Quantum Computing Introduction

Peter M. Kogge
Jonathan Baker
Univ. of Notre Dame

Excellent references:

- https://homes.cs.washington.edu/~oskin/quantum-notes.pdf
- https://arxiv.org/pdf/quant-ph/9809016.pdf
- http://www.dwavesys.com/
- "Quantum Annealing, Uses, Capabilities, and Potential," M. Thom – D-Wave
- http://www.research.ibm.com/quantum/
- "A Quantum Macro Assembler," Scott Pakin
- https://www.research.ibm.com/ibm-q/

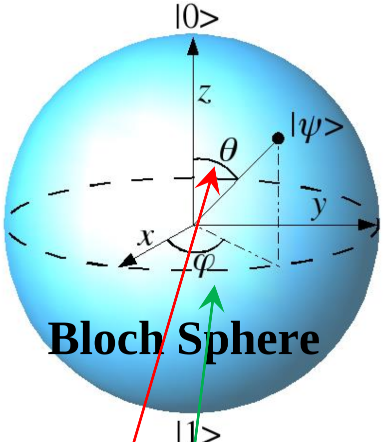
Intro

- Classical Computing: bit = “0” or “1”
- Quantum Computing: **qubit**
  - Takes on basis states **|0>** or **|1>** only when “measured”
  - Rest of time in a “superposition” of possible states

$$|\psi\rangle = \text{“state of qubit”} = a|0\rangle + b|1\rangle,$$

and a, b *complex numbers* and $|a|^2 + |b|^2 = 1$

  - $|a|^2$ is probability of being in |0> state
  - $|b|^2$ is probability of being in |1> state



- Physical phenomena
  - Photon: Vertical or Horizontal polarization
  - Electron: Spin up or Spin down
  - Atom: Discrete energy levels
- Computation: Take a vector $|\psi\rangle$ & “rotate” it.(2 angles)

Computation

- In classical logic, we use gates to manipulate the bits
- To manipulate a qubit, we use “quantum gates”
  - These gates can be represented as matrices
  - These matrices are unitary
    - $(U^T)^*U = I$
  - More importantly: **Reversible** (Invertible)
- The logic gates we are used to are **Irreversible**
- This means an operation on a qubit can be “undone”

Computation

- To perform a "quantum algorithm" perform some series of quantum gates
  - Apply matrices to the vector representing the quantum system
- *Evolution of state* in closed quantum system
  - $|\psi_{t1}\rangle = U\,|\psi_{t0}\rangle$ where t1 is some time after t0
  - and U some unitary matrix

A Simple Quantum "NOT" Gate

- Traditional logic ~1 = 0 and ~0 = 1
- Quantum: "Not" should reverse probabilities
- ~(a|0> + b|1>) = b|0> + a|1> (Switch coefficients)
- If we write value of a qubit as a vector $\begin{bmatrix} \mathbf{a} \\ \mathbf{b} \end{bmatrix}$
- Then can write ~ as $\begin{bmatrix} \mathbf{0} & \mathbf{1} \\ \mathbf{1} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{a} \\ \mathbf{b} \end{bmatrix}$
- Thus U for "not" is $\begin{bmatrix} \mathbf{0} & \mathbf{1} \\ \mathbf{1} & \mathbf{0} \end{bmatrix}$

Multi-Qubit Systems

- Given 2 independent qubits
  - |x> = a|0> + b|1>
  - |y> = c|0> + d|1>

$$|x> = \begin{bmatrix} a \\ b \end{bmatrix} \qquad |y> = \begin{bmatrix} c \\ d \end{bmatrix}$$

- "State" of composite system is "tensor product" |x>⊕|y>
  - ac|0>|0> + ad|0>|1> + bc|1>|0> + bd|1>|1>
  - = ac|00> + ad|01> + bc|10> + bd|11>

4 possible states of 2 bits = each a basis vector

- N independent bits thus have **2^N different basis vectors eqvt to 2^N classical bits**

