

Programming Quantum Computers (Introduction)

(Subtrack of Quantum Computing: An App-Oriented Approach)

Moez A. AbdelGawad

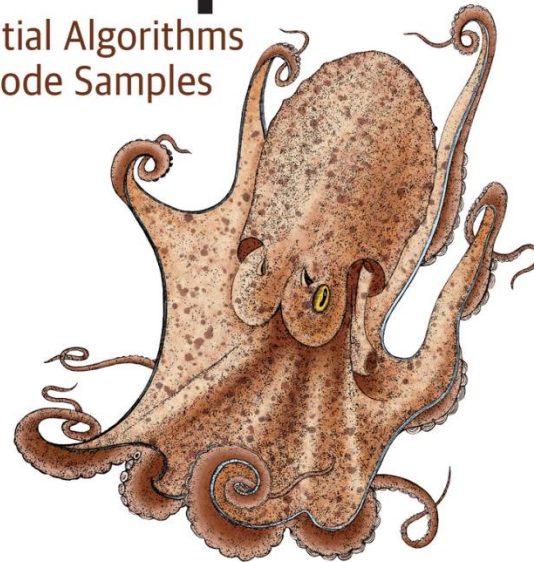
moez@{cs.rice.edu, alexu.edu.eg, srtacity.sci.eg}

Sat., Oct. 19th, 2019

O'REILLY®

Programming Quantum Computers

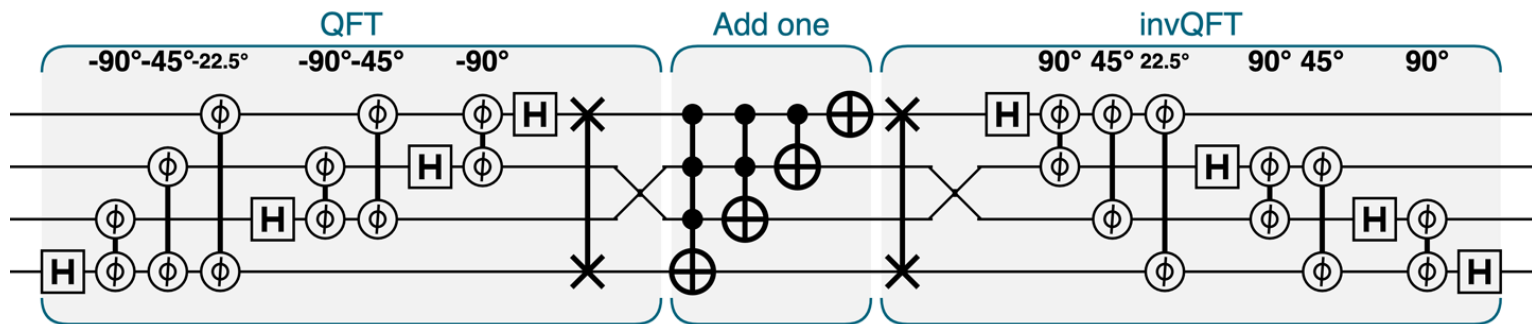
Essential Algorithms
and Code Samples



Eric R. Johnston, Nic Harrigan
& Mercedes Gimeno-Segovia

Quantum Computers are Real

- What are they useful for?
 - Let's discover, by programming them!
- A hands-on approach to programming QCs/QPUs.
 - By doing; i.e., by writing code & building programs.
 - Using simulators, since real QCs are harder-to-access (so far).
- Goals: Read, understand, write, and *debug* quantum programs.
 - Ones like the following.



PQC Outline (Tentative)

- **Introduction.**
 - One Lect. (Ch. 1 & 2)
 - Qubits, Superposition, and One-Qubit Primitives.
- Primitives.
 - One Lect. (Ch. 3 & 4)
 - Multiple Qubits and Entanglement.
 - A Program for Teleportation.
- Libraries.
 - Two Lects. (Ch. 5-8)
 - Arith. & Logic.
 - Amp. Amplification, QFT & Phase Est.
- Applications.
 - Two-Three Lects. (Ch. 9-13)
 - Real Data.
 - Search. Supersampling and QIP.
 - Shor's Factoring Algorithm, Quantum ML.

Lecture Outline

- Introduction & One Qubit Primitives.
 - QCEngine.
 - Qubits.
 - Physical, Logical. Bloch Sphere, Circle Notation.
 - Primitive Qubit Operations (PrimOps).
 - Simple Quantum Programs.
 - Random Generators, Quantum Spy Hunter.

