Frameworks actuales para el desarrollo de la Computación Cuántica

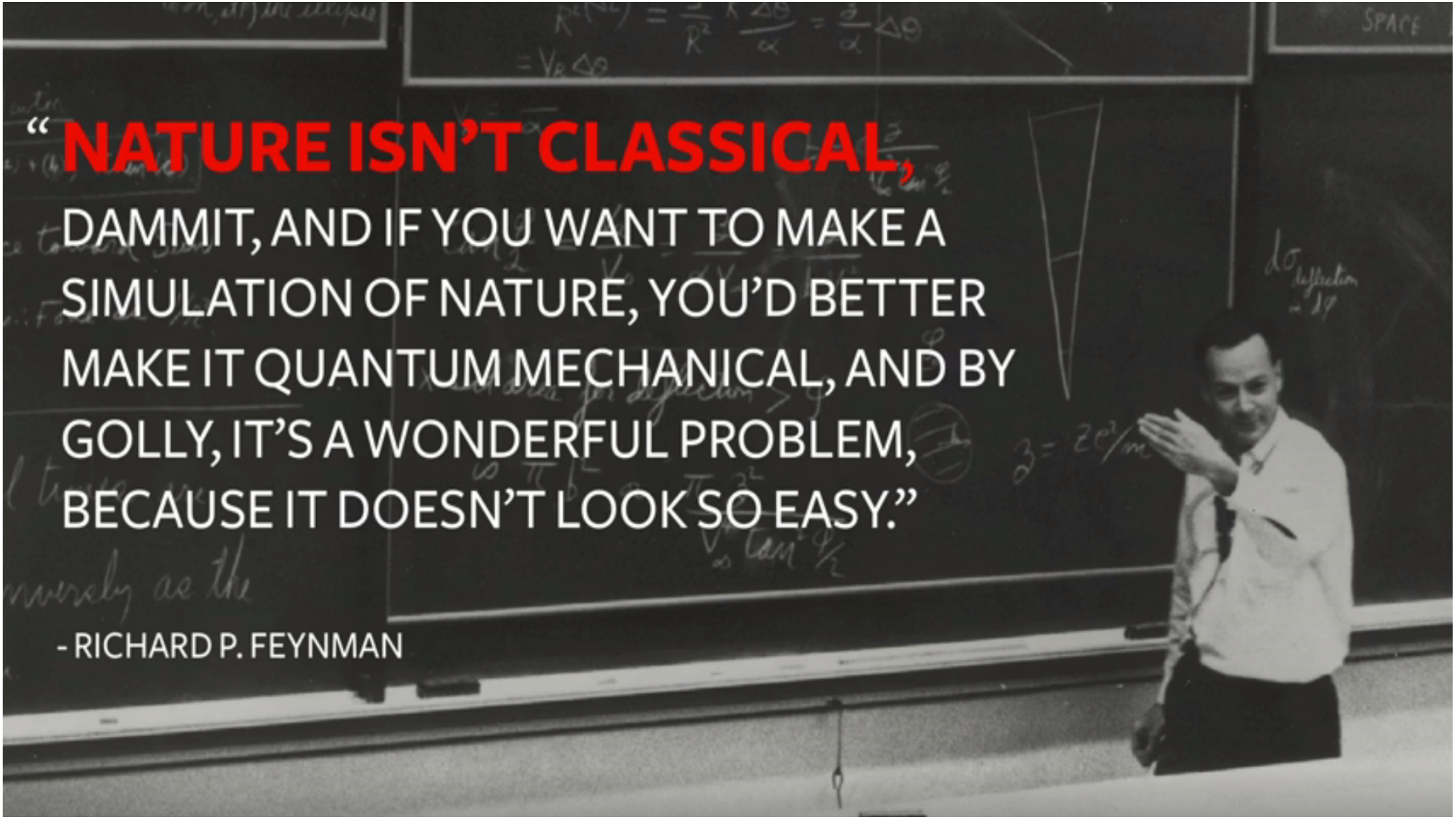
Jornadas Sarteco Cáceres – 18 de Septiembre 2019

Francisco J. Gálvez Ramírez
Arquitecto de Sistemas
fjgramirez@abdprof.com

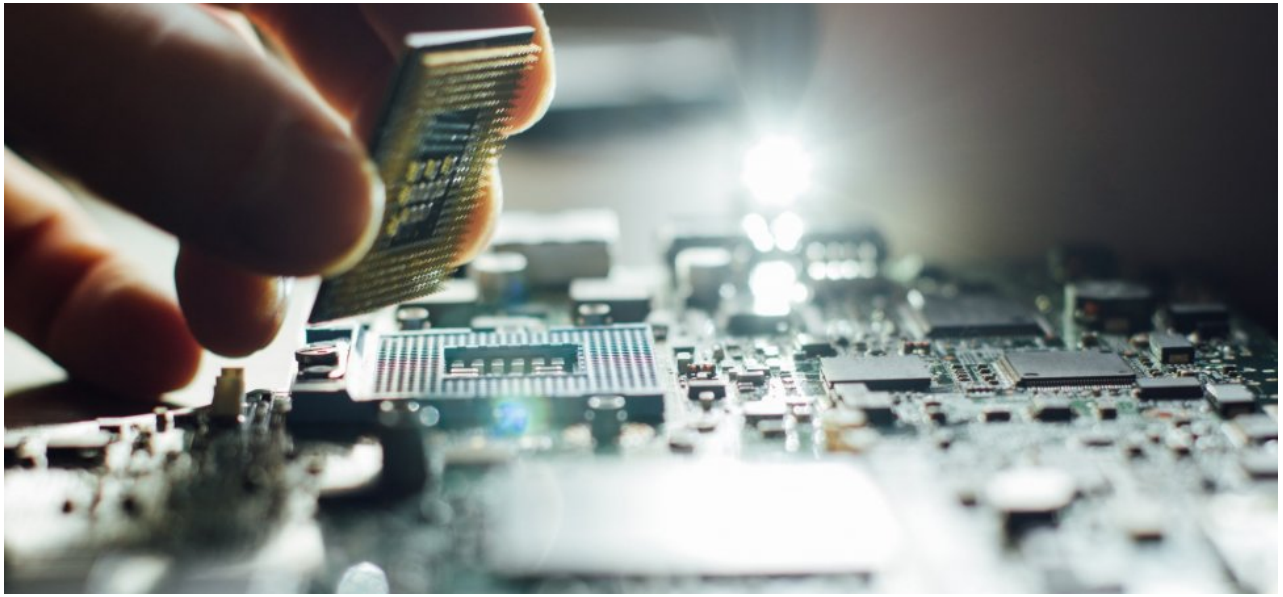
Introducción

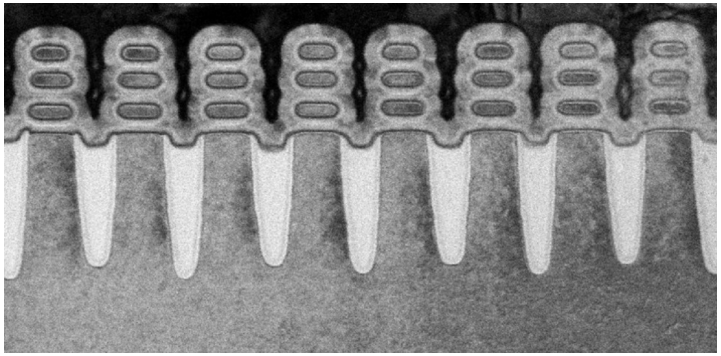
Agenda

1. Introducción
2. El Computador Cuántico
3. Infraestructuras Cuánticas
4. Computadores Cuánticos Actuales
5. Los Modelos de Programación
6. Frameworks de desarrollo Actuales


"NATURE ISN'T CLASSICAL,
DAMMIT, AND IF YOU WANT TO MAKE A SIMULATION OF NATURE, YOU'D BETTER MAKE IT QUANTUM MECHANICAL, AND BY GOLLY, IT'S A WONDERFUL PROBLEM, BECAUSE IT DOESN'T LOOK SO EASY."
- RICHARD P. FEYNMAN

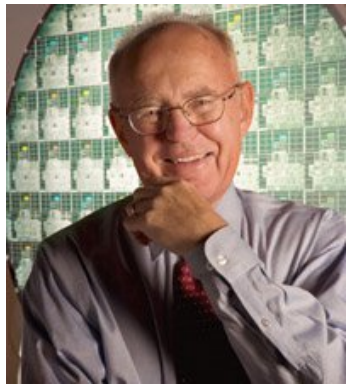
LEY DE MOORE





IBM 5nm chip

¿Cuanto tiempo nos queda para llegar al comportamiento cuántico de la naturaleza?



LA MÁQUINA DE TURING CUÁNTICA

David Deutsch, en la Universidad de Oxford describe el primer Computador Cuántico Universal

https://people.eecs.berkeley.edu/~christos/classics/Deutsch_quantum_theory.pdf

Quantum theory, the Church-Turing principle and the universal quantum computer

DAVID DEUTSCH*

Appeared in *Proceedings of the Royal Society of London A* **400**, pp. 97-117 (1985)†

(*Communicated by R. Penrose, F.R.S. — Received 13 July 1984*)

Abstract

It is argued that underlying the Church-Turing hypothesis there is an implicit physical assertion. Here, this assertion is presented explicitly as a physical principle: 'every finitely realizable physical system can be perfectly simulated by a universal model computing machine operating by finite means'. Classical physics and the universal Turing machine, because the former is continuous and the latter discrete, do not obey the principle, at least in the strong form above. A class of model computing machines that is the quantum generalization of the class of Turing machines is described, and it is shown that quantum theory and the 'universal

PRIMEROS ALGORITMOS CUÁNTICOS

- **Algorimo de Deusch –** Determina si una función es constante o balanceada

- **Algoritmo de Shor –** Factorización de grandes números

- **Algoritmo de Grover –** Búsquedas en espacios no estructurados

EL ALGORITMO DE DEUSTCH

f_1:

$0 \rightarrow 0$

$1 \rightarrow 0$

f_2:

$0 \rightarrow 1$

$1 \rightarrow 1$

f_3:

$0 \rightarrow 0$

$1 \rightarrow 1$

f_4:

$0 \rightarrow 1$

$1 \rightarrow 0$

Algoritmo de Deustch-Josza → Extensión del algoritmo de Deustch a registros con n valores

EL ALGORITMO DE SHOR

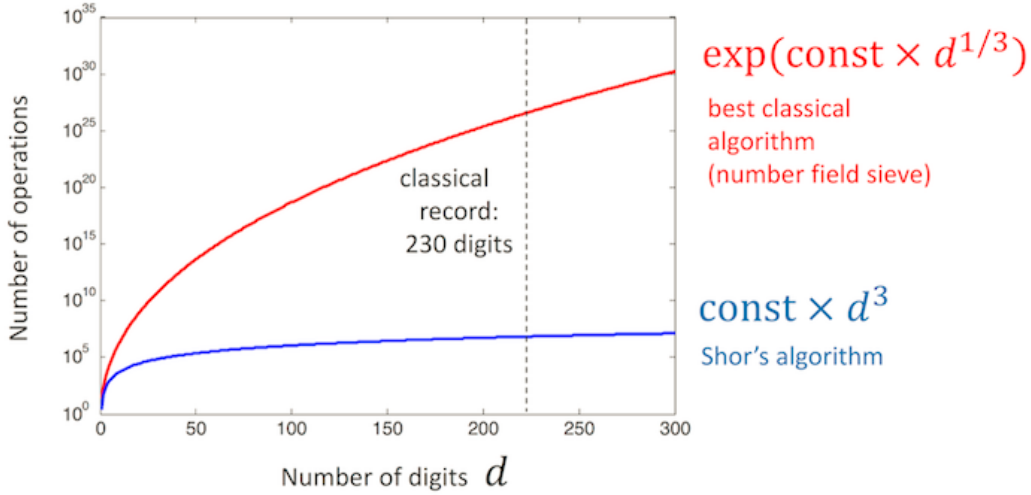
Numero de pasos que un ordenador clásico necesita dar para encontrar los factores primos de un número N de x dígitos

→ **Crece exponencialmente con x**

937 x 947 = N (fácil)

887339 = p x q (no tan fácil)

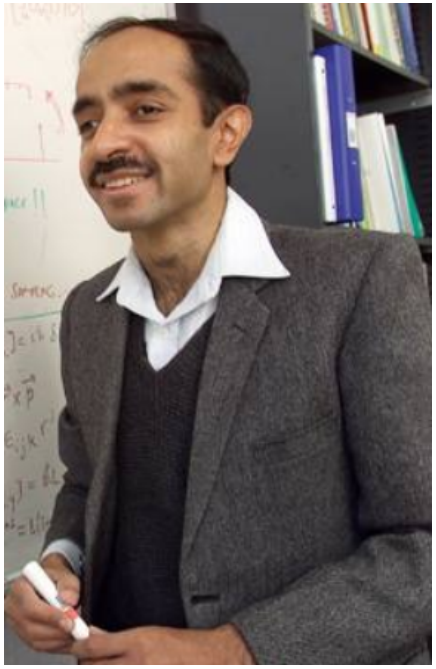
La robustez de la factorización es la base de algoritmo RSA



In 2001, IBM y la Universidad de Standford, ejecutaron por primera vez el algoritmo de Shor en un computador cuántico de **7 qubits desarrollado en Los Álamos.**

https://www-03.ibm.com/press/us/en/pressrelease/965.wss

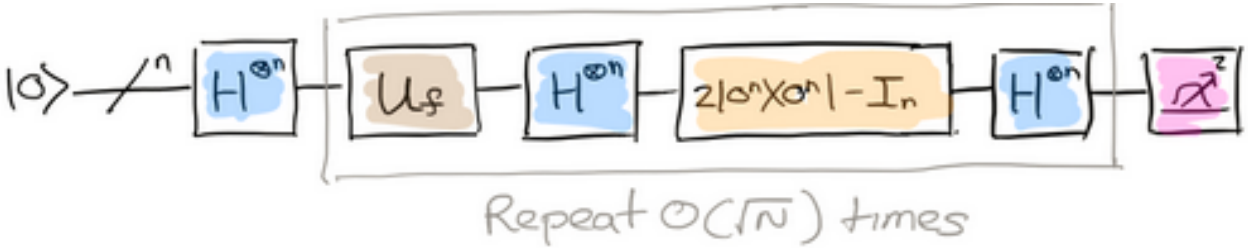
EL ALGORITMO DE GROVER



- ¿Cuantos intentos son necesarios para encontrar un elemento dentro de un conjunto de N elementos desordenados?

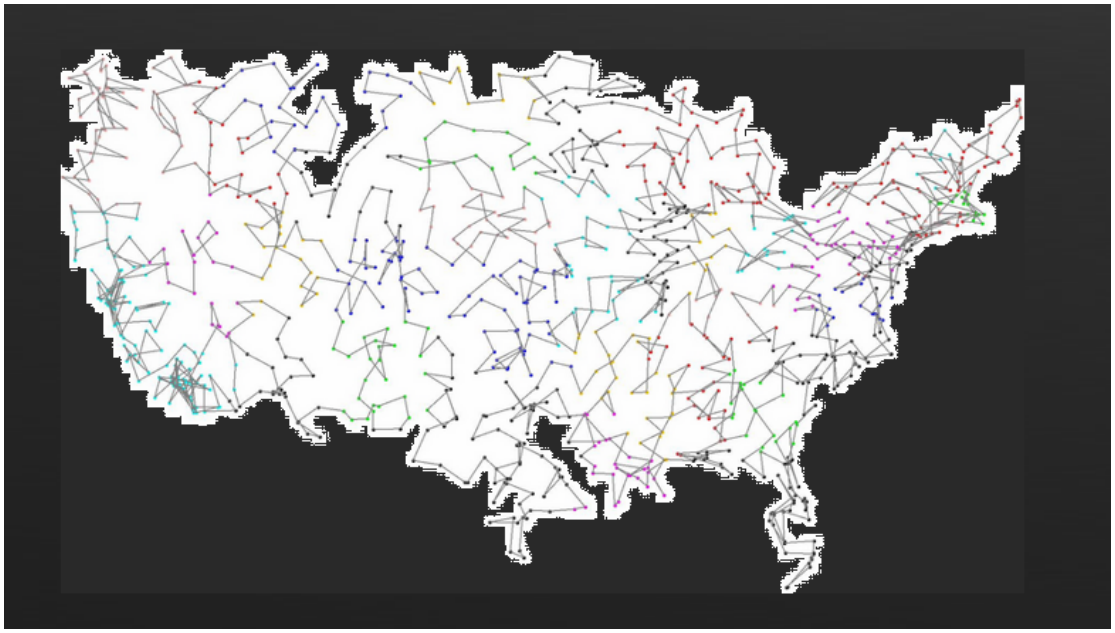
  → Son necesarios una media de N/2 intentos

- Un computador cuántico ejecutando el algoritmo de Grover lo haría en $\sqrt{N}$ intentos

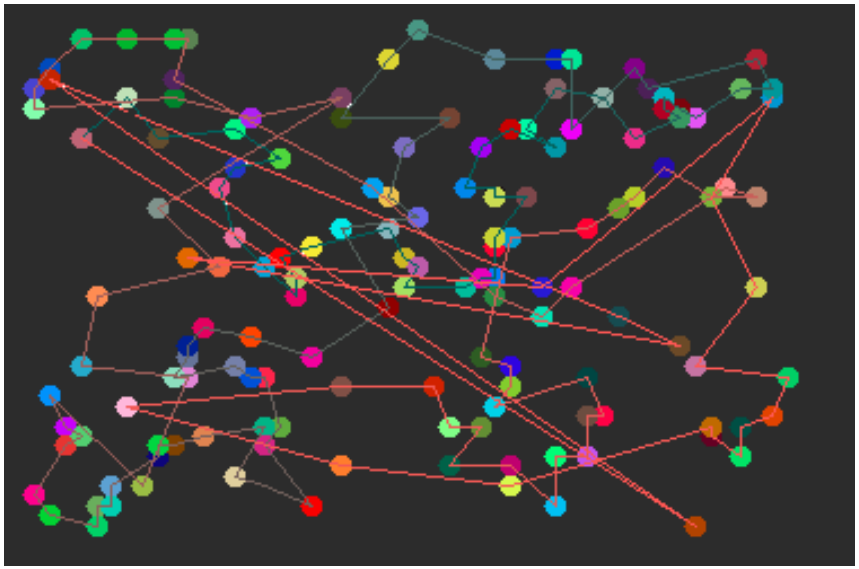


http://www.dma.eui.upm.es/MatDis/Seminario4/AlgoritmoGrover.pdf

Problema del viajante



Problemas de optimización.

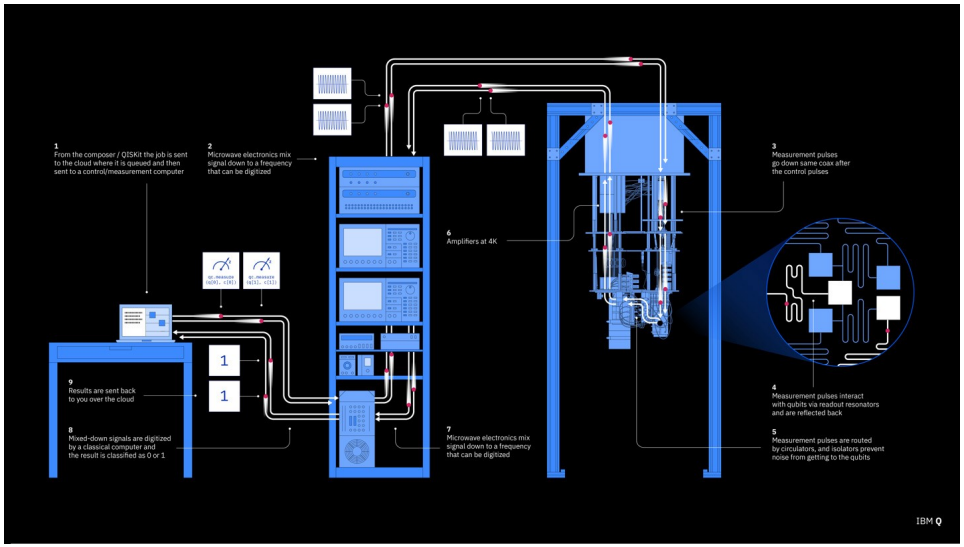


El Computador Cuántico

QUE ES UN COMPUTADOR CUÁNTICO

Características de un Computador Cuántico

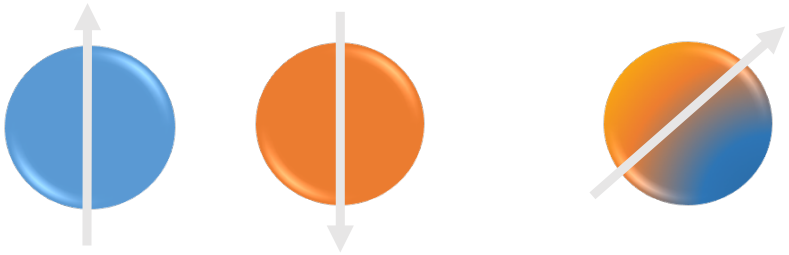
1. Se puede inicializar
2. Utiliza Bits Cuánticos (Qubits)
3. Trabaja con Paralelismo Cuántico
4. Mantiene la coherencia
5. Hace uso del Entrelazamiento



¿Que es un Qubit?

- El Qubit es el concepto cuantico de bit de información.

$|0>$ $|1>$ $\alpha|0> + \beta|1>$

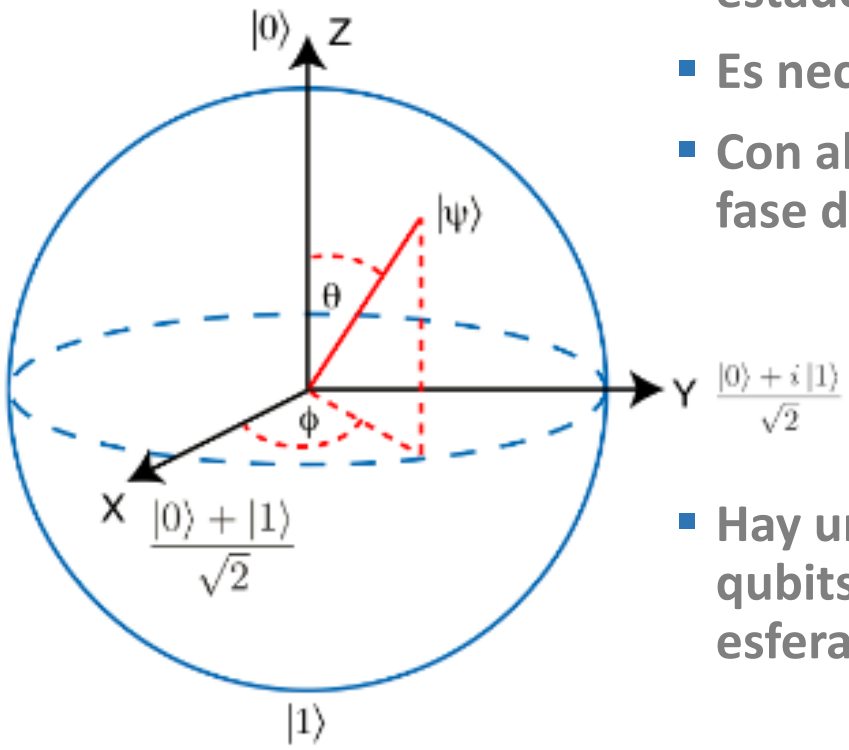


- No es un elemento o dispositvo.
- Concepto lógico que puede implementarse sobre un amplio rango de sistemas con comportamiento cuántico.
- Un Qubit puede representar dos estados: 0 y 1 (estados base)

Además un Qubit es capaz de trabajar con todas las posibles combinaciones que pueden tener lugar entre los estados base 0 and 1

$$\psi = \alpha|0> + \beta|1>$$

Representación de un Qubit en la esfera de Bloch



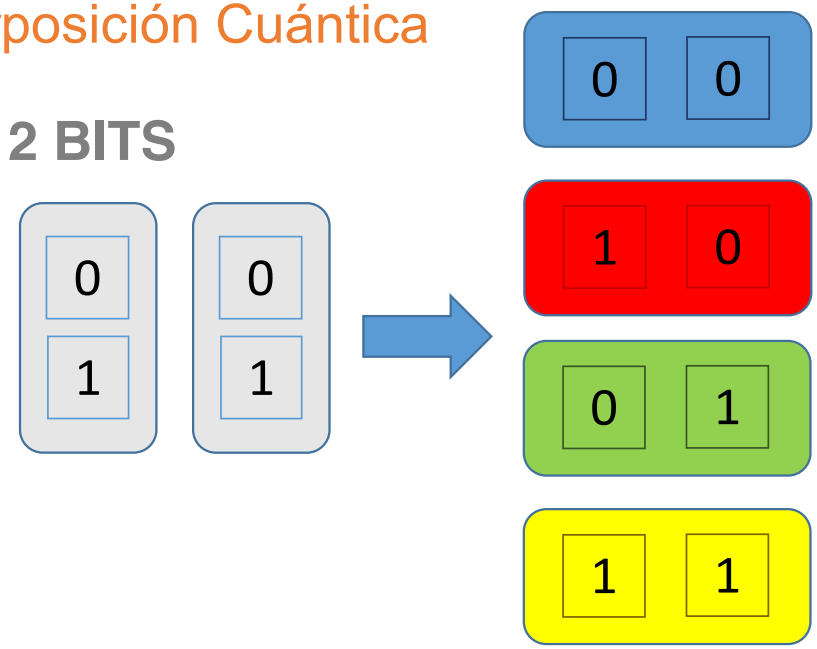
- La base estándar no es suficiente para describir un estado cuántico
- Es necesario especifica también una fase
- Con al esfera de Bloch se muestras las amplitudes y la fase de un estado.

$$|\psi\rangle = \cos\left(\frac{\theta}{2}\right)|0\rangle + e^{i\phi} sen\left(\frac{\theta}{2}\right)|1\rangle$$

