rigetti

A Platform for Quantum Programming

Will Zeng
Rigetti Computing

Stanford Platform Lab Seminar
16.30-17.30 in Gates 415
April 3, 2018

Q1. Why program a quantum computer?

Q2. How do I program a quantum computer?

Why program a quantum computer?

New power | New opportunity | Fundamental curiosity

Why program a quantum computer?

New power | New opportunity | Fundamental curiosity

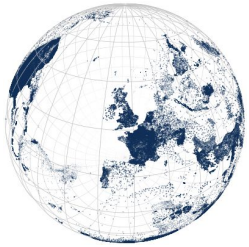
Quantum computing power* scales exponentially with qubits
N bits can exactly simulate **log N qubits**

This compute unit....

Commodore 64

AWS M4 Instance

1 Million x Commodore 64



Entire Global Cloud

1 Billion x
(1 Million x Commodore 64)

can exactly simulate:

10 Qubits

30 Qubits

60 Qubits

* We will be more precise later in the talk

Why program a quantum computer?

New power | New opportunity | Fundamental curiosity

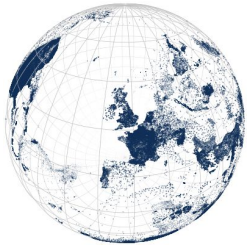
Quantum computing power* scales exponentially with qubits
N bits can exactly simulate **log N qubits**

This compute unit....

Commodore 64

AWS M4 Instance

1 Million x Commodore 64



Entire Global Cloud

1 Billion x
(1 Million x Commodore 64)

can exactly simulate: **10 Qubits** **30 Qubits** **60 Qubits**

Rigetti 19 qubits
available since Dec 2017

* We will be more precise later in the talk

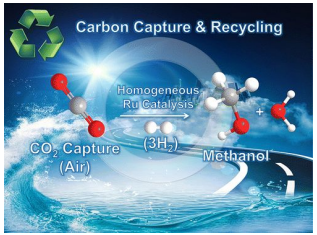
Why program a quantum computer?

New power | New opportunity | Fundamental curiosity

Alternative Energy Research

> Efficiently convert atmospheric CO_2 to methanol

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



Robotic Manufacturing

> Reduce manufacturing time and cost

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

Supply Chain Optimization

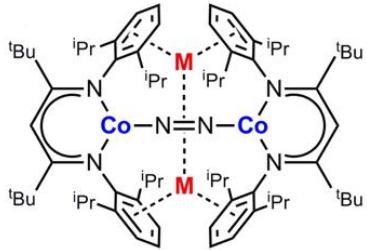
> Forecast and optimize for future inventory demand

> NP-hard scheduling and logistics map into quantum applications

Computational Materials Science

> Design of better catalysts for batteries

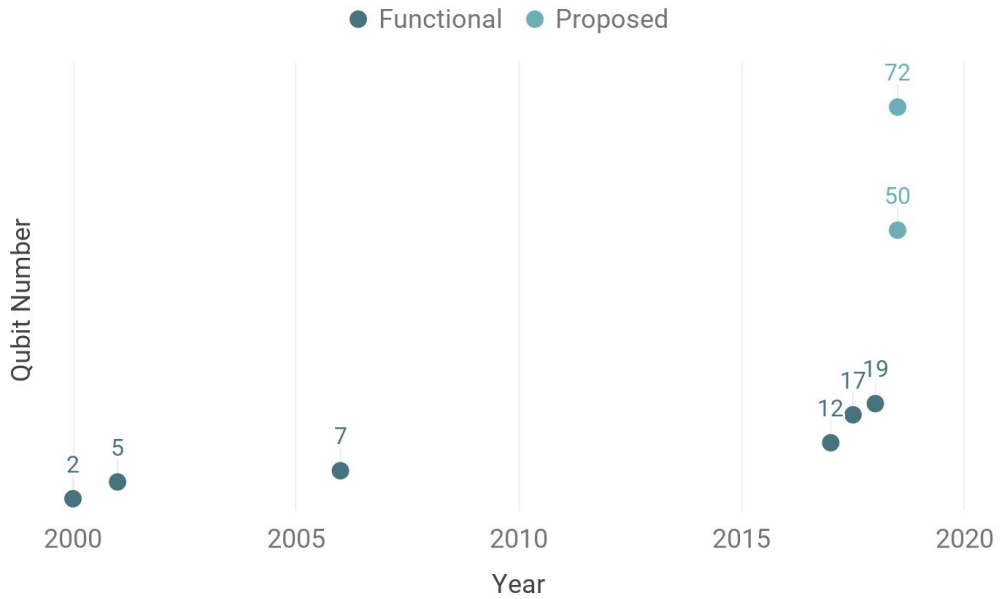
> Quantum algorithms for calculating electronic structure



Why program a quantum computer?

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

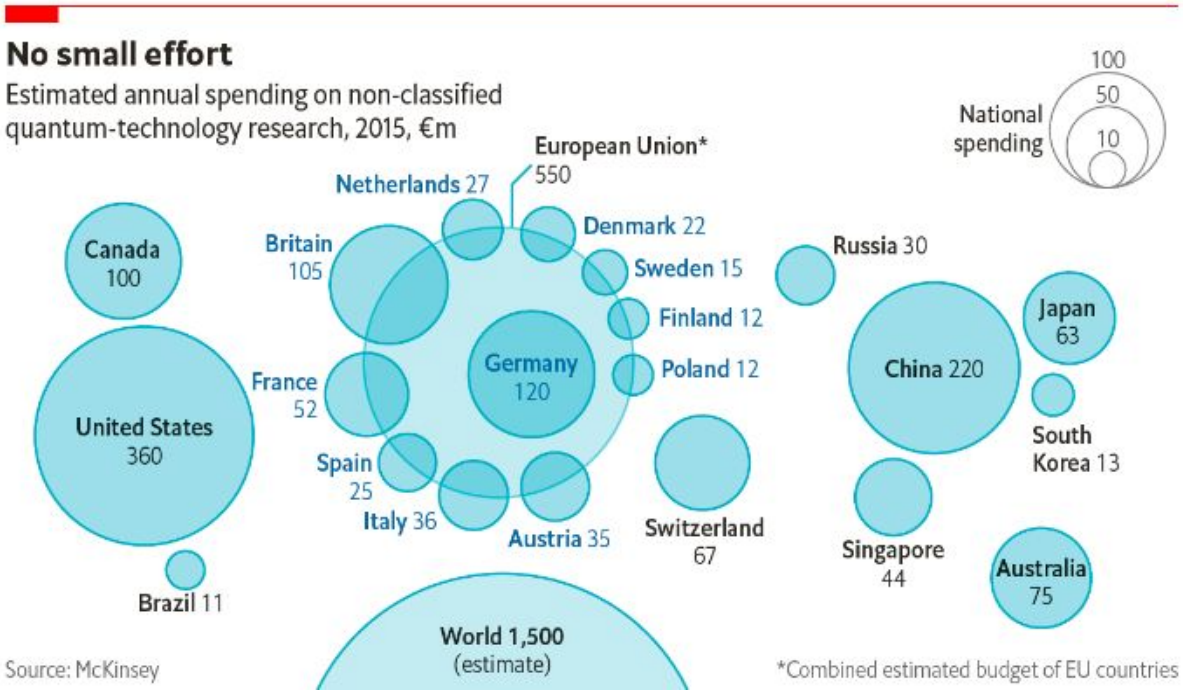
Quantum processors are scaling up quickly



Why program a quantum computer?

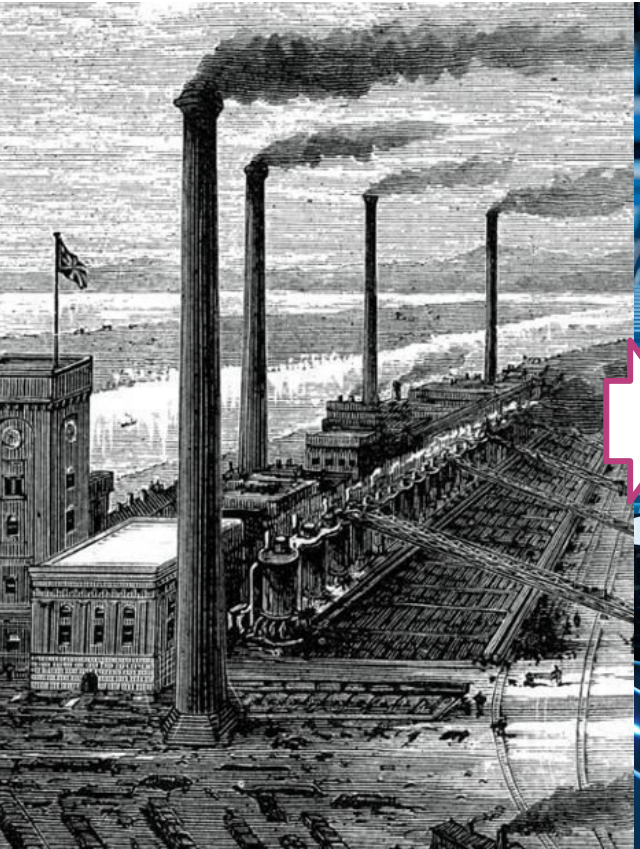
New power | **New opportunity** | Fundamental curiosity