Section I

QCENGINE AND QUBITS

QCEngine: An Online Quantum Simulator

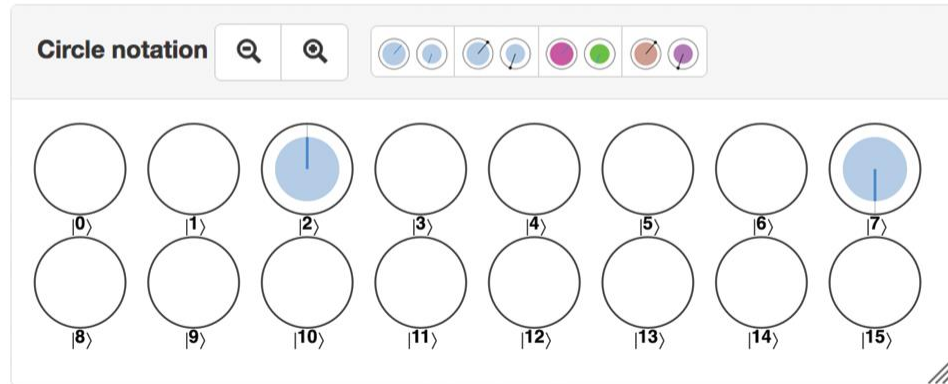
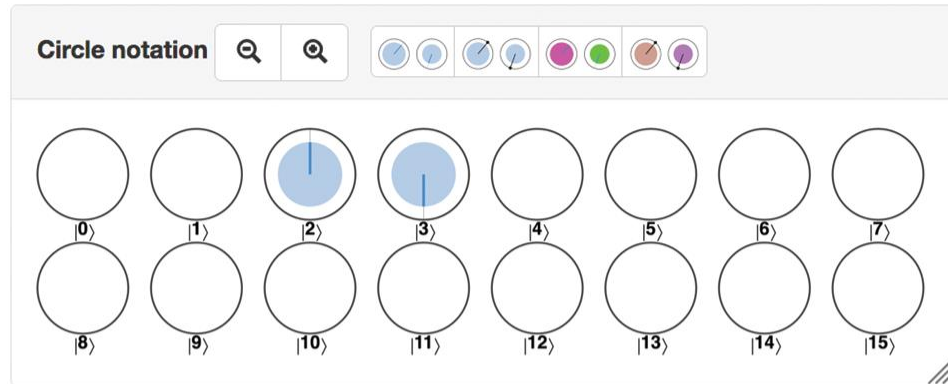
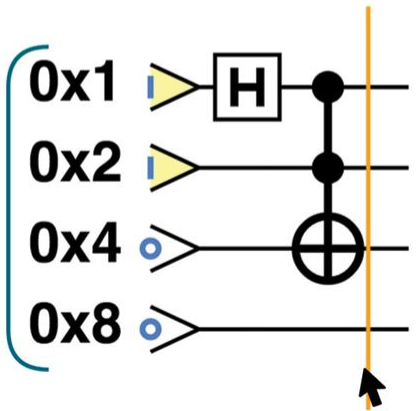
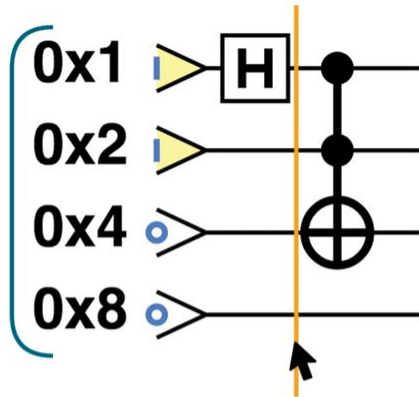
- <https://oreilly-qc.github.io/>

The screenshot shows the QCEngine web interface. At the top is a red navigation bar with the O'Reilly logo and links for 'Ideas', 'Learning Platform', 'Conferences', and 'Shop'. Below the navigation bar is a header section with the O'Reilly book cover for 'Programming Quantum Computers' on the left and the text 'Code Samples' on the right. The main content area is divided into three panels. The left panel, titled 'Run Program', shows a code editor with a 'Run Program' button and search icons. The code is as follows:

```
1 // Programming Quantum Computers
2 // by Eric Johnston, Nic Harrigan and Mercedes Gimeno-Segovia
3 // O'Reilly Media
4
5 // To run this online, go to http://oreilly-qc.github.io?p=2-1
6
7 // This sample generates a single random bit.
8
9 qc.reset(1); // allocate one qubit
10 qc.write(0); // write the value zero
11 qc.had(); // place it into superposition of 0 and 1
12 var result = qc.read(); // read the result as a digital bit
13
```

