Quantum Computing & Applications for Engineering

9/17/2024

Lecture 2: Making, Using, and Measuring Qubits

Today's Goals

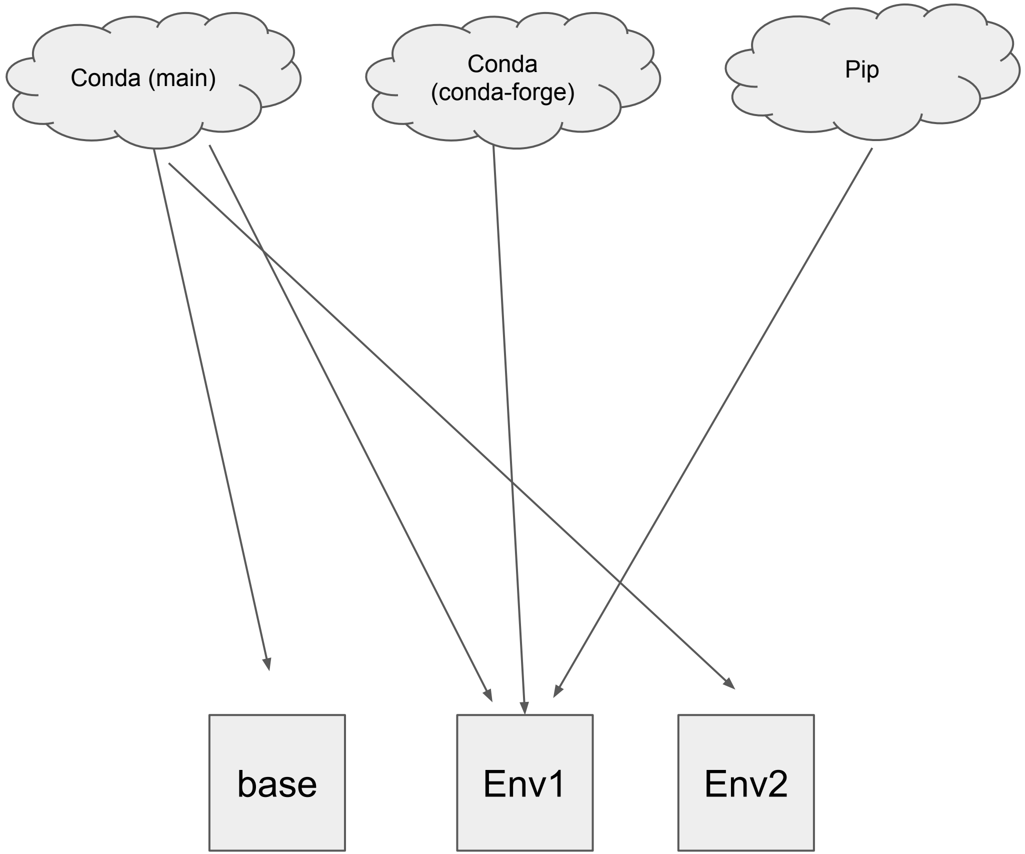
- Learn:
  - What makes a qubit a qubit?
  - How do we use them?
  - How can we measure their properties?
- Do:
  - Go over common issues
  - Some simple Qiskit patterns
  - Basic qubit experiments

Announcements

- Office Hours
  - Thanks to everyone who came!
  - Helps me understand where the gaps are, what to focus on.
- Common Issues
  - Conda and Jupyter Usage
  - Command Prompt/Terminal Usage
  - Programming practices and style
  - Outputting lists, printing to PDF
  - What the heck did I just run?

