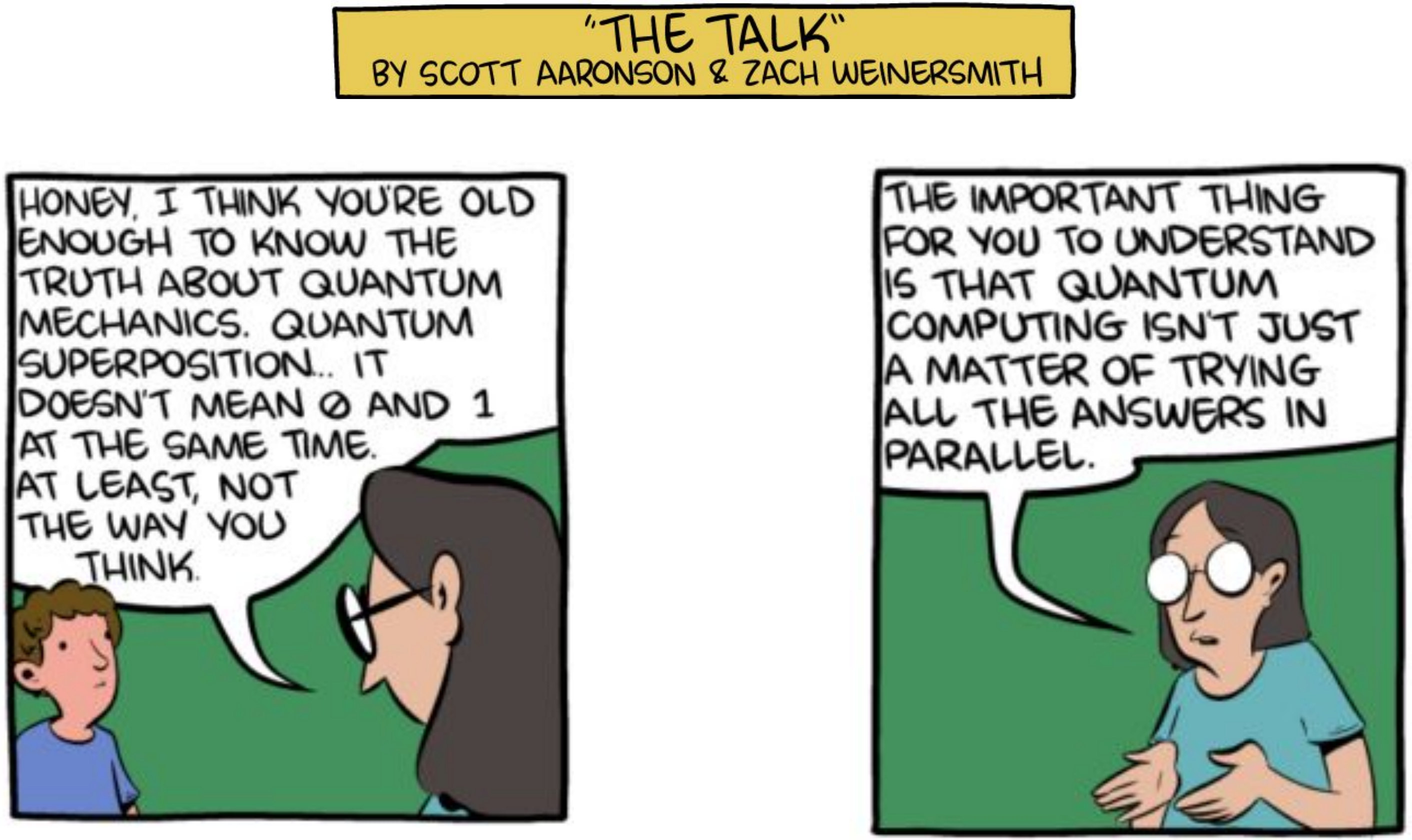
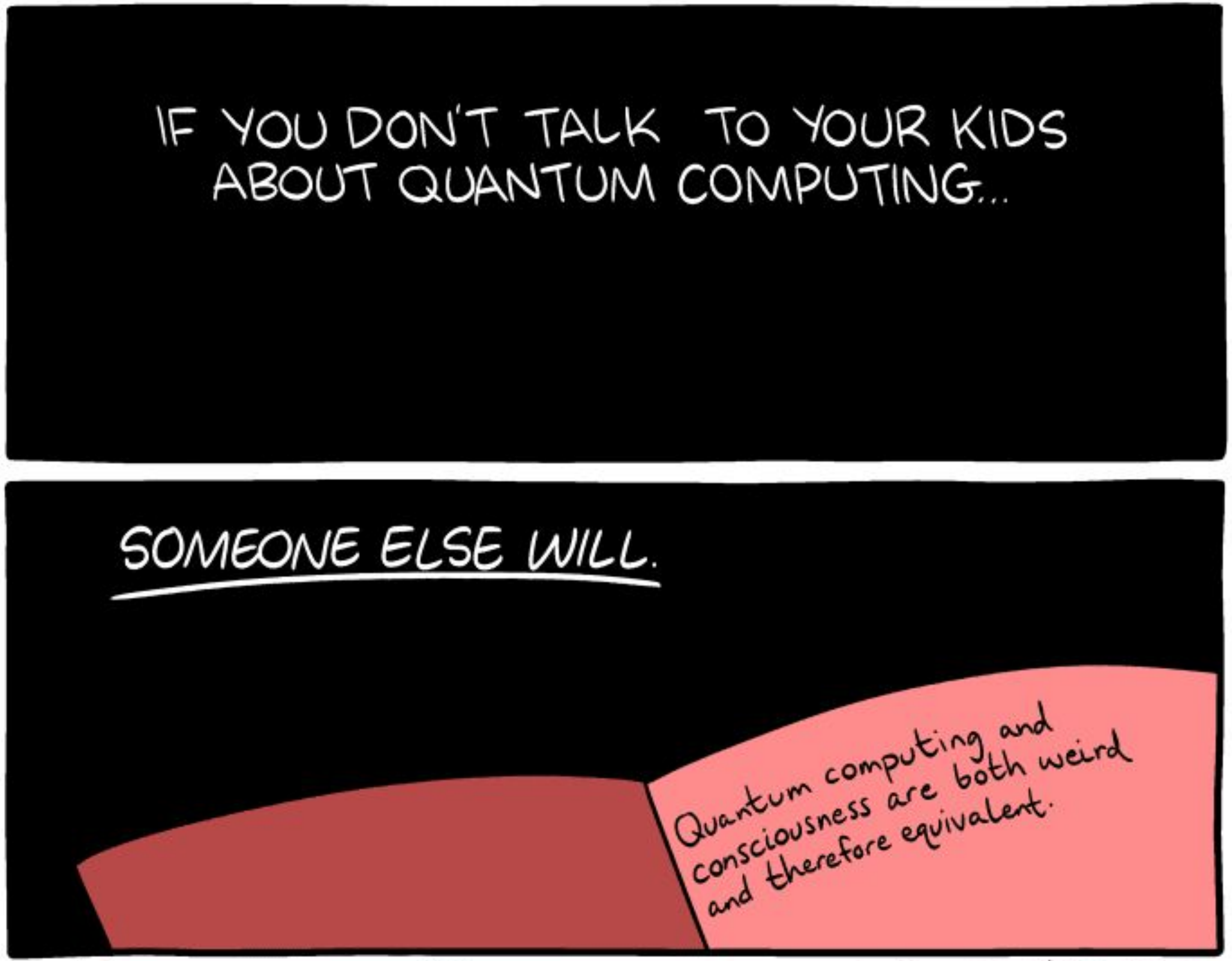
CS269: Quantum Computer Programming

Dan Boneh & Will Zeng + Guests


"THE TALK"
BY SCOTT AARONSON & ZACH WEINERSMITH
HONEY, I THINK YOU'RE OLD ENOUGH TO KNOW THE TRUTH ABOUT QUANTUM MECHANICS. QUANTUM SUPERPOSITION... IT DOESN'T MEAN 0 AND 1 AT THE SAME TIME. AT LEAST, NOT THE WAY YOU THINK.
THE IMPORTANT THING FOR YOU TO UNDERSTAND IS THAT QUANTUM COMPUTING ISN'T JUST A MATTER OF TRYING ALL THE ANSWERS IN PARALLEL.

http://www.smbc-comics.com/comic/the-talk-3


IF YOU DON'T TALK TO YOUR KIDS ABOUT QUANTUM COMPUTING...
SOMEONE ELSE WILL.
Quantum computing and consciousness are both weird and therefore equivalent.

scottaaronson.com/blog smbc-comics.com

This course is:

At the leading edge of a new technology, discipline, and industry

A programming-first approach

A great way to challenge yourself to think about computation in a totally new way

A way to learn “just enough” quantum physics

An **experiment!**

Course details

Online at: http://cs269q.stanford.edu

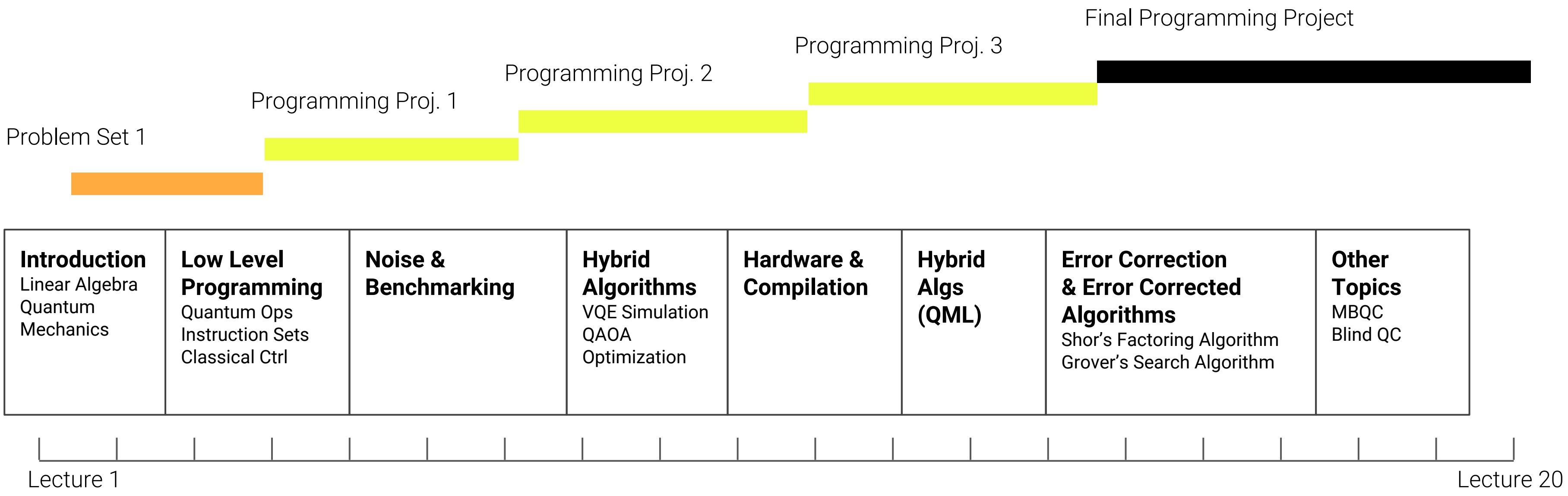
***Two** lectures per week. Tuesday, Thursday 10:30-11:50, McCullough 115*

There will be **one** written problem sets, **three** programming projects, and **one** final programming project.

Textbook: Quantum Computation and Quantum Information: 10th Anniversary Edition by Michael A. Nielsen and Isaac L. Chuang

Readings: posted online with the syllabus for each lecture. These are critical.

Course Topics & Timeline



Final Programming Project

Programming Proj. 3

Programming Proj. 2

Programming Proj. 1

Problem Set 1

Introduction
Linear Algebra
Quantum Mechanics

Low Level Programming
Quantum Ops
Instruction Sets
Classical Ctrl

Noise & Benchmarking

Hybrid Algorithms
VQE Simulation
QAOA
Optimization

Hardware & Compilation

Hybrid Algs (QML)

Error Correction & Error Corrected Algorithms
Shor’s Factoring Algorithm
Grover’s Search Algorithm

Other Topics
MBQC
Blind QC

Lecture 1

Lecture 20

Quantum Computing isn’t the answer to everything.

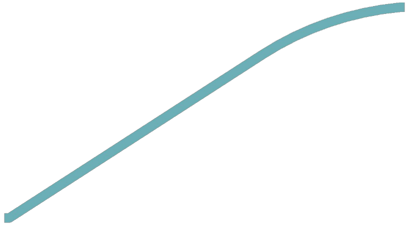
But it will almost certainly free us to **solve more problems.**

Today’s lecture:

Q1. Why program a quantum computer?

Q2. How do I program a quantum computer?

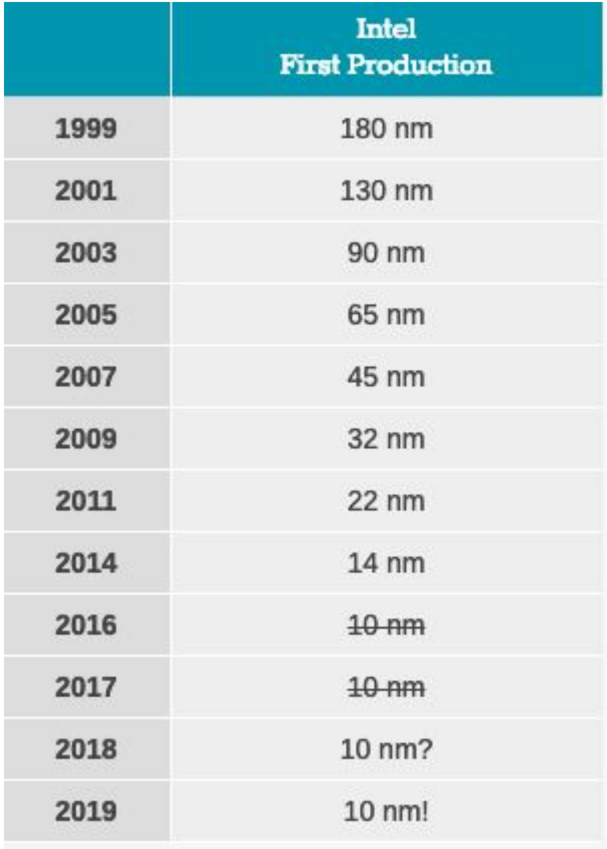
Classical computers have fundamental limits



Transistor scaling

Economic limits with 10bn for next node fab

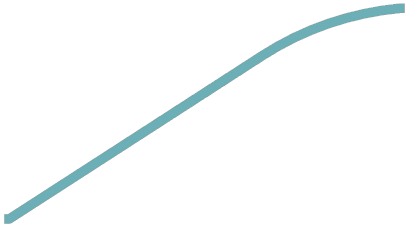
Ultimate single-atom limits



	Intel First Production
1999	180 nm
2001	130 nm
2003	90 nm
2005	65 nm
2007	45 nm
2009	32 nm
2011	22 nm
2014	14 nm
2016	~~10 nm~~
2017	~~10 nm~~
2018	10 nm?
2019	10 nm!

Source: https://www.anandtech.com/show/12693/intel-delays-mass-production-of-10-nm-cpus-to-2019

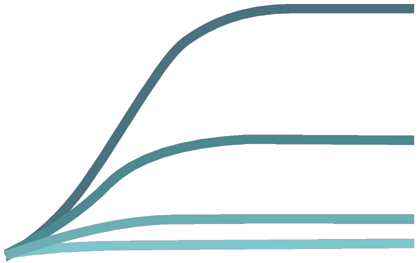
Classical computers have fundamental limits



Transistor scaling

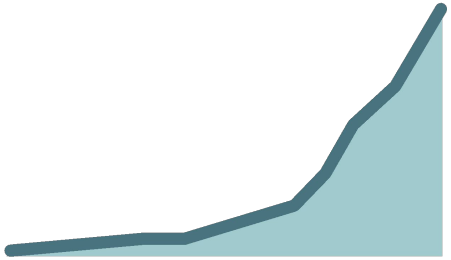
Economic limits with 10bn for next node fab

Ultimate single-atom limits



Returns to parallelization

Amdahl’s law

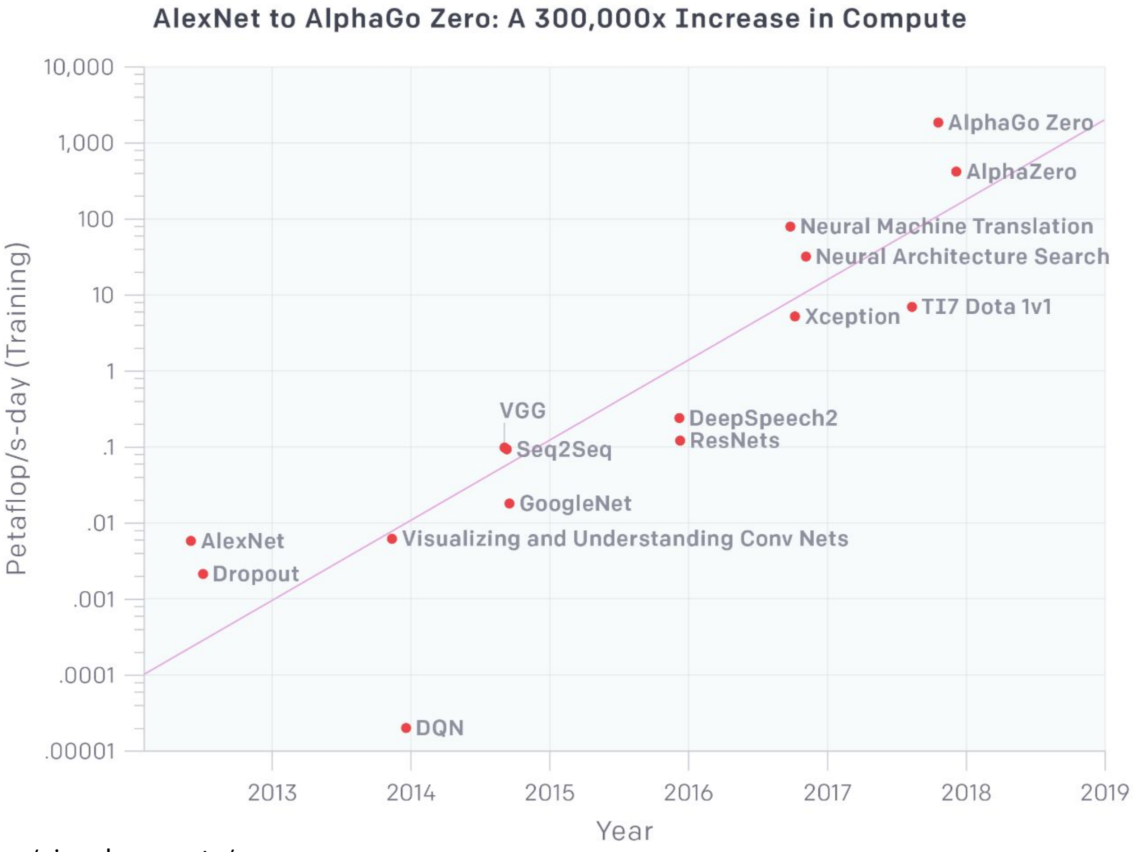


Energy consumption

Exascale computing project has its own power plant

Power density can melt chips

But Requirements for Compute Continue to Grow



Source: https://blog.openai.com/ai-and-compute/

And there's more we want to do

Simulation Driven Drug Design

Organic Batteries & Solar Cells

Artificial General Intelligence

Why build a quantum computer?

New power | New opportunity | Fundamental curiosity

Quantum computing power* scales exponentially with qubits

N bits can exactly simulate **log N qubits**

This compute unit....			
	Commodore 64	AWS M4 Instance	Entire Global Cloud
		1 Million x Commodore 64	1 Billion x (1 Million x Commodore 64)
can exactly simulate:	**10 Qubits**	**30 Qubits**	**60 Qubits**

Size of today's systems.
Note these are *imperfect qubits*.

* We will be more precise later in the lecture

Why build a quantum computer?

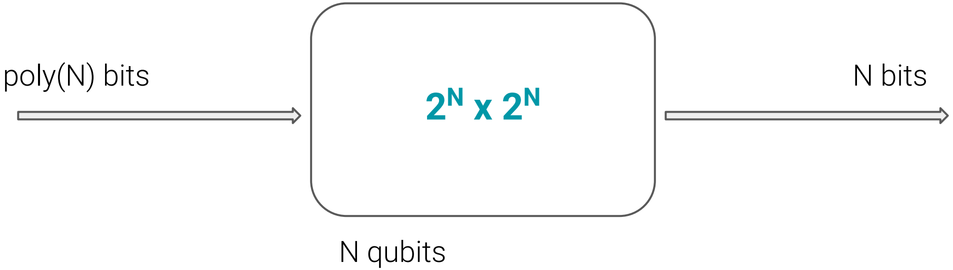
New power | New opportunity | Fundamental curiosity

For **N qubits** every time step (~100ns*) is an exponentially large **2^N x 2^N** complex **matrix multiplication**

Crucial details:

- limited number of multiplications (hundreds to thousands) due to noise
- not arbitrary matrices (need to be easily constructed on a QC)
- small I/O, **N-bits in and N-bits out**

The "big-memory small pipe" mental model for quantum computing



* for superconducting qubit systems

Why build a quantum computer?