- Hay una correspondencia de 1 → entre los estados de qubits puros $\mathbf{C}^2$ y los puntos R^3 sobre la superficie de una esfera de radio uno

Superposición Cuántica

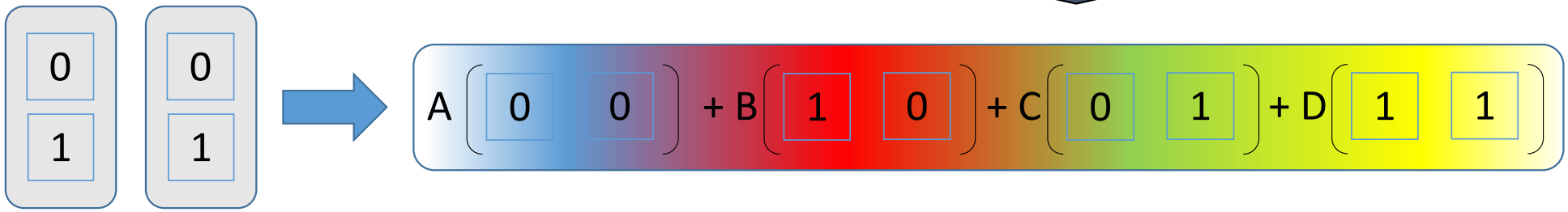
- **2 BITS**



4 Estados Independientes. El sistema puede estar en uno de estos cuatro estados.

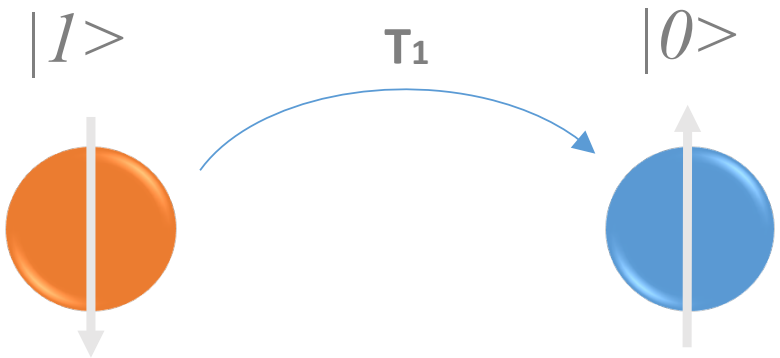
Un sistema que es combinación de cuatros posibles estados simultaneamente

- **2 QUBITS**



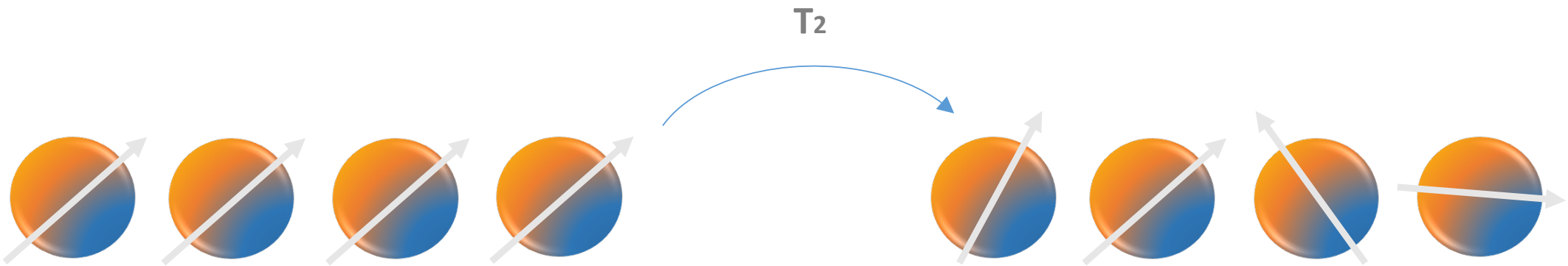
TIEMPOS DE COHERENCIA

T1 T2 Tiempos en los que un qubit o sistema de qubit se mantiene

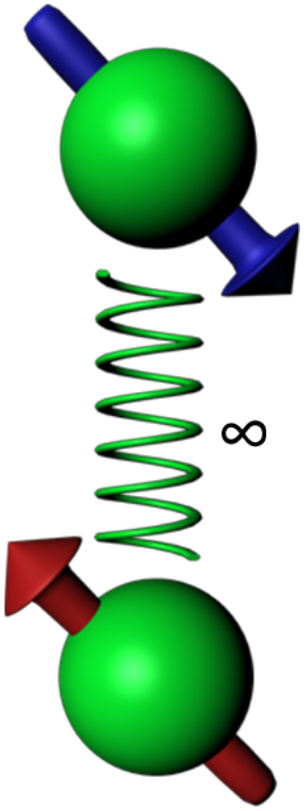


T1: es el tiempo en el que un qubit llevado estado excitado vuelve a su estado fundamental.

T2: Tiempo en el que un sistema de varios qubits pierde su coherencia.



Estados de Bell



- Un estado entrelazado no puede describirse en función de los estados de los qubits de lo componen, es decir:

  $|v_1\, v_2> \neq |v_1> \times |v_2>$

- Los Estados de Bell en el espacio de Hilbert vienen dados por:

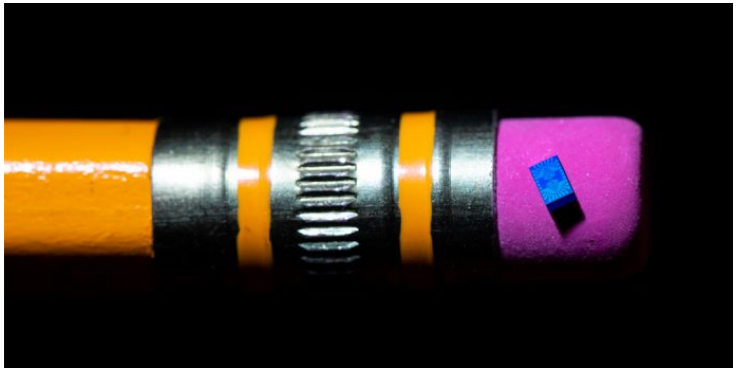
$$\left|\Phi^{+}\right\rangle = \frac{1}{\sqrt{2}}\left|00\right\rangle + \left|11\right\rangle \qquad \left|\Phi^{-}\right\rangle = \frac{1}{\sqrt{2}}\left|00\right\rangle - \left|11\right\rangle$$

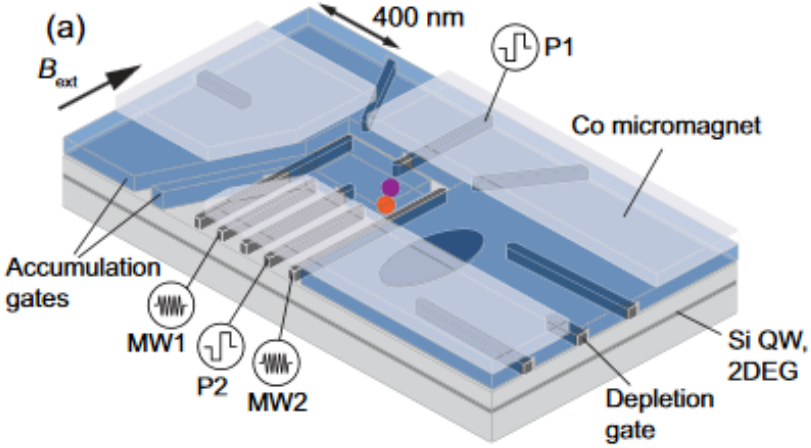
$$\left|\Psi^{+}\right\rangle = \frac{1}{\sqrt{2}}\left|01\right\rangle + \left|10\right\rangle \qquad \left|\Psi^{-}\right\rangle = \frac{1}{\sqrt{2}}\left|01\right\rangle - \left|10\right\rangle$$

Infraestructuras Cuánticas

Qubits de Spin

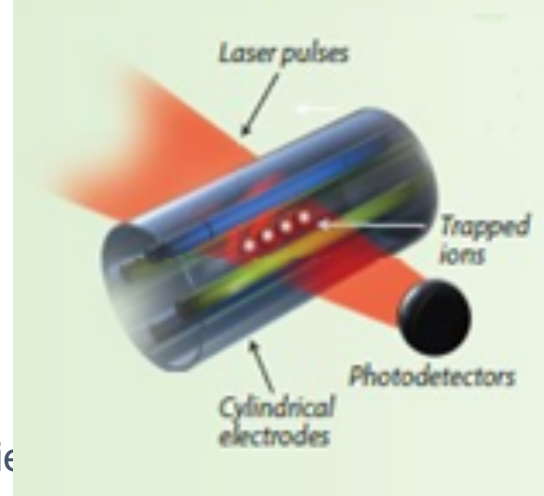
- Espines de e- o de núcleos en un substrato sólido.
- Estado de superposición varios segundos
- Compatibles con tecnologías de fabricación de chips actuales.
- El ruido proveniente del sustrato sólido puede dificultar su escalado.
- Pueden funcionar a temperaturas de 1K
- Muy pequeños

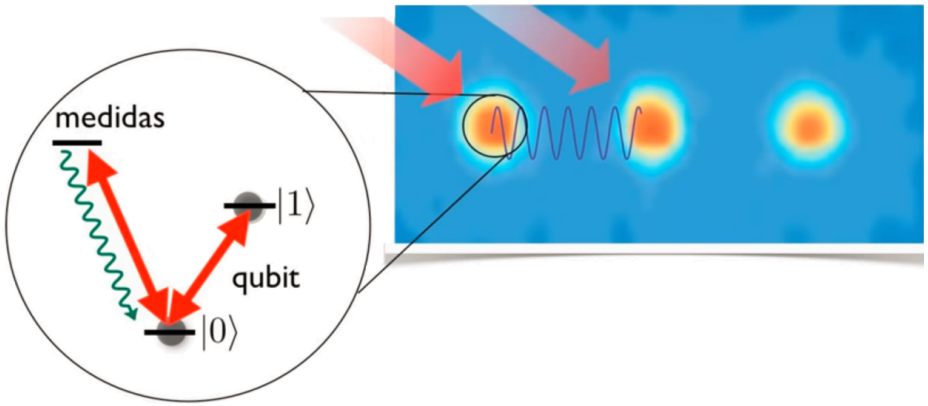


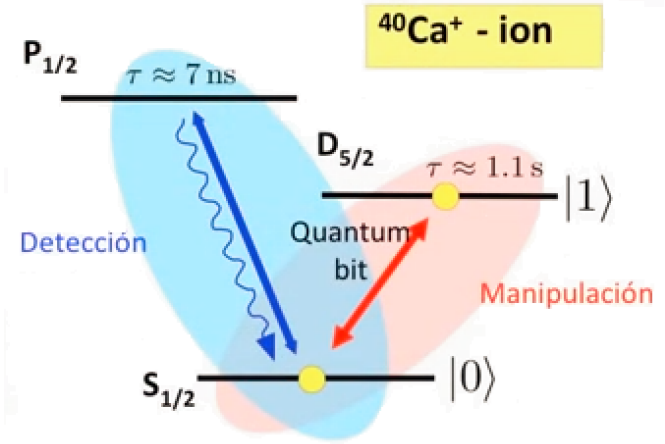


Trampas de Iones

- Iones atrapados en Campos Eléctricos
- Tienes pocos defectos
- Puertas lógicas con una tasa de error muy baja
- Requieren nuevas técnicas de fabricación
- Pulsos laser cronometrados para crear puertas lógicas y crear entrelazamie
- Codificar un qubit lógico con varios qubits físicos.

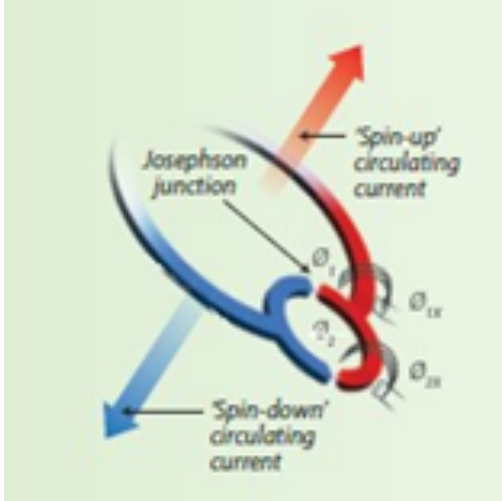


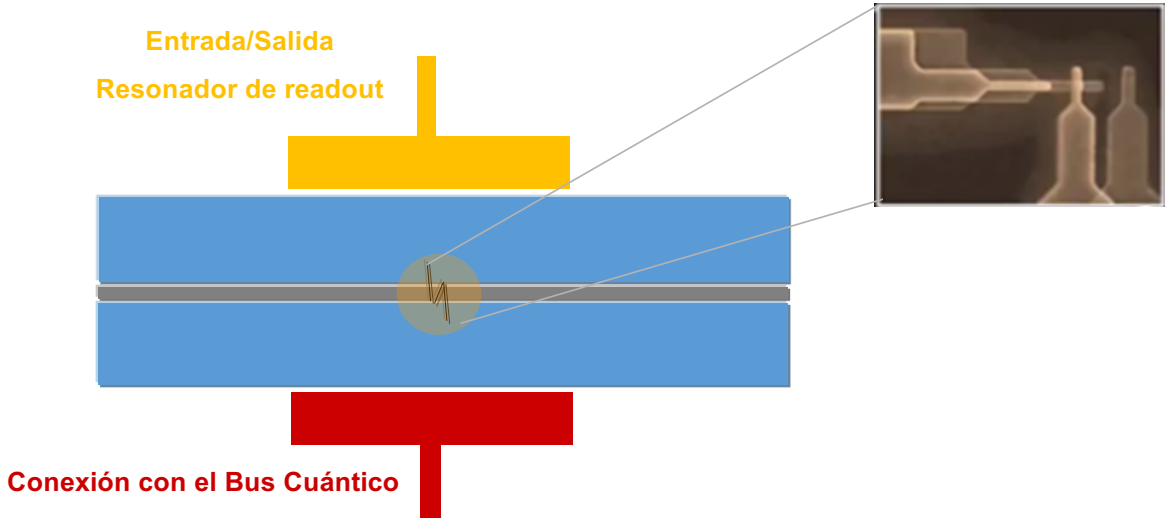


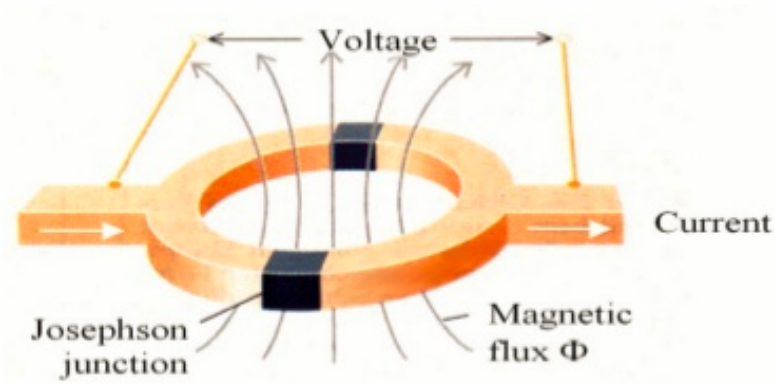


Circuitos Superconductores

- Superposición de Corrientes alrededor de un Superconductor
- Potencialmente fáciles de fabricar
- Tiempos de coherencia muy pequeños
- Requieren temperaturas de operación muy bajas.

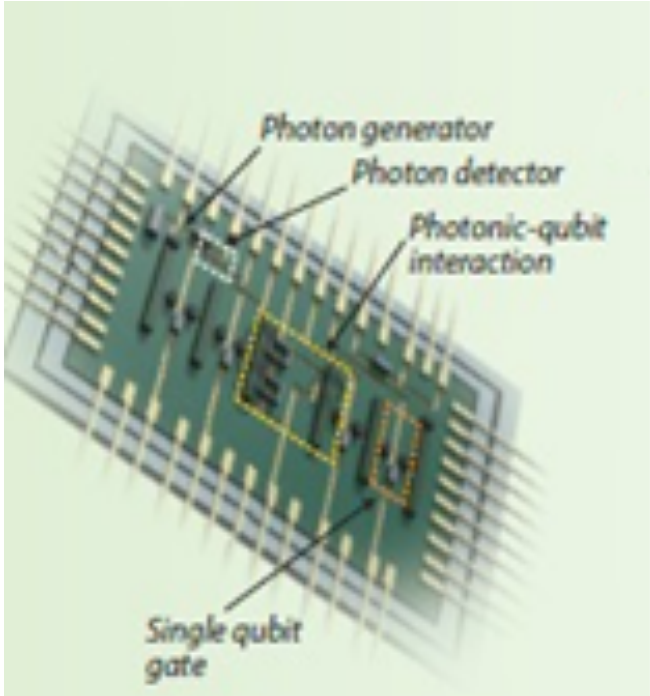


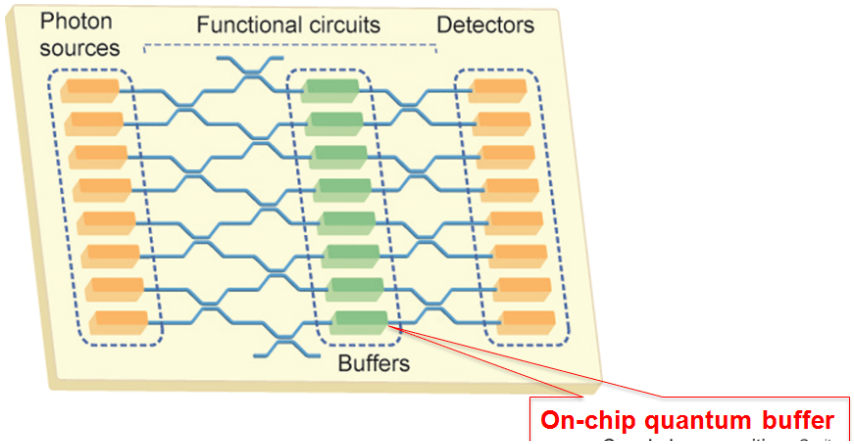




Circuitos Fotónicos

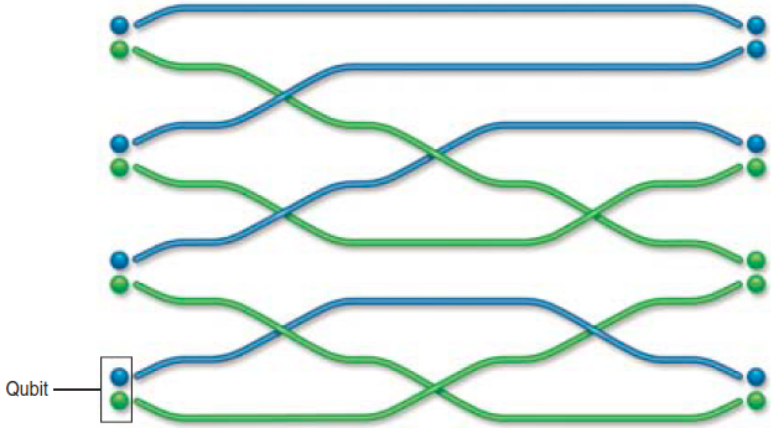
- Los qubits son fotones en circuitos de silicio.
- La redundancia de qubits se minimiza por la resistencia del fotón a la interferencia,
- Construcción de puetas lógicas fotónicas muy complicada
- La creación de fuentes de fótones individuales es un gran reto tecnológico

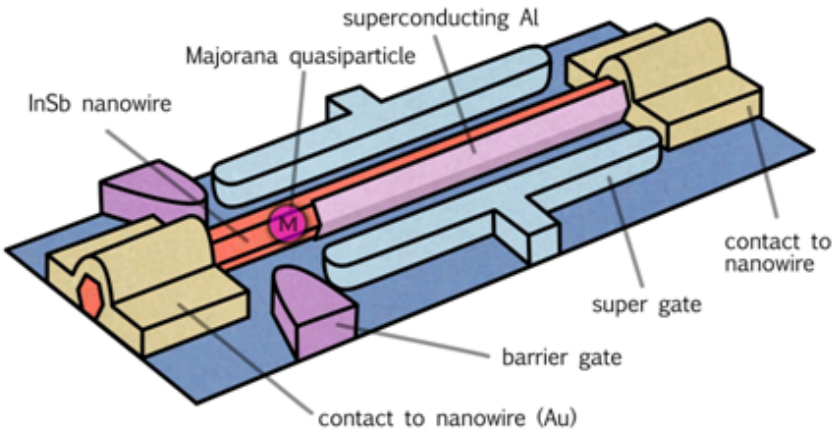




Qubits Topológicos

- Se utilizan particulas denominadas anyones
- Partícula de Majorana. Demostrada su existencia en 2012
- Se observan en montajes experimentales muy especificos (efecto Hall Cuantico Fraccionario)
- Gran Ventaja: Inmunes a pequeñas perturbaciones. Reducen drásticamente la corrección de errores.

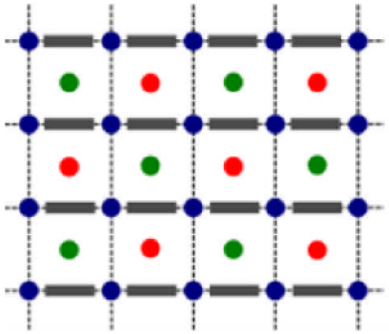




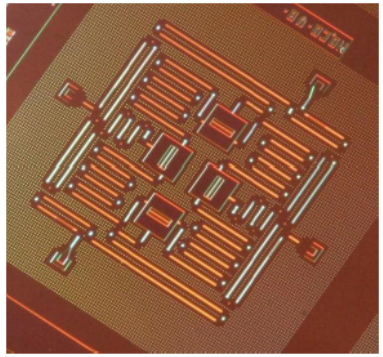
Los Computadores Cuánticos Actuales

INFRAESTRUCTURAS DE PROCESADORES CUÁNTICOS

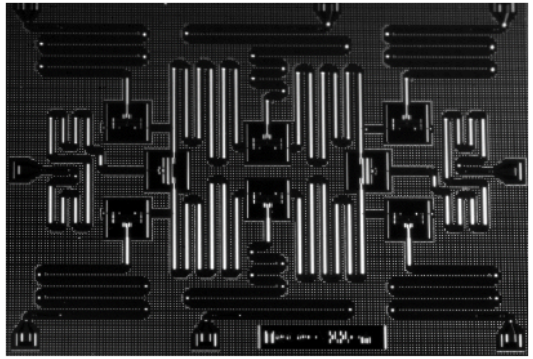
Implementation of quantum network



Resonator
Code qubit
X ancilla qubit
Z ancilla qubit

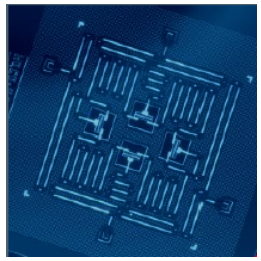


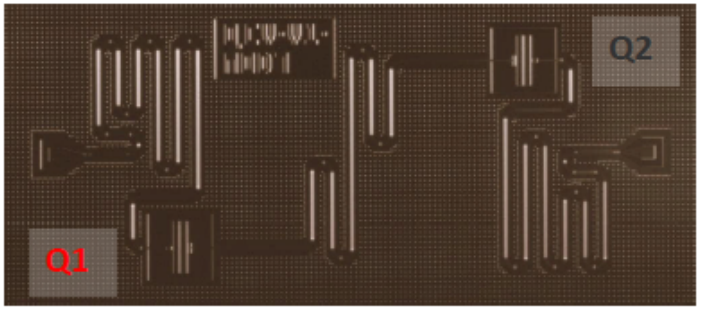
4Qubits/4Bus/4Readouts



8Qubits/4Bus/8Readouts

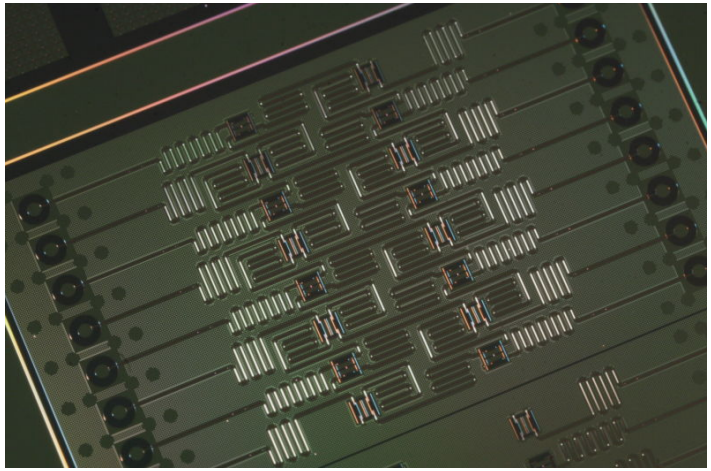
El esquema de acoplamiento del circuito QED se ha convertido en el estándar para acoplar y leer qubits superconductores a medida que los sistemas continúan escalando.