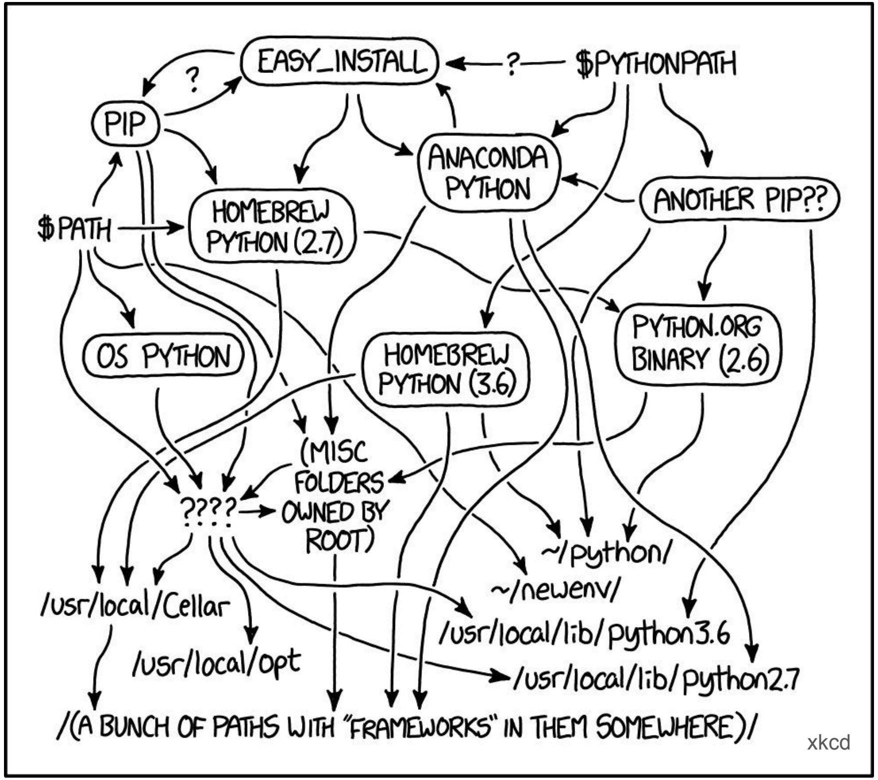
Anaconda Python

- Anaconda is a Python *distribution.*
  - Python itself is open source
  - Anaconda bundles it up with some common stuff to make life easier
- Main purpose is to manage environments.
  - Installing packages (`conda`/`pip`)
  - Maintaining consistency
- Possible to use different versions in different environments



Anaconda Python

- Each environment is a “box” with its own version of python, pip, and any other packages you want to install.
  - Environments are completely independent.
  - Package versions are cached/linked to avoid filling your hard drive with duplicates
- Bad idea to install in the base environment
  - Some packages are incompatible with each other and with certain versions of Python
  - Leave base alone, always try to make a new environment for different lines of work.
- For this class, we should only need the environment we created last time.



MY PYTHON ENVIRONMENT HAS BECOME SO DEGRADED THAT MY LAPTOP HAS BEEN DECLARED A SUPERFUND SITE.

Command Line Usage

- On Windows, Anaconda gives you a “navigator” interface.
  - On Mac/Linux too, but it’s less visible.
  - Let’s you do everything graphically.
- Use what you prefer.
  - Powershell (Windows)
  - Bash/zsh/fish (Mac/Linux/WSL)
  - None! (although I recommend learning a bit)
- Using the command line can be simpler.
  - Navigator can hang or crash.
  - Activating environments is more intuitive.
  - More control and history tracking.
- You can send outputs to a file:
  - Use the “>” operator.
  - Fastest way to do problem 1 on the HW.

Jupyter

- Jupyter is a program (written in Python) that runs within a Python environment.
    - Starting Jupyter starts a server that runs locally in the background.
    - You launch “kernels” in Python or other languages.
- Need to start Jupyter before you can open notebooks.
- Each notebook is a completely isolated session.
- Can navigate deeper into folders, but you can’t go higher than the one you started in.

Using Jupyter

- Each notebook consists of runnable “cells.”
  - Code, Markdown and Raw cells
  - Can move, split, copy, paste cells
- Cells can execute out of order.
  - Look at the number along the side to see what ran last.
  - Variables you delete will still be in memory.
- Exporting notebooks
  - The raw notebook format is “readable” but unwieldy JSON.
  - Can save as HTML or PDF.
  - PDF export requires LaTeX
  - Can try printing to PDF.

Python Style

- Imports
  - Best practice is to put them at the top of the code/notebook.
  - Use aliases or import individual objects to avoid retyping module names over and over.
  - Don't pollute the namespace!
- Reusing code
  - If you are copying/pasting more than 2-3x, consider using functions, classes and loops
- Formatting
  - Python standard (PEP8) recommends 80-character line lengths
- Comments
  - Comment your code or use Markdown cells
  - Helps with understanding and grading.
- I don't grade on style, but it will help with partial credit if I can understand what you're doing!

```
import qiskit
import qiskit as qs

from qiskit import QuantumCircuit
```

One blank line

~~from qiskit import~~ *

Danger!

```
QiskitRuntimeService(var1, var2,
```

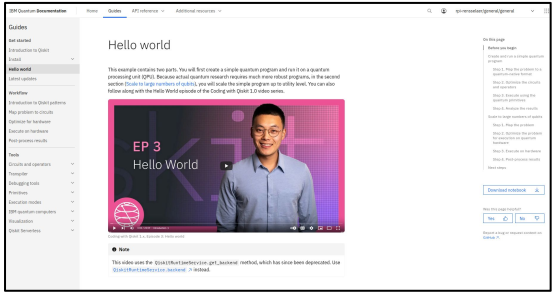
Needs to side-scroll on small monitors.

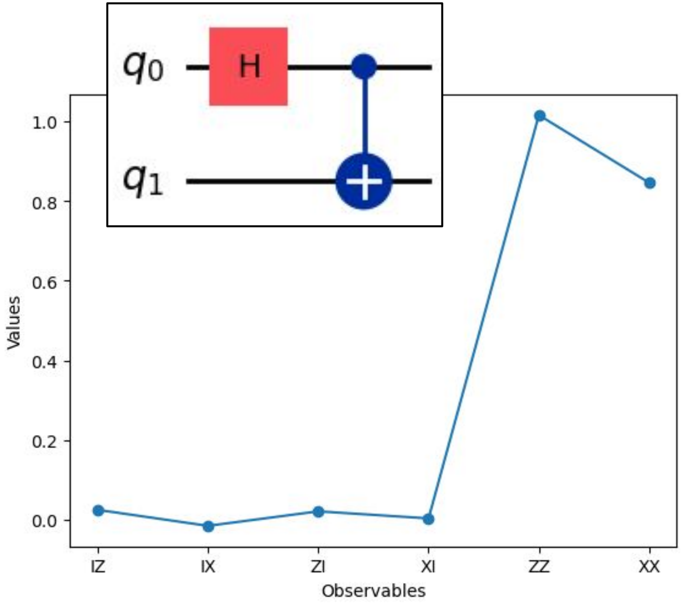
```
QiskitRuntimeService(var1,
                     var2,
                     var3,
                     var4)
```

Line breaks make it easier to read.

