Trabajo previo sobre arquitectura de computadora cuántica

Muchas tecnologías diferentes sobre computación cuántica se encuentran bajo investigación experimental [18], no obstante, cada arquitectura de un sistema escalable sigue siendo un problema de investigación abierto. Desde que DiVincenzo introdujo su criterio fundamental para una tecnología de computación cuántica viable [20], y Steane enfatizó la dificultad de diseñar sistemas capaces de ejecutar Corrección de Error Cuántico (CEC) adecuadamente [21,22], varios gru-

pos de investigadores han esbozado diversas taxonomías adicionales que abordan las necesidades arquitectónicas de sistemas a gran escala [23,24]. A modo de ejemplo, se han propuesto interconectores de pequeña escala para muchas tecnologías, pero los problemas de organizar los subsistemas usando estas técnicas en una arquitectura completa para un sistema de gran escala ha sido abordado solo por pocos investigadores. En particular, La cuestión de la heterogeneidad en la arquitectura del sistema ha recibido relativamente poca atención.

La subrutina más importante en computadoras cuánticas tolerante a fallos considerada entonces es por lejos la preparación de los estados ancilla (comodín) para circuitos tolerantes a fallos, dado que estos circuitos frecuentemente requieren extremadamente muchas ancillas. Taylor *et al.* propusieron un diseño con "bloques de ancillas" alternadas y ''bloques de datos'' en la disposición del dispositivo [25]. Steane introdujo la idea de ''fábricas'' para la creación de ancillas [26], para lo cual se examina en este trabajo para el caso del código de superficie. Isailovic *et al.* [27] estudió este problema para arquitectura de trampas de iones y se encontró que, para circuitos cuánticos típicos, aproximadamente el 90% de la computadora cuántica se debe dedicar a tales fábricas con el fin de calcular ''a la velocidad de los datos,'' o donde la producción de ancilla no es el proceso limitante en velocidad. Los resultados que se presentan aquí están muy de acuerdo con esta estimación. Metodi *et al.* consideran también la producción de ancillas en diseños de trampas de iones, poniendo el foco en cambio sobre estado ancilla de 3-cubits usada para la compuerta de Toffoli [28], lo cual es un camino alternativo para un conjunto de compuertas tolerante a fallos.

Algunos investigadores han estudiado la dificultad de mover datos en un procesador cuántico. Kielpinski *et al.* propusieron una tecnología escalable de trampa de iones utilizando áreas separadas para memoria y cálculo [29]. Dado que la corrección de error cuántico requiere ciclos rápidos a través de todos los cubits físicos en el sistema, esta aproximación es la mejor usada como una célula de unidad de replicado a través de un sistema más grande. Otros investigadores han propuesto sistemas homogéneos alrededor de este concepto básico. Una estructura común es un árbol *H* recursivo, el cual trabaja bien con un pequeño número de capas de un código de Calderbank-Shor-Steane, destinados específicamente a los sistemas de trampa de iones [30,31]. Oskin *et al.* [32], construye sobre la tecnología de Resonancia Magnética Nuclear (RMN) de estado sólido de Kane [33], proponiendo una red flexible de sitios, teniendo en cuenta explícitamente los temas de control clásico y el movimiento de los datos en los sistemas cuánticos escalables, pero sin un plan específico para CEC. En el caso de computación cuántica con circuitos superconductores, la arquitectura cuántica de von Neumann se considera específicamente dedicada al hardware para memorias cuánticas, registros de reducción a cero, y un bus cuántico, como el mencionado en [34].

El acoplamiento a larga distancia y la comunicación son retos importantes para los ordenadores cuánticos. Cirac *et al.* propusieron el uso de cubits fotónicos para distribuir el entrelazamiento entre átomos distantes [35], y otros investigadores han estudiado las perspectivas para compuertas no locales mediadas ópticamente [36–40]. Tales canales fotónicos podrían ser utilizados para realizar una computadora cuántica distribuida, escalable y modular [41]. Inversamente, Metodi *et al.* consideraron cómo usar compuertas locales y teleportación cuántica para mover cubits lógicos mediante sus arquitectura de arreglo lógico cuántico de trampa de iones [28]. Fowler *et al.* [42] investigaron la arquitectura de flujo de cubits en la juntura de Josephson considerando las dificultades extremas de rutear ambos los acopladores cuánticos y un gran número de líneas de control clásicas, produciendo una estructura con soporte para los códigos de Calderbank-Shor-Steane y los cubits lógicos organizados en una línea. Whitney *et al.* [43,44] han investigado la disposición automática y la optimización de circuitos diseñados específicamente para arquitecturas de trampa

de iones, mientras que Isailovic *et al.* [27,45] han estudiado la interconexión y las cuestiones de rendimiento de datos en sistemas similares de trampa de iones, con un énfasis sobre la preparación de ancillas para compuertas de teleportación [46].

Otros trabajos en los que se han estudiado las arquitecturas de computadoras cuánticas solo con acoplamiento del vecino más cercano entre cubits en un arreglo son los de [47-51], tecnología atractiva desde el punto de vista del diseño del hardware. Con los avances recientes en la operación de códigos topológicos y sus características deseables tales como el hecho de tener un umbral práctico alto y solo requerir interacciones de vecino más próximo, el esfuerzo de investígación se ha desplazado hacia arquitecturas capaces de construir y mantener grandes clústers de estados de 2 y 3 dimensiones [52-55]. Estos sistemas se basan en modelos de corrección topológica de errores [56], cuya tolerancia al error más alta se convierte frecuentemente en el costo de un sistema físico mayor, relativo a, por ejemplo, implementaciones basadas en el código de Steane [57]. El código de superficie [19], el cual se examina en este trabajo por su impacto sobre la arquitectura, pertenece a la familia topológica de códigos.

La atención reciente se ha dirigido a los modelos distribuidos de computación cuántica. Devitt *et al.* estudiaron como distribuir una red de computación cuántica de clúster de estados fotónicos sobre diferentes regiones geográficas [58]. El marco resumen de un multicomputador cuántico reconoce que los sistemas de gran escala demandan interconectores heterogéneos [59]; en la mayoría de las tecnologías de computación cuántica, no es posible construir sistemas monolíticos que contengan, el acoplamiento, y el control de miles de millones de cubits físicos. Van Meter *et al.* [60] extendieron éste marco arquitectónico con un diseño basado en el acoplamiento nanofotónico de los puntos cuánticos espín-electrón el cual usa explícitamente múltiples niveles de inter-conexión el cual va variando las fidelidades de acoplamiento (resultando en requerimientos de purificación variantes), tan bien como la habilidad de operar con un rendimiento muy bajo de dispositivos funcionales. A pesar de que el sistema propuesto tiene muchas características atractivas, la preocupación por la dificultad de fabricación de componentes ópticos de calidad adecuadamente alta y el deseo de reducir el tiempo del ciclo de red del código de superficie lleva al diseño del sistema propuesto en este capítulo.

3.1.2 Marco por capas

Una buena arquitectura debe tener una estructura simple al mismo tiempo que se maneja eficientemente los arreglos complejos de recursos en una computadora cuántica. Las arquitecturas multicapa son una aproximación convencional a efectos de resolver tales problemas de Ingeniería en muchos campos de la tecnología de la información. Además en [31] se presenta una arquitectura de software multicapa para diseñar computadoras cuánticas. La arquitectura de este capítulo, la cual describe el diseño físico de la computadora cuántica, consiste de cinco capas, donde cada capa tiene un conjunto prescrito de condiciones a cumplir. La interfaz entre dos capas se define por los servicios que una capa más baja provee a una por encima de ella. Para ejecutar una operación, una capa debe emitir comandos para la capa de abajo y procesar los resultados. Diseñar un sistema de esta manera asegura que las operaciones relativas son agrupadas juntas y que la organización del sistema es jerárquica. Este enfoque permite a los ingenieros cuánticos centrarse en retos individuales, mientras ver también como se ajusta un proceso en el diseño general. Al organizar la arquitectura en capas, se crea deliberadamente un diseño modular para la computa-dora cuántica.

El marco multicapa puede ser entendido por una pila de control compuesta de las cinco capas en la

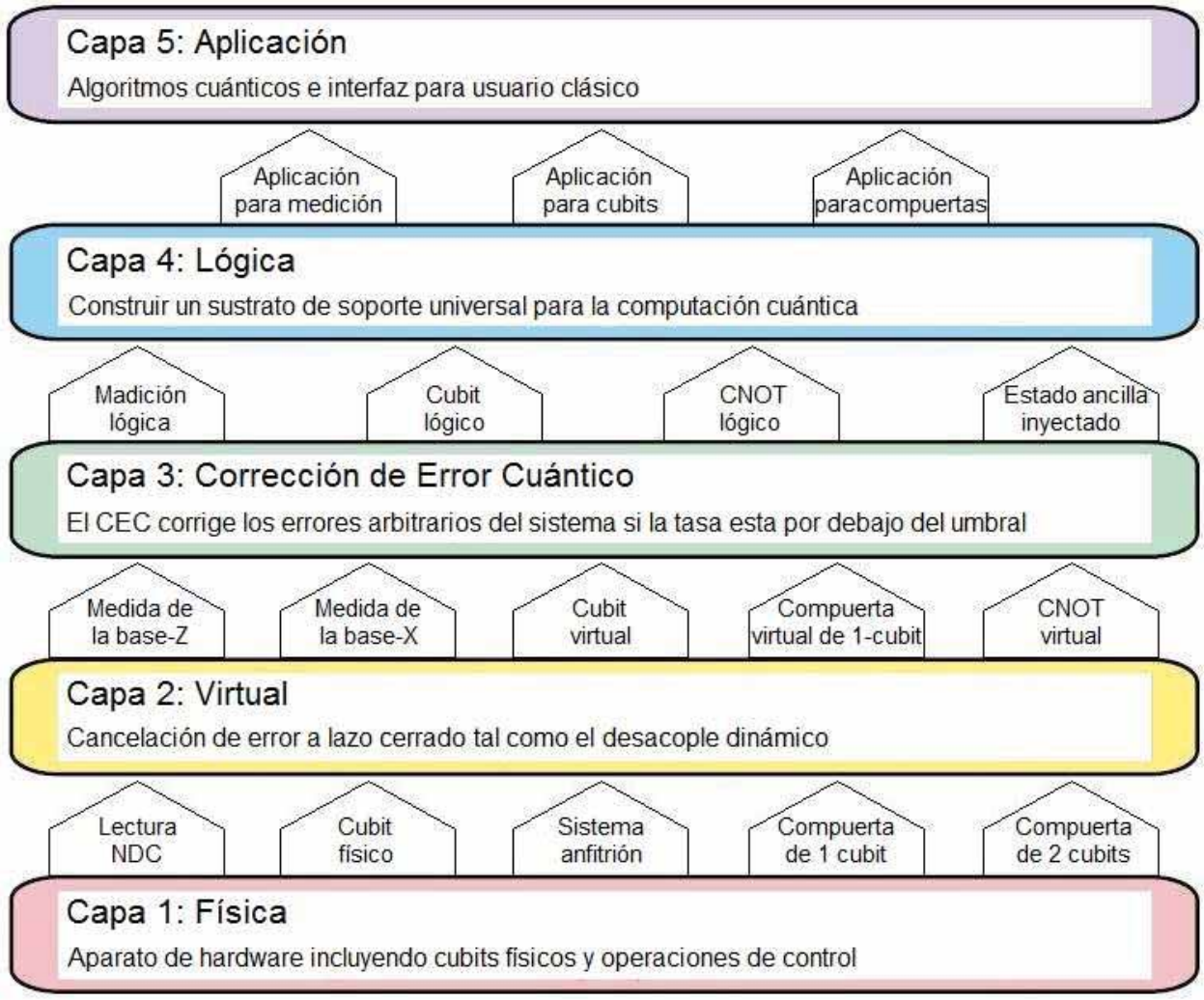


Fig.3.1 Pila de control multicapa que forma el marco de una arquitectura de computadora cuántica. Las flechas verticales indican los servicios provistos a una capa más alta.

arquitectura. La Fig.3.1 muestra un ejemplo de la pila de control para la arquitectura de puntos cuánticos que se propone en este capítulo, pero las interfaces particulares entre capas variarán de acuerdo al hardware físico, el esquema de corrección de error cuántico, y demás, que uno elige para implementarla. En el tope de la pila de control está la Capa de Aplicación, donde se implementa un algoritmo cuántico y se le proveen los resultados al usuario. La capa física inferior es anfitriona de los procesos físicos crudos que soportan a la computadora cuántica. Las capas entre (Virtual, CEC, y Lógico) son esenciales para formar los procesos cuánticos defectuosos en la Capa Física en un sistema de alta exactitud, tolerante a fallos [61] y los cubits y las compuertas cuánticas en la Capa de Aplicación.

3.1.3 Interacción entre capas

Dos capas se encuentran en una interfaz, la cual define cómo ellas intercambian instrucciones o los resultados de esas instrucciones. Muchos comandos diferentes están siendo ejecutados y procesados simultáneamente, de manera que podemos considerar también cómo interactúan las capas dinámicamente. Para que la computadora cuántica funcione eficientemente, cada capa debe

dar instrucciones a las capas de más abajo en una secuencia bien definida. No obstante, un Sistema robusto debe poder también manejar errores causados por dispositivos defectuosos.

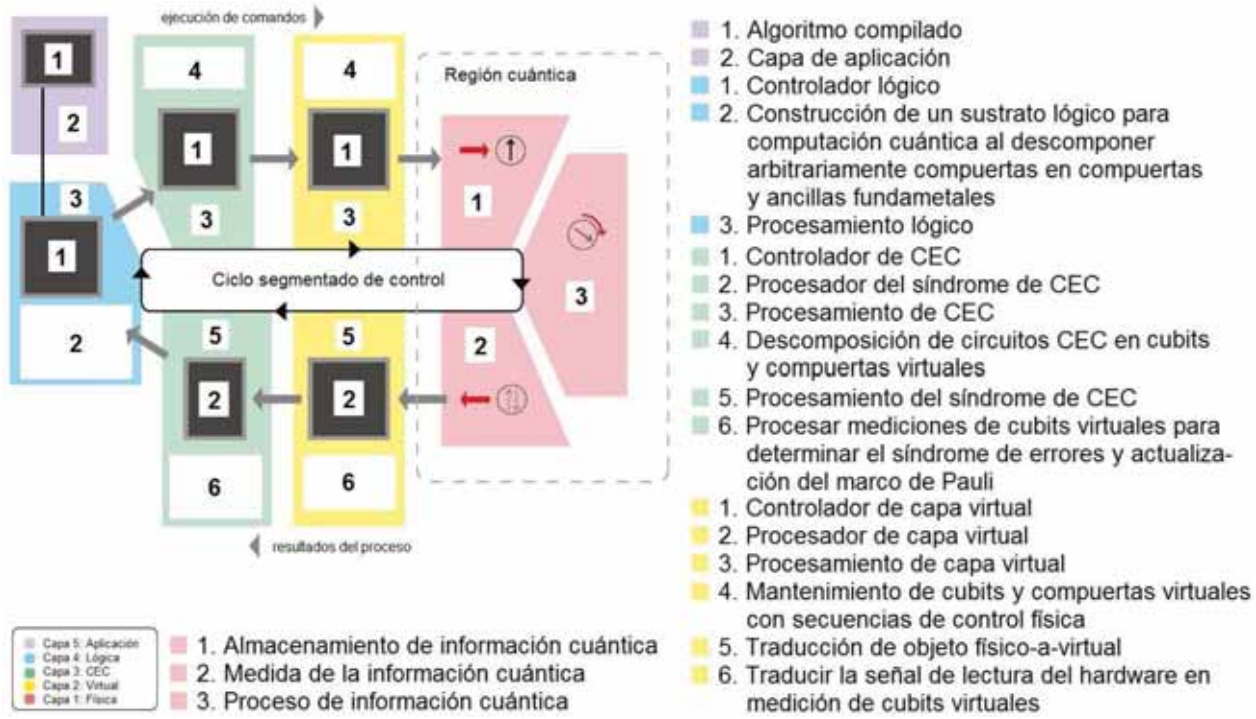


Fig.3.2 Ciclo de control primario de una computadora cuántica con la arquitectura multicapa. Considerando que la pila de control en la Fig. 1 dicta las interfaces entre capas, el ciclo de control determina el temporizado y la secuencia de operaciones. El recuadro punteado encierra a la capa física la cual indica que todos los procesos cuánticos suceden exclusivamente aquí, y las capas superiores procesan y organizan las operaciones de la capa física. La capa de aplicación es externa al bucle ya que funciona sin ninguna dependencia sobre el diseño de la computadora cuántica específica.

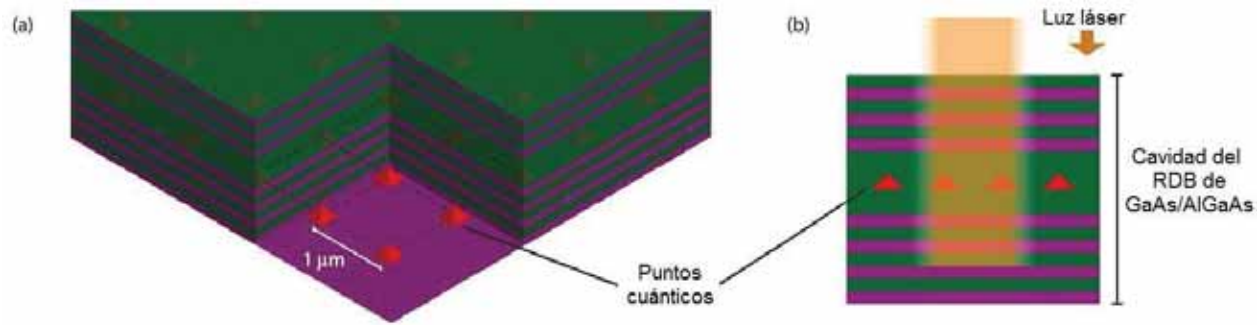


Fig.3.3 Puntos cuánticos en una microcavidad óptica planar que forma la base de la plataforma de hardware de PCECO. (a) Los puntos cuánticos están ordenados a un 1 μm de distancia en un arreglo cuadrado bidimensional. La trampa de puntos cuánticos es de electrones simples, y sus espines serán usados para procesamiento de información cuántica. (b) Vista lateral. Los espines del electrón son manipulados con pulsos láser enviados en la cavidad óptica desde arriba, y dos puntos cuánticos vecinos pueden ser acoplados por el campo óptico láser que los solapa. Las capas púrpura y verde son un crecido epitaxial de haces moleculares de AlGaAs y GaAs, Las capas alternas forman una cavidad óptica tipo reflector-de-Bragg-distribuido (RBD) la cual es planar, confinando la luz en la dirección vertical y extendiéndola a través del sistema en su totalidad en direcciones horizontales.

Para satisfacer ambos criterios, un bucle de control debe manipular operaciones en todas las capas simultáneamente al mismo tiempo que el procesamiento del síndrome de la medición tiene lugar para corregir errores. Un prototipo para este bucle de control se muestra en la Fig.3.2.

El ciclo de control primario define el comportamiento dinámico de la computadora cuántica en esta arquitectura ya que todas las operaciones deben interactuar con este bucle. El principal propósito del ciclo de control es implementar satisfactoriamente la corrección de error cuántico. La computadora cuántica debe funcionar lo suficientemente rápido como para corregir errores. Sin embargo, algunas operaciones de control incurren necesariamente en retardos, por lo que este ciclo no se limita a emitir un solo comando y esperar el resultado antes de continuar—la canalización es esencial [27,62]. Un tema relacionado consiste en que las operaciones en las diferentes capas se producen en escalas de tiempo drásticamente diferentes, como se discute más tarde en la Sec.3.7. La Fig.3.2 además describe la estructura de control necesaria para la computadora cuántica. Los procesadores en cada capa siguen las operaciones y comandos de edición actuales para las capas inferiores. De las capas 1 a la 4 interactúan en el bucle, mientras que la capa de Aplicación interfacea solo con la capa Lógica, puesto que es agnóstico acerca del diseño subyacente de la computadora cuántica (la cual se explica en la Sec.3.6).

3.1.4 Plataforma de hardware PCECO

El marco multicapa para computación cuántica fue desarrollado en tándem con una plataforma de hardware específica, conocida como Puntos Cuánticos con Espines Controlados Opticamente (PCECO). La plataforma PCECO utiliza espines electrónicos en puntos cuánticos para cubits. Los puntos cuánticos son ordenados en un arreglo bidimensional; la Fig. 3 muestra una representación en corte de la matriz de puntos cuánticos en el interior de la microcavidad óptica, la cual facilita el control de los espines electrónicos con pulsos láser. Se demuestra que el diseño PCECO resulta ser un candidato promisorio para computación cuántica de gran escala, comenzando con un análisis del hardware en la capa Física.

3.2 Capa I: Física

Los requerimientos esenciales para la capa Física están corporizados por el criterio de DiVincenzo [20], no obstante, también existe un interés en la performance del hardware cuántico. La escala de tiempo de las operaciones y los grados de los errores, ambos sistemáticos y aleatorios, son parámetros críticos que determinan el tamaño y velocidad de la computadora. En esta sección se discuten los componentes esenciales del hardware de una computadora cuántica, acom-pañada por la plataforma PCECO la cual se introduce como ejemplo. Llegamos a la conclusión mediante el análisis del rendimiento del hardware PCECO. Se advierte que muchos de los ele-mentos de hardware necesarios están todavía en fase de desarrollo experimental, pero elegimos los descritos a continuación como ejemplos para establecer escalas de tiempo que afectan a las capas superiores de la arquitectura.

3.2.1 Cubit físico

Una computadora cuántica debe tener la habilidad para almacenar información entre pasos del proceso; el objeto de cumplir con esta función se conoce convencionalmente como cubit físico. Un cubit físico puede ser más complejo que un Sistema de dos niveles, y este tema se aborda en la Capa 2 en la arquitectura, donde se utilizan las funciones de control para formar un verdadero bit cuántico como una unidad de información (véase la Sec.3.2). Ejemplos de cubits físicos inclu-

yen trampas de iones, modos de polarización fotónica, espines electrónicos, y estados cuánticos en circuitos superconductores [18]. El resto de la capa Física se dedica a controlar y medir al cubit físico.

El diseño de arquitectura multicapa es flexible en el sentido que la capa Física se puede adaptar a un hardware específico, tales como los cubits de circuitos superconductores, con cambio mínimo para las capas superiores tales como la corrección de error. El cubit físico que se considera en PCECO es el espín de un electrón ligado a un Punto Cuántico (PC) auto-ensamblado de InGaAs alrededor de un sustrato de GaAs [63–68]. Estos PCs pueden ser excitados ópticamente para estados trion (un electrón ligado y excitón), el cual emite luz con una longitud de onda de aproximadamente 900 *nm* cuando ellos decaen. Un campo magnético transversal divide los niveles de espín en dos estados fundamentales metaestables [69], el cual más tarde formará un sistema de dos niveles para un cubit virtual en la Capa 2. La separación de energía de los estados espín es importante por dos razones relativas al control del espín electrónico: primero, la división de la energía facilita el control con pulsos ópticos, como se explica en la Sec.3.2.3. Segundo, hay rotación de fase continua entre estados de espín $\left|\uparrow\right\rangle$ y $\left|\downarrow\right\rangle$ alrededor del eje σ_z sobre el cubit en la esfera de Bloch, la cual en conjunción con los pulsos ópticos temporizados proporciona un control unitario completo del vector espín electrón.

3.2.2 Sistema anfitrión

Para estos propósitos, el sistema anfitrión es el entorno de Ingeniería del cubit físico que hace de soporte al cálculo. Los ejemplos incluyen los campos de captura en el diseño de trampas de iones, las ondas guiadas en computación cuántica óptica, y el cristal de diamante que rodea a los centros de nitrógeno vacantes [18]. El sistema anfitrión define al entorno inmediato del cubit físico, el cual será importante para caracterizar el ruido que afecta a las operaciones cuánticas.

Ya hemos señalado que, en PCECO, el espín del electrón está ligado a un punto cuántico. Estos puntos cuánticos están empotrados en una microcavidad óptica, la cual facilitará las operaciones de las compuertas cuánticas via los pulsos láser. Para acomodar la matriz bidimensional del código de superficie detallado en la Capa 3, esta microcavidad debe ser planar en su diseño, así la cavidad se construye de dos espejos reflectores de Bragg distribuidos (RBD) los cuales se apilan verticalmente con una capa de cavidad entre ellos de $\lambda/2$, como se muestra en la Fig.3.3. Esta cavidad se cultiva en epitaxia de haces moleculares. Los PCs son empotrados en el centro de esta cavidad para maximizar la interacción con antinodos de los modos del campo de cavidad.

Usando epitaxial de haces moleculares, las cavidades de alta calidad (Q>10^5) se pueden cultivar con capas alternadas de GaAs/AlAs [70]. El núcleo en el punto cuántico y el sustrato alrededor tiene espín distinto de cero, lo cual constituye una fuente importante de ruido (ver Sec.3.2.6).

3.2.3 Mecanismo de compuerta de 1 cubit

La compuerta de 1-cubit manipula el estado de un único cubit físico. Este mecanismo de compuerta de 1-cubit todavía es un proceso físico; solo más tarde estas operaciones de control físico se combinan en una ‘‘compuerta virtual’’ (ver Sec.3.3). No obstante, es importante que la capa Física ofrezca un control suficiente del cubit físico. El control unitario completo de un cubit requiere al menos de dos grados de libertad ajustables, tal como la rotación alrededor de dos ejes sobre la esfera de Bloch, y otros tres parámetros ajustables libremente [71].

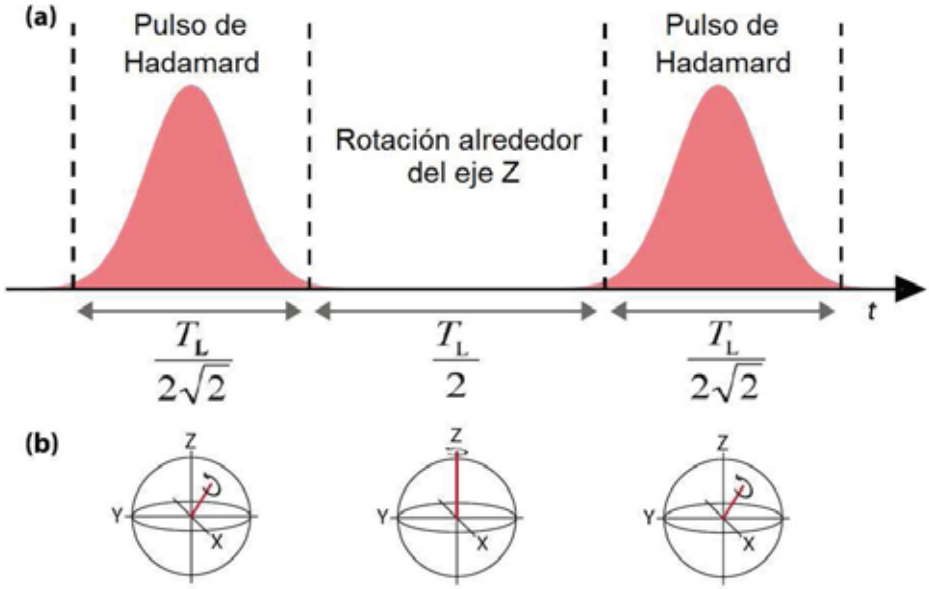


Fig.3.4 Pulsos de Hadamard en PCECO. (a) Una secuencia de pulso corto genera una rotación del eje-σ_x sobre el espín de la esfera de Bloch con dos pulsos de Hadamard y la precesión del eje-σ_z del campo magnético. La duración de los pulsos y el retardo entre ellos es proporcional al período de Larmor, T_L. (b) Los diagramas de la esfera de Bloch muestran los ejes de rotación en cada instante durante la secuencia.

Las operaciones de 1-cubit en PCECO son desarrolladas usando un campo magnético transversal y pulsos de láser ultrarápidos [68–72]. El campo magnético provee una rotación de frecuencia angular constante alrededor del eje-σ_z, mientras que los pulsos láser provocan una rotación de potencia dependiente alrededor de un eje ortogonal, el cual rotulamos con σ_x.

El primer comportamiento no ideal que consideramos es que el pulso láser tiene una duración finita, para que la precesión en σ_x y σ_z suceda al mismo tiempo, lo cual dificulta la manipulación del vector de espín de Bloch. Para solucionar esto, se introducen los ‘‘pulsos de Hadamard’’ [73]: Uno sintoniza la potencia y la duración del pulso láser para hacer que se maneje la rotación del pulso en el eje-σ_x igual en frecuencia angular a la precesión en el eje-σ_z del campo magnético, de esta forma el eje de rotación se convierte en $H = \frac{1}{\sqrt{2}}(\sigma_x + \sigma_z)$. Un ‘‘pulso π’’ alrededor de este eje es una compuerta de Hadamard. Al usar dos pulsos de Hadamard y la rotación $R_z(\theta)$ via precesión libre en el campo magnético, se puede construir cualquier rotación en el eje-σ_x por $R_x(\theta) = H.R_z(\theta).H$, como se muestra en la Fig.3.4. Al implementar pulsos de Hadamard, se pueden obtener operaciones de alta fidelidad con pulsos de duración finita. Un problema desafiante para PCECO es cómo el Sistema ejecuta millones de operaciones de control en paralelo.

Se tiene la visión de un sistema de imagen óptica consistente de una matriz de espejos de Sistemas Micro-Electro-Mecánicos (SMEMs) para dirigir los haces del láser de control individualmente hacia o lejos de los puntos cuánticos, junto con moduladores electro-ópticos para controlar con precisión la temporización de los pulsos láser.

3.2.4 Mecanismo de compuerta de 2 cubits

Las operaciones de 2-cubits acoplan dos cubits físicos, los cuales pueden generar entrelazamiento. Este mecanismo es crucial para computación cuántica, sin embargo a menudo es difícil de poner en práctica experimentalmente. Por ejemplo, las compuertas entrelazadas tales como CNOT son usadas frecuentemente en CEC, para el desarrollo rápido de mecanismos de compuertas de 2-cubits de alta fidelidad, lo cual es imperativo para el procesamiento de información a gran escala. En muchos casos, la compuerta de 2-cubits es el proceso que define la velocidad y exactitud de la computadora cuántica. La construcción práctica de una compuerta de 2-cubits escalable en PCECO sigue siendo el elemento más desafiante del hardware, por lo que varios métodos están actualmente en desarrollo. Una compuerta rápida de 2-cubits, completamente controlada ópticamente, debería ser ciertamente atractiva, y propuestas tempranas [64] identificaron la importancia de emplear las no linealidades de la cavidad Electro-Dinámica Cuántica (EDC). [64] sugiere la aplicación de dos láseres para el control de ambas compuertas, es decir, la de 1-cubit y la de 2-cubits; desarrollos más recientes han indicado que ambas compuertas de 1-cubit [72,74,75] y 2-cubits [76] pueden ser logradas usando solamente un único pulso óptico.

Consideremos una compuerta de 2-cubits via la interacción dispersa propuesta en [76]. La figura de mérito crítica para el Sistema de la cavidad EDC es el factor de cooperatividad C, el cual es proporcional al factor de calidad de la cavidad Q dividido por el volumen de la cavidad V. Para PCECO, prevemos un confinamiento en la cavidad transversal debido enteramente a la disposición planar ampliada de la microcavidad, en la cual los factores de cooperatividad son mejorados por la dependencia del ángulo de la respuesta de la cavidad, un efecto que se ampliará con alto índice de refracción de contraste en los espejos alternos de la pila del RBD [63]. Aunque los factores de cooperatividad existentes no logran de esta manera ser estimados por ser lo suficientemente altos para producir compuertas cuánticas con tasas de error suficientemente bajas para computación cuántica tolerante a fallos, las técnicas de control avanzadas y las codificaciones multiespín (tales como ‘‘cubits virtuales’’; ver Sec.3) pueden activar esta tecnología para funcionar con tasas de error aceptables. En [76] se estima que esta compuerta requerirá 10–100 *ns* para ejecutarse. Para el presente análisis se asume el valor 32 *ns*, el cual coincide con una compuerta virtual en la Sec.3.2. Otras mejoras en la velocidad o fidelidad de la compuerta o ambas pueden estar disponibles al introducir cambios en las interacciones usando microcavidad de polaritones [77]; estudiar esta posibilidad es el tema del trabajo futuro.

3.2.5 Lectura de medición

La medición es otro componente esencial de la computación cuántica. Como mínimo, uno debe poder leer el resultado final de un cálculo, no obstante la medición típicamente es usada extensamente en CEC tolerante a fallos. Por esta razón, las computadoras cuánticas pueden requerir medición que es comparable en velocidad y exactitud a las operaciones de control. Por otra parte, muchas situaciones llaman a la medición de No Demolición Cuántica (NDC), donde el cubit físico es proyectado en un autoestado del operador medición. Para ilustrar con un contraejemplo, consideremos un cubit definido por el basal y el primer estado excitado ópticamente de un PC. Un posible esquema de medición consiste en detectar la emisión de un fotón, la cual debería indicar que cubit estuvo en el estado excitado. No obstante, el estado final del cubit es el estado basal para cualquiera de los resultados de la medición, la cual es una medición destructiva, por lo que este procedimiento no se puede repetir. Por el contrario, la medición NDC es altamente deseable dado que se puede repetir, por lo que el ruido de lectura clásico puede ser reducido al tiempo promedio.

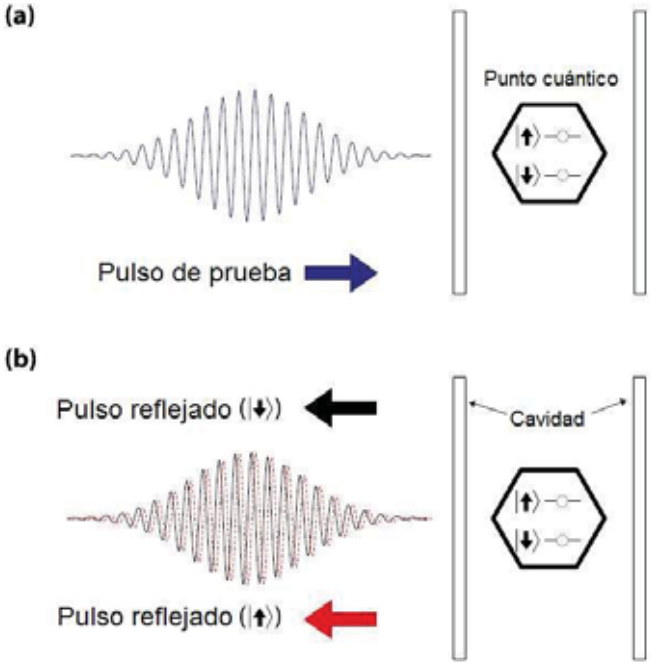


Fig.3.5 Diagrama de lectura para una no-demolición cuántica (NDC) dispersiva para PCECO. (a) Un pulso de prueba es enviado al interior de una microcavidad conteniendo un punto cuántico cargado. (b) La interacción de una cavidad mejorada dispersiva entre el pulso y el espín del electrón crea un desplazamiento de fase de estado-dependiente en la luz que deja la cavidad. La medición del desplazamiento de fase puede realizar una medición proyectiva sobre el espín del electrón.

PCECO requiere un esquema de medición NDC el cual está todavía en fase experimental. El mecanismo propuesto (mostrado en la Fig.3.5) se basa en la rotación de Faraday/Kerr. El principio físico subyacente es el siguiente: Un pulso de prueba fuera de resonancia incide sobre un PC, y el pulso óptico recibe un desplazamiento de fase diferente dependiendo de si el PC de electrones está en el estado de espín hacia arriba o hacia abajo. (Estos estados están separados en energía por el campo magnético externo.) Los fotodetectores sensitivos combinados con la medida de detección de homodina desplaza la fase para provocar una medición NDC proyectiva sobre el espín del electrón. En años recientes, varios resultados han demostrado la promesa de este mecanismo de medición: experimentos multidisparo a cargo de Berezovsky *et al.* [78] y Atatüre *et al.* [79] han medido desplazamientos de fase espín-dependientes en PC cargados, y Fushman *et al.* [80] observaron un gran desfasaje inducido por un PC neutral en una cavidad de cristal fotónica. Más recientemente, Young *et al.* observaron un desplazamiento de fase significativamente mejorado de un PC empotrado en una cavidad de microcolumnas [81].

3.2.6 Fuentes ruidosas y errores

El ruido y la decoherencia son los más grandes obstáculos para la computación cuántica escalable. En general, las fuentes de ruido que corrompen al cubit físico o degradan la fidelidad de las operaciones de control deberían ser caracterizados tan bien como sea posible. Para el presente análisis, se considera el entorno ruidoso para un espín electrónico en PCECO. El ruido primario en este sistema está desfasado, probablemente causado por la distribución no homogénea de los espines nucleares en el PC. El desfasaje conjunto está caracterizado por $T_2^* \approx 2\,ns$, mientras que el desfasaje intrínseco está caracterizado por $T_2 \approx 3\,\mu s$ [82]. Cuando el ruido experimentado por

un cubit está dominado por el desfasaje, uno puede contrarrestar la decoherencia mediante secuencias de control adaptadas a esta fuente de ruido [83]. La Sec.3.1 introduce un esquema de desacoplamiento diseñado específicamente para PCECO.

3.2.7 Resumen de rendimiento del hardware

En la Tabla 3.1 se resumen los tiempos de ejecución para las operaciones esenciales de la Capa 1 (C1) en PCECO. Estos son los procesos cuánticos que son los componentes básicos de las operaciones de información cuántica en las Capas 2 y superiores. No obstante, para un procesador cuántico completo, deberíamos considerar también el hardware de control clásico y los problemas de Ingeniería, tales como retardos, los cuales pueden ocurrir en un sistema grande. Por ejemplo, en [51] se consideran las implicaciones de los alambres de control clásicos, tales como los problemas de ruteo, el temporizado de señales, y la generación de calor en dispositivos de baja temperatura. Aunque la Ingeniería del hardware de control clásico es un problema importante, se encuentra fuera del alcance de nuestro análisis presente, razón por la cual nos la reservamos para un trabajo futuro.

Tabla 3.1 Parámetros para las operaciones cuánticas de la Capa 1. La precesión de fase de espín es determinada por la división de la energía de estado de espín debido a un campo magnético externo. Para implementar una compuerta de Hadamard, el tiempo del pulso del ancho de banda es $1/\sqrt{8}$ del período de Larmor (T_{Larmor}). Los tiempos para la operación de entrelazamiento y la medición NDC se estiman de la simulación.

Operación	Mecanismo	Duración	Notas
Precesión de fase de espín σ_x -axis	División del campo magnético en los niveles de energía de espín	40 ps	El entorno nuclear no homogéneo causa ensanchamiento espectral en la frecuencia de Larmor, que es la fuente de los procesos T_2^*
Pulso de rotación del estado de espín	Transición de Raman estimulada con ancho de banda de pulso óptico	14 ps	Desintonizado del rojo de las transiciones del estado trion-basal de los espines
Operación de entrelazamiento	Desplazamiento de fase no lineal de los estados de espín via acoplamiento a un modo de cavidad común	32 ns	Señal de láser CW modulada mediante un modulador electro-óptico (MEO)
Medición NDC	Desplazamiento de fase dispersiva de la luz reflejada de la cavidad planar	1 ns	Señal de láser CW modulada mediante un MEO.

3.3 Capa II: Virtual

En la capa Virtual se vuelcan primero los efectos cuánticos de la capa Física, como ser primitivas de información en cubits virtuales y compuertas cuánticas. Tal como se define en Ciencias de la Computación, un objeto virtual obedece a un conjunto predeterminado de conductas, sin especificar la estructura de dicho objeto. Como un ejemplo, un cubit virtual puede ser definido por un subespacio libre de decoherencia [84–86] construido de tres espines electrónicos; donde se considera como un todo, tres espines tienen muchos más grados de libertad que un único cubit. Similar comportamiento se puede observar en las compuertas cuánticas en PCECO, las cuales en la actualidad consisten de una secuencia de pulsos láser. Este proceso de transcripción de convertir muchos elementos físicos en un una unidad de información virtual es tarea de la Capa 2, y las funciones de esta capa se clarifican en esta sección. La Fig.3.6 da un pantallazo de los procesos de la capa Virtual en PCECO.

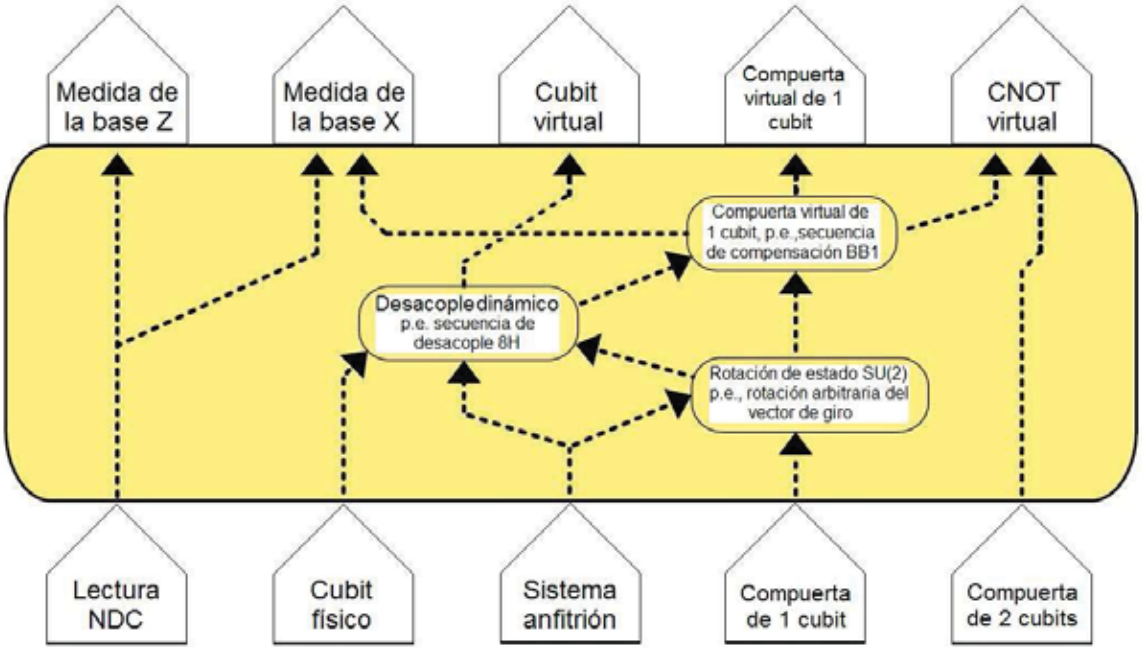


Fig.3.6 Los mecanismos de la capa virtual. Las salidas de la Capa 1 se combinan en secuencias controladas para producir cubits y compuertas virtuales. Las flechas indican como la salida de un proceso es usado por otro proceso.

En un sentido general, la capa Virtual hace robusta a la capa Física para los errores sistemáticos. Este efecto se observa tanto en los cubits como en las compuertas virtuales, donde hacemos cumplir simetrías en el sistema (mediante un cuidadoso diseño de las operaciones de control) las cuales causan errores correlacionados a cancelar por interferencia. El ejemplo más simple de este comportamiento es la secuencia de eco-espín de Hahn [87], y de hecho las técnicas de desacoplamiento desempeñan un papel destacado en la forma en la que construimos un cubit virtual.

3.3.1 Cubit virtual

Los cubits virtuales conforman los cubits físicos subyacentes en un Sistema de dos niveles el cual se aproxima a un cubit ideal. No obstante, el cubit virtual se modela de manera tal de tener una cierta cantidad finita de decoherencia, tal como el canal de despolarización [71]. En su caso, el desacoplamiento dinámico [89–90] y/o los subespacios libres de decoherencia [84–86] son usados para crear cubits virtuales de larga duración, y la decoherencia residual caracteriza la vida del cubit virtual. En lo que sigue, se considera cómo construir un cubit virtual con un PC cargado, incluyendo la mitigación de varios efectos no ideales en este sistema. En PCECO, el cubit virtual es creado de dos estados de espín metaestables de un electrón confinado para un PC. Como se discutió en la Sec.2.6, el Sistema físico crudo tiene un tiempo de desfasaje de $T_2^* \approx 2\ ns$ [82] causado por una distribución inhomogénea de espines nucleares en el entorno del electrón. Este tiempo desfasado es insuficiente para la CEC en la Capa 3, por lo que este sistema debe ser aumentado con técnicas de desacoplamiento dinámicas [91,92], los cuales extienden el tiempo de desfasaje del cubit virtual en el orden de microsegundos [82]. La construcción del cubit virtual en PCECO requiere que la Capa 2 pueda ocultar la complejidad de controlar el estado de espín del PC. Debido a que el vector de Bloch del cubit físico rota continuamente alrededor del eje-σ_z, los pulsos de control deben controlarse cuidadosamente para que realicen la operación deseada. Además, el control del espín del PC es complicado por lo inhomogéneo del ambiente del espín nucle-

ar el cual causa la rotación del eje-σ_z para proceder en una incierta frecuencia angular. Este problema es mitigado por la secuencia de Desacoplamiento Dinámico (DD), por lo que el sistema se desacopla del ruido ambiental y se pone precisamente en un marco de referencia controlada y en un tiempo predecible. La Fig.3.7(a) ilustra la secuencia de desacople ‘‘8H’’ (así llamada porque usa 8 pulsos de Hadamard), el cual es apropiado para usarse en PCECO. Esta secuencia de control es diseñada para desacoplar un cubit del ruido de desfasaje y para compensar los errores de pulsos sistemáticos en presencia de un fuerte pero lento plazo de deriva fluctuante en el Hamiltoniano del cubit, el cual es el caso para PC controlados ópticamente en presencia de un fuerte campo magnético.

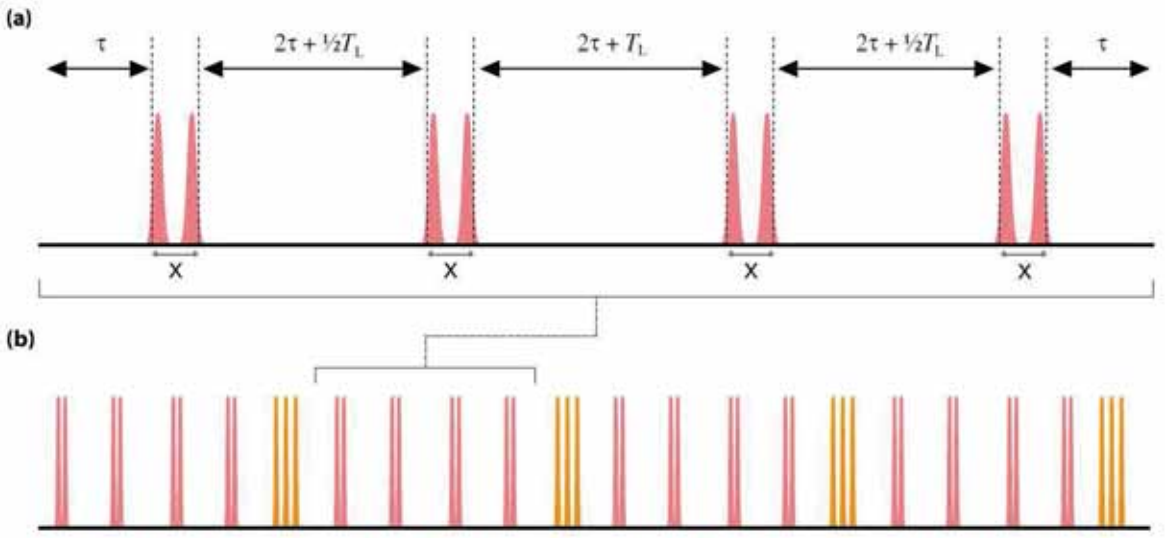


Fig.3.7 Una secuencia de desacople dinámica especial para PCECO, conocida como 8H la cual requiere 8 pulsos de Hadamard. T_L es el período de Larmor determinado por el campo magnético externo (ver Tabla 3.1). (a) Las especificaciones temporales para la secuencia 8H, donde τ es un tiempo arbitrario. Cada uno de los pares de pulsos promulga una rotación π alrededor del eje-σ_x del cubit virtual de la esfera de Bloch, como se uestra en la Fig.3.4. Para que 8H trabaje eficientemente, $\tau << T_2$. (b) Cuatro secuencias 8H en una fila entrelazada con compuertas arbitrarias formadas de 3 pulsos de Hadamard (naranja). La secuencia global forma una compuerta virtual por medio de una secuencia de compensación BB1.

Aunque las secuencias más largas consistentes de más pulsos pueden en teoría desacoplar para una fidelidad más alta, se ha elegido una secuencia de solo ocho pulsos de Hadamard para minimizar el tiempo de ejecución. En lugar de usar una secuencia más común como la de Carr-Purcell (CP) [93,94] o el Desacoplamiento Dinámico de Uhrig (DDU) [95], la secuencia en la Fig.3.7 es diseñada a medida para eliminar de primer orden los errores que se producen tanto en la evolución libre como en el control del cubit virtual. (CP y DDU no pueden cumplir esto último.) Se observa, sin embargo, que la secuencia 8H tiene una estructura similar a la secuencia de CP. La Fig.3.8 muestra la efectividad simulada de 8H comparada con CP y DDU. Se ha seleccionado $\tau = 1$ *ns* [de la Fig.3.7(a)], por lo que una iteración de la secuencia requiere 8 *ns*. Porque esta secuencia es específicamente diseñada para dar cuenta de los errores particulares a PCECO, el rendimiento excede los esquemas de DD más comunes. Sin embargo, 8H puede ser muy efectiva en otros sistemas de información cuántica donde los estados del cubit físico están separados en energía y los pulsos de control tienen una duración que es comparable al período de la precesión-libre (e.g., Larmor) del vector de cubit de Bloch.

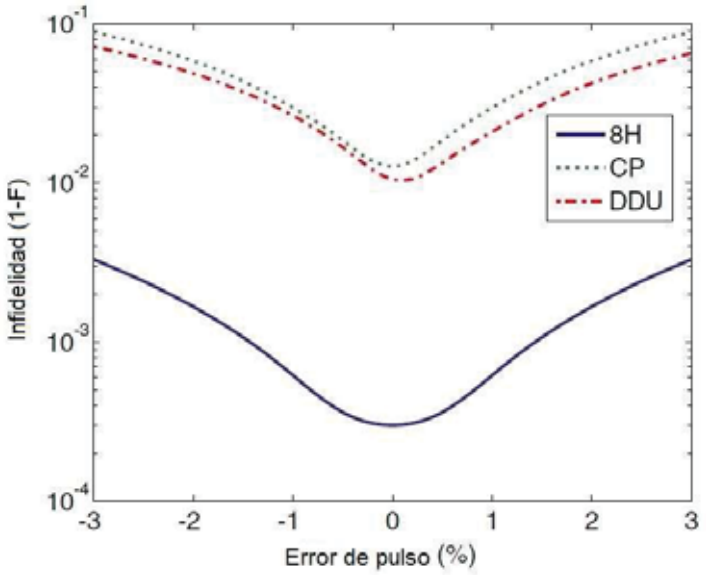


Fig.3.8 Simulación de la eficacia de desacoplamiento de la secuencia 8H comparada a CP y DDU (cada uno usando 4 X compuertas) en presencia del ruido de desfase y los errores de control. Aquí, ‘‘el error del pulso’’ es una desviación sistemática y relativa en la energía de cada pulso. En todos los casos, dos pulsos de Hadamard son combinados para producir una compuerta aproximada X, como en la Fig.3.4. El eje vertical es la infidelidad luego de la evolución de la secuencia en la Fig.3.7(a) con $\tau = 1$ *ns*; aquí, la infidelidad es $1 - F = 1 - \chi_{II}$, donde χ_{II} la matriz de los elementos identidad-a-identidad en el proceso de tomografía para la secuencia en la compuerta de desacople con ruido aleatorio. Dado que nuestro objetivo es ejecutar compuertas virtuales con $1 - F < 10^{-3}$, los errores de pulsos-láser deben ser menores que 1% a fin de que la tasa de error de memoria del cubit virtual sea adecuadamente baja.

3.3.2 Compuerta virtual

Las compuertas cuánticas manipulan el estado del cubit virtual al combinar las operaciones del control físico en la Capa 1 de una forma que crea interferencia destructiva de los errores. Las operaciones cuánticas deben ser implementadas por hardware físico, que es en última instancia hasta cierto punto defectuoso. Muchos errores son *sistemáticos*, a fin de que se correlacionan en el tiempo, incluso si son desconocidos para el diseñador del procesador cuántico. Las compuertas virtuales suprimen los errores sistemáticos tanto como sea posible en orden de satisfacer las demandas del sistema de corrección de error en la Capa 3.

Existen esquemas eficientes para eliminar los errores sistemáticos. Las secuencias de compensación pueden corregir los errores correlacionados en las operaciones de la compuerta en la Capa 1 [96,97]. Esta situación surge a menudo de los errores debidos a las imperfecciones en las operaciones de control, tales como las fluctuaciones en la intensidad del láser o la fuerza del acoplamiento de un electrón de PCs para un campo óptico (causado por imperfecciones de fabricación). Si estos errores están correlacionados sobre escalas de tiempo más grandes que las operaciones en esta arquitectura, una secuencia de compensación es efectiva para generar una compuerta virtual con error neto inferior que cada una de las compuertas constitutivas en la secuencia. Muchas secuencias de compensación son muy generales, por lo que la reducción de errores funciona sin conocimiento del tipo o magnitud del error. Las compuertas corregidas dinámicamente son un esquema alternativo donde uno sintoniza el Hamiltoniano dependiente del tiempo de las operaciones de control [98]. Más allá de estas técnicas de control a lazo abierto, también es deseable

caracterizar la precisión de las operaciones en la capa Virtual, en especial, compuertas multicubit y estados entrelazados. Evaluar sistemáticamente las operaciones cuánticas es un componente importante de un programa de investigación para el desarrollo de computadoras cuánticas y merece mayor investigación, sin embargo, esta fuera de nuestro alcance actual.

En PCECO, los pulsos ultrarápidos en la Capa 1 deberían inducir idealmente una rotación de estado en la base del espín (sistema de dos niveles), pero inevitablemente, el sistema físico sufrirá de cierta pérdida de fidelidad tanto por procesos sistemáticos como aleatorios. Tratamos de causar interferencia destructiva de los errores sistemáticos- tanto del medio ambiente como de los pulsos de control -mediante la incorporación de una secuencia de compensación BB1 dentro de un tren de secuencias de DDs 8H, como se muestra en la figura. 3.7 (b). La secuencia BB1 combina cuatro pulsos con uno desconocido, de sesgo sistemático de una manera que produce la acción de un único pulso que tiene mucha mayor fidelidad. Esta aproximación es motivada por las propiedades del cubit físico. El espín electrónico posee un fuerte pero lento plazo de deriva fluctuante en su Hamiltoniano a causa del campo magnético y el entorno del espín nuclear. La secuencia 8H trae el cubit ‘‘en el foco’’ (análogo al ‘‘eco-espín’’) solo en instantes prescritos, los cuales están cuando los pulsos BB1 son aplicados. Esta aproximación es más exacta que una secuencia BB1 sin reorientación debido al tiempo requerido para implementar las rotaciones sobre el cubit físico de la esfera de Bloch usando pulsos de Hadamard, por las mismas razones que 8H es más efectiva al desacoplar secuencias CP o DDU (como se muestra en la Fig.3.8). La secuencia de compensación BB1 requiere cuatro compuertas arbitrarias [96]; por lo tanto, la compuerta virtual con cancelación de error requiere 32 *ns*.

3.3.3 Medición de los cubits virtuales

La medición es una operación crucial que también debe ser aplicada al cubit virtual en una manera consistente con otros procesos de control. Por ejemplo, el DD impide mediciones por aislación de un cubit de las interacciones del entorno, así DD puede tener que ser suspendida durante la lectura. Dado que la medición juega un papel crucial en la corrección de errores, este mecanismo debe hacerse tan rápido y eficiente como sea posible, y un proceso de medición lento puede sufrir pérdida de fidelidad si el cubit físico adquiere decoherencia rápido sin DD. Incluso si el proceso de medición es mucho más rápido que la decoherencia del cubit, el ruido clásico en la señal de lectura de medición podría ser una preocupación. Si la capa física provee medición NDC, entonces la capa Virtual puede repetir la medición de un cubit virtual múltiples veces y superar el ruido en los circuitos de lectura mediante una mayoría por encuesta de los resultados discretos de medición. Esta es una forma simple y robusta de suprimir los errores de medición. Por ejemplo, si el pulso de medición óptica en PCECO requiere 1 *ns*, entonces la medición podría repetirse alrededor de 30 veces en la misma ventana de tiempo como una compuerta virtual. Otra posibilidad es acoplar un cubit virtual a una o más cubits ancillas que faciliten la medición [99]. En tal esquema, el proceso de medición en la capa Física podría ser destructivo, pero ya que solo la ancilla se destruye, la nueva acción en el cubit original es la medición de NDC que se puede repetir sin problemas.

La medición del cubit virtual en PCECO requiere que la secuencia DD se detenga, dado que la secuencia 8H interfiere con la lectura. Dado que el pulso de medición está en la base σ_z, las rotaciones alrededor del eje-σ_z del entorno magnético no afectan el resultado. Despreciar DD durante la medición es aceptable dado que el tiempo de relajación longitudinal (T_1) es muy largo comparado con la duración del pulso de medición [100,101].

3.4 Capa III: Corrección de Error Cuántico

La CEC tolerante a fallos es esencial para computación cuántica de gran escala. En la Sec.3.6.2 se analiza la implementación del algoritmo de factorización de Shor el cual requiere un error-por-compuerta del orden de 10^{-15}, el cual es simplemente inviable sobre hardware defectuoso, incluso utilizando las técnicas de la Capa 2 tales como el DD. La acción de corrección de error sobre un sistema de información cuántica es para bajar la entropía en la forma de un síndrome de error; en el proceso donde nuevos recursos son creados en la forma de *cubits y compuertas lógicas*. Considerando que la Capa 2 genera errores correlacionados a ser cancelados, la Capa 3 aisla y remueve los errores arbitrarios, siempre y cuando la tasa de error este por debajo de un umbral [102]. Si se cumple esta condición, el CEC puede producir en principio arbitrariamente cubits y compuertas lógicas de bajo error. Tal supresión completa de errores es necesaria debido a que los algoritmos cuánticos en la capa de Aplicación asumen que los cubits y las compuertas lógicas están libres de errores.

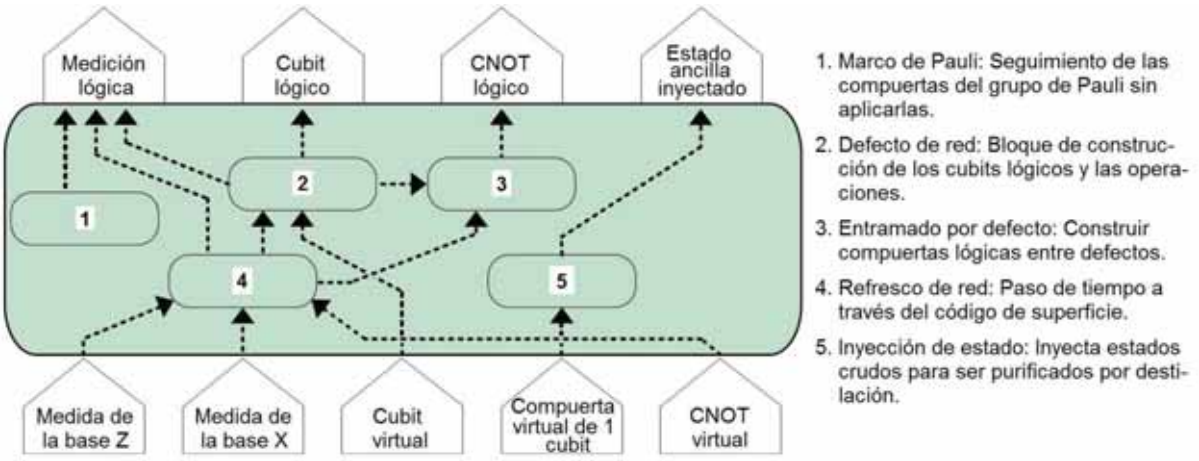


Fig.3.9 Procesos de traducción en la Capa 3 de PCECO. Un código de superficie es construido con cubits y compuertas virtuales, en última instancia dando cubits y operaciones lógicas. Las flechas en la parte baja son salidas de la Capa 2, mientras que las fleches en la parte superior son las salidas de la Capa 3. Las flechas negras punteadas indican que la salida de uno de los procesos es usada por otro proceso.

La CEC se ha convertido también en un campo muy amplio para ser abarcado completamente [71,103,104]. En su lugar, se analiza el caso de los códigos estabilizadores [105], y se considera específicamente el código de superficie [19,106,107] para PCECO. Se selecciona el código de superficie por su alto umbral y su geometría de interacción de vecino más próximo bidimensional. Esta sección se enfoca sobre los aspectos de CEC que son relevantes para una arquitectura de computadora cuántica, tales como determinar el tamaño código suficiente para una determinada aplicación, así como la forma de los errores que son seguidos por los marcos de Pauli en el hardware clásico. La Fig.3.9 muestra las operaciones de código de superficie en la Capa 3 para PCECO. Otros esquemas de corrección de errores además del código de superficie podrían ser también implementados en una arquitectura multicapa. Como ejemplo, los esquemas del código C4/C6 [108,109] y los códigos de Bacon-Shor [49-51,110] también han recibido significante atención como esquemas viables para computación cuántica tolerante a fallos. Debido a que la arquitectura en capas es modular, es posible reemplazar al código de superficie con otro esquema CEC siempre y cuando la capa Virtual soporte las operaciones necesarias del nuevo código. En general, el código CEC elegido impactará probablemente en muchos aspectos de un sistema de

computación cuántica, tales como la geometría del dispositivo, la conectividad, y la sensibilidad a los componentes defectuosos, de manera que la estructura y el comportamiento de la computadora se definen en gran parte por el código seleccionado. Por esta razón, se debería dedicar mucha atención a la optimización de la Capa 3 en cualquier arquitectura de computadora cuántica. Uno de los mensajes claves que vale la pena enfatizar es "el *umbral de la tasa de error* de un CEC se define como la tasa de error en la cual la corrección del error comienza a mostrar una ganancia neta en la protección de la información". Un sistema de CEC funcional debe operar por debajo del umbral, y un sistema *práctico* debe operar muy por debajo del umbral. En esta sección se demuestra que los recursos necesarios para la corrección de errores se vuelven manejables cuando la tasa de error del hardware se encuentra aproximadamente un orden de magnitud por debajo del umbral de la tasa de error del código elegido.

3.4.1 Estimando el poder necesario de corrección de errores

Se considera cómo estimar el grado de corrección de error requerido para una aplicación dada en virtud que esto determina la cantidad necesaria de recursos en la computadora. Los esquemas de CEC generan espacios de código protegido en un gran espacio de Hilbert formado de muchos cubits. La solución de compromiso para reducir los errores lógicos consiste en que en lugar de requerir un único cubit, la computadora cuántica ahora requiere muchos cubits virtuales para producir un cubit lógico. El número de cubits virtuales requeridos para un único cubit lógico es una cantidad importante en el uso de recursos. Esto depende de aspectos de rendimiento de la computadora cuántica teles como:

I. Error por compuerta virtual (EPCV: ε_V), la cual es una entrada a la Capa 3 desde la Capa 2,
II. umbral de EPCV del código de corrección de errores (ε_{umbral}),
III. distancia (d) del código,
IV. máximo error por compuerta lógica (ε_L), que está limitada en el extreme superior por los requerimientos de rendimiento del algoritmo cuántico en la Capa 5.

Para determinar ε_L, la aproximación más simple, al producto KQ, asume el peor caso. Si el algoritmo cuántico tiene un circuito con profundidad lógica K actuando sobre los Q cubits lógicos, entonces la probabilidad máxima de fallas está dada por

$$P_{falla} = 1-\left(1-\varepsilon_L\right)^{KQ} \approx KQ\varepsilon_L \tag{3.1}$$

para un pequeño ε_L. Por lo tanto, se exige que $\varepsilon_L << 1/KQ$. Dadas estas cantidades, el error promedio por compuerta lógica en un código operando bien por debajo del umbral puede aproximarse suficientemente [71,102,103,111–113] por

$$\varepsilon_L \approx C_1\left(C_2\frac{\varepsilon_V}{\varepsilon_{umbral}}\right)^{(d+1)/2} \tag{3.2}$$

donde C_1 es una constante determinada por la implementación específica del código, $C_2 \sim 1$, y, por supuesto, $\varepsilon_V \approx \varepsilon_{umbral}$. Los datos en [113] sugieren $C_1 \approx 0.13$ y $C_2 \approx 0.61$ para el código de superficie, el cual se usa ahora como un ejemplo. Dado un ε_V conocido y las cantidades de códi-

go específico $\{\varepsilon_{umbral}, C_1, C_2\}$, uno puede determinar la distancia *d* necesaria tal que la probabilidad de falla de un algoritmo cuántico completo sea suficientemente pequeña. Por comparación, Aliferis presenta un análisis similar para códigos concatenados tales como el código de Bacon-Shor [114].

Tabla 3.2 Parámetros determinantes del tamaño del código de superficie en PCECO para una implementación del algoritmo de factorización de Shor.

