CESGA

Infraestructuras Científicas y Técnicas Singulares


# Quantum Computing

**Dr. Andrés Gómez**
**Andres.gomez.tato@cesga.es**
**Apr. 2019**

# Lecture 3: Basic algorithms

- Quantum Paralelism
- Quantum Fourier Transform
- Amplitude amplification
- Phase estimation

# Quantum Paralelism

- The idea behind quantum paralelism is that you can apply a function to all the states in a superposition in **just one step**

- Let $\mathbf{U_f}$ an operator that implements the function ***f*** such that:

$$U_f\, |x, y> = |x, f(x) \oplus y>$$

- If we choose y=|0>:

$$U_f\, |x, 0> = |x, f(x)>$$

- And because **Uf is linear**, we can apply to any superpositon. For example, to the result states of Walsh-Hadamard operator:

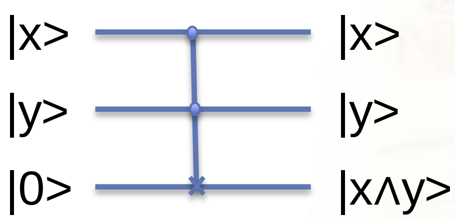
$$W = H^{\otimes n} \Rightarrow$$

$$U_f(W|0>\otimes |0>) = U_f(\frac{1}{\sqrt{N}}\sum_{i=0}^{N-1} |i>\otimes |0>) = \frac{1}{\sqrt{N}}\sum_{i=0}^{N-1} |i> |f(i)>$$

- BUT. All the solutions are entangled and we can get only one for each measurement

# Quantum Paralelism. Example

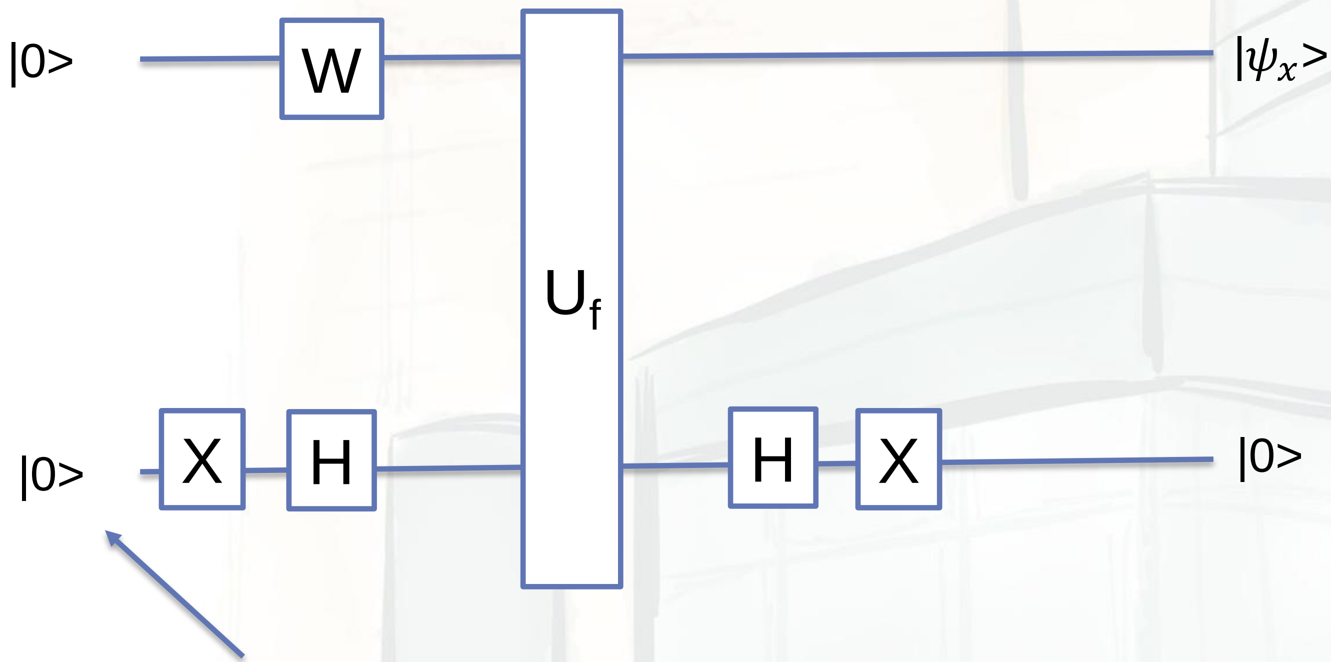
- Toffoli gate implement the classical AND operation on 2 bits




| $\|x\rangle$ | $\|y\rangle$ | $\|x \wedge y\rangle$ |
|---|---|---|
| 0 | 0 | 0 |
| 1 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 1 | 1 |

- So, $Toffoli(W|00> \otimes |0>) = \frac{1}{2}(|000> + |010> + |100> + |111>)$

# Quantum Subroutines




Ancilla or temporary qubit.

