



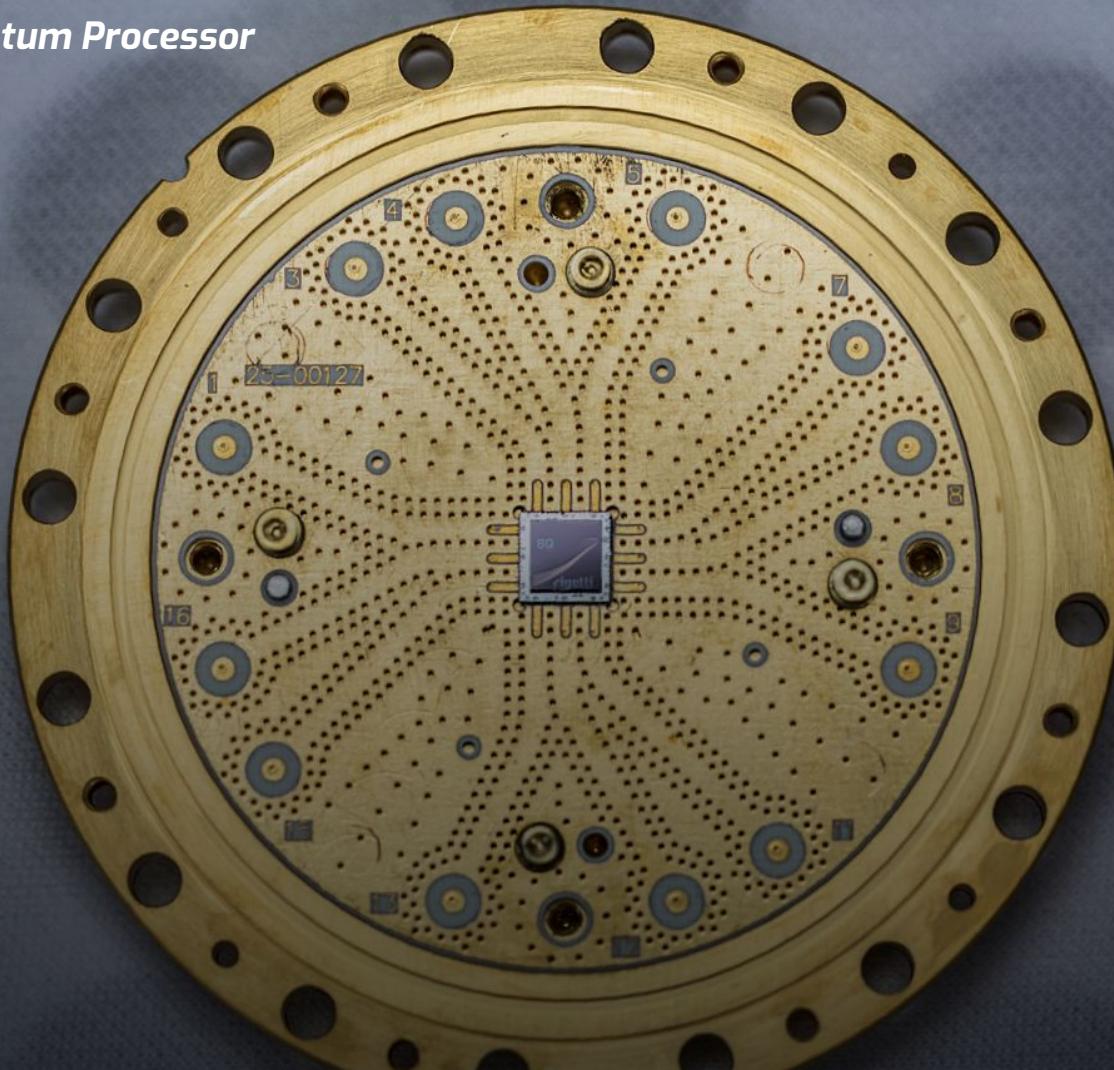
rigetti

Quantum Cloud Computing
Johannes Otterbach

TU Kaiserslautern
January 22, 2018



Rigetti 8-Qubit Quantum Processor



- > **Scalable Gate-model Quantum Processors**
 - > Superconducting Microwave Circuits
- > **Focus on near-term applications**
 - > Quantum/Classical Hybrid Algorithms
- > **Build towards fault-tolerance**
- > **Access over the cloud**
 - > Quantum computers as co-processors

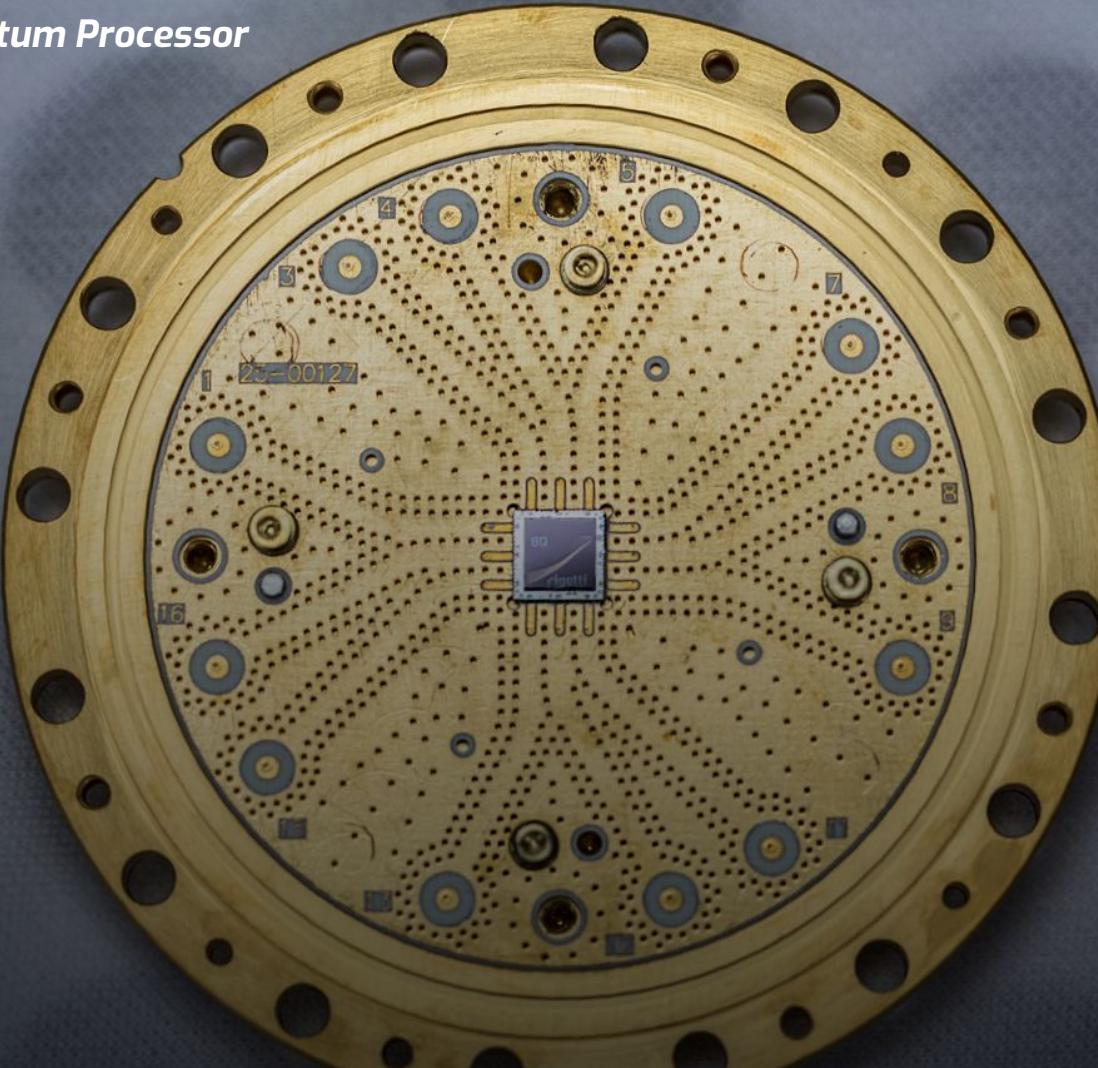
Rigetti 8-Qubit Quantum Processor

Founded in 2013 by
Chad Rigetti

Fab-1: Fremont, CA

~100 Employees

Forest: Quantum
computing over the
cloud



Venture back
startu

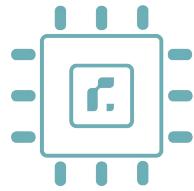
R&D lab: Berkeley, C

Full-stack scalab
superconducti
qubi

30-qubit Quantu
Virtual Machine



Full Stack Quantum Computing



Quantum
Processor



Hardware



Cloud based Quantum
Operating System



Applications



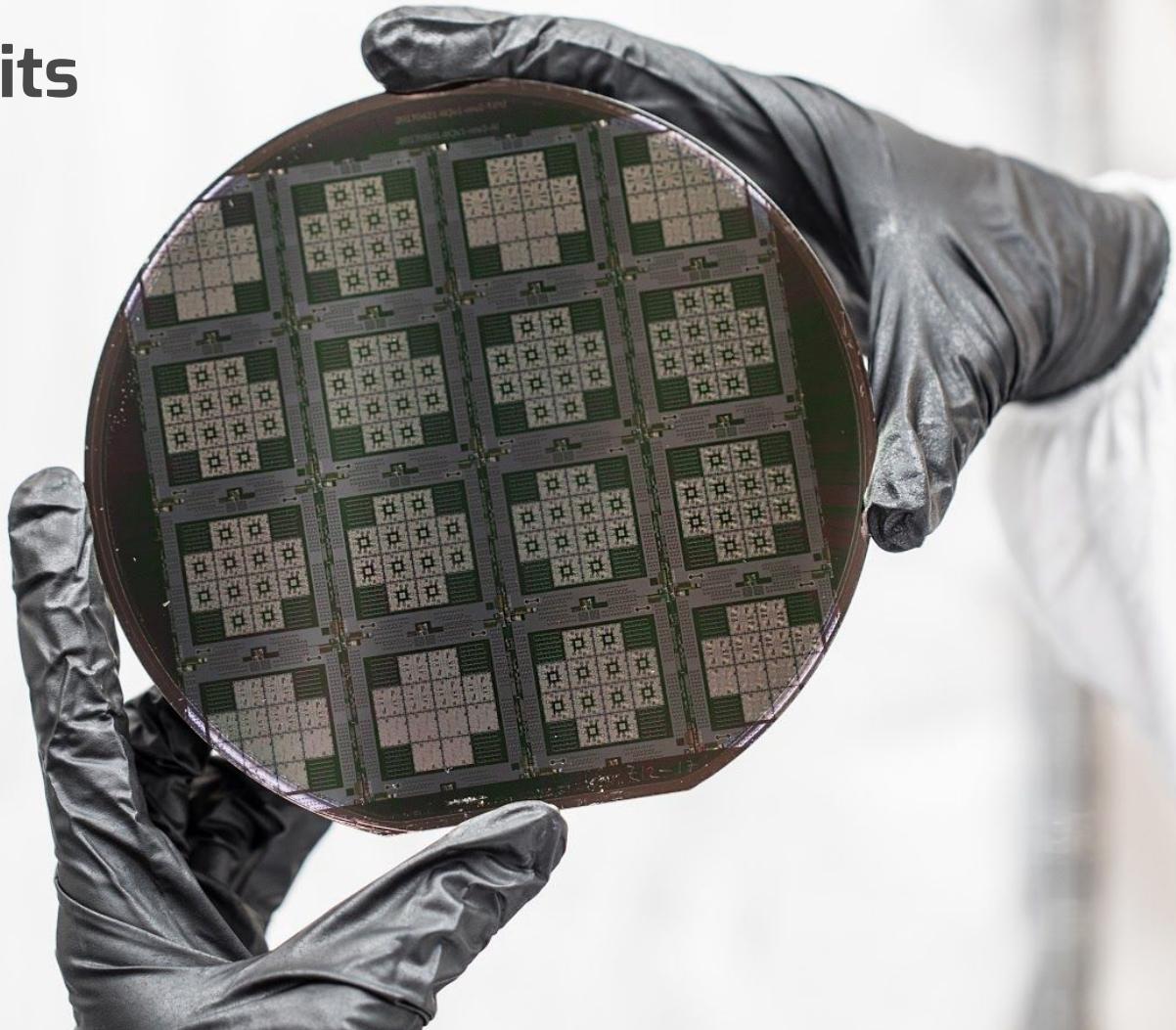
What do our **Qubits** look like ?