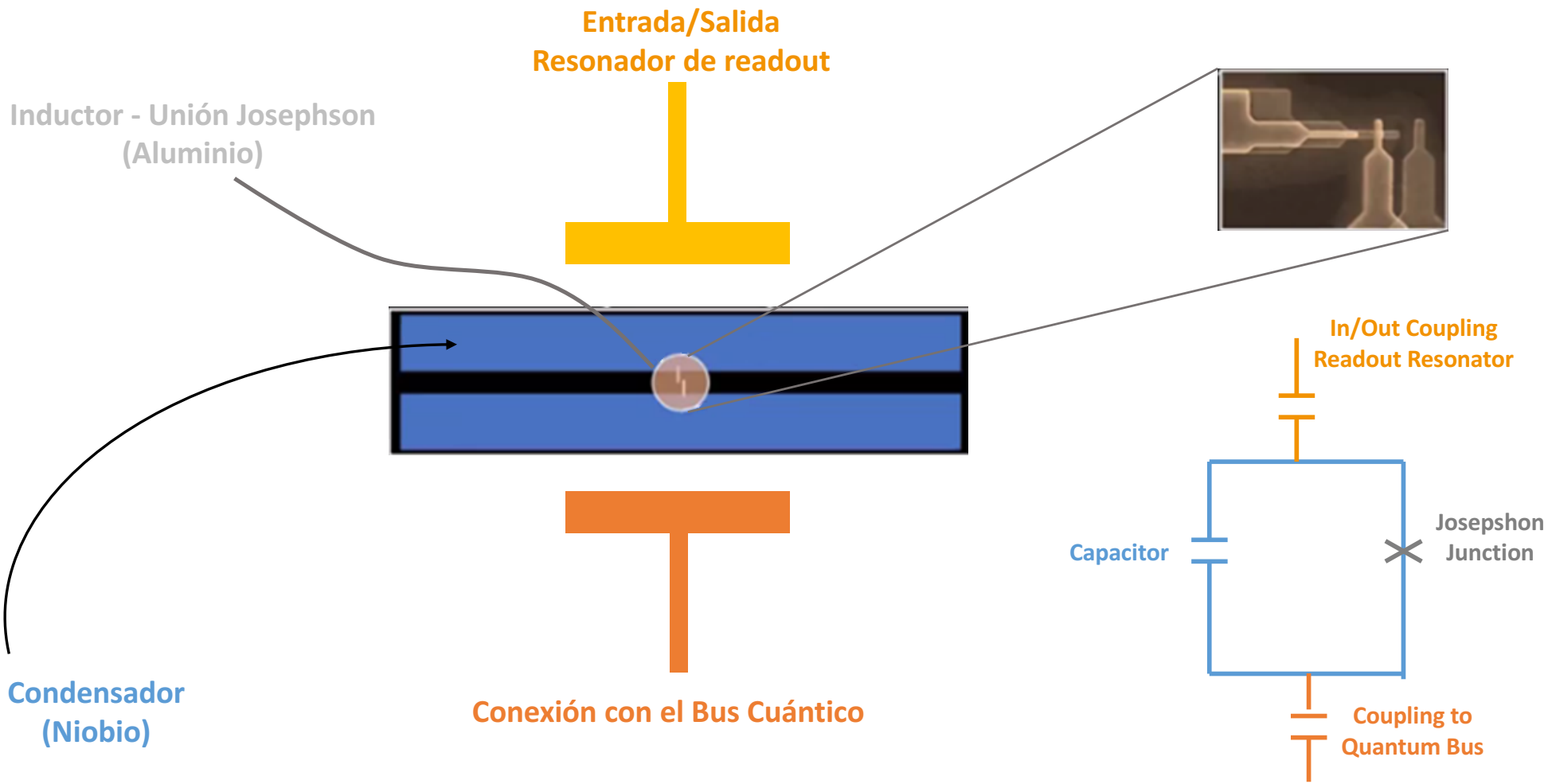
2Qubits/1Bus/2Readouts

Circuito QED: Un qubit superconductor está fuertemente interaccionado con un solo fotón en una cavidad de microondas.

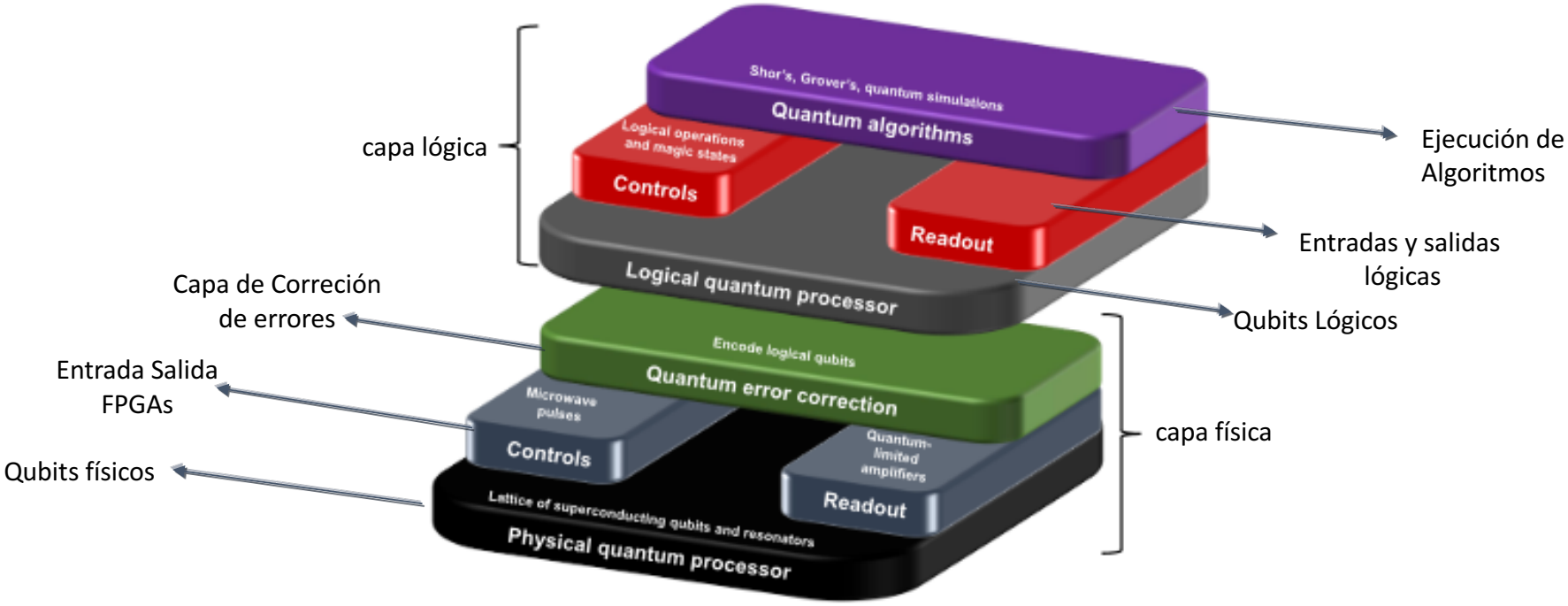


IBM's new 17-qubit quantum computer.

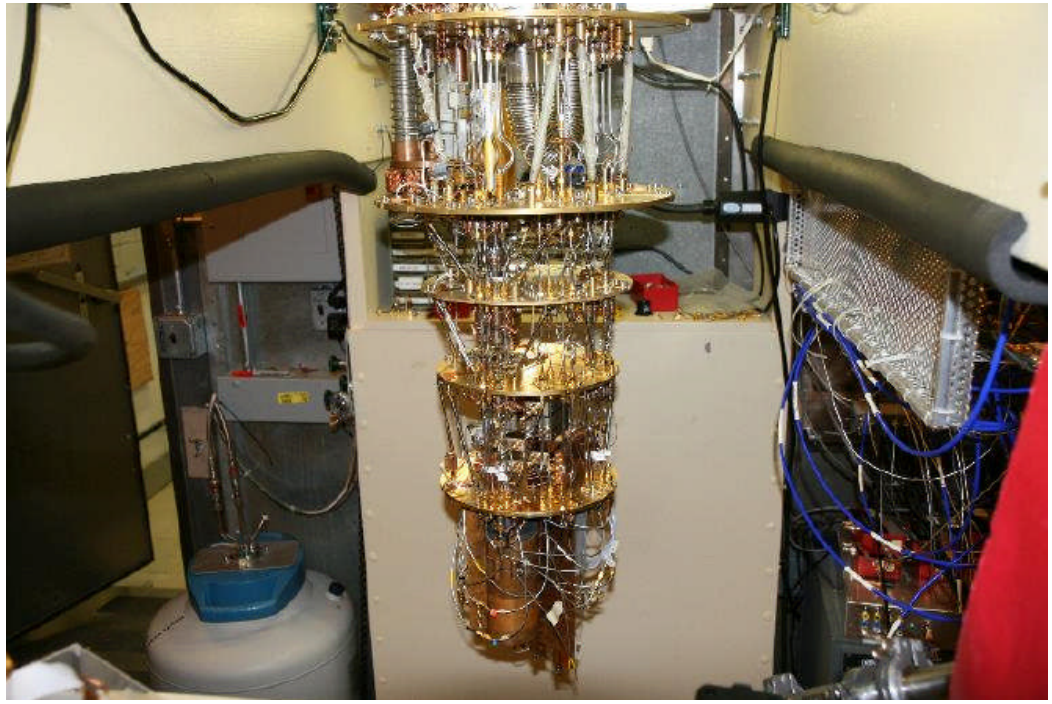
QUBITS SUPERCONDUCTORES

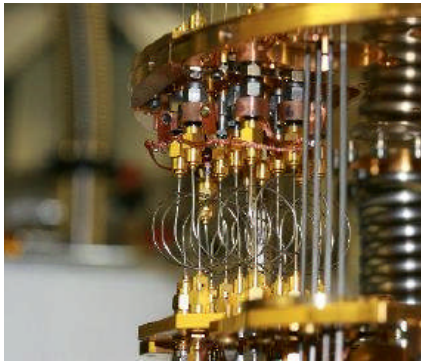


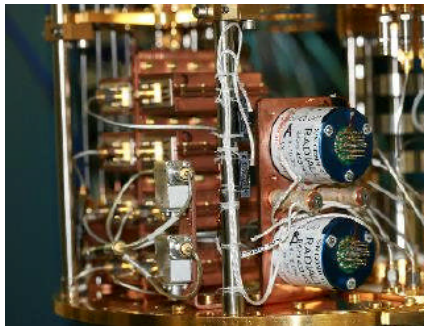
Estructura en capas de un procesador cuántico

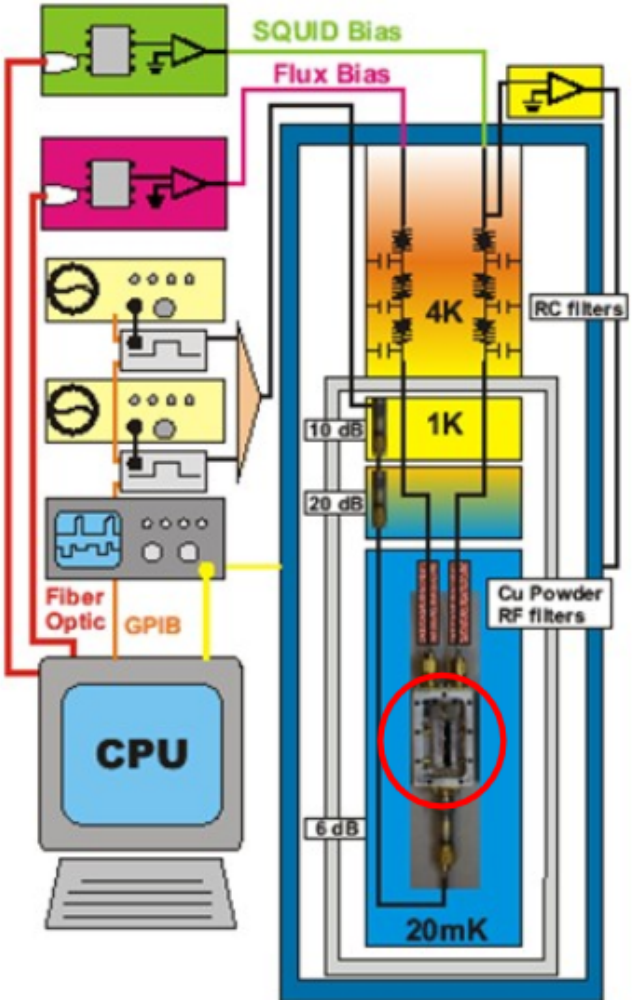


El Sistema de Refrigeración









Mezcla de ^{3}He y ^{4}He → Se enfria por debajo de 800mK

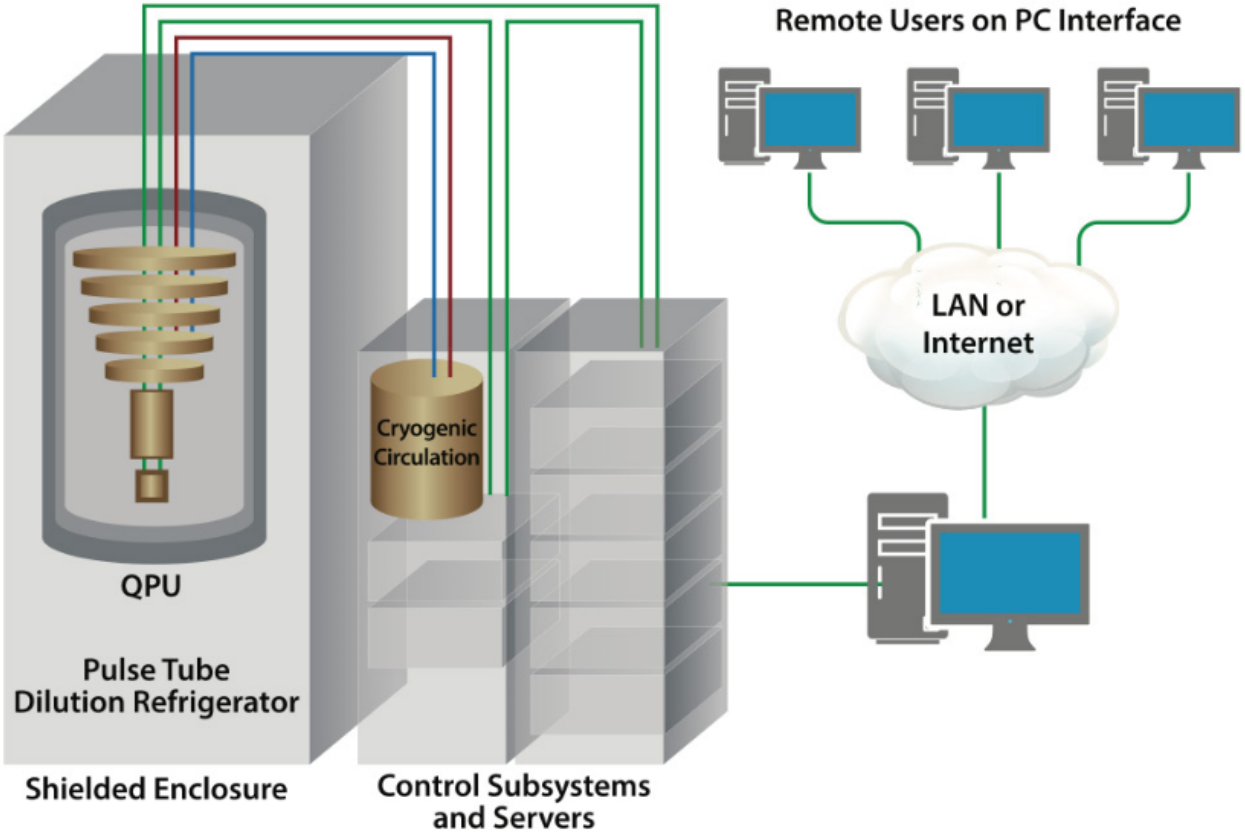
Tras el proceso de dilución, se alcanzan temperaturas de 5mK

Tecnología Industrial, existen varios proveedores en el mercado.

http://www.bluefors.com/

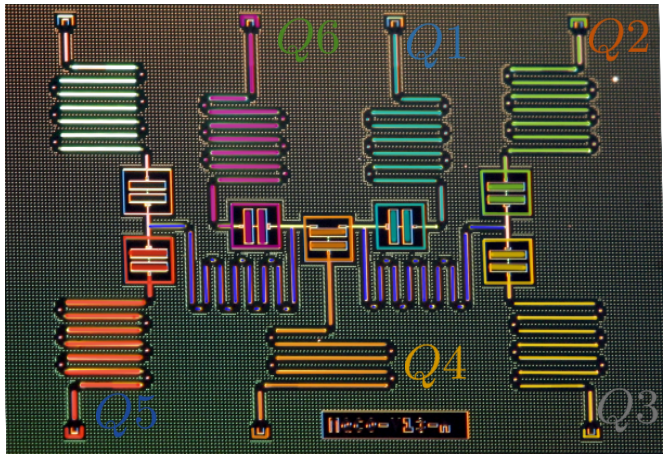
Visión Global

- Un computador cuántico se puede integrar con data centers estándar y entornos de alto rendimiento, tanto como cloud privada o público.



ARQUITECTURAS HÍBRIDAS

Los algoritmos híbridos pueden utilizarse para resolver problemas donde el objetivo sea minimizar la energía del sistema



Preparar un estado de prueba |ψ(θ)> y calcular su energía E(θ)

Utilizar un optimizador clásico para elegir un nuevo valor de θ

Ventajas:

- Utilización de circuitos cortos que se ejecuten en los tiempos de coherencia
- Mejorar las mejores estimaciones clásicas utilizando estados cuánticos

Los Modelos de Programación

Principales enfoques

Existen diversos iniciativas para abordar el tratamiento de problemas con un enfoque cuántico.

Las más populares (No las únicas) son:

- El Modelo de Circuitos
- El Temple Cuántico
- La Computación Topológica

El Modelo de Circuitos

EL MODELO DE CIRCUITOS

Puerta NOT

X

Transforrma el estado |0> en el estado |1> y viceversa. Rotación de π radianes alrededor del eje X

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

Puerta Z

Z

Deja inalterado el estado |0> y cambia el estado |1> a -|1>

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

Puerta H

H

Transforma cualquiera de los estados base en una combinación de ambos. Rotación de $\pi/2$ radianes alrededor del eje X y del eje Z

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

Puerta fase

S

Realiza una rotación de $\pi/2$ radianes alrededor del eje Z

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

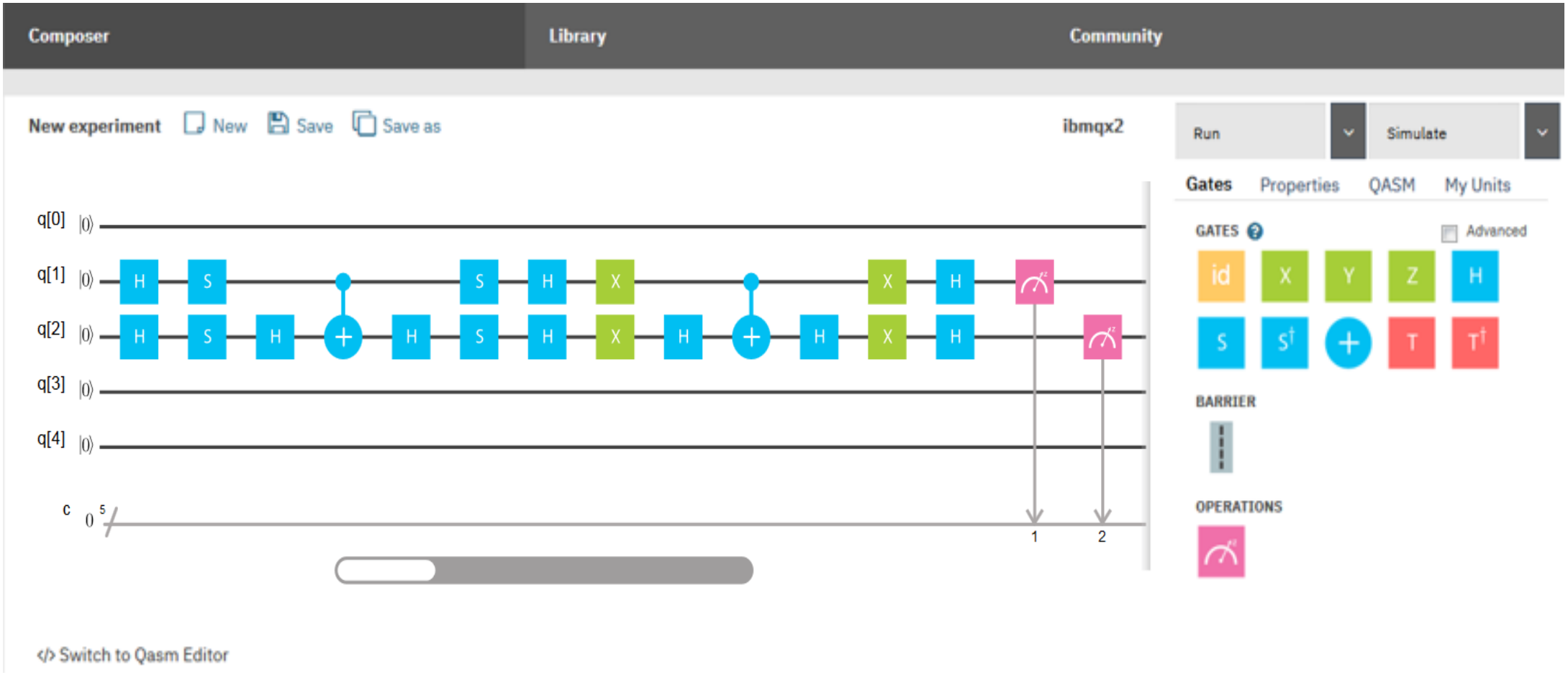
Puerta $\pi/4$

T

Realiza una rotación de $\pi/4$ radianes alrededor del eje Z

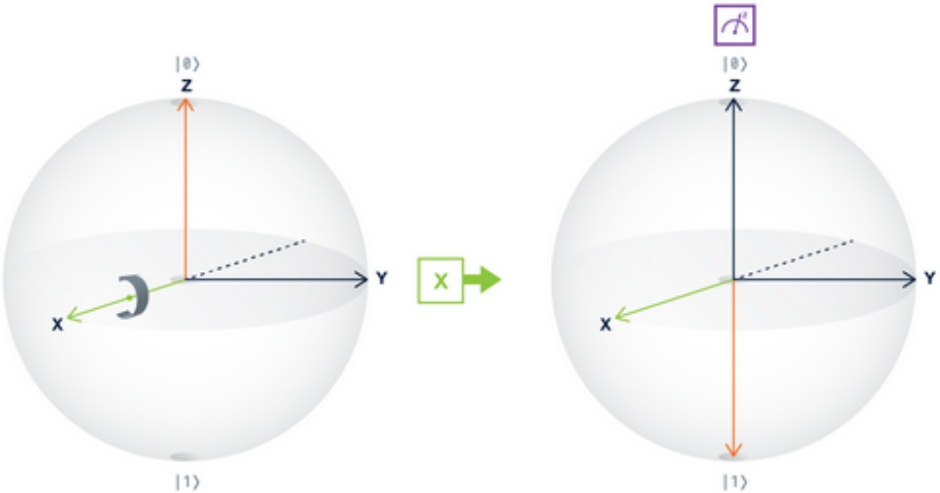
$$\begin{pmatrix} 1 & 0 \\ 0 & e^{i\pi/4} \end{pmatrix}$$

EL MODELO DE CIRCUITOS

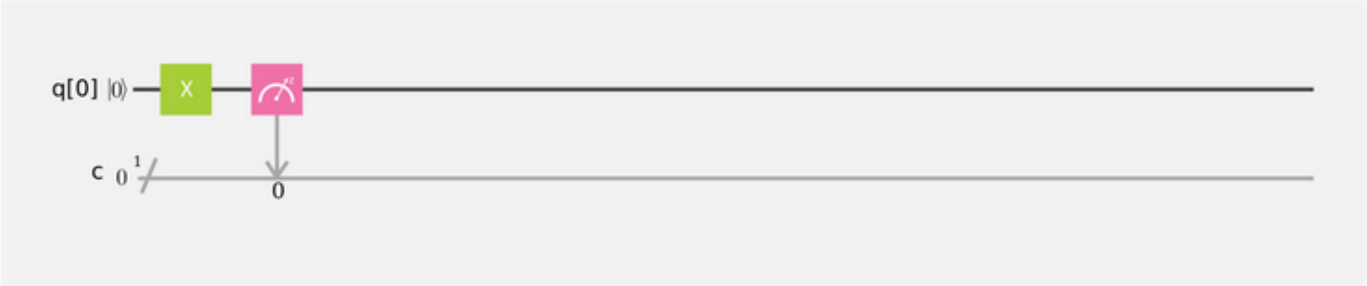


Puerta X

Cambio de estado base

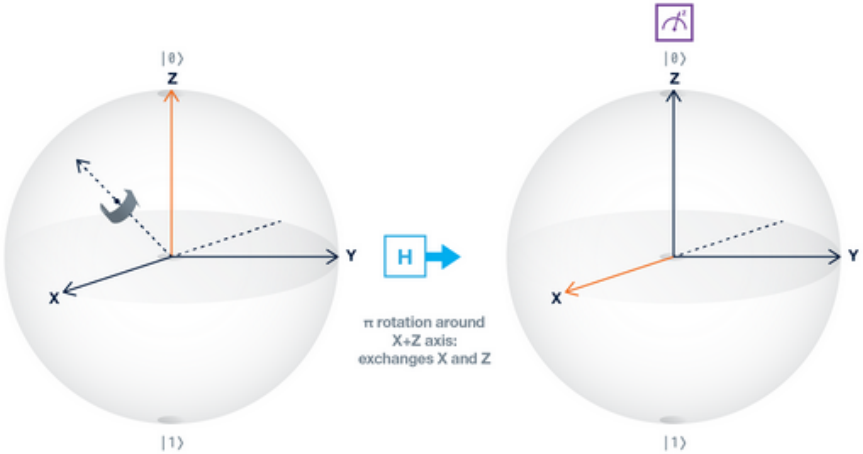


- **Transforrma el estado $|0\rangle$ en el estado $|1\rangle$ y viceversa**
- **Significa un rotaión de π radianes alrededor del eje X**

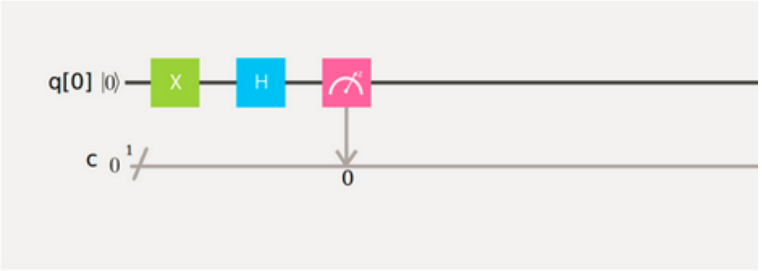


Puerta H

Superposición de estados base



- **Transforma cualquiera de los estados base $|0\rangle$ $|1\rangle$ o en una combinación de ambos.**
- **Supone una rotación de $\pi/2$ radianes alrededor del eje X y del eje Z**