Parámetros	Símbolo	Valor
Umbral de EPCV [113]	ε_{umbral}	9×10^{-3}
EPCV	ε_V	1×10^{-3}
Profundidad del circuito (ciclos de refresco de la red)	K	1.6×10^{11}
Cubits lógicos (Shor, [17])	Q	72708
Error por ciclo de refresco de red	ε_L	2.6×10^{-20}
Distancia del código de superficie	d	31
Cubits virtuales por cubit lógico	n_V/n_L	6240

La Ec.(3.2) ilustra que el EPCV debería ser $\varepsilon_V < 0.2\varepsilon_{umbral}$; de otra manera, la distancia del código, y por lo tanto el tamaño de la computadora cuántica, sería impracticablemente grande. La Tabla 3.2 provee un ejemplo de estos cálculos para la computadora cuántica PCECO. El EPCV (ε_V) también se asume, y los valores *K* y *Q* son para que el algoritmo de Shor factorice un estero de 1024-bit (ver Sec.3.6.2). Se requiere que $\varepsilon_L \leq 10^{-2} / KQ$, de modo que la probabilidad de error lógico del algoritmo cuántico es menor que 1%. La determinación de la fuerza necesaria para la corrección de errores también indica qué tan grande es la computadora cuántica en función de los cubits. Podemos estimar el número de cubits virtuales por cubit lógico, o n_V/n_L, teniendo en cuenta la superficie mínima necesaria para los dos defectos en la red, que constituyen un cubit un cubit lógico, separados por la distancia *d* en el código de superficie [19]. Para un conjunto típico de parámetros, que puedan ser necesarios en una aplicación de computación a gran escala tal como el algoritmo de factorización de Shor [115], son necesarios 6240 cubits virtuales para construir un cubit lógico. Esta es una sobrecarga no trivial, dado que los algoritmos cuánticos requieren un número sustancial de cubits lógicos, como se discute con mayor detalle en secciones subsecuentes. Por ejemplo, los algoritmos de simulación cuántica pueden requerir entre 1000 y 10000 cubits lógicos [116,117] mientras que la factorización de enteros puede requerir 100000 o más cubits lógicos, dependiendo de los métodos de cálculo aritmético [118]. Mas detalles sobre por qué tantos cubits lógicos son necesarios se dan en la Sec.3.5.2. Combinando el tamaño de los cálculos cuánticos con los requerimientos de corrección de error significa que las arquitecturas de computación cuántica de gran escala requerirán millones o miles de millones de cubits virtuales (y por lo tanto cubits físicos).

3.4.2 Marcos de Pauli

Un marco de Pauli [108,119] es una técnica de computación clásica simple y eficiente para seguir los resultados de aplicar una serie de compuertas de Pauli (X, Y, o Z) a cubits únicos. El teorema de Gottesman-Knill implica que el seguimiento de las compuertas de Pauli puede ser hecho eficientemente sobre una computadora clásica [120]. Muchos códigos de CEC, tales como el código de superficie, proyecta el estado codificado en una palabra de código perturbado con erróneamente aplicadas compuertas de Pauli de cubit único (relativo a estados en el espacio de código).

El síndrome, el cual es el conjunto de mediciones que identifica la configuración más probable del error, revela que estos errores de Pauli están, hasta estabilizadores indetectables y operadores lógicos, y la corrección de errores se consigue mediante la aplicación de esas mismas compuertas de Pauli a los cubits apropiados (dado que las compuertas de Pauli son Hermíticas y unitarias). No obstante, las compuertas cuánticas son defectuosas, y la aplicación de compuertas adicionales puede introducir más errores en el sistema. En lugar de aplicar cada operación de corrección, se puede mantener un seguimiento de cómo la operación de corrección de Pauli *debería ser aplicada*, y continuar con el cálculo. Esto es posible porque las operaciones necesarias para la corrección de errores están en el grupo de Clifford. Cuando en una medición de las bases de Pauli X, Y, o Z se realizan finalmente sobre un cubit, el resultado se modifica basado en la compuerta de Pauli correspondiente que debería haber sido aplicada anteriormente, como en la Fig.3.10. Esta compuerta de Pauli almacenada se la llama marco de Pauli [108,119], ya que, en lugar de aplicar una compuerta de Pauli, la computadora cuántica *cambia el marco de referencia para el cubit*, al cual se lo puede entender como la reasignación de los ejes sobre la esfera de Bloch, en lugar de mover el vector de Bloch.

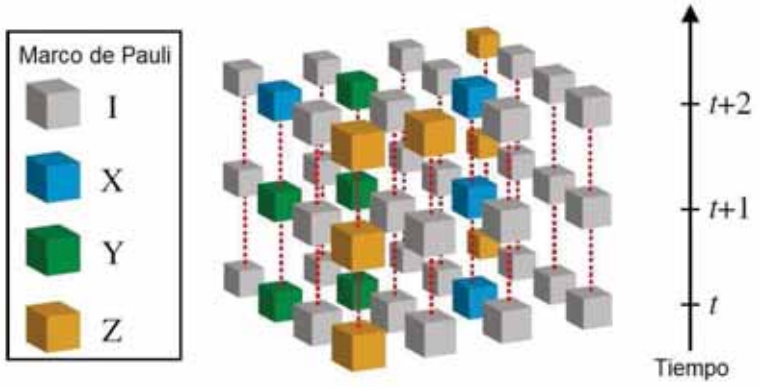


Fig.3.10 Ejemplo de un marco de Pauli evolucionando en el tiempo con las entradas correspondientes a los cubits virtuales que forman un código de superficie. Cada corte horizontal es un marco de Pauli en ese tiempo. Por ejemplo, si el cubit en la posición de la esquina superior frontal as medida en la base *X*, el resultado interpretado es la negación del resultado observado, dado que el marco de Pauli *Z* anticonmuta con esta base de medición.

El marco de Pauli se mantiene de la siguiente manera. Denotemos el marco de Pauli en el momento *t* como F_t:

$$F_t = \bigotimes_j P_t(j), \tag{3.3}$$

donde $P_t(j) = \{I, X, Y, Z\}$ es un elemento del grupo de Pauli correspondiente al cubit *j* en el momento *t*. Cualquier compuerta de Pauli en el circuito cuántico es multiplicada en el marco de Pauli y no es *implementada* en hardware, así $F_{t+1} = \left(\otimes_j U_{\{I,X,Y,Z\}}\right) F_t$ para todas las compuertas de Pauli $U_{\{I,X,Y,Z\}}$ en el circuito en el tiempo *t*. Otras compuertas U_C en el grupo de Clifford son *implementadas*, pero ellas transformarán al marco de Pauli por

$$F_{t+1} = U_C F_t U_C^{\dagger}, \tag{3.4}$$

Las operaciones de computadora cuántica proceden normalmente, con el único cambio del tipo de la medición final de ese cubit interpretado. El conjunto de las compuertas de Clifford es suficien-

te para la Capa 3, aunque la siguiente sección describe otro marco de Pauli para operaciones lógicas que no son de Clifford.
Enfatizamos que el marco de Pauli es un *objeto clásico* almacenado en la circuitería digital que manipula corrección del error. Los marcos de Pauli son sin embargo muy importantes para el funcionamiento de una computadora cuántica de código de superficie. La Capa 3 usa un marco de Pauli con una entrada por cada cubit virtual en el código de corrección de errores. Como se producen errores, el paso de procesamiento síndrome identifica al patrón más probable de los errores de Pauli. En lugar de aplicar directamente la etapa de recuperación, el marco de Pauli se adapta en la memoria clásica. Las compuertas de Pauli forman un grupo cerrado bajo la multiplicación (y la fase global del estado cuántico no es importante), así el marco de Pauli sigue solo a uno de cuatro valores (X, Y, Z, o I) para cada cubit virtual en la red.

3.5 Capa IV: Lógica

La capa Lógica toma los recursos tolerantes a fallas de la Capa 3 y crea un sustrato lógico para la computación cuántica universal. Esta tarea requiere procesamiento adicional de las compuertas y los cubits correctores de errores para producir cualquier compuerta arbitraria requerida en la capa de Aplicación, como se muestra en la Fig. 3.11.

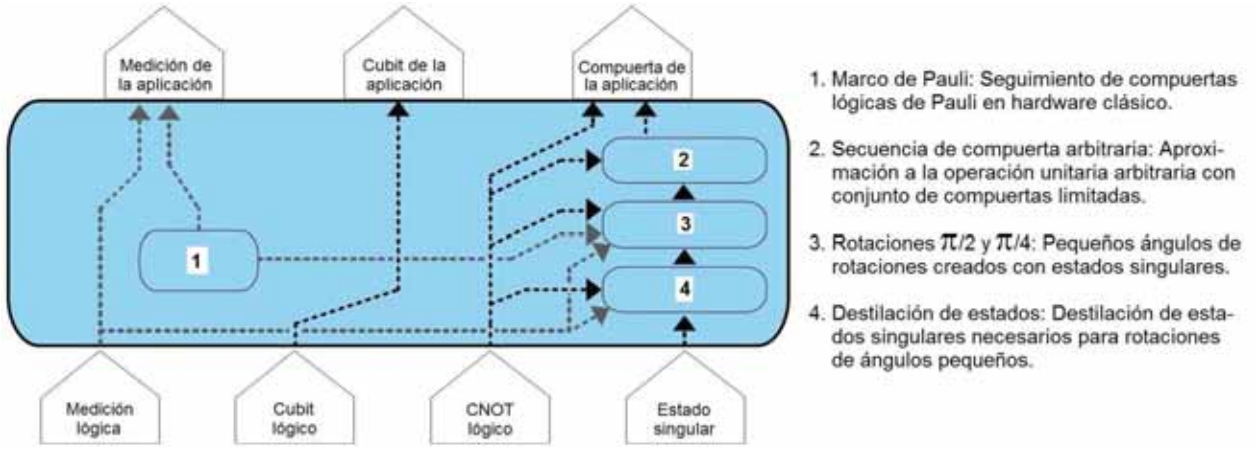


Fig.3.11 Organización del proceso en la capa lógica. Los cubits lógicos de la Capa 3 son inalterados, no obstante, los estados singulares defectuosos se destilan en estados de alta fidelidad $|Y\rangle = \frac{1}{\sqrt{2}}(|0\rangle + i|1\rangle)$ y $|A\rangle = \frac{1}{\sqrt{2}}(|0\rangle + e^{i\pi/4}|1\rangle)$.
Los estados destilados son usados para crear compuertas arbitrarias con circuitos cuánticos especializados [56,122–124].

El CEC provee solo un conjunto limitado de compuertas—Para ver por qué, consideremos que un número no finito de bits síndrome pueden distinguir arbitrariamente errores pequeños de compuertas de rotación. Un conjunto común de compuertas provistas por el CEC es el grupo de Clifford; aunque los circuitos de este conjunto pueden ser simulados eficientemente sobre una computadora clásica por el teorema de Gottesman-Knill [71], los grupos de Clifford forman la columna vertebral de los circuitos cuánticos. Sin embargo, algunos esquemas CEC, tales como el código de superficie, no ofrecen todo el grupo de Clifford sin algún tipo de ancilla. Identificamos al conjunto de compuertas tolerante a fallos generadas por la Capa 3 sin el uso de ancillas como las compuertas fundamentales. La compuerta Lógica entonces construye compuertas arbitrarias

de circuitos de *compuertas fundamentales* y las ancillas inyectadas en el código corrector de error. Por ejemplo, las arquitecturas de código de superficie inyecta y purifica las ancillas $|Y\rangle = \frac{1}{\sqrt{2}}(|0\rangle + i|1\rangle)$ y $|A\rangle = \frac{1}{\sqrt{2}}(|0\rangle + e^{i\pi/4}|1\rangle)$; entonces el código de superficie consume estas ancillas en circuitos cuánticos para producir compuertas $S = e^{i(\pi/4)\sigma_z}$ y $T = e^{i(\pi/8)\sigma_z}$, respectivamente [19,71]. En esta sección se discuten las funciones importantes de la capa Lógica: implementando los marcos lógicos de Pauli; destilando los estados ancilla como $|Y\rangle$ y $|A\rangle$; implementando el grupo completo de Clifford en el código de superficie sin las mediciones; y aproximando arbitrariamente las compuertas cuánticas para la capa de Aplicación.

3.5.1 Compuertas fundamentales y marco lógico de Pauli

Las compuertas fundamentales son provistas originalmente por el código de corrección de errores en la Capa 3. Por ejemplo, la Tabla 3.3 muestra las compuertas fundamentales usadas en PCECO. En la práctica, las compuertas de Pauli son implementadas con un marco de Pauli *lógica*, que es cualitativamente la misma que la de Pauli en la Capa 3 para cubits virtuales (Sec.3.6.2). No obstante, en la Capa 4 también podemos necesitar aplicar compuertas U_{NC} fuera del grupo de Clifford (GC). La compuerta que se implementa actualmente, U_{GC}^{I}, resulta de una transformada del marco de Pauli:

$$U_{NC}^{I} = F_t U_{NC} F_t^{\dagger}. \tag{3.5}$$

Notar la distinción entre esta expresión y la Ec.(3.4), donde el marco de Pauli es cambiado por una compuerta Clifford. El conjunto de compuertas fundamentales en la Tabla 3.3 es particular del código de superficie, y no es un grupo completo de Clifford dado que está desaparecida la compuerta de fase *S*. La Sec,3.5.3 ilustra un método para construir *S* usando una ancilla, sin la medición, que es eficiente ya que utiliza un número pequeño de compuertas fundamentales y la ancilla puede ser reusada. Las compuertas lógicas restantes para producir un conjunto universal en el código de superficie requieren estados ancilla los cuales son inyectados y destilados [19].

Tabla 3.3 Compuertas fundamentales en PCECO usando CEC de código de superficie [19]. El tiempo de ejecución aquí es una posible implementación, pero en muchos casos el cálculo del código de superficie puede ser deformado en otros circuitos topológicamente equivalentes lo cual da una ejecución más rápida a expensas de más recursos espaciales o viceversa.

Compuerta	Implementación	Tiempo de ejecución (pasos de la red)
X, Y, Z	Marco de Pauli	Instantáneos
CNOT	Entramado por defecto	13 [*d*/4]
H (Hadamard)	Desplazamiento de la red	13 [*d*/8]
MX, MZ (medición)	Estabilizadores de la medición	1

3.5.2 Destilación de estado-mágico

El método universal para hacer un conjunto universal de compuertas cuánticas de una forma *tolerante a fallos* es producir un cierto estado ancilla y usarlo en un circuito cuántico equivalente a la compuerta lógica deseada [61,71,114]. En algunos casos estos circuitos requieren medición

que consume la ancilla, por lo que el número de estados ancilla requirido es proporcional al número de compuertas en el algoritmo cuántico. Por ejemplo, los algoritmos discutidos en la Sec.3.6 requieren alrededor de 10^{12} o más estados ancilla, que típicamente se fabrican según sea necesario. Para complicar las cosas, las ancillas deben ser producidas por métodos que no son tolerante a fallos, tales como la inicialización de un cubit virtual y la aplicación de la compuerta virtual correspondiente. Este estado ancilla puede entonces ser *inyectado* en un código CEC en la Capa 3 [19], pero lleva consigo los errores en su producción. Afortunadamente, unos pocos ‘‘estados mágicos’’ pueden ser usados al ver varias ancillas de baja fidelidad y compuertas fundamentales para producir una ancilla de alta fidelidad. Una vez que la fidelidad de la ancilla es más alta que la fidelidad necesaria en la compuerta lógica, podemos construir arbitrariamente compuertas lógicas tolerante a fallos. Examinamos aquí los costos en recursos para este proceso; cada destilación es cara, y muchas ancillas deben ser destiladas. Se caracteriza el rendimiento de la destilación del estado mágico porque probablemente dominará los costos de los recursos de cualquier computadora cuántica que lo utilice. En consecuencia, esta es un área importante para futuras optimizaciones.

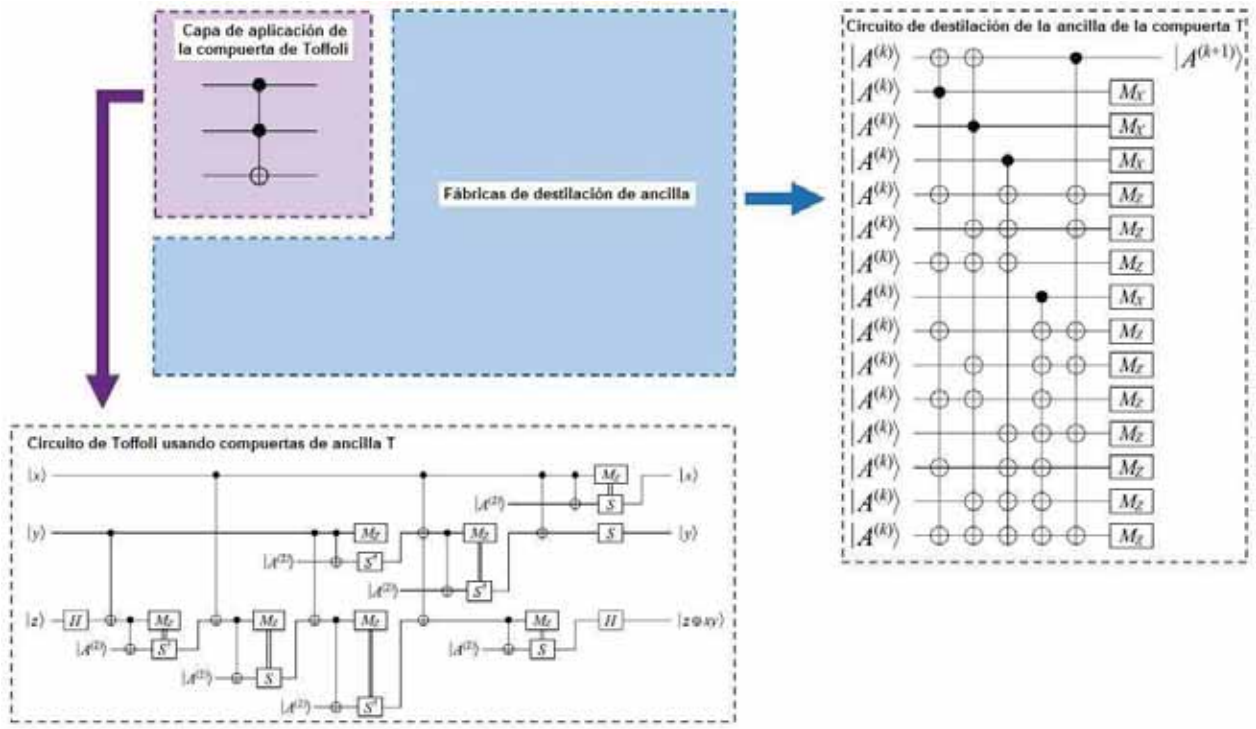


Fig.3.12 Una compuerta de Toffoli $\left(\left|x,y,z\right\rangle \rightarrow \left|x,y,z\oplus xy\right\rangle\right)$ en la capa de aplicación es construida con la asistencia de la capa lógica, usando la descomposición en [71]. Hay solo tres cubits de aplicación, pero sustancialmente más cubits lógicos son necesarios para circuitos de destilación en la Capa 4. Las ancillas $\left|A^{(2)}\right\rangle$ son el resultado de dos niveles de destilación ($\left|A^{(0)}\right\rangle$ es un estado inyectado) sobre la ancilla requerida para las compuertas *T*. Notemos que, en cada tiempo una ancilla es usada con medición, el marco de Pauli puede requerir ser actualizado. El circuito basado en ancilla para compuertas *S* (ver la Fig.3.13) no es mostrado aquí, por claridad.

Nos centramos primero en destilar el estado ancilla $\left|A\right\rangle = \frac{1}{\sqrt{2}}\left(\left|0\right\rangle + e^{i\pi/4}\left|1\right\rangle\right)$, el cual es usado para

construir la compuerta de fase T o $\pi/8$ [19,107,121]. En la próxima sección, la Fig.3.12 provee una ilustración de porqué este proceso es importante al mostrar la construcción tolerante a fallos de una compuerta de Toffoli en la Capa 5 usando recursos en la Capa 4; específicamente, los circuitos de destilación de ancilla constituyen más del 90% del esfuerzo de cálculo para una compuerta de Toffoli única. Como un resultado, el análisis en [17] sostiene que estos circuitos de destilación cuentan con la mayoría de los recursos de una computadora cuántica de código de superficie ejecutando el algoritmo de Shor. En particular, por cada cubit usado por el algoritmo, aproximadamente 10 cubits están trabajando atrás para generar las ancillas destiladas necesarias.

El circuito de destilación de ancilla en la Fig.3.12 muestra uno de los niveles de destilación de $|A\rangle$, pero un programa tan largo como el de Shor típicamente requerirá dos niveles (uno concatenado con otro). Además, dado que tal vez se necesitarán miles de millones de ancillas destilados $|A\rangle$ para el algoritmo, creamos una ‘‘fábrica de destilación’’ [26,60], la cual es una región específica de la computadora que continuamente produce estos estados lo más rápido posible.

La velocidad es importante, dado que la destilación de ancilla puede ser la etapa de tasa limitada en los circuitos cuánticos [27]. Cada circuito de destilación $|A\rangle$ requerirá 15 estados de nivel más bajo $|A\rangle$, pero no todos ellos serán usados al mismo tiempo. Por simplicidad usaremos un ciclo de reloj para cada compuerta igual al tiempo para implementar CNOT lógica (ver la Sec.3.7 para más detalles sobre este punto), de modo que, con la inicialización y la medición, el circuito de destilación requiere 6 ciclos. Al usar ancillas $|A\rangle$ solo cuando ellas son necesarias, el circuito puede ser compactado para requerir como máximo 12 cubits lógicos en un instante dado.

Caracterizamos el esfuerzo computacional por un ‘‘volumen circuital,’’ el cual es el producto del espacio de memoria lógica (i.e., área de la computadora) y el tiempo. El volumen del circuito de destilación de $|A\rangle$ es $V\left(\left|A^{(1)}\right\rangle\right)$ = (12 cubits lógicos) x (6 ciclos de reloj) = 72. Una destilación de dos niveles requerirá 16 circuitos de destilación, o un volumen circuital de $V\left(\left|A^{(2)}\right\rangle\right)=1152$. Una fábrica de destilación eficiente con un área $A_{fábrica}$ producirá un promedio $A_{fábrica}/V\left(\left|A^{(2)}\right\rangle\right)$ de ancillas destiladas por ciclo de reloj. Un análisis de este problema en el contexto del algoritmo de Shor está dado en [17], y en la Tabla 3.4 hay ua lista con un resumen de estos resultados.

Tabla 3.4 Análisis de recursos para una fábrica de destilación. Estas fábricas son cruciales para las computadoras cuánticas que requieren ancillas para compuertas universales. La destilación de estado-mágico usa la medición y compuertas de Clifford, de manera tal que el circuito puede ser deformado para reducir la profundidad e incrementar el área, o vice verza, al mismo tiempo que mantiene el volumen aproximadamente constante.

Parámetro	Símbolo	Valor
Profundidad del circuito		6 ciclos de reloj
Area del circuito	A_{destil}	12 cubits lógicos
Volumen del circuito	$V\left(\left\|A^{(1)}\right\rangle\right)$	72 cubits x ciclos
Tasa de fábrica (nivel n)	$R_{fábrica}\left(\left\|A^{(n)}\right\rangle\right)$	$A_{fábrica}/V\left(\left\|A^{(n)}\right\rangle\right)$ Ancillas/ciclo

3.5.3 Compuerta de fase lógica sin medición

La compuerta *S*, o compuerta fase, es la componente final del grupo de Clifford ausente del conjunto fundamental de las compuertas tolerantes a fallos discutidos en la Sec.3.5.1. Implementaciones previas del código de superficie presentan un método para crear esta compuerta para consumir un estado destilado $|Y\rangle$ en un circuito basado en medición proyectiva [19,107]. No obstante, esta aproximación fuerza a la computadora cuántica a destilar una ancilla $|Y\rangle$ de alta fidelidad para cada compuerta *S*, la cual puede ser muy costosa en ambas, las compuertas y los cubits fundamentales. Consideremos un método alternativa que usa la ancilla $|Y\rangle$ sin consumirla para implementar la compuerta *S*, la cual fue originalmente presentada en [114]. El circuito usa solo cuatro compuertas fundamentales y a diferencia de las técnicas previas, es determinística porque la medición no es necesaria. Dado que las ancillas $|Y\rangle$ no son consumidas, uno puede destilar un puñado de esos estados cuando una computadora cuántica está encendida, y entonces preservarlos para un uso posterior.

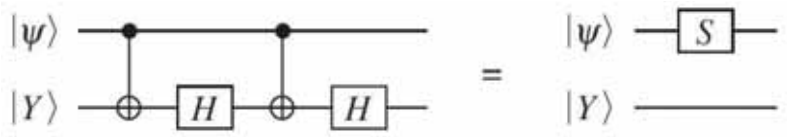


Fig.3.13 Circuito de descomposición para una compuerta lógica *S* que usa una ancilla $|Y\rangle$ pero no la consume. La operación inversa $S^{\dagger}$ puede ser creada al correr este circuito en reversa.

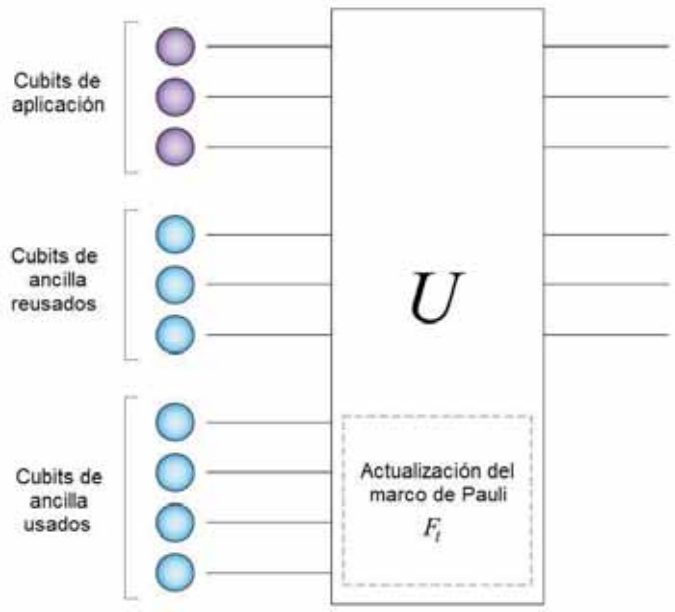


Fig.3.14 Construyendo una compuerta arbitraria en la capa lógica. Los cubits de aplicación están visibles para los algoritmos cuánticos, mientras los cubits de ancilla lógica facilitan el cálculo cuántico universal. Algunas ancilas son reusadas, tales como $|Y\rangle$ para compuertas *S*, mientras que otras ancillas son consumidas, tales como $|A\rangle$ para compuertas *T*. Frecuentemente, cuando una ancilla es consumida en un circuito que usa medición, el circuito es probabilístico, y el marco de Pauli es actualizado condicional sobre el resultado de la medición.

El circuito en la Fig.3.13 es equivalente a una simple compuerta *S* sobre el cubit de control. Esta equivalencia es porque $|Y\rangle$ es el $+i$ autoestado del operador iY, por lo que la compuerta controlada-iY impartirá una fase $+i$ solo si el cubit de control se encuentra en el estado $|1\rangle$, el cual es idéntico a la compuerta *S*. Notemos también que $S^\dagger$ puede ser creada al correr por detrás el circuito en la Fig.3.13. Esta técnica nos permite implementar al grupo de Clifford completo sin medición en el código de superficie. Por otra parte, dado que las compuertas *S* son usadas frecuentemente en algoritmos cuánticos, esta construcción de compuerta mejorada reduce sustancialmente la complejidad de una computadora cuántica debido a que son necesarias menos destilaciones de estados fuente-intensivos.

3.5.4 Aproximación arbitraria a las compuertas lógicas

La función primaria de la compuerta Lógica consiste en descomponer las compuertas unitarias arbitrarias del algoritmo cuántico en circuitos que contengan compuertas fundamentales disponibles de la capa del CEC. Los circuitos en la capa Lógica actúan sobre los cubits de la aplicación (usado explícitamente por el algoritmo cuántico) y los cubits lógicos de ancilla, ambos de los cuales facilitan la computación cuántica universal, como se muestra en la Fig.3.14. Dado que las compuertas cuánticas arbitrarias no están disponibles directamente, deben ser aproximadas de alguna manera, donde los recursos totales requeridos son una función de la exactitud de la aproximación. Cubrimos brevemente algunos de los métodos que se pueden emplear para producir tales compuertas arbitrarias; un estudio más completo de las técnicas se da en [122]. Al construir aproximaciones a una operación unitaria en la capa Lógica, se busca implementar un circuito cuántico que aproxima la operación unitaria deseada con un mínimo gasto en función de compuertas y ancillas producidas por la Capa 3. Denotamos la exactitud de aproximación como

$$\varepsilon_{aprox} = \sqrt{\frac{d - \left| tr\left(U^\dagger U_{aprox}\right)\right|}{d}} \tag{3.6}$$

donde U_{aprox} es el circuito cuántico tolerante a fallos que aproxima la unitaria deseada *U*, y *d* es la dimensionalidad de estas operaciones [123]. Existen varias técnicas para la aproximación a compuertas de cubit único arbitrarias, el cual puede ser generalizado a compuertas multicubit arbitrarias:

I. Las secuencias de aproximación de compuertas, tales como las producidas por el algoritmo de Solovay-Kitaev [71,124] o el algoritmo de Fowler [123], genera una secuencia de compuertas del conjunto tolerante a fallos (e.g., el conjunto {X, Y, Z, H, S, T}) que aproxima a la unitaria deseada *U*. La profundidad de estas secuencias escala como $O\left[\log^c\left(\varepsilon_{aprox}\right)\right]$ con $c \approx 4$ para secuencias Solovay-Kitaev y $O\left[\log\left(\varepsilon_{aprox}\right)\right]$ para secuencias de Fowler.

II. La fase de retroceso usa un registro especial de ancilla y un sumador cuántico para producir rotaciones de fase tolerantes a fallas [56,125,126]. La profundidad de los circuitos de fase de retroceso es $O\left[\log\left(\varepsilon_{aprox}\right)\right]$ o $O\left[\log\log\left(\varepsilon_{aprox}\right)\right]$, dependiendo del sumador cuántico [127–129]. El registro ancilla, el cual no es consumido y puede ser reusado, tiene un tamaño de *m* cubits para aproximar a una rotación de fase para una precisión de $\pi / 2^m$ radianes, el cual es también $O\left[\log\left(\varepsilon_{aprox}\right)\right]$.

III. ‘‘Las compuertas de teleportation’’ [46] que pueden producir circuitos cuánticos muy rápidos, requieren por lo general una ancilla de propósito especial para cada una de las compuertas que deben ser calculadas de antemano, las cuales demandan una computadora cuántica más grande y más compleja. Las compuertas de teleportation que aumentan el rendimiento en computación cuántica de gran escala son usadas extensamente en la arquitectura de [27] y en los algoritmos de simulación analizados en [122].

Elegir entre estos métodos depende de las capacidades de la arquitectura cuántica, tales como los cubits lógicos disponibles para cálculo paralelo, y sobre las características de rendimiento deseadas para la computadora.

3.6 Capa V: Aplicación

La capa de Aplicación es donde los algoritmos cuánticos son ejecutados. Los esfuerzos de la Capa 1 a la 4 han producido un sustrato de cálculo que suministra cualquier compuerta arbitraria que sea necesaria. La capa de Aplicación es por lo tanto la que no se ocupa de los detalles de implementación de la computadora cuántica—Es un entorno ideal de programación cuántica. No introducimos aquí ningún método algorítmico nuevo, sino que estamos interesados en la forma de estimar con precisión los recursos de computación cuántica necesarios para una aplicación objetivo. Este análisis puede indicar la posibilidad de un diseño propuesto de computadora cuántica, la cual es una consideración que vale la pena en la evaluación de las perspectivas a largo plazo de un programa de investigación de la computación cuántica.

Un ingeniero cuántico debería comenzar aquí en la Capa 5 con una aplicación específica en mente y trabajar por las capas para determinar el diseño del sistema necesario para realizar la funcionalidad deseada. Tomamos esta aproximación para PCECO al examinar dos interesantes algoritmos cuánticos: el algoritmo de factorización de Shor y la simulación de Química Cuántica. Un diseño riguroso del sistema está más allá del alcance del presente trabajo, no obstante, consideremos los recursos computacionales requeridos para cada aplicación con suficiente detalle como para que uno pueda evaluar el esfuerzo de Ingeniería necesario para diseñar una computadora cuántica basada en la tecnología PCECO.

3.6.1 Elementos de la capa de aplicación

La capa de Aplicación está compuesta de cubits y compuertas de aplicación que actúan sobre los cubits. Los cubits de aplicación son cubits lógicos usados explícitamente por un algoritmo cuántico (ver Fig.3.14). Como se discutió en la Sec.3.5, muchos cubits lógicos son usados también para destilar estados ancilla necesarios para producir un conjunto universal de compuertas, pero estos cubits lógicos de destilación no son visibles para el algoritmo en la Capa 5. Cuando un análisis de un algoritmo cuántico cita a un número de cubits sin referencia a la corrección de error tolerante a fallos, esto significa frecuentemente que se refiere al número de cubits de aplicación [116,130–132]. Similarmente, las compuertas de la capa de Aplicación son equivalentes en la mayoría respecto a las compuertas lógicas; la distinción se hace de acuerdo a qué recursos están visibles para el algoritmo o deliberadamente ocultos en la maquinaria de la capa Lógica, que ofrecen un margen de apreciación al diseñador del equipo. Un algoritmo cuántico podría solicitar cualquier compuerta arbitraria en la Capa 5, pero no todas las compuertas cuánticas son iguales en términos de costos de los recursos. Vemos en la Sec.3.5.2 que destilando ancillas $|A\rangle$, las cuales son necesarias para las compuertas *T*, es un proceso muy costoso. Por ejemplo, la Fig.3.12 muestra cómo las

Capas 4 y 5 se coordinan para producir una compuerta de Toffoli de la capa de Aplicación, que ilustra el grado en que la destilación de anchilla consume recursos en la computadora. Cuando se incluye la preparación de ancilla, las compuertas *T* pueden dar cuenta de más del 90% de la complejidad del circuito en un algoritmo cuántico tolerante a fallos (así como en [27]). Por esta razón, contamos con los recursos para su aplicación en términos de las compuertas de Toffoli.

Esta es una elección natural, debido a que el nivel de destilación de ancilla, el número de cubits virtuales, etc., dependen de la elección del hardware, la corrección del error, y muchos otros parámetros específicos del diseño. Por comparación, el número de compuertas de Toffoli es independiente de la máquina ya que esta cantidad depende sólo del algoritmo (al igual que el número de cubits de aplicación mencionados anteriormente).

Para determinar la corrección de error o los recursos de hardware para un algoritmo dado, uno puede tomar las estimaciones de los recursos de la Capa 5 y trabajar a través de las Capas 4 a 1, que es un ejemplo de modularidad en este marco arquitectónico. Usando el análisis en las secciones precedentes, una compuerta de Toffoli de la capa de Aplicación en PCECO tiene un tiempo de ejecución de 930 μs (31 ciclos de compuerta lógica incluyendo los circuitos de compuerta *S*, discutidos en la Sec.3.7).

3.6.2 Algoritmo de Shor

Tal vez, la aplicación más ampliamente difundida de las computadoras cuánticas sea el algoritmo de Shor, el cual descompone a un entero en sus factores primos [115]. Resolver eficientemente el problema de factoreo comprometería al criptosistema RSA (nombrado así en honor a las siglas de los apellidos de sus autores [133]). Debido a la importancia del algoritmo de Shor en el campo de la computación cuántica tolerante a fallos de gran escala, analizamos los recursos requeridos para factorizar un número de tamaño típico para la criptografía RSA.

Una clave de longitud típica para la criptografía de clave pública RSA es 1024 bits. Factorizar un número de este tamaño no es trivial, aun sobre una computadora cuántica, como lo muestra el siguiente análisis.

La Fig.3.15 muestra el tiempo de ejecución esperado sobre una PCECO para una iteración del algoritmo de Shor versus la longitud de la clave en bits para dos diferentes computadoras cuánticas: aquel en el que el tamaño del sistema aumenta con el tamaño del proble-ma, y uno donde el tamaño del sistema está limitado a 105 cubits lógicos (incluyendo cubits de aplicación). Para la computadora cuántica de tamaño fijo, el tiempo de ejecución comienza a crecer más rápido que la profundidad mínima del circuito cuando se factorizan números de 2048 bits y superiores. Fijar el tamaño de la máquina pone de relieve la importancia de las fábricas de destilación de la ancilla. Para esta instancia del algoritmo de Shor, alrededor del 90% de la máquina debería estar dedicada a la destilación; si los recursos insuficientes son dedicados a la destilación, la performance del algoritmo de factoreo se desploma. Por ejemplo, la factorización de 4096-bits dedica alrededor del 75% de la máquina a la destilación, pero se necesitarían alrededor de 3 veces la cantidad de fábricas para lograr máxima velocidad de ejecución de la traza inferior en la Fig.3.15. Muchos parámetros de diseño en una implementación del algoritmo de Shor se pueden ajustar como se desee; recogemos los detalles de nuestro análisis en la Sección 3.6.3.1. Deberíamos mencionar también aquí que el algoritmo de Shor es probabilístico, de manera tal que pueden ser requeridas unas pocas iteraciones [115].

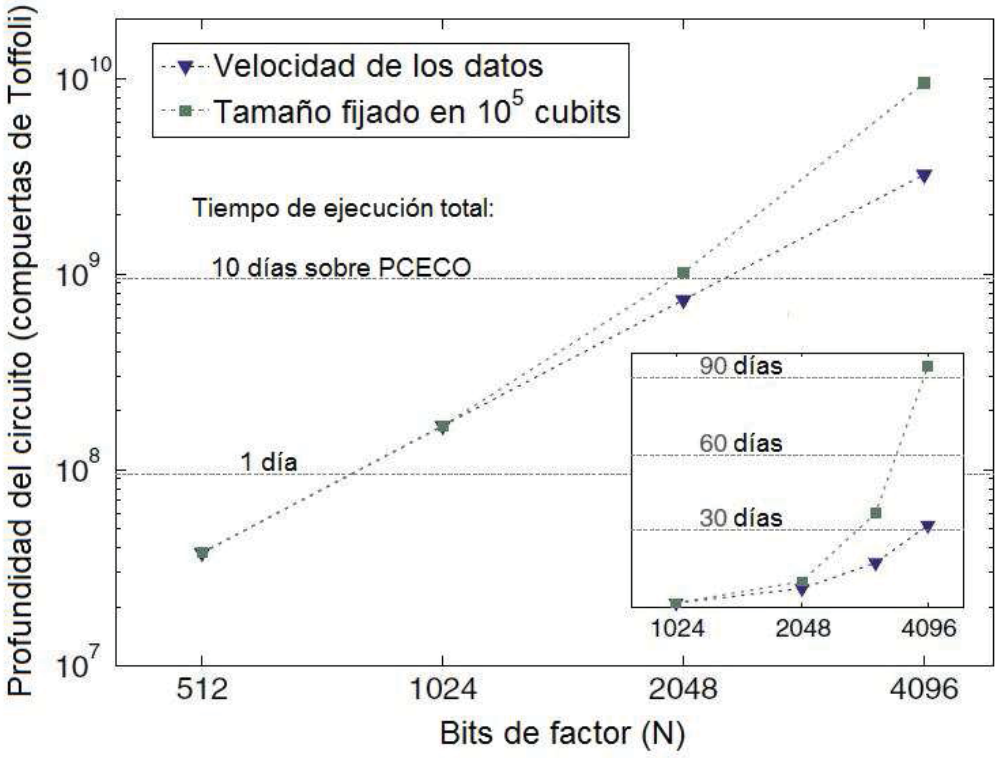


Fig.3.15 Tiempo de ejecución para el Algoritmo de Shor, usando la misma implementación circuital de [60]. El eje vertical muestra la profundidad circuital, en función de las compuertas de Toffoli, y el gráfico está rotulado con tiempo de ejecución estimado sobre la arquitectura PCECO. El trazo azul (con triángulos) es una computadora cuántica cuyo tamaño en escalas de cubits lógicos es necesaria para calcular la velocidad de los datos (no los retardos). El trazo verde (con cuadrados) es una máquina con 105 cubits lógicos, la cual experimenta un rápido incremento de los retardos en virtud de que el tamaño del problema aumenta más allá de 2048 bits, ya que los recursos disponibles son insuficientes para destilar ancillas para compuertas *T*, que son un componente necesario del algoritmo de Shor. El recuadro muestra los mismos datos sobre una escala vertical lineal, ilustrando cuando la computadora cuántica experimenta retardos por falta de cubits suficientes.

3.6.3 Simulación Cuántica

Las computadoras cuánticas fueron inspiradas por el problema que simular sistemas cuánticos sobre una computadora clásica es fundamentalmente difícil. Feynman postuló que un sistema cuántico debería simular otro mucho más eficientemente que un procesador clásico, y propuso un procesador cuántico para llevar a cabo dicha tarea [134]. La simulación cuántica es uno de los pocos algoritmos cuánticos conocidos que resuelve un problema útil que se cree insoluble en las computadoras clásicas, por lo que analizamos las necesidades de recursos para la simulación cuántica en la arquitectura cuántica que proponemos.

Consideramos en particular la simulación cuántica tolerante a fallos. Otros métodos de simulación se encuentran actualmente bajo investigación [135-137], pero están fuera del alcance de este trabajo. El ejemplo particular que hemos seleccionado es similar la ecuación de Schrödinger para un Hamiltoniano independiente del tiempo en la primera forma cuantizada, donde cada Hamiltoniano representa la configuración electrón-nuclear en una molécula [117,138]. Un caso de uso de tal simulación consiste en determinar los niveles de energía basal y de estado excitado en una

molécula. Seleccionamos la forma primero-cuantizada en lugar de segunda-cuan-tizada para un mejor escalado de los recursos en problemas de gran tamaño [139].

La Fig.3.16 muestra el tiempo necesario para ejecutar el algoritmo de simulación a efectos de determinar un estado propio de energía de la computadora PCECO como una función del tamaño del problema de simulación, expresado en número de electrones y núcleos. La forma primero-cuantizada almacena la información base de posición para una función de onda del electrón en un registro cuántico, y el Hamiltoniano completo es una función de cuerpos de una- y dos- iteraciones entre estos registros. Por estas razones, este método no depende de una estructura u ordenamiento molecular particular; por lo tanto, el método es muy general. Notemos que las escalas de tiempo calculadas linealmente en el tamaño del problema, están en oposición a la escala exponencial vista en los métodos clásicos. La precisión en las escalas de simulación con el número de pasos de tiempo simulados [116]; este ejemplo usa 210 pasos de tiempo para una precisión máxima de alrededor de 3 figuras significativas.

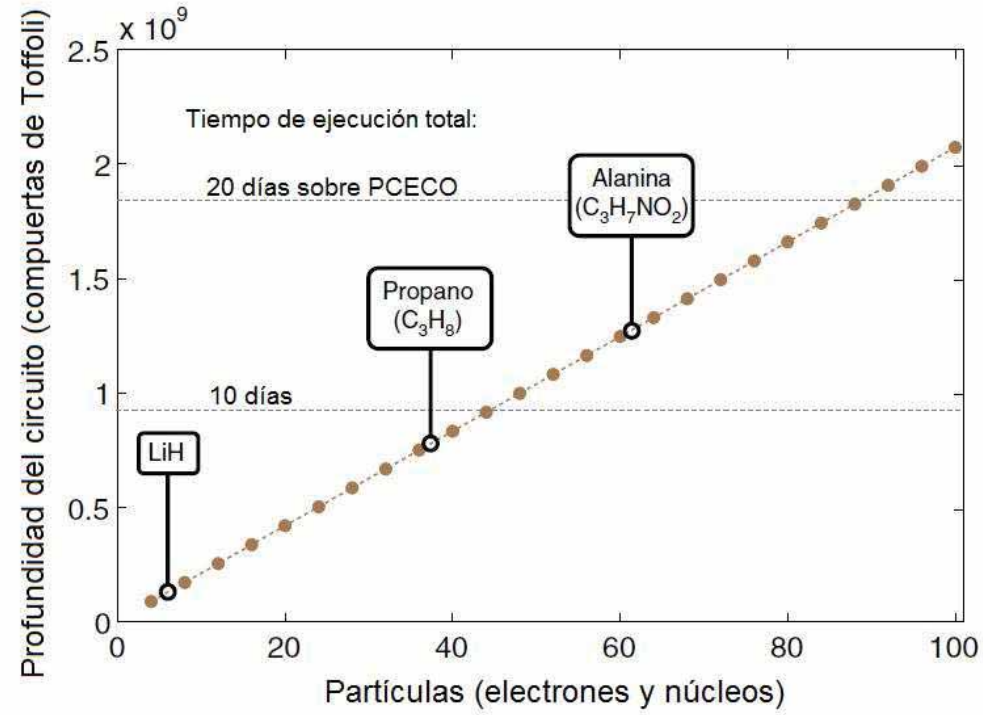


Fig.3.16 Tiempo de simulación para la ejecución de un Hamiltoniano molecular en forma de primera-cuantizada, como una función del tamaño del problema. El eje horizontal es el número de partículas que son simuladas, y el gráfico es rotulado con algunos ejemplos interesantes de la Química. El eje vertical es la profundidad circuital en compuertas de Toffoli, y el gráfico es rotulado con un tiempo de ejecución estimado sobre PCECO. Cada simulación usa 12-bits de precisión espacial en la función de onda y 210 pasos de tiempo para 10-bits de precisión de lectura, o como máximo alrededor de 3 cifras significativas. La escala lineal en el tiempo de ejecución del algoritmo versus el tamaño del problema que se debe al cálculo de energía potencial de dos cuerpos, lo cual constituye la mayoría del circuito cuántico. El número de cálculos de energía potencial se incrementa cuadráticamente con el tamaño del problema, a diferencia de la computación paralela donde se requiere tiempo de ejecución lineal.

Proveemos aquí un breve resumen de la complejidad computacional del algoritmo de Shor y la simulación cuántica en la forma primera-cuantizada.

3.6.3.1 Algoritmo de Shor

Adoptamos la misma implementación del algoritmo de Shor dada en Van Meter *et al.* [60]. Con objeto de determinar la performance del algoritmo de Shor en la Capa 5, debemos mirar cómo la Capa 4 prepara eficientemente a las compuertas de Toffoli. Supongamos que la computadora cuántica tiene una capacidad para 105 cubits lógicos; en general, uno puede intercambiar capacidad lógica y tiempo de ejecución algorítmica. Para factorizar un número de N-bit, son usados aproximadamente $6N$ cubits de aplicación por el algoritmo mismo, con el resto de los cubits lógicos usados para producir las cruciales ancillas $|A\rangle$. Son viables implementaciones con menos cubits de aplicación [130,132], pero la performance de tales circuitos es dramáticamente más lenta, especialmente si uno está restringido a un conjunto limitado de compuertas. Como se muestra en la Sec.3.5.2, una ronda de destilación $|A\rangle$ requiere un volumen de recursos computacionales con sección transversal de 12 cubits lógicos y tiempo de 6 ciclos CNOT. Organizamos el exceso de $(10^5 - 6N)$ cubits en fábricas las cuales destilan ancillas tan rápido como sea posible.

Al igual que antes, se define un ciclo de reloj de la Capa Lógica como una compuerta CNOT. Expresamos la tasa a la cual la fábrica genera ancillas por número medio de ancillas producida por ciclo de reloj. Dos niveles de destilación requerirán 16 circuitos de (15 en el primer nivel, 1 en el segundo nivel), el cual usa un volumen de circuito $V_{\text{destil}} = 16$ x (12 cubits lógicos) x (6 x ciclos de reloj). Para un área de sección de cruce dada $A_{\text{fábrica}}$ de la computadora cuántica dedicada a la destilación, la tasa máxima de la producción de ancilla esta dada por $A_{\text{fábrica}}/V_{\text{destil}}$. Los cálculos de estos valores están dados en Tabla 3.5.

Tabla 3.5 Las tasas de generación y de los consumos máximos para un computador cuántico de 10^5-cubit corriendo el algoritmo de factorización de Shor. Cuando la tasa de consumo de velocidad de datos es más alto que la tasa de destilación, el algoritmo de Shor experimenta retardos.

Bits de factor	Fábrica de ancilla, sección cruzada (cubits lógicos)	Tasa de dastilación ($|A\rangle$ por ciclo)	Máxima tasa de consumo ($|A\rangle$ por ciclo)
512	96928	84.1	32.1
1024	93856	81.5	57.8
2048	87712	76.1	105.1
4096	75424	65.5	192.7
8192	50848	44.1	355.7
16384	1696	1.5	660.6

Necesitamos determinar si el computador cuántico puede correr tan rápido como la profundidad del circuito en la Capa 5, o si la destilación de los estados $|A\rangle$ limita la performance. Usando una construcción del tipo de la Fig.3.14, la profundidad de la compuerta de Toffoli es 31 ciclos de reloj, donde cada compuerta S requiere 4 ciclos como se muestra en la Fig.3.13, y el circuito requiere 7 ancillas $|A\rangle$ destiladas. El circuito usa la construcción de sumador en avance con acarreo de [129], la cual requiere en total aproximadamente $10N$ compuertas de Toffoli con una profundidad de circuito de (alrededor de $4\log_2 N)t_{\text{Toffoli}}$, o alrededor de $124\log_2 N$ ciclos. Usando estas figuras, la tasa de consumo máximo de ancillas puede ser calculada, como se muestra en la Tabla 3.5. A medida que el tamaño del número a ser factorizado aumenta, una computadora cuántica de tamaño fijo es en algún punto incapaz de generar suficientes ancillas para correr el algoritmo a la máxima velocidad; cuando esto pasa, el tiempo de ejecución es limitado por el proceso de

destilación. Podemos hacer una estimación cruda de la Tabla 3.5 en la que una computadora cuántica eficiente para el algoritmo de Shor debe dedicar el 90% de sus recursos a la destilación. Con argumentos similares, una computadora cuántica de tamaño mínimo que mantiene sólo los qubits del algoritmo y uno de destilación de ancilla $|A\rangle$ al mismo tiempo, será muy lenta.

3.6.3.2 Simulación Cuántica

Utilizando el método de [117] para realizar la simulación en la primera-cuantizada. Cada función de onda del electrón es representada sobre una grilla cartesiana tridimensional con 12 bits de precisión en cada dimensión, lo cual requiere un registro cuántico de 36 cubits por partícula. Elegimos usar un conjunto diferente de sumadores y multiplicadores a los empleados en [117], optando en cambio por sumadores de acarreo de propagación únicos que son suficientes para una precisión de 12-bits [128]. Primero, el operador de energía potencial es calculado en la posición base. Transformamos la representación de la función de onda de la posición base al momento base con la Transformada de Fourier Cuántica (TFC), permitiendo una evaluación eficiente del operador de energía cinética. La transformada TFC inversa de nuestro sistema regresa a la posición base. La representación del circuito cuántico del sistema propagador $\mathcal{U}$ se representa en la Fig.3.17.

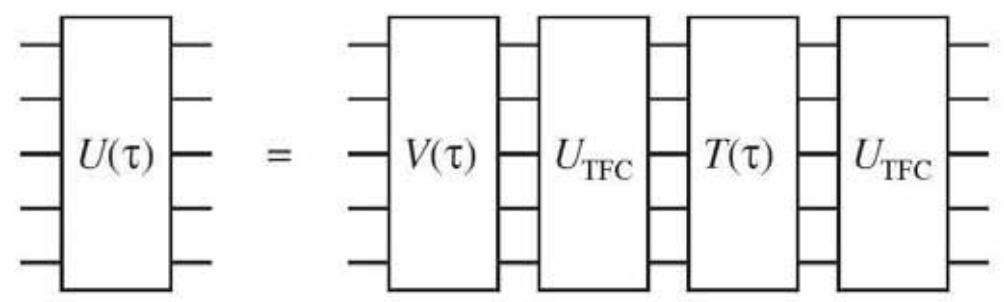


Fig.3.17 Circuito de representación para una iteración del propagador Hamiltoniano en forma primera-cuantizada. La TFC es implementada sobre la función de onda, transformando entre la base de posición y la base de momentum.

Los requerimientos de recursos para cada uno de los operadores cinético ($\mathcal{T}$), potencial ($\mathcal{V}$), y TFC se resumen en la Tabla 3.6. Los parámetros en la Tabla 3.6 fueron derivados asumiendo un cálculo paralelo de los términos del operador de conmutación; por ejemplo, la interacción de Coulomb entre partículas α y β puede ser calculada simultáneamente como γ y δ, porque estos términos en el Hamiltoniano conmuta y los circuitos son disjuntos [122]. Por otra parte, hemos usado aquí el análisis precedente para incluir en estas figuras el tamaño de las fábricas de ancilla, la cual es aproximadamente $260B$ cubits lógicos en order a simular un sistema de B partículas.

Tabla 3.6 Requerimientos de recursos para los operadores en la primera simulación molecular cuantizada con B partículas y una precisión espacial de 12-bit incluyendo la destilación de ancilla.

Operador	Tamaño máximo de memoria (cubits lógicos)	Profundidad del circuito (ciclos de reloj de la Capa 4)
Energía cinética	$334 \times \boldsymbol{B}$	1.55×10^5
Energía potencial	$369 \times \boldsymbol{B}$	$6.26 \times 10^5 \times \boldsymbol{B}$
TFC	$272 \times \boldsymbol{B}$	2.57×10^4

Para este algoritmo de simulación paralela, la producción de ancilla consume cerca del 70% de la computadora cuántica.

La construcción del circuito en la Fig.3.17 es sólo una iteración del sistema propagador. Estimar un autovalor de energía requiere simulación del sistema en pasos de tiempo discreto, de manera tal que el propagador es repetido muchas veces [116], como se muestra en la Fig.3.18. Luego de la evolución del propagador a lo largo de estos pasos de tiempo, el sistema se transforma en la autobase de energía por medio de una operación de TFC sobre el vector de tiempo $|t\rangle$ [71]. La precisión en la respuesta final está limitada por el número de bits en $|t\rangle$, por lo que para este análisis se supone que el sistema evolucionó durante 210 pasos de tiempo, lo cual ofrece al menos cerca de 3 dígitos decimales de precisión.

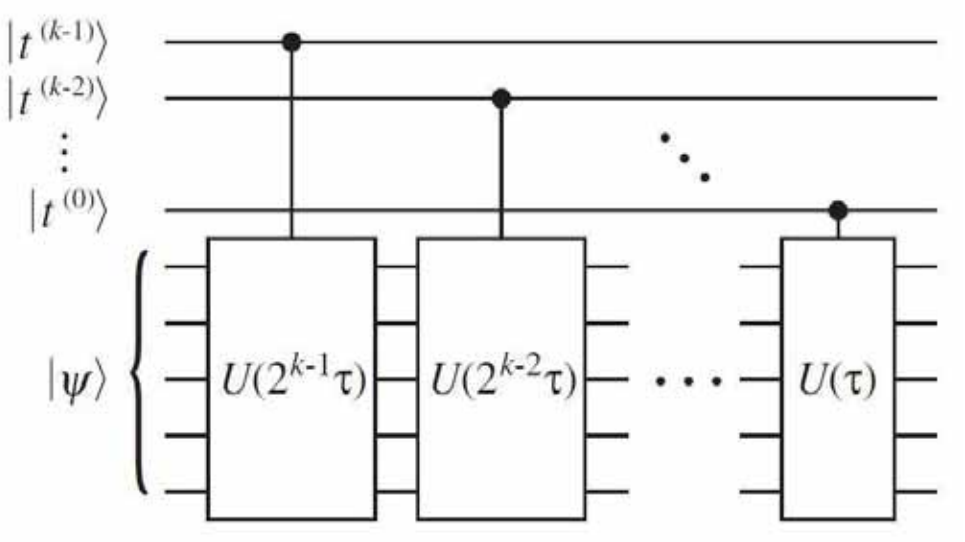


Fig.3.18 El tiempo de evolución del Hamiltoniano es producido al iterar el sistema propagador sobre muchos pasos de tiempo. Después de la evolución, una transformada de Fourier cuántica de las transformadas del vector de tiempo del sistema en la autobase de energía, permite una lectura del autovalor de energía.

3.6.4 Computación cuántica a gran escala

El algoritmo de factorización y la simulación cuántica representan interesantes aplicaciones de la computación cuántica de gran escala; para cada uno de los recursos computacionales requeridos de una arquitectura multicapa basada en PCECO listada en la Tabla 3.7. Los algoritmos son comparables en costos de recursos totales, como se refleja en el hecho de que estos dos problemas ejemplos requieren similar grado de corrección de error (de ahí un producto KQ muy similar). El algoritmo de simulación es más compacto que el de Shor, requiriendo, en particular, menor número de cubits lógicos para la destilación, lo que refleja el hecho de que este algoritmo realiza pocas operaciones aritméticas en paralelo. No obstante, el algoritmo de Shor tiene un tiempo de ejecución menor en este análisis. Ambos algoritmos pueden ser acelerados mediante el paralelismo si la computadora cuántica tiene más cubits lógicos para trabajar [118,122].

Tabla.3.7 Resumen de los recursos computacionales en una arquitectura multicapa basada en la plataforma PCECO, para el algoritmo de factorización de Shor un número de 1024-bit (la misma implementación que en [60]) y la simulación de estado basal de la molécula de alanina (C3H7NO2) usando primero una representación cuantizada.

Recurso computacional	Algoritmo de Shor (1024-bits)	Simulación molecular (alanina)
Capa 5: Cubits de aplicación.	6144	6650
Profundidad del circuito (Toffoli)	1.68×10^{8}	1.27×10^{9}
Capa 4: Cubits de destilación lógica.	66564	15860
Ciclos lógicos de reloj.	5.21×10^{9}	3.94×10^{10}
Capa 3: Distancia del código.	31	31
Error por ciclo de red.	2.58×10^{-20}	2.58×10^{-20}
Capa 2: Cubits virtuales.	4.54×10^{8}	1.40×10^{8}
Error por compuerta virtual.	1.00×10^{-3}	1.00×10^{-3}
Capa 1: Puntos cuánticos.	4.54×10^{8}	1.40×10^{8}
Area sobre el chip.	$4.54\ cm^2$	$1.40\ cm^2$
Tiempo de ejecución estimado.	1.81 días	13.7 días

3.7 Consideraciones temporales

El tiempo exacto, así como la secuencia de las operaciones son cruciales para hacer una arquitectura eficiente. En este marco, se presenta aquí, una capa superior en la arquitectura que depende del proceso en la capa inferior, de manera tal que el tiempo de la compuerta lógica está dictado por las operaciones CEC, y así sucesivamente. Este sistema de dependencia de los tiempos de las operaciones se representa por PCECO en la Fig.3.19. El eje horizontal está en una escala logarítmica en el tiempo para ejecutar una operación en una capa particular, mientras que las flechas indican una dependencia fundamental de una operación sobre otras operaciones en las capas inferiores. Examinando la Fig.3.19, vemos que las escalas de tiempo aumentan a medida que uno se va a capas más altas dado que una capa más alta debe frecuentemente debe emitir varios comandos a las capas inferiores. Usando PCECO como un ejemplo, la capa Virtual debe construir una compuerta virtual de 1-cubit de una secuencia de rotaciones de estado-espín.

Tabla.3.8 Tiempos de ciclo de reloj para las Capas 1 a 4 en nuestro análisis de PCECO. El tiempo del ciclo en cada capa es determinada por una operación fundamental de control. Muchas operaciones poseen alguna flexibilidad la cual debería permitir una compensación en el tiempo de ejecución y el tamaño del sistema, por lo que mejores métodos pueden ser descubiertos.

Capa	Ciclo del reloj	Operación limitante
4: Lógica	30 μs	CNOT lógico
3: CEC	256 ns	Refresco de red (circuito síndrome)
2:Virtual	32 ns	Compuerta virtual de 1 cunit
1: Física	8 ns	Frecuencia de repetición del pulso de láser

Este proceso incluye la duración de los pulsos del láser y los retardos entre pulsos, que se suman juntos para la duración total de la compuerta virtual.

Un punto crucial se muestra en la Fig.3.19 consistente en que el tiempo para implementar una compuerta cuántica lógica puede ser varios órdenes de magnitud mayor que la duración de cada proceso físico individual, como por ejemplo un pulso de láser.

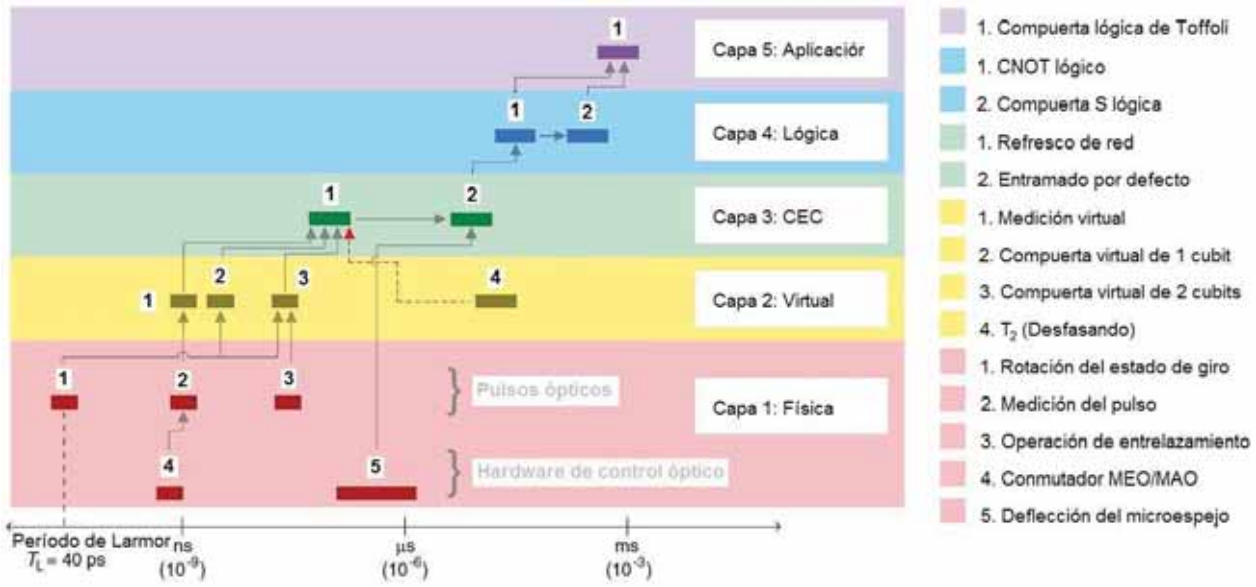


Fig.3.19 Escalas de tiempos relativos para operaciones críticas en PCECO en cada capa. Cada barra indica la escala de tiempo aproximada de una operación, y el ancho indica que algunos tiempos operación pueden variar con mejoras en la tecnología. Las flechas grises indican la dependencia de las operaciones de más alto orden sobre las capas más bajas. La flecha roja significa que el refresco de la red de código de superficie debe ser 2–3 órdenes de magnitud más rápido que el tiempo dasfasado con objeto de corregir los errores para funcionar. La capa de aplicación es representada con una compuerta de Toffoli, el cual es un bloque construido común de los algoritmos cuánticos. Las rutinas de algoritmos completos pueden variar significativamente, dependiendo de ambos; las capacidades de la computadora cuántica y la forma específica en que cada algoritmo es compilado, como es el caso de los cálculos extensos los cuales son realizados en paralelo.

Este aumento en el tiempo de operación es una consideración importante para los diseños de computadora cuántica que se basan en procesos físicos comparativamente más lentos. Al mismo tiempo, una computadora cuántica con un subconjunto de mecanismos de control muy rápidos está limitada por los más bajos procesos de compuerta esenciales, como PCECO puede solo operar tan rápido como la compuerta de entrelazamiento de 2-cubits en la Capa 1 lo permita. Para computación cuántica de gran escala, la velocidad de la lógica en las operaciones de corrección de errores constituye la figura de mérito crucial.

La Fig.3.19 pone de relieve también el hecho de que diferentes operaciones de control en la computadora ocurren a escalas de tiempo sustancialmente diferentes; lograr la sincronización de estos procesos es una función importante para una arquitectura de computadora cuántica. Para facilitar este proceso, cada capa en la arquitectura tiene una “frecuencia de reloj” interna, la cual es una característica de la escala de tiempo de las operaciones en esa capa.

Estos tiempos del ciclo del reloj para cada capa en PCECO se listan en la Tabla 3.8, junto con las operaciones que las definen. Incluso dentro de la misma capa, algunos procesos pueden tener diferentes períodos de tiempo de ejecución, por lo que establecer un ciclo de reloj sincroniza estas operaciones. En consecuencia, como una capa se basa en operaciones en una capa inferior, las dos capas están naturalmente sincronizadas. La sincronización por sí sola no es suficiente para hacer funcionar una computadora cuántica. Consideremos nuevamente el ciclo de control representado en la Fig.3.2. La extracción y el pro-cesamiento del síndrome del error se deben ejecutar en escalas de tiempo del mismo orden que la duración de una compuerta lógica o los errores se acumularán más rápido que lo que son detectados. Esta función se realiza via circuitería clásica,

no obstante, el esfuerzo de computación requerida puede no ser trivial. Las operaciones cuánticas rápidas pueden ser una carga, cuando la corrección de errores requiera cálculos (clásicos) complejos, como es el caso del código de superficie. Devitt *et al.* [54] y Fowler *et al.* [113,140] examinaron éste problema, encontrando que los requisitos de procesa-miento para la corrección de errores del código de superficie no son triviales; realizar estos cálculos ‘‘en vivo’’ cuando los resultados pueden ser necesarios en e.g., 10 μs podría ser uno de los más importantes problemas para la Ingeniería de una computadora cuántica. Sin embargo, los recientes avances es esta área sugieren que alguna combinación de software de algoritmo mejorado y hardware a medida pueden lograr el rendimiento necesario [140].

3.8 Arquitectura de computadora cuántica tolerante a fallos

Sobre la base de la teoría de la computación cuántica tolerante a fallos, se definen las bases para una arquitectura general que pueden minimizar dinámicamente la sobrecarga en la corrección de errores. En contraste con el modelo circuital usado en gran parte de la literatura de computación cuántica, la arquitectura propuesta aquí puede soportar eficientemente diferentes algoritmos y tamaño de datos. Los principales mecanismos que permitan a esta generalización son las rutas de datos confiables y la memoria cuántica eficiente [141]. En varios aspectos, computación cuántica es similar a computación clásica. Por ejemplo, los algoritmos cuánticos que tienen un flujo de control bien definido que manipula los elementos de datos individuales a lo largo de la ejecución.