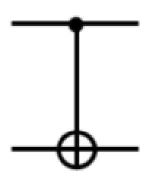
Quantum Entanglement (EPR)

- Pairs of particles where states cannot be described independently of each other
  - Even when particles separated by great distances
- Joint state space: |00>, |01>, |10>, |11>
- Once entangled, <u>cannot</u> separate out to individual spaces
- Any transformation on 1 bit affects state of other

Controlled Not

- Assume entangled $|\psi_1>,|\psi_2>$ in zero state
- Apply H = $\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$ to one bit = randomize it
  - Thus $|\psi'_1> = (1/\sqrt{2})|0> + (1/\sqrt{2})|1>$
- Apply C_{not} = $\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$
- Result = $(1/\sqrt{2})|00>+ (1/\sqrt{2})|11>$
- But there is no $|\psi_1>,|\psi_2>$ whose $\oplus$ gives this
  - 2 states cannot be considered independently

Other Quantum Gates

Gate	Notation	Matrix
NOT (Pauli-X)	$-\boxed{X}-$	$\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$
Pauli-Z	$-\boxed{Z}-$	$\begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$
Hadamard	$-\boxed{H}-$	$\frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$
CNOT (Controlled NOT)		$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$

http://www.mdpi.com/entropy/entropy-16-05290/article_deploy/html/images/entropy-16-05290f1-1024.png

Measuring A Quantum System

- At any given time, we cannot know what state a quantum bit is in
- In order to determine this, we must *measure* the system
  - By doing so, we collapse the system, and we cannot go back to where we left off.
  - Once you measure, the superposition is lost

Quantum Computation

- All known quantum algorithms solve "promise" problems
  - Structure of solution space promised to be of some form
- Use superposition, entanglement, & interference to extract info about structure
- Classically, must compute every point in solution space to obtain full knowledge
- Quantum computes every point using quantum parallelism
- Conceptually provides exponential speedup

Classical Logic on Quantum Computers

$$H = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$

$$X = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$

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

Together: These 3 gates can be used to simulate ALL of classical logic, on a quantum computer.

Classical Logic on Quantum Computers

- AND
- OR (= ¬AND)
- NOT
- etc.
- Every circuit in classic logic can be expressed using these gates
- Universal gates (NAND, NOR, Toffoli)

