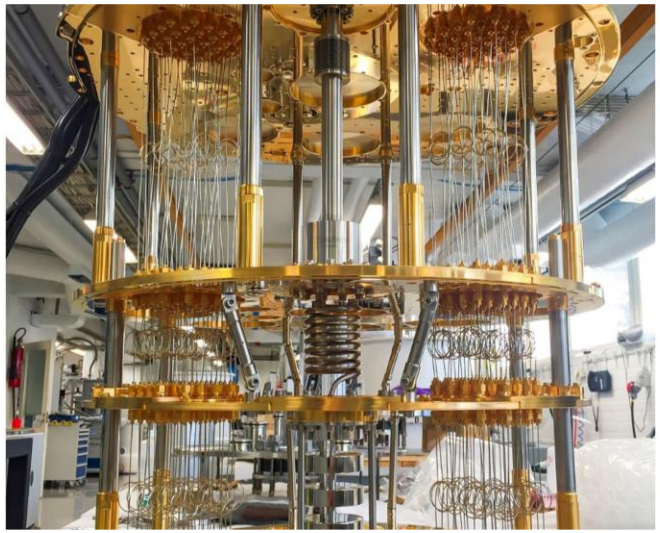
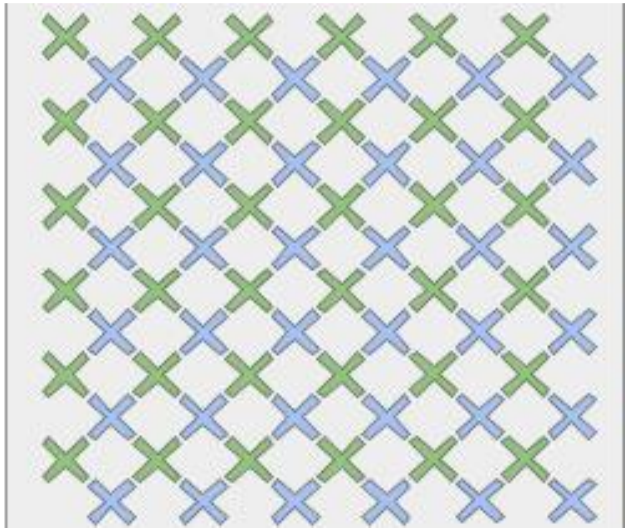
A Practical Guide to Quantum Computing

Rian Finnegan @xtellurian

In the News

- Microsoft announces creation of Majorana* particle at Delft University of Technology
  - *an elementary particle that is its own anti-particle
- Google creates 72 qubit chip – “Bristlecone”
- Australian startup Q-CTRL chosen by IBM to join global quantum computing network