What the heck did I run?

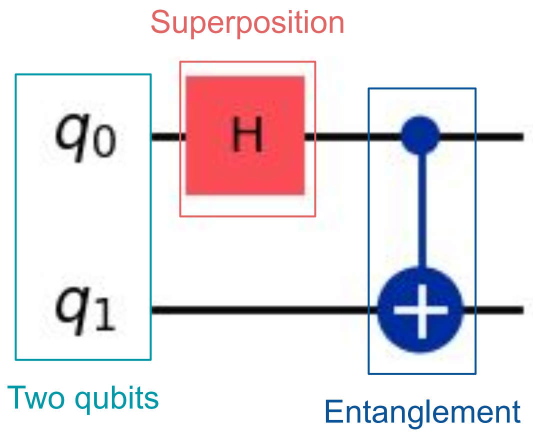
- The Qiskit “Hello World” example:
  - Simple 2-qubit test.
  - Meant to be a sanity check on our computing environment.
- We blindly copied/pasted a bunch of magic and got a plot.
  - Something about a circuit
  - Some stuff with estimators/observables
- What did it actually do?
  - Something very important!
  - We’ll spend this class and next working up to it.

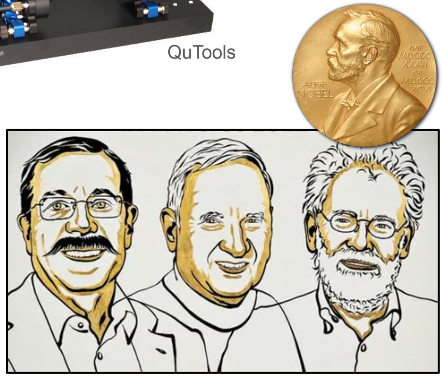




The Bell State

- Theory developed by John Stewart Bell in 1964.
- The simplest demonstration of quantum phenomena:
    - Superposition
    - Entanglement
    - Ruling out “hidden variables”
- Won the 2022 Nobel Prize
    - Alain Aspect, John Clauser, Anton Zeilinger.
    - Experimental validation and theoretical development.
- This is the building block of all quantum technologies.

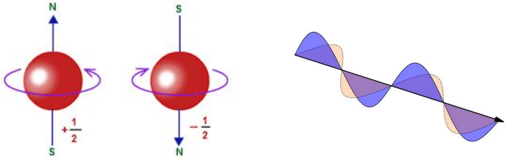




But first…

- …you need some qubits.
  - …and ways to operate on them.
  - …and ways to entangle them.
- …preferably stable ones.
- …preferably with a Python API.
  - We want to make a computer after all.
  - …and Python is the second best programming language for everything.

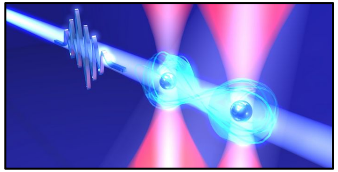
The DiVincenzo Criteria



Two-Level Systems

We need a quantum system that:

1. Have two (or more) well-characterized quantum states.
2. Can be reliably initialized into a known state.
3. Can be controlled through a set of universal operations.
4. Can be measured through a controllable readout process.
5. Have long coherence times.



Initialization, Control, Readout

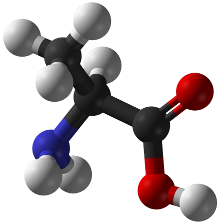
For networking, we also need:

6. The ability to convert stationary and flying qubits.
7. The ability to transmit flying qubits between locations.

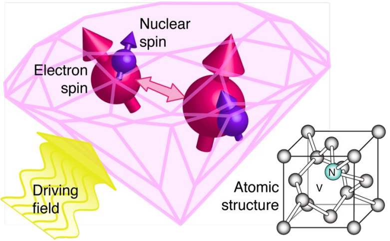
Isolation from environmental noise

D. DiVincenzo. (2000) “The Physical Implementation of Quantum Computation”, Fortschritte der Physik 48 (9–11): 771–783. (arXiv)