- Superconducting circuits

- Operated near absolute zero

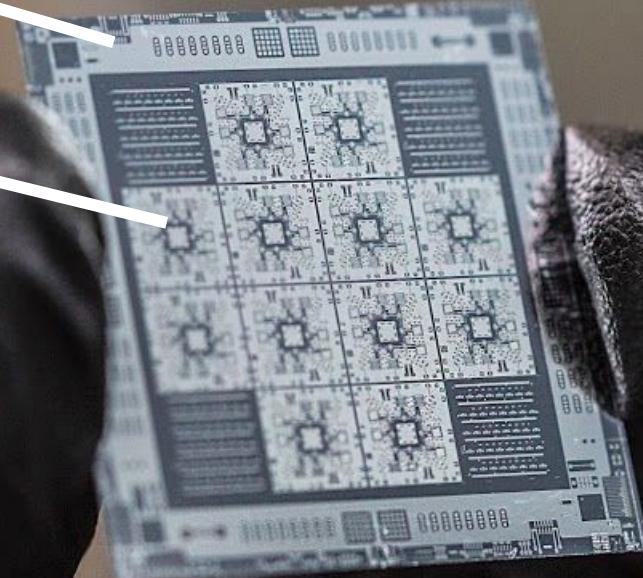
- Aluminum on silicon

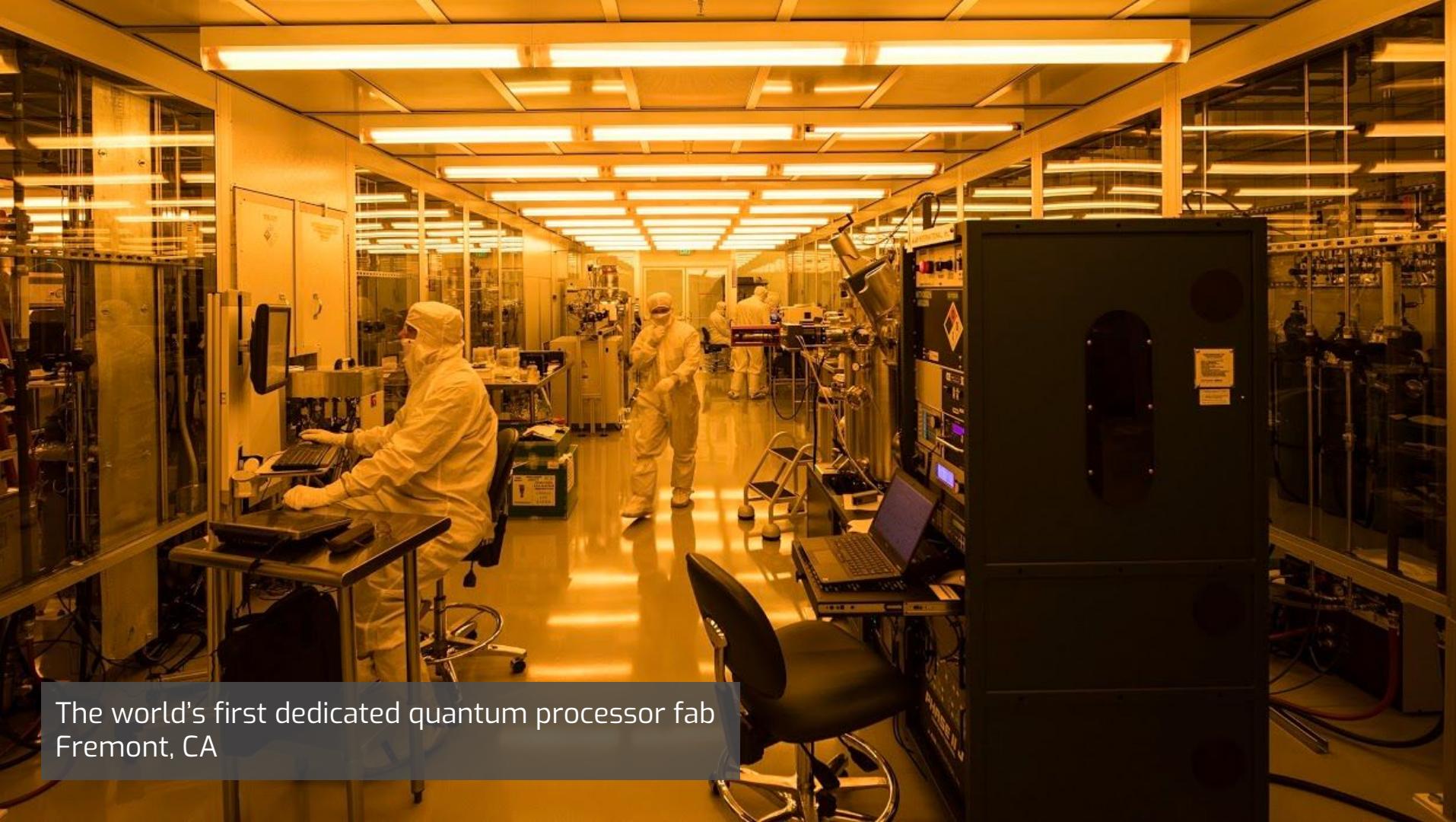
- Microwave signal delivery



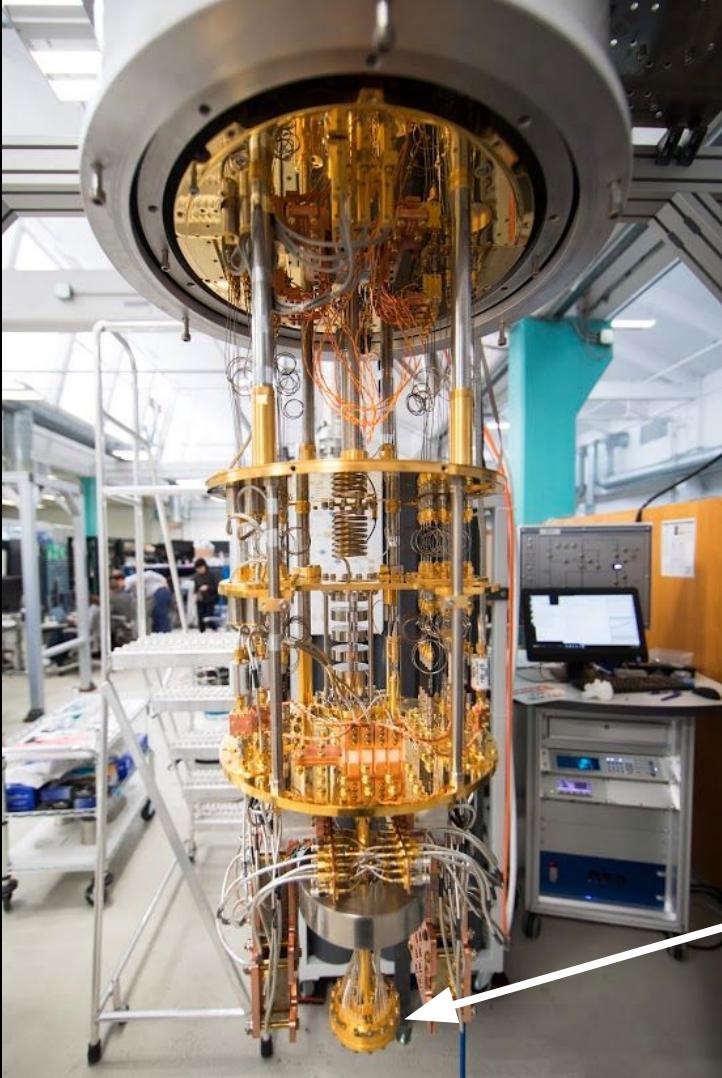
die

chip



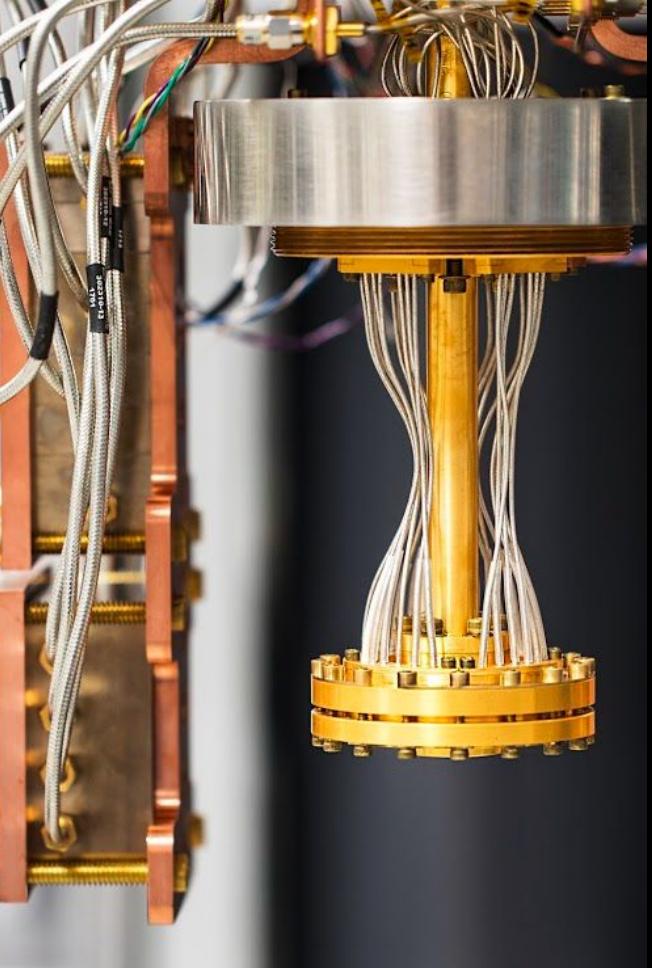


The world's first dedicated quantum processor fab
Fremont, CA



Signal Delivery

The Chip



> Packaging



> Cooling



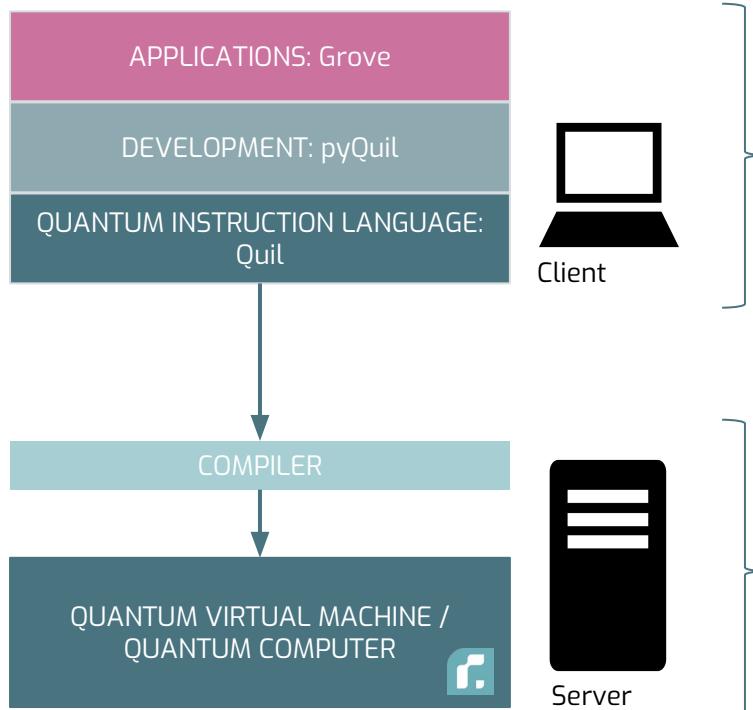
> Control Electronics

FOREST: Tools for experimental quantum programming

forest.rigetti.com

- > Write applications...
- > using tools...
- > that build quantum programs...

- > that compile onto quantum hardware...
- > that execute on a real or virtual quantum processor.



**Open-sourced on github
under Apache v2.0 license**

github.com/rigetticomputing/pyquil

github.com/rigetticomputing/grove

**Simulator in public-beta
Quantum HW**

forest.rigetti.com



Quantum Algorithms

Simulating quantum systems

1981

Original proposal by Feynman:
Simulating physics with computers

“The physical world is quantum mechanical, and therefore the proper problem is the simulation of quantum physics.”

1990

Quantum Algorithms

1994

Shor's factoring algorithm (Shor)

1995

Phase Estimation (PE) introduced (Kitaev)

1997

Hamiltonian Simulation by PE (Lloyd)

2002

Map of fermions to paulis (Somma)

2005

Molecular ground states w/ PE (Aspuru-Guzik)

2010

H₂ ground state using simulated QC (Whitfield)

2013

Hybrid Quantum-Classical Algorithms VQE (McClean)

2015

Approximate Combinatorial Optimization (Farhi)

TODAY

Simulating quantum systems

1981

Original proposal by Feynman:
Simulating physics with computers

"The physical world is quantum mechanical, and therefore the proper problem is the simulation of quantum physics."

1990

Quantum Algorithms

1994

Shor's factoring algorithm (Shor)

1995

Phase Estimation (PE) introduced (Kitaev)

1997

Hamiltonian Simulation by PE (Lloyd)

2002

Map of fermions to paulis (Somma)

2005

Molecular ground states w/ PE (Aspuru-Guzik)

2010

H₂ ground state using simulated QC (Whitfield)

2013

Hybrid Quantum-Classical Algorithms VQE (McClean)

2015

Approximate Combinatorial Optimization (Farhi)

2013

2013

Molecular ground states w/ VQE
Implementation on a photonics processor
(Peruzzo)

2014

Quantum Combinatorial Optimization (QAOA)
(Farhi), eg. MAX-CUT, MaxE3Lin2

2015

VQE + PE on superconducting qubits,
Small quantum machine learning examples

2016

Broader applications of VQE (Troyer, **Rubin**)

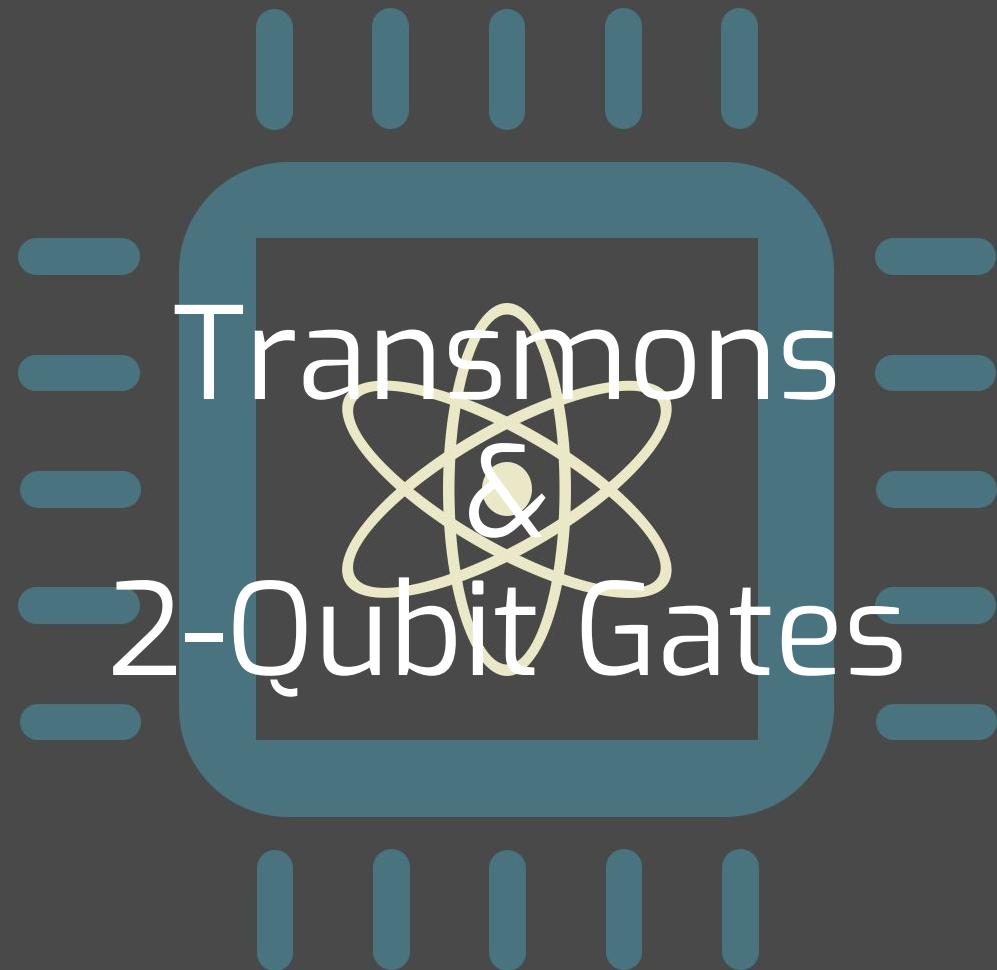
2017

Machine Learning (Aspuru-Guzik, **Otterbach**)

TODAY

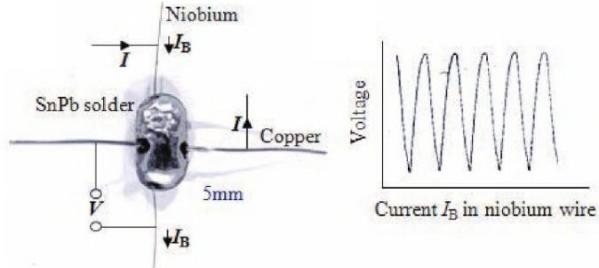
TODAY

Simulating quantum systems



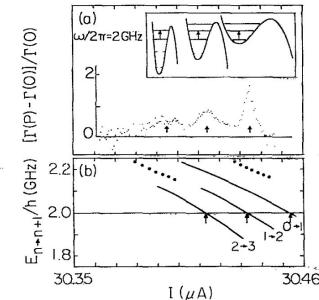
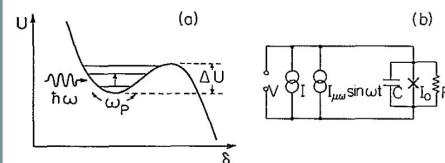
Early years of superconducting qubits

Josephson junctions



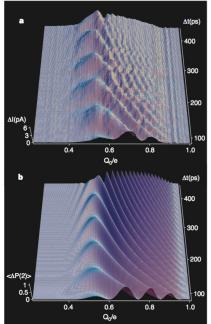
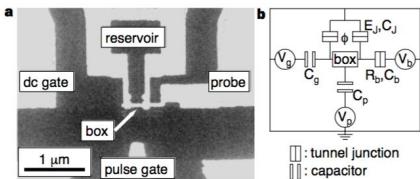
J. Clarke, 1966

Macroscopic coherence



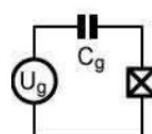
J. Martinis, M. Devoret, J. Clarke, PRB, 1987

Coherent control

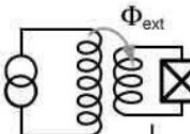


Y. Nakamura et al., Nature, 1999

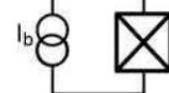
Qubit taxonomy



Charge



Flux

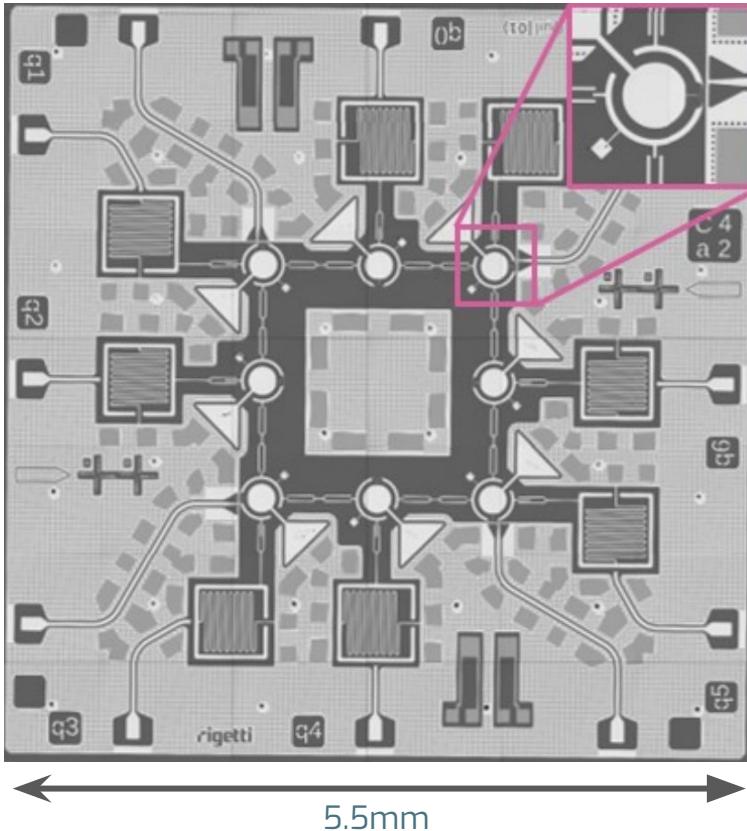


Phase

M. Devoret, A. Wallraff, J. Martinis, arXiv:0411174 (2004)

Our Implementation

Aluminum circuit on Silicon



Circuit Quantum Electrodynamics (cQED)

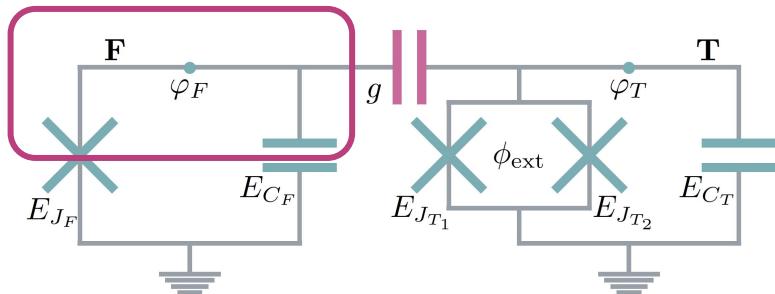
- Superconducting circuits at very low temps (0.01 Kelvin)
- Qubit = Circuit element made with Josephson Junctions (JJ's)



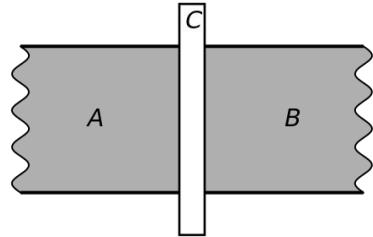
cQED - the transmon qubit(s)

- Josephson Junction - Cooper Pair Box
- Transmon regime: $E_J \gg E_C$
- JJ's are nonlinear
- Nonlinearity→two-level subspace→qubit
- Qubits coupled to linear resonators for state readout
- Jaynes-Cummings Hamiltonian:
Atomic transition coupled to cavity mode
- Two types of transmons: Fixed & Tunable

CPB



Josephson Effect



$$i\hbar \frac{\partial}{\partial t} \begin{pmatrix} \psi_A \\ \psi_B \end{pmatrix} = \begin{pmatrix} qV/2 & K \\ K & -qV/2 \end{pmatrix} \begin{pmatrix} \psi_A \\ \psi_B \end{pmatrix}$$