Las restricciones físicas sobre las tecnologías cuánticas también se parecen al dominio clásico. A pesar de que dos cubits pueden interactuar a distancia, el más fuerte y el menos propenso, la interacción se da siempre entre vecinos cercanos. Por otra parte, la interacción controlada requiere circuitería de soporte clásica, la cual debe ser apropiadamente ruteada a lo largo de todo el dispositivo. Aunque esta arquitectura de computador cuántico es similar a la arquitectura clásica, ciertos aspectos de la computación son únicos para el dominio cuántico. Como se muestra en la Fig.3.20, la arquitectura completa tiene tres componentes principales: la Unidad Aritmético-Lógica (UAL) cuántica, la memoria cuántica, y el organizador dinámico. En adición, la arquitectura usa una técnica novedosa de conectividad cuántica que explota la teleportación cuántica [142].

3.8.1 UAL cuántica

El núcleo de esta arquitectura es la UAL cuántica, la cual realiza operaciones cuánticas para cálculo y corrección de errores. Para llevar a cabo eficazmente cualquiera de las compuertas cuánticas especificadas sobre los datos cuánticos, la UAL aplica una secuencia de transformadas cuánticas básicas bajo control clásico [71]. Las transformadas incluyen

- la de Hadamard (una raíz-2, la transformada de Fourier de 1-cubit),
- la identidad (I, una NOP cuántica),
- la vuelta de bit (X, una NOT cuántica),
- la vuelta de fase (Z, la cual cambia el signo de las amplitudes),
- la vuelta de bit y fase (Y),
- la rotación de $\pi/4$ (S),
- la rotación de $\pi/8$ (T), y
- la NOT controlada (CNOT).

Estas compuertas forman unos de los conjuntos universales más pequeños posibles para computación cuántica. La tecnología cuántica física subyacente puede implementar estas compuertas eficientemente sobre datos codificados. Todas excepto CNOT operan sobre un cubit único solamente; la compuerta CNOT opera sobre dos cubits. Para llevar a cabo una tarea de corrección de error de alto nivel, la UAL aplica una secuencia de operaciones elementales. Debido a que esta tarea es necesaria para la computación cuántica tolerante a fallas, la UAL la realiza sobre datos codificados luego de la mayoría de las operaciones lógicas. Este procedimiento consume estados ancilla, los cuales ayudan en el cálculo de verificación de paridad. El hardware especializado proporciona estados estándar elementales que la UAL usa para manufacturar la ancilla requerida.

3.8.2 Memoria cuántica

La generalidad de la arquitectura se basa en una memoria cuántica eficiente. La clave es construir un banco de memorias cuánticas que son más fiables que los dispositivos de computación cuántica. También podemos usar unidades de "refresco" especializadas que son mucho menos complejas que nuestra UAL general. El almacenamiento de los cubits no sometidos a cálculo es muy similar al almacenamiento de las Memorias de Acceso Aleatorias Dinámicas (MAAD) convencionales.

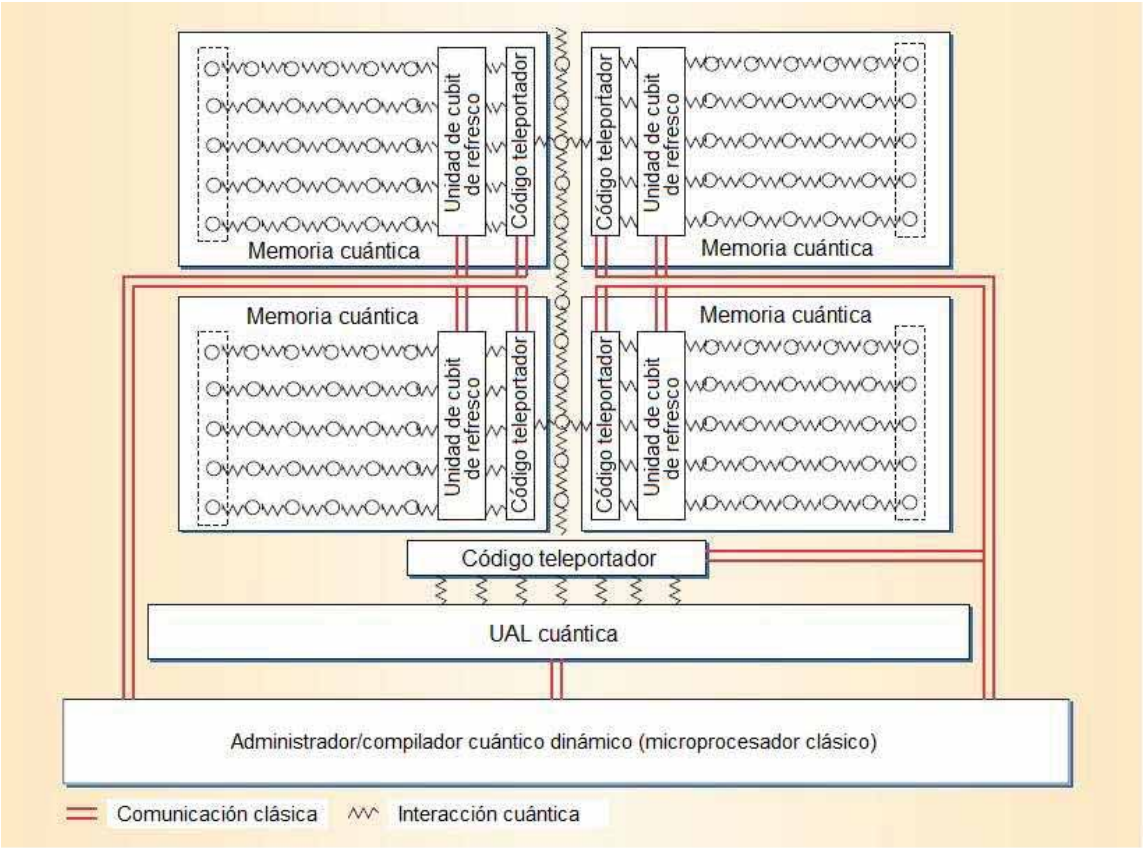


Fig.3.20 Arquitectura de computadora cuántica tolerante a fallas. La UAL cuántica realiza todas las operaciones cuánticas, los bancos de memorias cuánticas es compatible con la conversión de código eficiente, la teleportación transmite estados cuánticos sin enviar datos cuánticos, y el organizador dinámico controla todos los procesos.

Así como los capacitores individuales usados para MAAD fugan en el sustrato que lo rodea con el paso del tiempo, tal es el caso de las parejas de cubits en el entorno que los rodea y la decoherencia con el tiempo. Esto requiere un refresco periódico de los cubits lógicos individuales.

Como se muestra en la Fig.3.20, cada cubit del banco de memoria tiene una unidad de refresco dedicada que realiza periódicamente la detección de error y la recuperación sobre los cubits lógicos. Desde un punto de vista tecnológico, los subsistemas libres de decoherencia [84] los cuales proveen naturalmente las tasas de decoherencia más bajas para cubits estáticos, se podrían implementar tales como memorias cuánticas.

Esta arquitectura usa múltiples bancos de memoria cuántica. Esto no es para mejorar los tiempos de acceso del cubit lógico. De hecho, la tasa de error subyacente del mecanismo de almacenamiento físico de los cubits, la complejidad del algoritmo y el tamaño de los datos de entrada, así como el tiempo de operación de la UAL cuántica, el paralelismo, y el código de corrección de error que almacenan los cubits lógicos limita el tamaño del banco. Por ejemplo, si corremos el algoritmo de Shor sobre un número de 1,024-bit usando tecnología de memoria con una tasa de error de $p = 10^{-9}$, se estima que se debería usar 28,000 cubits físicos para representar alrededor de 1,000 cubits físicos usando dos niveles de recursión en un código de corrección de errores de 5-cubits. Por otro lado, si la tasa de error aumenta hasta $p = 10^{-6}$, la corrección del error debería requerir cuatro niveles de recursión para refrescar un banco de tamaño de solo 1.000 cubits físicos que debería almacenar solo dos cubits lógicos.

3.8.3 Alambres cuánticos

Mover información en una computadora cuántica es un reto. Las operaciones cuánticas deben ser reversibles, y no se pueden clonar cubits perfectamente—o sea, no podemos copiar su valor. No podemos sencillamente poner un cubit sobre un alambre y esperar a que se transmita el estado del cubit en consecuencia. En cambio, la arquitectura va a utilizar un concepto puramente cuántico para implementar alambres cuánticos: la teleportación [142]. Este procedimiento, el cual ha sido demostrado experimentalmente [143], transmite un estado cuántico entre dos puntos sin enviar realmente lo datos cuánticos. En su lugar, con la ayuda de un cierto estado estándar previamente compartido, la teleportación envía dos bits clásicos de datos de cada cubit.

La teleportación es superior a otros medios de entrega de estados cuánticos. Recordemos que una tecnología de estado sólido implementa cubits con átomos implantados en silicio [144,145]. Los cubits físicos no se pueden mover, pero podemos aplicar una operación de intercambio (swap) para pares progresivos de átomos para mover los valores de los cubits a lo largo de la línea de átomos. Mientras que podríamos usar una serie de compuertas cuánticas de intercambio para implementar alambres cuánticos, cada compuerta de intercambio se compone de tres compuertas CNOT, lo cual introduce errores en los cubits físicos—errores que generan sobrecarga adicional en los procedimientos de corrección. La teleportación en cambio usa compuertas de intercambio cuánticas que no son correctores de errores para distribuir cubits en un estado *gato* para la fuente y el destino del alambre. Un estado gato (nombrado en base al gato de Schrödinger) es un vector cubit con probabilidades igualmente distribuidas entre todos los bits puestos a 1 y todos los bits puestos a 0. Los cubits en un estado gato están entrelazados, y medir uno de los cubits determina únicamente el estado de todos los cubits en el vector de cubits. La teleportación usa un estado gato de dos-cubits. Este estado gato puede ser verificado por errores fácilmente e independientemente del cubit físico que está siendo transmitido. Si los errores tienen abrumado al estado gato, este puede ser desechado con poco daño para el proceso de transmisión. Una vez que existe

un estado gato correcto en ambos extremos, los cubits del estado gato teleportan al cubit físico a través de la distancia requerida.

3.8.4 Diseño multicapa

Para hacer realidad la ciencia con un computador cuántico, se deben haber coordinado avances en varios frentes [146]. La Fig.3.21 muestra una pila conceptual de subcampos en los que todos deben contribuir a una máquina útil completa. Las máquinas serán construidas para correr algoritmos específicos que existen en ciertas clases computacionales inaccesibles para las computadoras clásicas. Sin estos algoritmos, no habrá ningún incentivo económico para construir y desplegar máquinas. Sin corrección de error (ambos, teoría y práctica), las máquinas no pueden correr para cualquier espacio de tiempo útil. En el "fondo" de la pila se encuentra el cubit de almacenamiento, la compuerta y las tecnologías de transporte/interconexión, las cuales suministran campos de prueba útiles para la teoría cuántica en sí, sino que son de interés aquí principalmente como base para la construcción de computadoras cuánticas a gran escala. Ambas, el tope y el fondo de esta pila están muy ocupados con brillantes investigadores dedicados.

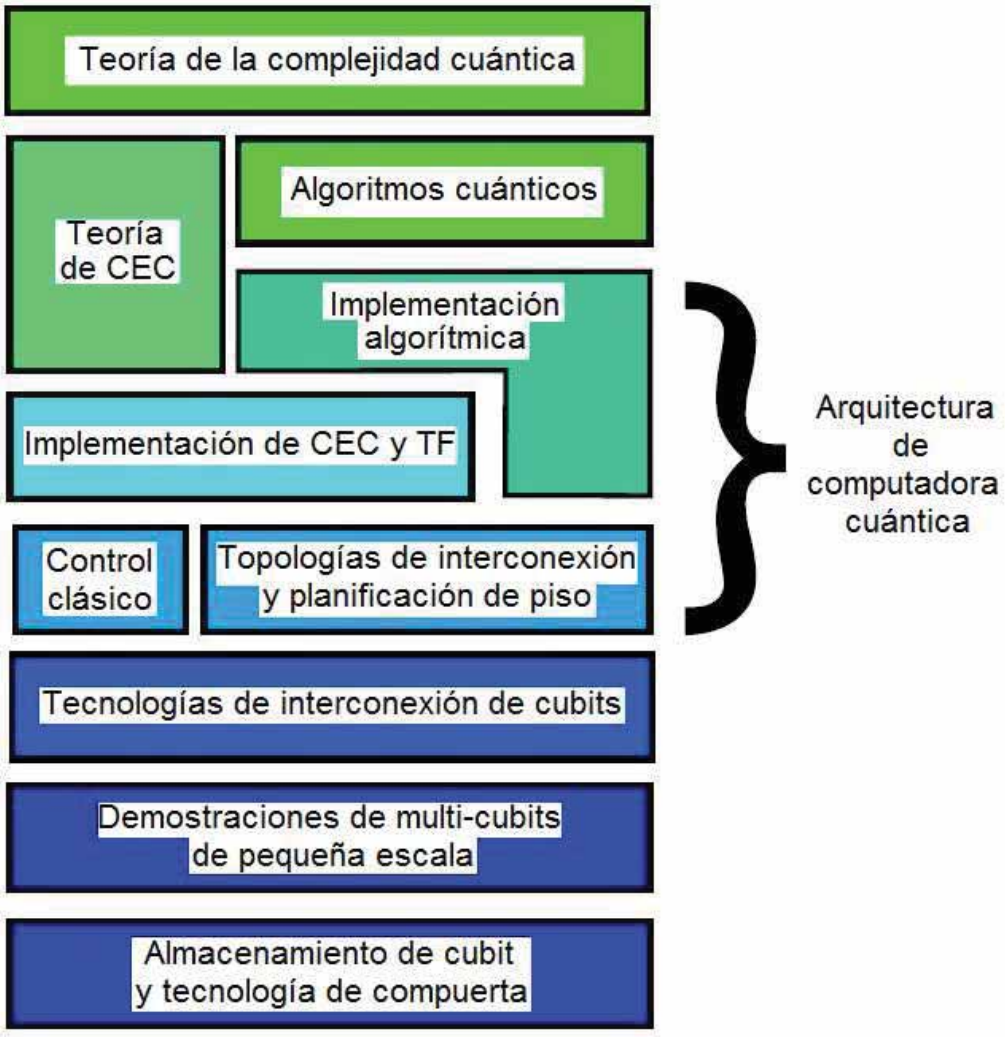


Fig.3.21 Arquitectura de computadora cuántica entre algunos subcampos de la computación cuántica. El CEC es la corrección de error cuántica; TF es la tolerancia a fallas. De [147].

La capa intermedia de esta figura, constituyendo el reino de la arquitectura de computadora cuántica, están menos densamente pobladas. Diseñar una arquitectura es el proceso de elección de las métricas para el éxito, definiendo el rendimiento deseado y los costos para los componentes de modo que puedan ser equilibrados, según se desee. La funcionalidad está dividida en subsistemas, y las interfaces entre los subsistemas también son definidas, junto con promesas correspondientes y supuestos sobre el comportamiento o las condiciones de mantenimiento.

Este puente es crítico, y un área con mucho lugar para la creatividad, además de trabajo riguroso. Un microprocesador es mucho más que simplemente una colección de transistores, y un sistema completo es mucho más que simplemente un microprocesador. Del mismo modo, las computadoras cuánticas serán mucho más que simplemente una colección uniforme de cubits.

Una arquitectura de computadora cuántica tiene mucho más que aprender de una arquitectura de computadora clásica, pero con algunas importantes salvedades. La primera de ellas consiste en que la naturaleza delicada de la información cuántica demanda que los elementos de memoria sean muy activos; en la mayoría de las concepciones, una computadora cuántica no realizará una fuerte distinción entre elementos de memoria y elementos computacionales. Segundo, son inexistentes largos cables o conexiones de larga distancia dentro de una computadora cuántica, requiriendo del autómata celular o vecino más cercano la transferencia de datos, la que siendo en el mejor de los casos de mala calidad, requiriere mucho más esfuerzo para transferir aunque más no sea un solo cubit de un lugar a otro usando purificación y teleportación. Entonces, los *principios* pueden ser aplicados, aunque las *respuestas* que llegaron probablemente diferirán sustancialmente de la arquitectura clásica de von Neumann.

Los recientes progresos en los experimentos, CEC, algoritmos, y arquitecturas han sido hechos sustancialmente más fácil por la aparición en los últimos años de excelentes comentarios en cada tópico, excepto arquitectura [85,135,148-154].

3.9 Conclusiones del capítulo

En este capítulo presentamos un marco multicapa para una arquitectura de computadora cuántica. El marco multicapa tiene dos tiene dos puntos fuertes: es modular, y facilita la tolerancia a fallas. La naturaleza multicapa de la arquitectura aconseja la modularidad, pero la característica que define a las capas que hemos elegido es la encapsulación. Cada una de las capas tiene un único e importante propósito así como los paquetes de las operaciones vinculadas a cumplir con dicho propósito. Adicionalmente, cada capa juega el rol de administrador de recursos, dado que frecuentemente muchas operaciones en la capa más baja son combinadas en la capa más alta. Puesto que las tecnologías de la computación cuántica van a evolucionar con el tiempo, las capas pueden necesitar un reemplazo en el futuro, y la encapsulación hace de la integración de los nuevos procesos una tarea más sencilla.

La tolerancia a fallos es en la actualidad el mayor reto para los computadores cuánticos, y la organización en capas es elegida deliberadamente para servir a esta necesidad. Podría decirse que las Capas 1 y 5 definen a cualquier computadora cuántica, pero las capas intermedias se dedican exclusivamente a la creación de la tolerancia a fallas de una manera inteligente. La Capa 2 utiliza un control simple para mitigar los errores sistemáticos, por lo que esta capa está posicionada cerca de la capa Física donde las técnicas como el DD y los subespacios libres de decoherencia son más efectivos. La Capa 3 es la anfitriona del CEC, que es esencial para modelo de circuito de

computadora cuántica de gran escala para sobre cualquier hardware, tales como ejecutar el algoritmo de Shor sobre un número de 1024 bits. Hay una interacción significativa entre las Capas 2 y 3, debido a que la Capa 2 mejora la eficacia de la Capa 3. Finalmente, la Capa 4 llena los vacíos en el conjunto de compuertas suministrado por la Capa 3 para formar cualquier operación unitaria deseada a la exactitud arbitraria, proporcionando de este modo un sustrato completo para la computación cuántica universal en la Capa 5.

PCECO, una plataforma de hardware específico que fue introducida en este trabajo, demuestra el poder del concepto de la arquitectura multicapa, pero también pone de relieve una serie de prometedoras tecnologías para la computación cuántica, que son particularmente notables por las escalas de tiempo rápido de las operaciones cuánticas, el alto grado de integración posible con la fabricación de estado sólido, y la adopción de varias tecnologías maduras de otros campos de la Ingeniería. Los tiempos de ejecución para las operaciones cuánticas fundamentales son discutidos en la Sec.3.2.7, pero la importancia de estos procesos rápidos se aprecia claramente en la Fig.3.15, donde la sobrecarga resultante de las compuertas virtuales en la Capa 2, CEC en la Capa 3, y las construcciones de compuertas en la Capa 4 incrementa el tiempo para implementar compuertas cuánticas de nanosegundos en la capa Física a milisegundos en la capa de Aplicación, o 6 órdenes de magnitud. En este contexto, una computadora cuántica necesita operaciones físicas muy rápidas.

Uno de los principales objetivos es entender mejor los recursos necesarios para construir una computadora cuántica que pueda resolver un problema intratable para las computadoras clásicas. Las figuras de mérito comunes para la evaluación de la tecnología de la computación cuántica son la fidelidad de la compuerta, el tiempo de operación, y el tiempo de coherencia del cubit. Esta investigación va más allá de mostrar cómo la conectividad y el rendimiento de control clásico son también cruciales. Diseñar una computadora cuántica requiere ver al sistema como un todo, de tal manera que las compensaciones y la compatibilidad entre componentes elegidos deben ser abordadas. Una foto holística es igualmente importante para comparar diferentes tecnologías de computación cuántica, tales como las trampas de iones o los circuitos superconductores. Este trabajo ilustra acerca de cómo aproximar el desafío completo de diseñar una computadora cuántica, para que se puedan adaptar estas técnicas al desarrollo de arquitecturas de otras tecnologías de computación cuántica que no hemos considerado aquí. Usando este diseño, diferentes sistemas propuestos pueden ser comparados en un marco común, que proporcione a los ingenieros cuánticos que aspiran a un lenguaje común para la determinación de la mejor tecnología de computación cuántica para una aplicación deseada.

Capítulo 4: Simulación de Algoritmos Cuánticos sobre GPGPU

Introducción

Ni siquiera dos décadas apenas han transcurrido desde que la computación cuántica se ha convertido en una disciplina prometedora [155]. En comparación con la computación convencional (clásica), los computadores cuánticos son los denominados dispositivos que procesan la información sobre la base de las leyes de la física cuántica y que podrían resolver algunos de los problemas de complejidad computacional no-polinómica en un tiempo mucho menor [71], es decir, polinómico. Aunque se trata de un enfoque muy esperanzador, en la actualidad todavía hay que enfrentarse a ciertas limitaciones difíciles. Por un lado, la tecnología actual sólo permite construir computadores cuánticos de dimensiones muy reducidas [156], y por otro lado se conoce sólo un número muy pequeño de algoritmos que pueden disminuir significativamente su complejidad computacional cuando se ejecutan sobre un computador cuántico [157-159].

El interés de simular el comportamiento de los computadores cuánticos es motivado por varios factores, que van desde las puramente teóricos, académicos y educativos, a la evaluación comparativa entre plataformas monoprocesador (convencionales) vs multiprocesador (por ejemplo, GPGPU: General-Purpose Computing on Graphics Processing Units). Además, su simulación constituye una herramienta muy útil en el desarrollo y prueba de nuevos algoritmos cuánticos (al menos para conjuntos de datos de pequeñas dimensiones).

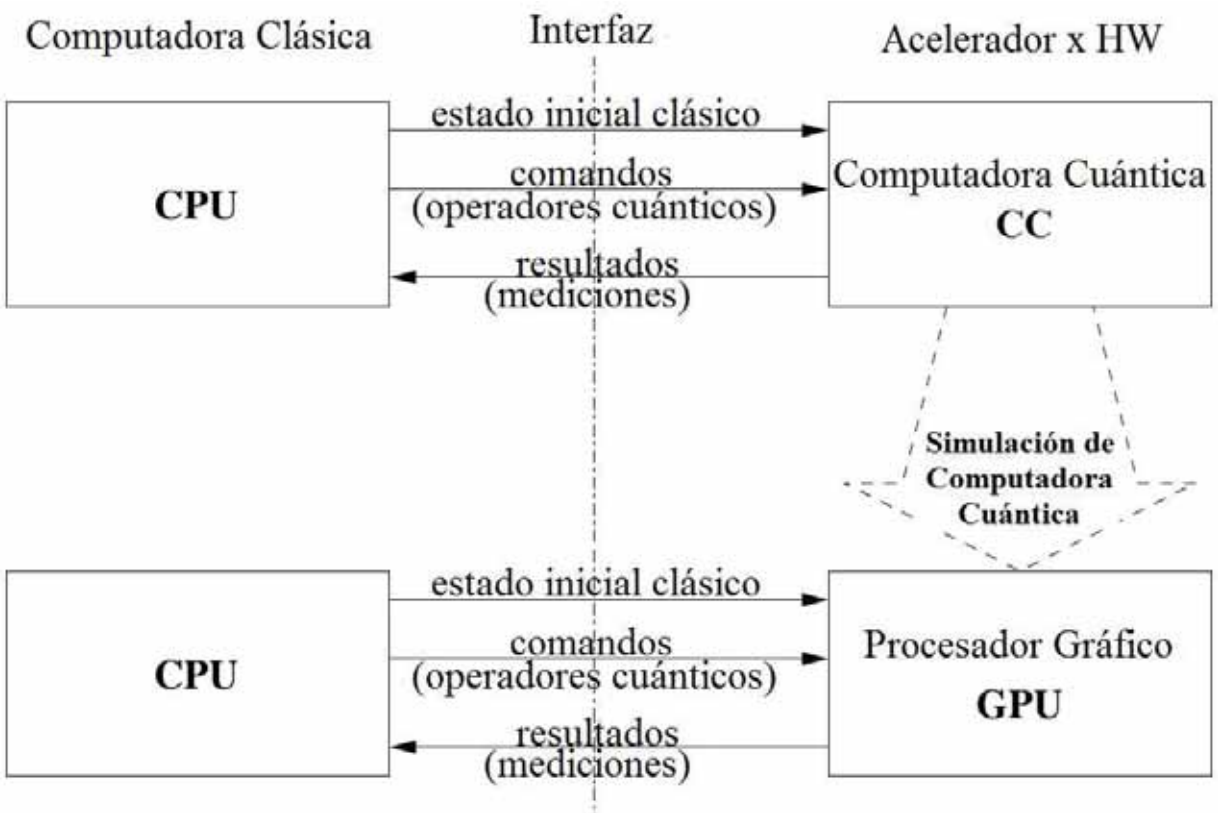


Fig.4.1: Modelo de computadora cuántica como un acelerador de hardware y su simulación.

La mayor parte del poder de los computadores cuánticos es debido al paralelismo cuántico que permite realizar operaciones simultáneas en un conjunto exponencial de datos superpuestos. Esta es la razón por la cual la simulación de este tipo de sistemas requiere un esfuerzo exponencial. El paralelismo es una herramienta adecuada para mitigar dichos requisitos computacionales y permitirá la emulación de los computadores cuánticos de mayor dimensión en un plazo razonable. El modelo que se muestra en la Fig.4.1 es el seguido en el trabajo de Ömer [160]. El computador cuántico actúa como un hardware de aceleración del procesador clásico, que envía las órdenes necesarias para resolver un problema concreto. De acuerdo con las leyes que lo rigen, no es posible conocer el estado exacto de este ordenador cuántico. Por lo tanto, la salida del algoritmo cuántico se puede obtener mediante un proceso de medición con cierta probabilidad. Este enfoque es seguido en nuestro simulador paralelo, el cual considera un computador cuántico ideal. La interfaz adoptada entre el computador clásico (anfitrión) y la plataforma de simulación es tomada de libquantum [161], uno de los programas de simulación más populares.

En este trabajo se muestra que los computadores cuánticos pueden ser eficientemente simulados sobre arquitecturas modernas basadas en las unidades de procesamiento gráfico, o Graphics Processing Units (GPU). Con este fin, se proponen varios métodos para la simulación paralela de los operadores básicos de un computador cuántico ideal enmarcado dentro del Compute Unified Device Architecture (CUDA) de NVIDIA [162].

4.1 Trabajo relacionado

El interés de los científicos, ingenieros y académicos en la computación cuántica ha dado lugar al desarrollo de varios simuladores de computación cuántica, cuya amplia lista se puede encontrar en [163]. Estos cubren una gran cantidad de lenguajes de programación y la mayoría de ellos están orientados a fines educativos o de demostración. Entre ellos podemos destacar libquantum [161] una biblioteca basada en C que permite escribir “programas cuánticos" que hacen uso de llamadas a funciones para compuertas cuánticas elementales. El libquantum es una biblioteca popular distribuida en formato fuente y cuyos componentes binarios están disponibles en algunas distribuciones de Linux. Además, forma parte del banco de prueba del SPEC CPU2006 [164].

Los esfuerzos para hacer frente a los altos requisitos de memoria y de cálculo se pueden encontrar en la literatura. Como soluciones de hardware, varios sistemas basados en Field Programmable Gate Array **(**FPGA) se han propuesto como aceleradores de hardware para los algoritmos de computación cuántica más importantes [165-168]. En cuanto al software, plataformas paralelas masivas proporcionan altos recursos informáticos con el fin de realizar simulaciones más rápidas para un elevado número de cubits. El principal desafío por superar es cómo distribuir/comunicar eficazmente los datos entre procesadores.

En el trabajo de Obenland y Despain [169] se llevaron a cabo simulaciones en paralelo sobre una CrayT3E y una IBM SP2 con una librería MPI de paso de mensajes en C, llegando hasta los 4.096 procesadores con un máximo de 28 cubits. En su trabajo Niwa *et. al.* [170], realizaron simulaciones paralelas ejecutadas sobre una SunEnterprise 4500 basada en UltraSPARCII, empleando la biblioteca de hilos de Solaris, alcanzando un máximo de 8 hilos con un tamaño máximo de 30 cubits. Simulaciones paralelas realizadas en [156] se llevaron a cabo en un clúster Beowulf de 16 nodos basado en procesadores Pentium 4 utilizando un modelo de paso de mensajes que permite realizar simulaciones de hasta 29 cubits. Simulaciones propuestos por Arnold [171] se ejecutaron sobre un clúster IBM P690 por medio de un modelo híbrido OpenMP

(memoria compartida)/ MPI (paso de mensajes), con un máximo de 1024 hilos logrando ejecuciones de hasta 37 cubits. Más recientemente, Raedt *et. al.* [172], llevó a cabo varias simulaciones donde se analizan y comparan las plataformas masivamente paralelas, como IBM Blue Gene/L, la IBM Regatta P690, la Hitachi SR11000, y la Cray X1E, entre otras. En este último trabajo, los programas fueron codificados en Fortran 90 con la biblioteca MPI, proporcionando ejecuciones con 4.096 hilos con tamaños de hasta 36 cubits.

4.2 Compute Unified Device Architecture (CUDA)

La Compute Unified Device Architecture (CUDA™) ha sido introducida por NVIDIA como una arquitectura de hardware y software tanto para la emisión y como así también para la gestión de cálculos en sus más recientes familias GPU, razón por la cual puede operar como un verdadero dispositivo de computación de datos paralelos genérico. Una extensión para el lenguaje de programación C se proporciona con el fin de desarrollar los códigos fuente.

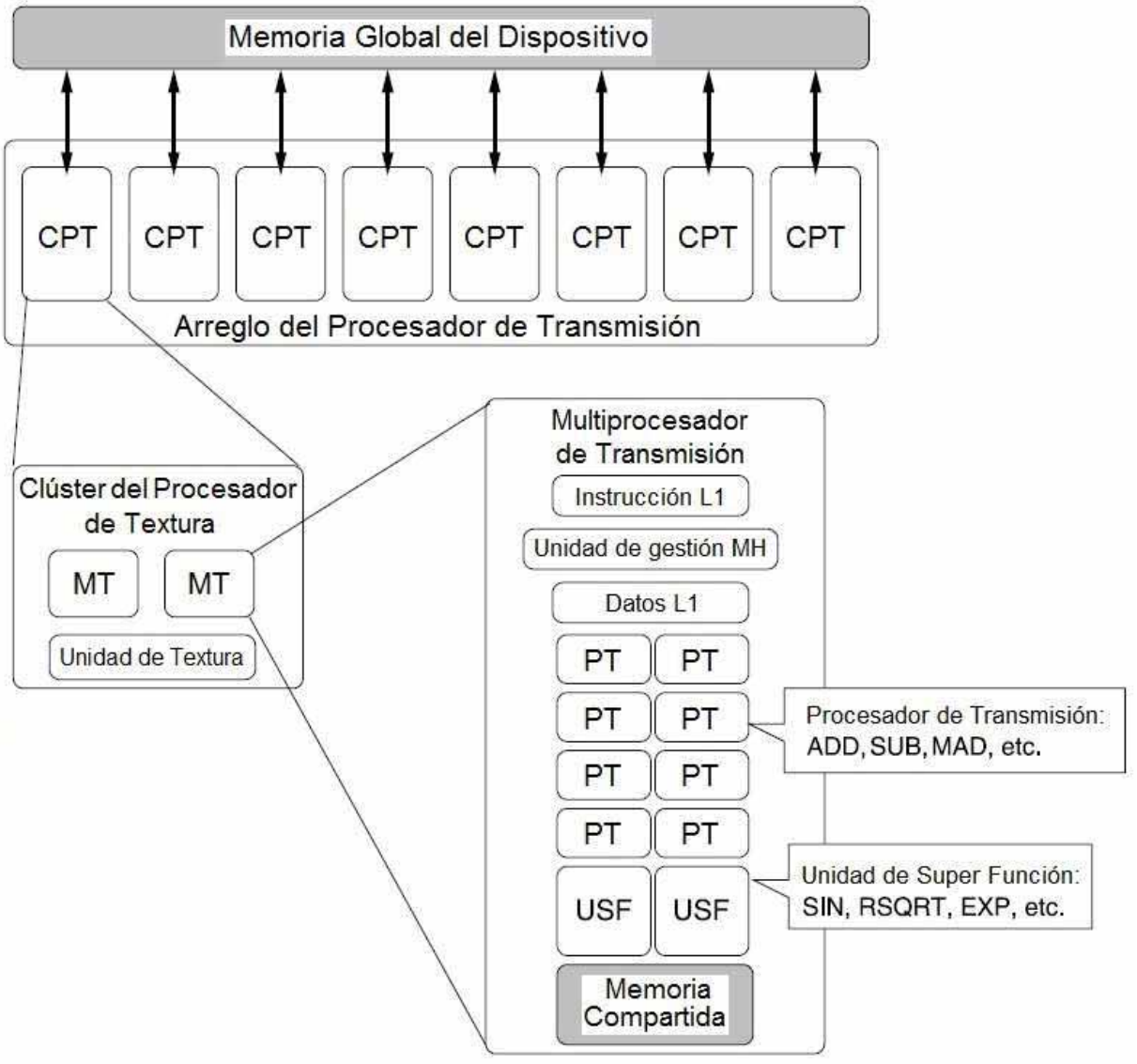


Fig.4.2: Organización de los procesadores y los espacios de memoria en CUDA.

Desde el punto de vista del hardware, el dispositivo GPU consta de un arreglo de multiprocesadores (MP) del tipo SIMT (Single Instruction Multiple Threads, en español: instrucciones únicas para varios hilos) cada uno de los cuales contiene varios procesadores de transmisión (PT), tal como se representa en la Fig.4.2 [162,173].

A este respecto, se encuentran disponible diferentes espacios de memoria. La memoria global del dispositivo es un espacio único accesible para todos los multiprocesadores, que actúa como la memoria del dispositivo principal, y es de una gran capacidad. Además, cada multiprocesador es dueño de una memoria privada en el chip, llamada memoria compartida o caché de datos en paralelo, la cual es de un tamaño más pequeño y menor latencia de acceso que la memoria global. Además, hay otros espacios de direccionamiento, omitidos en la figura, para propósitos más específicos, como son las memorias de textura y constante.

El modelo de ejecución de CUDA se basa en unas capas de abstracción jerárquica: cuadrículas, bloques, tramas e hilos como se muestra en la Fig.4.3.

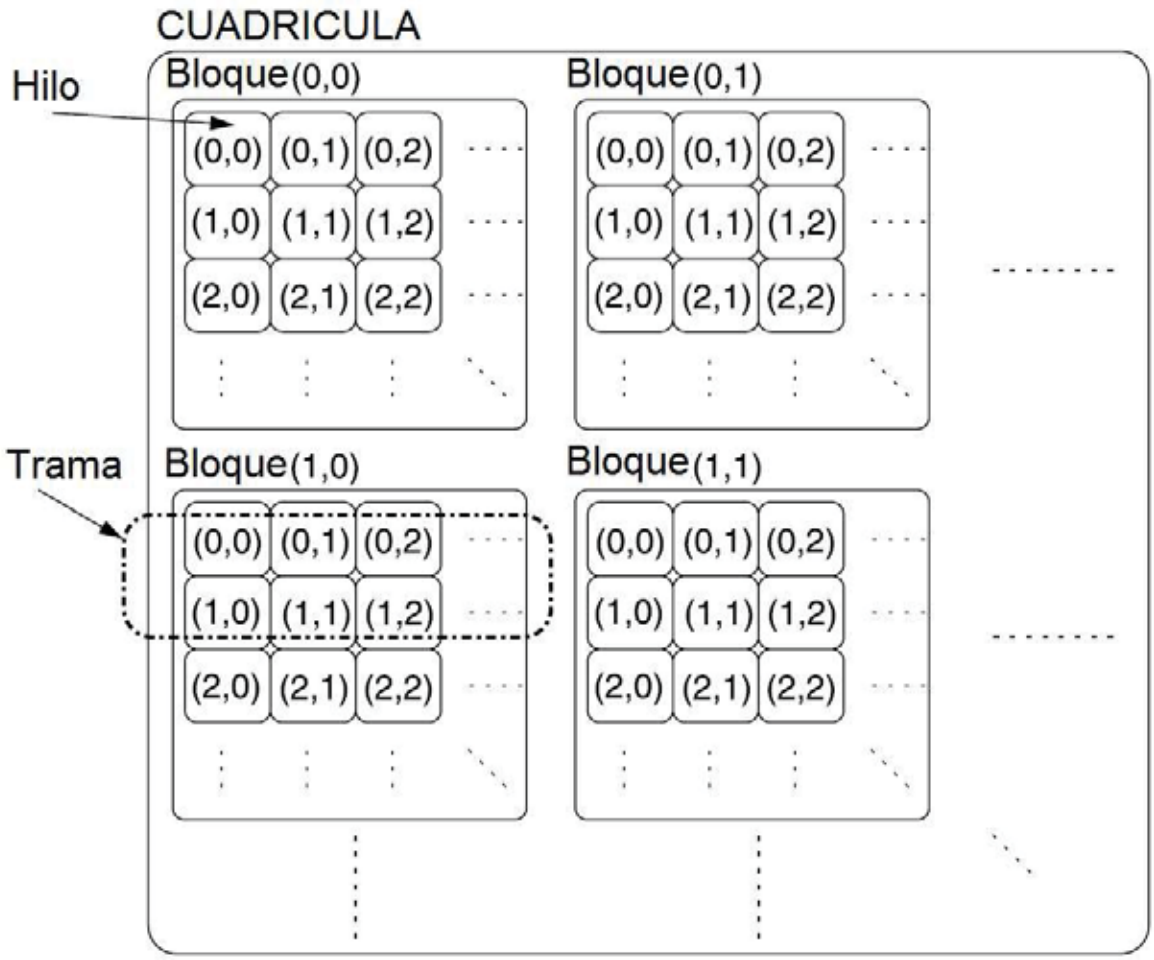


Fig.4.3: Esquema de ejecución SIMT en CUDA.

El hilo es la unidad de ejecución básica que es realmente mapeado sobre un procesador. Un bloque es un lote de hilos cooperando juntos sobre un multiprocesador y por lo tanto todos los

hilos en un bloque comparten la caché de datos paralelos. Una cuadrícula está compuesta por varios bloques, y dado que puede haber más bloques que procesadores, diferentes bloques de una cuadrícula están repartidos entre el conjunto de multiprocesadores. A su vez, una trama es un grupo de hilos ejecutándose de una manera SIMT, de manera tal que los hilos de un mismo bloque estarán repartidos en un multiprocesador trama por trama. En un determinado momento hilos de varios bloques pueden estar bajo una organización sobre el mismo multiprocesador. Estos son llamados bloques activos (hilos activos). Esta característica brinda una posibilidad para la latencia de la memoria oculta, en cuanto a las tramas de los hilos activos las cuales pueden ser despachadas mientras otras también activas están esperando por el acceso a la memoria global.

Dos clases de códigos son considerados: aquellos ejecutados por la CPU (del lado del anfitrión) y aquellos ejecutados por la GPU (del lado del dispositivo), llamados códigos del núcleo. La CPU es la responsable de transferir los datos entre las memorias del anfitrión (CPU) y el dispositivo (GPU) además de invocar el código del núcleo, estableciendo las dimensiones de la cuadrícula y el bloque. Tales núcleos están destinados a ser ejecutados de una manera SIMT lo largo de los procesadores. Un ejemplo sencillo se muestra en la Fig.4.4 donde en una matriz de punto flotante se inicializa en el dispositivo. Después de asignar el espacio de la matriz en la memoria global del dispositivo, el código del núcleo es invocado por el anfitrión. La función del núcleo es ejecutado por la GPU de una manera SIMT: cada hilo actualizará un elemento de la matriz global.

Lado del anfitrión (CPU)

```
void main(){
  int dimCuad=nBloques; /* Número de bloques */
  int dimBloque=N/nBloques;/* Hilos por bloque */
  float *A, *Ad;
  cudaMalloc(Ad, N);
  nucleo<<<dimCuad,dimBloque>>>(Ad);
  A=(float *)malloc(N*sizeof( float ));
  cudaMemcpy(A, Ad, N, cudaMemcpyDeviceToHost);
    ...
}
```

Lado del dispositivo (GPU)

```
__global__ void nucleo ( float *Ad){
 /* Hilo id en la cuadrícula completa */
    int id=bloqueIdx.x*bloqueDim.x+hiloIdx .x;

 /* La siguiente acción es ejecutada
    por hilos de una manera SIMT */

  Ad[id]=id ?0.0:1.0;

  }
```

Fig.4.4: Inicialización paralela del vector de estados para el valor |0> en el lado de la GPU.

El programador define el número de bloques, el número de hilos por bloque y su distribución (1D, 2D, 3D). Sin embargo, la dimensionalidad de las tramas es un parámetro de programación fijado por el sistema. El programador es responsable de la aplicación de las estrategias de codificación que resultarán en una buena planificación y el correcto equilibrio en la carga del trabajo. Además, existen algunas limitaciones, tales como el número máximo de hilos por bloque y el número máximo de bloques en cada dimensión. Los accesos a memoria y el esquema de sincronización son otros aspectos importantes a tener en cuenta. Las direcciones de las tramas emitidas por las instrucciones de acceso a memoria SIMT pueden agruparse obteniendo así un alto ancho de banda de memoria. Esto se conoce como condición de coalescencia que establece una velocidad de transferencia óptima cuando las posiciones de memoria consecutivas son accedidas por hilos consecutivos [162,174]. De lo contrario, el acceso será serializado y la latencia resultante será difícil de ocultar la ejecución de otras tramas del mismo bloque. La sincronización

global no es proporcionada del lado del dispositivo, sólo los hilos en un bloque pueden estar esperando a cada uno de los otros. Por consiguiente, el mecanismo de sincronización de bloques debe estar implementado de forma explícita por el anfitrión a través de invocaciones consecutivas del núcleo.

4.3 Simulación de datos en forma paralela

En esta sección se describirá el modelo computacional de la simulación en paralelo, basado en la arquitectura CUDA. Vamos a analizar las dependencias de los datos existentes con un enfoque paralelo de datos, ya que serán el principal límite a la explotación del paralelismo. En adelante, vamos a denotar el espacio de coeficientes, como $S=\{\alpha_0,\alpha_1,\ldots,\alpha_{N-1}\}$ (en realidad estos α_i representan literales, no los valores de los coeficientes. Así, el conjunto contendrá N elementos asociados a un vector de estados conocido $|\psi\rangle=(\alpha_0,\alpha_1,\ldots,\alpha_{N-1})$), siendo $N = 2^n$. Más precisamente, $S^{en}=\{\alpha_0^{en},\alpha_1^{en},\ldots,\alpha_{N-1}^{en}\}$ es el espacio de coeficiente en su estado inicial, antes de la aplicación de la transformación unitaria, y $S^{sal}=\{\alpha_0^{sal},\alpha_1^{sal},\ldots,\alpha_{N-1}^{sal}\}$ es el espacio de coeficientes después de la transformación.

Nuestro problema a modo de objetivo consiste en calcular el estado de salida para un registro ncubit, $|\psi^{sal}\rangle=\sum_{i=0}^{2^n-1}\alpha_i^{sal}|i\rangle$, a partir del estado inicial $|\psi^{en}\rangle=\sum_{i=0}^{2^n-1}\alpha_i^{en}|i\rangle$, dada una transformación U de tamaño $2^n \times 2^n$ $\left(|\psi^{sal}\rangle=U|\psi^{en}\rangle\right)$. Esto implica el cálculo de los valores para el conjunto de coeficientes de S^{sal}, utilizando como entrada el conjunto de coeficientes de S^{en}.

Observar que, en general, la aplicación de tal transformación tiene una complejidad computacional del orden $O(2^{2n})$. Un primer paso será establecer las dependencias de los datos entre los coeficientes de entrada y salida. Para una arquitectura de paso de mensajes, se determinarán las comunicaciones. En el modelo de paralelismo SIMT, esta dependencia afecta a los conflictos de acceso a memoria y la sincronización entre los diferentes hilos.

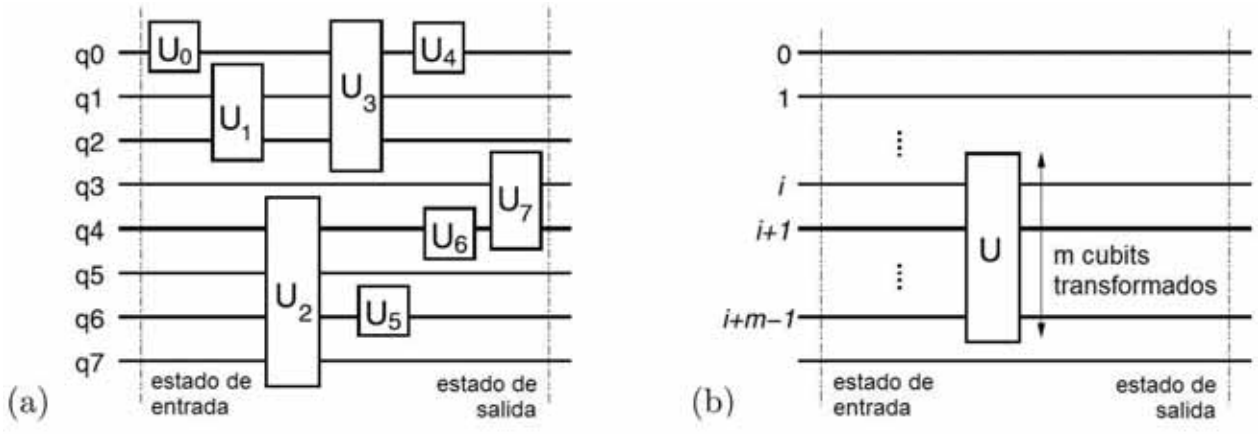


Fig.4.5: (a) Una computadora cuántica como una red de compuertas cuánticas;
(b) Una compuerta afectando al subconjunto de cubits.

Siendo conservador y como un caso trivial, se puede suponer que todos los coeficientes serán transformados y que cada coeficiente de salida dependerá de todos los coeficientes de entrada. Sin embargo, como se mencionó anteriormente, al expresar la transformación como una red de compuertas cuánticas [175,176], cada uno afectará a un subconjunto de cubits, Fig.4.5(a), con lo que las dependencias pueden ser manejadas a nivel de la compuerta.

Sin pérdida de generalidad, la aplicación de una compuerta sobre *m* cubits consecutivos partiendo desde el cubit *i* será considerada en función de analizar las dependencias de los datos, Fig.4.5(b) para un registro de *n* cubits $(i+m \le n)$. En este caso, la transformación resultante global será expresada como:

$$U_g = I^{\otimes n-m-i} \otimes U \otimes I^{\otimes i}, \tag{4.1}$$

donde la matriz asociada a los cubits *m* a ser transformada es:

$$U = \sum_{i=0}^{2^m-1} \sum_{j=0}^{2^m-1} u_{i,j} |i\rangle\langle j|,$$

de dimensión $2^m \times 2^m$.

Consideremos que la transformación será aplicada a un estado clásico inicial (un elemento de la base de espacio vectorial) cuya representación binaria es:

$$\psi^{en} = |b_{n-1}b_{n-2}\ldots b_1 b_0\rangle_2$$

con:

$$\begin{aligned}
\psi^{sal} &= U_g \otimes |b_{n-1}b_{n-2}\ldots b_{i+m}\rangle \\
&= \left(I^{\otimes n-m-i} \otimes U \otimes I^{\otimes i}\right) \otimes |b_{n-1}b_{n-2}\ldots b_0\rangle \\
&= |b_{n-1}b_{n-2}\ldots b_{i+m}\rangle \otimes U |b_{i+m-1}\ldots b_i\rangle \otimes |b_{i-1}b_{n-2}\ldots b_0\rangle \\
&= \begin{cases}
|b_{n-1}b_{n-2}\ldots b_{i+m}\rangle \otimes \left(\sum_{i=0}^{2^m-1} u_{i,0}|i\rangle\right) \otimes |b_{n-1}b_{n-2}\ldots b_{i+m}\rangle & si\ b_{i+m-1}\ldots b_i = 0 = 0\ldots00_{(2}, \\
|b_{n-1}b_{n-2}\ldots b_{i+m}\rangle \otimes \left(\sum_{i=0}^{2^m-1} u_{i,1}|i\rangle\right) \otimes |b_{n-1}b_{n-2}\ldots b_{i+m}\rangle & si\ b_{i+m-1}\ldots b_i = 1 = 0\ldots01_{(2}, \\
\vdots & \\
|b_{n-1}b_{n-2}\ldots b_{i+m}\rangle \otimes \left(\sum_{i=0}^{2^m-1} u_{i,2^m-1}|i\rangle\right) \otimes |b_{n-1}b_{n-2}\ldots b_{i+m}\rangle & si\ b_{i+m-1}\ldots b_i = 2^m-1 = 1\ldots11_{(2},
\end{cases}
\end{aligned} \tag{4.2}$$

En general, cada estado será una superposición lineal de estos estados, de manera tal que el coeficiente *k* del vector de estado final, con la expresión binaria $k = b_{n-1}b_{n-2}\ldots b_{0(2}$, puede ser escrito como una función del vector de estado inicial usando la Ec.(4.2):

$$\alpha_k^{sal} = \alpha^{sal}_{b_{n-1}b_{n-2}\ldots b_0}$$

$$= \begin{cases} \left(\sum_{j=0}^{2^m-1} u_{0,j}\, \alpha^{en}_{b_{n-1}b_{n-2}\ldots b_{i+m}|j|b_{i-1}b_{n-2}\ldots b_0} \right) & si\ b_{i+m-1}\ldots b_i = 0 = 0\ldots00_{(2}, \\ \left(\sum_{j=0}^{2^m-1} u_{1,j}\, \alpha^{en}_{b_{n-1}b_{n-2}\ldots b_{i+m}|j|b_{i-1}b_{n-2}\ldots b_0} \right) & si\ b_{i+m-1}\ldots b_i = 1 = 0\ldots01_{(2}, \\ \vdots & \\ \left(\sum_{j=0}^{2^m-1} u_{2^m-1,i}\, \alpha^{en}_{b_{n-1}b_{n-2}\ldots b_{i+m}|j|b_{i-1}b_{n-2}\ldots b_0} \right) & si\ b_{i+m-1}\ldots b_i = 2^m-1 = 1\ldots11_{(2}, \end{cases} \tag{4.3}$$

$$= \sum_{j=0}^{2^m-1} u_{\zeta,j}\, \alpha^{en}_{\left(k \wedge \left(2^{i+m}-2^i\right)\right)\oplus 2^j}$$

$$con\ \zeta = b_{i+m-1}\ldots b_i = \left\lfloor \frac{k}{2^i} \right\rfloor \bmod 2^m$$

La concatenación binaria de los números naturales está representada como |, y se aplica a ambos, es decir, dígitos binarios y números naturales, manteniendo un resultado de *n* bits. La conjunción lógica AND es $\wedge$, y la OR exclusiva o XOR es $\oplus$.

La Ec(4.3) permite calcular los coeficientes del estado cuántico final. Note que solo 2*m* coeficientes de entrada toman parte en el cálculo de cada coeficiente de salida, y además, que estos 2*m* coeficientes actúan como un grupo cerrado. Esto significa que cada uno de ellos solo toma parte en el cálculo de los coeficientes en el interior de ese grupo, siendo independiente del resto de los grupos.

Para un uso subsecuente, son introducidas formalmente varias definiciones. Consideremos un subconjunto del espacio de coeficientes $T \subset S$. Establecemos los valores de los coeficientes en *T* antes y después de la aplicación de una cierta transformación como T^{en}, T^{sal}, respectivamente. Para una transformación U_g, establecemos el conjunto de coeficientes de salida que son calculables usando solamente los coeficientes en *T* como $U_g(T)$.

Por ejemplo, consideremos $U_g = I^{n-2} \otimes U \otimes I$, una compuerta *U* de 1-cubit aplicada a el cubit 1 del registro. Si $T = \{\alpha_1, \alpha_3\}$, entonces denotaremos a $T^{en} = \left\{\alpha_1^{en}, \alpha_3^{en}\right\}$ y $T^{sal} = \left\{\alpha_1^{sal}, \alpha_3^{sal}\right\}$.

De la Ec.(4.3), el cálculo de los coeficientes α_1^{sal} y α_3^{sal} depende de los coeficientes α_1^{en} y α_3^{en}, entonces, $U_g\left(T^{en}\right) = \left\{\alpha_1^{sal}, \alpha_3^{sal}\right\}$. Actúa como un conjunto computacionalmente cerrado porque $T^{\text{out}} = U_g(T^{\text{in}})$. No obstante, si $T = \{\alpha_0, \alpha_4\}$ se sigue que $U_g(T^{\text{in}}) = \varnothing$, ya que no es posible calcular cualquier coeficiente de salida con sólo estos dos coeficientes. Entonces $T^{sal} = \left\{\alpha_0^{sal}, \alpha_4^{sal}\right\} \neq U_g\left(T^{en}\right)$. Las siguientes definiciones contemplan estas observaciones en un estilo más formal.

Definición 1. Sea U_g una transformación aplicada a un registro cuántico. Establecemos que un subconjunto de coeficientes $T \subset S$ es cerrado cuando $T^{\text{sal}} = U_g(T^{\text{in}})$.

Proclama 1. *La unión de dos subconjuntos cerrados para una cierta transformación resulta en un subconjunto cerrado.*

Definición 2. Un subconjunto cerrado de coeficientes en mínimo cuando, luego de remover uno sus coeficientes, no es posible calcular ningún coeficiente de salida para la transformación U_g usando los coeficientes que permanecen en el interior del subconjunto.

Proclama 2. *Para una cierta transformación U_g es siempre posible encontrar una partición del espacio de los coeficientes S en el interior de los subconjuntos cerrados de coeficientes.*

Proclama 3. *Si la transformación solo afecta m cubits de un total de n integrantes del registro, como en el caso de la Ec.*(4.1)*, existen $2^n/2^m$ diferentes subconjuntos cerrados mínimos con cardinalidad 2^m.*

Observar que hay particiones de *S* en conjuntos cerrados, no necesariamente mínimos, que pueden ser cerrados por más que una transformación afecte a diferentes cubits. Por ejemplo, para una compuerta de 1-cubit aplicada sobre el cubit 0, una partición sobre los subconjuntos cerrados mínimos es: $\{\{\alpha_0,\alpha_1\}\{\alpha_2,\alpha_3\}\{\alpha_4,\alpha_5\}\{\alpha_6,\alpha_7\}\}$, pero si la compuerta es aplicada al cubit 1, la partición se convierte en: $\{\{\alpha_0,\alpha_2\}\{\alpha_1,\alpha_3\}\{\alpha_4,\alpha_6\}\{\alpha_5,\alpha_7\}\}$. No obstante, podemos encontrar una partición en los conjuntos cerrados, aunque no mínima, que está cerrada tanto para la transformación 1-cubit aplicada al cubit 0, y la misma transformación aplicada al cubit 1: $\{\{\alpha_0,\alpha_1,\alpha_2,\alpha_3\}, \{\alpha_4,\alpha_5,\alpha_6,\alpha_7\}\}$.

Las siguientes dos características son de gran importancia dado que se refieren a la explotación de la localidad sobre la arquitectura objetivo.

Definición 3. Un conjunto cerrado $T \in S$ es c-coalescente si puede ser particionado en lotes disjuntos con al menos 2^c coeficientes en la forma: $\alpha_k, \alpha_{k+1}, \ldots, \alpha_{k+2^c-1}$, siendo *k* una potencia de 2.

Definición 4. Decimos que una partición de *S* es c-coalescente si todos los subconjutos que participan en la partición son c-coalescentes.

Proclama 4. *La unión de conjuntos cerrados coalescentes de orden "c", continúan siendo coalescentes de orden "c".*

Una característica notable de una partición en conjuntos cerrados, es que los coeficientes de Salida que pertenecen a diferentes subconjuntos cerrados se pueden calcular en paralelo.

A continuación, se estudia la forma de los conjuntos cerrados mínimos al cual pertenece un determinado coeficiente α_k, para diferentes configuraciones de interés.

4.3.1. Transformación diagonal unitaria

Es la forma de la matriz para la cual una transformación consiste en una matriz diagonal con valores $e^{j\phi_i}$. Observe que cada coeficiente del vector de estados es auto-transformado independientemente del resto. Como los conjuntos cerrados mínimos contienen sólo un elemento, es posible encontrar una partición mínima de *S* en solo uno de los elementos de los conjuntos cerrados. En este caso, el coeficiente α_k pertenecerá a un conjunto mínimo que se indica cómo

D_k, dado por $D_k = \{\alpha_k\}$. Observar que esta partición es válida para cualquier transformación diagonal, o su producto de Kronecker, ya que su resultado será también diagonal.

4.3.2. Compuerta de 1-cubit aplicada a un solo cubit

Consideremos la aplicación de la compuerta U de 1-cubit, sobre el cubit i de un registro n-cubit, resultando en una transformación global de la forma: $U_g = I^{\otimes(n-2-i)} \otimes U \otimes I^{\otimes i}$. De la Ec.(4.3) se puede inferir que el conjunto cerrado mínimo al cual pertenece el coeficiente α_k, está hecho de dos coeficientes cuyos subíndices difieren en el bit i. Denotemos este conjunto como G_k^i, siendo:

$$G_k^i = \{\alpha_k, \alpha_{k\oplus 2^i}\} \tag{4.4}$$

La partición en los conjuntos cerrados define una relación de equivalencia, en la cual los coeficientes que pertenecen al mismo conjunto cerrado forman una clase de equivalencia, así $G_k^i = G_{k\oplus 2^i}^i$. De esta forma, es necesario definir su representante canónico, que será aquel con el subíndice más bajo k de todos los coeficientes en G_k^i. Se debe tener en cuenta que estos conjuntos cerrados mínimos sólo serán coalescentes (orden 1) si i =0. Sin embargo, es posible obtener particiones de S en conjuntos cerrados no mínimos con una mayor cardinalidad y un mayor grado de coalescencia.

Proclama 5. *Con el fin de obtener un conjunto cerrado c-coalescente, para una compuerta de* 1-*cubit aplicada al cubit i, es necesario unir al menos un número de grupos* G_k^i *dados por*:

$$\begin{cases} 2^{c-1} & si\ i \le c, \\ 2^{c} & si\ i > c. \end{cases}$$

Proclama 6. *Un grupo cerrado c-coalescente para una compuerta de* 1-*cubit aplicada al cubit i tiene una cardinalidad dada por*:

$$\begin{cases} 2^{c} & si\ 0 \le i \le c-1, \\ 2^{c+1} & si\ i > c. \end{cases}$$

Es posible generalizar la expresión (4.4) para una compuerta aplicada a más de 1 cubit (cubits *a, b, c,* . . .). Entonces de la Ec.(4.3), resulta que el conjunto cerrado mínimo para un coeficiente α_k es:

$$G_k^{a,b,c,\ldots} = \bigcup_{u_a=0}^{1}\bigcup_{u_b=0}^{1}\bigcup_{u_c=0}^{1}\ldots\left\{\alpha_{k\oplus\left(u_a 2^a + u_b 2^b + u_c 2^c + \ldots\right)}\right\}. \tag{4.5}$$

Corolario 1. *Aplicando c compuertas de* 1-*cubit a los c cubits menos significativos resulta sobre una partición c-coalescente del espacio de coeficientes S tal como*:

$$\left\{G_0^{0,1,\ldots,c-1}, G_{2^c}^{0,1,\ldots,c-1}, G_{2.2^c}^{0,1,\ldots,c-1}, G_{3.2^c}^{0,1,\ldots,c-1}, \ldots\right\}.$$

4.3.3. Compuertas controladas

Aquí, se analiza la transformación $U_g = \begin{pmatrix} 1 & 0 & \\ 0 & 1 & \\ & & U \end{pmatrix}$ correspondiente a una compuerta controlada, siendo U la matriz asociada a una compuerta de 1-cubit. En general, se deduce de la Ec.(4.3) que el conjunto cerrado para el coeficiente α_k de una compuerta genérica de 2-cubits puede ser usada para una compuerta controlada sobre el cubit j controlada por el cubit i:

$$G_k^{i,j} = \left\{ \alpha_k, \alpha_{k \oplus 2^i}, \alpha_{k \oplus 2^i}, \alpha_{k \oplus 2^{i+j}} \right\}. \tag{4.6}$$

Siendo más precisos, el conjunto cerrado mínimo asociado con α_k se denota por:

$$C_k^{i,j} = \begin{cases} D_k^j = \{\alpha_k\} & si\, k \vee 2^i \neq k, \\ G_k^j = \left\{ \alpha_k, \alpha_{k \oplus 2^j} \right\} & si\, k \vee 2^i = k. \end{cases} \tag{4.7}$$

Los coeficientes α_k con $k \vee 2^i \neq k$ no son transformados, como $\alpha_k^{sal} = \alpha_k^{en}$. Entonces, solo la mitad de los coeficientes es transformada, con el mismo patrón como la compuerta de 1-cubit no controlada U aplicada al cubit objetivo. Desde el punto de vista de la coalescencia, con el fin de trabajar sobre los grupos cerrados c-coalescentes, es el cubit objetivo, j, aquel a ser considerado. Si $j < c$ se pueden tomar los mismos grupos cerrados, ya coalescentes. No es necesaria ninguna operación sobre los coeficientes que no se transforman en estos grupos. De otra manera, si $j \geq c$, se generan los conjuntos cerrados mínimos correspondientes a la mitad del espacio que se transforma (Ec.(4.7)), y entonces, se construyen conjuntos cerrados c-coalescentes al agruparlos.

4.3.4. Redes de compuertas

La simulación de una red de compuertas puede ser descompuesta en una secuencia de estados donde cada estado, U_i, tiene una menor complejidad que la transformación global $U_g = U_1 \cdot U_2 \cdot U_3 \ldots$. Se debe tener en cuenta que debido a que el producto de matrices no es conmutativo, una primera limitación a la descomposición de una red de compuertas en etapas es el orden de aplicación. En este punto, la próxima definición introduce lo que se denomina etapa de simulación. Quedará constituida por una secuencia de compuertas de la red, en el orden correcto, satisfaciendo dos características relacionadas con el tamaño de los conjuntos cerrados (cardinalidad) y el grado de localidad (coalescencia).

Definición 5. Se define un *estado de simulación* para una red de compuertas como la secuencia con más compuertas consecutivas, comenzando con una dada, en el orden original, que permite una partición del espacio de los coeficientes S en grupos, que satisfagan estas tres características:

(i) los grupos son cerrados para cada compuerta en el estado;

(ii) la cardinalidad de todos los grupos es como 2^r, para un cierto valor r;

(iii) todos los grupos son al menos c-coalescentes para un cierto c.

El algoritmo en la Fig.4.6 provee el conjunto de compuertas para un estado de simulación cumpliendo con la cardinalidad y los parámetros de coalescencia, r y c. El algoritmo comienza

tomando la primera compuerta de la red, aplicada al cubit *i*, cuyo conjunto mínimo cerrado se expresa en la Ec.(4.4). En cada iteración, G_k representa al mínimo conjunto cerrado para las compuertas siendo agregada a cada estado. Este conjunto puede duplicar su cardinalidad con cada nueva compuerta. El algoritmo se detiene con la última compuerta o cuando la cardinalidad de G_k supera 2^r. Como consecuencia, P_k representa un conjunto c-coalescente construido por la agregación de los conjunto s mínimos cerrados para todas las compuertas en el estado. Denotaremos cada conjunto cerrado P_k asociado a una etapa como el plano de coeficientes coalescentes. Con el fin de ilustrar el algoritmo se presentan varios ejemplos.

```
CrearSimulacionEtapa(r,c){
    G_k = {},  ∀ 0 ≤ k < N;
    Compuertas = {};
    while (existe_mas_compuertas()){
        (i, g)= obtenga_proxima_compuerta; /* Compuerta #g aplicada sobre el cubit #i */
        for (k=0; k<2^n, k++){
            /* Si 0 ≤ i ≤ c − 1, una partición de S puede ser encontrada en los
             * conjuntos coalescentes "c" ⋃_{u=0}^{2^c−1} G^i_{k⊕2^u} de cardinalidad 2^c
             * Si i ≥ c, una partición de S puede ser encontrada en los conjuntos
             * coalescentes "c" ⋃_{u=0}^{2^c−1} G^i_{k⊕2^u} de cardinalidad 2^{c+1}
             */
            G^i_k = {α_k, α_{k⊕2^i}}; /* Mínimo conjunto cerrado para la compuerta g */
            G^temp_k = G_k;
            porcada α_u ∈ G^temp_k{
            /* G_k es un conjunto cerrado para esta etapa incluyendo al candidato g */
                G_k = G_k ⋃ G^i_u;
            }
            if ( Card(G_k) > 2^r ) return Compuertas;
        }
        Compuertas = Compuertas ⋃{g};
        porcada α_k ∈ S {
            P_k = ⋃_{u=0}^{2^c−1} G_{k⊕2^u} /* P_k es una partición colescente "c" de S*/
        }
    }
    return Compuertas;
}
```

Fig.4.6: Un algoritmo para definir nuestra etapa de simulación orientada a la coalescencia.

Ejemplo 1 *(Aplicando el algoritmo comenzando con la compuerta dada).* Asumimos los estados con los parámetros *c* y *r* $(c < r \le n)$. Comenzando con una compuerta de 1-cubit o una controlada cuyo objetivo es el cubit *i*, surgen dos posibilidades:

• Si es $i \ge c$, el agrupamiento $P_k = \bigcup_{u=0}^{2^c-1} G^i_{k\oplus 2^u}$, definido en la Fig.4.6, es un conjunto cerrado para la transformación aplicada al cubit *i*, el cual es c-coalescente y de cardinalidad 2^c+1. Estos agrupamientos realizan una partición de *S* y verifica: $P_k = \bigcup_{u=0}^{2^c-1} G^i_{k\oplus 2^u} = G_k^{0,1,2,..,c-1,i}$.

