Introduction to Quantum Computing

SC21 Tutorial

Scott Pakin
Eleanor G. Rieffel

15 November 2021

LA-UR-19-31426

Resources

- **Things you'll need for the hands-on exercises**
  - Pencil and paper (or tablet or other writing mechanism)
  - A Web browser
- **Getting an early start now to save time later**
  - Bring up the tutorial slides so you can refer back to earlier material when necessary (instructions behind your badge)
  - Bring up Quirk (https://algassert.com/quirk) in a Web browser and click on the Edit Circuit button
- **Don't forget to complete the tutorial survey at the end of the day**

Agenda

- **Part I: Quantum-computing fundamentals**
  - High-level motivation, history, and status
  - Qubits, multi-qubit states, and quantum measurement

Morning break

- **Part II: Circuit-model quantum computing**
  - Review of notation
  - Quantum gates and quantum circuits

Lunch

  - Basic quantum algorithms
- **Part III: Quantum annealing**

Afternoon break

- **Part IV: Further quantum algorithms**
  - Quantum Alternating Operator Ansatz (QAOA)
  - Advancements in quantum algorithms
  - Concluding remarks

Quantum-Computing Fundamentals

Agenda

- **Part I: Quantum-computing fundamentals**
  - High-level motivation, history, and status
  - Qubits, multi-qubit states, and quantum measurement

Morning break

- **Part II: Circuit-model quantum computing**
  - Review of notation
  - Quantum gates and quantum circuits

Lunch

  - Basic quantum algorithms
- **Part III: Quantum annealing**

Afternoon break

- **Part IV: Further quantum algorithms**
  - Quantum Alternating Operator Ansatz (QAOA)
  - Advancements in quantum algorithms
  - Concluding remarks

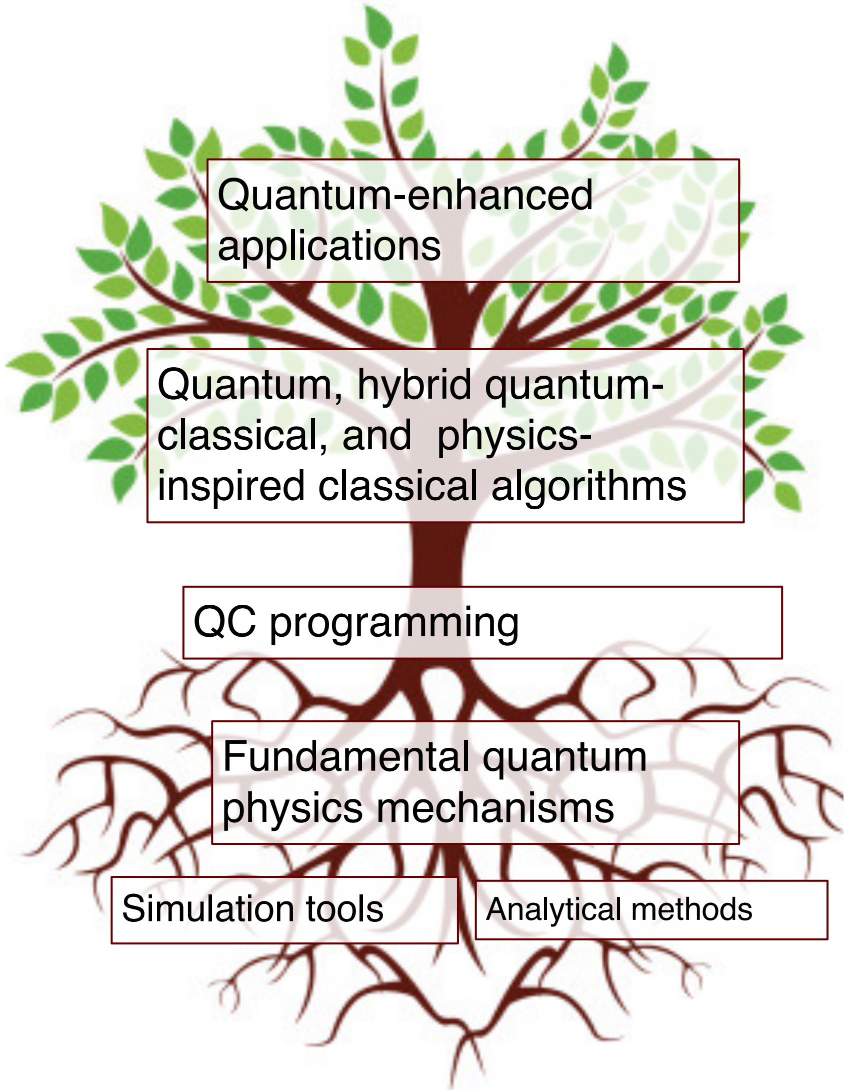
NASA's Stake in Quantum Computing

NASA constantly confronting massively challenging computational problems

- Computational capacity limits mission scope and aims

NASA's Pleiades
One of the top 25 fastest supercomputers in the world

NASA QuAIL mandate: ***Determine the potential for quantum computation*** to enable ***more ambitious NASA missions*** in the future

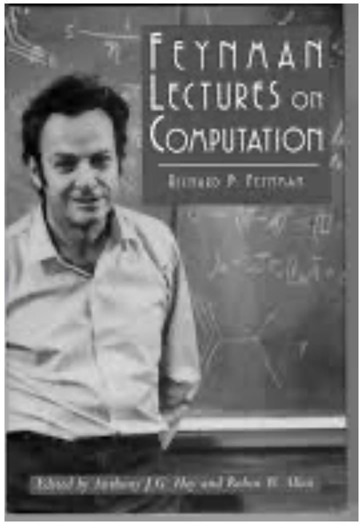


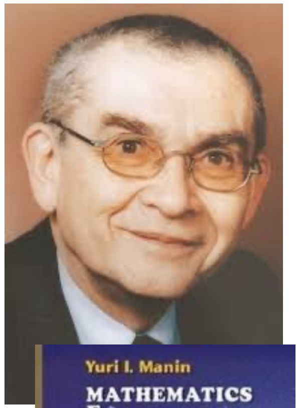
Quantum-enhanced applications
Quantum, hybrid quantum-classical, and physics-inspired classical algorithms
QC programming
Fundamental quantum physics mechanisms
Simulation tools
Analytical methods

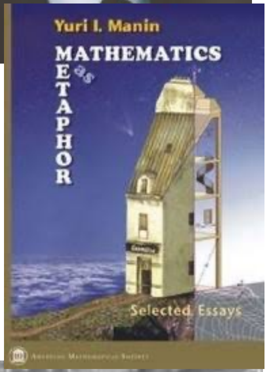
NASA Ames QuAIL team

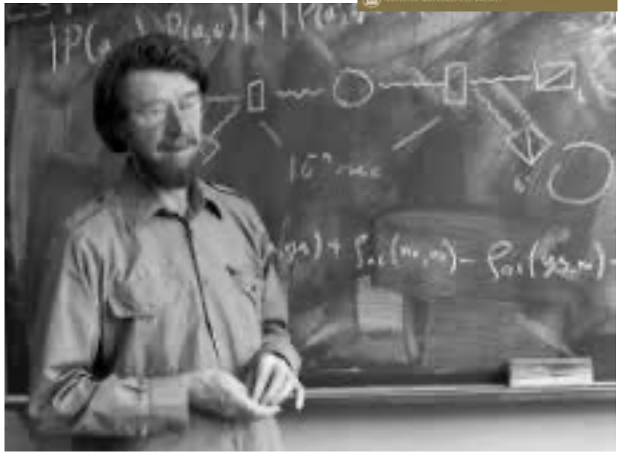
Birth of Quantum Computing

- **Feynman and Manin recognized in the early 1980s that certain quantum phenomena could not be simulated efficiently by a computer**
  - Phenomena related to quantum entanglement; Bell's inequality
- **Perhaps these quantum phenomena could be used to speed up more general computation?**





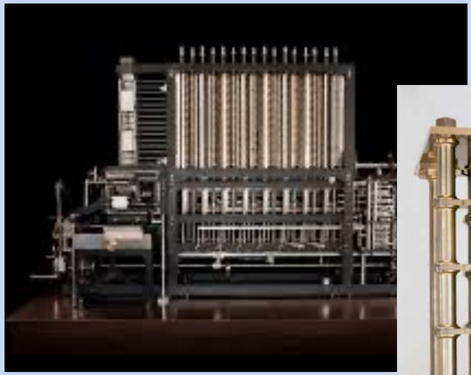
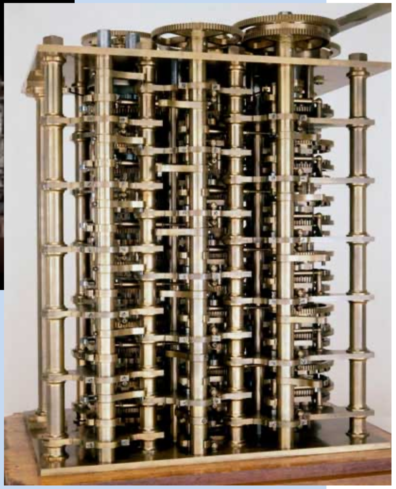




Computers as Classical Mechanical Machines

- **Babbage's analytical engine was a classical mechanical machine**
- **Turing machines**
  - The abstraction that underlies complexity theory and universal computing machines
  - Firmly rooted in classical mechanics
  - Described in classical mechanical terms
- **Abstraction allowed us ignore how classical computers are implemented physically**
  - When we program we don't think about the fundamental physics

- ***How do different models of physics affect how quickly we can compute?***

Babbage engine (Computer History Museum)

Computers as Quantum Mechanical Machines?

Fundamental questions

- **How do different models of physics affect how quickly we can compute?**
  - Suggests new computation-based physics principles
- **How would basing computation on a quantum mechanical model rather than a classical mechanical model change our notions of computing?**
  - Quantum physics is the physics of our universe
- **How quickly does nature allow us to compute?**

What a Quantum Computer is Not

- **Just because a computer uses quantum effects, does not mean it is a quantum computer**
  - All the computers in this building make use of quantum effects
  - The fundamental unit of computation, the bit, and the algorithms we design for computers did not change when quantum effects were used
- **A quantum computer has a fundamentally different way of encoding and processing information**
  - Quantum computers are quantum information processing devices
  - They process qubits instead of bits
  - They use quantum operations instead of logic gates
- **Also, just because a piece of hardware has a certain number of qubits, it isn’t necessarily a quantum computer**
  - A set of light switches, even a very large set, is not a classical computer

Certainty and Randomness in Quantum Computation

- **Any computation a classical computer can do, a quantum computer can do with roughly the same efficiency**
  - With the same probability of the outcome
  - If the classical computation is non-probabilistic, so is the quantum one
- **Like classical algorithms, some quantum algorithms are inherently probabilistic and others are not**
  - First quantum algorithms were not probabilistic
    - E.g. Deutsch-Jozsa algorithm solves problem with certainty that classical algorithms, of equivalent efficiency, could solve only with high probability
  - Shor’s algorithms are probabilistic
  - Grover’s is not intrinsically probabilistic
    - initial search algorithm was probabilistic, but
    - slight variants, which preserve the speed up, are non-probabilistic

Current Status of Quantum Algorithms

Quantum computing can do everything a classical computer can do

and

Provable quantum advantage known for a few dozen quantum algorithms

Unknown quantum advantage for everything else

Status of classical algorithms
- Provable bounds hard to obtain
  - Analysis is just too difficult
- Best classical algorithm not known for most problems
- Empirical evaluation required
- Ongoing development of classical heuristic approaches
  - Analyzed empirically: ran and see what happens
  - E.g. SAT, planning, machine learning, etc. competitions

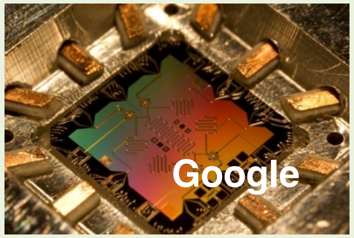
- **NISQ era supports unprecedented means for empirical analysis of quantum algorithms**
  - **Quantum heuristics come into their own**

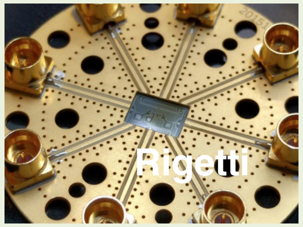
A handful of proven limitations on quantum computing

Conjecture: Quantum Heuristics will significantly broaden applications of quantum computing

Quantum Hardware

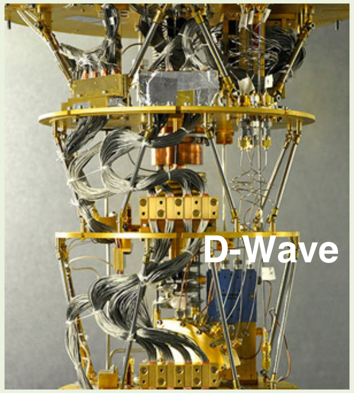
General Purpose:
Universal quantum processors

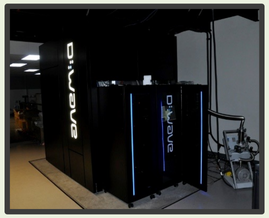




Special Purpose:
E.g. Quantum annealers

Noisy Intermediate-Scale Quantum (NISQ) devices





Superconducting processors
- *Google, IBM, Rigetti, Intel, …*

Trapped ion processors
- *IonQ, Honeywell, …*

Other approaches
- Optical
- Electron spins in silicon
- Topological, anyon based quantum computing

Number of qubits alone is not a good measure
- Analogy: billions of switches do not a classical computer make

Other key factors
- precision, speed, and generality of the control
  - particularly operations involving multiple qubits
- how long quantum coherence can be maintained
- stability over time
- speed with which processors can be calibrated

Quantum Computing has Entered the NISQ Era

Quantum supremacy has been achieved!

- Perform computations not possible on even the largest supercomputers in a reasonable amount of time

The international journal of science / 24 October 2019
nature
QUANTUM SUPREMACY
Classical supercomputer outperformed by quantum chip for the first time
Secret history
History of science
Bacterial activity

Cover article, Nature, 24 Oct 2019

Article

Quantum supremacy using a programmable superconducting processor

[illegible]

Google, NASA, ORNL collaboration

https://www.nature.com/articles/s41586-019-1666-5
https://www.nasa.gov/feature/ames/quantum-supremacy

… but not <u>useful</u> quantum supremacy.

- Currently too small to be useful for solving practical problems
- Early application to certified random number generation, but other applications require larger, more capable devices

Uses of these still limited, quantum devices?

(1) **Unprecedented opportunity** to explore and **evaluate algorithms**, both quantum and hybrid quantum-classical heuristic algorithms

(2) **Investigate quantum mechanisms** that may be harnessed for computational purposes

Insights gained feed into next generation

- **quantum algorithms**
- **quantum hardware**

One early target: Optimization

Other early targets: ML, Chem & Materials Simulation

Quantum Optimization Algs.: QA, AQO, QAOA

- **Common elements:**
  - Cost function: C(z)
  - Phase separation operator tied to the cost function,
    - Usually based on , but often including “penalty terms” to enforce constraints
  - Driver/Mixing operator
    - Most frequently, though we will see reasons to use other mixers

AQO (Adiabatic Q Opt)

- Evolution under $\boldsymbol{H(t)} = \boldsymbol{a(t)}H_C + \boldsymbol{b(t)}H_M$
- Slowly enough to stay in the ground subspace

QA (Q. Annealing)

- Evolution under $\boldsymbol{H(t)} = \boldsymbol{a(t)}H_C + \boldsymbol{b(t)}H_M$
- Many quick runs, thermal effect contribute

QAOA

- Alternate application of and
- For ***p*** alterations, the parameters are times/angles

Three Group Exercises

- **Before going on to a more technical part introducing the fundamentals of quantum computation**

Exercise 1

- **Which of the following best describes the current status of quantum algorithms?**
  a) Quantum algorithms can beat classical algorithms on every problem, we just need to build quantum computers on which to run them!
  b) Quantum algorithms have been studied since the early 1990s, and pretty much everything is known by now.
  c) While there are only a few dozen quantum algorithms known, quantum algorithms continue to be discovered, with many more algorithms likely to be identified as larger processors are built, enabling the evaluation of quantum heuristics.
  d) Quantum mechanics is the physics of the universe. Every algorithm is a quantum algorithm!

Exercise 2

- **Which statement best describes "quantum supremacy"?**
  a) "Quantum supremacy" was already achieved in the 1990s by Shor's algorithm, since it is a polynomial time algorithm whereas the best classical algorithms are superpolynomial time algorithms.
  b) It is well-known that quantum computers can beat classical computers, even supercomputers, at everything. "Quantum supremacy" is just a quick way of saying that.
  c) A quantum processor would demonstrate "Quantum supremacy" if it could perform in a practical amount of time a computation that could not be performed on even the world's largest supercomputers in a practical amount of time. It would be achieved even if it was demonstrated for only one computation and that computation was useless.
  d) "Quantum supremacy" will be achieved when quantum computers can run Shor's algorithm on cryptographically relevant numbers.

Exercise 3

- **Which statement best describes the relation between uncertainty and quantum algorithms?**
  a) Quantum mechanics is by nature uncertain—think the quantum uncertainty principle—so unlike classical algorithms, quantum algorithms are inherently probabilistic
  b) Classical algorithms can be translated to a form that can be run on quantum computers, so translations of classical algorithms that answer with certainty, still answer with certainty, but if an algorithm makes use of truly quantum effects, it cannot provide an answer with certainty
  c) Like classical algorithms, quantum algorithms fall in two categories, algorithms that provide an answer with certainty and probabilistic algorithms
  d) All algorithms, both quantum and classical, cannot provide a result with certainty—life is inherently uncertain.

Agenda

- **Part I: Quantum-computing fundamentals**
  - High-level motivation, history, and status
  - Qubits, multi-qubit states, and quantum measurement

Morning break

- **Part II: Circuit-model quantum computing**
  - Review of notation
  - Quantum gates and quantum circuits

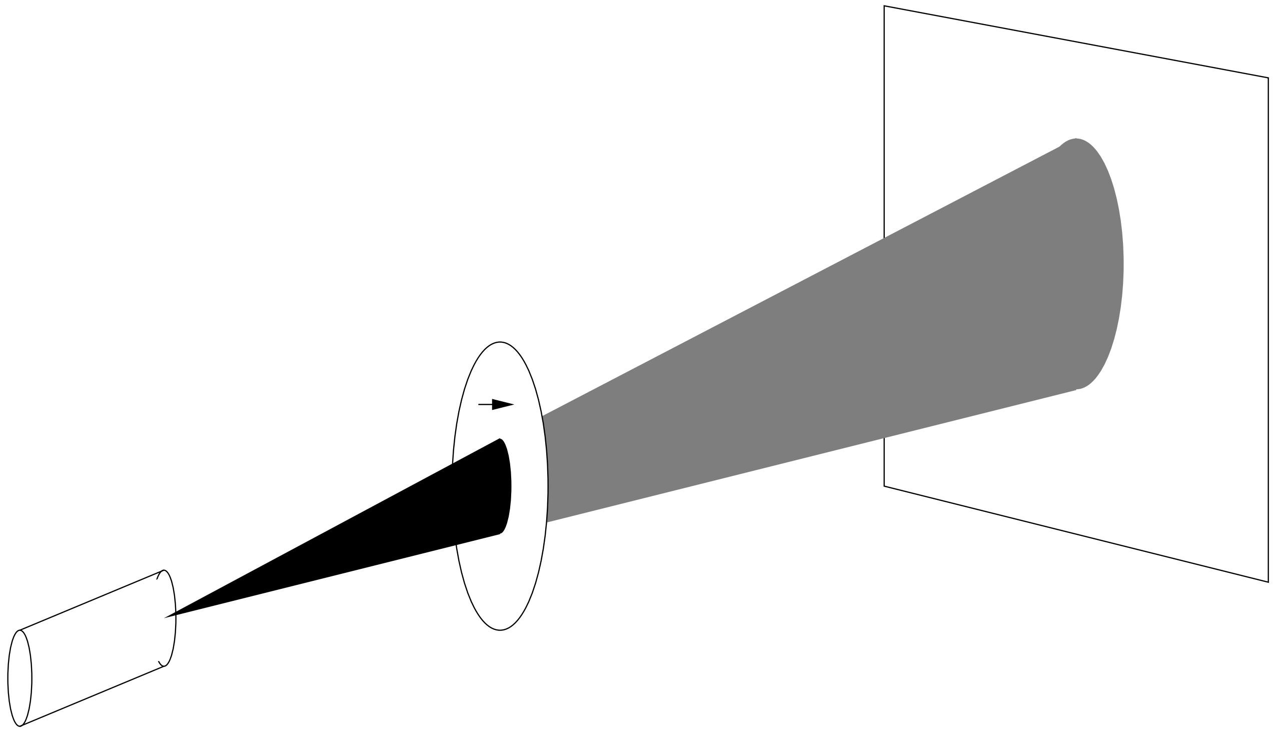
Lunch

  - Basic quantum algorithms
- **Part III: Quantum annealing**

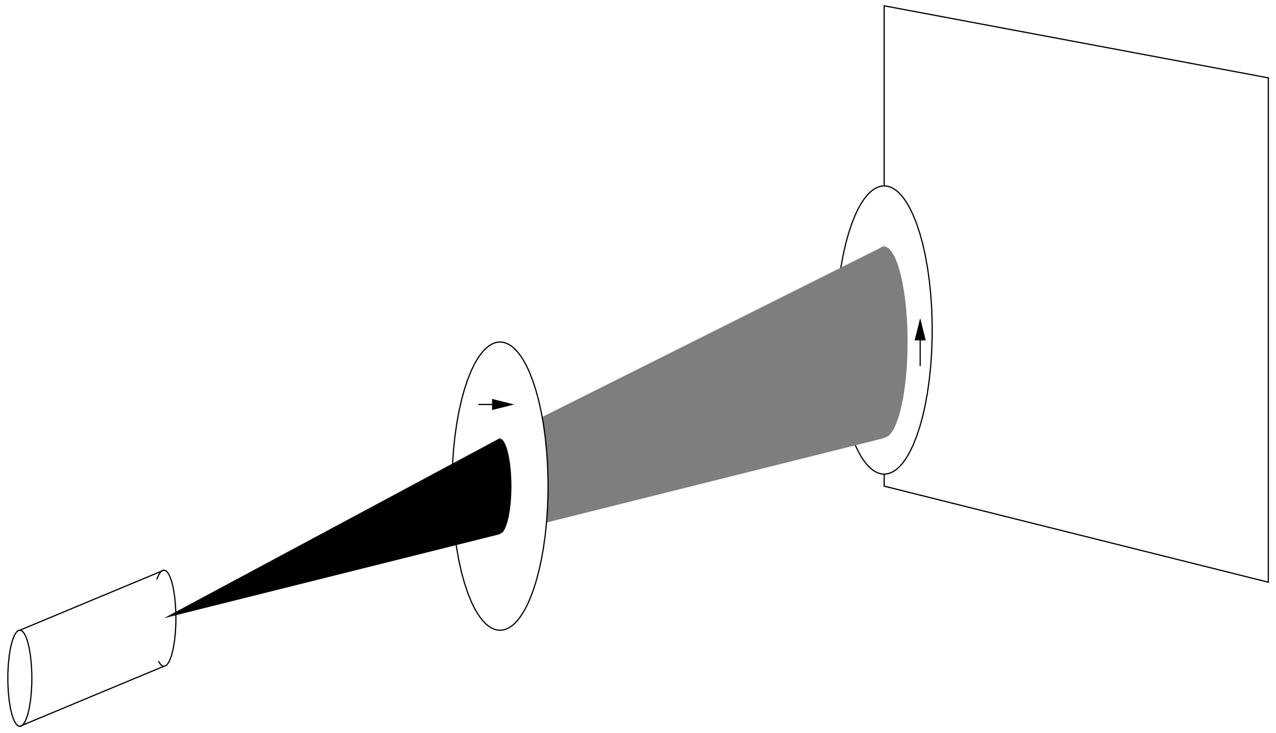
Afternoon break

- **Part IV: Further quantum algorithms**
  - Quantum Alternating Operator Ansatz (QAOA)
  - Advancements in quantum algorithms
  - Concluding remarks

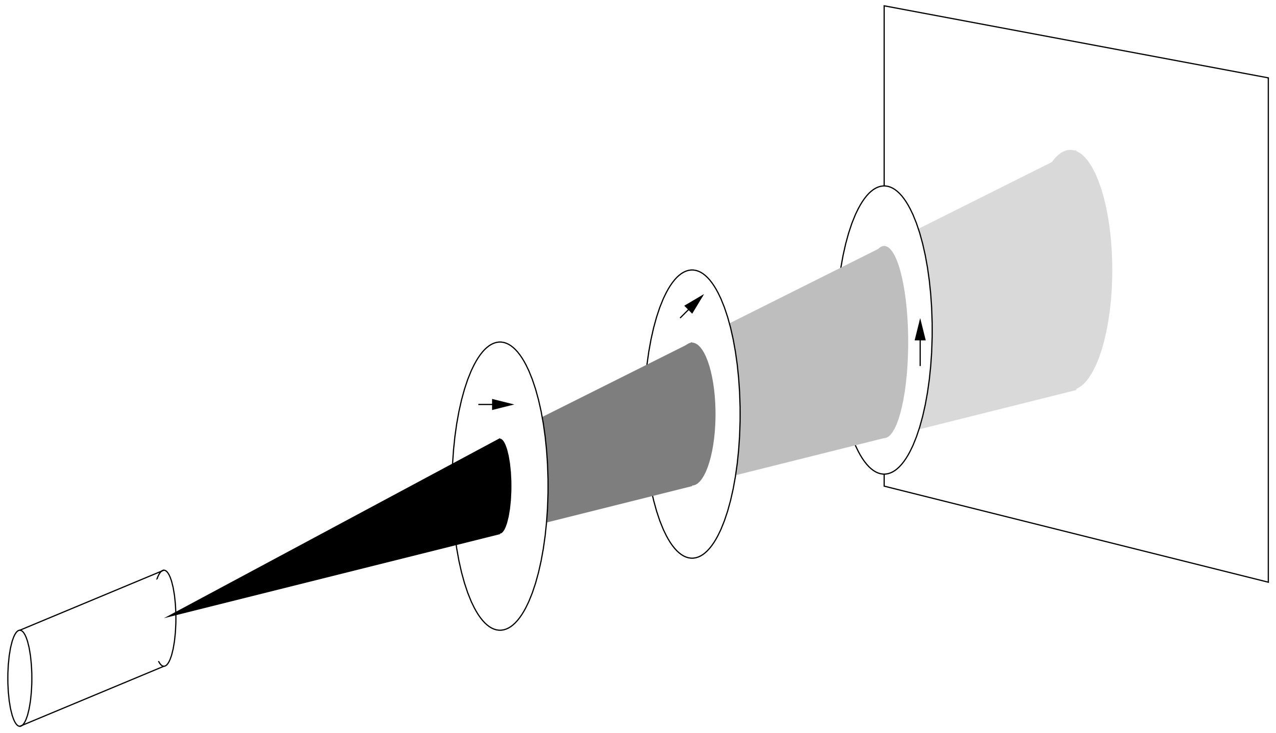
A Simple Experiment: Photon Polarization



A Simple Experiment: Photon Polarization



A Simple Experiment: Photon Polarization



Mathematically Representing Photon Polarization

Polarization state of a photon

- can be represented as a 2-dimensional vector of unit length

Taking horizontal $|{\rightarrow}\rangle$ and vertical $|{\uparrow}\rangle$ polarizations as a basis, an arbitrary polarization can be expressed as a superposition

$$|\psi\rangle = a|{\uparrow}\rangle + b|{\rightarrow}\rangle$$

with $|a|^2 + |b|^2 = 1$

(Allowing a and b to be complex numbers enables this formalism to describe circular polarization as well)