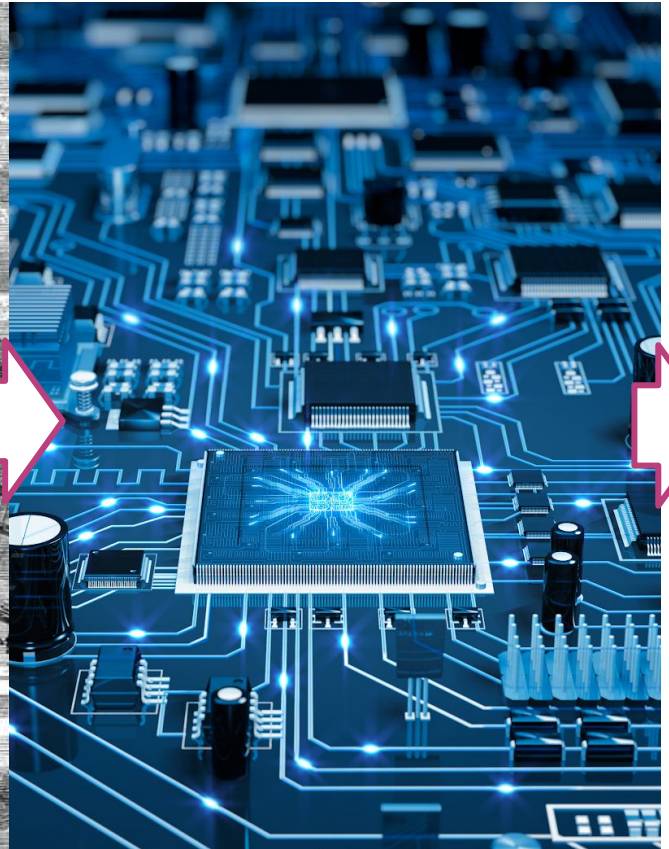
Investments across academia, government, and industry are global and growing



Why program a quantum computer?

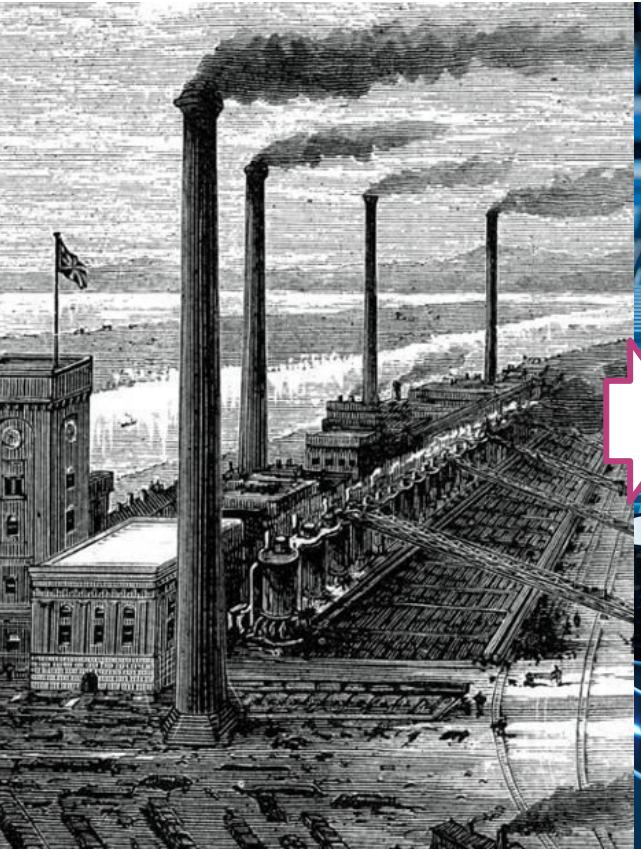
New power | New opportunity | **Fundamental curiosity**

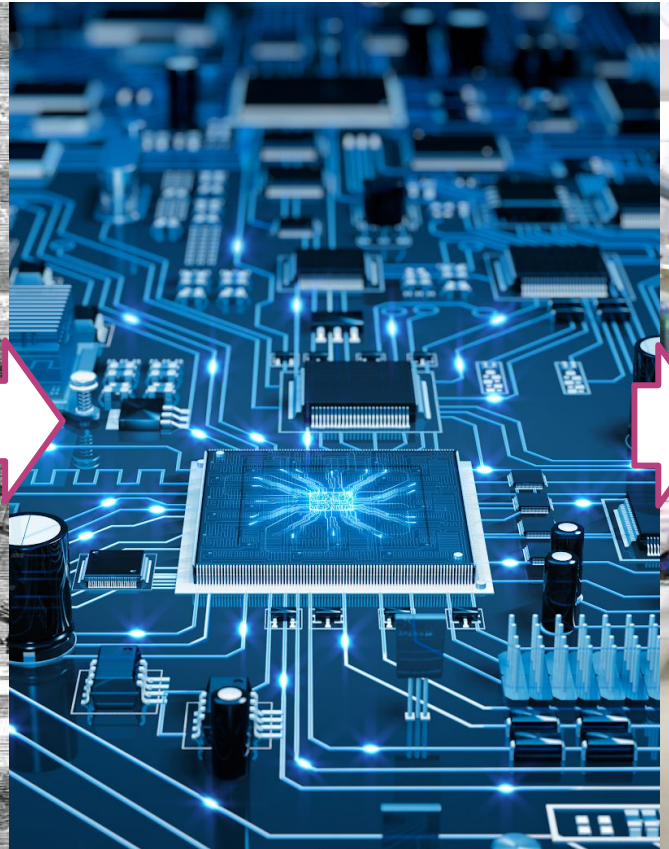


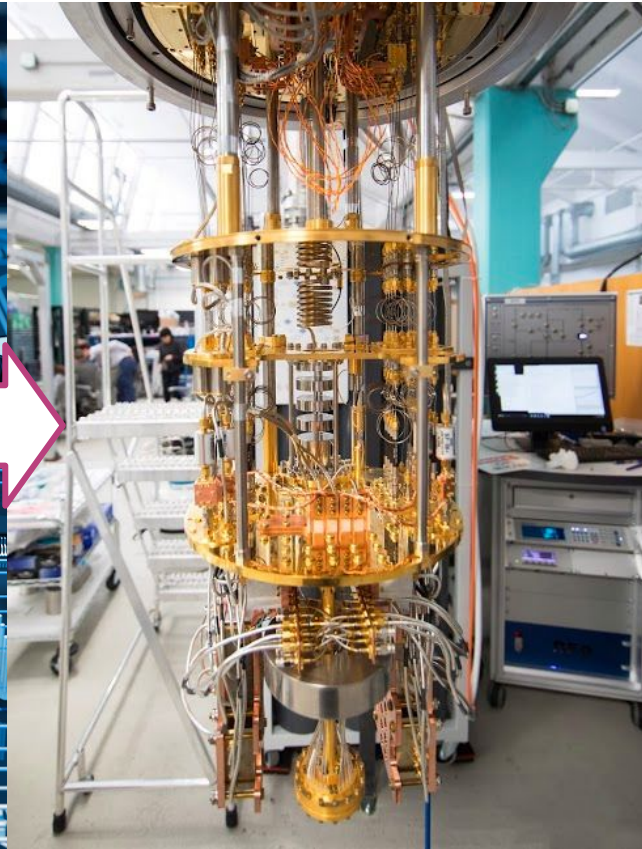


Why program a quantum computer?

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

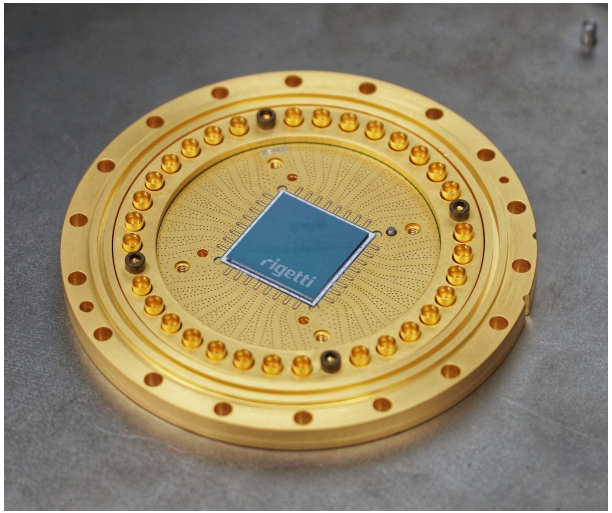
How do I program a quantum computer?

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

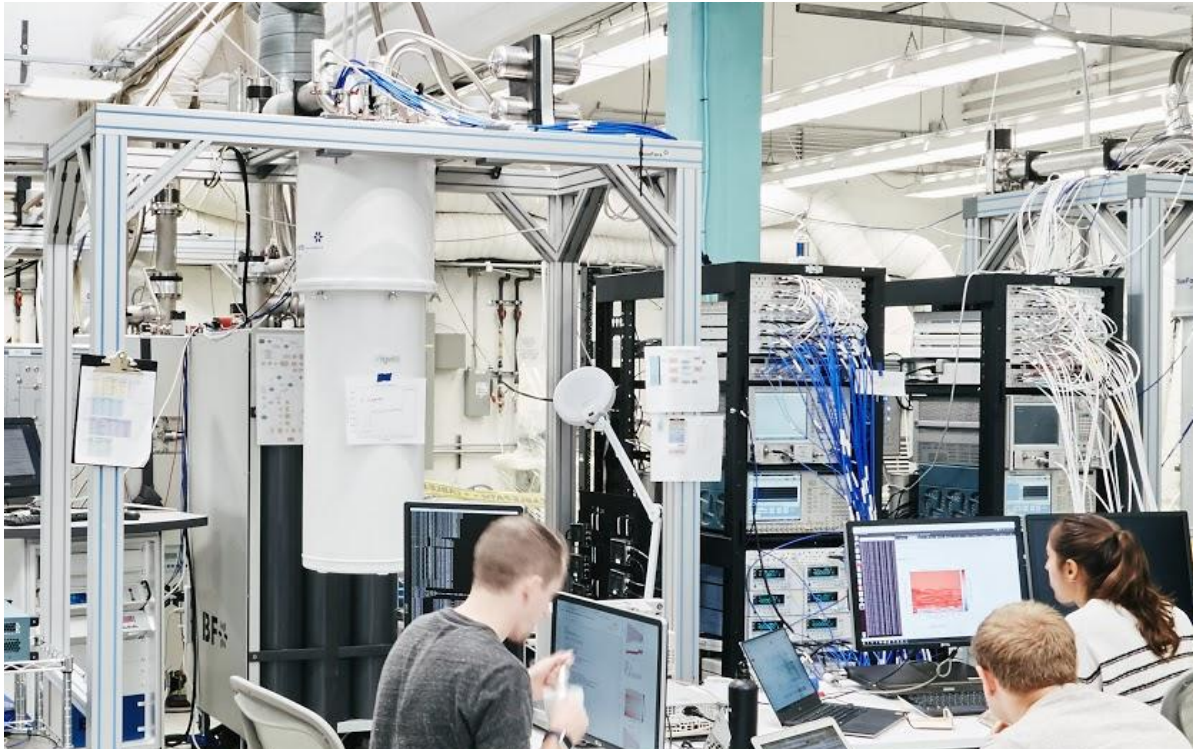
How do I program a quantum computer?