New power | New opportunity | Fundamental curiosity

Machine Learning

> Development of new training sets and algorithms

> Classification and sampling of large data sets

Supply Chain Optimization

> Forecast and optimize for future inventory demand

> NP-hard scheduling and logistics map into quantum applications

Robotic Manufacturing

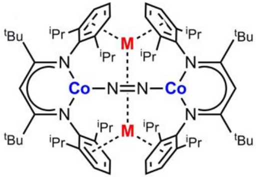
> Reduce manufacturing time and cost

> Maps to a Traveling Salesman Problem addressable by quantum constrained optimization

Computational Materials Science

> Design of better catalysts for batteries

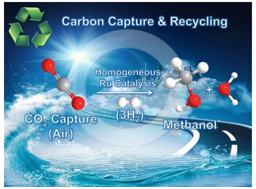
> Quantum algorithms for calculating electronic structure



Alternative Energy Research

> Efficiently convert atmospheric CO_2 to methanol

> Powered by existing hybrid quantum-classical algorithms + machine learning



What isn't on here: breaking RSA with Shor's algorithm

Quantum hardware development is accelerating

Plotted by number of qubits in development

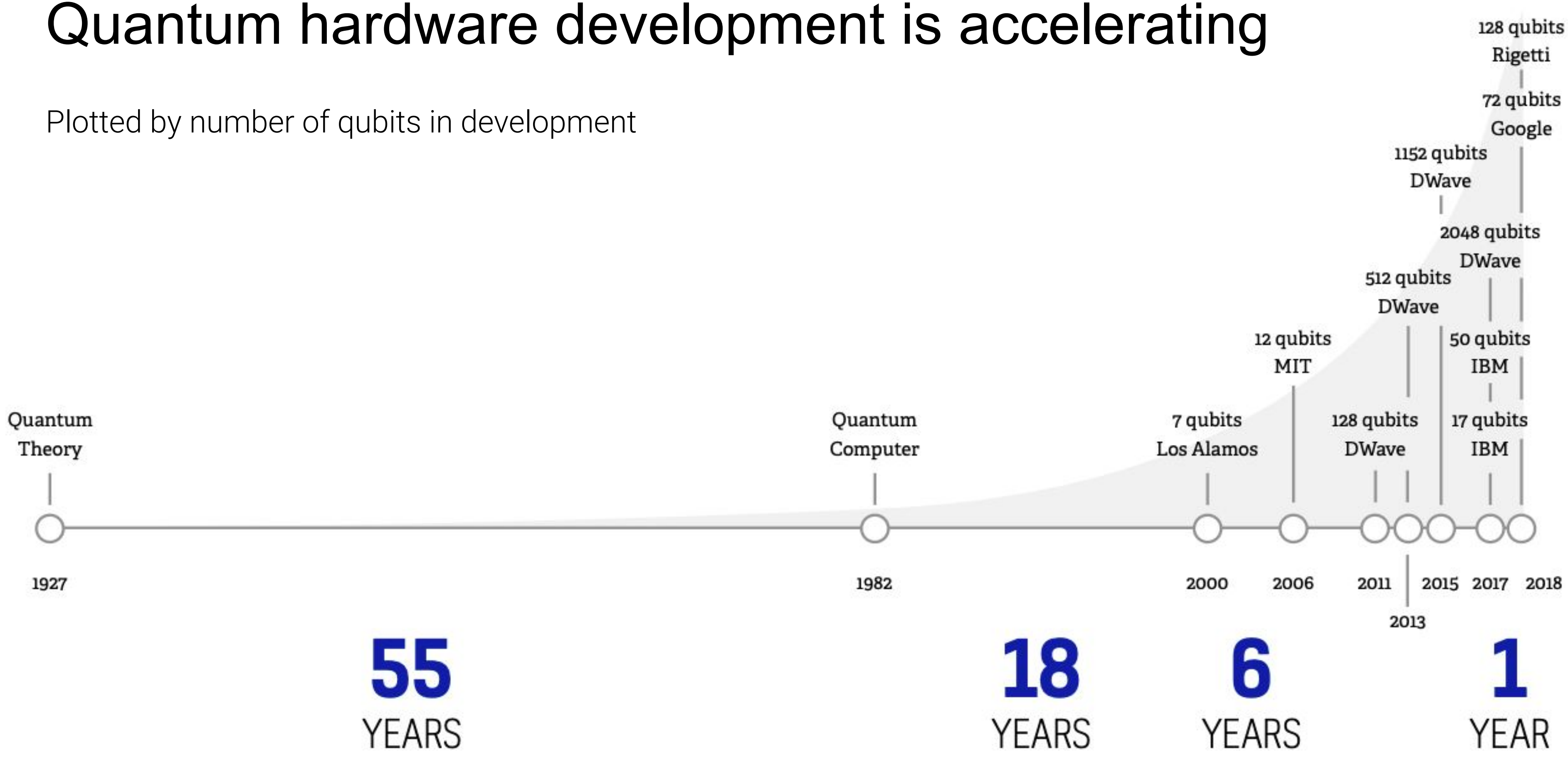


Image: Strangeworks

Quantum Hardware comes in many forms

Photonic Quantum Computers Use Light

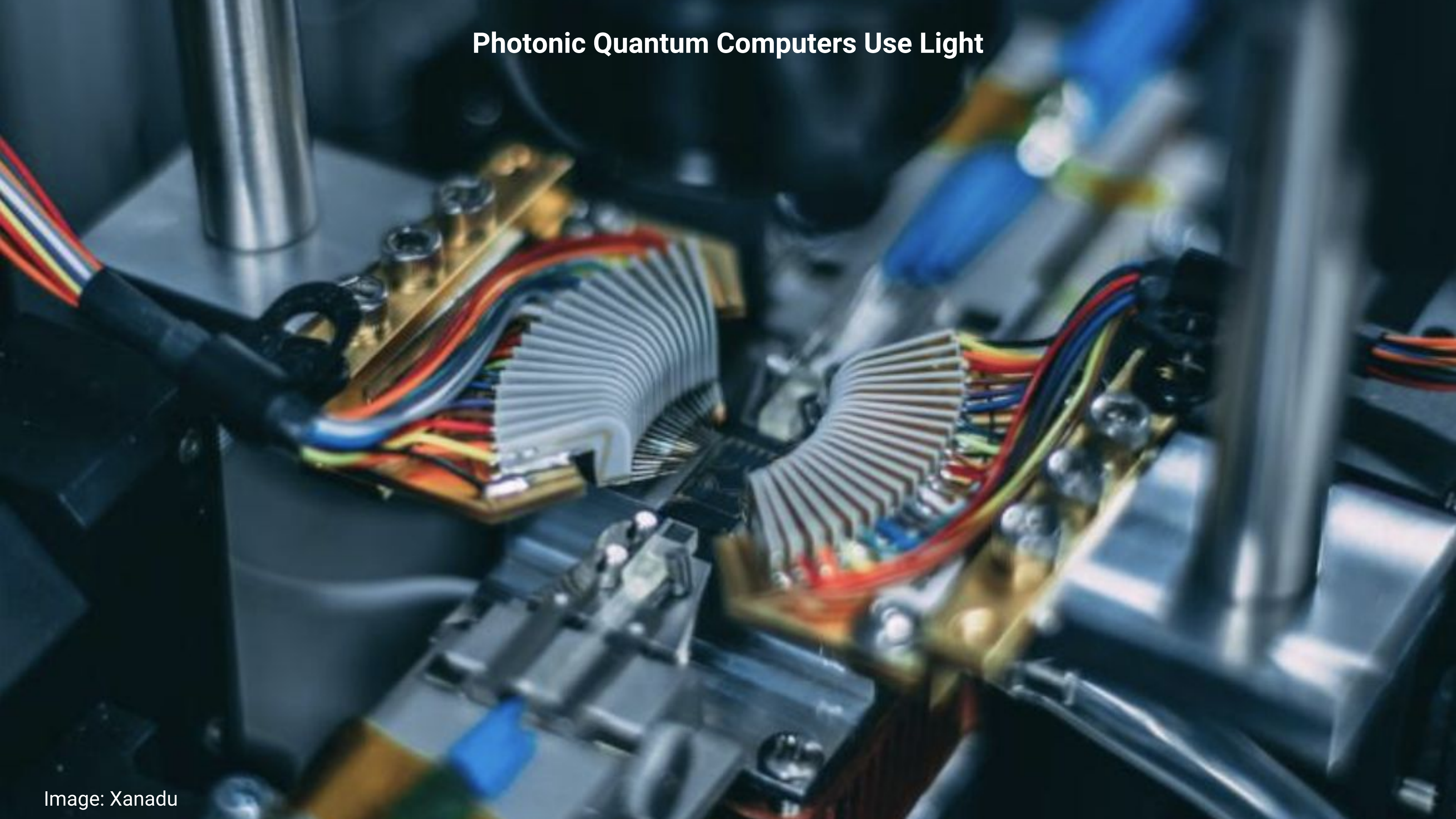
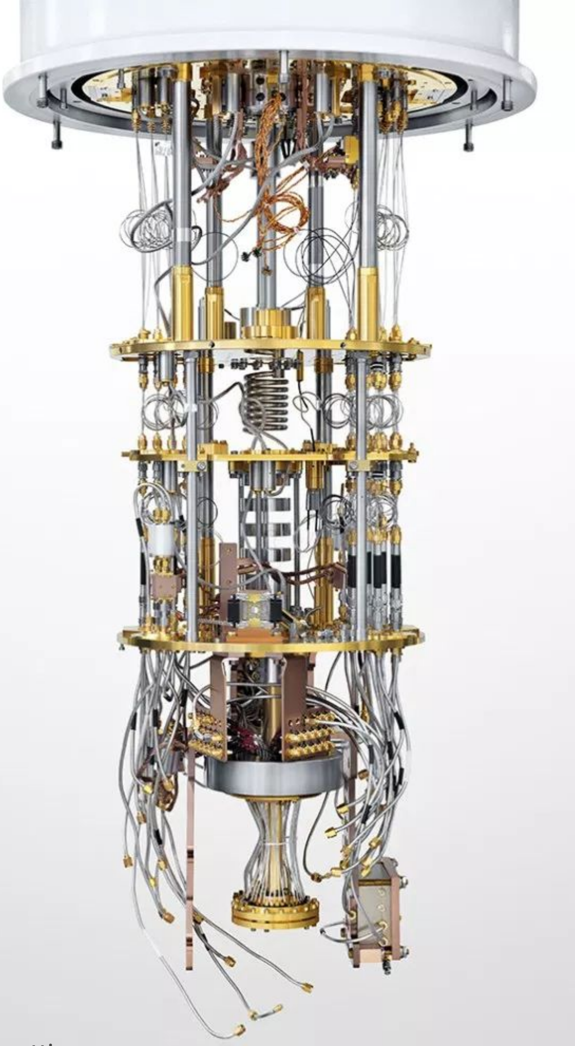


Image: Xanadu

Superconducting Qubits are Supercooled RF Circuits

Image: IBM



Images: Rigetti

Ion Trap Quantum Computers

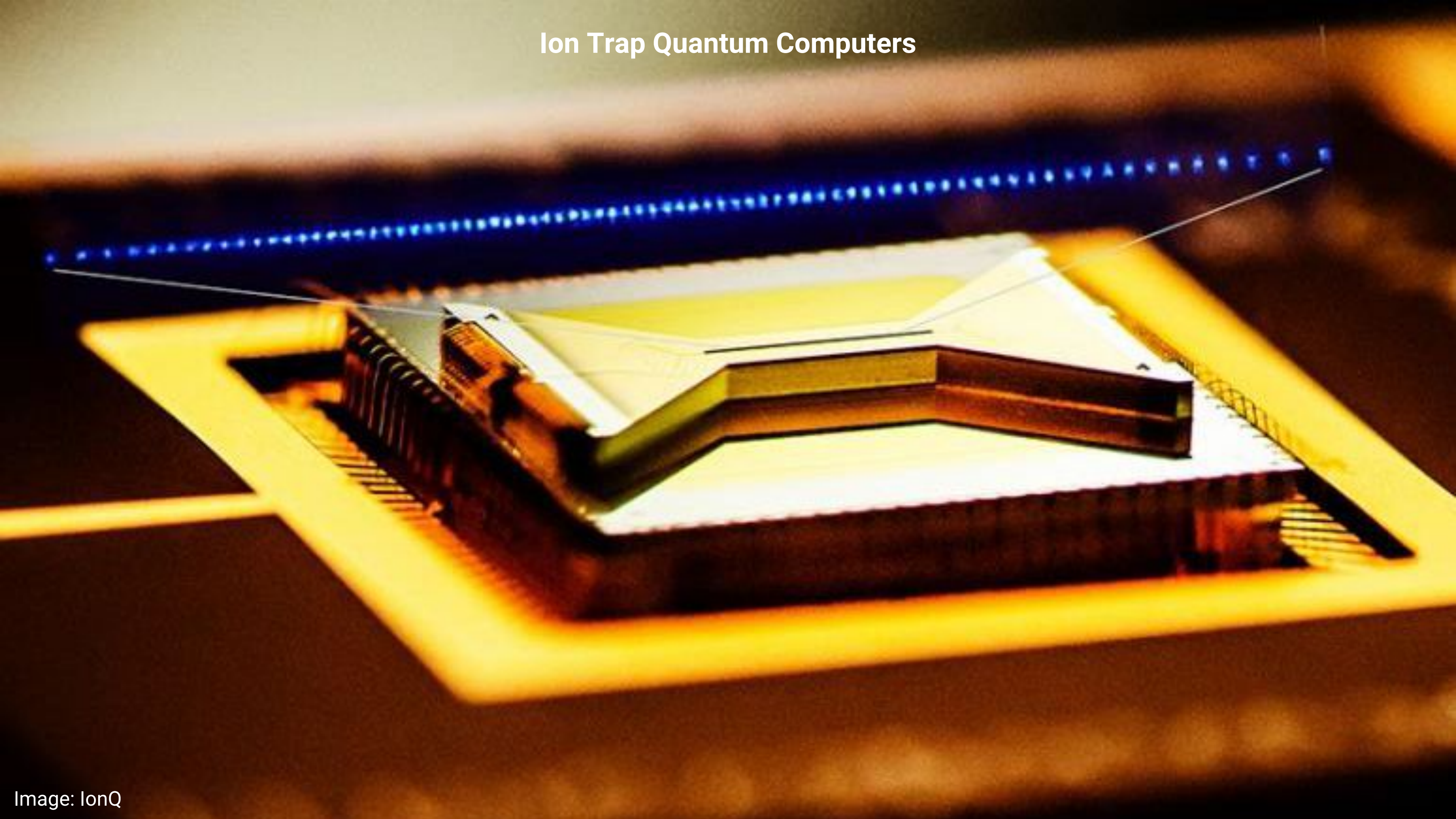
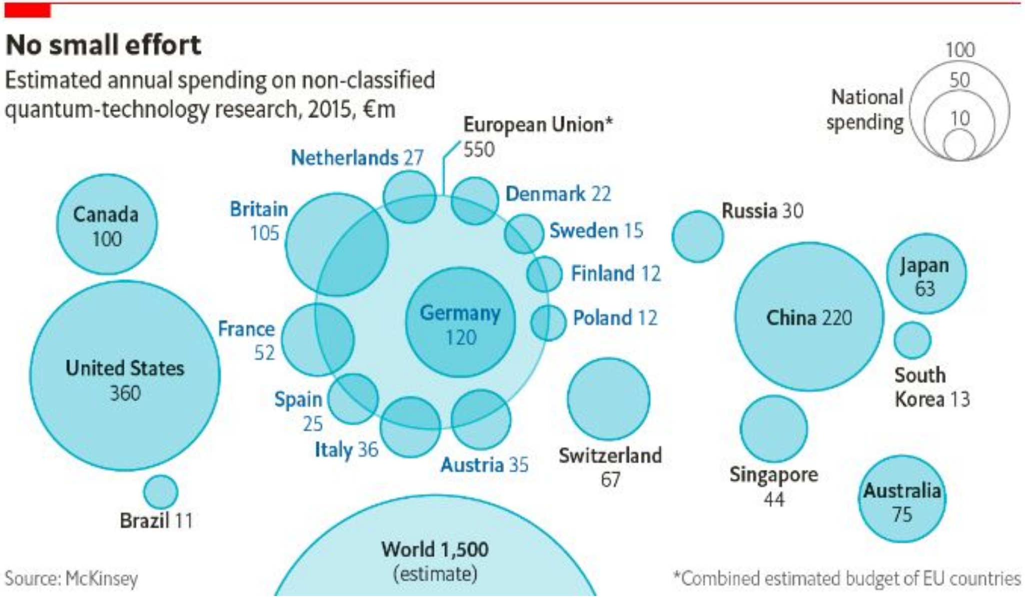


Image: IonQ

Why build a quantum computer?

New power | **New opportunity** | Fundamental curiosity

Investments across academia, government, and industry are global and growing



No small effort

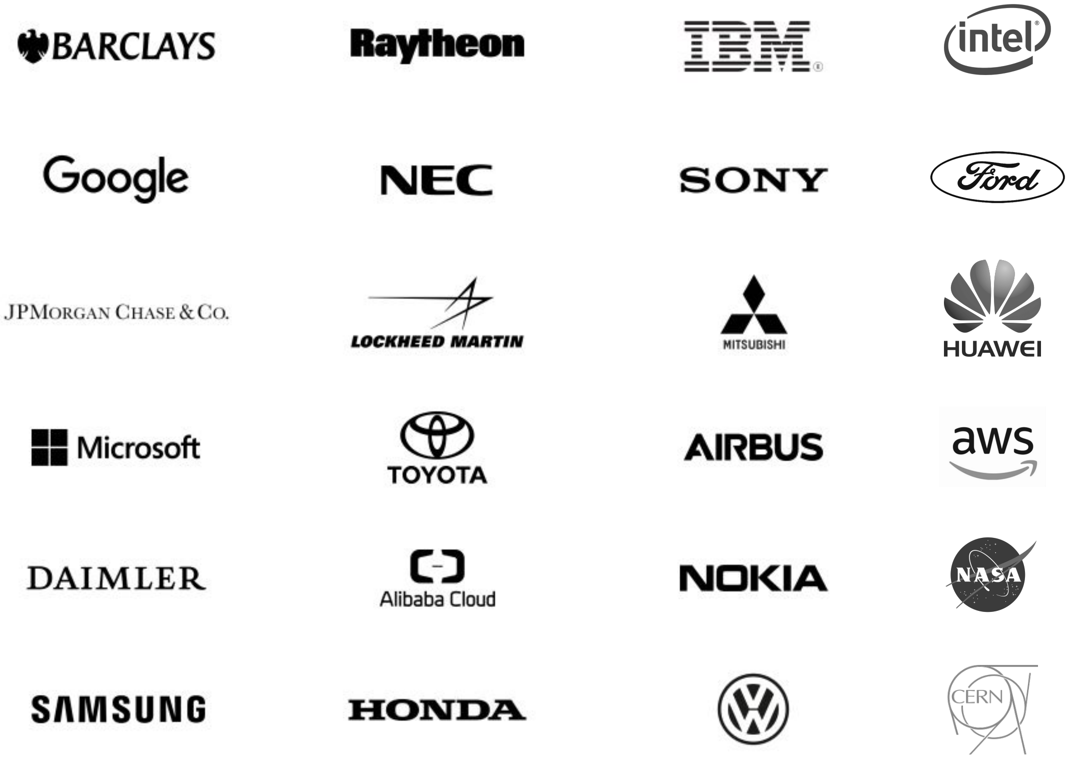
Estimated annual spending on non-classified quantum-technology research, 2015, €m