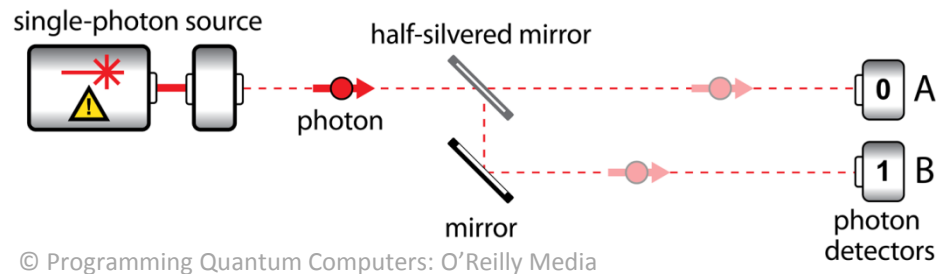
The middle panel, titled 'Program output', contains a text area with the placeholder text '(output prints here)'. The right panel, titled 'Program circuit', displays a quantum circuit diagram with a qubit initialized to 0x1, followed by a Hadamard (H) gate. Below the circuit diagram is a 'Circle notation' panel showing two circles representing the $|0\rangle$ and $|1\rangle$ states.

Debugging in QCEngine



What's a Qubit?

- Qubit = Quantum bit.
- Physical/Concrete Qubits vs. Logical/Abstract Qubits.
- Physical Qubit: Photon, electron, ion, ...
 - Needs physics knowledge.
 - Imperfect (decoherence). Needs error-correction.



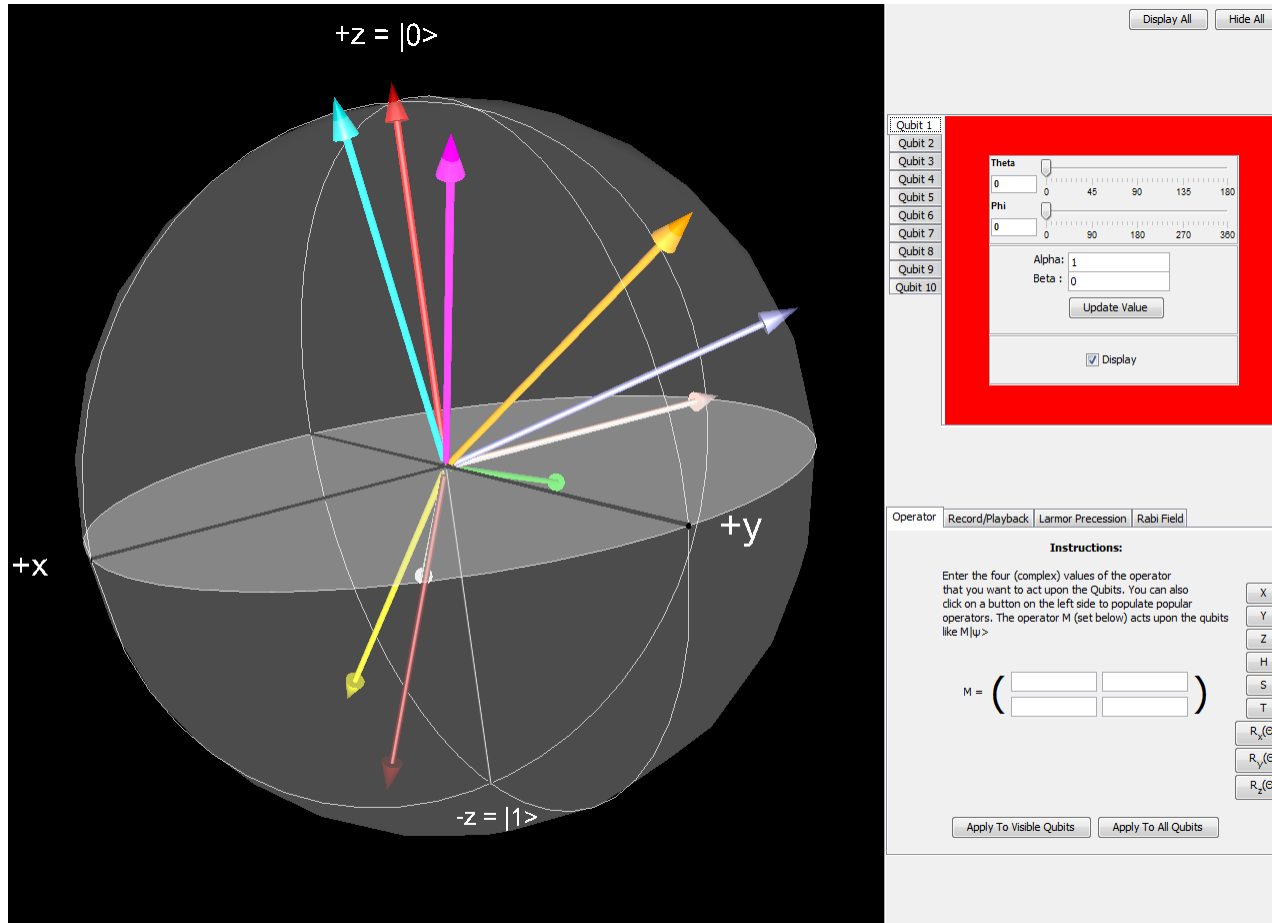
What's a Qubit?