- Fundamental commutation relation of cQED

$$[\hat{\Phi}, \hat{Q}] = i\hbar$$

- Equations of Motion

$$I_C = \dot{\rho}_A = -\dot{\rho}_B = I_0 \sin(\phi)$$

$$\dot{\phi} = \dot{\theta}_A - \dot{\theta}_B = \frac{qV}{\hbar}$$

Fixed SC Qubit

- Capacitor

$$I_C = C \frac{dV_C}{dt} = \frac{\hbar}{q} \ddot{\phi}$$

- Junction

$$I_J = I_0 \sin(\phi)$$

- Equation of motion

$$\frac{\hbar}{q} \ddot{\phi} + I_0 \sin(\phi) = I_b$$

- Using Lagrangian formalism and fundamental commutation relation to arrive at Hamiltonian

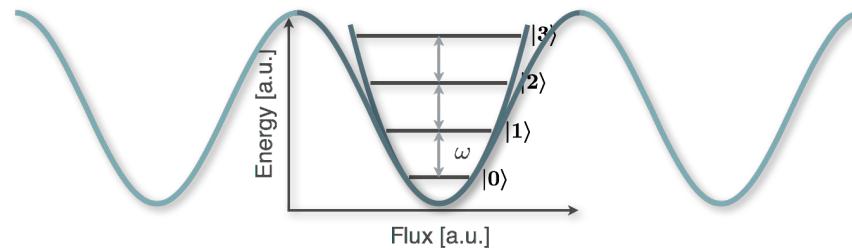
$$H = E_C \hat{n}^2 - E_J \cos(\hat{\phi}) - E_J \frac{I_b}{I_0} \hat{\phi}$$



Washboard and SC-Qubits

- Taylor expansion: Harmonic Oscillator

$$H = E_C \hat{n}^2 + \frac{1}{2} E_J \cos(\phi_0) (\hat{\phi} - \phi_0)^2$$



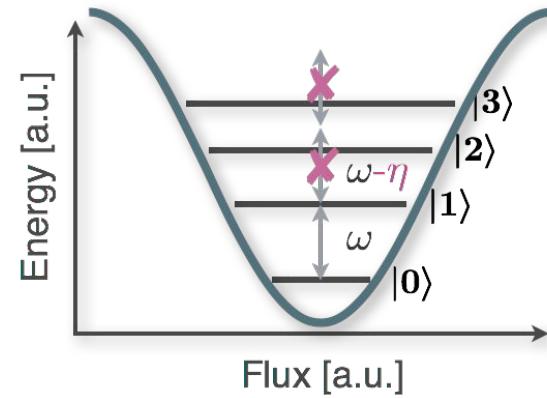
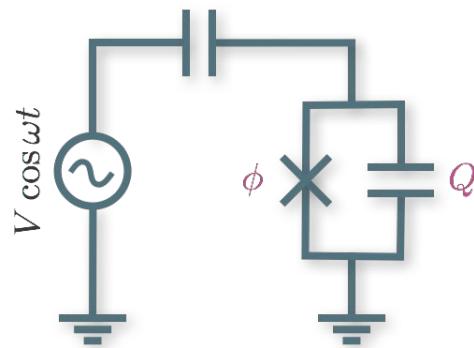
Washboard and SC-Qubits

- Taylor expansion: Harmonic Oscillator

$$H = E_C \hat{n}^2 + \frac{1}{2} E_J \cos(\phi_0) (\hat{\phi} - \phi_0)^2$$

- Higher order correction: Anharmonic spectrum
- Ground- and 1st excited state form Qubit states
- Transmon Regime

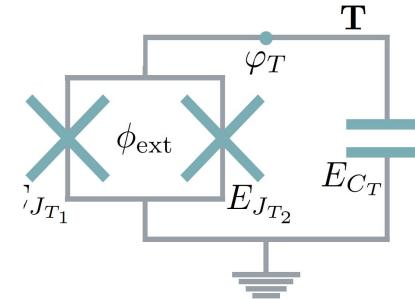
$$E_J \gg E_C$$



Tunable SC Qubit

- Phase sensitive SQUID
- Hamiltonian

$$H = 4E_C \hat{n}^2 - E_{J_1} \cos(\hat{\phi} - \varphi_{\text{ext}}) - E_{J_2} \cos(\hat{\phi})$$



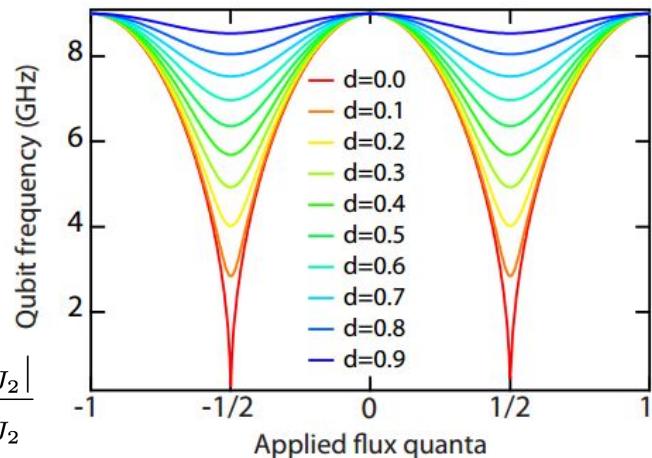
Symmetric junctions: $E_{J_1} = E_{J_2}$

Phases cancel
completely at $\Phi = \Phi_0/2$

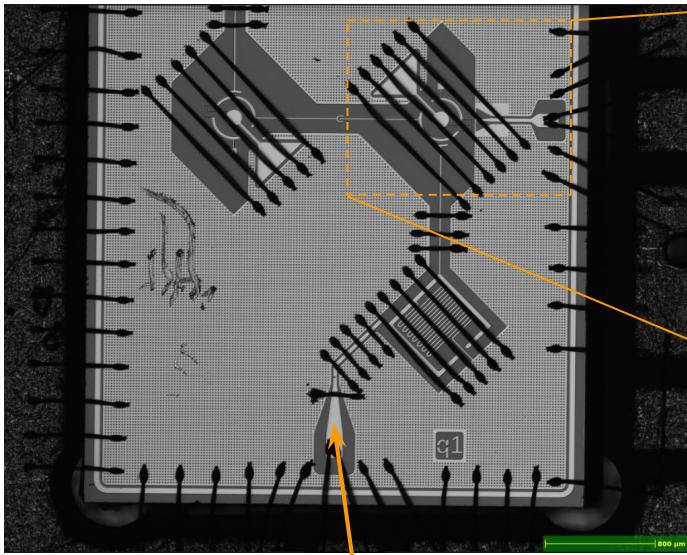
Asymmetric junctions: $E_{J_1} \neq E_{J_2}$

Phases cancel
incompletely at
 $\Phi = \Phi_0/2$

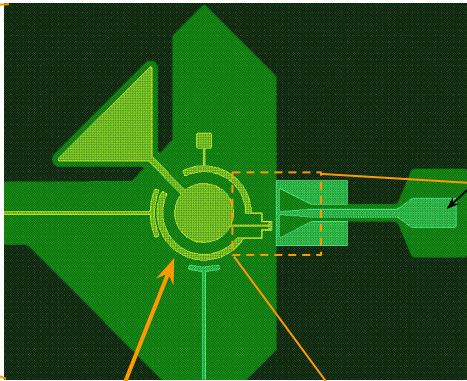
$$d = \frac{|E_{J_1} - E_{J_2}|}{E_{J_1} + E_{J_2}}$$



Experimental Device



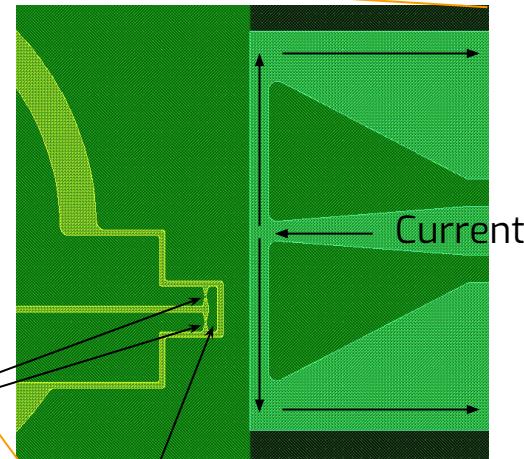
RF port



Tunable
transmon

Josephson
Junctions

Flux bias line (FBL)

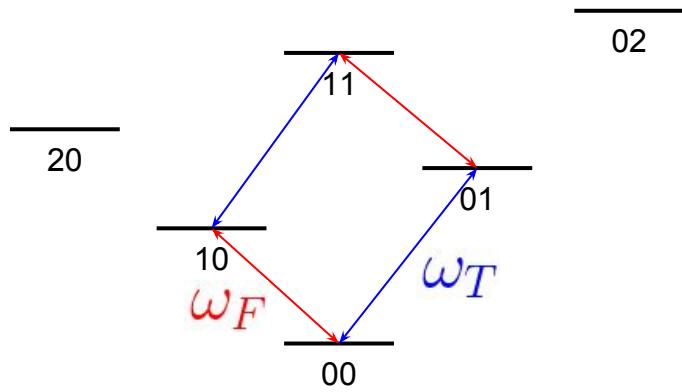


Current

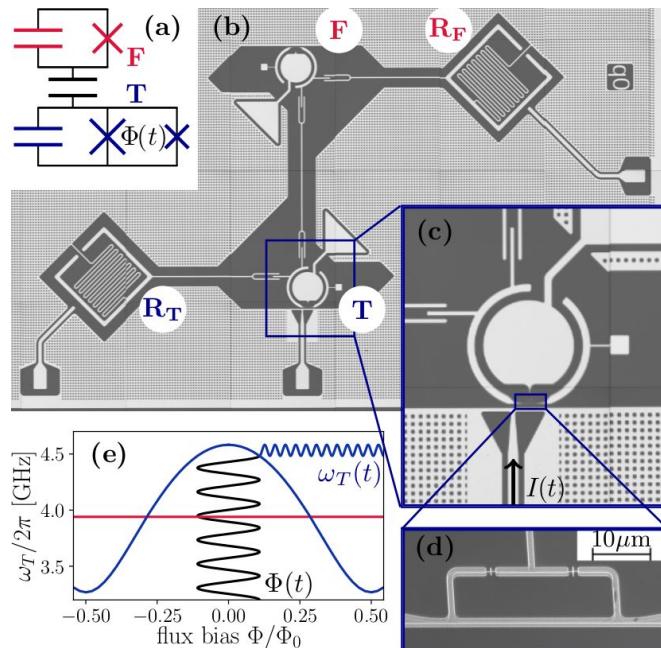
SQUID Loop



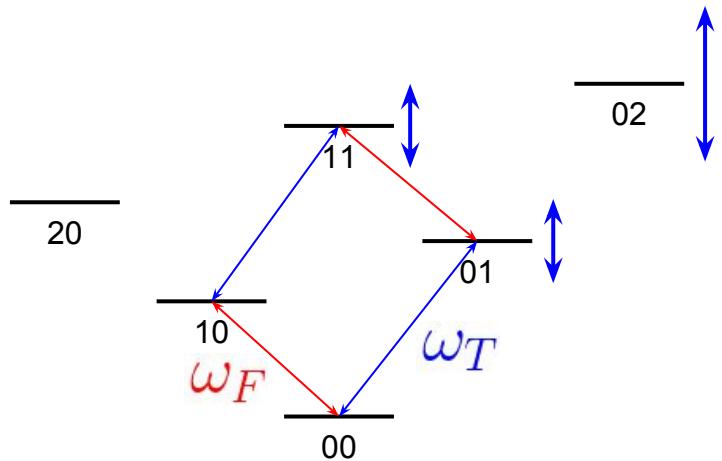
Parametric Entangling Gates



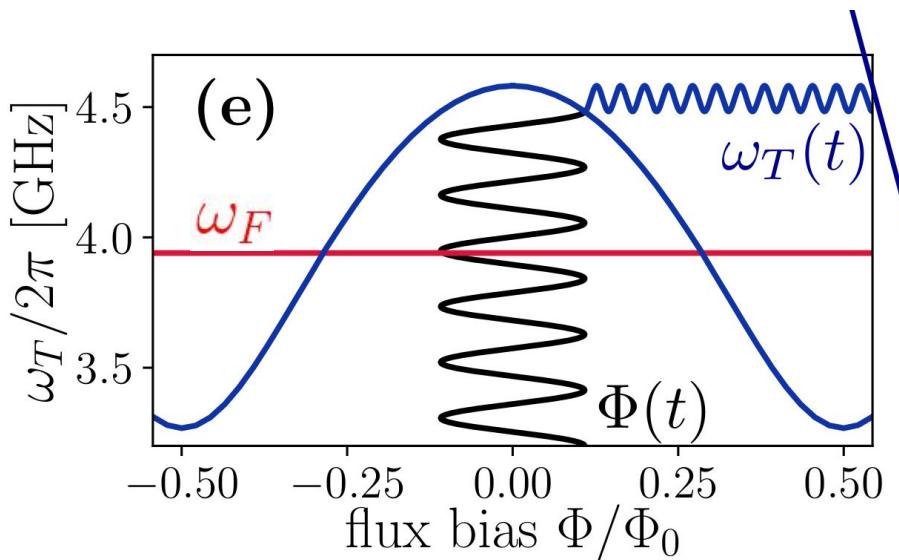
$$\Phi(t) = \bar{\Phi} + \tilde{\Phi} \cos(\omega_p t + \theta_p)$$



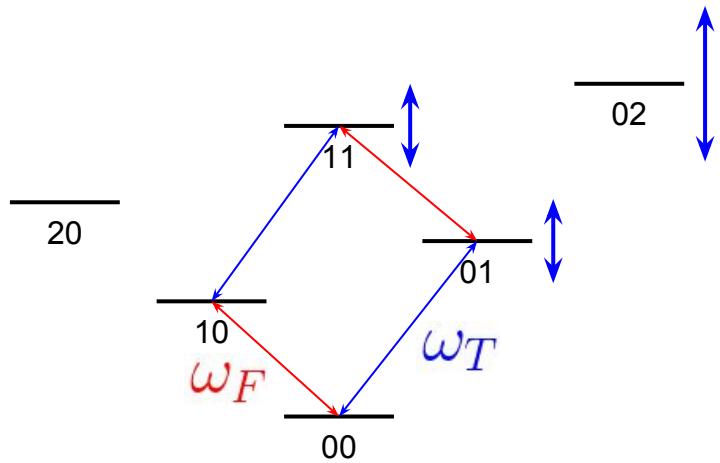
Parametric Entangling Gates



$$\Phi(t) = \bar{\Phi} + \tilde{\Phi} \cos(\omega_p t + \theta_p)$$
$$\omega_T(t) \approx \bar{\omega}_T(\bar{\Phi}) + \tilde{\omega}_T(\bar{\Phi}) \cos(2\omega_p t + 2\theta_p)$$

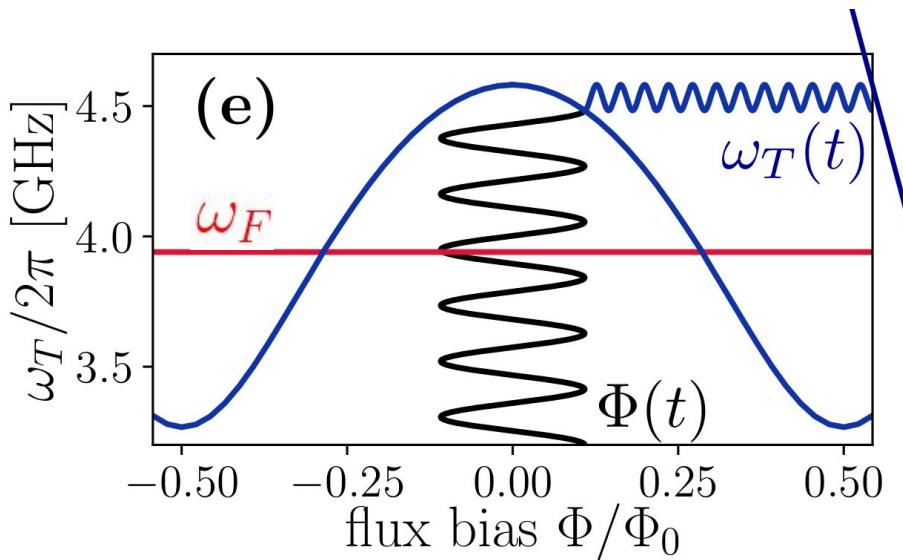


Parametric Entangling Gates

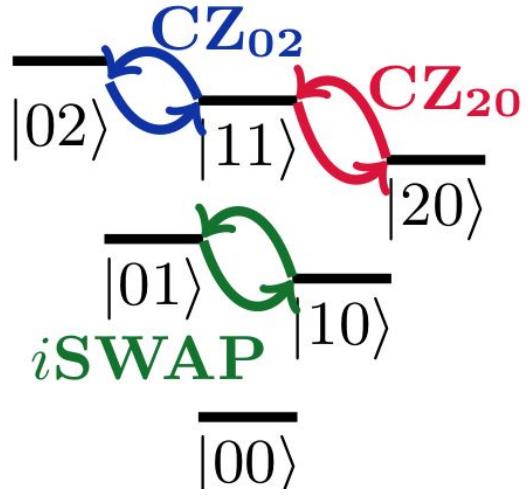


Resonance when
 $(\omega_T - \bar{\omega}_T) + 2\omega_p = \Delta_{ij}$
for some Δ between levels i, j

$$\Phi(t) = \bar{\Phi} + \tilde{\Phi} \cos(\omega_p t + \theta_p)$$
$$\omega_T(t) \approx \bar{\omega}_T(\bar{\Phi}) + \tilde{\omega}_T(\bar{\Phi}) \cos(2\omega_p t + 2\theta_p)$$