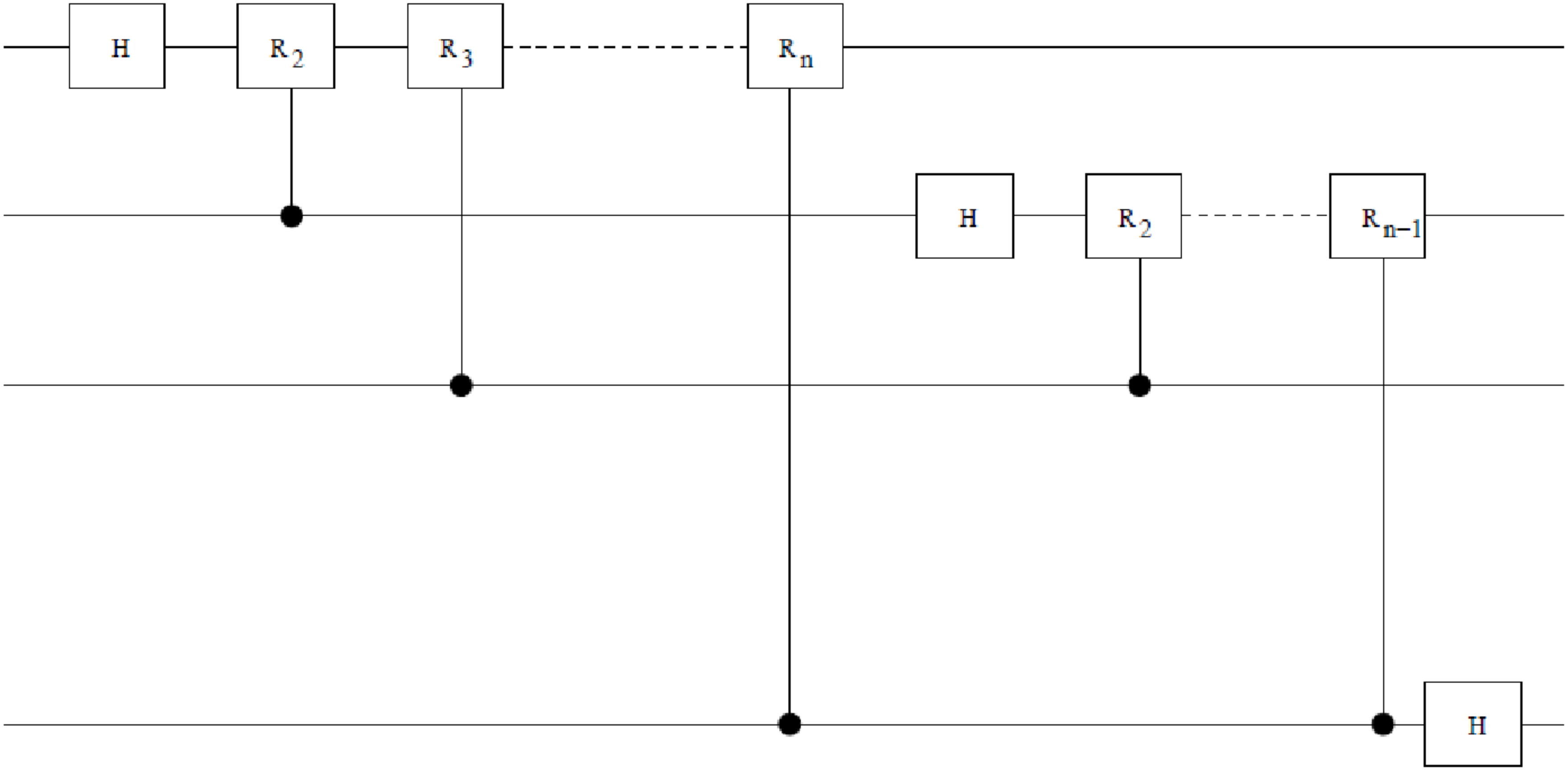
Satisfiability

- As we’ve seen in this class k-SAT is not fast on classical computers
- Question: Can we solve this problem faster by considering it on a quantum computer
- Recall that the gates are reversible
  - So if we set the output can we determine the inputs give that output
  - In fact, we can get all possible assignments this way

Known Algorithms

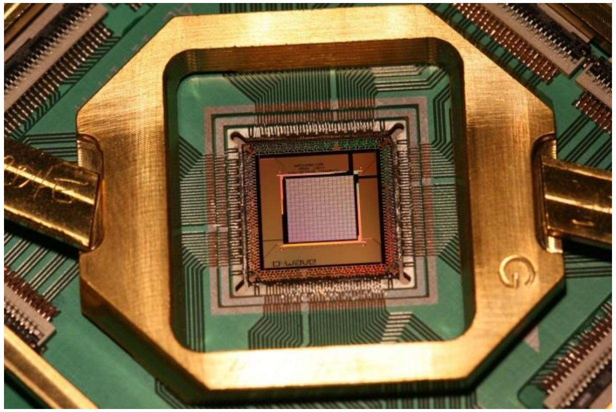
- Algorithms based on quantum Fourier transform
  - Deutsch–Jozsa algorithm
  - Simon's algorithm
  - Quantum phase estimation algorithm
  - **Shor's algorithm: Integer factorization problem** (NP-Hard solved in poly time)
  - Hidden subgroup problem
  - Boson sampling problem
  - Estimating Gauss sums
  - Fourier fishing and Fourier checking
- Algorithms based on amplitude amplification
  - Grover's algorithm
  - Quantum counting
- Algorithms based on quantum walks
  - Element distinctness problem
  - Triangle-finding problem
  - Formula evaluation
  - Group commutativity
- BQP-complete problems
  - Computing knot invariants
  - Quantum simulation