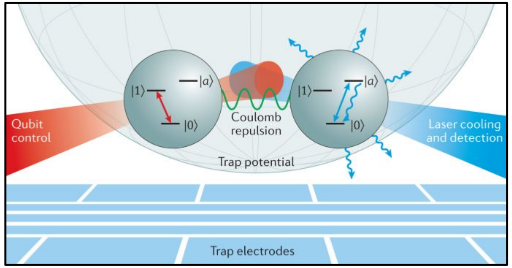
Criterion 1: Two-Level Quantum Systems



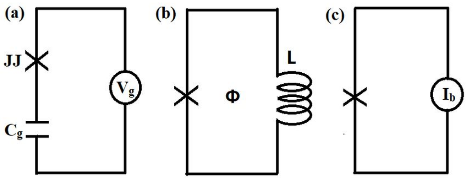
Nuclear Magnetic Resonance



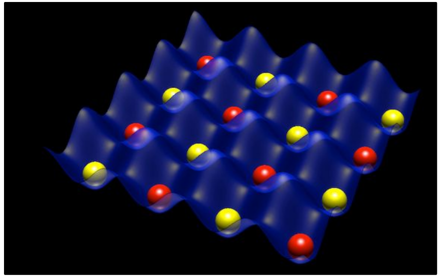
Crystal Spin Defects



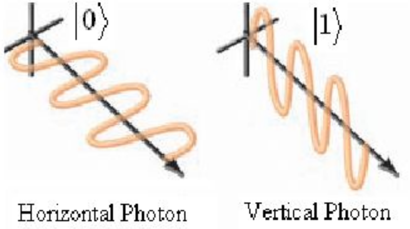
Trapped Ions



Superconducting Circuits



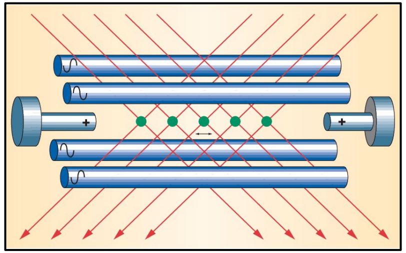
Neutral Atoms

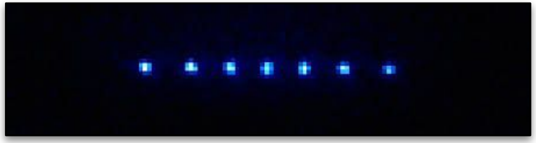


Single Photons

Trapped Ions

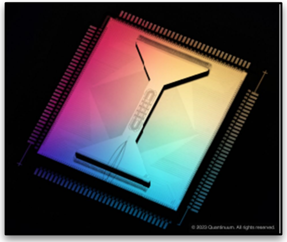
- Individual ions of Yb^+ or Ba^+
  - Trapped by electrostatic fields
  - Addressed and readout by lasers
- Scales to 100s of qubits per chip
  - Long coherence time (seconds to minutes)
  - All-to-all connections
  - Networking needed
  - Error correction has been demonstrated
- One of the first modalities tested
  - Spinoff of atomic clock technologies
  - Developed at NIST and Sandia

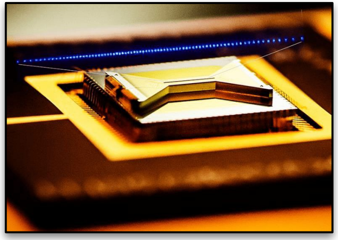




QUANTINUUM

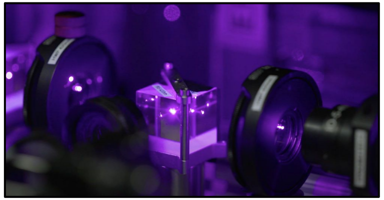
IONQ

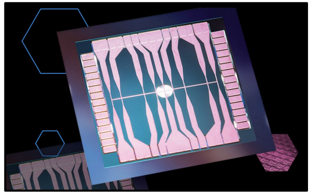


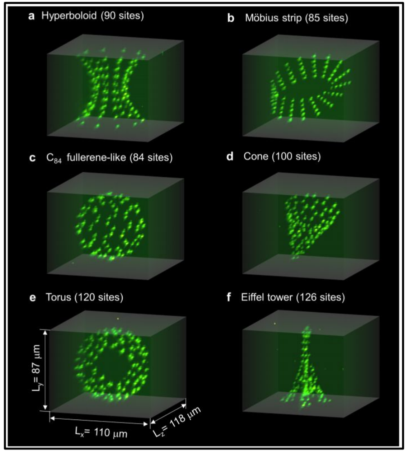


Neutral Atoms

- Atoms are cooled to nK temperatures with lasers
- Magentic fields and lasers are used to trap atoms in a grid
  - Nearest-neighbor connectivity
  - Atoms can be physically moved
- Very long coherence times
  - Several seconds
  - Error correction has been demonstrated
- Can also be used as quantum memory.







PASQAL

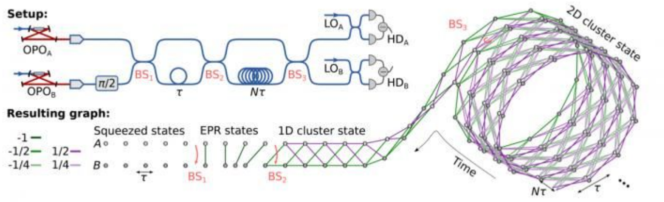
QuEra Computing Inc.