Source: McKinsey

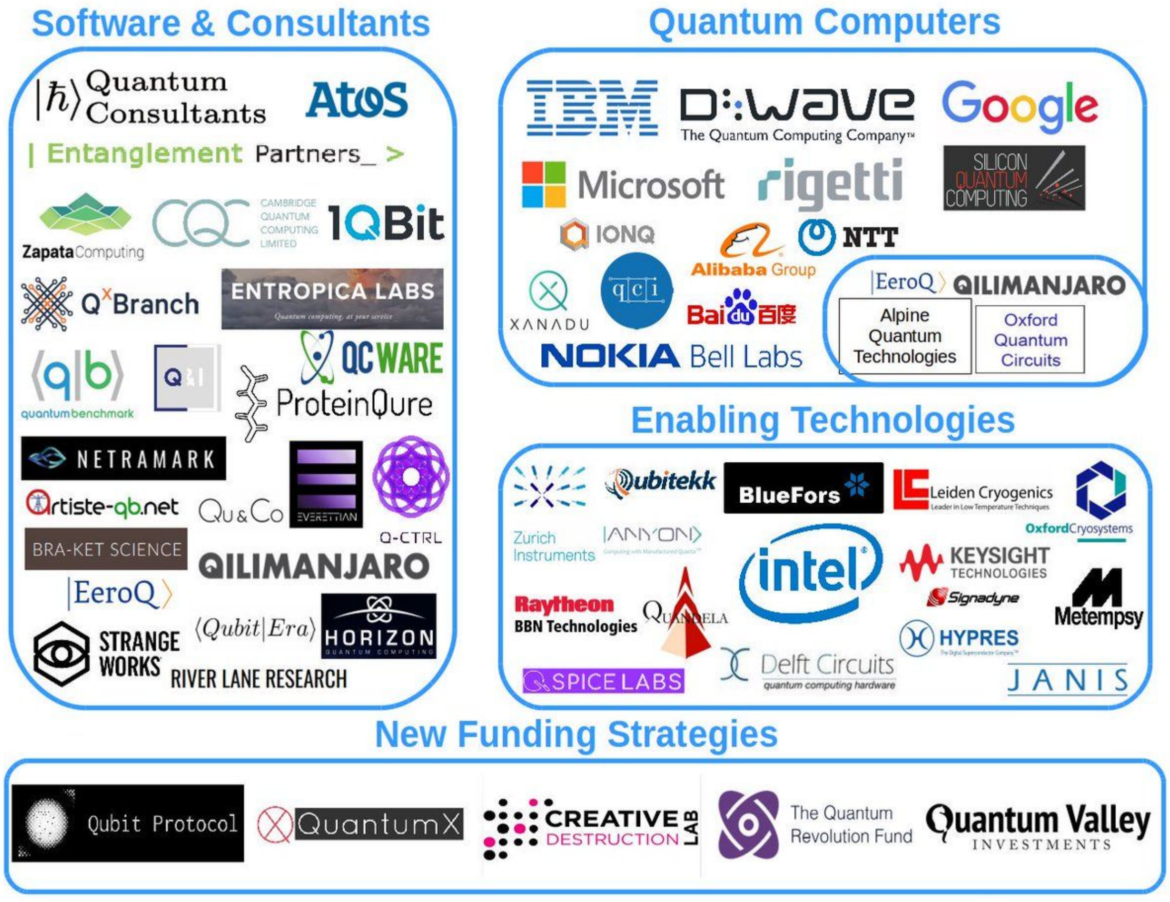
*Combined estimated budget of EU countries

Plus approximately $400M in global VC investment

Large Companies are involved



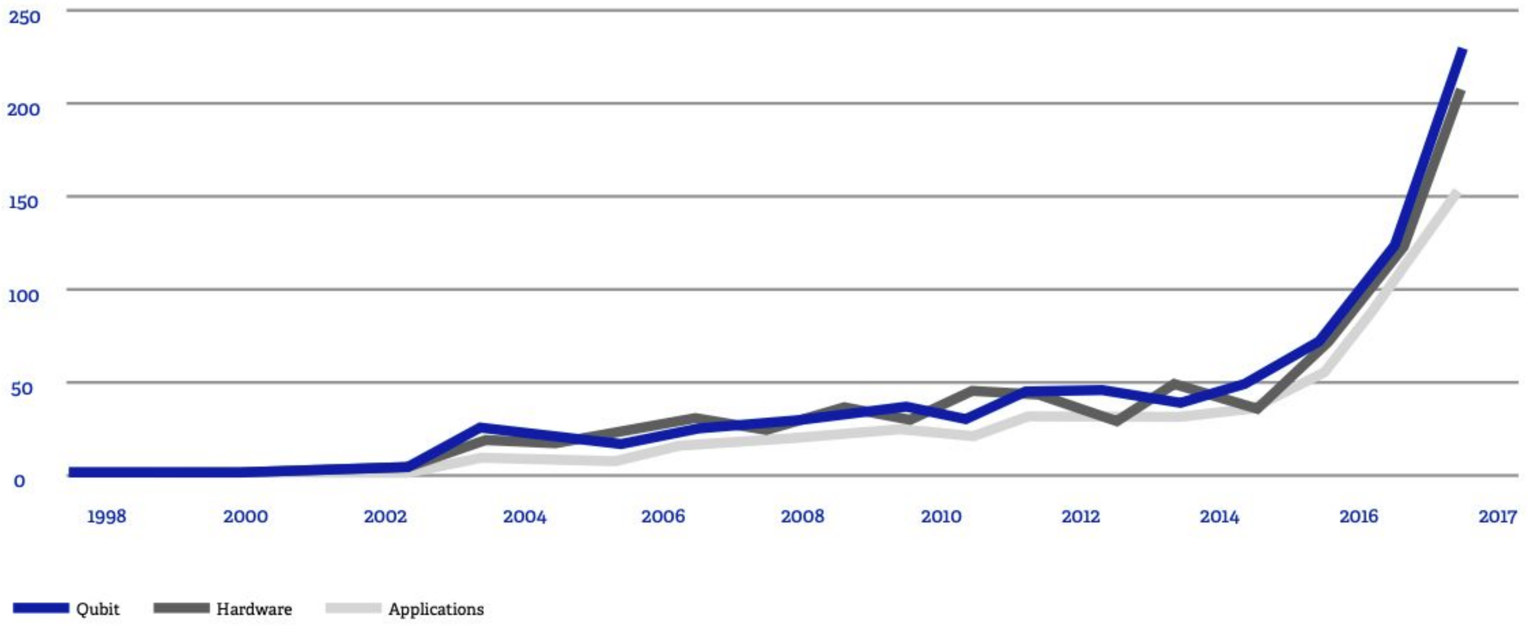
In a growing ecosystem of startups and incumbents



Representative list of players. A very active ecosystem!

QUANTUM WORLD ASSOCIATION

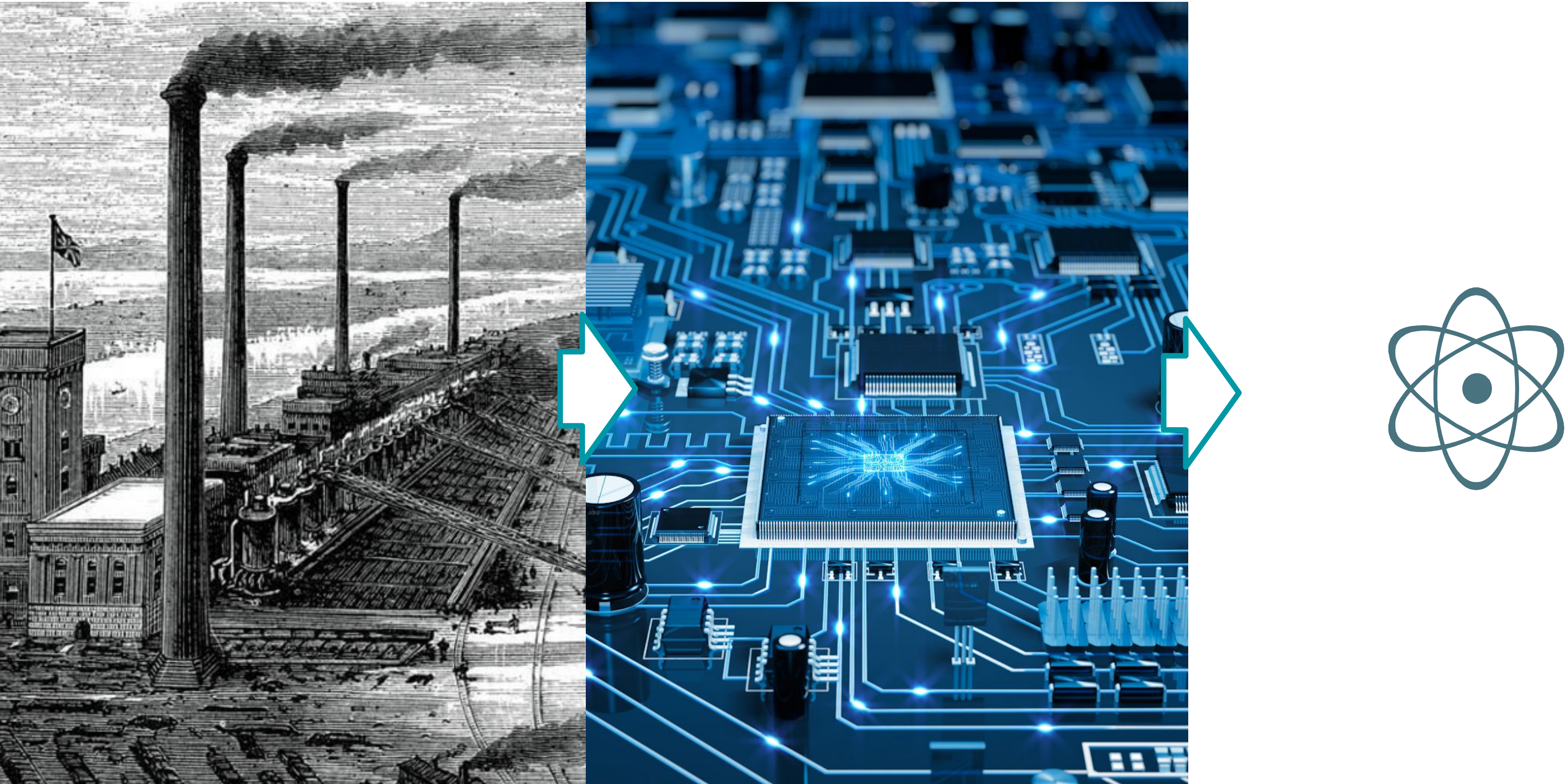
QUANTUM COMPUTING PATENT FAMILIES BY CATEGORY AND PUBLICATION YEAR



Based on 1,952 Quantum Computing patent documents from a worldwide search in Thomson Innovation; limited to one document per family, based of DWPI with US as primary country; documents can appear in more than one category; currently 293 documents for 2017.

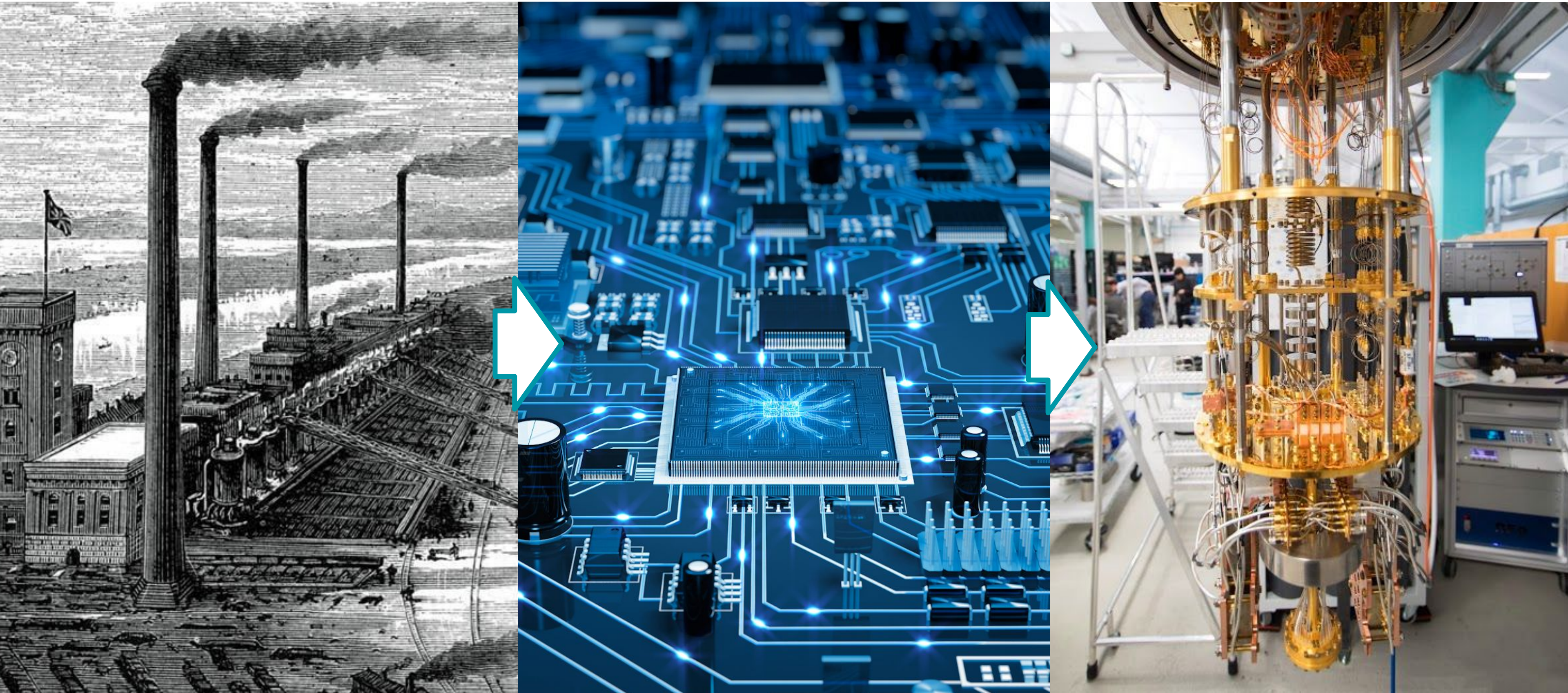
Why program a quantum computer?

New power | New opportunity | **Fundamental curiosity**



Why program a quantum computer?

New power | New opportunity | **Fundamental curiosity**



Why program a quantum computer?

New power | New opportunity | **Fundamental curiosity**

Quantum computing reorients the relationship between physics and computer science.

Every “function which would ***naturally*** *be regarded as computable” can be computed by the universal Turing machine.* - Turing

“... ***nature*** *isn't classical, dammit...”* - Feynman

Why program a quantum computer?

New power | New opportunity | **Fundamental curiosity**

Quantum computing reorients the relationship between physics and computer science.

Every "function which would ***naturally*** *be regarded as computable" can be computed by the universal Turing machine.* - Turing

"... ***nature*** *isn't classical, dammit..."* - Feynman

Physical phenomenon apply to information and computation as well.

> Superposition

> No-cloning

> Teleportation

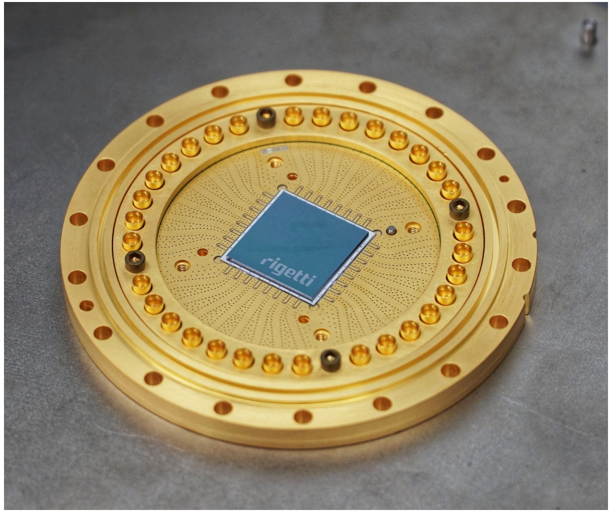
How do I program a quantum computer?

Hybrid Quantum Computers | Quantum Programming | Hybrid Programming | Hybrid Algorithms

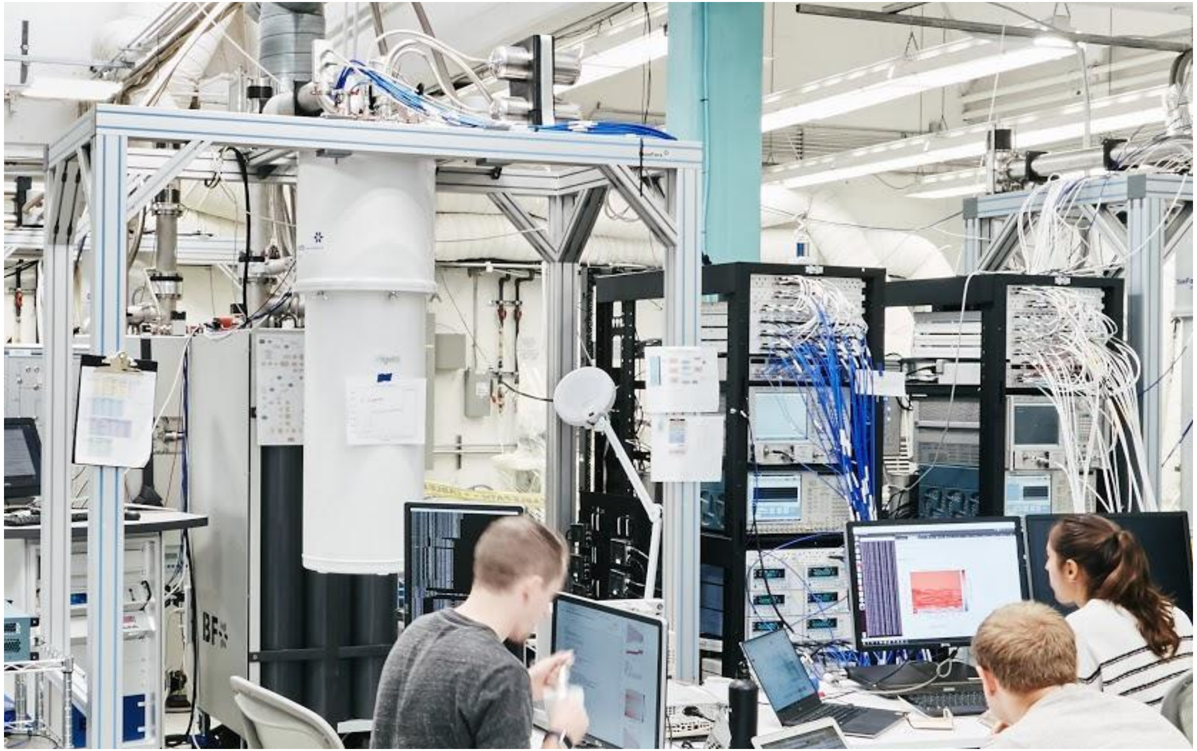
How do I program a quantum computer?

Hybrid Quantum Computers | Quantum Programming | Hybrid Programming | Hybrid Algorithms

Quantum computers have quantum processor(s) and classical processors



Quantum processor



Full quantum computing system

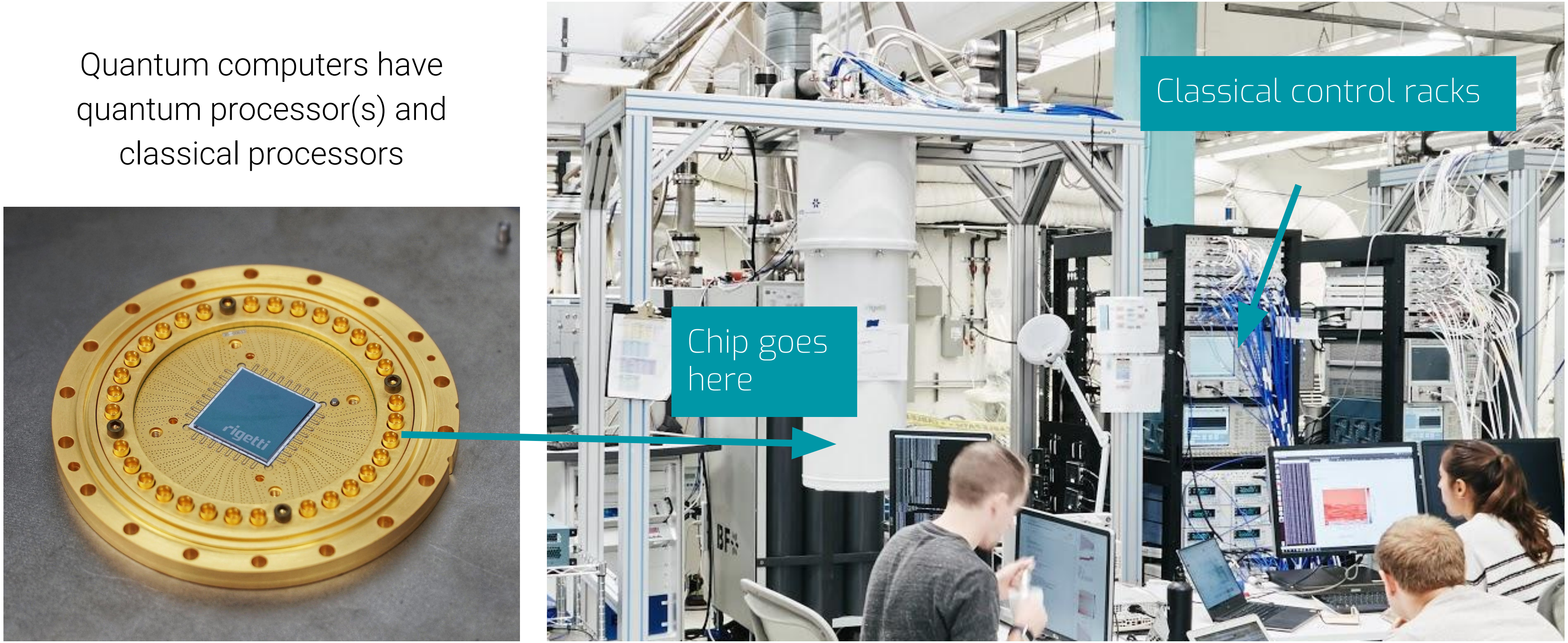
Otterbach et al. arXiv:1712.05771

Images: Rigetti

How do I program a quantum computer?