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

Quantum computers have quantum processor(s) and classical processors



Quantum processor

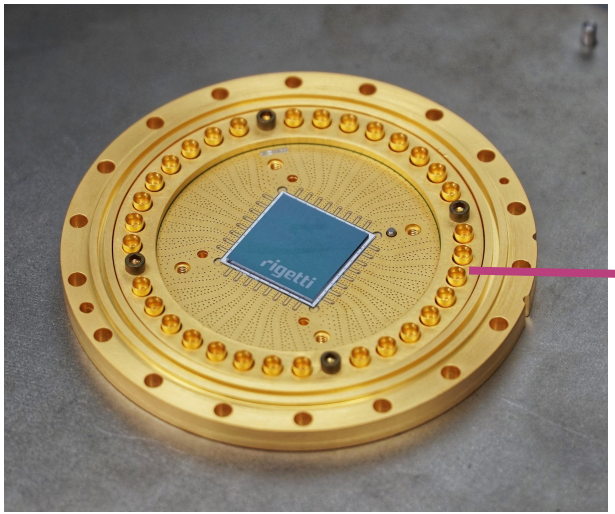


Full quantum computing system

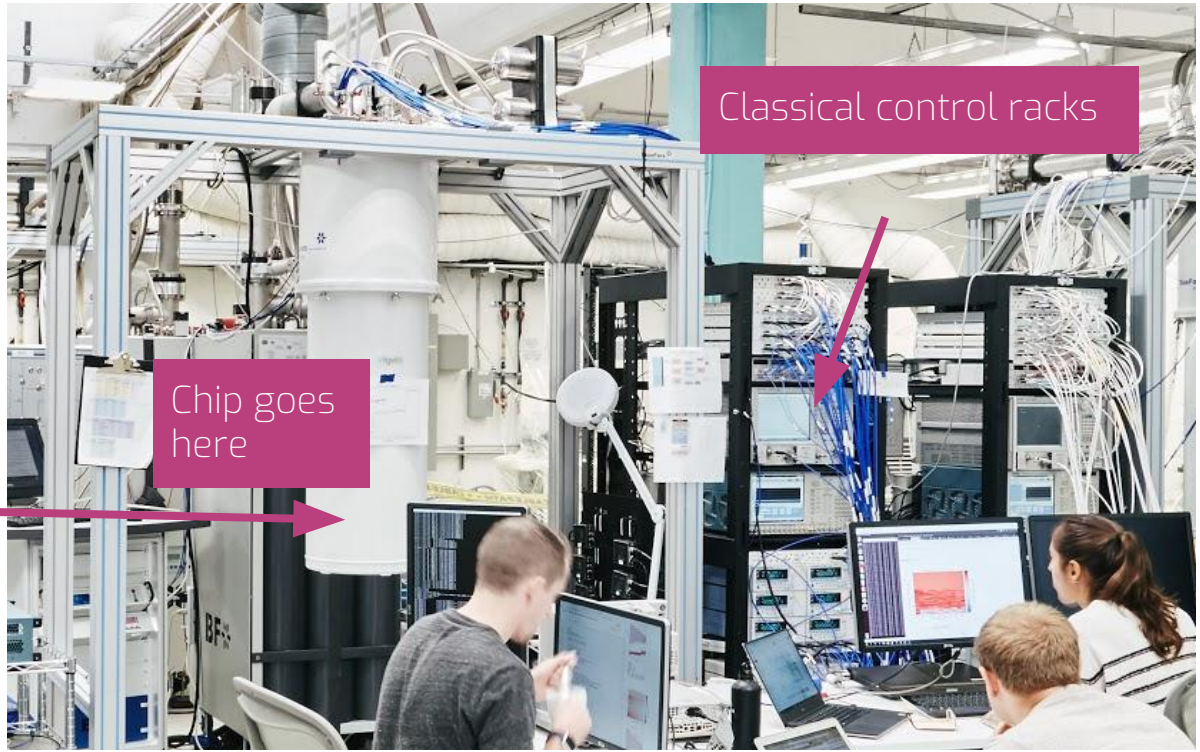
Otterbach et al. arXiv:1712.05771

How do I program a quantum computer?

Quantum computers have quantum processor(s) and classical processors



Quantum processor



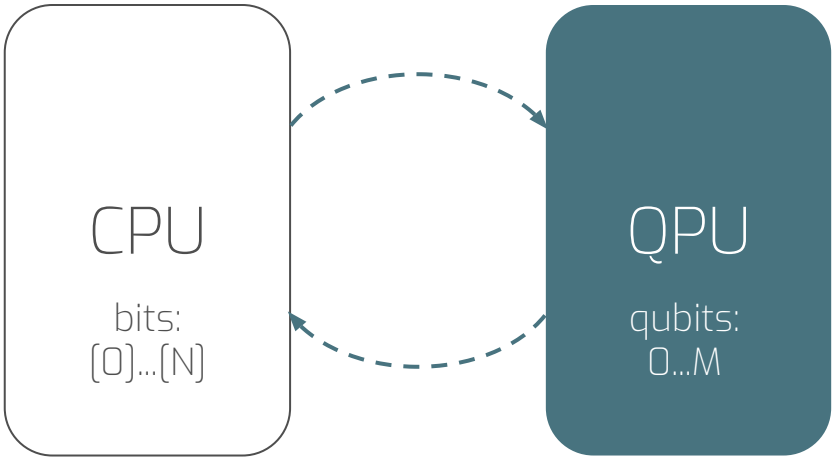
Full quantum computing system

Otterbach et al. arXiv:1712.05771

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

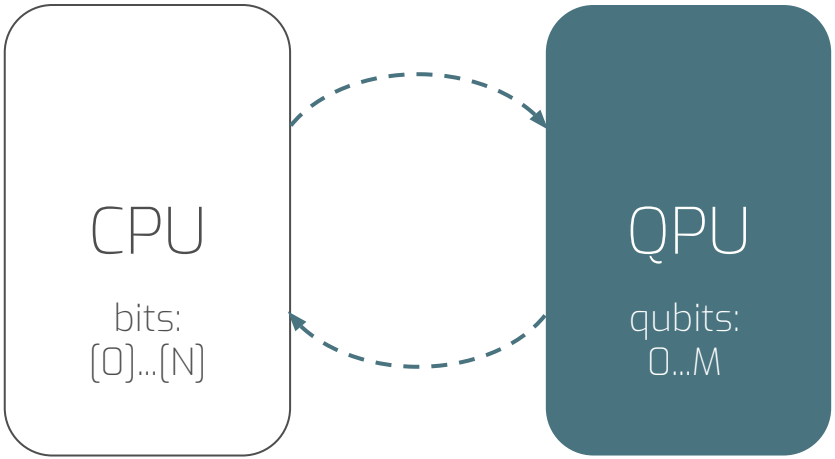


Smith, Curtis, Zeng. "A Practical Quantum Instruction Set Architecture" arXiv:1608.03355

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



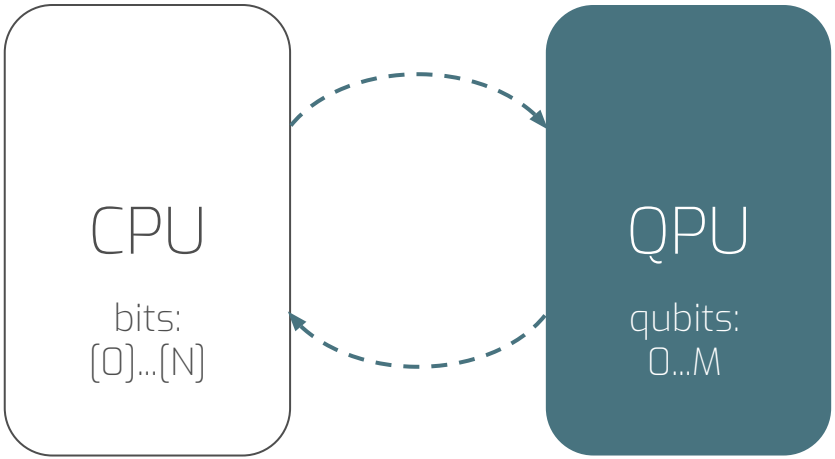
Forest is optimized for this.

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

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



Forest is optimized for this with our Quil [**01**⟩ instruction set.

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

How do I program a quantum computer?

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

How do I program a quantum computer?

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

Probability of 0

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 $b \in \{0, 1\}$	Real vector $\vec{b} = a\vec{0} + b\vec{1}$	$a + b \in \mathbb{R}_+$ $a + b = 1$

How do I program a quantum computer?