- Logical Qubit:
 - Simpler (no physics).
 - Ideal (no worrying about errors).
 - May correspond to more than one physical qubit.
 - For programming, we use *logical* qubits.
 - Portable quantum programs, independent of underlying hardware.
 - Basic qubit values (quantum states):
 - Quantum_Zero $|0\rangle$ and Quantum_One $|1\rangle$.

What's a (Logical) Qubit?

- Superposition: $c_0|0\rangle + c_1|1\rangle$
- Circle-notation:
 - Complex numbers c , and their conjugates c^* .
 - Amplitude: $c = a + ib = re^{i\theta}$ ($i = \sqrt{-1}$; a, b, r, θ are reals)
 - Magnitude: $r = |c| = \sqrt{c \times c^*} = \sqrt{a^2 + b^2}$
 - Phase: $\theta = \tan^{-1}(b/a)$
 - Probability: $r^2 = a^2 + b^2$ (square of magnitude)
 - $r_0^2 + r_1^2 = 1.0$ (sum of probabilities)
 - Relative phase (difference between phases): $\theta_1 - \theta_0 | 360^\circ$.
 - Visualizing Logical Qubits:
 - Bloch Sphere (Bloch Sphere Simulator, dotBloch App)
 - Three free variables (degrees of freedom) rather than four, or in fact just two (Theta and Phi).
 - To the rescue: **Circle-notation**
 - Visual, simple. No physics, no (complex) numbers!

Bloch Sphere Simulator

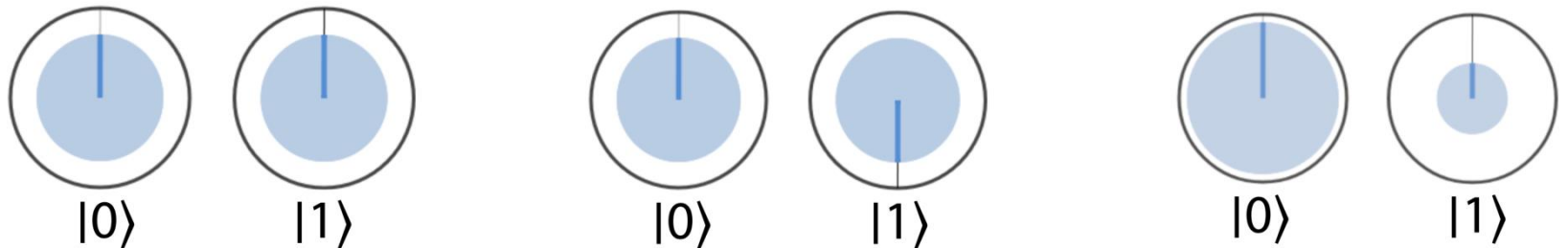


Circle-Notation

- Conventional bits, and qubits *after* readout.



- Qubits *before* readout (superposition).