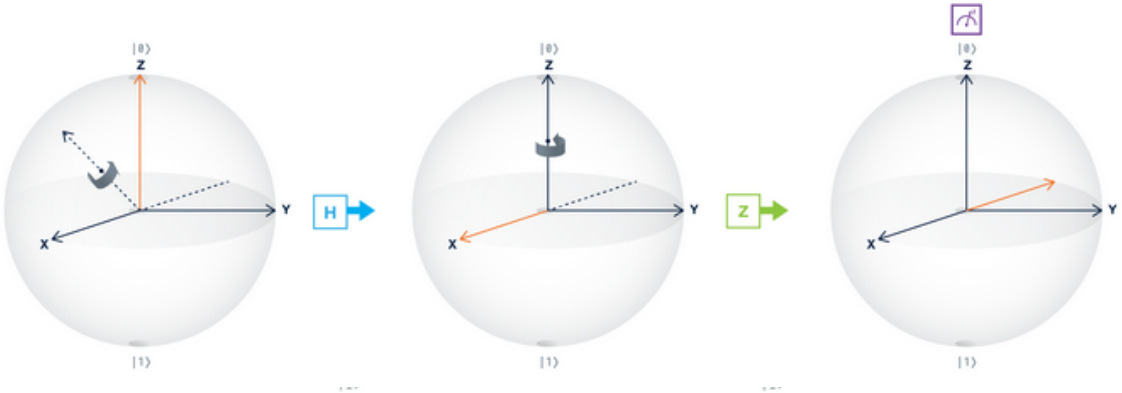


No se puede medir sobre el eje Z, pero se puede utilizar la puerta H para hacer un cambio de base

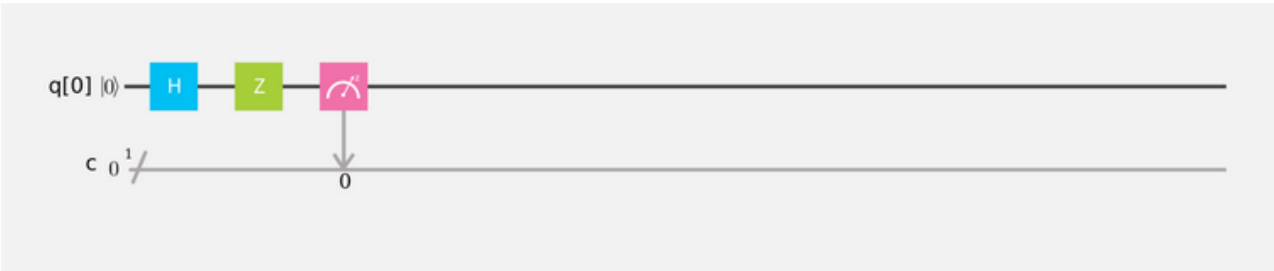
The Z Gate

Cambio de Fase - π

Z



- **Realiza una rotación de π radianes alrededor del eje Z**



Implementaciones Comerciales

IBM

rigetti

Microsoft

D:wave

IBM

Procesadores Cuánticos de 5 Qubits

Accesibles a traves de IBM Qua

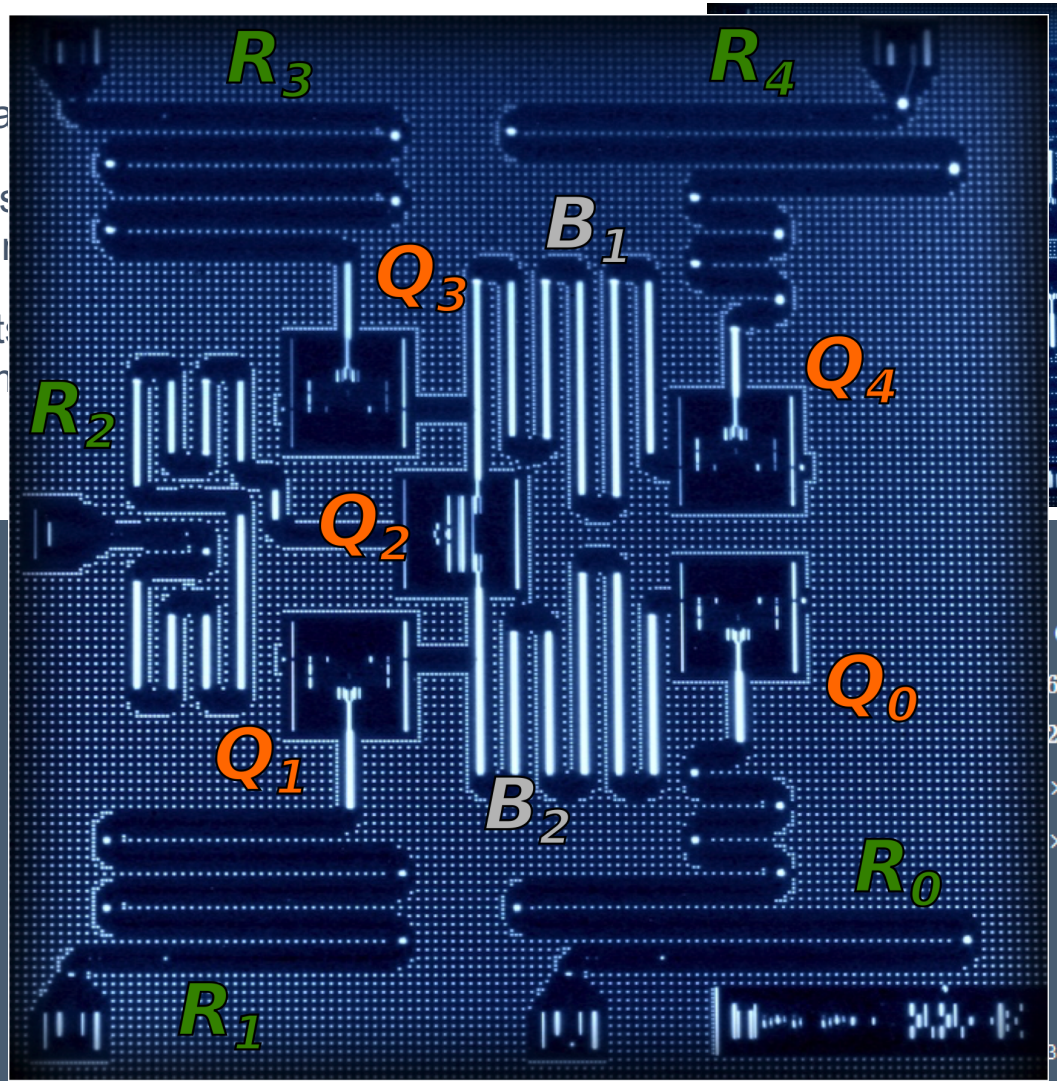
Q_n – **Qubits**

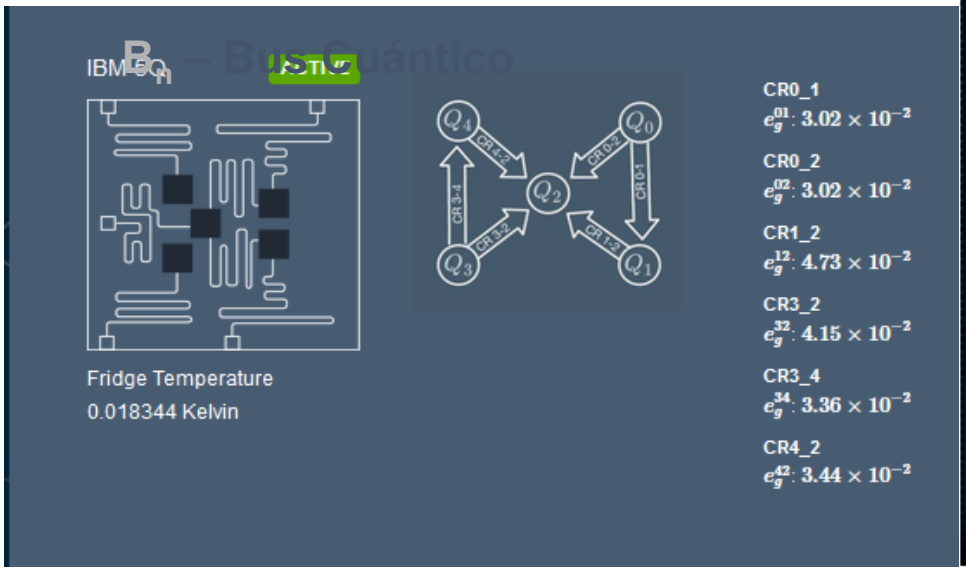
IBM QX es una iniciativa que permite a los
experimentos en el procesador cuar

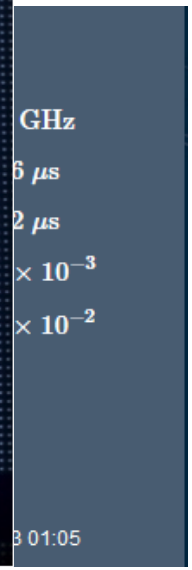
Dos Procesadores Cuánticos de 5 qubits
visualmente con IBM QX o en modo com
Python

R_n – **Readouts**

B_n – **Bus Cuántico**







Qiskit

- Framework de desarrollo de software para la creación de programas que puedan ejecutarse sobre un computador cuántico.

- Qiskit Terra provides the foundational roots for our software stack. Within Terra is a set of tools for composing quantum programs at the level of circuits and pulses, optimizing them for the constraints of a particular physical quantum processor, and managing the batched execution of experiments on remote-access backends. Terra is modularly constructed, simplifying the addition of extensions for circuit optimizations and backends. We welcome your contributions!

Qiskit

- Framework de desarrollo de software para la creación de programas que puedan ejecutarse sobre un computador cuántico.

- **Qiskit Terra.** Software a bajo nivel en contacto directo con los procesadores y los circuitos electrónicos y los pulsos RF que componen el computador cuántico
- **Qiskit Air**. Simulación de alto rendimiento para circuitos cuánticos.
- **Qiskit Aqua**. Algoritmos Cuánticos.
- **Qiskit Ignis.** Tratamiento de Errores

Qiskit Terra

- Software a bajo nivel en contacto directo con los procesadores y los circuitos electrónicos y los pulsos RF que componen el computador cuántico

```
from qiskit import QuantumCircuit, Aer, execute

qc = QuantumCircuit(2, 2)

qc.h(0)
qc.cx(0, 1)
qc.measure([0, 1], [0, 1])

backend = Aer.get_backend('qasm_simulator')
job_sim = execute(qc, backend)
sim_result = job_sim.result()

print(sim_result.get_counts(qc))
```

Qiskit Aer

- Simuladores optimizados en C++ para ejecutar circuitos compilados en Qiskit Terra.
- También dispone de modelos de ruido configurables para la realización de simuladores más realistas.

```python
from qiskit import QuantumCircuit, execute, Aer, IBMQ
from qiskit.providers.aer import noise

# Choose a real device to simulate
IBMQ.load_account()
provider = IBMQ.get_provider(group='open')
device = provider.get_backend('ibmq_16_melbourne')
properties = device.properties()
coupling_map = device.configuration().coupling_map

# Generate an Aer noise model for device
noise_model = noise.device.basic_device_noise_model(properties)
basis_gates = noise_model.basis_gates

# Generate a quantum circuit
qc = QuantumCircuit(2, 2)

qc.h(0)
qc.cx(0, 1)
qc.measure([0, 1], [0, 1])

# Perform noisy simulation
backend = Aer.get_backend('qasm_simulator')
job_sim = execute(qc, backend,
                  coupling_map=coupling_map,
                  noise_model=noise_model,
                  basis_gates=basis_gates)

sim_result = job_sim.result()
print(sim_result.get_counts(qc))
```

Qiskit Aqua

- Contiene librerias de algoritmos cúánticos sobre las que construir aplicaciones.
- Química, AI, Problemas de Optimización, Finannzas, etd.
- De diseño extensible, admite la incorporación de nuevos algorimtos y rutinas.

```
from qiskit import Aer
from qiskit.aqua.components.oracles import LogicalExpressionOracle
from qiskit.aqua.algorithms import Grover

sat_cnf = """
c Example DIMACS 3-sat
p cnf 3 5
-1 -2 -3 0
1 -2 3 0
1 2 -3 0
1 -2 -3 0
-1 2 3 0
"""

backend = Aer.get_backend('qasm_simulator')
oracle = LogicalExpressionOracle(sat_cnf)
algorithm = Grover(oracle)
result = algorithm.run(backend)

print(result["result"])
```

IGNIS

- Recoge funcionalidades para entender el ruido cuántico y realizar el tratamiento de errores.
  - Caracterización – Medida de los parámetros del ruido
  - Verificación – Verificar el rendimiento de las puertas y los pequeños circuitos o rutinas.
  - Mitigación – Ejecución de circuitos de calibración que se aplican a rutinas para compensar errores.

EXTENSIONS: MARKETPLACE

qiskit

qiskit-vscode 0.5.5
Simplifying Qiskit to make developing ...
qiskit — Installing

Qiksit-Snippets 0.0.8
Python snippets for IBM's quantum pro...
Akash — Install

Driver.cs | *Extension: qiskit-vscode* | Operation.qs

qiskit-vscode

qiskit.qiskit-vscode

qiskit | 6.488 | ★★★★☆ | Repository

Simplifying Qiskit to make developing quantum circuits and applications faster

Installing

Details Contributions Dependencies

Qiskit VSCode Extension

> Simplifying Qiskit to make developing quantum circuits and applications faster.

A Visual Studio Code extension with support for Qiskit and the OpenQASM language. It provides some useful commands to make easier launching jobs and visualizing results among others.

[Extension Development Host] - Qiskit.py

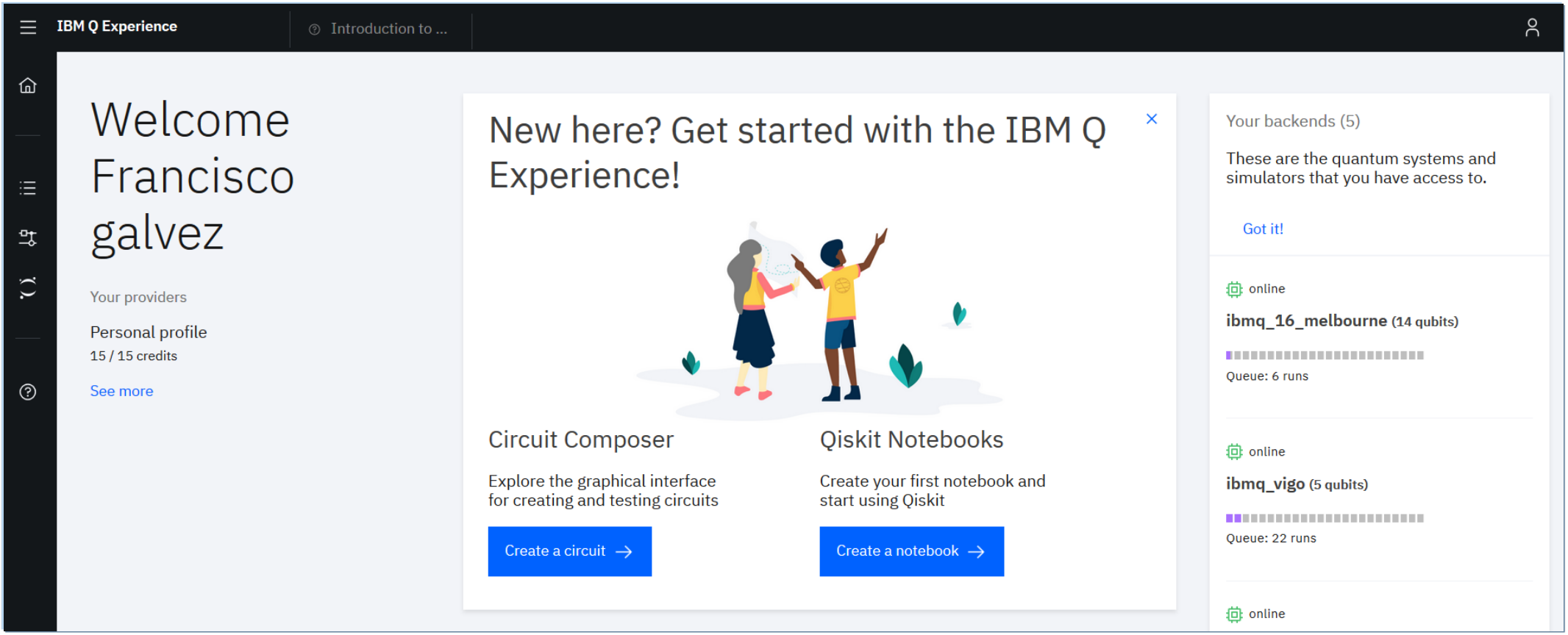
test.qasm | Qiskit.py

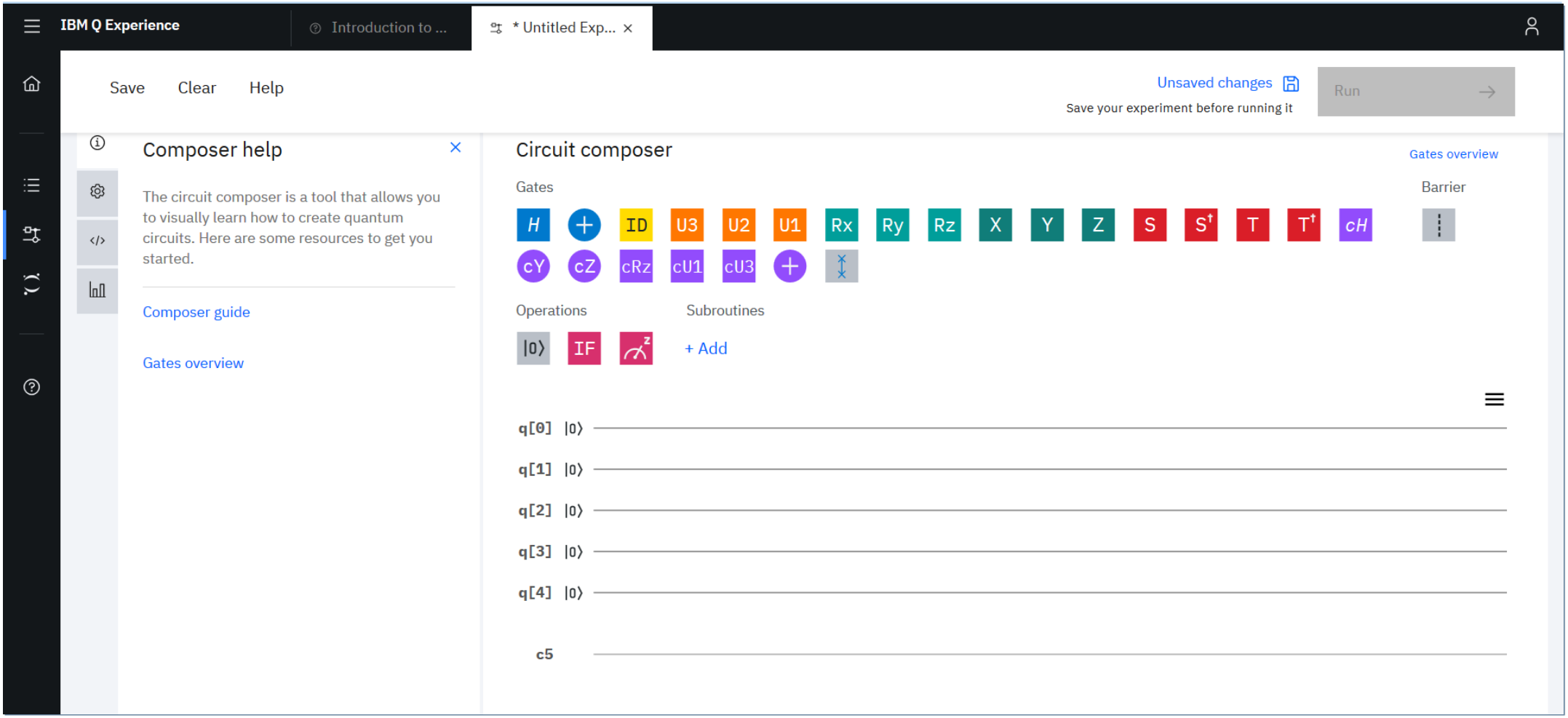
Yes

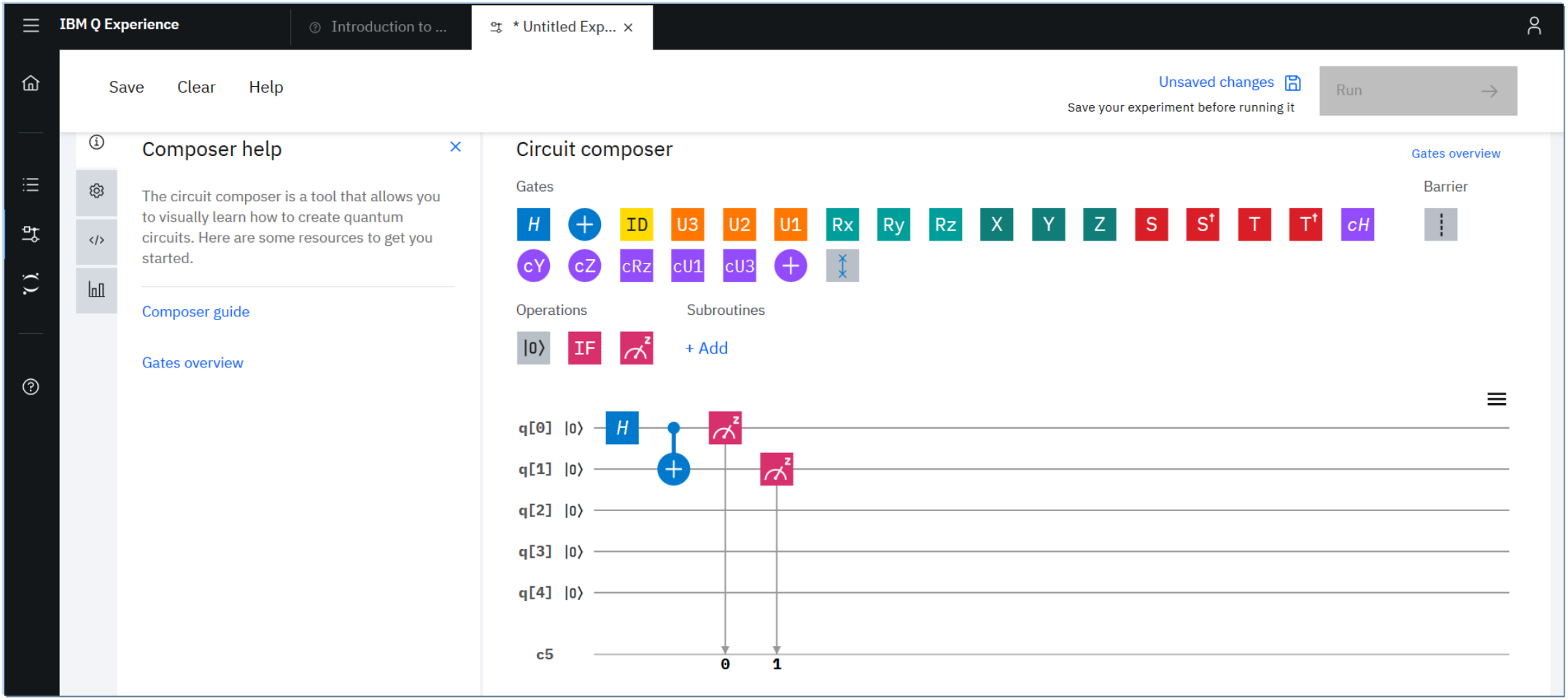
There's a new mandatory qiskit release: 0.5.3. You must upgrade it to enjoy this extension
(Press 'Enter' to confirm or 'Escape' to cancel)