Quantum Fourier Transform

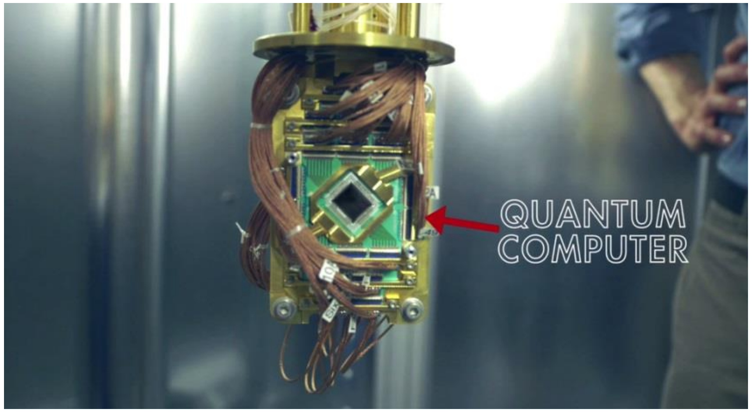


The D-WAVE Quantum Computer

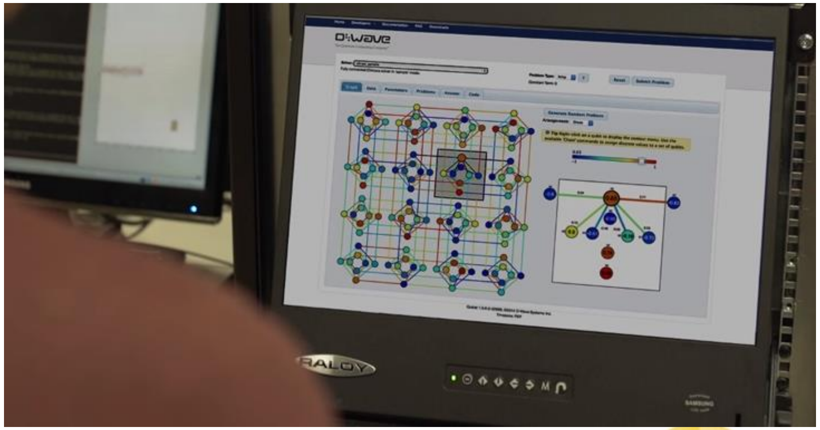
http://www.dwavesys.com/sites/default/files/styles/square_480x480/public/D-Wave%20Two%20in%20Lab.png?itok=PKiVp30g



http://www.dwavesys.com/sites/default/files/D-Wave%201000Q%20-%20lower%20res1.jpg