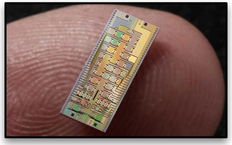
Infleqtion

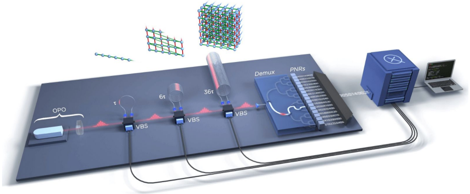
atom computing

Photons

- Photons can be used as qubits
  - Polarization
  - Resonant states in loops and cavities
- Integrated photonics can yield a high density of qubits
  - Millions of physical qubits
- Noise-resistance is a double-edged sword.
  - Reliable computations
  - Challenging to address and manipulate qubits.
- Necessary for networking.





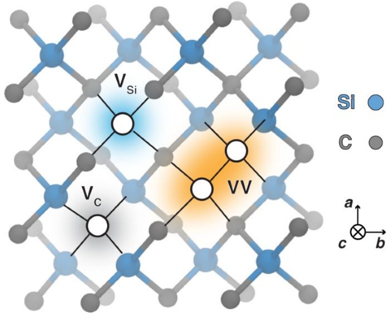


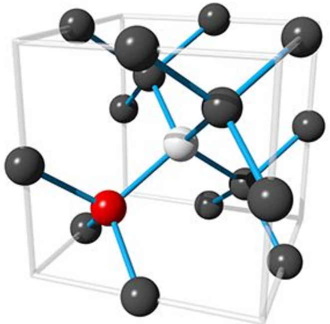
XANADU

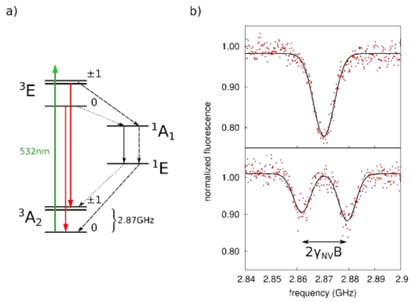
PsiQuantum

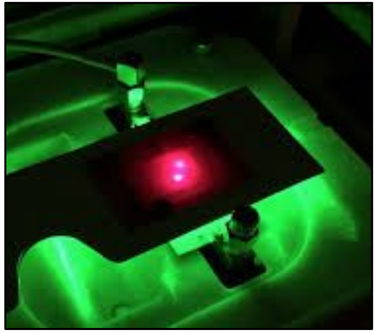
Crystal Defects

- Defects in crystals can create spin and energy structures in crystals
  - Wide-bandgap semiconductors
  - Diamond, SiC, hBN, …
- High temperature operation
  - Up to 800K for some devices!
  - Low temperature improves resolution.
- Mainly of interest for quantum sensing & networking.
  - Can make very tiny sensors
  - Can transduce light to RF and back





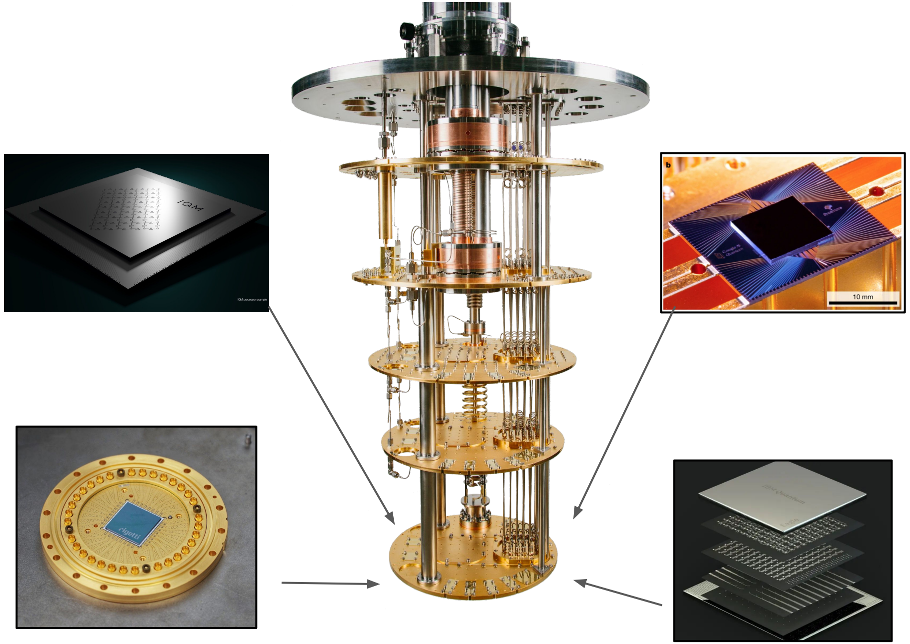




QUANTUM BRILLIANCE

Superconducting Circuits

- Superconducting Josephson junctions coupled to a capacitor or inductor.
  - ~10mK temperatures
  - Resonant oscillation modes at microwave frequencies
- Well-rounded performance
  - Fast gate times (~100s of ns)
  - Moderate coherence times (~100s of μs)
  - 100s of qubits, various connectivity



IQM

Google AI Quantum

rigetti

IBM Quantum

What kind should I pick?