Hybrid Quantum Computers | Quantum Programming | Hybrid Programming | Hybrid Algorithms

Quantum computers have quantum processor(s) and classical processors



Quantum processor

Full quantum computing system

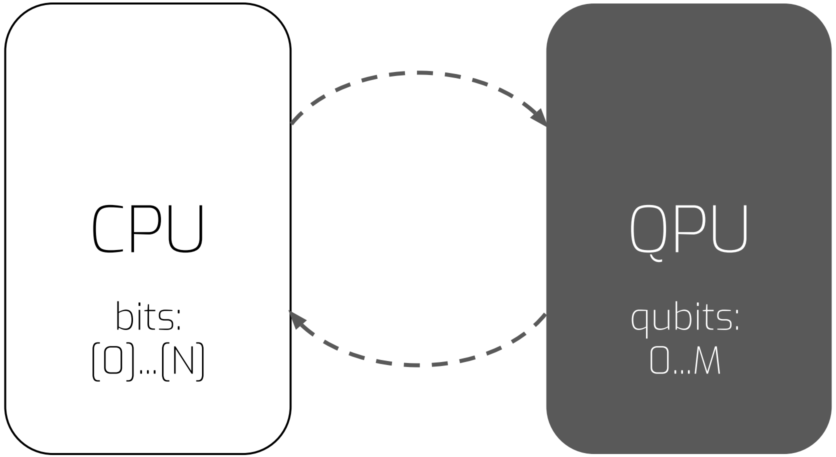
Otterbach et al. arXiv:1712.05771

Images: Rigetti

How do I program a quantum computer?

Hybrid Quantum Computers | Quantum Programming | Hybrid Programming | Hybrid Algorithms

Practical, valuable quantum computing is **Hybrid** Quantum/Classical Computing

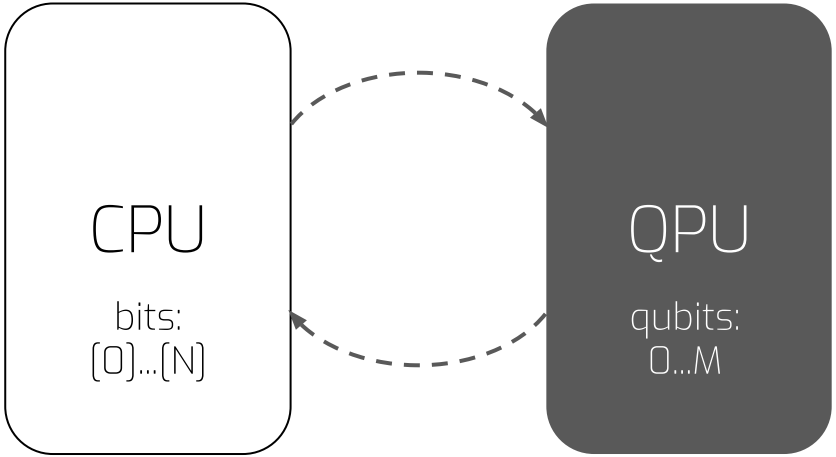


Smith, Curtis, Zeng. “A Practical Quantum Instruction Set Architecture” arXiv:1608.03355

How do I program a quantum computer?

Hybrid Quantum Computers | Quantum Programming | Hybrid Programming | Hybrid Algorithms

Practical, valuable quantum computing is **Hybrid** Quantum/Classical Computing

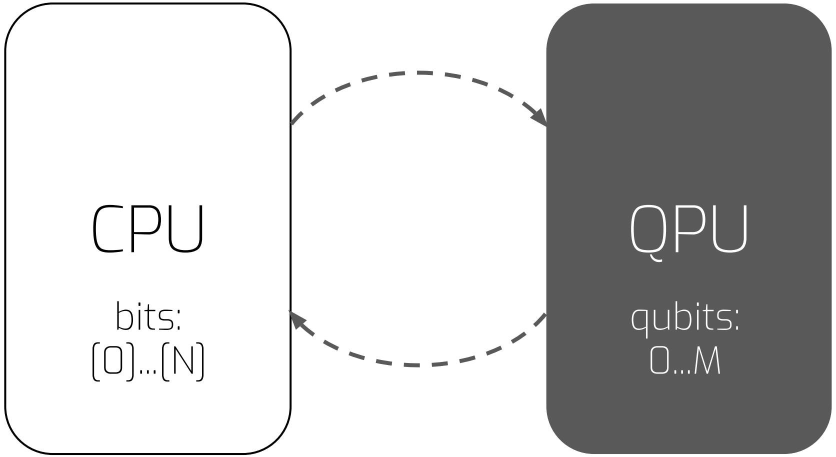


Smith, Curtis, Zeng. “A Practical Quantum Instruction Set Architecture” arXiv:1608.03355

How do I program a quantum computer?

Hybrid Quantum Computers | Quantum Programming | Hybrid Programming | Hybrid Algorithms

Practical, valuable quantum computing is **Hybrid** Quantum/Classical Computing



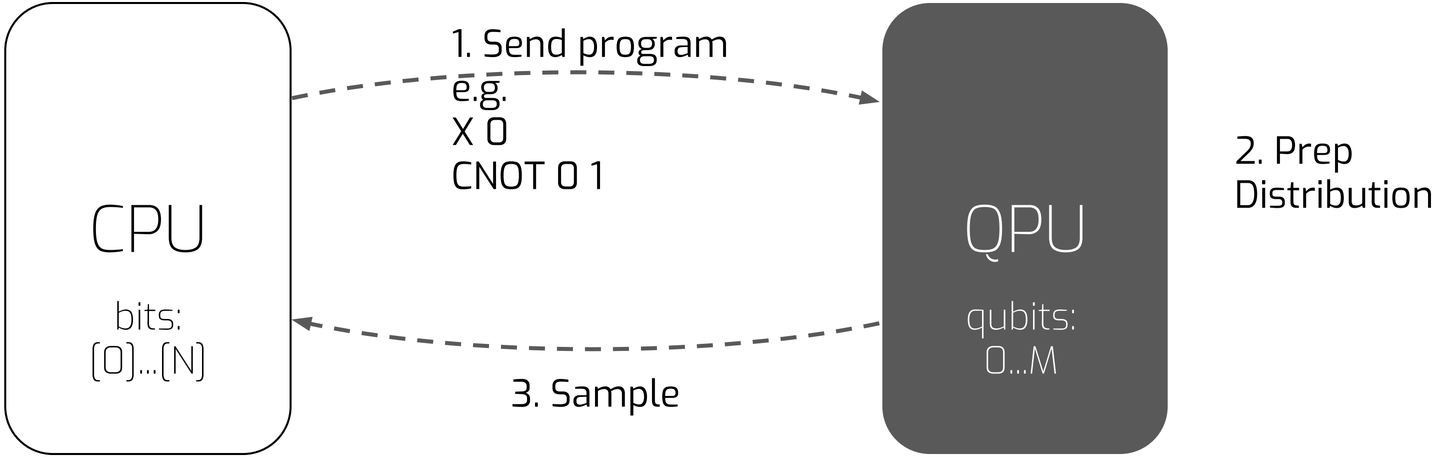
The Quil [**01**⟩ instruction set is optimized for this.

Smith, Curtis, Zeng. "A Practical Quantum Instruction Set Architecture" arXiv:1608.03355

How do I program a quantum computer?

Hybrid Quantum Computers | **Quantum Programming** | Hybrid Programming | Hybrid Algorithms

Quantum programming is preparing and sampling from complicated distributions



1. Send program
e.g.
X 0
CNOT 0 1

2. Prep
Distribution

3. Sample

How do I program a quantum computer?

	Bits	Probabilistic Bits	Qubits
State (single unit)	Bit $\in \{0, 1\}$	Real vector $a + b \in \mathbb{R}_+$ $\vec{b} = a\vec{0} + b\vec{1}$ $a + b = 1$	

How do I program a quantum computer?

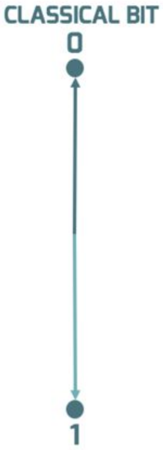
	Bits	Probabilistic Bits	Qubits
State (single unit)	Bit $\in \{0, 1\}$	Real vector $a + b \in \mathbb{R}_+$ $\vec{b} = a\vec{0} + b\vec{1}$ $a + b = 1$	

Probability of 0

Probability of 1

How do I program a quantum computer?

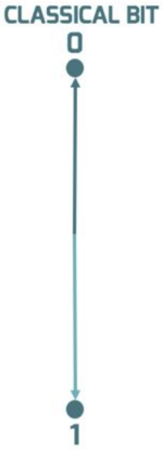
	Bits	Probabilistic Bits	Qubits
State (single unit)	Bit $\in \{0, 1\}$	Real vector $a + b \in \mathbb{R}_+$ $\vec{b} = a\vec{0} + b\vec{1}$ $a + b = 1$	



How do I program a quantum computer?

Hybrid Quantum Computers | **Quantum Programming** | Hybrid Programming | Hybrid Algorithms

	Bits	Probabilistic Bits	Qubits
State (single unit)	Bit $\in \{0, 1\}$	Real vector $a + b \in \mathbb{R}_+$ $\vec{b} = a\vec{0} + b\vec{1}$ $a + b = 1$	Complex vector $\alpha, \beta \in \mathbb{C}$ $\vec{\psi} = \alpha\vec{0} + \beta\vec{1}$ $\lvert\alpha\rvert^2 + \lvert\beta\rvert^2 = 1$



How do I program a quantum computer?

	Bits	Probabilistic Bits	Qubits
State (single unit)	Bit $\in \{0, 1\}$	Real vector $a + b \in \mathbb{R}_+$ $\vec{b} = a\vec{0} + b\vec{1}$ $a + b = 1$	Complex vector $\alpha, \beta \in \mathbb{C}$ $\vec{\psi} = \alpha\vec{0} + \beta\vec{1}$ $\lvert\alpha\rvert^2 + \lvert\beta\rvert^2 = 1$

CLASSICAL BIT
0
1

$|\alpha|^2$ = Probability of 0

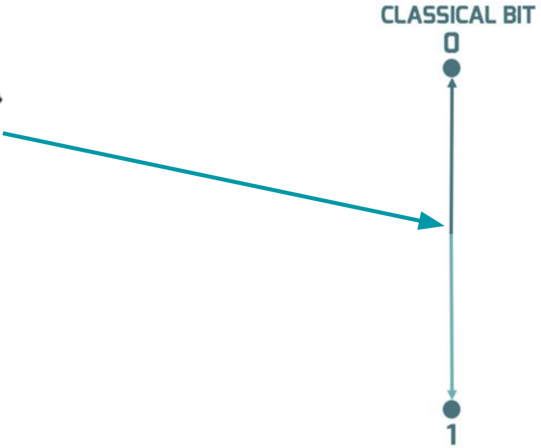
$|\beta|^2$ = Probability of 1

How do I program a quantum computer?

Hybrid Quantum Computers | **Quantum Programming** | Hybrid Programming | Hybrid Algorithms

	Bits	Probabilistic Bits	Qubits
State (single unit)	Bit $\in \{0, 1\}$	Real vector $a + b \in \mathbb{R}_+$ $\vec{b} = a\vec{0} + b\vec{1}$ $a + b = 1$	Complex vector $\alpha, \beta \in \mathbb{C}$ $\vec{\psi} = \alpha\vec{0} + \beta\vec{1}$ $\|\alpha\|^2 + \|\beta\|^2 = 1$

$$\vec{\text{coin}} = \frac{1}{2}\vec{0} + \frac{1}{2}\vec{1}$$



How do I program a quantum computer?

	Bits	Probabilistic Bits	Qubits
State (single unit)	Bit $\in \{0, 1\}$	Real vector $a + b \in \mathbb{R}_+$ $\vec{b} = a\vec{0} + b\vec{1}$ $a + b = 1$	Complex vector $\alpha, \beta \in \mathbb{C}$ $\vec{\psi} = \alpha\vec{0} + \beta\vec{1}$ $\|\alpha\|^2 + \|\beta\|^2 = 1$

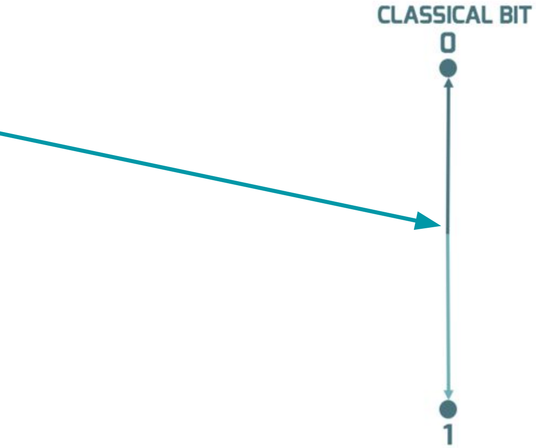
$$\vec{\text{coin}} = \frac{1}{2}\vec{0} + \frac{1}{2}\vec{1}$$

$$\text{qcoin} = \frac{1}{\sqrt{2}}\vec{0} + \frac{1}{\sqrt{2}}\vec{1}$$

$$\text{qcoin} = \frac{1}{\sqrt{2}}\vec{0} - \frac{1}{\sqrt{2}}\vec{1}$$

$$\text{qcoin} = \frac{1}{\sqrt{2}}\vec{0} - \frac{i}{\sqrt{2}}\vec{1}$$

...

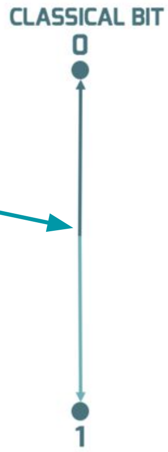


How do I program a quantum computer?

	Bits	Probabilistic Bits	Qubits
State (single unit)	Bit $\in \{0, 1\}$	Real vector $a + b \in \mathbb{R}_+$ $\vec{b} = a\vec{0} + b\vec{1}$ $a + b = 1$	Complex vector $\alpha, \beta \in \mathbb{C}$ $\vec{\psi} = \alpha\vec{0} + \beta\vec{1}$ $\lvert\alpha\rvert^2 + \lvert\beta\rvert^2 = 1$

$$\vec{\text{coin}} = \frac{1}{2}\vec{0} + \frac{1}{2}\vec{1}$$

$$\vec{\text{qcoin}}(\theta) = \frac{1}{\sqrt{2}}\vec{0} + \frac{e^{i\theta}}{\sqrt{2}}\vec{1}$$

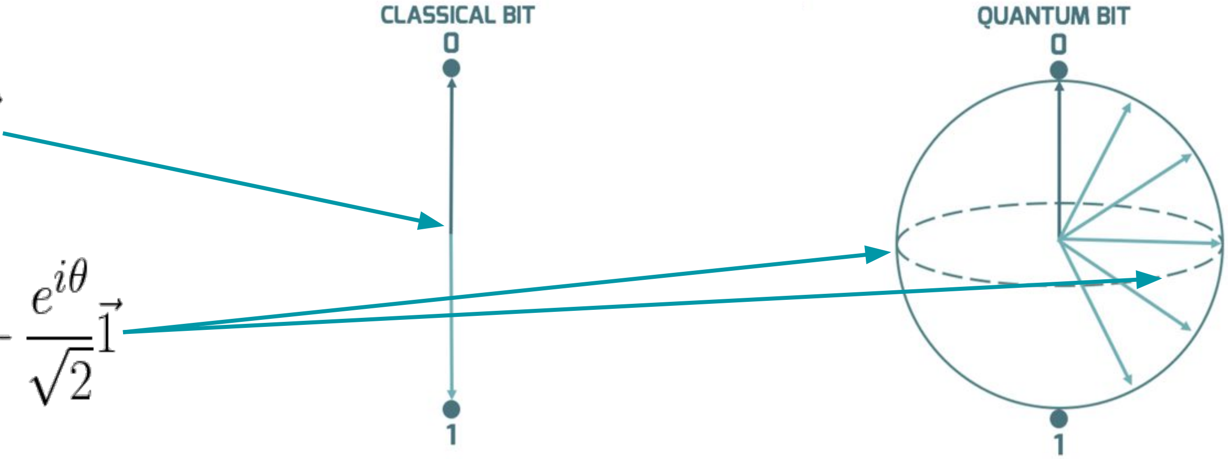


How do I program a quantum computer?

	Bits	Probabilistic Bits	Qubits
State (single unit)	Bit $\in \{0, 1\}$	Real vector $a + b \in \mathbb{R}_+$ $\vec{b} = a\vec{0} + b\vec{1}$ $a + b = 1$	Complex vector $\alpha, \beta \in \mathbb{C}$ $\vec{\psi} = \alpha\vec{0} + \beta\vec{1}$ $\lvert\alpha\rvert^2 + \lvert\beta\rvert^2 = 1$

$$\vec{\text{coin}} = \frac{1}{2}\vec{0} + \frac{1}{2}\vec{1}$$

$$\vec{\text{qcoin}}(\theta) = \frac{1}{\sqrt{2}}\vec{0} + \frac{e^{i\theta}}{\sqrt{2}}\vec{1}$$



How do I program a quantum computer?

	Bits	Probabilistic Bits	Qubits
State (single unit)	Bit $\in \{0, 1\}$	Real vector $a + b \in \mathbb{R}_+$ $\vec{b} = a\vec{0} + b\vec{1}$ $a + b = 1$	Complex vector $\alpha, \beta \in \mathbb{C}$ $\vec{\psi} = \alpha\vec{0} + \beta\vec{1}$ $\lvert\alpha\rvert^2 + \lvert\beta\rvert^2 = 1$
State (multi-unit)	Bitstring $x \in \{0, 1\}^n$	Prob. Distribution (stochastic vector) $\vec{s} = \{p_x\}_{x \in \{0,1\}^n}$	

$$\vec{s} = \bigotimes_{i}^{n} b_i$$

Probability of bitstring x

How do I program a quantum computer?

Hybrid Quantum Computers | **Quantum Programming** | Hybrid Programming | Hybrid Algorithms

	Bits	Probabilistic Bits	Qubits
State (single unit)	Bit $\in \{0, 1\}$	Real vector $a + b \in \mathbb{R}_+$ $\vec{b} = a\vec{0} + b\vec{1}$ $a + b = 1$	Complex vector $\alpha, \beta \in \mathbb{C}$ $\vec{\psi} = \alpha\vec{0} + \beta\vec{1}$ $\lvert\alpha\rvert^2 + \lvert\beta\rvert^2 = 1$
State (multi-unit)	Bitstring $x \in \{0, 1\}^n$	Prob. Distribution (stochastic vector) $\vec{s} = \{p_x\}_{x \in \{0,1\}^n}$	Wavefunction (complex vector) $\vec{\psi} = \{\alpha_x\}_{x \in \{0,1\}^n}$

$$\vec{s} = \bigotimes_{i}^{n} b_i$$

$$\vec{\psi} = \bigotimes_{i}^{n} \psi_i$$

How do I program a quantum computer?