# Exercise: Quantum Parallel Programming

OPEN QISKIT/DEUTSCH-JOZSA_ALGORITHM NOTEBOOK

# Quantum Parallelism

- When on a state $|\psi> = \sum_{i=0}^{N-1} a_i|i>$ , a superposition, one unitary gate is applied to only a one single qubit, all the amplitudes of state $|\psi>$ can be affected

- For example:

$$(I\otimes H)\begin{vmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \end{vmatrix} = \begin{vmatrix} 1 & 1 & 0 & 0 \\ 1 & -1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & -1 \end{vmatrix}\begin{vmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \end{vmatrix} = \begin{vmatrix} a_1 + a_2 \\ a_1 - a_2 \\ a_3 + a_4 \\ a_3 - a_4 \end{vmatrix}$$

galicia XUNTA DE GALICIA GOBIERNO DE ESPAÑA MINISTERIO DE ECONOMÍA Y COMPETITIVIDAD CSIC CONSEJO SUPERIOR DE INVESTIGACIONES CIENTÍFICAS Unión Europea Fondo Europeo de Desarrollo Regional "Una manera de hacer Europa"

# Exercise: Applying Quantum Parallelism

OPEN QISKIT/FIND_EDGE NOTEBOOK

# Quantum Fourier Transform

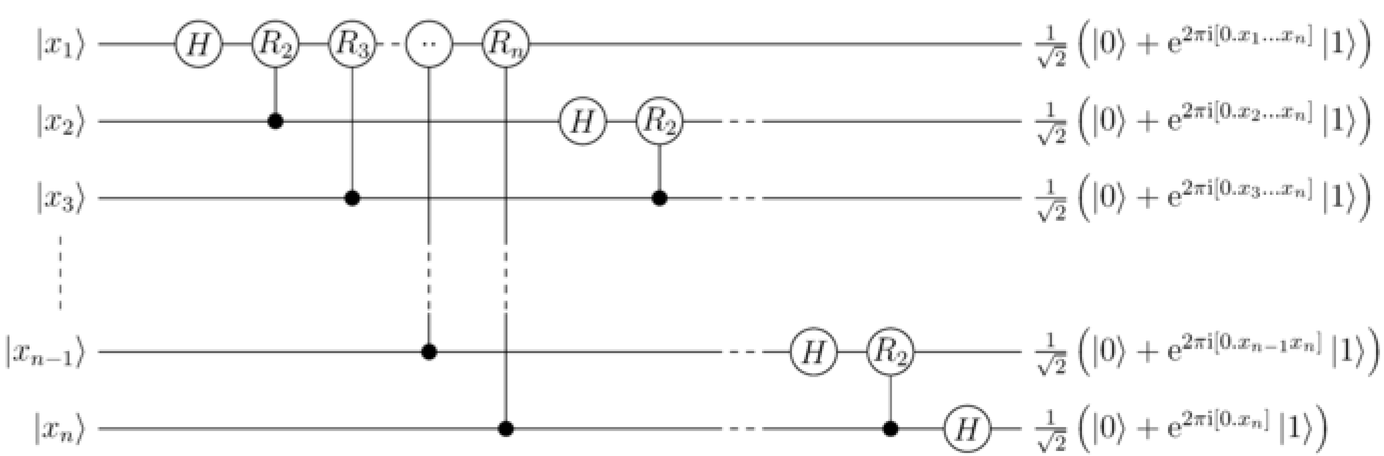
- Discrete Fourier Transform of a:[0,….,N-1]:$A(x) = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} a(k) e^{2\pi i \frac{kx}{N}}$
- Classical Fast Fourier Transform assumes that N = $2^n$
- Quantum Fourier Transform (QFT):
  - Amplitudes a(x) of state |x> is a function of x
  - So QFT($\sum_x a(x)|x>) = \sum_x A(x)|x>$
  - **If a(x) is a function of period r (r power of 2), A(x) are zero except for states multiple of N/r**

# Quantum Fourier Transform

- QFT(|k>) = $\frac{1}{\sqrt{N}}\sum_{x=0}^{N-1} e^{\frac{i2\pi kx}{N}}|x>$

- The way of calculate QFT is recursive as in classical FFT.

