




Emuladores de circuitos cuánticos

Guillermo Díaz Camacho
Fundación CESGA

El presente de la computación cuántica (I)

- Era NISQ (Noisy Intermediate-Scale Quantum)
- Es difícil fabricar chips con **muchos qubits**  • **Pocos qubits.**

El presente de la computación cuántica (I)

- Era NISQ (Noisy Intermediate-Scale Quantum)

- Es difícil fabricar chips con **muchos qubits**



- Pocos qubits.

- Es difícil fabricar **qubits con fidelidad**



- Poca profundidad.

El presente de la computación cuántica (I)

- Era NISQ (Noisy Intermediate-Scale Quantum)

- Es difícil fabricar chips con **muchos qubits**



- Pocos qubits.

- Es difícil fabricar **qubits con fidelidad**



- Poca profundidad.

- Es difícil realizar **puertas poco ruidosas**



- Pocas puertas.

El presente de la computación cuántica (I)

- Era NISQ (Noisy Intermediate-Scale Quantum)

- Es difícil fabricar chips con **muchos qubits**



- Pocos qubits.

- Es difícil fabricar **qubits con fidelidad**



- Poca profundidad.

- Es difícil realizar **puertas poco ruidosas**



- Pocas puertas.

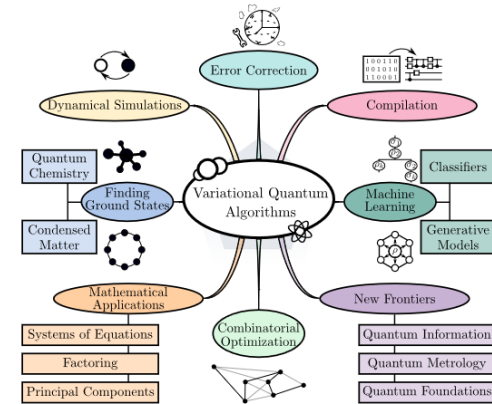
- Los qubits conectados tienen **crosstalk**



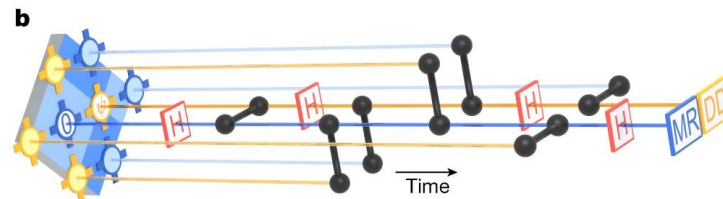
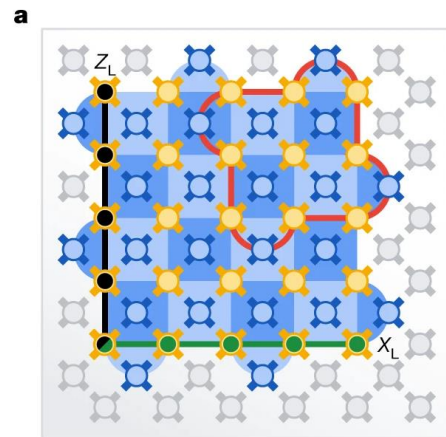
- Poco entrelazamiento.

El presente de la computación cuántica (II)

- Era NISQ (Noisy Intermediate-Scale Quantum)
- **Solución temporal: Algoritmos NISQ-friendly**
 - Ej: Algoritmos variacionales.
 - Se pueden ejecutar en ordenadores actuales: ¡investigación!



Cerezo et. al., *Nat. Rev. Phys.* (2021)



• Solución futura: Corrección/mitigación de errores

- Necesita:
 - Muchos qubits.
 - Muchas puertas extras.
 - Muy buena fidelidad.

Google Quantum AI, *Nature* (2023)

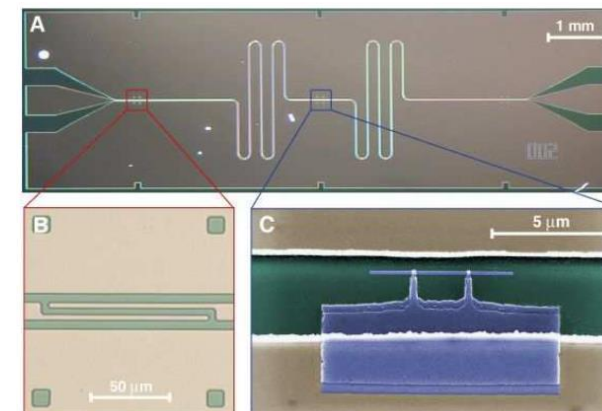
El presente de la computación cuántica (III)

- Pero mientras esperamos...
- **Simulación cuántica y emulación clásica de circuitos**

¿Emuladores o simuladores?