Hybrid Quantum Computers | **Quantum Programming** | Hybrid Programming | Hybrid Algorithms

	Bits	Probabilistic Bits	Qubits
State (single unit)	Bit $\in \{0, 1\}$	Real vector $a + b \in \mathbb{R}_+$ $\vec{b} = a\vec{0} + b\vec{1}$ $a + b = 1$	Complex vector $\alpha, \beta \in \mathbb{C}$ $\vec{\psi} = \alpha\vec{0} + \beta\vec{1}$ $\lvert\alpha\rvert^2 + \lvert\beta\rvert^2 = 1$
State (multi-unit)	Bitstring $x \in \{0, 1\}^n$	Prob. Distribution (stochastic vector) $\vec{s} = \{p_x\}_{x \in \{0,1\}^n}$	Wavefunction (complex vector) $\vec{\psi} = \{\alpha_x\}_{x \in \{0,1\}^n}$

$$\vec{s} = \bigotimes_i^n b_i$$

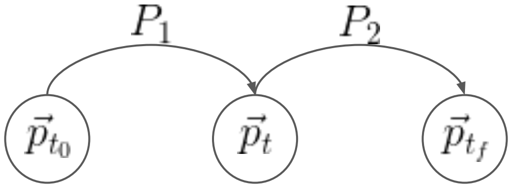
$$\vec{\psi} = \bigotimes_i^n \psi_i$$

$|\alpha_x|^2$ = Probability of bitstring x

How do I program a quantum computer?

Hybrid Quantum Computers | **Quantum Programming** | Hybrid Programming | Hybrid Algorithms

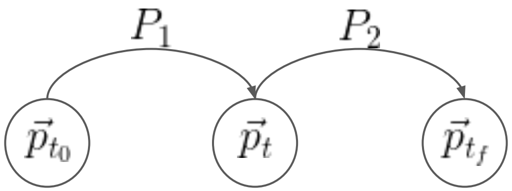
	Bits	Probabilistic Bits	Qubits
State (single unit)	Bit $\in \{0,1\}$	Real vector $a + b \in \mathbb{R}_+$ $\vec{b} = a\vec{0} + b\vec{1}$ $a + b = 1$	Complex vector $\alpha, \beta \in \mathbb{C}$ $\vec{\psi} = \alpha\vec{0} + \beta\vec{1}$ $\lvert\alpha\rvert^2 + \lvert\beta\rvert^2 = 1$
State (multi-unit)	Bitstring $x \in \{0,1\}^n$	Prob. Distribution (stochastic vector) $\vec{s} = \{p_x\}_{x \in \{0,1\}^n}$	Wavefunction (complex vector) $\vec{\psi} = \{\alpha_x\}_{x \in \{0,1\}^n}$
Operations	Boolean Logic	Stochastic Matrices $\sum_{j=1}^{S} P_{i,j} = 1.$	

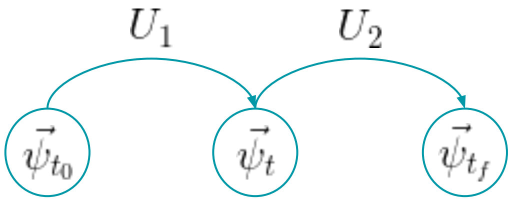


How do I program a quantum computer?

Hybrid Quantum Computers | **Quantum Programming** | Hybrid Programming | Hybrid Algorithms

	Bits	Probabilistic Bits	Qubits
State (single unit)	Bit $\in \{0, 1\}$	Real vector $a + b \in \mathbb{R}_+$ $\vec{b} = a\vec{0} + b\vec{1}$ $a + b = 1$	Complex vector $\alpha, \beta \in \mathbb{C}$ $\vec{\psi} = \alpha\vec{0} + \beta\vec{1}$ $\lvert\alpha\rvert^2 + \lvert\beta\rvert^2 = 1$
State (multi-unit)	Bitstring $x \in \{0, 1\}^n$	Prob. Distribution (stochastic vector) $\vec{s} = \{p_x\}_{x\in\{0,1\}^n}$	Wavefunction (complex vector) $\vec{\psi} = \{\alpha_x\}_{x\in\{0,1\}^n}$
Operations	Boolean Logic	Stochastic Matrices $\sum_{j=1}^{S} P_{i,j} = 1.$	Unitary Matrices $U^\dagger U = 1$





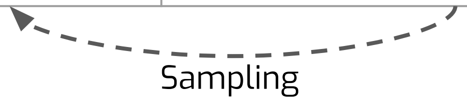
How do I program a quantum computer?

Hybrid Quantum Computers | **Quantum Programming** | Hybrid Programming | Hybrid Algorithms

	Bits	Probabilistic Bits	Qubits
State (single unit)	Bit $\in \{0, 1\}$	Real vector $a + b \in \mathbb{R}_+$ $\vec{b} = a\vec{0} + b\vec{1}$ $a + b = 1$	Complex vector $\alpha, \beta \in \mathbb{C}$ $\vec{\psi} = \alpha\vec{0} + \beta\vec{1}$ $\lvert\alpha\rvert^2 + \lvert\beta\rvert^2 = 1$
State (multi-unit)	Bitstring $x \in \{0, 1\}^n$	Prob. Distribution (stochastic vector) $\vec{s} = \{p_x\}_{x\in\{0,1\}^n}$	Wavefunction (complex vector) $\vec{\psi} = \{\alpha_x\}_{x\in\{0,1\}^n}$
Operations	Boolean Logic	Stochastic Matrices $\sum_{j=1}^{S} P_{i,j} = 1.$	Unitary Matrices $U^\dagger U = 1$
Component Ops	Boolean Gates	Tensor products of matrices	Tensor products of matrices

How do I program a quantum computer?

	Bits	Probabilistic Bits	Qubits
State (single unit)	Bit $\in \{0, 1\}$	Real vector $a + b \in \mathbb{R}_+$ $\vec{b} = a\vec{0} + b\vec{1}$ $a + b = 1$	Complex vector $\alpha, \beta \in \mathbb{C}$ $\vec{\psi} = \alpha\vec{0} + \beta\vec{1}$ $\|\alpha\|^2 + \|\beta\|^2 = 1$
State (multi-unit)	Bitstring $x \in \{0, 1\}^n$	Prob. Distribution (stochastic vector) $\vec{s} = \{p_x\}_{x\in\{0,1\}^n}$	Wavefunction (complex vector) $\vec{\psi} = \{\alpha_x\}_{x\in\{0,1\}^n}$
Operations	Boolean Logic	Stochastic Matrices $\sum_{j=1}^{S} P_{i,j} = 1.$	Unitary Matrices $U^\dagger U = 1$
Component Ops	Boolean Gates	Tensor products of matrices	Tensor products of matrices


Sampling

How do I program a quantum computer?

Hybrid Quantum Computers | **Quantum Programming** | Hybrid Programming | Hybrid Algorithms

	Bits	Probabilistic Bits	Qubits
State (single unit)	Bit $\in \{0, 1\}$	Real vector $a + b \in \mathbb{R}_+$ $\vec{b} = a\vec{0} + b\vec{1}$ $a + b = 1$	Complex vector $\alpha, \beta \in \mathbb{C}$ $\vec{\psi} = \alpha\vec{0} + \beta\vec{1}$ $\lvert\alpha\rvert^2 + \lvert\beta\rvert^2 = 1$
State (multi-unit)	Bitstring $x \in \{0, 1\}^n$	Prob. Distribution (stochastic vector) $\vec{s} = \{p_x\}_{x \in \{0,1\}^n}$	Wavefunction (complex vector) $\vec{\psi} = \{\alpha_x\}_{x \in \{0,1\}^n}$
Operations	Boolean Logic	Stochastic Matrices $\sum_{j=1}^{S} P_{i,j} = 1.$	Unitary Matrices $U^\dagger U = 1$
Component Ops	Boolean Gates	Tensor products of matrices	Tensor products of matrices

Sampling

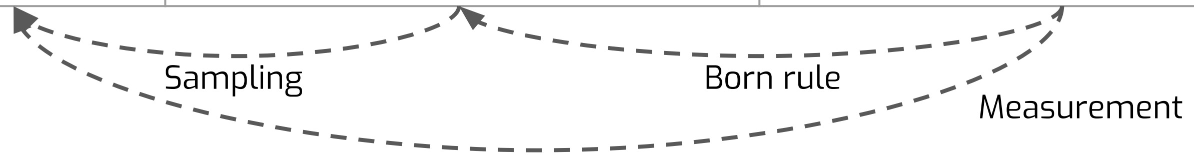
Born rule

$\lvert\alpha_x\rvert^2$ = Probability of bitstring x

How do I program a quantum computer?

Hybrid Quantum Computers | **Quantum Programming** | Hybrid Programming | Hybrid Algorithms

	Bits	Probabilistic Bits	Qubits
State (single unit)	Bit $\in \{0,1\}$	Real vector $a+b \in \mathbb{R}_+$ $\vec{b} = a\vec{0} + b\vec{1}$ $a+b=1$	Complex vector $\alpha, \beta \in \mathbb{C}$ $\vec{\psi} = \alpha\vec{0} + \beta\vec{1}$ $\|\alpha\|^2 + \|\beta\|^2 = 1$
State (multi-unit)	Bitstring $x \in \{0,1\}^n$	Prob. Distribution (stochastic vector) $\vec{s} = \{p_x\}_{x\in\{0,1\}^n}$	Wavefunction (complex vector) $\vec{\psi} = \{\alpha_x\}_{x\in\{0,1\}^n}$
Operations	Boolean Logic	Stochastic Matrices $\sum_{j=1}^{S} P_{i,j} = 1.$	Unitary Matrices $U^\dagger U = 1$
Component Ops	Boolean Gates	Tensor products of matrices	Tensor products of matrices



How do I program a quantum computer?

Hybrid Quantum Computers | **Quantum Programming** | Hybrid Programming | Hybrid Algorithms

Quil (Quantum Instruction Language) gives each quantum operation an instruction

```
<instruction> <qubit targets>
```

$$\xrightarrow{\text{Start in 0}} \quad \Psi = \begin{array}{cccc} [1, & 0, & 0, & 0] \\ 00 & 01 & 10 & 11 \end{array}$$

How do I program a quantum computer?

Hybrid Quantum Computers | **Quantum Programming** | Hybrid Programming | Hybrid Algorithms

Quil (Quantum Instruction Language) gives each quantum operation an instruction

```
<instruction> <qubit targets>
```

$$\Psi = [\underset{00}{1}, \underset{01}{0}, \underset{10}{0}, \underset{11}{0}]$$

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

```
X 0 # “quantum NOT”
```

How do I program a quantum computer?

Quil (Quantum Instruction Language) gives each quantum operation an instruction

```
<instruction> <qubit targets>
```

$$\Psi = [\underset{00}{1}, \underset{01}{0}, \underset{10}{0}, \underset{11}{0}]$$

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

```
X 0 # “quantum NOT”
```

$$\xrightarrow{\text{Apply X instr to 0th qubit}} \Psi = [\underset{00}{0}, \underset{01}{1}, \underset{10}{0}, \underset{11}{0}]$$

How do I program a quantum computer?

Hybrid Quantum Computers | **Quantum Programming** | Hybrid Programming | Hybrid Algorithms

Quil (Quantum Instruction Language) gives each quantum operation an instruction

```
<instruction> <qubit targets>
```

```
X 0 # “quantum NOT”
```

$$\Psi = \begin{array}{cccc} [1, & 0, & 0, & 0] \\ 00 & 01 & 10 & 11 \end{array}$$

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

$$\Psi = \begin{array}{cccc} [0, & 1, & 0, & 0] \\ 00 & 01 & 10 & 11 \end{array}$$

```
X 0
H 0 # Hadamard gate
```

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

How do I program a quantum computer?

Quil (Quantum Instruction Language) gives each quantum operation an instruction

```
<instruction> <qubit targets>
```

$$\Psi = \begin{array}{cccc} [1, & 0, & 0, & 0] \\ 00 & 01 & 10 & 11 \end{array}$$

```
X 0 # “quantum NOT”
```

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

$$\Psi = \begin{array}{cccc} [0, & 1, & 0, & 0] \\ 00 & 01 & 10 & 11 \end{array}$$

```
X 0
H 0 # Hadamard gate
```

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

$$\xrightarrow{\text{Apply H instr to 0th qubit}} \Psi = \begin{array}{cccc} [1/\sqrt{2}, & 1/\sqrt{2}, & 0, & 0] \\ 00 & 01 & 10 & 11 \end{array}$$

How do I program a quantum computer?

Quil (Quantum Instruction Language) gives each quantum operation an instruction

```
<instruction> <qubit targets>
```

$$\Psi = \begin{array}{cccc} [1, & 0, & 0, & 0] \\ 00 & 01 & 10 & 11 \end{array}$$

```
X 0 # “quantum NOT”
```

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

$$\Psi = \begin{array}{cccc} [0, & 1, & 0, & 0] \\ 00 & 01 & 10 & 11 \end{array}$$

```
X 0
H 0 # Hadamard gate
```

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

$$\Psi = \begin{array}{cccc} [1/\sqrt{2}, & 1/\sqrt{2}, & 0, & 0] \\ 00 & 01 & 10 & 11 \end{array}$$

```
CNOT 0 1
```

$$\mathrm{CNOT} = cX = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

How do I program a quantum computer?

Quil (Quantum Instruction Language) gives each quantum operation an instruction

```
<instruction> <qubit targets>
```

$$\Psi = [\underset{00}{1}, \underset{01}{0}, \underset{10}{0}, \underset{11}{0}]$$

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

```
X 0 # “quantum NOT”
```

$$\Psi = [\underset{00}{0}, \underset{01}{1}, \underset{10}{0}, \underset{11}{0}]$$

```
X 0
H 0 # Hadamard gate
```

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

$$\Psi = [\underset{00}{1/\sqrt{2}}, \underset{01}{1/\sqrt{2}}, \underset{10}{0}, \underset{11}{0}]$$

```
CNOT 0 1
```

$$\xrightarrow{\text{Apply CNOT instr to 0 and 1 qubits}} \Psi = [\underset{00}{1/\sqrt{2}}, \underset{01}{0}, \underset{10}{0}, \underset{11}{1/\sqrt{2}}]$$

$$\mathrm{CNOT} = cX = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

How do I program a quantum computer?

Hybrid Quantum Computers | **Quantum Programming** | Hybrid Programming | Hybrid Algorithms

Quil (Quantum Instruction Language) gives each quantum operation an instruction

```
<instruction> <qubit targets>
```

$$\Psi = \begin{array}{cccc} [1, & 0, & 0, & 0] \\ 00 & 01 & 10 & 11 \end{array}$$

```
X 0 # “quantum NOT”
```

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

$$\Psi = \begin{array}{cccc} [0, & 1, & 0, & 0] \\ 00 & 01 & 10 & 11 \end{array}$$

```
X 0
H 0 # Hadamard gate
```

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

$$\Psi = \begin{array}{cccc} [1/\sqrt{2}, & 1/\sqrt{2}, & 0, & 0] \\ 00 & 01 & 10 & 11 \end{array}$$

```
CNOT 0 1
```

$$\mathrm{CNOT} = cX = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

$$\Psi = \begin{array}{cccc} [1/\sqrt{2}, & 0, & 0, & 1/\sqrt{2}] \\ 00 & 01 & 10 & 11 \end{array}$$

Qubits 0 and 1 are ENTANGLED

How do I program a quantum computer?

Quil (Quantum Instruction Language) gives each quantum operation an instruction

```
<instruction> <qubit targets>

 X 0 # “quantum NOT”
 X 0
 H 0 # Hadamard gate
 CNOT 0 1
```

$$\Psi = [\underset{00}{1/\sqrt{2}},\ \underset{01}{0},\ \underset{10}{0},\ \underset{11}{1/\sqrt{2}}]$$

How do I program a quantum computer?

Hybrid Quantum Computers | **Quantum Programming** | Hybrid Programming | Hybrid Algorithms

Quil (Quantum Instruction Language) gives each quantum operation an instruction

```
<instruction> <qubit targets>

 X 0 # “quantum NOT”
 X 0
 H 0 # Hadamard gate
 CNOT 0 1

 # Move quantum data to classical data
 # MEASURE <qubit register> [<bit register>]

 MEASURE 0 [2]
```

$$\Psi = [\underset{00}{1/\sqrt{2}},\ \underset{01}{0},\ \underset{10}{0},\ \underset{11}{1/\sqrt{2}}]$$

How do I program a quantum computer?

Hybrid Quantum Computers | **Quantum Programming** | Hybrid Programming | Hybrid Algorithms

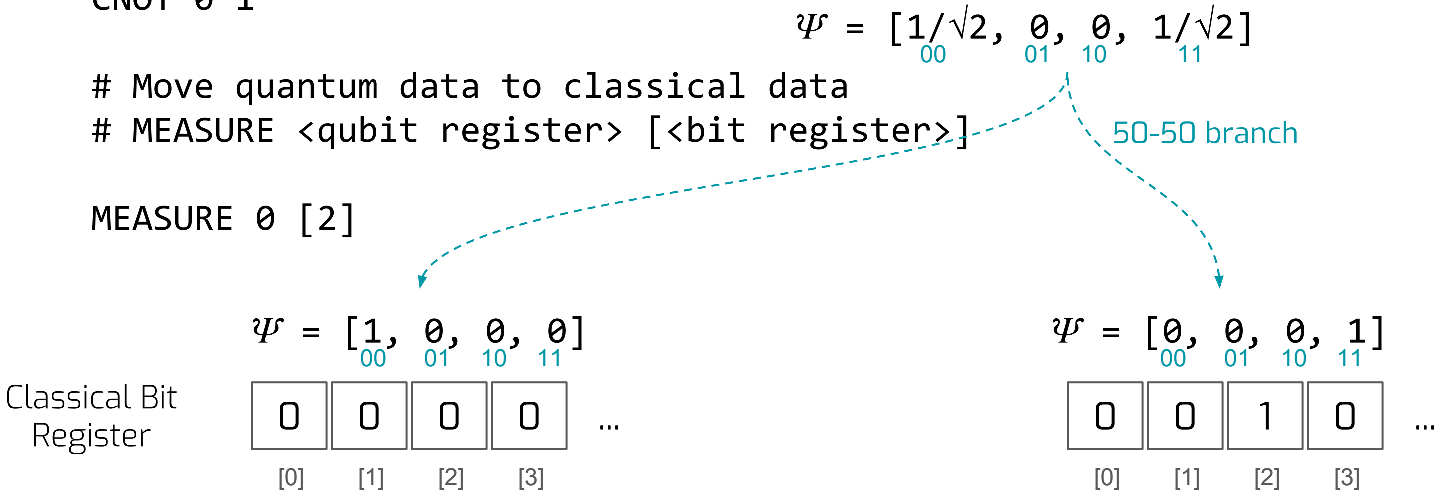
Quil (Quantum Instruction Language) gives each quantum operation an instruction

```
<instruction> <qubit targets>

 X 0 # “quantum NOT”
 X 0
 H 0 # Hadamard gate
 CNOT 0 1

 # Move quantum data to classical data
 # MEASURE <qubit register> [<bit register>]

 MEASURE 0 [2]
```



$\Psi = [1/\sqrt{2}, 0, 0, 1/\sqrt{2}]$

$\Psi = [1, 0, 0, 0]$

Classical Bit Register

[0]	[1]	[2]	[3]	
0	0	0	0	…

$\Psi = [0, 0, 0, 1]$

[0]	[1]	[2]	[3]	
0	0	1	0	…

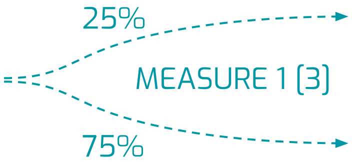
How do I program a quantum computer?

Some more examples of MEASUREMENT

Quantum Memory

Classical Memory

$$\Psi = [\underset{00}{1/2},\ \underset{01}{0},\ \underset{10}{0},\ \underset{11}{\sqrt{3/4}}]$$

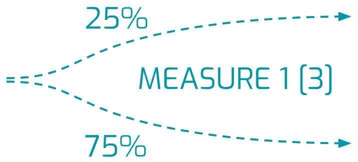


How do I program a quantum computer?

Hybrid Quantum Computers | **Quantum Programming** | Hybrid Programming | Hybrid Algorithms

Some more examples of MEASUREMENT

$$\begin{array}{ccccc} \Psi = [1/2, & 0, & 0, & \sqrt{3}/4] \\ 00 & 01 & 10 & 11 \end{array}$$



25%

MEASURE 1 [3]

75%

Quantum Memory

$$\begin{array}{ccccc} \Psi = [1, & 0, & 0, & 0] \\ 00 & 01 & 10 & 11 \end{array}$$

Classical Memory

0	0	0	0	...
[0]	[1]	[2]	[3]	

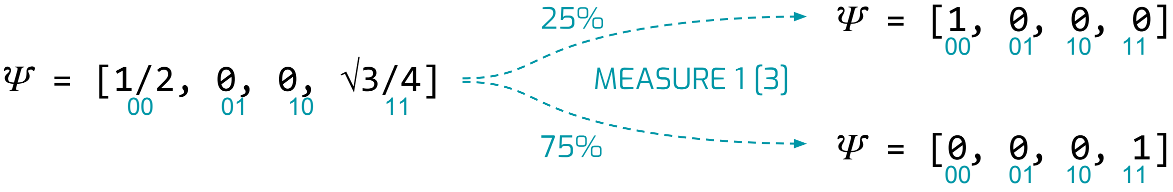
How do I program a quantum computer?