$|v\rangle$ is Dirac's notation for vectors. Means the same thing as $\vec{v}$ or $\mathbf{v}$, with v being the label for the vector

Measurement of Polarization

Polarization filters are quantum measuring devices

Quantum measurements always occur w.r.t. an orthogonal subspace decomposition associated with the measuring device

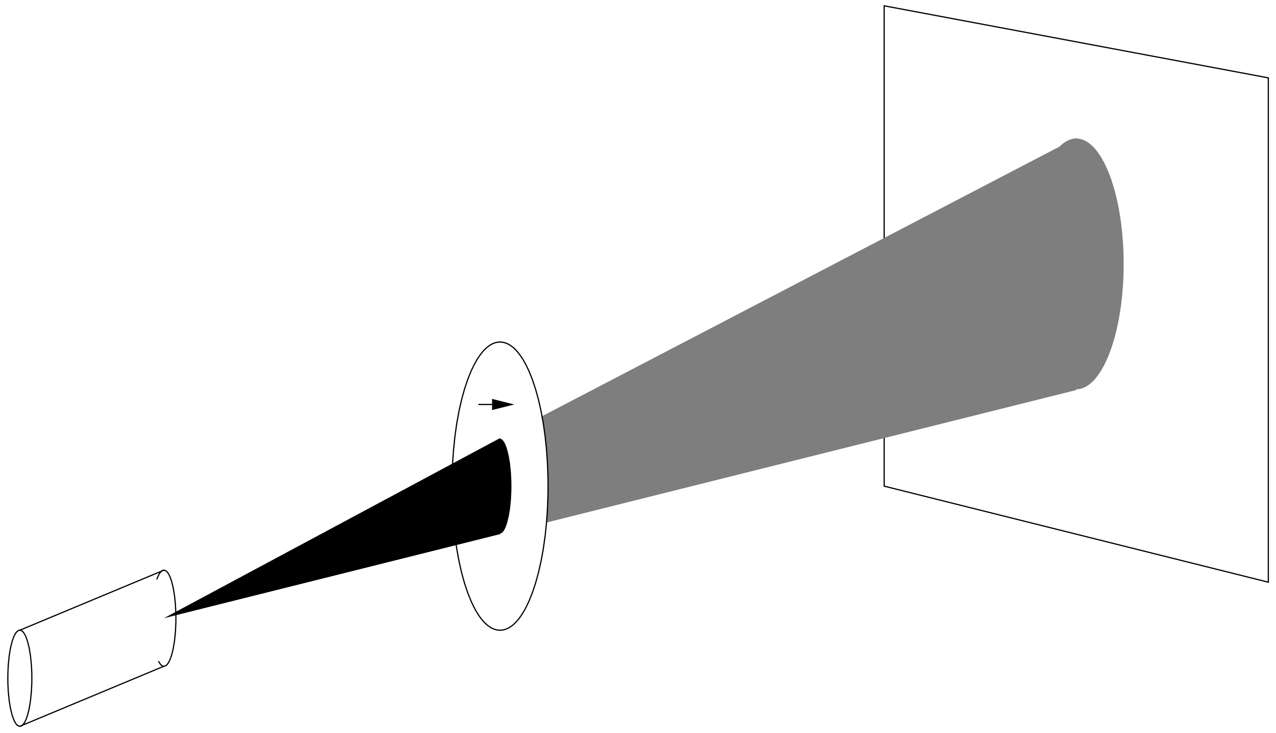
For a horizontal polarization filter, the basis in which it measures is $|{\rightarrow}\rangle$, together with its perpendicular $|{\uparrow}\rangle$

A photon with polarization $a|{\uparrow}\rangle + b|{\rightarrow}\rangle$ is measured by a horizontal filter
as $|{\uparrow}\rangle$ (absorbed) with probability $|a|^2$, and
$|{\rightarrow}\rangle$ (passed) with probability $|b|^2$

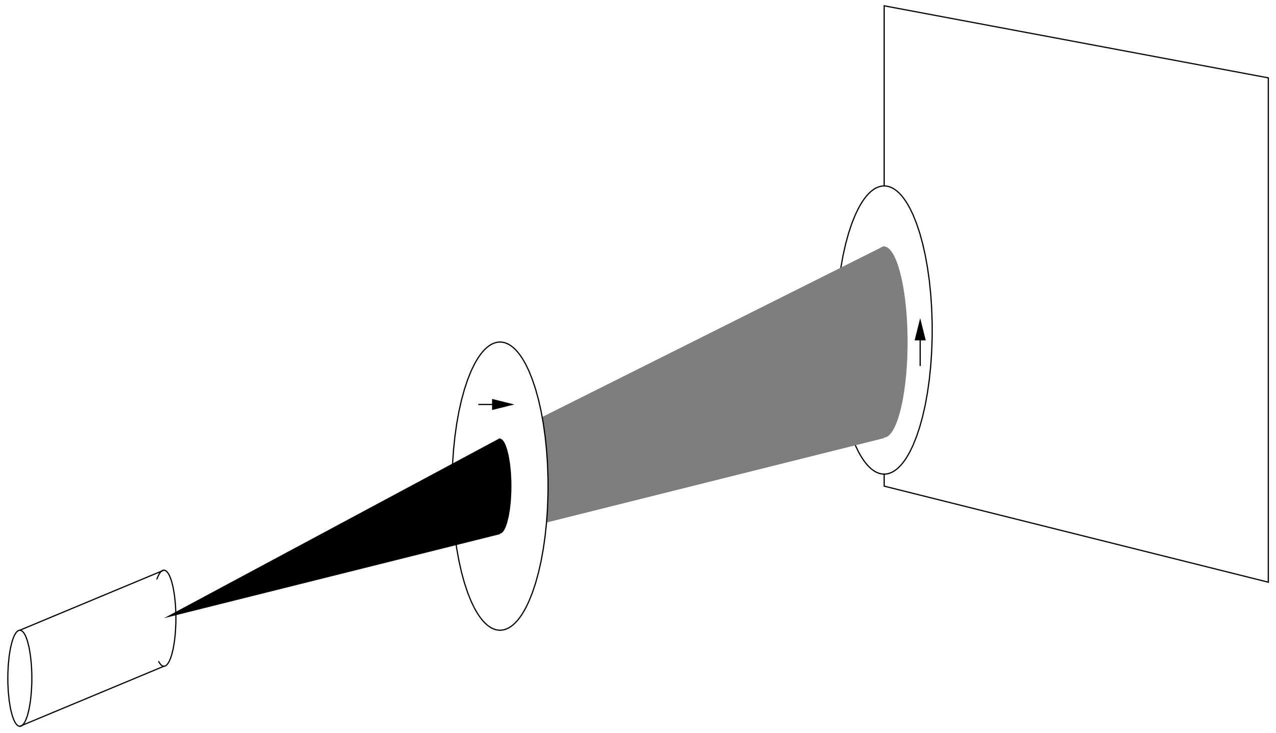
Any photon that has passed through the filter now has polarization $|{\rightarrow}\rangle$.

Polarization filters at other angles work in a similar way

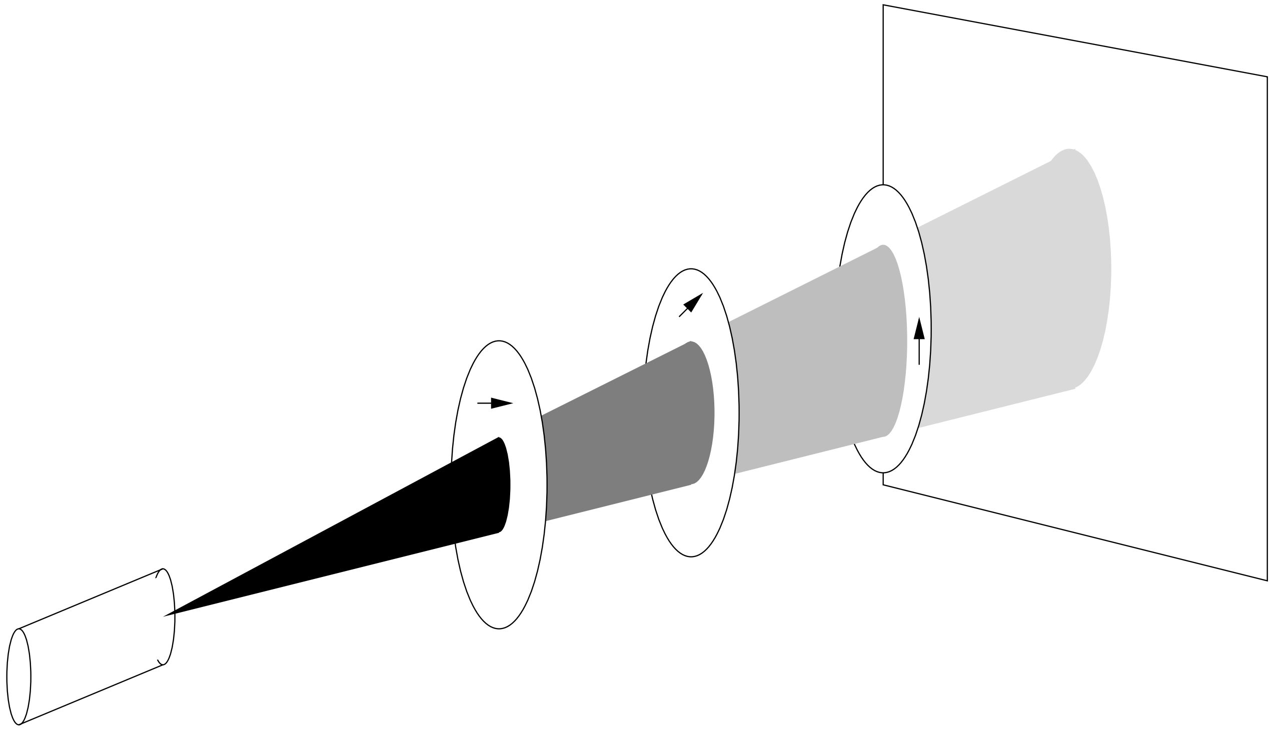
The Photon Polarization Experiment Revisited



The Photon Polarization Experiment Revisited



The Photon Polarization Experiment Revisited



Qubits (Quantum Bits)

Think polarization states of a photon!

Any 2-dimensional quantum system can be viewed as the fundamental unit of quantum computation, a *quantum bit* or *qubit*.
Qubit state space is a 2-dimensional complex vector space

A computational basis is chosen, denoted $|0\rangle$ and $|1\rangle$, and used to encode classical bit values 0 and 1

Possible qubit values $a|0\rangle + b|1\rangle$, for complex a, b with $|a|^2 + |b|^2 = 1$.

Unlike classical bits, qubits can be in superposition states such as $\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$ or $\frac{1}{\sqrt{2}}(|0\rangle - i|1\rangle)$

Measurement of Single Qubits

Measuring qubit $a|0\rangle + b|1\rangle$ in the computational basis $\{|0\rangle, |1\rangle\}$

- returns 0 with probability $|a|^2$
- returns 1 with probability $|b|^2$
- projects to state to the basis state corresponding to the measurement result

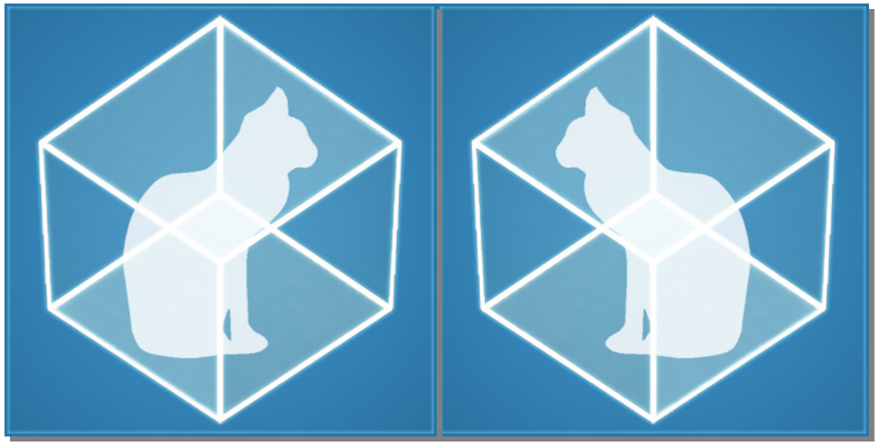
A qubit can be measured with respect to any orthogonal basis for its 2-dimensional state space

Only one classical bit of information can be extracted from one qubit

No cloning theorem: An unknown quantum state cannot be reliably copied

Multiple Qubits

- Qubits combine like quantum particles not classical objects
  - Quantum states combine via tensor products not direct products
  - The quantum state space, the space of possible states of n quantum particles, is exponentially larger than that of n classical objects
  - 2^n instead of 2n
- Entangled states make up the bulk of this space
- No classical analog: The state of entangled multiple particle systems cannot be described in terms of the states of the individual particles



High-level View of How State Spaces Combine

Let X be a vector space with basis $\{|\alpha_1\rangle, \ldots, |\alpha_n\rangle\}$ and Y be a vector space with basis $\{|\beta_1\rangle, \ldots, |\beta_m\rangle\}$

Classical state spaces combine via the **Cartesian product**

$X \times Y$ has basis
$\{|\alpha_1\rangle, \ldots, |\alpha_n\rangle, |\beta_1\rangle, \ldots, |\beta_m\rangle\}$

$$\begin{aligned} \dim(X \times Y) &= \dim(X) + \dim(Y) \\ &= n + m \end{aligned}$$

Quantum state spaces combine via the **tensor product**

$X \otimes Y$ has basis
$\{|\alpha_1\rangle \otimes |\beta_1\rangle, |\alpha_1\rangle \otimes |\beta_2\rangle, \ldots, |\alpha_n\rangle \otimes |\beta_m\rangle\}$

$$\begin{aligned} \dim(X \otimes Y) &= \dim(X) * \dim(Y) \\ &= n * m \end{aligned}$$

Scott will go over the mathematics and notation here in more detail in the next segment of the tutorial.

Measurement of Single Qubits

Measuring qubit $a|0\rangle + b|1\rangle$ in the computational basis $\{|0\rangle, |1\rangle\}$

- returns 0 with probability $|a|^2$
- returns 1 with probability $|b|^2$
- projects to state to the basis state corresponding to the measurement result

A qubit can be measured with respect to any orthogonal basis for its 2-dimensional state space

Only one classical bit of information can be extracted from one qubit

No cloning theorem: An unknown quantum state cannot be reliably copied

Exponential State Space

The quantum state of an n qubit system is a vector in a 2^n-dimensional space

If B is the state space of a single qubit spanned by $\{|0\rangle, |1\rangle\}$, then a 2-qubit system $B \otimes B$ has basis

$$\{|0\rangle \otimes |0\rangle, |0\rangle \otimes |1\rangle, |1\rangle \otimes |0\rangle, |1\rangle \otimes |1\rangle\},$$

often written

$$\{|00\rangle, |01\rangle, |10\rangle, |11\rangle\},$$

The standard computational basis for the 2^n-dimensional complex vector space $B \otimes B \dots B \otimes B$ of an n qubit system is

$$\{|00\dots00\rangle, |00\dots01\rangle, \dots, |11\dots10\rangle, |11\dots11\rangle\}$$

We'll use the notation $|5\rangle = |101\rangle$ when n is understood.

Entangled States

Entangled states cannot be written as tensor product of independent qubits

Example: An EPR pair $\frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$

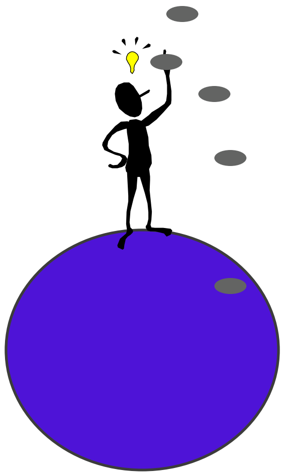
$$\begin{array}{rl} & (a_0|0\rangle + b_0|1\rangle) \otimes (a_1|0\rangle + b_1|1\rangle) \\ = & a_0 a_1|00\rangle + a_0 b_1|01\rangle + b_0 a_1|10\rangle + b_0 b_1|11\rangle \\ \neq & a_0 a_1|00\rangle + 0|01\rangle + 0|10\rangle + b_0 b_1|11\rangle \\ = & \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle) \end{array}$$

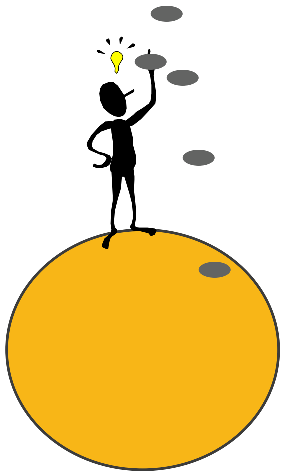
- Measurement of the first qubit yields either $|0\rangle$ or $|1\rangle$
- Measurement changes state to either $|00\rangle$ or $|11\rangle$
- Measurement of second qubit gives same result as first

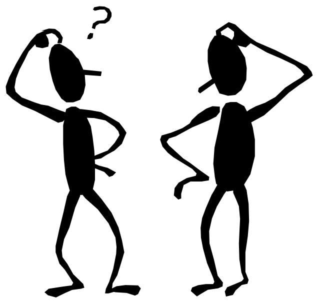
Similar results when measuring in other bases

Entanglement

- Two people see completely random results from coin tosses
- Completely correlated results!
- But no way to communicate
- Different relativistic frames disagree about who flipped the coin first





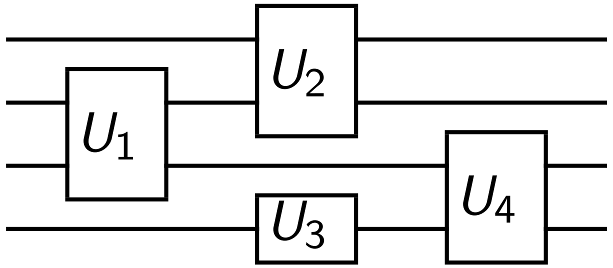


Quantum Computer (Circuit Model)

A quantum computation consists of

- initialization of n-qubit register ($|\psi\rangle$)
- quantum state transformation of register
  - sequence of primitive (1- or 2-qubit) operations (gates) U_i that collectively perform the transformation of the register
- measurement of some or all of the qubits of the register
- classical control throughout to
  - program which quantum steps to carry out
  - interpret results of quantum measurement

$$(I \otimes I \otimes U_4)(U_2 \otimes I \otimes U_3)(I \otimes U_1 \otimes I)|\psi\rangle$$



Notation

Agenda

- **Part I: Quantum-computing fundamentals**
  - High-level motivation, history, and status
  - Qubits, multi-qubit states, and quantum measurement

Morning break

- **Part II: Circuit-model quantum computing**
  - Review of notation
  - Quantum gates and quantum circuits

Lunch

  - Basic quantum algorithms
- **Part III: Quantum annealing**

Afternoon break

- **Part IV: Further quantum algorithms**
  - Quantum Alternating Operator Ansatz (QAOA)
  - Advancements in quantum algorithms
  - Concluding remarks

Tensor Products

- **The tensor product, $\otimes$, multiplies two vectors to produce a longer vector or two matrices to produce a larger matrix**
  - Unlike dot products or matrix multiplication, the two arguments do not have to have compatible dimensions
- **Operational semantics (loosely specified)**
  - Multiply each scalar on the left-hand-side vector/matrix by the entire right-hand-side vector/matrix
- **Vector example**
  - $\begin{pmatrix} a \\ b \end{pmatrix} \otimes \begin{pmatrix} c \\ d \end{pmatrix} = \begin{pmatrix} ac \\ ad \\ bc \\ bd \end{pmatrix}$, e.g., $\begin{pmatrix} 1 \\ 2 \end{pmatrix} \otimes \begin{pmatrix} 3 \\ 4 \end{pmatrix} = \begin{pmatrix} 1 \cdot 3 \\ 1 \cdot 4 \\ 2 \cdot 3 \\ 2 \cdot 4 \end{pmatrix} = \begin{pmatrix} 3 \\ 4 \\ 6 \\ 8 \end{pmatrix}$
- **Matrix example**
  - $\begin{pmatrix} a & b \\ c & d \end{pmatrix} \otimes \begin{pmatrix} e & f \\ g & h \end{pmatrix} = \begin{pmatrix} ae & af & be & bf \\ ag & ah & bg & bh \\ ce & cf & de & df \\ cg & ch & dg & dh \end{pmatrix}$, e.g., $\begin{pmatrix} 1 & 2 \\ 2 & 1 \end{pmatrix} \otimes \begin{pmatrix} 3 & 1 \\ 1 & 4 \end{pmatrix} = \begin{pmatrix} 3 & 1 & 6 & 2 \\ 1 & 4 & 2 & 8 \\ 6 & 2 & 3 & 1 \\ 2 & 8 & 1 & 4 \end{pmatrix}$

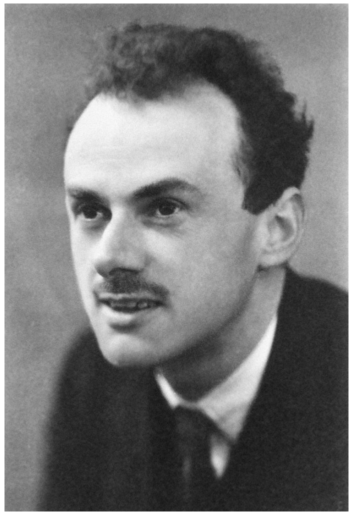
Basics of Dirac (a.k.a. Bra-Ket) Notation

- **Two components: bras and kets**

$$\langle\psi| \qquad\qquad |\psi\rangle$$

"Bra" *"Ket"*

Row vector (adjoint) Column vector



Paul Dirac
1902–1984

- **The label (e.g., "ψ") is merely a name and has no inherent meaning**
- **However, some conventions exist:**

$$|0\rangle \equiv \binom{1}{0} \qquad |1\rangle \equiv \binom{0}{1} \qquad |+\rangle \equiv \frac{1}{\sqrt{2}}\binom{1}{1} \qquad |-\rangle \equiv \frac{1}{\sqrt{2}}\binom{1}{-1}$$

Hence, e.g.,

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

- **Bra times ket: $\langle\psi|\phi\rangle$**
  - Inner product
  - Returns a scalar
- **Ket times bra: $|\phi\rangle\langle\psi|$**
  - Outer product
  - Returns a matrix

More on Dirac Notation

- **Ket-kets and bra-bras**
  - $|a\rangle|b\rangle$ includes an implicit tensor product: $|a\rangle \otimes |b\rangle$
  - We routinely simplify this even further to just $|ab\rangle$
  - Example: Given that $|0\rangle \equiv \begin{pmatrix}1\\0\end{pmatrix}$ and $|1\rangle \equiv \begin{pmatrix}0\\1\end{pmatrix}$, then $|01\rangle = |0\rangle \otimes |1\rangle = \begin{pmatrix}0\\1\\0\\0\end{pmatrix}$
- **Simple cases**
  - Given two orthogonal kets $|\uparrow\rangle$ and $|\downarrow\rangle$ that are each normalized (this is typical), $\langle\uparrow|\uparrow\rangle = \langle\downarrow|\downarrow\rangle = 1$ and $\langle\uparrow|\downarrow\rangle = \langle\downarrow|\uparrow\rangle = 0$
- **Convenient way to reason above linear transformations**
  - $|\mathrm{out}\rangle\langle\mathrm{in}|$ is an operator that maps $|\mathrm{in}\rangle$ to $|\mathrm{out}\rangle$ (i.e., by left multiplication) and anything orthogonal to $|\mathrm{in}\rangle$ to a zero vector: $|\mathrm{out}\rangle\langle\mathrm{in}|\mathrm{in}\rangle = |\mathrm{out}\rangle$; $|\mathrm{out}\rangle\langle\mathrm{in}|\mathrm{out}\rangle = \mathbf{0}$
- **Distributive properties**
  - Example: assuming $|x\rangle \perp |y\rangle$ and $\| |x\rangle \| = \| |y\rangle \| = 1$, $(|x\rangle\langle y| - i|y\rangle\langle x|)\,|x\rangle = |x\rangle\langle y|x\rangle - i|y\rangle\langle x|x\rangle = |x\rangle \cdot 0 - i|y\rangle \cdot 1 = -i|y\rangle$
  - Also, $|a\rangle\langle b| \otimes |c\rangle\langle d| = |ac\rangle\langle bd|$

Why Use Dirac Notation?