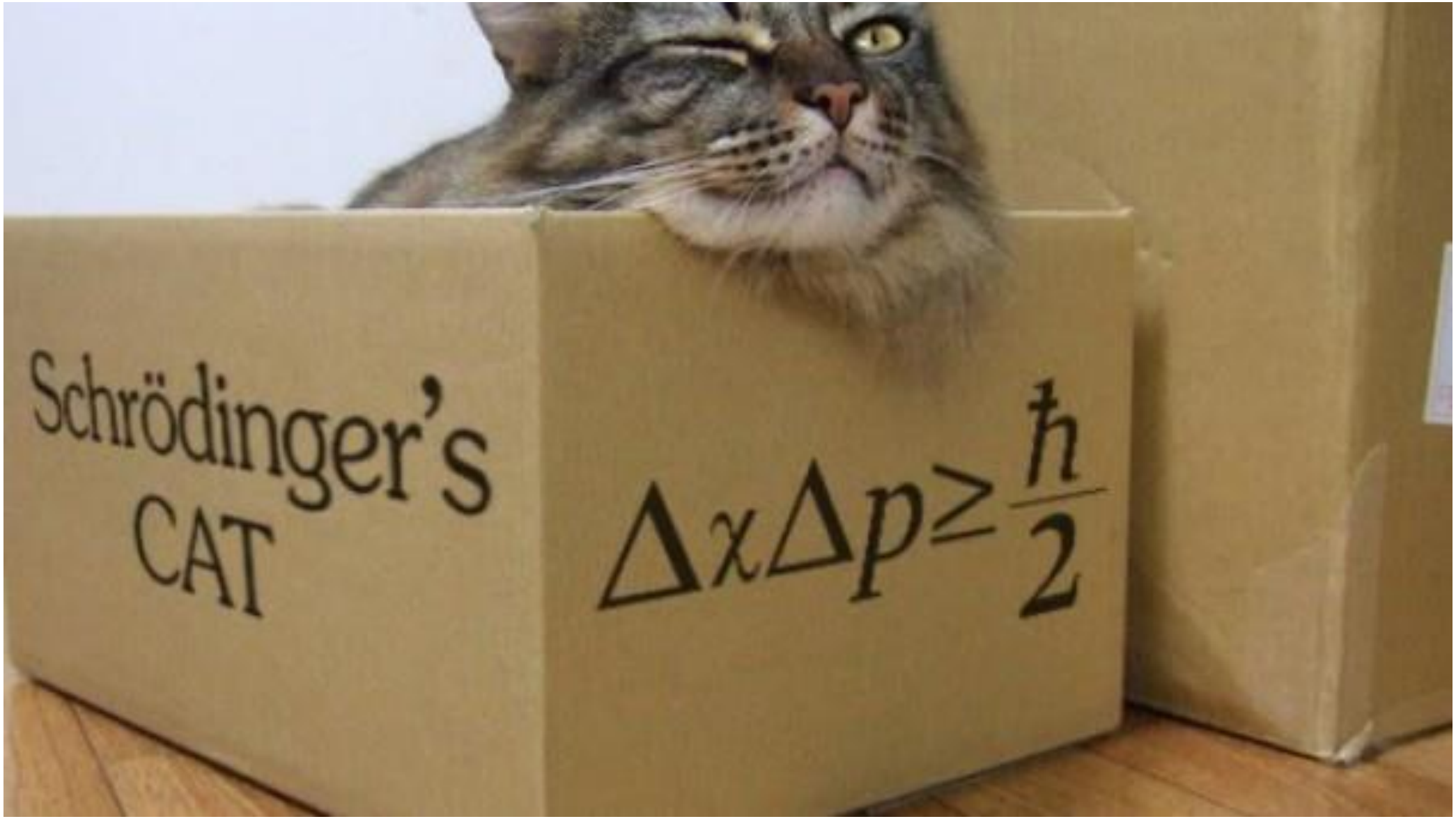




Climate change

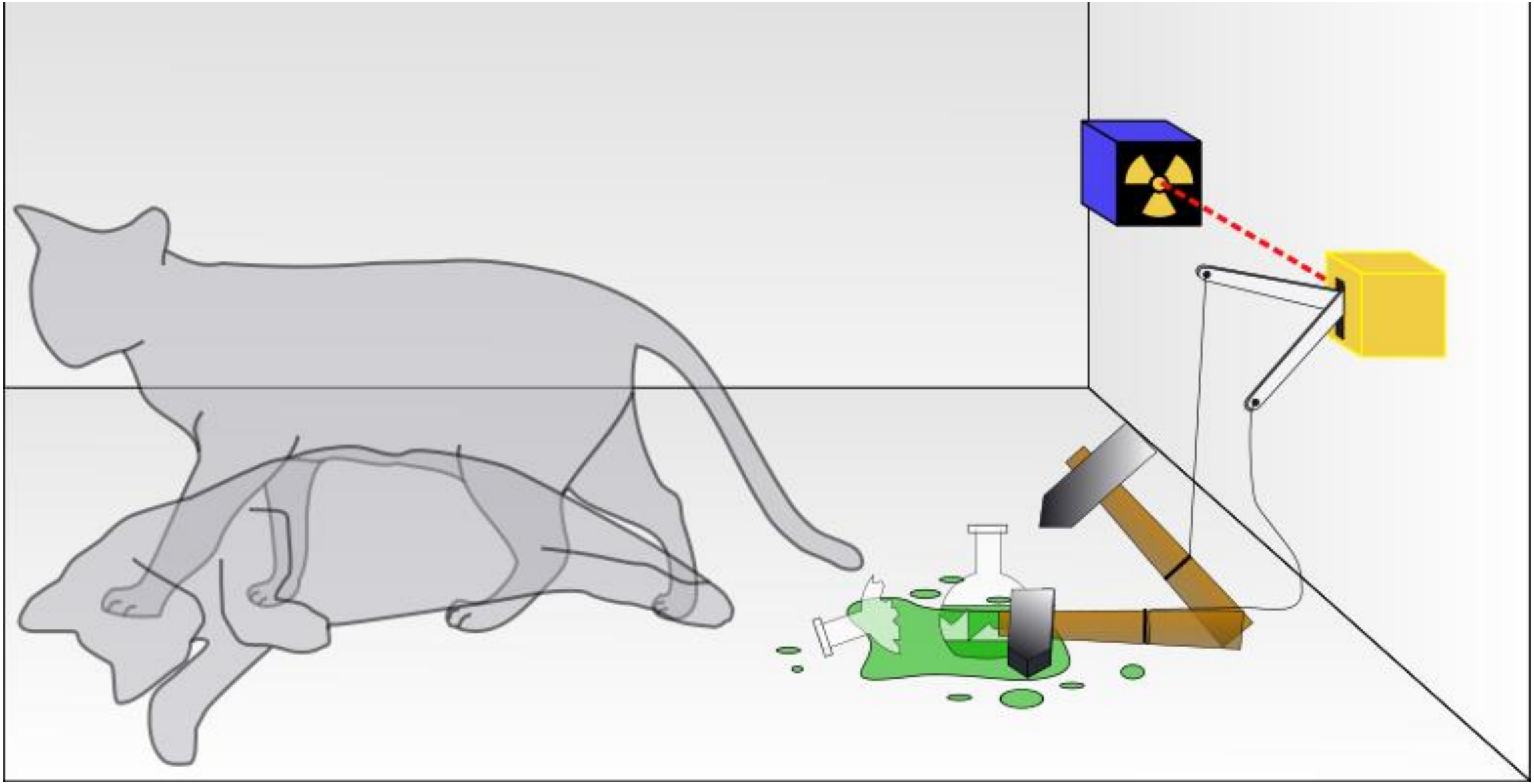
Food production

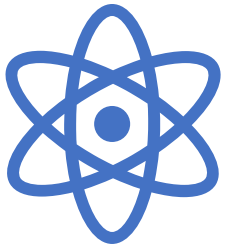
Antibiotic resistance


Schrödinger's
CAT
$\Delta x \Delta p \geq \frac{\hbar}{2}$

Schrödinger's Cat: Dead or Alive?

1. Schrödinger's cat: a cat, a flask of poison, and a radioactive source are placed in a sealed box.
2. If an internal monitor (e.g. Geiger counter) detects radioactivity (i.e. a single atom decaying), the flask is shattered, releasing the poison, which kills the cat.
3. The Copenhagen interpretation of quantum mechanics implies that after a while, the cat is *simultaneously* alive *and* dead. **[Superposition]**
4. Yet, when one looks in the box, one sees the cat *either* alive *or* dead not both alive *and* dead. **[Measurement]**
5. This poses the question of when exactly quantum superposition ends and reality collapses into one possibility or the other.



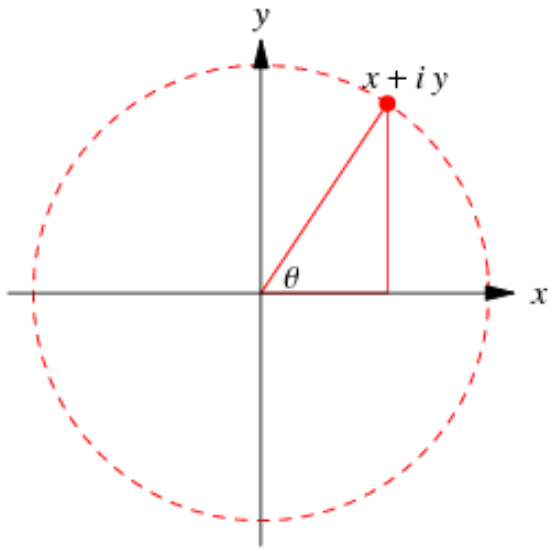


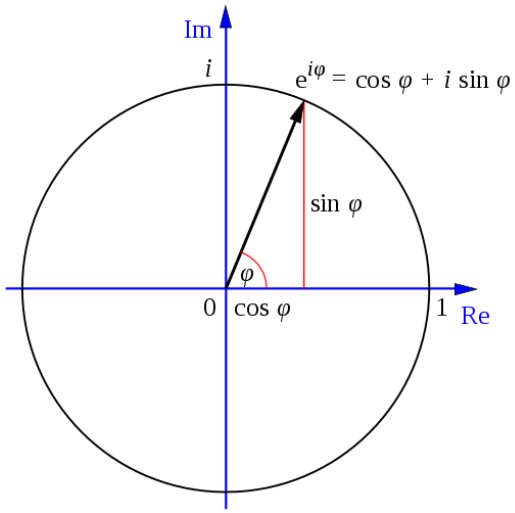
Quantum Theory

Theory: Complex Numbers

Define: $i^2 = -1$

Cartesian Form	Euler's Form
z = x + yi	$e^{ix} = \cos x + i \sin x$





Absolute Value/ Radius := $r = |z| = \sqrt{x^2 + y^2}.$

Theory: Dirac ‘Bra–ket’ Notation

$$\langle \Phi | \Psi \rangle$$

Φ = bra
Ψ = ket

$$|\Psi\rangle = \begin{pmatrix} \Psi_1 \\ \Psi_2 \end{pmatrix}, \qquad \Psi_1, \Psi_2 \in \mathbb{C},$$

where the notation $\Psi 1, \Psi 2 \in \mathbb{C}$

means that the components Ψ1, Ψ2 are *complex numbers*.

The notation $|\Psi\rangle$ is called Dirac (or “bra-ket”) notation and Ψ is simply a label used to refer to the *vector*.

We can write anything as the label, including English/Greek letters, numbers, or even words, for example:

$|A\rangle$, $|\beta\rangle$, $|3\rangle$, $|\text{cat}\rangle$ . . .

Triple Phasor Paradox

Theory: Quantum States

A quantum *system* is the mathematical representation of a physical system (such as a particle) as a **Hilbert space**.

An *operator* on a quantum system is a **matrix** in the appropriate Hilbert space. It represents an action performed on the system, such as a **measurement**, a **transformation**, or time **evolution**.

A *state* of a quantum system is a **vector with norm 1** in the appropriate Hilbert space. It represents the configuration of the system, and encodes the **possible outcomes of measurements** performed on that system.

|cat⟩ =1√2|dead⟩ - 1√2|alive⟩

$$|\psi\rangle = a|u\rangle + b|d\rangle.$$

$$|\Psi\rangle = \sum_{i=1}^{n} |B_i\rangle\langle B_i|\Psi\rangle.$$

Theory:
Classical
Bits

Bit (**Bi**nary Dig**it**):

Unit of information used in computing and digital communication.

2 possible values:

0 1

Compose bits to form larger numbers:

1001 => 9

Physically represented using transistors and classical logic gates

Theory: Quantum Bits (Qubits)

Qubit: The simplest possible quantum system

The standard basis of C2, which in this case is called the *computational basis*, consists of the following two basis states:

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

Note that the numbers inside the *kets* $|\cdot\rangle$ are just labels, as always. Any state of this system is thus represented as a superposition of these two basis states:

$$|\Psi\rangle = a\,|0\rangle + b\,|1\rangle$$

Quantum Development Kit *Preview*

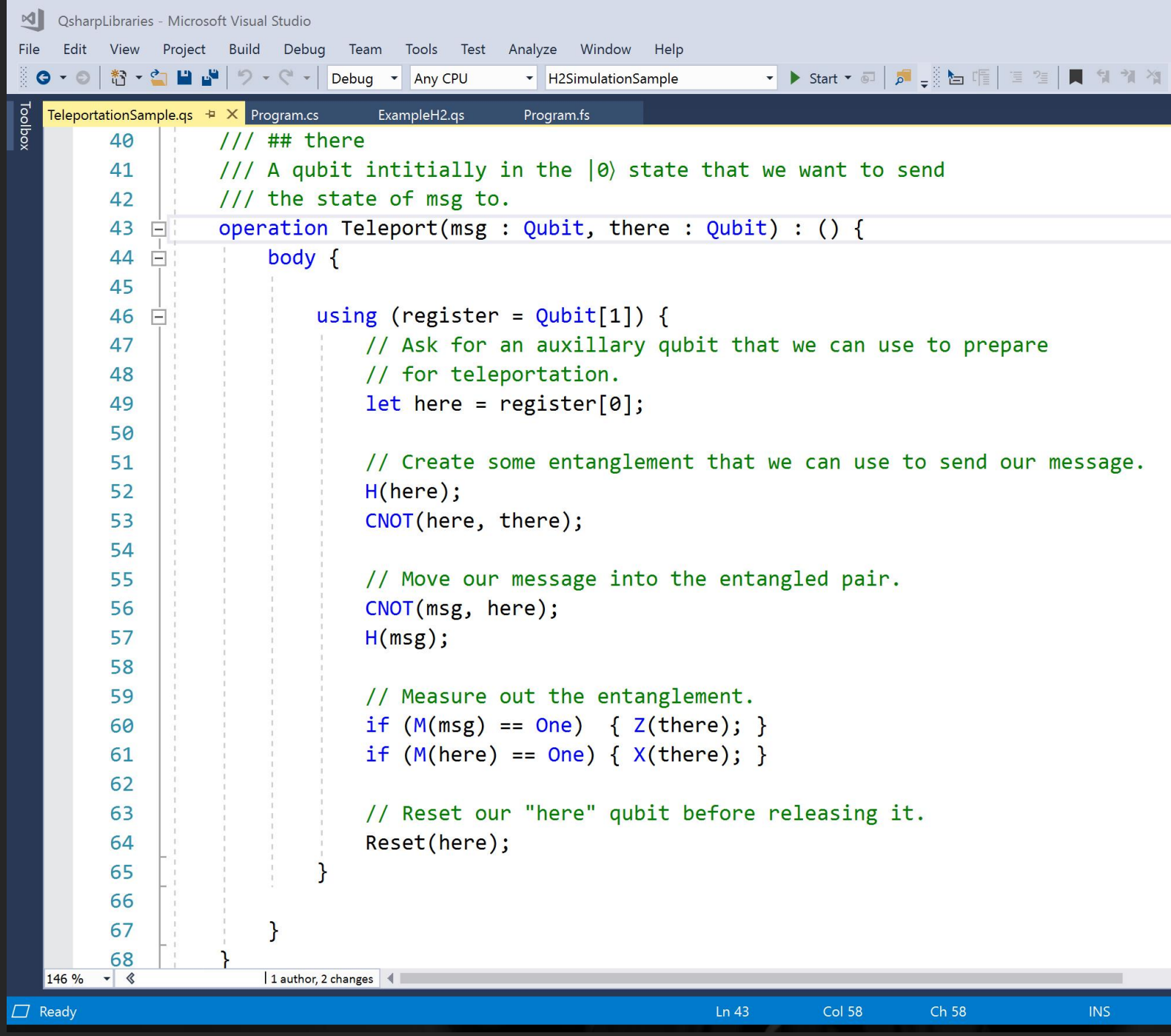
Quantum-focused programming language

Local & Azure simulators

Sample code & libraries

Quantum-focused programming language (Q#)

- Built ground-up for Quantum
- Support for Windows, macOS, and Linux
- Fully integrated into Visual Studio and VS Code
- Interoperability with Python (Windows only)
- Native type system



Local Simulator

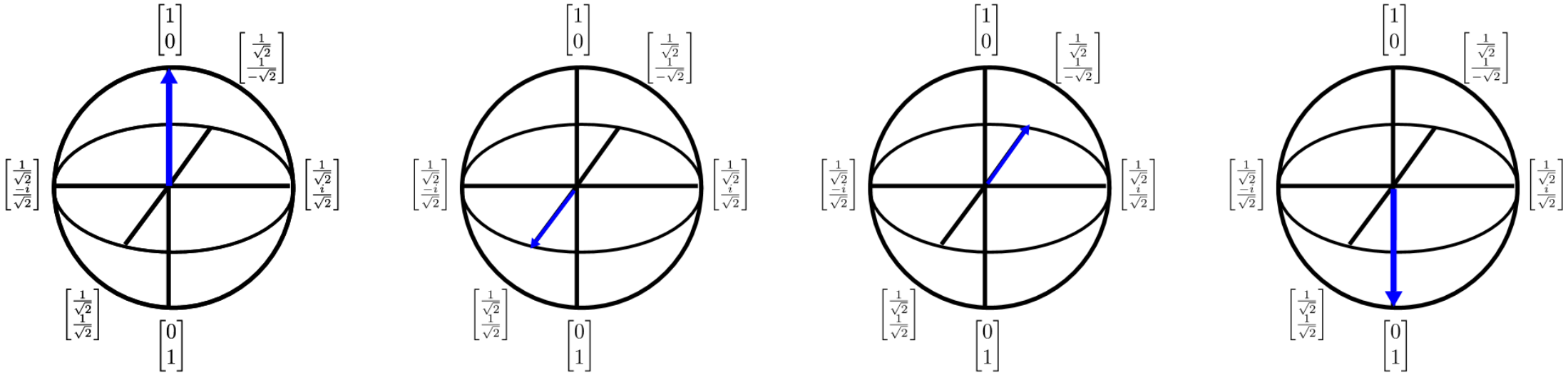
- Simulate a 30 qubit quantum computer
- Integrated into Visual Studio and VS Code
- Full debugging support