- If $R_m = \begin{bmatrix} 1 & 0 \\ 0 & e^{\frac{2\pi i}{2^m}} \end{bmatrix}$



# Exercise: Programming QFT

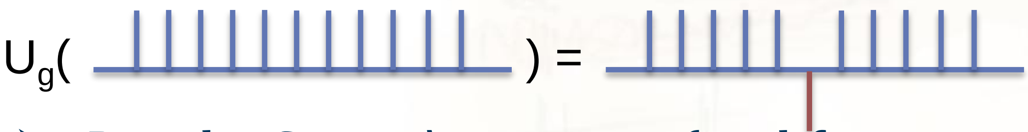
OPEN QISKIT/QFT NOTEBOOK

# Amplitude Amplification

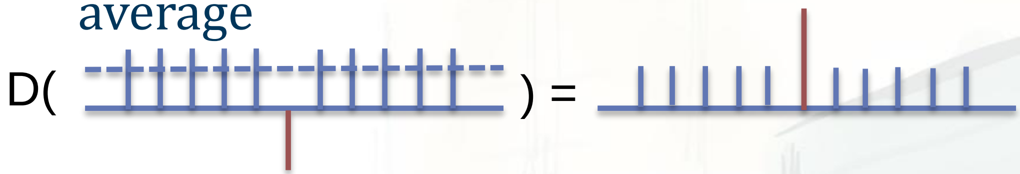
- Quantum computing is probabilistic.
- Want to maximize the opportunity of measuring the right answer in one shot.
- This means increase probability of the solutions. If possible, to 1.
- Solution: Amplitude amplification.
- If $U_g$ is the transformation on n qubits ($N=2^n$ states) that solve the problem, apply *k* times the transformation $DU_g$
  - What is $U_g$ ?
  - What is D?
  - How large is k?

# Amplitude Amplification

- $U_g$ is a transformation that change the sign to the solutions when appplied to the supperposition of all states




- D is the Grover's operator (or difusor operator) that **inverts** about the average




- If |G| is the probability of |x> if being a Good result, the number of time to apply the algorithm is:

$$k \approx \frac{\pi}{4}\sqrt{\frac{N}{|G|}}$$

# Grover's operator

$$D = \begin{bmatrix} \frac{2}{N} - 1 & \frac{2}{N} & \cdots & \frac{2}{N} \\ \frac{2}{N} & \frac{2}{N} - 1 & & \frac{2}{N} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{2}{N} & \frac{2}{N} & \cdots & \frac{2}{N} - 1 \end{bmatrix} = -\mathrm{W} S_0^{\pi} W$$

W is the **Walsh-Hadamard** transform:

$$H^{\otimes n} = H \otimes H \otimes \ \ldots \otimes H \otimes H$$

$S_0^{\pi}$ is the phase shit by π of the baseis vector |0>:

$$S_0^{\pi} = \begin{bmatrix} -1 & 0 & \cdots & 0 \\ 0 & 1 & & 0 \\ \vdots & & \ddots & \vdots \\ 0 & 0 & \cdots & 1 \end{bmatrix} = X^{\otimes n} \ C_n^{[0,n-1]}(Z) \, X^{\otimes n}$$