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

How do I program a quantum computer?

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

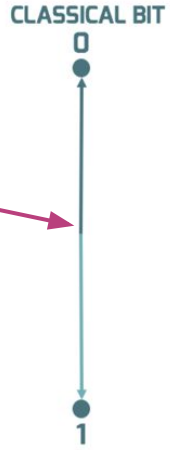
$|\alpha|^2$ = Probability of 0

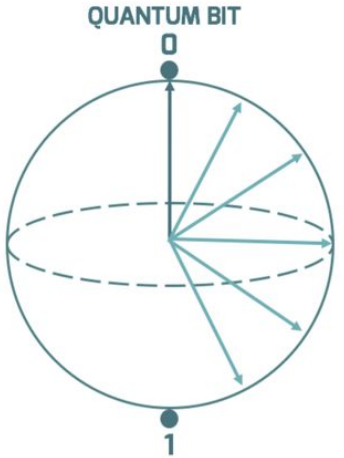
$|\beta|^2$ = Probability of 1

How do I program a quantum computer?

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

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



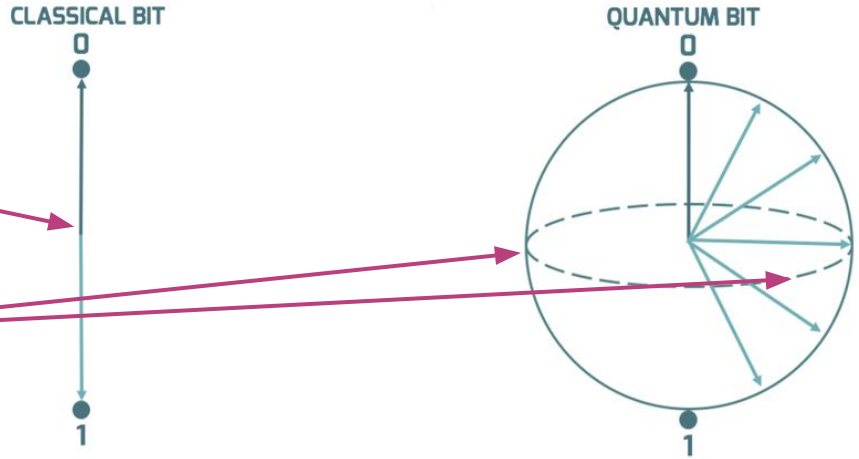


How do I program a quantum computer?

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

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

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



How do I program a quantum computer?

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

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

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

How do I program a quantum computer?

	Bits	Probabilistic Bits	Qubits
State (single unit)	Bit $b \in \{0,1\}$	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) $\vec{p} = p_{0\ldots0}, \ldots, p_x, \ldots, p_{1\ldots1}$	Wavefunction (complex vector) $\vec{\psi} = \alpha_{0\ldots0}, \ldots, \alpha_x, \ldots, \alpha_{1\ldots1}$

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

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

How do I program a quantum computer?

	Bits	Probabilistic Bits	Qubits
State (single unit)	Bit $b \in \{0,1\}$	Real vector $\vec{b} = a\vec{0} + b\vec{1}$ $a+b \in \mathbb{R}_+$ $a+b=1$	Complex vector $\vec{\psi} = \alpha\vec{0} + \beta\vec{1}$ $\alpha, \beta \in \mathbb{C}$ $\lvert\alpha\rvert^2 + \lvert\beta\rvert^2 = 1$
State (multi-unit)	Bitstring $x \in \{0,1\}^n$	Prob. Distribution (stochastic vector) $\vec{p} = p_{0\ldots0}, \ldots, p_x, \ldots, p_{1\ldots1}$	Wavefunction (complex vector) $\vec{\psi} = \alpha_{0\ldots0}, \ldots, \alpha_x, \ldots, \alpha_{1\ldots1}$

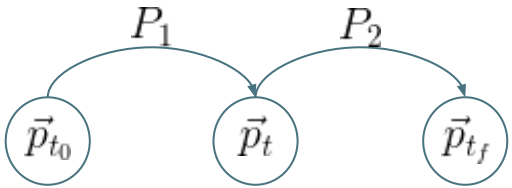
$$\vec{p} = \bigotimes_i^n b_i$$

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

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

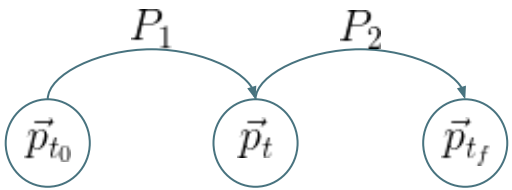
How do I program a quantum computer?

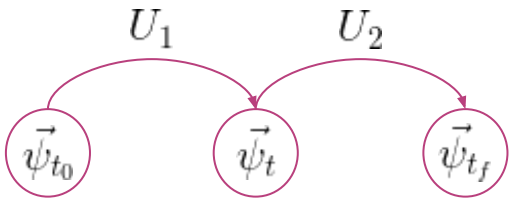
	Bits	Probabilistic Bits	Qubits
State (single unit)	Bit $b \in \{0,1\}$	Real vector $\vec{b} = a\vec{0} + b\vec{1}$ $a+b \in \mathbb{R}_+$ $a+b=1$	Complex vector $\vec{\psi} = \alpha\vec{0} + \beta\vec{1}$ $\alpha, \beta \in \mathbb{C}$ $\lvert\alpha\rvert^2 + \lvert\beta\rvert^2 = 1$
State (multi-unit)	Bitstring $x \in \{0,1\}^n$	Prob. Distribution (stochastic vector) $\vec{p} = p_{0\ldots0}, \ldots, p_x, \ldots, p_{1\ldots1}$	Wavefunction (complex vector) $\vec{\psi} = \alpha_{0\ldots0}, \ldots, \alpha_x, \ldots, \alpha_{1\ldots1}$
Operations	Boolean Logic	Stochastic Matrices $\sum_{j=1}^{S} P_{i,j} = 1.$	



How do I program a quantum computer?

	Bits	Probabilistic Bits	Qubits
State (single unit)	Bit $b \in \{0,1\}$	Real vector $\vec{b} = a\vec{0} + b\vec{1}$ $a+b \in \mathbb{R}_+$ $a+b=1$	Complex vector $\vec{\psi} = \alpha\vec{0} + \beta\vec{1}$ $\alpha, \beta \in \mathbb{C}$ $\lvert\alpha\rvert^2 + \lvert\beta\rvert^2 = 1$
State (multi-unit)	Bitstring $x \in \{0,1\}^n$	Prob. Distribution (stochastic vector) $\vec{p} = p_{0\ldots0}, \ldots, p_x, \ldots, p_{1\ldots1}$	Wavefunction (complex vector) $\vec{\psi} = \alpha_{0\ldots0}, \ldots, \alpha_x, \ldots, \alpha_{1\ldots1}$
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?

	Bits	Probabilistic Bits	Qubits
State (single unit)	Bit $b \in \{0,1\}$	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) $\vec{p} = p_{0\ldots0}, \ldots, p_x, \ldots, p_{1\ldots1}$	Wavefunction (complex vector) $\vec{\psi} = \alpha_{0\ldots0}, \ldots, \alpha_x, \ldots, \alpha_{1\ldots1}$
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?

	Bits	Probabilistic Bits	Qubits
State (single unit)	Bit $b \in \{0,1\}$	Real vector $\vec{b} = a\vec{0} + b\vec{1}$ $a+b \in \mathbb{R}_+$ $a+b=1$	Complex vector $\vec{\psi} = \alpha\vec{0} + \beta\vec{1}$ $\alpha, \beta \in \mathbb{C}$ $\lvert\alpha\rvert^2 + \lvert\beta\rvert^2 = 1$
State (multi-unit)	Bitstring $x \in \{0,1\}^n$	Prob. Distribution (stochastic vector) $\vec{p} = p_{0\ldots0}, \ldots, p_x, \ldots, p_{1\ldots1}$	Wavefunction (complex vector) $\vec{\psi} = \alpha_{0\ldots0}, \ldots, \alpha_x, \ldots, \alpha_{1\ldots1}$
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

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

How do I program a quantum computer?

	Bits	Probabilistic Bits	Qubits
State (single unit)	Bit $b \in \{0,1\}$	Real vector $\vec{b} = a\vec{0} + b\vec{1}$ $a+b \in \mathbb{R}_+$ $a+b=1$	Complex vector $\vec{\psi} = \alpha\vec{0} + \beta\vec{1}$ $\alpha, \beta \in \mathbb{C}$ $\lvert\alpha\rvert^2 + \lvert\beta\rvert^2 = 1$
State (multi-unit)	Bitstring $x \in \{0,1\}^n$	Prob. Distribution (stochastic vector) $\vec{p} = p_{0\ldots0}, \ldots, p_x, \ldots, p_{1\ldots1}$	Wavefunction (complex vector) $\vec{\psi} = \alpha_{0\ldots0}, \ldots, \alpha_x, \ldots, \alpha_{1\ldots1}$
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?

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

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

Start in 0 →

Ψ = [1, 0, 0, 0]
00 01 10 11

```
X 0 # “quantum NOT”
```

How do I program a quantum computer?

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

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

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

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

```
X 0 # “quantum NOT”
```

Apply X instr to 0th qubit →

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

How do I program a quantum computer?

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

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

```
X 0 # “quantum NOT”
```

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

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

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

```
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>
```

```
X 0 # “quantum NOT”
```

Ψ = [1, 0, 0, 0]
00 01 10 11

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

Ψ = [0, 1, 0, 0]
00 01 10 11

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

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

Apply H instr to 0th qubit →

Ψ = [1/√2, 1/√2, 0, 0]
00 01 10 11

How do I program a quantum computer?