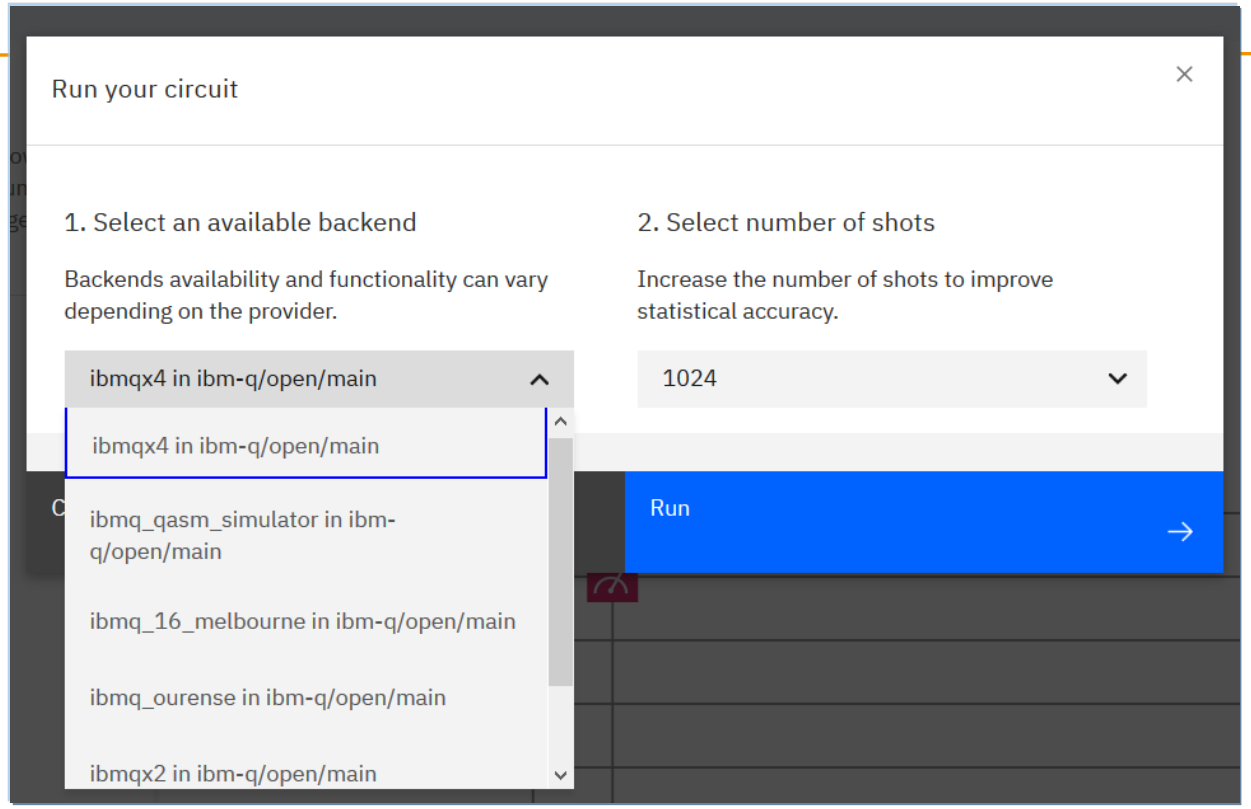
Run this Python

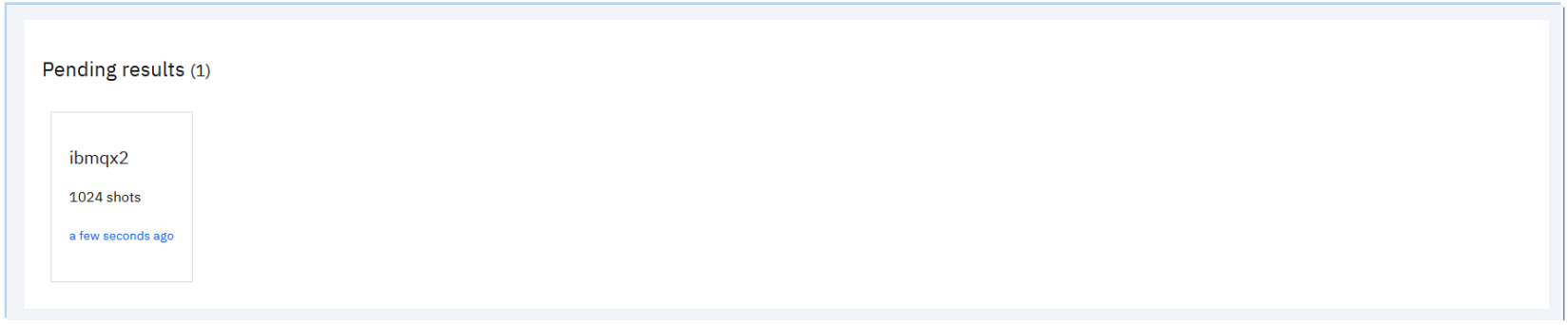
```
from qiskit import ClassicalRegister, Quan
from qiskit import QuantumCircuit, execute
def main():
    q = QuantumRegister(2)
    c = ClassicalRegister(2)
    qc = QuantumCircuit(q, c)
    qc.h(q[0])
    qc.cx(q[0], q[1])
```

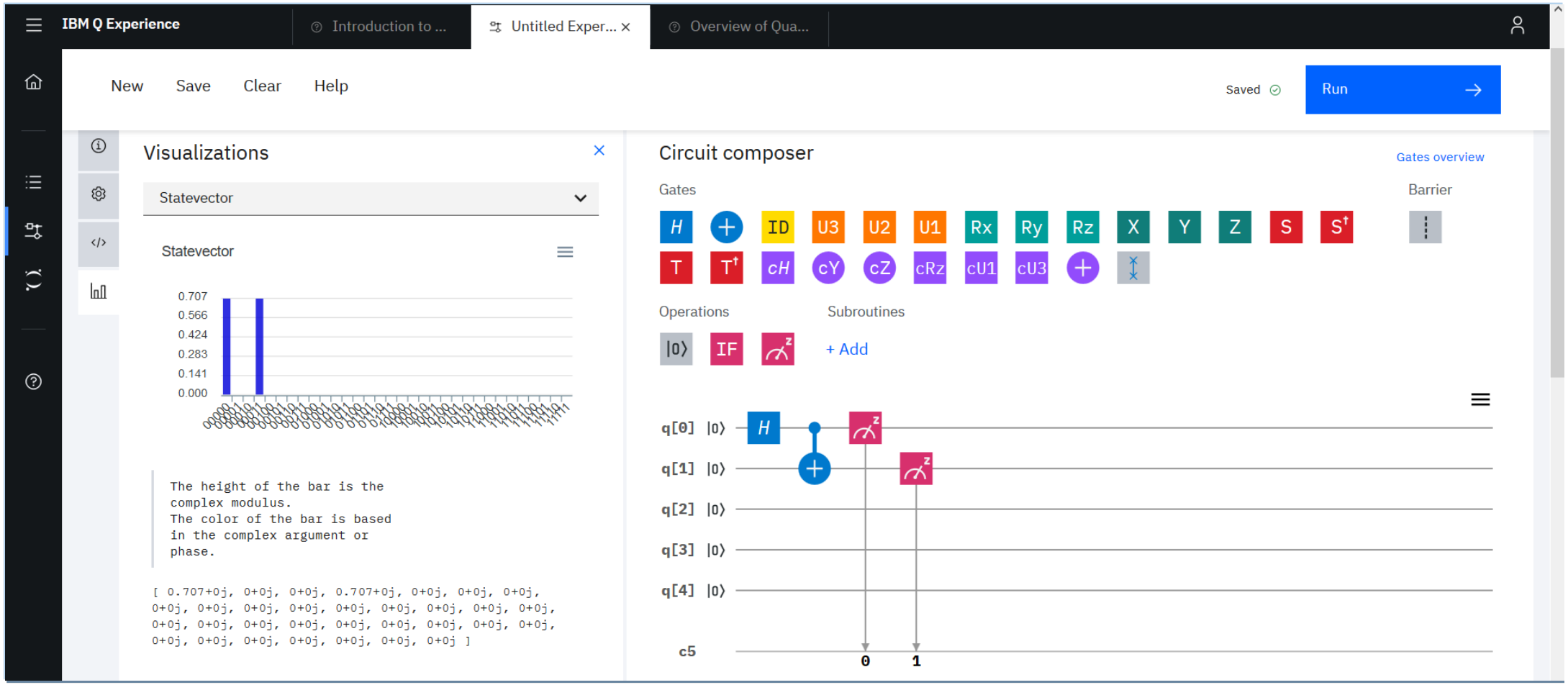

IBM Q Experience
Introduction to ...
Welcome Francisco galvez
Your providers
Personal profile
15 / 15 credits
See more
New here? Get started with the IBM Q Experience!
Circuit Composer
Explore the graphical interface for creating and testing circuits
Create a circuit
Qiskit Notebooks
Create your first notebook and start using Qiskit
Create a notebook
Your backends (5)
These are the quantum systems and simulators that you have access to.
Got it!
online
ibmq_16_melbourne (14 qubits)
Queue: 6 runs
online
ibmq_vigo (5 qubits)
Queue: 22 runs
online

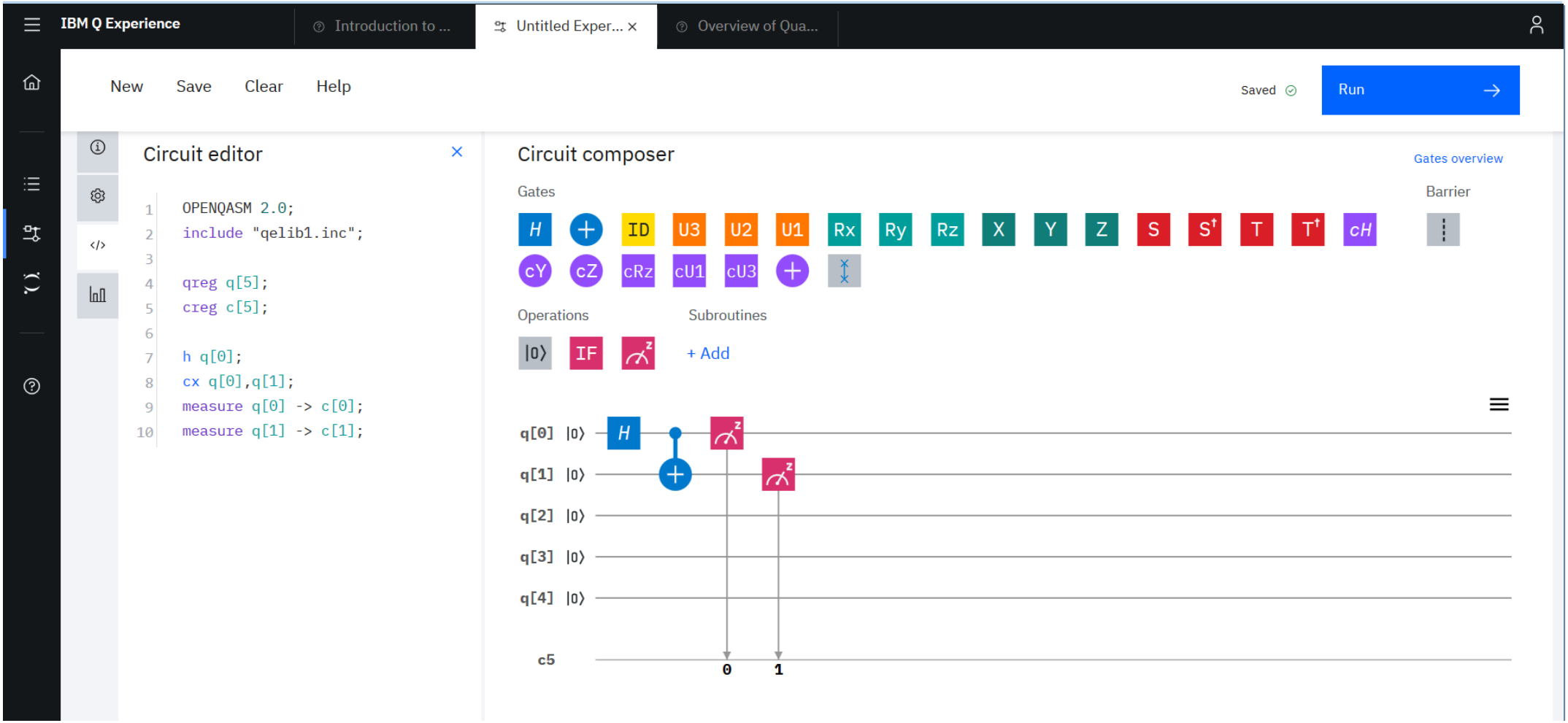

IBM Q Experience
Introduction to ...
* Untitled Exp... ×
Save
Clear
Help
Unsaved changes
Save your experiment before running it
Run
Composer help
The circuit composer is a tool that allows you to visually learn how to create quantum circuits. Here are some resources to get you started.
Composer guide
Gates overview
Circuit composer
Gates overview
Gates
Barrier
H
ID
U3
U2
U1
Rx
Ry
Rz
X
Y
Z
S
S†
T
T†
cH
cY
cZ
cRz
cU1
cU3
Operations
Subroutines
|0⟩
IF
+ Add
q[0] |0⟩
q[1] |0⟩
q[2] |0⟩
q[3] |0⟩
q[4] |0⟩
c5


IBM Q Experience
Introduction to ...
* Untitled Exp... ×
Save
Clear
Help
Unsaved changes
Save your experiment before running it
Run
Composer help
The circuit composer is a tool that allows you to visually learn how to create quantum circuits. Here are some resources to get you started.
Composer guide
Gates overview
Circuit composer
Gates overview
Gates
Barrier
H
ID
U3
U2
U1
Rx
Ry
Rz
X
Y
Z
S
S†
T
T†
cH
cY
cZ
cRz
cU1
cU3
Operations
Subroutines
|0⟩
IF
+ Add
q[0] |0⟩
q[1] |0⟩
q[2] |0⟩
q[3] |0⟩
q[4] |0⟩
c5
0
1


Run your circuit
1. Select an available backend
Backends availability and functionality can vary depending on the provider.
ibmqx4 in ibm-q/open/main
ibmqx4 in ibm-q/open/main
ibmq_qasm_simulator in ibm-q/open/main
ibmq_16_melbourne in ibm-q/open/main
ibmq_ourense in ibm-q/open/main
ibmqx2 in ibm-q/open/main
2. Select number of shots
Increase the number of shots to improve statistical accuracy.
1024
Run


Pending results (1)
ibmqx2
1024 shots
a few seconds ago


IBM Q Experience
Introduction to ...
Untitled Exper...
Overview of Qua...
New
Save
Clear
Help
Saved
Run
Visualizations
Statevector
Statevector
0.707
0.566
0.424
0.283
0.141
0.000
The height of the bar is the complex modulus.
The color of the bar is based in the complex argument or phase.
[0.707+0j, 0+0j, 0+0j, 0.707+0j, 0+0j, 0+0j, 0


IBM Q Experience
Introduction to ...
Untitled Exper...
Overview of Qua...
New
Save
Clear
Help
Saved
Run
Circuit editor
OPENQASM 2.0;
include "qelib1.inc";
qreg q[5];
creg c[5];
h q[0];
cx q[0],q[1];
measure q[0] -> c[0];
measure q[1] -> c[1];
Circuit composer
Gates overview
Gates
Barrier
H
ID
U3
U2
U1
Rx
Ry
Rz
X
Y
Z
S
S†
T
T†
cH
cY
cZ
cRz
cU1
cU3
Operations
Subroutines
|0⟩
IF
+ Add
q[0] |0⟩
q[1] |0⟩
q[2] |0⟩
q[3] |0⟩
q[4] |0⟩
c5
0
1

Elementos básicos de programa

- Quantum Program

```
QuantumProgram()
```

- Circuit

```
.create_circuit()
```

- Quantum Register

```
.create_quantum_registers()
```

- Classical Register

```
.create_classical_registers()
```

Circuit

- Primero, instanciar la clase y crear el registro

```
circuit = Q_program.get_circuit("Circuit")
```

Registers

- **Quantum Registers**

```
q2 = Q_program.create_quantum_register("q2", 2)
```

- **Classical Registers**

```
c2 = Q_program.create_classical_register("c2", 2)
```

```
# Creating registers
q2 = Q_program.create_quantum_registers("q2", 2)
c2 = Q_program.create_classical_registers("c2", 2)
```

Puertas Cuánticas

- Añadir puertas al circuito cuántico

```
# H (Hadamard) gate to the qubit 0 in the Quantum Register "qr"
circuit.h(quantum_r[0])

# Pauli X gate to the qubit 1 in the Quantum Register "qr"
circuit.x(quantum_r[1])

# Pauli Y gate to the qubit 2 in the Quantum Register "qr"
circuit.y(quantum_r[2])

# Pauli Z gate to the qubit 3 in the Quantum Register "qr"
circuit.z(quantum_r[3])

# CNOT (Controlled-NOT) gate from qubit 0 to the Qbit 2
circuit.cx(quantum_r[0], quantum_r[2])
```

El Lenguaje OpenQASM

- OpenQASM es una representacion intermedia para circuitos cuánticos
- Esta basado en QASM “quantum assembly language” (Chuang, *qasm2circ* (2005);
- Representa un bloque básico de un programa cuántico
- Foco en requisitos a corto plazo
- Pensado para que sea Hardware-agnostic

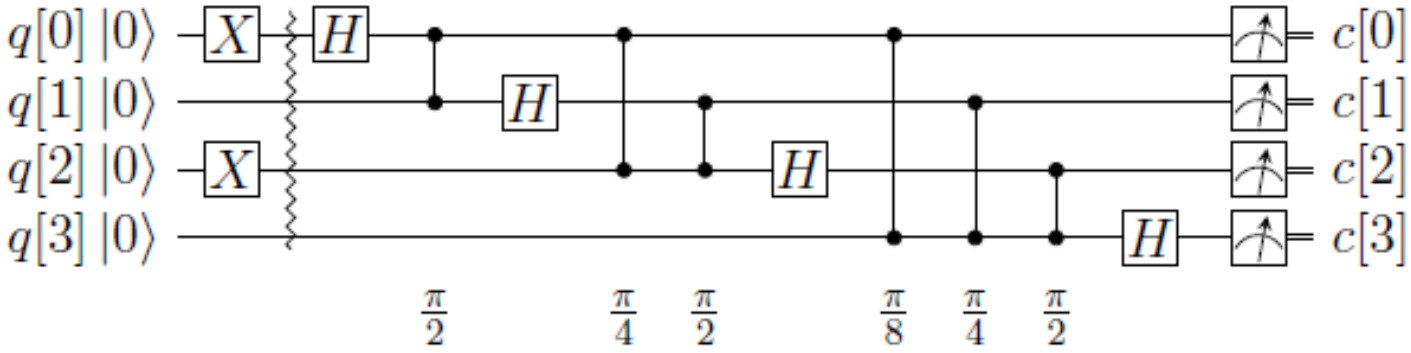
```
IBMQASM 2.0;
qreg name[size];
creg name[size];
include "filename";
gate name(params) qargs { body }
opaque name(params) qargs;
U(theta,phi,lambda) qubit|qreg;
```

```
CX qubit|qreg,qubit|qreg;
measure qubit|qreg -> bit|creg;
reset qubit|qreg;
gatename(params) qargs;
if(creg==int) qop;
barrier qargs;
```

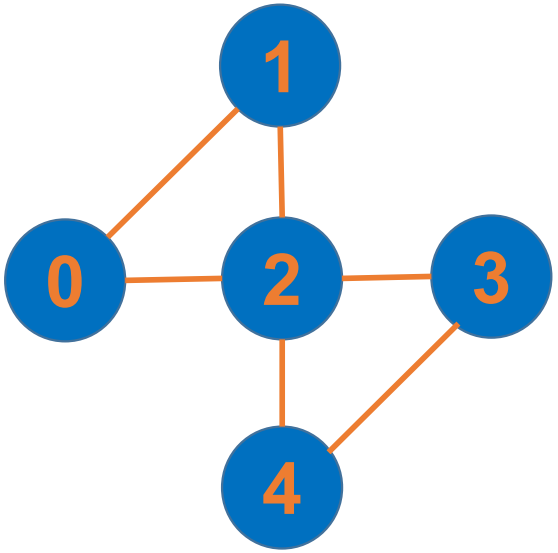
```
// quantum Fourier transform
IBMQASM 2.0;
include "qelib1.inc";
qreg q[4];
creg c[4];
x q[0];
x q[2];
barrier q;
h q[0];
cu1(pi/2) q[1],q[0];
h q[1];
cu1(pi/4) q[2],q[0];
cu1(pi/2) q[2],q[1];
h q[2];
cu1(pi/8) q[3],q[0];
cu1(pi/4) q[3],q[1];
cu1(pi/2) q[3],q[2];
h q[3];
measure q -> c;
```

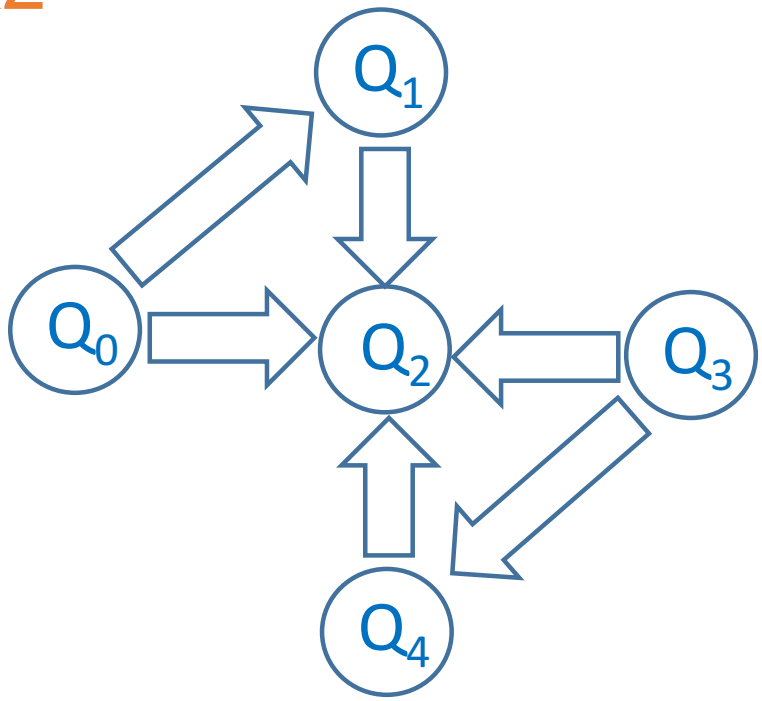
Ejemplo: Transformada Cuántica de Fourier

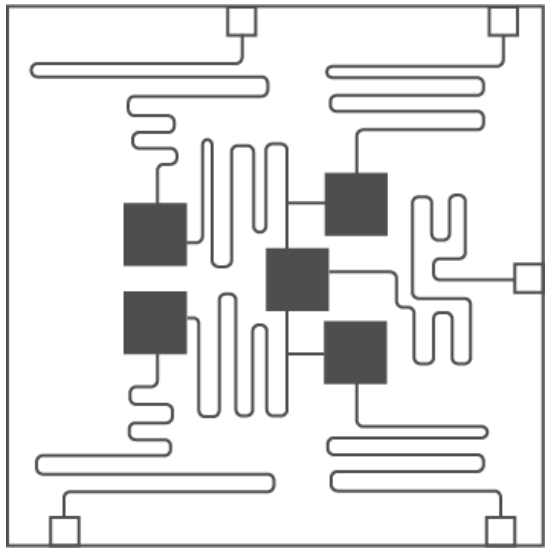
- La Transformada Cuántica de Fourier es un ejemplo de como pasar parámetros a las subrutinas de puertas.
- Este circuito aplica la QFT al estado $|q_0\, q_1\, q_2\, q_3\rangle = |1\,0\,1\,0\rangle$ y mide el resultado en la base computacional