- **Dirac notation makes it easier to reason about state transformations and probability amplitudes than it is when working with matrices**
- **Example**
  - What are the eigenvectors (quantum states) of the matrix

$$M \equiv \begin{pmatrix} \frac{1}{4} & \frac{\sqrt{3}}{4} \\ \frac{\sqrt{3}}{4} & \frac{3}{4} \end{pmatrix}$$

    that correspond to non-zero eigenvalues (probability amplitudes)? That is, what vector is taken to itself by left-multiplying by that matrix?
  - This is a *much* easier problem when written with bras and kets:

$$\text{Let } |w\rangle \equiv \frac{1}{2}|0\rangle + \frac{\sqrt{3}}{2}|1\rangle \text{ and } M \equiv |w\rangle\langle w|$$

  - Clearly, $|w\rangle$ itself is the eigenvector we're looking for because $|w\rangle\langle w|$ maps $|w\rangle$ to itself with probability amplitude 1: $|w\rangle\langle w|w\rangle = |w\rangle \cdot 1 = |w\rangle$

Notation Exercises

- **For the following exercises,**
  - Define $|w\rangle \equiv \frac{1}{2}|0\rangle + \frac{\sqrt{3}}{2}|1\rangle$, as on the previous slide
  - Please work with Dirac notation—don't convert to explicit vectors/matrices
- **Exercise 1: Evaluate $\langle \boldsymbol{w}|\boldsymbol{w}\rangle$**
  - Remember: This is an inner product so you should wind up with a scalar value
- **Exercise 2: Expand $|\boldsymbol{w}\rangle\langle \boldsymbol{w}|$**
  - Remember: This is an outer product so you should wind up with a matrix (represented as a sum of ket-bras)
- **Exercise 3: Apply $|\boldsymbol{w}\rangle\langle \boldsymbol{w}|$ to $|\boldsymbol{w}\rangle$**
  - Remember: This is a matrix times a vector so you should wind up with a vector
- **Exercise 4: Expand $|\boldsymbol{ww}\rangle$**
  - Remember: $|ww\rangle \equiv |w\rangle|w\rangle \equiv |w\rangle \otimes |w\rangle$ so you should wind up with a vector
- **Challenge exercise:**
  - Which of the following two states is entangled and which is separable?

$$\frac{1}{2}(|00\rangle + |01\rangle + |10\rangle - |11\rangle) \quad \text{vs.} \quad \frac{1}{2}(|00\rangle - |01\rangle - |10\rangle + |11\rangle)$$

  - Remember: Entangled states cannot be written as $(a_0|0\rangle + a_1|1\rangle) \otimes (b_0|0\rangle + b_1|1\rangle)$

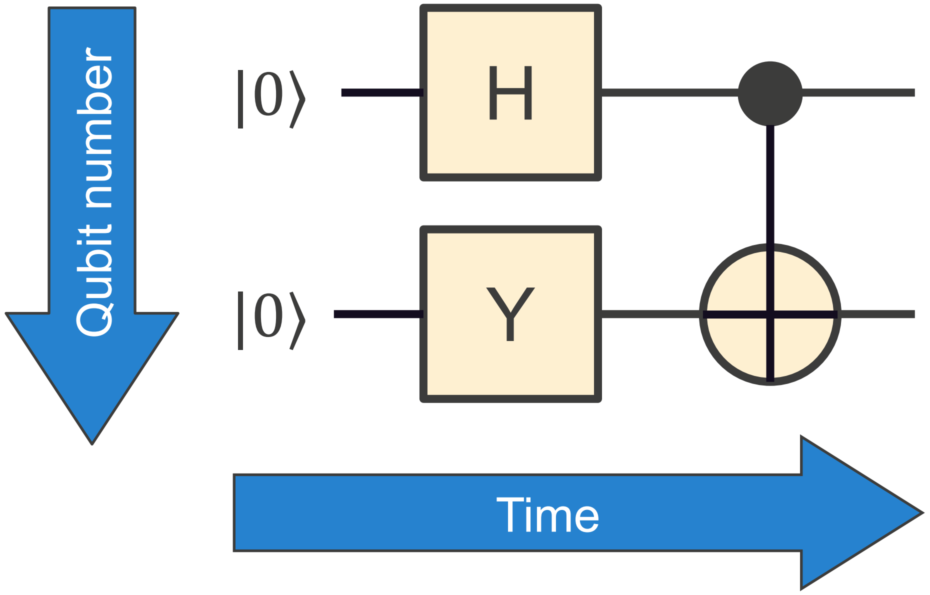
Notation Exercises: Solutions

- **Exercise 1: Evaluate $\langle \boldsymbol{w}|\boldsymbol{w}\rangle$**
  - $\langle w|w\rangle = \left(\frac{1}{2}\langle 0| + \frac{\sqrt{3}}{2}\langle 1|\right)\left(\frac{1}{2}|0\rangle + \frac{\sqrt{3}}{2}|1\rangle\right) = \frac{1}{4}\langle 0|0\rangle + \frac{\sqrt{3}}{4}\langle 0|1\rangle + \frac{\sqrt{3}}{4}\langle 1|0\rangle + \frac{3}{4}\langle 1|1\rangle = \frac{1}{4}\cdot 1 + \frac{\sqrt{3}}{4}\cdot 0 + \frac{\sqrt{3}}{4}\cdot 0 + \frac{3}{4}\cdot 1 = 1$
- **Exercise 2: Expand $|\boldsymbol{w}\rangle\langle \boldsymbol{w}|$**
  - $|w\rangle\langle w| = \left(\frac{1}{2}|0\rangle + \frac{\sqrt{3}}{2}|1\rangle\right)\left(\frac{1}{2}\langle 0| + \frac{\sqrt{3}}{2}\langle 1|\right) = \frac{1}{4}|0\rangle\langle 0| + \frac{\sqrt{3}}{4}|0\rangle\langle 1| + \frac{\sqrt{3}}{4}|1\rangle\langle 0| + \frac{3}{4}|1\rangle\langle 1|$
- **Exercise 3: Apply $|\boldsymbol{w}\rangle\langle \boldsymbol{w}|$ to $|\boldsymbol{w}\rangle$**
  - Hard way: $(|w\rangle\langle w|)|w\rangle = \left(\frac{1}{4}|0\rangle\langle 0| + \frac{\sqrt{3}}{4}|0\rangle\langle 1| + \frac{\sqrt{3}}{4}|1\rangle\langle 0| + \frac{3}{4}|1\rangle\langle 1|\right)\left(\frac{1}{2}|0\rangle + \frac{\sqrt{3}}{2}|1\rangle\right) = \left(\frac{1}{8}|0\rangle + 0 + \frac{\sqrt{3}}{8}|1\rangle + 0\right) + \left(0 + \frac{3}{8}|0\rangle + 0 + \frac{3\sqrt{3}}{8}|1\rangle\right) = \frac{1}{2}|0\rangle + \frac{\sqrt{3}}{2}|1\rangle$
  - Easy way: $|w\rangle(\langle w|w\rangle) = |w\rangle \cdot 1 = |w\rangle = \frac{1}{2}|0\rangle + \frac{\sqrt{3}}{2}|1\rangle$
- **Exercise 4: Expand $|\boldsymbol{ww}\rangle$**
  - $|ww\rangle = \left(\frac{1}{2}|0\rangle + \frac{\sqrt{3}}{2}|1\rangle\right)\left(\frac{1}{2}|0\rangle + \frac{\sqrt{3}}{2}|1\rangle\right) = \frac{1}{4}|00\rangle + \frac{\sqrt{3}}{4}|01\rangle + \frac{\sqrt{3}}{4}|10\rangle + \frac{3}{4}|11\rangle$

Notation Exercises: Solutions (cont.)

- **Challenge exercise: Entangled or separable?**
- **First state**
  - Can we express $\frac{1}{2}(|00\rangle + |01\rangle + |10\rangle - |11\rangle)$ as $(a_0|0\rangle + a_1|1\rangle) \otimes (b_0|0\rangle + b_1|1\rangle)$?
  - Multiplying out the latter gives us $a_0b_0|00\rangle + a_0b_1|01\rangle + a_1b_0|10\rangle + a_1b_1|11\rangle$
  - Goal is therefore to solve $\{a_0b_0 = 1, a_0b_1 = 1, a_1b_0 = 1, a_1b_1 = -1\}$ for $\{a_0, a_1, b_0, b_1\}$
  - Let's express all variables in terms of a_0. By the first equation, $b_0 = \frac{1}{a_0}$. By the second equation, $b_1 = \frac{1}{a_0}$. By the third equation, $a_1 = \frac{1}{b_0} = a_0$. But by the fourth equation, $b_1 = \frac{-1}{a_1} = \frac{-1}{a_0}$. Because $\frac{1}{a_0}$ cannot equal $\frac{-1}{a_0}$, we cannot express the state as $(a_0|0\rangle + a_1|1\rangle) \otimes (b_0|0\rangle + b_1|1\rangle)$ and it is therefore entangled.
- **Second state**
  - Can we express $\frac{1}{2}(|00\rangle - |01\rangle - |10\rangle + |11\rangle)$ as $(a_0|0\rangle + a_1|1\rangle) \otimes (b_0|0\rangle + b_1|1\rangle)$?
  - Goal is to solve $\{a_0b_0 = 1, a_0b_1 = -1, a_1b_0 = -1, a_1b_1 = 1\}$ for $\{a_0, a_1, b_0, b_1\}$
  - Let's express all variables in terms of a_0. By the first equation, $b_0 = \frac{1}{a_0}$. By the second equation, $b_1 = \frac{-1}{a_0}$. By the third equation, $a_1 = \frac{-1}{b_0} = -a_0$. The fourth equation confirms that $a_1 = \frac{1}{b_1} = -a_0$. Hence, $\{a_0, a_1, b_0, b_1\} = \{1, -1, 1, -1\}$ (remember: $|a_i|^2, |b_i|^2 \in [0,1]$) and the state is therefore separable.

The Circuit Model of Quantum Computing



- **A labelled box represents single-qubit operators (2×2 matrix)**
- **Symbol–vertical line–symbol represents a two-qubit operator (4×4 matrix)**
- **A quantum circuit is really just a piecewise representation of an enormous unitary matrix ($2^n \times 2^n$ for an *n*-qubit system)**
  - Above: $(CNOT)(H \otimes Y) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \left(\frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \otimes \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix} \right) = \frac{1}{\sqrt{2}} \begin{pmatrix} 0 & -i & 0 & -i \\ i & 0 & i & 0 \\ i & 0 & -i & 0 \\ 0 & -i & 0 & i \end{pmatrix}$

Mathematical Forms Commonly Encountered in QC

- **Magnitude of a complex number, | · |**
  - $|a + bi| \equiv \sqrt{a^2 + b^2}$
- **Vector and matrix adjoint, $A^\dagger$**
  - Complex-conjugate transpose
  - $\begin{pmatrix} a + bi \\ c + di \end{pmatrix}^\dagger \equiv \begin{pmatrix} a - bi & c - di \end{pmatrix}$
  - $\begin{pmatrix} a + bi & c + di \\ e + fi & g + hi \end{pmatrix}^\dagger \equiv \begin{pmatrix} a - bi & e - fi \\ c - di & g - hi \end{pmatrix}$
- **Matrix types**
  - Hermitian: $A = A^\dagger$
  - Unitary: $A^\dagger A = AA^\dagger = I$
- **Matrix exponentials**
  - For square matrix A, $e^A = \sum_{i=0}^{\infty} \frac{1}{k!} A^k$
  - In the above, $A^0 \equiv I$ for the I with the same dimensions as A

The Circuit Model

Agenda

- **Part I: Quantum-computing fundamentals**
  - High-level motivation, history, and status
  - Qubits, multi-qubit states, and quantum measurement

Morning break

- **Part II: Circuit-model quantum computing**
  - Review of notation
  - Quantum gates and quantum circuits

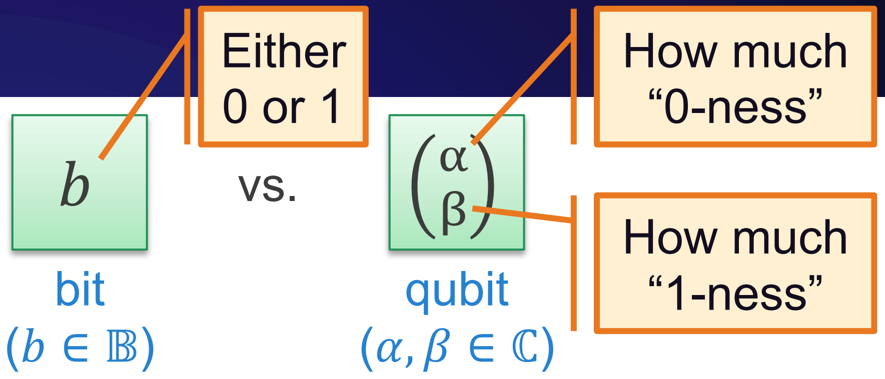
Lunch

  - Basic quantum algorithms
- **Part III: Quantum annealing**

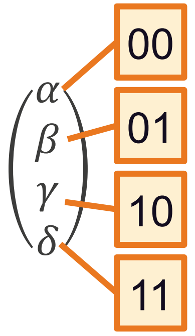
Afternoon break

- **Part IV: Further quantum algorithms**
  - Quantum Alternating Operator Ansatz (QAOA)
  - Advancements in quantum algorithms
  - Concluding remarks

Reminders



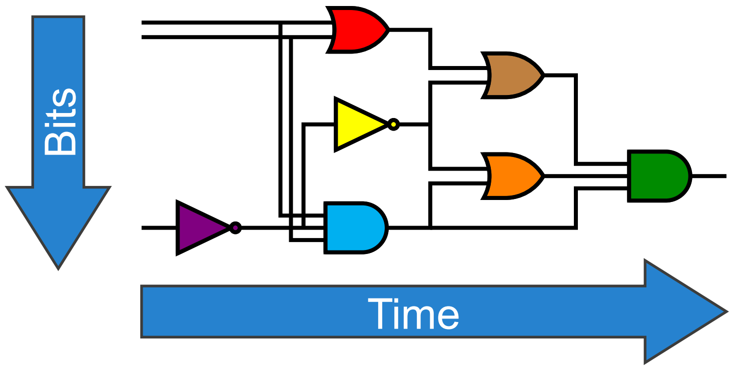
- **Unit of information**
  - Classical: Single bit, b
  - Quantum: Complex 2-vector, $|\psi\rangle = \binom{\alpha}{\beta}$
- **Measurement**
  - Measuring a qubit forces it to either 0 or 1
- **Superposition**
  - If qubit $|\psi\rangle = \binom{\alpha}{\beta} = \alpha\binom{1}{0} + \beta\binom{0}{1} = \alpha|0\rangle + \beta|1\rangle$, then it will be measured as 0 with probability $|\alpha|^2$ and as 1 with probability $|\beta|^2$
- **Multiple-qubit representation**
  - A two-qubit state is a complex 4-vector $|pq\rangle = \alpha|00\rangle + \beta|01\rangle + \gamma|10\rangle + \delta|11\rangle = \begin{pmatrix}\alpha\\ \beta\\ \gamma\\ \delta\end{pmatrix}$
  - An *n*-qubit state is a complex 2^n-vector

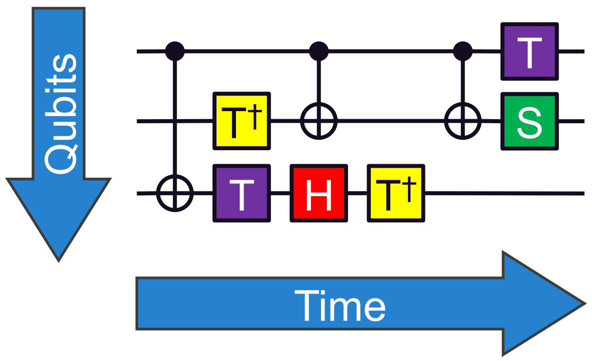


- **Entanglement**
  - The qubits in a two-qubit state are *entangled* if they can't be factored into $|p\rangle \otimes |q\rangle$
  - Example: $\frac{1}{2}(1 \quad -1 \quad 1 \quad -1)^{\mathsf{T}}$ can be factored into $\frac{1}{\sqrt{2}}\binom{1}{1} \otimes \frac{1}{\sqrt{2}}\binom{1}{-1}$ ($\therefore$ not entangled), but $\frac{1}{\sqrt{2}}(0 \quad 1 \quad 1 \quad 0)^{\mathsf{T}}$ cannot be factored ($\therefore$ entangled)

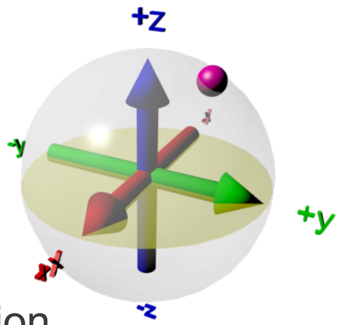
Basic Circuit-Model Concepts

- **Analogy to classical, digital circuits**



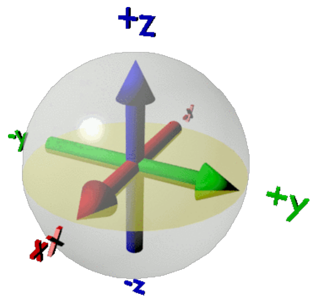


- **Differences**
  - Quantum circuits must be reversible (implication: same number of inputs and outputs for each gate and for the circuit as a whole)
  - Only combinational, not sequential, logic
- **Key point**
  - Abstract model of the operators to be applied—software not hardware
- **A qubit's state can be considered a point on the unit sphere**
- **Programmers manually control quantum effects**
  - Superposition: This qubit should be rotated by this amount in this direction
  - Entanglement (loosely): This qubit should conditionally rotate that qubit



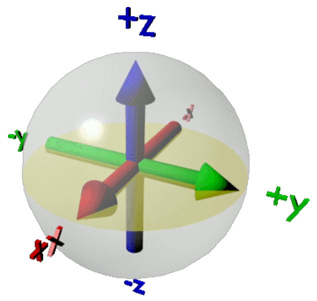
Manipulating Quantum States

- **Apply *operators* (a.k.a. *quantum gates*)**
  - Unitary matrices (corollary: all operations are reversible)
  - 2×2 for single-qubit gates, 4×4 for double-, 8×8 for triple-, etc.
- **Examples of single-qubit gates**
  - *X*, a.k.a. Pauli *x*, a.k.a. σ_x, a.k.a. NOT rotates by *π* radians around the *x* axis; it flips $|0\rangle \leftrightarrow |1\rangle$
  - *Y*, a.k.a. Pauli *y*, a.k.a. σ_y rotates by *π* radians around the *y* axis
  - *Z*, a.k.a. Pauli *z*, a.k.a. σ_z rotates by *π* radians around the *z* axis
  - Note that $XX = YY = ZZ = I$
- **A rotation in any direction by any amount is a gate**
  - Example: $\sqrt{\mathrm{NOT}}$ rotates by *π*/2 radians around the *x* axis



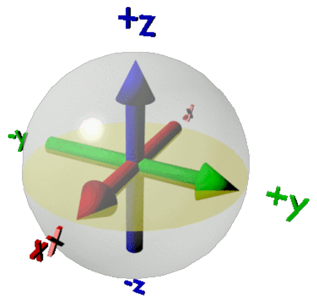
$$\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$$

Pauli *x* gate



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

Pauli *y* gate

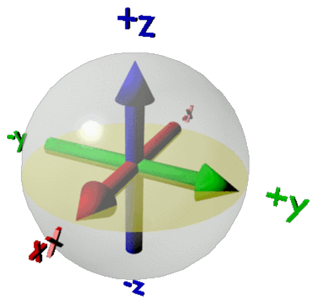


$$\begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$$

Pauli *z* gate

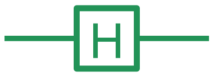
Manipulating Quantum States (cont.)

- **An important single-qubit gate**
  - *H*, a.k.a. Hadamard rotates by *π* radians around the diagonal pointing towards (+*x*, +*z*); it puts each of $|0\rangle$ and $|1\rangle$ into a perfect superposition of $|0\rangle$ and $|1\rangle$
    - $|0\rangle \rightarrow \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$, a.k.a. $|+\rangle$
    - $|1\rangle \rightarrow \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$, a.k.a. $|-\rangle$
  - Measurement of perfect superposition returns 0 and 1 with equal probability
  - *Surprise*: applying a Hadamard gate to a perfect superposition returns 0 or 1 with certainty (because $HH = I$)

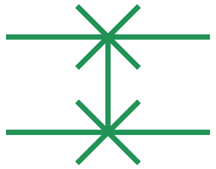


Hadamard gate

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

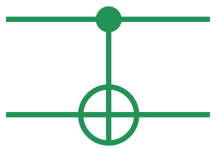


- **Examples of two-qubit gates**
  - SWAP: $\begin{pmatrix}1 & 0 & 0 & 0\\ 0 & 0 & 1 & 0\\ 0 & 1 & 0 & 0\\ 0 & 0 & 0 & 1\end{pmatrix}$



Swaps the values of the two qubits (i.e., maps $|ab\rangle \rightarrow |ba\rangle$)

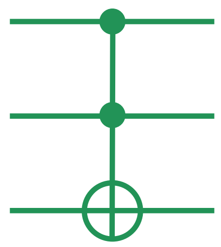
  - CNOT: $\begin{pmatrix}1 & 0 & 0 & 0\\ 0 & 1 & 0 & 0\\ 0 & 0 & 0 & 1\\ 0 & 0 & 1 & 0\end{pmatrix}$



Flips the second qubit if and only if the first qubit is 1 [“**if** *a* **then** $b \leftarrow \neg b$”] (essentially an XOR: $|ab\rangle \rightarrow |a\rangle|a \oplus b\rangle$)
  - Side effect of entangling the two qubits

A Useful Three-Qubit Gate

- **Toffoli gate**
  - A.k.a. controlled-controlled-not or CCNOT
  - CCNOT: $\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix}$



  - Flips the third qubit if and only if *both* of the first two qubits are 1
  - Maps $|abc\rangle \rightarrow |a\rangle|b\rangle|c \oplus ab\rangle$

Input	Output
$\|000\rangle$	$\|000\rangle$
$\|001\rangle$	$\|001\rangle$
$\|010\rangle$	$\|010\rangle$
$\|011\rangle$	$\|011\rangle$
$\|100\rangle$	$\|100\rangle$
$\|101\rangle$	$\|101\rangle$
$\|110\rangle$	$\|111\rangle$
$\|111\rangle$	$\|110\rangle$

- **Universal gate**
  - Can implement any classical Boolean function using only CCNOTs
  - **AND**: CCNOT(x, y, 0) → (x, y, $x \wedge y$)
  - **NOT**: CCNOT(1, 1, x) → (1, 1, $\neg x$)
  - **OR**: CCNOT(1, 1, CCNOT(CCNOT(1, 1, x), CCNOT(1, 1, y), 0)) → (1, 1, $\neg x$, $\neg y$, $x \vee y$)
  - **NAND**: CCNOT(x, y, 1) → (x, y, $\neg(x \wedge y)$)

Constructing a Gate from First Principles

- **What matrix implements a Pauli *X* (NOT) gate?**
  - We assume the standard basis, $|0\rangle \equiv \begin{pmatrix}1\\0\end{pmatrix}$ and $|1\rangle \equiv \begin{pmatrix}0\\1\end{pmatrix}$
- **Start with a truth table mapping inputs to outputs**

Input	Output
$\|0\rangle$	$\|1\rangle$
$\|1\rangle$	$\|0\rangle$

- **Define a corresponding operator**
  - One term per row, which maps input to output and all else to the zero vector
  - $X = |1\rangle\langle 0| + |0\rangle\langle 1|$
  - In matrix form, this would be $X = \begin{pmatrix}0\\1\end{pmatrix}\begin{pmatrix}1 & 0\end{pmatrix} + \begin{pmatrix}1\\0\end{pmatrix}\begin{pmatrix}0 & 1\end{pmatrix} = \begin{pmatrix}0 & 1\\1 & 0\end{pmatrix}$
- **Although defined using basis vectors, this works on superpositions, too**
  - Example: If $|\psi\rangle \equiv \sqrt{\frac{1}{4}}|0\rangle - \sqrt{\frac{3}{4}}|1\rangle$, then $X|\psi\rangle = -\sqrt{\frac{3}{4}}|0\rangle + \sqrt{\frac{1}{4}}|1\rangle$

Constructing a Larger Gate from First Principles

- **What operator/matrix implements a SWAP gate?**
  - This is a two-qubit gate with the semantics $|ab\rangle \rightarrow |ba\rangle$
- **The corresponding truth table is shown at right**
- **Construct an operator (same process as before but with more terms)**
  - $SWAP = |00\rangle\langle 00| + |10\rangle\langle 01| + |01\rangle\langle 10| + |11\rangle\langle 11|$

Input	Output
$\|00\rangle$	$\|00\rangle$
$\|01\rangle$	$\|10\rangle$
$\|10\rangle$	$\|01\rangle$
$\|11\rangle$	$\|11\rangle$

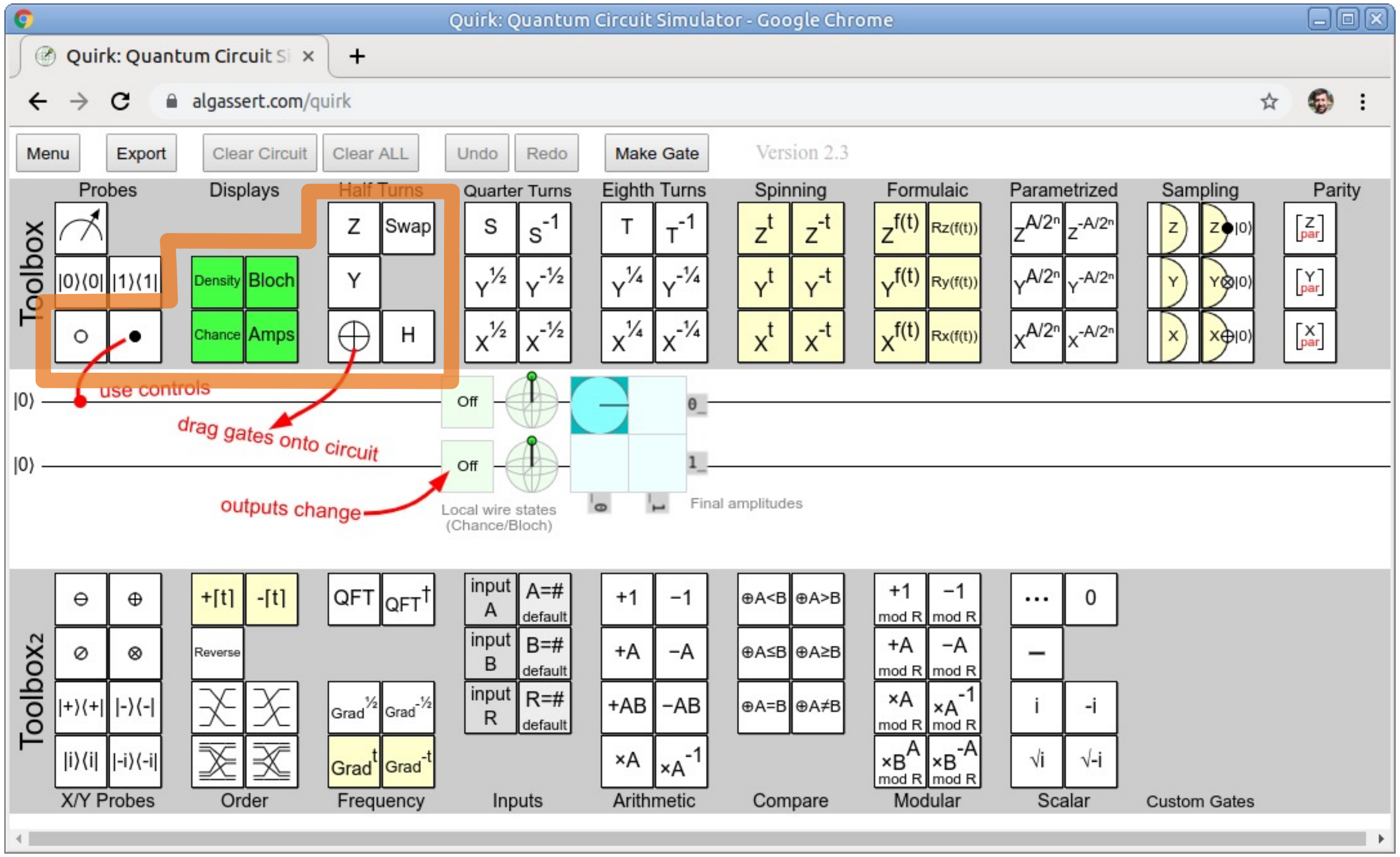
$$= \begin{pmatrix}1\\0\\0\\0\end{pmatrix}(1 \quad 0 \quad 0 \quad 0) + \begin{pmatrix}0\\0\\1\\0\end{pmatrix}(0 \quad 1 \quad 0 \quad 0) + \begin{pmatrix}0\\1\\0\\0\end{pmatrix}(0 \quad 0 \quad 1 \quad 0) + \begin{pmatrix}0\\0\\0\\1\end{pmatrix}(0 \quad 0 \quad 0 \quad 1)$$

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

Let’s Create a Quantum Circuit

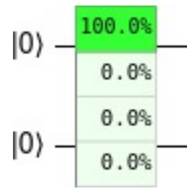
- **We’ll use the Quirk gate-model simulator for this task**
  - Go to https://algassert.com/quirk and click Edit Circuit
  - Easy to use; lots of features; runs entirely within a Web browser

For now, we’ll focus on just the most basic gates

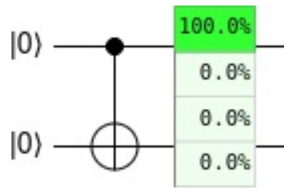


Let’s Create a Quantum Circuit (cont.)