Azure Simulator

- Available for quantum solutions needing over 40 qubit simulation

Demo

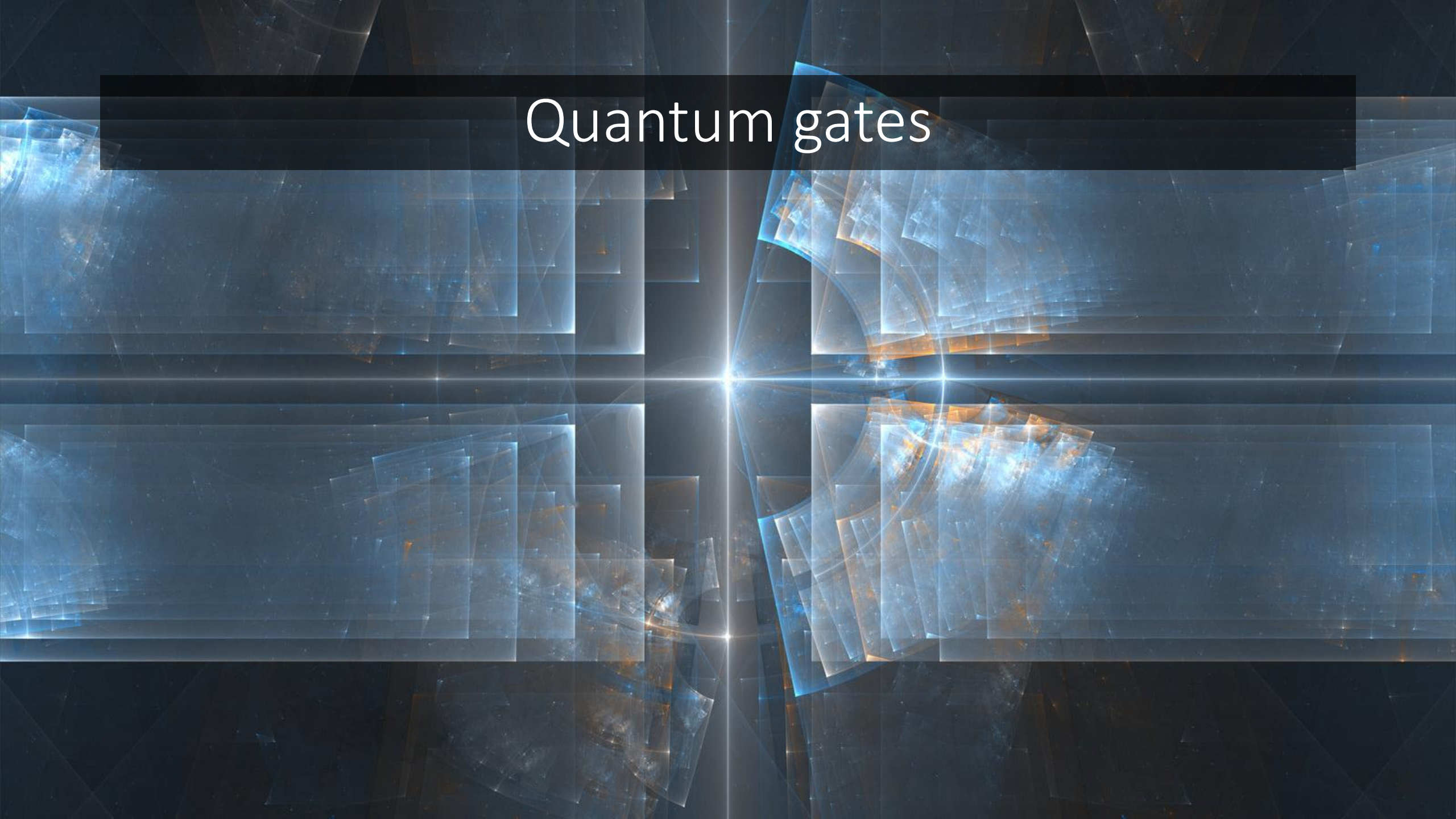
Qubits and Measurement in Q#



Representing a qubit with the Bloch sphere

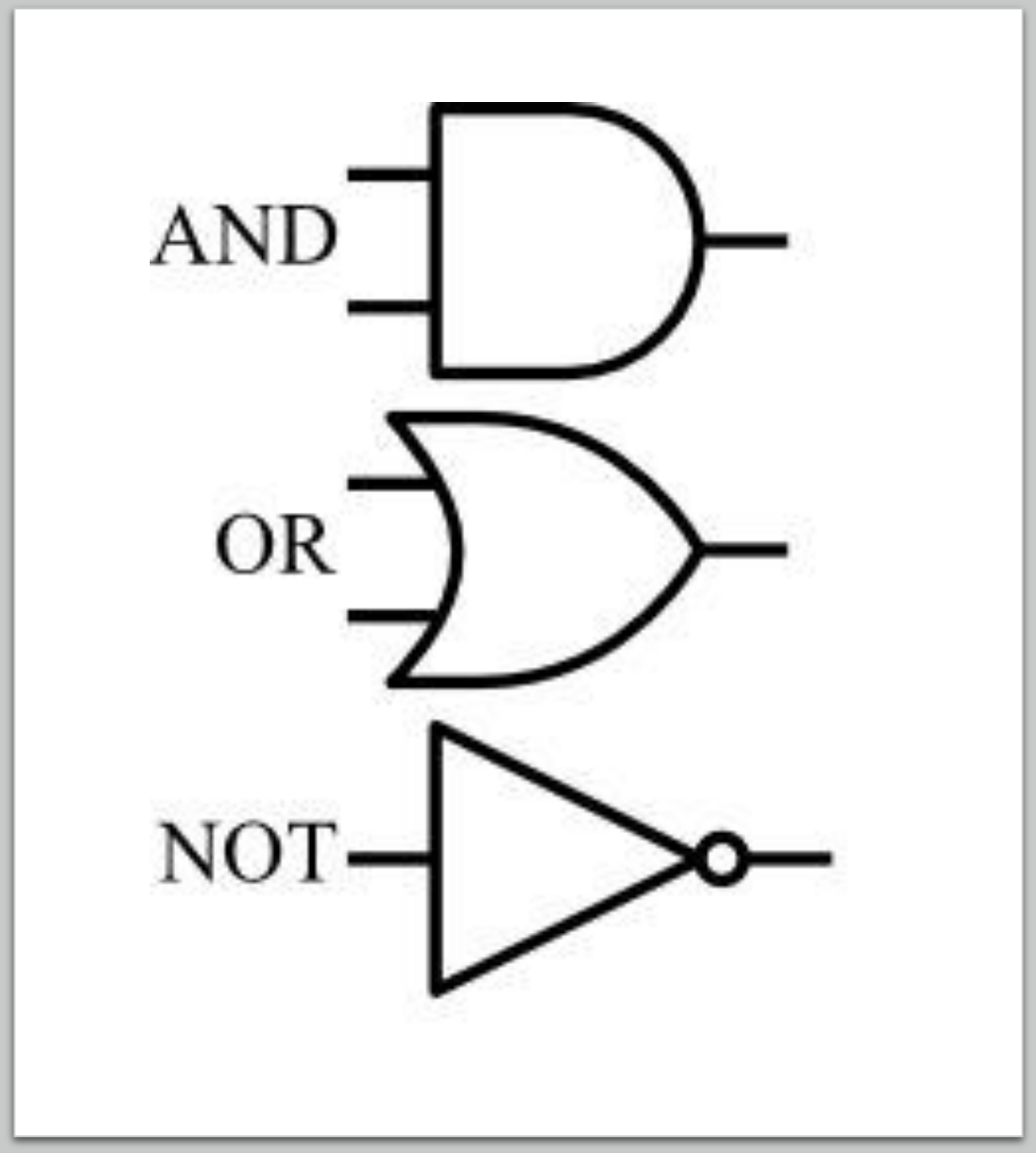
https://www.st-andrews.ac.uk/physics/quvis/simulations_html5/sims/blochsphere/blochsphere.html

Quantum gates



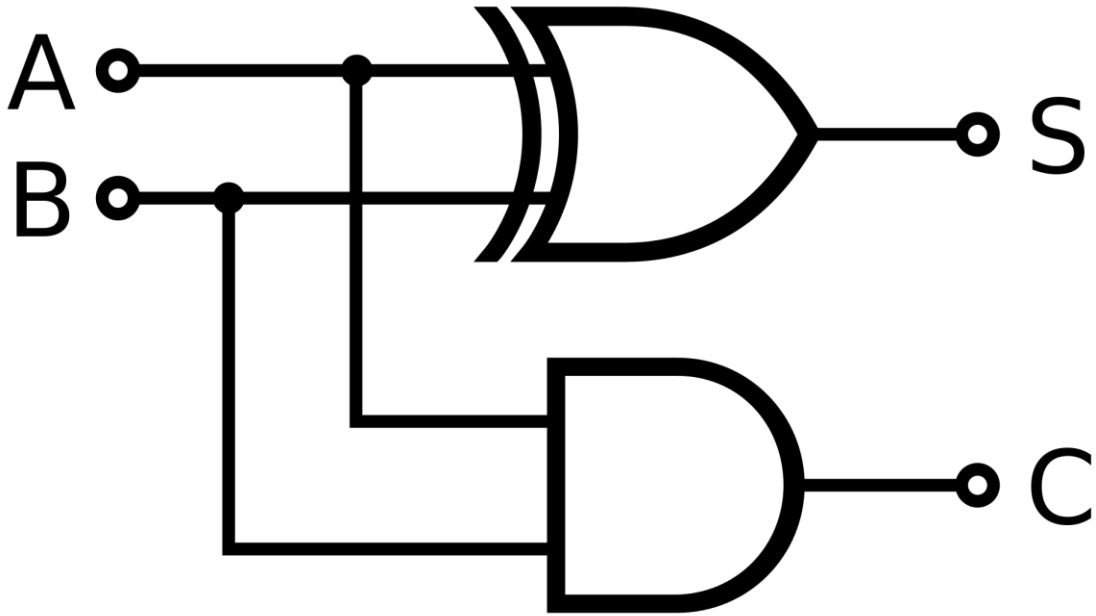
Classical computer gates

- E.g. AND, OR, NOT
- Non-linear
- Implemented by transistors, NAND gates, in silicone chips



