Scientific Computing with Quantum Computers

Colin P. Williams

JPL

Jet Propulsion Laboratory,
California Institute of Technology
Pasadena, CA 91109-8099
Email: Colin.P.Williams@jpl.nasa.gov, Tel: (818) 393 6998

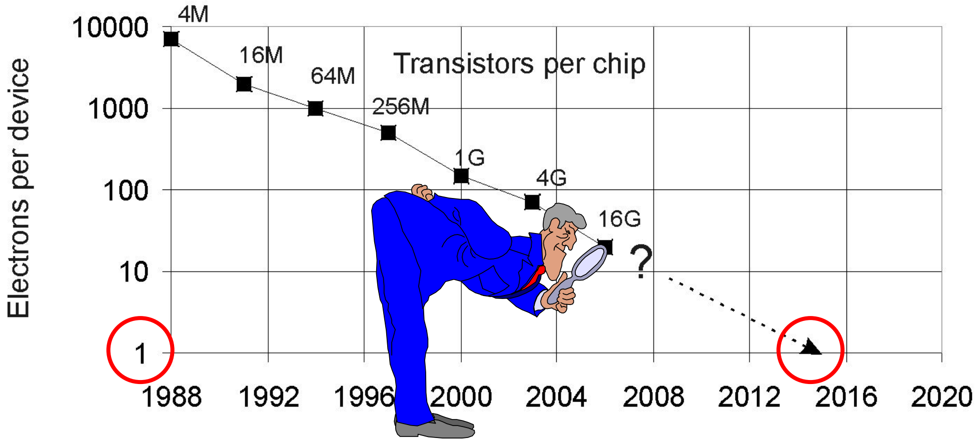
Quantum Algorithms Today

- **After 10 years and ~$1B worldwide spending, most significant quantum algorithm is still Shor's algorithm (1994)**
  - –Factoring composite integers (breaking public-key cryptosystems)
  - –Computing discrete logarithms (breaking elliptic curve cryptosystems)
- **Most U.S. $$$ on QC spent on:**
  - –Designing and implementing QC hardware that is scalable to run Shor's algorithm
  - –Relatively little being spent developing new quantum algorithms
- **Why?**
  - –Main funding agency has its "killer-ap" already. Weakens incentive for exploration.
  - –Sponsors strive for exponential speedups. Less glory in polynomial speedups.
  - –Culture: academic QC community not knowledgeable about HPC interests and issues
  - –Real-world computing data-dominated. QCs must encode data prior to processing it
    - Hidden cost can undermine any quantum complexity advantage
- **In this talk …**
  - –Quick overview of quantum computing
  - –Can quantum computers address scientific computing?
    - Quantum signal, image, data processing
    - Quantum algorithms for NP-Complete problems
    - Quantum simulation / quantum chemistry
  - –Discuss national strategy for finding significant new applications of QCs

Overview of Quantum Computing

Miniaturization Trend

- **Trend in miniaturization leading to quantum scales**



Miniaturization of computer technology as a function of time

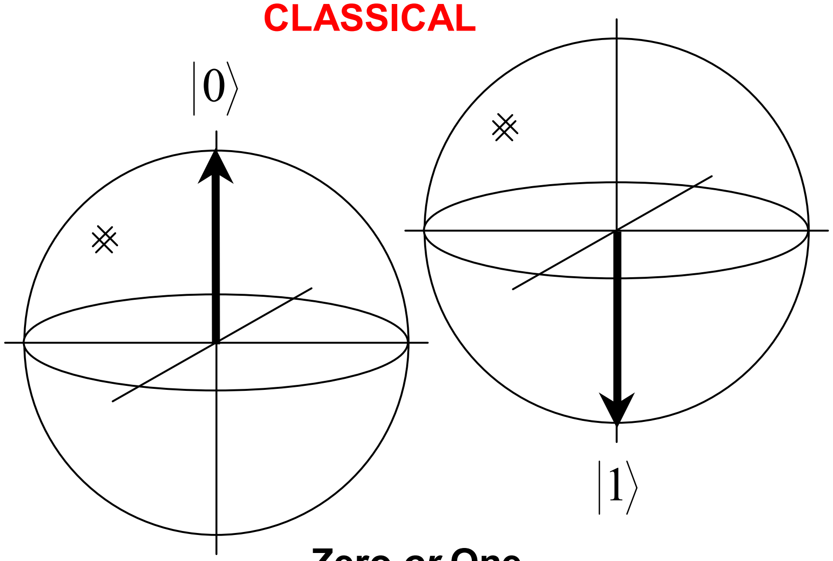
- **Gives computers access to new repertoire of physical effects**
  - Superposition, Interference, Entanglement, Non-locality, Non-determinism, Non-clonability
  - Allows fundamentally new ***kinds*** of algorithms

- **Nanotechnology may/may not exploit all quantum phenomena**
  - To maximize impact will need to harness ***uniquely*** quantum effects, e.g., entanglement

From Bits to Qubits

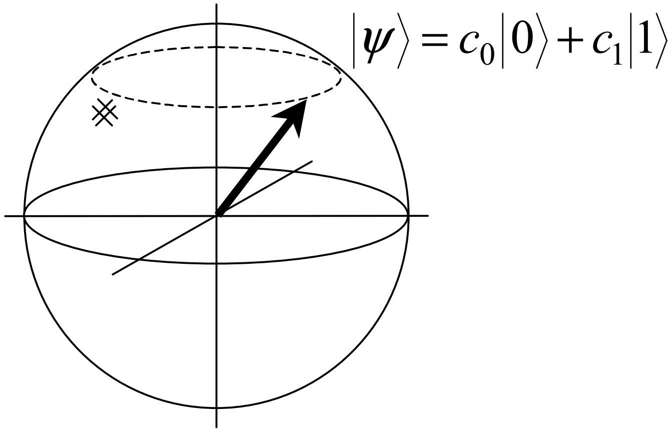
- **Use 2-state quantum systems for bits (0s and 1s) e.g. spins, polarized photons, atomic energy levels**

CLASSICAL



Zero *or* One

QUANTUM



Zero *and* One

- **A qubit can exist in a *superposition* state** $|\psi\rangle = c_0|0\rangle + c_1|1\rangle$ **s.t.** $|c_0|^2 + |c_1|^2 = 1$
- **Memory register, *n* qubits** $|\psi\rangle = c_0|000\ldots0\rangle + c_1|000\ldots1\rangle + \cdots + c_{2^n-1}|111\ldots1\rangle$
- **Potential for massive parallelism …but can't read out all answers**
  - *But can sometimes determine a collective property of the answers*

Entangled Qubits

- **Quintessential quantum property of qubits**
  - State of one qubit linked with that of another
- **Entangled state, e.g.,**

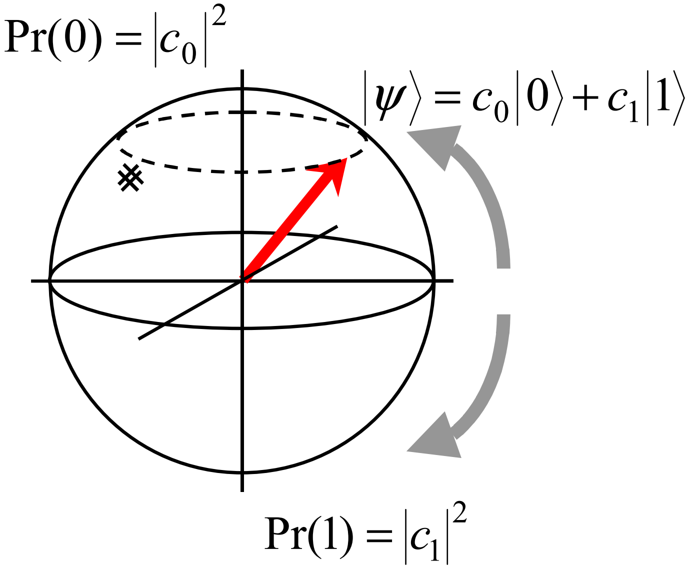
$$\frac{1}{\sqrt{2}}\left(|0\rangle_A|0\rangle_B+|1\rangle_A|1\rangle_B\right)\neq|\psi\rangle_A|\phi\rangle_B$$

- **Initially, neither "A" nor "B" has a definite bit value**
- **But measuring bit value of "A" determines that of "B" and vice versa**
- **Effect appears to propagate instantaneously independent of**
  - Distance between "A" and "B"
  - Nature of intervening medium
  - Recent experiments bound speed to > 10,000 c (Gisin, Geneva)

Reading Qubit Changes Qubit

- **Physically, “readout” depends on how qubit is implemented**
  - Spin-1/2 particle: measure spin orientation
  - Polarized photon: measure plane of polarization
  - Atomic energy levels: measure energy level
- **Non-deterministic outcome**

$\Pr(0) = |c_0|^2$



$|\psi\rangle = c_0|0\rangle + c_1|1\rangle$

$\Pr(1) = |c_1|^2$

- **Read qubit = project in** $\{|0\rangle, |1\rangle\}$ **basis**

Quantum Algorithms

- **Register evolves in accordance with Schrödinger eqn.**

$$i\hbar \frac{\partial \left|\psi\right\rangle}{\partial t} = H\left|\psi\right\rangle$$

- **with solution** $\left|\psi(t)\right\rangle = \exp(-iHt/\hbar)\left|\psi(0)\right\rangle = U\left|\psi(0)\right\rangle$

- **Make connection to computation:**

$$\begin{aligned} \left|\psi(0)\right\rangle &\leftrightarrow \text{input data} \\ U &\leftrightarrow \text{algorithm} \\ \left|\psi(t)\right\rangle &\leftrightarrow \text{output before measurement} \\ \left|00\ldots0\right\rangle \text{ or } \left|00\ldots1\right\rangle \text{ or } \cdots \text{ or } \left|11\ldots1\right\rangle &\leftrightarrow \text{output after measurement} \end{aligned}$$

Algorithm: Specification of a sequence of unitary transformations to apply to an input quantum state, followed by a measurement

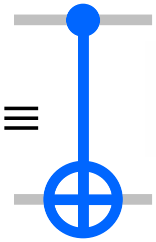
Quantum Circuits

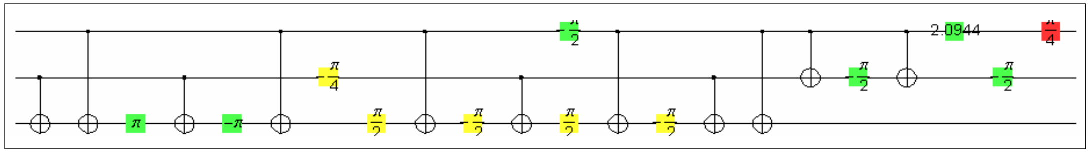
- **Quantum circuit is a decomposition of desired unitary matrix into sequence of single and pairwise quantum logic gates**

- **Only requires, e.g.**
  - *y*-rotations, *z*-rotations, phase-shifts, and controlled-NOT gates (CNOT)

$$R_y(\theta)=\begin{pmatrix}\cos\theta/2 & \sin\theta/2\\ -\sin\theta/2 & \cos\theta/2\end{pmatrix},\quad R_z(\xi)=\begin{pmatrix}e^{i\xi/2} & 0\\ 0 & e^{-i\xi/2}\end{pmatrix},\quad Ph(\theta)=\begin{pmatrix}e^{i\theta} & 0\\ 0 & e^{i\theta}\end{pmatrix}$$

$$CNOT=\begin{pmatrix}1&0&0&0\\0&1&0&0\\0&0&0&1\\0&0&1&0\end{pmatrix}\equiv$$



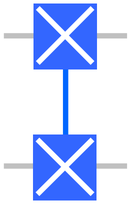


A typical quantum circuit

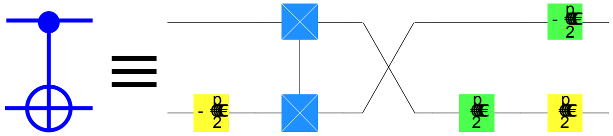
- **Other 2-qubit gates possible e.g.,**

$$iSWAP\equiv e^{i\left(\frac{\pi}{4}X\otimes X+\frac{\pi}{4}Y\otimes Y\right)}=\begin{pmatrix}1&0&0&0\\0&0&i&0\\0&i&0&0\\0&0&0&1\end{pmatrix}$$