• Por el contrario, si $0 \le i \le c-1$, el agrupamiento $P_k = \bigcup_{u=0}^{2^c-1} G^i_{k \oplus 2^u}$ también define a una partición c-coalescente, pero de cardinalidad 2^c. Por lo tanto, estos agrupamientos realizan una partición de *S* verificando: $P_k = \bigcup_{u=0}^{2^c-1} G^i_{k \oplus 2^u} = G_k^{0,1,2,\ldots,c-1}$.

En la próxima iteración, cuando se incorpora una nueva compuerta aplicada al cubit *j*, tres casos pueden ocurrir:

• Si $0 \le j \le c-1$ los conjuntos P_k que ya están cerrados tanto para *i* como para *j* no cambiarán. Entonces, el cubit *j* puede ser incorporado en el estado.

• Si *j* = *i*, los conjuntos P_k cambiarán. La compuerta puede ser incorporada directamente.

• Si $j \ge c$, $j \ne i$, un conjunto P_k se convertirá en $P_k = G_k^{0,1,2,\ldots,c-1,i,j}$, si $i \ge c$ o $P_k = G_k^{0,1,2,\ldots,c-1,j}$, si $i \le c-1$. En cualquier caso, esto significa que la cardinalidad de estos conjuntos c-coalescentes se duplica.

En general, el que la inserción de una nueva compuerta en el estado afecte al cubit *I* significa que los grupos c-coalescentes permanecerán igual cuando se satisfaga una de las siguientes condiciones: ya había una compuerta afectando al cubit *i* en el estado, o $0 \le i \le c-1$. Si no es así, la inserción de tal compuerta implica que la cardinalidad de los grupos cerrados c-coalescentes para el estado se duplica. El próximo resultado ilustra este hecho.

Proclama 7. *La cardinalidad de los grupos cerrados c-coalescentes generada desde el algoritmo en la Fig.4.6, para un estado de simulación, será* $2^{c+d} \le 2^r$ *si tal estado contiene*:

(i) *un número indeterminado de compuertas que afectan a los cubits entre* 0 *y* $c-1$, *y*

(ii) *las compuertas transforman la mayoría de los d cubits más significativos* (*extrictamente*) *que el cubit* $(c-1)$.

Corolario 2. *Como se require la máxima cardinalidad de la partición de S en los grupos cerrados c-coalescentes a ser como máximo* 2^r, *el número d de cubits, más significativos* (*extrictamente*) *cubit* $(c-1)$, *que puede ser transformado por el estado de simulación, que es como máximo* $r-c$, *es,* $d \le r-c$.

Ejemplo 2 *(compuerta factorizable-Kronecker* $U^{\otimes n}$ *).* Consideremos la transformación $U^{\otimes n}$ aplicada a un registro *n*-cubit, donde *U* es una compuerta genérica de 1-cubit. Comenzando por el cubit menos significativo, las compuertas se incorporarán en orden creciente. Luego de las primeras *c* compuertas, que se traducirá en una partición del espacio de coeficientes *S* en grupos cerrados c-coalescentes de cardinalidad 2^c: $G_k^{0,1,\ldots,c-1}$. Después de este punto, es posible adicionar $r-c$ compuertas más hasta que los grupos cerrados alcancen la más alta cardinalidad (2^r), dando lugar a una partición en $G_k^{0,1,\ldots,r-1}$. De esta manera, el primer estado (estado 0) comprende compuertas del cubit 0 al cubit *r-1*. El estado *p* incluirá aquellas compuertas aplicadas a los cubits desde $r+(r-c)(p-1)$ a $r+(r-c)p-1$, correspondientes a la partición en los grupos cerrados de la

forma $G_k^{0,1,\ldots,c-1,r+(r-c)(p-1),\ldots,r+(r-c)p-1}$. Para $n > r$, el número de estados va a seguir de la desigualdad $r+(r-c)(p-1) \le n-1 \le r+p(r-c)-1$. Este hecho se traduce en el siguiente resultado.

Proclama 8. *El número de estados producidos por el algoritmo en la Fig.4.6 para la transformación* $U^{\otimes n}$ *es*:

$$nEtapas = \begin{cases} 1, & si\ n \le r, \\ 1+\left\lceil \dfrac{n-r}{r-c} \right\rceil, & si\ n > r. \end{cases} \tag{4.7}$$

Notemos que de acuerdo a la Proclama 7 y el Corolario 2, para $U^{\otimes n}$, el número de cubits involucrados en el primer estado es como máximo r, estados intermedios subsiguientes implican $r-c$, pero el último estado puede tener menos que $r-c$ compuertas.

Ejemplo 3 *(Transformada de Fourier Cuántica (TFC)).* Este ejemplo analiza la descomposición en estados de la TFC, tal como se aplica en la Fig.4.7.

A diferencia de los ejemplos anteriores, aquí aparecen varias compuertas controladas. En general, su conjunto mínimo cerrado estará dado por la Ec.(4.6). No obstante, se puede considerar que los grupos cerrados para las compuertas controladas son equivalentes a aquellos de la compuerta de 1-cubit de la (Ec.(4.7) aplicada al bit objetivo), como se refiere a su análisis de la dependencia. El cubit de control solo indica si el cálculo asociado se realiza o no durante la simulación. Entonces, las compuertas controladas requieren la mitad del cálculo que las compuertas de 1-cubit con el mismo patrón de dependencias.

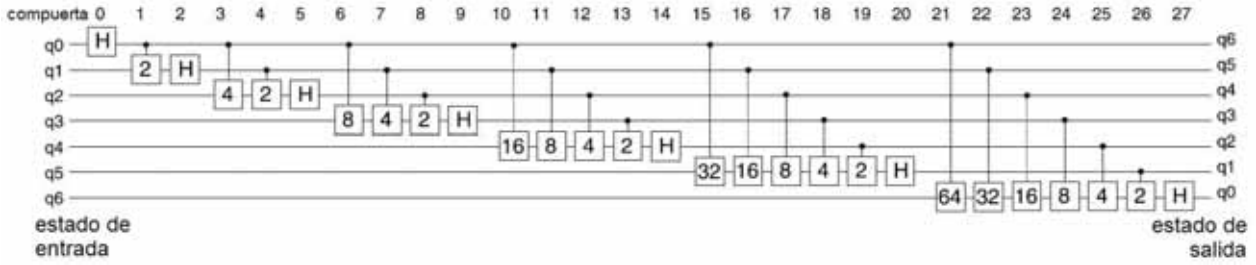


Fig.4.7: Una implementación de la TFC para un registro de 7 cubits.

La Tabla 4.1 ilustra la aplicación del algoritmo para valores particulares $c = 3$, $r = 5$. El número de estados resultantes es $\left\lceil \dfrac{n-r}{r-c} \right\rceil + 1 = 2$.

Tabla.4.1: Descomposición de la TFC en etapas de simulación observando coalescencia y parámetros de cardinalidad c = 3, r =5.

Compuerta	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27
Cubit de control	–	0	–	0	1	–	0	1	2	–	0	1	2	3	–	0	1	2	3	4	–	0	1	2	3	4	5	–
Cubit objetivo	0	1	1	2	2	2	3	3	3	3	4	4	4	4	4	5	5	5	5	5	5	6	6	6	6	6	6	6
Cardinalidad (P_k)	2^3	2^3	2^3	2^3	2^3	2^3	2^4	2^4	2^4	2^4	2^5	2^5	2^5	2^5	2^5	2^4	2^4	2^4	2^4	2^4	2^4	2^5	2^5	2^5	2^5	2^5	2^5	2^5
Etapa	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1

4.4 Implementación

Esta sección introduce la implementación del simulador desarrollado usando el modelo de programación en CUDA. El simulador lleva a cabo el cálculo del estado de salida de un computador cuántico teniendo en cuenta una transformación global U_g, como una secuencia de etapas.

El esquema general de la simulación se muestra en la Fig.4.8. El bucle más externo del lado del anfitrión atraviesa cada etapa invocando los códigos del núcleo, de modo que el cálculo de los coeficientes de salida se hace en paralelo en el dispositivo GPU.

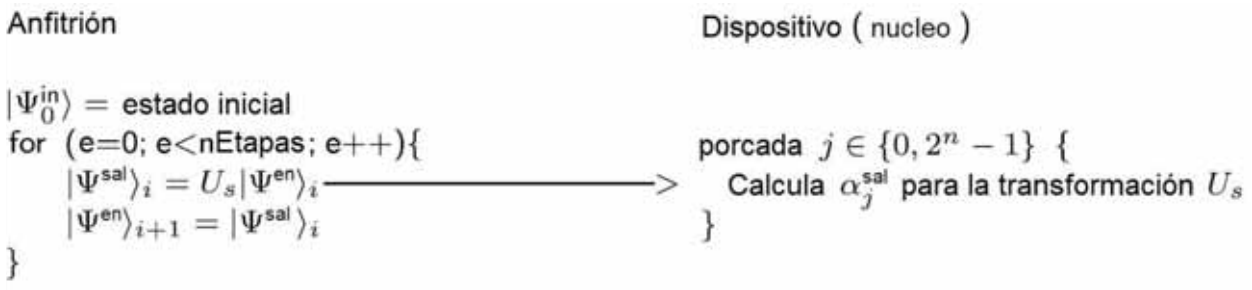


Fig.4.8: Esquema de simulación.

El enfoque para el núcleo se basa en la consideración de una partición del espacio coeficiente en conjuntos cerrados. Cada conjunto cerrado se puede procesar de forma independiente ya que no depende de ningún otro. Los conjuntos cerrados actúan como unidades de cálculo a nivel de bloque, y serán asignados a los bloques de CUDA, para que cada bloque esté a cargo del procesamiento de uno (o varios) de estos conjuntos. Los hilos en un bloque se encargarán de transformar los coeficientes de los conjuntos cerrados en el correspondiente paralelo de una ejecución SIMT.

Es necesario definir la granularidad de la asignación en dos niveles. Por un lado se debe definir cuáles son los grupos cerrados asignados a un bloque. Y a su vez, dentro del bloque, se debe definir cuántos y qué coeficientes de la función de cada proceso de hilo. Un hilo puede pasar de calcular sólo un par de coeficientes, hasta todo un grupo cerrado.

La dimensionalidad de la cuadrícula, es decir, el número de bloques y el número de hilos por bloque, determinará esta granularidad, a modo de aspectos condicionantes como la sincronización necesaria. En esta implementación, todo el vector de coeficientes reside en la memoria global del dispositivo. Sin embargo, como se refiere a la mejora del tiempo de ejecución, los hilos trabajan sobre copias locales de los conjuntos cerrados ubicados en la memoria compartida.

El desafío consiste en encontrar una función de asignación para los hilos en el espacio de cálculo (conjuntos cerrados) y los espacios de memoria (global y compartida) donde se colocan los coeficientes. Este hecho implica la definición de funciones de mapeo eficientes entre los identificadores de estos espacios, como se muestra en la Fig.4.9.

Desde el punto de vista del rendimiento, es importante tomar en cuenta la característica peculiar de memoria del modelo de memoria de CUDA. Con el fin de mantener un buen ancho de banda de transferencia y evitar la serialización de los hilos, es crucial llevar a cabo accesos coalescentes.

Fig.4.9: Espacios de hilos, cálculo y de coeficientes.

La estrategia de paralelización propuesta en este trabajo tendrá en cuenta las etapas definidas en la Definición 5. Este método aprovecha la localidad y las características de la jerarquía de memoria, lo que resulta en un buen desempeño. Vamos a designar a esta implementación conciente de coalescencia. Un caso particular es aquel en que la etapa consta de una sola compuerta. Esta implementación simplifica el código del programa, a expensas del rendimiento, y se utilizará sólo para fines de comparación. Esto último se conoce como ejecución compuerta-a-compuerta.

4.4.1 Implementación conciente de coalescencia:

Con el fin de obtener un alto rendimiento del sistema de memoria, es necesario el uso de la memoria compartida con forma de un caché de memoria rápida. Hay una diferencia importante: la carga de datos se hará de forma explícita. Desde el punto de vista del código del núcleo que se ejecutará en el GPU, este enfoque se traduce en tres pasos. En primer lugar, la copia de entrada del subconjunto de los coeficientes a ser transferidos para la memoria compartida. Este subconjunto tiene que ser cerrado para todas las compuertas en el estado. En segundo lugar, las compuertas que pertenecen a la etapa se aplicarán en este subgrupo. Este proceso tendrá lugar en la memoria compartida por lo que se hace necesaria una sincronización a nivel de hilo entre las transformaciones de compuerta. Por último, la etapa termina con la copia de salida de los coeficientes transformados, que serán devueltos a la memoria global. Estos coeficientes transformados vuelven a sus lugares de origen. Será esencial para llevar a cabo las operaciones de copia de entrada y salida de una manera coalescente.

La forma en que los coeficientes son asignados a los bloques de CUDA, y la forma en que se copian en la memoria compartida, se derivan de la Definición 5. Cada uno de los conjuntos resultantes P_k (Fig.4.6) asociados al estado pueden ser transferidos de una forma coalescente sin

haber dependencias entre diferentes P_k. De esta manera, cada P_k establecido resulta ser la carga de trabajo que se asignará a cada bloque de CUDA. El pseudocódigo de la Fig.4.10 describe los cálculos que cada bloque de CUDA debe llevar a cabo. Todos los cálculos, incluyendo las copias de entrada/salida, se realizarán en paralelo, siguiendo el modelo SIMT.

```
Copia_en(P_k)
hilossincro()
porcada U ∈ stapa {
        P_k = U(P_k)
        hilossincro()
}
Copia_sal(P_k)
```

Fig.4.10: Pseudocódigo núcleo para la simulación mental de coalescencia en el nivel de bloque.

Como un ejemplo, consideremos $c = 2$, $r = 4$. Una etapa de simulación para adaptarse a estas características se muestra en la Fig.4.11. El estado afecta a los cubits desde 0 hasta $c - 1$, y los dos cubits más significativos, i, j $(c \le i < j)$.

El algoritmo de la Fig.4.6 permite una partición del espacio coeficiente en planos P_k, al cual pertenece un coeficiente dado α_k . La Fig.4.11 muestra esta situación para un k tal que sus bits desde 0 a $c - 1$, i y j, son cero, que es

$$k \wedge \left(2^n - 1 - 2^i - 2^j - \sum_{u=0}^{c-1} 2^u\right) = k \, .$$

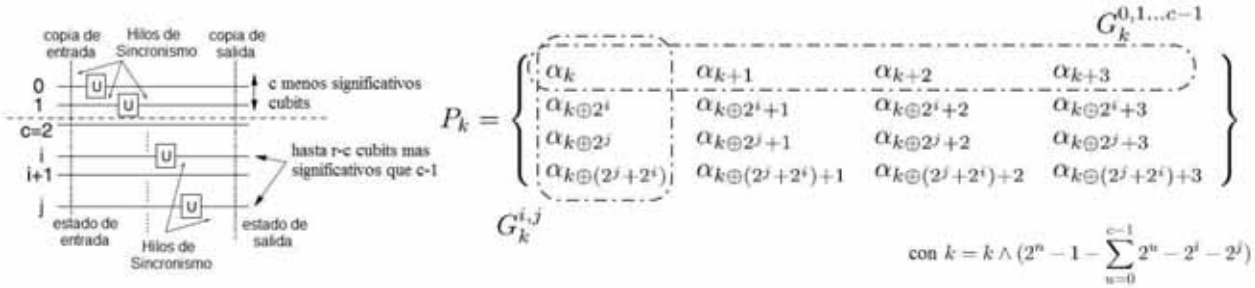


Fig.4.11: Ejemplo de plano coalescente para una etapa de simulación con *coalescencia* $c = 2$, *cardinalidad* $r = 4$.

Los conjuntos P_k pueden ser vistos como un plano hecho de $2^c \times 2^{r-c}$ coeficientes, en el interior del volumen de todos los coeficientes (Fig.4.12). Cada P_k establece una clase de equivalencia, de manera que cualquier coeficiente es un representante de esa clase. Por lo tanto, en este ejemplo, después de

$$k \wedge \left(2^n - 1 - \sum_{u=0}^{c-1} 2^u - 2^i - 2^j\right) = k$$

resulta que

$$P_k = P_{k+1} = \ldots = P_{k\oplus 2^i} = P_{k\oplus\left(2^j+2^i\right)+3} \, .$$

Se puede elegir el coeficiente α_k del valor más bajo de k como el representante canónico de la clase en cuestión. De esta forma, el plano P_k que contiene al coeficiente α_k, tendrá una función de canonización $P_{k_{canónica}} = P_k$ donde

$$k_{canónica} = k \wedge \left(2^n - 1 - \sum_{u=0}^{c-1} 2^u - 2^i - 2^j\right).$$

Generalmente, el subíndice para la función de canonización se obtiene poniendo a cero el valor de los bits 0 a $c-1$, y aquellos que están por encima de $c-1$ cubits deben ser transformados por la etapa. Como la asignación de bloques de CUDA se lleva a cabo en el nivel del grupo cerrado P_k, el número total de bloques será el número de planos, 2^{n-r}.

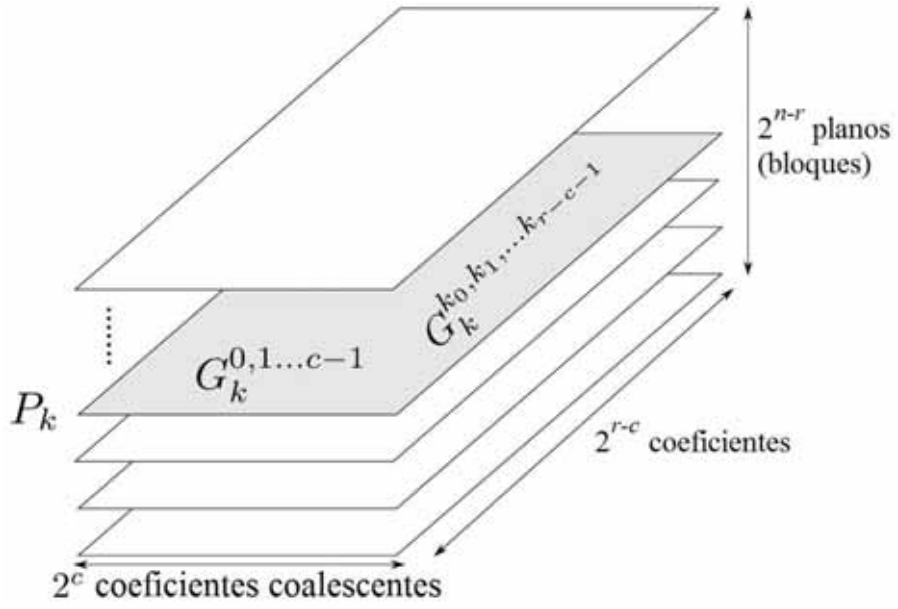


Fig.4.12: Espacio de coeficientes del vector de estados ordenado in P_k planos.

Continuando el ejemplo, la Fig.4.13 ilustra el procesamiento del plano P_0 por el bloque CUDA que le fue asignado. En esta primera etapa, los coeficientes que pertenecen a P_0 son copiados a la entrada de la memoria global a la memoria compartida manteniendo un grado de coalescencia 2^c. Luego de esto, todas las compuertas del estado son calculadas una a continuación de la otra, trabajando sobre la memoria compartida. Cada hilo será el encargado de calcular la transformación para un conjunto mínimo cerrado de esa compuerta, de hecho sólo para dos coeficientes, como se puede observar en la Fig.4.13. Como consecuencia, el número de hilos por bloque CUDA será la mitad del número de coeficientes en el interior del plano P_k, que es, $2^r/2$. Observar que los cálculos son llevados a cabo en el lugar, y se requiere una sincronización a nivel de hilo cada vez que se calcule una transformación de compuerta. Finalmente, en la copia de salida, los coeficientes ya procesados volverán a la memoria global. Esta transferencia también verifica la condición de coalescencia. Queda sin resolver cómo asignar cada uno de los 2^{n-r} planos P_k a los bloques. Los representantes canónicos, definidos previamente, tienen un subíndice de valor cero en los bits 0 a $c-1$, y también aquellos que están por encima de $c-1$ son transformados en la etapa. Entonces, el bloque identificado *bloqueId* puede ser mapeado a un plano P_k via inserción

de ceros en cada posiciones de la expresión binaria de *bloqueId*. De una manera más formal, el plano correspondiente al bloque *bloqueId* estará en este ejemplo, $P_{cero\_xmlitbit\_insert(bloqueId,\{0,1,\ldots,c,i,j\})}$, donde la función

$$cero\_bit\_insert\left(B,\{p_0,p_1\ldots,p_{M-1}\}\right)=\sum_{u=-1}^{M-1}\sum_{v=p_u+1}^{p_{u+1}-1}b_{v-(u+1)}2^v$$

con $p_{-1}=-1, p_M=m+M, p_0<p_1<p_2<\ldots<p_{M-1}$, para un B con expresión binaria

$$B=\sum_{u=0}^{m-1}b_u 2^u$$

inserta ceros en posiciones binarias $\{p_0, p_1, \ldots, p_{M-1}\}$. Entonces, si consideramos $i = 5$, $j = 6$, el bloque con identificador *bloqueId* = 11111_2 será el encargado del plano de cálculo

$$P_{cero\_bit\_insert(11111_2,\{0,1,5,6\})}=P_{110011100}.$$

Cuando no hay suficientes compuertas para constituir un plano de 2^r coeficientes cada bloque se encargará de procesar severos planos consecutivos de tal manera que la coalescencia efectiva se incrementa, hasta que se cumpla la cardinalidad 2^r. Luego de esto, el número de planos a ser procesados por un bloque será $2^r/Card(P_k)$.

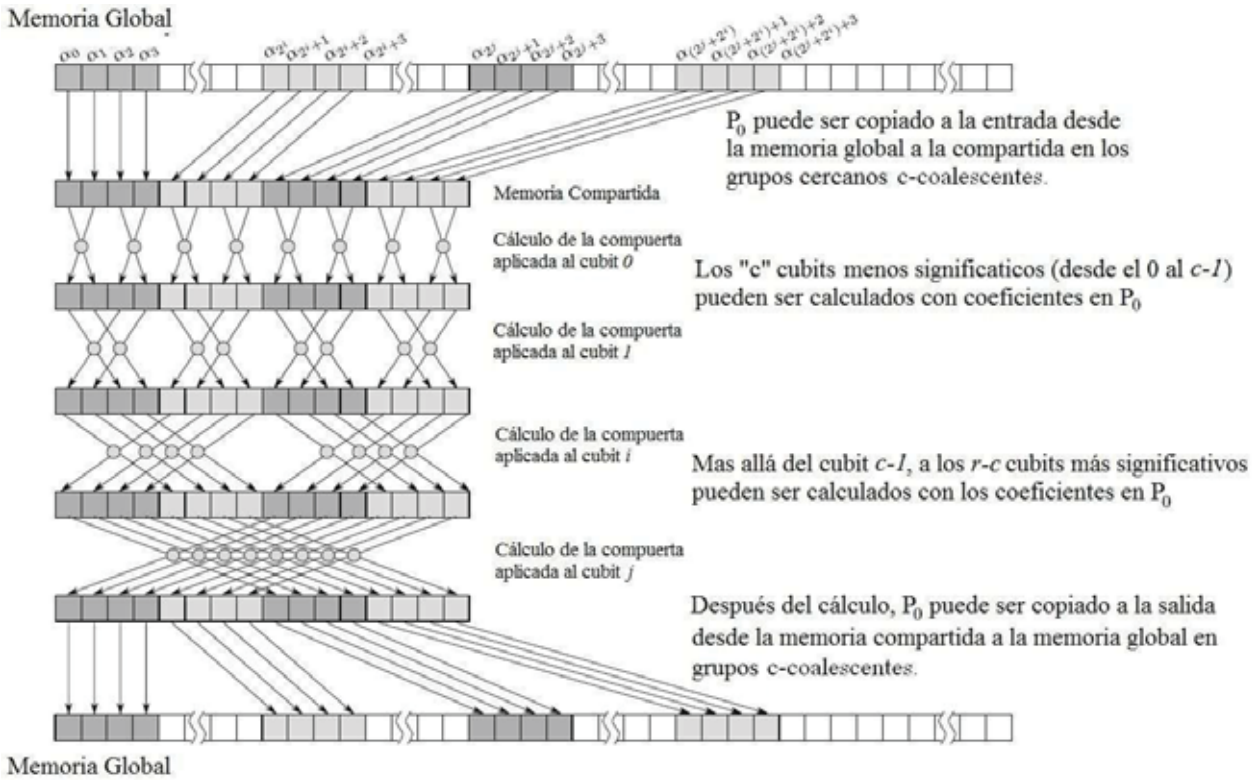


Fig.4.13: Estrategia mental de coalescencia para un bloque. Hilos consecutivos transfieren locaciones de memoria consecutiva durante el copiado de entrada/salida. Cada hilo está a cargo de un par de coeficientes que operan en el lugar.

4.4.2 Implementación compuerta-a-compuerta:

Con fines comparativos es interesante analizar el caso en el cual cada estado de simulación consiste de una única compuerta. En esta situación, no habrá ninguna reutilización de los coeficientes transferidos a la memoria compartida, siendo necesario una sincronización a nivel de anfitrión antes de comenzar con la siguiente compuerta. Para estos casos, una solución más simple es posible mediante la operación directamente sobre la memoria global, donde se almacenan todos los coeficientes. Aunque los tiempos de acceso a la memoria global serán peores, ninguna ventaja se logrará mediante la copia parcial de los coeficientes en la memoria compartida, ya que no se pueden volver a utilizar en absoluto.

Cada hilo es el encargado de calcular un grupo cerrado mínimo, operando en el lugar. Más formalmente, si el identificador de cuadrícula de hilo global de un hilo es

$tid = nBloques(bloqueId - 1)+hiloId$

tal hilo estará a cargo del cálculo del grupo cerrado

$$G^{i}_{cero\_bit\_insert(tid,i)} = \left\{ \alpha_{cero\_bit\_insert(t,i)}, \alpha_{cero\_bit\_insert(t,i)\oplus 2^{i}} \right\},$$

que como coeficientes del mismo grupo solo difiere en el bit *i*. Tener en cuenta que la carga de trabajo para cada hilo comprende no sólo el cálculo de sus dos coeficientes, sino también la lectura de la memoria global y escribir de nuevo en las mismas posiciones.

Observar que siempre habrá coalescencia si el qubit *i*, donde se aplica la compuerta, no es inferior a la dimensión de la trama. Sin embargo, cuando esta condición no se cumple, la falta de coalescencia dará lugar a una pérdida efectiva de ancho de banda. Para estos casos, será más eficiente hacer uso de la memoria compartida de manera similar a la aplicación conciente de coalescencia.

4.5 Resultados experimentales

En esta sección se analizan los resultados experimentales obtenidos después de la aplicación de las técnicas descritas anteriormente. Las simulaciones se ejecutan en una plataforma GeForce® GPU NVIDIA 8800GTX y el modelo de programación CUDA 1.1. Las principales características de la plataforma y CUDA se muestran en las Tablas 4.3 y 4.4.

Tabla.4.2: Características físicas principales de la plataforma GeFOrce8800GTX.

Características	Valor
Multiprocesadores en la GPU	16
Procesadores por multiprocesador	8
Frecuencia del reloj	1.35 GHz
Memoria Global	768 MB
Latencia de la Memoria Global	300+ ciclos
Memoria Compartida por multiprocesador	16 KB
Latencia de Memoria Compartida	4 ciclos
Registros de 32-bit por multiprocesador	8192

Tabla.4.3: Algunas características del modelo de programación de CUDA 1.1 relativos a la organización.

Características	Valor
Dimensión de la trama	32 hilos
Máx. número de hilos por bloque	512
Máx. dimensionalidad de una cuadrícula	65535x65535 bloques
Máx. dimensionalidad de un bloque	512 hilos por dimensión
Máx. número de bloques activos por multiprocesador	8
Máx. número de hilos activos por multiprocesador	768

Varias de las características físicas influyen directamente en los parámetros de simulación (*n*, *c* and *r*). Como cada coeficiente complejo está representado por dos 32-bit floats, el número máximo de cubits se deriva de los coeficientes asignables en la memoria global, $n = 26$ en este caso. Además, la memoria compartida limita el número de coeficientes asignados a cada bloque, es decir, la cardinalidad de los planos coalescentes P_k (parámetro *r*). Tener en cuenta que la cardinalidad se ha expresado como una potencia de 2 y otras variables de hilo también se colocan en la memoria compartida. De esta manera el número máximo de coeficientes que se puede asignar en la memoria compartida, resulta ser 2^{10} ($r = 10$). Por último, el parámetro *c*, que establece el grado de coalescencia está muy relacionado con la programación de los hilos en las tramas. Por lo tanto, aunque limitado por *r*, se espera un óptimo alrededor del número de hilos por trama. La Tabla 4.4 resume el impacto de las características de la plataforma.

Tabla.4.4: Impacto de las características de la plataforma sobre la simulación, asumiendo dos floats de 32-bit por coeficiente.

Caract. de la plataforma	Parámetros de simulación	Máx. valor
Dimen. de la Memoria Global	Número de cubits: n	26 cubits (n = 26)[a]
Dimen. de la Memoria Comp.	Coef. x plano de coalesc: 2^r	2^{10} coef. (r = 10)[b]
Máx. hilos por bloque	Número de hilos x bloque: 2^{r-1}	2^9 hilos (r = 10)
Dimen. de la trama	Grado de coales. en coef: 2^c	2^{r-1} (c = r-1)

[a] 768MB/(8 B/coef) = 96 Mcoef./global $\geq 2^n \Rightarrow \boldsymbol{n = 26}$, para cálculo en el lugar.
[b] 16KB/(8 B/coef) = 2^r + espacio extra para variables $\Rightarrow \boldsymbol{r = 10}$.

Los núcleos fueron compilados con *nvcc* de NVIDIA. Con el fin de obtener la mejor eficiencia, se fijan los parámetros de configuración (*r*, *c*), los cuales se definen como macros. Aunque se genera un binario diferente para cada configuración, esto ayuda a que el compilador optimice el binario para cada caso. De lo contrario se impediría varias optimizaciones del compilador, y aún más, el número de registros por hilo sería más alto, lo que limitaría el número de hilos activos en ejecución.

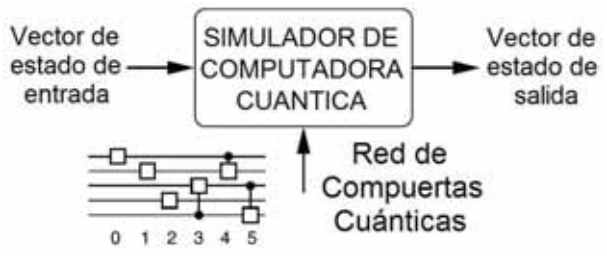


Fig.4.14: Operación del simulador de la computadora cuántica ideal propuesta.

Una entrada esencial para el simulador es la descripción de la red que se desea simular (Fig.4.14). Esto consiste en una serie de compuertas en el orden de simulación.

Experimentalmente se han abordado dos transformaciones cuánticas de interés: una transformación multicubit constituida por el producto de Kronecker de compuertas de 1-cubit ($U^{\otimes n}$) y la TFC. Dentro de un marco experimental, los factores que influyen en el rendimiento de simulador se van a analizar de modo que la mejor configuración se puede seleccionar para cada caso particular.

Para el primer caso, la transformación base ha sido la compuerta de Hadamard ($U = H$), que se traduce en la transformación Walsh. En este caso la red de compuertas de entrada es una secuencia de compuertas H, cada una aplicada a un cubit. En el caso de la TFC, como se muestra en la Fig.4.15(a), la red de compuertas consisten en una secuencia de $n(n + 1)/2$ compuertas. Sin embargo, es posible encontrar una configuración equivalente más eficiente agrupando las compuertas en pasos, como se muestra en la Fig.4.15(b), para la cual existente n transformaciones Q_i. Tener en cuenta que la complejidad computacional de Q_i es similar al de una compuerta de 1-cubit, dado que solo un cubit está controlado por otros menos significativos. Si no se proporciona ninguna otra información, todas las implementaciones analizadas siguientes se aplican sobre la base de la organización en pasos.

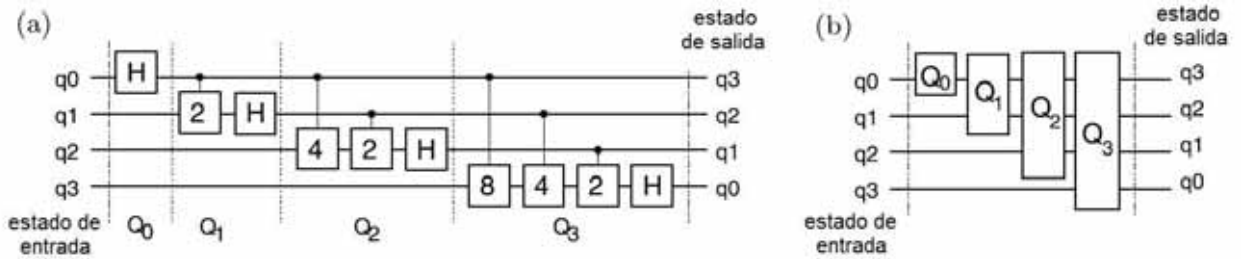


Fig.4.15: (a) Una implementación de la TFC usando Hadamard (1-cubit) y compuertas de desplazamiento de fase controlada (2-cubits). (b) Descomposición de la TFC en pasos.

4.5.1 Impacto de la coalescencia

La programación de los hilos en tramas se refleja en la condición coalescente, que afecta las etapas de copia de entrada/salida de la simulación. El parámetro c garantiza que lo que transfiere en-

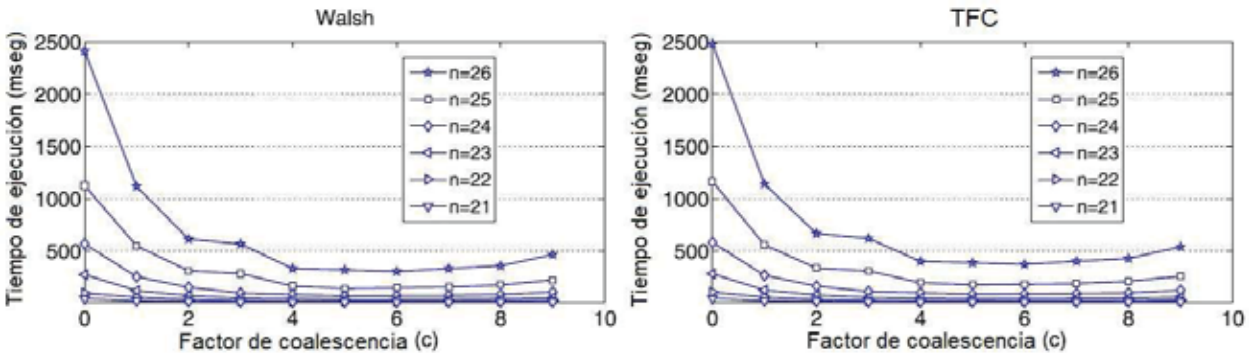


Fig.4.16: Impacto del factor de coalescencia sobre el tiempo de ejecución.

tre las memorias global y compartida se realiza con una mínima coalescencia de 2^c coeficientes. La Fig.4.16 muestra la influencia de c sobre el tiempo de ejecución de las transformadas de Walsh y TFC de tamaño de n cubits aplicados sobre un registro de n-cubits. Los datos fueron obtenidos para la máxima cardinalidad posible r = 10, y un número variable de cubits en el registro.

Los resultados experimentales confirman esto. Debe observarse que el tiempo de ejecución decrece a medida que la coalescencia se aproxima al tamaño de la trama. Un mínimo es alcanzado de c = 4, con un ligero incremento para valores mayores de c. Este incremento se debe al aumento en el número total de etapas con el incremento de c (Ec.(4.7)). Para una cierta cardinalidad, un c más pequeño se traduce en menos compuertas por etapa y entonces, un número mayor operaciones de copia de entrada/salida y operaciones de sincronización del anfitrión. Si es menos que 4, el tiempo de cálculo empeora drásticamente debido a la serialización de los accesos a la memoria global causado por la falta de coalescencia. Si está por encima de 7, el tiempo de ejecución se incrementa ligeramente debido a la sobrecarga producida por la apertura y cierre de un número cada vez mayor de etapas. Por lo tanto, la mejor performance se alcanza para los valores de c en el intervalo de 4 a 7.

4.5.2 Impacto de la cardinalidad

Las transformaciones aplicadas sobre un número variable de cubits consecutivos (de 0 a $i-1$) para un registro con n = 26 cubits, se muestran en las Figuras 4.17 y 4.18 para un amplio rango de configuraciones de simulación. Los gráficos del lado derecho muestran la ampliación de un área de interés particular. Notemos que las curvas son moduladas por la partición de la red de compuertas en las etapas de simulación (Ec.(4.7)). De esta forma, hay solo una etapa para los primeros puntos r, resultando en un crecimiento lineal del tiempo de ejecución para todas las configuraciones. Diferentes valores de r dan lugar a pasos cada $r-c$ cubits, correspondiente a la inicialización de cada nueva etapa. Esta sobrecarga es amortizada con la próximas $r-c$ compuertas. Un caso particular se da cuando $c = r-1$, para el cual las etapas más allá de la compuerta r tienen solo una compuerta. En este punto, las operaciones de copia de entrada/salida causa la sobrecarga principal, resultando en un gráfico lineal.

El parámetro de cardinalidad r no solo determina el número de etapas y compuertas por etapa, sino también el tamaño de los planos de coalescencia y por lo tanto, el número de bloques e hilos.

Para una transformación i-cubit, el número de bloques CUDA es 2^{i-r} , y el número de hilos por bloque es 2^{r-1}. Luego de estos, parece que sería óptimo tener en cuenta el tamaño máximo permitido, r = 10 (1 K coeficientes en la memoria compartida). No obstante, otro factor a tener en cuenta es la *ocupancia*, como varios bloques activos pueden coexistir en el mismo multiprocesador si hay suficientes registros y memoria compartida dispo-nible. La ocupancia es el radio entre el número total de hilos activos y el máximo admisible (ver la Tabla 4.3). Por lo tanto, si r = 10, entonces 2^{r-1} = 512 hilos/bloque, y no es posible tener más bloques activos en el mismo multiprocesador, ya que no hay espacio suficiente en la memoria compartida para ubicar más coeficientes. Cuando r = 9, es 2^{r-1} = 256 hilos/bloque, el número de coeficientes por bloque ocupa 4 KB. Este espacio libre en la memoria compartida puede ser usado por otros bloques, aumentando la ocupancia. La Tabla 4.5 muestra las restricciones impuestas sobre el número de bloques activos (*Nb*), mientras la Tabla 4.6 provee los valores de ocupancia para diferentes valores de r, destacando que es el factor limitante para cada uno de ellos.

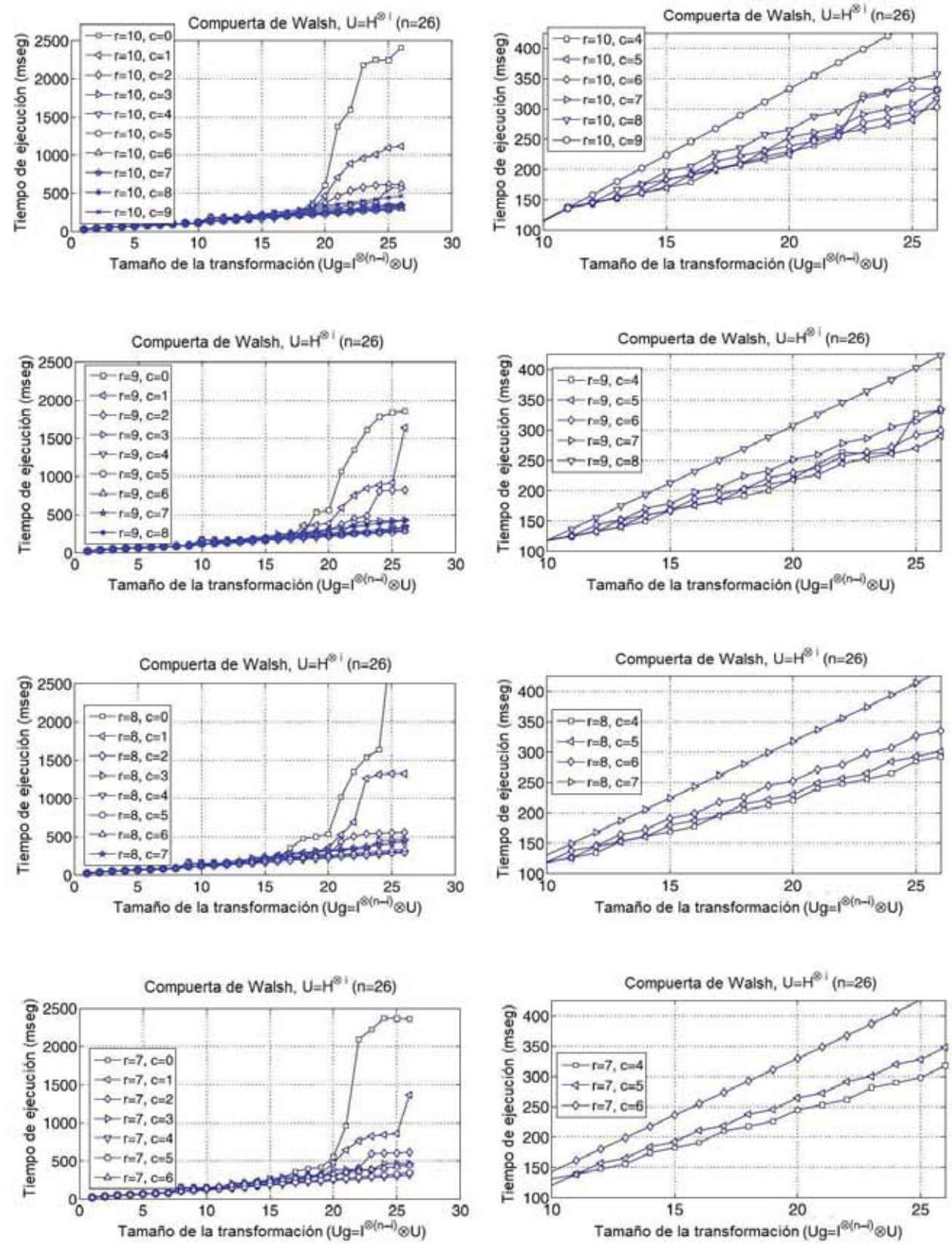


Fig.4.17: Resultados luego de aplicar la compuerta de Hadamard a un número variable de cubits de un registro de 26-cubits para diferentes valores de coalescencia y cardinalidad.

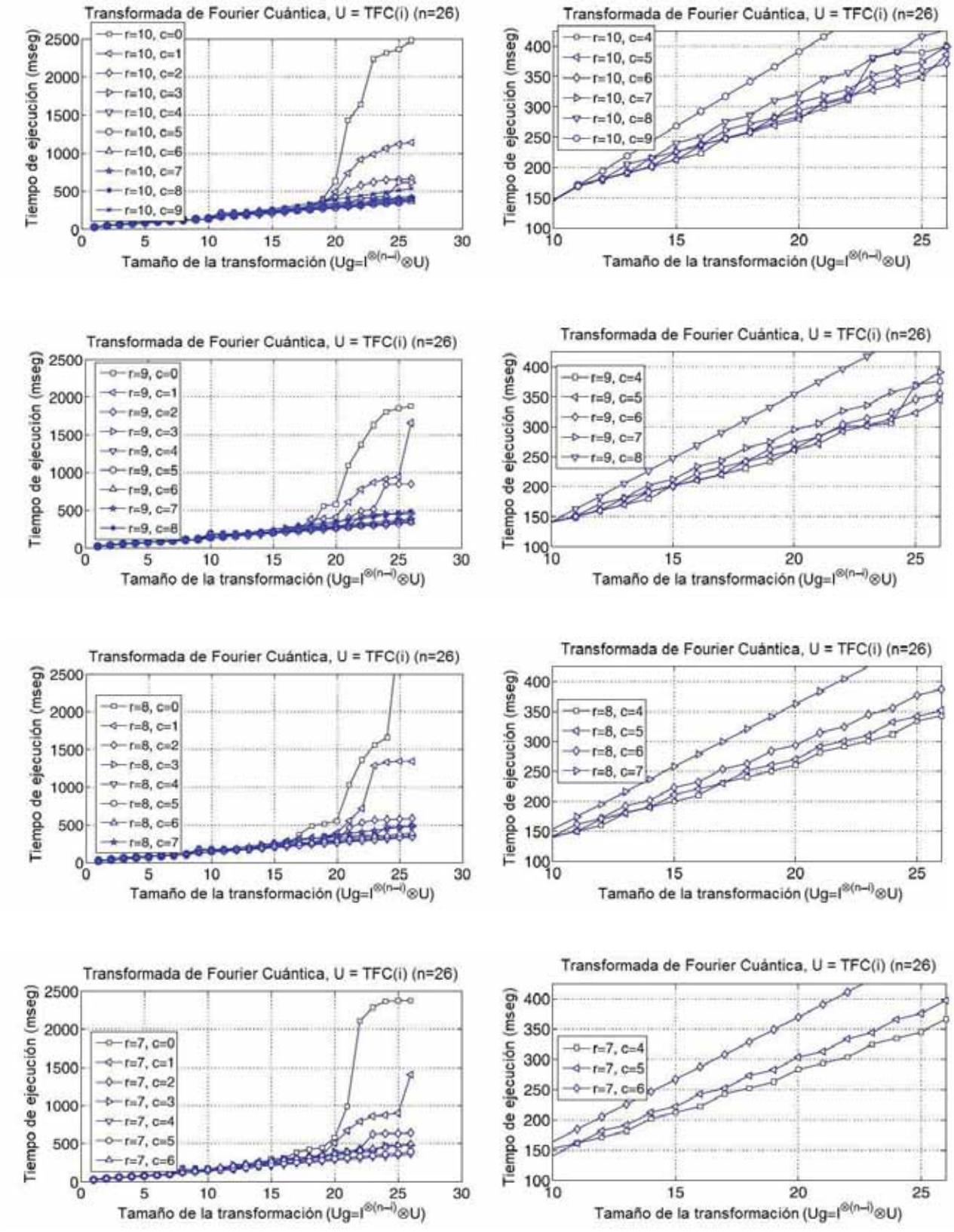


Fig.4.18: Resultados luego de aplicar la TFC a un número variable de cubits de un registro de 26-cubits para diferentes valores de de coalescencia y cardinalidad.

Una ocupancia máxima es alcanzada para configuraciones de $r = 9$ y $r = 8$. Otro factor limitante es el número de registros que cada hilo demanda, determinado en el tiempo de compilación. Todos nuestros códigos kernel del simulador están ajustados para cumplir con este límite.

Tabla 4.5: Los bloques activos por multiprocesador están restringidos por las características de programación de la plataforma.

Limitación	Número de bloques activos sujetos a
Número de bloques activos por multiprocesador	$N_b \leq 8$
Número de hilos activos por multiprocesador	$N_b . 2^{r-1} \leq 768$
Número de tramas activas por multiprocesador	$N_b . 2^{r-1} / 32 \leq 24$
Número de hilos por bloque	$2^{r-1} \leq 512$
Memoria compartida por multiprocesador	$N_b . 2^{r} \leq (2^{11}$-espacio extra para variables)
Número total de registros de 32 bits por multiprocesador	$N_b . 2^{r-1}$. registros por kernel ≤ 8192

Tabla 4.6 Ocupancia lograda mediante diferentes configuraciones.

r	*h/b*	*memcom/b*	*# b.a*	*# h.a*	Ocupancia	Factores limitantes
10	**512**	**8 KB** + vars.	1	512	2/3	*memcom/b, h/b*
9	256	**4 KB** + vars.	3	**768**	1	*memcom/b*
8	128	2 KB + vars.	6	**768**	1	-
7	64	1 KB + vars.	**8**	512	2/3	# a.b
6	32	512 B + vars.	**8**	256	1/3	# a.b

h/b = hilos por bloque; memcom/b = memoria compartida por bloque; b.a = bloques activos por multiprocesador; h.a. = hilos activos por multiprocesador; *vars*. = espacio extra para variables.

En un sentido estricto, si hay más hilos que procesadores por multiprocesador, una ocupancia abajo de la unidad no implica necesariamente un menor paralelismo efectivo. Sin embargo, una ocupancia más pequeña significa menos oportunidades para latencia oculta, lo que empeora el ancho de banda de la memoria efectiva.

Los factores determinantes del tiempo de cálculo para una cierta parametrización de la simulación (*n*, *r* y *c*) resulta ser la coalescencia, el número de bloques e hilos, la ocupancia y el tamaño de la memoria compartida. Observemos que el comportamiento de las curvas en diferentes experimentos es bastante regular. Casi en todos los casos, el valor óptimo es alcanzado para $r = 9$ cuando *c* se encuentra entre 4 y 7.

4.5.3 Ancho de banda

La Fig.4.19 muestra el tiempo de ejecución cuando se simula sólo una compuerta aislada ubicada en el cubit *i* de un registro de 26-cubit, con una entropía máxima del estado inicial. este experimento ilustra el hecho de que no usar la memoria compartida o no reusar datos resulta en una pobre performance. En general, cuando no hay datos a reusar, es mejor trabajar directamente en la memoria global (implementación compuerta-a-compuerta). No obstante, una falta de coalescencia es aun peor, como es el caso para los primeros cubits. Sin embargo, si hay más que una compuerta por etapa, se ha demostrado que la implementación conciente de coalescencia es ventajosa, ya que amortiza los gastos generales existentes.

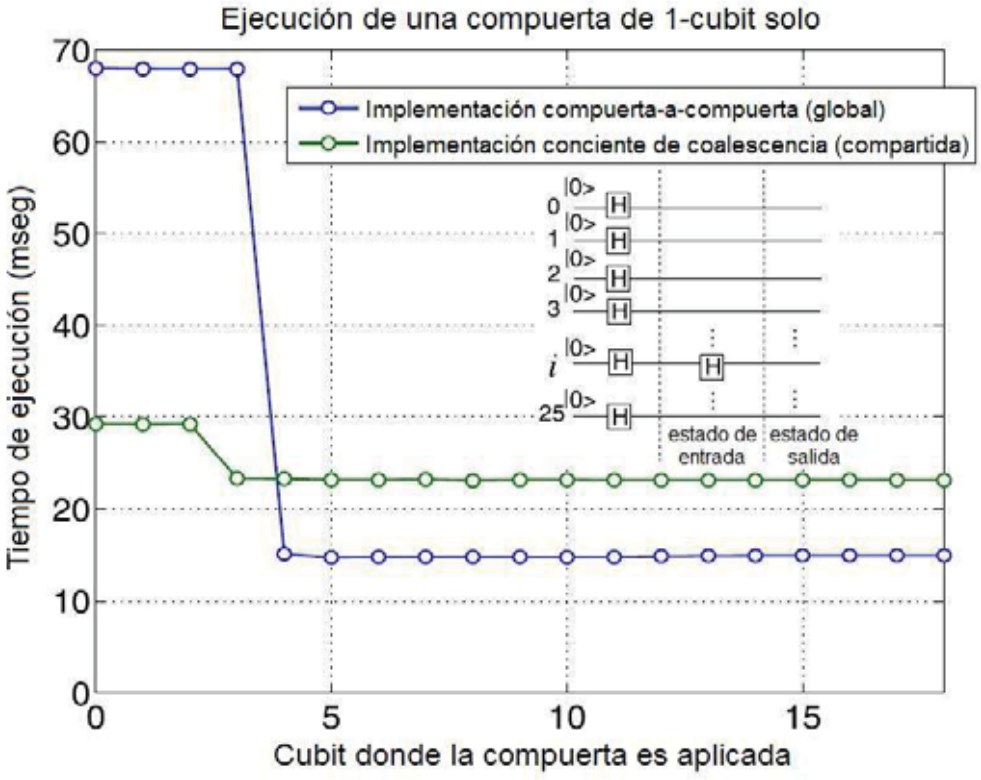


Fig.4.19: Compuerta de 1-cubit solo no amortiza el uso de la memoria compartida.

Hay dos fuentes de sobrecarga: transferencia entre espacios de memoria y el índice de cálculo. En orden de cuantificar estas sobrecargas, una transformación de identidad sintética ha sido codificada, implicando solamente copia de entrada/salida y los cálculos de los índices de asignación de los espacios de memoria, sin la carga útil computacional. Asumiendo que el tiempo de ejecución de la transformación bajo estudio (T_U) incluye la misma sobrecarga que la identidad de referencia (T_I), una medida de la eficiencia representa la carga sobre el tiempo total debería ser: $\eta = \frac{T_B - T_{Iref}}{T_B}$.

Se puede proponer un modelo de la performance para la implementación conciente de coalescencia, considerando severos componentes en el tiempo de ejecución: la carga computacional (T_{comp}), el índice de cálculo (T_{ind}), la copia de entrada/salida y los esfuerzos de sincronización del anfitrión (T_{cp}) y el tiempo de calentamiento asociado al núcleo de lanzamiento (I): $T_U = T_{comp} + T_{ind} + T_{cp} + I$.

Concerniente al tiempo de ejecución de un bloque, la carga y los tiempos de cálculo de los índices serán proporcionales al número de coeficientes a ser procesados por tal bloque (2^r) y el número de compuertas: $T_{comp} = K_1\, 2^r\, i$, $T_{ind} = K_2\, 2^r\, i$. El gasto general impuesto por la copia de entrada/salida será proporcional al número de ambos, las etapas (*nEtapas* después de la Ec.(4.7)) y los coeficientes a transferir. Si el gasto general de la primera etapa se considera independientemente, entonces $T_{cp} = K_3\, 2^r (nEtapas - 1)$. Por lo tanto, el término I incluirá la sobrecarga asociada a esta primera etapa tan bien como aquellos asociados al kernel de lanzamiento. De esta forma será:

$$\eta = \frac{K_1 2^r i}{(K_1+K_2)2^r i+K_3 2^r (nEtapas\text{-}1)+l},$$
$$con\ nEtapas = \begin{cases} 1, & if\ i \le r, \\ 1+\left\lceil \frac{i\text{-}r}{r\text{-}c} \right\rceil, & if\ i > r. \end{cases} \tag{4.8}$$

Con objeto de validar este modelo, se aplica un método de regresión no lineal. Los gráficos de estimación del error para los valores predichos se muestran en la Fig.4.20. Observemos que para parámetros que implican menos compuertas por etapa (menor que $r - c$), la sobrecarga se incrementa, como menos se aprovecha de las operaciones que producen la sobrecarga. Esto es similar cuando comienza una nueva etapa. Se produce una pequeña disminución local en las curvas, de manera tal que la sobrecarga asociada no será compensada hasta que no sean suficientes las compuertas.

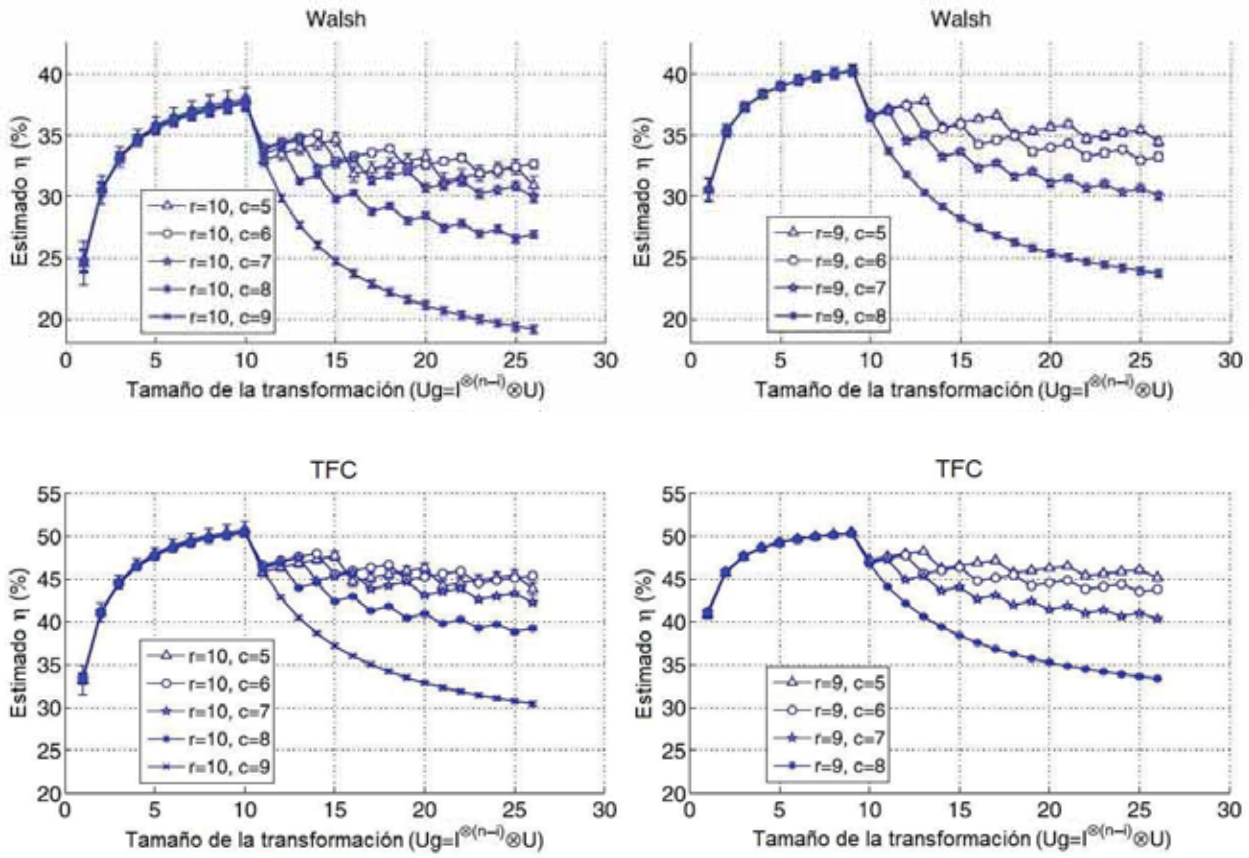


Fig.4.20: Gráficos del error de estimación del modelo, para Walsh y TFC aplicadas a un número variable cubits con $n = 26$.

La Tabla 4.7 muestra los resultados obtenidos de la predicción, que emite bajos errores relativos, en la mayoría de los casos por debajo del 2%. La Fig.4.21 muestra los resultados experimentales conformando los diferentes componentes de tiempo estimados. Estos gráficos no solo muestran los tiempos para Walsh, TFC, y la identidad de referencia, sino también para otras transformaciones sintéticas las cuales solo realizan copias de entrada/salida para las etapas, sin cálculo de índices o coeficientes.

Tabla 4.7: Estimación de parámetros para el modelo de ejecución. Los componentes del tiempo de ejecución son determinados para un tamaño de la transformación igual al tamaño del registro.

n	*r*	*c*	*radio c.i.*	Error relativo	Carga computacional	Cálculo del índice	Copia ent/sal
Walsh:							
26	8	5	28%	1.2%	35%	45%	20%
26	8	6	28%	1.2%	31%	41%	28%
26	8	7	28%	1.4%	24%	32%	44%
26	9	5	33%	1.4%	36%	47%	17%
26	9	6	32%	1.4%	34%	46%	20%
26	9	7	31%	1.4%	31%	41%	28%
26	9	8	31%	1.5%	25%	32%	43%
26	10	5	55%	3.1%	32%	45%	23%
26	10	6	28%	1.5%	33%	49%	18%
26	10	7	28%	1.6%	30%	45%	24%
26	10	8	27%	1.6%	28%	40%	32%
26	10	9	30%	2.0%	20%	28%	52%
TFC:							
26	8	5	12%	0.5%	43%	40%	17%
26	8	6	12%	0.5%	40%	36%	24%
26	8	7	11%	0.5%	33%	30%	37%
26	9	5	15%	0.5%	45%	42%	13%
26	9	6	12%	0.4%	44%	41%	15%
26	9	7	10%	0.4%	40%	38%	22%
26	9	8	11%	0.4%	33%	31%	36%
26	10	5	42%	2.2%	44%	37%	19%
26	10	6	12%	0.6%	45%	40%	15%
26	10	7	13%	0.7%	42%	37%	21%
26	10	8	13%	0.7%	39%	34%	27%
26	10	9	15%	0.9%	31%	26%	43%

Estos resultados destacan que las estrategias centradas en la explotación de la memoria compartida implican gastos inevitables. este hecho se traduce en una limitación de las más altas posibles *r* para el tamaño de la memoria compartida y la ocupancia.

4.5.4 Escalabilidad

Esta subsección analiza cómo las diferentes estrategias bajo estudio escalan con el tamaño del problema. Por razones comparativas, la Tabla 4.8 muestra el tiempo de ejecución (en *mseg*) de las estrategias paralelas, compuerta-a-compuerta y coalescencia conciente, y dos implementaciones secuenciales sobre una CPU. Para una coalescencia conciente, *r* y *c* se corresponden con la configuración más rápida. Además, el número de etapas de simulación generadas, de acuerdo a Ec.(4.7), se muestra en la columna de *nEstados*. Esto corresponde al número de núcleos del código a ser invocados. Los cocientes en la *última* columna representa al número de compuertas restantes en la última etapa con respecto al número de compuertas por etapas ($r-c$). Esta última fracción provee una medida del efecto frontera asociado a la última etapa. Un valor menor que 1 implica una reutilización de datos peor que todas las etapas anteriores.

Las ejecuciones secuenciales fueron conducidas sobre una PC de 2 GB RAM con un procesador Intel Core2 6400 @ 2.13 GHz. Los resultados fueron obtenidos con la librería *libquantum* [161] y también con una versión optimizada y simplificada, desarrollada por sus autores basados en el código fuente de *libquantum*. Esta versión minimiza las sobrecargas innecesarias, reteniendo la

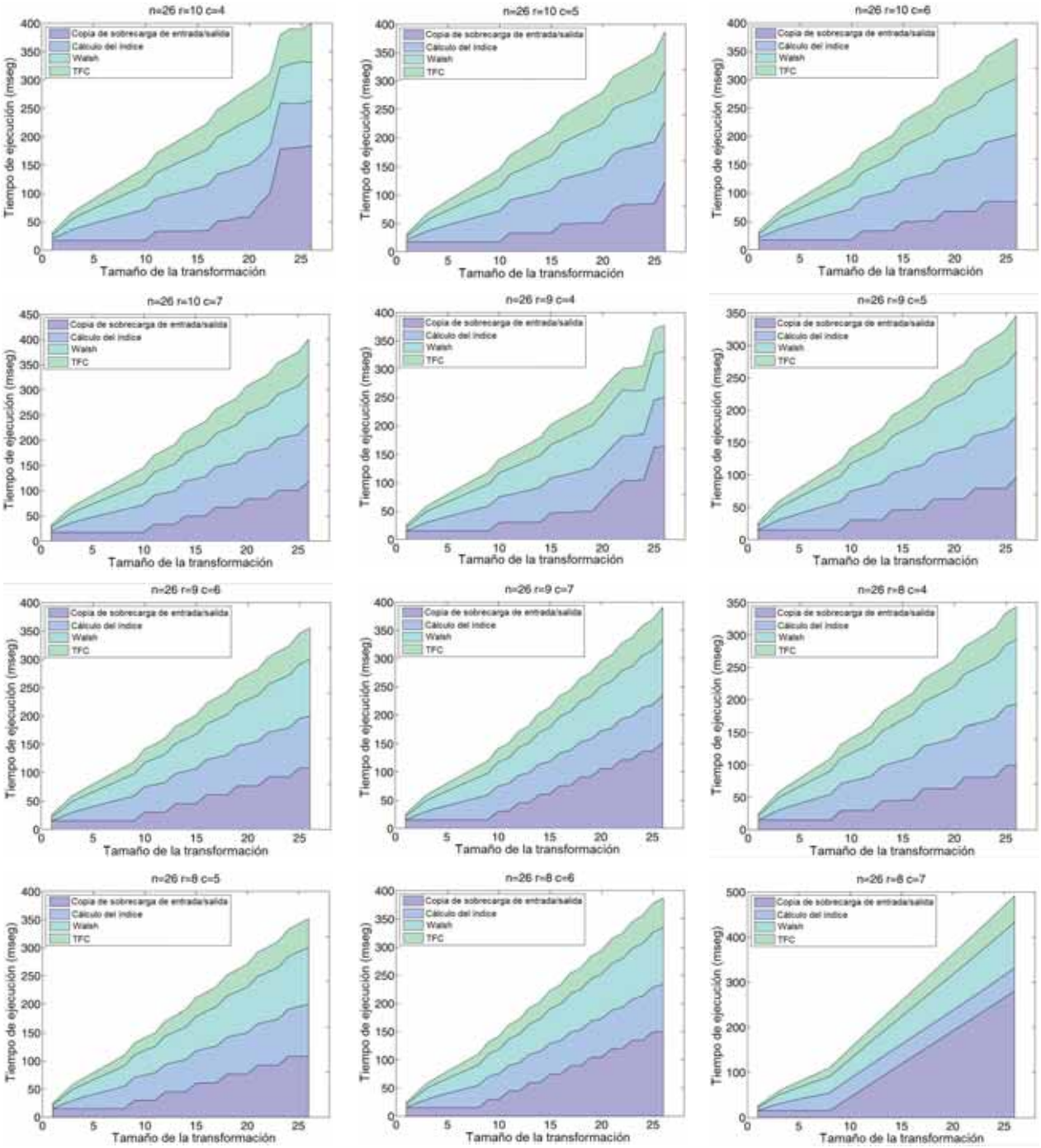


Fig.4.21: Componentes del tiempo de ejecución medidos para diferentes parámetros de simulación cuando las compuertas de Walsh y TFC son aplicadas a un número variable de cubits.

funcionalidad esencial requirida para los propósitos de prueba, los cuales permiten una comparación mágica con las ejecuciones en la GPU.