# Exercise: Grover's algorithm

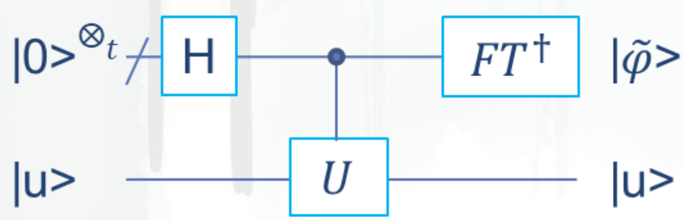
OPEN PROJECTQ/GROVER NOTEBOOK

# Phase estimation

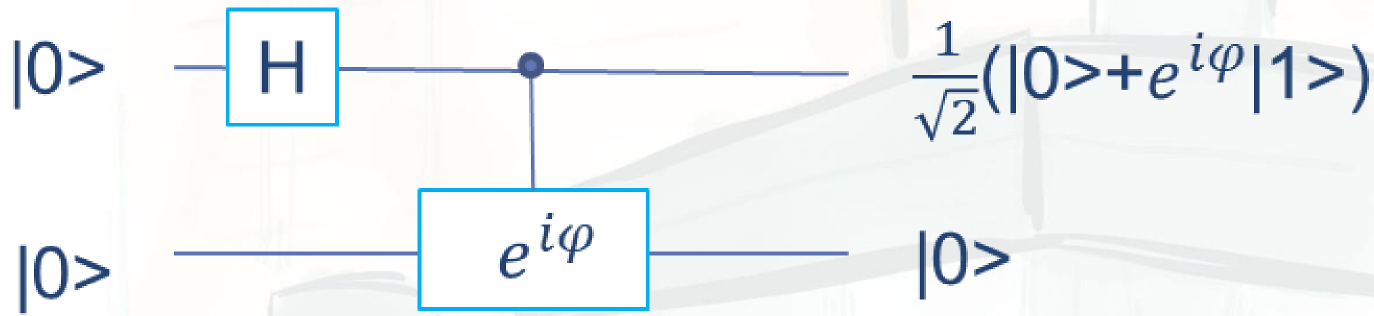
- Let **U** an unitary operation.
- Let |u> an eigenvector of U such that (which 0< **ϕ** <1):

$$U|u> = e^{i2\pi\phi}|u>$$

- Phase algorithm tray to estimate **ϕ** appying controled unitaries gates $\mathbf{U^{2n}}$, n ∈{0,1,...t}, being $\frac{1}{2^t}$ the precission of the approximation.
- The algorithm use **t** qubits to calculate the phase and **m** qubits to represent |u>.