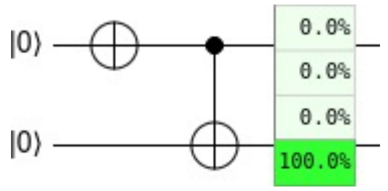
- **What state are we in initially?**
  - The $|00\rangle$ state
  - Place a Chance display on qubit 0 then extend it downwards to cover qubit 1, which shows all two-qubit probabilities



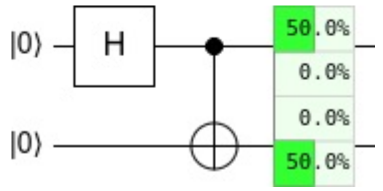
- **What if we add a CNOT from 0 to 1?**
  - So far, nothing happens ($|00\rangle \rightarrow |00\rangle$)



- **What if we put an X before the control?**
  - The state changes from $|00\rangle$ to $|11\rangle$

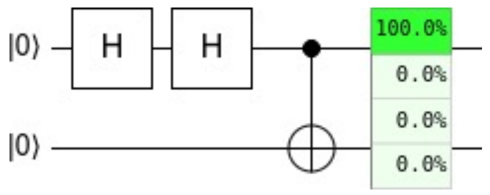


- **What if change the X to an H?**
  - We’re now in the state $|00\rangle + |11\rangle$
  - Because qubit 0 is now equally $|0\rangle$ and $|1\rangle$, it both flips and doesn’t flip qubit 1

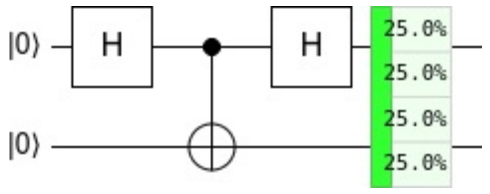


Let's Create a Quantum Circuit (cont.)

- **What if we double the H?**
  - We're back in the $|00\rangle$ state
  - H-H = I so qubit 0 is 0 and we therefore don't flip qubit 1



- **What if we move one of the Hs after the CNOT control?**
  - We're in the $|00\rangle + |01\rangle + |10\rangle - |11\rangle$ state



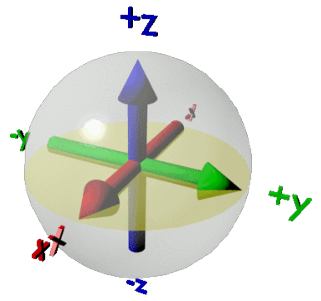
Whoa! What Just Happened?

- **Why does H-H-CNOT produce such a different result from H-CNOT-H?**
  - Let's step through the two cases slowly to see what each circuit does…
- **The H-H-CNOT case**
  - Timeline illustration (unnormalized):

$$\begin{tikzcd}
 & & \vert 00\rangle \\
 & & + \\
 & \vert 00\rangle \arrow[uur, "H_0"] \arrow[r] & \cancel{\vert 01\rangle} \\
\vert 00\rangle \arrow[ur, "H_0"] \arrow[dr] & + & + \quad \left.\right\} & \vert 00\rangle \arrow[r, "CNOT_{0\to1}"] & \vert 00\rangle \\
 & \vert 01\rangle \arrow[r] \arrow[ddr] & \vert 00\rangle \\
 & & + \\
 & & -\cancel{\vert 01\rangle}
\end{tikzcd}$$

The H gate (unnormalized)

Input	Output
$\vert 0\rangle$	$\vert +\rangle = \vert 0\rangle + \vert 1\rangle$
$\vert 1\rangle$	$\vert -\rangle = \vert 0\rangle - \vert 1\rangle$



- **The H-CNOT-H case**
  - Timeline illustration (unnormalized):

$$\begin{tikzcd}
 & & & \vert 00\rangle \\
 & & & + \\
 & \vert 00\rangle \arrow[r, "CNOT_{0\to1}"] & \vert 00\rangle \arrow[uur, "H_0"] \arrow[r] & \vert 01\rangle \\
\vert 00\rangle \arrow[ur, "H_0"] \arrow[dr] & + & + & + \\
 & \vert 01\rangle \arrow[r] & \vert 11\rangle \arrow[r] \arrow[ddr] & \vert 10\rangle \\
 & & & + \\
 & & & -\vert 11\rangle
\end{tikzcd}$$

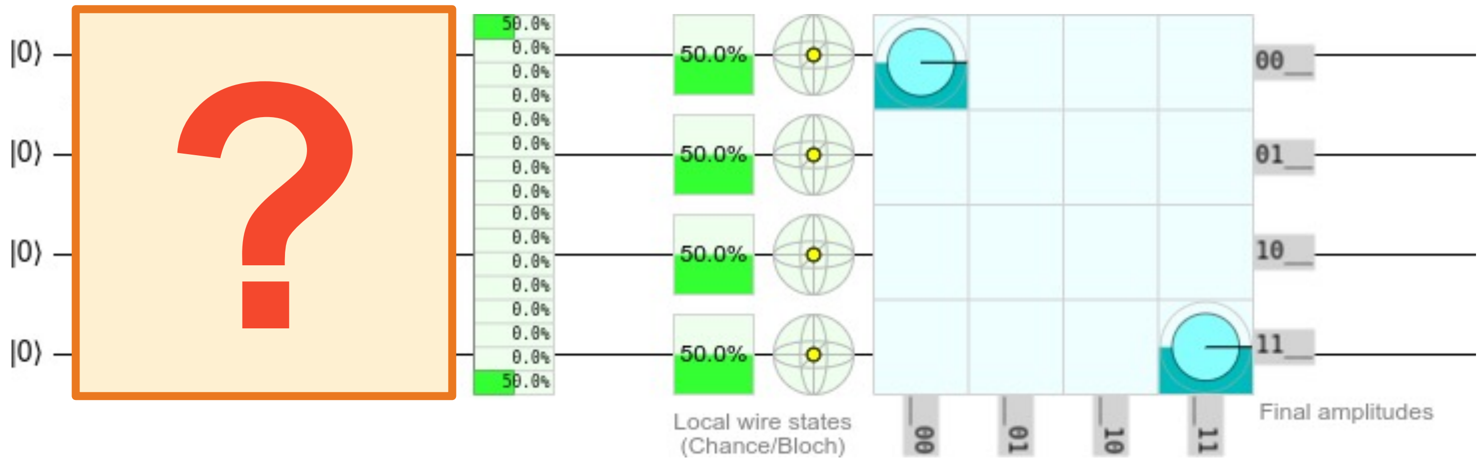
Whoa! What Just Happened? (cont.)

- **Formal explanation of the H-CNOT-H case**
  - Our circuit represents $(I \otimes H)(CNOT)(I \otimes H)$ applied to the input $|00\rangle$
  - $I \otimes H = (|0\rangle\langle 0| + |1\rangle\langle 1|) \otimes \frac{1}{\sqrt{2}}(|0\rangle\langle 0| + |1\rangle\langle 0| + |0\rangle\langle 1| - |1\rangle\langle 1|) = \frac{1}{\sqrt{2}}(|00\rangle\langle 00| + |01\rangle\langle 00| + |00\rangle\langle 01| - |01\rangle\langle 01| + |10\rangle\langle 10| + |11\rangle\langle 10| + |10\rangle\langle 11| - |11\rangle\langle 11|)$
  - $CNOT = |00\rangle\langle 00| + |11\rangle\langle 01| + |10\rangle\langle 10| + |01\rangle\langle 11|$
  - $\therefore (I \otimes H)|00\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |01\rangle)$
  - $\therefore (CNOT)(I \otimes H)|00\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$
  - $\therefore (I \otimes H)(CNOT)(I \otimes H)|00\rangle = \frac{1}{2}(|00\rangle + |01\rangle + |10\rangle - |11\rangle)$
- **Or, if you prefer a matrix formulation,**
  - $I \otimes H = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 & 0 & 0 \\ 1 & -1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & -1 \end{pmatrix}$; $\mathrm{CNOT} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix}$; and $|00\rangle = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix}$
  - So we get $\frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 & 0 & 0 \\ 1 & -1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & -1 \end{pmatrix}\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix}\frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 & 0 & 0 \\ 1 & -1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & -1 \end{pmatrix}\begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix} = \frac{1}{2}\begin{pmatrix} 1 \\ 1 \\ 1 \\ -1 \end{pmatrix}$

Different from what we showed on an earlier slide because we swapped the order of the control and target qubits

Hands-On Exercise: Construct a 4-Qubit GHZ State

- **Greenberger–Horne–Zeilinger (GHZ) state**
  - Entangled state, equally likely to be all zeros or all ones but never anything else
- **For this exercise, we'll construct a 4-qubit GHZ state in Quirk**
  - That is, we want to create a circuit that produces $\frac{1}{\sqrt{2}}(|0000\rangle + |1111\rangle)$
  - Here's what your solution should look like (and note that we extended the Chance display to cover four qubits):



- **We'll provide hints every few minutes to help you keep making progress**

Difficult Hands-On Exercise: a 2-bit Adder

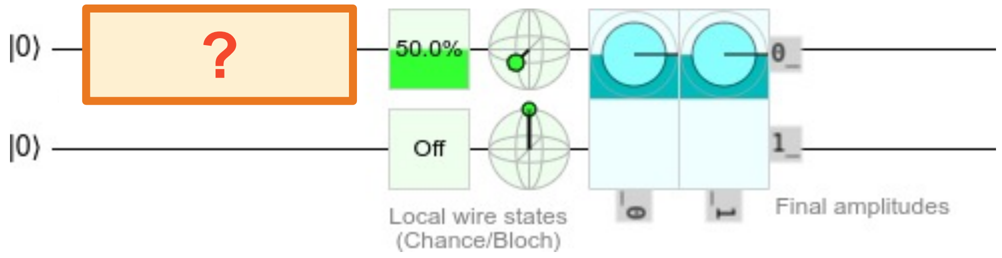
- **Skip this exercise unless you've already completed the previous exercise and want a bigger challenge**
- **With only 4 qubits, implement a 2-bit adder modulo 4**
  - $c_1 c_0 \leftarrow (a_1 a_0 + b_1 b_0) \bmod 4$
  - Input $\{a_1, a_0, b_1, b_0\}$ and output $\{c_1, c_0, x, y\}$ (where x and y are "don't cares")

- **Hint #1**
  - This can be implemented with exactly two CNOTs plus one Toffoli (CCNOT) gate
- **Hint #2**
  - CNOT flips the target bit iff the control bit is 1 (i.e., $|ct\rangle \rightarrow |c\rangle|c \oplus t\rangle$)
  - CCNOT flips the target bit iff both control bits are 1 (i.e., $|c_1 c_0 t\rangle \rightarrow |c_1\rangle|c_0\rangle|c_1 c_0 \oplus t\rangle$)
- **Hint #3**
  - From Digital Circuits 101,

$$\begin{array}{lcc} & a_1 & a_0 \\ + & b_1 & b_0 \\ \hline & a_1 \oplus b_1 \oplus a_0 b_0 & a_0 \oplus b_0 \end{array}$$

4-Qubit GHZ State: Hint #1

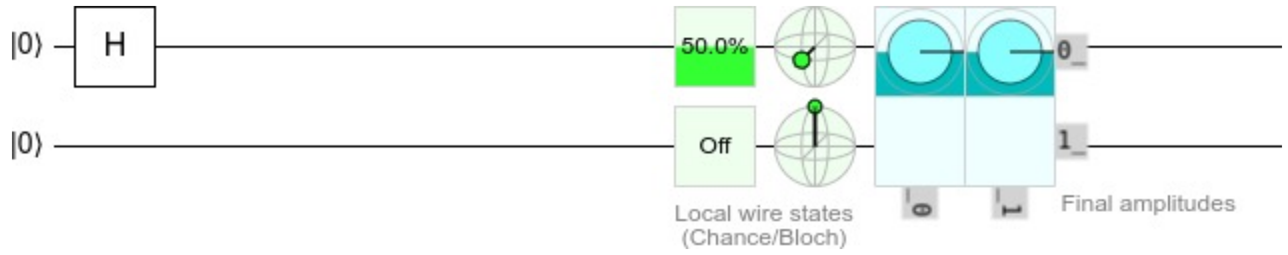
- **How would you create a 1-qubit GHZ state?**
  - That is, $\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$ (a.k.a. $|+\rangle$), a state that's equally likely to be $|0\rangle$ or $|1\rangle$
  - What gate have we seen that does this?
- **Solution format**



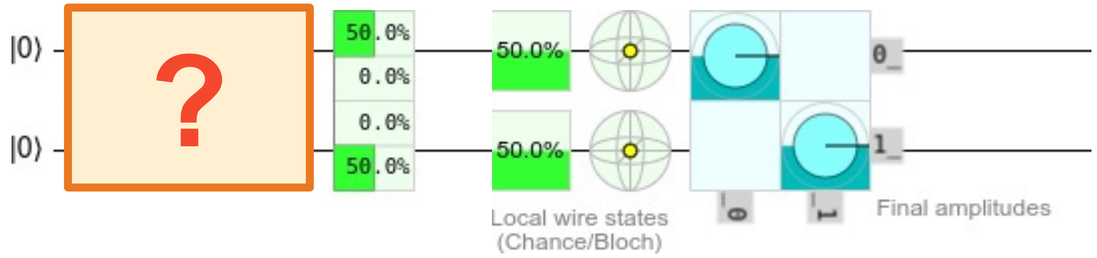
  - Quirk requires a minimum of two qubits so just leave qubit 1 alone
  - Technically, the above represents $\frac{1}{\sqrt{2}}(|00\rangle + |01\rangle)$

4-Qubit GHZ State: Hint #2

- **Solution to Hint #1: Creating a 1-qubit GHZ state**
  - All we need is an H gate to transform state $|00\rangle$ into state $\frac{1}{\sqrt{2}}(|00\rangle + |01\rangle)$



- **Hint #2: How would you create a 2-qubit GHZ state?**
  - That is, $\frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$, a state that's equally likely to be $|00\rangle$ or $|11\rangle$
  - Start from the Hint #1 state, $\frac{1}{\sqrt{2}}(|00\rangle + |01\rangle)$
  - How can we leave $|00\rangle$ alone but replace $|01\rangle$ with $|11\rangle$?
- **Solution format**

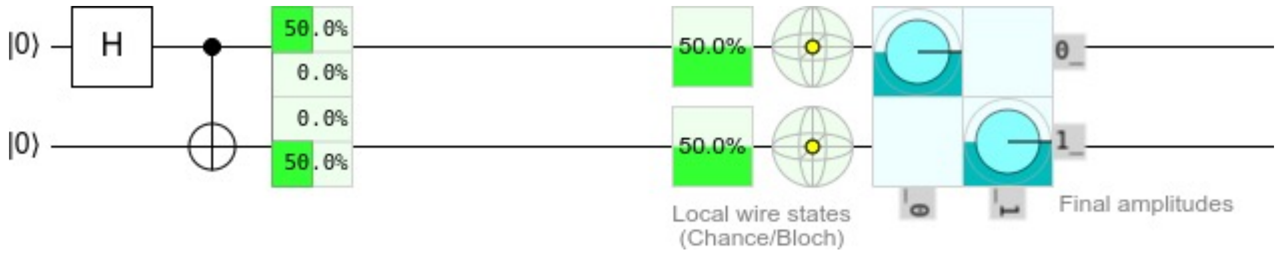


4-Qubit GHZ State: Hint #2′

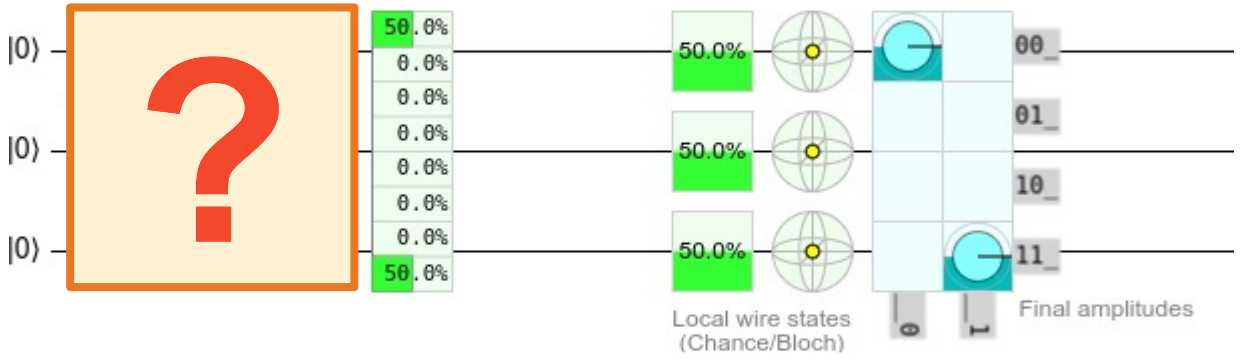
- **Hint #2: How would you create a 2-qubit GHZ state?**
  - That is, $\frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$, a state that's equally likely to be $|00\rangle$ or $|11\rangle$
  - Start from the Hint #1 state, $\frac{1}{\sqrt{2}}(|00\rangle + |01\rangle)$
  - How can we leave $|00\rangle$ alone but replace $|01\rangle$ with $|11\rangle$?
- **Hint #2′: What single 2-qubit gate performs the preceding mapping?**
  - Given $|ab\rangle$, negate a if and only if b is 1

4-Qubit GHZ State: Hint #3

- **Solution to Hint #2: Creating a 2-qubit GHZ state**
  - A CNOT gate performs the requisite mapping from $\frac{1}{\sqrt{2}}(|00\rangle + |01\rangle)$ to $\frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$
  - “If qubit 0 is 1, flip qubit 1” (from 0 to 1 in this case)

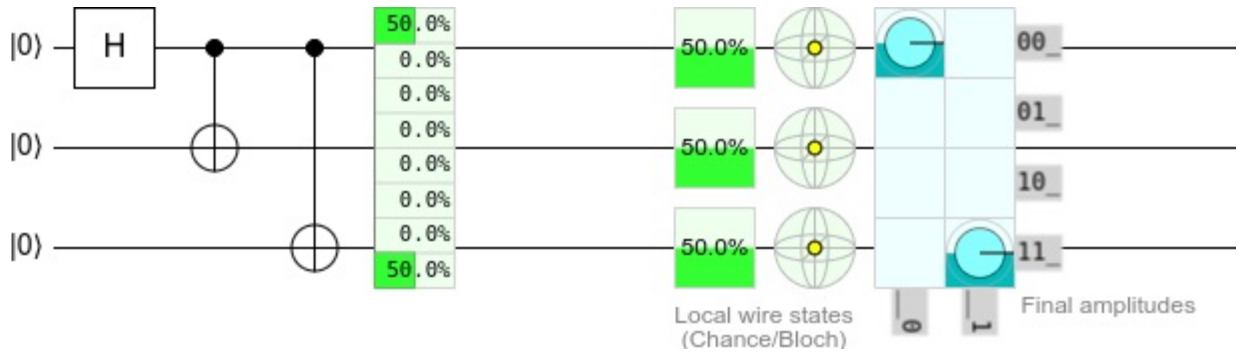


- **Hint #3: How would you create a 3-qubit GHZ state?**
  - Extend the above to 3 qubits: Given $\frac{1}{\sqrt{2}}(|000\rangle + |011\rangle)$, produce $\frac{1}{\sqrt{2}}(|000\rangle + |111\rangle)$
- **Solution format**



4-Qubit GHZ State: Hint #4

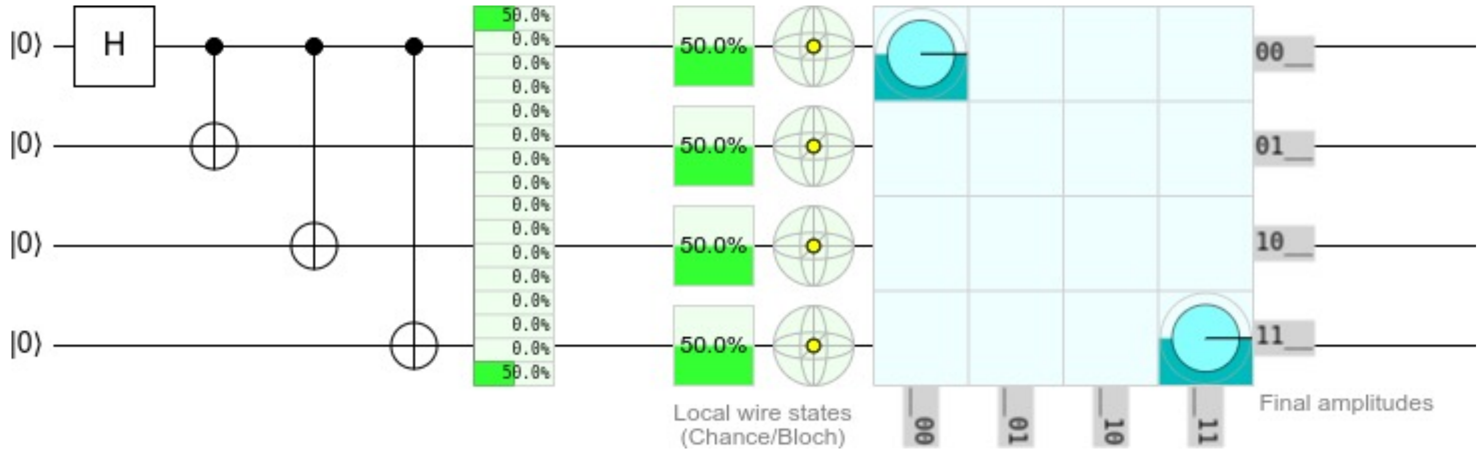
- **Solution to Hint #3: Creating a 3-qubit GHZ state**
  - We simply repeat what we did for Hint #2
  - A CNOT from qubit 0 to qubit 2 implements "If qubit 0 is 1, flip qubit 2"
  - Maps $\frac{1}{\sqrt{2}}(|000\rangle + |011\rangle)$ to $\frac{1}{\sqrt{2}}(|000\rangle + |111\rangle)$



- **Hint #4: What's the pattern? How can we scale up from a 3-qubit GHZ state to to a 4-qubit GHZ state?**

4-Qubit GHZ State: Solution

- **Solution to Hint #4: Scaling up**
  - We can keep applying CNOTs to copy qubit 0 to each subsequent qubit in turn to produce a circuit for the desired 4-qubit GHZ state



4-Qubit GHZ State: Solution (cont.)

- **Note how much easier it is to specify a quantum circuit gate-by-gate than to specify the complete unitary matrix to which it corresponds:**

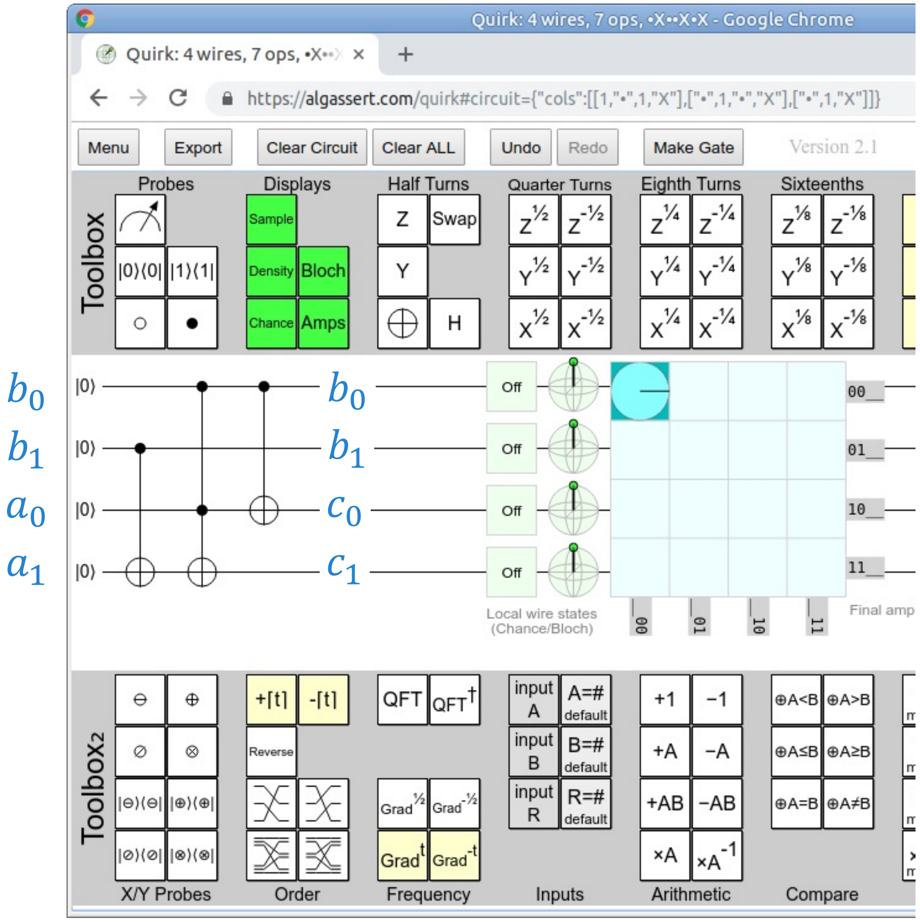
$$\frac{1}{\sqrt{2}}\begin{pmatrix}
1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 \\
0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\
0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\
1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{pmatrix}$$

- **Such matrices grow large quickly**
  - The above is a 16×16 matrix; producing a 5-qubit GHZ would require a 32×32 matrix

2-bit Adder: Solution

- **Problem**
  - With only 4 qubits, implement a 2-bit adder modulo 4: $c_1 c_0 \leftarrow (a_1 a_0 + b_1 b_0) \bmod 4$
- **Solution (explained by time step)**
  1. $\{a_1, a_0, b_1, b_0\}$
  2. $\{a_1 \oplus b_1, a_0, b_1, b_0\}$
  3. $\{a_1 \oplus b_1 \oplus a_0 b_0, a_0, b_1, b_0\}$
  4. $\{a_1 \oplus b_1 \oplus a_0 b_0, a_0 \oplus b_0, b_1, b_0\}$

  $= \{c_1, c_0, b_1, b_0\}$



Agenda

- **Part I: Quantum-computing fundamentals**
  - High-level motivation, history, and status
  - Qubits, multi-qubit states, and quantum measurement

Morning break

- **Part II: Circuit-model quantum computing**
  - Review of notation
  - Quantum gates and quantum circuits

Lunch

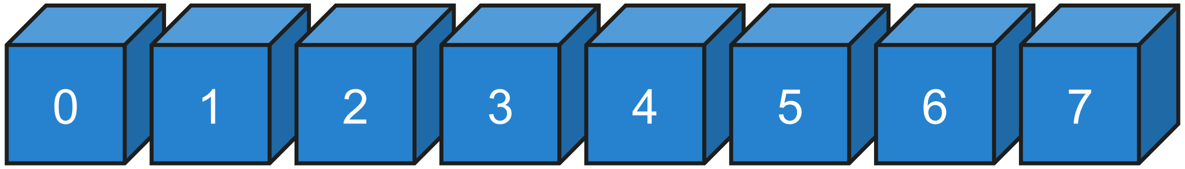
  - Basic quantum algorithms
- **Part III: Quantum annealing**

Afternoon break

- **Part IV: Further quantum algorithms**
  - Quantum Alternating Operator Ansatz (QAOA)
  - Advancements in quantum algorithms
  - Concluding remarks

Grover’s Algorithm

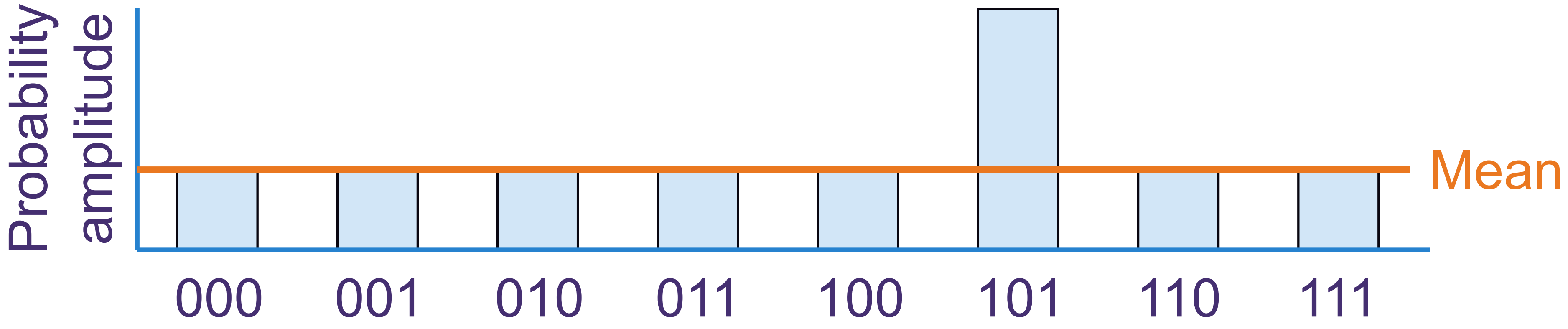
- **Which box contains the prize?**



  – Classically, must open all 8 boxes in the worst case
- **Let’s see how we can use quantum effects to do better than that…**
- **Given**
  – A power-of-two number of boxes
  – A guarantee that exactly one box contains the prize
  – An operator U_ω that, given a box number $|x\rangle$, negates the probability amplitude iff the box contains the prize (i.e., $U_\omega|x\rangle = -|x\rangle$ for $x = \omega$ and $U_\omega|x\rangle = |x\rangle$ for $x \neq \omega$)
- **Define the *Grover diffusion operator* as follows**
  – $|s\rangle \equiv \frac{1}{\sqrt{N}}\sum_{i=0}^{N-1}|x\rangle$ (i.e., the equal superposition of all states)
  – $U_s \equiv 2|s\rangle\langle s| - I$ (the Grover diffusion operator)

Grover's Algorithm (cont.)