En el caso de la transformada de Walsh, donde hay tantas compuertas como cubits, el tiempo de ejecución esperado para códigos secuenciales crece después de $O(n2^n)$ cuando se aplica a un registro de n-cubits. En el caso de la TFC, donde el número de compuertas es $n(n+1)/2$, el tiempo

Tabla 4.8: Tiempo de ejecución de los experimentos de Walsh y TFC para diferentes implementaciones.

n	Modelo de programación CUDA sobre GPU						Cód. sec. sobre GPU		speed-up
	comp-x-comp	Coalescencia conciente					libquantum	optimizado	t_{CPU}/t_{GPU}
	t (mseg)	*r*	*c*	*nEtapas*	*último*	*t* (mseg)	*t* (mseg)	*t* (mseg)	
Walsh:									
15	0.32	10	4	2	5/6	0.11	4.42	1.76	16.3
16	0.46	10	4	2	6/6	0.20	10.57	3.82	18.9
17	0.78	9	4	3	3/5	0.42	32.42	7.90	18.7
18	1.39	9	4	3	4/5	0.81	83.07	18.82	23.1
19	2.68	9	4	3	5/5	1.63	179.72	47.89	29.4
20	5.41	9	5	4	3/4	3.48	380.81	100.45	28.9
21	11.07	9	5	4	4/4	7.15	825.84	210.78	29.5
22	22.82	9	5	5	1/4	15.41	1688.36	439.86	28.5
23	47.28	9	5	5	2/4	31.73	3364.75	919.95	29.0
24	98.26	9	5	5	3/4	65.39	7066.15	1927.47	29.5
25	203.69	9	5	5	4/4	135.15	15077.27	4052.66	30.0
26	422.36	9	5	6	1/4	290.08	32729.87	8774.42	30.2
TFC:									
15	0.46	10	4	2	5/6	0.13	5.35	10.58	82.7
16	0.63	10	4	2	6/6	0.24	13.07	18.86	77.0
17	0.98	9	5	3	4/4	0.52	42.67	40.09	77.8
18	1.62	9	4	3	4/5	0.98	120.39	86.90	88.6
19	2.99	9	4	3	5/5	1.96	254.19	187.59	95.8
20	5.78	8	4	4	4/4	4.14	520.33	389.25	93.9
21	11.60	9	5	4	4/4	8.57	1107.78	816.98	95.3
22	23.71	8	4	5	2/4	18.26	2264.26	1715.42	93.9
23	48.84	8	4	5	3/4	37.67	4631.77	3588.86	95.3
24	101.04	9	4	4	5/5	76.60	9752.44	7492.01	93.8
25	209.12	9	5	5	4/4	161.58	22878.99	15642.21	96.8
26	433.03	8	4	6	2/4	343.30	55325.69	32692.91	95.2

de ejecución es más alto como se muestra para *libquantum*. Sin embargo, al agrupar compuertas TFC en pasos (ver la Fig.4.15) se mantiene el orden $O(n2^n)$. Esta última aproximación ha sido seguida en el código secuencial optimizado tan bien como en todas las implementaciones sobre GPU.

De la Tabla 4.8 se deriva un hecho importante que consiste en que la simulación conciente de coalescencia siempre presenta un mejor tiempo de ejecución que la de compuerta-a-compuerta. Esto indica que es obligatoria para explorar la jerarquía de memoria de la GPU cuando se busca un alto rendimiento. Observemos que, especialmente para un número alto de cubits, *n*, se alcanza el tiempo de ejecución más pequeño para los parámetros *r* y *c* ajustando las características del dispositivo. De esta forma, el *r* óptimo provee la máxima ocupancia en GPU, limitada por el tamaño de la memoria compartida. El parámetro *c*, estrechamente relacionado con los datos de coalescencia, es siempre 4 o 5, que es, una media trama (16 hilos) o una trama (32 hilos).

La columna extrema derecha, rotulada *speed-up*, es el cociente entre el tiempo de ejecución de la versión secuencial optimizada de CPU y la implementación conciente de coalescencia de la GPU. Los experimentos de TFC proveen un mejor speed-up cuando son comparados con Walsh debido a la mayor carga computacional de la TFC, que resulta en una sobrecarga de paralelización relativamente menos significante. Notemos que las implementaciones paralelas sobre la GPU exhiben una buena escalabilidad debido a que el speed-up no se degrada cuando aumenta el tamaño del problema.

A pesar de que es difícil comparar plataformas heterogéneas, la estrategia propuesta conciente de coalescencia esta cerca de alcanzar un speed-up de 100 en el mejor de los casos para TFC, cuando es comparada con la CPU. Esta es una figura considerable, tomando en cuenta que el dispositivo GPU tiene 128 procesadores de transmisión.

Las simulaciones del Capítulo 7 mostrarán las virtudes de los aportes de los 2 próximos capítulos precisamente sobre la mencionada plataforma. Las simulaciones incluirán en todos los casos cálculo paralelo intensivo tanto para Memorias Matriciales Correlacionadas Cuánticas (MMCCu), como así también para aplicaciones en Procesamiento Cuántico de Imágenes de varios tipos y su contraparte Booleano/Binario.

4.6 Conclusiones del capítulo

En este capítulo proponemos enfoques para la simulación de un computador cuántico ideal utilizando el modelo de programación CUDA en GPUs de NVIDIA, el cual ha sido analizado en este trabajo. Se puede señalar que la explotación explícita de la memoria rápida en el chip a través de latencia oculta, es un problema esencial cuando se persigue una alta eficiencia. Además, en este trabajo se desarrolla un marco de análisis que nos permite seleccionar los parámetros de simulación sobre algunos de los principales detalles de la plataforma.

Se han tenido en cuenta a tal efecto las características importantes como la localidad de referencia de memoria, el tamaño de la memoria en el chip y la ocupación de procesador. A partir de este análisis, las configuraciones para las simulaciones óptimas son experimentalmente obtenidas y evaluadas. Los resultados muestran que las estrategias propuestas son capaces de lograr altas aceleraciones en este tipo de multiprocesadores de fácil adquisición. En una GPU típica de 128 procesadores de transmisión un registro de 26-cubit puede ser simulado hasta casi cien veces más rápido que en código secuencial en una plataforma monoprocesador convencional contemporánea.

Capítulo 5: Procesos de Ortogonalización Clásico, Booleano y Cuántico con sus Inversas

Introducción

En este capítulo se describen los procesos de ortogonalización necesarios en cada ámbito (clásico, Booleano y cuántico) para ortogonalizar los patrones de entrada de lo que describiremos en el Capítulo 6 como una Memoria Matricial Correlacionada (MMC), con objeto de mejorar su entrenamiento y posterior funcionamiento. De esta manera obtendremos MMC clásicas, Booleanas y cuánticas, simples y mejoradas, donde por mejoradas entendemos una más alta eficiencia de almacenamiento y subsecuente reconocimiento de los patrones almacenados.

5.1 Procesos de Ortogonalización Clásicos (POCl)

LOS procesos de ortogonalización clásicos [177-181] juegan un rol clave en muchos métodos iterativos usados en MMC, Procesamiento de Señales e Imágenes, problemas de detección y estimación, procesos estocásticos, filtro de Kalman, Datamining y Bioinformática, entre otras muchas áreas de la Ciencia y la Ingeniería [182-199]. Estos procesos se llevan a cabo con diferentes necesidades y posibilidades de implementación. No obstante, en determinadas aplicaciones se requiere una determinada escala de integración, como es el caso de la tecnología de Muy Alta Escala de Integración (en inglés, Very-Large-Scale Integration: VLSI) por ejemplo, para implementar una versión adaptativa del POCl o de Gram-Schmidt, basada en una estructura tipo escalador [200,201], la cual se discutirá oportunamente. Por otra parte, este tipo de implementaciones son particularmente necesarias en el campo tan demandante de las comunicaciones.

Retornando al problema en cuestión, bajo ciertas condiciones las cuales serán oportunamente señaladas, la versión original de Gram-Schmidt se torna inestable, perdiendo el atributo ortogonalizador, por lo cual es reemplazada en dicho caso por una versión mejorada [177-181]. El problema reside en que todas las versiones mejoradas no poseen inversa, es decir, el atributo por el cual en base al conjunto de elementos de salida del proceso en cuestión se pueda reconstruir al conjunto de los elementos originales de entrada a dicho proceso. La solución al mencionado problema se la puede encontrar en [202,203], en la cual se presenta una versión mejorada (estable) con inversa, la cual resulta en la herramienta ideal en pos a la proyección de estas ideas en el ámbito cuántico. Finalmente, un algoritmo de ortonormalización estable, con inversa, alta y eficientemente paralelizable y de fácil implementación en forma de escalador (necesaria, por ejemplo, en filtros adaptativos) la cual es requerida cuando el anfitrión a bajo nivel es VLSI, FPGA, GPGPU, entre otros, es esencial para filtrado y compresión en Procesamiento de Señales e Imágenes, al tiempo que brindan un parangón para su replicación en el ámbito cuántico y su posible uso para mejorar la performance de las MMC.

5.1.1 Proceso de Ortogonalización de Gram-Schmidt (POGS)

5.1.1.1 Versión Algebraica

Dado un conjunto de vectores linealmente independientes pero no ortonormales, es posible emplear un proceso para transformarlos en un nuevo conjunto pero ortonormal; dicho proceso es diseñado para llevar a cabo la ortogonalización de Gram-Schmidt sobre los vectores de entrada [177-181]. Este proceso de transformación es lineal, manteniendo una correspondencia uno-a-uno

entre los vectores de entrada $\boldsymbol{v}_1$, $\boldsymbol{v}_2$, … , $\boldsymbol{v}_N$, y los vectores ortonormales resultantes $\boldsymbol{u}_1$, $\boldsymbol{u}_2$, … , $\boldsymbol{u}_N$, como se indica a continuación:

$$\{ \boldsymbol{v}_1 , \boldsymbol{v}_2 , \dots , \boldsymbol{v}_N \} \Leftrightarrow \{ \boldsymbol{u}_1 , \boldsymbol{u}_2 , \dots , \boldsymbol{u}_N \}$$

donde $\boldsymbol{u}_1 = \boldsymbol{v}_1$, y el resultante $\boldsymbol{u}_n$ está definido por [202,203]

$$u_n = v_n - \sum_{m=1}^{n-1} r_{m,n}\, u_m , \qquad n = 2, 3, \dots , N \tag{5.1}$$

con

$$r_{m,n} = \frac{u_m^T v_n}{u_m^T u_m} \tag{5.2}$$

donde $(.)^T$ significa transpuesta de (.). La ortogonalidad de los vectores de entrada puede ser también aproximada usando consideraciones estadísticas. Específicamente, si el espacio de dimensión de entrada *M* es grande y los vectores de entrada poseen elementos estadísticamente independientes, ellos estarán próximos a la ortogonalidad con respecto a los demás [202,203]. No obstante, el número de coeficientes $\boldsymbol{r}_{m,n}$ es

$$\boldsymbol{N_r} = \frac{\boldsymbol{N}(\boldsymbol{N}-1)}{2} \tag{5.3}$$

El cual es independiente de la dimensión de los vectores de entrada *M*, siendo *N* el número de vectores.

5.1.1.2 Versión Algorítmica

La versión algorítmica del POGS se basa en las Ecuaciones (5.1) y (5.2), y se muestra en la Fig.5.1, donde la matriz **v** (*M* x *N*) tiene como columnas a los vectores de entrada $\boldsymbol{v}_1$, $\boldsymbol{v}_2$, … , $\boldsymbol{v}_N$.

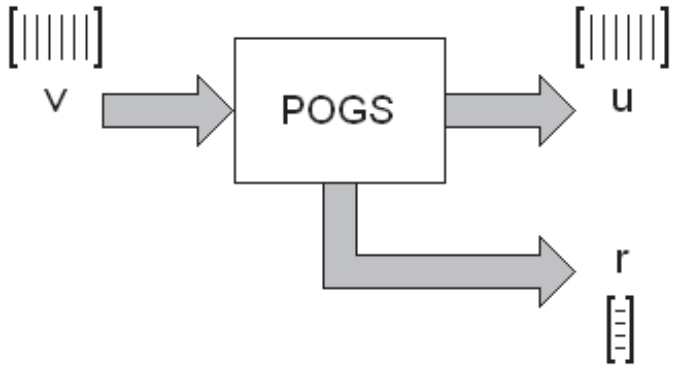


Fig. 5.1 Proceso de Ortogonalización de Gram-Schmidt (POGS).

Similarmente, los vectores ortonormales resultantes $\boldsymbol{u}_1$, $\boldsymbol{u}_2$, … , $\boldsymbol{u}_N$, constituyen las columnas de la matriz **u** (*M* x *N*). Este proceso dará lugar además a un vector **r** (N_r x 1), a partir de las Ecuaciones (5.1), (5.2) y (5.3). El algoritmo como una función de MATLAB® (intérprete de alto nivel) [202,203] es

```
function [u,r] = pogs(v)
u(:,1) = v(:,1);
i = 1;
[M,N] = size(v);
for n = 2:N
  acu = 0;
  for m = 1:n-1
    r(i) = ((v(:,n)')*u(:,m))/((u(:,m)')*u(:,m));
    acu = acu + r(i)*u(:,m);
    i = i+1;
  end
  u(:,n) = v(:,n) - acu;
end
```

5.1.2 Inversa del POGS (IPOGS)

Una versión algorítmica de la IPOGS es indispensable para múltiples aplicaciones [182-199], por lo tanto, a continuación desarrollamos la misma. No obstante, es importante volver a mencionar que su directa es inestable bajo ciertas condiciones [202,203].

5.1.2.1 Versión Algebraica para la IPOGS

Basándonos en consideraciones previas $\boldsymbol{v}_1 = \boldsymbol{u}_1$, y estando los resultantes $\boldsymbol{v}_n$ definidos por

$$v_n = u_n + \sum_{m=1}^{n-1} r_{m,n}\, u_m , \qquad n = 2, 3, \ldots , N \tag{5.4}$$

y considerando que los **u** y los **r** son recibidos en IPOGS, los **r** serán calculadas en POGS.

5.1.2.2 Versión Algorítmica para la IPOGS

Se basa en la Ec.(5.4) y la Fig.5.2.

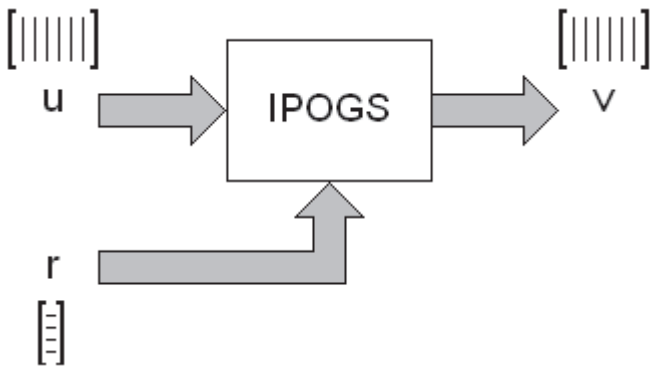


Fig.5.2 Inversa del Proceso de Ortogonalización de Gram-Schmidt (IPOGS).

El algoritmo como una función de MATLAB® sería:

```
function v = ipogs(u,r)
v(:,1) = u(:,1);
i = 1;
[M,N] = size(u);
for n = 2:N
  acu = 0;
  for m = 1:n-1
    acu = acu + r(i)*u(:,m);
    i = i+1;
  end
  v(:,n) = u(:,n) + acu;
end
```

Finalmente, y como oportunamente hemos mencionado, existe una versión modificada de la POGS tradicional [177-181], la cual es estable, y que es conocida como POGS Modificada (POGSMo) pero no posee inversa dado que se trata de un método *in-situ*, i.e., es un algoritmo destructivo con los vectores de entrada [177-181]. Esta es la razón por la cual se hace indispensable un nuevo algoritmo, el cual, siendo estable, tenga inversa.

5.1.3 POGS Mejorado (POGSMe)

5.1.3.1 Versión Algebraica para la Mejora de la Estabilidad

El algoritmo desarrollado es una versión modificada del muy tradicional POGS [177-181] (el cual es bien conocido por sus malas propiedades numéricas, ver [202,203]). Por otro lado, la inestabilidad tiene lugar cuando los denominadores de los elementos del vector **r** se aproximan a **0**. Por lo tanto, con objeto de asegurarnos la estabilidad es necesario aplicar el siguiente procedimiento a los vectores de entrada $\boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_N$:

1. $\boldsymbol{v}_i^T \boldsymbol{v}_i = \min (\boldsymbol{v}_1^T \boldsymbol{v}_1, \boldsymbol{v}_2^T \boldsymbol{v}_2, \ldots., \boldsymbol{v}_N^T \boldsymbol{v}_N)$

2. $(\boldsymbol{v}_i + \mathbf{bias}\, \mathbf{I})^T (\boldsymbol{v}_i + \mathbf{bias}\, \mathbf{I}) = 1$, where $\mathbf{I} = [\, 1\ 1\ \ldots.\ 1\,]^T$ siendo un vector (N x 1)

3. $(N)\, \mathbf{bias}^2 + (2\, \boldsymbol{v}_i^T \mathbf{I})\, \mathbf{bias} + (-1 + \boldsymbol{v}_i^T \boldsymbol{v}_i) = 0$

De tal modo que el mínimo denominador de los elementos del vector **r** sea igual a **1**.

Finalmente,

$$\mathbf{bias} = -(\boldsymbol{v}_i^T \mathbf{I} / N) + [(\boldsymbol{v}_i^T \mathbf{I} / N)^2 - (\boldsymbol{v}_i^T \boldsymbol{v}_i / N) + (1 / N)]^{1/2} \qquad (5.5)$$

5.1.3.2 Versión Algorítmica para la Mejora de la Estabilidad

El algoritmo resultante como una función de MATLAB® resultará en un corrector de la situación del denominador de los coeficientes **r** en función de la estimación del bias, permitiendo que el algoritmo no pierda la capacidad de seguir ortogonalizando a medida que aumenta la cantidad de columnas de la matriz **v**. La corrección final se basa en la Fig.5.3.

```
function [v,bias] = me(v) % "me" significa "mejora de la estabilidad"
[M,N] = size(v);
for n = 1:N
  w(n) = (v(:,n)')*v(:,n);
end
[minw,n] = min(w);
if minw < 1,
  bias = -(sum(v(:,n))/N)-(((sum(v(:,n))/N)^2)-(w(n)/N)+(1/N))^(1/2);
else
  bias = 0;
end
v = v + bias * ones(M,N); % ver Fig.5.3
```

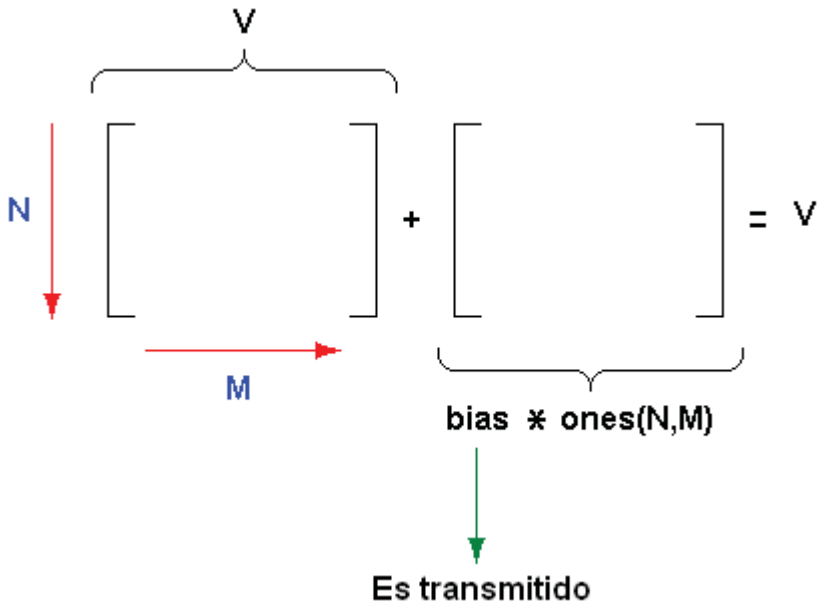


Fig.5.3 Inclusión del ***bias*** para la redefinición de la matriz V.

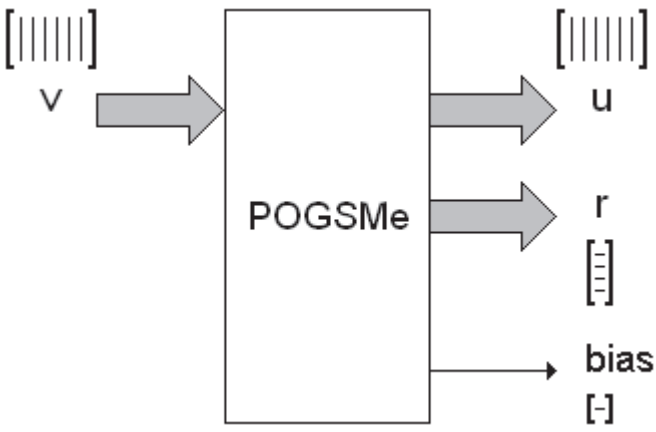


Fig.5.4 POGS mejorado en estabilidad (POGSMe).

5.1.3.3 Versión Algorítmica del POGSMe

La mejora de la estabilidad se basa en el desarrollo previo, el cual se muestra en la Fig.5.3. Por lo tanto, la versión algorítmica final del POGS mejorado en estabilidad, i.e., el POGSMe, puede ser observado en la Fig.5.4.

Los siguientes programas en MATLAB® representan al proceso completo

```
% Entrada de v
[v,bias] = me(v); % "me" significa "mejora de la estabilidad"
[u,r] = pogs(v);
% bias, u y r son transmitidos

% o bien

% Estrada de v
[u,r,bias] = pogsme(v); % POGSMe como una función
% bias, u y r son transmitidos

% donde

function [u,r,bias] = pogsme(v)
[M,N] = size(v);
for n = 1:N
  w(n) = (v(:,n)')*v(:,n);
end
[minw,n] = min(w);
if minw < 1,
  bias = -(sum(v(:,n))/N)-(((sum(v(:,n))/N)^2)-(w(n)/N)+(1/N))^(1/2);
else
  bias = 0;
end
v = v + bias * ones(M,N); % ver Fig.5.3
u(:,1) = v(:,1);
i = 1;
for n = 2:N
  acu = 0;
  for m = 1:n-1
    r(i) = ((v(:,n)')*u(:,m))/((u(:,m)')*u(:,m));
    acu = acu + r(i)*u(:,m);
    i = i+1;
  end
  u(:,n) = v(:,n) - acu;
end
```

En otras palabras, las rutinas anteriores establecen que se las puede definir en forma separada, en las funciones *me()* y *pogs()*, o bien, en una función única *pogsme()* que incluya los atributos de las dos anteriores, es decir,

• mejora de la estabilidad, y

• ortogonalización

Considerando la mejora de la estabilidad, el algoritmo que representa la inversa de la versión estable, es decir, POGSMe, será IPOGSMe y se puede observar en la Fig.5.5. Por otra parte, las si-

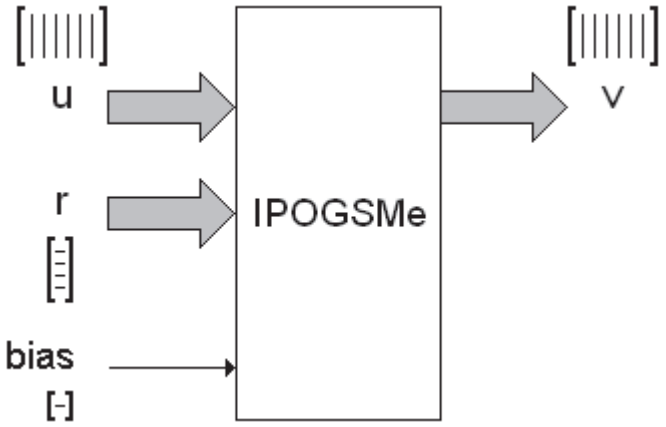


Fig.5.5 Inversa del POGSMe (IPOGSMe).

guientes rutinas representan al proceso mencionado completo en código de MATLAB®

```
% bias, u y r son recibidos
v = ipogs(u,r);
v = ime(v,bias); % "ime" significa "inversa de la mejora de la estabilidad"

% donde

function v = ime(v,bias) % "ime" significa "inversa de la mejora de la estabilidad"
[M,N] = size(v) ;
v = v - bias * ones(M,N); % ver Fig.5.3

% o bien

function v = ipogsme(u,r,bias)
v(:,1) = u(:,1);
i = 1;
[M,N] = size(u);
for n = 2:N
  acu = 0;
  for m = 1:n-1
    acu = acu + r(i)*u(:,m);
    i = i+1;
  end
  v(:,n) = u(:,n) + acu;
end
v = v - bias * ones(M,N); % ver Fig.5.3
```

5.1.4 Prueba de Performance

5.1.4.1 Tasa Dimensional de Entrada-Salida (TDES) sin mejora de la estabilidad

La TDES para el POGS (i.e., POGS-TDES) es:

$$POGS-TDES = \frac{size(\mathbf{u}) + size(\mathbf{r})}{size(\mathbf{v})} \tag{5.6}$$

Reemplazando en orden, las dimensiones correspondientes, en la Ec.(5.6)

$$\boldsymbol{POGS-TDES}=\frac{\boldsymbol{NM}+\boldsymbol{N}_r}{\boldsymbol{NM}} \tag{5.7}$$

Considerando la Ec.(5.3)

$$\boldsymbol{POGS-TDES}=\frac{\boldsymbol{NM}+\dfrac{\boldsymbol{N}(\boldsymbol{N}-1)}{2}}{\boldsymbol{NM}} \tag{5.8}$$

y simplificando

$$\boldsymbol{POGS-TDES}=\frac{2\boldsymbol{M}+\boldsymbol{N}-1}{2\boldsymbol{M}} \tag{5.9}$$

No obstante, mediante consideraciones prácticas se da $M >> N$ [202], entonces POGS-TDES ≅ 1, siendo M la dimensión del espacio de entrada, y N el número de vectores de tal espacio. Similarmente, y por idénticas consideraciones, el IPOGS-TDES ≅ 1, también.

5.1.4.2 Tasa Dimensional de Entrada-Salida (TDES) con mejora de la estabilidad

La TDES para POGSMe (i.e., POGSMe-TDES) es:

$$\boldsymbol{POGSMe-TDES}=\frac{\boldsymbol{size}(\mathbf{u})+\boldsymbol{size}(\mathbf{r})+1}{\boldsymbol{size}(\mathbf{v})} \tag{5.10}$$

Reemplazando en orden, las dimensiones correspondientes en la Ec.(5.10)

$$\boldsymbol{POGSMe-TDES}=\frac{\boldsymbol{NM}+\boldsymbol{N}_r+1}{\boldsymbol{NM}} \tag{5.11}$$

Considerando la Ec.(5.3)

$$\boldsymbol{POGSMe-TDES}=\frac{\boldsymbol{NM}+\dfrac{\boldsymbol{N}(\boldsymbol{N}-1)}{2}+1}{\boldsymbol{NM}} \tag{5.12}$$

y simplificando

$$\boldsymbol{POGSMe-TDES}=\frac{2\boldsymbol{NM}+\boldsymbol{N}^2-\boldsymbol{N}-2}{2\boldsymbol{NM}} \tag{5.13}$$

Por idénticas consideraciones al caso anterior, tendremos también POGSMe-TDES ≅ 1 y además IPOGSMe-TDES ≅ 1.

En otras palabras y como pudimos apreciar, tanto POGS-TDES ≅ 1, como IPOGS-TDES ≅ 1, POGSMe-TDES ≅ 1 y IPOGSMe-TDES ≅ 1, pero qué significa esto? Por lo pronto, esto es excepcionalmente positivo dado que nos esta indicando que la incorporación de un mecanismo de estabilización *in-pectore* no incrementa prácticamente en nada la complejidad computacional.

5.1.5 Rutinas empleadas

Las rutinas empleadas fueron las funciones: *pogsme()* e *ipogsme()*, pero además la función *po()*, la cual es utilizada a efectos de verificar la ortogonalidad generando productos internos de todos contra todos, los cuales en el conjunto de salida deben dar todos ceros.

```
M = input('M = ');
N = input('N = ');
v = rand(M,N);
[u,r,bias] = pogsme(v);
aux = po(u)'; % "po" significa "prueba de ortogonalidad"
v = ipogsme(u,r,bias);

% y

function aux = po(u) % "po" significa "prueba de ortogonalidad"
i = 1;
[M,N] = size(u);
for a = 1:N-1
  for b = a+1:N
    aux(i) = u(:,a)'*u(:,b);
    i = i+1;
  end
end
```

Como hemos mencionado, la segunda rutina prueba la ortogonalización de los vectores causada por el POGSMe, mientras que la primera usa a todas las otras rutinas y de esta manera y mediante las tres juntas se verifica el correcto funcionamiento de la IPOGSMe [202].

5.2 Procesos de Ortogonalización Booleanos (POB)

5.2.1 Versiones Algebraica y Algorítmica del POB

5.2.1.1 Ortogonalidad en el Sentido Booleano

Dado un conjunto de vectores $\mathbf{u}_k = [\ \mathbf{u}_{k1}, \mathbf{u}_{k2}, \ldots, \mathbf{u}_{kp}\ ]^T$ (donde $k = 1, 2, \ldots, q$, y $[.]^T$ significa traspuesta de [.]), *ellos serán ortonormales en un sentido Booleano*, si satisfacen el siguiente par de condiciones:

$$\mathbf{u}_k{}^T \wedge \mathbf{u}_j = 1 \text{ if } k = j \tag{5.14}$$

y

$$\mathbf{u}_k{}^T \wedge \mathbf{u}_j = 0 \text{ if } k \neq j \tag{5.15}$$

donde el término $\mathbf{u}_k{}^T \wedge \mathbf{u}_j$ representa la operación *interna AND* entre los vectores binarios $\mathbf{u}_k$ y $\mathbf{u}_j$, i.e.,

$$\begin{aligned}\mathbf{u}_k{}^T \wedge \mathbf{u}_j &= (\mathbf{u}_{k1} \wedge \mathbf{u}_{j1}) \vee (\mathbf{u}_{k2} \wedge \mathbf{u}_{j2}) \vee \bullet\bullet\bullet \vee (\mathbf{u}_{kp} \wedge \mathbf{u}_{jp}) \\ &= \bigvee_{n=1}^{p} (\mathbf{u}_{kn} \wedge \mathbf{u}_{jn})\end{aligned} \tag{5.16}$$

donde $\vee$ representa a la operación OR [204-206].

5.2.1.2 Independencia Lineal en el Sentido Booleano

Dado un conjunto de vectores binarios $\mathbf{u}_k = [\ \mathbf{u}_{k1}, \mathbf{u}_{k2}, \ldots, \mathbf{u}_{kp}\ ]^T$ (donde $k = 1, 2, \ldots, q$, y $[.]^T$ significa traspuesta de [.]), *ellos serán linealmente independientes en un sentido Booleano*, si satisfacen el siguiente par de condiciones:

1) no existe ningún elemento Booleano α con $\alpha \in [0,1]$ tal que: $\alpha \wedge \boldsymbol{u}_i = \boldsymbol{u}_j \ \forall \ \boldsymbol{i} \neq \boldsymbol{j}$

2) no existen los elementos Booleanos α_h con $\alpha_h \in [0,1]$ tal que: $\boldsymbol{u}_p = \bigvee_{h=h_i}^{h_f} (\alpha_h \wedge \boldsymbol{u}_h)$ $\forall \ \boldsymbol{p} \neq \boldsymbol{h}$

5.2.1.3 Versión Algebraica del POB

Dado un conjunto de vectores binarios de entrada, los cuales sean linealmente independientes (en el sentido Booleano), podemos utilizar un procedimiento Booleano para transformarlos en un conjunto ortonormal (en el sentido Booleano); este procedimiento es diseñado para llevar a cabo una ortonormalización *Booleana* sobre el conjunto de vectores de entrada no ortogonales. Esta forma de transformación se describe a continuación, en la cual se mantiene uno-a-uno la correspondencia entre los vectores binarios de entrada $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_q$ y los vectores binarios resultantes ortonormales de salida $\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_q$, es decir, $\{\ \mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_k\ \} \leftrightarrow \{\ \mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_k\ \}$. Al igual que en el caso clásico, aquí también se pretende la reconstrucción de los $\mathbf{u}$ en base a los $\mathbf{v}$.

De tal modo que comenzamos con $\mathbf{u}_1 = \mathbf{v}_1$, mientras que los restantes $\mathbf{u}_k$ serán definidos como

$$u_{j,k} = v_{j,k} \veebar \left(\bigvee_{i=1}^{k-1} \left(u_{j,i} \wedge v_{j,k} \right) \right) \tag{5.17}$$

con $k = 2, 3, \ldots, q$ y $j = 1, 2, \ldots, p$, y donde $\veebar$ representa a la operación XOR.

5.2.1.4 Versión Algorítmica del POB

En código de MATLAB®:

Primera versión (por elemento, i.e., escalar)

```
u(1,:) = v(1,:);
for j = 1:p
  for k = 2:q
    acu(j,k) = 0;
    for i = 1:k-1
      acu(j,k)= acu(j,k)|(u(j,i)&v(j,k));
    end
    u(j,k) = xor(v(j,k),acu(j,k));
  end
end
```

Segunda versión (por vector, i.e., vectorial)

```
u(:,1) = v(:,1);
for k = 2:q
  acu(:,k) = zeros(p,1);
  for i = 1:k-1
    acu(:,k) = acu(:,k)|(u(:,i)&v(:,k));
  end
  u(:,k) = xor(v(:,k),acu(:,k));
end
```

5.2.1.5 Simulación del Algoritmo POB

Basado en la Fig.5.6, contamos con 6 vectores binarios (columnas) de 7 elementos cada uno, llamado conjunto `v(:,k)`. Aplicando el algoritmo POB a este conjunto, se obtiene otro conjunto de vectores binarios, llamado `u(:,k)` pero cuyos vectores son ortonormales entre ellos. Observar que a la derecha del primer **1** de cada fila de `u(:,k)` solo habrá **0**'s exclusivamente.

La siguiente rutina en código de MATLAB® verifica la ortonormalidad del conjunto `u(:,k)`.

```
i = 0;
for n = 1:q-1
  for m = n+1:q
    i = i+1;
    y(:,i) = u(:,n)&u(:,m);
  end
end
```

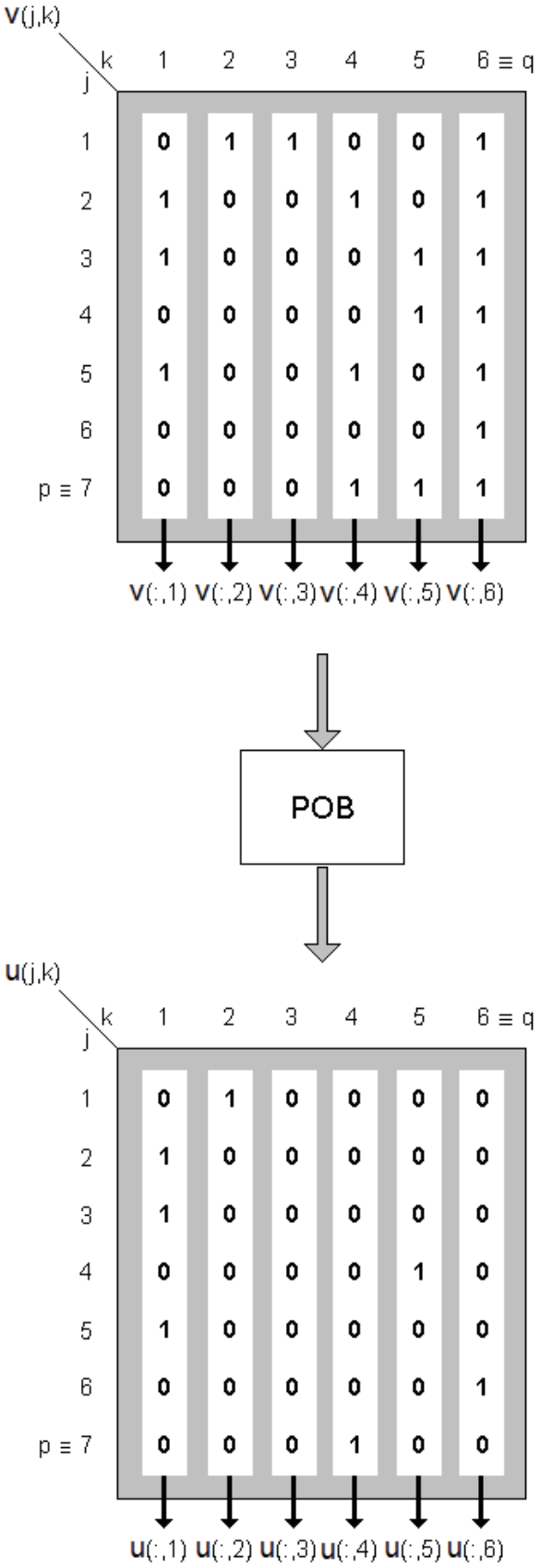


Fig.5.6 Ejemplo de aplicación del Algoritmo POB en sintaxis de MATLAB®.

5.2.2 Arquitectura Escalable como Red Ortogonalizadora Booleana Sistólica (ROBS)

La ROBS fue introducida por Mastriani [204,205] como un Proceso de Ortonormalización Booleano (POB) para convertir una base Booleana no-ortonormal, i.e., un conjunto de vectores binarios no-ortonormales (en un sentido Booleano) a una base Booleana ortonormal, i.e., un conjunto de vectores binarios ortonormales (en el sentido Booleano). El POB tiene un inmenso campo de aplicaciones, e.g.: Esteganografía, Redes de Hopfield, Memorias Matriciales Correlacionadas Booleanas (las cuales se tratarán en el Capítulo 6), procesamiento de imágenes bi-tono, compresión con pérdidas, Biometría (reconocimiento de iris, huella dactilar, rostro), mejora en la detección de bordes y segmentación de imágenes, entre otros. No obstante, es su forma escalonada a modo de red sistólica la que permite su mejor y más eficiente implementación en procesamiento de señales e imágenes. O sea, todas estas aplicaciones requieren ortonormalidad en un sentido Booleano [204,205]. Es importante mencionar que el POB es un algoritmo extremadamente estable, rápido y de fácil programación a bajo nivel y de implementación en FPGA y GPGPU [206].

Por lo tanto, en el ROBS dispondremos no sólo de los ya conocidos vectores linealmente independientes de entrada (en un sentido Booleano) o $v_1, v_2, \ldots, v_N$, y los vectores binarios ortonormales de salida, o $u_1, u_2, \ldots, u_N$, sino también de los vectores binarios residuales y que mencionaremos aquí como $s_1, s_2, \ldots, s_N$, los cuales representan el complemento ortogonal de los vectores de salida o $u_1, u_2, \ldots, u_N$.

5.2.2.1 Versión 1 del ROBS

Como se muestra en la Fig.5.7

$$s_i = v_i \veebar u_i \quad \forall\, i \text{, with } s_1 = 0 \tag{5.18}$$

$$u_i \wedge s_i = 0 \quad \forall\, i \tag{5.19}$$

donde $\veebar$ representa la operación XOR. La Ec.(5.19) representa el Principio Ortogonalidad en un sentido Booleano [204-206].

$$u_i \wedge u_k = 0 \quad \forall\, i \neq k \tag{5.20}$$

$$s_{i,j} \leq v_{i,j} \quad \forall\, i, j \tag{5.21}$$

Algoritmo:

$$u_j = v_j \quad \forall\, j \tag{5.22}$$

$$u_j = u_j \veebar (v_j \wedge u_i), \quad i \in [1, j\text{-}1], \; j \in [1, N] \tag{5.23}$$

5.2.2.2 Versión 2 del ROBS

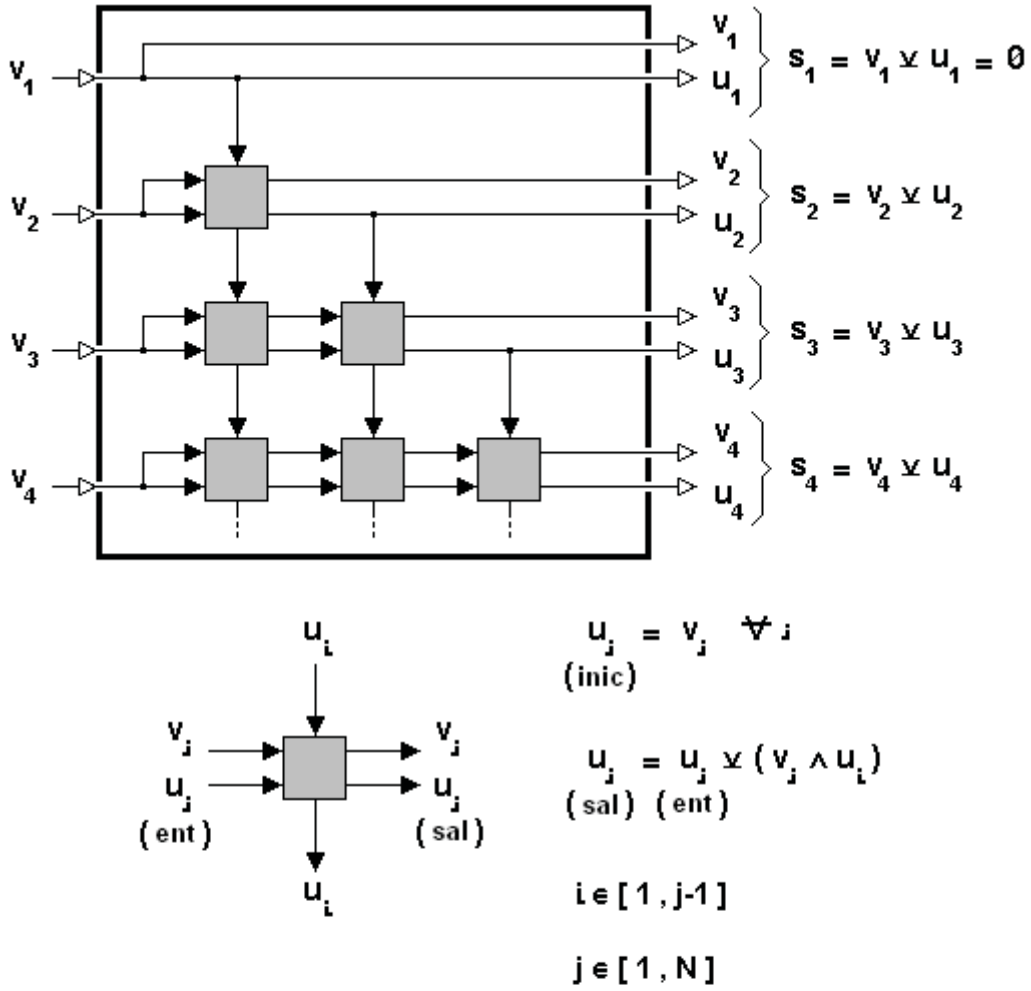
Como se muestra en la Fig.5.8

$$u_i = v_i \veebar s_i \quad \forall\, i \text{, con } u_1 = v_1 \text{, dado que } s_1 = 0 \tag{5.24}$$

Algoritmo:

$s_j = 0 \quad \forall\, j$ (5.25)

$s_j = s_j \vee (v_j \wedge (v_i \veebar s_i)), \quad i \in [1, j\text{-}1], \; j \in [1, N]$ (5.26)



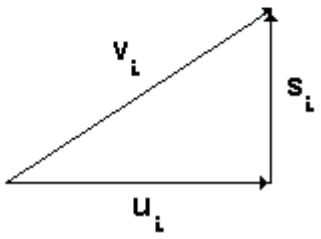
$u_j = v_j \quad \forall\, j$ (inic)

$u_j = u_j \veebar (v_j \wedge u_i)$ (sal) (ent)

$i \in [1, j\text{-}1]$

$j \in [1, N]$

Principio de Ortogonalidad en el Sentido Booleano

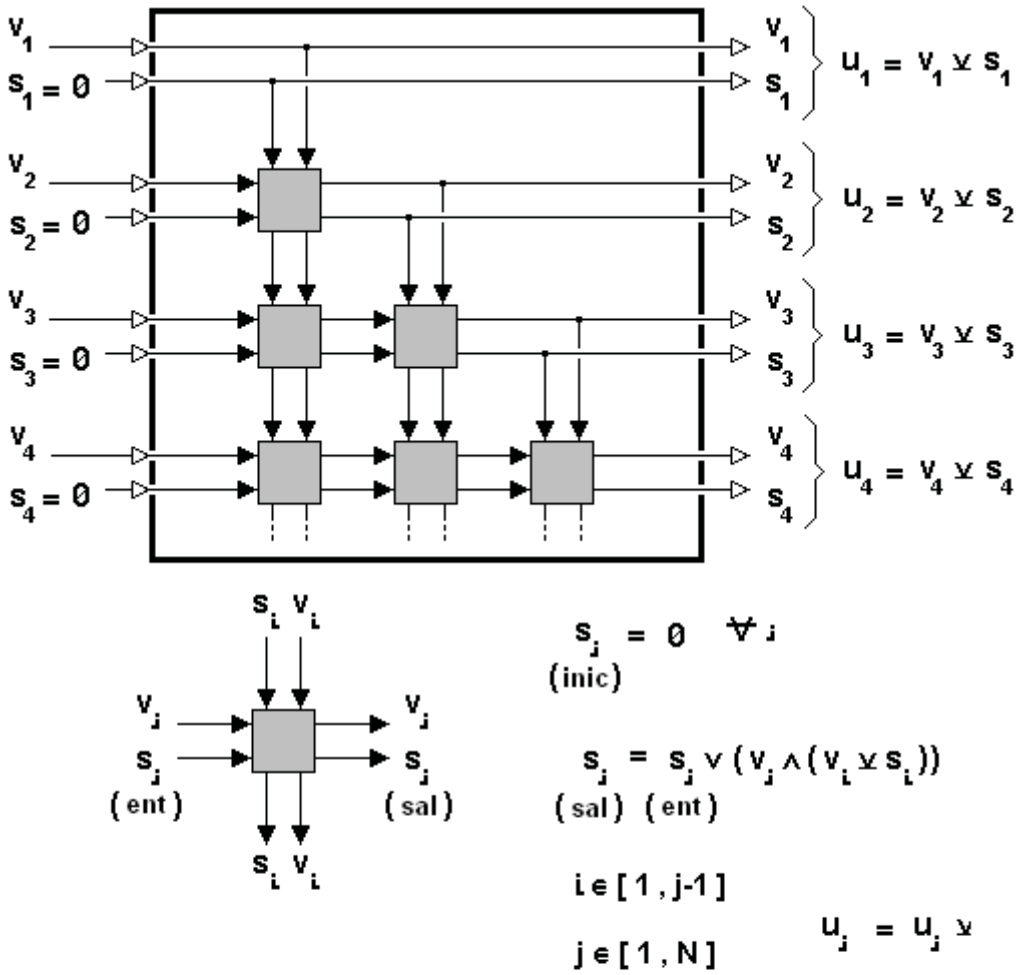


$u_i \wedge s_i = 0 \quad \forall\, i$

$u_i \wedge u_K = 0 \quad \forall\, i \neq K$

$s_i \leq v_i \quad \forall\, i$

Fig.5.7 ROBS, versión 1.



Principio de Ortogonalidad en el Sentido Booleano

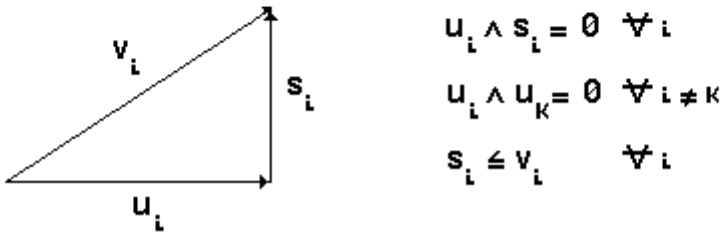


Fig.5.8 ROBS, versión 2.

Aunque esencialmente ambas arquitecturas producen lo mismo, la diferencia fundamental entre las mismas reside en qué se pone a disposición en forma explícita a la salida del ROBS a efectos de recuperar al conjunto de vectores de entrada en un procedimiento inverso. Por lo tanto, mientras en la primera versión (Fig.5.7) disponemos de los vectores **V** y **U**, y el complemento ortogonal de **U**, o sea, **S** se lo obtiene mediante el uso de la compuerta XOR entre **U** y **V**, en la segunda versión, obtenemos en forma directa a la salida los vectores **V** y **S**, mientras que **U** se obtendrá mediante la aplicación también es este caso de la operación XOR [204,205]. Estas diferencias serán críticas a efectos de elegir cuál de los dos modelos exportamos al mundo cuántico.

5.3 Proceso de Ortogonalización Cuántico (POCu)

5.3.1 Prolegómenos al Algebra Cuántica

5.3.1.1 Operaciones Aritméticas Necesarias

Producto de Kronecker entre cubits:

Dados un par de cubits $|\varphi_1\rangle$ y $|\varphi_2\rangle$, de la forma:

$$|\varphi_1\rangle = \alpha_1|0\rangle + \beta_1|1\rangle = \alpha_1\begin{bmatrix}1\\0\end{bmatrix} + \beta_1\begin{bmatrix}0\\1\end{bmatrix} = \begin{bmatrix}\alpha_1\\0\end{bmatrix} + \begin{bmatrix}0\\\beta_1\end{bmatrix} = \begin{bmatrix}\alpha_1\\\beta_1\end{bmatrix}, \quad \alpha_1, \beta_1 \in C$$

$$|\alpha_1|^2 + |\beta_1|^2 = 1, \tag{5.27}$$

y

$$|\varphi_2\rangle = \alpha_2|0\rangle + \beta_2|1\rangle = \alpha_2\begin{bmatrix}1\\0\end{bmatrix} + \beta_2\begin{bmatrix}0\\1\end{bmatrix} = \begin{bmatrix}\alpha_2\\0\end{bmatrix} + \begin{bmatrix}0\\\beta_2\end{bmatrix} = \begin{bmatrix}\alpha_2\\\beta_2\end{bmatrix}, \quad \alpha_2, \beta_2 \in C$$

$$|\alpha_2|^2 + |\beta_2|^2 = 1, \tag{5.28}$$

si efectuamos el producto de Kronecker entre ambos tendremos:

$$|\varphi_1\rangle \otimes |\varphi_2\rangle = \begin{bmatrix}\alpha_1\\\beta_1\end{bmatrix} \otimes \begin{bmatrix}\alpha_2\\\beta_2\end{bmatrix} = \begin{bmatrix}\alpha_1\alpha_2\\\beta_1\alpha_2\\\alpha_1\beta_2\\\beta_1\beta_2\end{bmatrix} = |\varphi_3\rangle$$

$$|\alpha_1\alpha_2|^2 + |\beta_1\alpha_2|^2 + |\alpha_1\beta_2|^2 + |\beta_1\beta_2|^2 = 1 \tag{5.29}$$

donde, como se puede observar fácilmente, $|\varphi_3\rangle$ es un cudit de 4 elementos [71].

Cociente de Kronecker entre cudit y cubit:

Dado un cudit de 4 elementos de la forma:

$$|\varphi_3\rangle = \gamma_{03}|0\rangle + \gamma_{13}|1\rangle + \gamma_{23}|2\rangle + \gamma_{33}|3\rangle = \gamma_{03}\begin{bmatrix}1\\0\\0\\0\end{bmatrix} + \gamma_{13}\begin{bmatrix}0\\1\\0\\0\end{bmatrix} + \gamma_{23}\begin{bmatrix}0\\0\\1\\0\end{bmatrix} + \gamma_{33}\begin{bmatrix}0\\0\\0\\1\end{bmatrix}$$

$$= \begin{bmatrix} \gamma_{03} \\ 0 \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ \gamma_{13} \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ \gamma_{23} \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ \gamma_{33} \end{bmatrix} = \begin{bmatrix} \gamma_{03} \\ \gamma_{13} \\ \gamma_{23} \\ \gamma_{33} \end{bmatrix}, \quad \gamma_{03}, \gamma_{13}, \gamma_{23}, \gamma_{33} \in C \tag{5.30}$$

$$|\gamma_{03}|^2 + |\gamma_{13}|^2 + |\gamma_{23}|^2 + |\gamma_{33}|^2 = 1.$$

y un cubit de la forma:

$$|\varphi_2\rangle = \alpha_2 |0\rangle + \beta_2 |1\rangle = \alpha_2 \begin{bmatrix} 1 \\ 0 \end{bmatrix} + \beta_2 \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} \alpha_2 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ \beta_2 \end{bmatrix} = \begin{bmatrix} \alpha_2 \\ \beta_2 \end{bmatrix}, \quad \alpha_2, \beta_2 \in C \tag{5.31}$$

$$|\alpha_2|^2 + |\beta_2|^2 = 1.$$

El cociente de Kronecker entre el cudit $|\varphi_3\rangle$ y el cubit $|\varphi_2\rangle$ será:

$$|\varphi_3\rangle \otimes / \otimes |\varphi_2\rangle = \begin{bmatrix} \gamma_{03} \\ \gamma_{13} \\ \gamma_{23} \\ \gamma_{33} \end{bmatrix} \otimes / \otimes \begin{bmatrix} \alpha_2 \\ \beta_2 \end{bmatrix} = \begin{bmatrix} \gamma_{03} / \alpha_2 \\ \gamma_{13} / \alpha_2 \end{bmatrix} = \begin{bmatrix} \gamma_{23} / \beta_2 \\ \gamma_{33} / \beta_2 \end{bmatrix} = |\varphi_4\rangle$$

$$|\gamma_{03} / \alpha_2|^2 + |\gamma_{13} / \alpha_2|^2 = 1 \tag{5.32}$$

y

$$|\gamma_{23} / \beta_2|^2 + |\gamma_{33} / \beta_2|^2 = 1$$

Operaciones entre qudits (elemento-a-elemento):

Dados dos cudits $|\Gamma\rangle$ y $|\Phi\rangle$ de las formas

$$|\Gamma\rangle = \gamma_{03} |0\rangle + \gamma_{13} |1\rangle + \gamma_{23} |2\rangle + \gamma_{33} |3\rangle = \gamma_{03} \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} + \gamma_{13} \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} + \gamma_{23} \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} + \gamma_{33} \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

$$= \begin{bmatrix} \gamma_{03} \\ 0 \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ \gamma_{13} \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ \gamma_{23} \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ \gamma_{33} \end{bmatrix} = \begin{bmatrix} \gamma_{03} \\ \gamma_{13} \\ \gamma_{23} \\ \gamma_{33} \end{bmatrix}, \quad \gamma_{03}, \gamma_{13}, \gamma_{23}, \gamma_{33} \in C \tag{5.33}$$

$$|\gamma_{03}|^2+|\gamma_{13}|^2+|\gamma_{23}|^2+|\gamma_{33}|^2=1.$$

y

$$|\Phi\rangle=\phi_{03}|0\rangle+\phi_{13}|1\rangle+\phi_{23}|2\rangle+\phi_{33}|3\rangle=\phi_{03}\begin{bmatrix}1\\0\\0\\0\end{bmatrix}+\phi_{13}\begin{bmatrix}0\\1\\0\\0\end{bmatrix}+\phi_{23}\begin{bmatrix}0\\0\\1\\0\end{bmatrix}+\phi_{33}\begin{bmatrix}0\\0\\0\\1\end{bmatrix}$$

$$=\begin{bmatrix}\phi_{03}\\0\\0\\0\end{bmatrix}+\begin{bmatrix}0\\\phi_{13}\\0\\0\end{bmatrix}+\begin{bmatrix}0\\0\\\phi_{23}\\0\end{bmatrix}+\begin{bmatrix}0\\0\\0\\\phi_{33}\end{bmatrix}=\begin{bmatrix}\phi_{03}\\\phi_{13}\\\phi_{23}\\\phi_{33}\end{bmatrix},\quad \phi_{03},\phi_{13},\phi_{23},\phi_{33}\in C \tag{5.34}$$

$$|\phi_{03}|^2+|\phi_{13}|^2+|\phi_{23}|^2+|\phi_{33}|^2=1.$$

Definimos a partir de ellos las siguientes operaciones aritméticas:

$$|\Gamma\rangle+|\Phi\rangle=\begin{bmatrix}\gamma_{03}\\\gamma_{13}\\\gamma_{23}\\\gamma_{33}\end{bmatrix}+\begin{bmatrix}\phi_{03}\\\phi_{13}\\\phi_{23}\\\phi_{33}\end{bmatrix}=\begin{bmatrix}\gamma_{03}+\phi_{03}\\\gamma_{13}+\phi_{13}\\\gamma_{23}+\phi_{23}\\\gamma_{33}+\phi_{33}\end{bmatrix} \tag{5.35a}$$

$$|\Gamma\rangle-|\Phi\rangle=\begin{bmatrix}\gamma_{03}\\\gamma_{13}\\\gamma_{23}\\\gamma_{33}\end{bmatrix}-\begin{bmatrix}\phi_{03}\\\phi_{13}\\\phi_{23}\\\phi_{33}\end{bmatrix}=\begin{bmatrix}\gamma_{03}-\phi_{03}\\\gamma_{13}-\phi_{13}\\\gamma_{23}-\phi_{23}\\\gamma_{33}-\phi_{33}\end{bmatrix} \tag{5.35b}$$

$$|\Gamma\rangle\times|\Phi\rangle=\begin{bmatrix}\gamma_{03}\\\gamma_{13}\\\gamma_{23}\\\gamma_{33}\end{bmatrix}\times\begin{bmatrix}\phi_{03}\\\phi_{13}\\\phi_{23}\\\phi_{33}\end{bmatrix}=\begin{bmatrix}\gamma_{03}\times\phi_{03}\\\gamma_{13}\times\phi_{13}\\\gamma_{23}\times\phi_{23}\\\gamma_{33}\times\phi_{33}\end{bmatrix} \tag{5.35c}$$

$$|\Gamma\rangle/|\Phi\rangle=\begin{bmatrix}\gamma_{03}\\\gamma_{13}\\\gamma_{23}\\\gamma_{33}\end{bmatrix}/\begin{bmatrix}\phi_{03}\\\phi_{13}\\\phi_{23}\\\phi_{33}\end{bmatrix}=\begin{bmatrix}\gamma_{03}/\phi_{03}\\\gamma_{13}/\phi_{13}\\\gamma_{23}/\phi_{23}\\\gamma_{33}/\phi_{33}\end{bmatrix} \tag{5.35d}$$

Como puede observarse, estas operaciones aritméticas entre cudits son elemento-a-elemento, y se

mantienen intactas entre cudits y arreglos de cudits, por lo cual no tiene sentido desarrollarlo en particular para este caso.

Definición de arreglo vectorial:

Dada el siguiente arreglo matricial (bidimensional) de cubits compuesto de *M* filas y *N* columnas

$$\begin{bmatrix} \begin{bmatrix} \alpha_{11} \\ \beta_{11} \end{bmatrix} & \begin{bmatrix} \alpha_{12} \\ \beta_{12} \end{bmatrix} & \cdots & \begin{bmatrix} \alpha_{1N} \\ \beta_{1N} \end{bmatrix} \\ \begin{bmatrix} \alpha_{21} \\ \beta_{21} \end{bmatrix} & \begin{bmatrix} \alpha_{22} \\ \beta_{22} \end{bmatrix} & \cdots & \begin{bmatrix} \alpha_{2N} \\ \beta_{2N} \end{bmatrix} \\ \vdots & \vdots & \ddots & \vdots \\ \begin{bmatrix} \alpha_{M1} \\ \beta_{M1} \end{bmatrix} & \begin{bmatrix} \alpha_{M2} \\ \beta_{M2} \end{bmatrix} & \cdots & \begin{bmatrix} \alpha_{MN} \\ \beta_{MN} \end{bmatrix} \end{bmatrix}$$

definiremos a cada una de sus columnas y filas como arreglos vectoriales (unidimensional), e.g., consideremos a la *i*-ésima columna

$$\begin{bmatrix} \begin{bmatrix} \alpha_{1i} \\ \beta_{1i} \end{bmatrix} \\ \begin{bmatrix} \alpha_{2i} \\ \beta_{2i} \end{bmatrix} \\ \vdots \\ \begin{bmatrix} \alpha_{Mi} \\ \beta_{Mi} \end{bmatrix} \end{bmatrix}$$

y a la *j*-ésima fila

$$\begin{bmatrix} \begin{bmatrix} \alpha_{j1} \\ \beta_{j1} \end{bmatrix} & \begin{bmatrix} \alpha_{j2} \\ \beta_{j2} \end{bmatrix} & \cdots & \begin{bmatrix} \alpha_{jN} \\ \beta_{jN} \end{bmatrix} \end{bmatrix}$$

las cuales emplearemos en las siguientes definiciones.

Producto interno de Kronecker para arreglos vectoriales de cubits:

Dados dos arreglos vectoriales $|U\rangle$ y $|V\rangle$ definidos como:

$$|U\rangle = \begin{bmatrix} \begin{bmatrix} \alpha_{U1} \\ \beta_{U1} \end{bmatrix} \\ \begin{bmatrix} \alpha_{U2} \\ \beta_{U2} \end{bmatrix} \\ \vdots \\ \begin{bmatrix} \alpha_{UM} \\ \beta_{UM} \end{bmatrix} \end{bmatrix} \quad y \quad |V\rangle = \begin{bmatrix} \begin{bmatrix} \alpha_{V1} \\ \beta_{V1} \end{bmatrix} \\ \begin{bmatrix} \alpha_{V2} \\ \beta_{V2} \end{bmatrix} \\ \vdots \\ \begin{bmatrix} \alpha_{VM} \\ \beta_{VM} \end{bmatrix} \end{bmatrix}, \text{ respectivamente.}$$

Definimos el producto interno entre los mismos como:

$$\langle U|V\rangle = \begin{bmatrix} \begin{bmatrix} \overline{\alpha}_{U1} \\ \overline{\beta}_{U1} \end{bmatrix} & \begin{bmatrix} \overline{\alpha}_{U2} \\ \overline{\beta}_{U2} \end{bmatrix} & \cdots & \begin{bmatrix} \overline{\alpha}_{UM} \\ \overline{\beta}_{UM} \end{bmatrix} \end{bmatrix} \begin{bmatrix} \begin{bmatrix} \alpha_{V1} \\ \beta_{V1} \end{bmatrix} \\ \begin{bmatrix} \alpha_{V2} \\ \beta_{V2} \end{bmatrix} \\ \vdots \\ \begin{bmatrix} \alpha_{VM} \\ \beta_{VM} \end{bmatrix} \end{bmatrix} = \begin{bmatrix} \overline{\alpha}_{U1}\alpha_{V1} + \overline{\alpha}_{U2}\alpha_{V2} + \cdots + \overline{\alpha}_{UM}\alpha_{VM} \\ \overline{\beta}_{U1}\alpha_{V1} + \overline{\beta}_{U2}\alpha_{V2} + \cdots + \overline{\beta}_{UM}\alpha_{VM} \\ \overline{\alpha}_{U1}\beta_{V1} + \overline{\alpha}_{U2}\beta_{V2} + \cdots + \overline{\alpha}_{UM}\beta_{VM} \\ \overline{\beta}_{U1}\beta_{V1} + \overline{\beta}_{U2}\beta_{V2} + \cdots + \overline{\beta}_{UM}\beta_{VM} \end{bmatrix} \tag{5.36}$$

Como pueden observarse, hay dos aspectos relevantes en la ecuación anterior:

1) El guión sobre ciertos elementos significa complejo conjugado.

2) El resultado del producto interno de dos arreglos vectoriales de cubits es un cudit.

Producto externo (tensorial) de Kronecker para arreglos vectoriales de cubits:

Dados los arreglos vectoriales $|U\rangle$ y $|V\rangle$ anteriormente definidos, su producto externo (tensorial) resulta ser:

$$|U\rangle\langle V| = \begin{bmatrix} \begin{bmatrix} \alpha_{U1} \\ \beta_{U1} \end{bmatrix} \\ \begin{bmatrix} \alpha_{U2} \\ \beta_{U2} \end{bmatrix} \\ \vdots \\ \begin{bmatrix} \alpha_{UM} \\ \beta_{UM} \end{bmatrix} \end{bmatrix} \begin{bmatrix} \begin{bmatrix} \overline{\alpha}_{V1} \\ \overline{\beta}_{V1} \end{bmatrix} & \begin{bmatrix} \overline{\alpha}_{V2} \\ \overline{\beta}_{V2} \end{bmatrix} & \cdots & \begin{bmatrix} \overline{\alpha}_{VM} \\ \overline{\beta}_{VM} \end{bmatrix} \end{bmatrix}$$

$$= \begin{bmatrix} \begin{bmatrix} \alpha_{U1}\overline{\alpha}_{V1} \\ \beta_{U1}\overline{\alpha}_{V1} \\ \alpha_{U1}\overline{\beta}_{V1} \\ \beta_{U1}\overline{\beta}_{V1} \end{bmatrix} & \begin{bmatrix} \alpha_{U1}\overline{\alpha}_{V2} \\ \beta_{U1}\overline{\alpha}_{V2} \\ \alpha_{U1}\overline{\beta}_{V2} \\ \beta_{U1}\overline{\beta}_{V2} \end{bmatrix} & \cdots & \begin{bmatrix} \alpha_{U1}\overline{\alpha}_{VM} \\ \beta_{U1}\overline{\alpha}_{VM} \\ \alpha_{U1}\overline{\beta}_{VM} \\ \beta_{U1}\overline{\beta}_{VM} \end{bmatrix} \\ \begin{bmatrix} \alpha_{U2}\overline{\alpha}_{V1} \\ \beta_{U2}\overline{\alpha}_{V1} \\ \alpha_{U2}\overline{\beta}_{V1} \\ \beta_{U2}\overline{\beta}_{V1} \end{bmatrix} & \begin{bmatrix} \alpha_{U2}\overline{\alpha}_{V2} \\ \beta_{U2}\overline{\alpha}_{V2} \\ \alpha_{U2}\overline{\beta}_{V2} \\ \beta_{U2}\overline{\beta}_{V2} \end{bmatrix} & \cdots & \begin{bmatrix} \alpha_{U2}\overline{\alpha}_{VM} \\ \beta_{U2}\overline{\alpha}_{VM} \\ \alpha_{U2}\overline{\beta}_{VM} \\ \beta_{U2}\overline{\beta}_{VM} \end{bmatrix} \\ \vdots & \vdots & \ddots & \vdots \\ \begin{bmatrix} \alpha_{UM}\overline{\alpha}_{V1} \\ \beta_{UM}\overline{\alpha}_{V1} \\ \alpha_{UM}\overline{\beta}_{V1} \\ \beta_{UM}\overline{\beta}_{V1} \end{bmatrix} & \begin{bmatrix} \alpha_{UM}\overline{\alpha}_{V2} \\ \beta_{UM}\overline{\alpha}_{V2} \\ \alpha_{UM}\overline{\beta}_{V2} \\ \beta_{UM}\overline{\beta}_{V2} \end{bmatrix} & \cdots & \begin{bmatrix} \alpha_{UM}\overline{\alpha}_{VM} \\ \beta_{UM}\overline{\alpha}_{VM} \\ \alpha_{UM}\overline{\beta}_{VM} \\ \beta_{UM}\overline{\beta}_{VM} \end{bmatrix} \end{bmatrix} \tag{5.37}$$

Como podemos observar de esta última ecuación, se trata de un arreglo matricial (bidimensional) de cudits.