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

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

```
X 0 # “quantum NOT”
```

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

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

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

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

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

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

```
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>
```

```
X 0 # “quantum NOT”
```

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

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

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

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

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

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

```
CNOT 0 1
```

Apply CNOT instr to 0 and 1 qubits →

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

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

How do I program a quantum computer?

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

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

```
X 0 # “quantum NOT”
```

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

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

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

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

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

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

```
CNOT 0 1
```

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

Qubits 0 and 1 are ENTANGLED

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

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>

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

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

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
```

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

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

MEASURE 0 [2]
```

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

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

MEASURE 0 [2]
```

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

50-50 branch

$\Psi = [1, 0, 0, 0]$ (00, 01, 10, 11)

$\Psi = [0, 0, 0, 1]$ (00, 01, 10, 11)

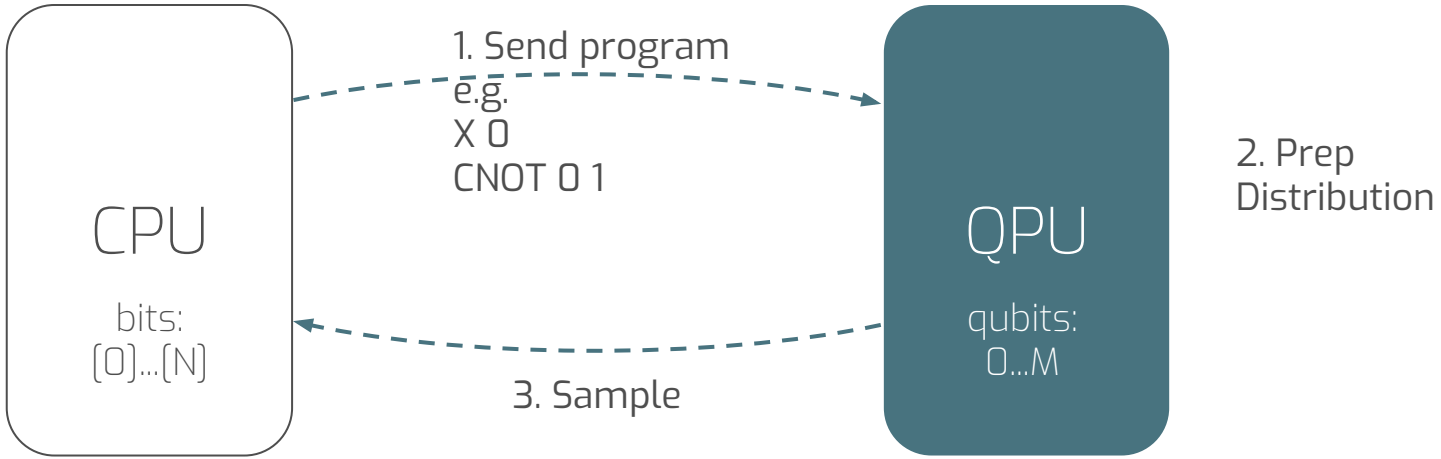
Classical Bit Register

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

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

How do I program a quantum computer?

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]
.
.
.
```

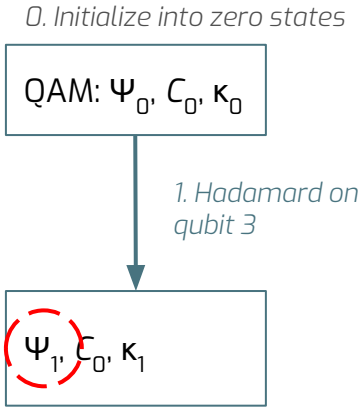
The Quil Programming Model

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)




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

The Quil Programming Model

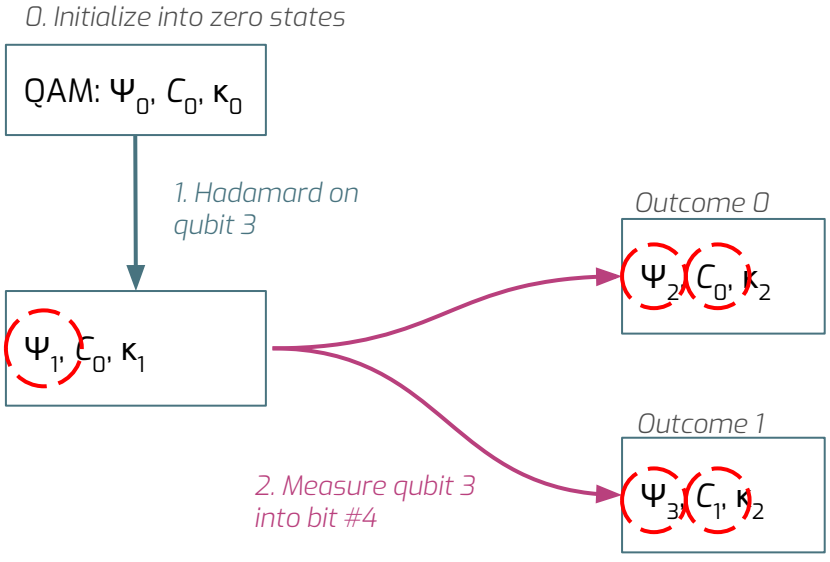
Targets a **Quantum Abstract Machine (QAM)** with a syntax for representing state transitions