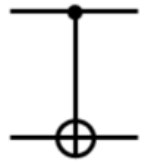
Classical Logic Circuits

Half Adder
AND / XOR gates

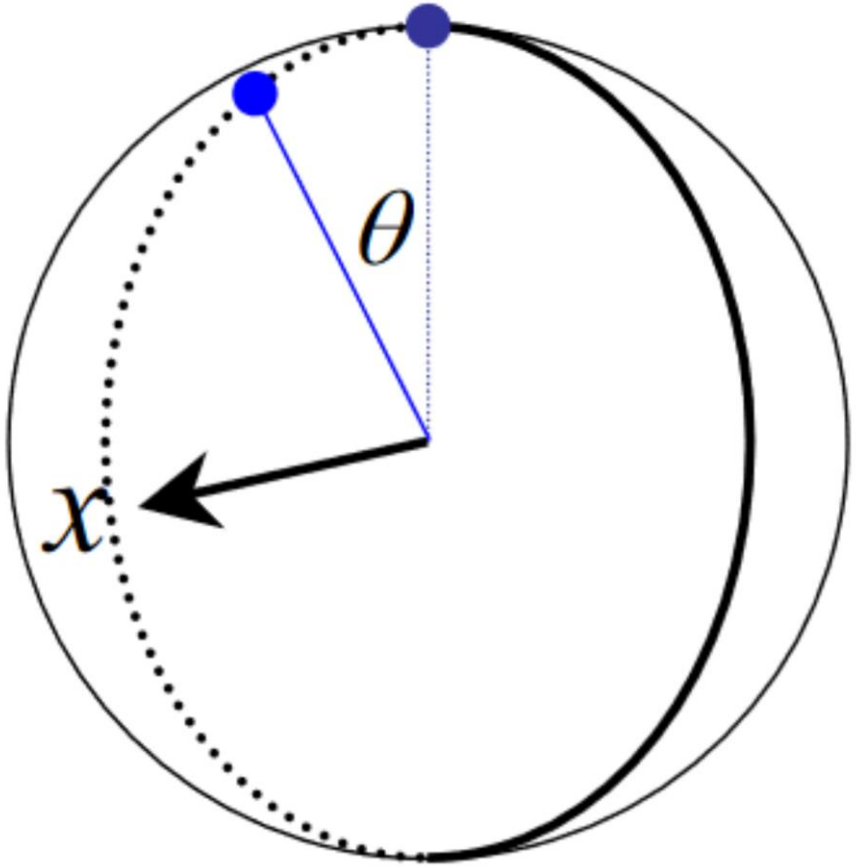


A	B	Sum (S)	Carry (C)
0	0	0	0
1	0	1	0
0	1	1	0
1	1	0	1

Quantum Gates

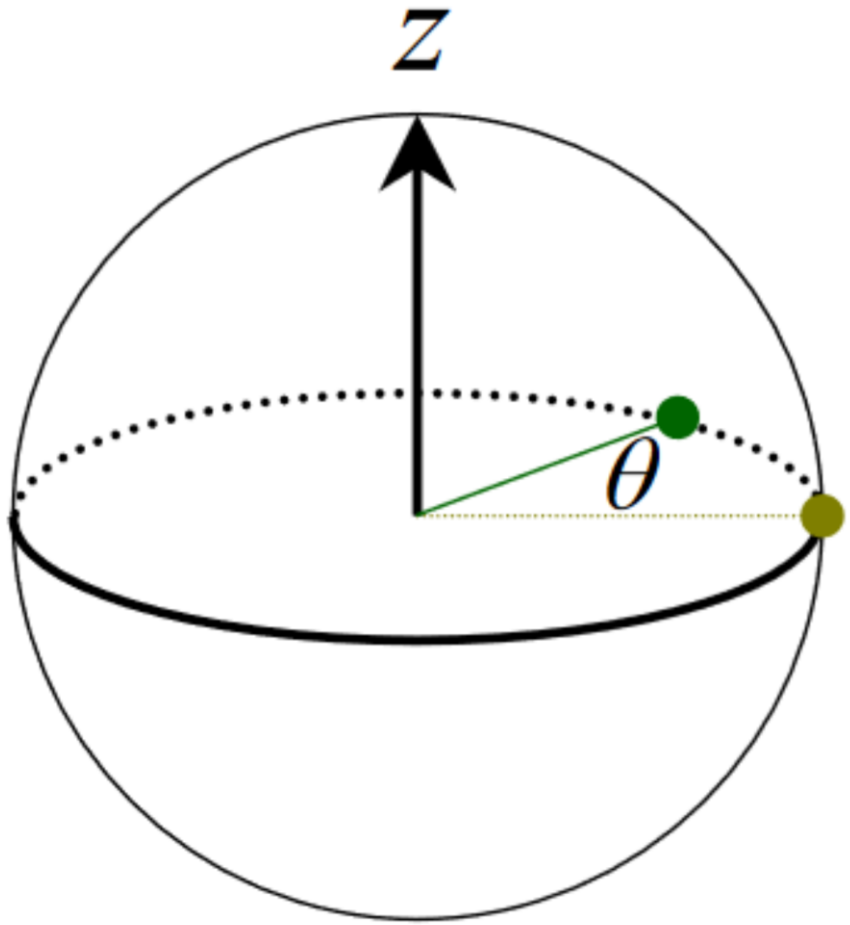
Gate	Notation	Matrix
NOT (Pauli-X)		$\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$
Pauli-Z		$\begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$
Hadamard		$\frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$
CNOT (Controlled NOT)		$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 &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

X gate



- $X = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}$
  - Names: Pauli X, X, NOT, bit flip, σ_x
- Perform equivalent of NOT gate in traditional computing

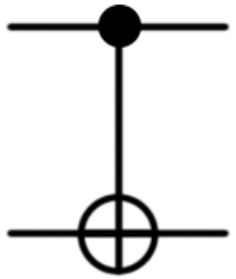
Z gate



- $Z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$
  - Names: Pauli Z, Z, phase flip, σ_z
- Rotation around the Z-axis of the Bloch sphere by 180 degrees

Hadamard gate

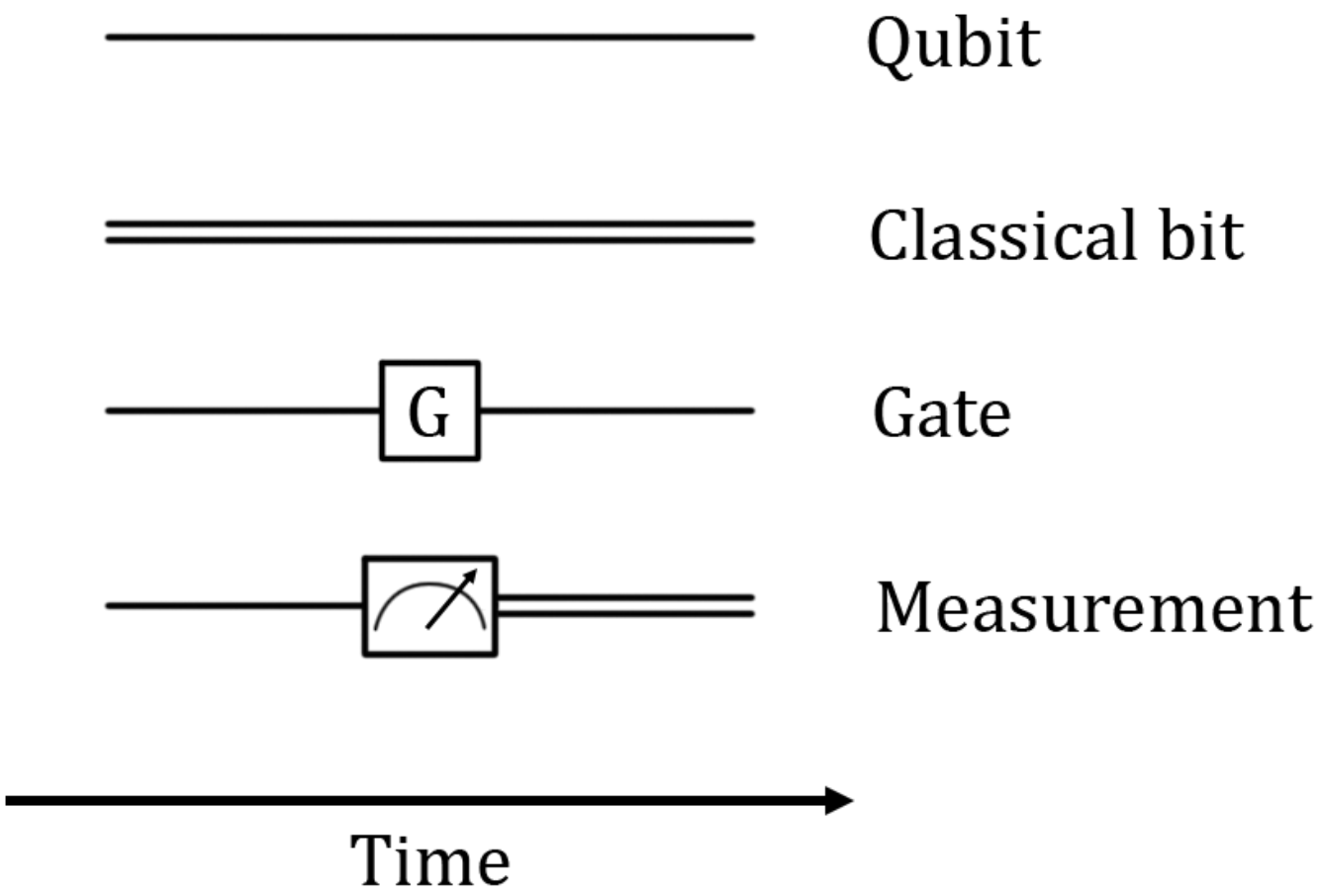
- Name: Hadamard, H
- 180 degree rotation around the diagonal X+Z axis of the Bloch sphere.
- Creates the superposition state
- $H = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$
- |0> will be transformed into $\frac{|0> + |1>}{\sqrt{2}}$
- |1> will be transformed into $\frac{|0> - |1>}{\sqrt{2}}$