5.3.1.2 Notación propuesta

A partir de este punto y por simplicidad utilizaremos la notación clásica para el tratamiento de toda operación aritmética entre arreglos vectoriales y matriciales, e.g., si debemos realizar el producto interno entre los arreglos $|U\rangle$ y $|V\rangle$ se expresará como $U^T V$ donde los elementos de U^T serán los complejos conjugados de los elementos de *U,* aunque el resultado será idéntico al de la Ec.(5.36). Similares consideraciones tomaremos para el caso del producto externo (tensorial) entre dichos arreglos vectoriales, i.e., UV^T, aunque el resultado será también idéntico a su contraparte en notación de Dirac, la cual en este caso corresponde a la Ec.(5.37). En otras palabras, de aquí en más y hasta el final de este capítulo y al solo efecto de simplificar las expresiones, ignoraremos cuando por simplicidad convenga a la notación de Dirac.

5.3.2 Red Ortogonalizadora Cuántica Sistólica (ROCS)

Como podemos observar en la Fig.5.9 la arquitectura de la ROCS consiste en una red sistólica escalonada para lo cual el mapeo entrada-salida se sintetiza en la expresión siguiente:

$\{ \boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_N \} \Leftrightarrow \{ \boldsymbol{u}_1, \boldsymbol{u}_2, \ldots, \boldsymbol{u}_N \}, \{ \boldsymbol{r}_{m,n}$, con $n = 2{:}N$ y $m = 1{:}n{-}1 \}, \{ \boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_N \}$

donde "*a = b:c*" significa *"a desde b hasta c*".

Dicho mapeo tiene que ver con una disposición muy propia de este entorno, nos referimos al ámbito cuántico mediante el cual un algoritmo cuántico debe ser reversible [71]. A este respecto y consecuente con lo dicho, podemos identificar como herencia del mundo clásico dos versiones acerca de un ortogonalizador algebraico, a saber: el clásico y el modificado.

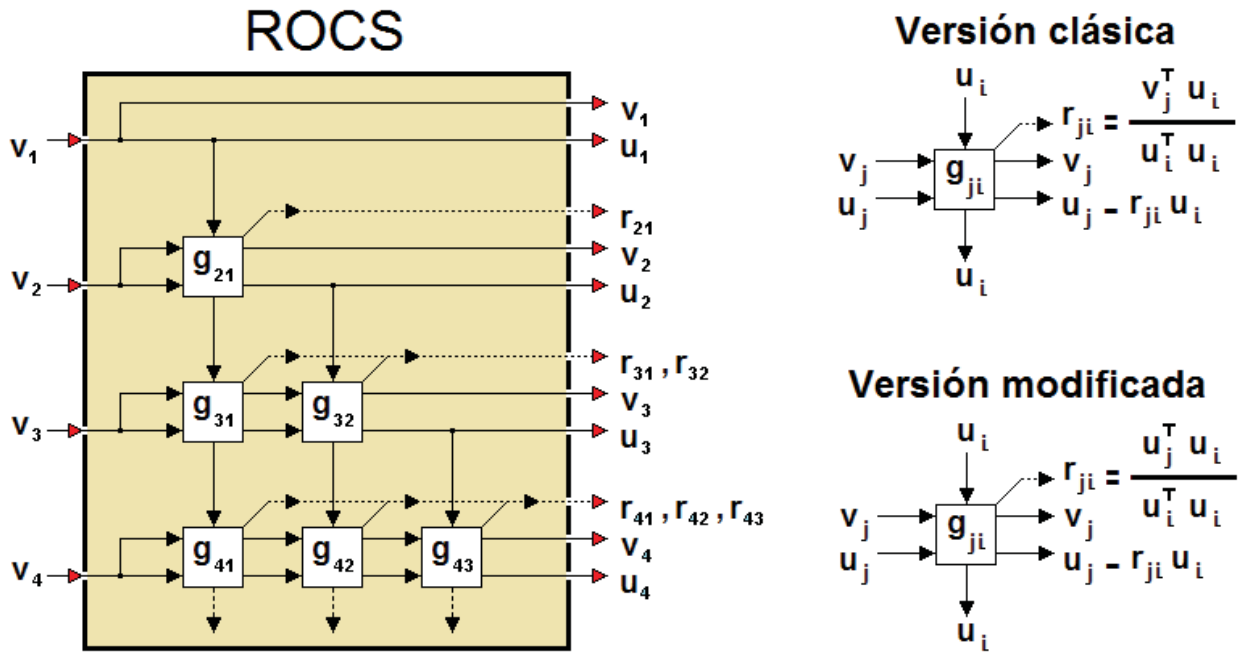


Fig.5.9 Arquitectura ROCS.

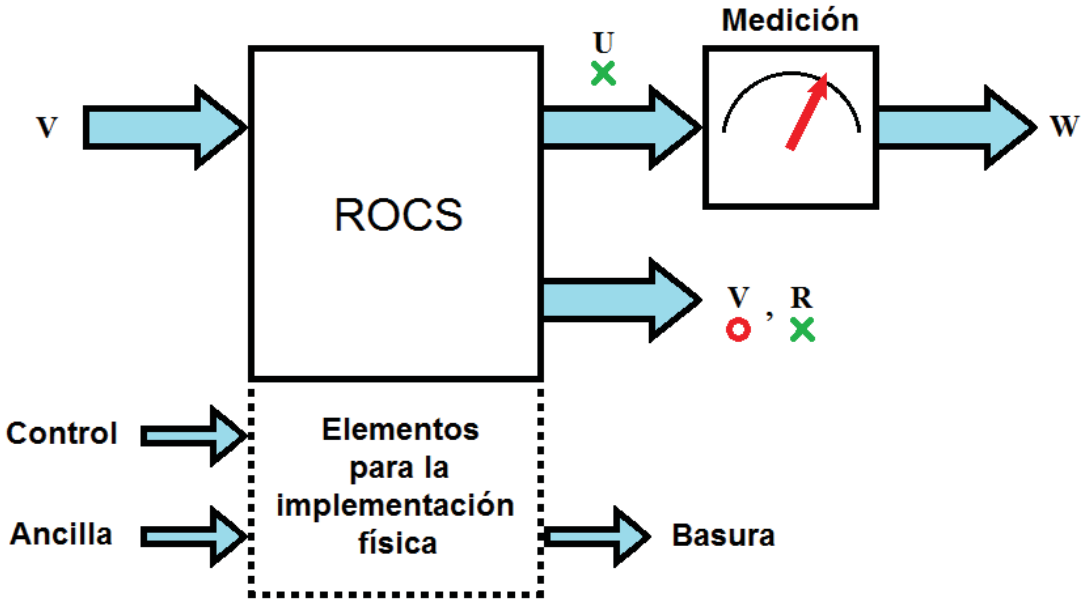


Fig.5.10 Contexto de implementación de ROCS conjuntamente con los elementos de la capa física (en la parte inferior del gráfico) y los detalles de las mediciones de los vectores **U** (en la parte superior derecha del gráfico). Las X verde debajo de **U** y **R** significan que son los elementos mínimos indispensables, tales que a partir de ellos se puede reconstruir la entrada **V** en el caso clásico. El O rojo representa lo propio para el caso modificado. En otras palabras, es solo en la versión clásica en la cual no necesitamos **V** a la salida para el proceso inverso al ROCS.

Ya fue explicado en este mismo capítulo que la versión clásica es más lacónica y cartesiana permitiendo obtener la inversa de una manera más directa en base a los arreglos vectoriales cuánticos de salida ***U*** y al arreglo matricial de los coeficientes de proyección r_{ij}, conocida aquí como ***R***, ver Fig.5.10. No obstante, esta versión (la clásica) se torna inestable bajo ciertas circunstancias

[207-211]. Por otra parte, la versión modificada es más estable, pero a costa de perder el atributo de reversibilidad de una manera directa. Por lo tanto, mientras en la versión clásica podemos obtener la inversa en base al par (***U***, ***R***) lo cual se puede observar en la Fig.5.10 en que estos elementos poseen bajo si una **X**, en cambio, en la versión modificada la única manera de obtener la reversibilidad es trasladando directamente las componentes de ***V*** a la salida, razón por la cual debajo de este elemento en la Fig.5.10 aparece un **O**, al solo efecto de distinguirlo.

Versión Clásica:

Se comienza con $\boldsymbol{u}_1 = \boldsymbol{v}_1$, y el resultante $\boldsymbol{u}_n$ está definido por [207-211]

$$u_n = v_n - \sum_{m=1}^{n-1} r_{m,n}\, u_m, \qquad n = 2, 3, \ldots, N \tag{5.38}$$

con

$$r_{m,n} = \frac{u_m^T v_n}{u_m^T u_m} \tag{5.39}$$

Versión Modificada:

Es idéntica a la anterior, con la salvedad que los coeficientes r_{ij} dependen solo de las salidas, i.e.:

$$r_{m,n} = \frac{u_m^T u_n}{u_m^T u_m} \tag{5.40}$$

En ambas versiones las unidades g_{ij} poseen tres entradas y cuatro salidas

5.3.3 Ortogonalidad antes y después de la medición

Vamos a demostrar con un simple ejemplo la robustez de la ortogonalidad de los métodos mencionados al probar la ortogonalidad también luego del operador medición.

Antes de la Medición:

Comencemos con las tres primeras salidas obtenidas de la versión clásica

$$u_1 = v_1 \tag{5.41}$$

$$u_2 = v_2 - \frac{u_1^T v_2}{u_1^T u_1} u_1 \tag{5.42}$$

$$u_3 = v_3 - \frac{u_1^T v_3}{u_1^T u_1} u_1 - \frac{u_2^T v_3}{u_2^T u_2} u_2 \tag{5.43}$$

Si probamos la ortogonalidad entre ellas, realizando productos internos de a pares entre ellos, obtendremos:

$$u_1^T u_2 = u_1^T v_2 - \frac{u_1^T v_2}{\cancel{u_1^T u_1}} \cancel{u_1^T u_1} = 0 \tag{5.44}$$

$$u_1^T u_3 = u_1^T v_3 - \frac{u_1^T v_3}{\cancel{u_1^T u_1}} \cancel{u_1^T u_1} - \frac{u_2^T v_3}{u_2^T u_2} \cancel{u_1^T u_2} = 0 \tag{5.45}$$

$$u_2^T u_3 = u_2^T v_3 - \frac{u_1^T v_3}{u_1^T u_1} \cancel{u_2^T u_1} - \frac{u_2^T v_3}{\cancel{u_2^T u_2}} \cancel{u_2^T u_2} = 0 \tag{5.46}$$

Donde las tachaduras oblicuas significan que se cancelan mutuamente los mismos en numerador y denominador. Las horizontales significan que son **0** por haberse calculado en un paso previo.

Después de la Medición:

En la Ec.(1.27) de la Sección 1.2.4 del Capítulo 1 definimos

$$|\psi\rangle \xrightarrow{(\lambda_n)} \frac{\boldsymbol{P_n}|\psi\rangle}{\sqrt{\langle\psi|\boldsymbol{P_n}|\psi\rangle}} \tag{5.47}$$

En otras palabras, el colapso de la función de onda $|\psi\rangle$ luego de la medición, siendo λ_n el *n*-ésimo autovalor y $\boldsymbol{P_n}$ es la matrix de autocorrelación de los vectores propios de la matriz asociada al algoritmo en cuestión, en este caso el relativo al ROCS. Llevado a nuestro problema y sintaxis, y considerando la Fig.5.10 tendremos:

$$|\boldsymbol{u}_1\rangle \longrightarrow |\boldsymbol{w}_1\rangle = \frac{\boldsymbol{M}|\boldsymbol{u}_1\rangle}{\sqrt{\langle\boldsymbol{u}_1|\boldsymbol{M}|\boldsymbol{u}_1\rangle}} \tag{5.48}$$

$$|\boldsymbol{u}_2\rangle \longrightarrow |\boldsymbol{w}_2\rangle = \frac{\boldsymbol{M}|\boldsymbol{u}_2\rangle}{\sqrt{\langle\boldsymbol{u}_2|\boldsymbol{M}|\boldsymbol{u}_2\rangle}} \tag{5.49}$$

$$|\boldsymbol{u}_3\rangle \longrightarrow |\boldsymbol{w}_3\rangle = \frac{\boldsymbol{M}|\boldsymbol{u}_3\rangle}{\sqrt{\langle\boldsymbol{u}_3|\boldsymbol{M}|\boldsymbol{u}_3\rangle}} \tag{5.50}$$

Retomando –por simplicidad–la notación clásica, tendríamos:

$$\boldsymbol{u}_1 \longrightarrow \boldsymbol{w}_1 = \frac{\boldsymbol{M}\boldsymbol{u}_1}{\sqrt{\boldsymbol{u}_1^T \boldsymbol{M}\boldsymbol{u}_1}} \tag{5.51}$$

$$\boldsymbol{u}_2 \longrightarrow \boldsymbol{w}_2 = \frac{\boldsymbol{M}\boldsymbol{u}_2}{\sqrt{\boldsymbol{u}_2^T \boldsymbol{M}\boldsymbol{u}_2}} \tag{5.52}$$

$$\boldsymbol{u}_3 \longrightarrow \boldsymbol{w}_3 = \frac{\boldsymbol{M}\boldsymbol{u}_3}{\sqrt{\boldsymbol{u}_3^T \boldsymbol{M}\boldsymbol{u}_3}} \tag{5.53}$$

$$w_1^T w_2 = \left(\frac{\Gamma u_1}{\sqrt{u_1^T \Gamma u_1}} \right)^T \left(\frac{\Gamma u_2}{\sqrt{u_2^T \Gamma u_2}} \right) = \frac{(\Gamma u_1)^T \Gamma u_2}{\sqrt{u_1^T \Gamma u_1} \sqrt{u_2^T \Gamma u_2}}$$

$$= \frac{u_1^T \Gamma^T \Gamma u_2}{\sqrt{u_1^T \Gamma u_1} \sqrt{u_2^T \Gamma u_2}} = \frac{u_1^T I u_2}{\sqrt{u_1^T \Gamma u_1} \sqrt{u_2^T \Gamma u_2}} = \frac{\cancel{u_1^T u_2}}{\sqrt{u_1^T \Gamma u_1} \sqrt{u_2^T \Gamma u_2}} = 0 \qquad \left(de\ la\ Ec.5.44,\ u_1^T u_2 = 0 \right) \tag{5.54}$$

$$w_1^T w_3 = \left(\frac{\Gamma u_1}{\sqrt{u_1^T \Gamma u_1}} \right)^T \left(\frac{\Gamma u_3}{\sqrt{u_3^T \Gamma u_3}} \right) = \frac{(\Gamma u_1)^T \Gamma u_3}{\sqrt{u_1^T \Gamma u_1} \sqrt{u_3^T \Gamma u_3}}$$

$$= \frac{u_1^T \Gamma^T \Gamma u_3}{\sqrt{u_1^T \Gamma u_1} \sqrt{u_3^T \Gamma u_3}} = \frac{u_1^T I u_3}{\sqrt{u_1^T \Gamma u_1} \sqrt{u_3^T \Gamma u_3}} = \frac{\cancel{u_1^T u_3}}{\sqrt{u_1^T \Gamma u_1} \sqrt{u_3^T \Gamma u_3}} = 0 \qquad \left(de\ la\ Ec.5.45,\ u_1^T u_3 = 0 \right) \tag{5.55}$$

$$w_2^T w_3 = \left(\frac{\Gamma u_2}{\sqrt{u_2^T \Gamma u_2}} \right)^T \left(\frac{\Gamma u_3}{\sqrt{u_3^T \Gamma u_3}} \right) = \frac{(\Gamma u_2)^T \Gamma u_3}{\sqrt{u_2^T \Gamma u_2} \sqrt{u_3^T \Gamma u_3}}$$

$$= \frac{u_2^T \Gamma^T \Gamma u_3}{\sqrt{u_2^T \Gamma u_2} \sqrt{u_3^T \Gamma u_3}} = \frac{u_2^T I u_3}{\sqrt{u_2^T \Gamma u_2} \sqrt{u_3^T \Gamma u_3}} = \frac{\cancel{u_2^T u_3}}{\sqrt{u_2^T \Gamma u_2} \sqrt{u_3^T \Gamma u_3}} = 0 \qquad \left(de\ la\ Ec.5.46,\ u_2^T u_3 = 0 \right) \tag{5.56}$$

Siendo ***I*** la matriz identidad y considerando la ortogonalidad de las salidas previas a la medición, podemos observar la robustez del ROCS al permitir la ortogonalidad más allá de las mediciones, lo cual es clave para las aplicaciones relativas al Capítulo 6.

Finalmente, podemos observar que la parte baja de la Fig.5.10 se corresponde con los elementos necesarios para la implementación del ROCS en la capa física, esto es la fábrica de ancilla, los cubits de control y la inevitable generación espuria de basura a la salida [71].

5.3.4 Verificación de la ortogonalidad y su interpretación

Dado el siguiente arreglo matricial de cubits

$$U = \begin{bmatrix} \begin{bmatrix} \alpha_{11} \\ \beta_{11} \end{bmatrix} & \begin{bmatrix} \alpha_{12} \\ \beta_{12} \end{bmatrix} & \begin{bmatrix} \alpha_{13} \\ \beta_{13} \end{bmatrix} \\ \begin{bmatrix} \alpha_{21} \\ \beta_{21} \end{bmatrix} & \begin{bmatrix} \alpha_{22} \\ \beta_{22} \end{bmatrix} & \begin{bmatrix} \alpha_{13} \\ \beta_{13} \end{bmatrix} \\ \begin{bmatrix} \alpha_{31} \\ \beta_{31} \end{bmatrix} & \begin{bmatrix} \alpha_{32} \\ \beta_{32} \end{bmatrix} & \begin{bmatrix} \alpha_{13} \\ \beta_{13} \end{bmatrix} \end{bmatrix} \tag{5.57}$$

el cual consideraremos que resulta de la salida del ROCS, si deseamos verificar en este contexto la ortogonalidad entre sus columnas, por ejemplo, entre la 1º y la 2º de las mismas, dicho producto interno debería dar:

$$\langle u_1 | u_2 \rangle = \begin{vmatrix} \alpha_{11}\alpha_{12} \\ \beta_{11}\alpha_{12} \\ \alpha_{11}\beta_{12} \\ \beta_{11}\beta_{12} \end{vmatrix} + \begin{vmatrix} \alpha_{21}\alpha_{22} \\ \beta_{21}\alpha_{22} \\ \alpha_{21}\beta_{22} \\ \beta_{21}\beta_{22} \end{vmatrix} + \begin{vmatrix} \alpha_{31}\alpha_{32} \\ \beta_{31}\alpha_{32} \\ \alpha_{31}\beta_{32} \\ \beta_{31}\beta_{32} \end{vmatrix} = \begin{vmatrix} \alpha_{11}\alpha_{12} + \alpha_{21}\alpha_{22} + \alpha_{31}\alpha_{32} \\ \beta_{11}\alpha_{12} + \beta_{21}\alpha_{22} + \beta_{31}\alpha_{32} \\ \alpha_{11}\beta_{12} + \alpha_{21}\beta_{22} + \alpha_{31}\beta_{32} \\ \beta_{11}\beta_{12} + \beta_{21}\beta_{22} + \beta_{31}\beta_{32} \end{vmatrix} = \begin{vmatrix} 0 \\ 1 \\ 1 \\ 0 \end{vmatrix} = \frac{|0|}{|1|} = \frac{|1\rangle}{|0\rangle} = |0\rangle \qquad (5.58)$$

Como podemos observar de la Ec.(5.58), al realizar el producto interno de las dos primeras columnas del arreglo matricial ***U*** y dado que cada columna consta de tres cubits cada una, surgen tres cudits, los cuales al ser sumados elemento-a-elemento deben ser igualados en orden a aquello que corresponde en función de hallar finalmente un cero cuántico, i.e., $|0\rangle$, por cuanto se supone que este resultado final responde a la aplicación del producto interno de dos arreglos vectoriales ortogonales entre si. Esto se da mediante un proceso progresivo por agrupaciones, i.e., el cudit resultado se separa en dos cubits, uno arriba del otro, conformando en orden, arriba un uno cuántico $|1\rangle$ y abajo un cero cuántico $|0\rangle$. Los cuales a su vez constituyen otro cero cuántico final $|0\rangle$. Este criterio via agrupaciones parciales es el que debe observarse con objeto de interpretar correctamente los productos internos entre los elementos del conjunto de salida del ROCS.

5.4 Conclusiones del capítulo

En este capítulo presentamos los siguientes aportes:

1. el Proceso de Ortogonalización Booleano (POB) con su inversa,

2. el Proceso de Ortogonalización de Gram-Schmidt Mejorado (POGSMe) con su inversa,

3. el Proceso de Ortogonalización Cuántico (POCu) con su inversa,

4. la Red Ortogonalizadora Booleana Sistólica (ROBS),

5. la Red Ortogonalizadora Cuántica Sistólica (ROCS), y

6. una métrica que llamamos Tasa Dimensional de Entrada-Salida (TDES) la cual fue creada para monitorear el impacto del mejorador de la estabilidad del Proceso Ortogonalizador de Gram-Schmidt en el costo computacional final.

Los Aportes 1 a 5 permitieron la implementación de la primera Memoria Matricial Correlacionada Cuántica (MMCCu) eficiente, es decir, la MMCCu Mejorada (MMCCuMe) del Capítulo 6.

Es importante dejar de relieve que los Aportes 3 y 5 constituyen dos nuevos Algoritmos Cuánticos a nivel de Capa de Aplicación, que viene a completar las herramientas necesarias a efectos de implementar finalmente el Procesamiento Cuántico de Señales e Imágenes, impensables sin un explícito algoritmo de ortogonalización eficiente en dicho ámbito.

Capítulo 6: Memorias Matriciales Correlacionadas Cuántica, Booleana y Clásica

Introducción

En este capítulo se describen en detalle los mecanismos de entrenamiento (training) y funcionamiento -una vez entrenada la misma- (recall) de una Memoria Matricial Correlacionada Cuántica (MMCCu) [212]. Se hará hincapié en la baja en su rendimiento de memorización producto de la necesidad de un algoritmo eficiente de ortogonalización cuántico de los patrones de entrenamiento. Finalmente, estableceremos un parangón entre la versión cuántica deducida aquí, con las versiones clásica y Booleana provenientes de anteriores trabajos. En otras palabras, expondremos los tres tipos de memorias en un orden inverso al del capítulo anterior.

6.1 Memoria Matricial Correlacionada Cuántica (MMCCu)

Esta sección nos introduce en la Memoria Matricial Correlacionada Cuántica (MMCCu) y en su versión Mejorada (MMCCuMe), las cuales son particularmente útiles si se decide trabajar con memorias cuánticas. Con objeto de desarrollar la MMCCuMe recurriremos al Proceso Ortogonalizador Cuántico (POCu) desarrollado en el capítulo anterior el cual emplearemos para convertir un conjunto de arreglos vectoriales cuánticos no ortonormales, a una base de arreglos vectoriales ortonormales entre sí. Este trabajo muestra que es posible mejorar la performance de memorización de la MMCCu gracias al algoritmo POCu. Además, el algoritmo MMCCuMe posee un gran campo de aplicaciones, e.g.: Esteganografía, como un mejorador del funcionamiento de las redes de Hopfield, Procesamiento de Imágenes Médicas, etc. Finalmente, es importante mencionar que la MMCCuMe es extremadamente fácil de implementar en firmware, con especial énfasis en FPGA y GPGPU [212].

6.1.1 Prolegómenos

Antes de comenzar a desarrollar la MMCCu debemos primero establecer claramente el concepto de Memoria Asociativa Cuántica (MACu) la cual además puede ser clasificada como lineal o no lineal, dependiendo del modelo adoptado para sus neuronas. En el caso lineal, las neuronas actúan (en una primera aproximación) como un combinador lineal [212]. A efectos de ser más específicos, dados los arreglos vectoriales cuánticos de datos $|a\rangle$ y $|b\rangle$ los cuales denotan el estímulo (entrada) y la respuesta (salida) de una MACu, respectivamente. En una MACu lineal, la relación Estrada-salida se describe por

$$|b\rangle = M|a\rangle \tag{6.1}$$

donde ***M*** se la conoce como MACu. La matriz ***M*** especifica la red de conectividad de la MACu.

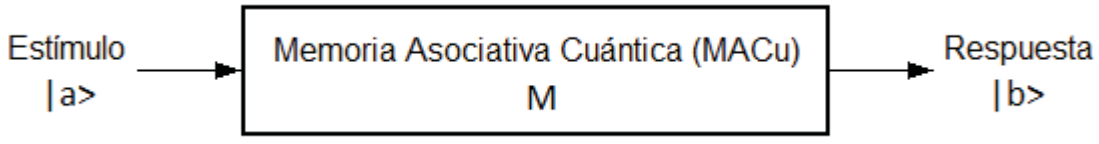


Fig.6.1: Diagrama en bloques de la MACu.

La Fig.6.1 representa un diagrama en bloques que se corresponde con una MACu lineal (MACuL). Por otra parte, en una MACu no lineal (MACuNL), tendremos una relación entrada-salida de la forma

$$|b\rangle = \varphi\left(M;|a\rangle\right)|a\rangle \tag{6.2}$$

donde, en general, φ(.;.) es una función no lineal de la MACu y el arreglo vectorial cuántico de entrada.

Aunque esta forma de matriz (en el caso clásico) fue introducida en los 60 y ha sido estudiada extensamente desde entonces, y el uso de la ortogonalización no es nuevo para mejorar el almacenamiento de dichas redes, ver [212], desafortunadamente, no existía una versión Memoria Matricial Correlacionada Clásica (MMCCl) cuando los patrones de entrada (*key pattern*) $|a_k\rangle$ y los patrones memorizados (*memorized pattern*) $|b_k\rangle \; \forall \, k$ sean cuánticos.

6.1.2 MMCCu

En esta sección desarrollaremos en extenso un caso particular de MACu conocido como MMCCu, sus ventajas y su principal desventaja, la cual nos lleva a la inclusión de un procedimiento basado en la POCu para mejorar su performance de memorización. No obstante, comenzaremos con el concepto de memoria y sus características para luego abordar el concepto de memoria cuántica. Finalmente, abordaremos la principal métrica que permite ser el monitor idóneo del training y recall de una MMCu y que cosiste en la correlación testigo de los patrones.

6.1.2.1 Concepto de Memoria

La discusión acerca de tareas de aprendizaje, particularmente las áreas de asociación de patrones, nos conduce inevitablemente a pensar y reflexionar acerca del concepto de *memoria*. En un contexto neurobiológico, la memoria se refiere a las alteraciones neuronales soportadas relativamente inducidas por la interacción de un organismo con su entorno [213]. Sin tal posibilidad no puede existir el concepto de memoria. Es más, para que una memoria sea útil debe ser accesible al sistema nervioso en función de influenciar futuros comportamientos. No obstante, un patrón de actividad debe ser inicialmente almacenado en memoria a través de un *proceso de aprendizaje* [214-221].

La memoria, y el aprendizaje están intrínsecamente conectados. Cuando es aprendido un patrón de actividad particular, es almacenado en el cerebro donde puede ser recordado más tarde cuando se lo requiera. La memoria puede ser dividida en memoria de "término-corto" y "término-largo", dependiendo del tiempo de retención [213]. La memoria de *término-corto* se refiere a una compilación del conocimiento representando al estado "actual" del entorno. Cualquier discrepancia entre el conocimiento almacenado entre la memoria de término-corto y un estado "nuevo" es usada para adaptar a la memoria de término-corto. La memoria de *término-largo*, por su parte, se refiere al conocimiento almacenado en un tiempo largo o permanentemente.

En esta sección estudiaremos un tipo de memoria asociativa, la cual nos ofrecerá las siguientes características:

- La memoria es distribuida.

- Ambos, i.e., el patrón de estímulo (key) y el patrón de respuesta (almacenado) de una memoria asociativa consisten de vectores de datos.

- La información es almacenada en memoria mediante la creación de un patrón espacial de actividad neuronal a través de un gran número de neuronas.

- La información contenida en un estímulo no determina solamente su ubicación de almacenaje en memoria sino también una dirección para su recuperación.

- Aunque las neuronas no representan unidades de cálculo confiables y de bajo-ruido, la memoria exhibe un alto grado de resistencia al ruido y daño de una clase difusiva.

- Pueden existir interacciones entre los patrones individuales almacenados en la memoria. (De cualquier otra manera la memoria tendría que ser excepcionalmente grande para que pueda acomodar el almacenamiento de un gran número de patrones en perfecto aislamiento el uno del otro). Por lo tanto, estas son las posibilidades de la memoria de cometer *errores* durante el proceso de recuperación.

En una *memoria distribuida*, la cuestión básica de interés son las actividades simultáneas o casi simultáneas de muchas neuronas diferentes, las cuales son el resultado de un estímulo externo o interno. Las actividades neuronales forman un patrón especial en el interior de la memoria que contiene información acerca del estímulo. Por consiguiente, se dice que la memoria lleva a cabo un mapeo distribuido que transforma un patrón de actividad en el espacio de entrada en otro patrón de actividad en el espacio de salida. Podemos ilustrar algunas propiedades importantes de una memoria distribuida mapeando al considerar una red neuronal idealizada que consiste de dos capas de neuronas [222-227].

6.1.2.2 Memoria Cuántica

En el análisis matemático siguiente, asumimos que las Redes Neuronales Cuánticas (RNCu) son lineales. La implicancia de esta suposición es que cada neurona actúa como un combinador lineal. Para proceder con el análisis suponemos que un patrón de actividad $|x_k\rangle$ ocurre en la capa de entrada de la red y que un patrón de actividad $|y_k\rangle$ ocurre simultáneamente en la capa de salida. La cuestión que queremos considerar aquí es la de aprender la forma de asociación entre los patrones $|x_k\rangle$ y $|y_k\rangle$. Los patrones $|x_k\rangle$ e $|y_k\rangle$ están representados por arreglos vectoriales, escritos en su forma expandida como:

$$|x_k\rangle = \left[|x_{k1}\rangle, |x_{k2}\rangle, \ldots, |x_{km}\rangle\right]^T \tag{6.3}$$

y

$$|y_k\rangle = \left[|y_{k1}\rangle, |y_{k2}\rangle, \ldots, |y_{km}\rangle\right]^T \tag{6.4}$$

De aquí en adelante nos referiremos a *m* como la *dimensionalidad de la red* [212]. Los elementos de ambos $|x_k\rangle$ y $|y_k\rangle$ pueden asumirse como valores positivos y negativos. No obstante, siempre dentro de los valores admitidos dentro de la esfera de Bloch para cubits puros [71].

En el caso de esta red, la cual ya hemos asumido como lineal, la asociación del arreglo vectorial de estímulo (key) $|x_k\rangle$ con el arreglo vectorial memorizado $|y_k\rangle$ puede ser descripta en la forma de un arreglo matricial cuántico como:

$$|y_k\rangle = W(k)|x_k\rangle, \quad k = 1, 2, \ldots, q \tag{6.5}$$

Para desarrollar una descripción detallada de la Ec.(6.5), tenemos

$$|y_{ki}\rangle = \sum_{j=1}^{m} w_{ij}(k)|x_{kj}\rangle, \quad i = 1, 2, \ldots, m \tag{6.6}$$

donde $w_{ij}(k), j = 1, 2, \ldots, m$ es el peso sináptico de la neurona *i* correspondiente al *k*-ésimo par de patrones asociados. Usando la notación matricial, podemos expresar y_{ki} en la forma equivalente

$$|y_{ki}\rangle = [w_{i1}(k), w_{i2}(k), \ldots, w_{im}(k)] \begin{bmatrix} |x_{k1}\rangle \\ |x_{k2}\rangle \\ \vdots \\ |x_{km}\rangle \end{bmatrix}, i = 1, 2, \ldots, m \tag{6.7}$$

Podemos definir una Memoria Matricial Cuántica (MMCu) dimensión de *m*-por-*m* que describa la sumatoria de las matrices de pesos para el conjunto entero de patrones de asociaciones, como sigue:

$$M = \sum_{k=1}^{q} w(k) \tag{6.8}$$

La definición de la MMCu dada en la Ec.(6.7) puede ser reestructurada en la forma de una recursión, como se muestra a continuación

$$M_k = M_{k-1} + W(k), \quad k = 1, 2, \ldots, q \tag{6.9}$$

donde el valor inicial M_0 es cero (i.e., los pesos sinápticos en la memoria son todos inicialmente cero), y el valor final M_q es idénticamente igual a *M* como se define en la Ec.(6.8). De acuerdo a la formula recursiva de la Ec.(6.9), el término M_{k-1} es el valor anterior de la MMCu resultante de los (*k*-1) patrones de asociación, y M_k es el valor actualizado a la luz del incremento $W(k)$ producido por la *k*-ésima asociación. Notemos, no obstante, que cuando $W(k)$ es adicionado a M_{k-1}, el incremento $W(k)$ pierde su identidad distintiva entre la mezcla de contribuciones de la forma M_k. A pesar de la mezcla sináptica de las diferentes asociaciones, la información acerca de los estímulos no se puede haber perdido, como se demuestra en la secuela. Notemos además, que conforme el número *q* de patrones almacenados se incremente, la influencia de un nuevo patrón sobre la memoria -como un todo- se reduce progresivamente.

6.1.2.3 Entrenamiento de la MMCCu (training)

Supongamos que la MACu ha aprendido la matriz de memoria *M* a través de las asociaciones de los patrones de entrada (key) y los memorizados descritas por $|x_k\rangle \to |y_k\rangle$, donde $k = 1, 2, \ldots, q$. Postulamos a $\hat{M}$, el cual representa a un estimado de la matriz de memoria *M* en función de estos patrones [212]:

$$\hat{M} = \sum_{k=1}^{q} |y_k\rangle\langle x_k| \tag{6.10}$$

El término $|y_k\rangle\langle x_k|$ representa el *producto externo* entre el patrón de entrada (key) $|x_k\rangle$ y el patrón memorizado $|y_k\rangle$. Este producto externo es un *estimado* de la matriz de pesos *W*(k) que mapea el patrón de salida $|y_k\rangle$ sobre el patrón de entrada $|x_k\rangle$. Dado que los patrones $|x_k\rangle$ e $|y_k\rangle$ son ambos arreglos vectoriales cuánticos de *m*-por-1, desde luego y por ende, se deduce que el producto externo $|y_k\rangle\langle x_k|$, y por lo tanto, el estimado $\hat{M}$, son arreglos matriciales cuánticos de *m*-por-*m*.

Por otra parte, la Ec.(6.10) puede ser reformulada en una forma equivalente más conveniente

$$\begin{aligned} \hat{M} &= \begin{bmatrix} \langle y_1| \\ \langle y_2| \\ \vdots \\ \langle y_q| \end{bmatrix} \left[|x_1\rangle, |x_2\rangle, \ldots, |x_q\rangle \right] \\ &= |Y\rangle\langle X| \end{aligned} \tag{6.11}$$

donde

$$\langle X| = \left[|x_1\rangle, |x_2\rangle, \ldots, |x_q\rangle \right] \tag{6.12}$$

y

$$\langle Y| = \left[\langle y_1|, \langle y_2|, \ldots, \langle y_q| \right] \tag{6.13}$$

La matriz ***X*** tiene una dimensión de *m*-por-*q* y se encuentra enteramente compuesta del conjunto de patrones de entrada (key) usados en el proceso de aprendizaje; por lo cual recibe el nombre de matriz de aprendizaje (*key matrix*). La matriz ***Y*** también tiene una dimensión de *m*-por-*q* y está compuesta del correspondiente conjunto de patrones memorizados, razón por la cual es llamada *matriz memorizada*.

La ecuación (6.11) puede ser reestructurada en una forma recursiva la cual facilitará notablemente la implementación computacional de la misma, a saber:

$$\hat{M}_k = \hat{M}_{k-1} + |y_k\rangle\langle x_k|, \qquad k = 1, 2, \ldots, q \tag{6.14}$$

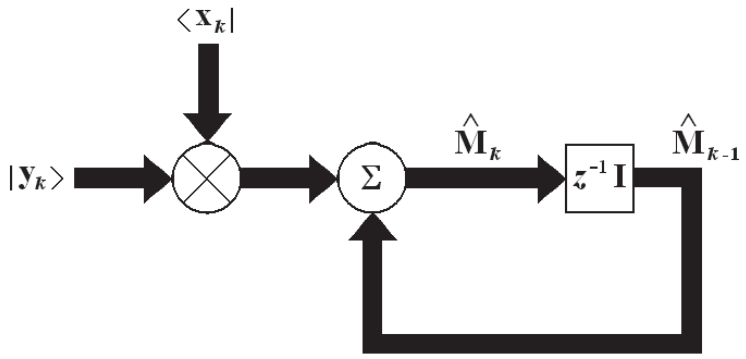


Fig.6.2: Representación gráfica del flujo de señal de la Ec(5.X).

Un gráfico de representación del flujo de señales de esta recursión en presentado en la Fig.(6.2). De acuerdo a este gráfico de flujo de señales y a la fórmula recursiva de la Ec.(6.14), la matriz $\hat{M}_{k-1}$ representa a un estimado previo de la MMCu; y $\hat{M}_k$ representa su valor actualizado a la luz de la nueva asociación llevada a cabo por la memoria sobre los patrones $|x_k\rangle$ e $|y_k\rangle$. Comparando la recursión de la Ec.(6.14) con aquella de la Ec.(6.9), podemos ver que el producto externo $|y_k\rangle\langle x_k|$ representa un estimado de la matriz de pesos W(k) correspondiente a la k-ésima asociación de los patrones de entrada (key) $|x_k\rangle$ y los memorizados $|y_k\rangle$. La Fig.6.3 representa el mecanismo por el cual el patrón memorizado (estado de salida) se reinyecta en la entrada como patrón key (estado de entrada) estableciendo el mecanismo necesario para la posterior recuperación.

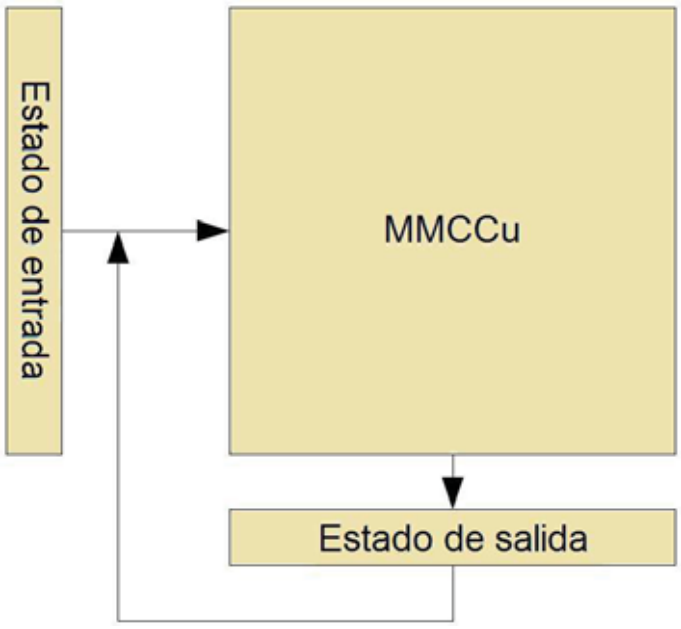


Fig.6.3: Arquitectura de entrenamiento.

6.1.2.4 Funcionamiento una vez entrenada de la MMCCu (recall)

El problema principal planteado por el uso de una MACu es la dirección y recuperación de los

patrones almacenados en la memoria. Al solo efecto de explicar un aspecto de este problema, si $\hat{M}$ representa a la matriz de memoria de una MACu, la cual ha sido completamente aprendida a través de su exposición a *q* patrones de asociaciones de acuerdo con la Ec.(6.10). Si un patrón de entrada (key) $|x_j\rangle$ es recogido de una manera aleatoria y reaplicado como *estímulo* a la memoria, produciendo la *respuesta* (Fig.6.3)

$$|y\rangle = \hat{M}\,|x_j\rangle \tag{6.15}$$

Lo cual queda inequívocamente establecido desde el punto de vista gráfico en la arquitectura de recuperación de la Fig.6.4. Como podemos observar y a diferencia de la arquitectura de entrenamiento de la Fig.6.3 la cual es a lazo cerrado, esta arquitectura será a lazo abierto, sin la reinyección del estado de salida en la entrada.

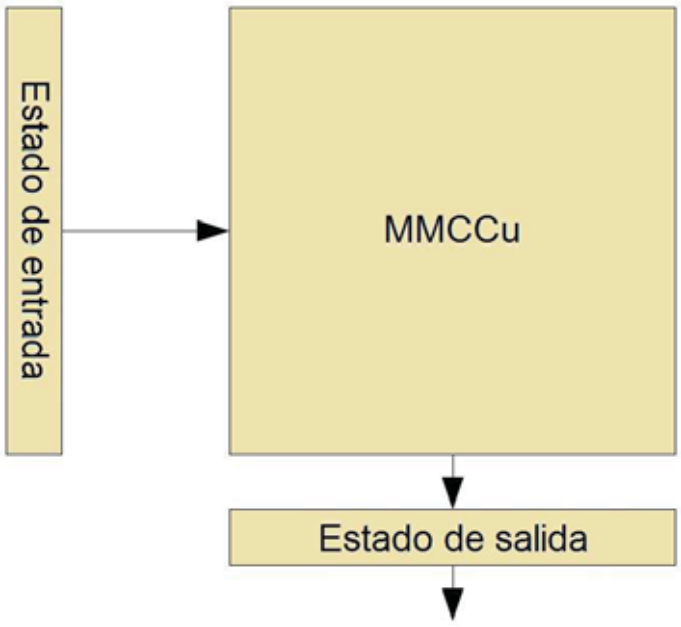


Fig.6.4: Arquitectura de recuperación.

Si ahora sustituimos la Ec.(6.10) en la (6.15), obtendremos

$$\begin{aligned} |y\rangle &= \sum_{k=1}^{m} |y_k\rangle\langle x_k|\,|x_j\rangle \\ &= \sum_{k=1}^{m} \langle x_k|\,|x_j\rangle\,|y_k\rangle \end{aligned} \tag{6.16}$$

donde, en la segunda línea, podemos reconocer que $\langle x_k|\,|x_j\rangle$ es un escalar igual al *producto interno* de los arreglos vectoriales cuánticos (key) $|x_k\rangle$ y $|x_j\rangle$. Podemos reescribir la Ec.(6.16) como

$$|y\rangle = \left(\langle x_j|\,|x_j\rangle\right)|y_j\rangle + \sum_{\substack{k=1 \\ k\neq j}}^{m} \left(\langle x_k|\,|x_j\rangle\right)|y_k\rangle \tag{6.17}$$

Si cada uno de los patrones key $|x_1\rangle,|x_2\rangle,\ldots,|x_q\rangle$ es normalizado a efectos de tener energía unitaria; es decir,

$$\begin{aligned} E_k &= \sum_{l=1}^{m} |x_{kl}\rangle |x_{kl}\rangle \\ &= \langle x_k | x_k \rangle \\ &= 1, \qquad k = 1,2,\ldots,q \end{aligned} \tag{6.18}$$

De acuerdo a esto, podemos simplificar la respuesta de la memoria para el estímulo (patrón key) $|x_j\rangle$ de la forma

$$|y\rangle = |y_j\rangle + |v_j\rangle \tag{6.19}$$

donde

$$|v_j\rangle = \sum_{\substack{k=1 \\ k\neq j}}^{m} \langle x_k | x_j \rangle |y_k\rangle \tag{6.20}$$

El primer término del lado derecho de la Ec.(6.19) representa la respuesta "deseada" $|y_j\rangle$; y puede, por lo tanto, ser vista como la "señal" componente de la respuesta actual $|y\rangle$. El segundo término $|v_j\rangle$ es un "arreglo vectorial de ruido cuántico" que surge debido a la *interferencia* entre el arreglo vectorial cuántico (key) $|x_j\rangle$ y todos los otros arreglos vectoriales cuánticos (key) almacenados en memoria. El arreglo vectorial de ruido cuántico $|v_j\rangle$ es el responsable de la generación de errores durante la recuperación.

En el contexto de un espacio lineal de señales, podremos definir el *coseno del ángulo* entre un par de arreglos vectoriales cuánticos $|x_j\rangle$ y $|x_k\rangle$ como el producto interno de $|x_j\rangle$ y $|x_k\rangle$ dividido por el producto de sus normas Hermíticas individuales o longitudes, en otras palabras,

$$\cos\left(|x_k\rangle,|x_j\rangle\right) = \frac{\langle x_k | x_j \rangle}{\||x_k\rangle\|\,\||x_j\rangle\|} \tag{6.21}$$

El símbolo $\||x_k\rangle\|$ significa la norma Hermítica del arreglo vectorial $|x_k\rangle$, definida como la raíz cuadrada de la energía de $|x_k\rangle$:

$$\begin{aligned} \||x_k\rangle\| &= \sqrt{\langle x_k | x_k \rangle} \\ &= \sqrt{E_k} \end{aligned} \tag{6.22}$$

Retornando a la línea de razonamiento original, notemos que los arreglos vectoriales cuánticos (key) están normalizados a efectos de tener energía unitaria, de acuerdo a la Eq.(6.18). Por lo tanto, podemos reducir la definición de la Ec.(6.21) a

$$\cos\left(\left|x_k\right\rangle,\left|x_j\right\rangle\right)=\left\langle x_k\left|x_j\right.\right\rangle \tag{6.23}$$

Podemos entonces redefinir el arreglo vectorial de ruido cuántico de la Ec.(6.20) como

$$\left|v_j\right\rangle=\sum_{\substack{k=1\\k\neq j}}^{m}\cos\left(\left|x_k\right\rangle,\left|x_j\right\rangle\right)\left|y_k\right\rangle \tag{6.24}$$

Ahora podemos observar que si los keys son *ortogonales* (i.e., perpendiculares entre sí en un sentido Hermítico), entonces

$$\cos\left(\left|x_k\right\rangle,\left|x_j\right\rangle\right)=\left|0\right\rangle,\quad k\neq j \tag{6.25}$$

y por lo tanto, el arreglo vectorial de ruido cuántico $\left|v_j\right\rangle$ es idénticamente cero. En tal caso, la respuesta $\left|y\right\rangle$ es igual a $\left|y_j\right\rangle$. La *memoria asocia perfectamente* si los keys de un conjunto *ortonormal*; i.e., si ellos satisfacen el siguiente par de condiciones:

$$\left\langle x_k\left|x_j\right.\right\rangle=\begin{cases}\left|1\right\rangle, & k=j\\ \left|0\right\rangle, & k\neq j\end{cases} \tag{6.26}$$

Supongamos ahora que los keys forman un conjunto ortonormal, según lo establecido en la Ec.(6.26). Cuál es entonces el límite en la *capacidad de almacenamiento* de la MACu? Dicho de otra manera, cuál es el número más grande de patrones que pueden ser realmente almacenados? La respuesta a esta pregunta fundamental reside en el rango del arreglo matricial cuántico de memoria $\hat{M}$. En el mundo clásico, el rango de una matriz esta definido como el número de columnas independientes (o filas) de la matriz. En el capítulo anterior, vimos que existe una contraparte a esto en el Algebra Cuántica. No obstante, lo que esto quiere decir es que, si *r* es el rango de una matriz rectangular de dimensiones *l*-by-*m*, entonces tendremos $r\leq\min(l,m)$. En el caso de una memoria de correlación cuántica, la memoria matricial cuántica $\hat{M}$ es un arreglo matricial cuántico de *m*-por-*m* cubits, donde *m* es la dimensionalidad del espacio de entrada. Por lo tanto, el rango de la memoria matricial cuántica *M* esta limitada por la dimensionalidad *m*. Entonces, podemos establecer formalmente que el número de patrones que pueden ser realmente almacenados en una MMCCu nunca puede exceder la dimensionalidad del espacio de entrada.

6.1.2.5 MMCCu Mejorada (MMCCuMe)

De acuerdo a lo explicado en la sección anterior, la mejora en la memoria consiste en remover el arreglo vectorial de ruido cuántico $\left|v_j\right\rangle$, que surge de la descomposición ortogonal del arreglo vectorial cuántico de entrada $\left|x_k\right\rangle$ o key, el cual es conducido a través del procedimiento explicado en el capítulo anterior consistente en el POCu, mediante el cual dado un conjunto de entrada $\left|x_1\right\rangle,\ldots,\left|x_d\right\rangle$ en el espacio vectorial en *Z* en el cual podemos definir el producto interno necesario para esta implementación. El POCu producirá un conjunto base ortonormal $\left|z_1\right\rangle,\ldots,\left|z_d\right\rangle$ para el espacio vectorial *Z*. Este procedimiento nos permitirá acceder automáticamente al MMCCuMe a

partir de la combinación en orden del POCu seguido de la MMCCu, ver Fig.6.3. No es dificil verificar que los arreglos vectoriales cuánticos $|z_1\rangle,\ldots,|z_d\rangle$ forman un conjunto ortonormal el cual es también un base para *Z*. Entonces, cualquier espacio vectorial dimensionalmente finito de dimensión *d* tiene una base ortonormal, $|z_1\rangle,\ldots,|z_d\rangle$ [23-30]. Ahora, los arreglos vectoriales cuánticos de entrada a la MMCCu son los **z** en lugar de los **x**. Ambos bloques (POCu y MMCCu) constituyen la MMCCu Mejorada (MMCCuMe), ver la Fig.6.3.

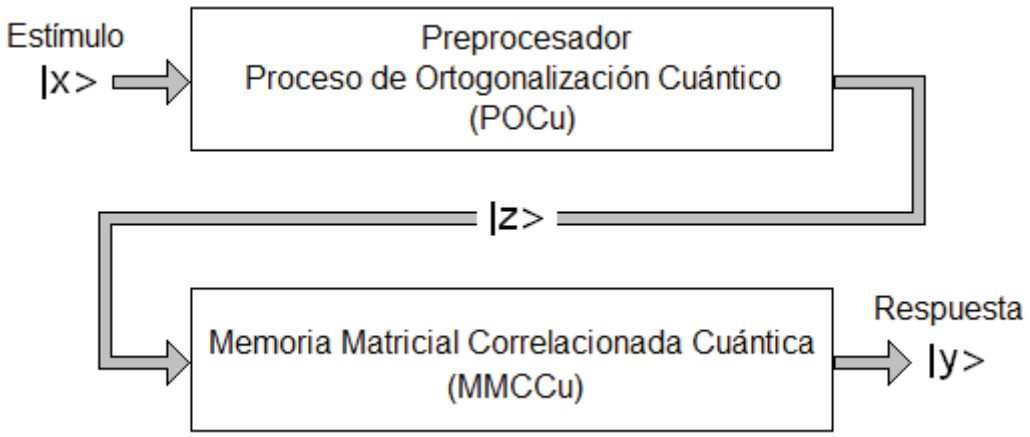


Fig.6.5: Ambos bloques: POCu y MMCCu constituyen la MMCCuMe.

6.1.2.6 Correlación testigo de los patrones de training y recall

La métrica más conspicua a efectos de evaluar la performance de almacenamiento y recuperación de este tipo de memorias surge del siguiente análisis.

Dados los siguientes arreglos matriciales de patrones, **X** (en azul) e **Y** (en verde) podemos definir entonces la matriz de correlación cruzada $\mathbf{R_{XY}}$ la cual expresaremos simbólicamente como:

$$\mathbf{R_{XY}} = \mathbf{X} \times \mathbf{Y} \tag{6.27}$$

la cual podemos observar ubicuamente en detalle en la Fig.6.6.

Partiendo de la correlación cruzada:

$$C_{X,Y} = \frac{1}{N \times N} \sum_{\substack{i=1 \\ j=1}}^{N,N} |X_i\rangle\langle Y_j| \qquad \forall\ i, j \tag{6.28}$$

desarrollamos ahora las autocorrelaciones individuales:

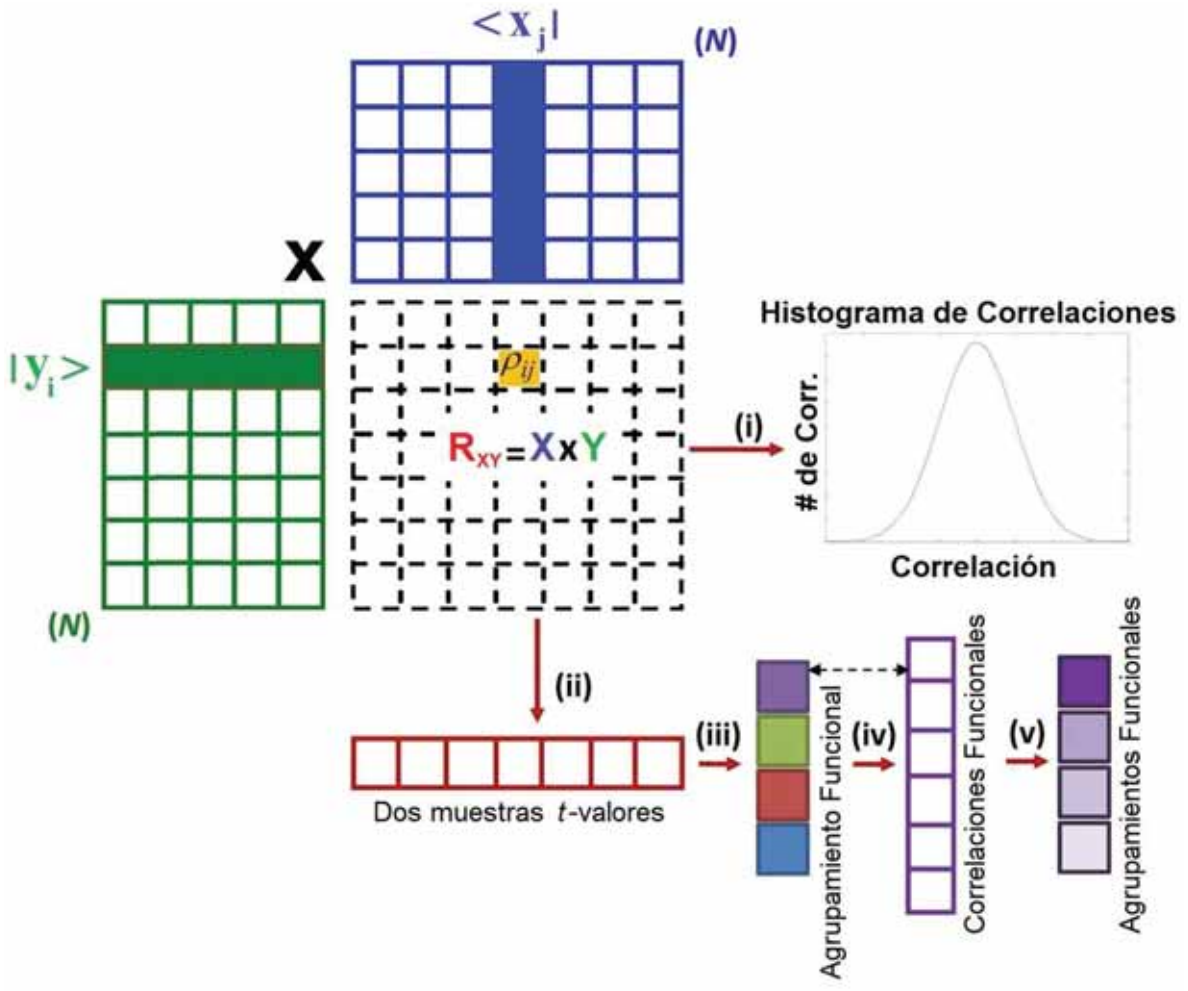


Fig.6.6: Generación de la MMCCu a partir de las correlaciones cruzadas de los arreglos de patrones.

$$C_{X,X} = \frac{1}{N \times N} \sum_{\substack{i=1 \\ j=1}}^{N,N} |X_i\rangle\langle X_j| \qquad \forall\ i, j \tag{6.29}$$

y

$$C_{Y,Y} = \frac{1}{N \times N} \sum_{\substack{i=1 \\ j=1}}^{N,N} |Y_i\rangle\langle Y_j| \qquad \forall\ i, j \tag{6.30}$$

A partir de las cuales podemos definir:

$$\sigma_X = \sqrt{C_{X,X}} \tag{6.31}$$

y

$$\sigma_Y = \sqrt{C_{Y,Y}} \tag{6.32}$$

Finalmente,

$$\rho_{ij} = \frac{C_{XY}^{ij}}{\sigma_X^{ij} \times \sigma_Y^{ij}} \tag{6.33}$$

Esta última ecuación la podemos observar cristalizada tanto en el corazón de la Fig.6.6 como en ambos casos de la Fig.6.7 para los dos casos posibles, i.e., antes y después de la ortogonalización de los conjuntos de patrones. También en esta ecuación tuvimos que recurrir (a la derecha del símbolo igual) al uso del supraíndice a efectos de representar la ubicuidad (mediante los índices *i* y *j*) del coeficiente ρ al solo efecto de simplificar su notación.

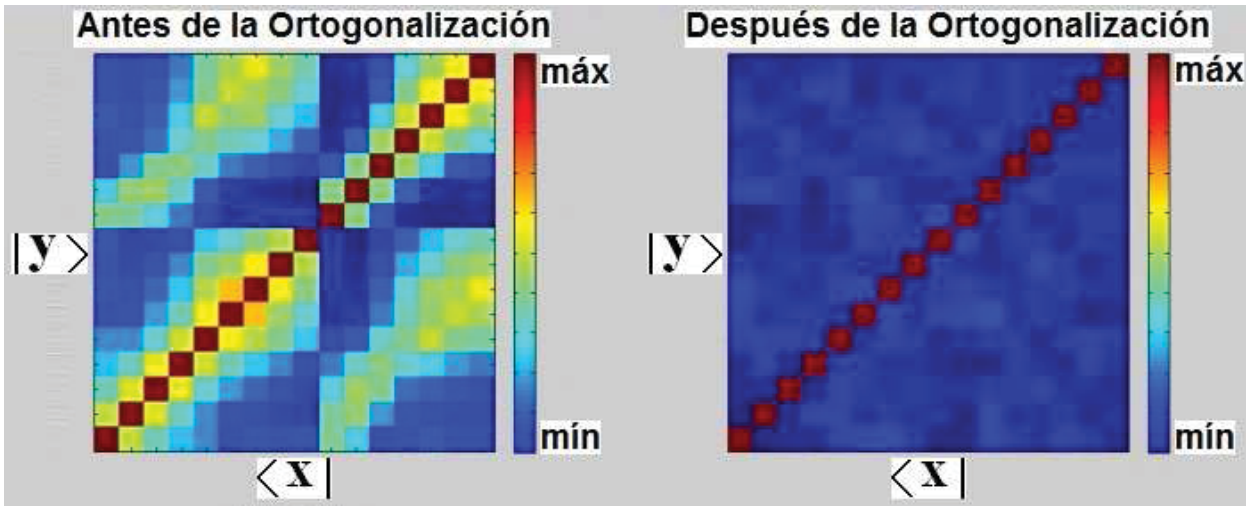


Fig.6.7: R_{XY} antes y después de la ortogonalización de los patrones. En la figura de la izquierda podemos observar que el ruido que representa la existencia de correlaciones intercanales hace colapsar la performance de almacenamiento, cosa que es saneada con la ortogonalización de los patrones donde tanto el ruido como las correlaciones intercanal desaparecen, obteniéndose así una memorización sin errores.

Finalmente, la Fig.6.6 se completa con una serie de módulos adicionales complementarios durante el proceso de recuperación (*recall*) los cuales sirven para realizar análisis estadísticos, diádicos (de a dos muestras) y estructurales, los cuales representan los usos más comunes de este tipo de memorias en la mayoría de sus usos en el mundo clásico [213, 226 y 227].

6.2 Memoria Matricial Correlacionada Booleana Mejorada (MMCBMe)

Esta sección nos introducirá al funcionamiento de la Memoria Matricial Correlacionada Booleana Mejorada (MMCBMe) a partir del empleo de un Proceso de Ortonormalización Booleano (POB) para precisamente, ortonormalizar los patrones ingresantes a una Memoria Matricial Correlacionada Booleana (MMCB) [228, 229], la cual es particularmente útil para trabajar en distintas aplicaciones que requieran el empleo de memorias binarias [230-235].

En el capítulo anterior se presentó un novedoso POB para convertir un conjunto de patrones Booleanos no-ortonormales, i.e., un conjunto no-ortonormal de vectores binarios (en un sentido Booleano) a una base Booleana ortonormal, i.e., un conjunto de vectores binarios ortonormales (en sentido Booleano). Esta sección muestra (al igual que en el caso cuántico) que es posible mejorar la performance de una MMCB gracias al Algoritmo POB. Además, el Algoritmo BOP posee un sinnúmero de aplicaciones en los más variados campos de la Ciencia y la Ingeniería, e.g.: Eteganografía, Redes de Hopfield, Procesamiento de Imágenes Bi-tono, etc. Finalmente, es importante mencionar que el POB (y a diferencia de su contraparte clásica) es estable, siendo por otra parte extremadamente rápido, de fácil paralelización y simple de implementar en FPGA y GPGPU [228, 229].

6.2.1 Memoria Matricial Correlacionada Booleana (MMCB)

Supongamos que una memoria asociativa ha aprendido la matriz de memoria $\boldsymbol{M}$ a través de las asociaciones de patrones binarios (key) y los correspondientes memorizados descritos por $\mathbf{a}_k \rightarrow \mathbf{b}_k$, donde $k = 1, 2, \ldots, q$, siendo $\mathbf{a}_k$ un vector patrón binario (key) de *p-by-1* elementos, i.e., $\mathbf{a}_k = [\, \mathbf{a}_{k1}, \mathbf{a}_{k2}, \ldots, \mathbf{a}_{kp} \,]^T$, y $\mathbf{b}_k$ un vector patrón memorizado binario de *p-by-1*, i.e., $b_k = [\, b_{k1}, b_{k2}, \ldots, b_{kp} \,]^T$, i.e., cuyos valores son exclusivamente 0 o 1.

Podemos postular entonces que $\boldsymbol{M}$, denota un *estimado* de la matriz de memoria $\boldsymbol{M}$ en función de estos patrones como sigue

$$\boldsymbol{M} = \bigvee_{k=1}^{q} (\mathbf{b}_k \wedge \mathbf{a}_k^T) \tag{6.34}$$

donde $\wedge$ representa la operación lógica AND, y el término $\mathbf{b}_k \wedge \mathbf{a}_k^T$ representa la *operación exterior AND entre vectores binarios* del patrón (key) $\mathbf{a}_k$ y el patrón memorizado $\mathbf{b}_k$.

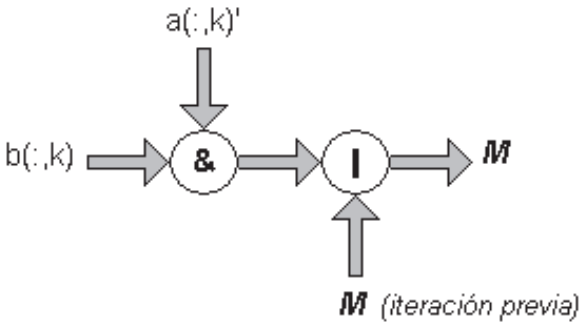


Fig.6.8 Representación gráfica del flujo de señal de la MMCB en sintaxis de MATLAB®.

Por lo tanto, en código de MATLAB® tendremos:

```
M = zeros(p,p);
for k = 1:q
   M = M | (b(:,k) & a(:,k)');
end
```

Entonces, ***M*** es una memoria matricial binaria estimada de *p-por-p*. Donde **|** representa a la operación OR, e & representa a la operación AND en código de MATLAB®, como podemos observar de la Fig.6.8.