Parametric Entangling Gates



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

Resonance when

$$(\omega_T - \overline{\omega_T}) + 2\omega_p = \Delta_{ij}$$

for some Δ between levels i, j



Randomized Benchmarking

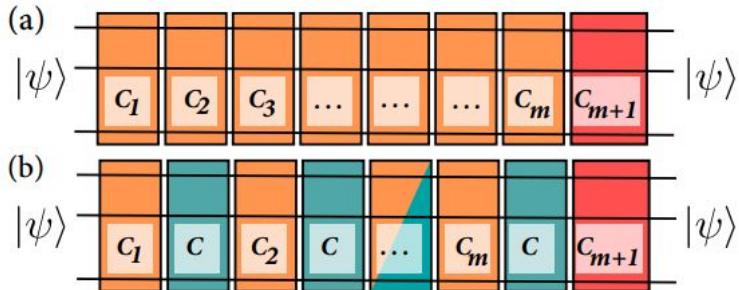
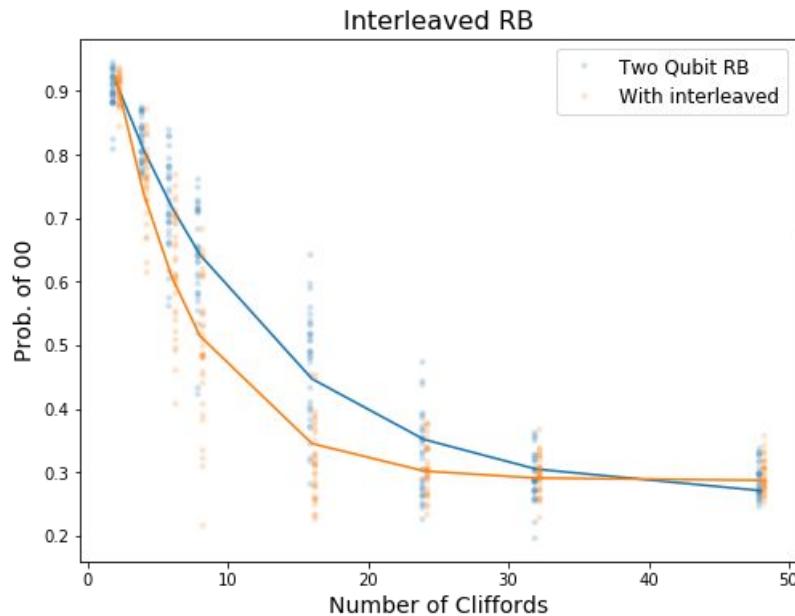


Image from Magesan

Interleaved RB

- A. Run sequences of 2Q Cliffords
- B. Run sequences with gate C interleaved

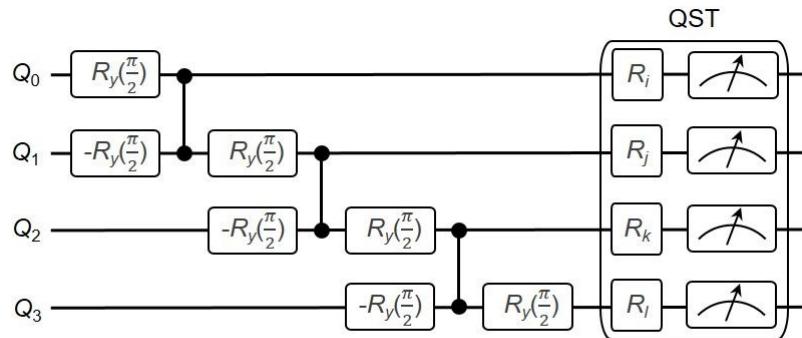
Attempt to isolate the infidelity due to C



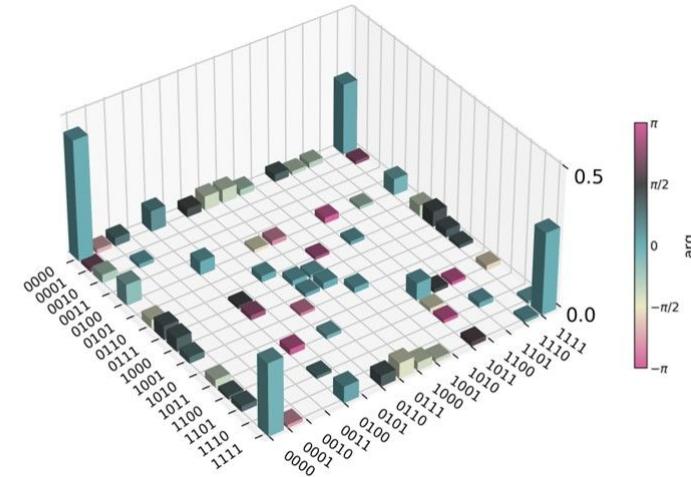
2Q gate type	Avg error per 2Q gate
iSwap	5.9%
CZ_{02}	8.5%
CZ_{20}	8.7%

4 qubit entangled state verification

Quantum Circuit



Quantum State Tomography



Fidelity of 79%



Quil and the Quantum Abstract Machine

A hybrid classical/quantum programming model.

FSM & QAM

FSM

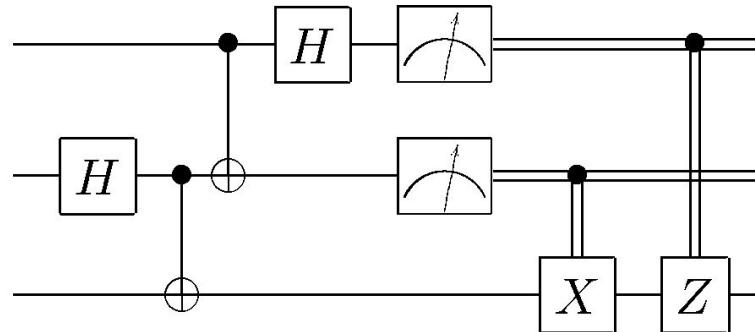
- Classic Bits 0 and 1 encode the state
- Universal Gate Sets
 - NOT + AND
 - NOT + OR
 - AND + XOR
 - ...
- Execution state (next instruction)

QAM

- Qubits $|0\rangle$ and $|1\rangle$
- Universal Quantum Gate Set
 - CNOT + Single Qubit Gates
- Classic Bits 0 and 1 encode the state
- Universal Gate Sets
 - NOT + AND
 - NOT + OR
 - AND + XOR
 - ...
- Execution state (next instruction)



Quil is **portable** and **hybrid**.



The Quil Programming Model

Targets a **Quantum Abstract Machine (QAM)**

- > **Quil** is the instruction language and is how you interact with the machine
- > It is a syntax for representing state transitions.



The Quil Programming Model

Targets a **Quantum Abstract Machine (QAM)**

- > **Quil** is the instruction language and is how you interact with the machine
- > It is a syntax for representing state transitions.

Ψ : Quantum state (qubits) → quantum instructions

Quil Example

H 3

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

MEASURE 3 [4]

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

JUMP-WHEN @END [5]

.

.

.



The Quil Programming Model

Targets a **Quantum Abstract Machine (QAM)**

- > **Quil** is the instruction language and is how you interact with the machine
- > It is 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)**

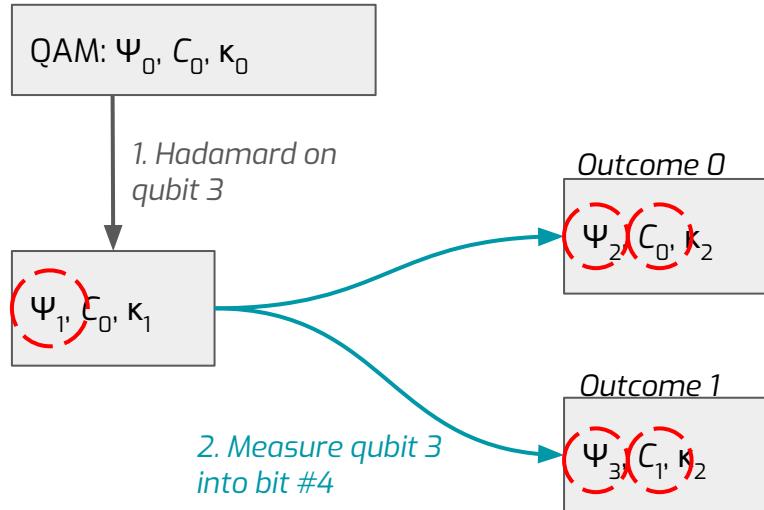
- > **Quil** is the instruction language and is how you interact with the machine
- > It is a syntax for representing state transitions.

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



Quil Example

H 3

MEASURE 3 [4]

JUMP-WHEN @END [5]

.



The Quil Programming Model

Targets a **Quantum Abstract Machine (QAM)**

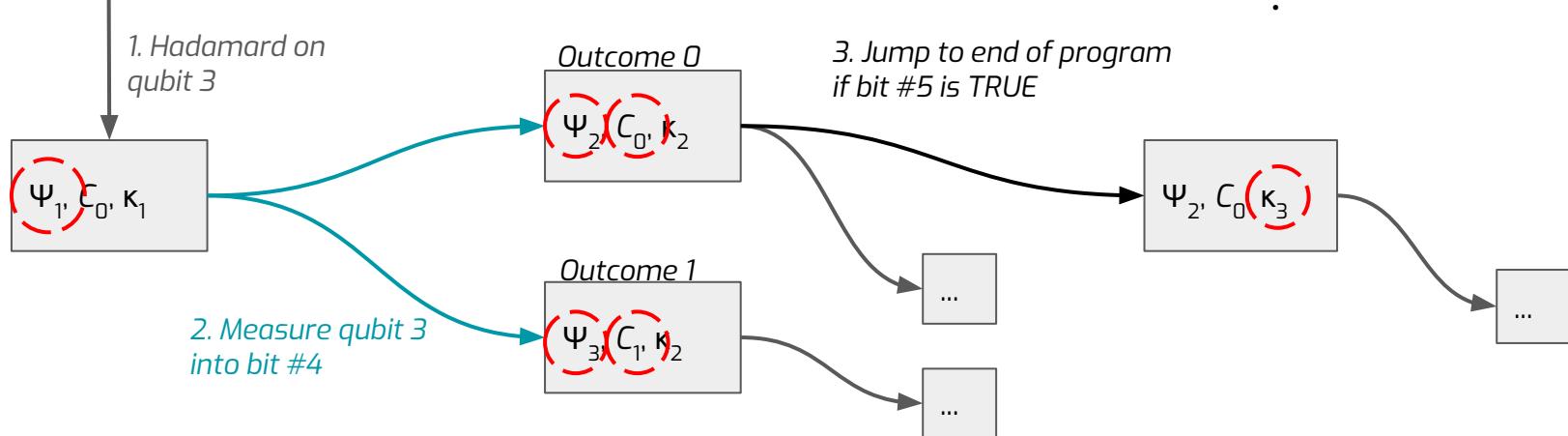
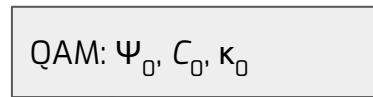
- > **Quil** is the instruction language and is how you interact with the machine
- > It is a syntax for representing state transitions.

Ψ : Quantum state (qubits) → quantum instructions

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

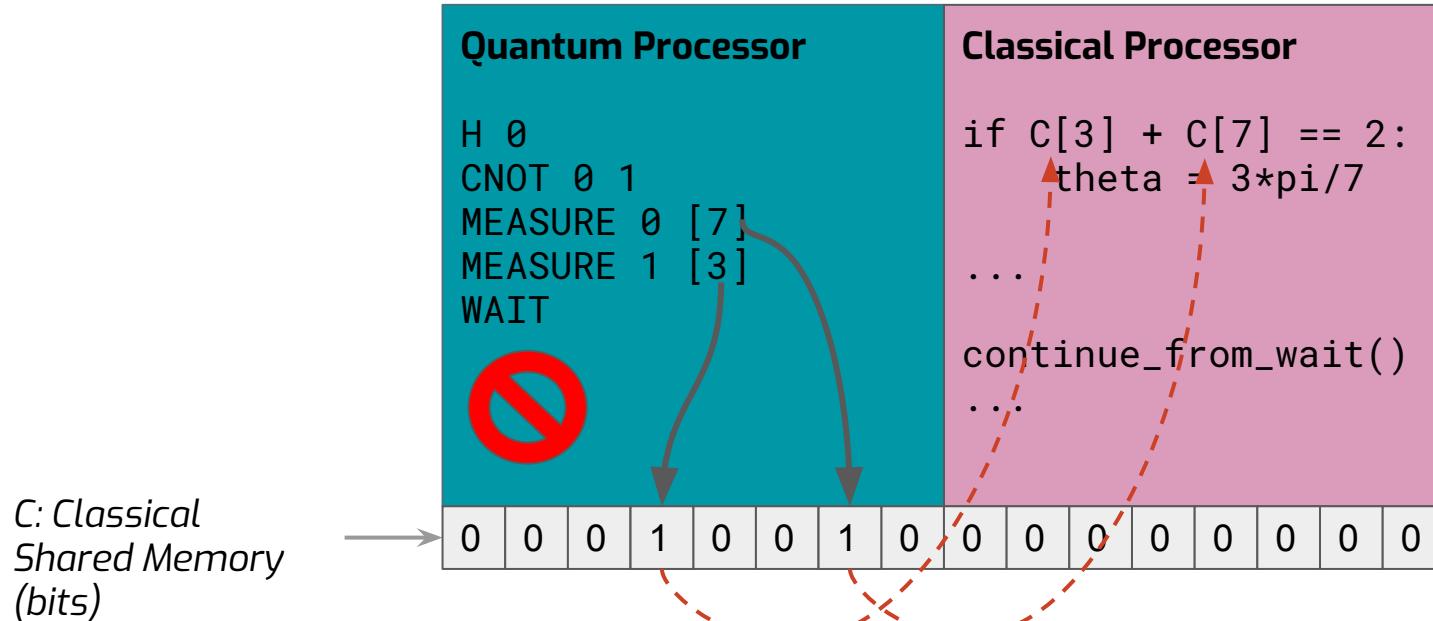
κ : Execution state (program) → control instructions (e.g., jumps)

0. Initialize into zero states



Interacting with a Classical Computer

- > The Quantum Abstract Machine has a **shared classical state**.
- > The QAM becomes a practical device with this shared state.
- > Classical computers can take over with classical/quantum synchronization.



Formal Details: The Quil White Paper

arXiv:[1608.03355](https://arxiv.org/abs/1608.03355)

A Practical Quantum Instruction Set Architecture

Robert S. Smith, Michael J. Curtis, William J. Zeng

Rigetti Computing

775 Heinz Ave.

Berkeley, California 94710

Email: {robert, spike, will}@rigetti.com

Abstract—Quantum computing technology has advanced rapidly in the last few years. Physical systems—superconducting qubits in particular—promise scalable gate-based hardware. Alongside these advances, new algorithms have been discovered that are adapted to the relatively smaller, noisier hardware that will become available in the next few years. These tend to be hybrid classical/quantum algorithms, where the quantum hardware is used in a co-processor model. Here, we introduce an abstract machine architecture for describing these algorithms, along with a language for representing computations on this machine, and discuss a classically simulable implementation architecture. *Keywords*—quantum computing, software architecture

IV	Quil Examples	7
IV-A	Quantum Fourier Transform	7
IV-B	Static and Dynamic Implementation of QVE	8
IV-B1	Static implementation	8
IV-B2	Dynamic implementation	8
V	A Quantum Programming Toolkit	9
V-A	Overview	9



Quantum Teleportation in Quil

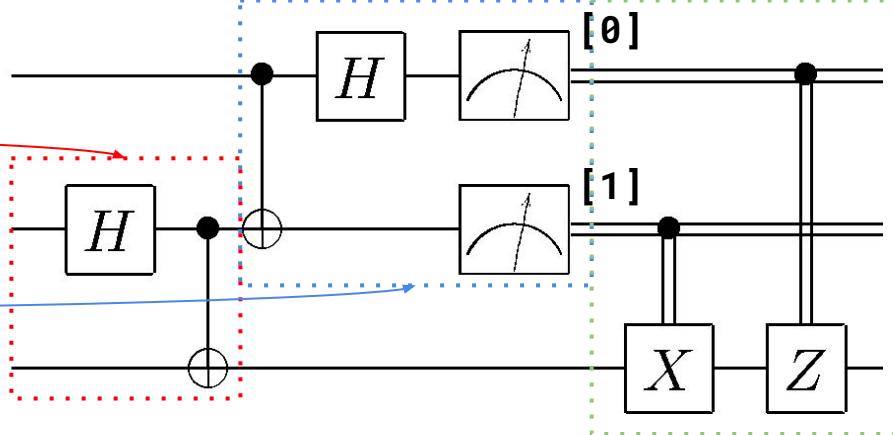
```
DEFQCIRCUIT TELEPORT A q B:  
    # Bell pair  
    H      A  
    CNOT   A B  
  
    # Teleport  
    CNOT   q A  
    H      q  
    MEASURE q [0]  
    MEASURE A [1]
```

Alice's ancilla q

Alice A
Bob B

```
# Classically communicate measurements  
JUMP-UNLESS @SKIP [1]  
X B  
LABEL @SKIP  
JUMP-UNLESS @END [0]  
Z B  
LABEL @END
```

```
# If Alice's qubits are 0 and 1  
# and Bob's is 5  
TELEPORT 0 1 5
```



Teleportation Truth Table

- Bell State: $|00\rangle + |11\rangle$
- Ancilla: $\alpha|0\rangle + \beta|1\rangle$

