Quantum Computing

Colin P. Williams

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

Sponsored by National Security Agency (NSA) and Advanced Research and Development Activity (ARDA)

Overview

- **NASA / JPL's interest in Quantum Technologies**
  - Quantum Computers
    - Faster solution of certain hard computational problems
    - Unmatchable by any conventional computer
  - Quantum Communications
    - Superdense information compression
    - Securing command & control of orbital assets
  - Quantum Sensors
    - Gyroscopes / Accelerometers / Magnetometers
    - Gravity Gradiometers (underground sensing)
    - Gravity Wave Detectors
  - Quantum Lithography

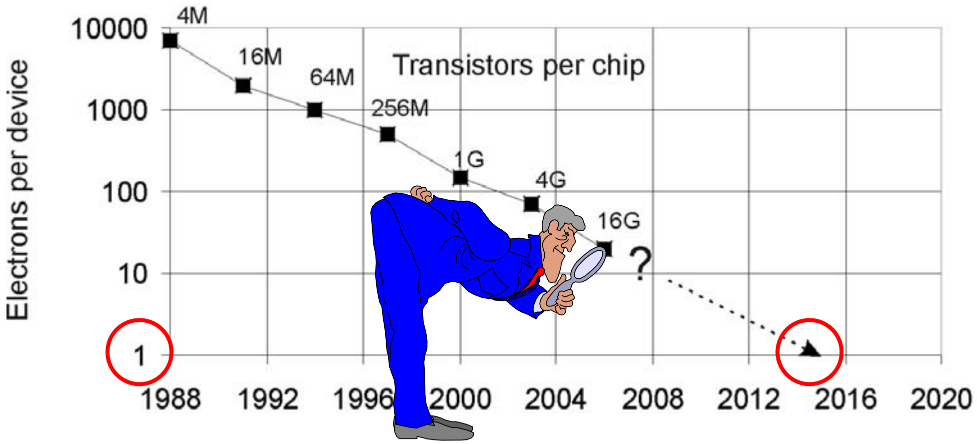
- **In this Talk …**
  - What are Quantum Computers?
  - Why are they Interesting?
  - State-of-the-Art Quantum Computing Hardware at JPL
    - Automated Quantum Computer Circuit Design
    - Superconducting and Linear Optics Quantum Computing Hardware
  - Spin-off technologies from Quantum Computing

Overview

- **What is a quantum computer?**
  - –From bits to qubits
  - –Quantum memory registers
  - –Quantum computation
- **What can you do with a quantum computer?**
  - –Quantum algorithms
- **How do you make a quantum computer?**
  - –Quantum algorithms to quantum circuit designs
  - –Quantum circuits designs to quantum hardware
- **JPL interest in quantum computing**
  - –NASA-relevant quantum algorithms
  - –Spin-off quantum technologies

Miniaturization Trend

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



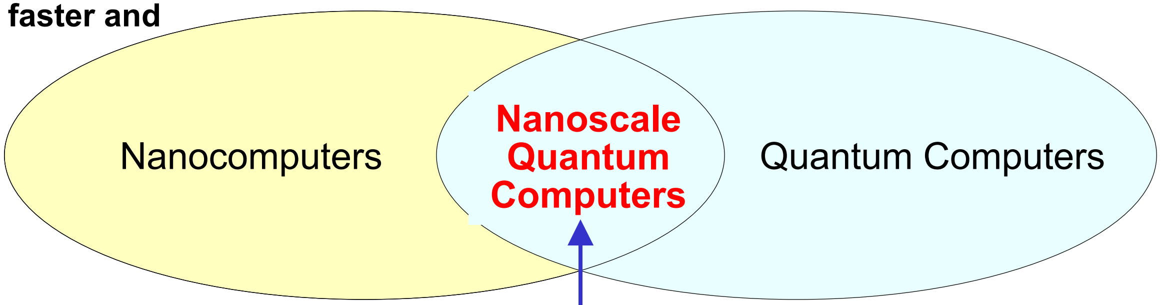
Miniaturization of computer technology as a function of time

- **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

Nanotechnology c.f. Quantum Technology

- Nanocomputers compared with quantum computers



More efficient than conventional computers on some problems but need not be “small” at all e.g., NMR quantum computers

Equivalent to conventional computers, but faster and more compact

Most interesting class: faster, smaller, more energy efficient, and algorithmically superior to conventional computers

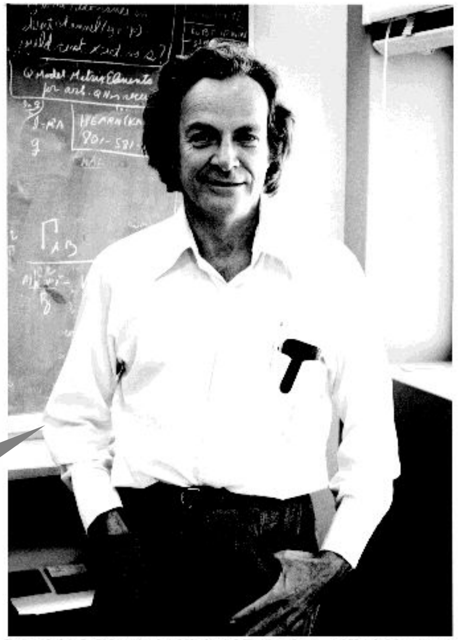
- Use nanofabrication techniques to assemble quantum computing hardware

At Quantum Level Commonsense Fails

- **Theory of computation harbors implicit assumptions**
  - which cease to be true at quantum scales

- **What are these assumptions?**
  - Bit always has a value
  - This value is 0 or 1
  - Bit can be copied without error
  - Reading a bit does not change it
  - Reading a bit has no affect on other (unread) bits

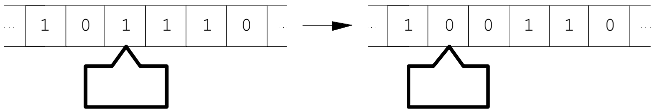
- **For qubits, each assumption can fail**



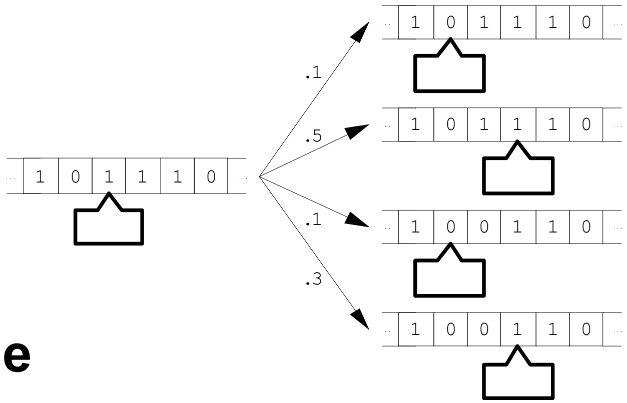
"Because nature isn't classical dammit!"
Richard Feynman

Fundamental Shift in Foundations

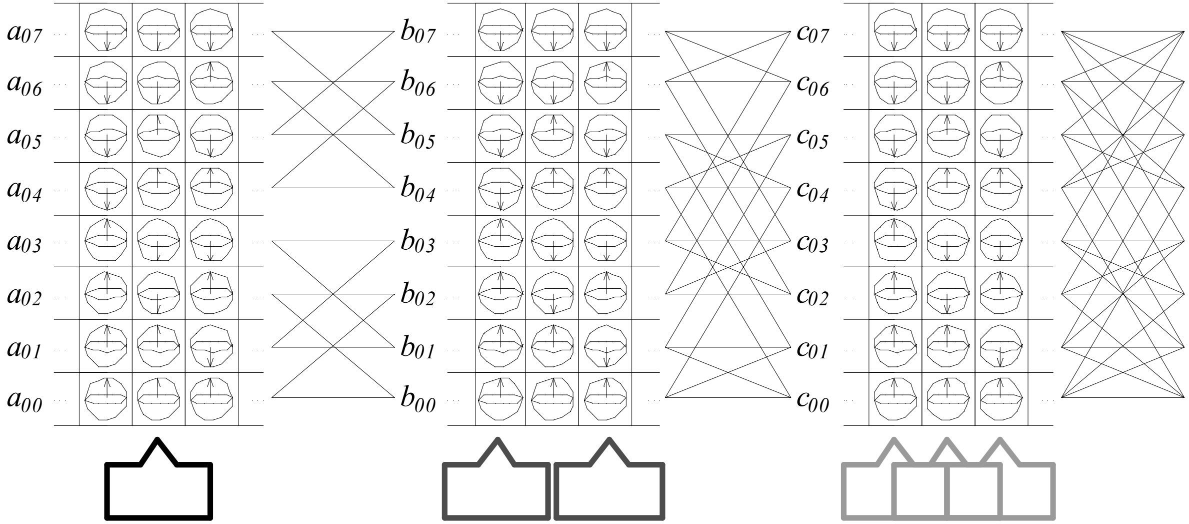
Turing Machine



Probabilistic Turing Machine



- … becomes Quantum Turing Machine

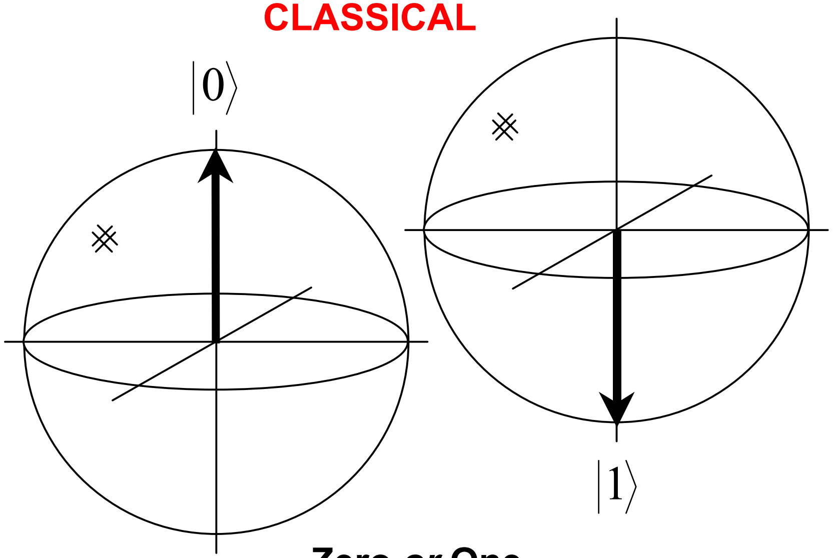


- All computational paths pursued simultaneously

From Bits to Qubits

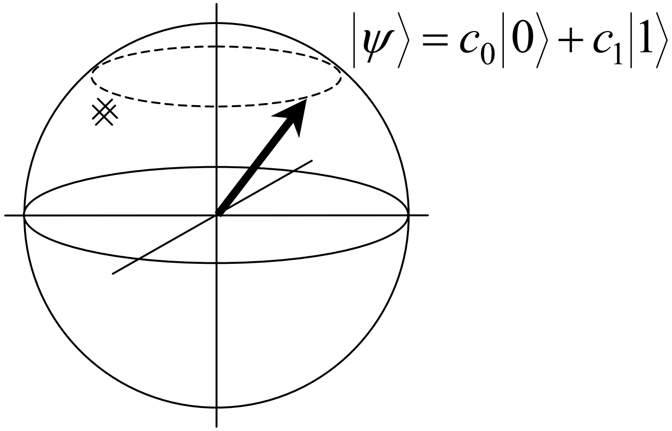
- Use 2-state quantum systems for bits (0s and 1s) e.g. spins, polarized photons, atomic energy levels

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
- Can only read 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.,**

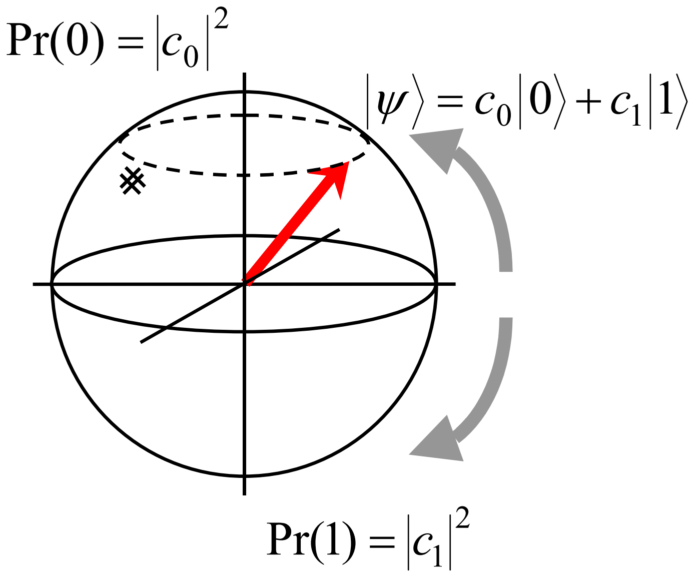
$$\frac{1}{\sqrt{2}}\left(|0\rangle_A|0\rangle_B + |1\rangle_A|1\rangle_B\right) \neq |\psi\rangle_A|\phi\rangle_B$$

- **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.**

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

- **with solution** $|\psi(t)\rangle = \exp(-iHt/\hbar)|\psi(0)\rangle = U|\psi(0)\rangle$
- **Make connection to computation:**

$$|\psi(0)\rangle \leftrightarrow \text{input data}$$

$$U \leftrightarrow \text{algorithm}$$

$$|\psi(t)\rangle \leftrightarrow \text{output before measurement}$$

$$|00\ldots0\rangle \text{ or } |00\ldots1\rangle \text{ or } \cdots \text{ or } |11\ldots1\rangle \leftrightarrow \text{output after measurement}$$

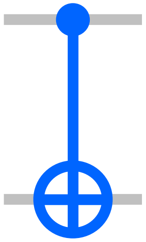
Algorithm: Specification of a sequence of unitary transformations to apply to an input quantum state, followed by a measurement

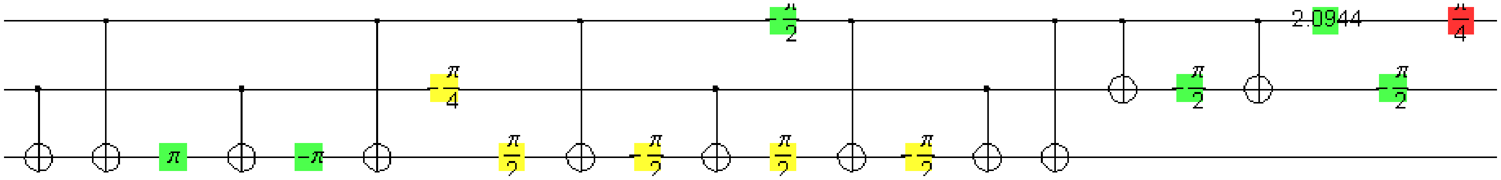
Quantum Circuits

- **Quantum circuit is a decomposition of desired unitary matrix into sequence of single and pairwise quantum logic gates**
- **Only requires**
  - *y*-rotations, *z*-rotations, phase-shifts, and controlled-NOT gates (CNOT)

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

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





What Makes Quantum Computers So Interesting?