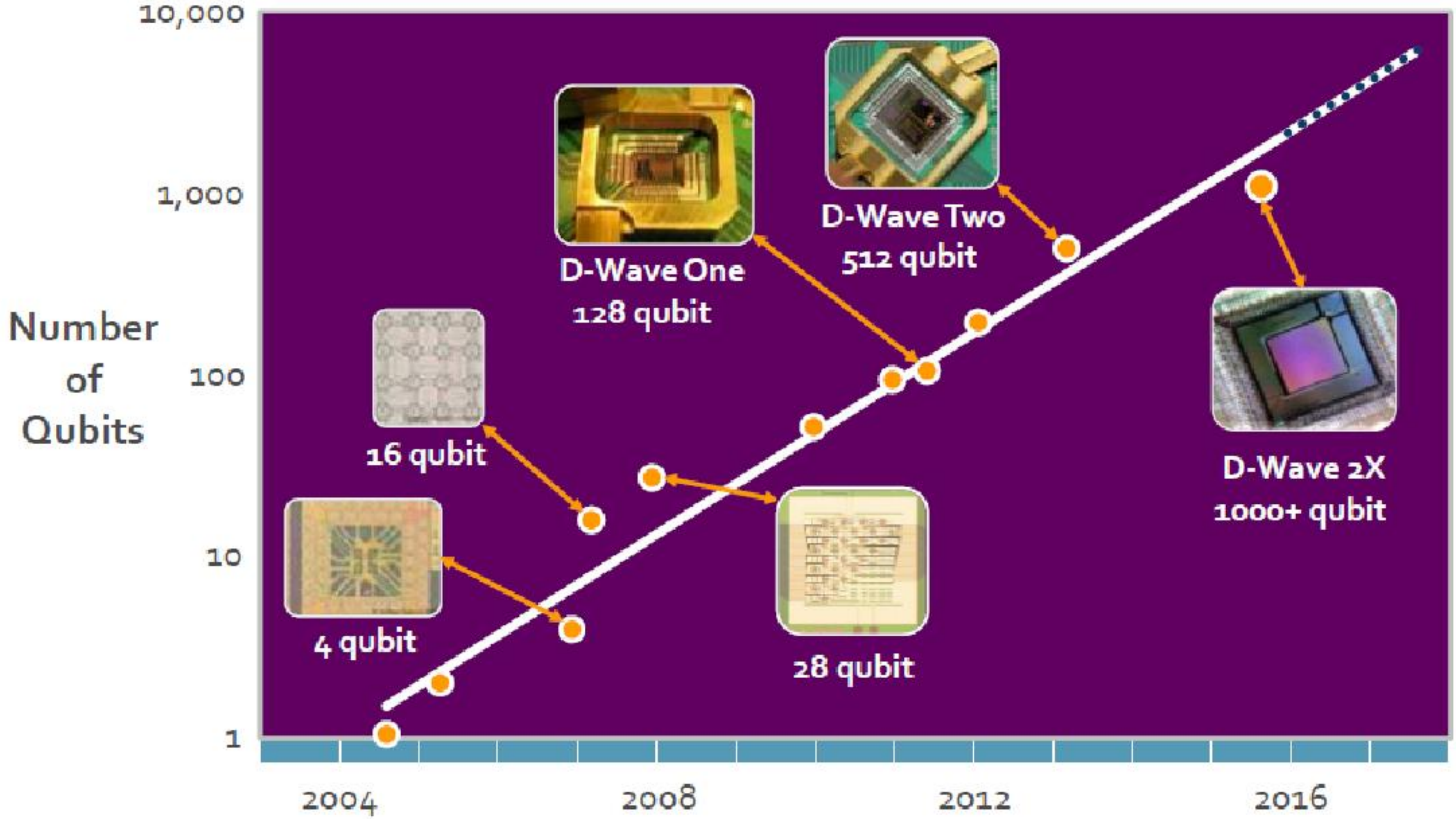


https://www.google.com/url?sa=i&rct=j&q=&esrc=s&source=images&cd=&cad=rja&uact=8&ved=0ahUKEwiGq7b_w-LQAhUC8IMKHYi3DvYQjRwIBw&url=http%3A%2F%2Fwww.techrepublic.com%2Farticle%2Fquantum-leap-d-waves-next-quantum-computing-chip-offers-a-1000x-speed-up%2F&psig=AFQjCNHxO1Wckorw-xqm_Clq78dYoEJoJw&ust=1481215492810282



http://3.bp.blogspot.com/-cKKMKqBBNBw/VmgmsYXuLOI/AAAAAAAAAltY/zMxL5ncod-Y/s1600/google-d-wave-quantum-computer-3.jpg