State $ A B q\rangle$	CNOT q A	$H q$	Classic C-X A B	Classic C-Z q B
$\alpha 000\rangle$	$\alpha 000\rangle$	$ 00\rangle_{Aq} (\alpha 0\rangle + \beta 1\rangle)_B$	$[0, 0]_{Aq}; \alpha 0\rangle + \beta 1\rangle$	$\alpha 0\rangle + \beta 1\rangle$
$\beta 001\rangle$	$\beta 101\rangle$	$ 11\rangle_{Aq} (\alpha 1\rangle - \beta 0\rangle)_B$	$[1, 1]_{Aq}; \alpha 0\rangle - \beta 1\rangle$	$\alpha 0\rangle + \beta 1\rangle$
$\alpha 110\rangle$	$\alpha 110\rangle$	$ 10\rangle_{Aq} (\alpha 1\rangle + \beta 0\rangle)_B$	$[1, 0]_{Aq}; \alpha 0\rangle + \beta 1\rangle$	$\alpha 0\rangle + \beta 1\rangle$
$\beta 111\rangle$	$\beta 011\rangle$	$ 01\rangle_{Aq} (\alpha 0\rangle - \beta 1\rangle)_B$	$[0, 1]_{Aq}; \alpha 0\rangle - \beta 1\rangle$	$\alpha 0\rangle + \beta 1\rangle$



pyQuil generates Quil

```
from pyquil.gates import X, CNOT, H, Z, RX, I
from pyquil.api import QVMConnection
from pyquil.quil import Program
import numpy as np

qvm = QVMConnection()

alice_register = 0
ancilla_register = 1

flip_correction_branch = Program(X(1))
phase_correction_branch = Program(Z(1))

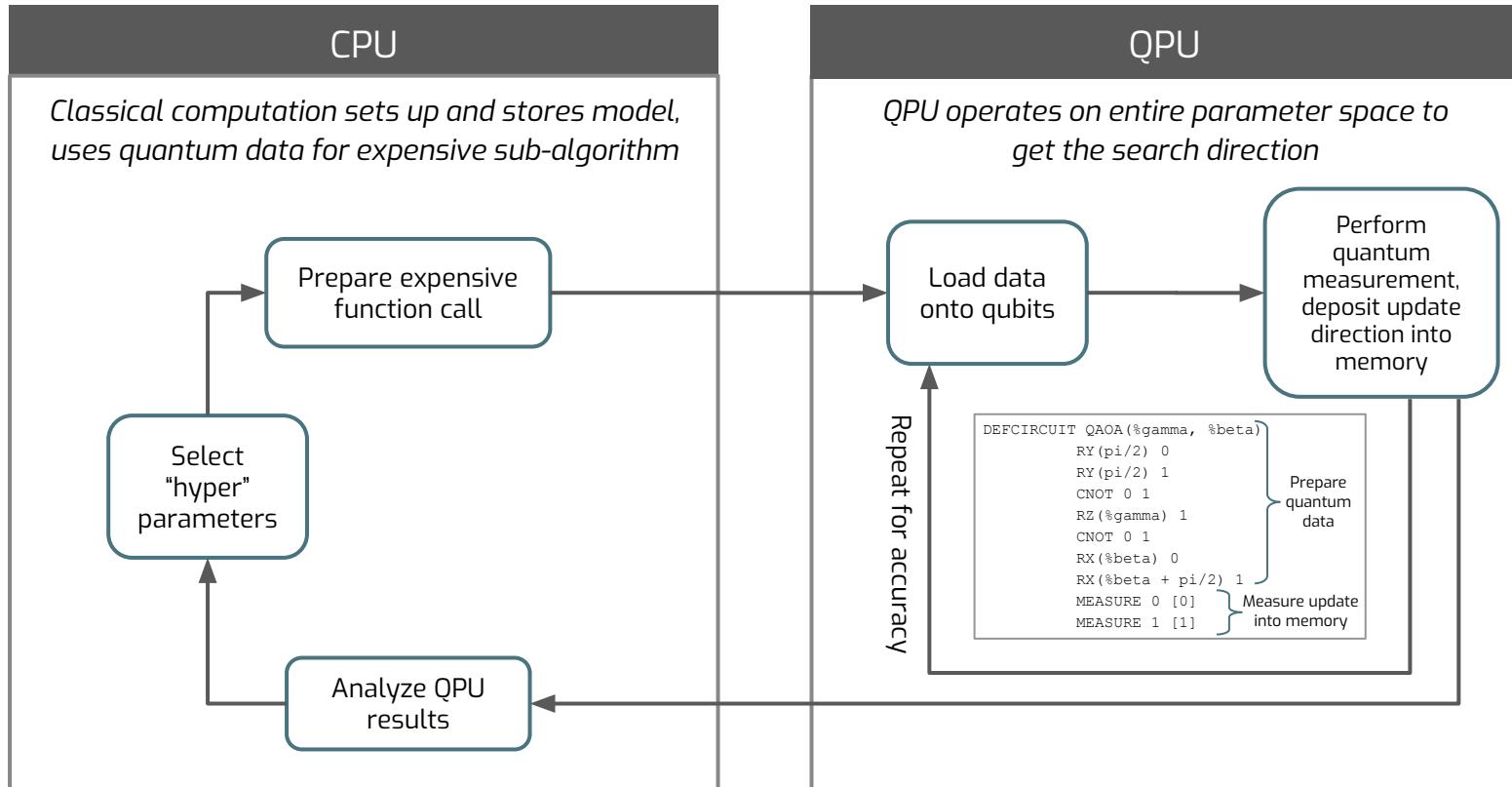
prog = (Program()
        .inst(H(0))
        .inst(CNOT(0, 1))
        .inst(RX(0.2 * np.pi, 2))
        .inst(CNOT(2, 0))
        .inst(H(2))
        .measure(0, alice_register)
        .measure(2, ancilla_register)
        .if_then(alice_register, flip_correction_branch)
        .if_then(ancilla_register, phase_correction_branch))

qvm.run_and_measure(prog, list(prog.get_qubits()), trials=10)
```

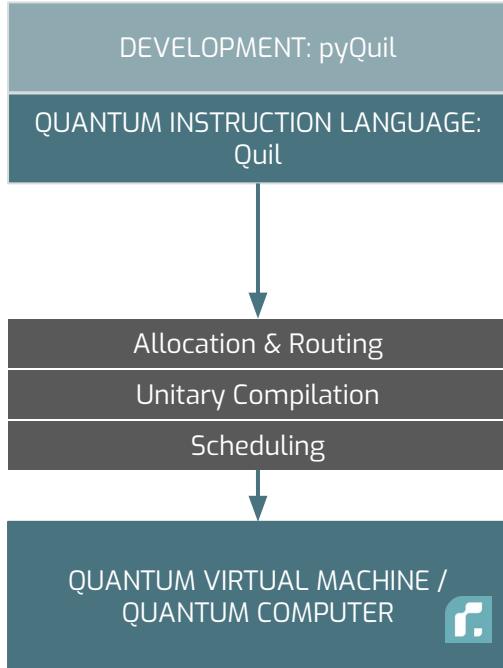
```
H 0
CNOT 0 1
RX(pi/5) 2
CNOT 2 0
H 2
MEASURE 0 [0]
MEASURE 2 [1]
JUMP-WHEN @THEN1 [0]
JUMP @END2
LABEL @THEN1
X 1
LABEL @END2
JUMP-WHEN @THENS5 [1]
JUMP @END6
LABEL @THENS5
Z 1
LABEL @END6
```



Hybrid Quantum Computing



Compilation



Some useful tools

- QVM with up to 37 qubits (on AWS)
- Can simulate arbitrary 1 & 2 qubit noise by definition of Kraus maps

```
def damping_channel(p=.01):
    damping_op = np.array([[0, sqrt(p)],
                          [0, 0]])
    residual_kraus = np.diag([1, sqrt(1-p)])
    return [residual_kraus, damping_op]

# overload identity gate I on qc 0 with p=1% damping probability
p.define_noisy_gate("I", [0], damping_channel(0.01))
```

- Compilation layer
 - Specify arbitrary circuit and compile to natural gate set
 - Compression optimizations build in
 - Simple layout optimization: Quantum Circuit to Chip Topology
 - E.g. compiled RB circuit to identity
- Simple prototyping before requesting QPU

```
from pyquil.api import QVMConnection

cxn = QVMConnection()
cxn.run_and_measure(prog, ...)
```

```
from pyquil.api import QPUConnection

cxn = QPUConnection("19Q-Acorn")
cxn.run_and_measure(prog, ...)
```

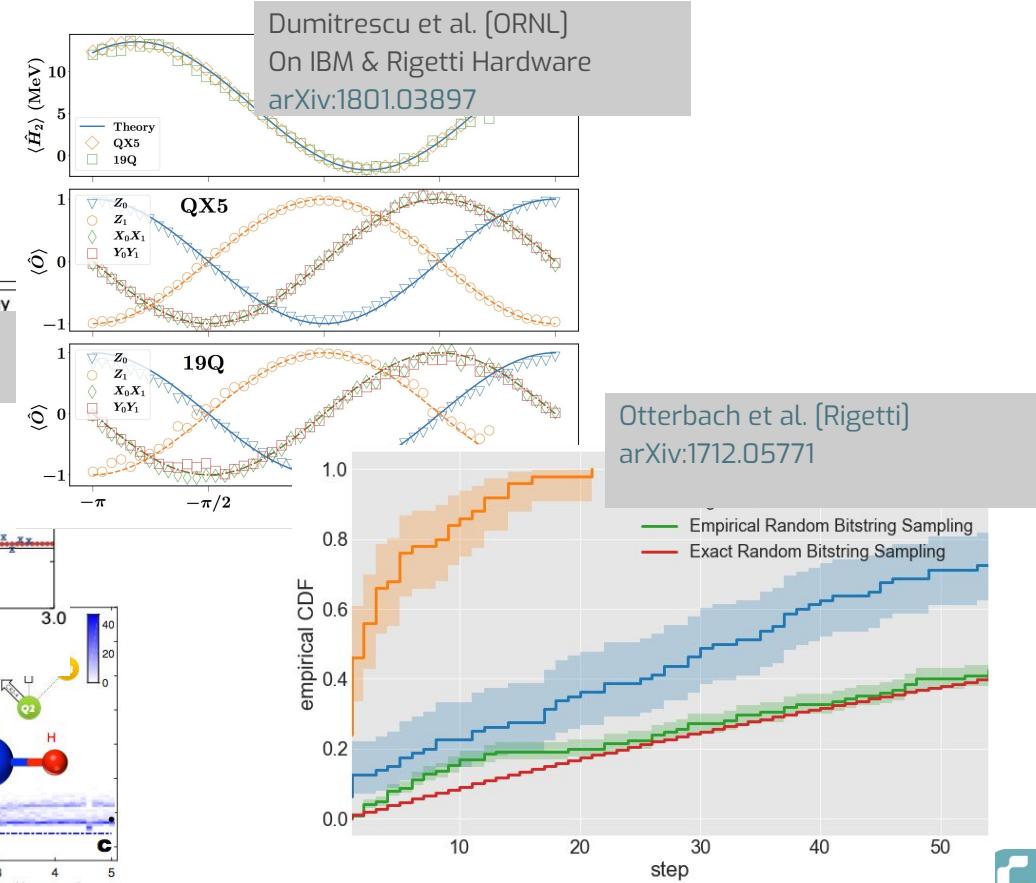
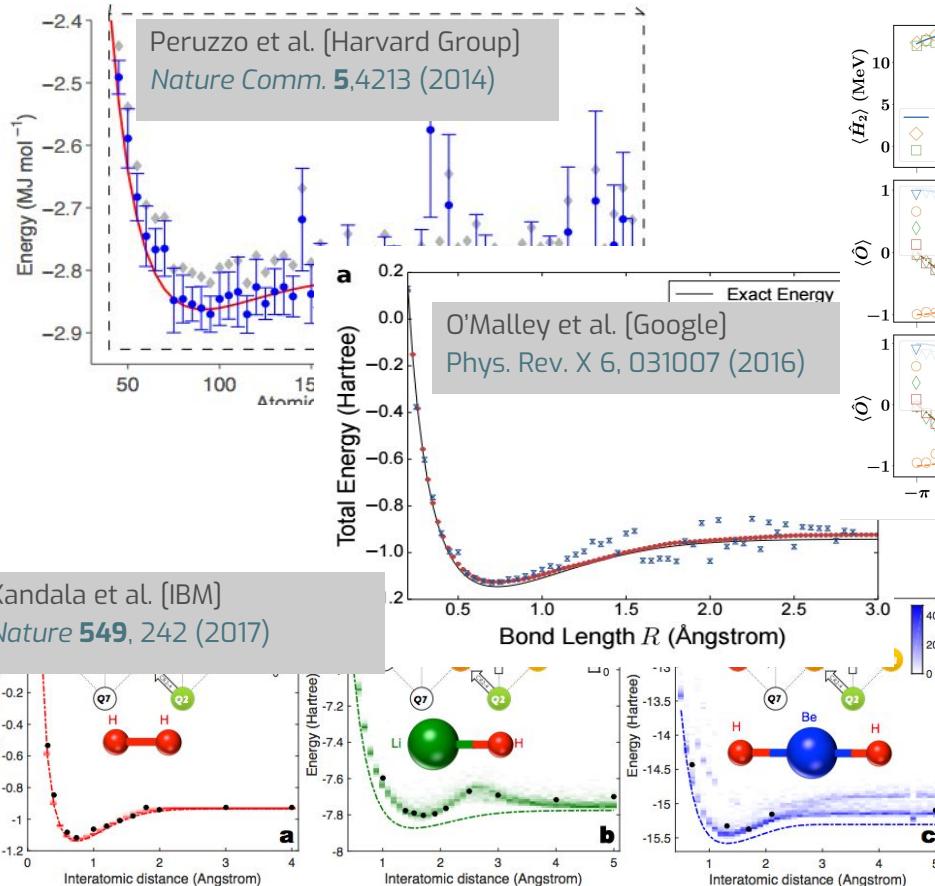


NISQ

Applications

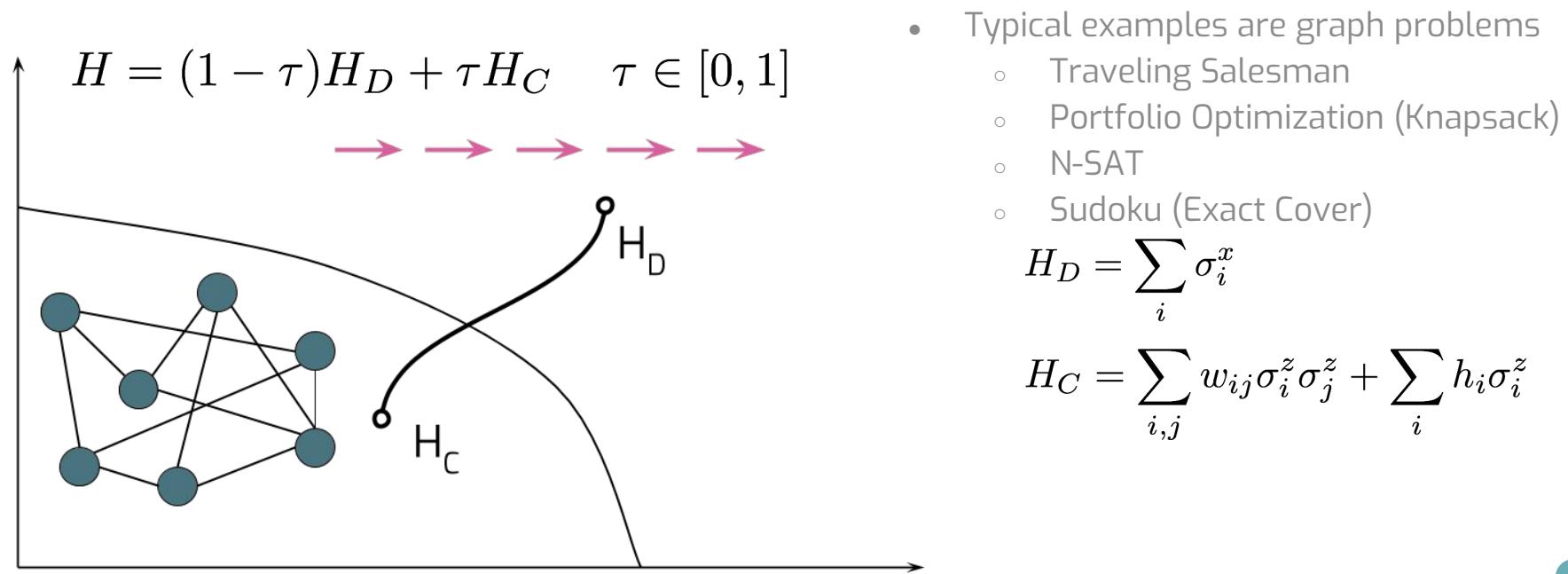
NISQ - Near-term Intermediate Scale Quantum

Preskill, arXiv:[1801.00862](https://arxiv.org/abs/1801.00862)



QAOA - Quantum Approximate Optimization Algorithm

- Motivated through Adiabatic Quantum Computing
- Encode solution to hard NP-complete problem in ground-state of H_C
- Start at easy to prepare initial state with Hamiltonian H_D



Trotterization

- Gate model of AQC (Farhi, Goldstone, Gutman, arxiv:1411.4028)

$$\begin{aligned} U &= e^{-i((1-\tau)H_D + \tau H_C)t} \\ &= \lim_{p \rightarrow \infty} [e^{-i(1-\tau)H_D t/p} e^{-i\tau H_C t/p}]^p \\ &\rightarrow \prod_{p=1}^{\infty} e^{-i\beta_p H_D} e^{-i\gamma_p H_C} \end{aligned}$$

- Angles β_p, γ_p need not be small
- Theory for $p \rightarrow \infty$ is exact; in practice small p is already good
- No specification on how to find optimal β_p, γ_p



Example: Maxcut

“Maximize disagreement on a colored graph”