- **Simuladores cuánticos:**

- **Sistemas cuánticos** físicos que simulan otros sistemas
- **Hardware** construido con un propósito específico

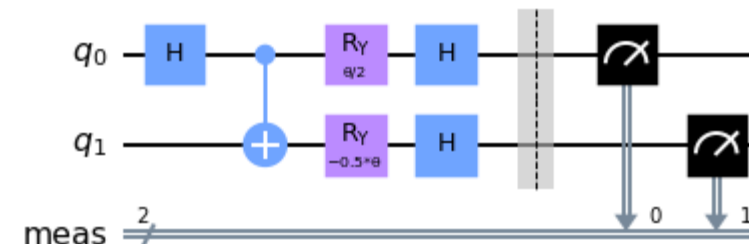


A. Wallraff et al., *Nature* (2004)

- **Cuidado:** No hay consenso claro!
Así es como lo llamamos nosotros!

- **Emuladores clásicos de circuitos cuánticos:**

- **Ordenadores clásicos** simulando circuitos cuánticos
- **Software** clásico que reproduce mecánica cuántica



¿Para qué **SÍ** usar emuladores?

- No tengo acceso a un computador cuántico
 - ¡Puedo usar mi ordenador!
- Quiero estudiar circuitos en condiciones ideales
 - **¡Investigación!**
 - Debugging
 - Estudiar performance, escalado, etc ...
- Quiero evaluar el efecto del ruido
 - ¡Corrección/mitigación de errores!

¿Para qué **NO** usar emuladores?

- ¿Resolver problemas grandes?
 - Requiere muchos recursos! (**memoria, tiempo y energía**).
- ¿Buscar ventaja cuántica?
 - Si un algoritmo se puede simular clásicamente, **por definición no hay ventaja cuántica.**
 - Hay algoritmos clásicos muy eficientes!
- ¿Algoritmos de inspiración cuántica?
 - **¡Siguen siendo clásicos!**

Tipos de emuladores

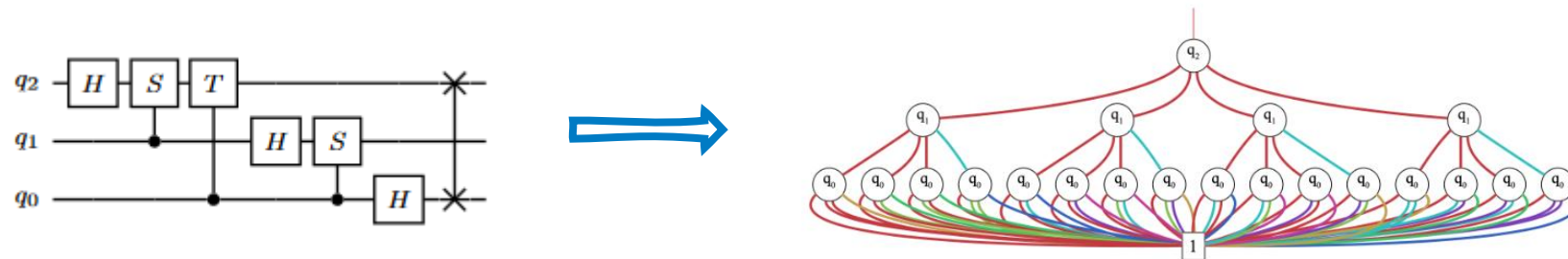
- Emulador de statevector (tipo **Schrödinger**):

- Guardan el vector de estado entero: **coste exponencial de memoria**
- Representación completa

$$|0000\rangle \longrightarrow [1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]^T \xrightarrow{F_N = \frac{1}{\sqrt{N}}} \begin{bmatrix} 1 & 1 & 1 & 1 & \dots & 1 \\ 1 & \omega_n & \omega_n^2 & \omega_n^3 & \dots & \omega_n^{N-1} \\ 1 & \omega_n^2 & \omega_n^4 & \omega_n^6 & \dots & \omega_n^{2(N-1)} \\ 1 & \omega_n^3 & \omega_n^6 & \omega_n^9 & \dots & \omega_n^{3(N-1)} \\ \vdots & \vdots & \vdots & \vdots & \dots & \vdots \\ 1 & \omega_n^{N-1} & \omega_n^{2(N-1)} & \omega_n^{3(N-1)} & \dots & \omega_n^{(N-1)(N-1)} \end{bmatrix}$$

- Emulador de árbol de decisiones (tipo **Feynman** o híbridos **Schrodinger-Feynman**):