Quantum Annealer Scaling

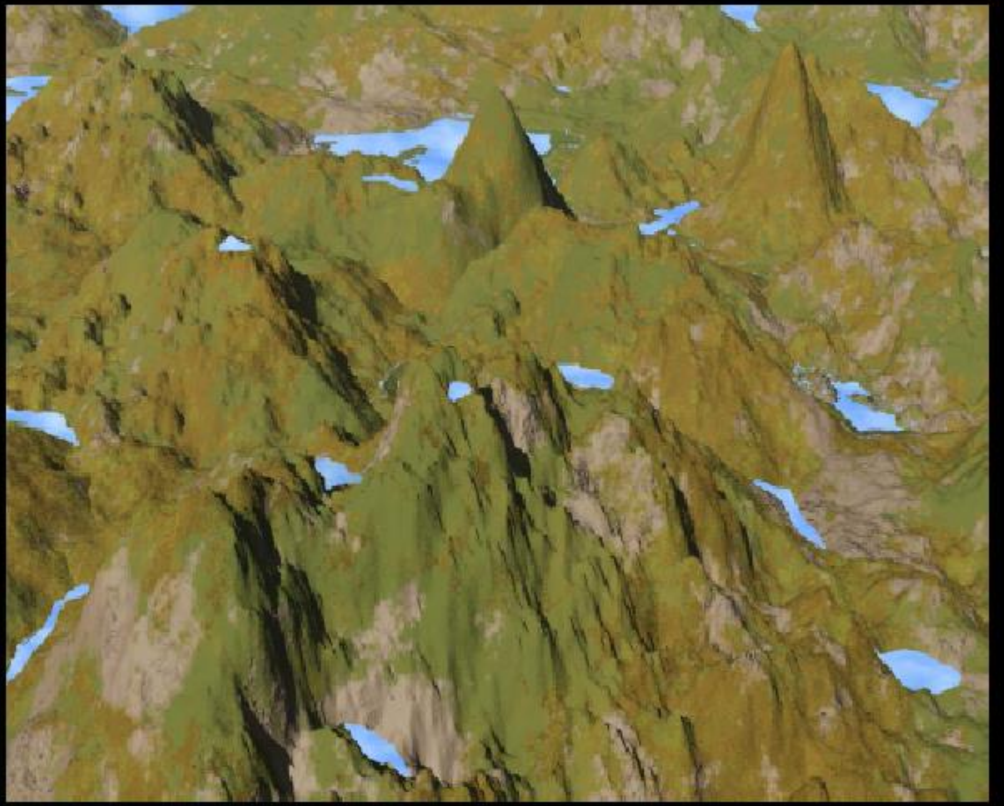


D-Wave
The Quantum Computing Company

Courtesy: D-Wave

Energy Landscape

- Space of solutions defines an energy landscape & best solution is lowest valley
- Classical algorithms must walk over this landscape
- Quantum annealing uses quantum effects to go through the mountains



Courtesy: D-Wave

Describing a Circuit Quantumly

Listing 2. QASM version of Figure 2 (circsat.qasm)

```
# Solve a circuit-satisfiability problem.

!include <gates>

!use_macro not1 not_x4
not_x4.$A = x3
not_x4.$Y = $x4

!use_macro or2 or_x5
or_x5.$A = x1
or_x5.$B = x2
or_x5.$Y = $x5

!use_macro not1 not_x6
not_x6.$A = $x4
not_x6.$Y = $x6

!use_macro and3 and_x7
and_x7.$A = x1
and_x7.$B = x2
and_x7.$C = $x4
and_x7.$Y = $x7

!use_macro or2 or_x8
or_x8.$A = $x5
or_x8.$B = $x6
or_x8.$Y = $x8

!use_macro or2 or_x9
or_x9.$A = $x6
or_x9.$B = $x7
or_x9.$Y = $x9

!use_macro and3 and_x10
and_x10.$A = $x8
and_x10.$B = $x9
and_x10.$C = $x7
and_x10.$Y = x10
```