- **QCs take fewer steps than classical computers**
  - Not technological (faster chip) advantage
  - But complexity (fewer steps) advantage
  - Unmatchable by any classical computer
  - Potential breakthrough in solving hard computational problems

- **QCs are reversible computers**
  - Potentially energy efficient
  - Energy expended in computation is recoverable

- **QCs perform tasks that no classical computer can do**
  - Quantum teleportation
  - Utterly secure communication
  - Simulations of physical systems too complex to describe exactly / explicitly

Quantum Algorithms

Quantum Algorithms

- **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 (Fijany / Williams)
  - –Hitting Times for Quantum Random Walks (Ambainis/Childs/Farhi/Gutmann)
  - –Solving Pell's Equation (Hallgren)

- **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)
  - –Counting Number of Solutions (Brassard/Hoyer/Tapp)
  - –Evaluating High-dimensional Numerical Integrals (Abrams / Williams)
  - –Template Matching (Jozsa)

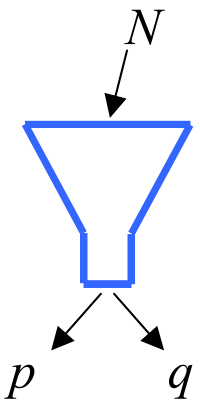
Quantum Algorithm for Factoring Integers

Factoring Integers

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

***N* = 114381625757888867669235779976146612010218296721242362562561842 9…**

…35706935245733897830597123563958705058989075147599290026879543541



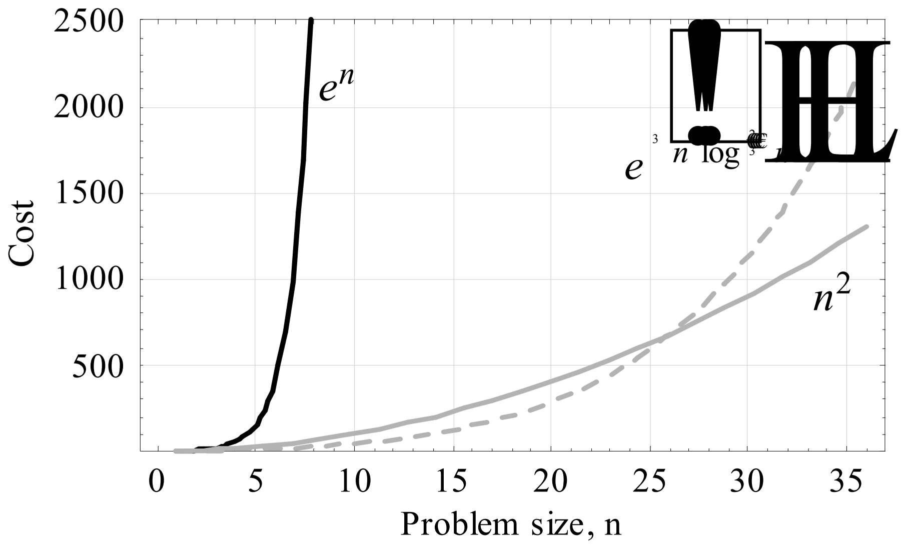
***p* = 3276913299326670954996198819083446141317764296799294253979828853 3**

***q* = 3490529510847650949147849619903898133417764638493387843990820577**

© Colin P. Williams 2004

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**

© Colin P. Williams 2004

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 $N = p\,q$**
    - *… because can then find private key, (d, N), from e d = 1 mod (p-1)(q-1)*
- **So fast-factoring would make most current e-commerce transactions vulnerable to eavesdropping / fraud**

Example of RSA

```
In[29]:= {$PublicKey, $PrivateKey} = CreatePublicKeyAndPrivateKey[20];

         Picking p: p = 3097172369

         Picking q: q = 3782480549

         Hence n = p q = 11714994242642750581

         Picking large integer d, co-prime to n: d = 7520374751991265811

         Computing modular inverse, e, from e d = 1 mod (p-1)(q-1): e = 9871244581433966043

         Public  Key is {e, n} = {9871244581433966043, 11714994242642750581}

         Private Key is {d, n} = {7520374751991265811, 11714994242642750581}

In[30]:= cipherText = EncryptRSA["I'm hungry. Let's eat! ", $PublicKey]

Out[30]= {6337662632885750605, 4223282963866241971, 8515734954729530610,
          5721050265798001277, 3125477641371647366, 8785778425474049423, 1160959880272455170,
          1843196734898219670, 4095890900271762030, 5711708545539327862, 5188837378111696662}

In[31]:= DecryptRSA[cipherText, $PrivateKey]

Out[31]= I'm hungry. Let's eat!
```

© Colin P. Williams 2004

Factoring via Period Finding

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

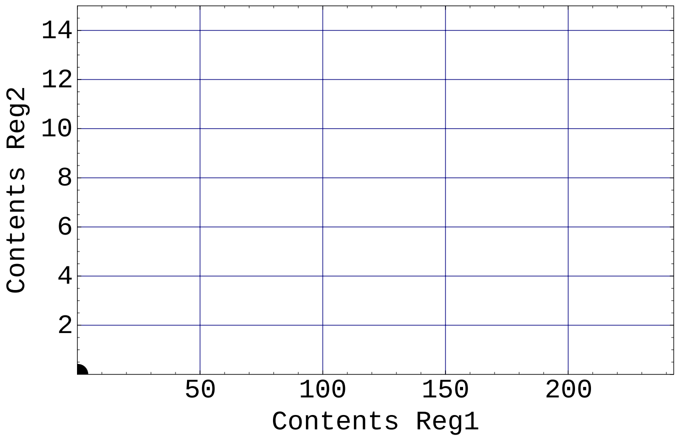
- ❑ **Classical (*inefficient*) algorithm**

- ❑ **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, … = $\boxed{1, 2, 4, 8,}$ 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*

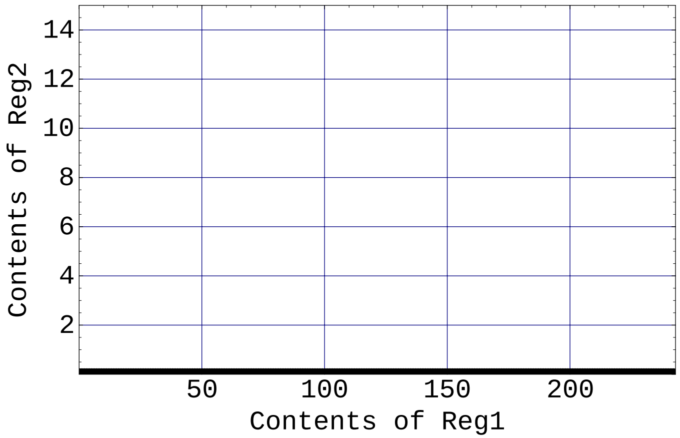
- ❑ **But there is a fast quantum algorithm for period finding**
  - – *Based on **sampling** from Fourier transform of this periodic sequence*

Quantum Factoring I: Periodic State

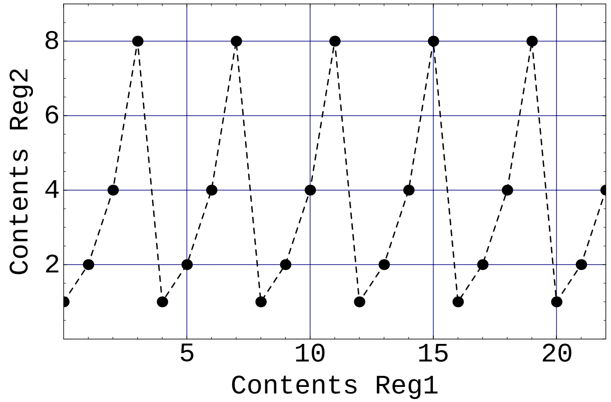
Initialize Reg1 & Reg2 as ¨0,0>



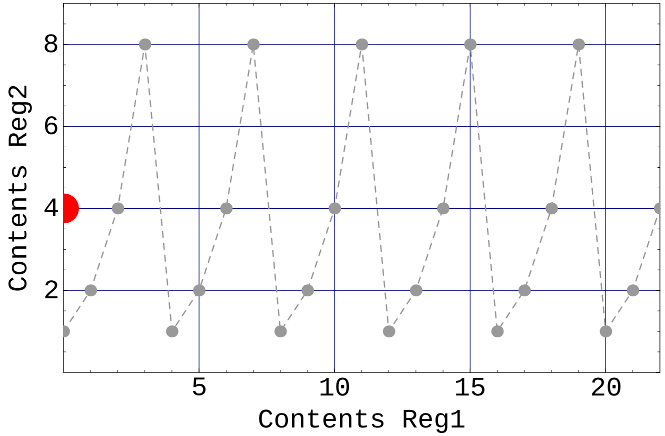
Load Reg1 with 1 Sqrt@qD Sum@¨a,0>, 8a,0,q-1<D