  - … and equally efficient as CNOT



iSWAP **circuit icon**



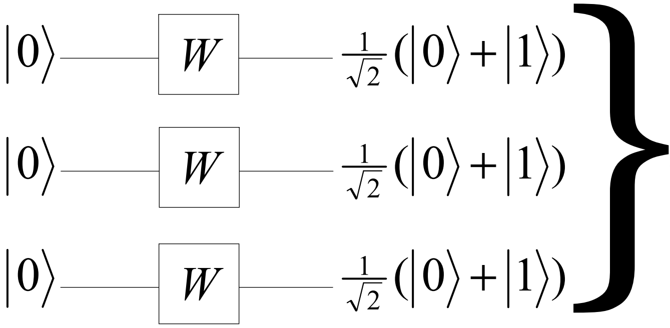
Walsh-Hadamard Gates

- **Some gates are especially useful, e.g., Walsh-Hadamard gates**

$$W = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \equiv -\boxed{W}-$$

- W **gates create a superposition of exponentially many (i.e., 2^n) terms in polynomially many (i.e., n) operations**

$$\underbrace{(W|0\rangle)\otimes(W|0\rangle)\otimes(W|0\rangle)}_{n \text{ operations}} = \underbrace{\frac{1}{2\sqrt{2}}\left(|000\rangle+|001\rangle+|010\rangle+|011\rangle+|100\rangle+|101\rangle+|110\rangle+|111\rangle\right)}_{2^n \text{ components}}$$



$|0\rangle$ — $\boxed{W}$ — $\frac{1}{\sqrt{2}}(|0\rangle+|1\rangle)$

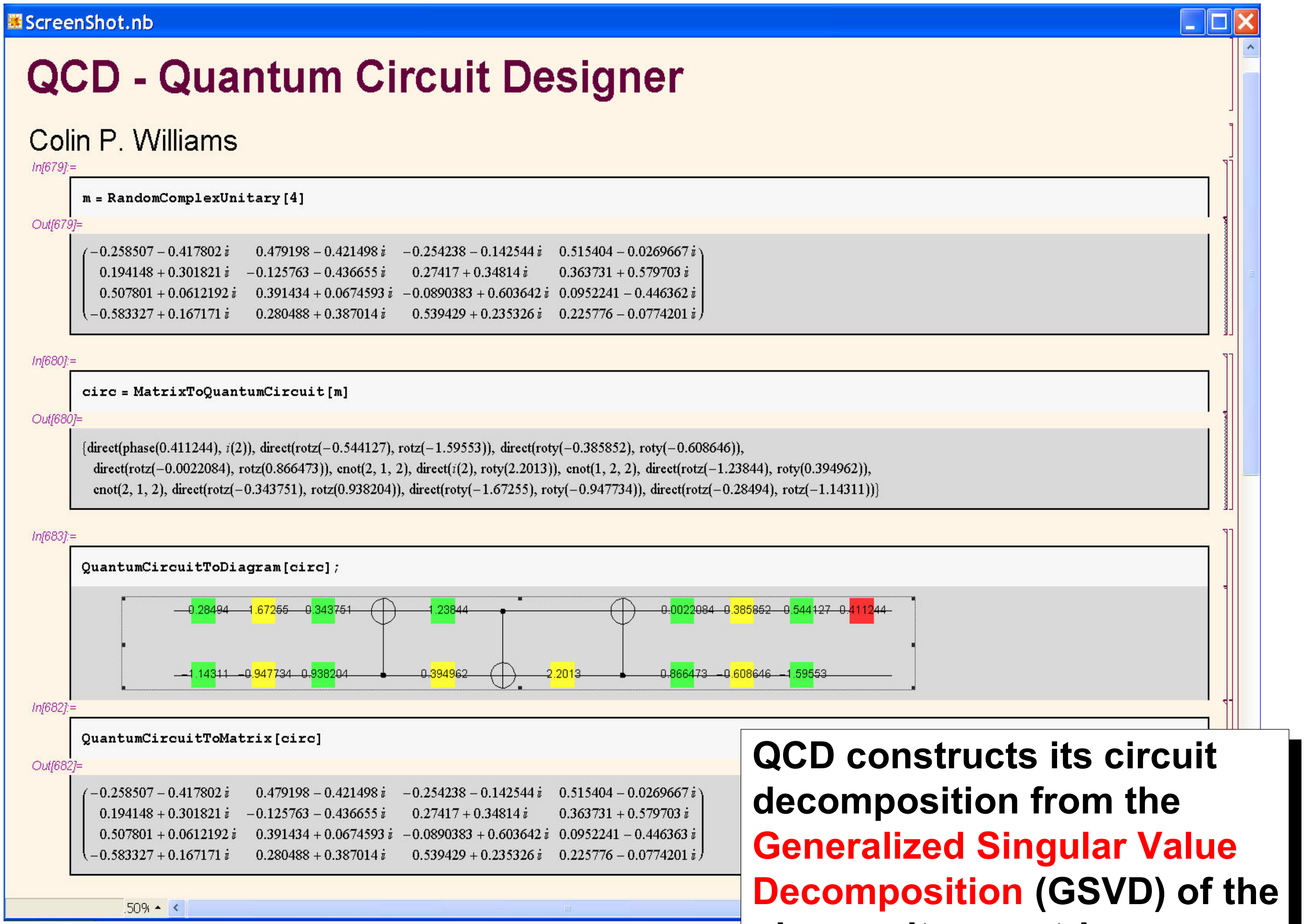
$|0\rangle$ — $\boxed{W}$ — $\frac{1}{\sqrt{2}}(|0\rangle+|1\rangle)$

$|0\rangle$ — $\boxed{W}$ — $\frac{1}{\sqrt{2}}(|0\rangle+|1\rangle)$

$$\equiv \frac{1}{2\sqrt{2}}\left(|000\rangle+|001\rangle+|010\rangle+|011\rangle+|100\rangle+|101\rangle+|110\rangle+|111\rangle\right)$$

Automated Design of Quantum Circuits

- Given a $2^n \times 2^n$ unitary, QCD finds a circuit for it



QCD - Quantum Circuit Designer

Colin P. Williams

In[679]:=

```
m = RandomComplexUnitary[4]
```

Out[679]=

$$\begin{pmatrix} -0.258507-0.417802\,i & 0.479198-0.421498\,i & -0.254238-0.142544\,i & 0.515404-0.0269667\,i \\ 0.194148+0.301821\,i & -0.125763-0.436655\,i & 0.27417+0.34814\,i & 0.363731+0.579703\,i \\ 0.507801+0.0612192\,i & 0.391434+0.0674593\,i & -0.0890383+0.603642\,i & 0.0952241-0.446362\,i \\ -0.583327+0.167171\,i & 0.280488+0.387014\,i & 0.539429+0.235326\,i & 0.225776-0.0774201\,i \end{pmatrix}$$

In[680]:=

```
circ = MatrixToQuantumCircuit[m]
```

Out[680]=

```
{direct(phase(0.411244), i(2)), direct(rotz(−0.544127), rotz(−1.59553)), direct(roty(−0.385852), roty(−0.608646)),
  direct(rotz(−0.0022084), rotz(0.866473)), cnot(2, 1, 2), direct(i(2), roty(2.2013)), cnot(1, 2, 2), direct(rotz(−1.23844), roty(0.394962)),
  cnot(2, 1, 2), direct(rotz(−0.343751), rotz(0.938204)), direct(roty(−1.67255), roty(−0.947734)), direct(rotz(−0.28494), rotz(−1.14311))}
```

In[683]:=

```
QuantumCircuitToDiagram[circ];
```

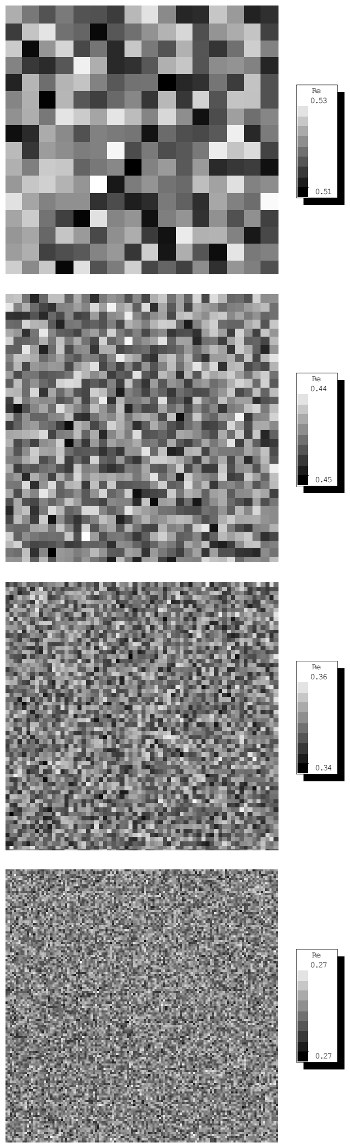
In[682]:=

```
QuantumCircuitToMatrix[circ]
```