- Roughly a dozen competing methods
  - Different benefits and tradeoffs
  - Everyone thinks theirs is the “one true way.”
- It’s not clear yet which hardware will “win.”
  - It took 20 years for silicon-based CMOS* devices to dominate classical hardware
  - It took another 20 years for x86 to become the dominant CPU architecture
- There may never be a winner.
  - Superconductors are fast, but require cryogenics and have finite connectivity.
  - Ions are stable, but slow and require networking to scale.
  - Atoms can scale and are stable, but are very slow.
  - Photons are so stable they’re challenging to control.
  - Crystal defects are moderately fast, but hard to manufacture.
  - …

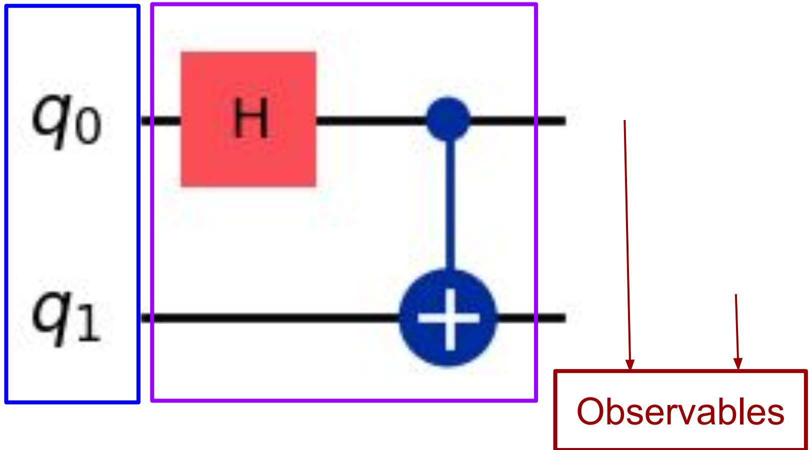
*Complementary Metal-Oxide-Semiconductor - basically all microelectronics today

All different, yet all (mostly) the same

- Regardless of the hardware implementation, the theory is the same.
- Schrodinger’s equation
  - Apply a Hamiltonian to a quantum state
  - The energy values contain information on the state.
- The exact values for these will vary.

$$\boxed{E}\boxed{\Psi} = \boxed{\mathbf{H}}\boxed{\Psi}$$

Energy (scalar)
Hamiltonian (matrix)
Qubits’ State (vector)

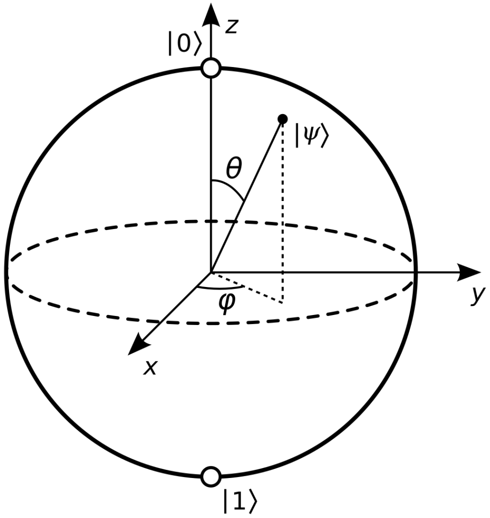


Quantum States

- The state of any qubit (or group of qubits) is represented by a vector.
  - The elements of the vector are complex numbers.
  - The length of the vector is always 1.
- Single qubit states are plotted on the Bloch Sphere (unit sphere).
- The Z-axis is taken to be the computational basis.
  - The |0> and |1> values are the basis states
  - Qubits are measured with respect to the computational basis.

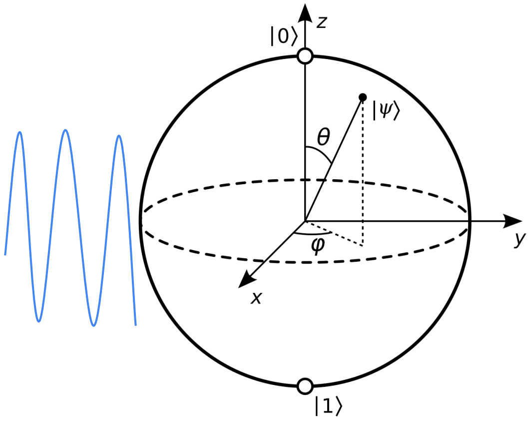
$$|0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \qquad |1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle \qquad |\alpha^2 + \beta^2| = 1$$

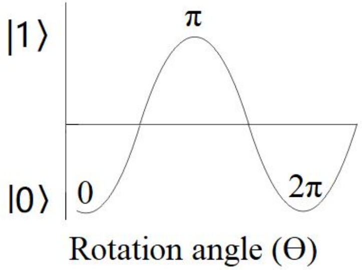


Criteria 2-4: Initialization, Control and Readout

- We use photons to interact with qubits
  - Radio/Microwave (RF) - Superconducting, spins
  - IR/Visible/UV (Laser) - Ions, neutral atoms, crystal defects
- Carefully timed RF/Laser pulses cause qubits to change state.
  - Align a signal generator to the qubit's resonant frequency
  - Send the signal to a modulated RF or laser source
- Duration and phase of the pulses determine the resulting qubit state.