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]
.
.
.
```



The Quil Programming Model

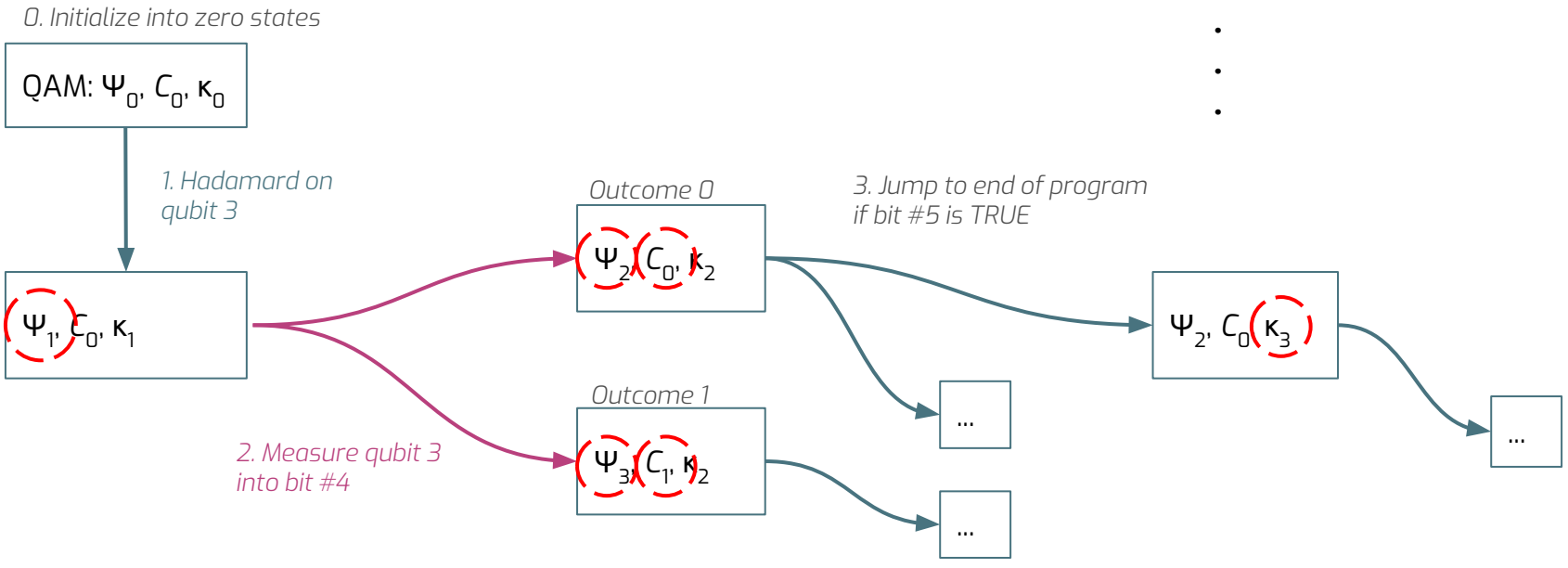
Targets a **Quantum Abstract Machine (QAM)** with a syntax for representing state transitions

Ψ: Quantum state (qubits) → quantum instructions

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

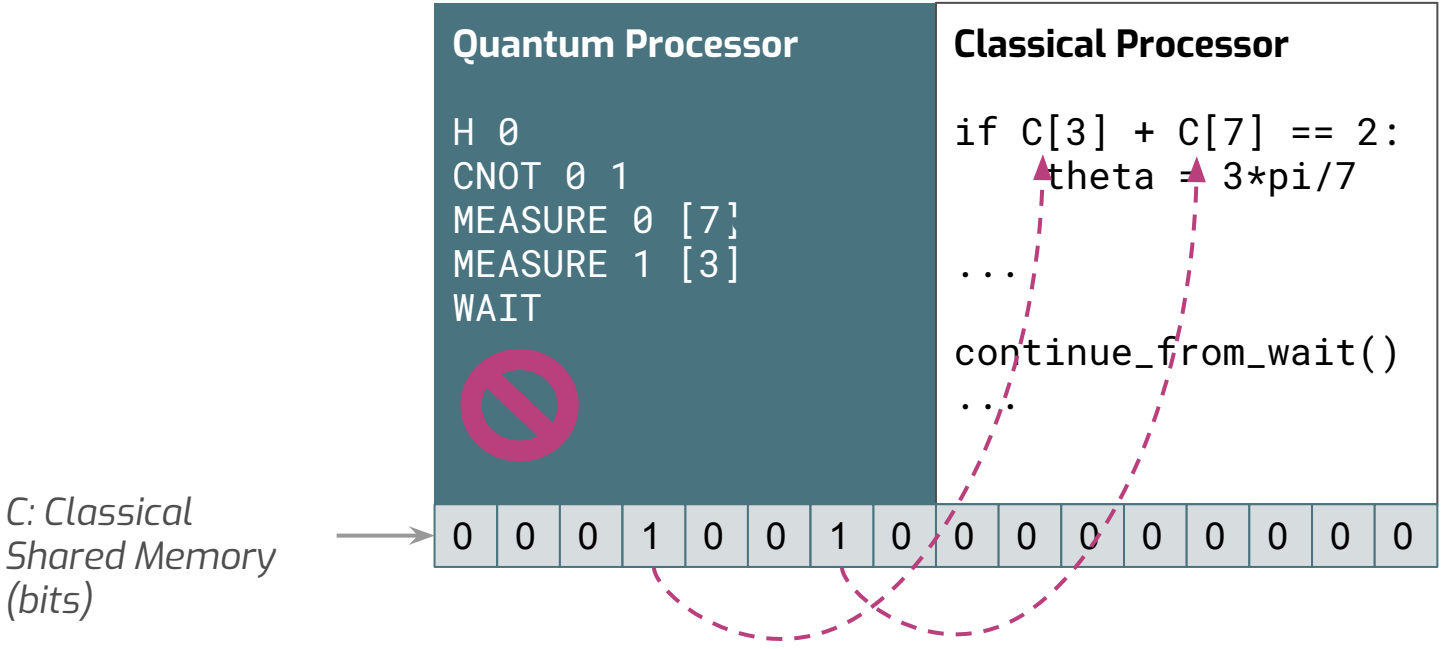
κ: Execution state (program)→ control instructions (e.g., jumps)

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



The Quil Programming Model

Quantum/classical synchronization with **shared classical state**.



Forest

A Hybrid Cloud Quantum Programming Environment

Cloud Compute Backends

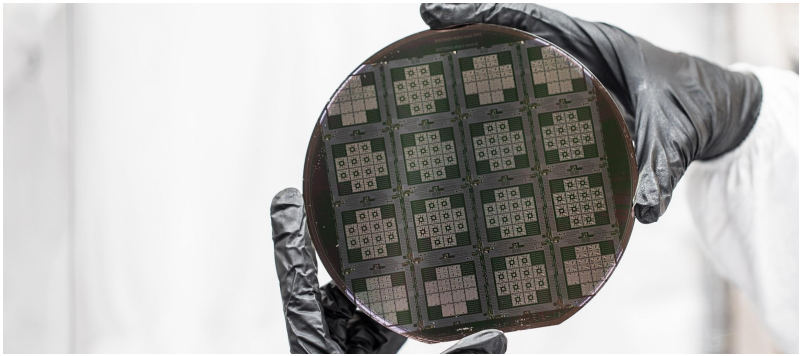
- 19Q
- Quantum Virtual Machine

Superconducting Processors & Simulators

Software Dev Kit

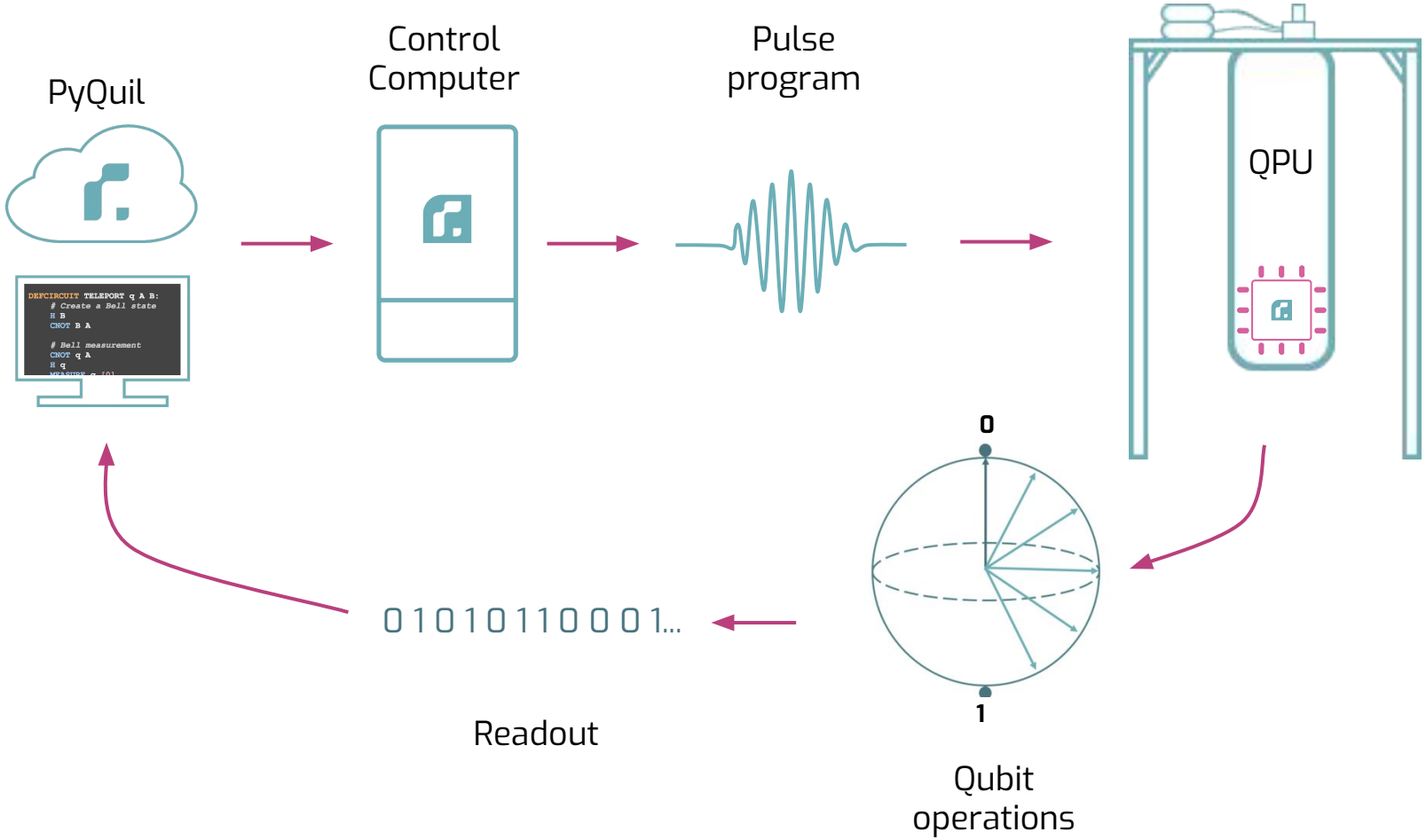
- Quil Instr. Set
- pyQuil
- grove (VQE, QAOA, etc ...)

Open Source Python libraries



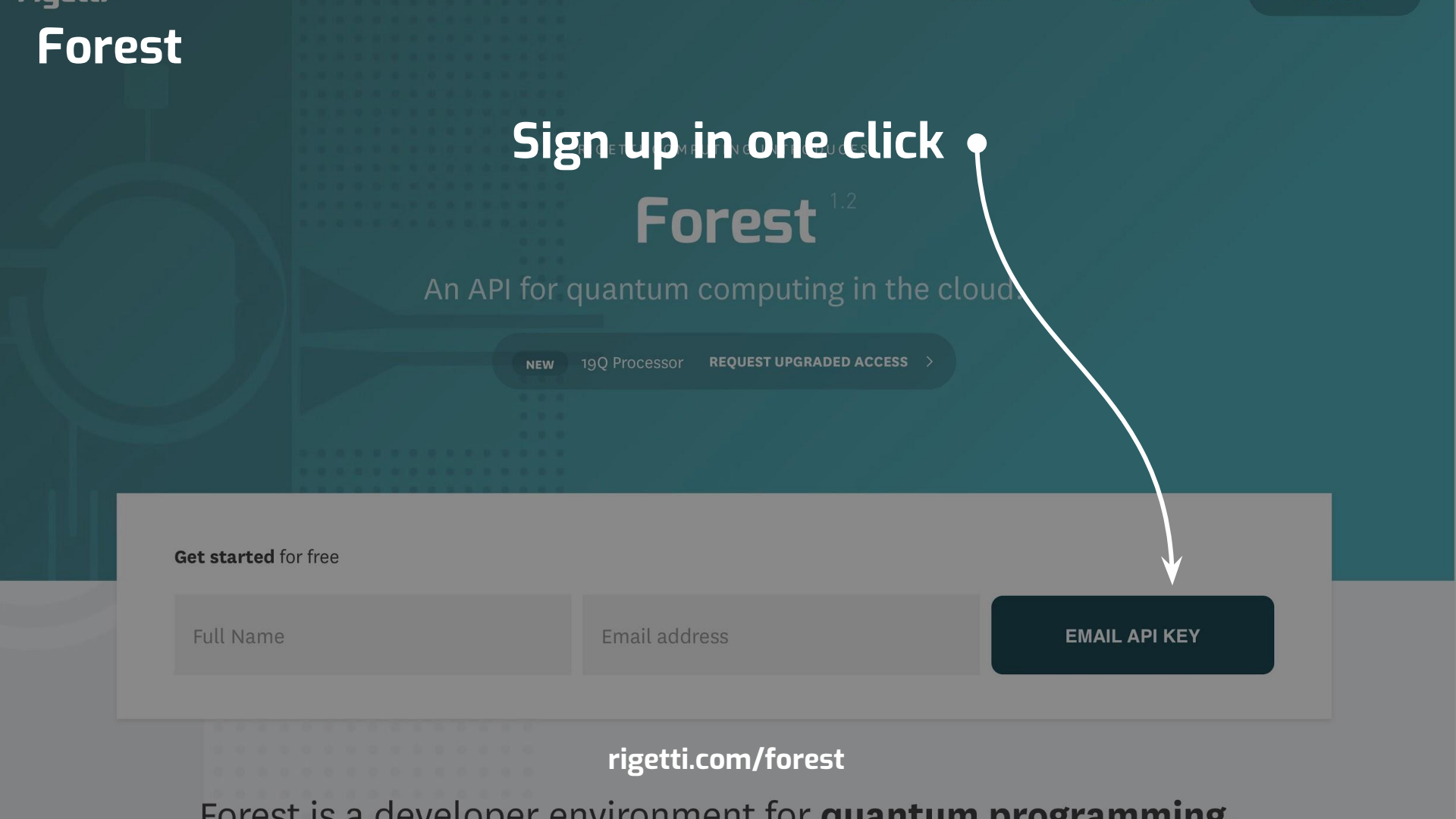
Join us on Github

github.com/rigetticomputing


PyQuil
Control Computer
Pulse program
QPU
DEFCIRCUIT TELEPORT q A B:
Create a Bell state
H B
CNOT B A
Bell measurement
CNOT q A
H q
0
1
0 1 0 1 0 1 1 0 0 0 1...
Readout
Qubit operations

Forest

Sign up in one click



rigetti.com/forest

Forest is a developer environment for **quantum programming**

How do I program a quantum computer?

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

We need hybrid programming because of errors

Chance of hardware error in a classical computer:

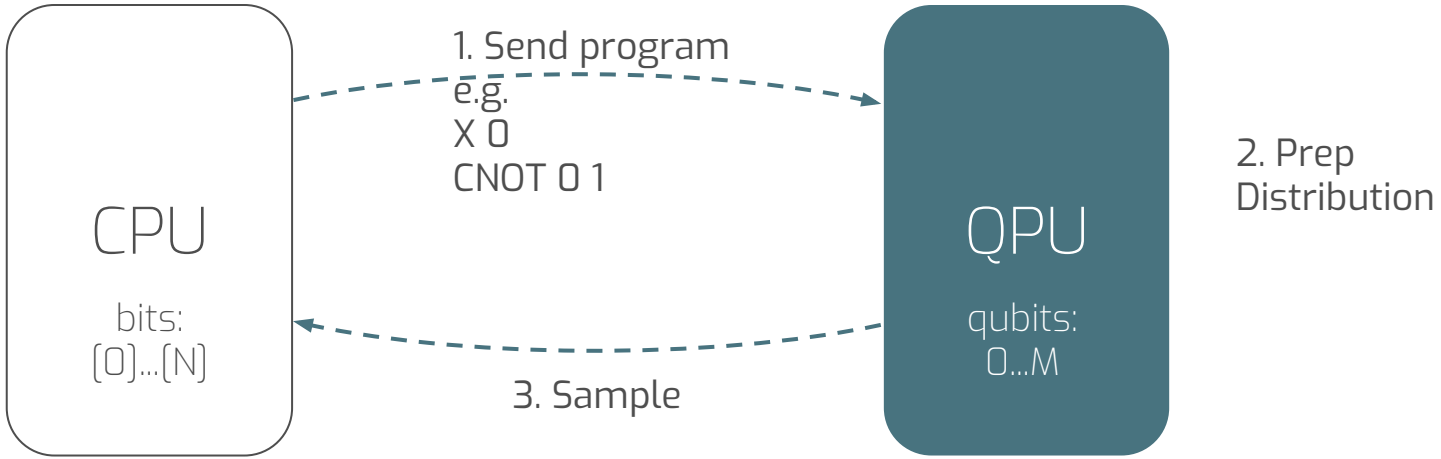
0.000,000,000,000,000,000,000,1 %

Chance of hardware error in a quantum computer:

0.1%

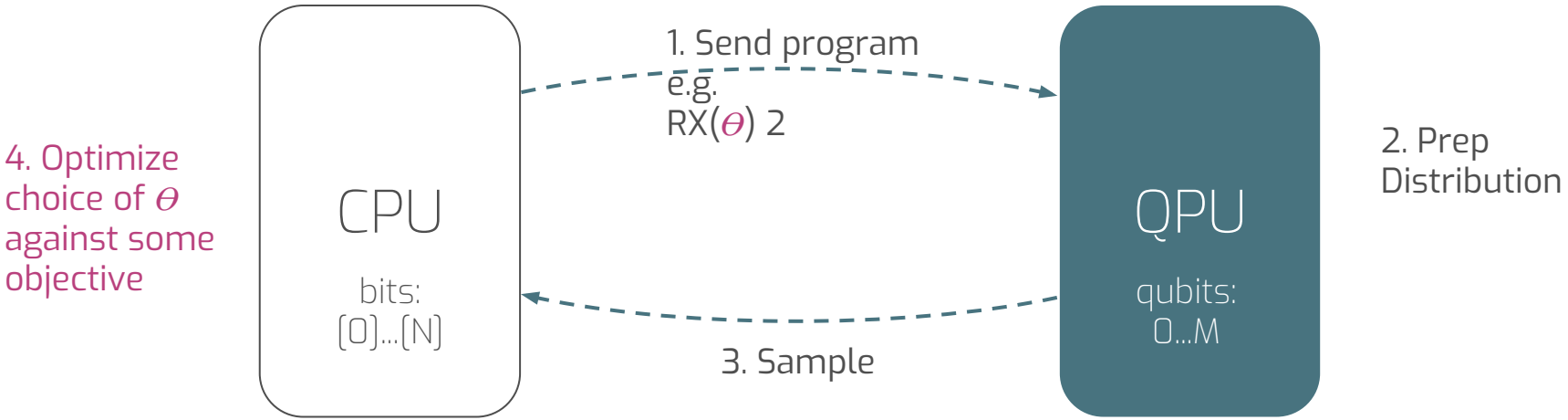
How do I program a quantum computer?

Quantum programming is preparing and sampling from complicated distributions



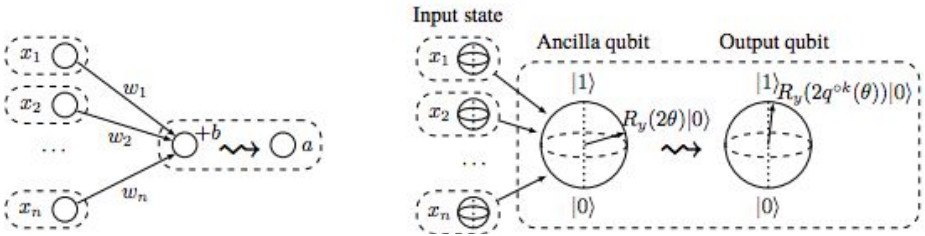
How do I program a quantum computer?

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



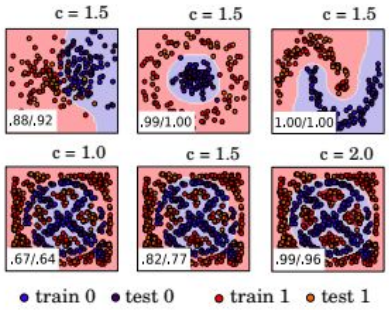
Quantum Machine Learning

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

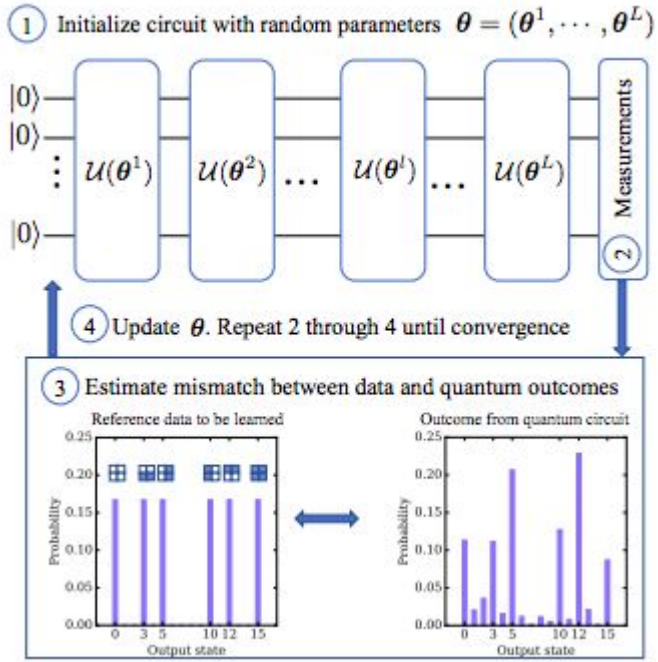


> Quantum circuit learning. (Mitarai et al. 2018)