Hybrid Quantum Computers | **Quantum Programming** | Hybrid Programming | Hybrid Algorithms

Some more examples of MEASUREMENT

Quantum Memory

Classical Memory



$\Psi = [\underset{00}{1/2}, \underset{01}{0}, \underset{10}{0}, \underset{11}{\sqrt{3}/4}]$

MEASURE 1 [3]

25%

$\Psi = [\underset{00}{1}, \underset{01}{0}, \underset{10}{0}, \underset{11}{0}]$

0	0	0	0	…
[0]	[1]	[2]	[3]	

75%

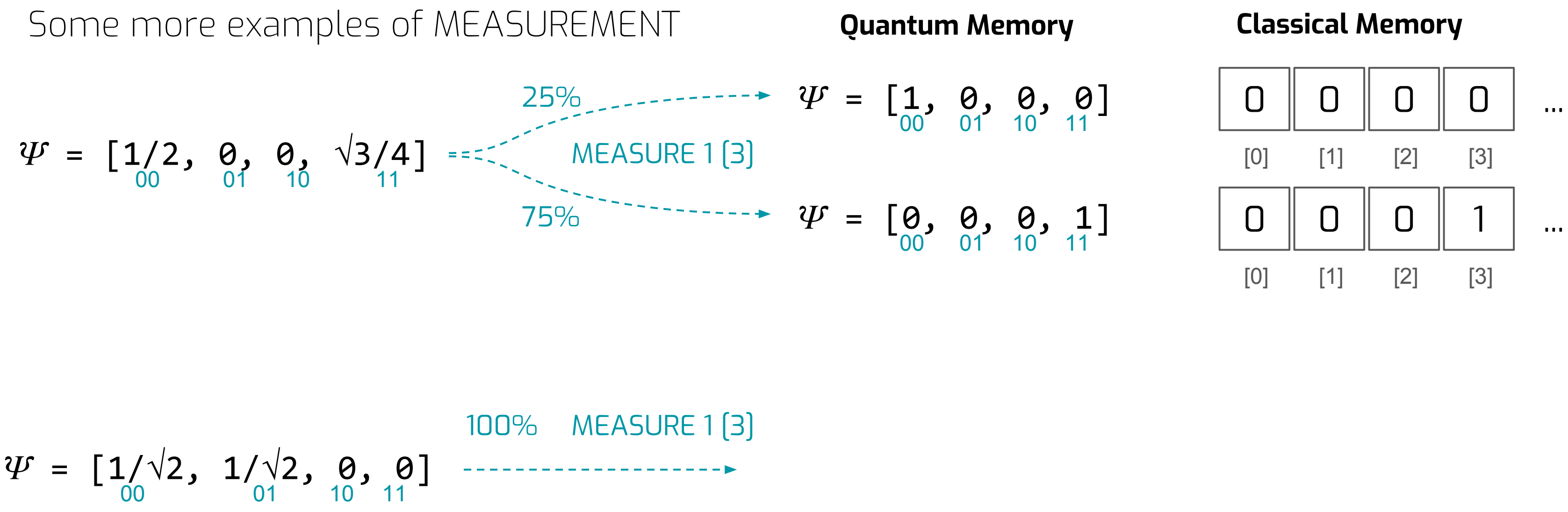
$\Psi = [\underset{00}{0}, \underset{01}{0}, \underset{10}{0}, \underset{11}{1}]$

0	0	0	1	…
[0]	[1]	[2]	[3]	

How do I program a quantum computer?

Hybrid Quantum Computers | **Quantum Programming** | Hybrid Programming | Hybrid Algorithms

Some more examples of MEASUREMENT



Quantum Memory

Classical Memory

$\Psi = [\underset{00}{1/2},\ \underset{01}{0},\ \underset{10}{0},\ \underset{11}{\sqrt{3/4}}]$

MEASURE 1 [3]

25%

$\Psi = [\underset{00}{1},\ \underset{01}{0},\ \underset{10}{0},\ \underset{11}{0}]$

0	0	0	0	...
[0]	[1]	[2]	[3]	

75%

$\Psi = [\underset{00}{0},\ \underset{01}{0},\ \underset{10}{0},\ \underset{11}{1}]$

0	0	0	1	...
[0]	[1]	[2]	[3]	

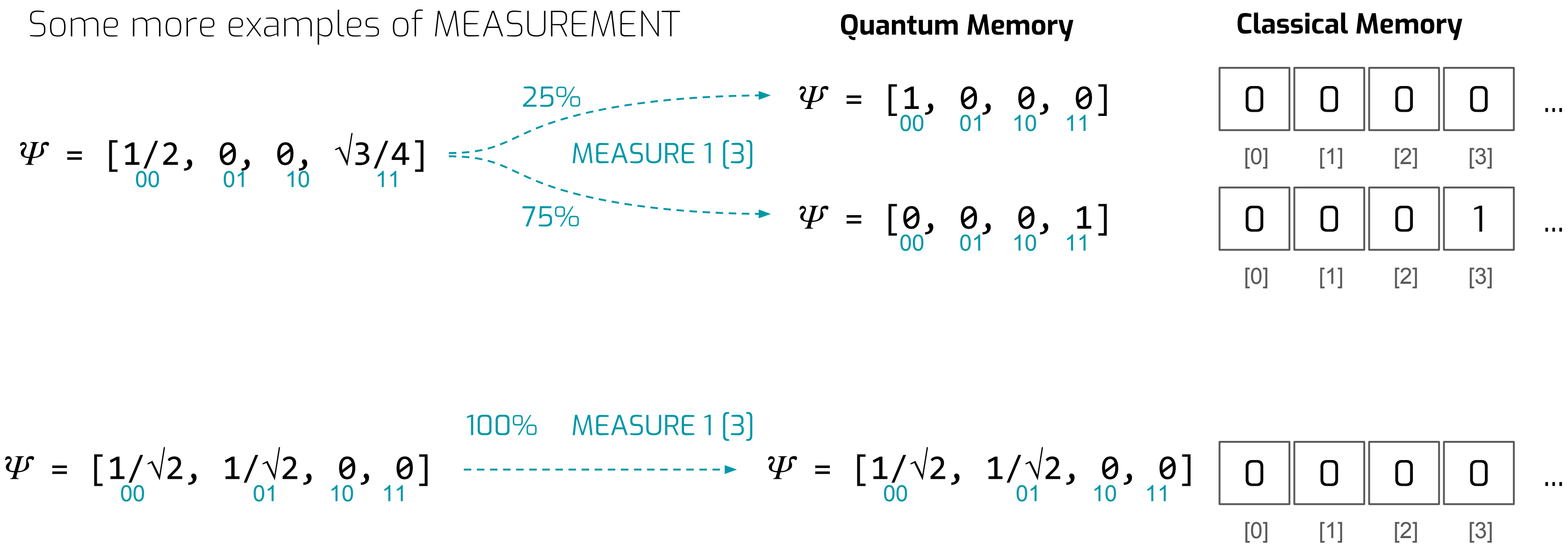
$\Psi = [\underset{00}{1/\sqrt{2}},\ \underset{01}{1/\sqrt{2}},\ \underset{10}{0},\ \underset{11}{0}]$

100% MEASURE 1 [3]

How do I program a quantum computer?

Hybrid Quantum Computers | **Quantum Programming** | Hybrid Programming | Hybrid Algorithms

Some more examples of MEASUREMENT



Quantum Memory

Classical Memory

$\Psi = [\underset{00}{1/2}, \underset{01}{0}, \underset{10}{0}, \underset{11}{\sqrt{3}/4}]$

MEASURE 1 [3]

25%

$\Psi = [\underset{00}{1}, \underset{01}{0}, \underset{10}{0}, \underset{11}{0}]$

0	0	0	0	...
[0]	[1]	[2]	[3]	

75%

$\Psi = [\underset{00}{0}, \underset{01}{0}, \underset{10}{0}, \underset{11}{1}]$

0	0	0	1	...
[0]	[1]	[2]	[3]	

$\Psi = [\underset{00}{1/\sqrt{2}}, \underset{01}{1/\sqrt{2}}, \underset{10}{0}, \underset{11}{0}]$

100% MEASURE 1 [3]

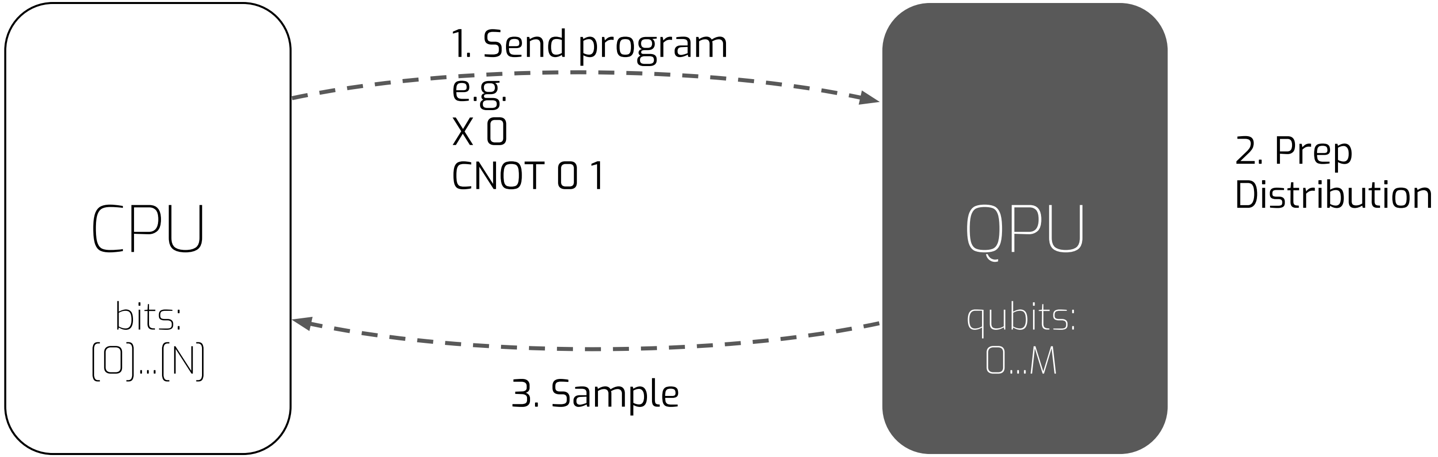
$\Psi = [\underset{00}{1/\sqrt{2}}, \underset{01}{1/\sqrt{2}}, \underset{10}{0}, \underset{11}{0}]$

0	0	0	0	...
[0]	[1]	[2]	[3]	

How do I program a quantum computer?

Hybrid Quantum Computers | **Quantum Programming** | Hybrid Programming | Hybrid Algorithms

Quantum programming is preparing and sampling from complicated distributions



The Quil Programming Model

Targets a **Quantum Abstract Machine (QAM)** with a syntax for representing state transitions

Ψ: Quantum state (qubits) → quantum instructions

C: Classical state (bits) → classical and measurement instructions

κ: Execution state (program)→ control instructions (e.g., jumps)

```
# Quil Example
H 3
MEASURE 3 [4]
JUMP-WHEN @END [5]
.
.
.
```

The Quil Programming Model

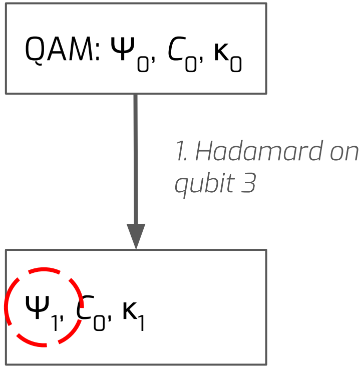
Targets a **Quantum Abstract Machine (QAM)** with a syntax for representing state transitions

Ψ: Quantum state (qubits) → quantum instructions

C: Classical state (bits) → classical and measurement instructions

κ: Execution state (program)→ control instructions (e.g., jumps)

0. Initialize into zero states



QAM: Ψ_0, C_0, κ_0

1. Hadamard on qubit 3

Ψ_1, C_0, κ_1

```
# Quil Example
H 3
MEASURE 3 [4]
JUMP-WHEN @END [5]
.
.
.
```

The Quil Programming Model

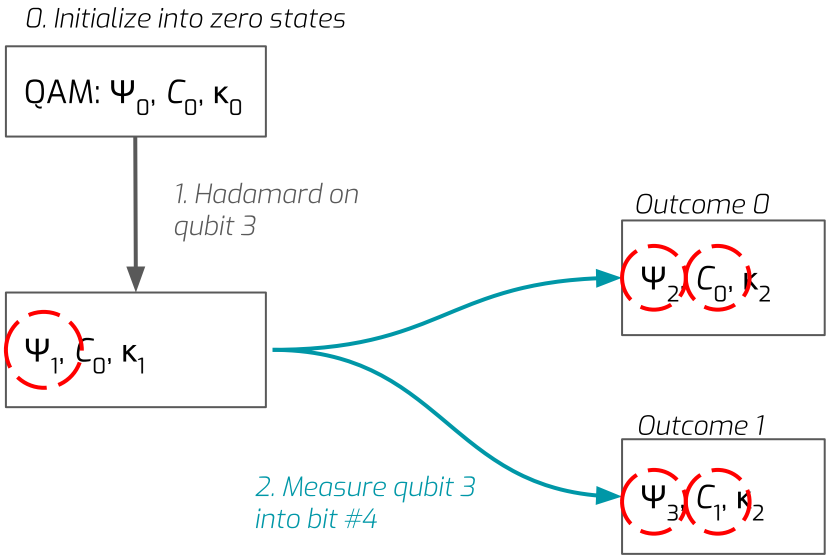
Targets a **Quantum Abstract Machine (QAM)** with a syntax for representing state transitions

Ψ: Quantum state (qubits) → quantum instructions

C: Classical state (bits) → classical and measurement instructions

κ: Execution state (program) → control instructions (e.g., jumps)

```
# Quil Example
H 3
MEASURE 3 [4]
JUMP-WHEN @END [5]
.
.
.
```



0. Initialize into zero states

QAM: Ψ_0, C_0, κ_0

1. Hadamard on qubit 3

Ψ_1, C_0, κ_1

2. Measure qubit 3 into bit #4

Outcome 0

Ψ_2, C_0, κ_2

Outcome 1

Ψ_3, C_1, κ_2

The Quil Programming Model

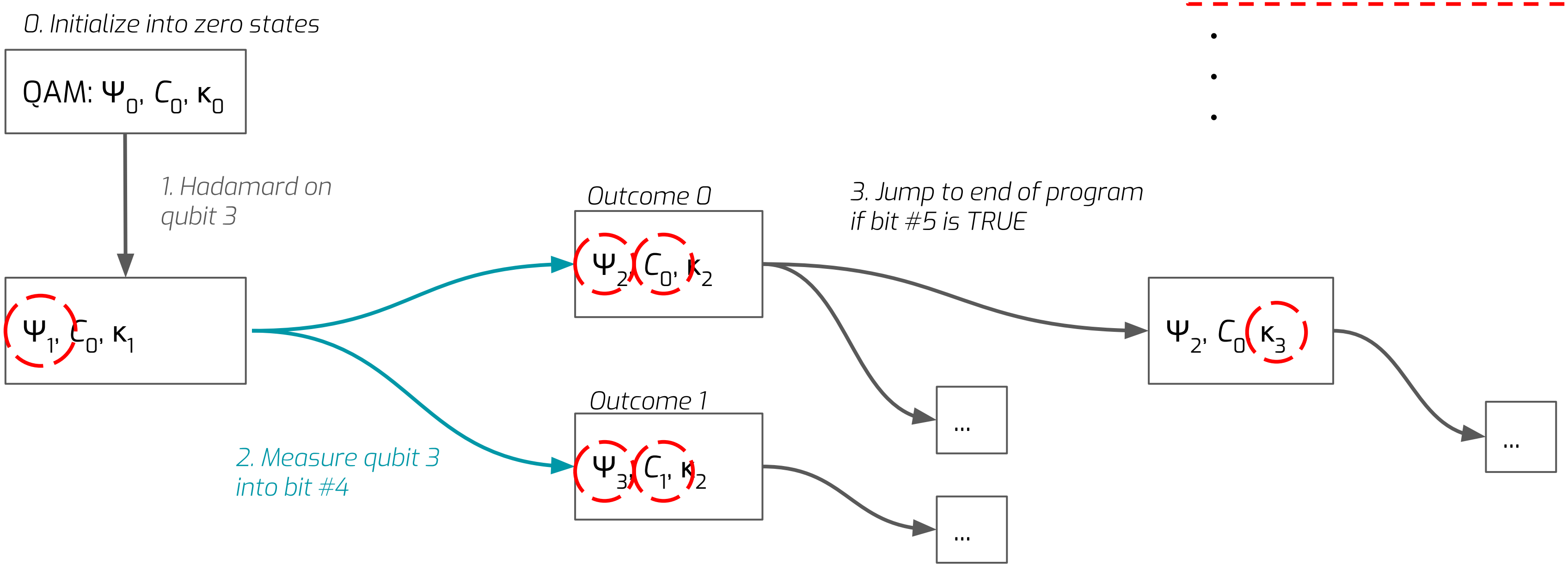
Targets a **Quantum Abstract Machine (QAM)** with a syntax for representing state transitions

Ψ: Quantum state (qubits) → quantum instructions

C: Classical state (bits) → classical and measurement instructions

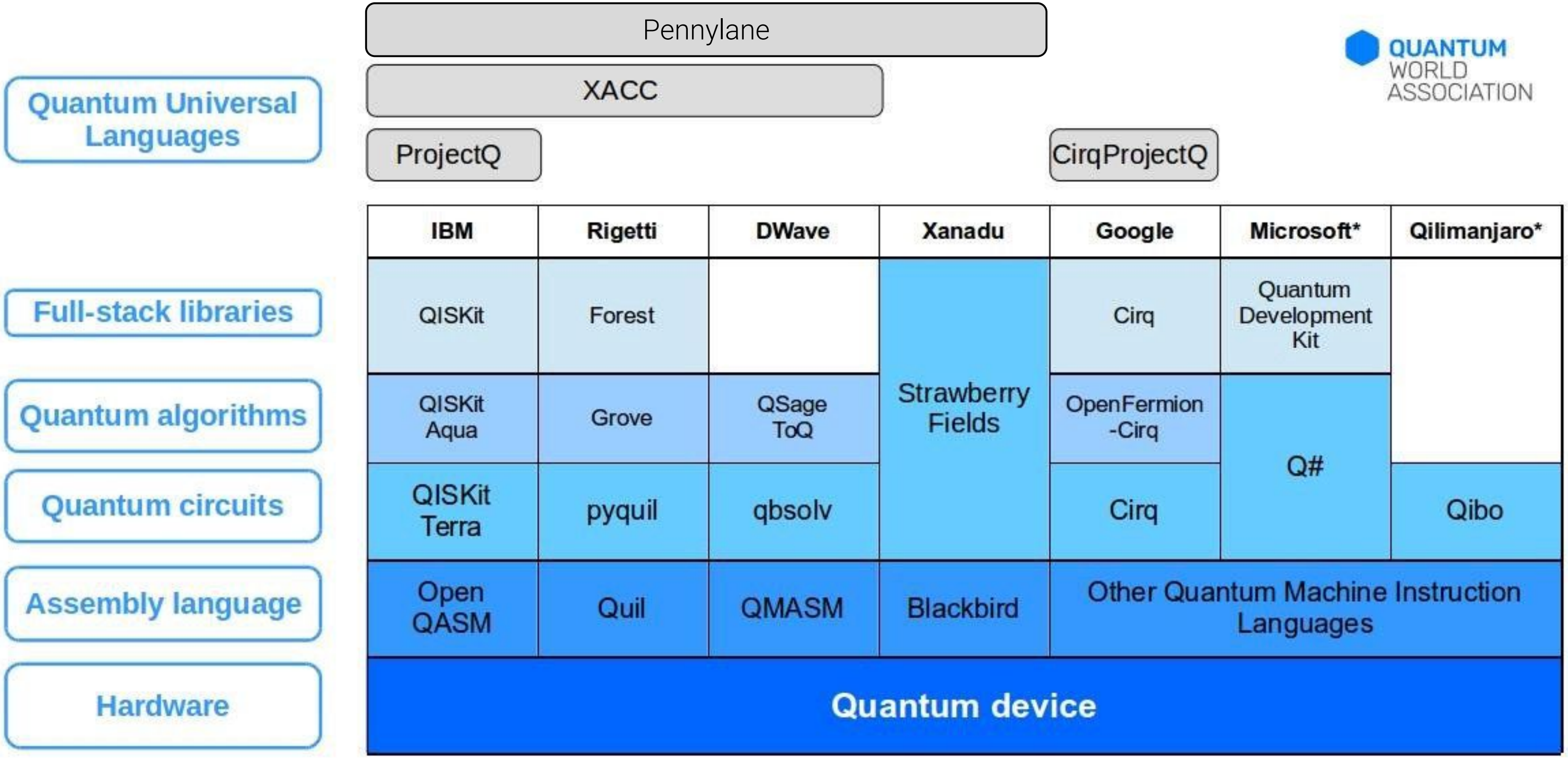
κ: Execution state (program)→ control instructions (e.g., jumps)

```
# Quil Example
H 3
MEASURE 3 [4]
JUMP-WHEN @END [5]
.
.
.
```