CNOT gate

- $CNOT = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$
  - Controlled Not
- Acts on 2 qubits – flips the first, if the second is state 1

Diagram representation



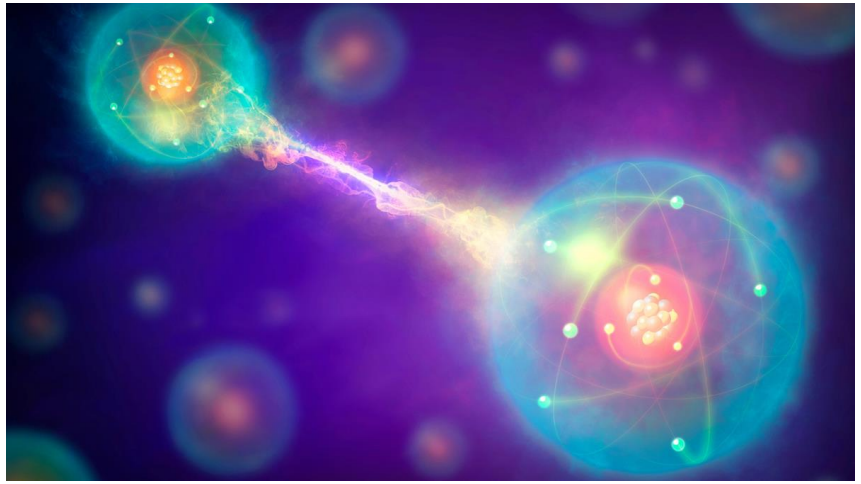
Demo

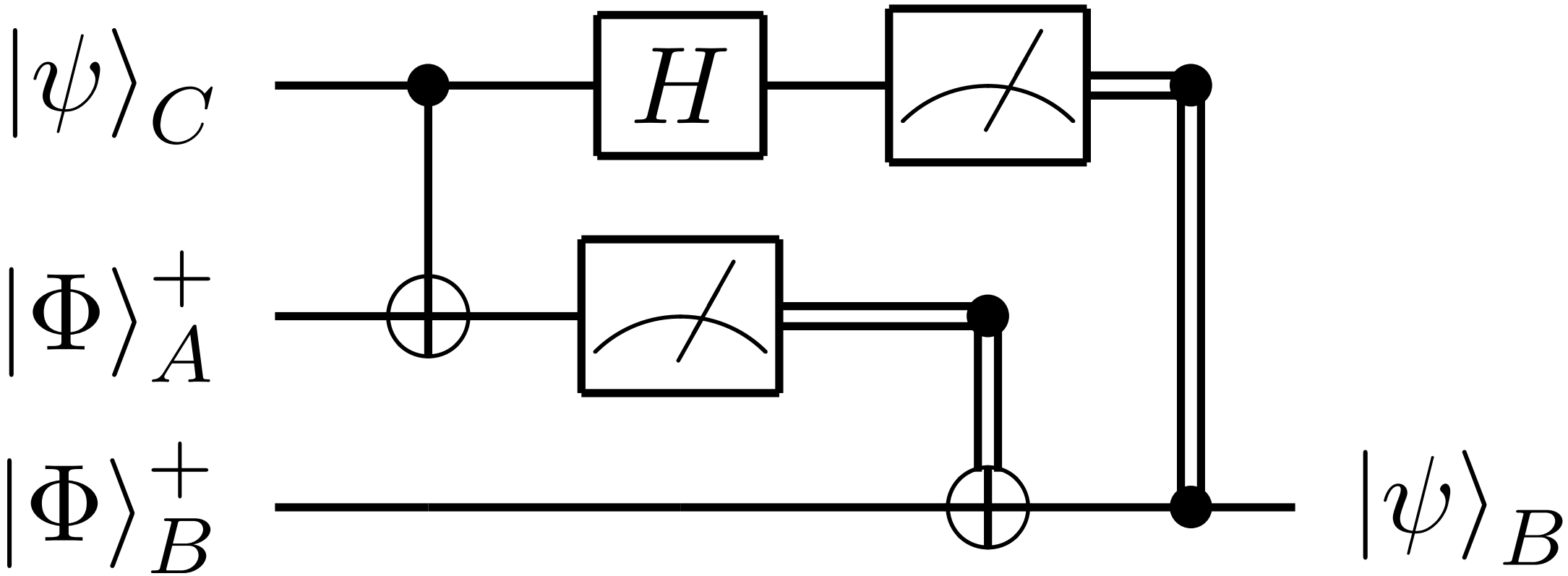
Quantum Gates in Q#

Quantum entanglement (Bell State)

- Simplest example of **entanglement**
- Named after John Bell's famous inequality
- Measurement of one qubit will assign a value to the other qubit immediately (teleportation)

$$|\Phi^+\rangle = \frac{1}{\sqrt{2}}(|0\rangle_A \otimes |0\rangle_B + |1\rangle_A \otimes |1\rangle_B).$$