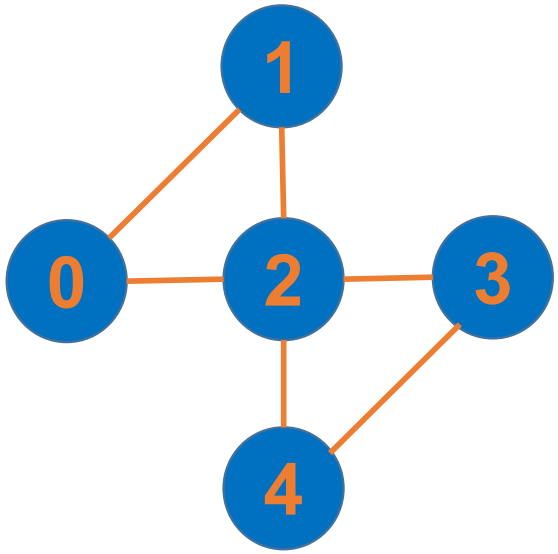
Yorktown – ibmqx2

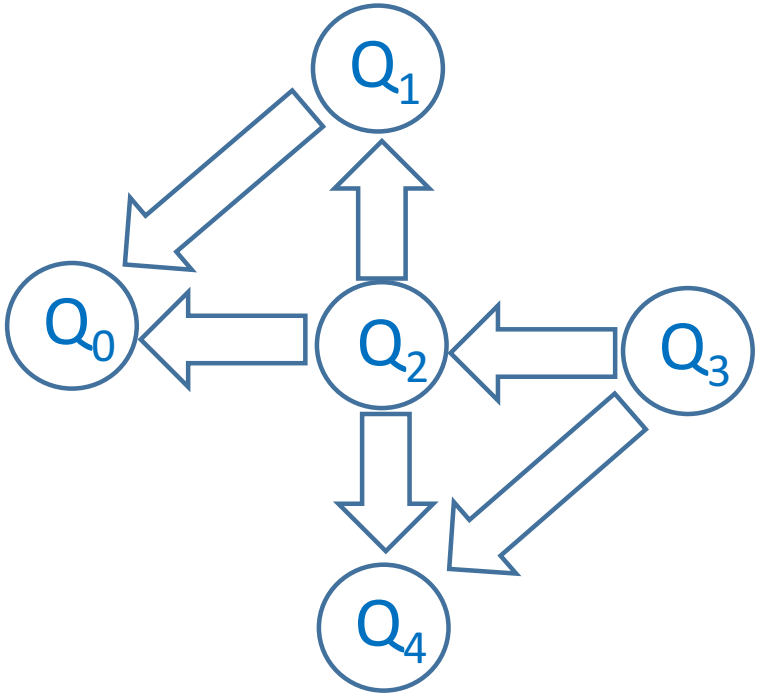


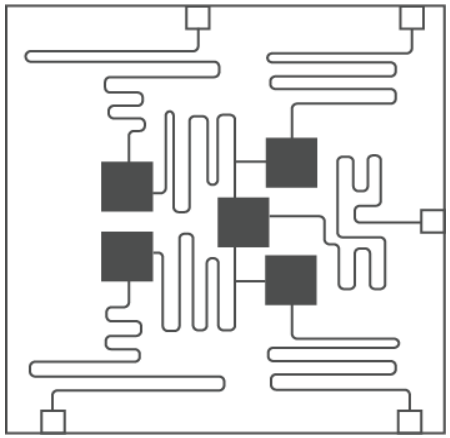


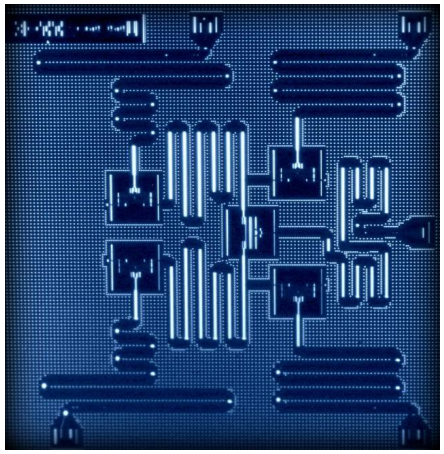


Tenerife – ibmqx4

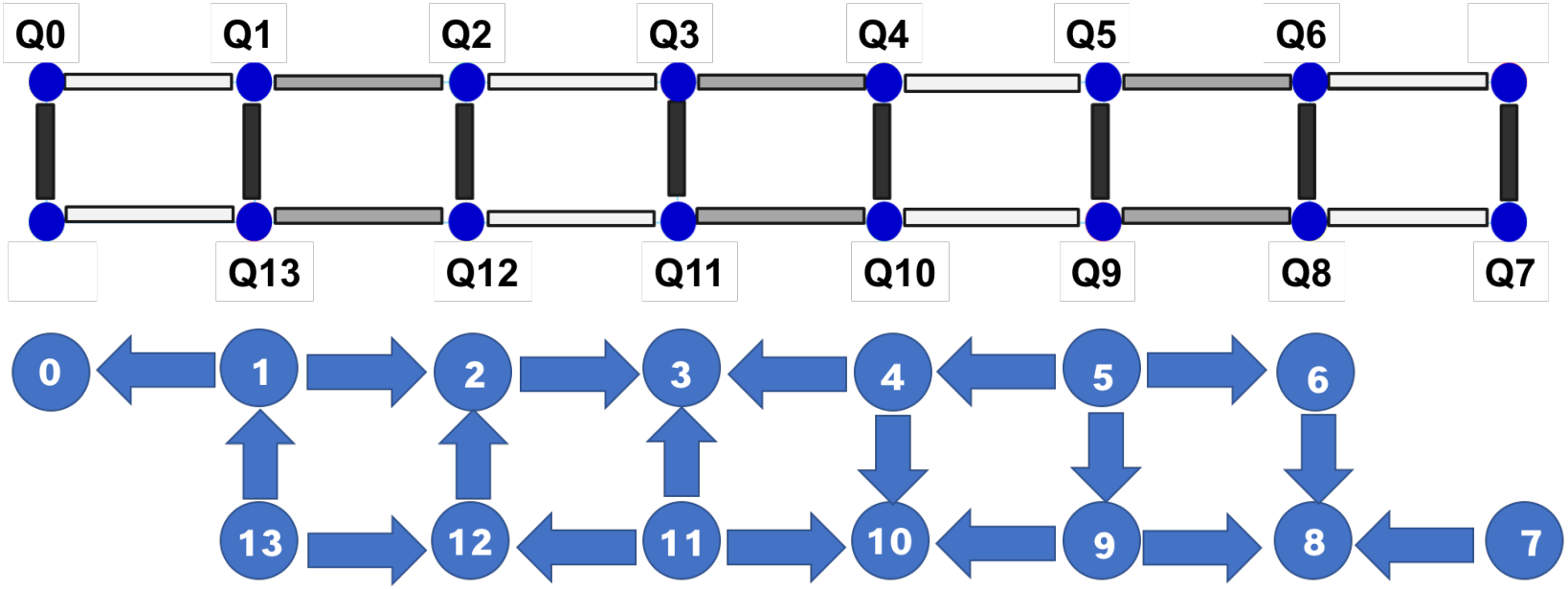


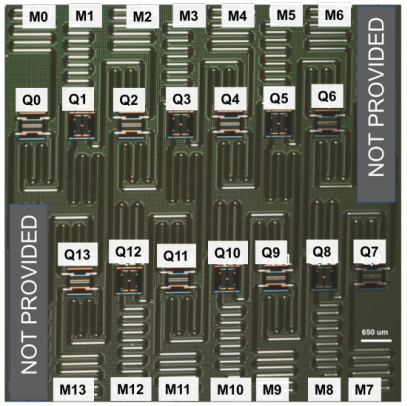




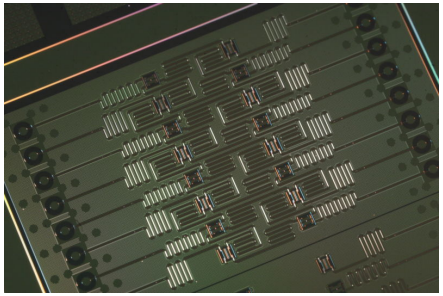


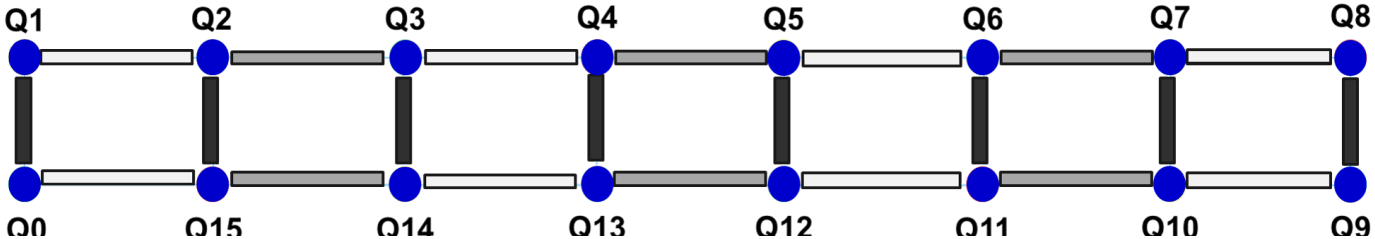
Melbourne

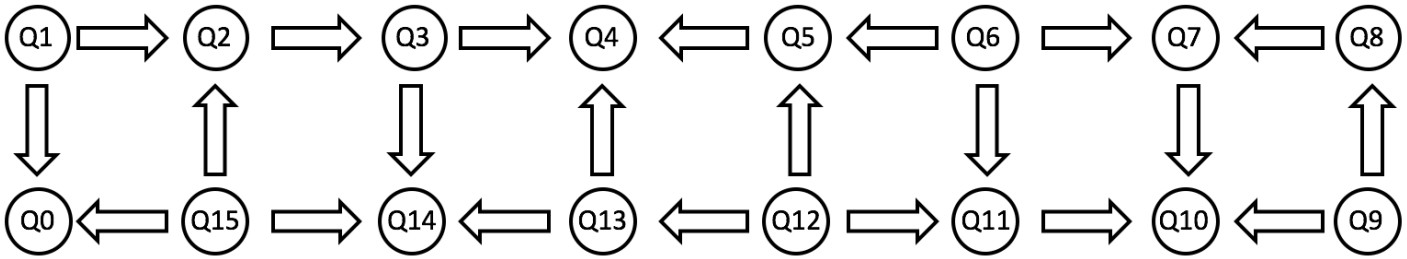

Q0
Q1
Q2
Q3
Q4
Q5
Q6
Q13
Q12
Q11
Q10
Q9
Q8
Q7
0
1
2
3
4
5
6
13
12
11
10
9
8
7

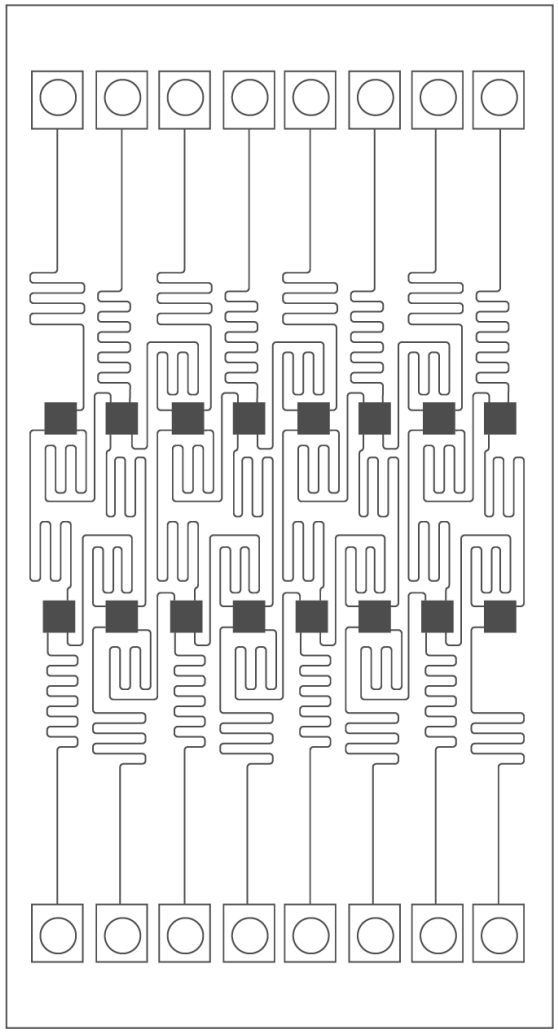

M0
M1
M2
M3
M4
M5
M6
Q0
Q1
Q2
Q3
Q4
Q5
Q6
NOT PROVIDED
NOT PROVIDED
Q13
Q12
Q11
Q10
Q9
Q8
Q7
650 um
M13
M12
M11
M10
M9
M8
M7

Rueschlikon

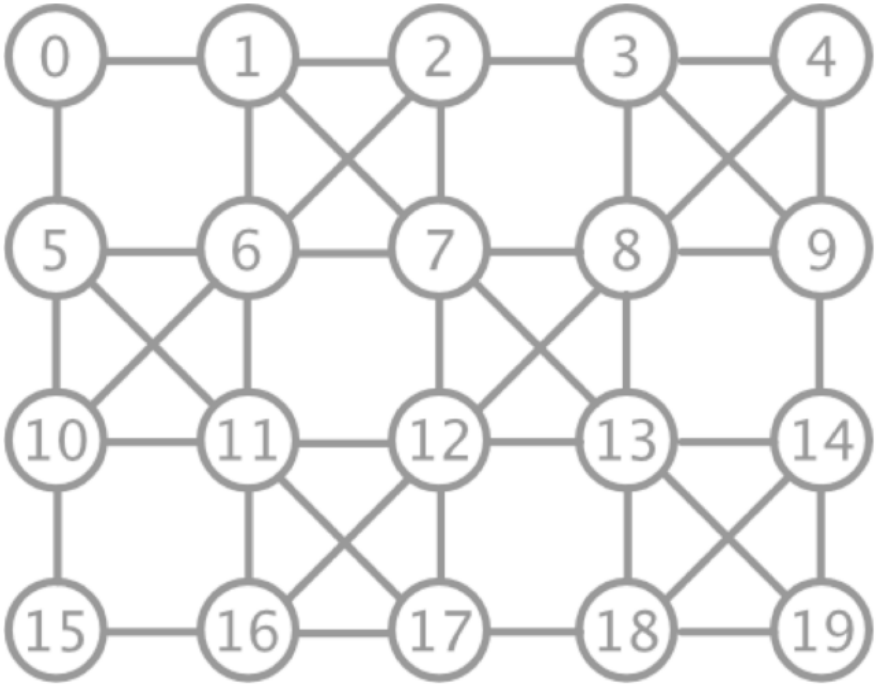








Tokio



Average measurements

Frequency (GHz)	4.97
T1 (μs)	89.59
T2 (μs)	58.66
Gate error (10^{-3})	1.84
Readout error (10^{-2})	8.38

Procesador Cuántico de 20 Qubits<

Disponbie desde finales de 2017 y orientado a uso empresarial

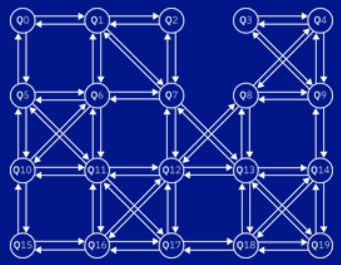
Mejores tiempos de coherencia. Se llega a una media de **90ms**.

Punto de partida para comenzar a trabajar con programas realmente útiles.

- 20 Qubits de propósito empresarial y programable con QASM y Python
- Qubits más optimizados e interconectados

Backend: QS1_1 (20 Qubits)

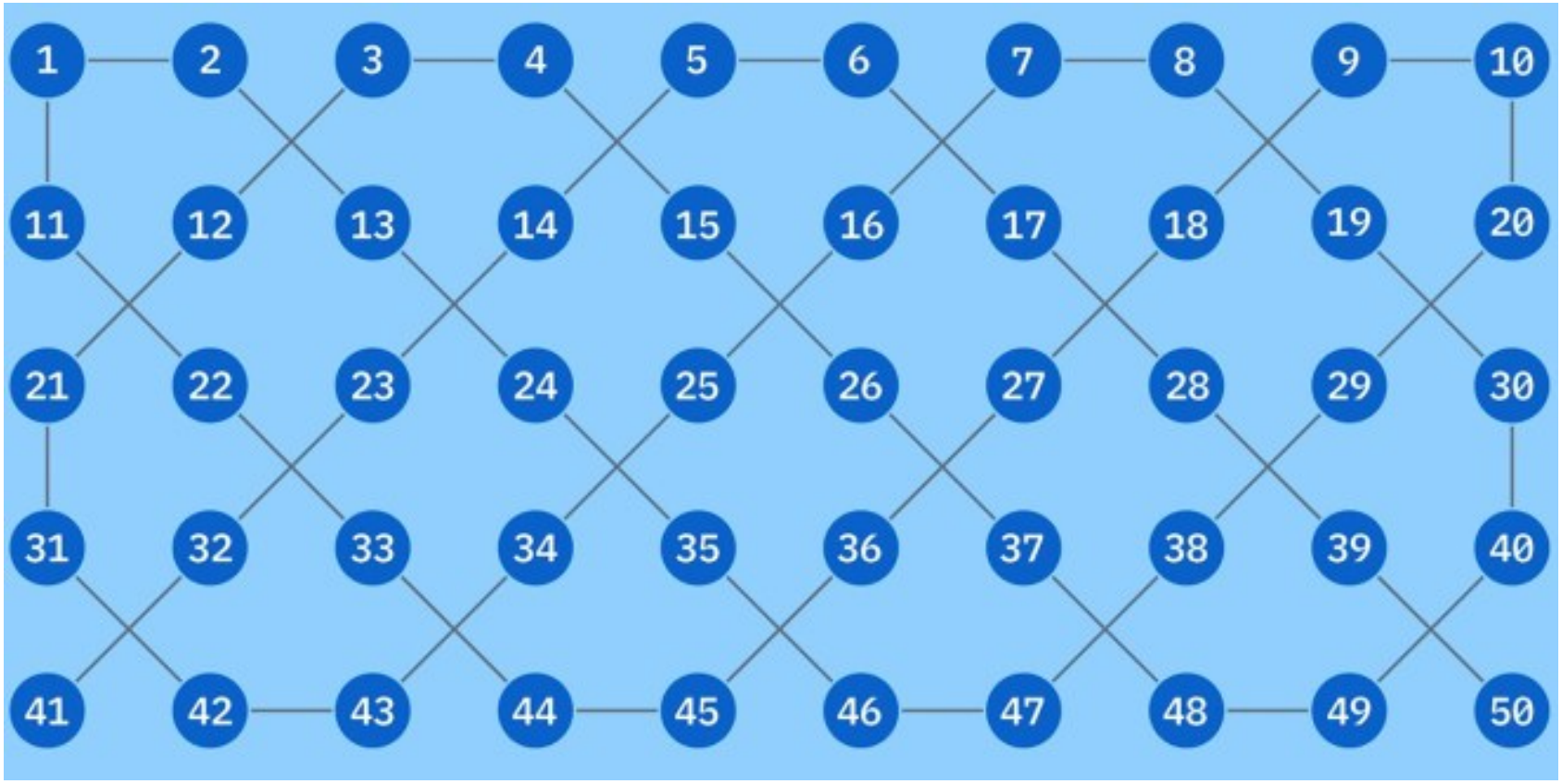
ACTIVE | AVAILABLE TO HUBS, PARTNERS, AND MEMBERS OF THE IBM Q NETWORK



Date Calibration: 2017-12-13 16:58:26

	Q0	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10	Q11	Q12	Q13	Q14	Q15	Q16	Q17	Q18	Q19
Frequency (GHz)	4.84	4.48	4.88	5.03	5.01	4.91	4.89	4.50	4.80	5.05	4.99	5.06	4.86	4.36	4.45	4.41	4.90	4.97	4.94	4.85
T1 (µs)	71.94	78.41	62.65	57.68	47.50	58.22	72.70	173.53	68.23	61.05	84.10	55.20	85.81	98.52	108.49	82.75	87.30	64.71	80.82	101.41
T2 (µs)	18.27	15.63	22.23	29.33	45.38	39.44	38.11	60.35	36.62	39.80	15.60	37.48	46.19	94.16	34.38	61.09	41.38	40.24	42.42	68.43
Gate error (10^{-3})	17.06	30.68	7.97	12.43	3.52	2.48	6.59	1.68	1.05	3.75	1.02	1.99	1.32	1.09	3.21	0.77	1.68	2.01	2.47	1.54
Readout error (10^{-2})	6.45	27.35	10.40	21.75	8.75	13.35	20.15	30.95	17.75	18.60	13.70	14.85	7.60	6.55	23.60	14.40	18.15	14.15	22.20	10.25
MultiQubit gate error (10^{-3})	CX0_1 4.04	CX1_0 4.04	CX2_1 5.82	CX3_4 3.45	CX4_3 3.45	CX5_0 13.93	CX6_1 3.47	CX7_1 7.32	CX8_4 3.79	CX9_3 6.28	CX10_5 4.69	CX11_6 5.77	CX12_7 1.93	CX13_7 5.27	CX14_13 5.68	CX15_10 1.61	CX16_15 1.59	CX17_12 1.93	CX18_13 4.75	CX19_13 6.20
	CX0_5 13.93	CX1_2 5.82	CX2_7 5.86	CX3_9 6.28	CX4_8 3.79	CX5_6 3.70	CX6_5 3.70	CX7_2 5.86	CX8_9 6.33	CX9_4 3.60	CX10_6 4.65	CX11_10 1.13	CX12_8 2.63	CX13_8 17.72	CX14_9 5.30	CX15_16 1.59	CX16_11 2.63	CX17_16 2.00	CX18_14 2.91	CX19_14 2.31
		CX1_6 3.47			CX4_9 3.60	CX5_10 4.69	CX6_11 5.77	CX7_6 5.87	CX8_12 2.63	CX9_8 6.33	CX10_11 1.13	CX11_5 1.55	CX12_11 2.41	CX13_12 3.89	CX14_18 2.91		CX16_12 2.03	CX17_11 1.46	CX18_17 1.70	CX19_18 1.43
		CX1_7 7.32				CX5_11 1.55	CX6_7 5.87	CX7_13 5.27	CX8_13 17.72	CX9_14 5.30	CX10_15 1.61	CX11_12 2.41	CX12_17 1.93	CX13_14 5.68	CX14_19 2.31		CX16_17 2.00	CX17_18 1.70	CX18_19 1.43	
							CX6_10 4.65	CX7_12 1.93				CX11_16 2.63	CX12_13 3.89	CX13_18 4.75						
												CX11_17 1.46	CX12_16 2.03	CX13_19 6.20						

50 Qubits


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
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50

rigetti

rigetti

ABDProf DEVELOPMENT

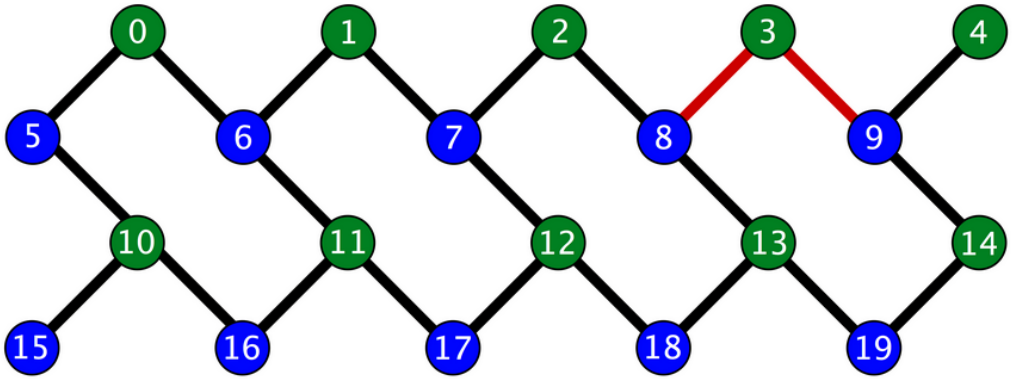
Procesador universal de 19 Qubits

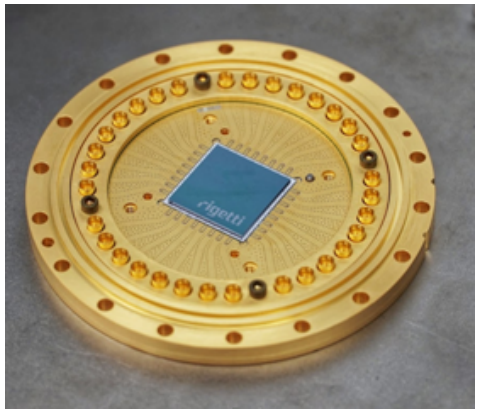
- “Acorn” 20 qubits de tipo transmon con diseño planar
- Rigetti está investigando configuraciones en 3D
- El Qubit 3 no está operativo

venture funding raised by Rigetti: 69.2 million.

Qubits Ajustables

Qubits Fijos





Entorno de Desarrollo

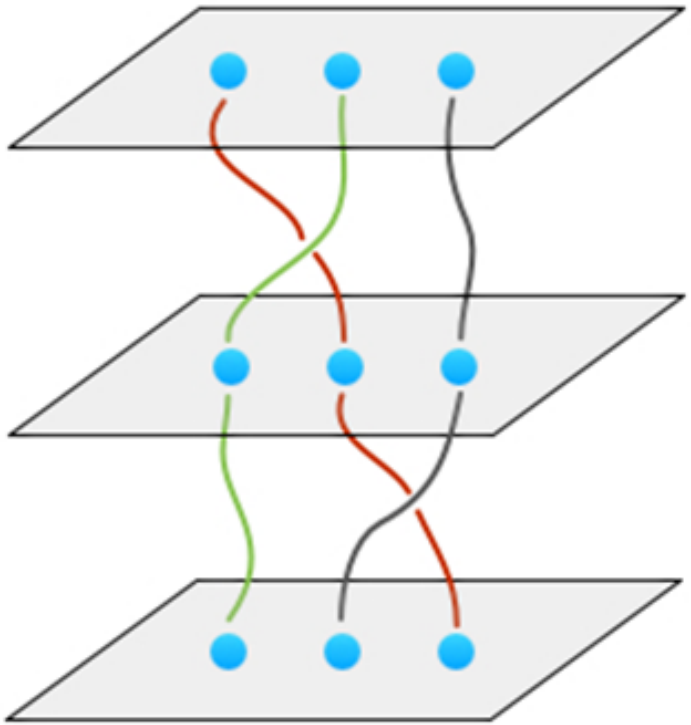
- Forest → Entorno de desarrollo en Python
- Acceso a simuladores
- Acceso al procesador de 19 qubits previa solicitud

Forest apuesta por un modelo híbrido cuántico-clásico, integrándose directamente con la infraestructura de Cloud y tratando el computador cuántico como un acelerador.

Microsoft

Visión de Microsoft

- A medio largo plazo, Microsoft apuesta por la computación topológica.
- Microsoft lleva investigando más de una década como construir un qubit topológico
- La Computación Topológica se basa en la existencia de una quasipartícula anunciada teoricamente en 1937 por Etore Majorana.
- Los qubit topológicos presentan grandes ventajas:
  - Inmunes a pequeñas pertubaciones
  - Facilitarían el escalado de los computadores cuánticos



Microsoft Quantum Development Kit - QDK

https://github.com/microsoft/Quantum

- Lanzado a principios de 2018, include:
- Incluye el lenguaje de programación llamado Q# y el compilador
  - Muestras y tutoriales de aprendizaje
  - Varios simuladores
  - Herramientas de estimación de recursos
- Desde julio de 2019 QKD es Open Source

README.md

Microsoft Quantum Development Kit Samples

launch binder

These samples demonstrate the use of the Quantum Development Kit for a variety of different quantum computing tasks. Most samples are provided as a Visual Studio 2017 C# or F# project under the `QsharpSamples.sln` solution.

Each sample is self-contained in a folder. Most of the samples consist of a Q# source file with detailed comments explaining the sample and a short classical program (either `Program.cs` (C#), `Program.fs` (F#), or `host.py` (Python)) that calls into the Q# operations and functions. There are some samples that are written as an interactive Jupyter notebook thus require no classical host.

LIQUi|>

- LIQUi|> es una arquitectura Software para computación cuántica que incluye lenguajes, algoritmos y simuladores.
- Desarrollado por el grupo de Arquitectura Cuantica de Microsoft se utiliza para traducir algoritmos escritos en lenguajes de alto nivel a programacion a nivel de máquina.
- Simula Hamiltonianos, Circuitos cuánticos y modelos de ruido cuánticos.
- El stack incluye un simulador de 30 qubits con una maquina de 32 GB de RAM.
- El número más grande que ha podido factorizar ha sido uno de 13 bits con 27 qubits, medio millon de puertas lógicas y 5 dias de ejecución.