Out[682]=

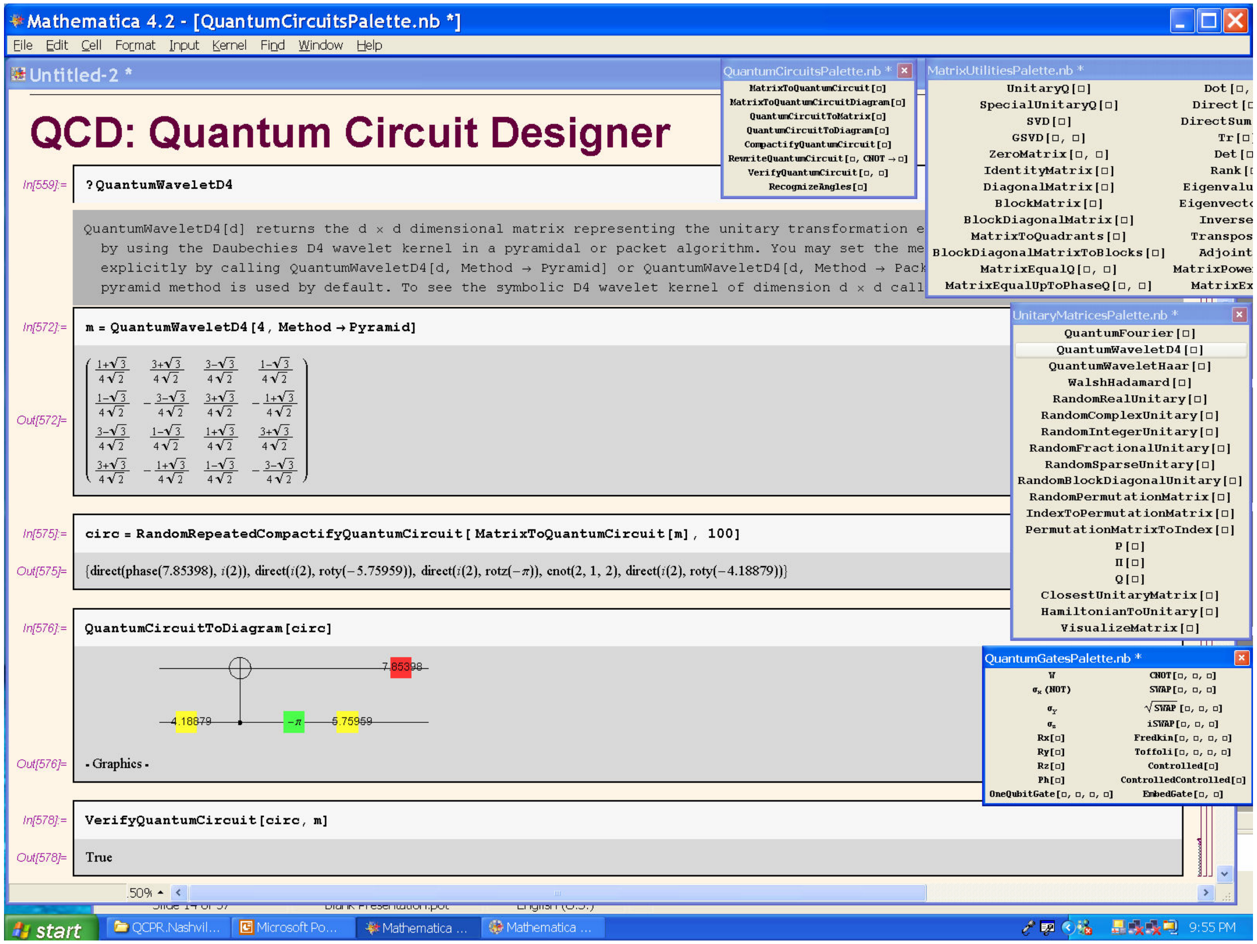
$$\begin{pmatrix} -0.258507-0.417802\,i & 0.479198-0.421498\,i & -0.254238-0.142544\,i & 0.515404-0.0269667\,i \\ 0.194148+0.301821\,i & -0.125763-0.436655\,i & 0.27417+0.34814\,i & 0.363731+0.579703\,i \\ 0.507801+0.0612192\,i & 0.391434+0.0674593\,i & -0.0890383+0.603642\,i & 0.0952241-0.446363\,i \\ -0.583327+0.167171\,i & 0.280488+0.387014\,i & 0.539429+0.235326\,i & 0.225776-0.0774201\,i \end{pmatrix}$$

QCD constructs its circuit decomposition from the **Generalized Singular Value Decomposition** (GSVD) of the given unitary matrix



“Useful” Unitaries have Compact Circuits

- **e.g., QWT (D4 kernel in pyramid algorithm)**
  - QCD finds compact form automatically



Mathematica 4.2 - [QuantumCircuitsPalette.nb *]
File Edit Cell Format Input Kernel Find Window Help
Untitled-2 *
QuantumCircuitsPalette.nb *
MatrixUtilitiesPalette.nb *
UnitaryMatricesPalette.nb *
QuantumGatesPalette.nb *
7.85398
4.18879
−π
5.75959
50%
start
QCPR Nashvil...
Microsoft Po...
Mathematica ...
Mathematica ...
9:55 PM

QCD: Quantum Circuit Designer

In[559]:= ?QuantumWaveletD4

```
QuantumWaveletD4[d] returns the d × d dimensional matrix representing the unitary transformation e
  by using the Daubechies D4 wavelet kernel in a pyramidal or packet algorithm. You may set the me
  explicitly by calling QuantumWaveletD4[d, Method → Pyramid] or QuantumWaveletD4[d, Method → Pack
  pyramid method is used by default. To see the symbolic D4 wavelet kernel of dimension d × d call
```

In[572]:= m = QuantumWaveletD4[4, Method → Pyramid]

Out[572]=

$$\begin{pmatrix} \frac{1+\sqrt{3}}{4\sqrt{2}} & \frac{3+\sqrt{3}}{4\sqrt{2}} & \frac{3-\sqrt{3}}{4\sqrt{2}} & \frac{1-\sqrt{3}}{4\sqrt{2}} \\ \frac{1-\sqrt{3}}{4\sqrt{2}} & -\frac{3-\sqrt{3}}{4\sqrt{2}} & \frac{3+\sqrt{3}}{4\sqrt{2}} & -\frac{1+\sqrt{3}}{4\sqrt{2}} \\ \frac{3-\sqrt{3}}{4\sqrt{2}} & \frac{1-\sqrt{3}}{4\sqrt{2}} & \frac{1+\sqrt{3}}{4\sqrt{2}} & \frac{3+\sqrt{3}}{4\sqrt{2}} \\ \frac{3+\sqrt{3}}{4\sqrt{2}} & -\frac{1+\sqrt{3}}{4\sqrt{2}} & \frac{1-\sqrt{3}}{4\sqrt{2}} & -\frac{3-\sqrt{3}}{4\sqrt{2}} \end{pmatrix}$$

In[575]:= circ = RandomRepeatedCompactifyQuantumCircuit[MatrixToQuantumCircuit[m], 100]

Out[575]= {direct(phase(7.85398), i(2)), direct(i(2), roty(−5.75959)), direct(i(2), rotz(−π)), cnot(2, 1, 2), direct(i(2), roty(−4.18879))}

In[576]:= QuantumCircuitToDiagram[circ]

Out[576]= -Graphics-

In[578]:= VerifyQuantumCircuit[circ, m]

Out[578]= True

QuantumCircuitsPalette.nb:

MatrixToQuantumCircuit[□]
MatrixToQuantumCircuitDiagram[□]
QuantumCircuitToMatrix[□]
QuantumCircuitToDiagram[□]
CompactifyQuantumCircuit[□]
RewriteQuantumCircuit[□, CNOT → □]
VerifyQuantumCircuit[□, □]
RecognizeAngles[□]

MatrixUtilitiesPalette.nb:

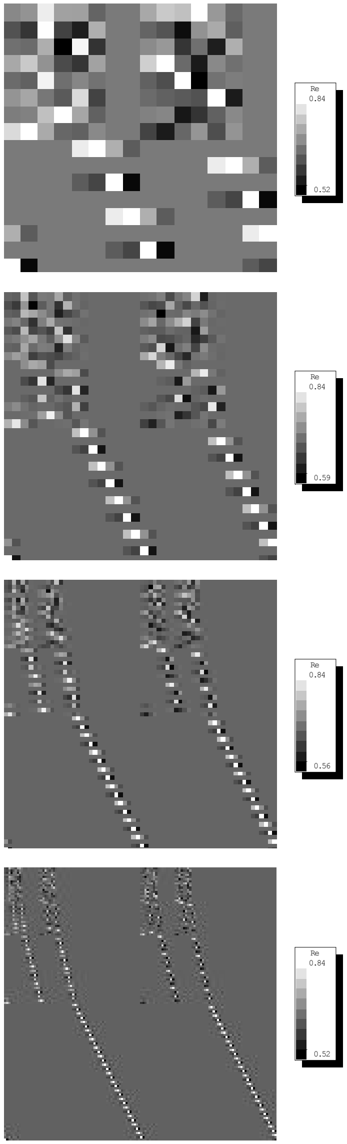
UnitaryQ[□]	Dot[□,
SpecialUnitaryQ[□]	Direct[□
SVD[□]	DirectSum
GSVD[□, □]	Tr[□
ZeroMatrix[□, □]	Det[□
IdentityMatrix[□]	Rank[□
DiagonalMatrix[□]	Eigenvalu
BlockMatrix[□]	Eigenvecto
BlockDiagonalMatrix[□]	Inverse
MatrixToQuadrants[□]	Transpos
BlockDiagonalMatrixToBlocks[□]	Adjoint
MatrixEqualQ[□, □]	MatrixPowe
MatrixEqualUpToPhaseQ[□, □]	MatrixEx

UnitaryMatricesPalette.nb:

QuantumFourier[□]
QuantumWaveletD4[□]
QuantumWaveletHaar[□]
WalshHadamard[□]
RandomRealUnitary[□]
RandomComplexUnitary[□]
RandomIntegerUnitary[□]
RandomFractionalUnitary[□]
RandomSparseUnitary[□]
RandomBlockDiagonalUnitary[□]
RandomPermutationMatrix[□]
IndexToPermutationMatrix[□]
PermutationMatrixToIndex[□]
P[□]
Π[□]
Q[□]
ClosestUnitaryMatrix[□]
HamiltonianToUnitary[□]
VisualizeMatrix[□]

QuantumGatesPalette.nb:

W	CNOT[□, □, □]
σ_x (NOT)	SWAP[□, □, □]
σ_y	$\sqrt{\text{SWAP}}$[□, □, □]
σ_z	iSWAP[□, □, □]
Rx[□]	Fredkin[□, □, □, □]
Ry[□]	Toffoli[□, □, □, □]
Rz[□]	Controlled[□]
Ph[□]	ControlledControlled[□]
OneQubitGate[□, □, □, □]	EmbedGate[□, □]



See A. Fijany and C. P. Williams, “Quantum Wavelet Transforms: Fast Algorithms & Complete Circuits”, Springer-Verlag, LNCS Vol. 1509, (1998), pp.10-33.

Quantum Speedups

- **Exponential Speedup**
  - Deciding whether a function is constant or balanced (Deutsch)
  - Sampling from Fourier Transform (Simon)
  - Factoring Integers (Shor)
  - Simulating Quantum Systems (Abrams / Lloyd)
  - Computing Eigenvalues (Abrams)
  - Sampling from Wavelet Transform (Hoyer, Fijany / Williams, Klappenecker)
  - Sampling from Discrete Cosine, Hartley Transforms (Markov)
  - Solving Pell's Equation (Halgren)

- **Polynomial Speedup**
  - Searching unstructured virtual databases (Grover)
  - Solving NP-Complete/NP-Hard problems (Cerf / Grover / Williams)
  - Finding function collisions (Brassard)
  - Estimating Means, Medians, Maxima and Minima (Grover, Nayak/Wu, Abrams/Williams)
  - Counting Number of Solutions (Brassard/Hoyer/Tapp)
  - Evaluating High-dimensional Numerical Integrals (Abrams / Williams)
  - Template Matching (Jozsa)

Quantum Signal, Image and Data Processing

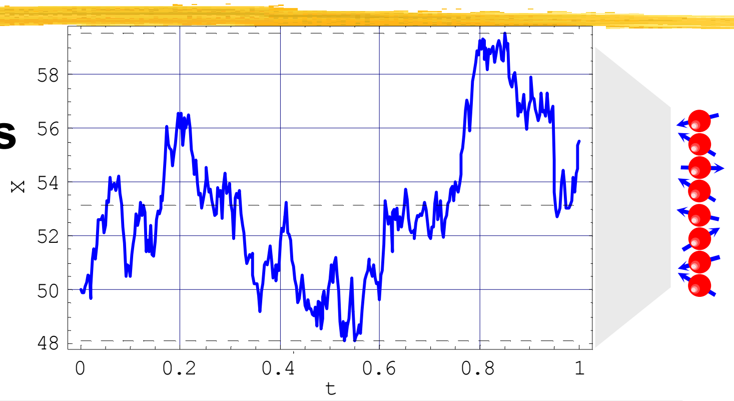
Signal, Image and Data Processing

- **Signal, image and data processing fundamentally different on a quantum computer than classical computer**
  - –Classical-to-quantum data encoding
    - Polynomial (possibly linear) in size of data cost
  - –Quantum processing
    - Some operations yield exponential speedups
    - e.g., quantum versions of Fourier, wavelet, and cosine transforms
  - –Quantum-to-classical readout
    - Cannot “see” result in conventional sense
    - Can sample from, or obtain collective properties of, processed signal, image or data
- **Can process an image exponentially more efficiently, report on a property of interest, but be unable to display the result**
  - –Quantum world strongly distinguishes truth from proof
- **Let’s look at how to enter data into a quantum computer**