Put superposition x^a mod n in Reg2
1 Sqrt@qD Sum@¨a, x^a mod n>,8a,0,q-1<D

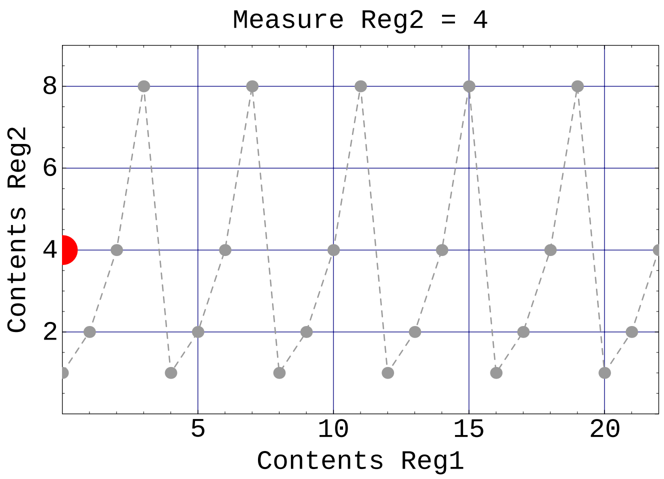


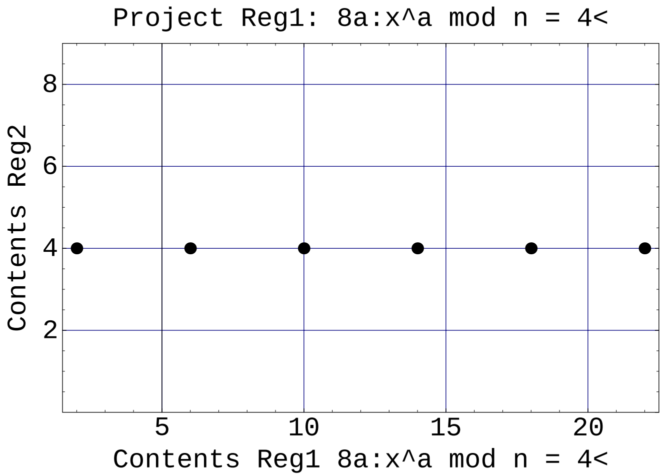
Measure Reg2 = 4



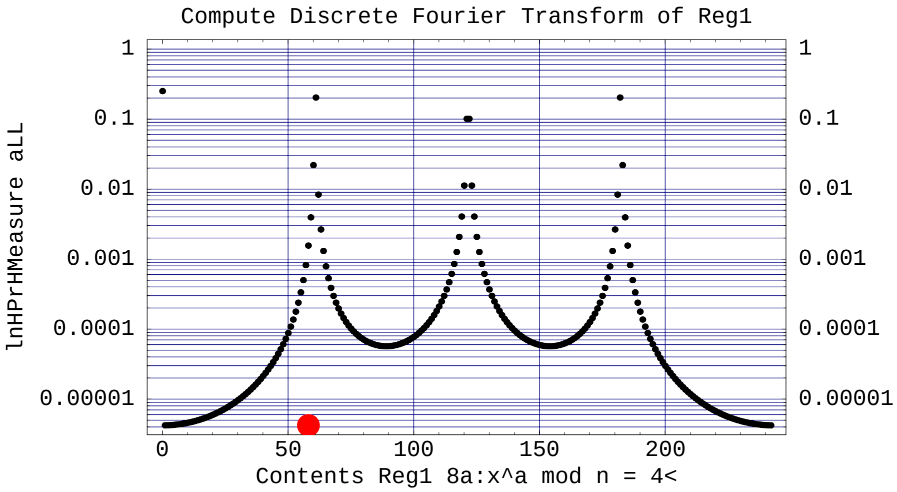
© Colin P. Williams 2004

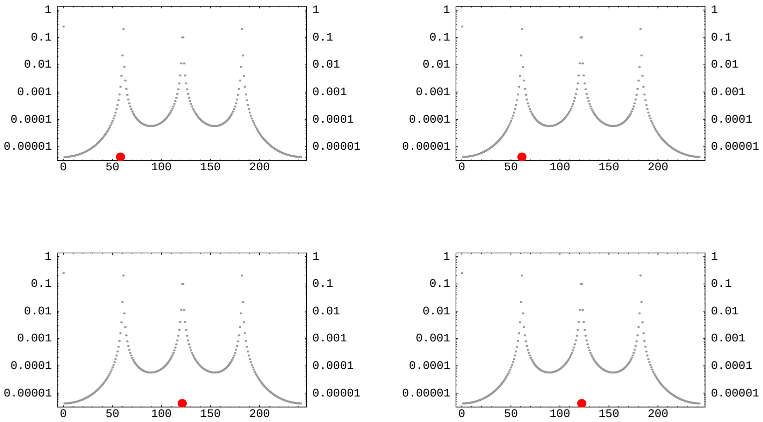
Quantum Factoring II: Find Period





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

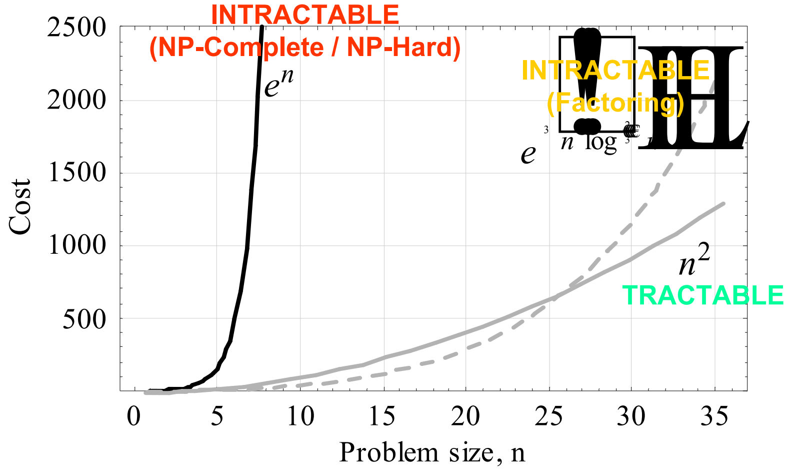




Quantum Algorithm for Solving NP-Complete Problems

NASA-Relevant Computations

- **Autonomy relies on solving NP-Complete/NP-Hard problems**
  - –Diagnosis
  - –Planning
  - –Scheduling
  - –Combinatorial Optimization
  - –Learning
  - –Constraint Satisfaction
  - –etc …



Solving one type of NP-Hard problem efficiently would solve ALL types of NP-Hard problems efficiently as you can easily interconvert them

- **Image Interpretation**
  - –Change detection
  - –Superresolution
  - –Pattern recognition
- **Can’t tame NP-Hard problems with conventional computers**
- **But quantum computers can speed up computations by:**
  - –Exponential factor,
  - –Polynomial factor, or
  - –Not at all
  - –So possibility exists for fundamental ***algorithmic*** advance

Example: 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 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**

How Quantum Search Works

$$f_t(x) = \begin{cases} 0 & x \neq t \\ 1 & x = t \end{cases}$$

- **Knowledge of database encoded in an “oracle” function**
  - *x* is the index of an item in the database
  - Target entry has index *x* = *t*
  - Oracle returns $f_t(t) = 1$, $f_t(x) = 0$ otherwise

- **Use “oracle” to build an “amplitude amplification operator”, Q**

$$\hat{Q} = -\hat{U} \cdot \hat{I}_s \cdot \hat{U}^{-1} \cdot \hat{I}_{f_t}$$

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

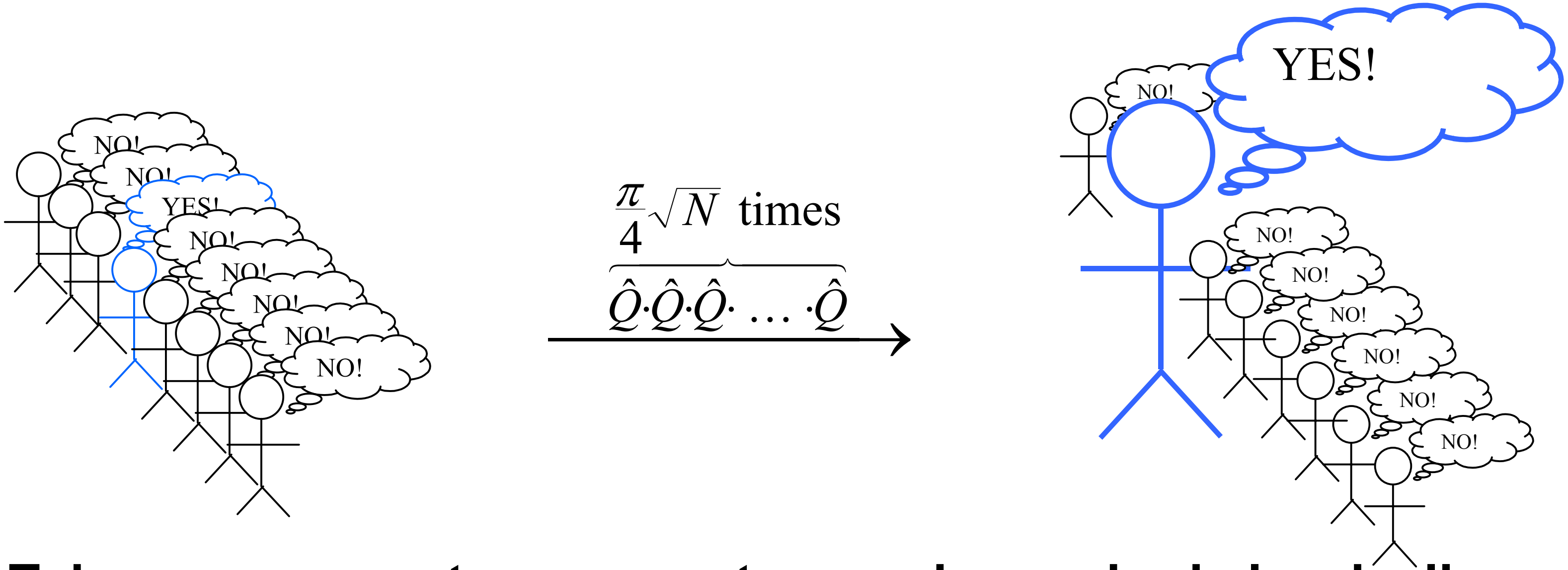
Amplitude Amplification Boosts "Signal"

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**

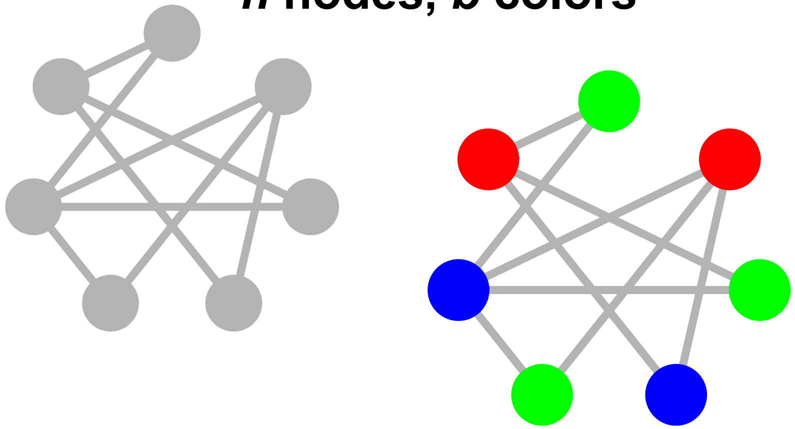
Step 4: Read register – will obtain target index with probability *O*(1)


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**

What about the NP-Hard Problems?

n nodes, *b* colors



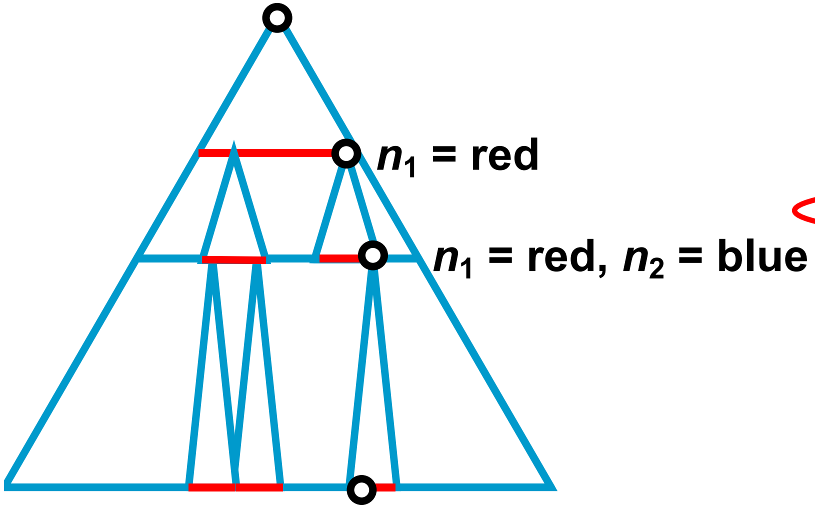
Nested Quantum Search

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

Induces tree-structured search space



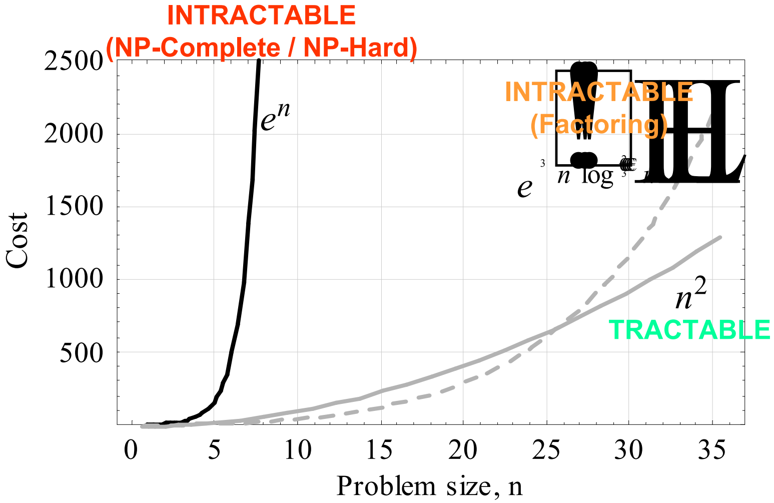
Comparison

- **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).

An Alternative Approach : the Quantum Adiabatic Algorithm

- 3-SAT: Given *n* Boolean variables, $x_1, x_2, \ldots, x_n$, find an assignment of True or False to each one that makes a sentence, like the following, True:

$$\underbrace{(x_1 \vee \neg x_3 \vee x_4) \wedge (\neg x_2 \vee x_3 \vee x_4) \wedge \ldots \wedge (x_1 \vee x_5 \vee x_6)}_{n \text{ variables}, \quad m \text{ clauses}, \quad 3 \text{ variables per clause}}$$



QUANTUM ADIABATIC ALGORITHM

- Encode 3-SAT problem instance to be solved in a Hamiltonian, H_1, s.t. its solution is the ground state of H_1
- Start system off in the ground state of some other (easy to arrange) Hamiltonian H_0
- Slowly change the system, in T increments, so that at time *t*, its instantaneous Hamiltonian, $H(t/T)$ is a weighted combination of H_0 and H_1, i.e.

$$H(\tfrac{t}{T}) = (1 - \tfrac{t}{T})H_0 + \tfrac{t}{T}H_1$$

- At time $t = T$, measure the system
- If you go slowly enough, i.e., “adiabatically”, Adiabatic Theorem says you should end up in the ground state of H_1 (and hence solve problem)

ADIABATIC THEOREM

- If smallest gap between ground state and first excited state is $g_{\min} = \min_{0 \le t \le T} [E_1(t) - E_0(t)]$
- Matrix element between corresponding eigenstates is

$$\left\langle \frac{dH}{dt} \right\rangle_{1,0} = \langle E_1; t | \frac{dH}{dt} | E_0; t \rangle$$

- Then overlap between final (actual) state and desired (ground) state will be $\left| \langle E_0; T | \psi(T) \rangle \right|^2 \ge 1 - \varepsilon^2$
- Provided $\dfrac{\left| \left\langle \frac{dH}{dt} \right\rangle_{1,0} \right|}{g_{\min}^2} \le \varepsilon$