> 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)



The Variational Quantum Eigensolver

Used for the electronic structure problem in quantum chemistry

1. MOLECULAR DESCRIPTION

e.g. Electronic Structure Hamiltonian

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

2. MAP TO QUBIT REPRESENTATION

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

e.g. DI-HYDROGEN

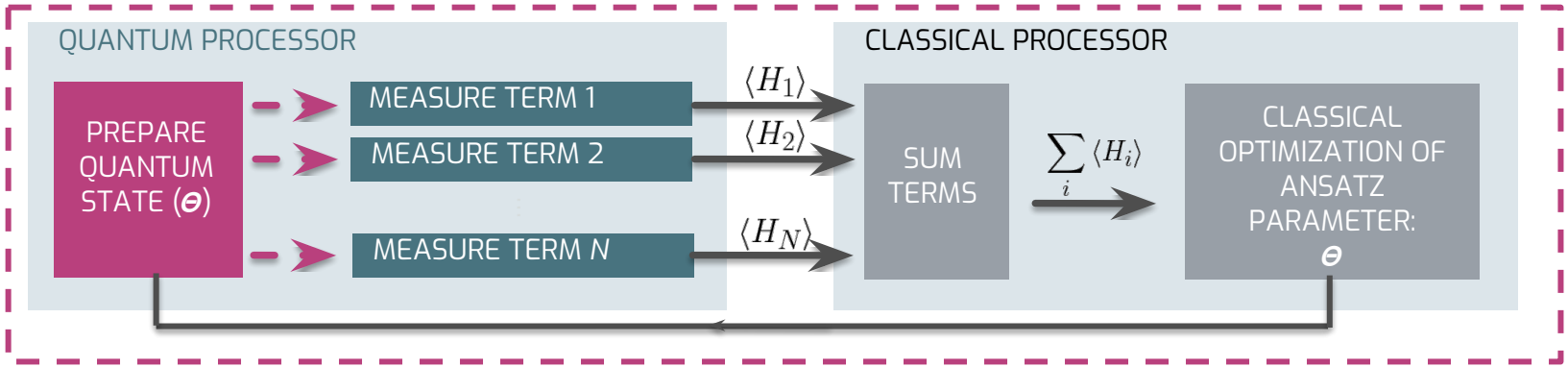
$$\begin{aligned} H = {} & f_0 \mathbb{1} + f_1 Z_0 + f_2 Z_1 + f_3 Z_2 + f_1 Z_0 Z_1 \\ & + f_4 Z_0 Z_2 + f_5 Z_1 Z_3 + f_6 X_0 Z_1 X_2 + f_6 Y_0 Z_1 Y_2 \\ & + f_7 Z_0 Z_1 Z_2 + f_4 Z_0 Z_2 Z_3 + f_3 Z_1 Z_2 Z_3 \\ & + f_6 X_0 Z_1 X_2 Z_3 + f_6 Y_0 Z_1 Y_2 Z_3 + f_7 Z_0 Z_1 Z_2 Z_3 \end{aligned}$$

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



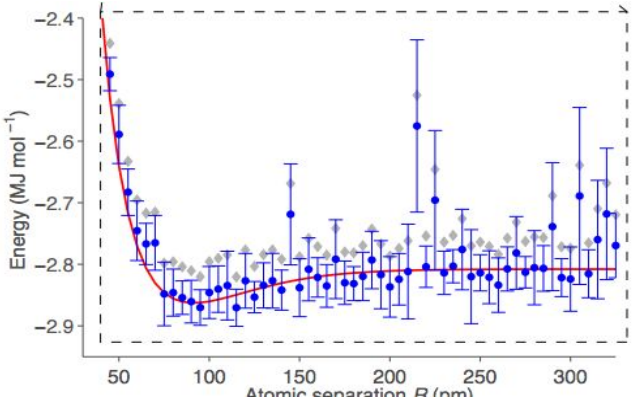
O'Malley, P. J. J., et al. (2015). Scalable Quantum Simulation of Molecular Energies. *arXiv:1512.06860*.

Wecker, D., et al. (2015). Progress towards practical quantum variational algorithms. *Physical Review A*, *92*(4), 042303.

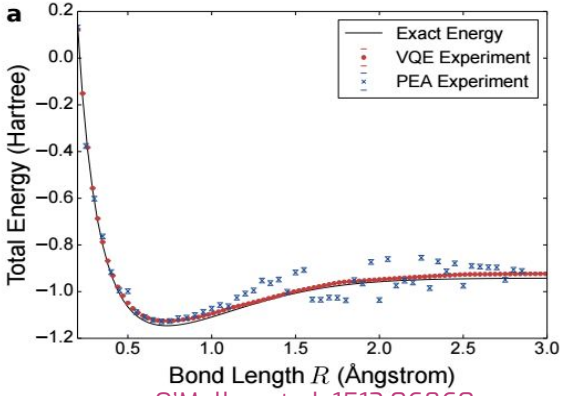
McClean, J. R. et al. (2015). The theory of variational hybrid quantum-classical algorithms. *arXiv:1509.04279*.

Peruzzo, A., et al. (2014). A variational eigenvalue solver on a photonic quantum processor. *Nature communications*, *5*.

VQE Simulations on Quantum Hardware

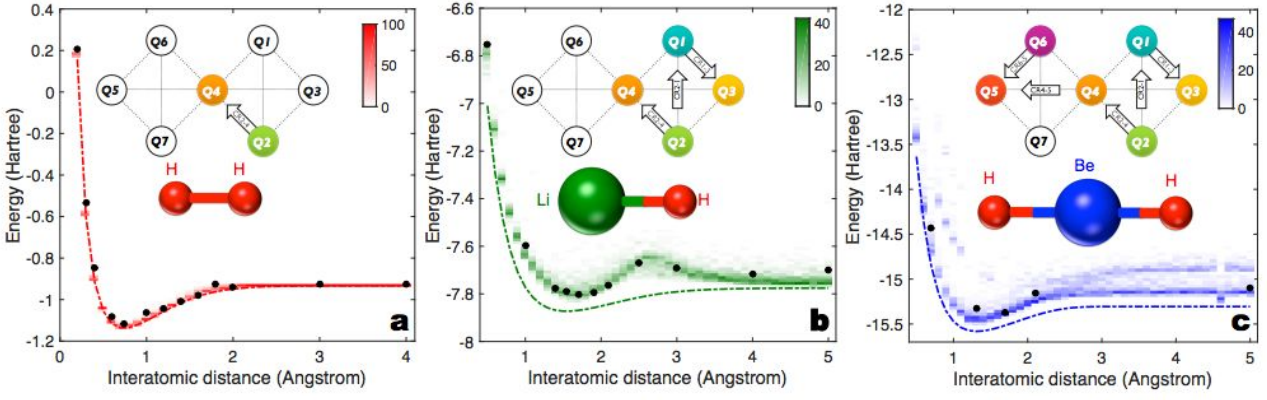


Peruzzo et al. 1304.3061



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

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

Traveling Salesperson Scheduling

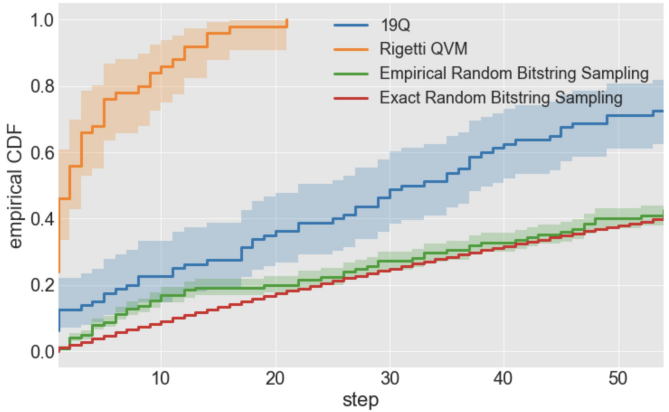
Hadfield et al. 2017 (1709.03489)

K-means clustering

Otterbach et al. 2017 (1712.05771)

Boltzmann Machine Training

Verdon et al. 2017 (1712.05304)



QAOA in **Forest**

In ***14*** lines of code

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

graph = [(0, 1), (1, 2), (2, 3)]
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())

program = init_state_prog + qaoa_ansatz([0., 0.5], [0.75, 1.])

qvm = QPUConnection()
qvm.run_and_measure(program, qubits=nodes, trials=10)
```