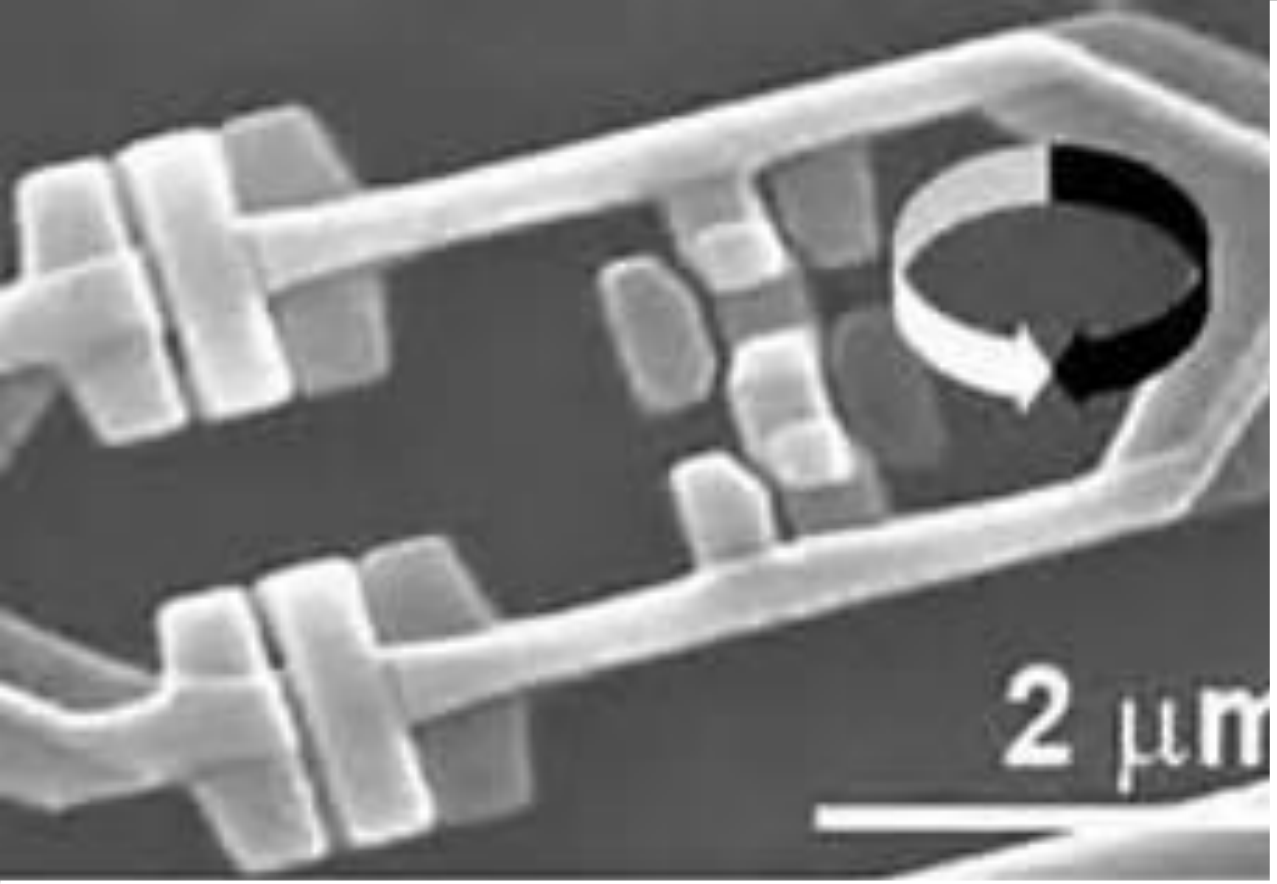


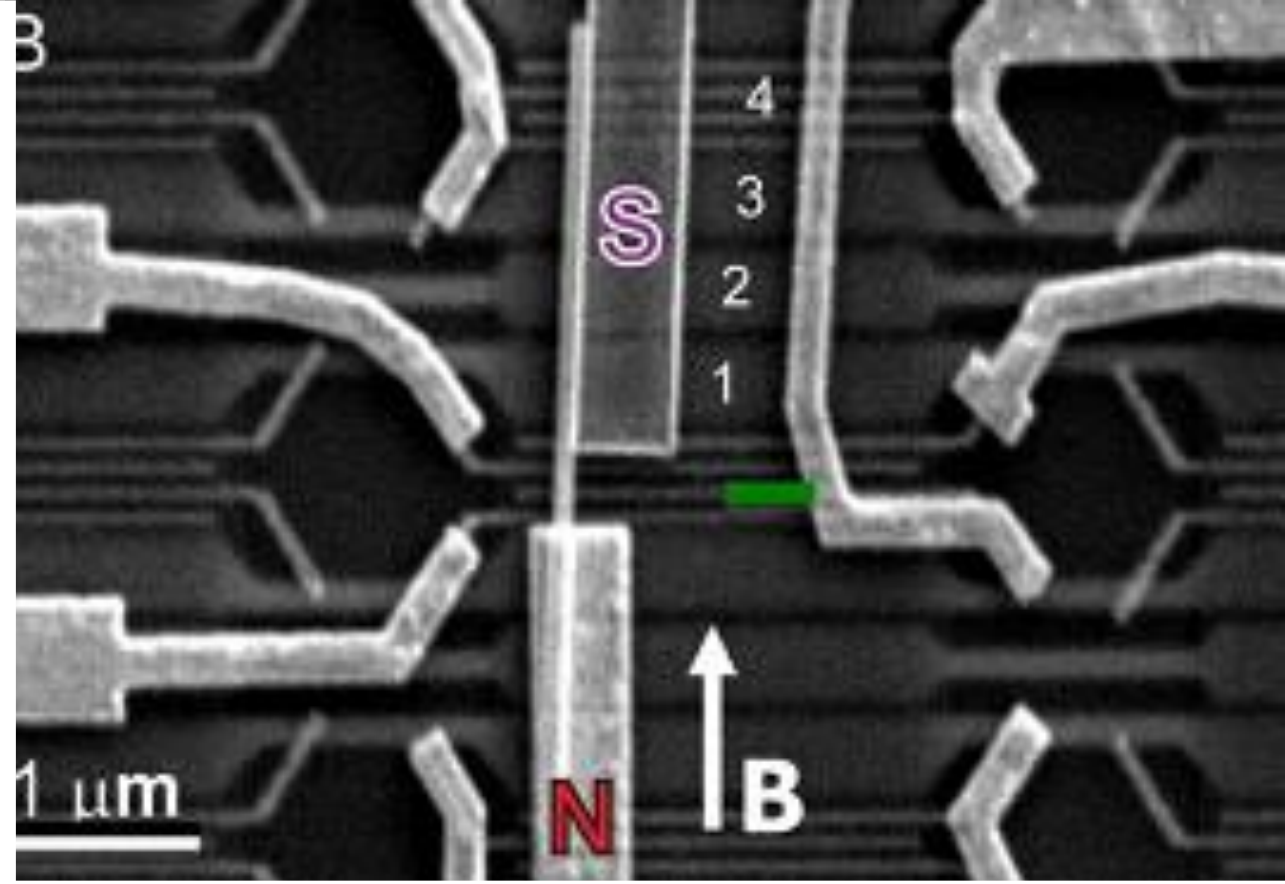


Teleportation Circuit Diagram

Demo

Entanglement & Teleportation in Q#





Quantum hardware technologies

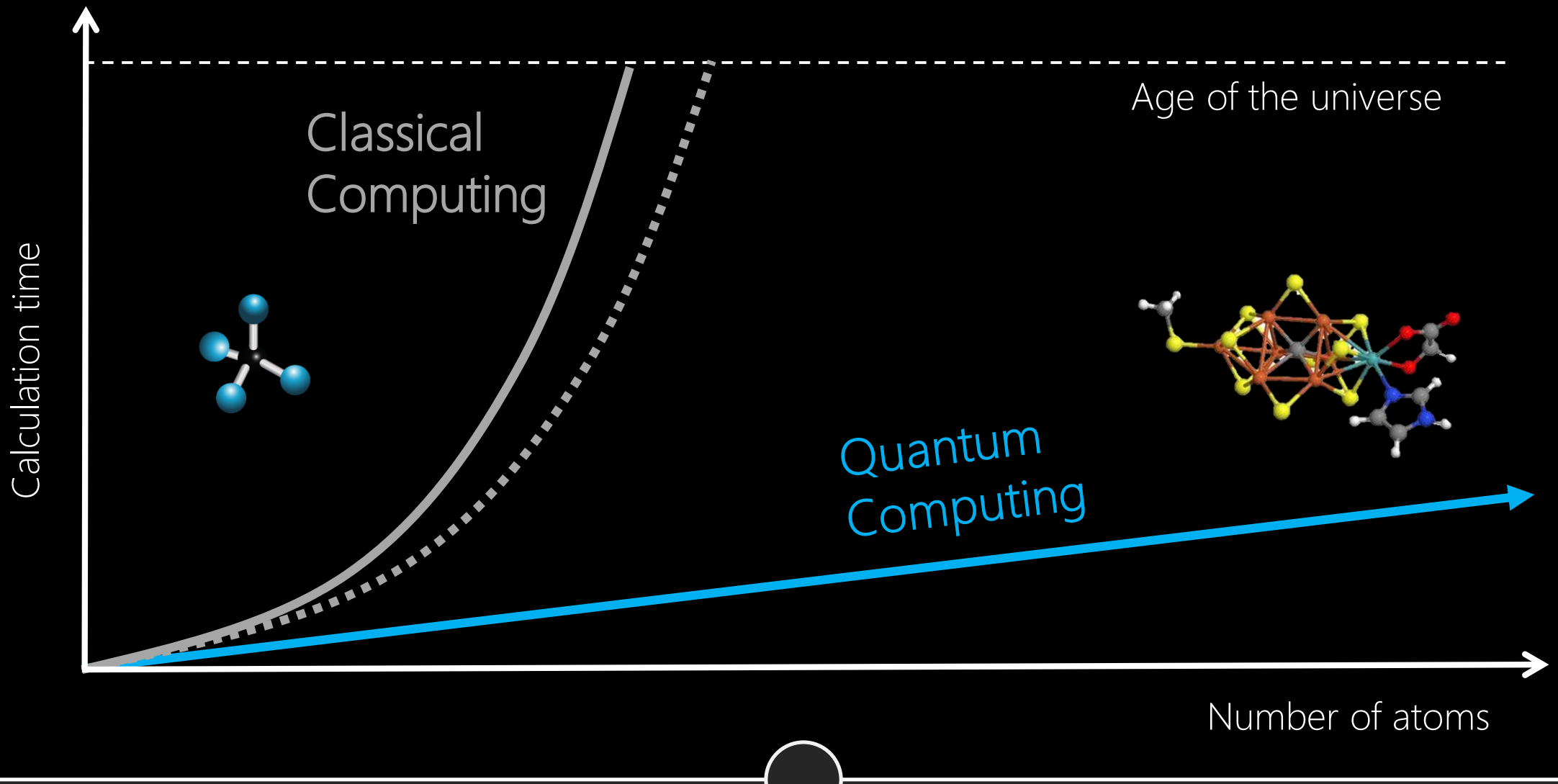
Charlie Marcus' lab in Copenhagen, Denmark

One of our primary experimental collaborators

Photos courtesy of: Professor Charlie Marcus

What should we do with a quantum computer?