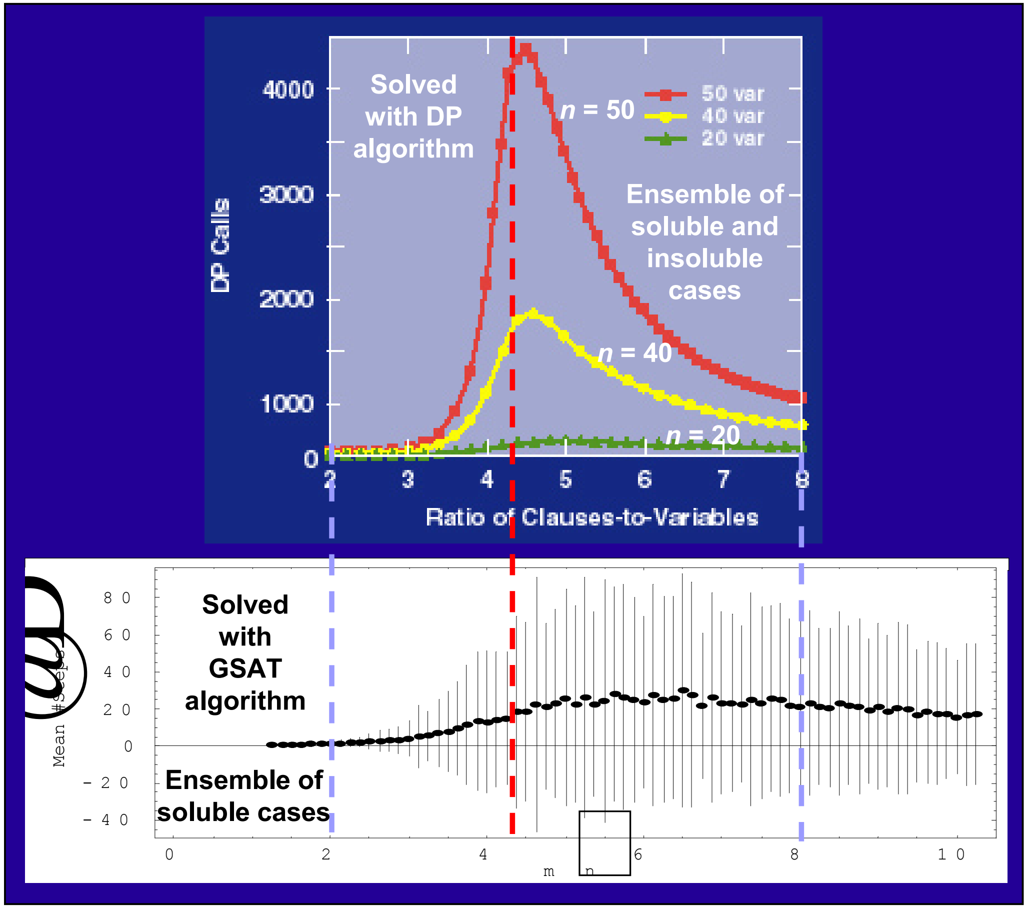
How Does Cost of Adiabatic Algorithm Scale with Problem Size?

- **What is known (analytically)?**
  - Early numerical studies hinted at a polynomial scaling ✓
  - Farhi proves scaling is polynomial for “easy” problems ✓
  - Ruskai proves minimum gap is non-zero ✓
  - van Dam, Mosca, **Vazirani** exhibit problem for which scaling is provably exponential ✘
  - Farhi et al. circumvent such instances by choosing a different interpolation path ✓
  - Roland and Cerf nest one adiabatic algorithm within another to achieve an adiabatic solution of an NP-Complete problem that is faster than adiabatic version of Grover’s algorithm on that problem ✓

- **True scaling (for hard problems) is unknown analytically**
  - Can it be estimated reliably by extrapolating simulations?

Beware of Extrapolations from Small Scale Simulations

- Numeric scaling prediction based on extrapolation from *n* = 10, 15, 20 variable instances of 3-SAT
- From classical computer science we know such scaling is not very reliable
- Questionable to assess scaling from small-scale (*n* < 50) numerical simulations
- Quantum adiabatic algorithm for *n* = 50 is well beyond what we can simulate classically



- Need an *analytic* model of scaling of the quantum adiabatic algorithm

Mapping
Quantum Algorithms into Quantum Circuits

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:**

$$\left|\psi(0)\right\rangle \leftrightarrow \text{input data}$$

$$U \leftrightarrow \text{algorithm}$$

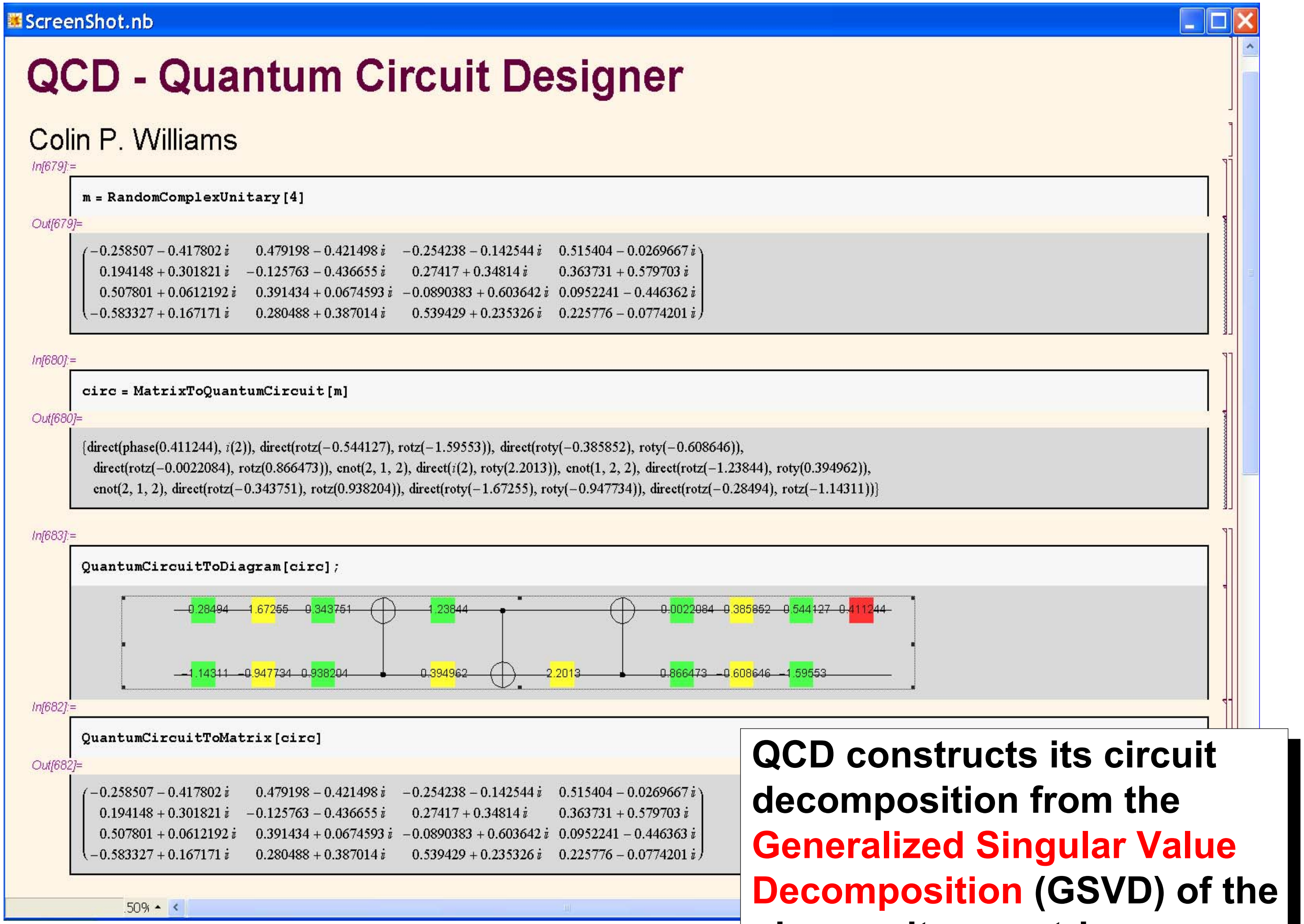
$$\left|\psi(t)\right\rangle \leftrightarrow \text{output before measurement}$$

$$\left|00\ldots0\right\rangle \text{ or } \left|00\ldots1\right\rangle \text{ or } \cdots \text{ or } \left|11\ldots1\right\rangle \leftrightarrow \text{output after measurement}$$

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

QCD: Quantum Circuit Design Tool

- QCD: *Mathematica*-based circuit design tool



ScreenShot.nb

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]:=

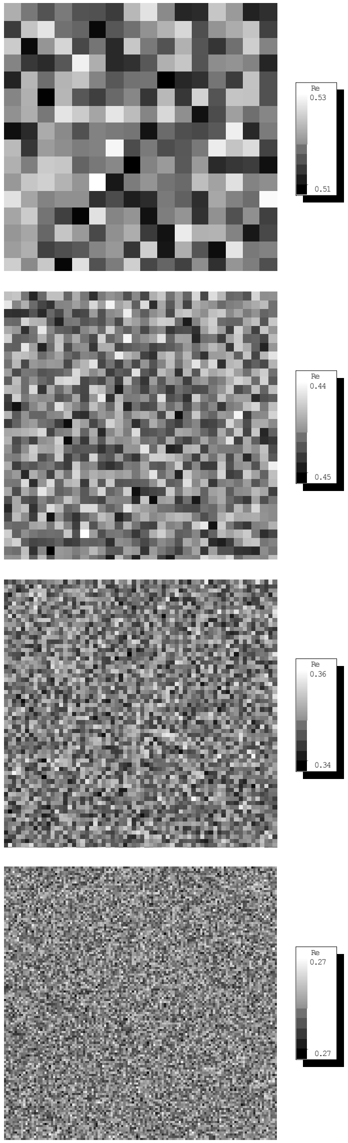
```
QuantumCircuitToDiagram[circ];
```

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

Generalized Singular Value Decomposition

- GSVD exploits fact that blocks of a partitioned unitary matrix have highly related singular value decompositions *(see Golub & van Loan, "Matrix Computations", p.77)*
- GSVD decomposition of a $2^n \times 2^n$ unitary matrix

$$\boxed{U = \underbrace{\left(\begin{array}{c|c} U_{00} & U_{01} \\ \hline U_{10} & U_{11} \end{array}\right)}_{2^n} = \left(\begin{array}{c|c} L_0 & 0 \\ \hline 0 & L_1 \end{array}\right) \cdot \left(\begin{array}{c|c} D_{00} & D_{01} \\ \hline D_{10} & D_{11} \end{array}\right) \cdot \left(\begin{array}{c|c} R_0 & 0 \\ \hline 0 & R_1 \end{array}\right)}$$

- L_0, L_1, R_0, R_1, are $2^{n-1} \times 2^{n-1}$ unitary matrices
- $D_{00} = D_{11} = diag(C_1, C_2, \ldots, C_{2^{n-1}})$
- $D_{10} = -D_{01} = diag(S_1, S_2, \ldots, S_{2^{n-1}})$

Apply Recursively …

- Recurse until factors are direct sums of 1-qubit gates

$$\left(\begin{array}{c|c} L_0 & 0 \\ \hline 0 & L_1 \end{array}\right) = \left(\begin{array}{c|c} \left(\begin{array}{c|c} L_0' & 0 \\ \hline 0 & L_1' \end{array}\right)\cdot\left(\begin{array}{c|c} D_{00}' & D_{01}' \\ \hline D_{10}' & D_{11}' \end{array}\right)\cdot\left(\begin{array}{c|c} R_0' & 0 \\ \hline 0 & R_1' \end{array}\right) & 0 \\ \hline 0 & \left(\begin{array}{c|c} L_0'' & 0 \\ \hline 0 & L_1'' \end{array}\right)\cdot\left(\begin{array}{c|c} D_{00}'' & D_{01}'' \\ \hline D_{10}'' & D_{11}'' \end{array}\right)\cdot\left(\begin{array}{c|c} R_0'' & 0 \\ \hline 0 & R_1'' \end{array}\right) \end{array}\right)$$

$$= \left(\begin{array}{cc|cc} L_0' & & & 0 \\ & L_1' & & \\ \hline & 0 & L_0'' & \\ & & & L_1'' \end{array}\right)\cdot\underbrace{\left(\begin{array}{cc|cc} D_{00}' & D_{01}' & & 0 \\ D_{10}' & D_{11}' & & \\ \hline & 0 & D_{00}'' & D_{01}'' \\ & & D_{10}'' & D_{11}'' \end{array}\right)}_{}\cdot\left(\begin{array}{cc|cc} R_0' & & & 0 \\ & R_1' & & \\ \hline & 0 & R_0'' & \\ & & & R_1'' \end{array}\right)$$

1-qubit gates

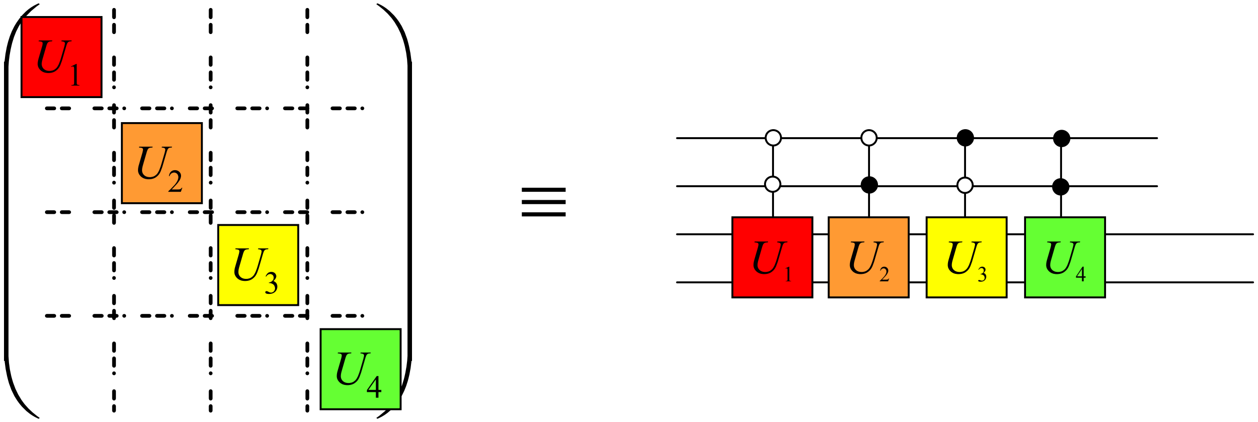
1-qubit gates

Central matrix = blocks of tri-banded matrices.
Needs special handling

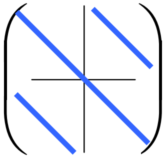
© Colin P. Williams 2004

Block-Diagonals “Easy”