- **The basic algorithm is fairly straightforward to apply:**
  - Put each of the *n* qubits in a superposition of $|0\rangle$ and $|1\rangle$
  - For $\sqrt{2^n}$ iterations
    - Apply U_ω to the state
    - Apply U_s to the state
- **How does that work?**
  - Gradually shifts the probability amplitude to qubit ω from all the other qubits
  - When we measure, we'll get a result of ω with near certainty

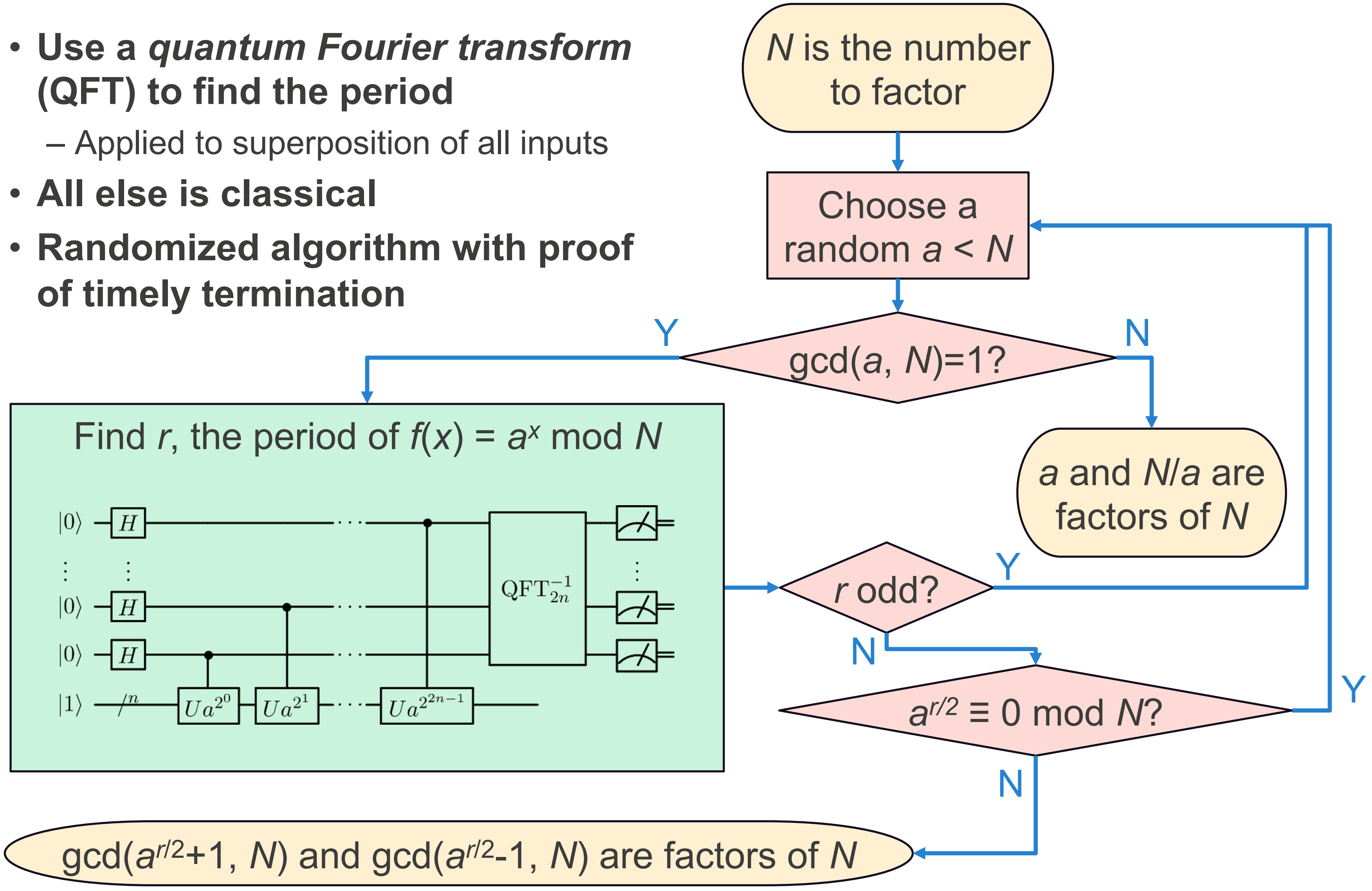


Shor’s Algorithm

- **Factor 1,274,093,332,123,426,680,869 into a product of two primes**
  - Okay, it’s 135,763,451,261×9,384,656,329
- **Observations**
  - Given that N is the product of two primes, p and q
  - Given some a that is divisible by neither p nor q
  - Then the sequence $\{a^1 \bmod N, a^2 \bmod N, a^3 \bmod N, a^4 \bmod N, a^5 \bmod N, \ldots\}$ will repeat every r elements (the sequence’s *period*)
  - As Euler discovered (ca. 1760), r always divides $(p-1)(q-1)$
- **Example**
  - Let a be 2 and N be 15 (=3×5)
  - Then $a^x \bmod N = \{2, 4, 8, 1, 2, 4, 8, 1, 2, 4, 8, 1, 2, 4, 8, 1 \ldots\}$ so r is 4
  - Lo and behold, 4 divides $(3-1)(5-1)=8$
- **Approach**
  - Once we know the period, r, it’s not too hard to find N’s prime factors p and q
  - Unfortunately, finding r is extremely time-consuming…for a classical computer

Shor’s Algorithm (cont.)

- **Use a *quantum Fourier transform* (QFT) to find the period**
  - Applied to superposition of all inputs
- **All else is classical**
- **Randomized algorithm with proof of timely termination**



N is the number to factor
Choose a random *a* < *N*
gcd(*a*, *N*)=1?
Y
N
a and *N*/*a* are factors of *N*
Find *r*, the period of *f*(*x*) = a^x mod *N*
$|0\rangle$
$|0\rangle$
$|0\rangle$
$|1\rangle$
H
n
Ua^{2^0}
Ua^{2^1}
$Ua^{2^{2n-1}}$
QFT_{2n}^{-1}
r odd?
Y
N
$a^{r/2} \equiv 0 \bmod N$?
Y
N
gcd($a^{r/2}$+1, *N*) and gcd($a^{r/2}$-1, *N*) are factors of *N*

Quantum Annealing

Agenda

- **Part I: Quantum-computing fundamentals**
  - High-level motivation, history, and status
  - Qubits, multi-qubit states, and quantum measurement

Morning break

- **Part II: Circuit-model quantum computing**
  - Review of notation
  - Quantum gates and quantum circuits

Lunch

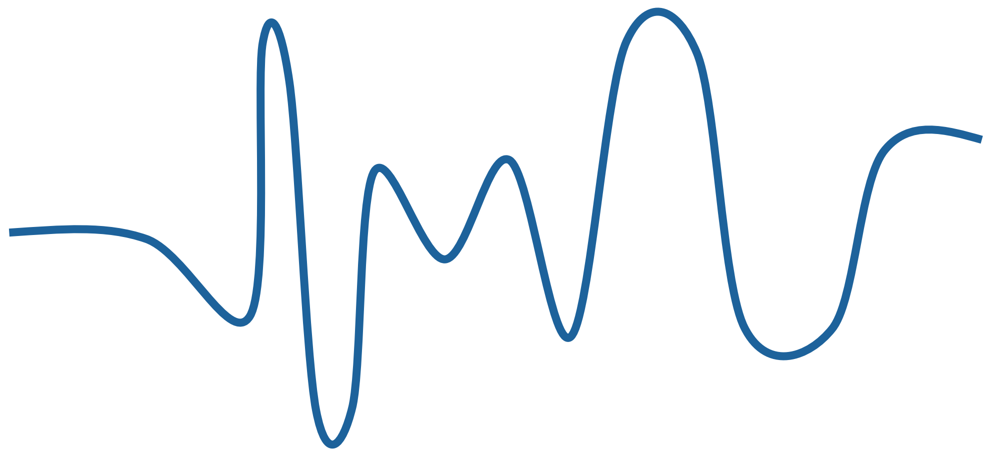
  - Basic quantum algorithms
- **Part III: Quantum annealing**

Afternoon break

- **Part IV: Further quantum algorithms**
  - Quantum Alternating Operator Ansatz (QAOA)
  - Advancements in quantum algorithms
  - Concluding remarks

Simulated Annealing

- **Classical (and classic) optimization approach**
- **Find the coordinates of the minimum value in an energy landscape**
- **Conceptual approach**
  - Drop a bunch of rubber balls on the landscape, evaluating the function wherever they hit
  - Hope that one of the balls will bounce and roll downhill to the global minimum
- **Challenge: Commonly get stuck in a local minimum**
- **Solution: *Quantum tunneling***
  - A quantum annealer can tunnel through narrow energy barriers, regardless of height



Quantum Annealing

- **A quantum annealer solves only a single, parameterized, problem**
- **Find the minimum energy (value) of a given, "problem" Hamiltonian**

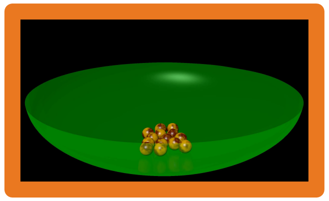
$$\mathcal{H} = \sum_{i=0}^{N-1} h_i \sigma_i + \sum_{i=0}^{N-2} \sum_{j=i+1}^{N-1} J_{i,j} \sigma_i \sigma_j$$

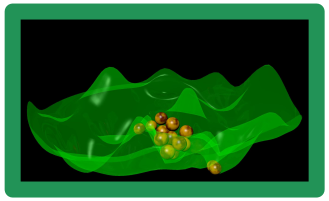
- **What the hardware does**
  - Find the $\sigma_i \in \{-1, +1\}$ that minimize $\mathcal{H}$ given coefficients $h_i \in \mathbb{R}$ and $J_{i,j} \in \mathbb{R}$
  - In other words, a quantum-annealing program is merely a list of h_i and $J_{i,j}$ values
- **This is a *classical* problem with a *classical* solution**
  - Quantum effects are used internally to work towards the goal

How Quantum Annealing Works

- **The hardware combines our problem Hamiltonian with an initial Hamiltonian as follows to produce a *time-dependent* Hamiltonian:**

$$\mathcal{H}(s) = \underbrace{-\frac{\Delta(s)}{2}\sum_{i=0}^{N-1}\sigma_i^x}_{} + \underbrace{\frac{\varepsilon(s)}{2}\left(\sum_{i=0}^{N-1} h_i\sigma_i^z + \sum_{i=0}^{N-2}\sum_{j=i+1}^{N-1} J_{i,j}\sigma_i^z\sigma_j^z\right)}_{}$$





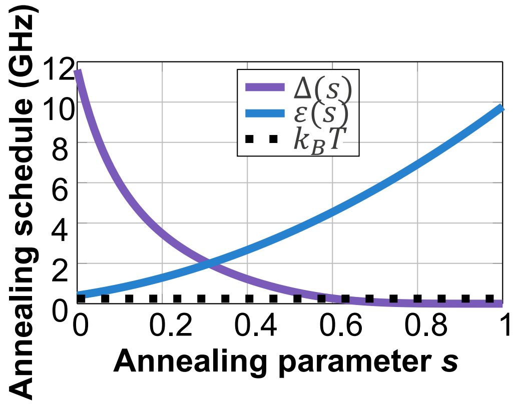


Image credit: King, Hoskinson, Lanting, Andriyash, and Amin (2016)

- **Start in a trivial energy landscape**
  - Qubits initialized to the solution to this known, trivial problem
- **Gradually transition to the problem state**
  - Decrease transverse-field strength
  - Increase longitudinal-field strength
- **Premise (adiabatic theorem)**
  - Sufficiently gradual transition → qubits remain in solution state

Problem Generality

- **Are there really that many problems that can be expressed as $\arg\min_\sigma \mathcal{H}(\sigma) = \sum_{i=0}^{N-1} h_i\sigma_i + \sum_{i=0}^{N-2}\sum_{j=i+1}^{N-1} J_{i,j}\sigma_i\sigma_j$?**
- **Yes!**
- **Any polynomial cost function over discrete variables with polynomial constraints can be mapped to this form**
- **Examples abound**
  - List taken from Andrew Lucas, "Ising formulations of many NP problems", DOI 10.3389/fphy.2014.00005

- Number partitioning
- Graph partitioning
- Cliques
- Binary integer linear programming
- Exact cover
- Set packing
- Vertex cover
- Satisfiability
- Minimal maximal matching
- Set cover
- Knapsack with integer weights
- Graph coloring
- Clique cover
- Job sequencing with integer weights
- Hamiltonian cycles and paths
- Traveling salesman
- Minimal spanning tree with a maximal degree constraint
- Steiner trees
- Directed and undirected feedback vertex sets
- Feedback edge set
- Graph isomorphisms

Alternative Problem Formulations

- **Physics version**
  - Minimize the energy of a 2-local Ising-model Hamiltonian function

    $\mathcal{H}(\sigma) = \sum_{i=0}^{N-1} h_i \sigma_i + \sum_{i=0}^{N-2} \sum_{j=i+1}^{N-1} J_{i,j} \sigma_i \sigma_j$

    with $h_i \in \mathbb{R}$, $J_{i,j} \in \mathbb{R}$, and $\sigma_i \in \{-1, +1\}$
  - We'll call this version "Ising" for short

- **Operations Research version**
  - Minimize the value of a quadratic pseudo-Boolean objective function

    $Obj(x) = \sum_{i=0}^{N-1} a_i x_i + \sum_{i=0}^{N-2} \sum_{j=i+1}^{N-1} b_{i,j} x_i x_j$

    with $a_i \in \mathbb{R}$, $b_{i,j} \in \mathbb{R}$, and $x_i \in \{0, 1\}$
  - Known as a *quadratic unconstrained binary optimization* (QUBO) or an *unconstrained binary quadratic programming* (UBQP) problem
  - Sometimes expressed in matrix form:

    $Obj(x) = x^{\mathrm{T}} Q x$

    for either symmetric or upper-triangular matrix Q with the a_i on the diagonal and the $b_{i,j}$ on the off-diagonals

- **Conversion between the two versions requires only a linear transformation**
  - Substitute $x_i = (\sigma_i + 1)/2$ or $\sigma_i = 2x_i - 1$
- **Some problems are more natural to express with one formulation than the other**

Setting up a Trivial Problem

- **Goal: Define a two-variable function that is minimized when $A = B$**
  - That is, we want a quantum annealer to return either {TRUE, TRUE} or {FALSE, FALSE} but not {TRUE, FALSE} or {FALSE, TRUE}
- **Approach #1 (Ising)**
  - Set up and solve a system of inequalities with valid solutions evaluating to some arbitrary value k and invalid solutions evaluating to any value $> k$

σ_A	σ_B	$h_A\sigma_A + h_B\sigma_B + J_{A,B}\sigma_A\sigma_B$	Must be
-1	-1	$-h_A - h_B + J_{A,B}$	$= k$
-1	$+1$	$-h_A + h_B - J_{A,B}$	$> k$
$+1$	-1	$+h_A - h_B - J_{A,B}$	$> k$
$+1$	$+1$	$+h_A + h_B + J_{A,B}$	$= k$

  - One possibility: $\mathcal{H}(\sigma_A, \sigma_B) = -\sigma_A\sigma_B$ (i.e., $h_A = h_B = 0$ and $J_{A,B} = -1$), with $k = -1$
  - Not unique; in this case, any $J_{A,B} < 0$ will do

Setting up a Trivial Problem (cont.)

- **Approach #2 (QUBO)**
  - Add one *penalty term* for each invalid solution
  - Penalty terms evaluate to a positive number when given a specific invalid solution or zero when given any other solution

x_A	x_B	Penalty
0	0	0
0	1	$(1-x_A)x_B$
1	0	$x_A(1-x_B)$
1	1	0

  - One possibility: $Obj(x_A, x_B) = (1-x_A)x_B + x_A(1-x_B) = x_A + x_B - 2x_Ax_B$
  - Not unique; scaling by any positive value will also work

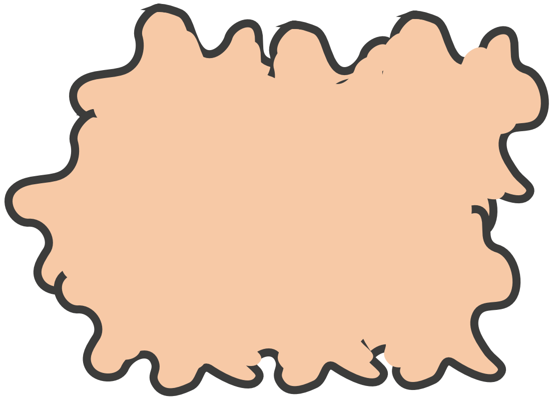
A Convenient Property

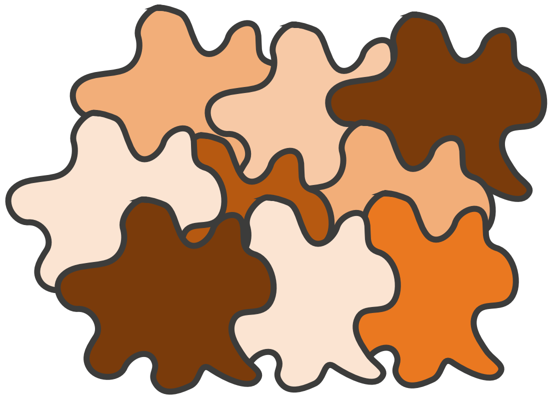
- **Both Ising and QUBO expressions are additive**
- **That is $\mathcal{H}_1 + \mathcal{H}_2$ is minimized at the intersection of the set of values that minimize $\mathcal{H}_1$ and the set of values that minimize $\mathcal{H}_2$**
- **Likewise for $\boldsymbol{Obj_1}$ and $\boldsymbol{Obj_2}$**
- **If the intersection is empty, the result can be hard to reason about**
  - A quantum annealer *always* returns an answer because every Ising Hamiltonian/QUBO has a minimum value

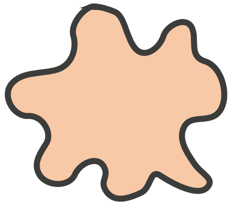
- **Example**
  - Define a four-variable function that is minimized when $A = B = C = D$
  - Approach: Add multiple two-variable function instances from the previous slides
  - Ising solution: $\mathcal{H}_4(\sigma_A, \sigma_B, \sigma_C, \sigma_D) = \mathcal{H}_2(\sigma_A, \sigma_B) + \mathcal{H}_2(\sigma_B, \sigma_C) + \mathcal{H}_2(\sigma_C, \sigma_D) = -\sigma_A\sigma_B - \sigma_B\sigma_C - \sigma_C\sigma_D$
  - QUBO solution: $Obj_4(x_A, x_B, x_C, x_D) = Obj_2(x_A, x_B) + Obj_2(x_B, x_C) + Obj_2(x_C, x_D) = x_A + 2x_B + 2x_C + x_D - 2x_Ax_B - 2x_Bx_C - 2x_Cx_D$

One Way to Program a Quantum Annealer

- **Constructing an $\mathcal{H}_4$ from multiple instances of an $\mathcal{H}_2$ suggests a more general programming methodology**
  – Break down a problem into a set of repeated subproblems (“building blocks”)
  – Solve the subproblems by hand in the *reverse* direction from what the quantum annealer would do: Given the variables (σ_i or x_i), solve for the coefficients ($h_i/J_{i,j}$ or $a_i/b_{i,j}$)
  – Combine the subproblems into a complete problem
  – Solve the complete problem on the quantum annealer in the forward direction: Given the coefficients, solve for the variables

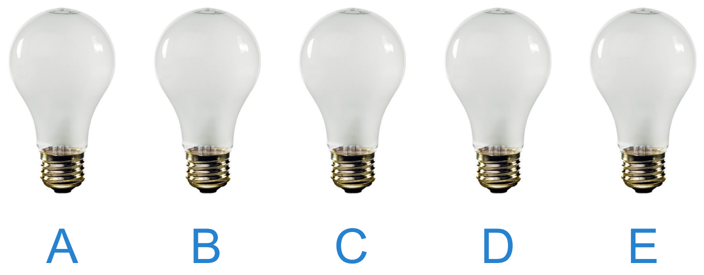

Big, complicated problem


Break down into repeated subproblems


Solve simple subproblem by hand then combine to solve complicated problem

Example of Decomposing a Problem

- **Problem**
  - Configure five lights, labeled A–E, such that exactly one of A, B, and C is on, exactly one of B, C, and D is on, and exactly one of C, D, and E is on
- **Subproblem to solve**
  - Exactly 1 of 3 lights must be on—will apply to {A, B, C}, {B, C, D}, and {C, D, E}



Ising Solution

σ_0	σ_1	σ_2	$\sum_{i=0}^{N-1} h_i\sigma_i + \sum_{i=0}^{N-2} \sum_{j=i+1}^{N-1} J_{i,j}\sigma_i\sigma_j$	Must be
-1	-1	-1	$-h_0 - h_1 - h_2 + J_{0,1} + J_{0,2} + J_{1,2}$	$> k$
-1	-1	$+1$	$-h_0 - h_1 + h_2 + J_{0,1} - J_{0,2} - J_{1,2}$	$= k$
-1	$+1$	-1	$-h_0 + h_1 - h_2 - J_{0,1} + J_{0,2} - J_{1,2}$	$= k$
-1	$+1$	$+1$	$-h_0 + h_1 + h_2 - J_{0,1} - J_{0,2} + J_{1,2}$	$> k$
$+1$	-1	-1	$+h_0 - h_1 - h_2 - J_{0,1} - J_{0,2} + J_{1,2}$	$= k$
$+1$	-1	$+1$	$+h_0 - h_1 + h_2 - J_{0,1} + J_{0,2} - J_{1,2}$	$> k$
$+1$	$+1$	-1	$+h_0 + h_1 - h_2 + J_{0,1} - J_{0,2} - J_{1,2}$	$> k$
$+1$	$+1$	$+1$	$+h_0 + h_1 + h_2 + J_{0,1} + J_{0,2} + J_{1,2}$	$> k$

One solution: $\mathcal{H}_{1\text{of}3}(\sigma_0, \sigma_1, \sigma_2) = \sigma_0 + \sigma_1 + \sigma_2 + \sigma_0\sigma_1 + \sigma_0\sigma_2 + \sigma_1\sigma_2$, with $k = -2$

QUBO Solution

x_0	x_1	x_2	Penalty
0	0	0	$(1-x_0)(1-x_1)(1-x_2)$
0	0	1	0
0	1	0	0
0	1	1	$(1-x_0)x_1x_2$
1	0	0	0
1	0	1	$x_0(1-x_1)x_2$
1	1	0	$x_0x_1(1-x_2)$
1	1	1	$x_0x_1x_2$

$$Obj(x) = \boxed{-3x_0x_1x_2} + 2x_0x_1 + 2x_0x_2 + 2x_1x_2 - x_0 - x_1 - x_2 + 1$$

Uh oh! Cubic terms aren't allowed in a *quadratic* unconstrained binary optimization problem.