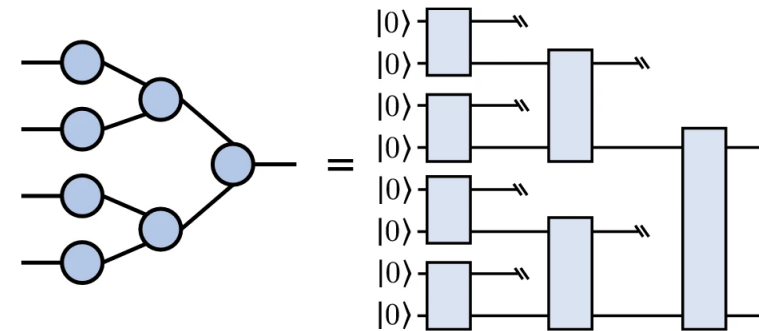
- Separan el estado en trozos: **menos costoso en memoria**
- Árbol de decisiones: **número exponencial de pasos**



- Wille, Burgholzer, Artner, *IEEE* (2021)

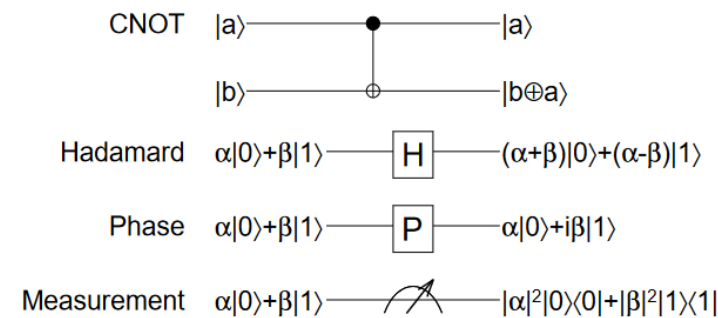
Otros emuladores

- Emuladores Tensor Networks (Ej: MPS)
 - Muy rápidos y necesitan menos memoria.
 - Requieren poco entrelazamiento en el sistema (Area Law)



- Guala et. al., *Sci. Rep.* (2023)

- Emuladores Stabilizer/Clifford
 - Polinomial en tiempo
 - Sólo circuitos con puertas Clifford



- Aaronson, Gottesman, *Phys Rev. A* (2004)

Limitaciones (statevector)

- ¡Memoria!

- Crece **exponencialmente** con N qubits
- Vector de estado: $2^{(N+4)}$ bytes $|\psi\rangle$
- Matriz densidad: $4^{(N+4)}$ bytes $\rho = |\psi\rangle\langle\psi|$
 - Necesario para emular ruido

- ¡Tiempo!

- Matrices también exponenciales.
- Paralelización en HPC (GPUs)

- FT3:

- 256 nodos ILK: 256 Gb
- 64 nodos A100: 256 Gb
- 1 nodo Optane: 8 Tb

$$|\psi\rangle$$

$$\rho = |\psi\rangle\langle\psi|$$

Qubits	Vector
10	16 Kb
20	16 Mb
30	16 Gb
33	137 Gb
38	4.4 Tb
50	18 Pb

Qubits	Matriz
10	268 Mb
14	68 Gb
15	275 Gb
17	4.4 Tb
20	281 Tb

¿Qué emuladores tenemos en CESGA?

- Instalados en Finisterrae III:

- Qiskit (Aer)
- Tket/PyTket
- Yao



- Como entorno:

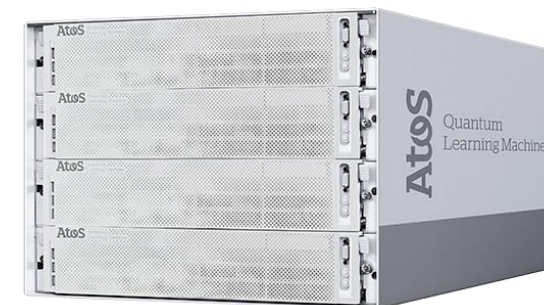
- Cirq
- PennyLane
- Qibo



PENNYLANE

- Hardware específico!

- Quantum Learning Machine (QLM, Atos)



- FX700 (Qulacs, Fujitsu)



Gracias por
vuestra atención

gdiaz@cesga.es



Software cuántico...

¿Qué es qué?

- **QASM: Quantum Assembly Language**
 - Convierte instrucciones de alto nivel a puertas en ordenadores cuánticos
 - Agnóstico! Todo tipo de QPUs, y también emuladores
- **Software Development Kits (SDKs):**
 - Full-Stack: Escribir, compilar y ejecutar
 - Primitivas: Trabajar a alto nivel
- **Algoritmos de inspiración cuántica**
 - Algoritmos clásicos, **no hay nada cuántico**
 - Ej: Tensor Networks, QPSO...

- **Librerías de Python, Open Source**
(~~pensadas para sus sistemas~~)

- Qiskit (IBM)
- Tket/PyTket (Quantinuum)
- Cirq (Google)
- Braket (AWS)
- Ocean (D-Wave)
- QLM (Atos)

- **Otros SDK**

- Pennylane (Xanadu)
- Qibo (CERN)
- ProjectQ (ETH Zurich)
- Q# (Microsoft, C#)
- Yao (Julia)
- Qulacs (Python/C++)



PENNYLANE