6.2.2 Recuperación (recall)

El problema fundamental que sobreviene con el uso de una Memoria Asociativa Booleana (MAB) es la dirección y la recuperación de los patrones almacenados en memoria, siendo en este sentido similar a una Memoria Asociativa Clásica (MACl) [213]. Para explicar un aspecto de este problema, si ***M*** denota la matriz de memoria de una MAB, la cual ha sido completamente aprendida a través de su exposición a *q* patrones de asociaciones de acuerdo con la Ec.(6.34). Si un patrón key $\mathbf{a}_j$ es presentado de manera aleatoria y reaplicado como *estímulo* a la memoria, produciendo la *respuesta*

$$\mathbf{b} = \boldsymbol{M} \wedge \mathbf{a}_j \tag{6.35}$$

El término $\boldsymbol{M} \wedge \mathbf{a}_j$ representa la *operación AND* entre la matriz de memoria Booleana (MMB) ***M*** y el patrón key $\mathbf{a}_j$. Sustituyendo la Ec.(6.34) en la (6.35), tendremos

$$\begin{aligned} \mathbf{b} &= \bigvee_{k=1}^{q} (\mathbf{b}_k \wedge \mathbf{a}_k^T) \wedge \mathbf{a}_j \\ &= \bigvee_{k=1}^{q} (\mathbf{a}_j \wedge \mathbf{a}_k^T) \wedge \mathbf{b}_k \end{aligned} \tag{6.36}$$

donde, en la segunda línea, se reconoce que $\mathbf{a}_k{}^T \wedge \mathbf{a}_j$ es un elemento Booleano igual a la *operación interna AND entre los vectores binarios* para los vectores key $\mathbf{a}_k$ y $\mathbf{a}_j$. Por otra parte, podemos reescribir la Ec.(6.36) como

$$\mathbf{b} = \left(\left(\mathbf{a}_j^T \wedge \mathbf{a}_j\right) \wedge \mathbf{b}_j\right) \vee \left(\bigvee_{(k=1)\wedge(k\neq j)}^{q} (\mathbf{a}_k^T \wedge \mathbf{a}_j) \wedge \mathbf{b}_k \right) \tag{6.37}$$

Ahora vemos que si los vectores key poseen *ortonormalidad Booleana* (i.e., son perpendiculares entre ellos en el sentido Booleano), entonces la respuesta **b** es igual a $\mathbf{b}_j$. De acuerdo con esto, podemos establecer que la memoria Booleana asocia perfectamente si los vectores key forman un *conjunto ortonormal* (en un sentido Booleano); i.e, ellos satisfacen el siguiente par de condiciones:

$$\mathbf{a}_k{}^T \wedge \mathbf{a}_j = 1 \text{ if } k = j \tag{6.38a}$$

y

$$\mathbf{a}_k{}^T \wedge \mathbf{a}_j = 0 \text{ if } k \neq j \tag{6.38b}$$

Supongamos ahora que los vectores key forman efectivamente un conjunto ortonormal (en un sentido Booleano), como se prescribe en la Ec.(6.38). Cuál es entonces el límite en la *capacidad de almacenamiento* de la MAB ? Dicho de otra manera, cuál es el número más grande de patrones que pueden ser realmente almacenados? La respuesta a esta pregunta fundamental se encuentra en el rango de MMB $\boldsymbol{M}$. El rango de una matriz Booleana está definido como el número de columnas independientes (o bien filas, en el sentido Booleano) de dicha matriz, siendo $rango \leq min(p,p)$. La MMB $\boldsymbol{M}$ es una matriz de p-por-p, donde p es la dimensionalidad del espacio de entrada. Por lo tanto, el rango de la MMB $\boldsymbol{M}$ está limitado por la dimensionalidad p. Entonces, podemos establecer formalmente que el número de patrones que pueden ser realmente almacenados nunca pueden exceder la dimensionalidad del espacio de entrada. Esto guarda también una coincidencia absoluta con su contraparte cuántica.

6.3 Memoria Matricial Correlacionada Clásica Mejorada (MMCClMe)

Finalmente, en esta sección desarrollamos el concepto original de la Memoria Matricial Correlacionada Clásica (MMCCl) [213, 236-248], para la cual, empleando el POGS [202] a efectos de ortonormalizar los patrones de entrenamiento, obtendremos la Memoria Matricial Correlacionada Clásica Mejorada (MMCClMe). Esta última goza de una inmejorable capacidad de almacenamiento, como veremos a continuación.

6.3.1 Memoria Matricial Correlacionada Clásica (MMCCl)

Consecuente con la sección anterior, en la versión clásica también supondremos que una memoria asociativa ha aprendido la matriz de memoria $\boldsymbol{M}$ a través de las asociaciones de patrones clásicos (key) y los correspondientes memorizados descritos por $\mathbf{a}_k \rightarrow \mathbf{b}_k$, donde $k = 1, 2, \ldots, q$, siendo $\mathbf{a}_k$ un vector patrón clásico (key) de *p-por-1* elementos, i.e., $\mathbf{a}_k = [\, \mathbf{a}_{k1}, \mathbf{a}_{k2}, \ldots, \mathbf{a}_{kp} \,]^T$, y $\mathbf{b}_k$ un vector patrón memorizado clásico de *p-por-1*, i.e., $b_k = [\, b_{k1}, b_{k2}, \ldots, b_{kp} \,]^T$, i.e., cuyos valores pueden tomar cualquier valor real, aunque por lo general se ecualizan sus valores en el entorno [-1,+1].

Podemos postular entonces que $\boldsymbol{M}$, denota un *estimado* de la matriz de memoria $\boldsymbol{M}$ en función de estos patrones como sigue

$$\boldsymbol{M} = \sum_{k=1}^{q} \left(\mathbf{b}_k \mathbf{a}_k^T \right) \tag{6.39}$$

donde $(\bullet)^T$ significa *transpuesto de* $(\bullet)$, y el término $\mathbf{b}_k \mathbf{a}_k^T$ representa la *operación producto externo entre los vectores* del patrón (key) $\mathbf{a}_k$ y el patrón memorizado $\mathbf{b}_k$.

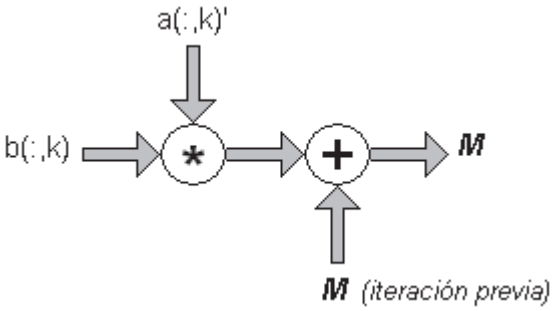


Fig.6.9 Representación gráfica del flujo de señal de la MMCCl en sintaxis de MATLAB®.

Por lo tanto, en código de MATLAB® tendremos:

```
M = zeros(p,p);
for k = 1:q
   M = M + (b(:,k) * a(:,k)');
end
```

Entonces, $\boldsymbol{M}$ es una memoria matricial clásica estimada de *p-por-p*. Como podemos observar de la Fig.6.9 el esquema adaptativo es muy simple.

6.3.2 Recuperación (recall)

El problema fundamental que sobreviene con el uso de una Memoria Asociativa Clásica (MACl) es el mismo que en sus contrapartes cuánticos y Booleanos, es decir, dicho problema es la dirección y la recuperación de los patrones almacenados en memoria, siendo en este sentido similar a una Memoria Asociativa Cuántica (MACu) [212]. Para explicar un aspecto de este problema, si $\boldsymbol{M}$ denota la matriz de memoria de una MACl, la cual ha sido completamente aprendida a través de su exposición a q patrones de asociaciones de acuerdo con la Ec.(6.39). Si un patrón key $\mathbf{a}_j$ es presentado de manera aleatoria y reaplicado como *estímulo* a la memoria, produciendo la *respuesta*

$$\mathbf{b} = \boldsymbol{M}\,\mathbf{a}_j \tag{6.40}$$

El término $\boldsymbol{M}\,\mathbf{a}_j$ representa la *operación de producto clásico* entre la Matriz de Memoria Clásica (MMCl) $\boldsymbol{M}$ y el patrón key $\mathbf{a}_j$. Sustituyendo la Ec.(6.39) en la (6.40), tendremos

$$\begin{aligned} \boldsymbol{b} &= \sum_{k=1}^{q} \left(\mathbf{b}_k \mathbf{a}_k^T\right) \mathbf{a}_j \\ &= \sum_{k=1}^{q} \left(\mathbf{a}_j \mathbf{a}_k^T\right) \mathbf{b}_k \end{aligned} \tag{6.41}$$

donde, en la segunda línea, se reconoce que $\mathbf{a}_j \mathbf{a}_k^T$ es un elemento clásico igual a la *operación producto interno entre los vectores clásicos* para los vectores key $\mathbf{a}_k$ y $\mathbf{a}_j$. Por otra parte, podemos reescribir la Ec.(6.41) como

$$\boldsymbol{b} = \left(\left(\mathbf{a}_j^T \mathbf{a}_j\right)\mathbf{b}_j\right) + \left(\sum_{(k=1)\wedge(k\neq j)}^{q} \left(\mathbf{a}_k^T \mathbf{a}_j\right)\mathbf{b}_k\right) \tag{6.42}$$

Ahora vemos que si los vectores key poseen *ortonormalidad clásica* (i.e., son perpendiculares entre ellos en el sentido clásio), entonces la respuesta $\mathbf{b}$ es igual a $\mathbf{b}_j$. De acuerdo con esto, podemos establecer que la memoria clásica asocia perfectamente si los vectores key forman un *conjunto ortonormal* (en el sentido clásico); i.e, ellos satisfacen el siguiente par de condiciones:

$$\mathbf{a}_k{}^T \mathbf{a}_j = 1 \text{ if } k = j \tag{6.43a}$$

y

$$\mathbf{a}_k{}^T \mathbf{a}_j = 0 \text{ if } k \neq j \tag{6.43b}$$

Supongamos ahora que los vectores key forman efectivamente un conjunto ortonormal (en un sentido clásico), como se prescribe en la Ec.(6.43). Cuál es entonces el límite en la *capacidad de almacenamiento* de la MACl ? Dicho de otra manera, cuál es el número más grande de patrones que pueden ser realmente almacenados? La respuesta a esta pregunta fundamental se encuentra en el rango de MMCl $\boldsymbol{M}$. El rango de una matriz clásica está definido como el número de columnas independientes (o bien filas, en el sentido clásico) de dicha matriz, siendo $rango \leq min(p,p)$. La MMCl $\boldsymbol{M}$ es una matriz de p-por-p, donde p es la dimensionalidad del espacio de entrada. Por lo tanto, el rango de la MMCl $\boldsymbol{M}$ está limitado (aquí también) por la dimensionalidad p. Entonces, podemos establecer formalmente que el número de patrones que pueden ser real-

mente almacenados nunca pueden exceder la dimensionalidad del espacio de entrada. Esto guarda también una coincidencia absoluta con sus contrapartes cuántica y Boolena.

6.4 Conclusiones del capítulo

Presentamos aquí los siguientes aportes:

1. una versión mejorada de las ya conocidas Memorias Matriciales Correlacionadas Boolenas (MMCB), es decir, la MMCB mejorada (MMCBMe) en base al innovador Proceso de Ortonormalización Booleano (POB) del capítulo anterior,

2. la Memoria Matricial Correlacionada Cuántica (MMCCu), y

3. la MMCCu Mejorada (MMCCuMe) en base al Proceso de Ortogonalización Cuántico (POCu) implementado en forma escalonada y conocida como la Red Ortogonalizadora Cuántica Sistólica (ROCS) en el capítulo anterior.

Mientras este último aporte se constituye en la última Red Neuronal Cuántica (RNCu), completamos el conjunto de herramientas tendientes a:

a) el arribo de la representación subsimbólica del conocimiento (provenientes de la Inteligencia Artificial) al ámbito cuántico, y

b) una mejora en los procedimientos conducentes al reconocimiento de patrones en dicho ámbito.

Capítulo 7: Simulaciones

Introducción

En este capítulo se llevan a cabo 6 simulaciones: 3 de MMCCu y 3 de Algebra y Procesamiento Cúantico de Imágenes, al solo efecto probatorio de los atributos de ortogonalizador del POCu. En el caso de los primeros 3 experimentos, los patrones cuánticos (memorias) son generados a partir del software *open source* del Laboratorio Clarendon y del Keble College de la Universidad de Oxford, Inglaterra, llamado QNNLab. Este software genera patrones de entrenamiento simulados para varios de los tipos de redes neuronales cuánticas (RNCu), no así para la MMCCu, dado que hasta este trabajo no existían. No obstante, emplearemos aquellos utilizados en las memorias de otros modelos de RNCu, concretamente las conocidas como perceptrones multicapa cuánticos (PMCu).

En las primeras 3 simulaciones trataremos de enfatizar la performance de almacenamiento de la MMCCu, antes y después de la aplicación del POCu a las memorias de entrenamiento (o sea, antes y después de su ortonormalización), lo cual impactará directamente en la correlación testigo de los patrones de training y recall de la Sección 6.1.2.6 del Capítulo 6.

7.1 Experimento 1

Para este experimento se dispone de 26 patrones de 35 cubits cada uno. Cada patrón representa un símbolo de un alfabeto. Dichos símbolos guardan gran correlación morfológica entre ellos, razón por la cual pierden distinguibilidad a la hora de pretender recuperar los almacenados. Esto produce directamente la presencia de ruido de recall el cual baja drásticamente la performance de reconocimiento del símbolo. Como podemos apreciar en detalle en la Fig.7.1, la columna de la izquierda (con los 3 gráficos) representa en orden:

1. la correlación testigo de los patrones de training y recall,
2. los errores de acierto y
3. el error de almacenamiento vs el nivel de ruido

En los tres métricas mencionadas para el caso sin ortonormalizar, la columna de la derecha representa lo mismo pero para el caso que incluye normalización de los patrones.

Los dos gráficos superiores de dicha figura muestran que en el caso sin ortonormalizar (estado original) las memorias están fuertemente correlacionadas, a través de coeficientes de valores apreciables por fuera de la diagonal principal, lo cual evidencia un compromiso morfológico entre los símbolos en cuestión. Esto representa una intercanalización que hace perder distinguibilidad a los patrones en pos de su recuperación, lo cual se interpreta como un ruido de recall. Mientras que en el caso con ortonormalización de los patrones podemos ver claramente que se obtiene una matriz casi diagonal, eliminando la intercanalización y mejorando la performance de almacenamiento.

Los dos gráficos del medio representan el error de acierto con y sin ortonormalización. Como podemos apreciar, mientras en el caso sin ortonormalizar solo se dieron 4 aciertos sobre 26, en el caso que incluye la ortonormalización de los patrones, todos fueron aciertos.

Finalmente, los dos gráficos inferiores muestran el error de almacenamiento vs el nivel de ruido, el cual en el caso sin ortonormalizar es casi insensible a este respecto, i.e., la performance en mala para todo nivel de ruido, mientras para el caso ortonormalizado –y como es de esperarse para una memoria según los lineamientos establecidos en el capítulo anterior– el error de almacenamiento crece en forma lineal con el ruido.

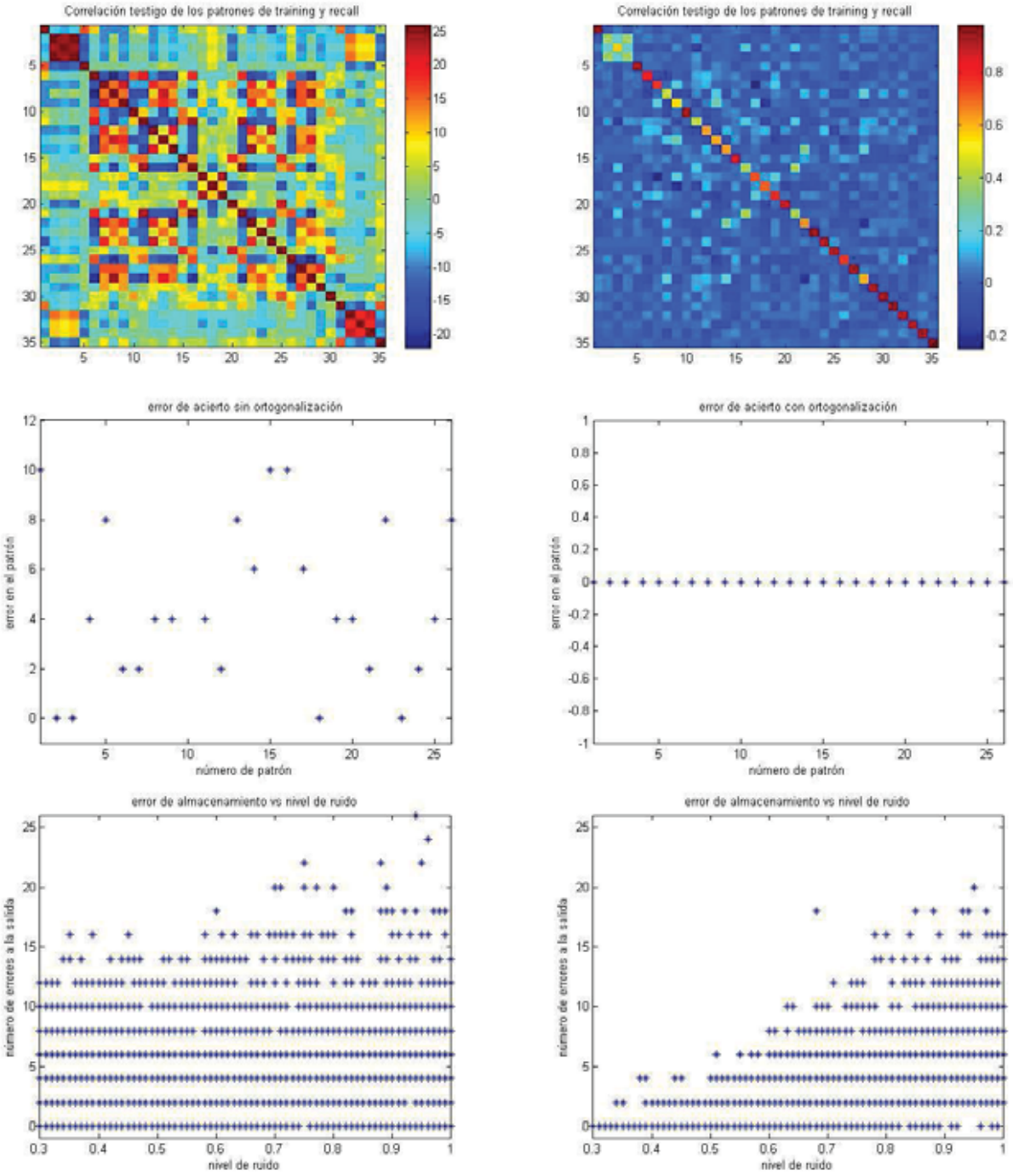


Fig.7.1: 26 patrones de 35 cubits cada uno. La columna de la izquierda (con los 3 gráficos) representa en orden la correlación testigo de los patrones de training y recall, los errores de acierto y el error de almacenamiento vs el nivel de ruido; para el caso sin ortonormalizar. La columna de la derecha representa lo mismo pero para el caso que incluye normalización.

7.2 Experimento 2

Similares consideraciones al igual que funcionamiento se da en este experimento respecto del exterior considerando que disponemos aquí de más patrones y más grandes, lo cual evidencia un saludable comportamiento lineal con las dimensiones en este tipo de memoria cuántica.

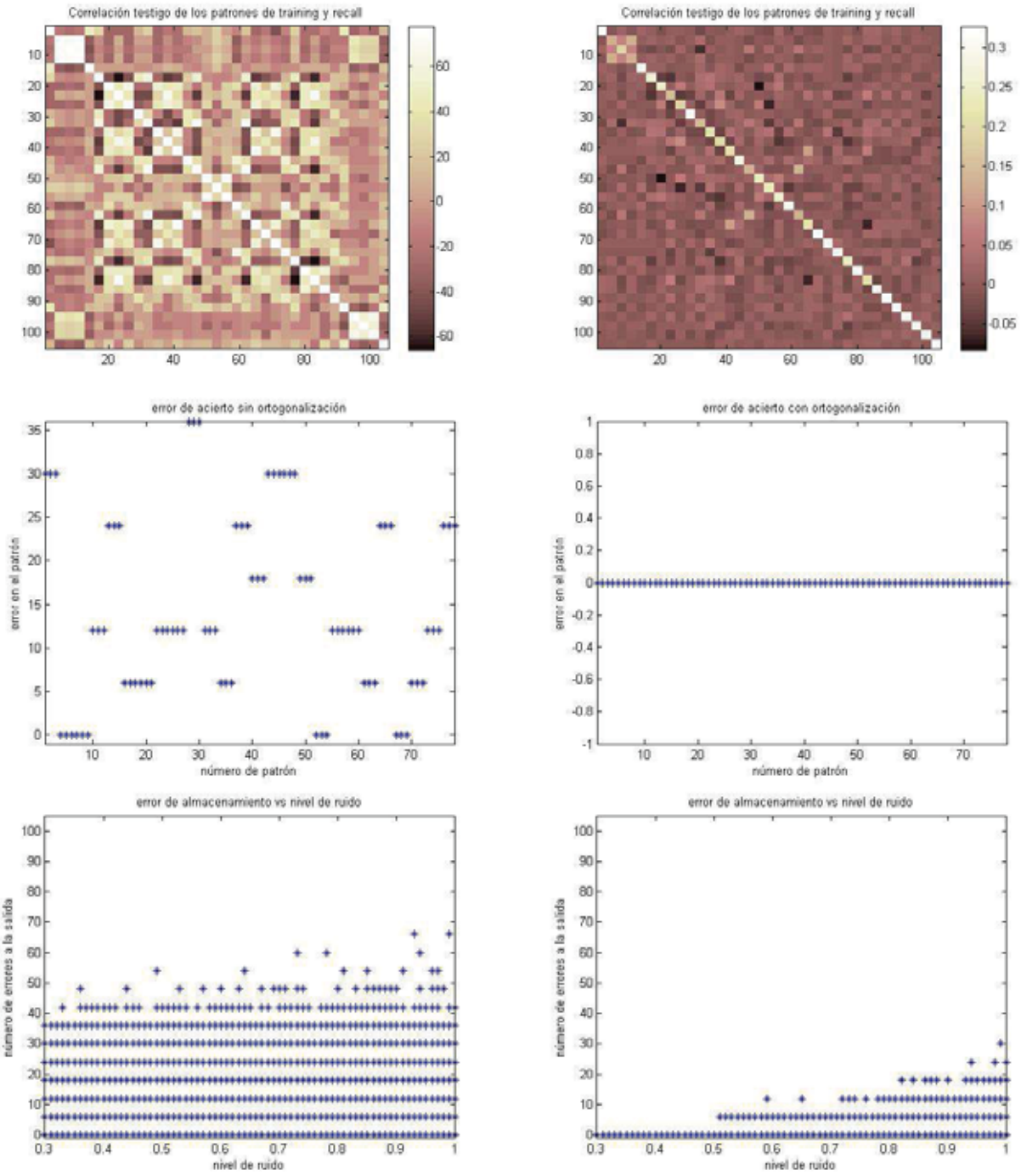


Fig.7.2: 78 patrones de 105 cubits cada uno. La columna de la izquierda (con los 3 gráficos) representa en orden la correlación testigo de los patrones de training y recall, los errores de acierto y el error de almacenamiento vs el nivel de ruido; para el caso sin ortonormalizar. La columna de la derecha representa lo mismo pero para el caso que incluye normalización.

7.3 Experimento 3

Idénticas consideraciones que en los dos casos anteriores se dan para éste, con la salvedad que aquí se trata de muchos más patrones y considerablemente más grandes, no obstante, se guardan las mismas relaciones y conclusiones que las formuladas en el caso anterior.

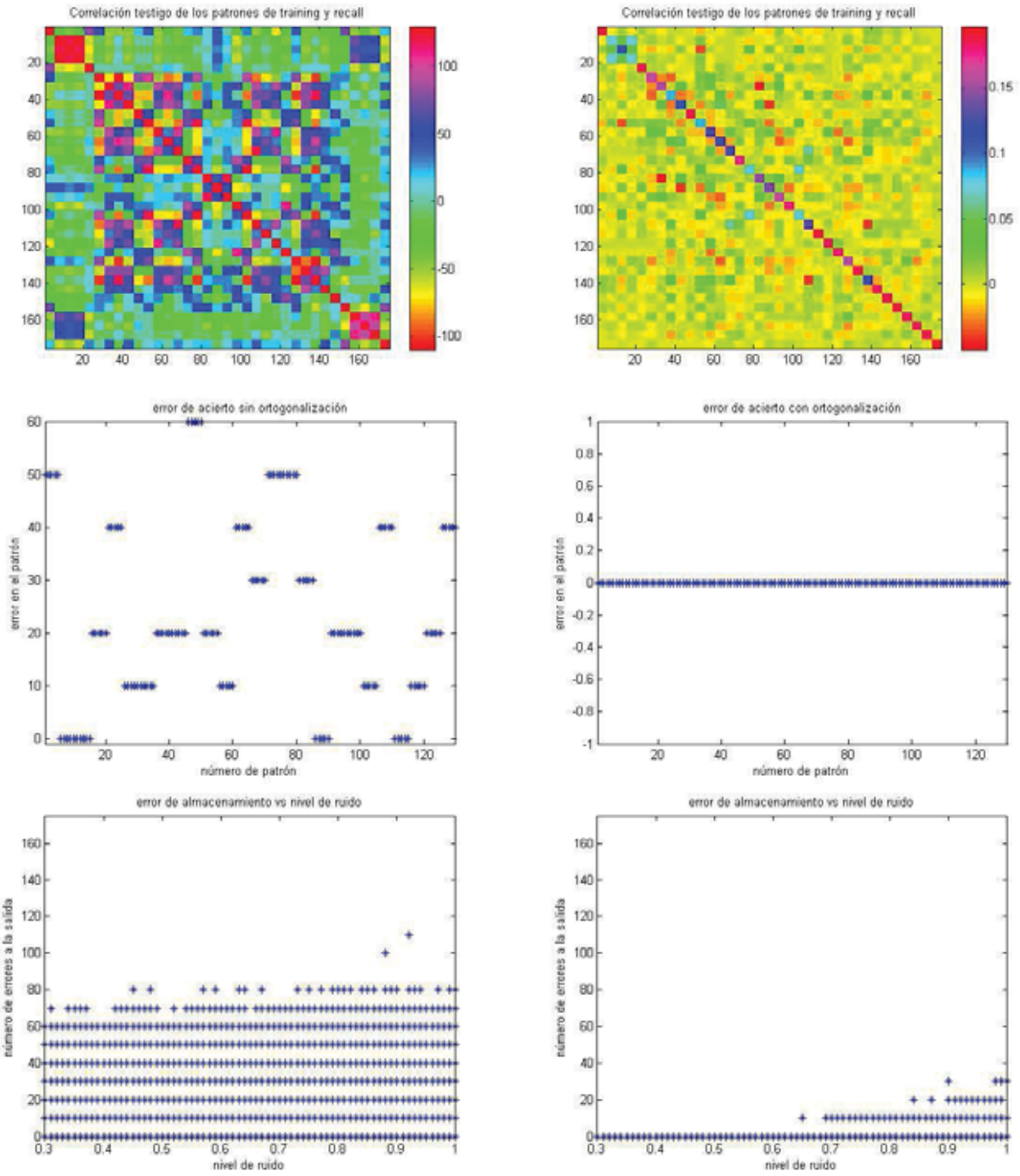


Fig.7.3: 130 patrones de 175 cubits cada uno. La columna de la izquierda (con los 3 gráficos) representa en orden la correlación testigo de los patrones de training y recall, los errores de acierto y el error de almacenamiento vs el nivel de ruido; para el caso sin ortonormalizar. La columna de la derecha representa lo mismo pero para el caso que incluye normalización.

Finalmente, en las últimas 3 simulaciones trataremos de enfatizar la performance de ortogonalización del POCu aplicado a arreglos uni, bi y tridimensionales, los cuales representan señales, imágenes (multimediales, documentales, satelitales, biométricas, etc.) y video o bien imágenes multi e hiper-espectrales satelitales o tomografías o resonancias magnéticas seriadas, respectivamente.

Un aspecto muy importante aclarar aquí es que a los efectos de un más eficiente procedimiento de simulación del POCu sobre una máquina clásica (CPU) para estos 3 últimos experimentos, utilizaremos el POB, haciendo el parangón entre [verdadero, falso] del mundo Booleano, [0, 1] del mundo binario y [$|0\rangle$, $|1\rangle$] del mundo cuántico.

7.4 Experimento 4

En esta simulación probaremos la virtud ortogonalizadora del POCu para un conjunto de vectores columna cuánticos, como se muestra en la parte derecha de la Fig.7.4. La parte izquierda de la misma figura nos muestra la contraparte Booleana/Binaria del conjunto dado.

La columna izquierda de la Fig.7.4 nos muestra un conjunto de vectores Booleanos/Binarios antes y después de aplicar el POB. Lo propio en la parte derecha de la misma figura para el caso cuántico. Como puede observarse, luego de los respectivos ortogonalizadores no hay unos a la derecha del primer uno. En ambos casos, la columna de la izquierda representa el primer patrón a considerar, y teniendo en cuenta que todo procedimiento de ortogonalización es en si mismo *no democrático*, la única posibilidad valedera en ambos casos en virtud de los productos internos respectivos definidos en el Capítulo 5, es que en la fila donde haya un uno no puede haber otro, esto es estricto, excluyente y en orden de precedencia de izquierda a derecha, es decir, del primero al último. Estos resultados muestran inequívocamente la eficiencia del ortogonalizador cuántico, el cual ha mostrado el mismo comportamiento para más de 50 ejemplos de este tipo simulados, de los cuales aquí solo mostramos uno.

Finalmente, en la parte derecha de la figura, espín azul hacia arriba significa $|0\rangle$, mientras que espín rojo hacia abajo representa $|1\rangle$.

7.5 Experimento 5

En esta simulación se descompone una imagen en sus respectivos elementos Booleanos/Binarios constitutivos, como muestra la Fig.7.5, en la cual se puede observar la descomposición (bit-slicer) de una versión de Lena de 512x512 pixeles en grises con 8 bit-x-pixel en sus 8 planos de bits (bit-planes) y el posterior re-ensamble (bit-reassembler) con objeto de recuperar la imagen original.

La Fig.7.6, en cambio nos muestra la acción del ROBS, el cual nos permite obtener los bit-planes de **U** y **S** a partir del original de Lena **V**. Como puede observarse **U** y S son ortogonales entre sí, al igual que los bit-planes de **U** entre sí. Unos aspectos importantes a tener en cuenta son: a) Una vez re-ensamblado **U**, esta debería dar una imagen con dithering (el cual sirve para simular profundidad de color en Procesamiento de Imágenes, Artes Electrónicas y Videojuegos), b) **U** se puede seguir descomponiendo como el caso de **V**, y c) Al ser ortogonales entre sí, los elementos de **U** forman una base que proviene de Lena para ser usada en todo sistema de proyección de cualquier otra imagen dada de esa dimensión y bit-por-pixeles en esa base con fines varios en Procesamiento de Imágenes en general y compresión de imágenes en particular.

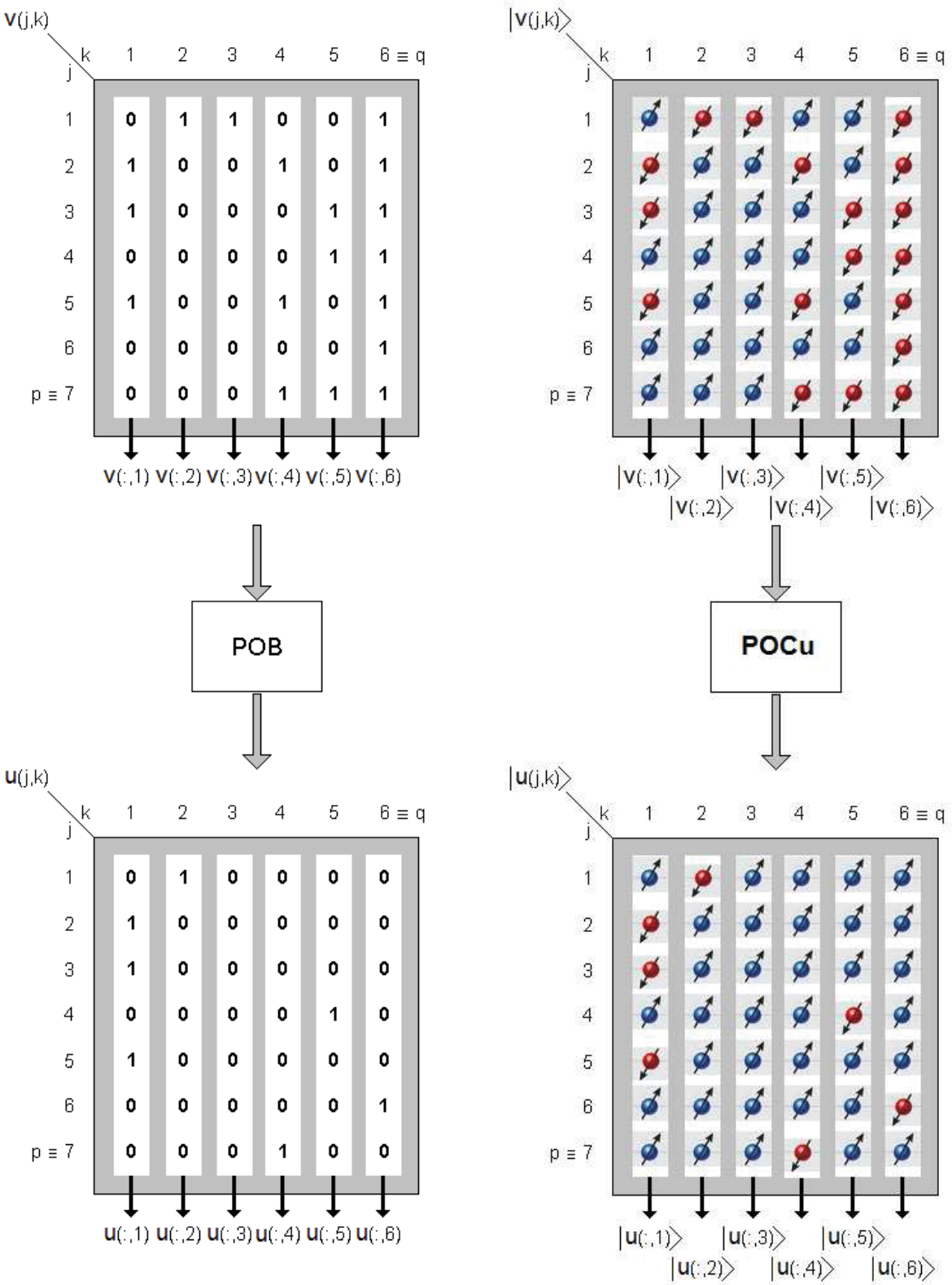


Fig.7.4: La figura de la izquierda nos muestra un conjunto de vectores Booleanos/Binarios antes y después de aplicar el POB. Lo propio en la figura de la derecha para el caso cuántico. Como puede observarse, luego de los ortogonalizadores no hay unos a la derecha del primer uno. En la figura de la derecha, espín azul hacia arriba significa $|0\rangle$, mientras que espín rojo hacia abajo representa $|1\rangle$.

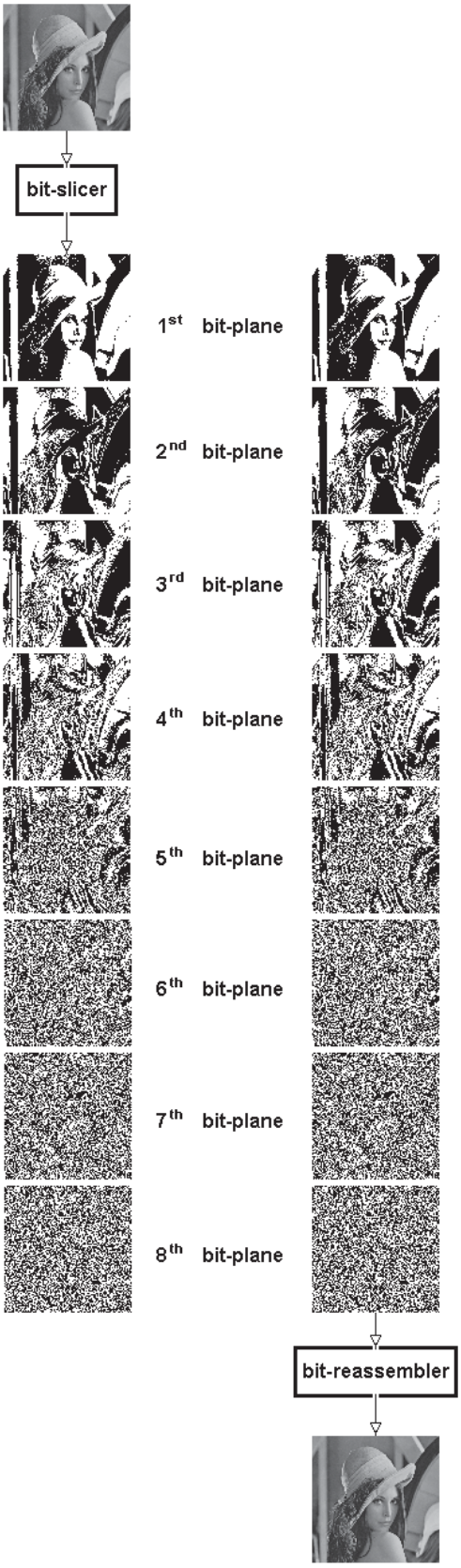


Fig.7.5: En esta figura se puede observar la descomposición (bit-slicer) de una versión de Lena de 512x512 pixeles en grises con 8 bit-x-pixel en sus 8 planos de bits (bit-planes) y el posterior re-ensamble (bit-reassembler) con objeto de recuperar la imagen original.

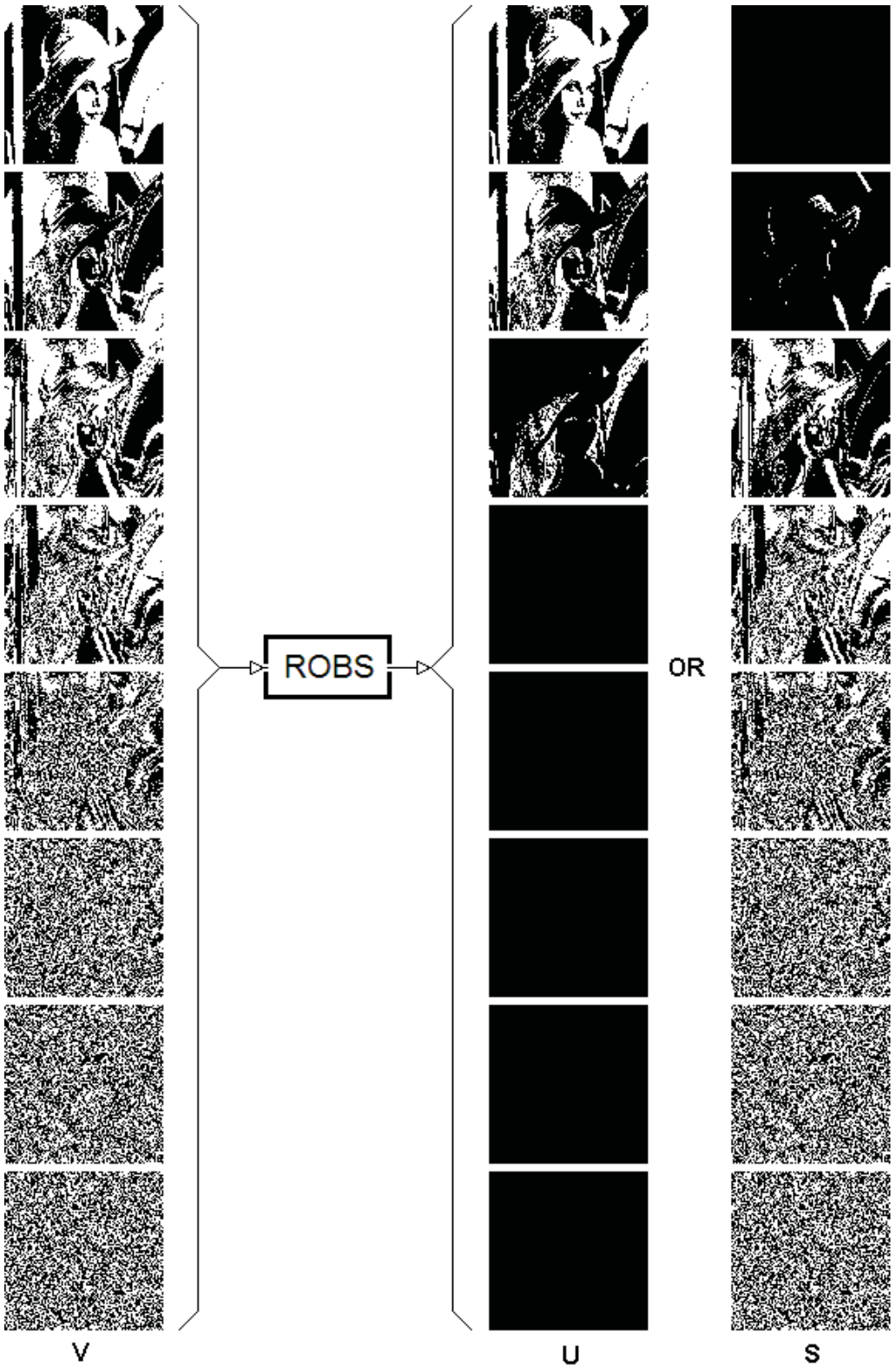


Fig.7.6: El ROBS nos permite obtener los bit-planes de **U** y **S** a partir del original de Lena **V**. Como puede observarse **U** y S son ortogonales entre sí, al igual que los bit-planes de **U** entre sí. Unos aspectos importantes a tener en cuenta son: a) Una vez re-ensamblado **U** debería dar una imagen con dithering, b) **U** se puede seguir descomponiendo como el caso de **V**, y c) Al ser ortogonales entre sí, los elementos de **U** forman una base que proviene de Lena para un sistema de proyección de cualquier otra imagen de esa dimensión y bit-por-pixeles en esa base con fines varios en Procesamiento de Imágenes en general y compresión de imágenes en particular.

V

U

S

Fig.7.7: Siendo la imagen superior Lena original, la del medio representa el ensamble de los elementos de U, lo cual corrobora que se constituye una imagen con dithering, y la de abajo la imagen residual, de la cual en futuros trabajos deberá explorarse sus potencialidades en el campo del Procesamiento Cuántico de Imágenes, en virtud de sus posibilidades de permitir continuar descomponiéndose como **V** recursivamente, con un vasto terreno por delante para su aplicación en compresión y segmentación de imágenes de todo tipo.

La Fig.7.7 contiene en su parte superior a Lena original, mientras que la del medio representa el ensamble de los elementos de U, lo cual corrobora que se constituye una imagen con dithering, y finalmente la de abajo nos proporciona la imagen residual, de la cual en futuros trabajos deberá explorarse sus potencialidades en el campo del Procesamiento Cuántico de Imágenes, en virtud de sus posibilidades de permitir continuar descomponiéndose como **V** recursivamente, con un vasto terreno por delante para su aplicación en compresión y segmentación de imágenes de todo tipo.

7.6 Experimento 6

En esta simulación se consideran un grupo de cubos Booleanos/Binarios, donde cada cubo representa una banda espectral en diferentes campos del Arte, Ciencia y Tecnología, lo cual puede observarse en la Fig.7.8.

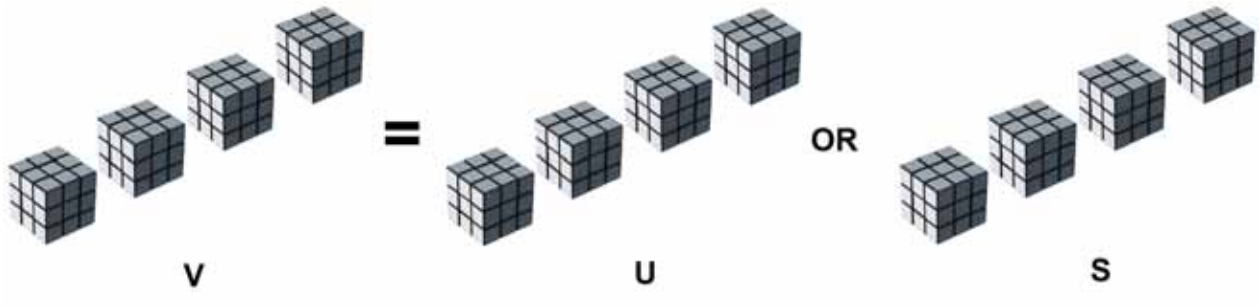


Fig.7.8: En esta figura **V** esta constituida por un conjunto de cubos binarios, los cuales pueden representar a la descomposición en sus respectivos bit-planes de bandas multi e hiper-espectrales de un satélite, o bien, slices de una tomografía o resonancia magnética seriada.

En este procedimiento se genera una base de cubos Booleanos/Binarios siendo su procedencia del tipo de la Fig.7.9, la cual esta compuesta de los slices de una tomografía seriada oxiaxial de cabeza con un slicing de 1 milímetro entre slices de 12 bit-por-pixel cada uno y de 1024x1024 pixeles. En la Fig.7.10 podemos observar el ordenamiento de los slices de la tomografía computada seriada oxiaxial de la figura anterior, la cual en total suman 34 slices.

Por otra parte, en la Fig.7.11 podemos observar una imagen del satélite SPOT 5 multi-espectral de 16 bandas, el cual genera las imágenes de la Fig.7.12, donde se presentan las bandas espectrales de dicho satélite con el detalle el primer bit-plane para cada una.

Finalmente, la Fig.7.13 nos muestra las bandas espectrales de la Mona Lisa, las cuales van del infra-rojo al ultra-violeta pasando por el rango visible. Un caso similar lo representan las bandas de la Fig.7.14, con bandas generadas mediante diferentes fuentes de la misma obra, entre los cuales se incluye hasta Rayos X. Por último, la Fig.7.15 representa la imagenología multi-espectral para documentos de Artes Electrónicas.

En todos los casos se generó base, tanto para el caso Booleano/Binario como su contraparte cuántica simulada, al alcanzarse dos grupos de bandas, es decir, **U** y **S**, a partir de la **V** original, y también se comprobó la ortogonalidad de los elementos **U** entre si y con los **S**.

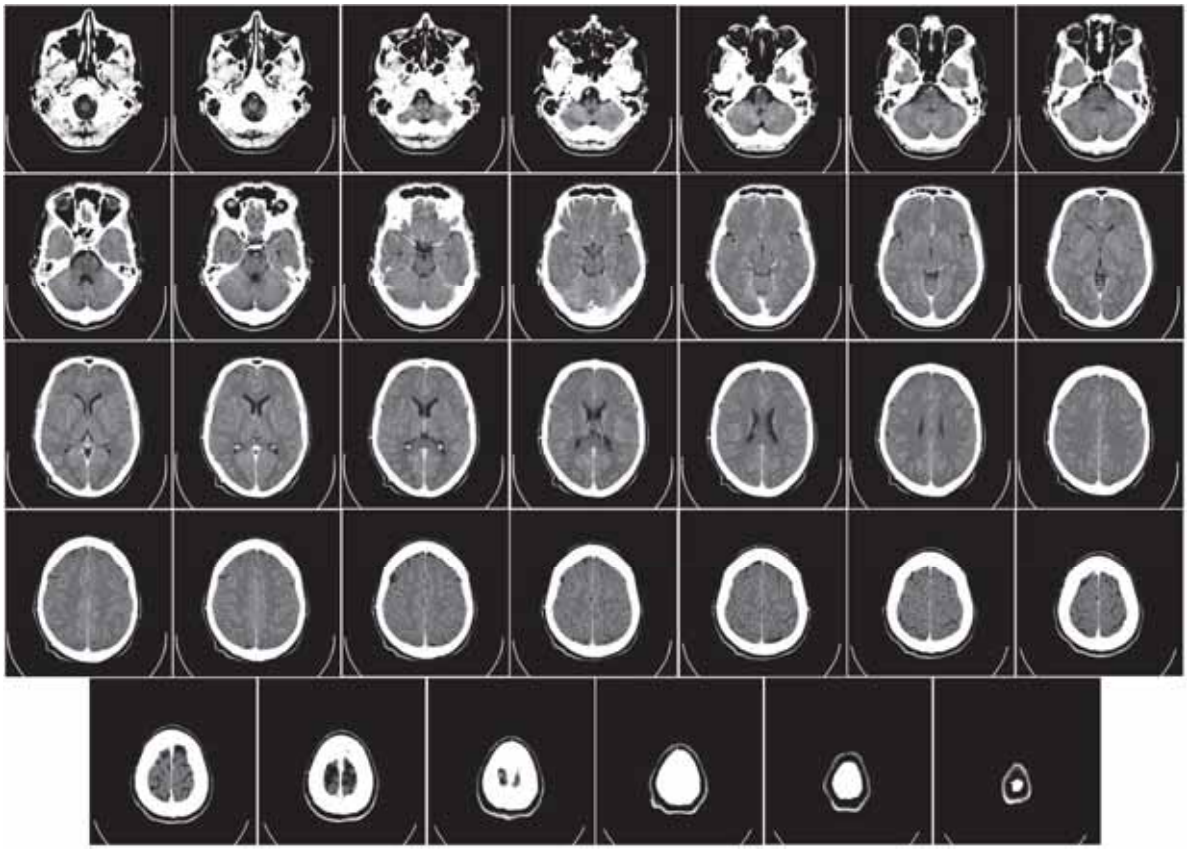


Fig.7.9: Slices de una tomografía seriada oxiaxial de cabeza con un slicing de 1 milímetro entre slices de 12 bit-por-pixel cada uno y de 1024x1024 pixeles.

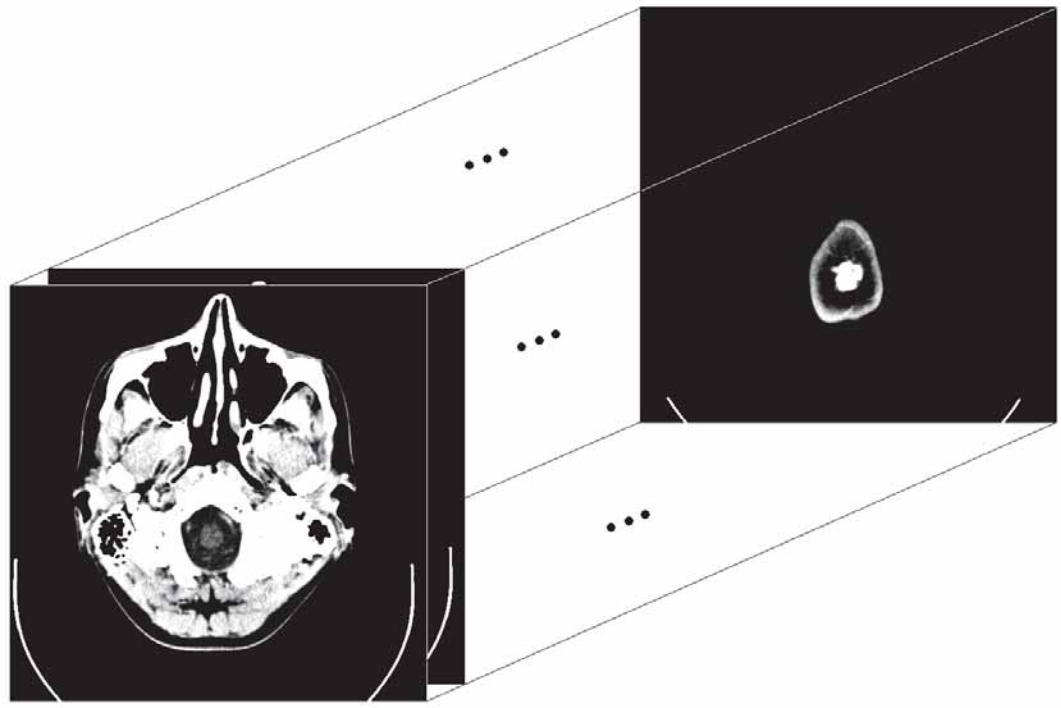


Fig.7.10: Ordenamiento de los slices de la tomografía computada seriada oxiaxial de la figura anterior. En total suman 34 slices.

Fig.7.11: Satélite SPOT 5 multi-espectral de 16 bandas.

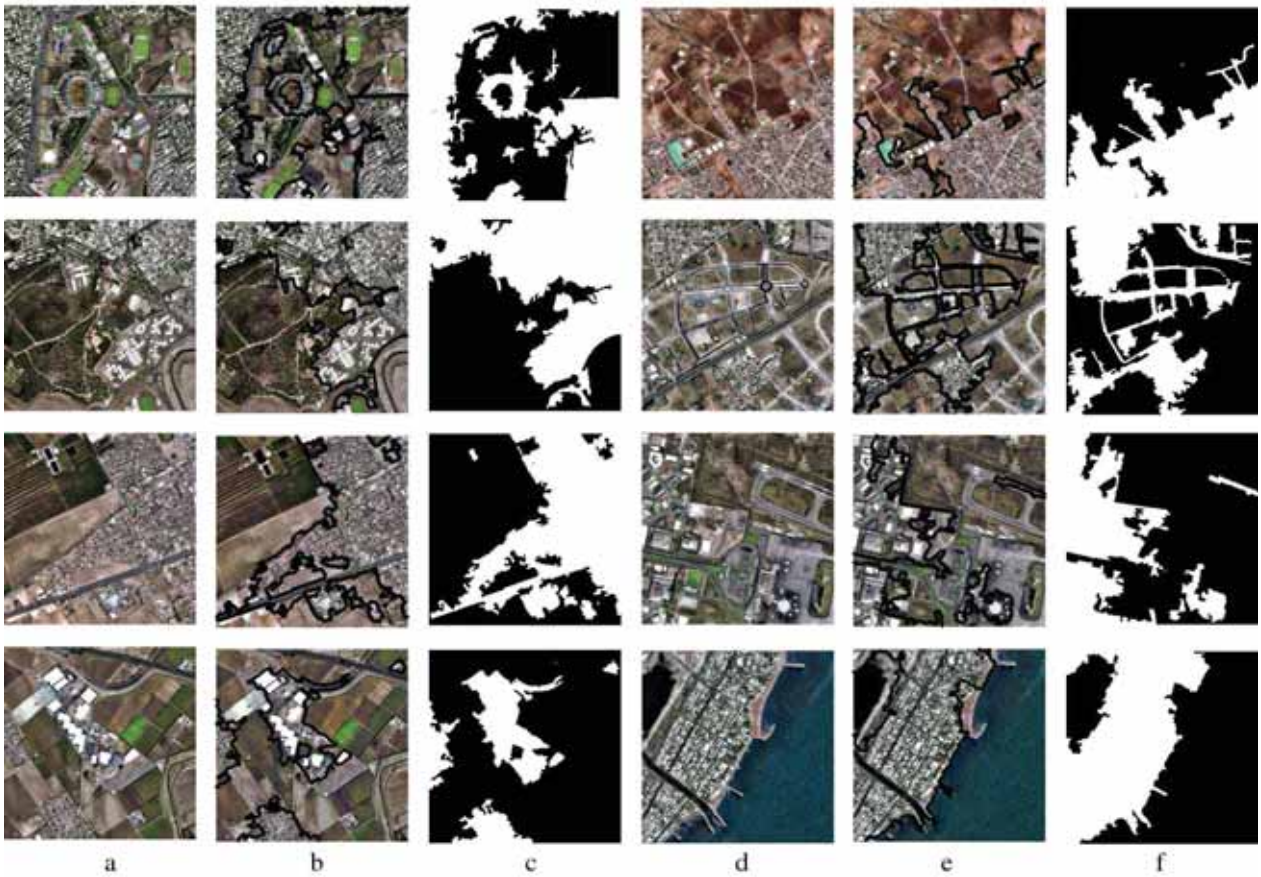


Fig.7.12: Bandas espectrales del satélite SPOT 5 con el detalle el primer bit-plane de cada banda.

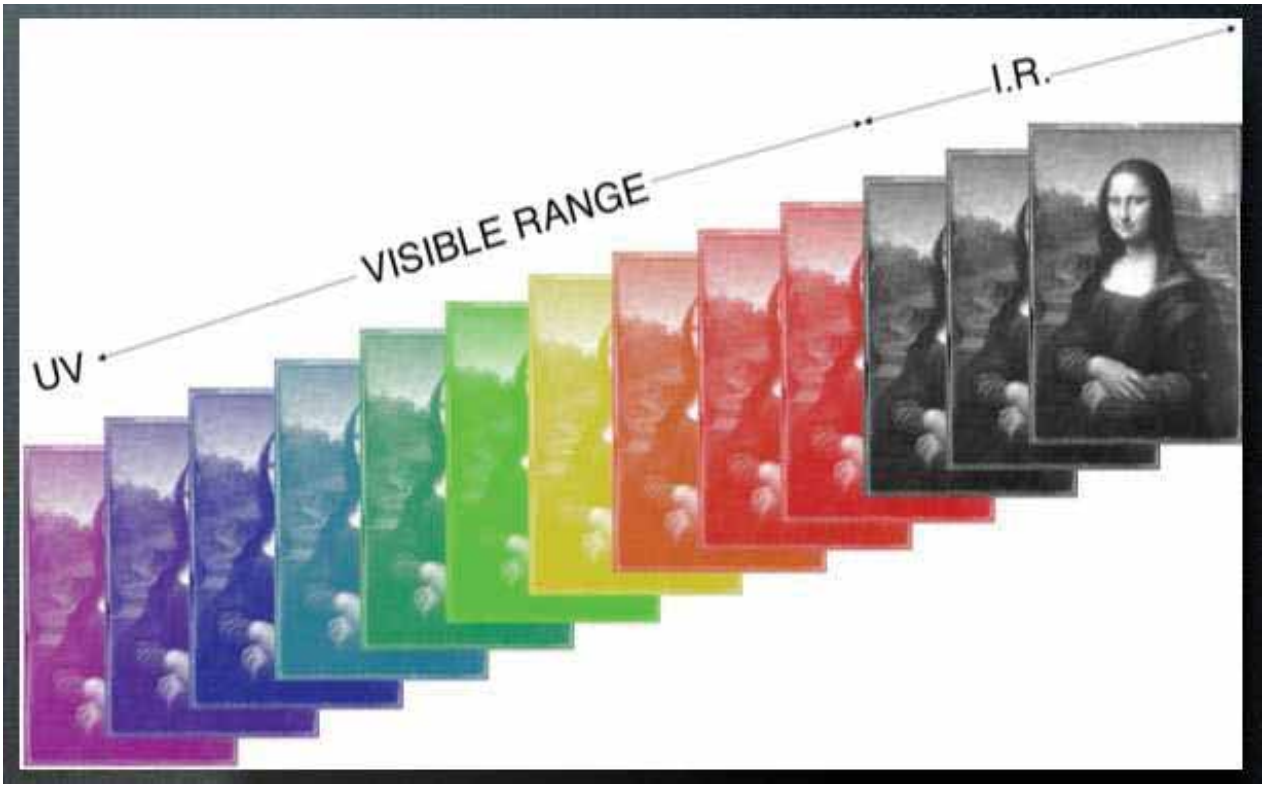


Fig.7.13: Bandas espectrales de la Mona Lisa que van del infra-rojo al ultra-violeta pasando por el rango visible.

Fig.7.14: Bandas generadas mediante diferentes fuentes de la misma obra.

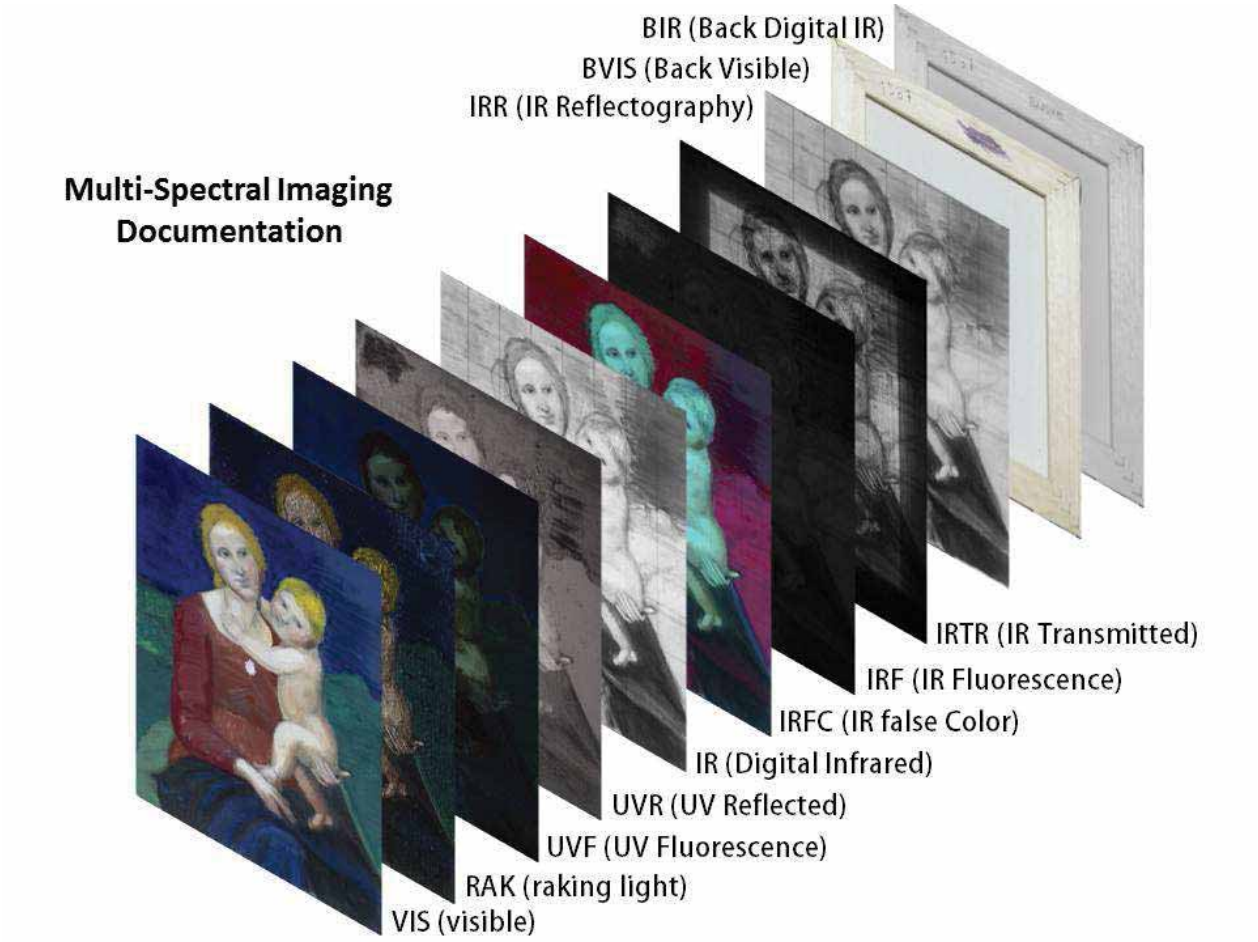


Fig.7.15: Imagenología multi-espectral para documentos de Artes Electrónicas.

7.7 Conclusiones del capítulo

Se presentan aquí 6 simulaciones, La primeras 3 a partir de los patrones cuánticos generados con el emulador QNNLab de la Universidad de Oxford. En base a los mismos se simularon procesos de almacenamiento sobre la MMCCu desarrollada en el Capítulo 6 con y sin la ortonormalización cuántica desarrollada en el Capítulo 5. Los resultados son elocuentes en lo concerniente a los errores de recuperación, que es otra forma de expresar el aumento de la performance de memorización gracias a la superación del ruido en base al empleo de la ortonormalización cuántica.

Las últimas 3 simulaciones, en cambio, tienen que ver con aplicaciones de las herramientas mencionadas al Procesamiento Cuántico de Imágenes multimediales, médicas (tomografías y resonancias seriadas), satelitales (multi e hiper-espectrales) y aquellas relativas a documentos de Artes Electrónicas.

Finalmente, el contenido de este capítulo es en si mismo un aporte al área del conocimiento en cuestión, por cuanto es la primera vez en la literatura que se realizan mediante la tecnología cuántica e implementadas sobre GPGPU simulaciones de MMCCu y MMCB, así como aquellas relativas al Procesamiento Cuántico de Imágenes y además con la performance alcanzada.

Capítulo 8: Conclusiones

Conclusiones

Capítulo 1:
En este capítulo exploramos los fundamentos de la Mecánica Cuántica, su lógica simbólica en general y la notación de Dirac en particular, enfatizando las nociones de espín, entrelazamiento cuántico y estados y desigualdades de Bell.

Capítulo 2:
En este capítulo presentamos los conceptos fundamentales de la Computación Cuántica, desde la noción de cubit hasta la Máquina de Turing Cuántica pasando por la noción de medición, compuertas y algoritmos cuánticos.

Capítulo 3:
En este capítulo presentamos un marco multicapa para una arquitectura de computadora cuántica. El marco multicapa tiene dos tiene dos puntos fuertes: es modular, y facilita la tolerancia a fallas. La naturaleza multicapa de la arquitectura aconseja la modularidad, pero la característica que define a las capas que hemos elegido es la encapsulación. Cada una de las capas tiene un único e importante propósito así como los paquetes de las operaciones vinculadas a cumplir con dicho propósito. Adicionalmente, cada capa juega el rol de administrador de recursos, dado que frecuentemente muchas operaciones en la capa más baja son combinadas en la capa más alta. Puesto que las tecnologías de la computación cuántica van a evolucionar con el tiempo, las capas pueden necesitar un reemplazo en el futuro, y la encapsulación hace de la integración de los nuevos procesos una tarea más sencilla.

La tolerancia a fallos es en la actualidad el mayor reto para los computadores cuánticos, y la organización en capas es elegida deliberadamente para servir a esta necesidad. Podría decirse que las Capas 1 y 5 definen a cualquier computadora cuántica, pero las capas intermedias se dedican exclusivamente a la creación de la tolerancia a fallas de una manera inteligente. La Capa 2 utiliza un control simple para mitigar los errores sistemáticos, por lo que esta capa está posicionada cerca de la capa Física donde las técnicas como el DD y los subespacios libres de decoherencia son más efectivos. La Capa 3 es la anfitriona del CEC, que es esencial para modelo de circuito de computadora cuántica de gran escala para sobre cualquier hardware, tales como ejecutar el algoritmo de Shor sobre un número de 1024 bits. Hay una interacción significativa entre las Capas 2 y 3, debido a que la Capa 2 mejora la eficacia de la Capa 3. Finalmente, la Capa 4 llena los vacíos en el conjunto de compuertas suministrado por la Capa 3 para formar cualquier operación unitaria deseada a la exactitud arbitraria, proporcionando de este modo un sustrato completo para la computación cuántica universal en la Capa 5.