Quantum Computing Programming Languages

QUANTUM WORLD ASSOCIATION



Quantum Universal Languages	Pennylane						
	XACC						
	ProjectQ				CirqProjectQ		
	IBM	**Rigetti**	**DWave**	**Xanadu**	**Google**	**Microsoft***	**Qilimanjaro***
Full-stack libraries	QISKit	Forest		Strawberry Fields	Cirq	Quantum Development Kit	
Quantum algorithms	QISKit Aqua	Grove	QSage ToQ		OpenFermion -Cirq	Q#	
Quantum circuits	QISKit Terra	pyquil	qbsolv		Cirq		Qibo
Assembly language	Open QASM	Quil	QMASM	Blackbird	Other Quantum Machine Instruction Languages		
Hardware	**Quantum device**						

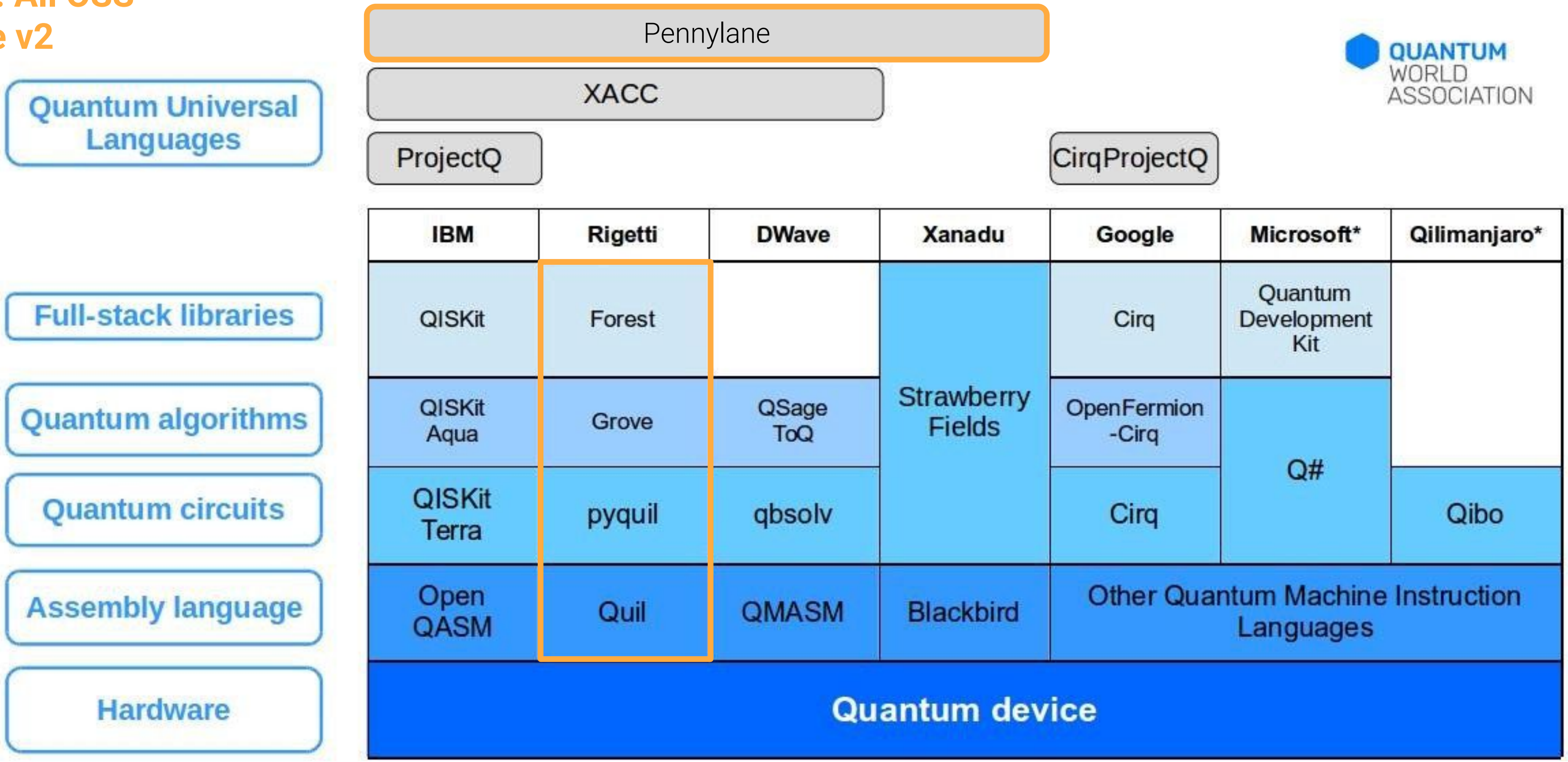
* Hardware under development. Quantum programs are run on their own simulators.

“Quantum Language” is refered with no distinction both as a quantum equivalence of a programming language and as a library to write quantum programs supported by some well-known classical programming language.

© Alba Cervera-Lierta for the QWA (2018)

Main tools in this course. All OSS Apache v2

Quantum Computing Programming Languages



QUANTUM WORLD ASSOCIATION

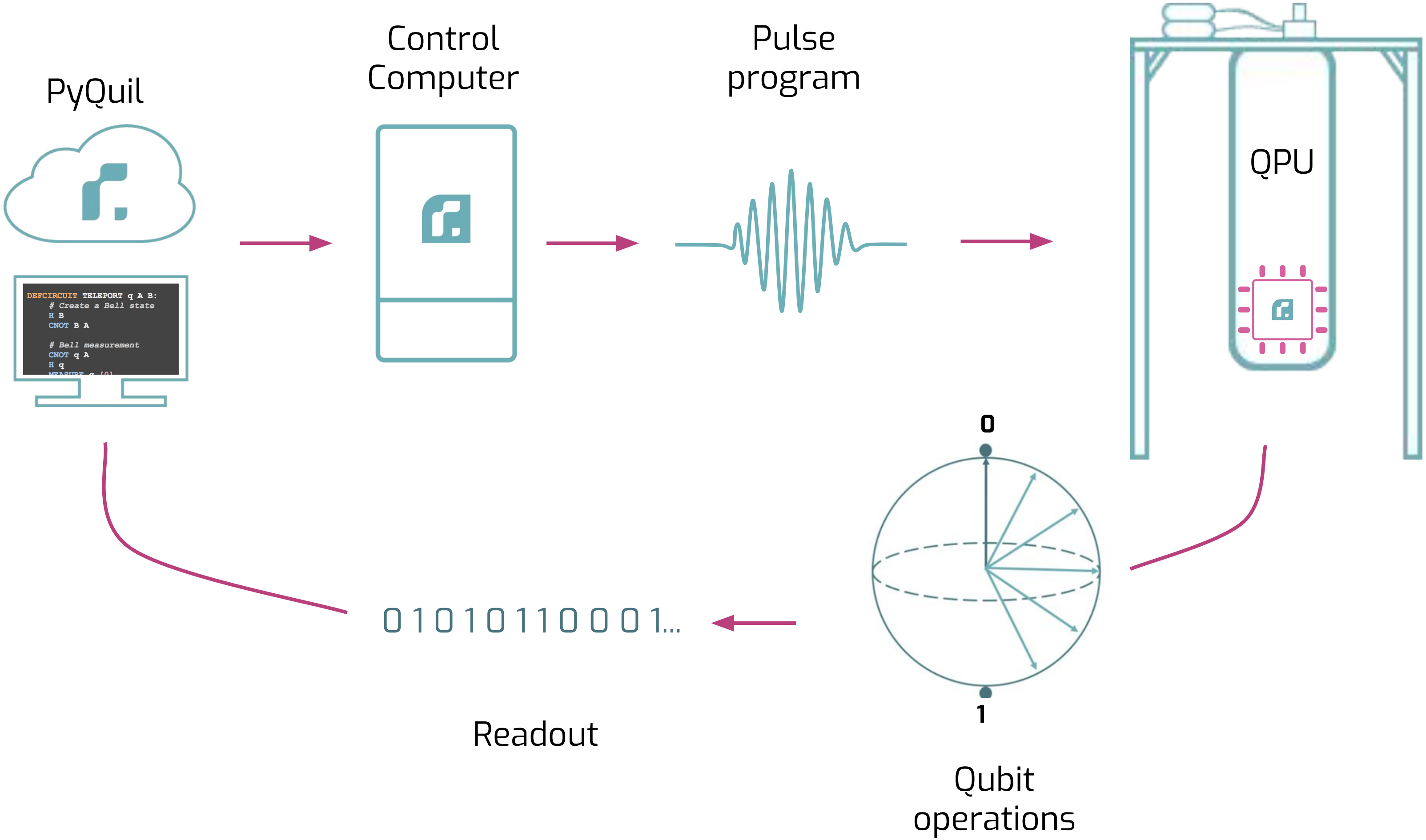
Quantum Universal Languages: Pennylane, XACC, ProjectQ, CirqProjectQ

	IBM	Rigetti	DWave	Xanadu	Google	Microsoft*	Qilimanjaro*
Full-stack libraries	QISKit	Forest		Strawberry Fields	Cirq	Quantum Development Kit	
Quantum algorithms	QISKit Aqua	Grove	QSage ToQ		OpenFermion -Cirq	Q#	
Quantum circuits	QISKit Terra	pyquil	qbsolv		Cirq		Qibo
Assembly language	Open QASM	Quil	QMASM	Blackbird	Other Quantum Machine Instruction Languages		
Hardware	Quantum device						

* Hardware under development. Quantum programs are run on their own simulators.

“Quantum Language” is refered with no distinction both as a quantum equivalence of a programming language and as a library to write quantum programs supported by some well-known classical programming language.

© Alba Cervera-Lierta for the QWA (2018)



PyQuil

```
DEFCIRCUIT TELEPORT q A B:
    # Create a Bell state
    H B
    CNOT B A

    # Bell measurement
    CNOT q A
    H q
    MEASURE q [0]
```

Control Computer

Pulse program

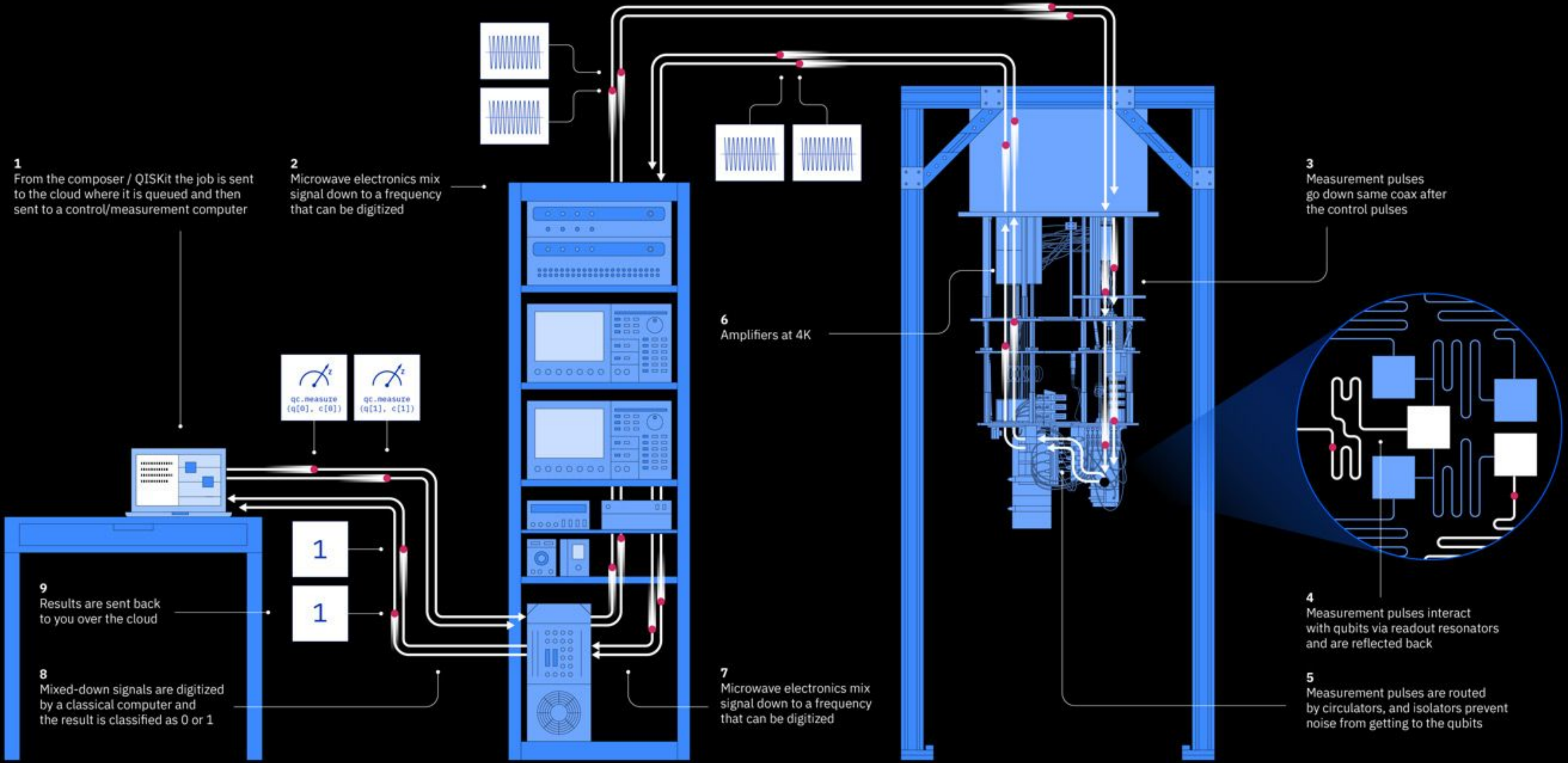
QPU

Qubit operations

0 1 0 1 0 1 1 0 0 0 1...

Readout

Image: Rigetti



1
From the composer / QISKit the job is sent to the cloud where it is queued and then sent to a control/measurement computer

2
Microwave electronics mix signal down to a frequency that can be digitized

3
Measurement pulses go down same coax after the control pulses

4
Measurement pulses interact with qubits via readout resonators and are reflected back

5
Measurement pulses are routed by circulators, and isolators prevent noise from getting to the qubits

6
Amplifiers at 4K

7
Microwave electronics mix signal down to a frequency that can be digitized

8
Mixed-down signals are digitized by a classical computer and the result is classified as 0 or 1

9
Results are sent back to you over the cloud

How do I program a quantum computer?

We need hybrid programming because of errors

Chance of hardware error in a classical computer:

0.000,000,000,000,000,000,000,1 %

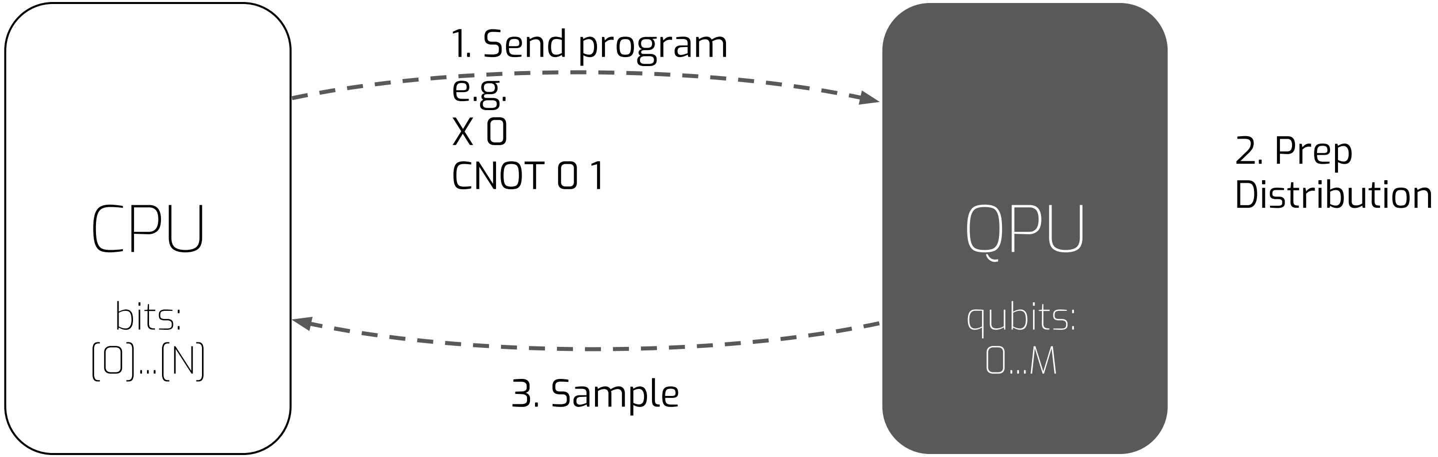
Chance of hardware error in a quantum computer:

0.1%

How do I program a quantum computer?

Hybrid Quantum Computers | Quantum Programming | Hybrid Programming | **Hybrid Algorithms**

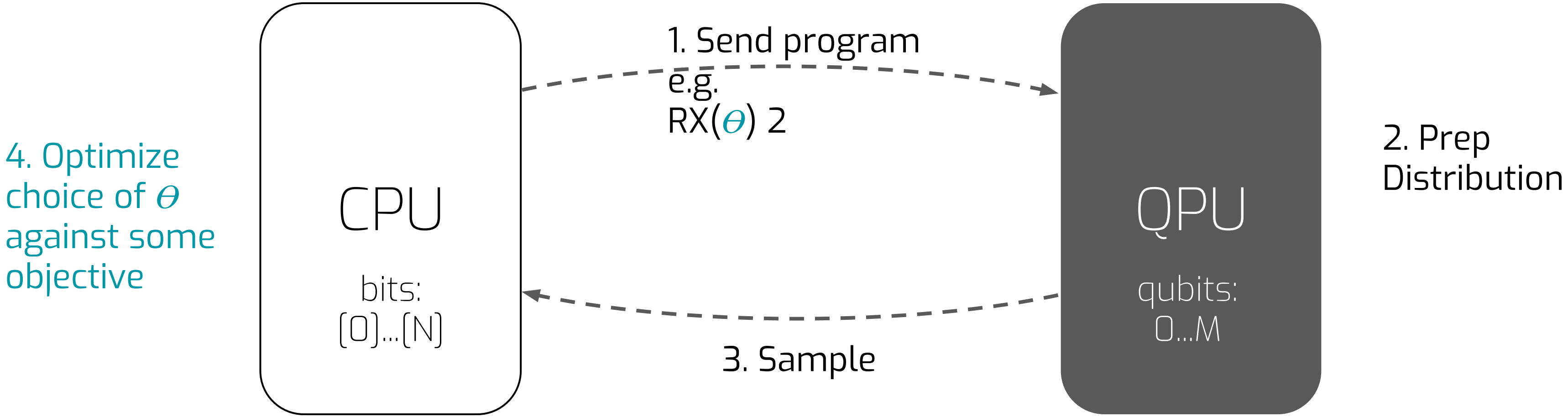
Quantum programming is preparing and sampling from complicated distributions



How do I program a quantum computer?