Corrected QUBO Solution

x_0	x_1	x_2	Penalty
0	0	0	$(1-x_0)(1-x_1)(1-x_2)$
0	0	1	0
0	1	0	0
0	1	1	$(1-x_0)x_1x_2$
1	0	0	0
1	0	1	$x_0(1-x_1)x_2$
1	1	0	$x_0x_1(1-x_2)$
1	1	1	$4x_0x_1x_2$

$$Obj(x) = 2x_0x_1 + 2x_0x_2 + 2x_1x_2 - x_0 - x_1 - x_2 \cancel{+1}$$

Offset four coefficients of -1 with one coefficient of $+4$

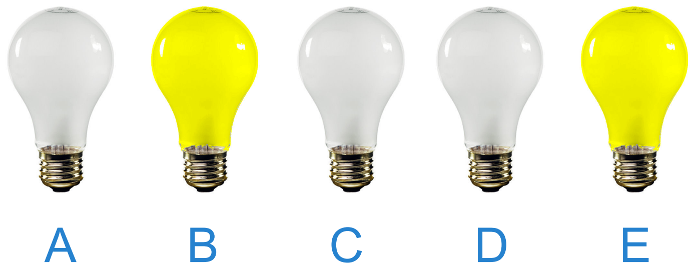
Constants can be discarded, as they don't affect the values of x that minimize $Obj(x)$

Constructing the Full Problem

- **Given the solution to the subproblem,**
  - $\mathcal{H}_{1\text{of}3}(\sigma_0, \sigma_1, \sigma_2) = \sigma_0 + \sigma_1 + \sigma_2 + \sigma_0\sigma_1 + \sigma_0\sigma_2 + \sigma_1\sigma_2$ (Ising-model formulation)

  we can simply add instances of that to define our full problem:
  - $\mathcal{H}(A, B, C, D, E) = \mathcal{H}_{1\text{of}3}(A, B, C) + \mathcal{H}_{1\text{of}3}(B, C, D) + \mathcal{H}_{1\text{of}3}(C, D, E)$

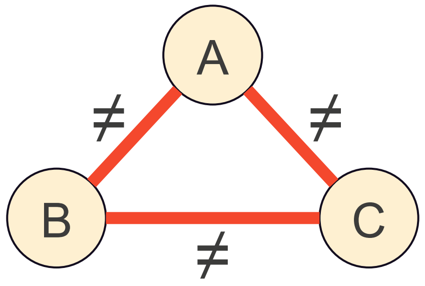
  which expands to
  - $\mathcal{H}(A, B, C, D, E) = A + 2B + 3C + 2D + E + AB + AC + 2BC + BD + 2CD + CE + DE$
- **This can be passed to a quantum annealer system for solution**
  - Hint: There exist three valid solutions out of 32 possible configurations of the five lights



Frustration

- **One can think of the role of a quantum annealer as a constraint-satisfaction engine**
- **So far, we've seen a few examples of constraints:**
  - Constrain $A = B$
  - Constrain exactly one of three variables to be "on"
  - Constrain exactly one of A, B, and C, exactly one of B, C, and D, and exactly one of C, D, and E to be "on"
- **Consider another trivial constraint: $\boldsymbol{A \neq B}$**
  - $\mathcal{H}_{\neq}(\sigma_A, \sigma_B) = \sigma_A \sigma_B$

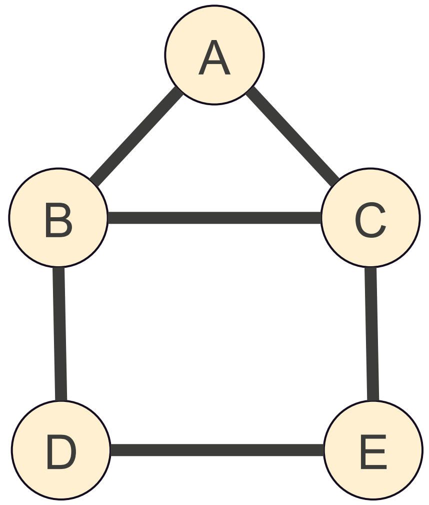
    or
  - $Obj_{\neq}(x_A, x_B) = -x_A - x_B + 2x_A x_B$
- **What will a quantum annealer do when presented the following?**
  - $\mathcal{H}(A,B,C) = \mathcal{H}_{\neq}(A,B) + \mathcal{H}_{\neq}(B,C) + \mathcal{H}_{\neq}(C,A)$
  - That is, $A \neq B$, $B \neq C$, and $C \neq A$



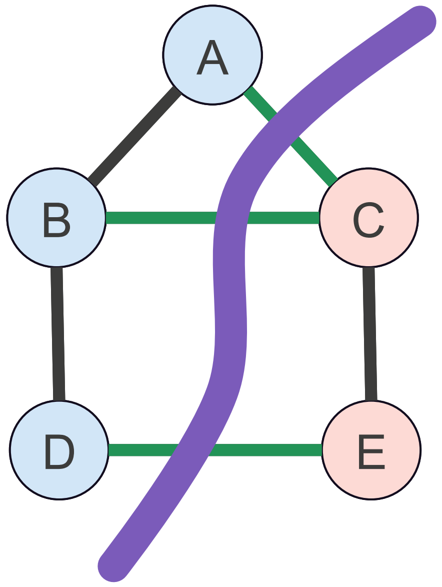
- **This is known as a *frustrated system***
  - Cannot satisfy all constraints
- **A quantum annealer will satisfy as much as it can**
  - "No solution" is not a possibility—it's minimizing, not solving, $\mathcal{H}$
  - Key to a quantum annealer's power

Exploiting Frustration

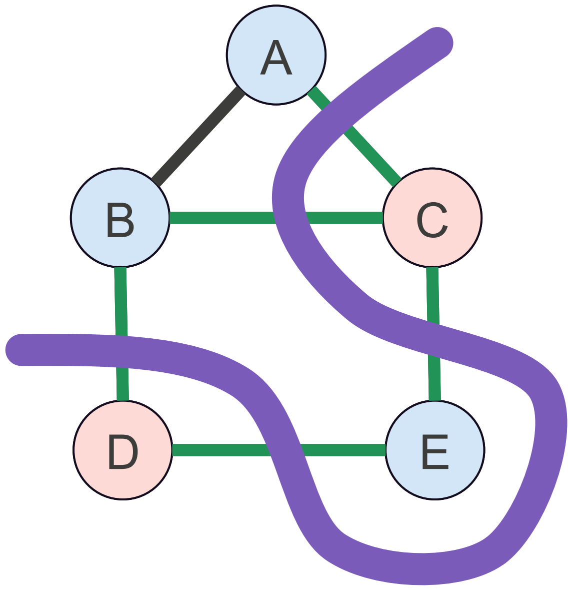
- **Example: Find the maximum cut of a graph**
  - Goal is to divide the vertices of an undirected graph into two partitions to maximize the number of edges that span partitions
  - This is an NP-hard problem



A example graph



Partitioning leading to a cut of size 3 (not maximal)

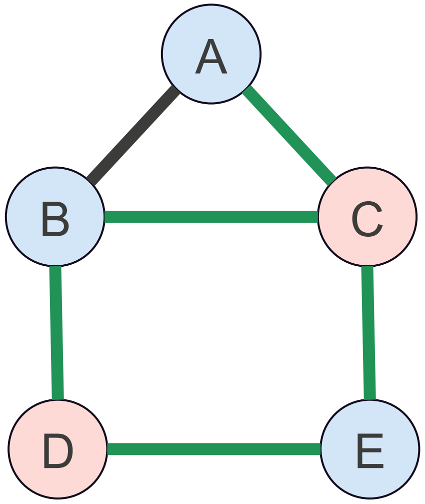


Partitioning leading to a cut of size 5 (maximal)

Maximum Cut on a Quantum Annealer

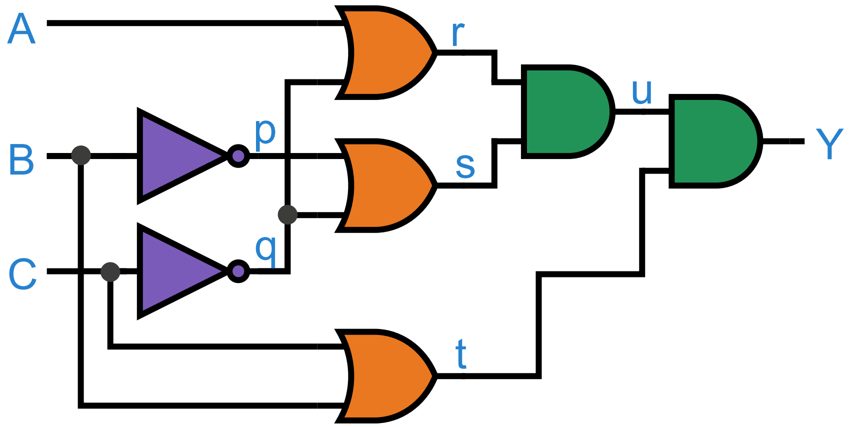
- **Maximizing the number of edges that span partitions = maximizing the number of adjacent vertices with different colors**
- **Approach: Specify $\sigma_i \neq \sigma_j$ for all adjacent vertices σ_i and σ_j**
- **These may not all be satisfiable, but the quantum annealer will satisfy as many as possible, as this is what minimizes $\mathcal{H}$**
  - For example, in the graph shown to the right we have $\mathcal{H}(-1,-1,+1,+1,-1) = (-1)(-1) + (-1)(+1) + (-1)(+1) + (-1)(+1) + (+1)(-1) + (+1)(-1) = -4$
  - An example of a very poor partitioning is $\mathcal{H}(+1,+1,+1,+1,+1,+1) = (+1)(+1) + (+1)(+1) + (+1)(+1) + (+1)(+1) + (+1)(+1) + (+1)(+1) = +6$

$$\mathcal{H}(A,B,C,D,E)$$
$$= AB + AC + BC + BD + CE + DE$$



Hands-On Exercise: Circuit Satisfiability

- **Given a digital circuit (Boolean expression), what inputs, if any, produce a TRUE output?**
  - NP-complete problem
  - Requires $O(2^n)$ function evaluations in the worst case for an arbitrary circuit
  - Generally approached with heuristic approximations
- **Let's program a quantum annealer to propose solutions to the circuit shown to the right**
  - Still approximate—quantum annealers do not guarantee correct solutions

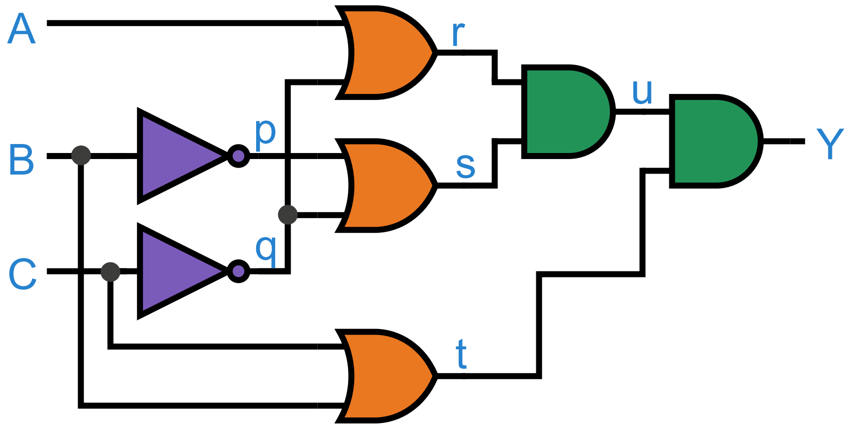


$$Y = (A \vee \neg C) \wedge (\neg B \vee \neg C) \wedge (B \vee C)$$

Hands-On Exercise: Circuit Satisfiability (cont.)

- **Split the problem into subproblems**

$$
\begin{array}{ll}
p = \neg B & t = B \lor C \\
q = \neg C & u = r \land s \\
r = A \lor q & Y = u \land t \\
s = p \lor q & Y = \top
\end{array}
$$



$$Y = (A \lor \neg C) \land (\neg B \lor \neg C) \land (B \lor C)$$

  - Goal is for the quantum annealer to satisfy all of those constraints
- **Identify the building blocks we need to define to implement the above**
  - $Obj_{\neg}(x_0, x_1)$, minimized when $x_0 \neq x_1$
  - $Obj_{\lor}(x_0, x_1, x_2)$, minimized when $x_2 = x_0 \lor x_1$ (i.e., $x_2 = 1$ if at least one of x_0 or x_1 is 1)
  - $Obj_{\land}(x_0, x_1, x_2)$, minimized when $x_2 = x_0 \land x_1$ (i.e., $x_2 = 1$ only if both x_0 and x_1 are 1)
  - $Obj_{\top}(x_0)$, minimized when $x_0 = 1$
- **Combine subproblems into a complete problem for a quantum annealer**
  - $Obj(A, B, C, p, q, r, s, t, u, Y) = Obj_{\neg}(B, p) + Obj_{\neg}(C, q) + Obj_{\lor}(A, q, r) + Obj_{\lor}(p, q, s) + Obj_{\lor}(B, C, t) + Obj_{\land}(r, s, u) + Obj_{\land}(u, t, Y) + Obj_{\top}(Y)$

Hands-On Exercise: Circuit Satisfiability (cont.)

- **Your task**
  - Implement—with pen and paper—as many of the four building blocks as time permits
- **QUBO #1 (True)**

x_0	Penalty
0	🟥
1	🟩

$Obj_{\top}(x_0) =$

- **QUBO #2 (Not)**

x_0	x_1	Penalty
0	0	🟥
0	1	🟩
1	0	🟩
1	1	🟥

$Obj_{\neg}(x_0, x_1) =$

- **QUBO #3 (Or) and QUBO #4 (And)**

x_0	x_1	x_2	Penalty$_{\text{OR}}$	Penalty$_{\text{AND}}$
0	0	0	🟩	🟩
0	0	1	🟥	🟥
0	1	0	🟥	🟩
0	1	1	🟩	🟥
1	0	0	🟥	🟩
1	0	1	🟩	🟥
1	1	0	🟥	🟥
1	1	1	🟩	🟩

$Obj_{\vee}(x_0, x_1, x_2) =$

$Obj_{\wedge}(x_0, x_1, x_2) =$

Circuit Satisfiability: Hint #1

- **Penalty column should be a function of the input columns**
- **For all four QUBOs, we don't want to penalize valid rows at all**
  - Assign a penalty of 0 (technically, a constant function returning 0) to all valid rows
- **For all four QUBOs, we want to penalize invalid rows**
  - Assign a function that maps that row's inputs to a positive number and all other inputs to zero

➔ **What pattern can we use to generate such functions?**
  - Look over the previous QUBO examples in this slide deck
  - Answer in next hint
- **Once you've defined all the per-row functions, add them up to get the final objective function**

Circuit Satisfiability: Hint #2

- **Hint #1 asked how we should define our penalty functions**
- **Remember that all inputs x_i to a QUBO are binary (0 or 1)**
- **Expression to penalize a single input**
  - Goal is to have $f(\text{valid}) = 0$ and $f(\text{invalid}) > 0$
  - If we want $x_0 = 0$ and want to penalize $x_0 = 1$, a penalty function of $f(x) = x$ honors both $f(0) = 0$ and $f(1) > 0$
  - If we want $x_0 = 1$ and want to penalize $x_0 = 0$, a penalty function of $f(x) = 1 - x$ honors both $f(1) = 0$ and $f(0) > 0$

x_0	x_1	x_2	Penalty
0	0	0	$(1 - x_0)(1 - x_1)(1 - x_2)$
0	0	1	$(1 - x_0)(1 - x_1)x_2$
0	1	0	$(1 - x_0)x_1(1 - x_2)$
0	1	1	$(1 - x_0)x_1x_2$
1	0	0	$x_0(1 - x_1)(1 - x_2)$
1	0	1	$x_1(1 - x_2)x_3$
1	1	0	$x_0x_1(1 - x_2)$
1	1	1	$x_0x_1x_2$

- **Expression to penalize multiple inputs**
  - Take the product of all single-input expressions
  - Table to the right shows how to penalize any three-variable input
  - Note that each row evaluates to a penalty of 0 if given any input other than the row's associated $\{x_0, x_1, x_2\}$ values

Circuit Satisfiability: Solution to the TRUE and NOT QUBOs

- **Here's how one can define $Obj_\top$ and $Obj_\neg$:**

QUBO #1: TRUE

x_0	Penalty
0	$1 - x_0$
1	0

$Obj_\top(x_0) = -x_0$

QUBO #2: NOT

x_0	x_1	Penalty
0	0	$(1 - x_0)(1 - x_1)$
0	1	0
1	0	0
1	1	$x_0 x_1$

$Obj_\neg(x_0, x_1) = 2x_0x_1 - x_0 - x_1$

- **Note that we the omitted the constant term (1) in both cases**
  - Doesn't affect the property that all valid rows have the same value (in both cases, -1)
  - Doesn't affect the property that all invalid rows evaluate to larger objective values (in both cases, 0) than any valid row

Circuit Satisfiability: Hint #3

- For QUBO #3 (OR), can you produce an initial set of penalty functions?
- Why can't the corresponding objective function be run as is on a quantum annealer?

Circuit Satisfiability: Hint #4

- Hint #3 asked for an initial version of QUBO #3 (OR)
- The list of initial penalty functions is shown to the right
- The objective function can't be run as is on a quantum annealer because it's not a quadratic function (∴ not a QUBO)

- To complete QUBO #3 we need to get rid of the cubic term
- We can multiply a penalty function by any positive number

➔ What factors should we use for each of the four **invalid** rows?
  - Find each row's cubic coefficient and scale the rows such that ∑positive + ∑negative = 0

QUBO #3: OR

x_0	x_1	x_2	Penalty
0	0	0	0
0	0	1	$(1-x_0)(1-x_1)x_2$
0	1	0	$(1-x_0)x_1(1-x_2)$
0	1	1	0
1	0	0	$x_0(1-x_1)(1-x_2)$
1	0	1	0
1	1	0	$x_0x_1(1-x_2)$
1	1	1	0

$$Obj_V(x_0,x_1,x_2) = \boxed{2x_0x_1x_2} - x_0x_1 - 2x_0x_2 - 2x_1x_2 + x_0 + x_1 + x_2$$

Circuit Satisfiability: Solution to the OR QUBO

- **What factor should we use for each of the four invalid rows?**
  - Find each row's cubic coefficient and scale the rows such that ∑positive + ∑negative = 0

QUBO #3: OR

x_0	x_1	x_2	Hint #4 penalty	Cubic coefficient	Final penalty	Obj_V
0	0	0	0	0	0	0
0	0	1	$(1-x_0)(1-x_1)x_2$	$-1\cdot -1\cdot +1 = +1$	$(1-x_0)(1-x_1)x_2$	1
0	1	0	$(1-x_0)x_1(1-x_2)$	$-1\cdot +1\cdot -1 = +1$	$(1-x_0)x_1(1-x_2)$	1
0	1	1	0	0	0	0
1	0	0	$x_0(1-x_1)(1-x_2)$	$+1\cdot -1\cdot -1 = +1$	$x_0(1-x_1)(1-x_2)$	1
1	0	1	0	0	0	0
1	1	0	$x_0x_1(1-x_2)$	$+1\cdot +1\cdot -1 = -1$	$3x_0x_1(1-x_2)$	3
1	1	1	0	0	0	0

$$Obj_V(x_0,x_1,x_2) = x_0x_1 - 2x_0x_2 - 2x_1x_2 + x_0 + x_1 + x_2$$

Circuit Satisfiability: Solution to the AND QUBO

- **Solving AND follows exactly the same pattern as solving OR**
  - Find each row's cubic coefficient and scale the rows such that ∑positive + ∑negative = 0

QUBO #4: AND

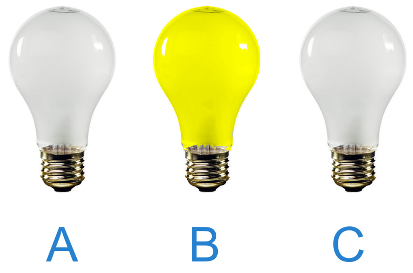
x_0	x_1	x_2	Initial penalty	Cubic coefficient	Final penalty	$Obj_\wedge$
0	0	0	0	0	0	0
0	0	1	$(1-x_0)(1-x_1)x_2$	$-1\cdot-1\cdot+1=+1$	$3(1-x_0)(1-x_1)x_2$	3
0	1	0	0	0	0	0
0	1	1	$(1-x_0)x_1x_2$	$-1\cdot+1\cdot+1=-1$	$(1-x_0)x_1x_2$	1
1	0	0	0	0	0	0
1	0	1	$x_0(1-x_1)x_2$	$+1\cdot-1\cdot+1=-1$	$x_0(1-x_1)x_2$	1
1	1	0	$x_0x_1(1-x_2)$	$+1\cdot+1\cdot-1=-1$	$x_0x_1(1-x_2)$	1
1	1	1	0	0	0	0

$$Obj_\wedge(x_0,x_1,x_2) = x_0x_1 - 2x_0x_2 - 2x_1x_2 + 3x_2$$

Two Models of Quantum Computing

- **Circuit-model quantum computing and annealing-model quantum computing seem to have nothing in common**
- **What's the connection?**
- **Let's revisit our problem Hamiltonian for "1 of 3":**

$$\mathcal{H}_{1\text{of}3}(\sigma_0, \sigma_1, \sigma_2) = \sigma_0 + \sigma_1 + \sigma_2 + \sigma_0\sigma_1 + \sigma_0\sigma_2 + \sigma_1\sigma_2$$



- **Treat each term of $\mathcal{H}_{1\text{of}3}$ as implicitly being of the form $a_0 \otimes a_1 \otimes a_2$ where $a_0, a_1, a_2 \in \{I, Z\}$**
  - Each explicit σ_i in $\mathcal{H}_{1\text{of}3}$ replaced by a Z and each implicit σ_i replaced by an I:

$$\mathcal{H}_{1\text{of}3} = (Z \otimes I \otimes I) + (I \otimes Z \otimes I) + (I \otimes I \otimes Z) + (Z \otimes Z \otimes I) + (Z \otimes I \otimes Z) + (I \otimes Z \otimes Z)$$

  - Implication is that $\mathcal{H}_{1\text{of}3}$ is an 8×8 matrix

Reminder: $I = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$ and $Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$

Two Models of Quantum Computing (cont.)

- **From the previous slide,**
  $\mathcal{H}_{1of3} = (Z \otimes I \otimes I) + (I \otimes Z \otimes I) + (I \otimes I \otimes Z) + (Z \otimes Z \otimes I) + (Z \otimes I \otimes Z) + (I \otimes Z \otimes Z)$
- **Because both I and Z are diagonal matrices, $\mathcal{H}_{\mathbf{1of3}}$ is a diagonal matrix:**

$$\mathcal{H}_{1\mathrm{of}3} = \begin{array}{c}\begin{pmatrix} 6 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}\end{array} \begin{array}{c} 111 \\ 110 \\ 101 \\ 100 \\ 011 \\ 010 \\ 001 \\ 000 \end{array}$$

- **This matrix's minimal elements lie exactly on the "1 of 3" rows!**
- **But how can we use $\mathcal{H}_{\mathbf{1of3}}$ in a quantum circuit that actually finds those minimal elements?**

➔ **Employ a general-purpose quantum optimization algorithm**
  – And this is the subject of the next section of the tutorial…

Further Quantum Algorithms

Agenda

- **Part I: Quantum-computing fundamentals**
  - High-level motivation, history, and status
  - Qubits, multi-qubit states, and quantum measurement

Morning break

- **Part II: Circuit-model quantum computing**
  - Review of notation
  - Quantum gates and quantum circuits

Lunch

  - Basic quantum algorithms
- **Part III: Quantum annealing**

Afternoon break

- **Part IV: Further quantum algorithms**
  - Quantum Alternating Operator Ansatz (QAOA)
  - Advancements in quantum algorithms
  - Concluding remarks

Target for NISQ Evaluation: Quantum Optimization Heuristics

- **Instances of combinatorial optimization problems**
  - Current approach: classical heuristics algorithms
  - NISQ hardware provides means to evaluate quantum heuristic algorithms

One strategy: Try the simplest algorithm that might work!

- **Quantum heuristics**
  - Combine cost-function-based operator with a mixing operator
  - AQO, QA, QAOA
  - Other ideas welcome!

- **Evaluation techniques**
  - Analytic, numerical, experimenting on NISQ hardware

Target for NISQ Evaluation: Quantum Optimization Heuristics

- **Diverse optimization goals**
  - Exact optimization with guarantees
  - Approximate opt. with guarantees
  - Good heuristic, without guarantees
  - Fair sampling; portfolio sampling

- **Sampling goals, e.g. for machine learning (ML)**
  - Sampling thermal distribution corresponding to cost function (sampling from Boltzmann distributions used in ML)

Quantum Optimization Algorithms: AQO, QA, QAOA

- **Common elements: Given cost function C(z),**
- **Phase separation operator based on the cost function,**
  - Usually based on $H_P = -C(z)|z\rangle\langle z|$, often including additional “penalty terms” to enforce constraints
- **Driver/Mixing operator**
  - Most frequently $H_M = \sum_j Xj$, though we will shortly see other mixers

AQO

- Evolution under $\boldsymbol{H}(\boldsymbol{t}) = \boldsymbol{a}(\boldsymbol{t})H_P + \boldsymbol{b}(\boldsymbol{t})H_M$
- Slowly enough to stay in the ground subspace

QA

- Evolution under $\boldsymbol{H}(\boldsymbol{t}) = \boldsymbol{a}(\boldsymbol{t})H_P + \boldsymbol{b}(\boldsymbol{t})H_M$
- Many quick runs, thermal effect contribute

QAOA

- Alternate application of H_P and H_M
- For $\boldsymbol{p}$ alterations, the parameters are $\mathbf{2}\boldsymbol{p}$ times/angles $\boldsymbol{\gamma_1}, \boldsymbol{\beta_1}, \dots \boldsymbol{\gamma_P}, \boldsymbol{\beta_p}$

Quantum Optimization Algorithms: AQO, QA, QAOA

- **Common elements: Given cost function C(z),**
- **Phase separation operator based on the cost function,**
  - Usually based on $H_P = -C(z)|z\rangle\langle z|$, often including additional “penalty terms” to enforce constraints
- **Driver/Mixing operator**
  - Most frequently $H_M = \sum_j Xj$, though we will shortly see other mixers

AQO

- Evolution under

$\boldsymbol{H}(\boldsymbol{t}) = \boldsymbol{a}(\boldsymbol{t})H_P + \boldsymbol{b}(\boldsymbol{t})H_M$

- Slowly enough to stay in the ground subspace

QA

- Evolution under

$\boldsymbol{H}(\boldsymbol{t}) = \boldsymbol{a}(\boldsymbol{t})H_P + \boldsymbol{b}(\boldsymbol{t})H_M$

- Many quick runs, thermal effect contribute

QAOA

- Alternate application of H_P and H_M
- For ***p*** alterations, the parameters are $\boldsymbol{2p}$ times/angles $\boldsymbol{\gamma_1, \beta_1, \ldots \gamma_P, \beta_p}$

Quantum Alternating Operator Ansatz

- **Based on the Quantum Approximate Optimization Algorithm**
  - A gate model heuristic due to Farhi et al.
  - Iterates between two Hamiltonians, p times
    - Phase separation (cost function dep.)
    - Mixing
- **Early results by Farhi and co-authors**
  - $p \to \infty$: from AQO
    - Converges to optimum for $p \to \infty$
  - $p = 1$: from IQP circuits
    - Provably hard to sample output efficiently classically (up to standard complexity theory conjectures)
    - Beat existing classical approx. ratio on MaxE3Lin2, but inspired better classical algorithm
- **Later results**
  - New algorithm for Grover's unstructured search problem
    - achieves $\sqrt{N}$ query complexity by different means

Z. Jiang et al., **Near-optimal quantum circuit for Grover's unstructured search using a transverse field**, PRA 95 (6), 062317, 2017.