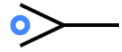


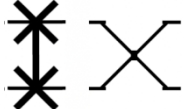



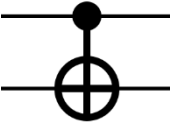
2019-10-19 Copyright © 2019-24 Moez A. AbdelGawad

$$0.707|0\rangle + 0.707|1\rangle \quad 0.707|0\rangle - 0.707|1\rangle \quad 0.95|0\rangle + 0.34|1\rangle$$

Section II

PRIMITIVE QUANTUM OPERATIONS AND SIMPLE QUANTUM PROGRAMS

Primitive Quantum Operations (PrimOps)

- **Read**  `val qc.read(t)`
- **Write**  `qc.write(t, val)`
- **No-Op**  `qc.nop()`
- **Not** 
- **Swap (2 qubits)** 
- **Hadamard (Superposition)** 
- **Phase (deg.)** 
- **Root-Not** 
- **Cond. Op. (C-Not, C-Swap, C-Phase)** 
 - Multi-qubits (2 or more).

```
val qc.read(t)
```

```
qc.write(t, val)
```

```
qc.nop()
```

```
qc.not(t)
```

```
qc.exchange(t1|t2)
```

```
qc.had(t)
```

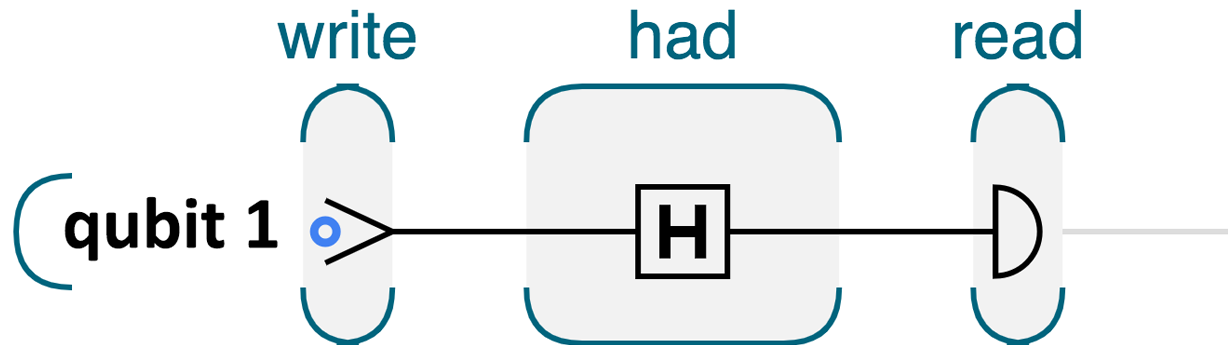
```
qc.phase(angle, t)
```

```
qc.rootnot(t)
```

```
qc.cnot(t, c)
```

```
qc.cnot(t, c1|c2)
```

Our First Quantum Program (Hands-on: Let's Play Online!)



Perfect Random Bit Generator

PrimOps: More Details

- **Read:** Converts a superposition to black/white (measurement). Random, according to probabilities.
 - Superposition, including phase, is irreversibly destroyed (Subsequent reads return same value).
- **Write:** Puts a definite black/white value (only $|0\rangle$ and $|1\rangle$ are allowed).
- **No-Op:** Does nothing (“time passage”).

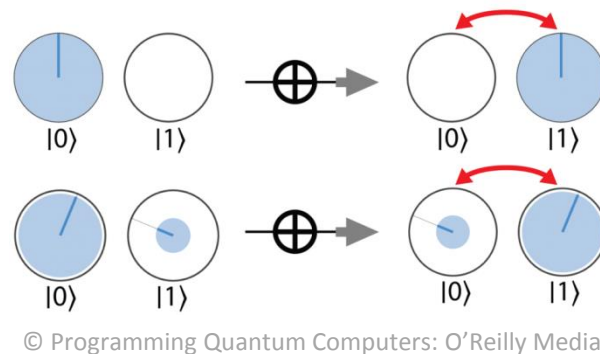
- **Not:** Flips/swaps probabilities and phase. Reversible.
- **Swap:** Exchanges states of two qubits.

- **Hadamard:** Puts into superposition ($|0\rangle$ goes to perfect 50%-50% superposition, $|1\rangle$ goes to same but with relative phase 180°). Reversible. (Averaging. Best understood as matrix.)
- **Phase:** Shifts phase of $|1\rangle$ relative to that of $|0\rangle$.
 - **Rot-X** and **Rot-Y** (on Bloch sphere).
- **Root-Not:** Squared (repeated twice) gives **Not**.