Hybrid Quantum Computers | Quantum Programming | Hybrid Programming | **Hybrid Algorithms**

By parameterizing quantum programs we can train them to be robust to noise



4. Optimize choice of θ against some objective

CPU

bits:
[0]...[N]

1. Send program
e.g.
RX(θ) 2

2. Prep Distribution

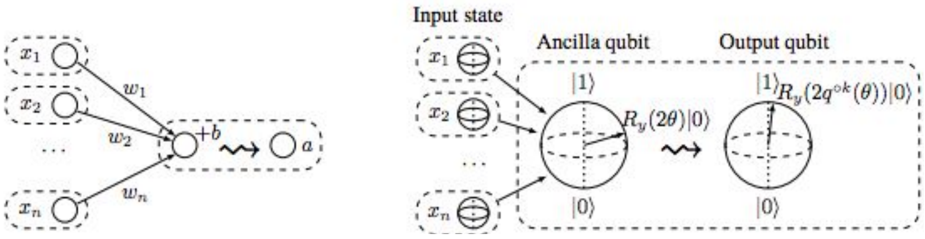
QPU

qubits:
0...M

3. Sample

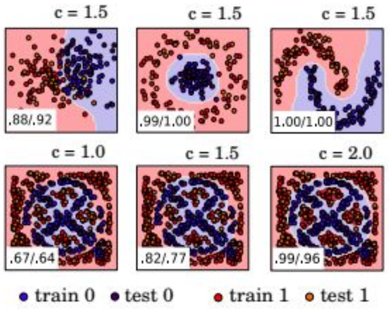
Quantum Machine Learning

> Quantum neuron: an elementary building block for machine learning on quantum computers. (Cao et al. 2017)

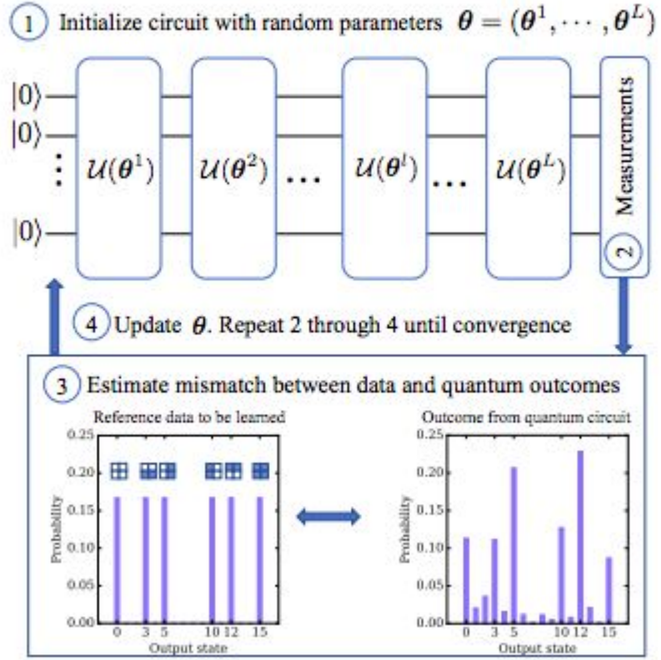


> Quantum circuit learning. (Mitarai et al. 2018)

> Quantum machine learning in feature Hilbert spaces. (Schuld and Killoran 2018)



A generative modeling approach for benchmarking and training shallow quantum circuits. (Benedetti et al. 2018)


1 Initialize circuit with random parameters $\boldsymbol{\theta} = (\boldsymbol{\theta}^1, \cdots, \boldsymbol{\theta}^L)$
$|0\rangle$
$|0\rangle$
$|0\rangle$
$\mathcal{U}(\boldsymbol{\theta}^1)$
$\mathcal{U}(\boldsymbol{\theta}^2)$
$\mathcal{U}(\boldsymbol{\theta}^l)$
$\mathcal{U}(\boldsymbol{\theta}^L)$
Measurements
2
4 Update $\boldsymbol{\theta}$. Repeat 2 through 4 until convergence
3 Estimate mismatch between data and quantum outcomes
Reference data to be learned
Outcome from quantum circuit
Probability
0.25
0.20
0.15
0.10
0.05
0.00
0
3
5
10
12
15
Output state

The Variational Quantum Eigensolver

Used for the electronic structure problem in quantum chemistry

1. MOLECULAR DESCRIPTION

e.g. Electronic Structure Hamiltonian

$$H = \sum_{i,j<i}^{N_n} \frac{Z_i Z_j}{|R_i - R_j|} + \sum_{i=1}^{N_e} \frac{-\nabla^2_{r_i}}{2} - \sum_{ij}^{N_n,N_e} \frac{Z_i}{|R_i - r_j|} + \sum_{i,j<i}^{N_e} \frac{1}{|r_i - r_j|}.$$

2. MAP TO QUBIT REPRESENTATION

e.g. Bravyi-Kitaev or Jordan-Wigner Transform

e.g. DI-HYDROGEN

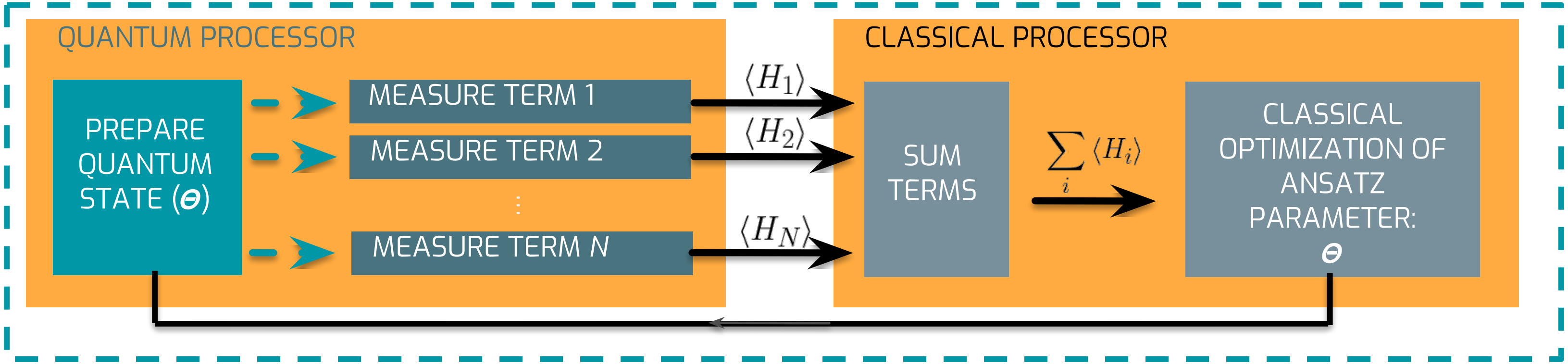
$$\begin{aligned} H = {} & f_0 \mathbb{1} + f_1 Z_0 + f_2 Z_1 + f_3 Z_2 + f_1 Z_0 Z_1 \\ & + f_4 Z_0 Z_2 + f_5 Z_1 Z_3 + f_6 X_0 Z_1 X_2 + f_6 Y_0 Z_1 Y_2 \\ & + f_7 Z_0 Z_1 Z_2 + f_4 Z_0 Z_2 Z_3 + f_3 Z_1 Z_2 Z_3 \\ & + f_6 X_0 Z_1 X_2 Z_3 + f_6 Y_0 Z_1 Y_2 Z_3 + f_7 Z_0 Z_1 Z_2 Z_3 \end{aligned}$$

3. PARAMETERIZED ANSATZ

e.g. Unitary Coupled Cluster
Variational Adiabatic Ansatz

$$\frac{\langle \varphi(\vec{\theta})|\, H \,|\varphi(\vec{\theta})\rangle}{\langle \varphi(\vec{\theta})|\varphi(\vec{\theta})\rangle} \geq E_0$$

4. RUN Q.V.E. QUANTUM-CLASSICAL HYBRID ALGORITHM



QUANTUM PROCESSOR

PREPARE QUANTUM STATE (*ϴ*)

MEASURE TERM 1

MEASURE TERM 2

⋮

MEASURE TERM *N*

$\langle H_1 \rangle$

$\langle H_2 \rangle$

$\langle H_N \rangle$

CLASSICAL PROCESSOR

SUM TERMS

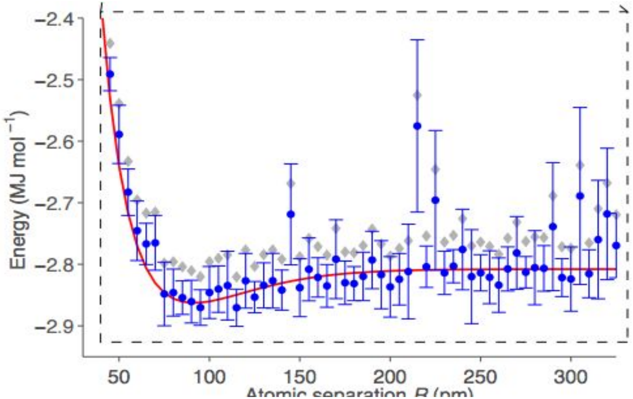
$\sum_i \langle H_i \rangle$

CLASSICAL OPTIMIZATION OF ANSATZ PARAMETER: *ϴ*

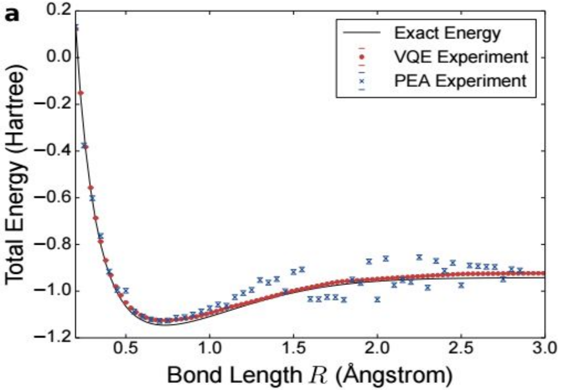
O'Malley, P. J. J., et al. (2015). Scalable Quantum Simulation of Molecular Energies. *arXiv:1512.06860.*
Wecker, D., et al. (2015). Progress towards practical quantum variational algorithms. *Physical Review A*, *92*(4), 042303.

McClean, J. R. et al. (2015). The theory of variational hybrid quantum-classical algorithms. *arXiv:1509.04279.*
Peruzzo, A., et al. (2014). A variational eigenvalue solver on a photonic quantum processor. *Nature communications*, *5*.

VQE Simulations on Quantum Hardware

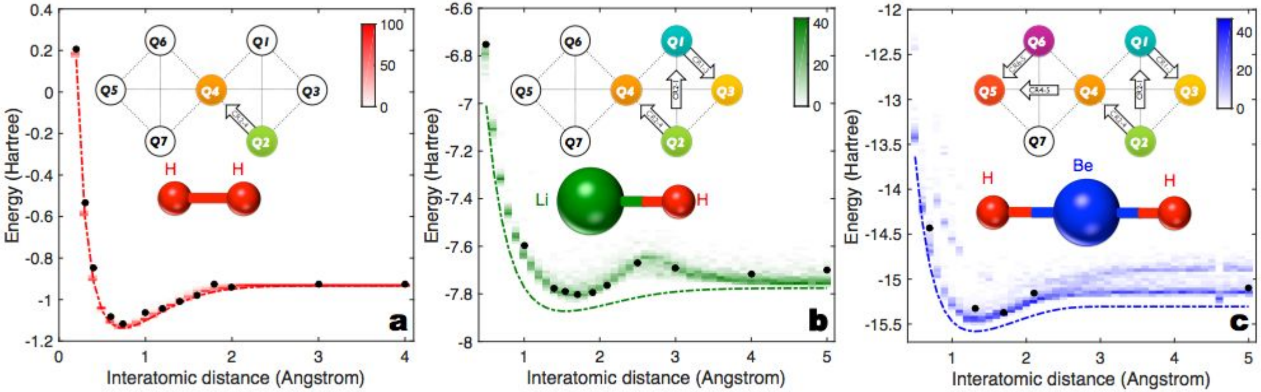


Peruzzo et al. 1304.3061



O’Malley et al. 1512.06860

Kandala et al. 1704.05018



Quantum Approximate Optimization Algorithm

(QAOA) Hybrid algorithm used for constraint satisfaction problems

Given binary constraints:

$$z \in \{0,1\}^n$$

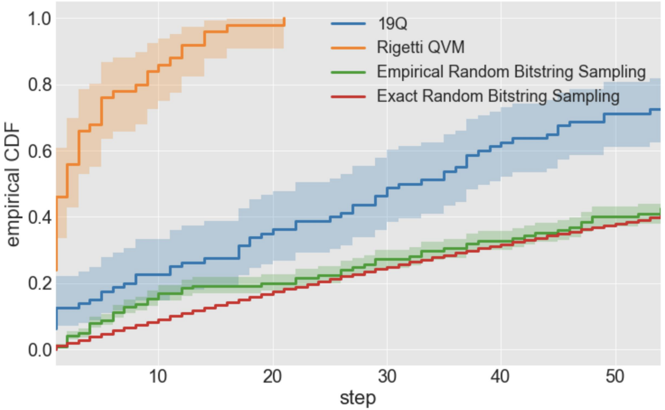
$$C_a(z) = \begin{cases} 1 & \text{if } z \text{ satisfies the constraint } a \\ 0 & \text{if } z \text{ does not .} \end{cases}$$

MAXIMIZE

$$\boxed{C(z) = \sum_{a=1}^{m} C_a(z)}$$

Traveling Salesperson   Scheduling   K-means clustering   Boltzmann Machine Training

Hadfield et al. 2017 (1709.03489)   Otterbach et al. 2017 (1712.05771)   Verdon et al. 2017 (1712.05304)



QAOA in **Forest**

In ***19*** lines of code

```
from pyquil import Program
from pyquil.api import WavefunctionSimulator
from pyquil.gates import H
from pyquil.paulis import sZ, sX, sI, exponentiate_commuting_pauli_sum

graph = [(0, 1), (1, 2), (2, 3), (3, 0)]
nodes = range(4)

init_state_prog = sum([H(i) for i in nodes], Program())
h_cost = -0.5 * sum(sI(nodes[0]) - sZ(i) * sZ(j) for i, j in graph)
h_driver = -1. * sum(sX(i) for i in nodes)

def qaoa_ansatz(betas, gammas):
    return sum([exponentiate_commuting_pauli_sum(h_cost)(g) + \
       exponentiate_commuting_pauli_sum(h_driver)(b) \
              for g, b in zip(gammas, betas)], Program())

def qaoa_cost(params):
    half = int(len(params)/2)
    betas, gammas = params[:half], params[half:]
    program = init_state_prog + qaoa_ansatz(betas, gammas)
    return WavefunctionSimulator().expectation(prep_prog=program, pauli_terms=h_cost)

minimize(qaoa_cost, x0=[0., 0.5, 0.75, 1.], method='Nelder-Mead', options={'disp': True})
```

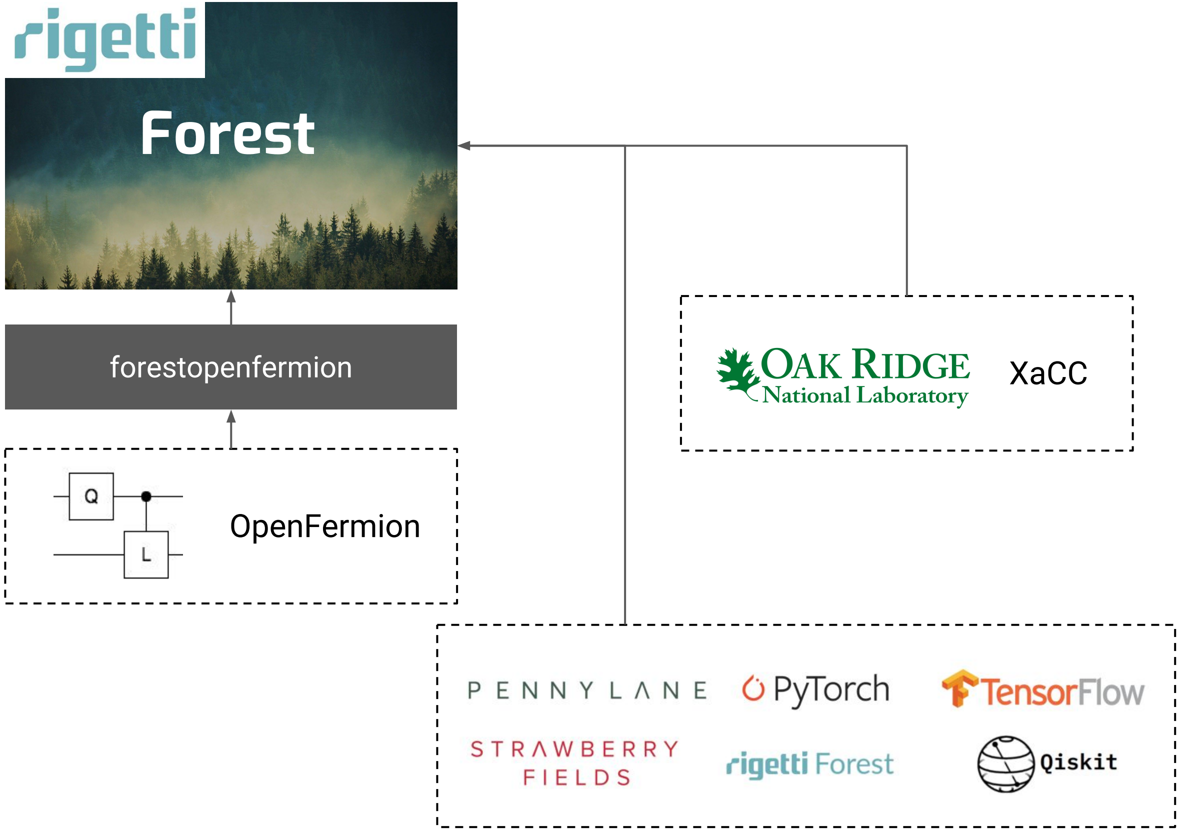
Open areas in quantum programming

> Debuggers

> Optimizing compilers

> Application specific packages

> **Adoption and implementations**



forestopenfermion

OpenFermion

XaCC

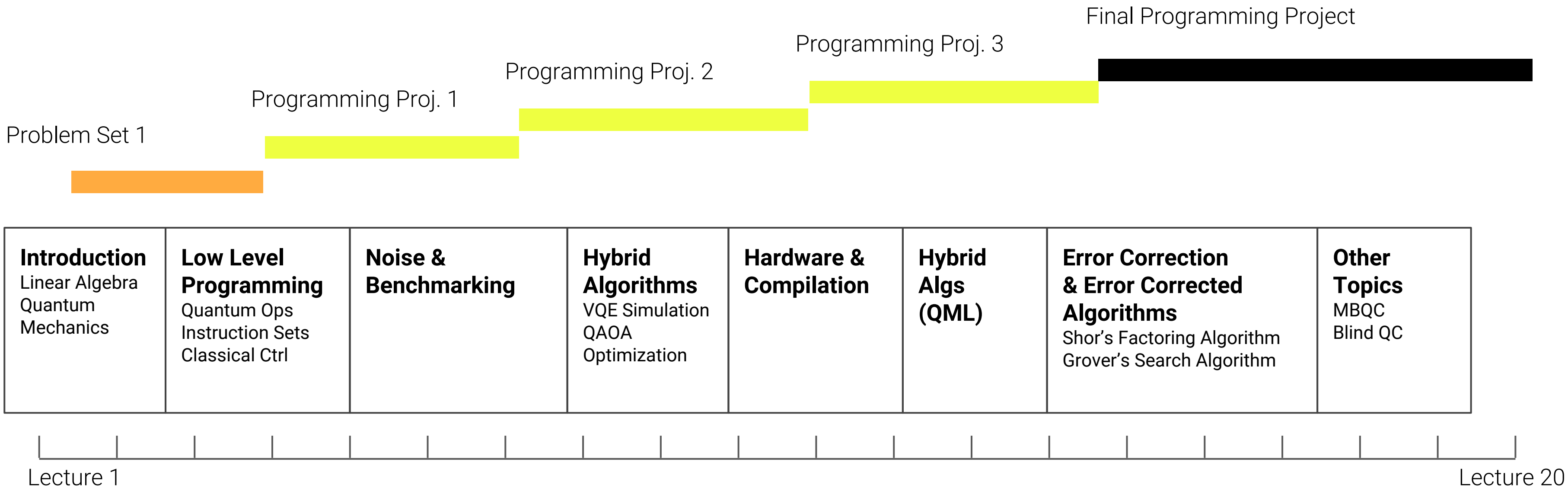
Q1. Why program a quantum computer?

New power | New opportunity | Fundamental curiosity

Q2. How do I program a quantum computer?

Hybrid quantum programming (usually) **in Python!**

Course Topics & Timeline



Problem Set 1

Programming Proj. 1

Programming Proj. 2

Programming Proj. 3

Final Programming Project

Introduction
Linear Algebra
Quantum Mechanics

Low Level Programming
Quantum Ops
Instruction Sets
Classical Ctrl

Noise & Benchmarking

Hybrid Algorithms
VQE Simulation
QAOA
Optimization

Hardware & Compilation

Hybrid Algs (QML)

Error Correction & Error Corrected Algorithms
Shor’s Factoring Algorithm
Grover’s Search Algorithm

Other Topics
MBQC
Blind QC

Actions for between now and the next lecture:

1. Read the syllabus.

2. Read Mike & Ike Chapters 1 & 2. Especially review Sections 2.2, 2.3 & 2.6.

3. Review Linear Algebra. You will need:

   Vectors and linear maps
   Bases and linear independence
   Pauli Matrices
   Inner Products
   Eigenvalues & Eigenvectors
   Adjoints
   Hermitian Operators
   Unitary Matrices
   Tensor Products
   Matrix Exponentials
   Traces
   Commutators and Anti-commutators

4. Download and install pyQuil: https://pyquil.readthedocs.io