Quantum Development Kit for Visual Studio Code

- Las aplicacioens desarrolladas con Microsoft Quantum Development Kit constan de dos partes
  - **Parte Cuántica**: Algoritmos Cuánticos implementados con Q#
  - **Parte Clásica**: Un programa clásico en Python o C#, que invoca las operaciones de Q#

clasico

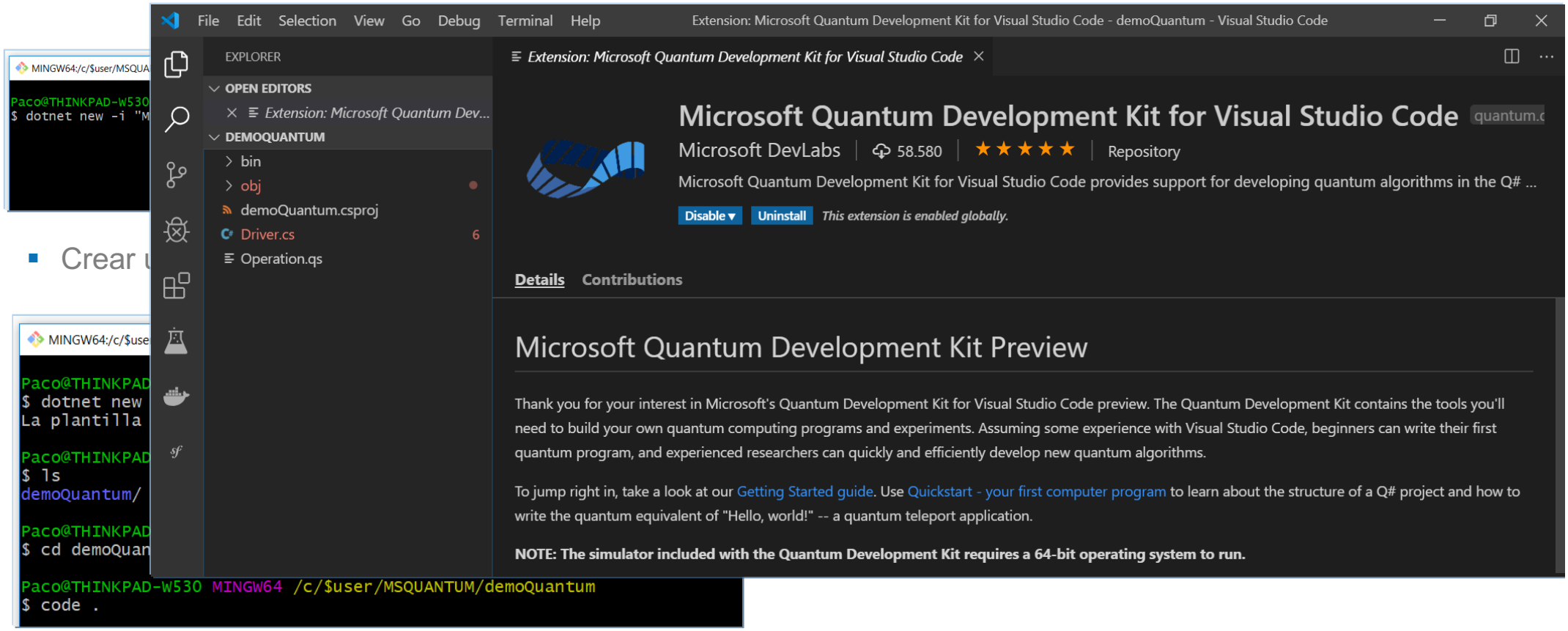
```
{
    using (var qsim = new QuantumSimulator())
    {
        // Try initial values
        Result[] initials = new Result[] { Result.Zero, Result.One };
        foreach (Result initial in initials)
        {
            var res = BellTest.Run(qsim, 1000, initial).Result;
            var (numZeros, numOnes) = res;
            System.Console.WriteLine(
                $"Init:{initial,-4} 0s={numZeros,-4} 1s={numOnes,-4}");
        }
    }

    System.Console.WriteLine("Press any key to continue...");
    Console.ReadKey();
}
}
}
```

Cuántico

```
operation BellTest (count : Int, initial: Result) : (Int, Int) {

    mutable numOnes = 0;
    using (qubit = Qubit()) {

        for (test in 1..count) {
            Set (initial, qubit);
            let res = M (qubit);

            // Count the number of ones we saw:
            if (res == One) {
                set numOnes += 1;
            }
        }
        Set(Zero, qubit);
    }

    // Return number of times we saw a |0> and number of times we saw a |1>
    return (count-numOnes, numOnes);
}
```

Quantum Development Kit for Visual Studio Code

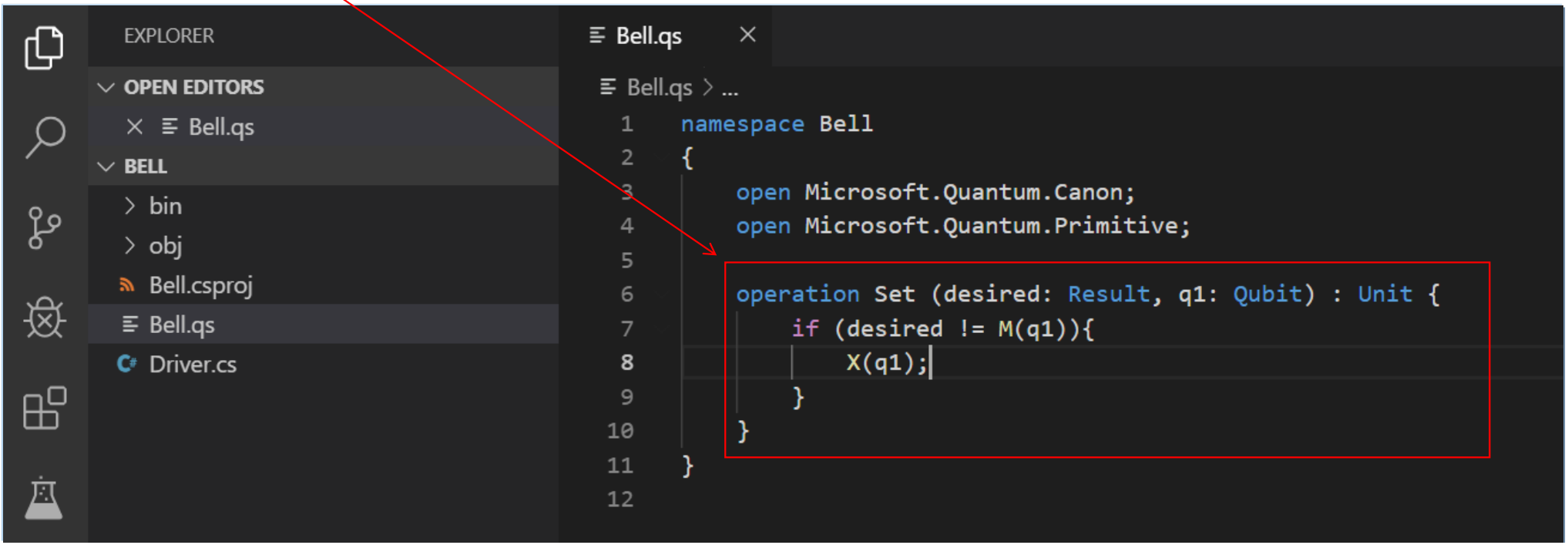
- Instalar las plantillas el proyecto Q#



- Crear u

Quantum Development Kit for Visual Studio Code

- La operación es la unidad básica de ejecución cuántica en Q#:



Quantum Development Kit - Operaciones Básicas

- Cambiando el estado del qubit

```
{
    X(qubit);
        let res = M (qubit);
```

- Creando Superposición

```
    H(qubit);
        let res = M (qubit);

```

- Generando Entrelazamiento

```
    using ((q0, q1) = (Qubit(), Qubit())) {

        Set (initial, q0);
        Set (Zero, q1);

        H(q0);
        CNOT(q0,q1);
        let res = M (q0);
}
```

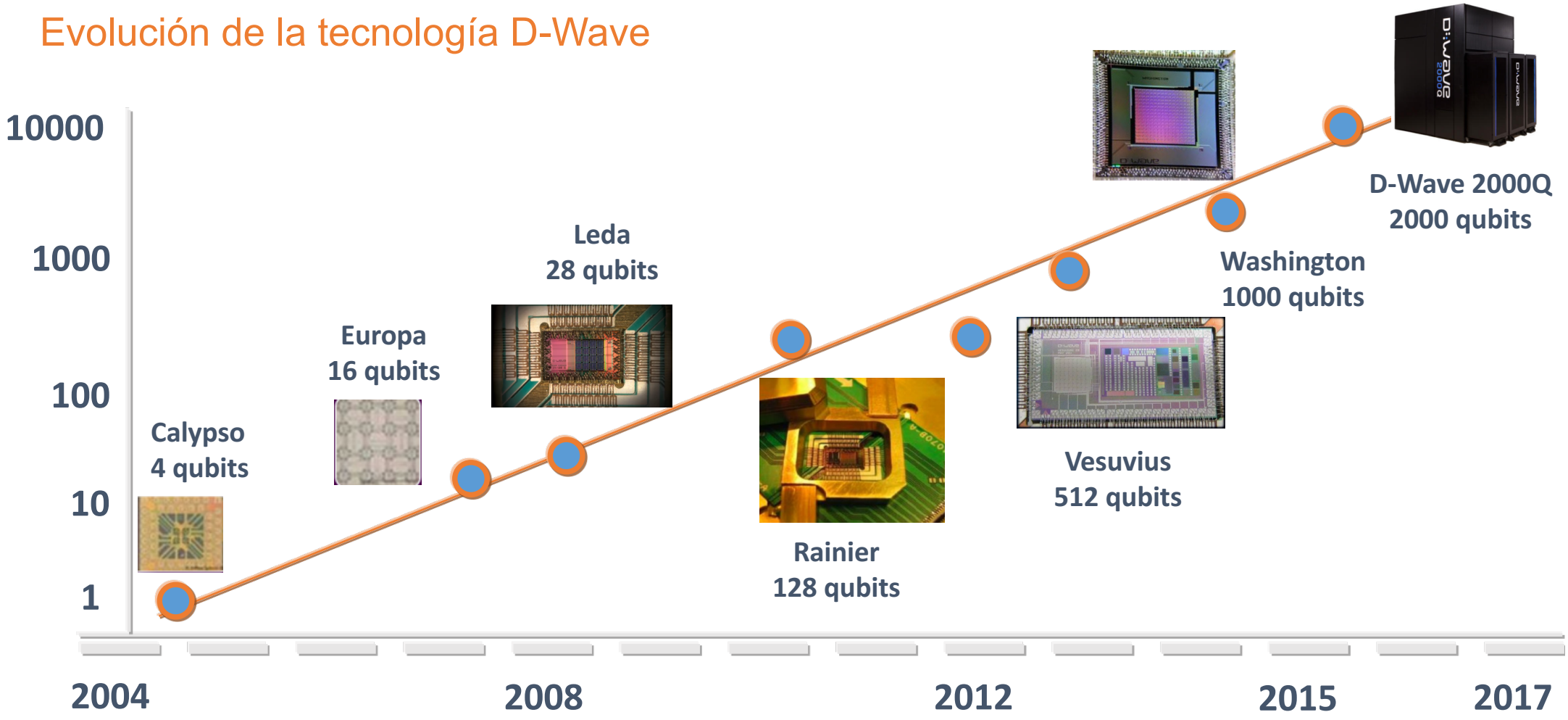
D-Wave

The Quantum Computing Company™

D-Wave Systems

- D-Wave 2000Q → 2048 Qubits
- Qubits Superconductores
- Computación Cuántica Adiabática
- No es Universal
- Resolución de problemas de Optimización.
- No ha demostrado Quantum Supremacy

Evolución de la tecnología D-Wave



D-Wave
The Quantum Computing Company™

ABDProf DEVELOPMENT

Aplicaciones y Oportunidades de Negocio

- Primera Empresa en construir un computador cuántico
- Clientes declarados
  - Lockheed Martin/USC
  - Google/NASA Ames/USRA
  - Los Alamos National Lab
  - Cybersecurity-1
  - Oak Ridge National Lab
- Otros Clientes en ciernes.

Security & Mission Planning

Detecting computer viruses & network intrusion

Scheduling resources and optimal paths

Determining set membership

Analyzing graph properties

Factoring integers

Machine Learning & Computer Science

- Detecting statistical anomalies
- Finding compressed models
- Recognizing images and patterns
- Training neural networks
- Verifying and validating software
- Classifying unstructured data
- Diagnosing circuit faults

Financial Modeling

- Detecting market instabilities
- Developing trading strategies
- Optimizing trading trajectories
- Optimizing asset pricing and hedging
- Optimizing portfolios

Healthcare & Medicine

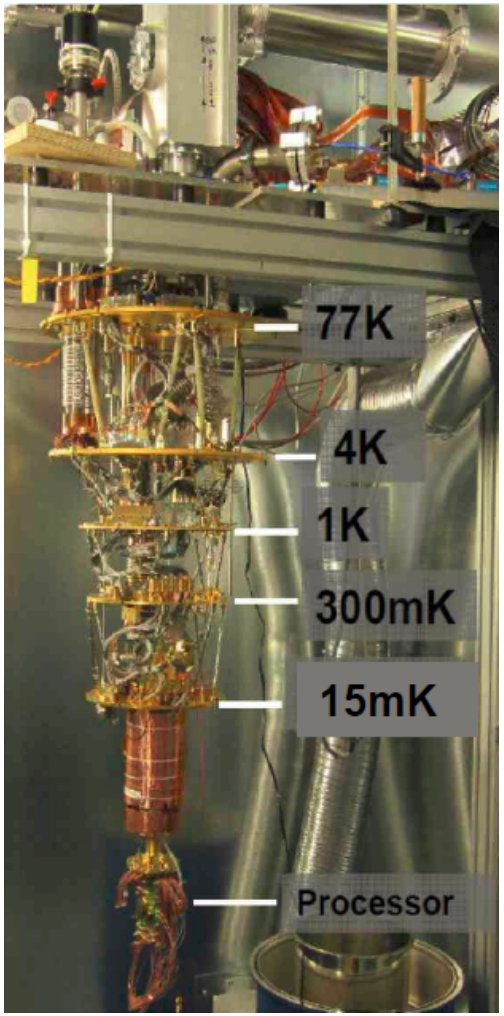
- Detecting fraud
- Generating targeted cancer drug therapies
- Optimizing radiotherapy treatments
- Creating protein models

Aislamiento

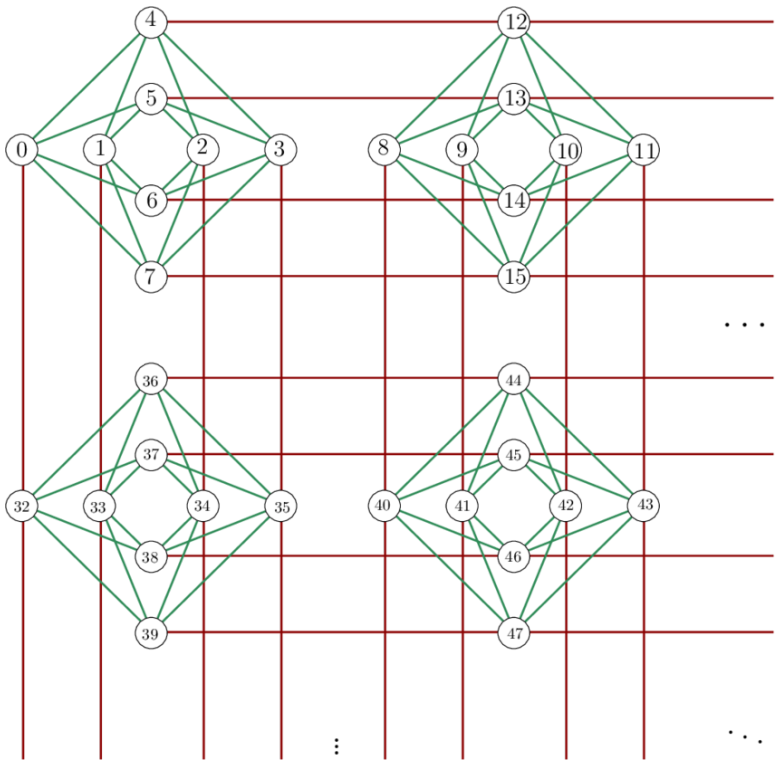
- Caja de Faraday Totalmente aislada de campos externos.
- 16 Capas de aislamiento entre el chip cuántico y el mundo exterior
- Campo magnético interno es < nT. Aprox. 50.000 veces inferior al campo magnético terrestre
- Alto vacío: 10.000 millones de veces inferior a la presión atmosférica.

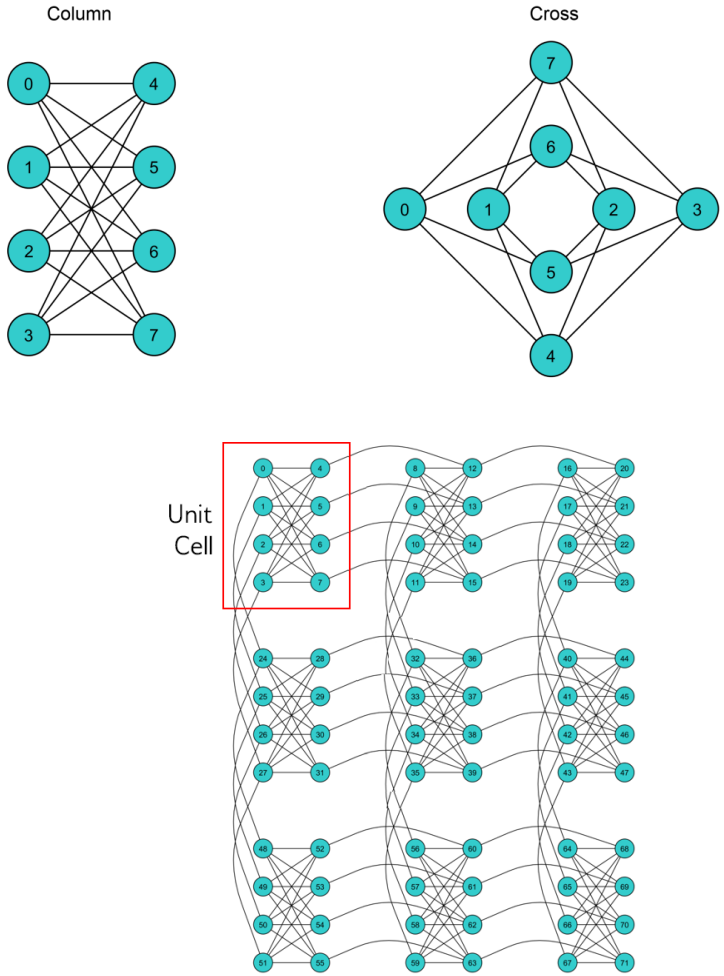
Sistema de Enfriamiento

- Temperaturas cercanas al cero absoluto (15 mK)
- Refrigeración de Dilución. Mezcla de ^{3}He y ^{4}HE
- Consumo de Energía 25 kW
  - Consumo constante con el escalado.



Arquitectura Chimera

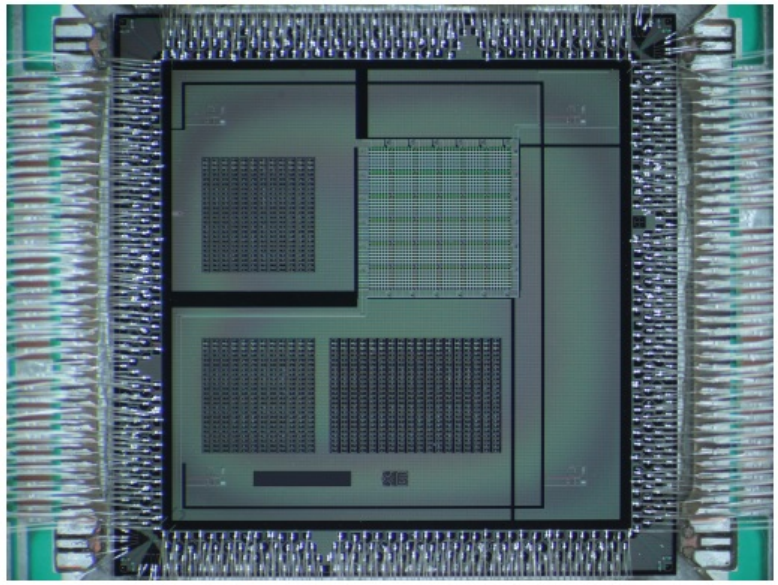




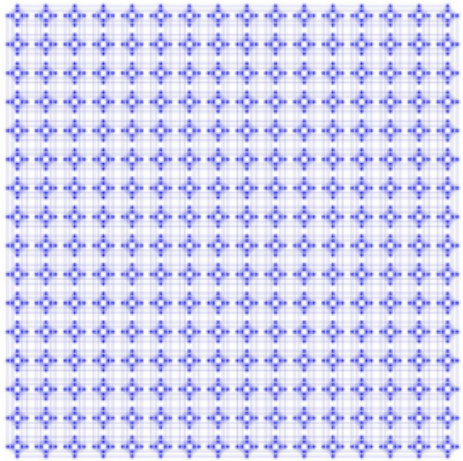
Nueva Arquitectura P6 680 (pegasus)

- Visualización de topologias: de Quimera a Pegasus → https://www.dwavesys.com/sites/default/files/D-Wave_Pegasus_GIF.gif.
- D-Wave ha anunciado un sistema de 5640 Qubits

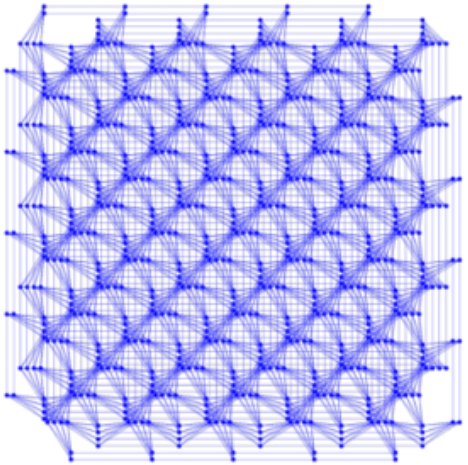
Pegasus P6 - 680 Qubit Prototype



Chimera C16 - DW 2000Q



Pegasus P6 - 680 Qubit Prototype

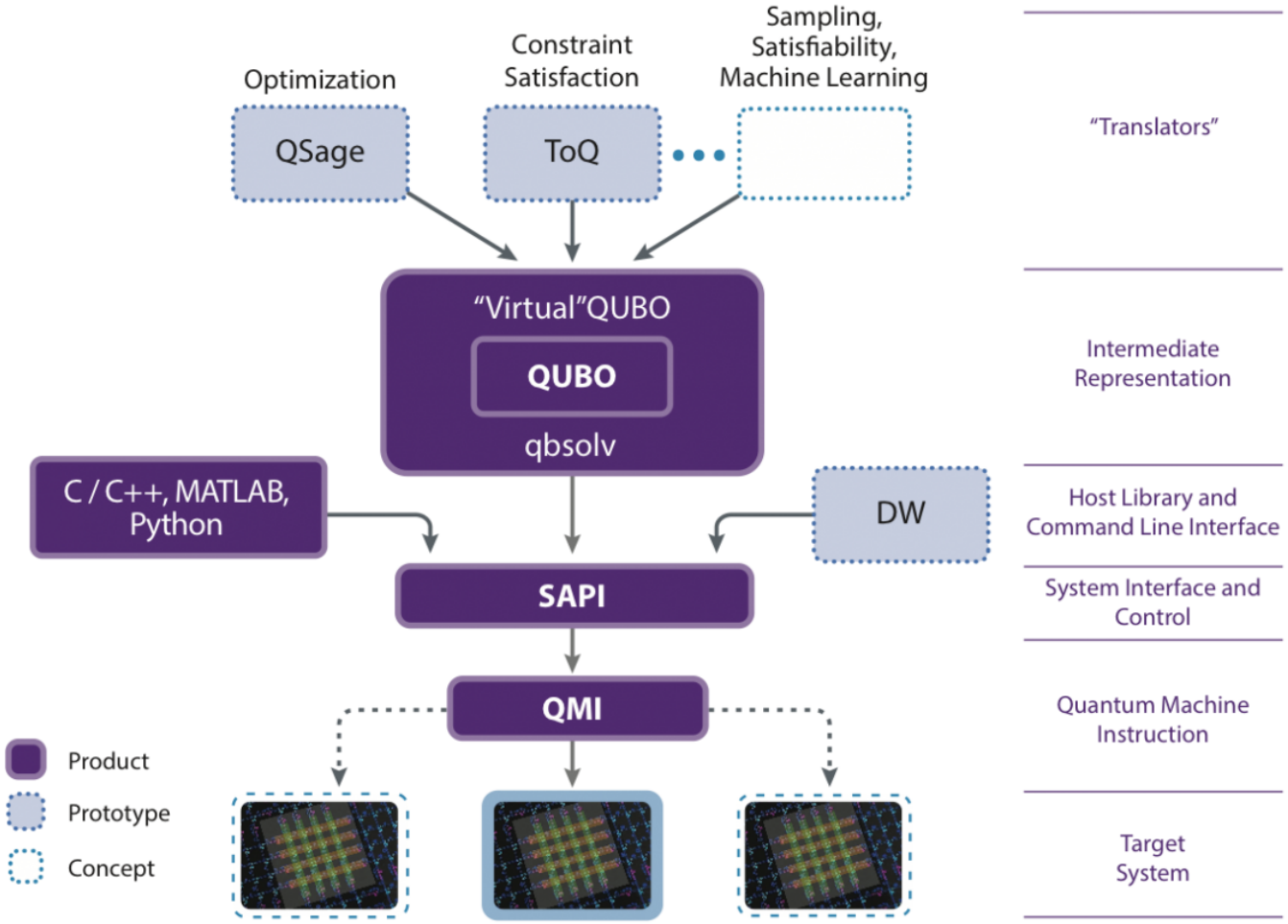


Device Count	C16	P6	P16
Qubits	2048	680	5640
Couplers	6000	4784	40484
max degree	6	15	15

Nueva Arquitectura P6 680 (pegasus)

- Mas Conectividad entre qubits (de 5 a 15)
- Menor ruido. (Qubits más robustos y mejora aislados)
- Más Qubit por procesador (de 2000 a 5640)
- Nuevas herramientas de desarrollo (Ocean) y opciones de arquitectura (Hybrid).
- Acceso más flexible (Leap)
- Ongoing Releases:

D-Software Environment

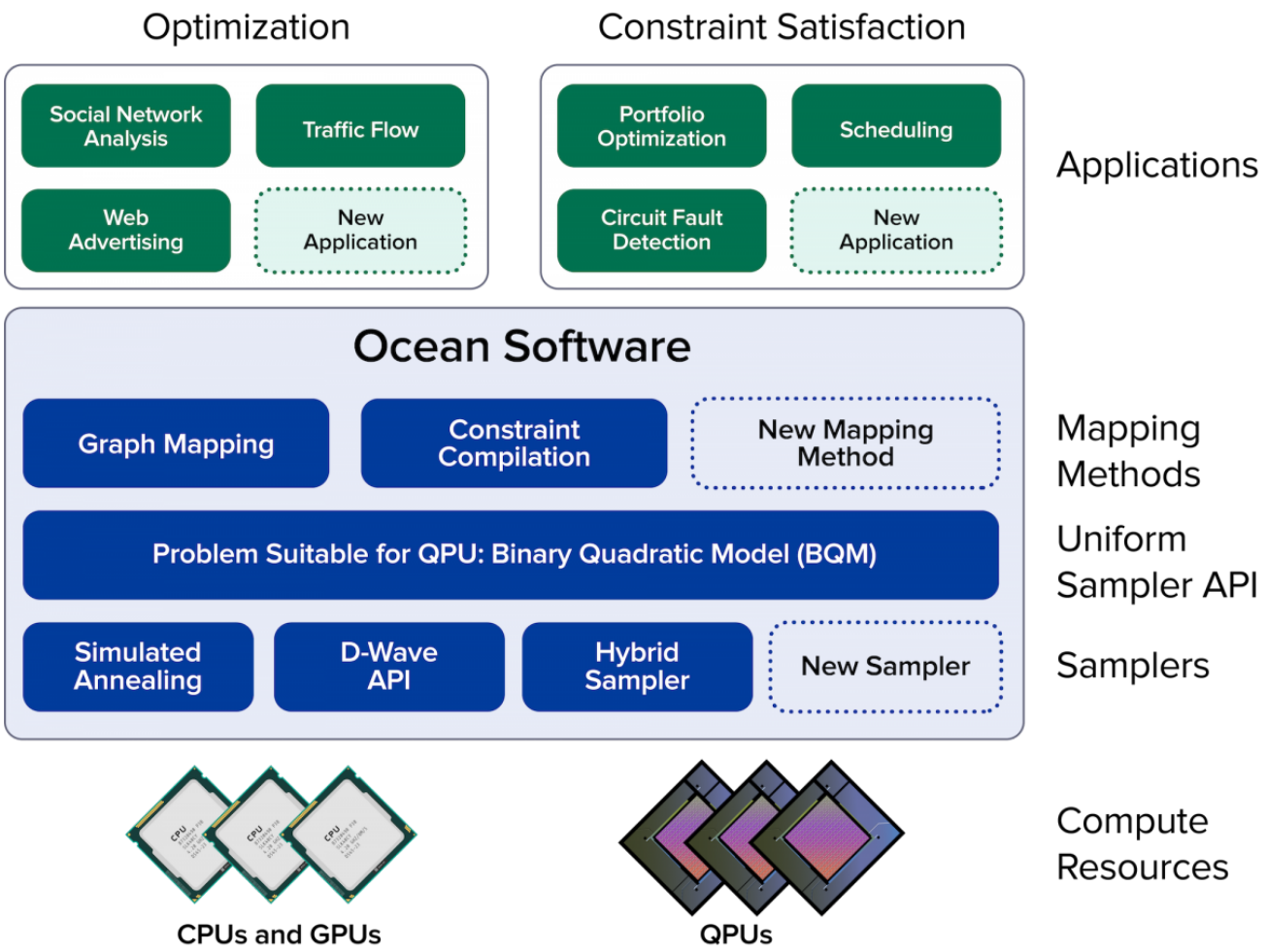


- API Restful para C/C++, Python y Matlab

Quadratic Unconstrained Binary Optimization (QUBO) problem

qbsolv → lee las instancias QUBO
Particiona el problema
Resuelve subproblemas
Recombina resultados

D-Software Environment



D:Wave
The Quantum Computing Company™

ABDProf DEVELOPMENT

cloud.dwavesys.com/leap/login/

D:Wave Leap

Leap In

EMAIL ADDRESS

jane@gmail.com

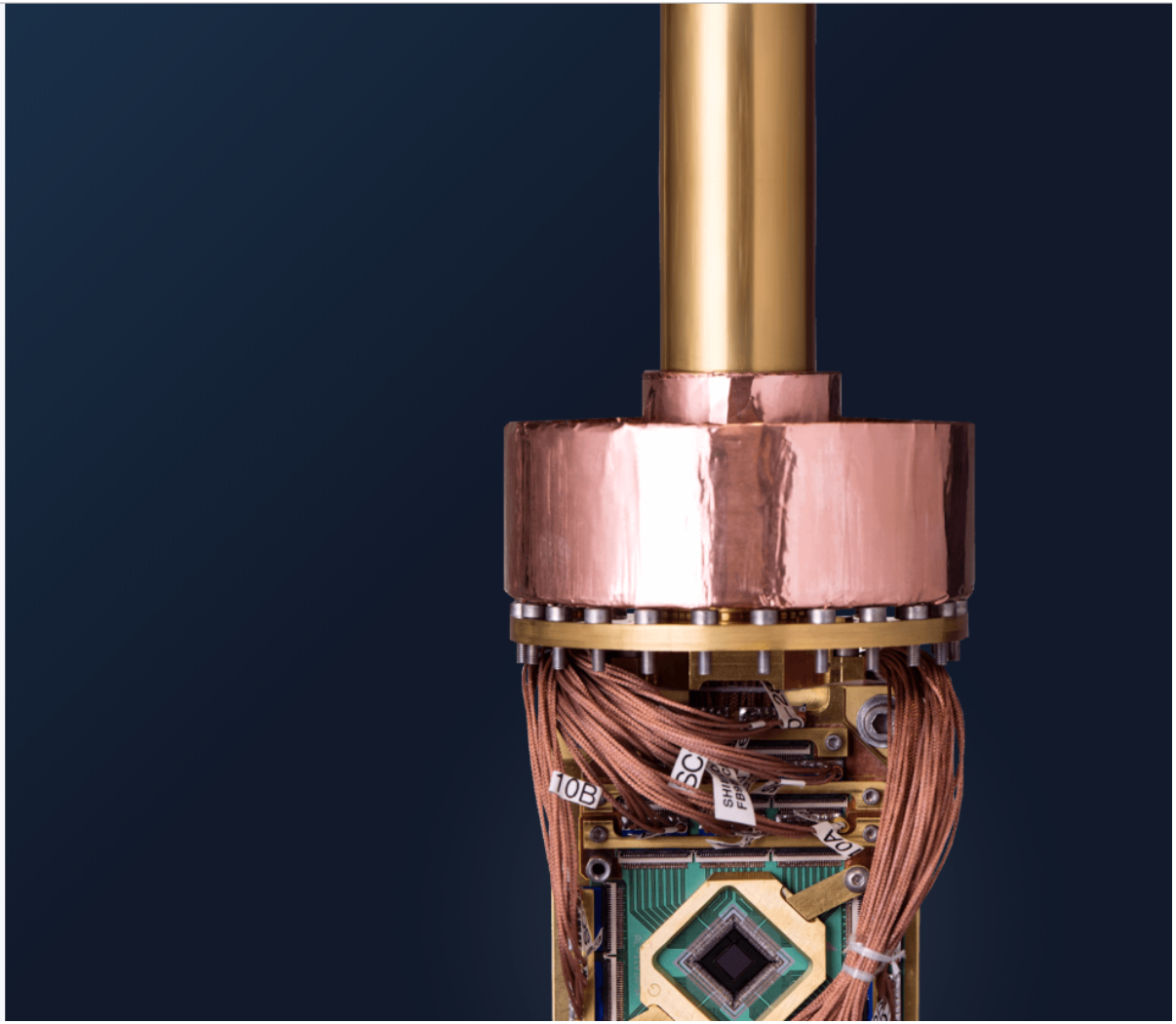
PASSWORD

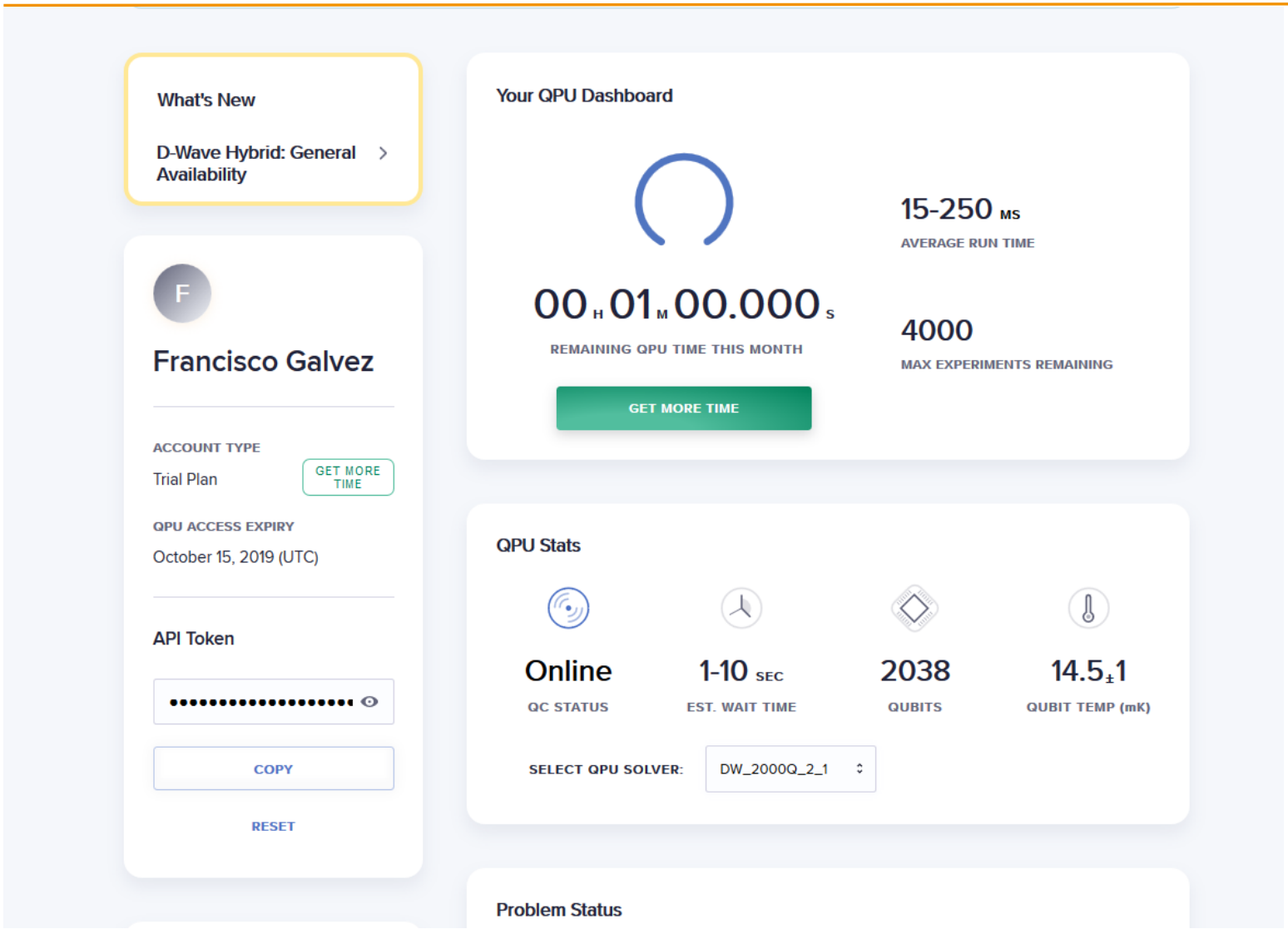
Forgot password?
Having trouble logging in?

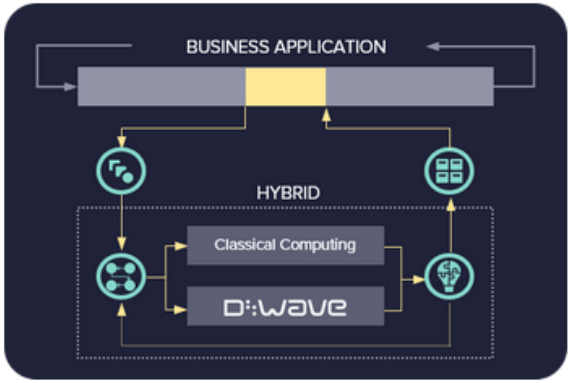
LOG IN

Don't have an account? Sign up

Lost activation link? Resend link







- Leap – Entorno Cloud de progamación
- Ocean SDK – Kit de desarrollo de Softwre

D-Wave Leap Dashboard Resources Community Help Francisco Galvez

Dashboard

BACK DISMISS

Install our SDK

STEP 1

D-Wave Python Library

We recommend that you work in a virtual environment. If you are new to Python virtual environments, see our Getting Started guidelines. To set up the required dependencies, run the following command from your terminal:

```
pip install dwave-ocean-sdk &&
dwave config create
```

COPY

STEP 2

Your API token

Follow the prompts and paste the following into your terminal.

••••••••••••••••••••••••••••

COPY

STEP 3

Solve Problems

You should now be able to connect to a D-Wave system. Verify with the **dwave ping** command. Solve your own problems or begin with our end-to-end examples.

D-Wave
The Quantum Computing Company™

ABDProf DEVELOPMENT

Factoring with the D-Wave System

In the factoring demo you saw how the D-Wave system factored an integer by running a multiplication circuit in reverse.

This notebook demonstrates how you can solve a constraint satisfaction problem (CSP) on a quantum computer with the example of factoring.

1. Factoring as a Constraint Satisfaction Problem describes the factoring problem as an example CSP.
2. Formulating the Problem for a D-Wave System shows how such CSPs can be formulated for solution on a quantum computer.
3. A Simple Example codes a small CSP to clarify the solution technique.
4. Factoring on the Quantum Computer codes a factoring problem for solution on the D-Wave system.
5. Further Information details some points touched on in previous sections and examines more closely the results returned from the quantum computer.

This notebook should help you understand both the techniques and Ocean software tools used for solving CSPs on D-Wave quantum computers.

New to Jupyter Notebooks? JNs are divided into text or code cells. Pressing the **Run** button in the menu bar moves to the next cell. Code cells are marked by an "In: []" to the left; when run, an asterisk displays until code completion: "In: [*]".

Factoring as a Constraint Satisfaction Problem

The complexity class for classical integer factoring is believed to be between P and NP-hard. Although research has yielded algorithms that perform faster than the intuitive trial division, including Fermat's algorithm, Pollard's two algorithms, and sieve algorithms, it's still an open question whether a classical algorithm exists that can factor in polynomial time. For quantum computing, Shor's algorithm runs in polynomial time (the D-Wave system does not run this algorithm).

This notebook solves factoring on a D-Wave quantum computer by formulating it as a *constraint satisfaction problem*. CSPs require that all a problem's variables be assigned values that result in the satisfying of all constraints. For factoring, the problem's constraints are that the two variables representing factors, a and b, be assigned only natural numbers and that their multiplication be equal to the factored number, P.

Among CSPs are hard problems well suited to solution on quantum computers. For example, the map-coloring problem is to color all regions of a map such that any two regions sharing a border have different colors (see a D-Wave system solve a four-color map-coloring problem here: Ocean software examples). The job-shop scheduling problem is to schedule multiple jobs done on several machines with constraints on the machines' execution of tasks. You can apply the solution technique shown here to many CSPs.

Formulating the Problem for a D-Wave System

How can we formulate the factoring problem in a way that a D-Wave quantum computer can understand?

D-Wave systems solve binary quadratic models (BQM), the Ising model traditionally used in statistical mechanics and its computer-science equivalent, the quadratic unconstrained binary optimization (QUBO) problem. Given N variables $x_1, \ldots, x_N$, where each variable x_i can have binary values 0 or 1, the system finds assignments of values that minimize

- Notebooks
  - Factoring
  - Structural Imbalance
  - Anneal Scheduling
  - Others…

CAMPOS DE APLICACIÓN

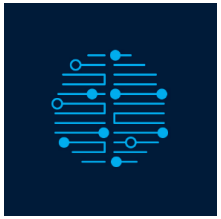
CAMPOS DE APLICACIÓN

Cryptografia y Seguridad

- Mantener claves privadas a salvo de hackers e intrusiones, no importa donde se almacenen o se procesen.

Medicina y Materiales

- Simular y entender las moléculas y sus interacciones mejor que un computador clásico.



Machine Learning

- Acelerar significativamente el aprendizaje automático y las tareas analíticas tales como el análisis topológico.

Searching Big Data

- Busquedas, conexiones y relaciones de forma mucho más rápida que los ordenadores actuales.

CAMPOS DE APLICACIÓN

Criptografía y Seguridad

- **Distribución de Claves Cuánticas**

  Uso de propiedades cuánticas para intercambio de claves criptográficas, que se utilizan para cifrar mensajes a través de un canal inseguro.

IDQ FROM VISION TO TECHNOLOGY MagiQ Quintessence labs

- **La internet Cuántica**

  Comunicación cuántica de largo alcance, conexión a los procesadores cuánticos y mayores capacidades de transmision más segura

- **Criptografía Post-Cuantica**

  Necesidad de algoritmos criptográficos cuánticos resistentes a los nuevos estándares criptográficos de clave pública

Eighth International Conference on Post-Quantum Cryptography

PQCrypto 2017

Utrecht, The Netherlands, June 26–28, 2017

https://2017.pqcrypto.org

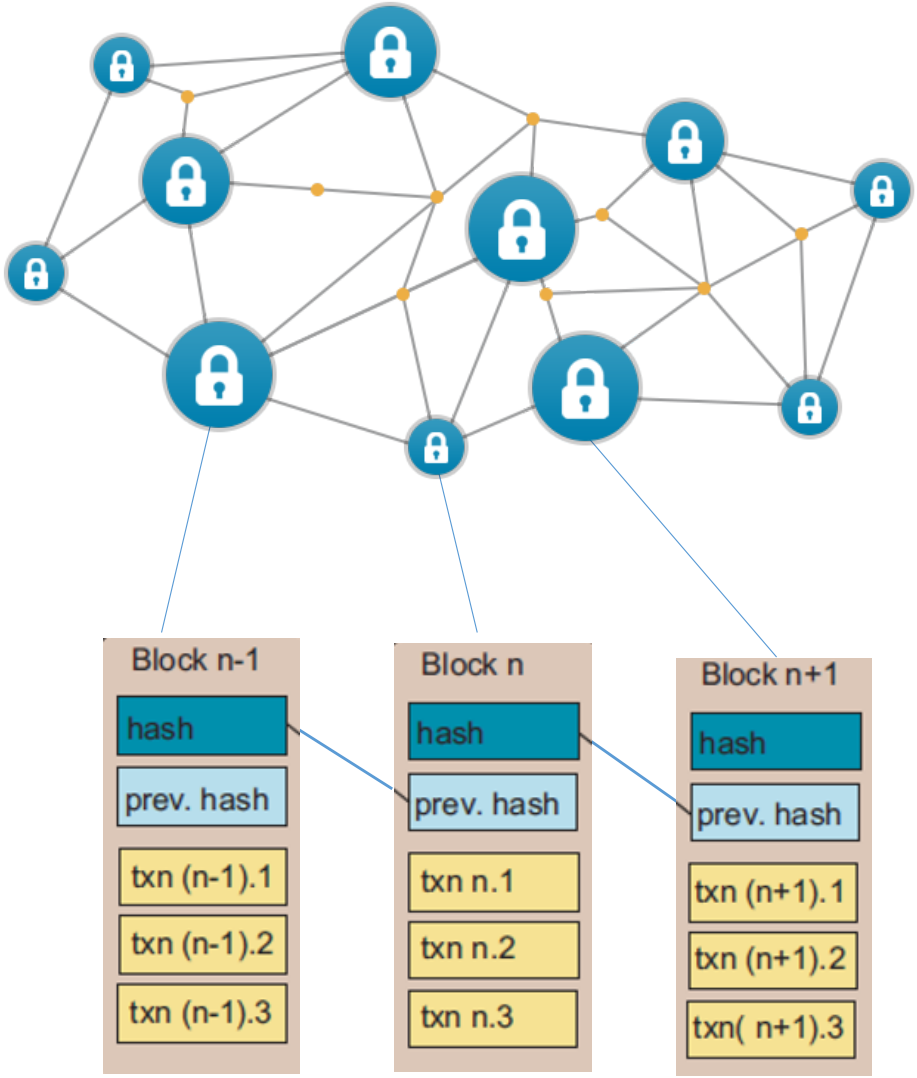
November 30, 2016

Google's Post-Quantum Cryptography Experiment Successful

Quantum Computers Create A Need For New Cryptography Methods.

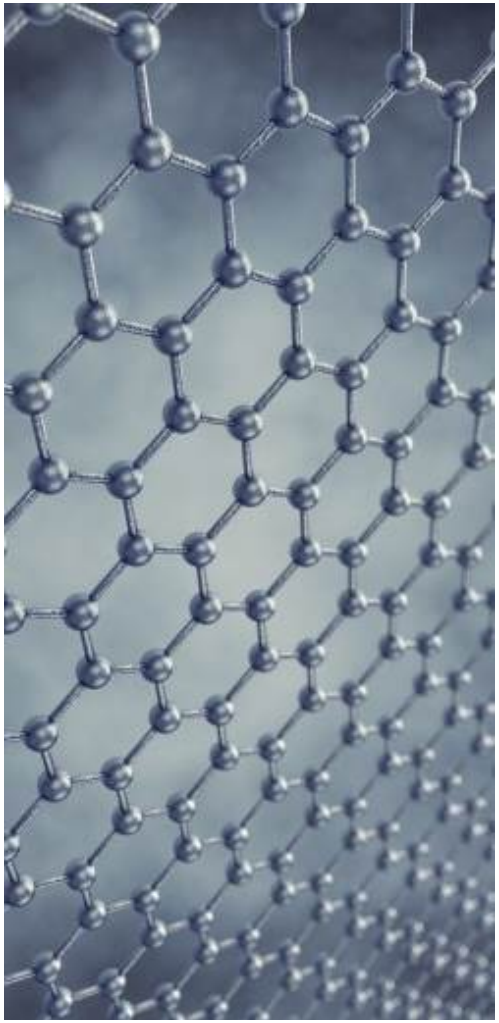
Ejemplo Seguridad en Blockchain

- Utiliza Hash key.
- Cada bloque contiene un valor hash relativo a su contenido, y además incluye un hash del bloque previo
- La seguridad de Blockchain puede mejorarse utiliando esquemas de firma digital post-quantum para firmar las transacciones
- El uso de Claves Cuánticas Distribuidas (QKD) garantiza un mayor nivel de seguridad mediante las leyes de la física cuántica

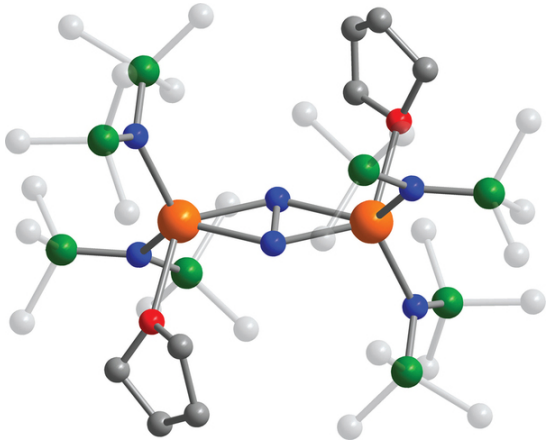


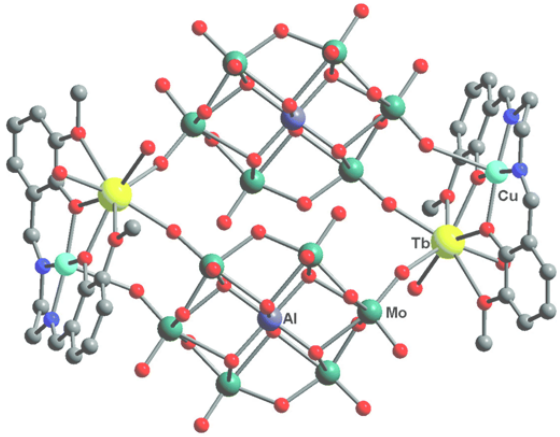
CAMPOS DE APLICACIÓN

Aplicaciones: Nuevos Materiales



- Simulación de modelos electrónicos reales con varios cuerpos
- Simulación de nuevas estructuras y experimentación de las interacciones entre los elementos que las componen.
- Actualmente se emplean algoritmos basados en modelos de Hartree-Fock





Machine Learning y Deep Learning

npj | Quantum Information

www.nature.com/npjqi

ARTICLE OPEN

Demonstration of quantum advantage in machine learning

Diego Ristè[1], Marcus P. da Silva[1], Colm A. Ryan[1], Andrew W. Cross[2], Antonio D. Córcoles[2], John A. Smolin[2], Jay M. Gambetta[2], Jerry M. Chow[2] and Blake R. Johnson[1]

The main promise of quantum computing is to efficiently solve certain problems that are prohibitively expensive for a classical computer. Most problems with a proven quantum advantage involve the repeated use of a black box, or oracle, whose structure encodes the solution. One measure of the algorithmic performance is the query complexity, i.e., the scaling of the number of oracle calls needed to find the solution with a given probability. Few-qubit demonstrations of quantum algorithms, such as Deutsch–Jozsa and Grover, have been implemented across diverse physical systems such as nuclear magnetic resonance, trapped ions, optical systems, and superconducting circuits. However, at the small scale, these problems can already be solved classically with a few oracle queries, limiting the obtained advantage. Here we solve an oracle-based problem, known as learning parity with noise, on a five-qubit superconducting processor. Executing classical and quantum algorithms using the same oracle, we observe a large gap in query count in favor of quantum processing. We find that this gap grows by orders of magnitude as a function of the error rates and the problem size. This result demonstrates that, while complex fault-tolerant architectures will be required for universal quantum computing, a significant quantum advantage already emerges in existing noisy systems.

npj Quantum Information (2017)3:16 ; doi:10.1038/s41534-017-0017-3

Gracias

Francisco Javier Gálvez Ramirez
fjgramirez@abdprof.com