Score 0

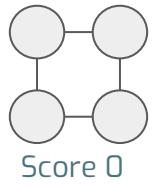


Score 0

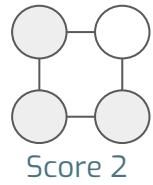


Score +1

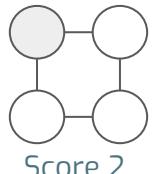
4-node “ring of disagrees”



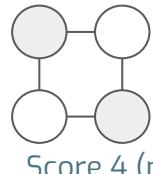
Score 0



Score 2



Score 2



Score 4 (max)



December 2016



Example: Maxcut

- Initial state is ground state of H_D : $| \rightarrow \rangle = H^{\otimes n} |0\rangle$
- Run the QAOA prescription:

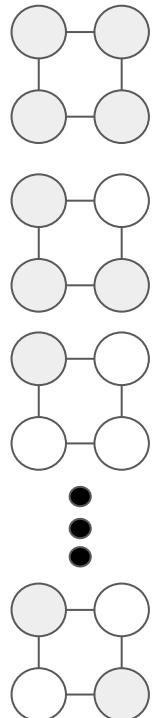
$$|\beta, \gamma\rangle = \prod_{p=1}^{\infty} U_p V_p H^{\otimes n} |0\rangle$$

- Intuitively: Superposition of bitstring configurations

$$|\beta, \gamma\rangle \equiv \sqrt{p_1} \begin{array}{c} \text{Diagram of a 4-node complete graph } K_4 \\ \text{with edges between all pairs of nodes} \end{array} + \sqrt{p_2} \begin{array}{c} \text{Diagram of a 4-node complete graph } K_4 \\ \text{with edges between all pairs of nodes, except the bottom-left edge} \end{array} + \dots + \sqrt{p_{16}} \begin{array}{c} \text{Diagram of a 4-node complete graph } K_4 \\ \text{with edges between all pairs of nodes, except the top-right edge} \end{array}$$



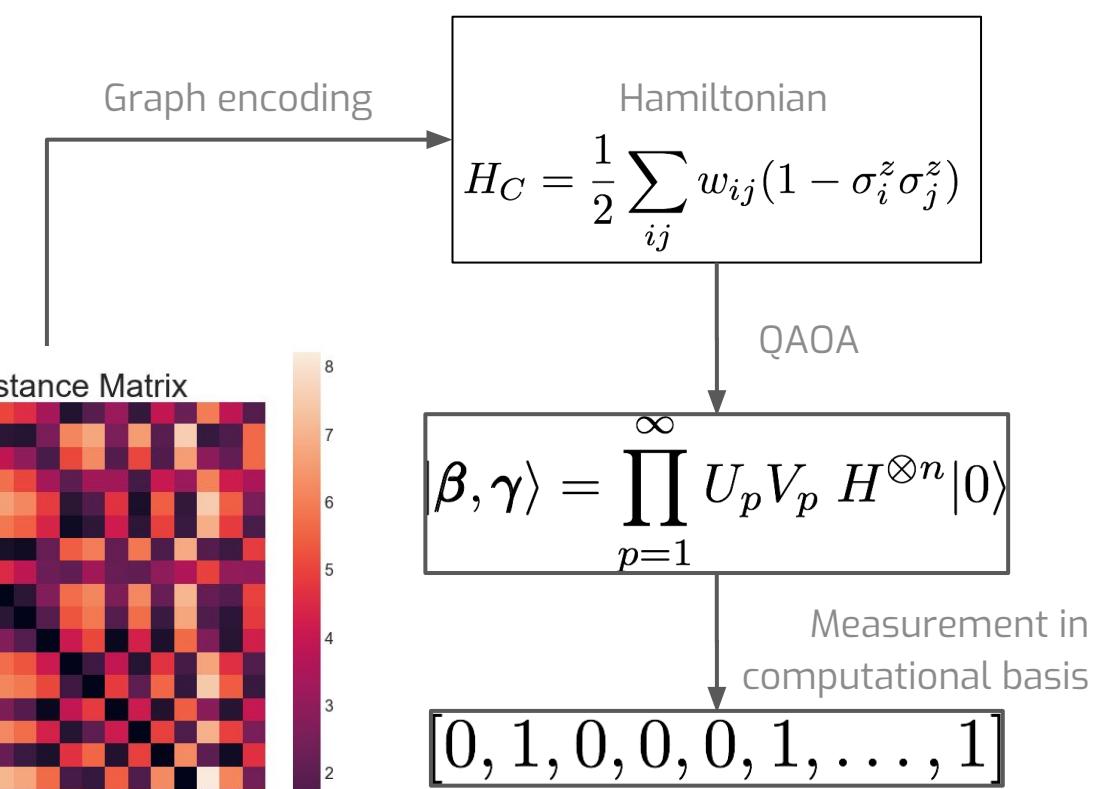
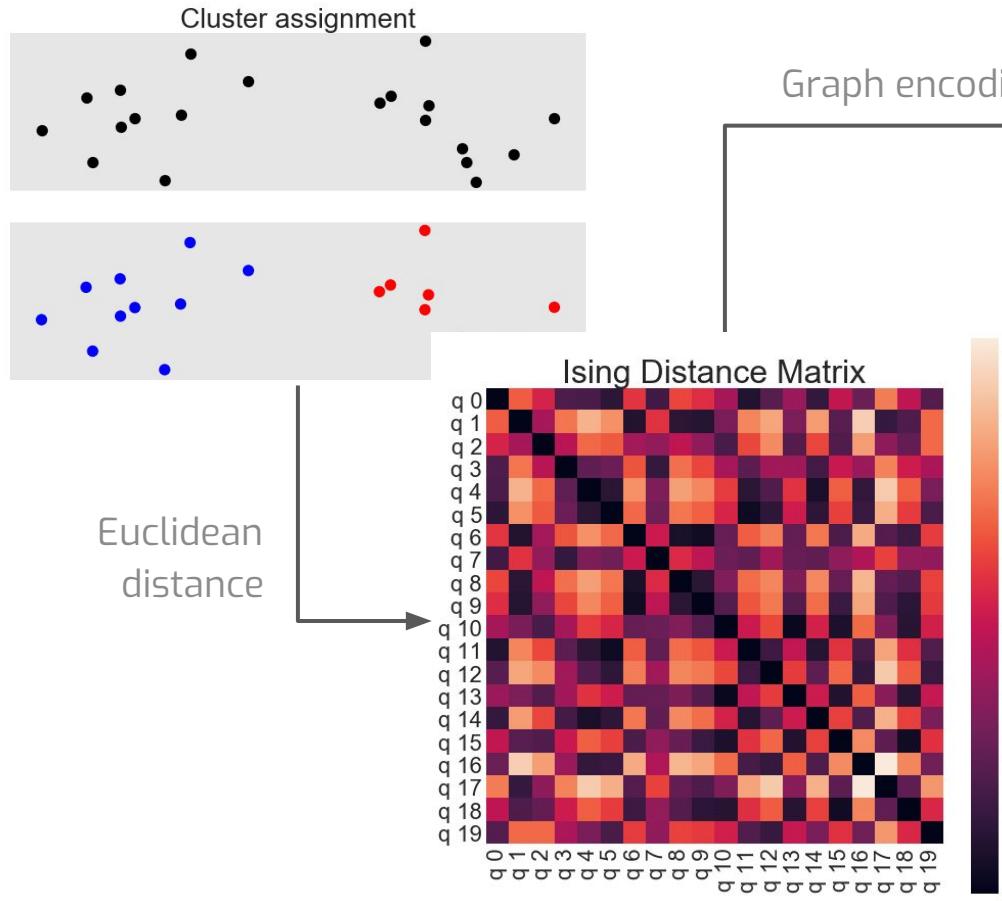
Probability distributions over bit strings



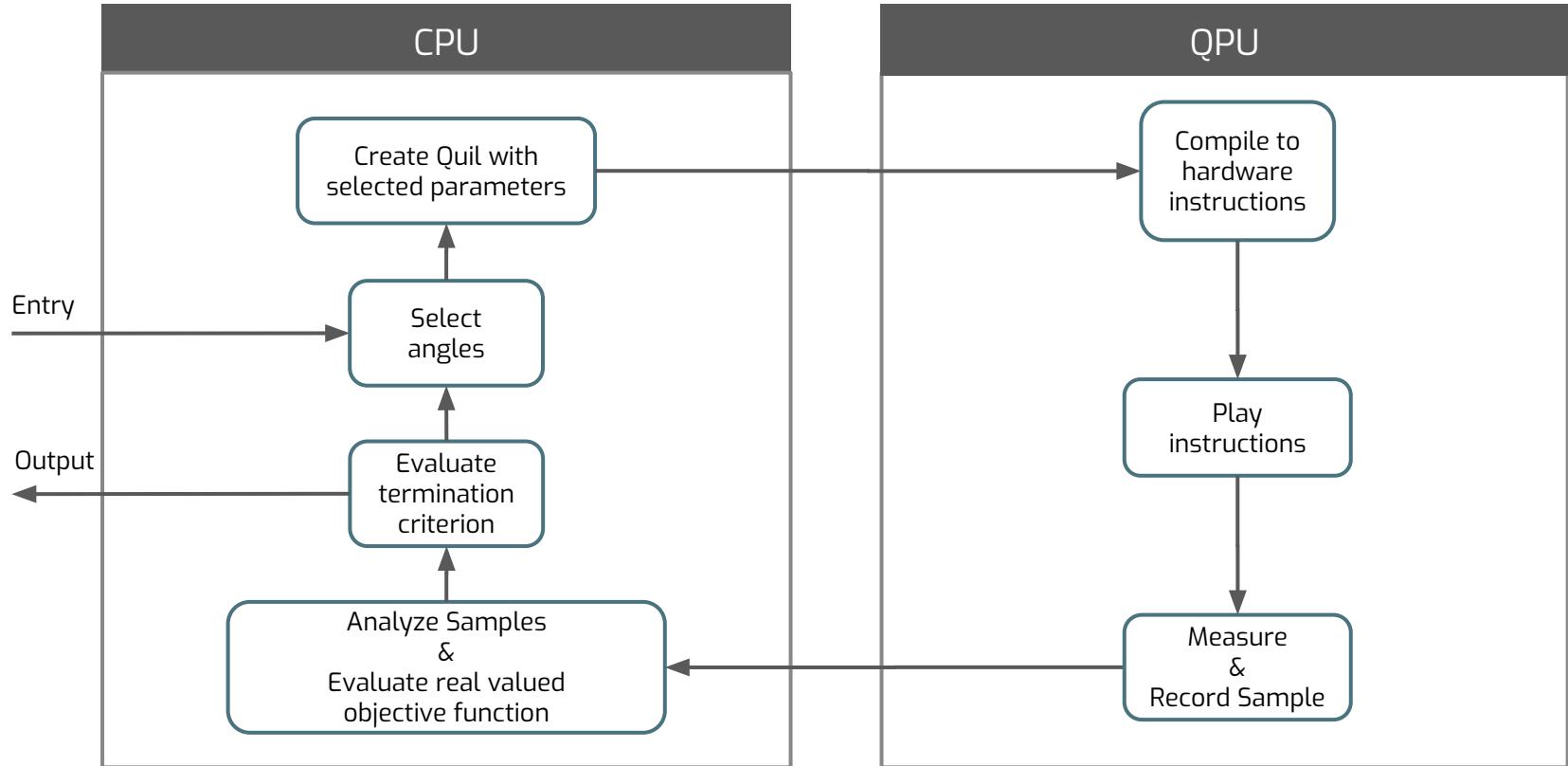
Cost	Choice $ \beta_1, \gamma_1\rangle$	Choice $ \beta_2, \gamma_2\rangle$	Choice $ \beta_3, \gamma_3\rangle$
0	$p = 0.1$	$p = 0.01$	$p = 0.2$
2	$p = 0.3$	$p = 0.15$	$p = 0.01$
2	$p = 0.05$	$p = 0.2$	$p = 0.0$
4	$p = 0.01$	$p = 0.51$	$p = 0.25$



(Weighted) Maxcut as Clustering



When are we done?



Objective Function

- Loss/Reward Function:

$$c_{\beta, \gamma} : \{0, 1\}^n \mapsto \mathbb{R}$$

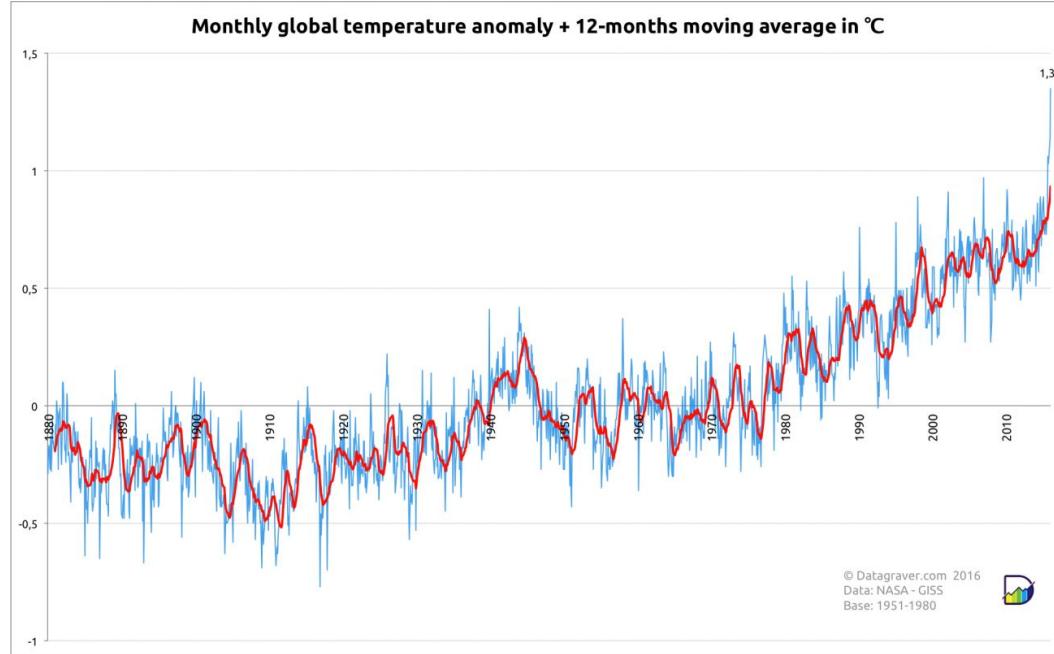
"quality of a sampled bit-string"

- Objective Function

$$f : X_{\beta, \gamma} \mapsto \mathbb{R},$$

$$(\beta, \gamma) \rightarrow \text{STAT}_c(c; \beta, \gamma)$$

- Extreme values
- Mean
- ...

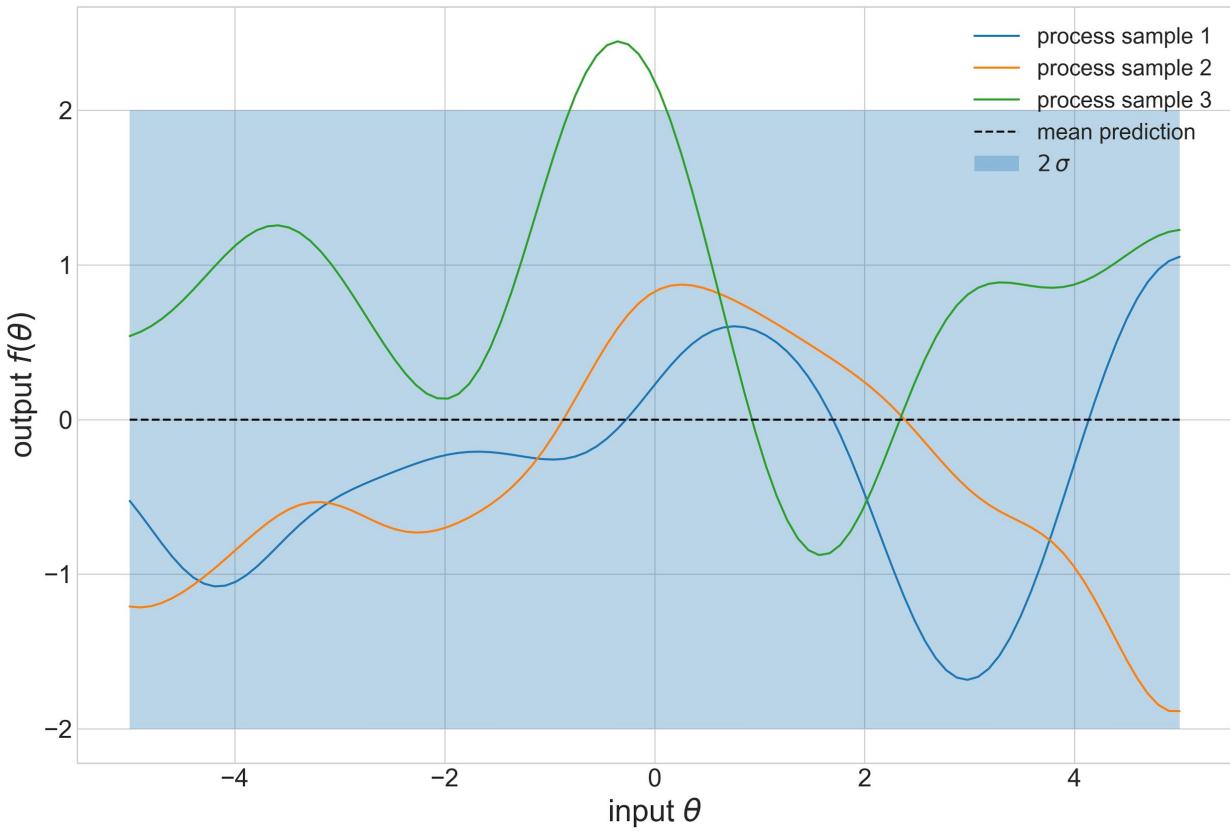


- Find the optimum of objective:
 - No easy access to gradients, need derivative free methods
 - Nelder-Mead
 - Bayesian Methods
 - ...