- Once you have a block-diagonal form can easily map this into an equivalent “conditional” quantum logic circuit

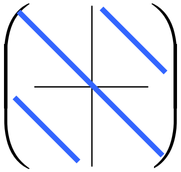


Tri-banded to Block-diagonal

- **Central matrix** $\left(\begin{array}{c|c} D_{00} & D_{01} \\ \hline D_{10} & D_{11} \end{array}\right) \equiv$  **is always tri-banded**

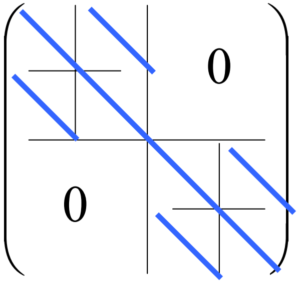
- **Can map tri-banded matrix to block-diagonal matrix using qubit reversal matrices,** P_n ***(cascaded SWAP gates)***

$$\left(\begin{array}{c|c} D_{00} & D_{01} \\ \hline D_{10} & D_{11} \end{array}\right) \equiv$$



$$= P_n^{-1} \cdot \left(\begin{array}{c|c} \blacksquare & 0 \\ \hline 0 & \blacksquare \end{array}\right) \cdot P_n$$

$$\left(\begin{array}{cc|cc} D'_{00} & D'_{01} & & \\ \hline D'_{10} & D'_{11} & \multicolumn{2}{c}{0} \\ \hline & & D''_{00} & D''_{01} \\ \cline{3-4} \multicolumn{2}{c|}{0} & D''_{10} & D''_{11} \end{array}\right) \equiv$$

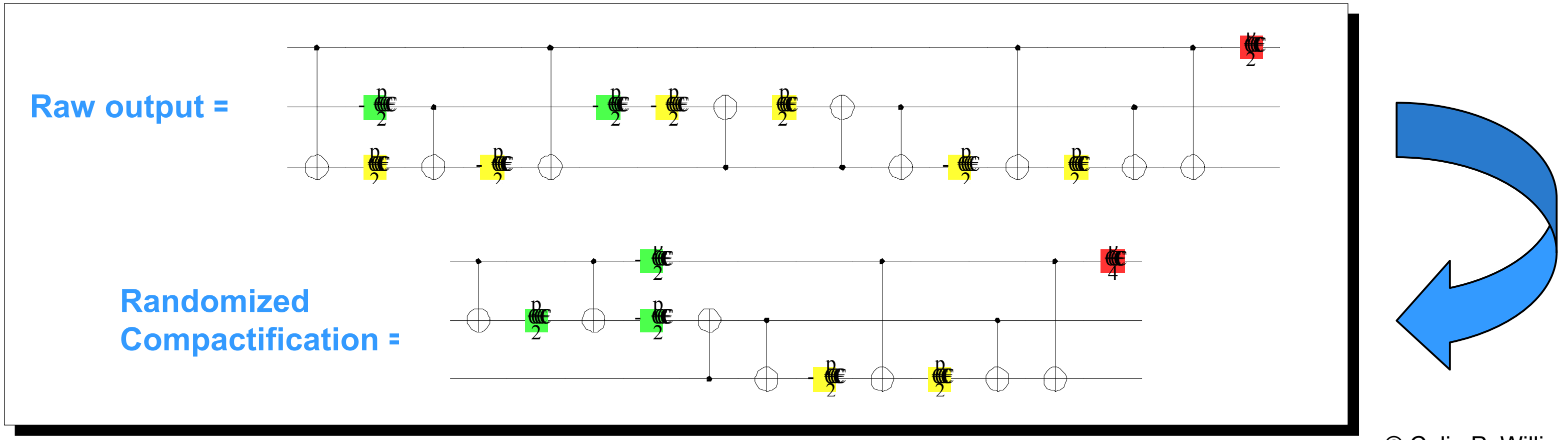


$$= (P_{n-1}^{-1} \oplus P_{n-1}^{-1}) \cdot \left(\begin{array}{c|c|c|c} \blacksquare & 0 & 0 & 0 \\ \hline 0 & \blacksquare & 0 & 0 \\ \hline 0 & 0 & \blacksquare & 0 \\ \hline 0 & 0 & 0 & \blacksquare \end{array}\right) \cdot (P_{n-1} \oplus P_{n-1})$$

Circuit Compactification

- **Output from GSVD can be compactified using randomized scheme**
  - Select a sub-circuit, computes implied unitary matrix, redesigns a circuit for it, and accepts the result if of lower depth
  - Compactifies across boundaries of adjacent conditional gates, e.g.,

$$\text{Target matrix} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix}$$

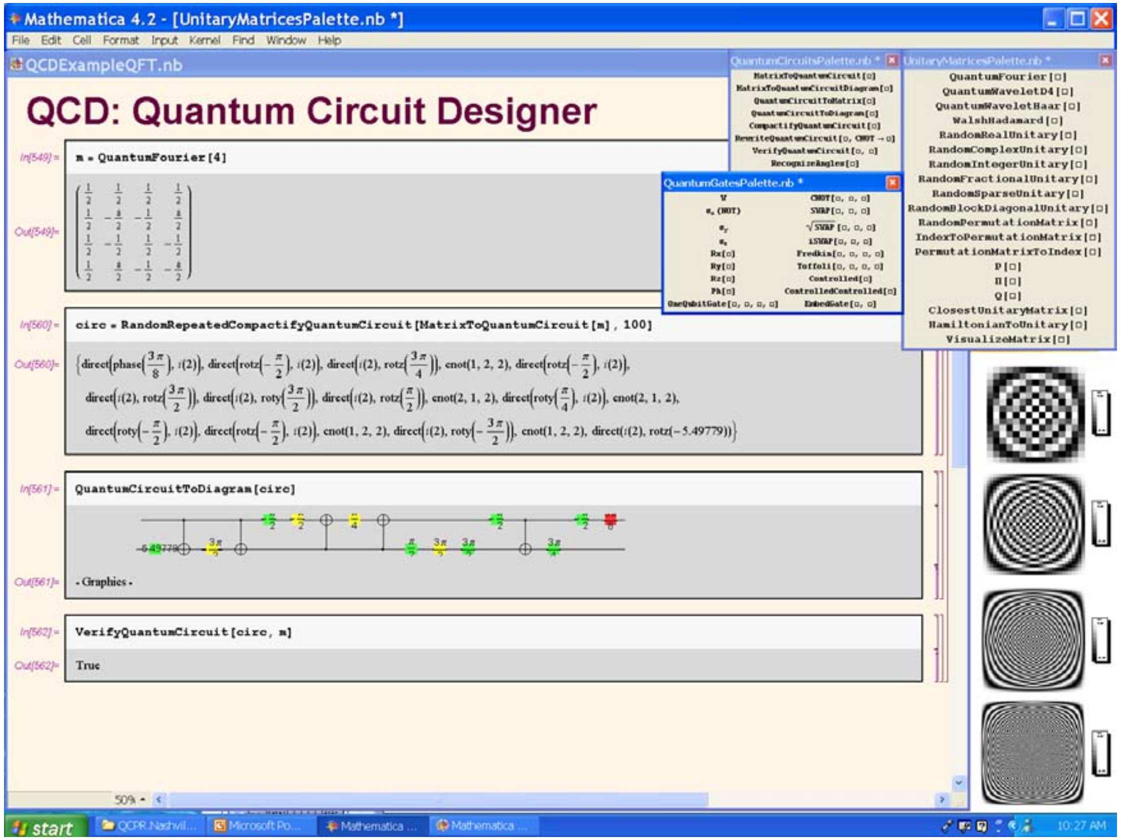


Raw output =

Randomized Compactification =

Quantum Fourier Transform

- **QCD can detect special structure if it exists**
  - E.g. QCD finds a compact circuit for QFT
  - Comparable to direct conversion of usual QFT circuit which involves conditional gates


Mathematica 4.2 - [UnitaryMatricesPalette.nb *]
File Edit Cell Format Input Kernel Find Window Help
QCDExampleQFT.nb
QuantumCircuitsPalette.nb *
UnitaryMatricesPalette.nb *
QuantumGatesPalette.nb *
50%
start
QCPR Nashvil...
Microsoft Po...
Mathematica ...
Mathematica ...
10:27 AM

QCD: Quantum Circuit Designer

In[549]:= `m = QuantumFourier[4]`

Out[549]=

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

In[560]:= `circ = RandomRepeatedCompactifyQuantumCircuit[MatrixToQuantumCircuit[m], 100]`

Out[560]=

$$\{\text{direct}(\text{phase}(\tfrac{3\pi}{8}), i(2)), \text{direct}(\text{rotz}(-\tfrac{\pi}{2}), i(2)), \text{direct}(i(2), \text{rotz}(\tfrac{3\pi}{4})), \text{cnot}(1, 2, 2), \text{direct}(\text{rotz}(-\tfrac{\pi}{2}), i(2)), \text{direct}(i(2), \text{rotz}(\tfrac{3\pi}{2})), \text{direct}(i(2), \text{roty}(\tfrac{3\pi}{2})), \text{direct}(i(2), \text{rotz}(\tfrac{\pi}{2})), \text{cnot}(2, 1, 2), \text{direct}(\text{roty}(\tfrac{\pi}{4}), i(2)), \text{cnot}(2, 1, 2), \text{direct}(\text{roty}(-\tfrac{\pi}{2}), i(2)), \text{direct}(\text{rotz}(-\tfrac{\pi}{2}), i(2)), \text{cnot}(1, 2, 2), \text{direct}(i(2), \text{roty}(-\tfrac{3\pi}{2})), \text{cnot}(1, 2, 2), \text{direct}(i(2), \text{rotz}(-5.49779))\}$$

In[561]:= `QuantumCircuitToDiagram[circ]`

Out[561]= - Graphics -

In[562]:= `VerifyQuantumCircuit[circ, m]`

Out[562]= True

QuantumCircuitsPalette.nb:

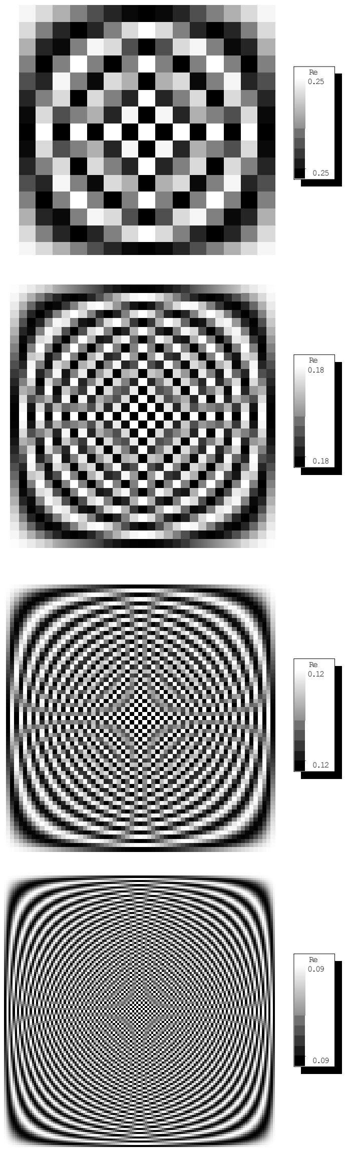
```
MatrixToQuantumCircuit[□]
MatrixToQuantumCircuitDiagram[□]
QuantumCircuitToMatrix[□]
QuantumCircuitToDiagram[□]
CompactifyQuantumCircuit[□]
RewriteQuantumCircuit[□, CNOT → □]
VerifyQuantumCircuit[□, □]
RecognizeAngles[□]
```

QuantumGatesPalette.nb:

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

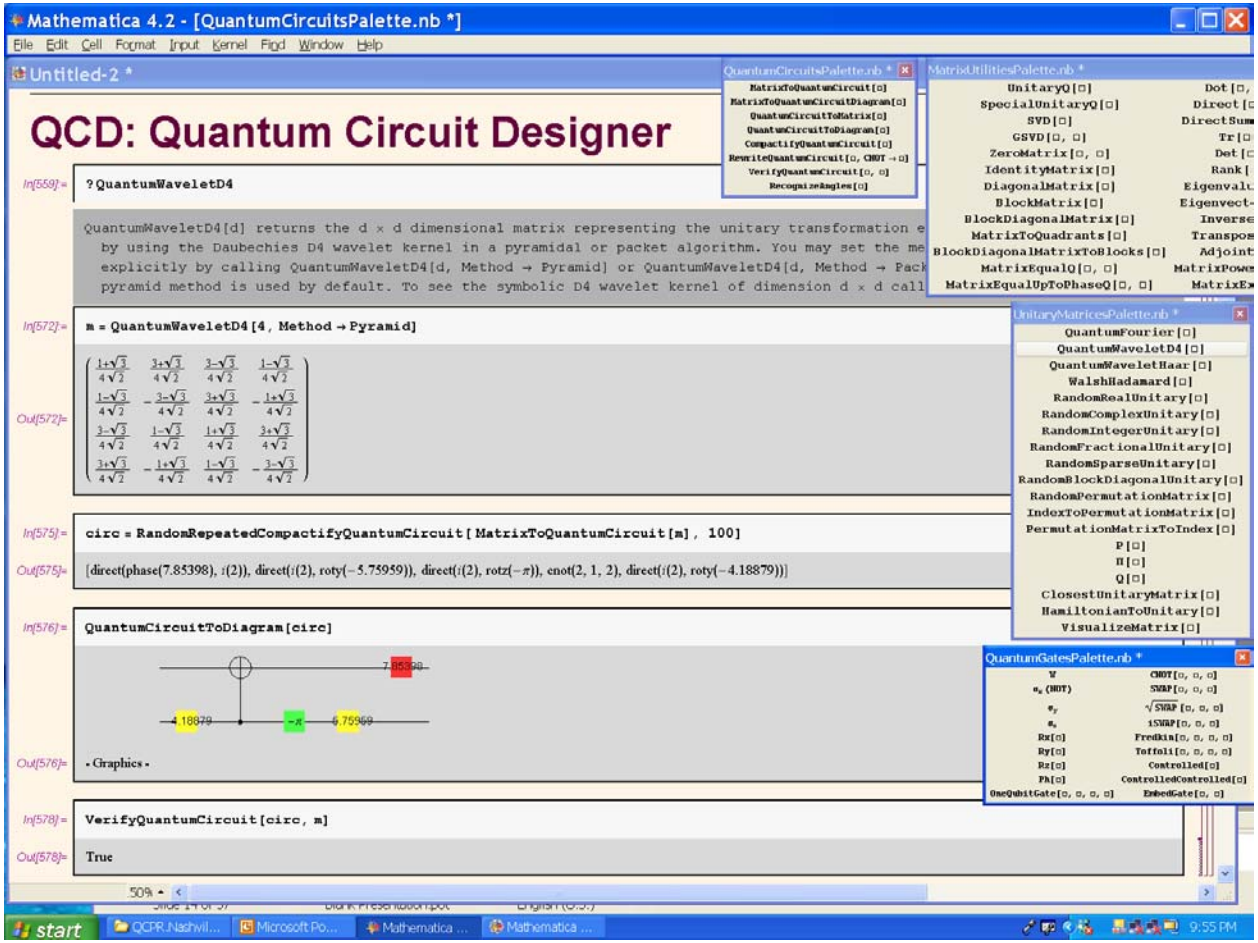
UnitaryMatricesPalette.nb:

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



Quantum Wavelet (D4) Transform

- **QWT (in pyramid algorithm) also has special structure**
  - QCD also finds compact circuits for QWT


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 *

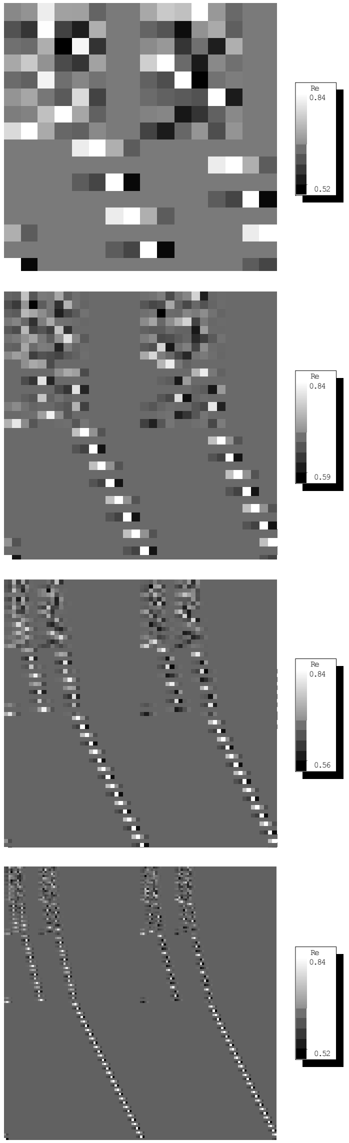
```
UnitaryQ[□]                           Dot[□,
SpecialUnitaryQ[□]                    Direct[□
SVD[□]                                DirectSum
GSVD[□, □]                            Tr[□
ZeroMatrix[□, □]                      Det[□
IdentityMatrix[□]                     Rank[
DiagonalMatrix[□]                     Eigenvalu
BlockMatrix[□]                        Eigenvect
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[□, □, □]
σx (NOT)                       SWAP[□, □, □]
σy                             √SWAP[□, □, □]
σz                             iSWAP[□, □, □]
Rx[□]                          Fredkin[□, □, □, □]
Ry[□]                          Toffoli[□, □, □, □]
Rz[□]                          Controlled[□]
Ph[□]                          ControlledControlled[□]
OneQubitGate[□, □, □, □]       EmbedGate[□, □]
```



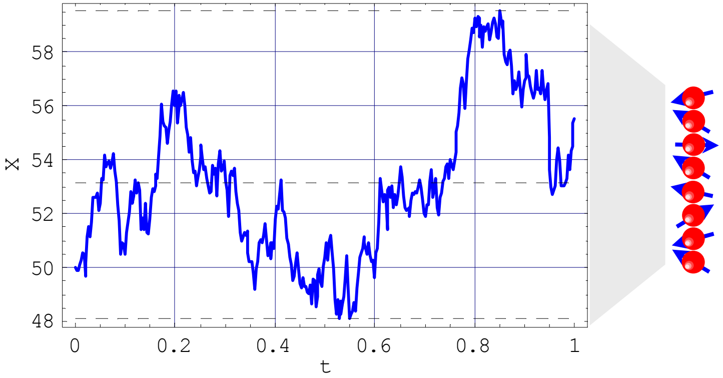
Signal, Data and Image Processing

- **Earth Sciences and Space Sciences Enterprises**
- **Signal, image and data processing fundamentally different on a quantum computer than classical computer**
  - –Classical-to-quantum data encoding
    - •Linear 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

Quantum Computer Hardware

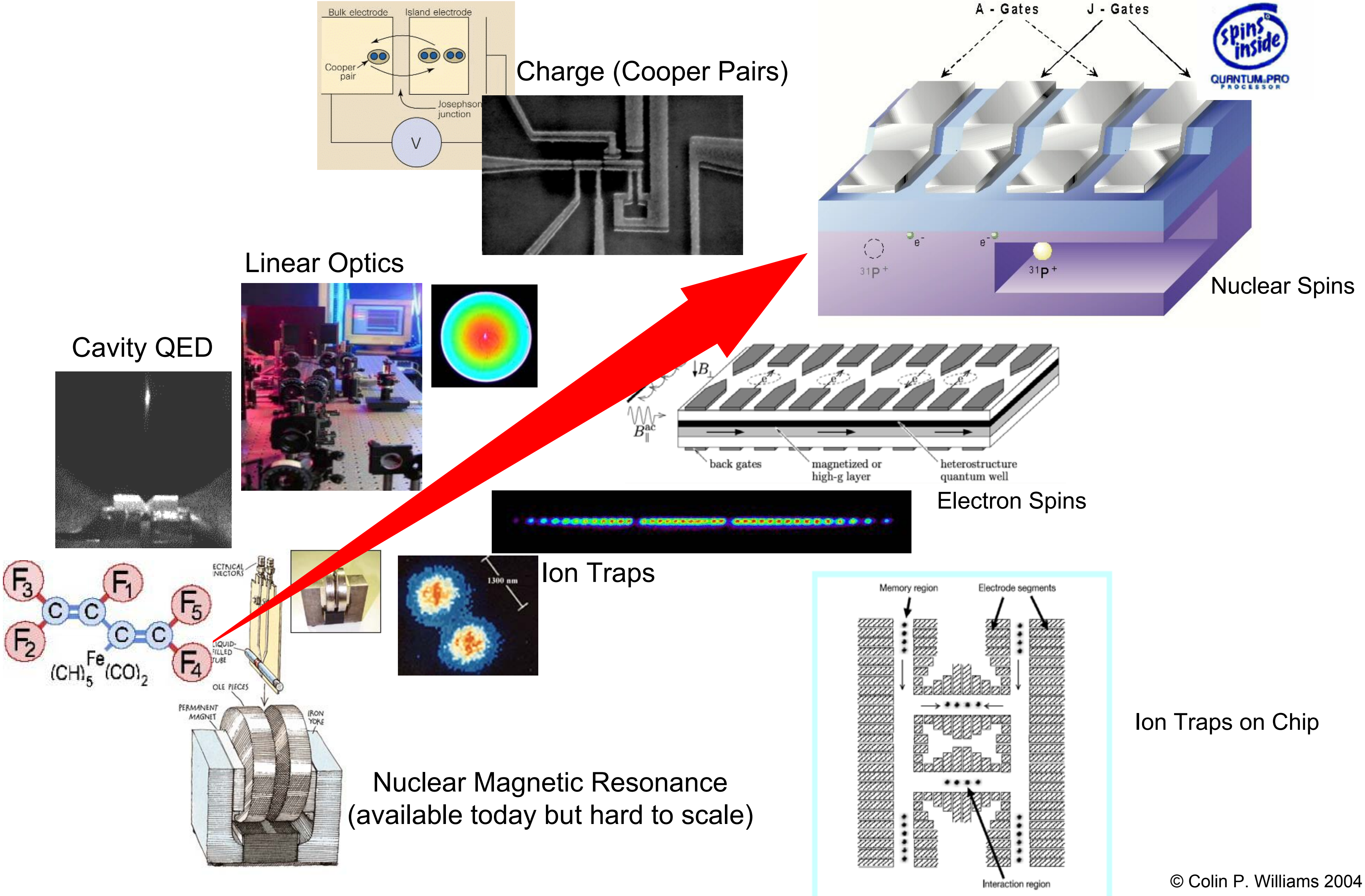
What is Needed to make a Quantum Computer?

- ***Necessary*** **criteria for a system to serve as a quantum computer**

Requirement	*Explanation*
Qubits	*There are quantum states that can serve as qubits*
Initialization	*All qubits can be placed in a standard starting state*
Static Memory	*Qubits must not change during storage*
Unitary Operations	*Can do unitary operations on arbitrary subsets of qubits*
Conditional Operations	*Operation performed on one qubit depends upon of another*
Readout	*The value of any qubit accessible via measurement operation*
Isolation	*Qubits must not interact with environment in between readouts*
Error Correction	*Unknown (and unknowable) errors can be corrected*

- **Detailed information on all major schemes available at**
  - –http://qist.lanl.gov (ARDA's Quantum Computing Roadmap)
  - –http://xxx.lanl.gov/archive/quant-ph (Preprint server for all things quantum)

Architectures Converging on Nanoelectronics



Charge (Cooper Pairs)

Nuclear Spins

Linear Optics

Cavity QED

Electron Spins

Ion Traps

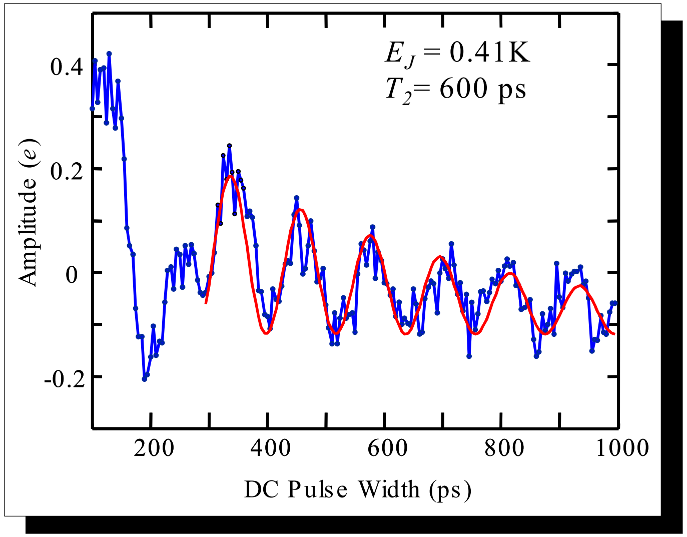
Ion Traps on Chip

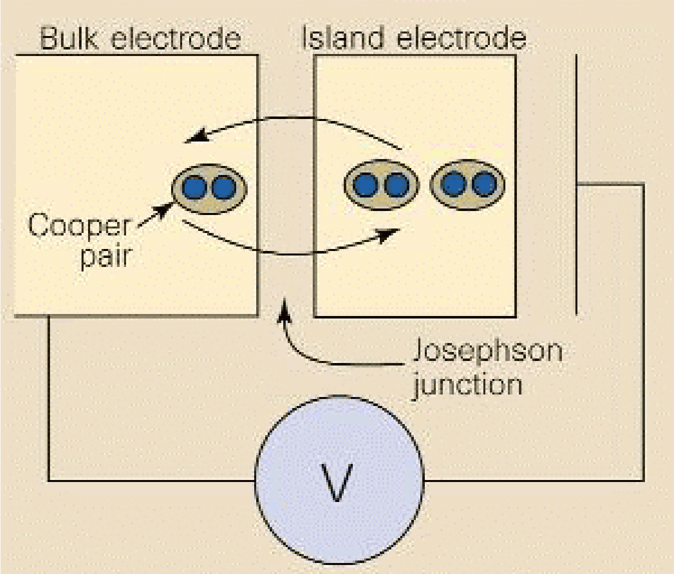
Nuclear Magnetic Resonance
(available today but hard to scale)

Superconductor-Based Quantum Hardware at JPL

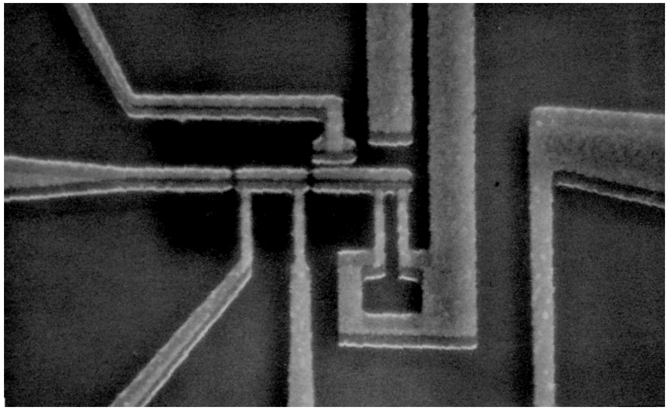
Charge-based Qubits

- **Qubit as a Single Cooper Pair Box**[1,2]
  - Cooper pairs tunnel through Josephson junction onto island
  - Qubit encoded as the number of Cooper pairs on the island
  - Coherent oscillations in the number of pairs





❑ *Charge-based qubit fabricated at JPL using e-beam lithography*



SCB-based qubit fabricated in Aluminum using e-beam lithography.