Data-Entry on a Quantum Computer

- **Encode 2^n data values as the amplitudes of just *n* qubits**

$$|\psi\rangle = \sum_{i=0}^{2^n-1} c_i |i\rangle \equiv \begin{pmatrix} c_0 \\ c_1 \\ \vdots \\ c_{2^n-1} \end{pmatrix}$$



Algorithm DataEntry:

Step 1: Normalize the data vector, and pad it to length $2^{\lceil \log_2 \|c\| \rceil}$, i.e., compute $c_i' \leftarrow \dfrac{c_i}{\sum_i |c_i|^2}$

Step 2: Interpret c_i' as the amplitudes of the pure state $|\psi'\rangle$

Step 3: w.l.o.g. assume amplitude $c_0' \neq 0$ (otherwise permute basis until $c_0' \neq 0$)

Step 4: Construct the matrix *M* defined by:

$$M = \left(\begin{array}{c|cccc} c_0' & & & & \\ \hline c_1' & 1 & & & \\ \vdots & & 1 & & \\ \vdots & & & 1 & \\ c_{2^n-1}' & & & & 1 \end{array}\right)$$

Step 5: Use Gram-Schmidt process to fix first column as $|\psi'\rangle$ and compute orthonormal columns for the rest of the matrix

Step 6: Map this unitary matrix into an equivalent quantum circuit using QCD circuit design tool

Output: A circuit for synthesizing an arbitrary data input to a quantum computer

Common Signal Processing Unitaries Exponentially Faster on a Quantum Computer

- **Discrete Fourier Transform (DFT)**

$$(x_0, x_1, \ldots, x_{N-1}) \xrightarrow{DFT} (y_0, y_1, \ldots, y_{N-1}): \quad y_k = \frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} x_j e^{2\pi i\, jk/N}$$

- **Quantum Fourier Transform (DFT)**
  - Same except “vector” is now stored in amplitudes of a superposition state

$$\sum_{j=0}^{N-1} x_j |j\rangle \xrightarrow{QFT} \sum_{k=0}^{N-1} y_k |k\rangle: \quad |j\rangle \to \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} e^{2\pi i\, jk/N} |k\rangle$$

  - Defining $|j\rangle \equiv |j_1, j_2, \ldots, j_n\rangle: \quad j = j_1 2^{n-1} + j_2 2^{n-2} + \cdots + j_n 2^0$
  - Defining $0.j_\ell j_{\ell+1} \ldots j_m \equiv j_\ell / 2 + j_{\ell+1} / 4 + \cdots + j_m / 2^{m-\ell+1}$
  - Can factor mapping of basis state into following product state

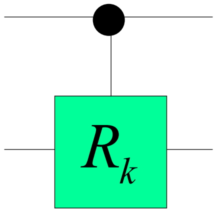
$$|j\rangle \equiv |j_1 j_2 \ldots j_n\rangle \to \frac{1}{2^{n/2}} \left[\left(|0\rangle + e^{2\pi i 0.j_n} |1\rangle\right) \left(|0\rangle + e^{2\pi i 0.j_{n-1} j_n} |1\rangle\right) \cdots \left(|0\rangle + e^{2\pi i 0.j_1 j_2 \cdots j_n} |1\rangle\right) \right]$$

  - Hence QFT maps basis states into product states and therefore has an efficient quantum circuit factorization

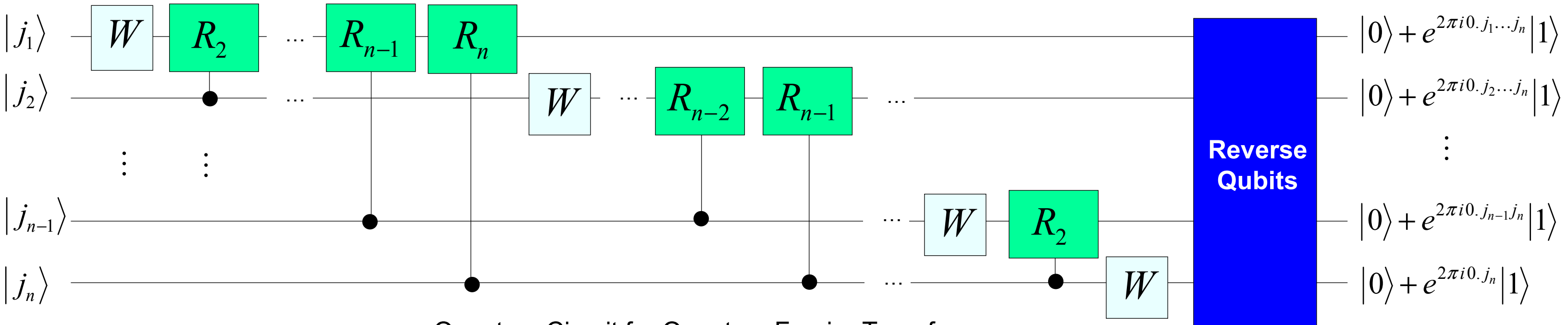
Quantum Circuit for QFT

- **Need Walsh-Hadamard and controlled 1-qubit gates**

$$W = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}; \quad R_k = \begin{pmatrix} 1 & 0 \\ 0 & e^{2\pi i/2^k} \end{pmatrix}$$



$$\equiv \left(\begin{array}{cc:cc} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ \hdashline 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & e^{2\pi i/2^k} \end{array}\right)$$


$|j_1\rangle$
$|j_2\rangle$
$|j_{n-1}\rangle$
$|j_n\rangle$
W
R_2
R_{n-1}
R_n
W
R_{n-2}
R_{n-1}
W
R_2
W
Reverse Qubits
$|0\rangle + e^{2\pi i 0.j_1 \dots j_n}|1\rangle$
$|0\rangle + e^{2\pi i 0.j_2 \dots j_n}|1\rangle$
$|0\rangle + e^{2\pi i 0.j_{n-1} j_n}|1\rangle$
$|0\rangle + e^{2\pi i 0.j_n}|1\rangle$

Quantum Circuit for Quantum Fourier Transform

- **DFT (actually FFT) of length *N* vector takes O(*N* log *N*) steps**
- **QFT of length *N* vector takes O((log*N*)^2) steps**
  - Exponential speedup
  - But cannot see the result, can only compute with it or sample from it

- **QFT maps eigenstates into phase factors**
  - QFT^{-1} maps phase factors into eigenstates

Quantum Algorithms for NP-Complete Problems

Foundation is Quantum Search Algorithm

- **Invented by Lov Grover, Bell Labs, in 1996**
  - –L. Grover, "*A Fast Quantum Mechanical Algorithm for Database Search*", in Proceedings of the 28th Annual ACM Symposium on the Theory of Computing (1996) pp212-219.
  - –G. Brassard, "*Searching a Quantum Phone Book*", Science, January 31st (1997) pp.627-628.
- **Problem: Find the name of the person in a (*virtual*) telephone directory who has a prescribed telephone number**
  - –Suppose N entries in directory
  - –Classical: need $O(N)$ queries in worst case
  - –Quantum: need $O(N^{1/2})$ queries in worst case
- **Gives *polynomial* speedup**
- **Use as subroutine in higher-level quantum algorithms**
- **To use quantum search to search real datasets must …**
  - –Replace the "oracle" in Grover's original algorithm with a polynomial cost tester circuit (returns true if input is a solution, false otherwise)

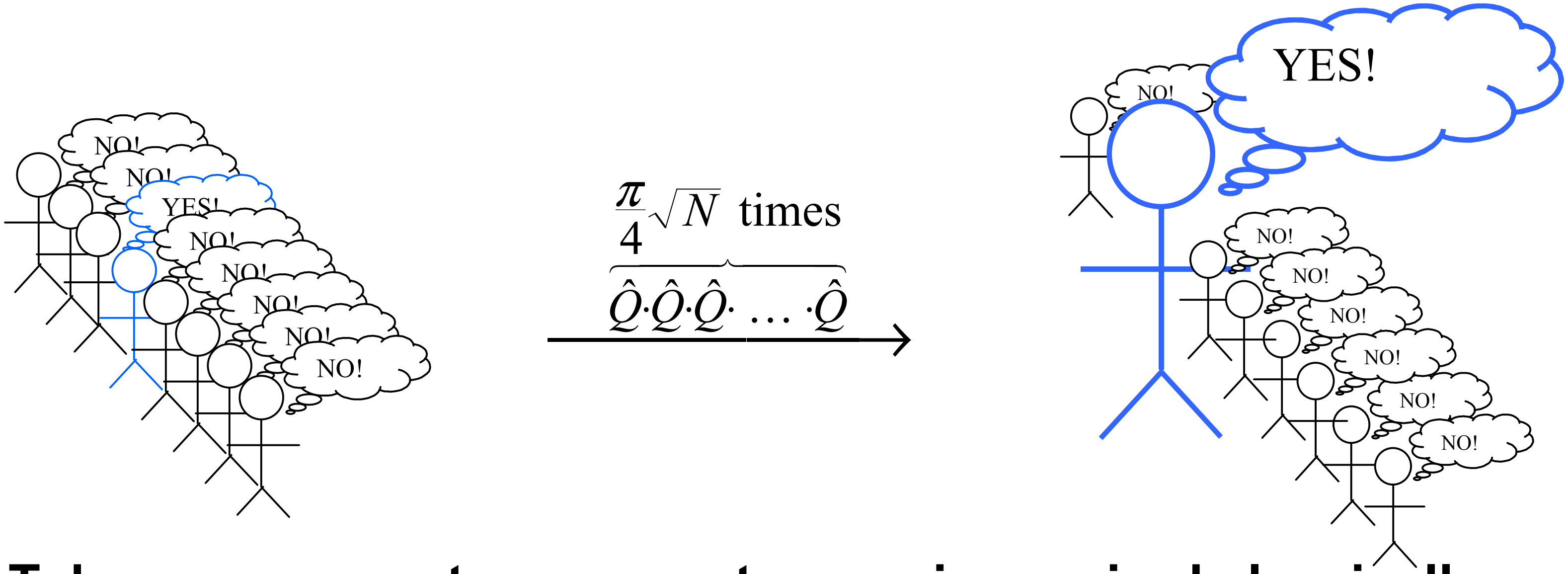
Amplitude Amplification Boosts “Signal”

Step 1: Create equally weighted superposition of all *N* candidates

Step 2: Synthesize amplitude amplification op.

Step 3: Apply *Q* $\frac{\pi}{4}\sqrt{N}$ **times**

Step 4: Read register – will obtain target index with probability *O*(1)


NO!
NO!
YES!
NO!
NO!
NO!
NO!
NO!
$\frac{\pi}{4}\sqrt{N}$ times
$\overbrace{\hat{Q}\cdot\hat{Q}\cdot\hat{Q}\cdot\ \ldots\ \cdot\hat{Q}}$
YES!
NO!
NO!
NO!
NO!
NO!
NO!
NO!