PCECO, una plataforma de hardware específico que fue introducida en este trabajo, demuestra el poder del concepto de la arquitectura multicapa, pero también pone de relieve una serie de prometedoras tecnologías para la computación cuántica, que son particularmente notables por las escalas de tiempo rápido de las operaciones cuánticas, el alto grado de integración posible con la fabricación de estado sólido, y la adopción de varias tecnologías maduras de otros campos de la Ingeniería. Los tiempos de ejecución para las operaciones cuánticas fundamentales son discutidos en la Sec.3.2.7, pero la importancia de estos procesos rápidos se aprecia claramente en la

Fig.3.15, donde la sobrecarga resultante de las compuertas virtuales en la Capa 2, CEC en la Capa 3, y las construcciones de compuertas en la Capa 4 incrementa el tiempo para implementar compuertas cuánticas de nanosegundos en la capa Física a milisegundos en la capa de Aplicación, o 6 órdenes de magnitud. En este contexto, una computadora cuántica necesita operaciones físicas muy rápidas.

Uno de los principales objetivos es entender mejor los recursos necesarios para construir una computadora cuántica que pueda resolver un problema intratable para las computadoras clásicas. Las figuras de mérito comunes para la evaluación de la tecnología de la computación cuántica son la fidelidad de la compuerta, el tiempo de operación, y el tiempo de coherencia del cubit. Esta investigación va más allá de mostrar cómo la conectividad y el rendimiento de control clásico son también cruciales. Diseñar una computadora cuántica requiere ver al sistema como un todo, de tal manera que las compensaciones y la compatibilidad entre componentes elegidos deben ser abordadas. Una foto holística es igualmente importante para comparar diferentes tecnologías de computación cuántica, tales como las trampas de iones o los circuitos superconductores. Este trabajo ilustra acerca de cómo aproximar el desafío completo de diseñar una computadora cuántica, para que se puedan adaptar estas técnicas al desarrollo de arquitecturas de otras tecnologías de computación cuántica que no hemos considerado aquí. Usando este diseño, diferentes sistemas propuestos pueden ser comparados en un marco común, que proporcione a los ingenieros cuánticos que aspiran a un lenguaje común para la determinación de la mejor tecnología de computación cuántica para una aplicación deseada.

Capítulo 4:
En este capítulo proponemos enfoques para la simulación de un computador cuántico ideal utilizando el modelo de programación CUDA en GPUs de NVIDIA, el cual ha sido analizado en este trabajo. Se puede señalar que la explotación explícita de la memoria rápida en el chip a través de latencia oculta, es un problema esencial cuando se persigue una alta eficiencia. Además, en este trabajo se desarrolla un marco de análisis que nos permite seleccionar los parámetros de simulación sobre algunos de los principales detalles de la plataforma.

Se han tenido en cuenta a tal efecto las características importantes como la localidad de referencia de memoria, el tamaño de la memoria en el chip y la ocupación de procesador. A partir de este análisis, las configuraciones para las simulaciones óptimas son experimentalmente obtenidas y evaluadas. Los resultados muestran que las estrategias propuestas son capaces de lograr altas aceleraciones en este tipo de multiprocesadores de fácil adquisición. En una GPU típica de 128 procesadores de transmisión un registro de 26-cubit puede ser simulado hasta casi cien veces más rápido que en código secuencial en una plataforma monoprocesador convencional contemporánea.

Capítulo 5:
En este capítulo presentamos los siguientes aportes:

1. el Proceso de Ortogonalización Booleano (POB) con su inversa,

2. el Proceso de Ortogonalización de Gram-Schmidt Mejorado (POGSMe) con su inversa,

3. el Proceso de Ortogonalización Cuántico (POCu) con su inversa,

4. la Red Ortogonalizadora Booleana Sistólica (ROBS),

5. la Red Ortogonalizadora Cuántica Sistólica (ROCS), y

6. una métrica que llamamos Tasa Dimensional de Entrada-Salida (TDES) la cual fue creada para monitorear el impacto del mejorador de la estabilidad del Proceso Ortogonalizador de Gram-Schmidt en el costo computacional final.

Los Aportes 1 a 5 permitieron la implementación de la primera Memoria Matricial Correlacionada Cuántica (MMCCu) eficiente, es decir, la MMCCu Mejorada (MMCCuMe) del Capítulo 6.

Es importante dejar de relieve que los Aportes 3 y 5 constituyen dos nuevos Algoritmos Cuánticos a nivel de Capa de Aplicación, que viene a completar las herramientas necesarias a efectos de implementar finalmente el Procesamiento Cuántico de Señales e Imágenes, impensables sin un explícito algoritmo de ortogonalización eficiente en dicho ámbito.

Capítulo 6:
Presentamos aquí los siguientes aportes:

7. una versión mejorada de las ya conocidas Memorias Matriciales Correlacionadas Boolenas (MMCB), es decir, la MMCB mejorada (MMCBMe) en base al innovador Proceso de Ortonormalización Booleano (POB) del capítulo anterior,

8. la Memoria Matricial Correlacionada Cuántica (MMCCu), y

9. la MMCCu Mejorada (MMCCuMe) en base al Proceso de Ortogonalización Cuántico (POCu) implementado en forma escalonada y conocida como la Red Ortogonalizadora Cuántica Sistólica (ROCS) en el capítulo anterior.

Mientras este último aporte se constituye en la última Red Neuronal Cuántica (RNCu), completamos el conjunto de herramientas tendientes a:

a) el arribo de la representación subsimbólica del conocimiento (provenientes de la Inteligencia Artificial) al ámbito cuántico, y

b) una mejora en los procedimientos conducentes al reconocimiento de patrones en dicho ámbito.

Capítulo 7:
Se presentan aquí 6 simulaciones, La primeras 3 a partir de los patrones cuánticos generados con el emulador QNNLab de la Universidad de Oxford. En base a los mismos se simularon procesos de almacenamiento sobre la MMCCu desarrollada en el Capítulo 6 con y sin la ortonormalización cuántica desarrollada en el Capítulo 5. Los resultados son elocuentes en lo concerniente a los errores de recuperación, que es otra forma de expresar el aumento de la performance de memorización gracias a la superación del ruido en base al empleo de la ortonormalización cuántica.

Las últimas 3 simulaciones, en cambio, tienen que ver con aplicaciones de las herramientas mencionadas al Procesamiento Cuántico de Imágenes multimediales, médicas (tomografías y

resonancias seriadas), satelitales (multi e hiper-espectrales) y aquellas relativas a documentos de Artes Electrónicas.

Finalmente, el contenido de este capítulo es en si mismo un aporte al área del conocimiento en cuestión, por cuanto es la primera vez en la literatura que se realizan mediante la tecnología cuántica e implementadas sobre GPGPU simulaciones de MMCCu y MMCB, así como aquellas relativas al Procesamiento Cuántico de Imágenes y además con la performance alcanzada.

Conclusiones Finales:
En este trabajo se han presentado 10 innovaciones que van desde nuevos algoritmos cuánticos, pasando por nuevos procedimientos de ortogonalización de arreglos vectoriales cuánticos, hasta una nueva red neuronal cuántica para el reconocimiento de patrones en este ámbito, sin dejar de mencionar una colección práctica de simulaciones en el ámbito de la nueva disciplina conocida como Procesamiento Cuántico de Imágenes.

Trabajos Futuros y Áreas Abiertas:
Una primera línea de investigación que surge automáticamente de este trabajo tienen que ver con el desarrollo de la versión cuántica de los otros tipo de redes neuronales [213], además de los ya implementados modelos de Behrman *et al.*, Chrisley, Menneer-Narayanan y Ventura [250] y las memorias matriciales correlacionadas [212]. Estas implementaciones proveerán al ámbito de la Computación Cuántica de una herramienta para el reconocimiento de patrones cuánticos, así como para el uso en Inteligencia Artificial Cuántica, y motores no lineales de búsqueda como en el caso de los requeridos para los procesos del tipo Descubrimiento del Conocimiento a partir de los Datos (en inglés, Knowledge Discovery Data) con el uso de Minería de Datos (en inglés, Data-Mining) y Reservorio de Datos (en inglés, Data-Warehousing).

Una segunda línea de investigación concierne a la proliferación del Procesamiento Cuántico de Imágenes en todas sus formas, es decir, filtrado, mejoramiento, restauración, compresión y super-resolución cuántico de imágenes, siendo las mismas multimediales, satelitales, médica, documentales, biométricas, entre otras. Además, es importante explotar a futuro mediante esta tecnología una área del procesamiento de imágenes conocido como Morfología Matemática, en virtud del parangón entre las unidades elementales de información de esta última, es decir, los bits y los cubits de la tecnología cuántica para el caso puro y ortogonal.

Glosario: Acrónimos

Capítulo 1:	
Bra “| >”	Vector Columna
CHSH	Clauser-Horne-Shimony-Holt
EPR	Einstein-Podolsky-Rosen
Ket “< |”	Vector Fila
LOCC	Operaciones Locales y Comunicaciones Clásicas
Capítulo 2:	
AND	Salida 1 si y solo si ambas entradas son 1
Cbit	Bit Clásico
CMINUS	Giro de π
CNOT	NO-controlada
COPY	Convierte un bit en dos iguales
CPHASE	Corrimiento de fase al cubit objetivo cuando el cubit de control está en |1>
mcd	máximo común dividor
mod	módulo
MT	Máquina de Turing
MTC	Máquina de Turing Cuántica
MUT	Máquina Universal de Turing
MUTC	Máquina Universal de Turing Clásica
NAND	Produce salida 0 si y solo si ambas entradas son 1
NOT	Inversor Lógico
OR	Salida 0 si y solo si ambas entradas son 0
OR exclusiva	Salida 1 solo si ambas entradas son diferentes
TFD	Transformada de Fourier Discreta
Capítulo 3:	
CEC	Corrección de Error Cuántico
CP	Carr-Purcell
DD	Desacoplamiento Dinámico
DDU	Desacoplamiento Dinámico de Uhrig
EDC	Electro-Dinámica Cuántica
EPCV	Error Por Compuerta Virtual
GC	Grupo de Clifford
MAAD	Memorias de Acceso Aleatorias Dinámicas
MEO	Modulador Electro-Óptico
NDC	No Demolición Cuántica
PC	Punto Cuántico
PCECO	Puntos Cuánticos con Espines Controlados Ópticamente
RBD	Reflector-de-Bragg-Distribuido
RMN	Resonancia Magnética Nuclear
RSA	Rivest-Shamir- Adleman
SMEM	Sistemas Micro-Electro-Mecánicos
TF	Tolerancia a Fallas

TFC	Transformada de Fourier Cuántica
UAL	Unidad Aritmético-Lógica
Capítulo 4:	
CC	Computadora Cuántica
CPT	Clúster del Procesador de Textura
CPU	Central Processing Unit
CrayT3E	Supercomputador de segunda generación de Cray Research
CrayX1E	Supercomputador de procesador escalar y memoria de acceso no uniforme fabricado y vendido por Cray Inc.
CUDA	Compute Unified Device Architecture
FPGA	Field Programmable Gate Array
GeForce	Modelo de GPU de NVIDIA
GPU	Graphics Processing Units
GPGPU	General-Purpose GPU
Hitachi SR11000	Supercomputador con sistema operativo AIX
IBM	International Business Machines
IBM Blue Gene/L	Supercomputador de la serie Blue Gene desarrollado en colaboración entre IBM y Lawrence Livermore National Laboratory
IBM P690	Servidor UNIX durante la época de los procesadores POWER4
IBM Regatta P690	Supercomputador paralelo con procesadores POWER4
IBM SP2	RISC System/6000 Escalable, Sistema POWERparallel
MH	Multi Hilo
MP	Multiprocesadores
MPI	Message Passing Interface
MT	Multiprocesador de Transmisión
NVIDIA	Empresa fabricante de GPUs y creadora de CUDA
OpenMP	API para programación paralela memoria-compartida multiplataforma en C/C++ y Fortran.
PT	Procesador de Transmisión
SIMT	Single-Instruction Multiple-Thread
SPEC	Standard Performance Evaluation Corporation
SWAPS	Enroques
UltraSPARCII	Implementación del microprocesador de la SPARC V9 con Arquitectura de Conjunto de Instrucciones (ISA)
USF	Unidad de Super Función
XOR	OR exclusiva
Capítulo 5:	
ME	Mejora de la Estabilidad
MMC	Memoria Matricial Correlacionada
IME	Inversa de la Mejora de la Estabilidad
IPOGS	Inversa del Proceso de Ortogonalización de Gram-Schmidt
IPOGSMe	Inversa del Proceso de Ortogonalización de Gram-Schmidt Mejorado
PO	Prueba de Ortogonalidad
POB	Proceso de Ortogonalización Booleano

POCl	Proceso de Ortogonalización Clásico
POCu	Proceso de Ortogonalización Cuántico
POGS	Proceso de Ortogonalización de Gram-Schmidt
POGSMe	Proceso de Ortogonalización de Gram-Schmidt Mejorado
POGSMo	Proceso de Ortogonalización de Gram-Schmidt Modificado
ROBS	Red Ortogonalizadora Booleana Sistólica
ROCS	Red Ortogonalizadora Cuántica Sistólica
TDES	Tasa Dimensional de Entrada-Salida
VLSI	Very-Large-Scale Integration
Capítulo 6:	
MAB	Memoria Asociativa Booleana
MACl	Memoria Asociativa Clásica
MACu	Memoria Asociativa Cuántica
MACuL	MACu Lineal
MACuNL	MACu No Lineal
MMB	Matriz de Memoria Booleana
MMCB	Memoria Matricial Correlacionada Booleana
MMCl	Matriz de Memoria Clásica
MMCBMe	Memoria Matricial Correlacionada Booleana Mejorada
MMCCl	Memoria Matricial Correlacionada Clásica
MMCClMe	Memoria Matricial Correlacionada Clásica Mejorada
MMCCu	Memoria Matricial Correlacionada Cuántica
MMCCuMe	Memoria Matricial Correlacionada Cuántica Mejorada
Capítulo 7:	
PMCu	Perceptrones Multicapa Cuánticos
RNCu	Red Neuronal Cuántica

NOTA: Todos los acrónimos son compatibles para atrás y no se repiten.

Bibliografía

Capítulo 1:

[1] **Mendoza Vázquez D. G.**, *Introducción a la Teoría de la Información Cuántica*, tesis de Licenciatura en Ciencias de la Computación, Universidad Nacional Autónoma de México, México D.F., 2010.

[2] **Sansone, G**. "Elementary Notions of Hilbert Space." §1.3 in Orthogonal Functions, rev. English ed. New York: Dover, pp. 5-10, 1991.

[3] **Cohen-Tannoudji C., Liu B. y Laloe F.**, "Quantum Mechanics", Vol I, 1977.

[4] **Young T.**, "Experiments and Calculations Relative to Physical Optics", Abstracts of the Papers Printed in the Philosophical Transactions of the Royal Society of London, Volume 1, pp. 131-132.

[5] **Clauser J. F., Horne M.A., Shimony A. y Holt R. A.**, "Proposed experiment to test local hidden-variable theories", Phys. Rev. Lett. 23, 880-884 (1969).

[6] **Vedral V., Plenio M. B., Rippin M. A., y Knight P. L.**, Optics Section, "Quantifying Entanglement", American Physical Society, p. 2275–2279, 1997.

Capítulo 2:

[7] **Mermin N. D.**, "Quantum Computer Science, An Introduction", Cambridge University Press, 2007.

[8] **Rosinger E. E.**, "Basics of Quantum Computation (Part 1)", University of Pretoria, arXiv:quant-ph/0407064v1.

[9] **Casati B. G. y Strini G.**, "Principles of Quantum Computation and Information Vol. 1", World Scientific, 2004.

[10] **Ekert A. y Jozsa R.**, "Quantum Computation and Shor´s Factorithm Algorithm, Review of Modern Phiscs", 68, (1996) 733-753.

[11] **Shor P. W.**, "Algorithms for quantum computation: discrete logarithms and factoring". Proc 35th Ann. Sym. on found of Comp. Sci., (1994) 124-134.

[12] **Deutsch D.**, "Quantum Theory, The Church-Turing principle and the universal quantum computing", Proceeds of the Royal Society of London, 400, pp. 97-117 (1985).

[13] **Nielsen M. A. y Chuang I. L.**, Physics Review Letters 79, 321 (1997).

[14] **Berstein, Vazirani**. "Quantum Complexity Theory", Proceedings of the 25th Annual ACM Symposium on Theory of Computing.

[15] **Rieffel E. G. y Polak W.**, "An Introduction to Quantum Computing for Non-Physicists", FX Palo Alto Labratory, arXiv:quant-ph/9809016v2.

[16] **Fouché W., Heidema J. y Jones G.**, "Deutsch´s Universal Quantum Turing Machine (Revisted)", arXiv:quant-ph/0701108v1.

Capítulo 3:

[17] **Cody Jones N., Van Meter R., Fowler A. G., Mcmahon P. L., Kim J., Ladd T. D., y Yamamoto Y.**, Layered Architecture for Quantum Computing, Physical Review X, 2, 031007 (2012).

[18] **Ladd T. D., Jelezko F., Laflamme R., Nakamura Y., Monroe C., y O'Brien J. L.**, Quantum Computers, Nature (London) 464, 45 (2010).

[19] **Fowler A. G., Stephens A. M., y Groszkowski P.**, High-Threshold Universal Quantum Computation on the Surface Code, Phys. Rev. A 80, 052312 (2009).

[20] **DiVincenzo D. P.**, The Physical Implementation of Quantum Computation, Fortschr. Phys. 48, 771 (2000).

[21] **Steane A. M.**, Quantum Computer Architecture for Fast Entropy Extraction, Quantum Inf. Comput. 2, 297 (2002) [http://dl.acm.org/citation.cfm?id=2011480].

[22] **Steane A. M.**, How to Build a 300 Bit, 1 Giga-Operation Quantum Computer, Quantum Inf. Comput. 7, 171 (2007) [http://dl.acm.org/citation.cfm?id=2011718].

[23] **Spiller T. P., Munro W. J., Barrett S. D. y Kok P.**, An Introduction to Quantum Information Processing: Applications and Realizations, Contemp. Phys. 46, 407 (2005).

[24] **Van Meter R. y Oskin M.**, Architectural Implications of Quantum Computing Technologies, ACM Journal on Emerging Technologies in Computing Systems 2, 31 (2006).

[25] **Taylor J. M., Engel H.-A., Dür W., Yacoby A., Marcus C. M., Zoller P., y Lukin M. D.**, Fault-Tolerant Architecture for Quantum Computation Using Electrically Controlled Semiconductor Spins, Nature Phys. 1, 177 (2005).

[26] **Steane A. M.**, Space, Time, Parallelism and Noise Requirements for Reliable Quantum Computing, Fortschr. Phys. 46, 443 (1998).

[27] **Isailovic N., Whitney M., Patel Y., y Kubiatowicz J.**, in 35th International Symposium on Computer Architecture, Beijing, China 2008 (ISCA '08) (IEEE Conference Publications, Los Alamitos, 2008), p. 177.

[28] **Metodi T. S., Thaker D. D., Cross A. W., Chong F. T., y Chuang I. L.**, in MICRO-38: Proceedings of the 38th Annual International Symposium on Microarchitecture (IEEE Computer Society, Los Alamitos, 2005), p. 305.

[29] **Kielpinski D., Monroe C., y Wineland D.**, Architecture for a Large-Scale Ion-Trap Quantum Computer, Nature (London) 417, 709 (2002).

[30] **Copsey D., Oskin M., Metodiev T., Chong F. T., Chuang I., y Kubiatowicz J.**, in Proceedings of the Fifteenth Annual ACM Symposium on Parallel Algorithms and Architectures (SPAA '03) (ACM, New York, 2003), p. 65.

[31] **Svore K. M., Aho A.V., Cross A.W., Chuang I., y Markov I. L.**, A Layered Software Architecture for Quantum Computing Design Ttools, Computer 39, 74 (2006).

[32] **Oskin M., Chong F. T., Chuang I. L., y Kubiatowicz J.**, in 30th International Symposium on Computer Architecture, 2003 (ISCA '03) (IEEE Conference Publications, San Diego, California, 2003), p. 374.

[33] **Kane B. E.**, A Silicon-Based Nuclear Spin Quantum Computer, Nature (London) 393, 133 (1998).

[34] **Mariantoni M., Wang H., Yamamoto T., Neeley M., Bialczak R. C., Chen Y., Lenander M., Lucero E., O'Connell A. D., Sank D., Weides M., Wenner J., Yin Y., Zhao J., Korotkov A. N., Cleland A. N., y Martinis J. M.**, Implementing the Quantum von Neumann Architecture with Superconducting Circuits, Science 334, 61 (2011).

[35] **Cirac J. I., Zoller P., Kimble H. J., y Mabuchi H.**, Quantum State Transfer and Entanglement Distribution among Distant Nodes in a Quantum Network, Phys. Rev. Lett. 78, 3221 (1997).

[36] **van Enk S. J., Kimble H. J., Cirac J. I., y Zoller P.**, Quantum Communication with Dark Photons, Phys. Rev. A 59, 2659 (1999).

[37] **Steane A. M. y Lucas D. M.**, Quantum Computing with Trapped Ions, Atoms and Light, Fortschr. Phys. 48, 839 (2000).

[38] **Duan L.-M., Lukin M. D., Cirac J. I., y Zoller P.**, Long-Distance Quantum Communication with Atomic Ensembles and Linear Optics, Nature (London) 414, 413 (2001).

[39] **Van Meter R., Nemoto K., y Munro W. J.**, Communication Links for Distributed Quantum Computation, IEEE Trans. Comput. 56, 1643 (2007).

[40] **Duan L.-M. y Monroe C.**, Colloquium: Quantum Networks with Trapped Ions, Rev. Mod. Phys. 82, 1209 (2010).

[41] **Kim J. y Kim C.**, Integrated Optical Approach to Trapped Ion Quantum Computation, Quantum Inf. Comput. 9, 181 (2009) [http://dl.acm.org/citation.cfm?id=2011782].

[42] **Fowler A. G., Thompson W. F., Yan Z., Stephens A. M., Plourde B. L. T., y Wilhelm F. K.**, Long-Range Coupling and Scalable Architecture for Superconducting Flux Qubits, Phys. Rev. B 76, 174507 (2007).

[43] **Whitney M., Isailovic N., Patel Y., y Kubiatowicz J.**, in Proceedings of the 4th International Conference on Computing Frontiers (CF '07) (ACM, New York, 2007), p. 83.

[44] **Whitney M., Isailovic N., Patel Y., y Kubiatowicz J.**, in 36th International Symposium on Computer Architecture 2009 (ISCA '09) (ACM, Austin, Texas, 2009), p, 383.

[45] **Isailovic N., Patel Y., Whitney M., y Kubiatowicz J.**, in 33rd International Symposium on Computer Architecture, 2006 (ISCA'06) (IEEE Conference Publications, Boston, Massachusetts, 2006), p. 366.

[46] **Gottesman D. y Chuang I. L.**, Demonstrating the Viability of Universal Quantum Computation Using Teleportation and Single-Qubit Operations, Nature (London) 402, 390 (1999).

[47] **Levy J.**, Quantum-Information Processing with Ferroelectrically Coupled Quantum Dots, Phys. Rev. A 64, 052306 (2001).

[48] **Fowler A. G., Devitt S. J., y Hollenberg L. C. L.**, Implementation of Shor's Algorithm on a Linear Nearest Neighbour Qubit Array, Quantum Inf. Comput. 4, 237 (2004) [http://dl.acm.org/citation.cfm?id=2011828].

[49] **Aliferis P. y Cross A. W.**, Subsystem Fault Tolerance with the Bacon-Shor Code, Phys. Rev. Lett. 98, 220502 (2007).

[50] **Levy J. E., Ganti A., Phillips C. A., Hamlet B. R., Landahl A. J., Gurrieri T. M., Carr R. D., y Carroll M. S.**, The Impact of Classical Electronics Constraints on a Solid-State Logical Qubit Memory, arXiv:0904.0003v1.

[51] **Levy J. E., Carroll M. S., Ganti A., Phillips C. A., Landahl A. J., Gurrieri T. M., Carr R. D., Stalford H. L., y Nielsen E.**, Implications of Electronics Constraints for Solid-State Quantum Error Correction and Quantum Circuit Failure Probability, New J. Phys. 13, 083021 (2011).

[52] **Weinstein Y. S., Hellberg C. S., y Levy J.**, Quantum-Dot Cluster-State Computing with Encoded Qubits, Phys. Rev. A 72, 020304 (2005).

[53] **Stock R. y James D. F. V.**, Scalable, High-Speed Measurement-Based Quantum Computer Using Trapped Ions, Phys. Rev. Lett. 102, 170501 (2009).

[54] **Devitt S. J., Fowler A. G., Tilma T., Munro W. J., y Nemoto K.**, Classical Processing Requirements for a Topological Quantum Computing System, Int. J. Quant. Info. 08, 121 (2010).

[55] **Devitt S. J., Stephens A. M., Munro W. J., y Nemoto K.**, Integration of Highly Probabilistic Sources into Optical Quantum Architectures: Perpetual Quantum Computation, New J. Phys. 13, 095001 (2011).

[56] **Kitaev A. Yu., Shen A. H., y Vyalyi M. N.**, Classical and Quantum Computation (American Mathematical Society, Providence, 2002), 1st ed.

[57] **Oskin M., Chong F. T., y Chuang I. L.**, A Practical Architecture for Reliable Quantum Computers, Computer 35, 79 (2002).

[58] **Devitt S. J., Munro W. J., y Nemoto K.**, High Performance Quantum Computing, Progr. Informat. 8, 49 (2011).

[59] **Van Meter R., Nemoto K., Munro W. J., y Itoh K. M.**, Distributed Arithmetic on a Quantum Multicomputer, Computer Architecture News 34, 354 (2006).

[60] **Van Meter R., Ladd T. D., Fowler A. G., y Yamamoto Y.**, Distributed Quantum Computation Architecture Using Semiconductor Nanophotonics, Int. J. Quant. Info. 08, 295 (2010).

[61] **Preskill J.**, Fault-Tolerant Quantum Computation, arXiv:quant-ph/9712048.

[62] **Shen J. P. y Lipasti M. H.**, Modern Processor Design: Fundamentals of Superscalar Processors (McGraw-Hill Higher Education, New York, 2005).

[63] **Björk G., Pau S., Jacobson J., y Yamamoto Y.**, Wannier Exciton Superradiance in a Quantum-Well Microcavity, Phys. Rev. B 50, 17336 (1994).

[64] **Imamoglu A., Awschalom D. D., Burkard G., DiVincenzo D. P., Loss D., Sherwin M., y Small A.**, Quantum Information Processing Using Quantum Dot Spins and Cavity QED, Phys. Rev. Lett. 83, 4204 (1999).

[65] **Bonadeo N. H., Chen G., Gammon D., y Steel D. G.**, Single Quantum Dot Nonlinear Optical Spectroscopy, Phys. Status Solidi B 221, 5 (2000).

[66] **Guest J. R., Stievater T. H., Li X., Cheng J., Steel D. G., Gammon D., Katzer D. S., Park D., Ell C., Thränhardt A., Khitrova G., y Gibbs H. M.**, Measurement of Optical Absorption by a Single Quantum Dot Exciton, Phys. Rev. B 65, 241310 (2002).

[67] **Hours J., Senellart P., Peter E., Cavanna A., y Bloch J.**, Exciton Radiative Lifetime Controlled by the Lateral Confinement Energy in a Single Quantum Dot, Phys. Rev. B 71, 161306 (2005).

[68] **Yamamoto Y., Ladd T. D., Press D., Clark S., Sanaka K., Santori C., Fattal D., Fu K.-M., Höfling S., Reitzenstein S., y Forchel A.**, Optically Controlled Semiconductor Spin Qubits for Quantum Information Processing, Phys. Scr. T137, 014010 (2009).

[69] **Bayer M., Ortner G., Stern O., Kuther A., Gorbunov A. A., Forchel A., Hawrylak P., Fafard S., Hinzer K., Reinecke T. L., Walck S. N., Reithmaier J. P., Klopf F., y Schäfer F.**, Fine Structure of Neutral and Charged Excitons in Self-Assembled In(Ga)As=(Al)GaAs Quantum Dots, Phys. Rev. B 65, 195315 (2002).

[70] **Reitzenstein S., Hofmann C., Gorbunov A., Strauß M., Kwon S. H., Schneider C., Löffler A., Höfling S., Kamp M., y Forchel A.**, AlAs=GaAs Micropillar Cavities with Quality Factors Exceeding 150.000, Appl. Phys. Lett. 90, 251109 (2007).

[71] **Nielsen M. A. y Chuang I. L.**, Quantum Computation and Quantum Information, Cambridge University Press, Cambridge, England, 2004.

[72] **Press D., Ladd T. D., Zhang B., y Yamamoto Y.**, Complete Quantum Control of a Single Quantum Dot Spin Using Ultrafast Optical Pulses, Nature (London) 456, 218 (2008).

[73] **De Greve K., McMahon P. L., Cody Jones N., Schneider C., Kamp M., Hoefling S., y Yamamoto Y.**, All-Optical, Ultrafast Hadamard Gates: High-Fidelity, Direct Implementation of a Quantum Information Primitive (unpublished).

[74] **Economou S. E., Sham L. J., Wu Y., y Steel D. G.**, Proposal for Optical U(1) Rotations of Electron Spin Trapped in a Quantum Dot, Phys. Rev. B 74, 205415 (2006).

[75] **Clark S. M., Fu K.-M. C., Ladd T. D., y Yamamoto Y.**, Quantum Computers Based on Electron Spins Controlled by Ultrafast Off- Resonant Single Optical Pulses, Phys. Rev. Lett. 99, 040501 (2007).

[76] **Ladd T. D. y Yamamoto Y.**, Simple Quantum Logic Gate with Quantum Dot Cavity QED Systems, Phys. Rev. B 84, 235307 (2011).

[77] **Quinteiro G. F., Fernández-Rossier J., y Piermarocchi C.**, Long-Range Spin-Qubit Interaction Mediated by Microcavity Polaritons, Phys. Rev. Lett. 97, 097401 (2006).

[78] **Berezovsky J., Mikkelsen M. H., Gywat O., Stoltz N. G., Coldren L. A., y Awschalom D. D.**, Nondestructive Optical Measurements of a Single Electron Spin in a Quantum Dot, Science 314, 1916 (2006).

[79] **Atatüre M., Dreiser J., Badolato A., y Imamoglu A.**, Observation of Faraday Rotation from a Single Confined Spin, Nature Phys. 3, 101 (2007).

[80] **Fushman I., Englund D., Faraon A., Stoltz N., Petroff P., y Vuckovic J.**, Controlled Phase Shifts with a Single Quantum Dot, Science 320, 769 (2008).

[81] **Young A. B., Oulton R., Hu C. Y., Thijssen A. C. T., Schneider C., Reitzenstein S., Kamp M., Höfling S., Worschech L., Forchel A., y Rarity J. G.**, Quantum-Dot-Induced Phase Shift in a Pillar Microcavity, Phys.Rev. A 84, 011803 (2011).

[82] **Press D., De Greve K., McMahon P. L., Ladd T. D., Friess B., Schneider C., Kamp M., Höfling S., Forchel A., y Yamamoto Y.**, Ultrafast Optical Spin Echo in a Single Quantum Dot, Nature Photon. 4, 367 (2010).

[83] **Aliferis P. y Preskill J.**, Fault-Tolerant Quantum Computation against Biased Noise, Phys. Rev. A 78, 052331 (2008).

[84] **Lidar D. A., Chuang I. L., y Whaley K. B.**, Decoherence-Free Subspaces for Quantum Computation, Phys. Rev. Lett. 81, 2594-2597 (1998).

[85] **Lidar D. A. y Whaley K. B.**, in Irreversible Quantum Dynamics, Lecture Notes in Physics, Vol. 622, edited by F. Benatti and R. Floreanini (Springer, Berlin, 2003), p. 83 [http://www.springerlink.com/content/m26k4533p0v85331/].

[86] **Grace M., Brif C., Rabitz H., Walmsley I., Kosut R., y Lidar D.**, Encoding a Qubit into Multilevel Subspaces, New J. Phys. 8, 35 (2006).

[87] **Hahn E. L.**, Spin echoes, Phys. Rev. 80, 580 (1950).

[88] **Viola L., Knill E., y Lloyd S.**, Dynamical Decoupling of Open Quantum Systems, Phys. Rev. Lett. 82, 2417 (1999).

[89] **Khodjasteh K. y Lidar D. A.**, Fault-Tolerant Quantum Dynamical Decoupling, Phys. Rev. Lett. 95, 180501 (2005).

[90] **Biercuk M. J., Uys H., VanDevender A. P., Shiga N., Itano W. M., y Bollinger J. J.**, Optimized Dynamical Decoupling in a Model Quantum Memory, Nature (London) 458, 996 (2009).

[91] **Viola L. y Knill E.**, Robust Dynamical Decoupling of Quantum Systems with Bounded Controls, Phys. Rev. Lett. 90, 037901 (2003).

[92] **Ng H. K., Lidar D. A., y Preskill J.**, Combining Dynamical Decoupling with Fault-Tolerant Quantum Computation, Phys. Rev. A 84, 012305 (2011).

[93] **Carr H. Y. y Purcell E. M.**, Effects of Diffusion on Free Precession in Nuclear Magnetic Resonance Experiments, Phys. Rev. 94, 630 (1954).

[94] **Haeberlen U. y Waugh J. S.**, Coherent Averaging Effects in Magnetic Resonance, Phys. Rev. 175, 453 (1968).

[95] **Uhrig G. S.**, Keeping a Quantum Bit Alive by Optimized π-Pulse Sequences, Phys. Rev. Lett. 98, 100504 (2007).

[96] **Brown K. R., Harrow A. W., y Chuang I. L.**, Arbitrarily Accurate Composite Pulse Sequences, Phys. Rev. A 70, 052318 (2004).

[97] **Tomita Y., Merrill J. T., y Brown K. R.**, Multi-Qubit Compensation Sequences, New J. Phys. 12, 015002 (2010).

[98] **Khodjasteh K. y Viola L.**, Dynamically Error-Corrected Gates for Universal Quantum Computation, Phys. Rev. Lett. 102, 080501 (2009).

[99] **Cappellaro P., Emerson J., Boulant N., Ramanathan C., Lloyd S., y Cory D. G.**, Entanglement Assisted Metrology, Phys. Rev. Lett. 94, 020502 (2005).

[100] **Elzerman J. M., Hanson R., van Beveren L. H. W., Witkamp B., Vandersypen L. M. K., y Kouwenhoven L. P.**, Single-Shot Read-Out of an Individual Electron Spin in a Quantum Dot, Nature (London) 430, 431 (2004).

[101] **Kroutvar M., Ducommun Y., Heiss D., Bichler M., Schuh D., Abstreiter G., y Finley J. J.**, Optically Programmable Electron Spin Memory Using Semiconductor Quantum Dots, Nature (London) 432, 81 (2004).

[102] **Aharonov D. y Ben-Or M.**, in Proceedings of the Twenty-Ninth Annual ACM Symposium on Theory of Computing (STOC ’97) (ACM, New York, 1997), p. 176.

[103] **Preskill J.**, Reliable Quantum Computers, Proc. R. Soc. A 454, 385 (1998).

[104] **Devitt S. J., Nemoto K., y Munro W. J.**, The Idiots Guide to Quantum Error Correction, arXiv:0905.2794.

[105] **Gottesman D.**, Ph.D. thesis, California Institute of Technology, Pasadena, CA, 1997.

[106] **Raussendorf R. y Harrington J.**, Fault-Tolerant Quantum Computation with High Threshold in Two Dimensions, Phys. Rev. Lett. 98, 190504 (2007).

[107] **Raussendorf R, Harrington J, y Goyal K.**, Topological Fault-Tolerance in Cluster State Quantum Computation, New J. Phys. 9, 199 (2007).

[108] **Knill E.**, Quantum Computing with Realistically Noisy Devices, Nature (London) 434, 39 (2005).

[109] **Aliferis P., Gottesman D., y Preskill J.**, Accuracy Threshold for Postselected Quantum Computation, Quantum Inf. Comput. 8, 181 (2008) [http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.148.2937].

[110] **Bacon D.**, Operator Quantum Error-Correcting Subsystems for Self-Correcting Quantum Memories, Phys. Rev. A 73, 012340 (2006).

[111] **Fowler A. G., Wang D. S., y Hollenberg L. C. L.**, Surface Code Quantum Error Correction Incorporating Accurate Error Propagation, Quantum Inf. Comput. 11, 8 (2011) [http://dl.acm.org/citation.cfm?id=2011385].

[112] **Wang D. S., Fowler A. G., y Hollenberg L. C. L.**, Surface Code Quantum Computing with Error Rates over 1%, Phys. Rev. A 83, 020302 (2011).

[113] **Fowler A. G., Whiteside A. C., y Hollenberg L. C. L.**, Towards Practical Classical Processing for the Surface Code, Phys. Rev. Lett. 108, 180501 (2012).

[114] **Aliferis P.**, Ph.D. thesis, California Institute of Technology, Pasadena, CA, 2007.

[115] **Shor P. W.**, Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer, SIAM J. Comput. 26, 1484 (1997).

[116] **Aspuru-Guzik A., Dutoi A. D., Love P. J., y Head-Gordon M.**, Simulated Quantum Computation of Molecular Energies, Science 309, 1704 (2005).

[117] **Kassal I., Jordan S. P., Love P. J., Mohseni M., y Aspuru-Guzik A.**, Polynomial-Time Quantum Algorithm for the Simulation of Chemical Dynamics, Proc. Natl. Acad. Sci. U.S.A. 105, 18681 (2008).

[118] **Van Meter R. y Itoh K. M.**, Fast Quantum Modular Exponentiation, Phys. Rev. A 71, 052320 (2005).

[119] **DiVincenzo D. P. y Aliferis P.**, Effective Fault-Tolerant Quantum Computation with Slow Measurements, Phys. Rev. Lett. 98, 020501 (2007).

[120] **Anders S. y Briegel H. J.**, Fast Simulation of Stabilizer Circuits Using a Graph-State Representation, Phys. Rev. A 73, 022334 (2006).

[121] **Bravyi S. y Kitaev A.**, Universal Quantum Computation with Ideal Clifford Gates and Noisy Ancillas, Phys. Rev. A 71, 022316 (2005).

[122] **Cody Jones N., Whitfield J. D., McMahon P. L., Yung M.-H., Van Meter R., Aspuru-Guzik A., y Yamamoto Y.**, Simulating Chemistry Efficiently on Fault-Tolerant Quantum Computers, arXiv:1204.0567.

[123] **Fowler A. G.**, Constructing Arbitrary Steane Code Single Logical Qubit Fault-Tolerant Gates, Quantum Inf. Comput. 11, 867 (2011) [http://dl.acm.org/citation.cfm?id=2230946].

[124] **Dawson C. M. y Nielsen M. A.**, The Solovay-Kitaev Algorithm, Quantum Inf. Comput. 6, 81 (2006) [http://dl.acm.org/citation.cfm?id=2011685].

[125] **Yu. Kitaev A.**, Quantum Measurements and the Abelian Stabilizer Problem, arXiv:quant-ph/9511026v1.

[126] **Cleve R., Ekert A., Macchiavello C., y Mosca M.**, Quantum Algorithms Revisited, Proc. R. Soc. A 454, 339 (1998).

[127] **Vedral V., Barenco A., y Ekert A.**, Quantum Networks for Elementary Arithmetic Operations, Phys. Rev. A 54, 147 (1996).

[128] **Cuccaro S. A., Draper T. G., Kutin S. A., y Moulton D. P.**, A New Quantum Ripple-Carry Addition Circuit, arXiv:quant-ph/0410184.

[129] **Draper T. G., Kutin S. A., Rains E. M., y Svore K. M.**, A Logarithmic-Depth Quantum Carry-Lookahead Adder, Quantum Inf. Comput. 6, 351 (2006) [http://dl.acm.org/citation.cfm?id=2012090].

[130] **Beauregard S.**, Circuit for Shor's Algorithm Using 2*n*+3 Qubits, Quantum Inf. Comput. 3, 175 (2003) [http://dl.acm.org/citation.cfm?id=2011525].

[131] **Zalka C.**, Shor's Algorithm with Fewer (Pure) Qubits, arXiv:quant-ph/0601097.

[132] **Takahashi Y. y Kunihiro N.**, A Quantum Circuit for Shor's Factoring Algorithm Using 2*n*+2 Qubits, Quantum Inf. Comput. 6, 184 (2006) [http://dl.acm.org/citation.cfm?id=2011669].

[133] **Rivest R. L., Shamir A., y Adleman L.**, A Method for Obtaining Digital Signatures and Public-Key Cryptosystems, Commun. ACM 21, 120 (1978).

[134] **Feynman R.**, Simulating Physics with Computers, Int. J. Theor. Phys. 21, 467 (1982).

[135] **Buluta I. y Nori F.,** Quantum Simulators. *Science*, 326(5949):108–111, 2009.

[136] **Barreiro J. T., Müller M., Schindler P., Nigg D., Monz T., Chwalla M., Hennrich M., Roos C. F., Zoller P., y Blatt R.**, An Open-System Quantum Simulator with Trapped Ions, Nature (London) 470, 486 (2011).

[137] **Biamonte J. D., Bergholm V., Whitfield J. D., Fitzsimons J., y Aspuru-Guzik A.**, Adiabatic Quantum Simulators, AIP Adv. 1, 022126 (2011).

[138] **Zalka C.**, Simulating Quantum Systems on a Quantum Computer, Proc. R. Soc. A 454, 313 (1998).

[139] **Kassal I., Whitfield J. D., Perdomo-Ortiz A., Yung M.-H., y Aspuru-Guzik A.**, Simulating Chemistry Using Quantum Computers, Annu. Rev. Phys. Chem. 62, 185 (2011).

[140] **Fowler A. G., Whiteside A. C., y Hollenberg L. C. L.**, Towards Practical Classical Processing for the Surface Code: Timing Analysis, arXiv: 1202.5602.

[141] **Oskin M., Chong F. T., y Chuang I. L.**, A Practical Architecture for Reliable Quantum Computers, IEEE Computer Magazine, pp.79-87, Jan. 2002

[142] **Bennett C. H. et al.**, "Teleporting an Unknown Quantum State via Dual Classical and EPR Channels," *Physical Rev. Letters*, vol. 70, 1993, pp. 1895-1899.

[143] **Bouwmeester D. *et al.***, "Experimental Quantum Teleportation," *Nature*, vol. 390, 1997, pp. 575-579.

[144] **Kane B.**, "A Silicon-Based Nuclear Spin Quantum Computer," *Nature*, vol. 393, 1998, pp. 133-137.

[145] **DiVincenzo D. P. y Loss D.**, "Quantum Information Is Physical," *Superlattices and Microstructures*, vol. 23, 1998, p. 419.

[146] **Van Meter R.**, State of the Art in Quantum Computer, pp.1-14, Aug.13, 2011. http://aqua.sfc.wide.ad.jp/publications/van-meter-quantum-architecture-handout.pdf

[147] **Van Meter III R. D.**, *Architecture of a Quantum Multicomputer Optimized for Shor's Factoring Algorithm*. PhD thesis, Keio University, 2006. Available as arXiv:quant-ph/0607065.

[148] **Ladd T. D., Jelezko F., Laflamme R., Nakamura Y., Monroe C., y O'Brien J. L.**. Quantum computers. *Nature*, 464:45–53, March 2010.

[149] **Devitt S. J., Nemoto K., y Munro W. J.,** The idiots guide to quantum error correction. arXiv:0905.2794v2 [quant-ph], 2009.

[150] **Dür W. y Briegel H. J.,** Entanglement purification and quantum error correction. *Rep. Prog. Phys.*, 70:1381–1424, 2007.

[151] **Vandersypen L. M. K. y Chuang I.,** NMR techniques for quantum computation and control. *Rev. Modern Phys.*, 76:1037, 2004.

[152] **Bacon D. y van Dam W.,** Recent progress in quantum algorithms. *Communications of the ACM*, 53(2):84–93, February 2010.

[153] **Mosca M.**, Quantum algorithms. *Arxiv preprint arXiv:0808.0369*, 2008.

[154] **Brown K. L., Munro W. J., y Kendon V. M.**, Using quantum computers for quantum simulation. *Entropy*, 12(11):2268–2307, 2010.

Capítulo 4:

[155] **Gutiérrez E., Romero S., Trenas M. A. and Zapata E. L.**, Quantum computer simulation using the CUDA programming model, Computer Physics Communications 181 (2010) pp. 283–300.

[156] **Glendinning I. y Ömer B.**, Parallelization of the QC-lib quantum computer simulator library, in: Lecture Notes in Computer Science, vol. 3019, 2004, pp. 461–468.

[157] **Deutsch D. y Jozsa R.**, Rapid solution of problems by quantum computation, Proceedings of Royal Society of London Series A 439 (1992) 553–558.

[158] **Grover L.**, A fast quantum mechanical algorithm for database search, in: Annual ACM Symposium on the Theory of Computation, 1996, pp. 212–219.

[159] **Shor P.**, Algorithms for quantum computation: Discrete logarithm and factoring, in: Symposium on Foundations of Computer Science, 1994, pp. 124–134.

[160] **Ömer B.**, QCL: A programming language for quantum computers, at http://tph.tuwien.ac.at/~oemer/qcl.html.

[161] **Butscher B. y Weimer H.**, Libquantum library, at http://www.libquantum.de.

[162] NVIDIA CUDA: Programming guide, and SDK, at http://www.nvidia.com/cuda.

[163] Quantum computer simulators, at http://www.quantiki.org/wiki/index.php/ List_of_QC_simulators.

[164] SPEC 2006: CPU-intensive benchmark suite, at http://www.spec.org.

[165] **Aminian M., Saeedi M., Zamani M. y Sedighi M.**, FPGA-based circuit model emulation of quantum algorithms, in: IEEE Computer Society Annual Symposium on VLSI, 2008, pp. 399–404.

[166] **Fujishima M.**, FPGA-based high-speed emulator of quantum computing, in: IEEE Int'l Conference on Computer Design, 2004.

[167] **Khalid A., Zilic Z. y Radecka K.**, FPGA emulation of quantum circuits, in: IEEE Int'l Conference on Field-Programming Technology, 2003.

[168] **Udrescu L. P. M. y Vladutiu M.**, Using HDLs for describing quantum circuits: A framework for efficient quantum algorithm simulation, in: Computing Frontiers Conference, 2004.

[169] **Obenland K. y Despain A.**, A parallel quantum computer simulator, at http://arxiv.org/abs/quant-ph/9804039, 1998.

[170] **Niwa J., Matsumoto K. y Imai H.**, General-purpose parallel simulator for quantum computing, Phys. Rev. A 66 (6) (Dec. 2002) 062317–062328.

[171] **Arnold G., Lippert T., Pomplun N. y Richter M.**, Large scale simulation of ideal quantum computers on SMP-clusters, in: Parallel Computing: Current & Future Issues of High-End Computing, Proceedings of the International Conference ParCo 2005, in: John von Neumann Institute for Computing Series (NIC), vol. 33, Central Institute for Applied Mathematics, Jülich, Germany, ISBN 3-00-017352-8, 2005, pp. 447–454; DBLP, http://dblp.uni-trier.de.

[172] **Raedt K. D., Michielsen K., Raedt H. D., Trieu B. y Arnold M. G.**, Massively parallel quantum computer simulator, Computer Physics Communications 176 (2007) 121–136.

[173] **El Zein A., Mccreath E., Rendell A. y Smola A.**, Performance evaluation of the NVIDIA GeForce 8800 GTX GPU for machine learning, in: Lecture Notes in Computer Science, vol. 5101, 2008, pp. 466–475.

[174] **Ha P., Tsigas P. y Anshus O.**, The synchronization power of coalesced memory accesses, in: Lecture Notes in Computer Science, vol. 5218, 2008 pp. 320–334.

[175] **Barenco A., Bennett C., Cleve R., DiVicenzo D., Margolus N., Shor P., Sleator T., Smolin J. y Weinfurter H.**, Elementary gates for quantum computation, Phys. Rev. A 52 (5) (1995) 3457–3467.

[176] **Deutsch D.**, Quantum computational networks, Proceedings of Royal Society of London Series A 425 (1989) 73–90.

Capítulo 5:

Clásico

Algoritmos

[177] **Golub G. H. y van Loan C. F.**, *Matrix Computations*, (3rd Edition) Johns Hopkins University Press, Baltimore, 1996.

[178] **Hadley G.**, *Linear Algebra*. Mass.: Wesley, 1961.

[179] **Strang G.**, *Linear Algebra and Its Applications*. New York: Academic Press, 1980.

[180] **Leon S. J.**, "Linear Algebra with Applications" Maxwell MacMillan, New York, 1990.

[181] **Stewart G. W.**, *Introduction to Matrix Computations.* Orlando: Academic Press, 1973.

Aplicaciones

[182] **Lodhi H. y Guo Y.**, "Gram-Schmidt kernels applied to structure activity analysis for drug design", in Proc. of 2nd. Workshop on Datamining in Bioinformatics, Edmonton, Alberta, Canada, July 23rd. 2002, pp. 37-42.

[183] **Mozingo R. A. y Miller T. W.**, *Introduction to Adaptive Arrays*. New York: Wiley, 1980.

[184] **Compton R. T.**, *Adaptive Antennas: Concepts and Performance*. New Jersey: Prentice-Hall, 1988.

[185] **Makhoul J.**, "A class of all-zero lattice digital filters: properties and applications," IEEE Trans. Acoust. Speech and Signal Process., vol. ASSP-26, pp.304-314, 1978.

[186] **Ahmed N. y Youn D. H.**, "On a realization and related algorithm for adaptive prediction," IEEE Trans. Acoust. Speech and Signal Process., vol. ASSP-28, pp.493-4974, 1980.

[187] **Gallivan K. A. y Leiserson C. E.**, "High-performance architectures for adaptive filtering based on the Gram-Schmidt algorithm," SPIE, Real Time Signal Process. VII, vol.495, pp.30-36, 1984.

[188] **Al-Dhahir N. y Cioffi J. M.**, "Efficiently computed reduced-parameter input-aided MMSE equalizers for ML detection: A unified approach," IEEE Trans. on Info. Theory, vol. 42, pp. 903-915, May 1996.

[189] **Melsa P. J. W., Younce R. C., y Rhors C. E.**, "Impulse response shortening for discrete multitone transceivers," IEEE Trans. on Communications, vol. 44, pp. 1662-1672, Dec. 1996.

[190] **Al-Dhahir N. y Cioffi J. M.**, "Optimum finite-length equalization for multicarrier transceivers," IEEE Trans. on Communications, vol. 44, pp. 56-63, Jan. 1996.

[191] **Lu B., Clark L. D., Arslan G., y Evans B. L.**, "Divide-and-conquer and matrix pencil methods for discrete multitone equalization," IEEE Trans. on Signal Processing, in revision.

[192] **Arslan G., Evans B. L., y Kiaei S.**, "Equalization for Discrete Multitone Receivers To Maximize Bit Rate," IEEE Trans. on Signal Processing, vol. 49, no. 12, pp. 3123-3135, Dec. 2001.

[193] **Farhang-Boroujeny B. y Ding M.**, "Design Methods for Time Domain Equalizer in DMT Transceivers", IEEE Trans. on Communications, vol. 49, pp. 554-562, March 2001.

[194] **Cioffi J. M.**, A Multicarrier Primer. Amati Communication Corporation and Stanford University, T1E1.4/97-157, Nov. 1991.

[195] **Ding M., Redfern A. J., y Evans B. L.**, "A Dual-path TEQ Structure For DMT-ADSL Systems", Proc. IEEE Int. Conf. on Acoustics, Speech, and Signal Proc., May 13-17, 2002, Orlando, FL, accepted for publication.

[196] **Acker K. V., Leus G., Moonen M., van de Wiel O., y Pollet T.**, "Per tone equalization for DMT-based systems," IEEE Trans. on Communications, vol. 49, no. 1, pp. 109-119, Jan 2001.

[197] **Trautmann S., y Fliege N. J.**, "A new equalizer for multitone systems without guard time" IEEE Communications Letters, Vol.6, No. 1 pp. 34 -36, Jan 2002.

[198] **Tuma M., y Benzi M.**, "A robust preconditioning technique for large sparse least squares problems", in Proc. of XV. Householder Symposium, Peebles, Scotland, June 17-21, 2002. pp.8.

[199] **Mastriani M.**, "*Denoising and Compression in Wavelet Domain via Projection onto Approximation Coefficients*," International Journal of Signal Processing, Volume 5, Number 1, pp.20-30, 2008.

Redes Sistólicas

[200] **Haykin S.**, *Modern Filters*. New York: Macmillan, 1990.

[201] **Haykin S.**, *Adaptive Filter Theory*. New Jersey: Prentice-Hall, 1991.

Versión estable propia

[202] **Mastriani M., y Naiouf M.**, "*Enhanced Gram-Schmidt Process for Improving the Stability in Signal and Image Processing"*, International Journal of Mathematical Sciences, WASET, vol. 7, Nro. 4, 2013, pp.7-11.

[203] **Mastriani M.**, "*Fast Cosine Transform to increase speed-up and efficiency of Karhunen-Loève Transform for lossy compression of SAR imagery"*, WSEAS Transactions on Signal Processing, ID: 5714-160.

Booleano

[204] **Mastriani M.**, "*Systholic Boolean Orthonormalizer Network in Wavelet Domain for Microarray Denoising*," International Journal of Signal Processing, Volume 2, Number 4, pp.273-284, 2005.

[205] **Mastriani M.**, "*Systholic Boolean Orthonormalizer Network in Wavelet Domain for SAR Image Despeckling"*, WSEAS Transactions on Signal Processing, ID: 5714-125.

[206] **Mastriani M.**, "*New Tool for Boolean Signal Processing"*, WSEAS Transactions on Signal Processing, ID: 5714-126.

Cuántico

[207] **Cherkes I., Klaiman S., y Moiseyev N.**, Spanning the Hilbert Space With an Even Tempered Gaussian Basis Set, International Journal of Quantum Chemistry, Vol. 109, 2996–3002, 2009.

[208] **Plenio M.**, Quantum Mechanics, Imperial College, 2nd term 2002.
http://www.lsr.ph.ic.ac.uk/~plenio/lecture.pdf

[209] **Martikainen J.-P.**, Quantum information and computing, Lecture 2: Essential quantum mechanics Department of Physical Sciences, University of Helsinki.
http://www.helsinki.fi/˜jamartik

[210] **Riseborough P. S.**, Quantum Mechanics I, April 1, 2013.
https://math.temple.edu/~prisebor/qm1.pdf

[211] **Riseborough P. S.**, Advanced Quantum Mechanics, April 1, 2013.
https://math.temple.edu/~prisebor/Advanced.pdf

Capítulo 6:

Cuántico

[212] **M. Mastriani, M. Naiouf**, "Quantum Enhanced Correlation Matrix Memories via States Orthogonalisation", International Journal of Electronics Science and Engineering, WASET, vol. 7, Nro. 8, 2013, pp.1131-1136.

Concepto de Memoria

[213] **Haykin S.**, *Neural Networks: A Comprehensive Foundation,* Macmillan, New York, 2002.

[214] **Mastriani M.**, "Self-Restorable Stochastic Neuro-Estimation using Forward-Propagation Training Algorithm," *Proc. of INNS*, Portland, OR, **1**, 404-411, 1993a.

[215] **Mastriani M.**, "Self-Restorable Stochastic Neurocontrol using Back-Propagation and Forward-Propagation Training Algorithms," *Proc. of ICSPAT*, Santa Clara, CA, **2**, 1058-1071 (1993b).

[216] **Mastriani M.**, "Pattern Recognition Using a Faster New Algorithm for Training Feed-Forward Neural Networks," *Proc. of INNS*, San Diego, CA, **3**, 601-606 (1994a).

[217] **Mastriani M.**, "Predictor of Linear Output," *Proc. IEEE Int. Symposium on Industrial Electronics*, Santiago, Chile, 269-274, 1994b.

[218] **Mastriani M.**, "Predictor-Corrector Neural Network for doing Technical Analysis in the Capital Market," *Proc. AAAI International Symposium on Artificial Intelligence*, Monterrey, México, 49-58, 1994c.

[219] **Willshaw D. J. y von der Malsburg C.**, "How patterned neural connections can be set up by self-organization," *Proc. of the Royal Society of London*, **Series B 194**, 431-445, 1976.

[220] **Kohonen T.**, "Physiological interpretation of the self-organizing map algorithm." *Neural Networks* **6**, 895-905 (1993).

[221] **Palm G.**, Neural Assemblies: An Alternative Approach, New York: Springer-Verlag, 1982.

[222] **Willshaw D. J. *et al***, "Non-holographic associative memory," *Nature* (London) **222**, 960-962, 1969.

[223] **Baum E. B., Wilczek F., Moody J.**, "Internal Representation for Associative Memories", Biological Cybernetics, 1998, Vol. 59, pp. 217-228.

[224] **Hobson S., y Austin J.**, "Improved Storage Capacity in Correlation Matrix Memories Storing Fixed Weight Codes," in *19th International Conference on Artificial Neural Networks: Part I*, Limassol, 2009, pp. 728-736.

[225] **Kustrin D., y Austin J.**, "Connectionist Propositional Logic: A Simple Correlation Matrix Memory Based Reasoning System," in *Emergent neural computational architectures based on neuroscience: towards neuroscience-inspired computing*, Stefan Wermter, Jim Austin, and David Willshaw, Eds. New York, United States of America: Springer-Verlag, 2001, pp. 534-546.

[226] **Turner M., y Austin J.**, "Matching performance of binary correlation matrix memories," *Transactions of the Society for Computer Simulation International*, vol. 14, no. 4, pp. 1637-1648, Dec. 1997.

[227] **Kohonen, T.**, "Correlation Matrix Memories." *IEEE Trans. on Computers*, **C-21**, 353-359, 1972.

Boleano

[228] **Mastriani M.**, *"Enhanced Boolean Correlation Matrix Memory",* (RNL02), X RPIC Reunión de Trabajo en Procesamiento de la Información y Control, 8 al 10 de Octubre 2003, San Nicolás, Buenos Aires, Argentina, vol. 2, pp. 470-473.

[229] **Mastriani M.**, "*Enhanced Boolean Correlation Matrix Memory"*, WSEAS Transactions on Signal Processing, ID: 5714-127.

Binario

[230] **Turner M., y Austin J.**, Matching performance of binary correlation matrix memories. Neural Networks; 10:1637-1648, 1997.

[231] **Turner M., y Austin J.**, Matching performance of binary correlation matrix memories, Transactions of the Society for Computer Simulation International, vol. 14, no. 4, pp. 1637-1648, December 1997.

[232] **Zhou P., Austin J., y Kennedy J.**, A high performance k-NN classifier using a binary correlation matrix memory. Advances in Neural Information Processing Systems, Vol. 11, CA, MIT press, 1999.

[233] **Zhou P., Austin J., y Kennedy J.**, A Binary Correlation Matrix Memory k-NN Classifier with Hardware Implementation, British Machine Vision Conference.

[234] **Armstrong J. R.**, Boolean Weightless Neural Network Architectures, A thesis submitted in partial fulfilment for the requirements of the degree of DOCTOR OF PHILOSOPHY, Applied Digital Signal and Image Processing (ADSIP) Centre, School of Computing, Engineering and Physical Sciences, University of Central Lancashire in collaboration with BAE SYSTEMS, April 2011.

[235] **Hodge V. J., y Austin J.**, An evaluation of standard retrieval algorithms and a binary neural approach. Neural Networks, 14 (3). pp. 287-303., 2001.

Clásico

[236] **Baum E. B., Wilczek F., y Moody J.**, "Internal Representation for Associative Memories", Biological Cybernetics, Vol. 59, pp. 217-228, 1998.

[237] **Hobson S., y Austin J.**, Improved Storage Capacity in Correlation Matrix Memories Storing Fixed Weight Codes, Proceedings of the 19th International Conference on Artificial Neural Networks, Lecture Notes in Computer Science 5768, pp. 728-736, Springer-Verlag, 2009.

[238] **Aykin E., y O'Keefe S.**, A Fuzzy Classifier Based on Correlation Matrix Memories, Computer Science Department, The University of York Proceedings of the 10th WSEAS International Conference on FUZZY SYSTEMS, p. 63-68, 2009.

[239] **Meyer R.**, A Fuzzy Binary Correlation Matrix Memory Classification System, Master Thesis, University of York, September 13, 2011.

[240] **Austin J., Hobson S., Burles N., y O'Keefe S.**, A rule chaining architecture using a correlation matrix memory. Int Conf on Artif Neural Networks, pp 49-56. doi: 10.1007/978-3-642-33269-2 7, 2012.

[241] **Austin J.**, Correlation Matrix Memories for Knowledge Manipulation, International Conference on Neural Networks, Fuzzy Logic and Soft Computing, Iizuka, Japan. August 17, l994.

[242] **Austin J., y Lees K.**, A search engine based on neural correlation matrix memories, Neurocomputing, vol. 35, no. 1, pp. 55–72, November 2000.

[243] **Shanker K. P. S., Turner A., Sherly E., y Austin J.**, Sequential data mining using correlation matrix memory. Int. Conf. on Networks and Information Technology, pp. 470-472. doi: 10.1109/ICNIT.2010.5508469, 2010.

[244] **Hobson S.**, Correlation Matrix Memories: Improving Performance for Capacity and Generalisation. Ph.D. Thesis, University of York, Computer Science, September 2011.

[245] **Hobson S., y Austin J.**, Improved storage capacity in correlation matrix memories storing fixed weight codes. Int. Conf. on Artificial Neural Networks, pp. 728-736. doi: 10.1007/978-3-642-04274-4 75, 2009.

[246] **Walker R.**, A MATLAB Tool Kit For CMM, 3rd Year Project, Department of Computer Science, University of York, Supervisor: Dr Simon O'Keefe 11th March 2008.

[247] **Kohonen T.**, Correlation Matrix Memories, IEEE Transactions on Computers, C-21, pp. 353-359, 1972.

[248] **Wilde I. F.**, Neural Networks, Mathematics Department King's College London, London, WC2R 2LS, UK.

Capítulo 7:

Simulación

[249] **Burles N.**, Quantum Parallel computation with neural networks, Department of Computer Science, University of York, Submitted in part fulfilment for the degree of MEng, 10 May 2010.

Capítulo 8:

Quantum Neural Networks

[250] **Garman J. A.**, A Heuristic Review of Quantum Neural Networks, Department of Physics, Imperial College London, Submitted in partial fulfilment of the requirements for the degree of Master of Science, 6 October 2011.

Sinopsis Curricular

Nombre y apellido: **Mario Mastriani**
Nacionalidad: Argentina y Europea
Celular: +54-9-11-6314-3000
E-mail: mmastriani@cema.edu.ar
Web page: http://www.ucema.edu.ar/institucional/personal_profesores_new.php?id=12742
http://untref.edu.ar/academico/investigacion-y-desarrollo/laboratorios-y-centros-de-investigacion/?scroll=188
http://www.neotvlab.net/lis/

Sinopsis Curricular:

• Ingeniero Electrónico, UBA, Septiembre 1989
• Doctor en Ingeniería, UBA, Mayo 2006
• Doctor en Ciencias de la Computación, UBA, Marzo 2009
• Doctor en Ciencia y Tecnología, UNGS, Octubre 2011
• Doctor en Ciencias Informáticas, UNLP, Septiembre 2014
• Posición Posdoctoral, UNTreF, Agosto 2014

• Referee de 15 journals internacionales de la IEEE, Springer-Verlag, Taylor & Francis, IET, SPIE, OSA, Elsevier, etc.

• Ex coordinador de la Carrera de Ingeniería en Computación de la Univ. Nac. de Tres de Febrero (UNTreF)

• Docente-Investigador: Categoría 1

• Profesor Titular de Procesamiento de Imágenes de la Carrera de Ingeniería en Computación de la UNTreF

• Director del Laboratorio de Imágenes y Señales (LIS) del Sistema UVT-CPA-NeoTVLabs de la UNTreF

• Ex profesor del curso de Doctorado de la Facultad de Ingeniería de la UBA:
Tópicos avanzados en procesamiento de señales e imágenes

• Ex profesor de la Maestría en Optoelectrónica de la Facultad de Ingeniería de la UBA:
1) Diagnóstico y tratamiento médico.
2) Aplicaciones Médicas.

• Ex director del Laboratorio de Investigación y Desarrollo en Nuevas Tecnologías (LIDeNTec) de ANSES

• Coordinador del área de Innovación Técnica de la Dirección General de Informática e Innovación Tecnológica (GIIT) de ANSES

• Ex evaluador FONCyT en Tecnología Informática, Comunicaciones y Electrónica de la Agencia

• Ex evaluador Consejo de Investigación Científica (CIC) de la Provincia de Bs. As. en Tecnología Informática, de la Comunicaciones y Electrónica

• Ex evaluador CONEAU para las carreras de Ingeniería Informática

• Ex responsable del grupo de focalización Synthetic Aperture Radar (SAR) de CONAE

• Ex contratista de INVAP S.E.

• Ex contratista de la Satellogic S.A.

• Ex representante por UNTreF ante el Programa de Desarrollo de Software para la TV Digital Interactiva del Ministerio de Planificación Federal, Inversión Pública y Servicios

• 49 publicaciones científicas internacionales con proceso de referato severo en journals

• 51 conferencias científicas en el país y el exterior

• Libros:
- "Técnicas de Mejoramiento de Imágenes de Radar de Apertura Sintética: Supresión del Speckle" Editorial Académica Española, 2011, Madrid, ISBN 978-3-8454-8262-0.
- "Compresión de Imágenes con Pérdidas: Rápida y Eficiente" Editorial Académica Española, 2011, Madrid, ISBN 978-3-8454-9062-5.
- "Supercompresión de Imágenes y Video con Aplicaciones" Editorial Académica Española, 2011, Madrid, ISBN 978-3-8465-7249-8.

• 2 capítulos en libros