[1] Y. Nakamura, Yu. A. Pashkin, and J. S. Tsai, Nature 398, 786 (1999).

[2] P. Echternach, C. P. Williams, et al. "Universal Quantum Gates for Single Cooper Pair Box Based Quantum Computing," Quantum Information and Computation, Vol. 1, (2001) 143-150 (also at http://xxx.lanl.gov/abs/quant-ph/0112025).

Hamiltonian for 2-Qubit Gate

- **Qubit-Qubit interaction Hamiltonian**

$$\hat{H}_2 = \begin{pmatrix} -\frac{E_1}{2} - \frac{E_2}{2} & -\tfrac{1}{2}E_{J_2}(\Phi_2) & -\tfrac{1}{2}E_{J_1}(\Phi_1) & 0 \\ -\tfrac{1}{2}E_{J_2}(\Phi_2) & -\frac{E_1}{2} + \frac{E_2}{2} & -\tfrac{1}{2}E_{J_C}(\Phi_C) & -\tfrac{1}{2}E_{J_1}(\Phi_1) \\ -\tfrac{1}{2}E_{J_1}(\Phi_1) & -\tfrac{1}{2}E_{J_C}(\Phi_C) & \frac{E_1}{2} - \frac{E_2}{2} & -\tfrac{1}{2}E_{J_2}(\Phi_2) \\ 0 & -\tfrac{1}{2}E_{J_1}(\Phi_1) & -\tfrac{1}{2}E_{J_2}(\Phi_2) & \frac{E_1}{2} + \frac{E_2}{2} \end{pmatrix}$$

- **Specialize** $n_{C_1} = n_{C_2} = \tfrac{1}{2}$ **and** $E_{J_1} = E_{J_2} = 0$

$$\hat{U}_2 = \exp(-\frac{i\hat{H}_2 t}{\hbar})$$

Induced Unitary Transformation

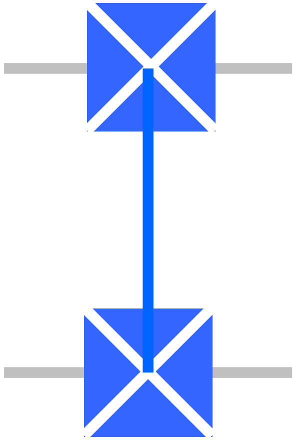
$$\hat{U}_2 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\left(\frac{E_{J_C}\Delta t}{2\hbar}\right) & i\sin\left(\frac{E_{J_C}\Delta t}{2\hbar}\right) & 0 \\ 0 & i\sin\left(\frac{E_{J_C}\Delta t}{2\hbar}\right) & \cos\left(\frac{E_{J_C}\Delta t}{2\hbar}\right) & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

The iSWAP Gate

- **Can make any 1-Qubit gate**
- **But no obvious way to make CNOT**
- **However, can make a new 2-qubit gate called "iSWAP"**

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

iSWAP **circuit icon**

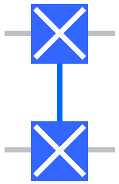


- **Question: is iSWAP as useful as CNOT ?**
  - Is set of all 1-qubit gates $\cup$ iSWAP a universal gate set?
  - Are iSWAP circuits as efficient as CNOT circuits?

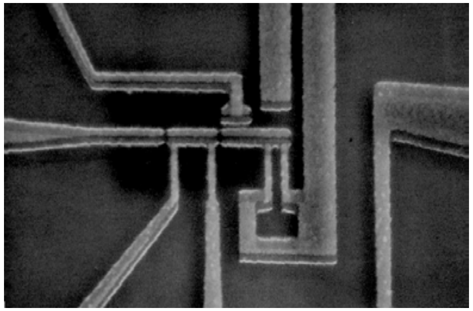
iSWAP-based circuits

- **iSWAP is an alternative entangling gate to CNOT**

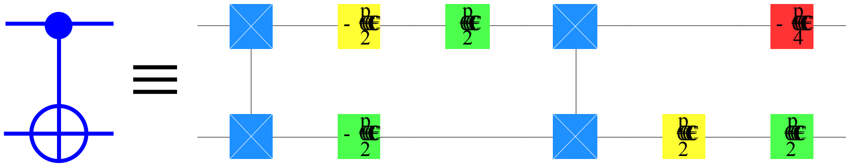
$$iSWAP \equiv e^{i\left(\frac{\pi}{4}X\otimes X+\frac{\pi}{4}Y\otimes Y\right)} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & i & 0 \\ 0 & i & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$



$iSWAP$ **circuit icon**

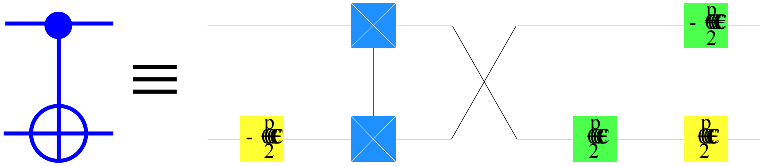


- **Proof of Universality: Write CNOT as iSWAPs and 1-qubit gates**



CNOT in two iSWAPs

- **Proof of Efficiency: Write CNOT as iSWAPs, SWAPs, & 1-qubit gates**



CNOT in one iSWAP & one SWAP

Hybrid Charge-Phase Qubits

- **Charge qubits are susceptible to fluctuations in background charges**

- **Other superconducting qubits possible**
  - E.g., the 3JJ phase qubit[1]
  - Superposition of a right and left circulating currents
  - "Long" coherence time (2.5μs)
  - But how to do 1-shot readout?

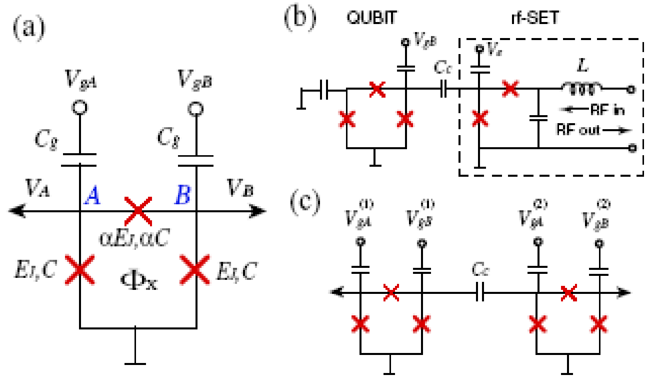
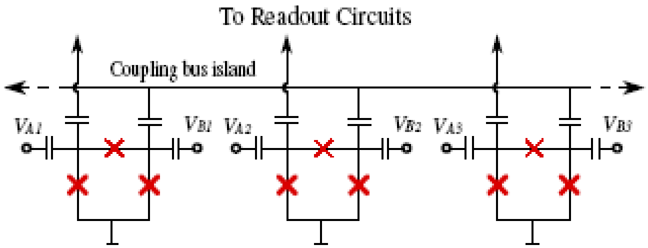


FIG. 1: (a) 3JJ qubit with two gate voltages as a Charge-phase qubit. (b) Single qubit coupled to an rf-SET as a read-out device. (c) Two capacitively coupled qubits.

- **D-Wave invented hybrid qubit[2]**
  - Dual of Saclay hybrid charge/phase qubit
  - Uncertainty in phase leads to localization in charge
  - Hence can infer phase state by measuring charge using an RF-SET (developed for reading charge-based qubits)



Coupling several qubits via a bus 1.

- **JPL now collaborating with D-Wave to make these phase/charge hybrid qubits**

[1] J. Mooij et al., Science 285, 1036 (1999))
[2] M. Amin, see http://xxx.lanl.gov/abs/cond-mat/0311220

Linear-Optics Quantum Hardware

Linear Optics Quantum Computer

- **Original optical QCs required elements with strong nonlinearities**

- **Schemes using only linear elements, and photo-detectors now known to be possible**
  - Non-linearity in detectors replaces non-linearities in elements
  - E. Knill et al., Nature 409, 46 (2001), arXiv:quant-ph/0006088

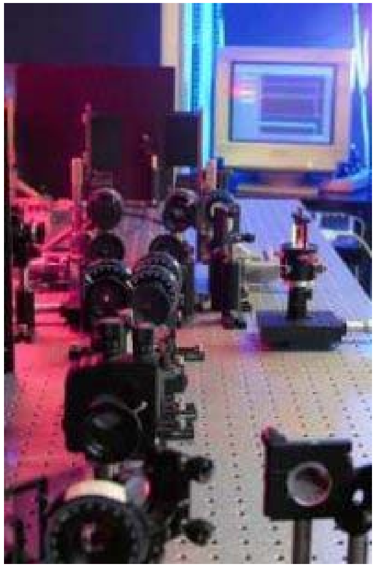
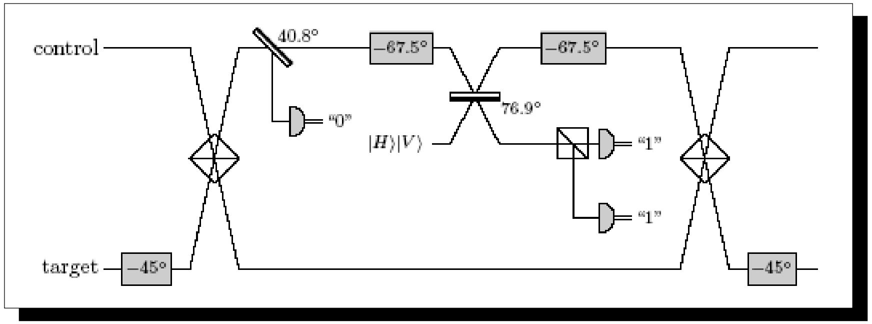


Photo courtesy Univ. Queensland

- **"Dual-rail" logic encoding:**
  - Logical "$|0\rangle$" ≡ $|1\rangle_A\,|0\rangle_B$ and logical "$|0\rangle$" ≡ $|0\rangle_A\,|1\rangle_B$
  - Modes "*A*" and "*B*" may be two spatial modes, or two polarization modes in same spatial mode

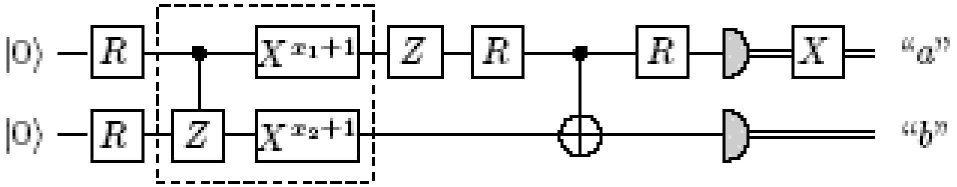
- **1-qubit gates**
  - waveplates and phase delays
- **2-qubit gates**
  - Non-deterministic CNOT and CSIGN gates (shown)



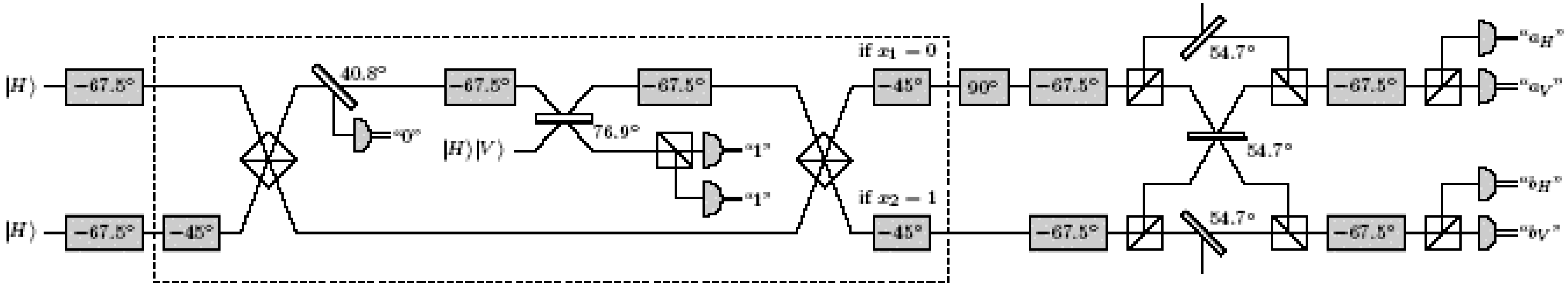
Polarization-encoded CSIGN gate (equivalent up to 1-qubit gates to CNOT)

LOQC Quantum Search Algorithm

- **Simplified quantum circuit for 2-qubit Grover algorithm**[1]



- **Equivalent LOQC interferometer set-up**



[1] J. Dodd et al. http://xxx.lanl.gov/abs/quant-ph/0306081

JPL Interests in LOQC

- Quantum computing (in collaboration with Oz QC groups)
- Using LOQC tricks in quantum communications & quantum sensors