- **Takes square root as many steps as is required classically**
- **Fundamental algorithmic advance that is only possible on a quantum computer**

Grover’s Algorithm

Grover’s Algorithm for Unstructured Quantum Search

Step 1. Given a black box oracle, embodied as a quantum circuit for computing the operator $\hat{I}_{f_t}$ which inverts the sign of the target eigenstate $|t\rangle$

Step 2. Construct the operator $\hat{Q} = -\hat{U} \cdot \hat{I}_s \cdot \hat{U} \cdot \hat{I}_{f_t}$ where

- $|s\rangle$ is a superposition of equally weighted indices (unbiased starting state)
- $|t\rangle$ is the (unknown) target index that you are seeking
- $\hat{I}_s = 1 - 2|s\rangle\langle s|$ inverts the sign of the starting state
- $\hat{I}_{f_t} = 1 - 2|t\rangle\langle t|$ is the unitary operator representing the oracle
- $\hat{U}$ is any unitary matrix having only non-zero elements

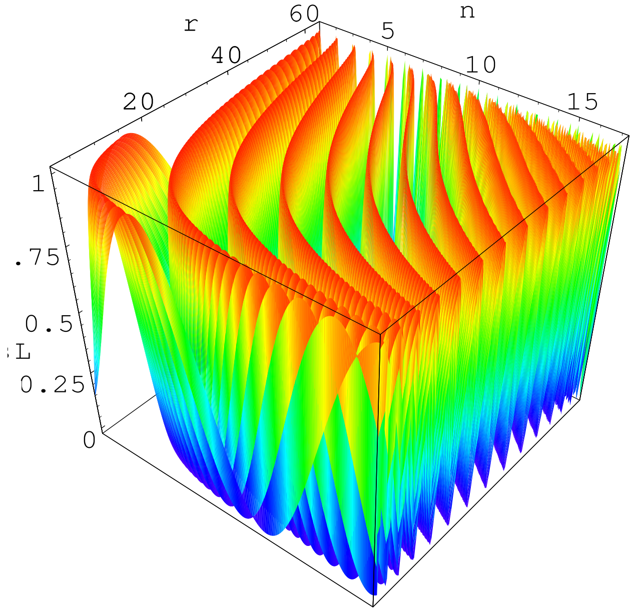
Step 3. Compute $|\psi\rangle = \hat{Q}^k \hat{U}|s\rangle$ i.e., $k = \frac{\pi}{4}\sqrt{N}$ repetitions of $\hat{Q}$ on $\hat{U}|s\rangle$

Step 4. Read the values of the first *n* bits of $|\psi\rangle$

With high probability, result will be the target index $|t\rangle$

Success Probability Oscillates

- **Probability of success of quantum search as a function of the number of solutions (targets), *r*, and the number of amplitude amplification operations, *n***



- **For fixed number of solutions, success probability rises and falls**
- **Quantum search is like baking a souffle**
  - It is possible to over-bake the batter and ruin the result

Do Slightly Better (on Average) with Punctuated Quantum Search

Punctuated Quantum Search (speeding up average case quantum search)

Step 1. Initialize registers as if you were about to perform quantum search

Step 2. Perform amplitude amplification for $i < \frac{\pi}{4}\sqrt{N}$ iterations

Step 3. Read the values of the first *n* bits of $|\psi\rangle$

With some probability < 1, result will be the target index $|t\rangle$

Success probability after i iterations of Grover's algorithm is

$$p(i) \approx \left(\tfrac{1}{\sqrt{N}}\cos\left(\tfrac{2i}{\sqrt{N}}\right) + \sin\left(\tfrac{2i}{\sqrt{N}}\right)\right)^2$$

Expected number of steps to success using punctuated quantum search

$$C_{avg} = ip(i) + 2ip(i)(1-p(i)) + 3ip(i)(1-p(i))^2 + \ldots$$

$$= \sum_{j=1}^{\infty} jip(i)(1-p(i))^{j-1} = \frac{i}{p(i)}$$

Hence, can find value of *i* that minimizes expected cost

Result: Punctuated quantum search is 12% faster (on average) than Grover search

Quantum Search with Parallelism

- **Augment quantum search with classical parallelism**
  - –Consider *k* independent agents, searching in parallel
  - –Imposes lower demands on the coherence time per processor

***k*-Parallel Quantum Search**

Predicted optimal number of amplitude amplification operations to use for each of *k* agents performing punctuated quantum searches in parallel, when there are *r* solutions amongst *N* candidates is:

$$n_{\text{optimal}}(r,N,k) \approx \frac{1}{2}\left(\sqrt{\frac{5-15k+\sqrt{5}\sqrt{-31-30k+225k^2}}{-3+15k^2}}\sqrt{\frac{N}{r}}-1\right)$$

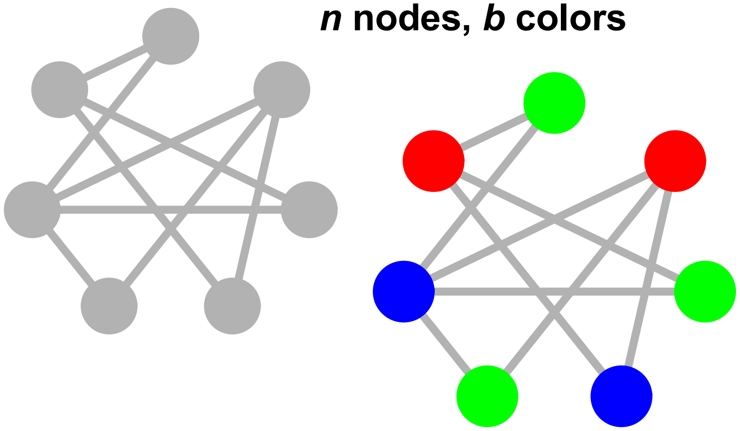
Formula has practical utility. In any implementation, can only maintain coherence for time $n_{\text{coherence}}$. By setting $n_{\text{coherence}} = n_{\text{optimal}}(r,N,k)$ we can invert this equation to determine how many agents we need to solve the problem within the attainable coherence time.

See R. M. Gingrich, C. P. Williams, and N. J. Cerf, “Generalized Quantum Search with Parallelism,” Physical Review A, Vol. 61, (2000).

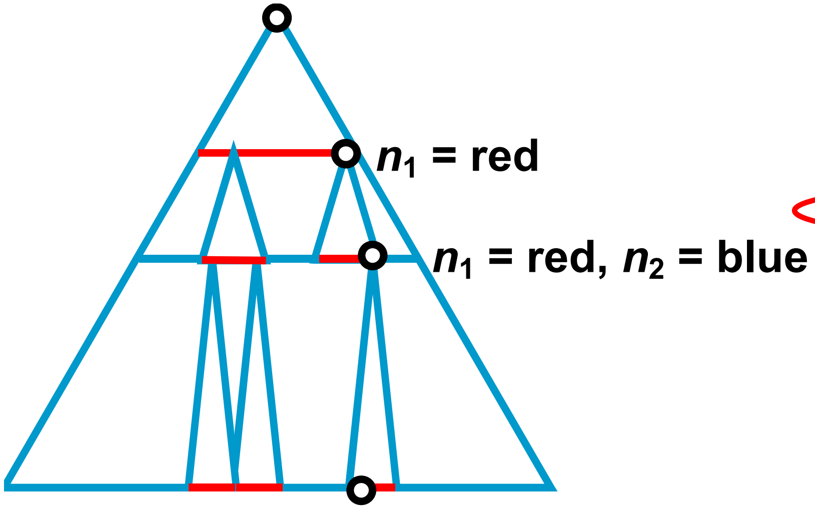
Nested Quantum Search Beats Naïve Quantum Search *and* Good Classical Algorithm

NP-Complete Problems

n nodes, *b* colors



Induces tree-structured search space



Nested Quantum Search

Step 1: Superposition of consistent partial solutions at intermediate level

Step 2: Perform amplitude amplification in the subspace of their descendants

Step 3: Nest Step 1 inside Step 2

Comparison

- **Best classical tree search O($b^{0.446n}$)**
- **Naïve Quantum Search O($b^{0.5n}$)**
- **Structured Quantum Search O($b^{0.333n}$)**

- N. Cerf, L. Grover, C. P. Williams, "*Nested Quantum Search and Structured Problems*," Phys. Rev. A, 61, 032303, 9th February (2000)
- C. P. Williams, "*Quantum Search Algorithms in Science and Engineering*", Colin P. Williams, Computing in Science and Engineering, IEEE Computer Society, April (2001).

Quantum Simulation

The Problem

- **How do you simulate the evolution of a quantum system, and extract useful predictions, *efficiently?***
  - –Problem appears to be impossible classically for all but the simplest systems

- **All phases of the quantum simulation must be efficient**
  - – Preparing the initial state
  - – Evolving the state
  - – Extracting an answer

- **In 1982 Feynman speculated quantum computers might simulate quantum systems efficiently**
  - –Bosons (yes) / fermions (???)
  - –Lloyd showed this is true for bosons and fermions
  - –… and we now know its true for anyons too
  - –Lloyd/Abrams then constructed an explicit quantum algorithm for computing properties of a quantum system

The Solution

- **Idea: use the evolution of one quantum system to emulate the evolution of another**

$$\begin{tikzcd}
{|s(0)\rangle} \arrow[r, "M", "\text{Encode}"'] \arrow[d] & {|\psi(0)\rangle} \arrow[d, "\text{Evolve}"] & \text{Initialize} \\
{|s(t)\rangle} & {|\psi(t)\rangle} \arrow[l, "M^{-1}"', "\text{Decode}"] & \text{Measure}
\end{tikzcd}$$

Quantum Simulation in Small Increments

- **Schrödinger equation with time independent, local, Hamiltonian**

$$i\hbar \frac{\partial |\psi\rangle}{\partial t} = H|\psi\rangle$$

  - i.e., $H = \sum_{\ell=1}^{L} H_\ell$ and only involves few-body interactions
  - Solution $|\psi(t)\rangle = e^{-iHt/\hbar}|\psi(0)\rangle = U|\psi(0)\rangle$

- **If you can find an efficient quantum circuit for U, then you're done**
  - But, in general, this is hard and/or costly to do exactly
  - Moreover, if $\exists \ell, \ell' : [H_\ell, H_{\ell'}] \neq 0$ then $\exp(-iHt/\hbar) = \exp(-i\sum_{\ell=1}^{L} H_\ell t/\hbar) \neq \prod_{\ell=1}^{L} \exp(-iH_\ell t/\hbar)$

- **How then do you build a circuit for U ?**
  - Break evolution up into small increments

$$|\psi(t)\rangle = \underbrace{e^{-iH\Delta t/\hbar} e^{-iH\Delta t/\hbar} \cdots e^{-iH\Delta t/\hbar}}_{M \text{ factors}} |\psi(0)\rangle$$

$$= (\prod_{j=1}^{M} U(\Delta t))|\psi(0)\rangle \quad \text{where} \quad U(\Delta t) = e^{-iH\Delta t/\hbar} \approx \prod_{\ell=1}^{L} e^{-iH_\ell \Delta t/\hbar} + O((\Delta t)^2)$$

  - If each H_ℓ is local, there is an efficient quantum circuit for $\exp(-iH_\ell \Delta t/\hbar)$

Higher-Order Approximations

- **Trotter formula (e.g.,** $H = H_1 + H_2$**)**
  - –Basis of approximation rests on a limit

$$\lim_{n\to\infty} (e^{-iH_1 t/n} e^{-iH_2 t/n})^n = e^{-i(H_1+H_2)t}$$

- **Simplest Trotter approximation**

$$e^{-iH\Delta t} = e^{-iH_1\Delta t} e^{-iH_2\Delta t} + O((\Delta t)^2)$$

- **Higher-order Trotter approximations**

$$\begin{aligned} e^{-iH\Delta t} &= e^{-iH_1\frac{\Delta t}{2}} e^{-iH_2\Delta t} e^{-iH_1\frac{\Delta t}{2}} + O((\Delta t)^3) \\ &= e^{-iH_2\frac{\Delta t}{2}} e^{-iH_1\Delta t} e^{-iH_2\frac{\Delta t}{2}} + O((\Delta t)^3) \end{aligned}$$

Extracting Answers Efficiently from Quantum Simulations

Extracting Answers from Quantum Simulations

- **You cannot "read" the answer, i.e., the full solution $|\psi(t)\rangle$**
  - –It is stored in a quantum state
  - –If you try to read it, you collapse the state

- **What can you do with it?**
  - –You can sample from it
  - –You can repeat the simulation and perform state tomography to estimate $|\psi(t)\rangle$
  - –… but this is likely to be inefficient
  - –You can use it as an input to another quantum algorithm (the key!!)