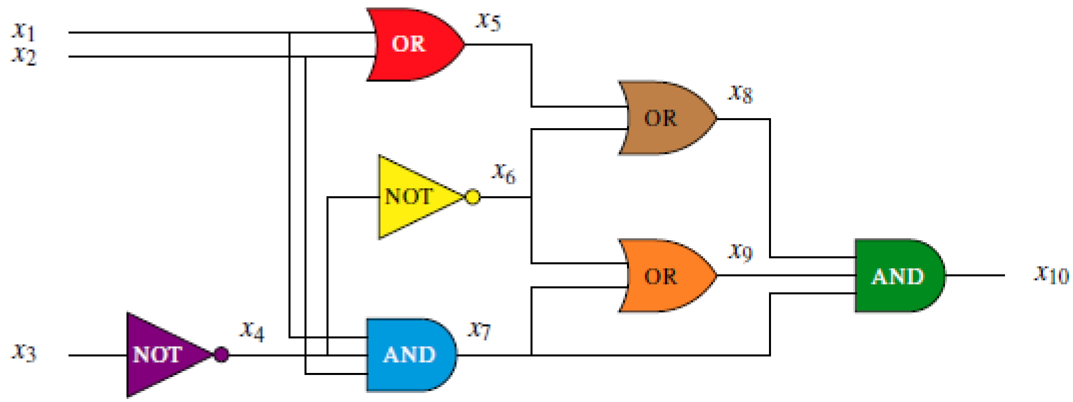
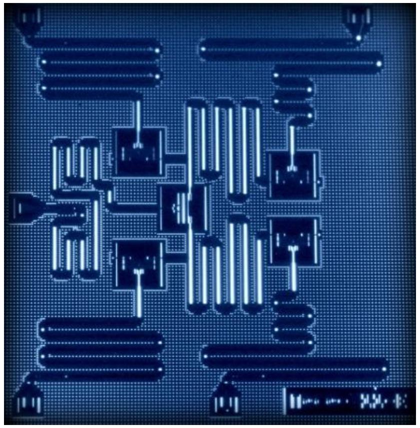


Fig. 2. A sample logic circuit

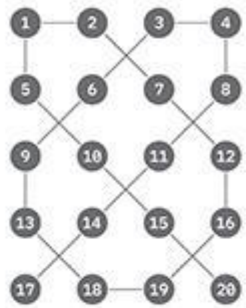
What happens if we force x10 to 1?
We solve the corresponding SAT problem!

S. Pakin, "A quantum macro assembler," ***2016 IEEE High Performance Extreme Computing Conference (HPEC)*****, Waltham, MA, USA, 2016, pp. 1-8.**
doi: 10.1109/HPEC.2016.7761637
URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=7761637&isnumber=7761574