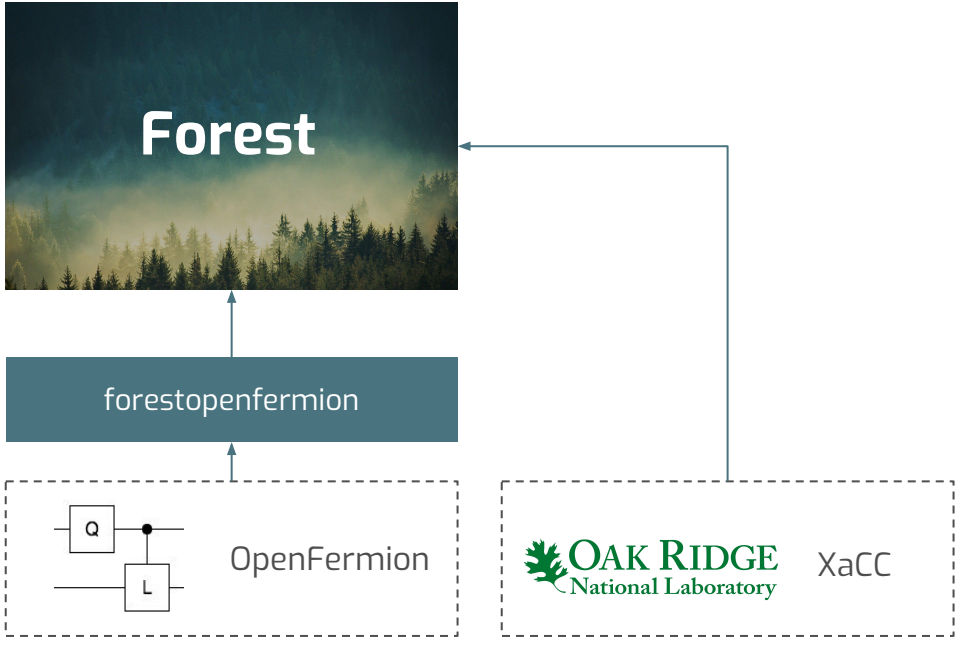
Open areas in quantum programming

> Debuggers

> Optimizing compilers

> Application specific packages

> **Adoption and implementations**



Q1. Why program a quantum computer?

New power | New opportunity | Fundamental curiosity

Q2. How do I program a quantum computer?

Hybrid quantum programming using Forest and Quil