- **If we feed $|\psi(t)\rangle$ into another quantum algorithm, we can …**
  - –Obtain mean values of operators efficiently
  - –Calculate correlation functions
  - –Measure (some) spectral properties
  - –Estimate a ground state eigenvalue / eigenvector

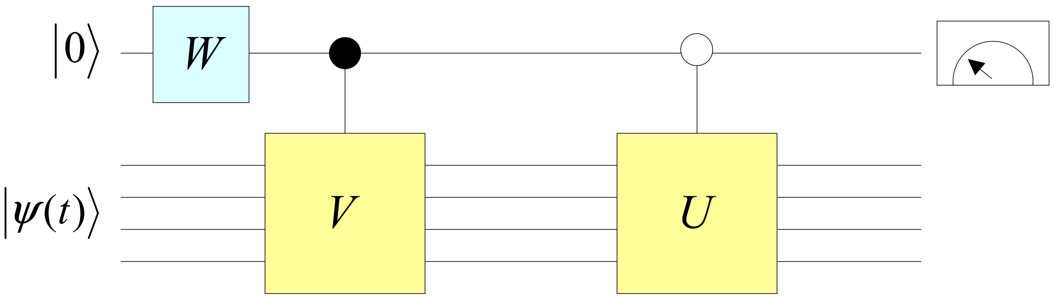
Ancilla-Assisted Measurements

- **Suppose we computed $|\psi(t)\rangle$ because we would like to estimate**

$$\langle\psi(t)|U^{+}V|\psi(t)\rangle = \langle U^{+}V\rangle$$

–Where U, V are unitary

- **With $2\sigma_{+} = \sigma_{x} + i\sigma_{y}$ the following circuit is found to measure $\langle U^{+}V\rangle$**



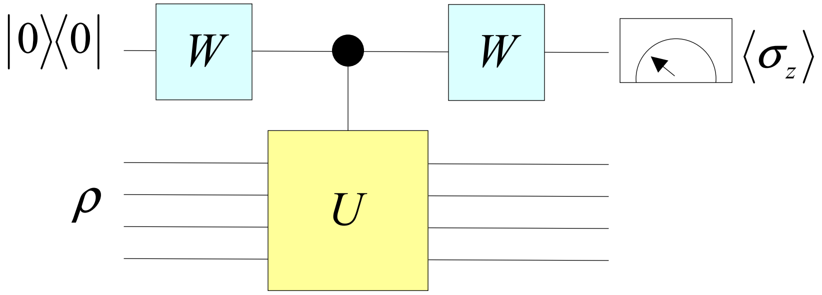
Measure expectation value of $2\sigma_{+}$

$$\langle 2\sigma_{+}\rangle = \langle\psi(t)|U^{+}V|\psi(t)\rangle$$

- **Only a single output qubit is monitored**
  - –Estimating the state of a single qubit can be done efficiently
  - –Then, if the Controlled-0-*U* and Controlled-1-*V* can be implemented efficiently …
  - –… the (polynomial cost) quantum simulation (needed to create the input to this circuit) need only be repeated polynomially many times

Tomography c.f. Spectroscopy

- **Circuits for measuring** $\mathrm{Re}[\mathrm{Tr}(\rho U)]$ **and** $\mathrm{Im}[\mathrm{Tr}(\rho U)]$ **for any unitary operator** U
  - *Find* $\langle\sigma_z\rangle = \mathrm{Re}[\mathrm{Tr}(\rho U)]$ *and* $\langle\sigma_x\rangle = -\mathrm{Im}[\mathrm{Tr}(\rho U)]$



Measure expectation value of σ_z

- **N.B. polarization measurement reveals a property that depends on both ρ and *U***
  - *Hence can use this circuit to extract information about ρ is U is known (tomography)*
  - *Or to extract information about U if ρ is known (spectroscopy)*

Quantum Algorithm VI

Eigenvalue Estimation

"…quantum computers of tens to hundreds of qubits can match and exceed the capabilities of classical full configuration interaction (FCI) calculations"

Prof. A. Aspuru-Guzik, Dept. Chemistry
UC Berkeley

the nanohelix perpendicular to its axial direction, has been measured by AFM with use of the force-displacement (*F-Z*) curve (*17*). Through the static compression of a nanohelix lying on a silicon substrate with an AFM tip (Fig. 4B) (fig. S5) and careful calibration of the sensitivity, *S*, of the photodetector (*11*, *18*), which is the inverse of the slope of the *F-Z* curve (fig. S6), the transverse spring constant of the nanohelix, K_{helix}, was found to be (*11*) 3.9, 4.6, 4.5, and 5.3 N/m for one to four turns, respectively, of the nanohelix as labeled in Fig. 4B. The measured K_{helix} is near the theoretically estimated transverse spring constant of 4.2 N/m (*11*), and so the elastic modulus, E, values of the nanohelix derived from the measured spring constant (*11*) are 42, 49, 48, and 57 GPa for the four turns, respectively, which agrees well with the elastic modulus measured for straight ZnO nanobelts by mechanical resonance with the use of in situ TEM (*19*).

The perfect helix we observed is of great interest not only for understanding the growth behavior of polar-surface–driven growth processes in the wurtzite system, such as GaN, AlN, and InN, but also for investigating fundamental physics and optical phenomena. The piezoelectric and semiconducting properties of ZnO suggest that the nanohelix could be a fundamental unit for investigating electromechanically coupled nanodevices by using the superlattice piezoelectric domains. The nanohelix is likely to have important applications in sensors, transducers, resonators, and photonics.

References and Notes

1. S. Amelinckx *et al.*, *Science* 265, 635 (1994).
2. R. P. Gao, Z. L. Wang, S. S. Fan, *J. Phys. Chem. B* 104, 1227 (2000).
3. H. F. Zhang, C. M. Wang, E. C. Buck, L. S. Wang, *Nano Lett.* 3, 577 (2003).
4. H. F. Zhang, C. M. Wang, L. S. Wang, *Nano Lett.* 2, 941 (2002).
5. O. G. Schmidt, K. Eberl, *Nature* 410, 168 (2001).
6. X. Y. Kong, Z. L. Wang, *Nano Lett.* 3, 1625 (2003).
7. X. Y. Kong, Y. Ding, R. Yang, Z. L. Wang, *Science* 303, 1348 (2004).
8. W. L. Hughes, Z. L. Wang, *J. Am. Chem. Soc.* 126, 6703 (2004).
9. Z. L. Wang *et al.*, *Adv. Funct. Mater.* 14, 944 (2004).
10. Z. W. Pan, Z. R. Dai, Z. L. Wang, *Science* 291, 1947 (2001).
11. See Materials and Methods provided on *Science* Online.
12. The {11̄22} are typical twin planes for ZnO (*20*).
13. P. X. Gao, Z. L. Wang, *Small*, in press; published online (http://dx.doi.org/10.1002/smll.200500165).
14. The growth of the nanostructure occurred at a local temperature of ~700° to 800°C in an argon atmosphere, under which the possibility of neutralizing the surface polar charges by adsorbing foreign molecules is unlikely, provided the growth occurred fairly quickly. This result is consistent with our previous studies [see (*7*–*9*)].
15. From the energy point of view, a polar-surface–dominated nanobelt tends to fold itself into a ring or spring for reducing electrostatic energy (*6*). The nanoring is stable if the ratio between nanobelt thickness, *t*, and the radius, *R*, of the nanoring is smaller than ~3% (*8*). Nanorings with *t*/*R* > 3% are energetically unfavorable to form.
16. The aforementioned data rule out the possibility that the nanohelix is formed by bilayer surface strain (*5*) for two reasons. First, there is no bilayer strained structure in our system; and second, the difference in surface stress on the (0001)-Zn and (0001̄)-O of ZnO, if any, has negligible effect on the formation of nanorings or nanobows (*8*). The nanohelix is likely to be formed by two processes: the rigid structural alteration due to the formation of the superlattice and the spontaneous polar-charge-induced bending (*6*). However, *t* of the nanobelt is ~20 nm and typical *R* of the nanohelix is ~150 to 350 nm (Fig. 1). Thus the *t*/*R* is ~6 to 13%, which is much larger than the threshold value (~3%) permitted for forming a nanoring by electrostatic polar charges (*15*). Also, considering the great reduction in area of the polar surfaces (for ~50%) after forming the superlattice as well as the helical shape of the structure, the role played by polar charges in forming a helix is dramatically reduced. Therefore, the dominant mechanism for forming the nanohelix is by rigid lattice rotation and twisting as a result of superlattice formation.
17. E. W. Wong, P. E. Sheehan, C. M. Lieber, *Science* 277, 1971 (1997).
18. A. Volodin *et al.*, *Phys. Rev. Lett.* 84, 3342 (2000).
19. X. D. Bai, P. X. Gao, Z. L. Wang, E. G. Wang, *Appl. Phys. Lett.* 82, 4806 (2003).
20. Y. Dai, Y. Zhang, Z. L. Wang, *Solid State Commun.* 126, 629 (2003).
21. Thanks to support from NSF, the NASA Vehicle Systems Program and Department of Defense Research and Engineering, the Defense Advanced Research Projects Agency, and the Chinese Academy of Sciences.

Supporting Online Material
www.sciencemag.org/cgi/content/full/309/5741/1700/DC1
Materials and Methods
Figs. S1 to S6

23 June 2005; accepted 2 August 2005
10.1126/science.1116495

Simulated Quantum Computation of Molecular Energies

Alán Aspuru-Guzik,[1*†] Anthony D. Dutoi,[1*] Peter J. Love,[2] Martin Head-Gordon[1,3]

The calculation time for the energy of atoms and molecules scales exponentially with system size on a classical computer but polynomially using quantum algorithms. We demonstrate that such algorithms can be applied to problems of chemical interest using modest numbers of quantum bits. Calculations of the water and lithium hydride molecular ground-state energies have been carried out on a quantum computer simulator using a recursive phase-estimation algorithm. The recursive algorithm reduces the number of quantum bits required for the readout register from about 20 to 4. Mappings of the molecular wave function to the quantum bits are described. An adiabatic method for the preparation of a good approximate ground-state wave function is described and demonstrated for a stretched hydrogen molecule. The number of quantum bits required scales linearly with the number of basis functions, and the number of gates required grows polynomially with the number of quantum bits.

Feynman observed that simulation of quantum systems might be easier on computers using quantum bits (qubits) (*1*). The subsequent development of quantum algorithms has made this observation concrete (*2*–*6*). On classical computers, resource requirements for complete simulation of the time-independent Schrödinger equation scale exponentially with the number of atoms in a molecule, limiting such full configuration interaction (FCI) calculations to diatomic and triatomic molecules (*7*). Computational quantum chemistry is therefore based on approximate methods that often succeed in predicting chemical properties for larger systems, but their level of accuracy varies with the nature of the species, making more complete methods desirable (*8*).

Could quantum computation offer a new way forward for exact methods? Despite the formal promise, it has not been demonstrated that quantum algorithms can compute quantities of chemical importance for real molecular systems to the requisite accuracy. We address this issue by classically simulating quantum computations of the FCI ground-state energies of two small molecules. Although the basis sets used are small, the energies are obtained to the precision necessary for chemistry. Absolute molecular energies must be computed to a precision (greater than six decimal places) that reflects the smaller energy differences observed in chemical reactions (~0.1 kcal/mol). These simulations show that quantum computers of tens to hundreds of qubits can match and exceed the capabilities of classical FCI calculations.