- More into math? Complex vectors, matrices, linear algebra, tensor products,

PrimOps: More Details (using circle-notation)

- **Not:**



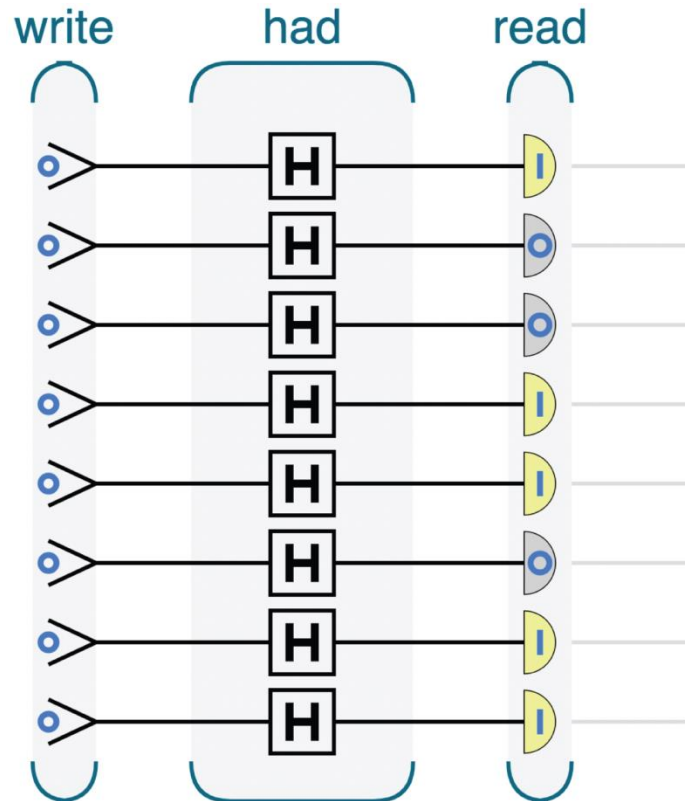
- Similarly for other primops.
 - **Phase:** Rotates $|1\rangle$ (right circle) counterclockwise.
 - **H (Hadamard):** Average and difference (matrix).
 - Any questions?
- Combinations/compositions, and equivalences.
 - Equiv. are basis for transformations and optimizations.
 - For example, $H \circ Z \circ H = \oplus$ and $H \circ \oplus \circ H = Z$.

No 'Copy' Operation

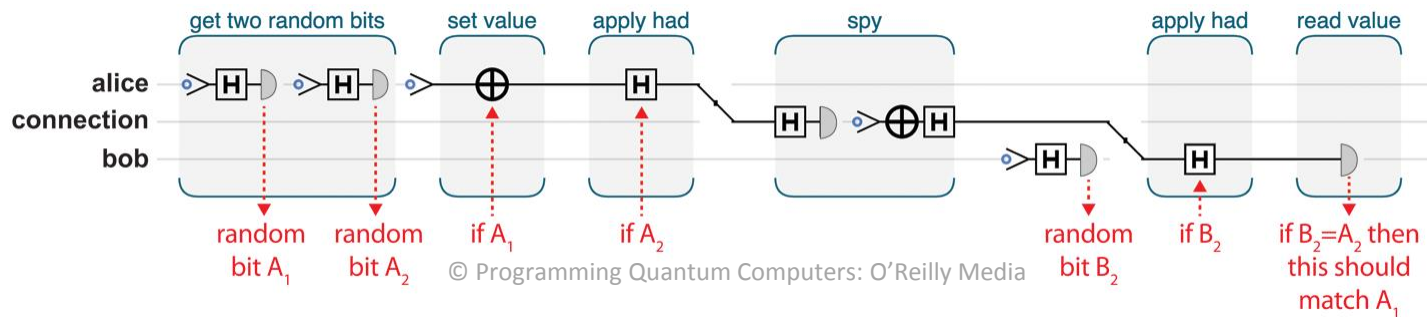
- A fundamental counterintuitive property of QC.
 - Quantum states cannot be replicated.
 - Not even using entanglement. Originals are always destroyed.
- App-specific workarounds typically employed.

A Slightly More Advanced Program

- Can you guess what does program do?



Quantum Spy Hunter



- Using the amazing laws of quantum physics/computing (e.g., no copy op.) to our advantage.
- Using 50 qubits, probability of spy *not* getting caught is less than $\frac{1}{1,000,000}$.
 - Precisely is $(3/4)^{50}$.
 - Every 10 qubits divide the probability of not catching spy by more than 10 (precisely, by 17.75. Every 5 qubits divide by 4.2).

Discussion

Q & A

Next Lecture Appetizer!

- In next lecture:
 - Multi-Qubit Operations
 - Controlled Ops.
 - Entanglement.
 - Teleportation!
 - On IBM's quantum computer!

Thank You