Image from <https://datagraver.com/case/world-temperature-anomalies-for-februari-2016>

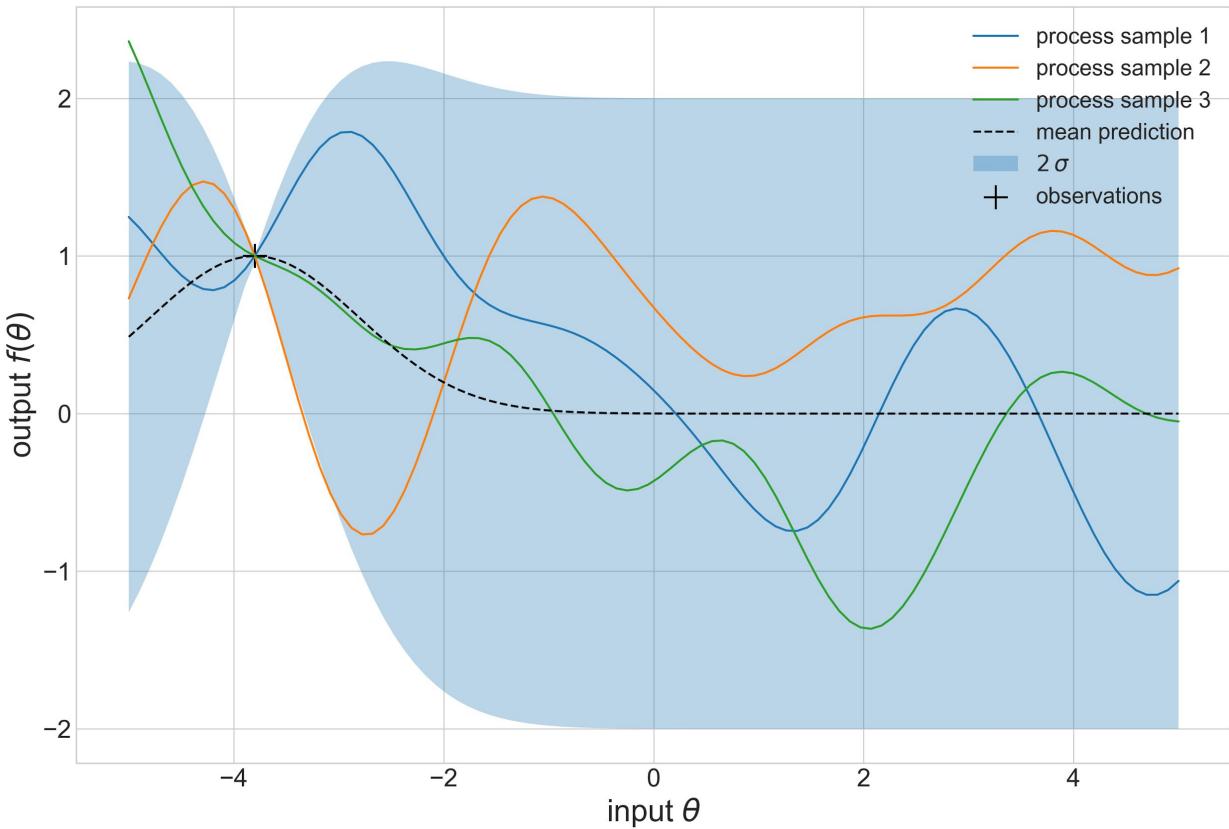
Bayesian Optimization

- Gaussian Process Prior of objective function



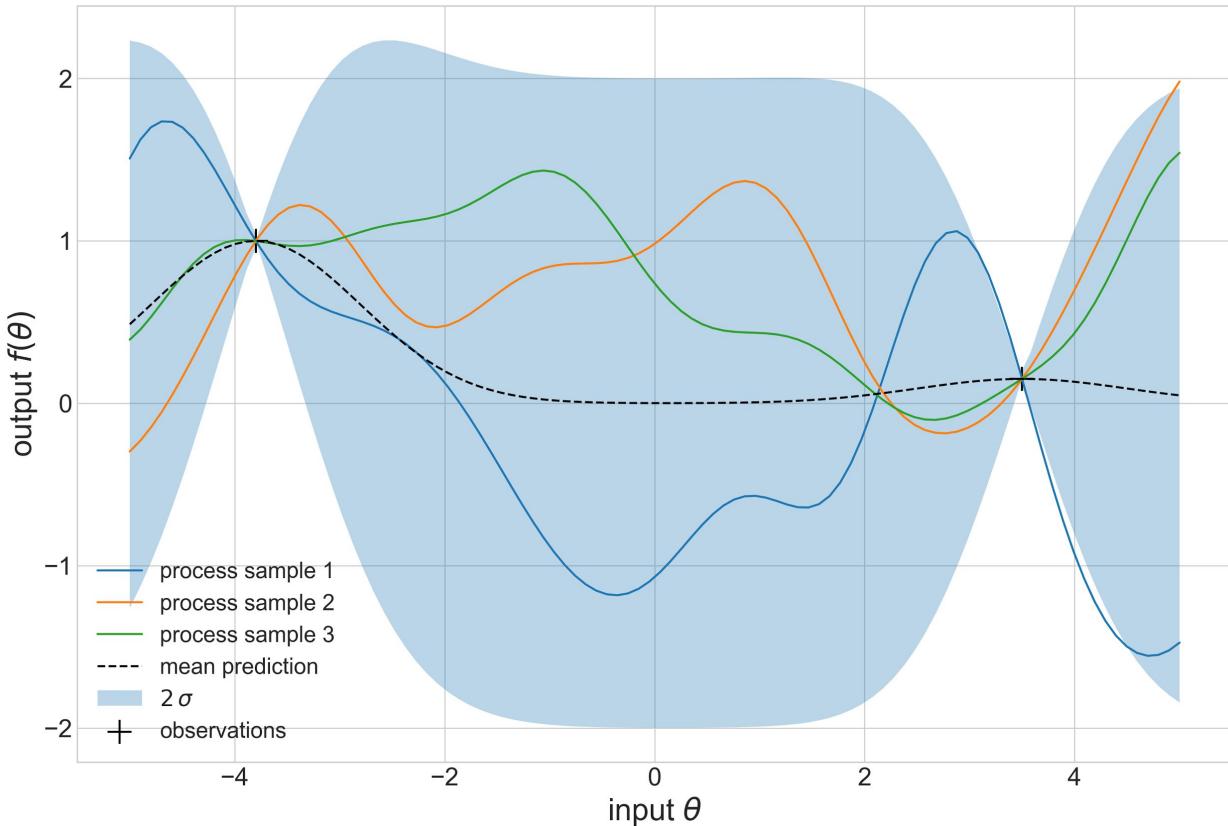
Bayesian Optimization

- Gaussian Process Prior of objective function
- Measure and update Prior



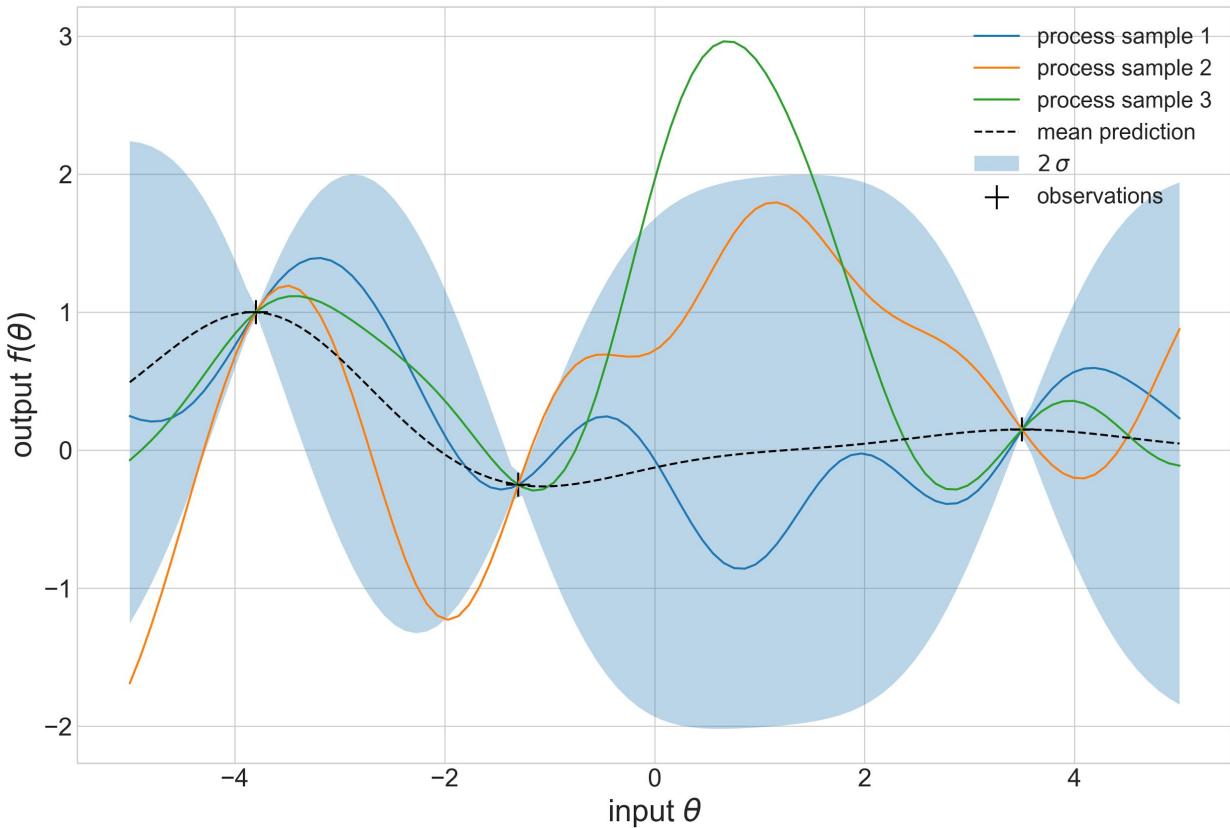
Bayesian Optimization

- Gaussian Process Prior of objective function
- Measure and update Prior
- Choose next point to measure and update



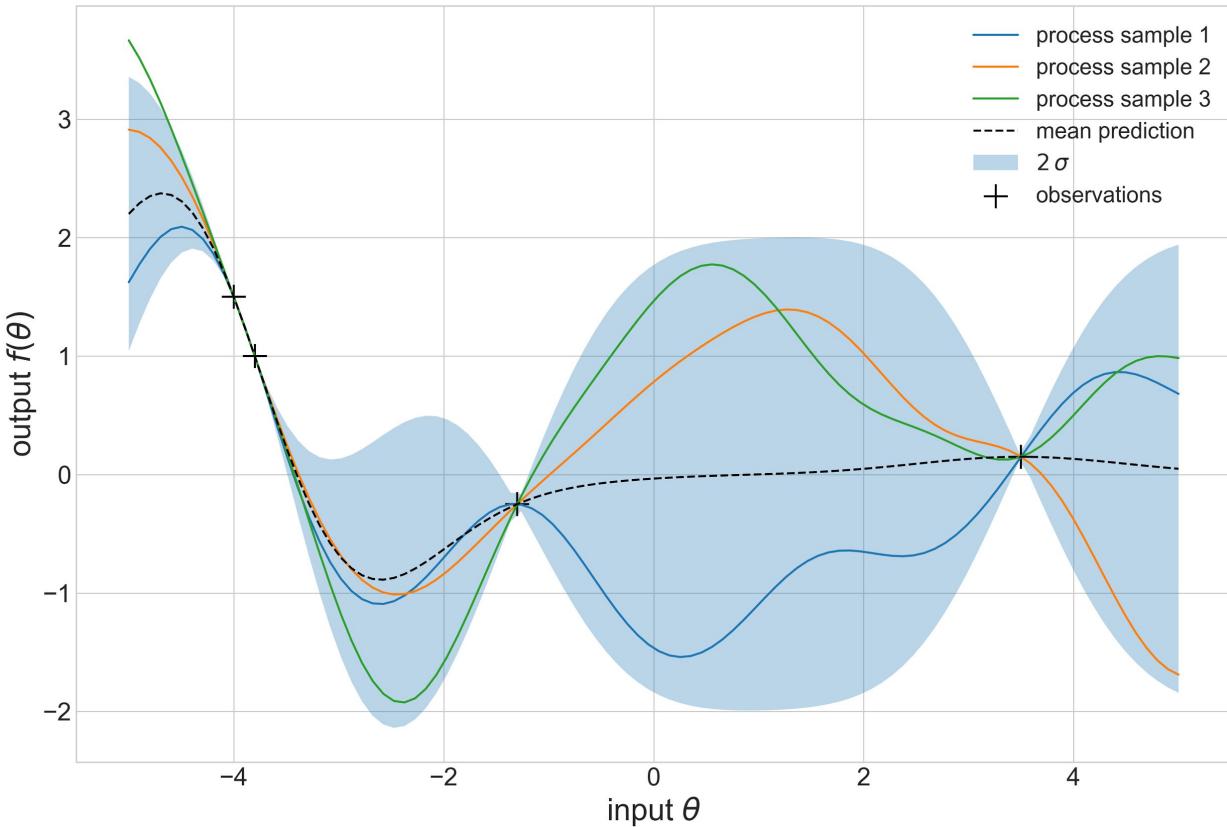
Bayesian Optimization

- Gaussian Process Prior of objective function
- Measure and update Prior
- Choose next point to measure and update
- Again



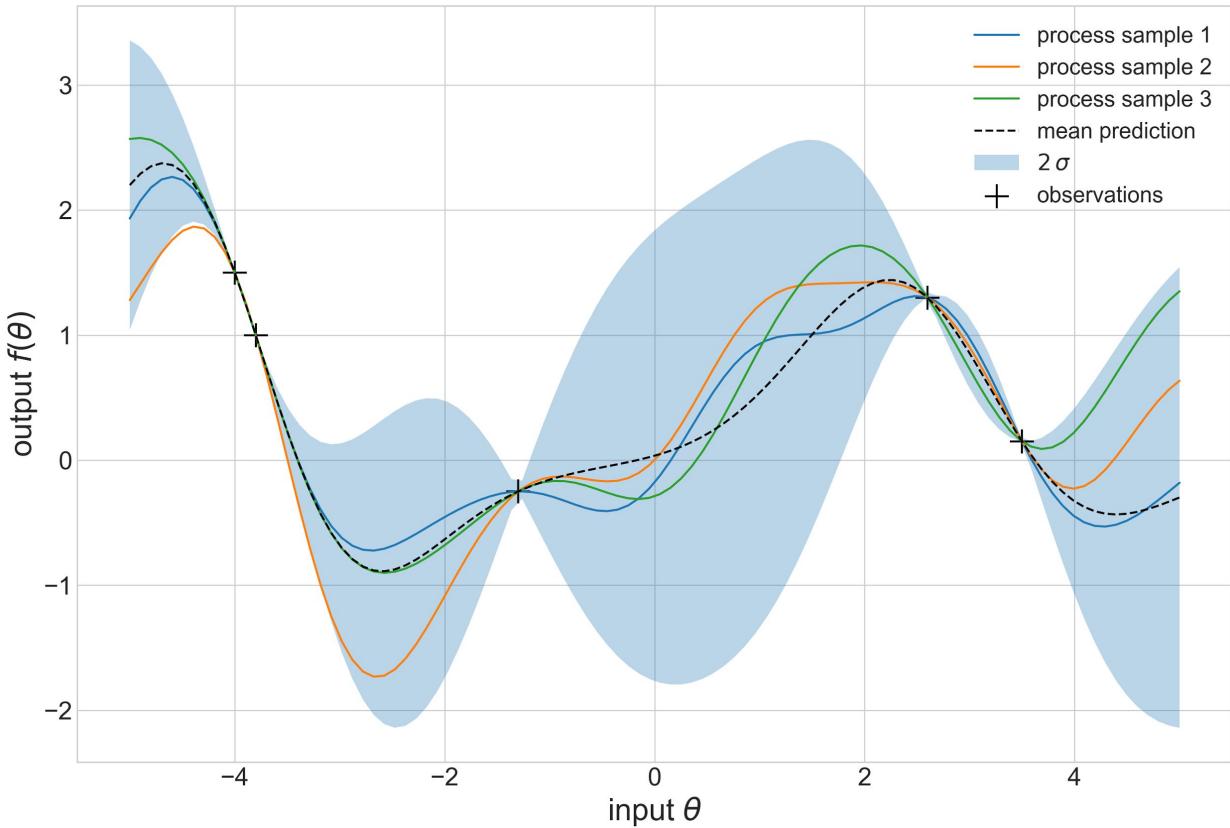
Bayesian Optimization

- Gaussian Process Prior of objective function
- Measure and update Prior
- Choose next point to measure and update
- Again
- ...