A molecular ground-state energy is the lowest eigenvalue of a time-independent Schrödinger equation. The phase-estimation algorithm (PEA) of Abrams and Lloyd (*3*, *4*) can

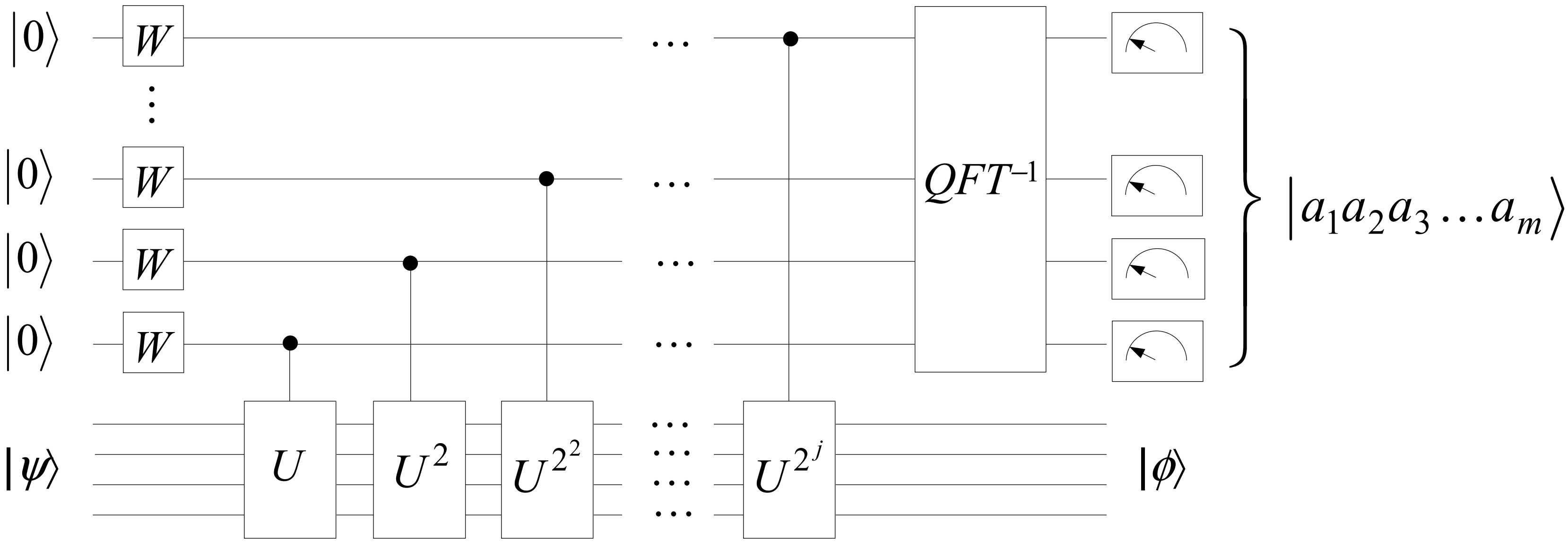
[1] Department of Chemistry, University of California Berkeley, CA, USA. [2] D-Wave Systems, Inc., 4401 Still Creek Drive, Suite 100, Burnaby, BC V5C 6G9, Canada. [3] Chemical Sciences Division, Lawrence Berkeley National Laboratory, Berkeley, CA 94720, USA.

*These authors contributed equally to this work.

†To whom correspondence should be addressed. E-mail: alan@aspuru.com

Quantum Chemistry: Finding Eigenvalues

- **Problem: Given a Hamiltonian, H, and a state, $|\psi\rangle$, which approximates a true eigenstate, $|\phi\rangle$**
- **Goal: Determine the corresponding eigenvalue,** $e^{2\pi i\phi} : 0 \le \phi \le 1$
- **Some observations:**
  - –First notice an eigenvector of H is also an eigenvector of $U = e^{-iHt/\hbar}$
  - –If H is "local", U (and powers of U) can be implemented efficiently
- **Let** $a/2^m = 0.a_1a_2\ldots a_m$ **(in binary) is the best *m*-bit estimate of ϕ**
  - *–Then the following circuit computes a_1, a_2, ..., a_m with probability 0.405*



How Does Eigenvalue Estimation Work?

- **Hamiltonian, H, and a state, $|\psi\rangle$, which approximates a true eigenstate, $|\phi_u\rangle$**
- **Goal: Determine the corresponding eigenvalue, a_u**

Algorithm EigenvalueEstimation:

Step 1: Initialize state to $|0\rangle|\psi\rangle = |0\rangle\sum_u d_u|\phi_u\rangle$ where $|\phi_u\rangle$ are eigenvectors of *U*

Step 2: Apply Walsh-Hadamard's to the *b* ancillae to create $\frac{1}{\sqrt{2^b}}\sum_{j=0}^{2^b-1}|j\rangle\sum_u d_u|\phi_u\rangle$

Step 3: Apply powers of *U*, conditioned on ancillae $\frac{1}{\sqrt{2^b}}\sum_{j=0}^{2^b-1}|j\rangle U^j\sum_u d_u|\phi_u\rangle$

Step 4: Since *U* unitary its eigenvalues can be written as $e^{2\pi i a_u}$ where $a_u \in \Re$

Step 5: Re-write last state using eigenvalues and change order of summation

$$\frac{1}{\sqrt{2^b}}\sum_u\sum_{j=0}^{2^b-1} d_u e^{2\pi i j a_u}|j\rangle|\phi_u\rangle$$

Step 6: Perform an inverse QFT in the ancillae $\sum_u d_u\left(\sum_{j=0}^{2^b-1} g(a_u, j)|j\rangle\right)|\phi_u\rangle$

where $g(a_u, j) = \begin{cases} \dfrac{\sin(\pi(2^b a_u - j))e^{\pi i(a_u - j2^{-b})(2^b-1)}}{2^b\sin(\pi(a_u - j2^{-b}))} & 2^b a_u \neq j \\ 1 & 2^b a_u = j \end{cases}$

Step 7: Measure the ancillae qubits to obtain outcome *j* with probability

$p_j = \sum_u |d_u|^2 |g(a_u, j)|^2$ and project second register into state $\sum_u \frac{d_u g(a_u, j)}{\sqrt{p_j}}|\phi_u\rangle$

Summary

- **Quantum computing allows new kinds of algorithms that cannot be run as efficiently on conventional HPCs**
- **Some problems can be solved exponentially faster on QCs**
  - –Factoring integers, quantum simulation, sampling from many of the "standard" unitary transforms (DFT, DCT, DWT, Hartley, Fractional Fourier etc)
- **Some can be solved polynomially faster on QCs**
  - –NP-Complete problems
- **Some problems cannot be sped up at all**
- **With just 50 qubits can simulate physical systems beyond the reach of current supercomputers**

Conclusions

- **QCs will never replace *all* applications of HPCs**
  - –Why? QCs do not speed up all computations
  - –Even if the computation can be sped up sometimes a user might require “proof” (computational history) as well as “truth”
- **Nevertheless, QCs could have niche roles …**
  - –Code-breaking: PKI & elliptic-curve (exponential); DES/AES (polynomial)
  - –Certain simulations of quantum systems (e.g., aspects of quantum chemistry)
  - –Certain signal, image and data processing tasks
    - Need (Quantum Encoding + Quantum Processing + Readout)(×#Repeats) << Classical Processing
  - –Deciding certain mathematical propositions
- **U.S. should intensify efforts to find “significant” quantum algorithms**
  - –QCs only route to *fundamental* (qualitative) algorithmic advances
  - –Efficiency unmatchable by any classical HPC (current or future)
- **Impact**
  - – Commercially significant applications of QCs will engage U.S. industry, leverage government investments, accelerate hardware development, improve quality, and enhance competitiveness
- **Need *applications* teams (focus to date has been hardware teams)**
  - –Need to partner quantum computer scientists with domain experts
    - Quantum computer scientists don’t understand needs of user communities
    - User communities don’t appreciate what is an isn’t feasible with QCs
  - –Proper teaming is effective (e.g., D-Wave’s partnership with quantum chemists at UC Berkeley)
- **Don’t need large QCs to start to see an impact!**

Quantum Algorithm I

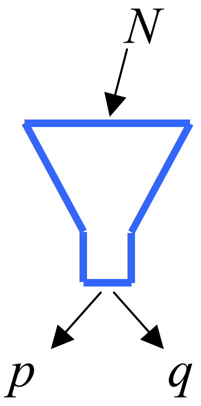
Shor's Algorithm for Factoring Composite Integers

Factoring Integers

- **Multiplication easy** $p \times q = N$
- **Factoring hard** $N \to p, q$

***N* = 114381625757888867669235779976146612010218296721242362562561842 9…**

…35706935245733897830597123563958705058989075147599290026879543541

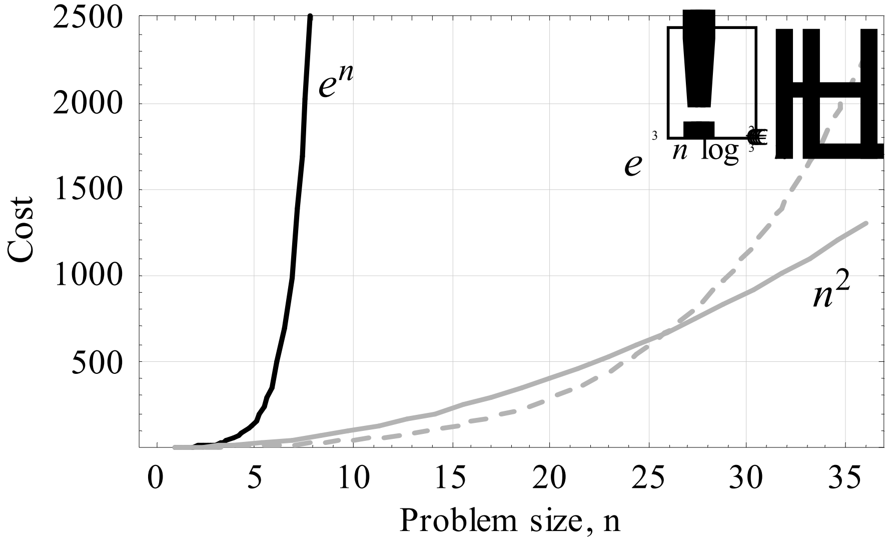


***p* = 32769132993266709549961988190834461413177642967992942539798288533**

***q* = 3490529510847650949147849619903898133417764638493387843990820577**

Complexity of Factoring Integers

- **Number Field Sieve** $O(e^{n^{1/3}(\log n)^{2/3}})$ **sub-exponential (hard!)**



- **Why does anyone care?**
- **Security of widely used public key cryptosystems rests on the presumption that factoring is hard, e.g., RSA**

RSA Public Key Cryptosystem

Create Keys	1. Find two primes, and compute their product $N = p\, q$ 2. Find integer d coprime to $(p\text{-}1)(q\text{-}1)$ 3. Compute e from $e\, d = 1 \bmod (p\text{-}1)(q\text{-}1)$ 4. Broadcast public key (e,N) , keep private key (d,N) secret
Encrypt	5. Represent message P as a sequence of integers $\{M_i\}$ 6. Encrypt M_i using public key and rule $E_i = M_i^e \bmod N$
Decrypt	7. Decrypt using private key and rule $M_i = E_i^d \bmod N$ 8. Reconvert the $\{M_i\}$ back to the plaintext P