Pulses cause the qubit statevector to rotate



The pulse duration determines the new state

Quantum Gates

- Pulses act as single quantum computational instructions, or gates.
- A square 2^nx2^n matrix
  - Sometime called an operator
  - Defines how probability amplitudes are exchanged.
- Unitary matrix
  - Measurement probability among all basis states is conserved.
  - Measurement is always a real number.
  - Self-adjoint.

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

$$\mathbf{Y} = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}$$

$$\mathbf{Z} = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$$

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

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

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

$$U^\dagger U = I$$
$$UU^\dagger = I$$

$$A = \begin{pmatrix} 1+i & 2-i \\ 3i & 4 \end{pmatrix} \qquad A^\dagger = \begin{pmatrix} 1-i & -3i \\ 2+i & 4 \end{pmatrix}$$

Note: The above oversimplifies a semester-long math/physics course into a single slide.

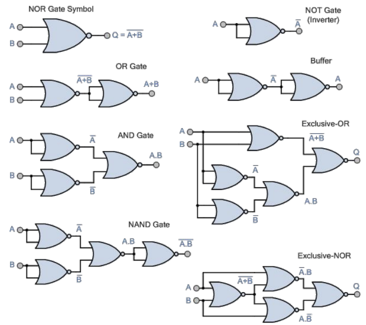
Quantum Gates

- It can be shown that there is a <u>universal</u> gate set.
    - Any other gates can be built from a handful of single and two-qubit gates.
    - Classical analogy - NAND gate
- Two-qubit gates that create entanglement cannot be decomposed into separate single-qubit operations.

$$\mathbf{U}(\theta, \phi, \lambda) = \begin{pmatrix} \cos\left(\frac{\theta}{2}\right) & -e^{-i\lambda}\sin\left(\frac{\theta}{2}\right) \\ e^{i\phi}\sin\left(\frac{\theta}{2}\right) & e^{i(\phi+\lambda)}\cos\left(\frac{\theta}{2}\right) \end{pmatrix}$$

<u>A general 1-qubit gate</u>

$$\mathbf{T} = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\frac{pi}{4}} \end{pmatrix}$$



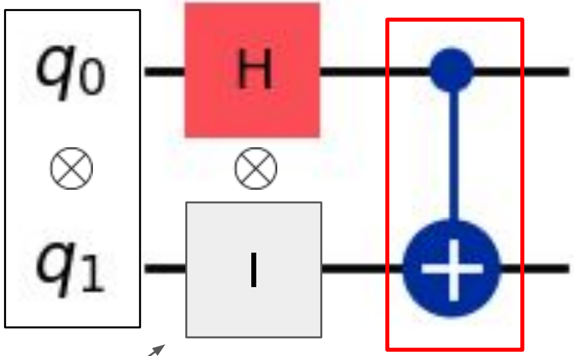
<u>Classical Universal Gates</u>

Composing Things

$$|0\rangle \otimes |0\rangle = \begin{pmatrix}1\\0\end{pmatrix} \otimes \begin{pmatrix}1\\0\end{pmatrix} = \begin{pmatrix} 1\begin{pmatrix}1\\0\end{pmatrix} \\ 0\begin{pmatrix}1\\0\end{pmatrix}\end{pmatrix} = \begin{pmatrix}1\\0\\0\\0\end{pmatrix} = |00\rangle$$

$$|0\rangle^{\otimes 2} = |00\rangle$$

- We (mathematically) assemble individual gates and qubits using the tensor product.
- Mechanically, you "tile" the elements onto each other.
- Entangled things cannot be decomposed.
    - Tensor product states need 2N storage elements
    - Entangled states require 2^N storage elements.
    - This is why classical representations of quantum states scale exponentially.

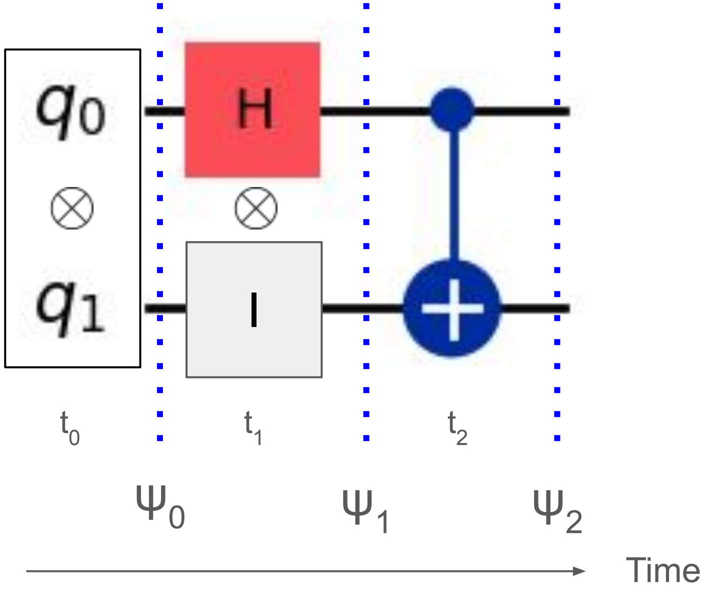


(Identity matrix is implied)

This is a 4x4 that cannot be decomposed into two 2x2's!