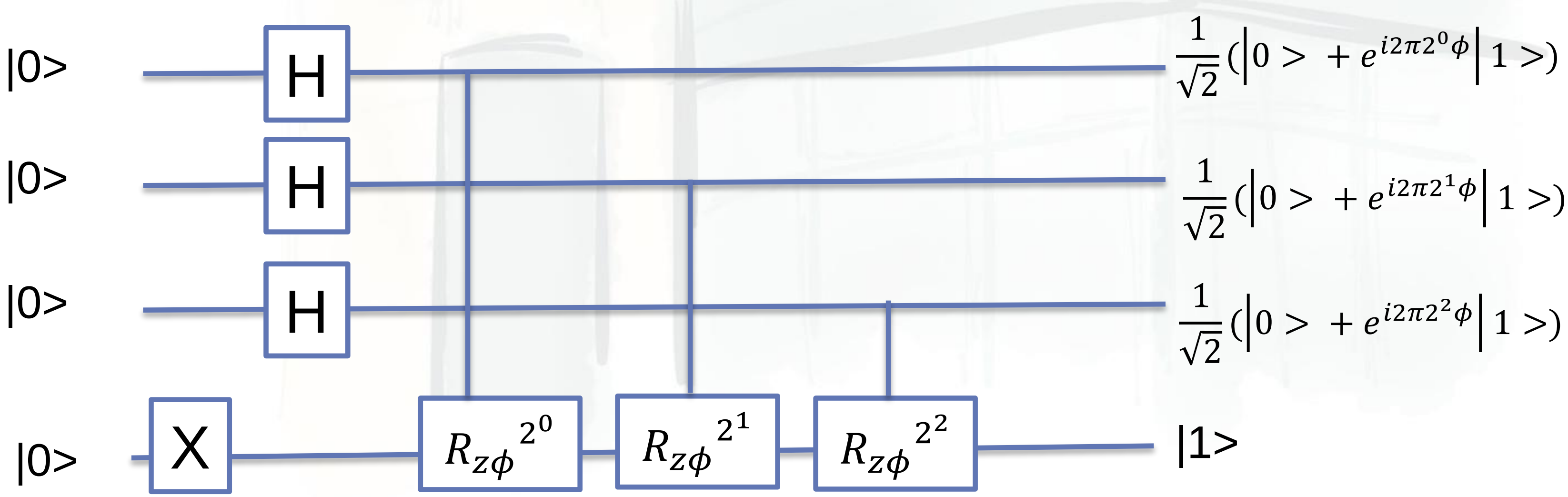


# Phase Kickback



# Phase estimation

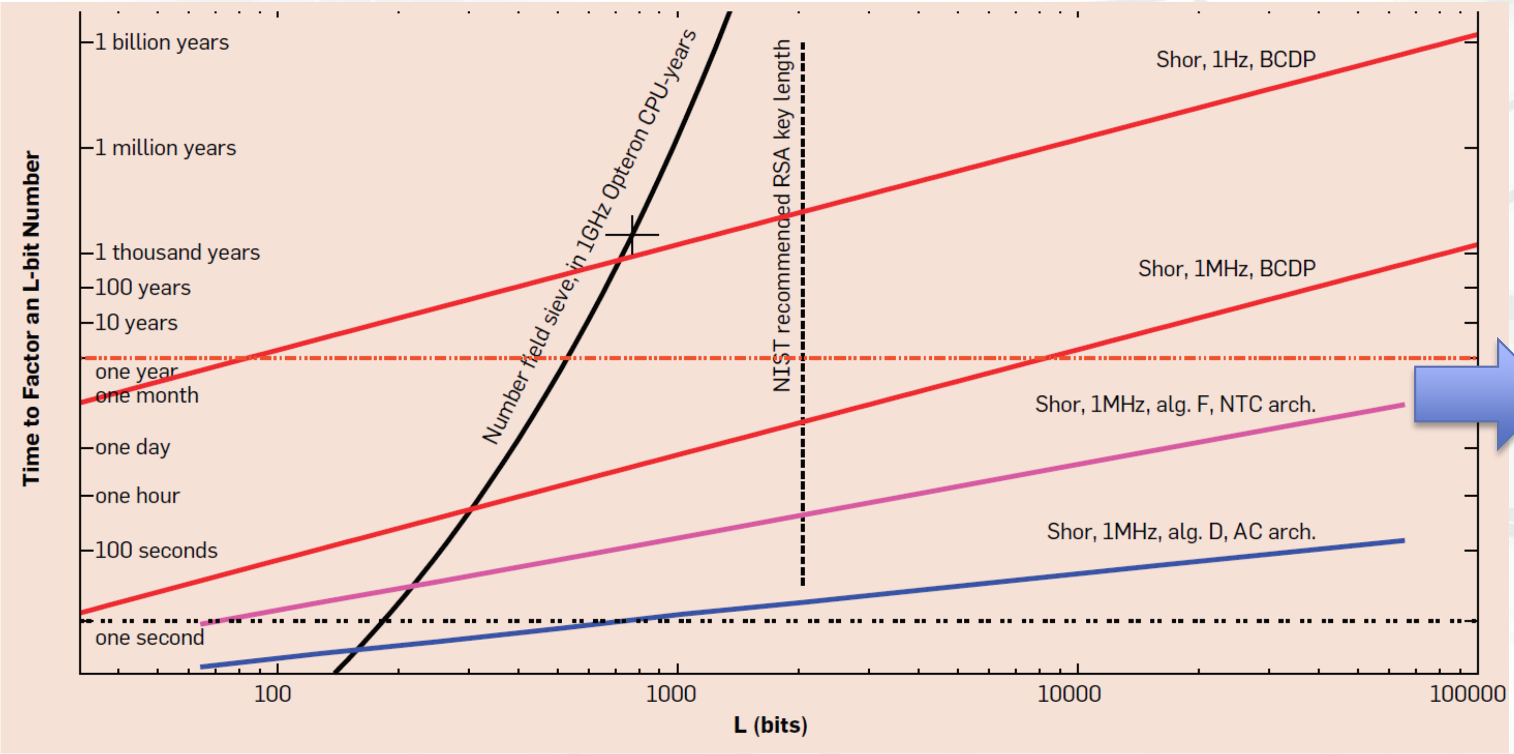
- Steps:
  - Allocate quantum register for n qubits ($q_n$).
  - Allocate a second quatum register for m qubits ($q_m$)
  - Initialice $q_m$ to |u>
  - Apply Walsh-Hadamard to $q_n$
  - Apply  Controled($U^{2n}$,n) on |u> for all n qubit  in $q_n$
  - Make $QFT^{-1}$ on $q_n$
  - Measure $q_n$
  - From the measurements (as integers) calculate **ϕ**



# Exercise: Phase Estimation

OPEN PROJECTQ/PHASE_ESTIMATION NOTEBOOK

# Shor's Factoring Algorithm




Post-Quantum Cryptography is needed (?)

AC: Abstract Concurrent Architecture. Supports ccNOT, concurrency and gate operands any distance apart
NTC: Neighbor-only, Two-qubit-gate, Concurrent architecture. Qbits in a line. Not ccNOT. Only two-qubits gates. Only neighborng qubits can operate
BCDP: Beckman, Chari, Devabhaktuni, and Preskill's algoritm

Van Meter, R., & Horsman, C. (2013). A blueprint for building a quantum computer. *Communications of the ACM*, *56*(10), 84. http://doi.org/10.1145/2494568
Meter, R. D. Van. (2006). Architecture of a Quantum Multicomputer Optimized for Shor's Factoring Algorithm. Arxiv:quant-ph/0607065

# Exercise: Shor's algorithm

OPEN PROJECTQ/SHOR NOTEBOOK

# Thanks!

# Questions?