- As public key (e, N) known, can crack RSA if you can factor N into $N = p\, q$ by computing private key (d, N) from $e\, d = 1 \bmod (p\text{-}1)(q\text{-}1)$

Factoring via Period Finding

- Can factor integers by finding period of a function related to the factors

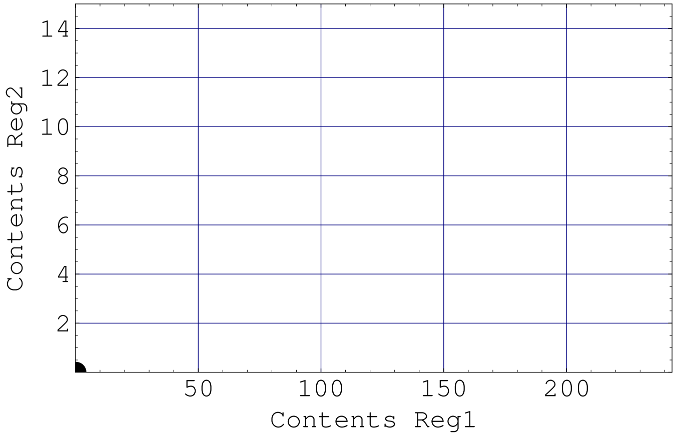
- Classical (*inefficient*) algorithm

- Example: factor $N = 15$
  - Choose random integer x that is coprime to N
    - e.g. $x = 2$ will suffice because gcd(2, 15) = 1
  - Compute the sequence of integers x^i mod N, giving:
    - 2^0 mod 15, 2^1 mod 15, … = 1, 2, 4, 8, 1, 2, 4, 8, 1, 2, 4, 8 …
  - Sequence is periodic, with period $r = 4$
  - Factors of N given by gcd($x^{r/2} \pm 1$, N)
  - Gives $15 = p\ q$ where p = gcd(5,15) = 5, q = gcd(3,15) = 3

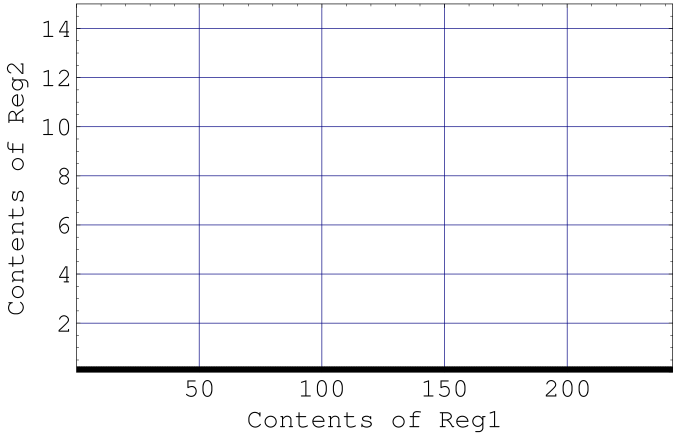
- But there is a fast quantum algorithm for period finding
  - Based on sampling from Fourier transform of this periodic sequence

Quantum Factoring I: Periodic State

Initialize Reg1 & Reg2 as ¨0,0>

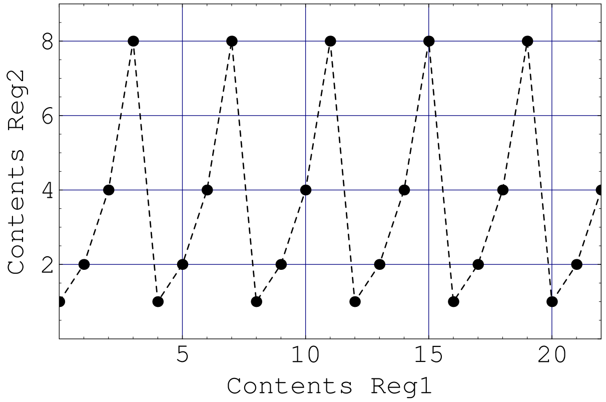


Load Reg1 with 1 Sqrt@qD Sum@¨a,0>, 8a,0,q-1<D

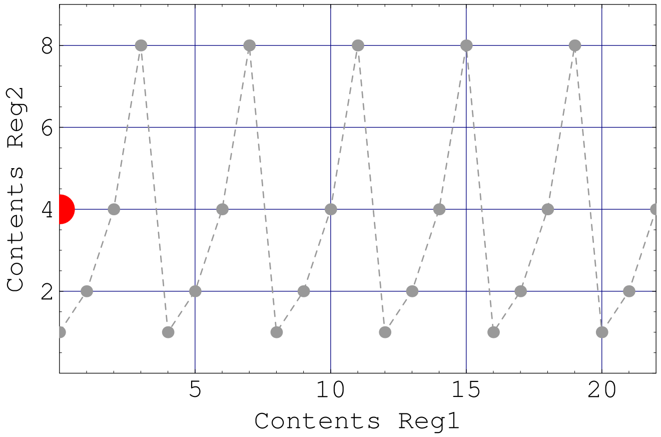


Put superposition x^a mod n in Reg2

1 Sqrt@qD Sum@¨a, x^a mod n>,8a,0,q-1<D

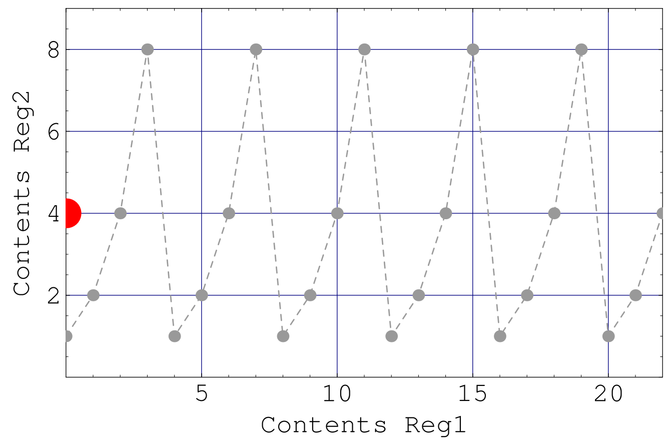


Measure Reg2 = 4

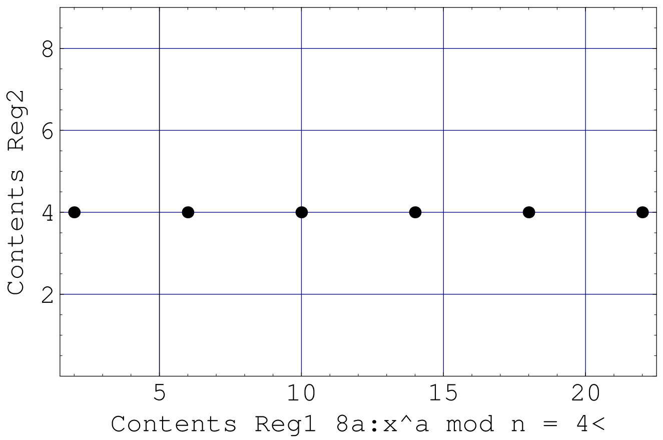


Quantum Factoring II: Find Period

Measure Reg2 = 4

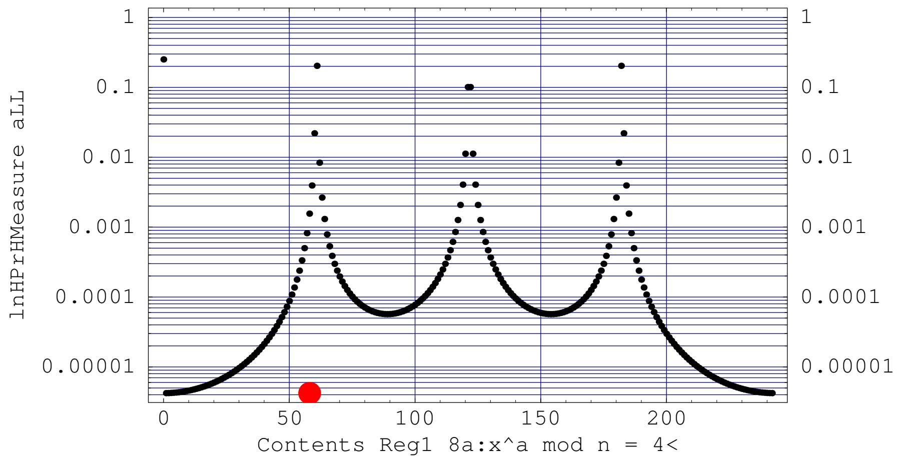


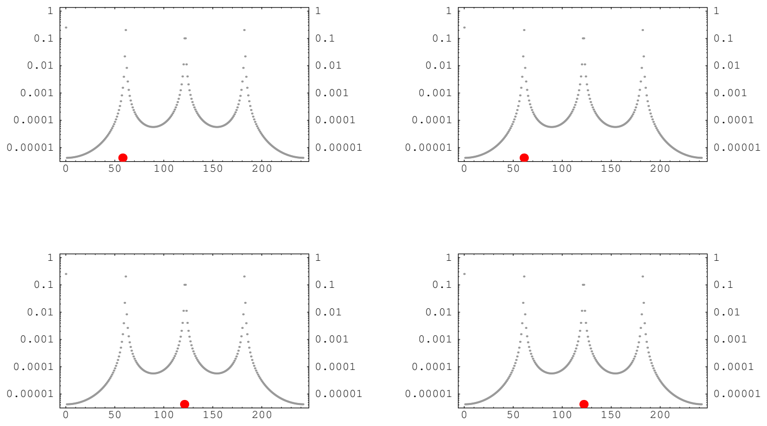
Project Reg1: 8a:x^a mod n = 4<



Repeat Shors Algm OHlnHqLL times.
Obtain samples from DFT in Reg1

Compute Discrete Fourier Transform of Reg1





Shor's Algorithm

Shor's Algorithm for Factoring a Composite Integer n

Step 1. Pick q such that $2n^2 \le q \le 3n^2$

Step 2. Pick random x s.t. gcd(x, n) = 1

Step 3. Repeat steps (a)-(g) O(log q) times using the same value of x each time

(a) Initialize two registers, *A* and *B*, to the state $|0,0\rangle$

(b) Load *A* with all integers in range 0 to q–1, $|\psi\rangle = \frac{1}{\sqrt{q}}\sum_{a=0}^{q-1}|a,0\rangle$

(c) Apply $U_{f_{n,x}} : |a,0\rangle \xrightarrow{U_{f_{n,x}}} |a, x^a (\bmod n)\rangle$ to create $|\psi\rangle = \frac{1}{\sqrt{q}}\sum_{a=0}^{q-1}|a, x^a (\bmod n)\rangle$

(d) Read *B*, obtaining the result $|k\rangle$. The state of the registers becomes $|\psi\rangle = \frac{1}{\sqrt{\|A\|}}\sum_{a' \in A}|a',k\rangle$

(e) Apply QFT to *A* $|a'\rangle \xrightarrow{QFT} \frac{1}{\sqrt{q}}\sum_{c=0}^{q-1} e^{2\pi i a' c}|c\rangle$

(f) Read *A*, obtaining a sample, λ from the QFT. λ will be an integer that is close to some integer multiple, *j*, of *q*/*r* where *r* is the desired period

(g) Estimate *j* by computing the continued fraction expansion of λ/*q*

Step 4. By repeating Steps (a)-(g) we obtain a set of samples of multiples of 1/*r*. Hence determine *r*.

Step 5. With *r* known, the factors of n can be obtained from:

$$\text{factor1} = \gcd(x^{r/2-1}, n) \qquad \text{factor2} = \gcd(x^{r/2+1}, n)$$