Composing Things

- Recall matrix-vector multiplication
    - Tip the vector on its side.
    - Drop it through the matrix, multiplying the aligned elements.
    - Sum the results.
- Quantum circuits read left to right
    - Each qubit has a “wire” or world-line.
    - Time flows to the right.
    - Gates are applied at each step
- Equivalent math goes right to left



$$|\psi_0\rangle = |0\rangle \otimes |0\rangle$$

$$|\psi_1\rangle = (\mathbf{H} \otimes \mathbf{I})|\psi_0\rangle$$

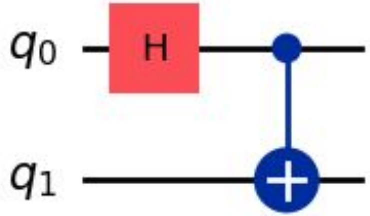
$$|\psi_2\rangle = \mathbf{CX}|\psi_1\rangle$$

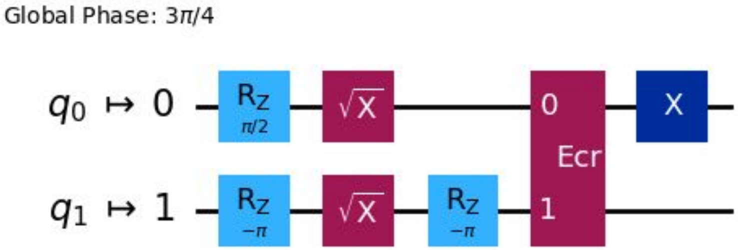
Generating Entanglement

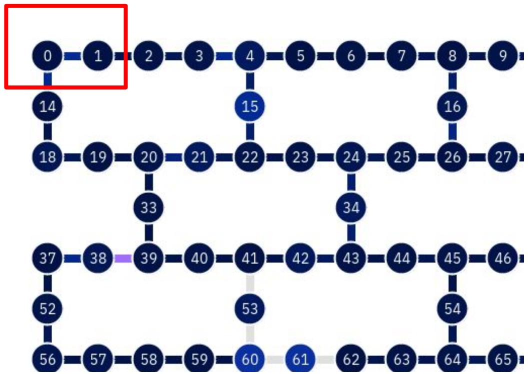
- Entanglement is how we compose qubits into collective systems.
  - The multi qubit system acts as a collective, regardless of physical separation.
  - Individual qubits participating in entanglement are correlated with their partners.
- Physically, we generate entanglement with a coupled pulse that acts on both qubits simultaneously.
- Mathematically, we represent entangling gates with “controlled gates”
- Entanglement is what enables us to represent more information

Transpilation

- The signal generators for quantum computers typically only support a few “native” operations.
- Entangling gates can only be applied on qubits that are physically near each other.
- The transpiler tries to find an optimum, mathematically-equivalent configuration.







Measurement

- Measurement is performed with respect to an “observable.”
  - Observable is a unitary matrix
  - Measurement projects the qubit statevector onto the observable axis according to its probability amplitudes.
  - Matrix eigenvectors are the possible outcomes
- A qubit is a “3D-bit” so we may need to look at it a few different ways.
- Pauli Z, X operators are the most common observables.
  - Z-axis: Standard 0/1 basis
  - X-axis: “Hadamard basis”
- Many repeated measurements give us the <u>expectation value</u> of the observable.
- Measurement ends the quantum part of the computation.
  - The measured qubits can no longer participate.
  - They can be re-initialized and used for other things

Doing all of this on a real machine

- IBM launched their cloud devices in 2016
  - 5 qubits was a lot!
  - 14 was “premium”
  - Other vendors followed this model
- Composer - Draw circuit diagrams by hand
  - Good for playing around and understanding
  - Cumbersome for big circuits
- Qiskit - Write circuits in Python
  - Define qubits and gates using quantum circuits
  - Define measurement and experiment types using Primitives
  - Connect to hardware or backend simulators

Break

Activity: Making things in Qiskit

Services & Backends

- The *service* runs locally and manages interactions with *backends*
- A backend is a quantum device or a simulator
- Demo: Setup a service and a backend

Quantum Circuits

- Quantum circuits are the programs we run on the machine.
- Start by creating an empty circuit
  - Append Gates
  - Draw it (optional)
- Use the Statevector object for debugging small circuits.
- Demo

Transpilation

- We need to transpile the circuits we made so they can run on the native gates.
- Qiskit provides a “pass manager” that performs this step automatically.
    - You can write your own pass managers too

Running Stuff

- Qiskit IBM Runtime provides “Primitives” for running quantum circuits on the hardware.
  - Sampler - Run the circuit and measure the qubits. Repeat for a predefined number of “shots”
  - Estimator - Compute the expectation value of an observable in a quantum circuit. Runs on top of a sampler and implements its own post-processing to save you some steps.
- We will focus on samplers this week

Post-Processing Results

- After running a job, we need to do something with the results
    - Statistics
    - Plotting
    - …
- In this example, we plot a histogram of the counts of each state we measured.