- Disproving the Church Turing Thesis
- Complexity Classes
- Quantum Supremacy
- Shor’s Algorithm
- Grover’s Algorithm



Addressing classically intractable problems

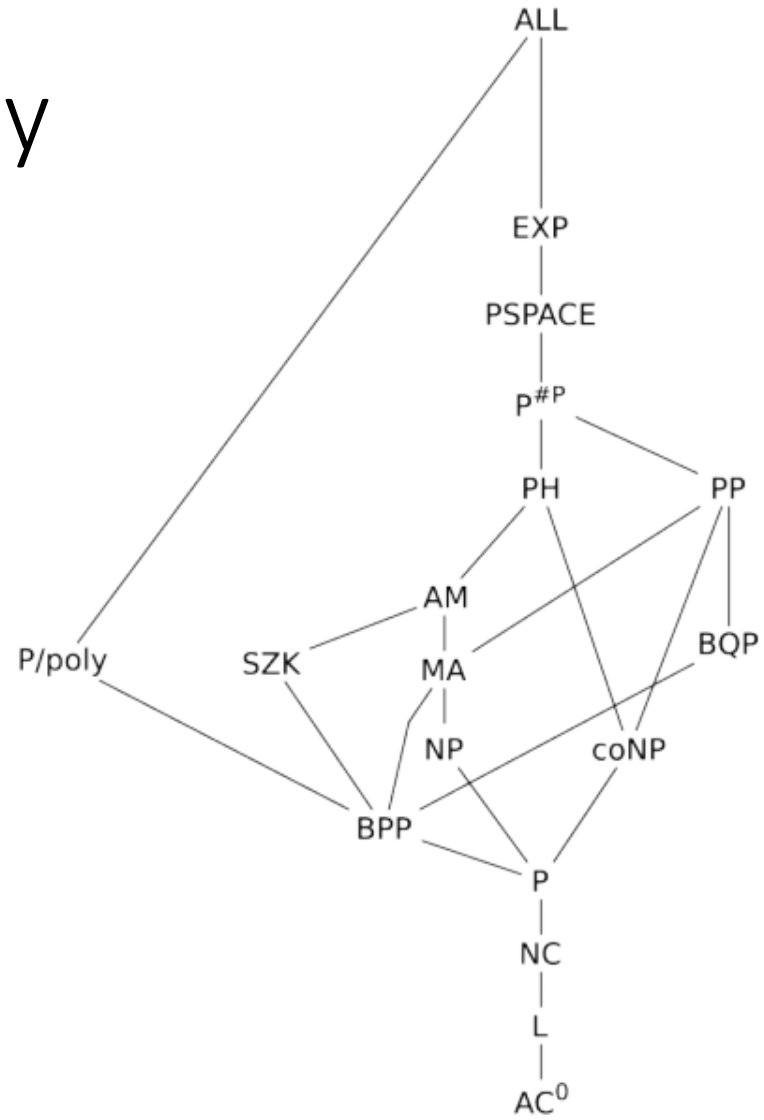
Classes of Computational Complexity

P: The class of problems which we can solve efficiently using a conventional algorithm -> class of all problems solvable by a *deterministic Turing machine bounded in time* by a **polynomial** function of the input length.

NP: the set of problems such that someone can convince you of a *yes* answer in a reasonable amount of time.

BPP: The class of problems that are solvable efficiently by randomized algorithms.

BQP: The class of all problems that admit an efficient solution by quantum computers.



(Disproving)
The Church-Turing Thesis

CT thesis is a statement about computability.

"Everything that is computable by any device you can physically build, is also computable by a Turing machine".

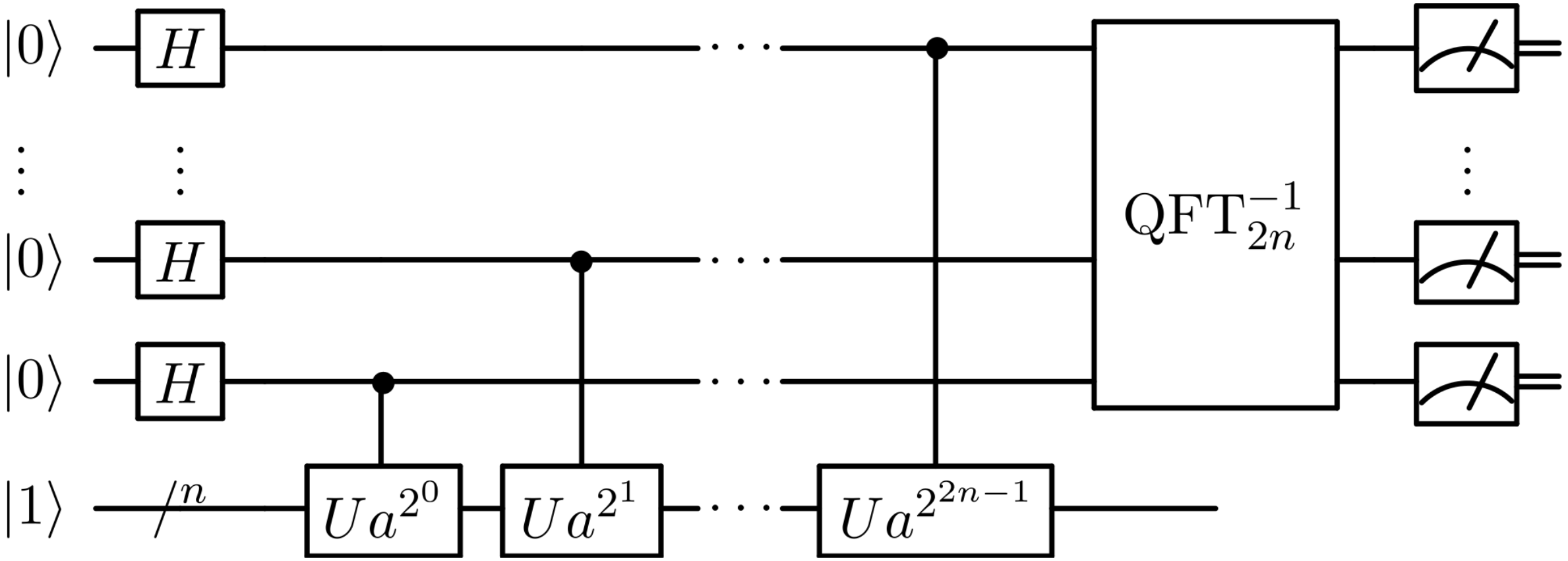
Extended: efficiently computable. I.e. anything we can compute in polynomial time , complexity class P or BPP.

Quantum Supremacy

Quantum >> Classical

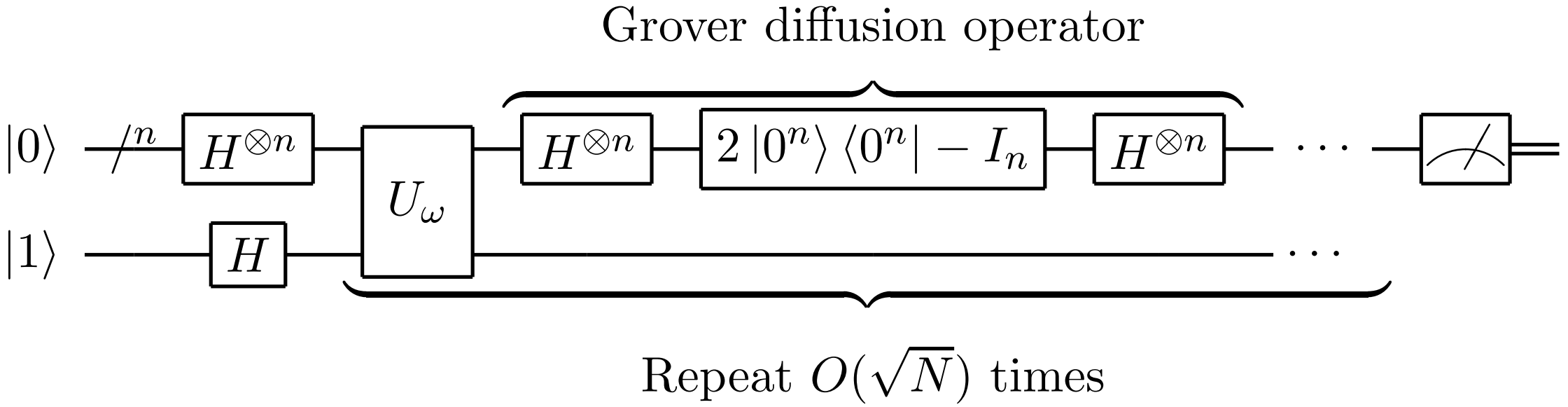
- Classically impossible
- Quantumly possible
- Useless but Hard

- Prove: Non-existence of a classical algorithm to do what a QC can do



Shor's Algorithm

Factor an integer N in O((log N)2(log log N)(log log log N))



Grover's Algorithm

Finds the unique input to a black box function that produces a particular output value, using just O(√N) evaluations of the function.


https://aka.ms/learnquantum