Quantum Alternating Operator Ansatz

- **Advantages**
  - Supports more general mixing operators, providing massive improvements in implementability
  - Incorporates hard constraints into mixer instead of as a penalty term; algorithm explores only feasible subspace, often exponentially smaller, so more efficient search
  - Reworked QAOA acronym to support applications to exact optimization and sampling as well as approximate optimization
- **We have mapped many problems to extended QAOA formalism**
  - Focused on scheduling and network problems
  - Including Max-*k*-colorable subgraph

S. Hadfield et al., **From the Quantum Approximate Optimization Algorithm to a Quantum Alternating Operator Ansatz**, Algorithms 12 (2), 34 2019, arXiv:1709.03489

Gate-model quantum optimization heuristic:
Quantum Alternating Operator Ansatz

Generalization of Quantum Approximation Optimization Algorithm

$$Q_p(\boldsymbol{\beta},\boldsymbol{\gamma}) = U_{\mathrm{M}}(\beta_p)U_{\mathrm{P}}(\gamma_p)\cdots U_{\mathrm{M}}(\beta_1)U_{\mathrm{P}}(\gamma_1)\,|\boldsymbol{\beta},\boldsymbol{\gamma}\rangle$$

$$\boxed{e^{-i\gamma H_{\mathrm{P}}}}$$

$$\boxed{\prod_j e^{\boxed{-i\beta_k H_j}}}$$

$$\boxed{|\boldsymbol{\beta},\boldsymbol{\gamma}\rangle = Q_p(\boldsymbol{\beta},\boldsymbol{\gamma})\,|s\rangle}$$

Phase separator: unitary for which

- The energy spectrum of **H_P encodes the problem's objective function**
- $H_P = -C(z)|z\rangle\langle z|$

Mixer: unitary which:

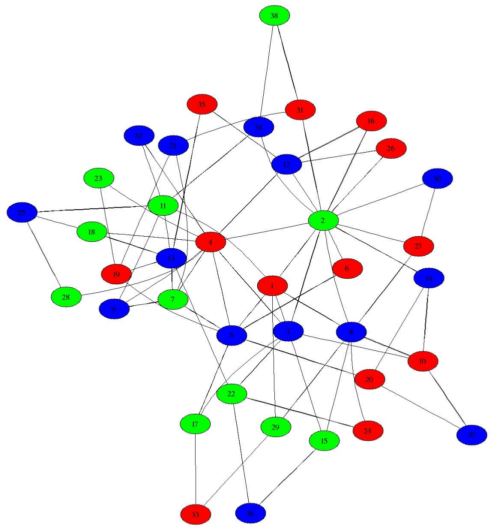
- Preserves the feasible subspace
- Provides nonzero transitions between all feasible states
- Not necessarily time evolution of a single local Hamiltonian
- β_k depends on the level $1 \leqq k \leqq p$, but independent of H_j

Initial state which:

- is a superposition of one or more solutions in the feasible subspace
- can be prepared efficiently

Example: QAOA for Max-κ-Colorable Subgraph

- Problem: Given a graph $G = (V, E)$, and k colors $1, \dots, k$, find a color assignment maximizing the # of properly colored edges
- Properly colored edge means endpoint vertices have been assigned different colors



- "One-hot" Encoding: nk variables

  $x_{uj} = 1$ iff vertex *u* is colored color *j*

- Must avoid invalid colorings
  - e.g. if a vertex is labeled as both red **and** blue, or not colored at all

- Optimization: Write cost function as

  $C(x) = m - \sum_{(uv)\in E}^{m} \sum_{j=1}^{k} x_{uj} x_{vj}$

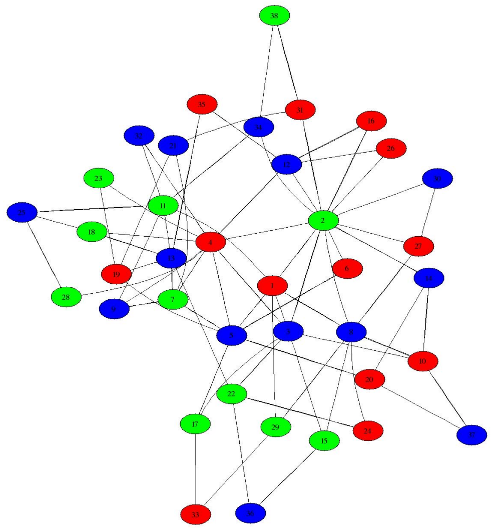
- Requires *n* constraints: one for each vertex *u*

$$\sum_{j=1}^{k} x_{uj} = 1$$

Example: QAOA for Max-κ-Colorable Subgraph

- **Could add constraints to the cost function to enforce penalties**
  - standard approach in quantum annealing
- **Better: design mixer to keep the evolution in the feasible subspace, which keeping the phase separator simple**

- **Use swap mixer on the colors rather than bit flip mixer**
  - instead of $\sum_j Xj$, use sum of swap operators, $|00\rangle\langle 00| + |10\rangle\langle 01| + |01\rangle\langle 10| + |11\rangle\langle 11|$ one for each pair of (adjacent) colors



- **Feasible subspace is exponentially smaller search space than entire Hilbert space**
  - While still exponentially large
- **Initial state choice**
  - Any classical feasible state
    - e.g., all colored red
  - Any superposition of feasible states
    - e.g., superposition of all colors (W state)

Partitioned Mixer Example: Max-κ-Colorable Subgraph

Partitioned Mixers: Products of $U_Bv = e^{-iBv}$. Don't commute, so different orders give different mixers

$$U_{\text{parity}}(\beta) = U_{\text{last}}(\beta) U_{\text{even}}(\beta) U_{\text{odd}}(\beta),$$

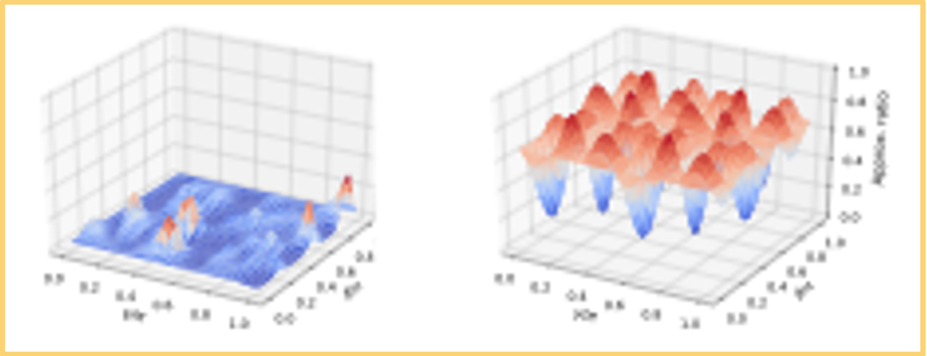
where

$$U_{\text{odd}}(\beta) = \prod_{a \text{ odd},\ a \neq n} e^{-i\beta(X_a X_{a+1} + Y_a Y_{a+1})},$$

$$U_{\text{even}}(\beta) = \prod_{a \text{ even}} e^{-i\beta(X_a X_{a+1} + Y_a Y_{a+1})},$$

$$U_{\text{last}}(\beta) = \begin{cases} e^{-i\beta(X_d X_1 + Y_d Y_1)}, & \kappa \text{ odd}, \\ I, & \kappa \text{ even}. \end{cases}$$

Numerical Results: Advantage of XY-mixer



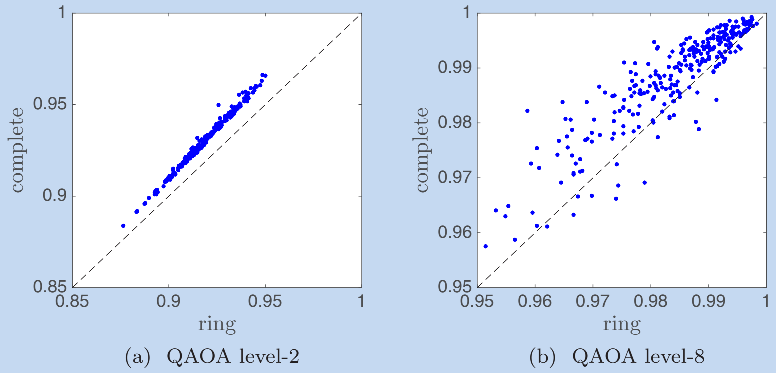
Numerical obtained approximation ratio for 3-coloring of a triangle graph. (Left) QAOA with standard X-mixer. (Right) QAOA with XY-mixer.

Confirmed advantage of mixers that maintain evolution within feasible subspace

Ratio of dimension of feasible subspace to full Hilbert space of shrinks exponentially with n:

$$\frac{\dim(\mathcal{H}_{\mathrm{fea}})}{\dim(\mathcal{H})} = \frac{\kappa^n}{2^{n\kappa}} = \left(\frac{\kappa}{2^\kappa}\right)^n$$

Ring vs. complete mixer



Approximation ratio on all 4-colorable, connected graphs with 7 vertices

Trade off in depth between initial state generation and QAOA iterations

- W state vs single classical state

Zhihui Wang, Nicholas C. Rubin, Jason M. Dominy, Eleanor G. Rieffel, **XY-mixers: analytical and numerical results for QAOA,** *PRA 101 (1), 012320, 2020*

XY-mixer retains advantage under noise

M Streif, M Leib, F Wudarski, E Rieffel, Z Wang, **Quantum algorithms with local particle number conservation: noise effects and errorcorrection**, Physical Review A 103 (4), 042412, 2021

Parameter Setting for QAOA

McClean, Boixo, Smelyanskiy, Babbush, Neven, Barren plateau paper (Nature Comm, 2018)

- Processes for finding parameters can get stuck in large, featureless plateaux

QAOA for Grover's unstructured search – simple, periodic parameters recover square-root query complexity

- (Not a Trotterization of Roland-Cerf adiabatic algoruthm, since Trotterization loses the square-root query complexity)

For the trivial problem of MaxCut on ring, good parameters harder to find than might be expected

- Fermionic view helps with analysis
- Control theory gives insight into parameter landscape
- Mbeng, Fazio, Santoro advanced our results, finding a regular set of parameters that approximated adiabatic schedule arXiv:1906.08948

Bravyi et al., Classical Algorithms for Quantum Mean Values, arXiv:1909.11485

Z. Jiang et al., **Near-optimal quantum circuit for Grover's unstructured search using a transverse field**, PRA 95 (6), 062317, 2017

Z. Wang et al., **The Quantum Approximation Optimizatio n Algorithm for MaxCut: A Fermionic View**, PRA 97 (2), 8)

Analytical Framework for QAOA: Overview

Key quantities for parameterized quantum circuits such as QAOA expressed as expectation values

Can obtain *exact parameter series expressions* for these quantities via “the Heisenberg picture”

For alternating ansätze, similar terms reappear in the series as the number p of layers is increased

For QAOA, the resulting terms of the series intuitively relate to classical functions (derived from the cost function) that reflect problem structure and choice of mixing operator

Applications:

- ***leading-order behavior and series approximations for QAOA$_p$***
- **encapsulate several existing *performance results for QAOA$_1$***
- **extensions to QAOA$_p$ with $p \geq 1$**
- **additional results including *generalizations* to constrained optimization**

Stuart Hadfield, Tad Hogg, Eleanor G. Rieffel, Analytical Framework for Quantum Alternating Operator Ansätze, arXiv:2105.06996

Connections between quantum annealing schedules and QAOA parameter setting

Brady, Baldwin, Bapat, Kharkov, V. Gorshkov arXiv:2003.08952

- generically, for a fixed amount of time, optimal procedure has bang-bang structure of QAOA at the beginning and end, but a smooth annealing structure in between

Zhou, Wang, Choi, Pichler, Lukin arXiv:1812.01041

- Learned optimal parameters
- Identified regular subfamily of optimal parameters, resembling digitized smooth evolution
  - For easy problems, resembled adiabatic schedules
  - For hard problems, resembled diabatic schedules

Mbeng, Fazio, Santoro, Connects adiabatic adiabatic schedules with optimal QAOA parameters for the easy problem of MaxCut on a Ring, arXiv:1906.08948

Yang, Rahmani, Shabani, H Neven, C Chamon, PRX 2017

- Pontryagin's minimum principle implies optimal evolution schedules must be bang-bang, up to some caveats

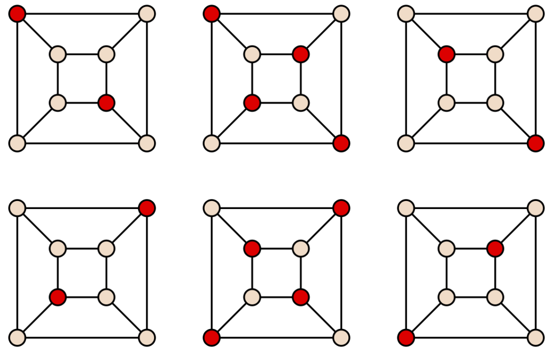
Status of QAOA

- **Unclear as of yet as to whether it provides a speed up beyond a few examples**
  – that is true for any NISQ quantum optimization algorithm!
- **Parameter setting is challenging**
  – we will return to this topic shortly
  – relation between parameter setting in QAOA and annealing schedule choice in quantum annealing
- **Two types of QAOA algorithms:**
  – Gate-model, with mixing operator and phase separation operator made up of 1- and 2-qubit gates
  – Large-scale Hamiltonians applied, possible on some hardware being developed for both annealers and universal QCs

Cf. Neill et al., A blueprint for demonstrating quantum supremacy with superconducting qubits (2017)

Exercise: Map Independent Set to QAOA

- **MIS (Map Independent Set) Problem: Given a graph G, find largest mutually disjoint subset of vertices $S \subset V$,** where $|V| = n$



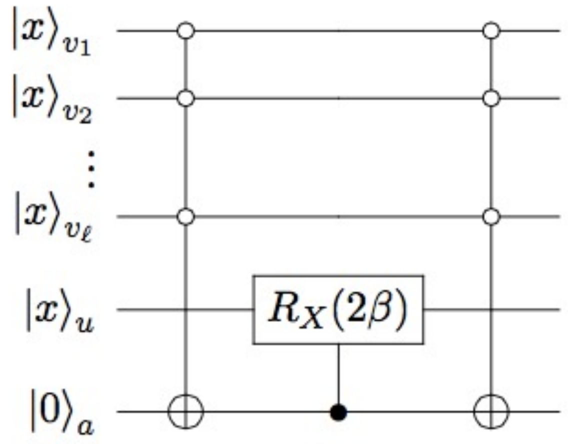
- **Specify**
  - Representation (meaning of binary variables)
  - Cost function
  - Mixing operator
  - Initial state

- **Feel free to express in terms of classical operations, including controlled-NOTs, even multiply controlled-NOTs**
  - Then convert to Hamiltonian or unitary

- **Bonus exercise: Map Max-3-Colorable-Induced Subgraph to QAOA**
  - Problem statement: Given a graph G, find largest subset of vertices $S \subset V$, such that all edges between $s \in S$ are properly colored (endpoints have different colors)
  - Hint: Use concepts from both Max-Colorable-Subgraph and MIS mappings

A Solution: One Possible MIS to QAOA Mapping

- **Representation**
  - Use $\boldsymbol{n}$ binary variables $\boldsymbol{x_u}$, one for each vertex in the graph. These variables will be mapped to $\boldsymbol{n}$ qubits. Variable $\boldsymbol{x_u} = \mathbf{1}$ indicates vertex $\boldsymbol{u} \in \boldsymbol{S}$
- **Initial State:** Empty set: $|\mathbf{00\ldots0}\rangle$



- **Cost function**
  - cost function to maximize $\quad \boldsymbol{c(S) = |S| = \sum_{u \in V} x_u}$
  - maps to Hamiltonian $\quad \boldsymbol{C = \frac{1}{2}\sum_{u \in V}(I - Z_j)}$
- **Mixer**
  - Can add (or remove) a vertex $\boldsymbol{u}$ to $\boldsymbol{S}$ if none of $\boldsymbol{u}$'s neighbors are already in $\boldsymbol{S}$
  - For each vertex, construct the partial mixer (a controlled X-rotation)

$$\boldsymbol{exp(-i\beta B_u) = \Lambda_{\bar{x}_{v_1}\bar{x}_{v_2}\ldots\bar{x}_{v_\ell}}\,(\exp(-i\beta X_u))}$$

where $\quad \boldsymbol{ndhd(u) = \{v_1, \ldots v_\ell\}}$

  - Total mixer: $\boldsymbol{U_M(\beta) = \prod_{u \in V} exp(-i\beta B_u)}$

where we must choose a product order

For more explanation, and answer to bonus exercise, see

Hadfield et al. From the Quantum Approximate Optimization Algorithm to a Quantum Alternating Operator Ansatz. arXiv:1709.03489 (2017)

Agenda

- **Part I: Quantum-computing fundamentals**
  - High-level motivation, history, and status
  - Qubits, multi-qubit states, and quantum measurement

Morning break

- **Part II: Circuit-model quantum computing**
  - Review of notation
  - Quantum gates and quantum circuits

Lunch

  - Basic quantum algorithms
- **Part III: Quantum annealing**

Afternoon break

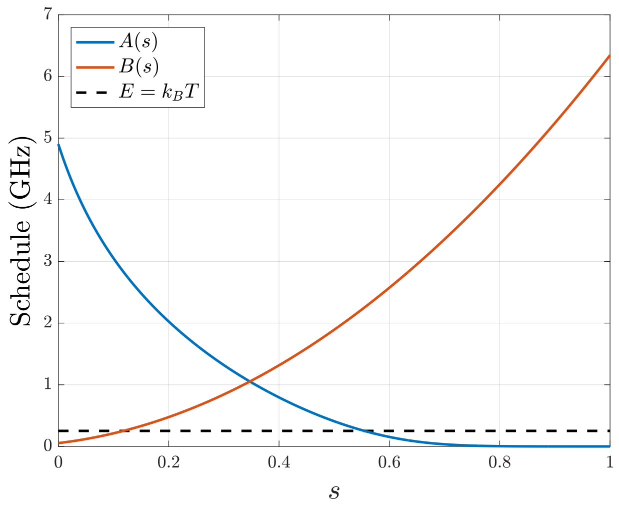
- **Part IV: Further quantum algorithms**
  - Quantum Alternating Operator Ansatz (QAOA)
  - Advancements in quantum algorithms
  - Concluding remarks

Pause Feature on the D-Wave 2000Q

- **Annealing Hamiltonians**
  - $H(s) = A(s)H_d + B(s)\,H_p,\ 0 \leqq s \leqq 1$
- **Annealing schedule: s(t)**

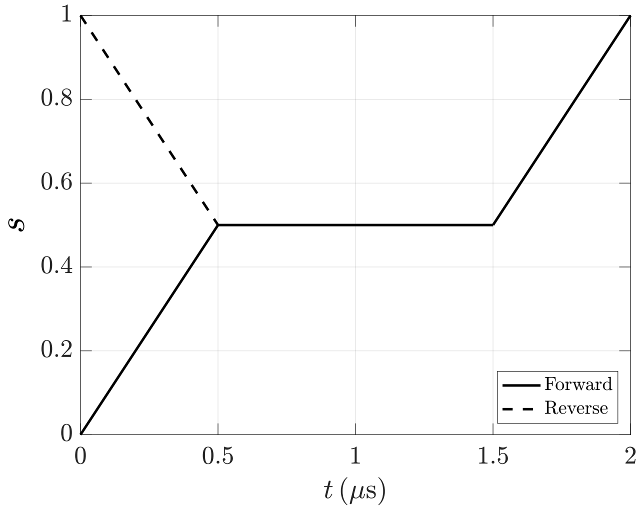
- **Previously, schedule restricted to adjustment of overall time, t_a**
  - $s(t) = t/t_a$

- **New pause feature supports more flexible schedules**
  - given total anneal time of $t_a + t_p$
  - anneal normally to pause location s_p
  - hold H constant from s_p to $s_p + t_p$
  - continue normally after $s_p + t_p$



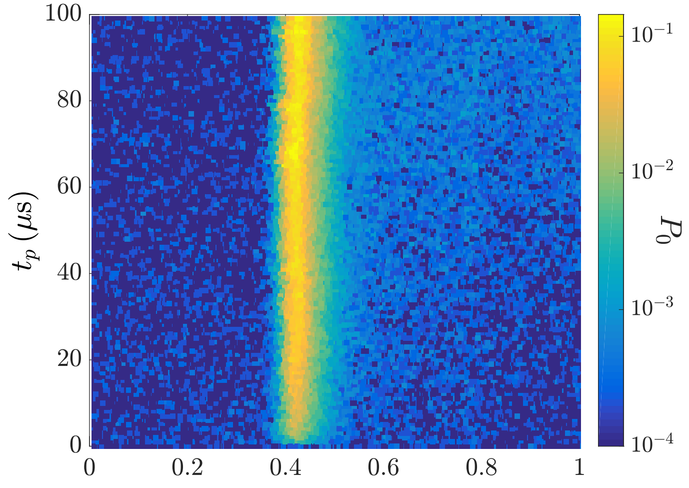
Standard D-Wave 2000Q schedule

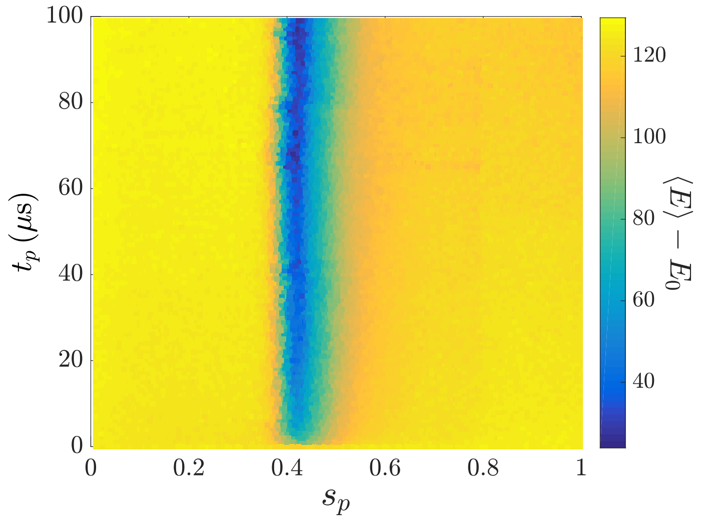
Example pause schedule



Pause Results for One Typical Problem

- **Performance for pause schedules**
  - heat map of probability of solution P_0 as function of pause location s_p and pause length t_p
  - heat map of average energy (above ground state) of solution as function of pause location s_p and pause length t_p
- **Results for single 800-qubit problem**
  - total anneal time ta = 1 μs
  - Each point: 10,000 anneals, using 5 gauges
  - $P_0 = 10^{-4}$ for anneal without pause
- **Orders of magnitude improvement for pausing in narrow region of location parameter s_p**

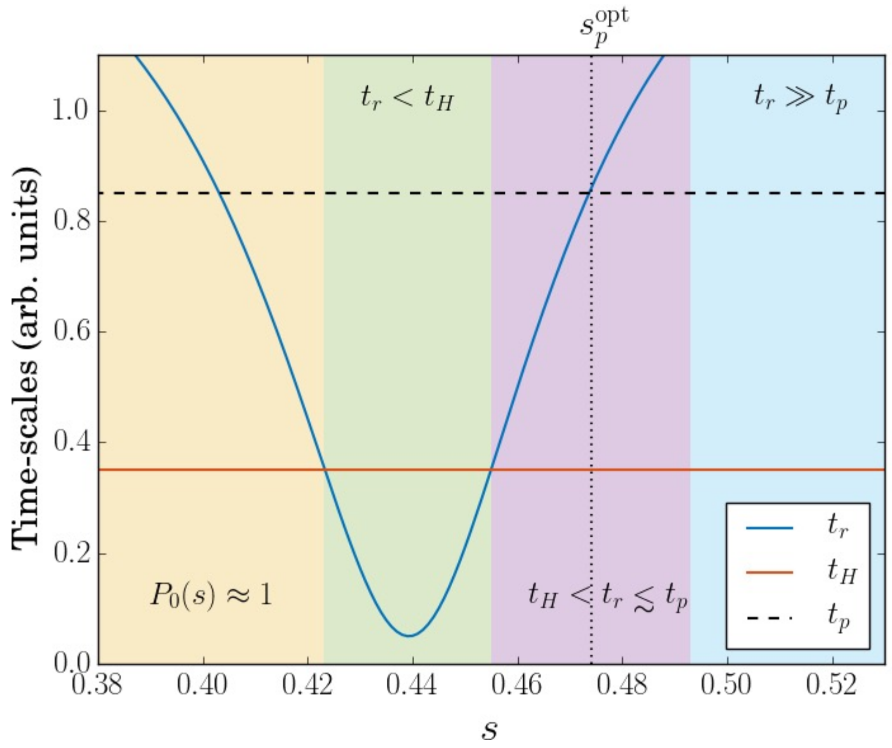




J. Marshall, D. Venturelli, I. Hen, E. Rieffel, The power of pausing: advancing understanding of thermalization in experimental quantum annealers, Physical Review Applied 11 (4), 044083, 2019, arXiv:1810.05881

Theory and Relevant Time Scales

- **Early times: Ground state is well-separated by rest of spectrum, so $P_0 \sim 1$**
- **Gap narrows: $t_r < t_H$ Relaxation rate increases, potential for thermal excitation leading to instantaneous thermalization**
- **Gap widens: $t_H < t_r \lesssim t_p$ Instantaneous thermalization no longer occurs, but a pause may enable significant thermalization**
- **Late times: $t_p << t_r$ Energy levels well-separated so even with a pause of length t_p, thermalization cannot occur**

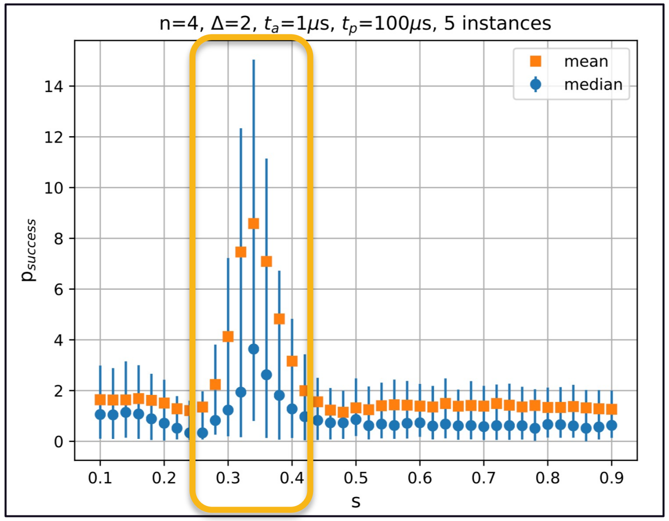


Cartoon of distinct regions with different behavior focusing on most dynamic part of the anneal

- *t_r = relaxation rate*
- *t_H = Hamiltonian evolution time scale.*
  For t_r < t_H, the system instantaneously thermalizes (we plot t_H as a line only for the purpose of easy visualization of the regions)

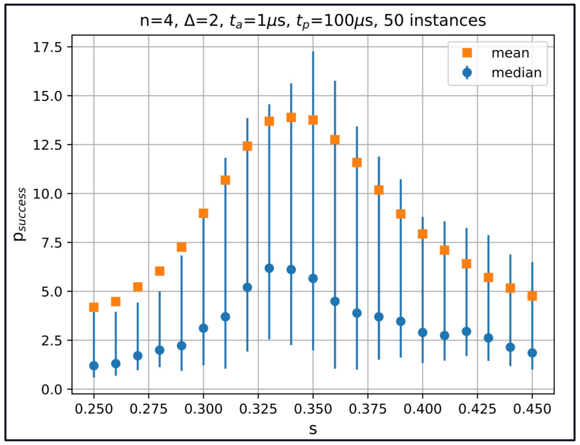
Effectiveness of Pause on Embedded Problems

- **Problem class: *Minimum Weighted Spanning Tree with degree constraints***
  - Factor of five improvement in the mean probability of success observed for 50, N=4 problem instances (20–35 variables; 50–125 qubits when embedded)
  - Consistent pause location across instances
  - Similar results for N=5 problems (not shown)



Mean and median probability of success as a function of the annealing pause for 50 N=4 instances, 1 ms anneal, 50K reads, J_{ferro} = -2. Pause location for
- *(left) ranging from 0.1 to 0.9*
- *(bottom) near the peak success prob.*

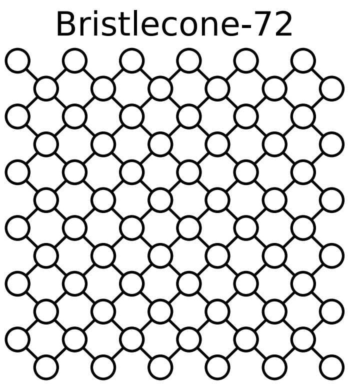
(Error bars are at the 35th and 65th percentiles)

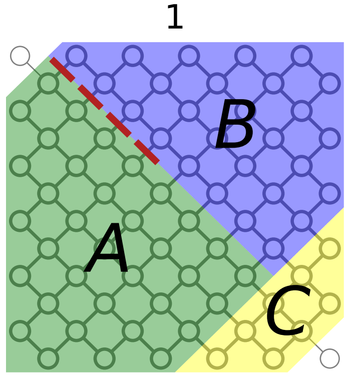


Open question: why is the effect less pronounced for these embedded problems?