- Heralded 2-photon entanglement source

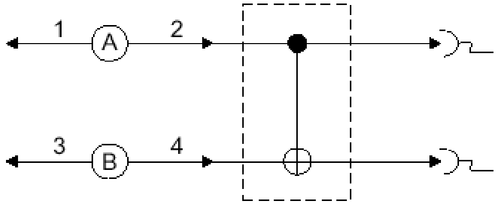
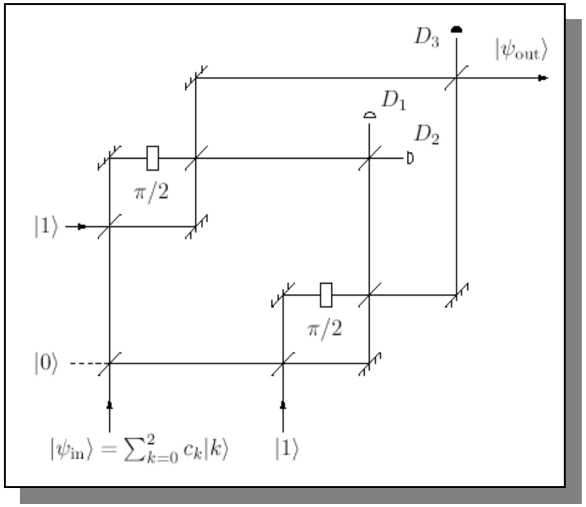


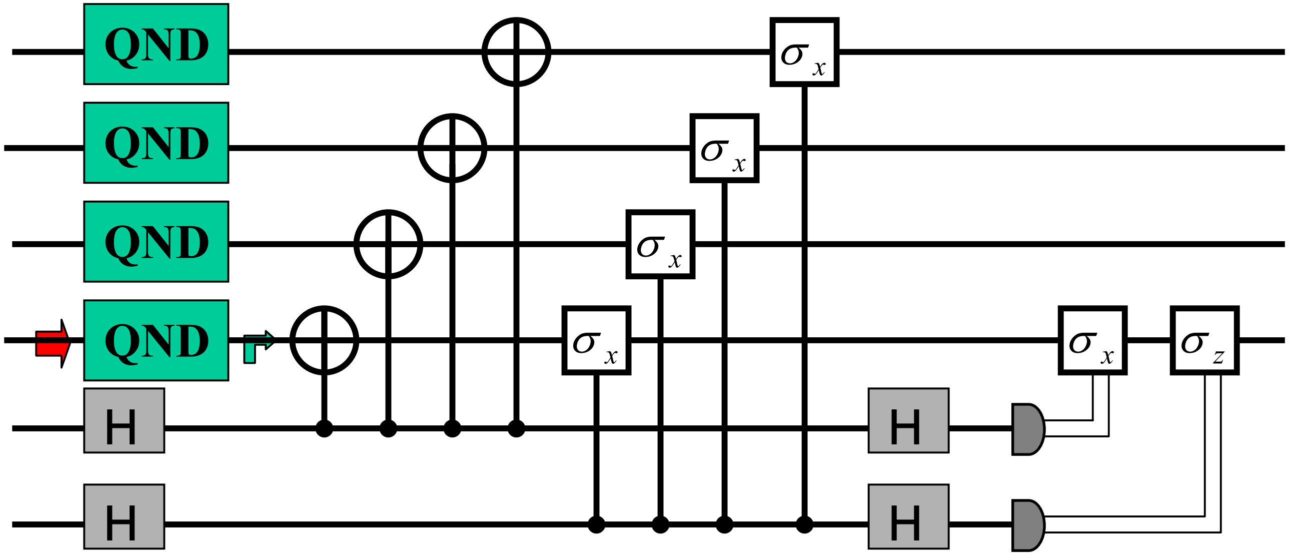
FIG. 1: An overview of the proposed method to generate heralded two-photon entanglement. A and B are envisioned as independent down-conversion sources, and the controlled-NOT gate, followed by polarization-sensitive single-photon detectors, is used to swap post-selected entanglement of modes 2 and 4 onto the remaining photons in modes 1 and 3.


D_3
$|\psi_{\text{out}}\rangle$
D_1
D_2
$\pi/2$
$|1\rangle$
$\pi/2$
$|0\rangle$
$|\psi_{\text{in}}\rangle = \sum_{k=0}^{2} c_k |k\rangle$
$|1\rangle$

- Quantum Non-Demolition Detector

Building Components that Enable more Sophisticated Devices

- QND device allows us to devise a method for correcting for photon *losses (in transmission down a fiber)*

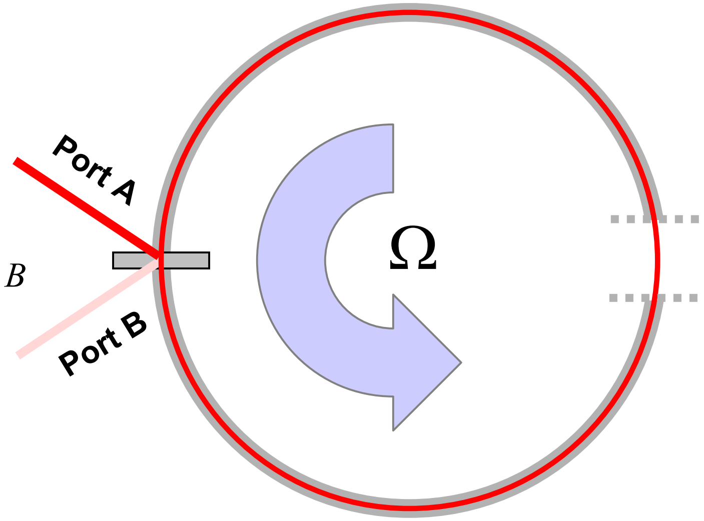


Quantum Gyroscopes

One-Port Fock Input

- **Exactly N photons per second in Port A and just vacuum in Port B**

$$|\psi\rangle = |N\rangle_A |0\rangle_B$$



$$\Delta\phi_{\text{1-Port}} = O\left(\frac{1}{\sqrt{N}}\right)$$

Phase sensitivity

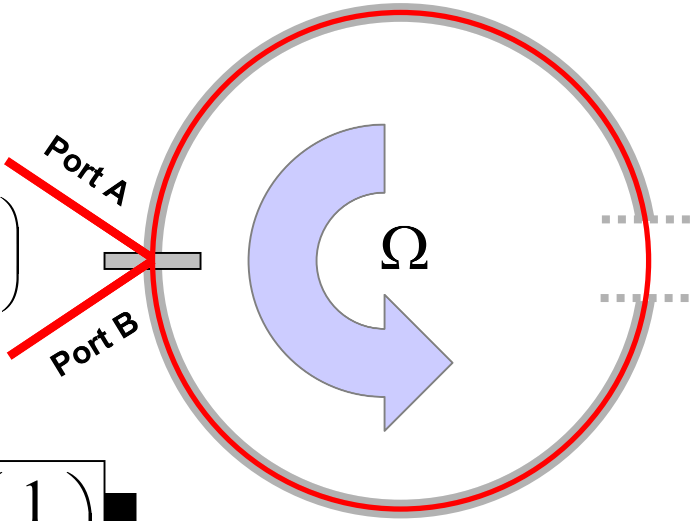
Entanglement

- **Multi-particle quantum state that cannot be factored into a definite state for each particle**
  - e.g., $|\psi\rangle = \frac{1}{\sqrt{2}}(|N\rangle_A|0\rangle_B + |0\rangle_A|N\rangle_B)$
  - Either *N* particles in path *A* and none in path *B* …,
  - … or none in path *A* and *N* in path *B*
  - State not definite until particle-number in a path is measured (counted)

Two-Port Entangled Fock State

- Entangled Fock state fed into ports A and B
- Almost equal numbers of photons per port

$$|\psi\rangle = \frac{1}{\sqrt{2}}\left(\left|\frac{N+1}{2}\right\rangle_A \left|\frac{N-1}{2}\right\rangle_B + \left|\frac{N-1}{2}\right\rangle_A \left|\frac{N+1}{2}\right\rangle_B \right)$$



$$\boxed{\Delta\phi_{2\text{-port}} = O\left(\frac{1}{N}\right)}$$

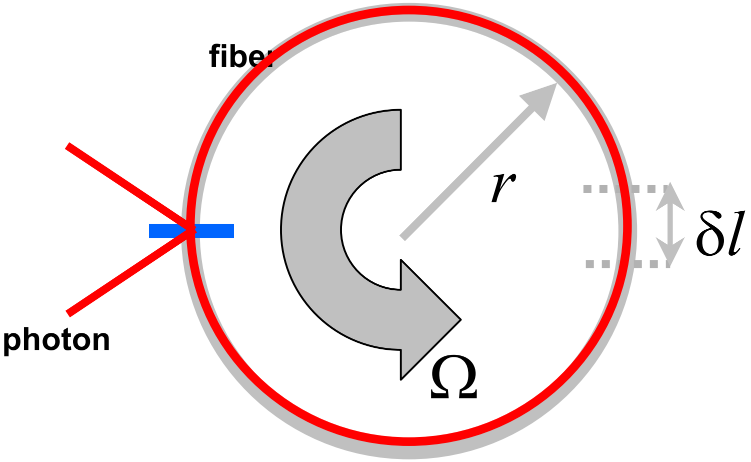
Phase sensitivity

Quantum / Classical Sensitivity

- **Minimum detectable rotation rate, $\Delta\Omega$**
  - If *N* = total number of particles passing through device per unit time
  - ~ 10^{16} photons per sec

- **Classically,** $\Delta\Omega_{\text{one-port}} \propto \frac{1}{\sqrt{N}}$

- **Quantumly,** $\Delta\Omega_{\text{two-port}} \propto \frac{1}{N}$



- **Hence 2-port quantum optical gyro 10^8 times more sensitive to rotation than equivalent 1-port optical gyro!**

Quantum Gyroscopy Applications

- **Precise rotation sensing needed for**
  - Altitude/attitude control
  - Recovery in turbulent flight
  - Drone formation-flying
  - Inertial navigation
  - Instrument pointing & stabilization
  - Unjammable GPS
  - Autonomous vehicles
  - Covert navigation

- **Quantum gyroscope is feasible**
  - Expected to be ~ 10^6 to 10^{10} times more sensitive to rotation than existing gyros!
  - "Correlated Input-port, Matter-wave Interferometer: Quantum Noise Limits to the Atom-laser Gyroscope", J. P. Dowling, Phys. Rev. A, Vol. 57, No. 6, June (1998)

Quantum Lithography

Quantum Lithography

Conventional view: feature spacing limited by wavelength of light used (Rayleigh criterion):

Spacing = λ/(2 sin(*θ*))

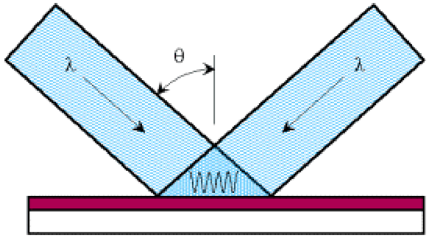
But by interfering quantum entangled photons |0>|*N*>+|*N*>|0> we obtain:

Spacing = λ/(2 *N* sin(*θ*))

Beat Rayleigh criterion by factor of *N*

Interferometric Lithography

Finer (Sub-wavelength) Lines using Entangled Light

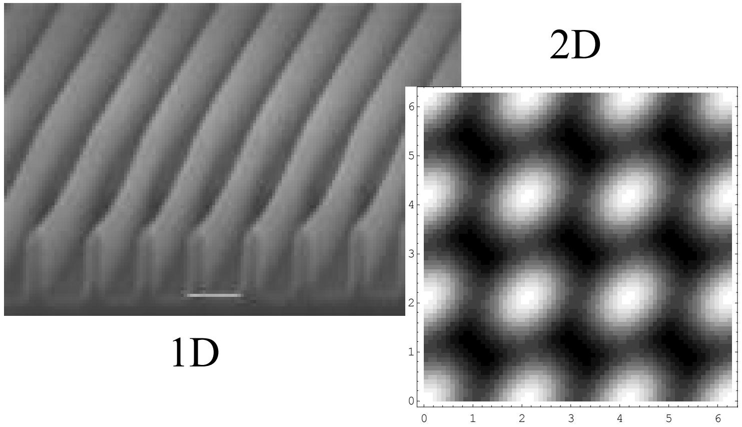


$$\text{period} = \frac{\lambda}{2\sin(\theta)}$$

135 nm at 257 nm/80°

"*Quantum Interferometric Optical Lithography: Exploiting Entanglement to Beat the Diffraction Limit*", A.. Boto, P. Kok, D. Abrams, S. Braunstein, C. P. Williams, and J. Dowling, Physical Review Letters, Vol 85, 13, (2000) pp.2733-2736

Linear improvement of *N* gives density improvement of *N*^{2}



Currently know how to do *N* = 2, 3, 4 in principle, *N* can be arbitrarily large

Ideal for ultra-fine diffraction gratings (uses in extreme spectroscopic astronomy)

More complex 2D patterns achieved by using multiple exposures using different photon input states

Input states are "Fock states" – highly non-classical light

Conclusions

- **Quantum computing allows fundamentally new kinds of algorithms**
- **Some problems can be solved exponentially faster on QCs**
  - –Factoring integers, and quantum simulation
- **Some can be solved polynomially faster on QCs**
  - –NP-Complete problems
- **With just 50 qubits can simulate physical systems beyond the reach of current supercomputers**
- **Contact:**
  - –Email: Colin.P.Williams@jpl.nasa.gov
  - –Tel: (818) 393 6998