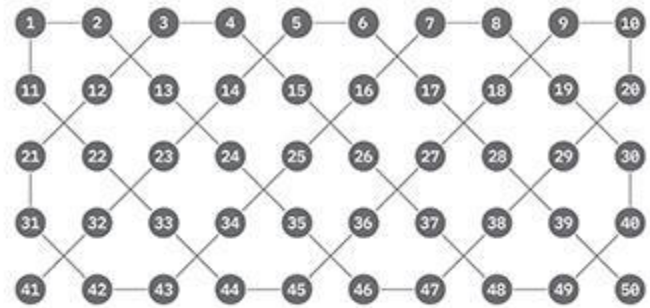
IBM Quantum Computer



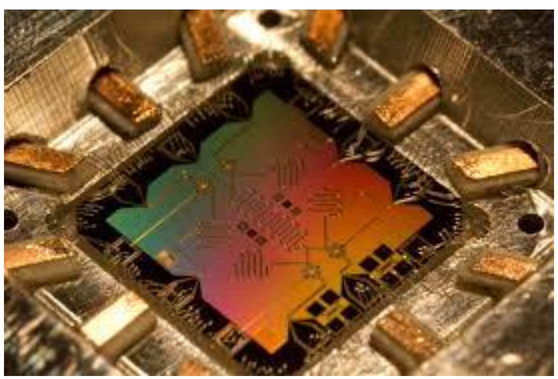
A
qubit

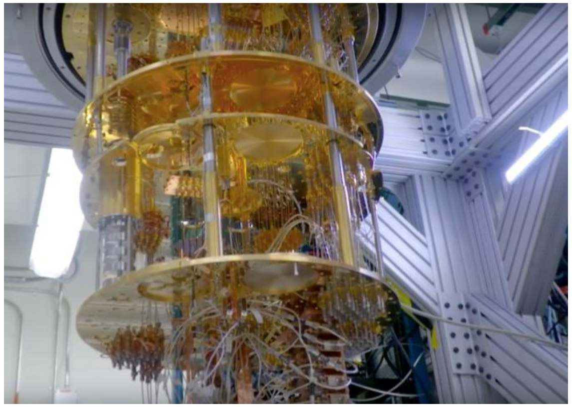


Current 20 qubit
Computer



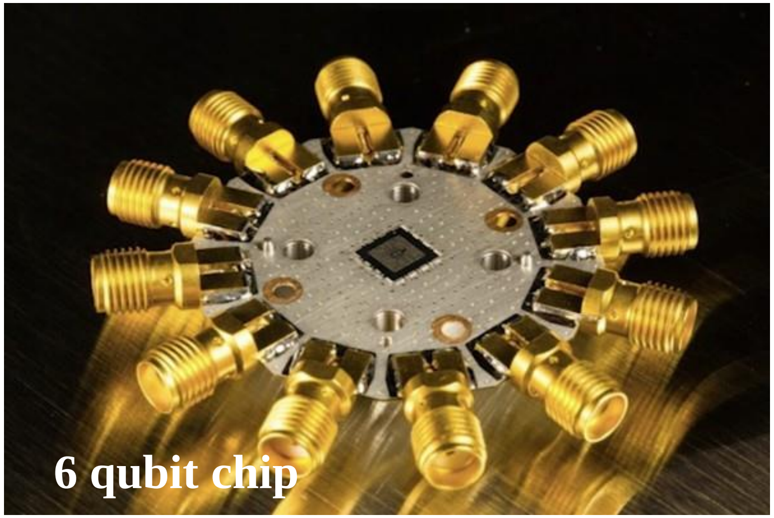
50 qubit chip in test



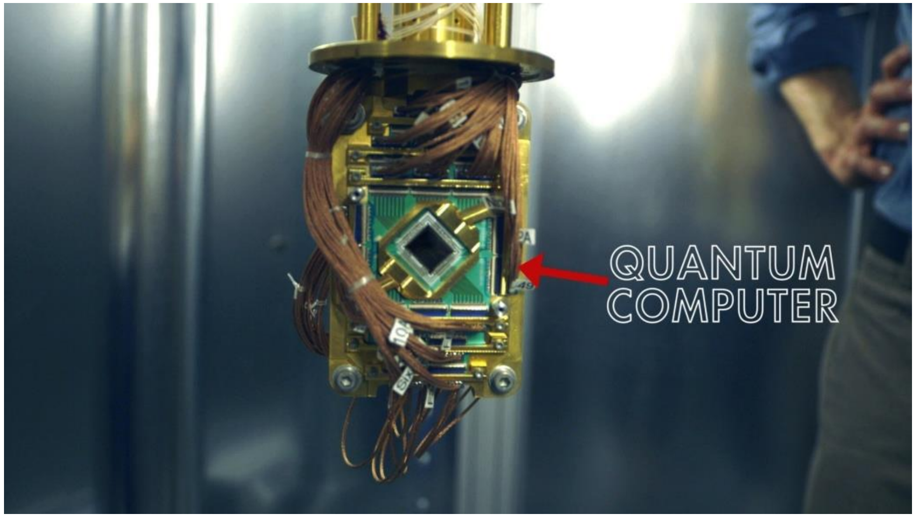


https://www.research.ibm.com/ibm-q/

Google Quantum Computer



https://cdn.technologyreview.com/i/images/chip2.jpg?sw=600&cx=0&cy=0&cw=1363&ch=912



http://fossbytes.com/wp-content/uploads/2014/09/maxresdefault.jpg