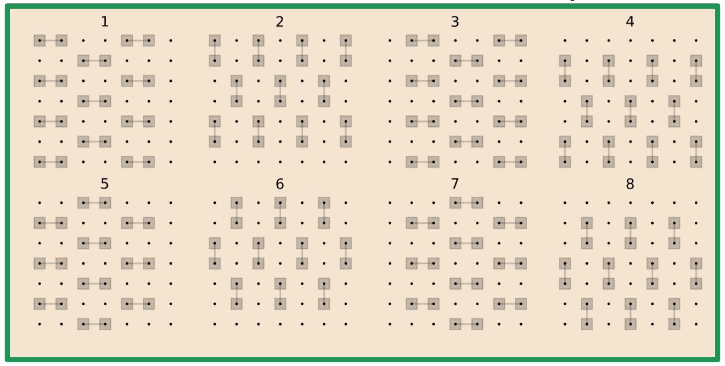
Classical HPC Simulation of Quantum Circuits

- **Advanced the state-of-the-art**
  - can simulate larger quantum circuits than any previous approach
  - judicious use of cuts within a tensor network contraction
  - HPC memory tricks and trade-offs
  - can flexibly incorporate fidelity goal
- **Largest computation run on NASA HPC clusters**
  - 60-qubit subgraph, depth 1+32+1
  - 116,611 processes on 13,059 nodes, peak of 20 PFLOPS, 64% of max
  - across Pleiades, Electra, Hyperwall
- **Applications**
  - quantum supremacy experiments
  - benchmark emerging quantum hardware
  - empirically explore quantum algorithms





Computed exact amplitudes for 72 qubit Bristlecone random circuit, depth 1+32+1



Villalonga et al., ***A flexible high-performance simulator for the verification and benchmarking of quantum circuits implemented on real hardware***. arXiv:1811.09599

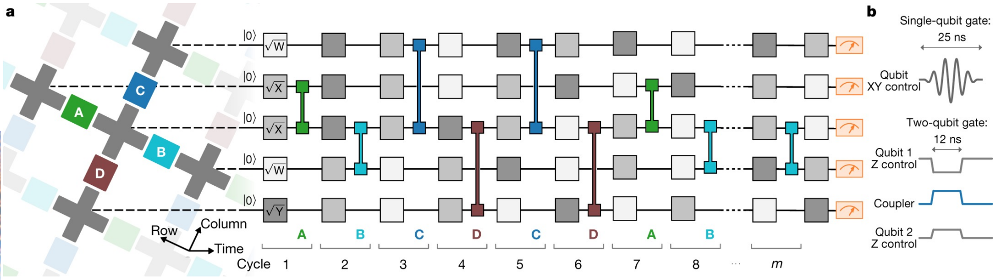
Villalonga et al., ***Establishing the Quantum Supremacy Frontier with a 281 Pflop/s Simulation***, arXiv:1905.00444

Quantum Supremacy

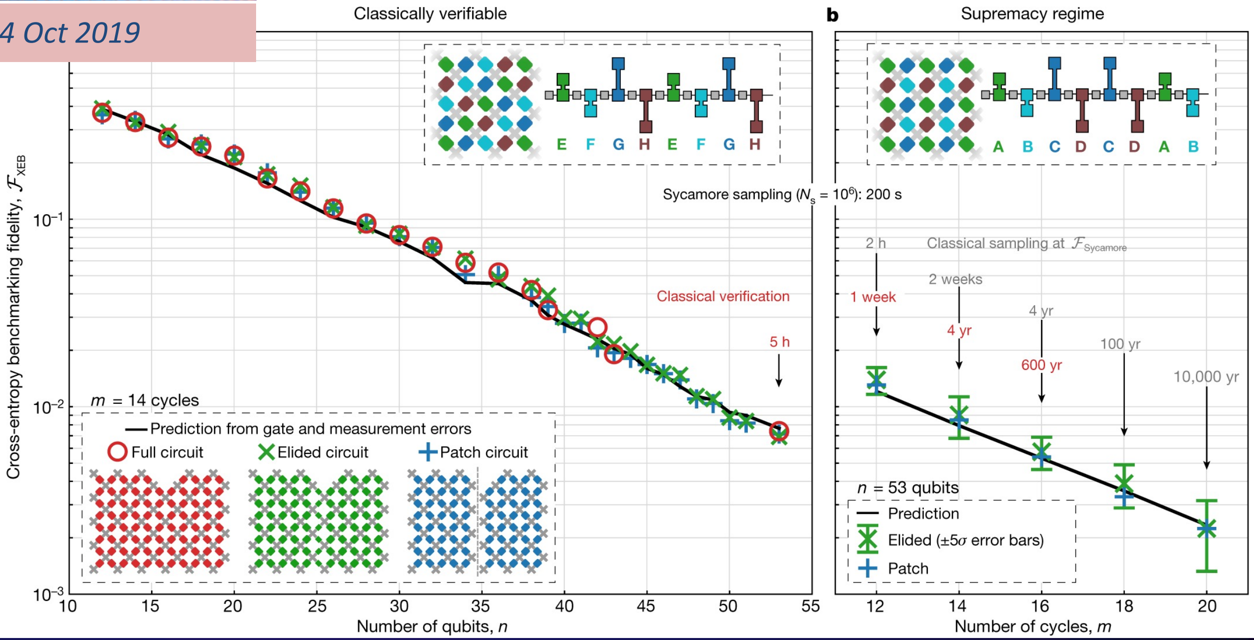
- **How do you verify a computation that no other hardware can do?**
  - check smaller version
  - check pieces
  - check variants that are simpler to simulate
- **Entering an era of unprecedented ways to explore quantum algorithms**
- **Era of quantum heuristics.**
- **These explorations will broaden the known application of quantum computing**

nature
QUANTUM SUPREMACY
Classical supercomputer outperformed by quantum chip for the first time

24 Oct 2019



***Example random quantum circuit**. Every cycle includes a layer of single- and two-qubit gates. The single-qubit gates are chosen randomly from a set of three gates. The sequence of two-qubit gates follow a tiling pattern, coupling each qubit sequentially to its four neighbors.*



Unified Framework for Optimization → PySA

- Modern C++17 with template metaprogramming for high level of abstraction
- Compile time optimization to improve performance

- **UFO features and state-of-the-art algorithm implementations:**
  - Compile time optimization
  - Optimize natively k-local Ising Hamiltonians, with $k \geq 2$
  - Algorithms:
    - Parallel Tempering
    - Ergodic and non-ergodic Isoenergetic cluster moves
    - Approximate solution using mean-field theory
    - [Planned in PySA] Thermal cycling
    - [Planned in PySA] Simulated Quantum Annealing
    - [Planned in PySA] Hard-embedding of constraints
  - Fully working C-APIs
  - Python libraries with full state access [Planned in PySA]
- [Planned in PySA] GPU implementation

NASA UFO

We continuously update UFO [PySA] with optimized code for state-of-the-art classical optimization, including physics inspired approaches we have developed

Qubit Routing on NISQ Processors

Compilation of an algorithms to a NISQ processor requires

- Decomposition into native gates
- Qubit routing

Qubit routing moves qubit states to locations where the required gates can act on them

- Can be done by inserting SWAPs into a circuit composed of native

Main result: Any unordered set of k-qubit gates on distinct k-qubit subsets of n logical qubits can be ordered and parallelized in $O(n^{k-1})$

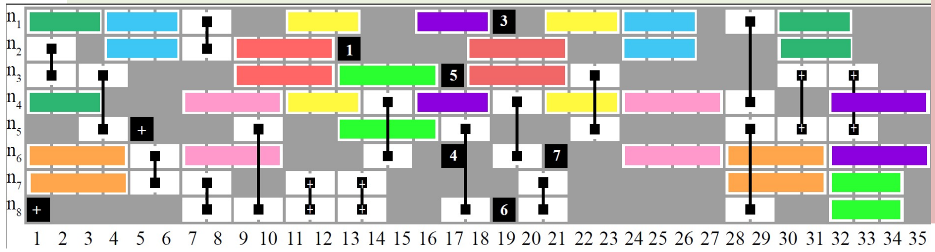
Applications:

- QAOA
- Quantum simulation of Fermionic systems
  - Single Trotter step
  - uses fermionic swap gates to change ordering of a Jordan-Wigner string

Bryan O'Gorman, William J. Huggins, Eleanor G. Rieffel, K. Birgitta Whaley, **Generalized swap networks for near-term quantum computing**, arXiv:1905.05118

Temporal Planning for Qubit Routing

- **Qubit routing can be phrased as a temporal planning problem**
  - minimize *makespan*
- **Can incorporate**
  - nearest-neighbor h/w constraints
  - varying quantum gate times
  - crosstalk
- **Initial experiments focused on**
  - QAOA circuits for Maxcut because of their high number of commuting gates
  - Rigetti hardware proposal with varying gates between neighboring qubits



- **Mapped circuit compilation problem to a temporal planning problem, compared state-of-the-art temporal planners**
- **Demonstrated temporal planning is a viable approach to circuit compilation**
- **More recently, combined temporal planning with constrained programming**
- **Currently extending to more complex circuits such as QAOA for graph coloring**
- **Expressive framework can incorporate further hardware requirements, incl. noise tradeoffs, as we learn them**

D. Venturelli et al., Compiling quantum circuits to realistic hardware architectures using temporal planners, *Quantum Science and Technology* (2018)

Classical HPC Simulation of Quantum Circuits

Advanced the state-of-the-art

- simulates larger quantum circuits than previous approaches

- judicious use of cuts within a tensor network contraction

- HPC memory tricks and trade-offs

- can flexibly incorporate fidelity goal

Largest computation run on NASA HPC clusters

- 60-qubit subgraph, depth 1+32+1

- 116,611 processes on 13,059 nodes, peak of 20 PFLOPS, 64% of max

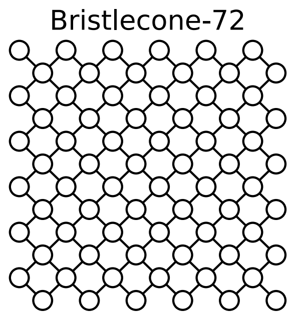
- across Pleiades, Electra, Hyperwall

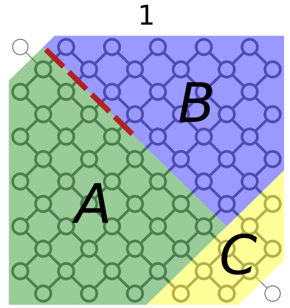
Applications

- quantum supremacy experiments

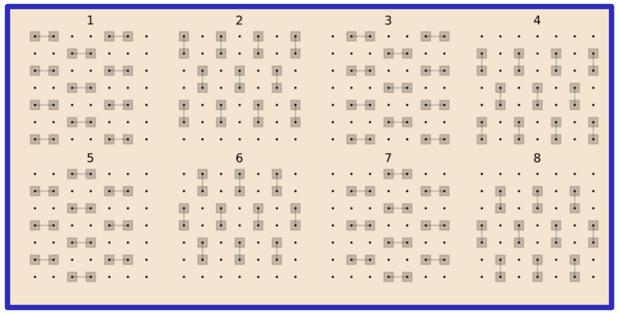
- benchmark emerging quantum hardware

- empirically explore quantum algorithms





Computed exact amplitudes for 72 qubit Bristlecone random circuit, depth 1+32+1



Villalonga et al., ***A flexible high-performance simulator for the verification and benchmarking of quantum circuits implemented on real hardware***. arXiv:1811.09599
Villalonga et al., ***Establishing the Quantum Supremacy Frontier with a 281 Pflop/s Simulation***, arXiv:1905.00444

Open source qFlex code: https://github.com/ngnrsaa/qflex

HybridQ: A Hybrid Quantum Simulator for Large Scale Simulations

Hardware agnostic quantum simulator, designed to simulate large scale quantum circuits.

Can run tensor contraction simulations, direct evolution simulation and Clifford+T simulations using the same syntax

Features:

- Fully compatible with Python (3.8+)
- Low-level optimization achieved by using C++ and Just-In-Time (JIT) compilation with JAX and Numba,
- It can run seamlessly on CPU/GPU and TPU, either on single or multiple nodes (MPI) for large scale simulations, using the exact same syntax
- User-friendly interface with an advanced language to describe circuits and gates, including tools to manipulate/simplify circuits.

Recent Improvements:

- Commutations rules are used to simplify circuits (useful for QAOA)
- Expansion of density matrices as superpositions of Pauli strings accepts arbitrary non-Clifford gates,
- Open-source (soon!) project with continuous-integration, multiple tests and easy installation using either `pip` or `conda`

Open source code available at https://github.com/nasa/HybridQ

Agenda

- **Part I: Quantum-computing fundamentals**
  - High-level motivation, history, and status
  - Qubits, multi-qubit states, and quantum measurement

Morning break

- **Part II: Circuit-model quantum computing**
  - Review of notation
  - Quantum gates and quantum circuits

Lunch

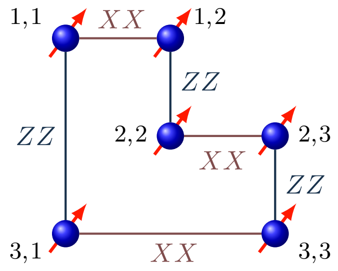
  - Basic quantum algorithms
- **Part III: Quantum annealing**

Afternoon break

- **Part IV: Further quantum algorithms**
  - Quantum Alternating Operator Ansatz (QAOA)
  - Advancements in quantum algorithms
  - Concluding remarks

Quantum Error Mitigation

- **Error suppression: Inhibits transitions out of the ground subspace**
- **Error correction: Counteracts transitions that have happened**
- **Quantum error correction initially thought impossible!**
  - No cloning principle: an unknown quantum state cannot be copied reliably without destroying the original
- **Quantum information theory was just too interesting**
  - Steane and Shor & Calderbank saw a way to finesse what had seemed insurmountable barriers to quantum error correction
- **Now quantum error correction is one of the most developed areas**
  - beautiful, almost magical, effects!
  - uses properties of quantum measurement and entanglement to its advantage
- **Stabilizer code formulation most common**
- **More recently, advantages of subsystem codes**
  - e.g. Jiang & Rieffel, Non-commuting two-local Hamiltonians for quantum error suppression, Q Info Processing (2017)



Fault Tolerance

- Error suppression and correction mechanisms cannot be done perfectly
- Fault tolerance: Ensures error suppression/correction do not introduce more problems than they solve
- Imprecise implementation of mechanisms may cause errors Even accurate implementation may magnify errors
  - can take correctable errors to uncorrectable ones
- Threshold theorems: There exists an error rate threshold rth below which indefinitely long quantum computations can be carried out robustly
- In the gate model, a number of different threshold theorems are known. Specific theorems involve precise statements of error model, precision of implementation, resource quantification, distance measure
- How to establish a threshold theorem for adiabatic quantum computing remains a major open question

Measurement-Based Quantum Computing (MBQC)

- **Outline of measurement-based quantum computation**
  - Start in a highly entangled state that serves as the quantum resource
    - Cluster states, graph states, …
  - Make series of single-qubit measurements that can depend on previous measurement results
  - Interpret the results of the measurements to obtain a final answer
- **Properties**
  - Computational power equivalent to standard quantum computation
  - Separation between classical and quantum aspects of the computation
  - Entanglement decreases; also called one-way quantum computing
- **Resource states for MBQC**
  - Some states too entangled to serve as a resource!
  - Classically hard to sample from output distributions of non-adaptive MBDL!
  - Open questions remain, e.g. Rieffel & Wiseman, Discord in relation to resource states for measurement-based quantum computation, PRA (2014)

Status of Quantum Algorithms

- **Anything a classical computer can do, a quantum computer can do**
- **Provable quantum advantage known for a few dozen quantum algorithms**
- **Data from Quantum Algorithms Zoo: speed up over classical**
  - Exponential: 2
  - Superpolynomial: 29
  - Polynomial: 28
  - Constant: 1
  - Varies: 4
  - Total: 64
  - https://quantumalgorithmzoo.org/
- **Rapidly expanding opportunity for empirical testing on emerging quantum hardware**

Conjecture: Quantum heuristics will significantly broaden of applications of quantum computing

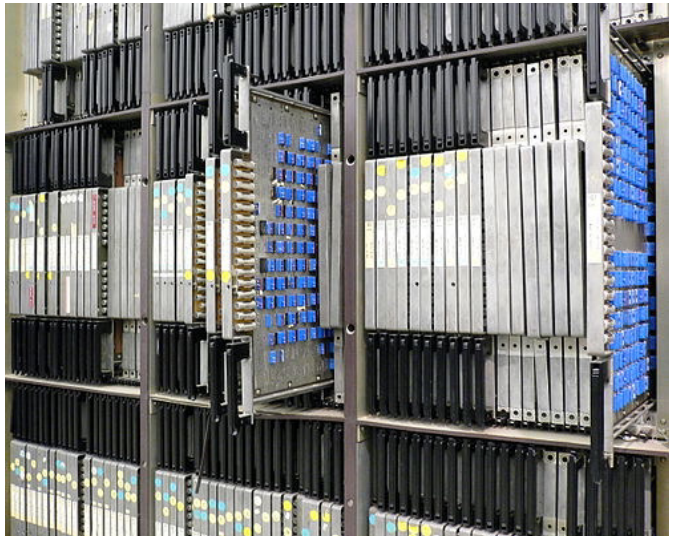
What is the chance that the only cases in which quantum computing provides a speed up is in cases when we can prove it does?

Impact of Quantum Information Processing Viewpoint on Classical Computer Science

- **Analogies**
  - Complex analysis enables computation of real integrals
  - Probabilistic algorithms inform analysis of deterministic algorithms
- **Quantum computational security reductions for purely classical encryption schemes**
  - Regev's lattice-based encryption scheme
  - One of the security reductions for Gentry's fully homomorphic encryption scheme
- **Improved classical simulations of quantum systems**

- **Insights into classical complexity theory**
  - Aaronson found a short, almost trivial, proof of a property of the complexity class **PP** by showing that it is the same as the quantum complexity class **PostBQP**
- **Drucker & de Wolf (2009) survey "Quantum proofs for classical theorems"**

A Historical Perspective

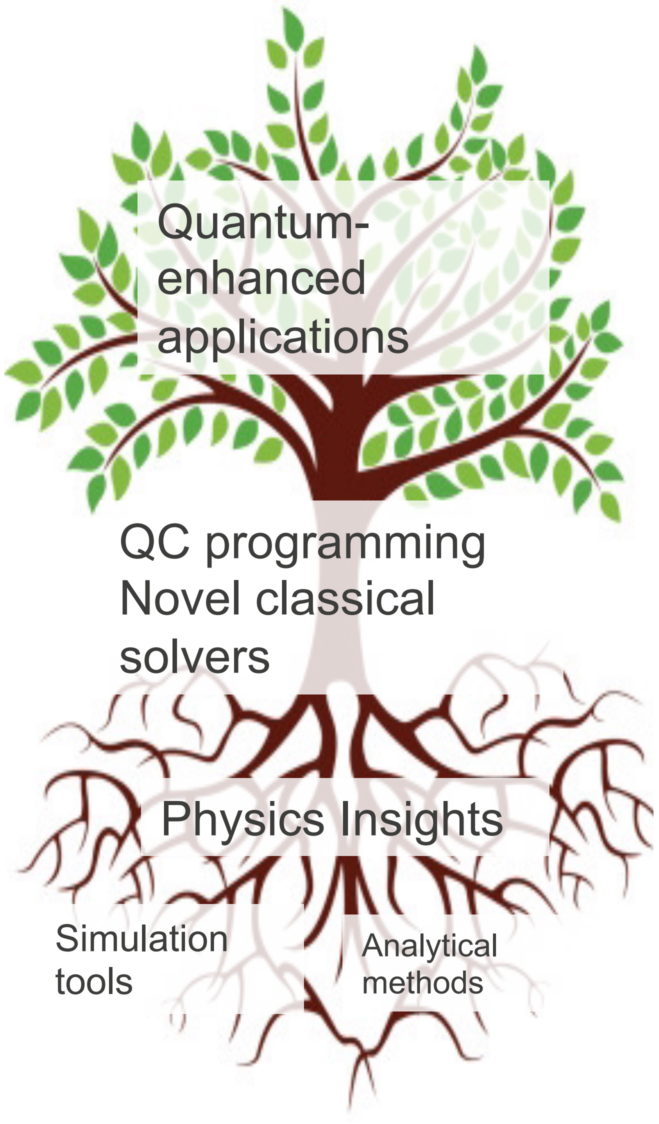


NASA Ames director Hans Mark brought Illiac IV to NASA Ames in 1972

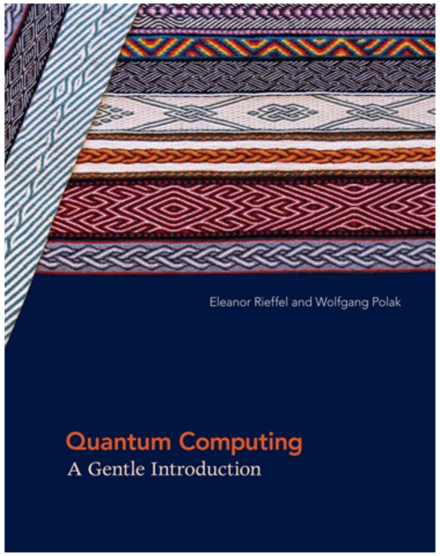
- **Illiac IV—first massively parallel computer**
  - 64 64-bit FPUs and a single CPU
  - 50 MFLOP peak, fastest computer at the time
- **Finding good problems and algorithms was challenging**
- **Questions at the time**
  - How broad will the applications be of massively parallel computing?
  - Will computers ever be able to compete with wind tunnels?

Take Away Points

- **Next year will be even more exciting!**
  - Emerging quantum hardware performing computations beyond the reach of even the largest supercomputers
- **Many open questions remain:**
  - When will scalable quantum computers be built, and how?
    - How quickly can special purpose quantum computing devices be built?
  - How broad will the impact of quantum computation be? What will the ultimate impact of quantum heuristics be?
  - How best to harness quantum effects for computational purposes?
- **Deep connection between physics and computer science**
  - How fast does nature let us compute?



Further Reading



Eleanor Rieffel and Wolfgang Polak
Quantum Computing: A Gentle Introduction
MIT Press, March 2011

And references therein

Overviews of NASA QuAIL team work

Eleanor G. Rieffel, Stuart Hadfield, Tad Hogg, Salvatore Mandrà, Jeffrey Marshall, Gianni Mossi, Bryan O'Gorman, Eugeniu Plamadeala, Norm M. Tubman, Davide Venturelli, Walter Vinci, Zhihui Wang, Max Wilson, Filip Wudarski, Rupak Biswas, ***From Ansätze to Z-gates: a NASA View of Quantum Computing***, arXiv:1905.02860

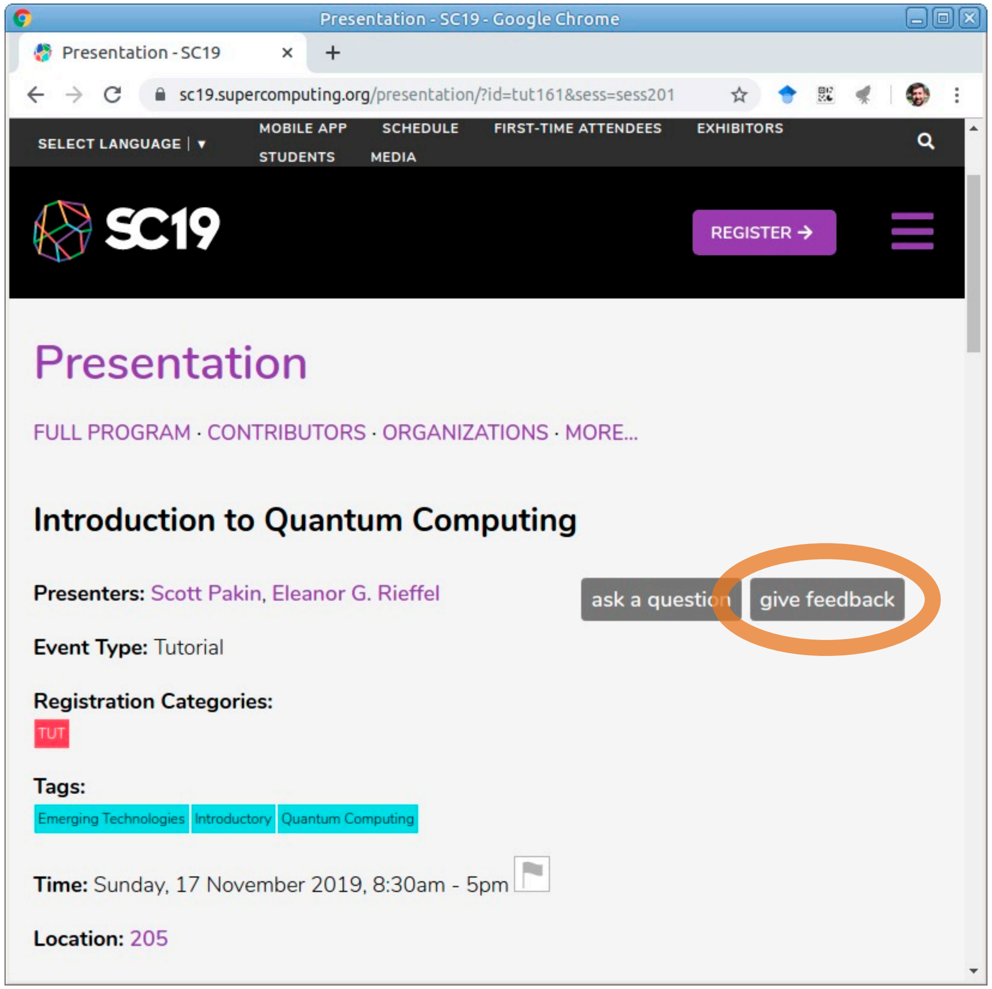
Rupak Biswas, Zhang Jiang, Kostya Kechezhi, Sergey Knysh, Salvatore Mandrà, Bryan O'Gorman, Alejandro Perdomo-Ortiz, Andre Petukhov, John Realpe-Gómez, Eleanor Rieffel, Davide Venturelli, Fedir Vasko, Zhihui Wang, ***A NASA Perspective on Quantum Computing: Opportunities and Challenges***, arXiv:1704.04836

Additional Resources

- **Free access to *real* quantum computers**
  - IBM Quantum Experience (circuit model): https://www.ibm.com/quantum-computing/
  - D-Wave Leap (annealing model): https://cloud.dwavesys.com/leap
- **Additional software for high-level programming of D-Wave systems**
  - Prolog: QA Prolog (https://github.com/lanl/QA-Prolog)
  - C: C to D-Wave (https://github.com/lanl/c2dwave)
  - Verilog: edif2qmasm (https://github.com/lanl/edif2qmasm)
  - Macro assembly language: QMASM (https://github.com/lanl/qmasm)
  - QUBO/Ising Google Sheet (https://tinyurl.com/y6wkkkm3)—use *File→Make a copy* to store an editable version in Google Drive or *File→Download as* to save locally
- **HPC quantum-circuit simulator**
  - qFlex (https://github.com/ngnrsaa/qflex)
- **Student internships available**
  - NASA QuAIL at NASA Ames Research Center (https://ti.arc.nasa.gov/tech/dash/groups/quail/)
  - LANL Quantum Computing Summer School Fellowship: https://quantumcomputing.lanl.gov/

Tutorial Survey

- **Please send us feedback so we know how to improve the tutorial for next time**
- **Two ways to find the review form:**
  1. Scan the QR code on the screen outside this room
  2. From https://sc19.supercomputing.org/, click on *Schedule* (at the top), then *Introduction to Quantum Computing*, and finally *give feedback*



Presentation - SC19 - Google Chrome
Presentation - SC19
sc19.supercomputing.org/presentation/?id=tut161&sess=sess201
SELECT LANGUAGE | ▾
MOBILE APP
SCHEDULE
FIRST-TIME ATTENDEES
EXHIBITORS
STUDENTS
MEDIA
SC19
REGISTER →
FULL PROGRAM · CONTRIBUTORS · ORGANIZATIONS · MORE...
ask a question
give feedback

Presentation

Introduction to Quantum Computing

Presenters: Scott Pakin, Eleanor G. Rieffel

Event Type: Tutorial

Registration Categories:
TUT

Tags:
Emerging Technologies Introductory Quantum Computing

Time: Sunday, 17 November 2019, 8:30am - 5pm

Location: 205