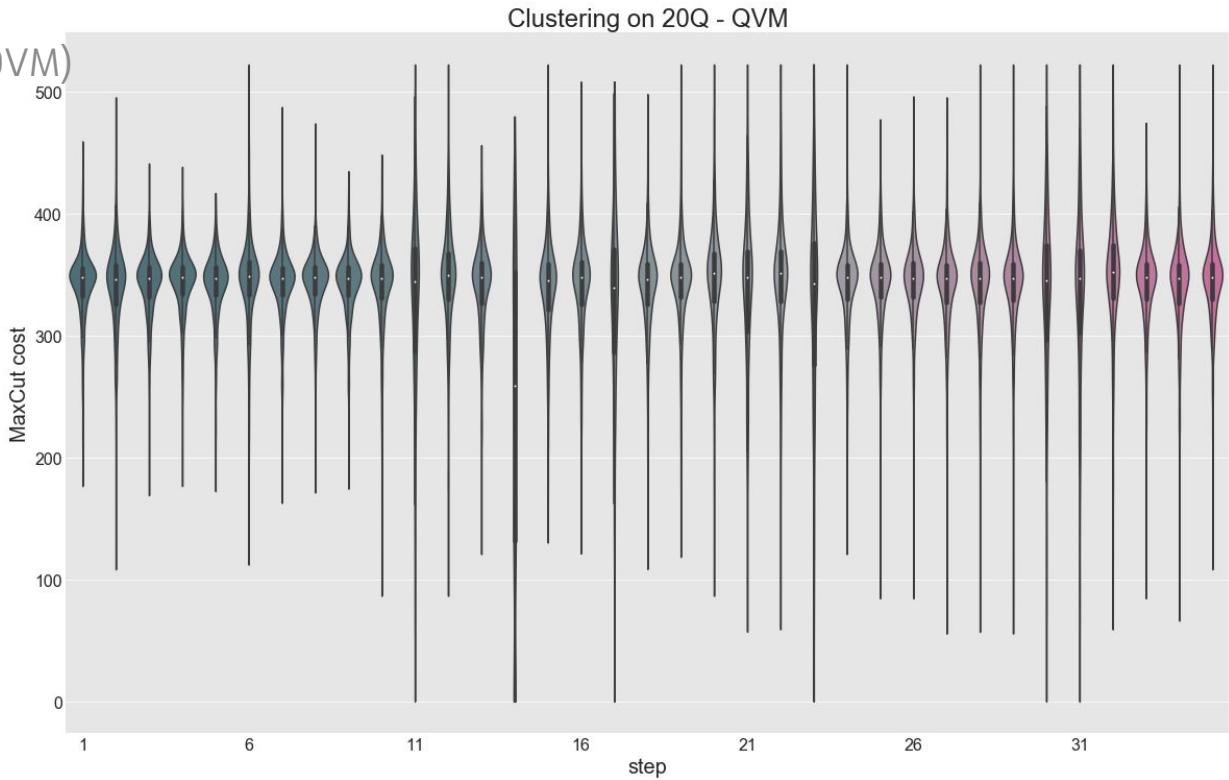
Bayesian Optimization

- Gaussian Process Prior of objective function
- Measure and update Prior
- Choose next point to measure and update
- Again
- ...
- ...



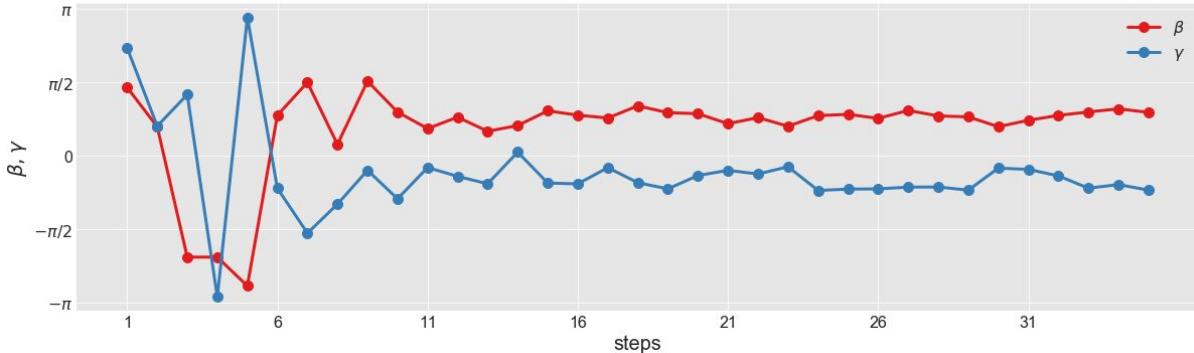
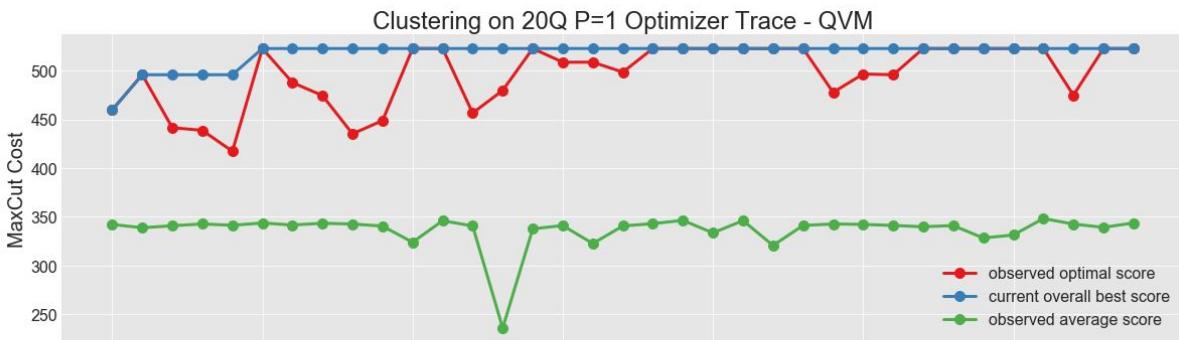
That's how it looks in practice

- Fully Connected Graph for clustering
- Noiseless Simulator (Rigetti QVM)
- p=1 QAOA



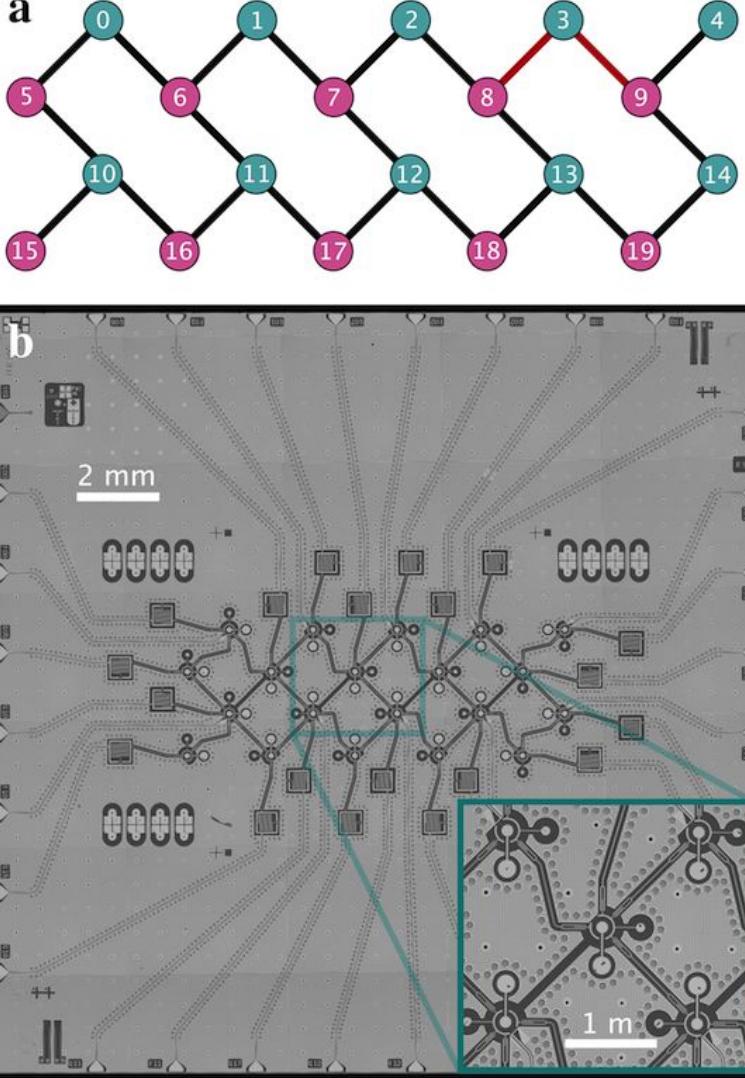
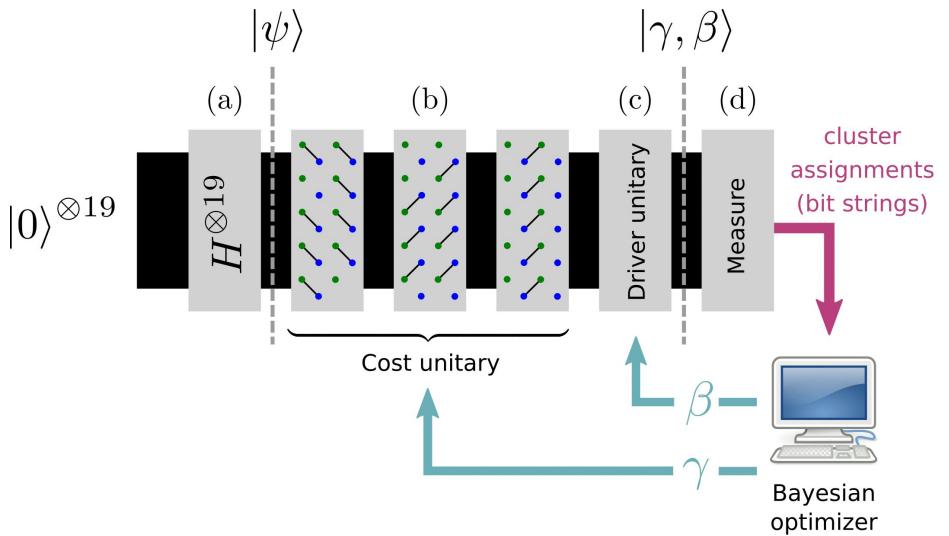
That's how it looks in practice

- Fully Connected Graph for clustering
- Noiseless Simulator (Rigetti QVM)
- p=1 QAOA



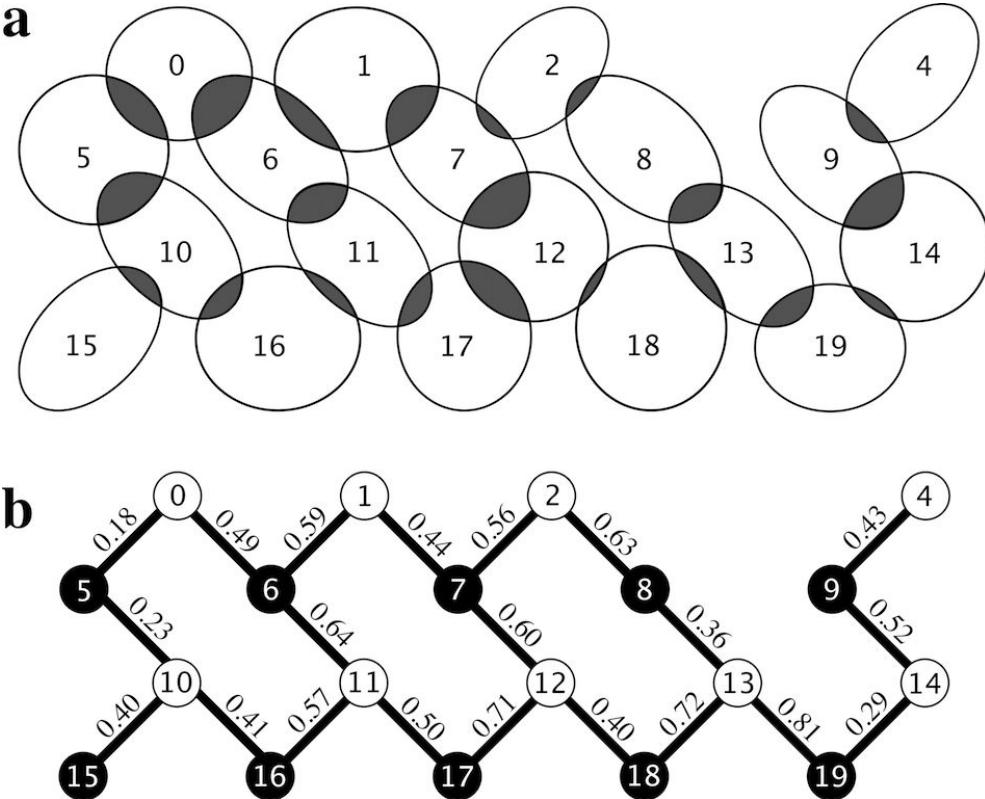
Clustering on a 19-Q Chip

- Chip topology requires smart gate sequence for QAOA to execute all gates on a vertex
- Staggering gate applications according to edge-coloring

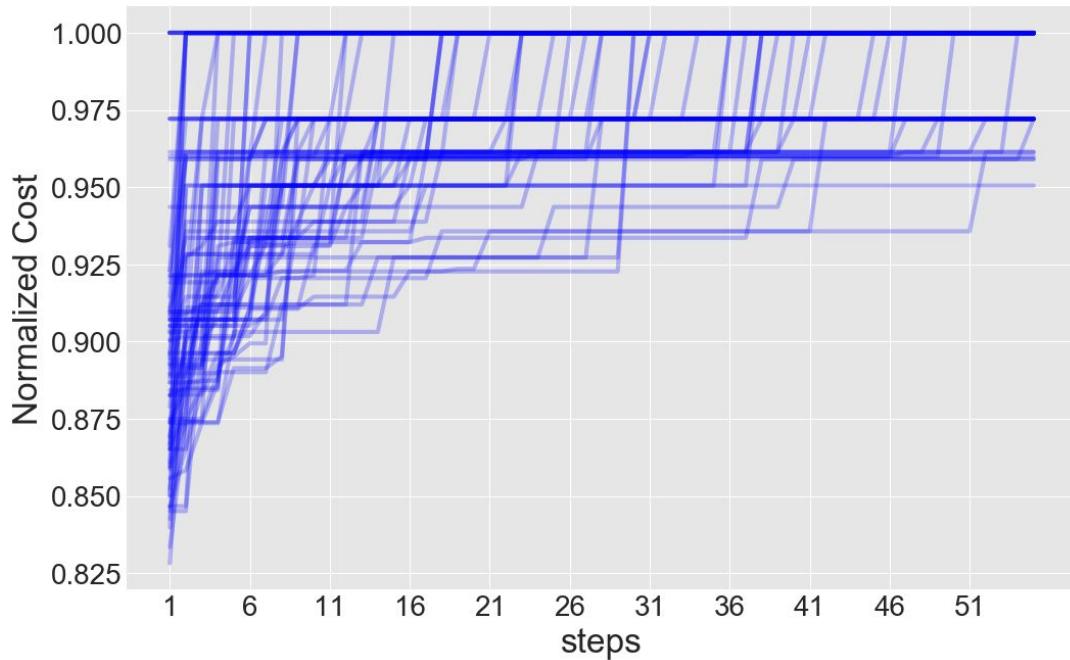


Clustering on a 19-Q Chip

- Moderate coherence times
- Moderate 2Q fidelities
- Moderate readout fidelities
- Demonstration with chip specific problem taylored to the topology
- *Overlap* problem similar to VLSI design



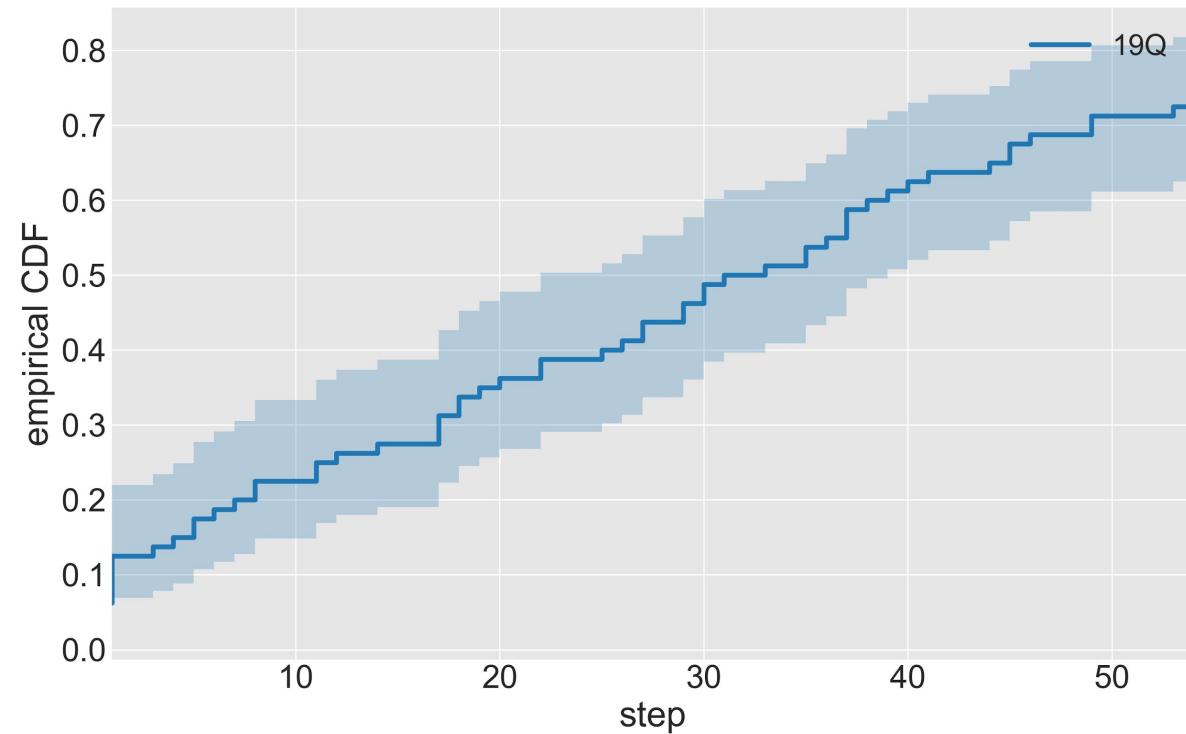
Putting it all together



- 83 trial runs on the QPU
- p=1 QAOA, i.e. single application of U and V
- Algorithm finds the optimum most of the time
- Calculate eCDF form the traces



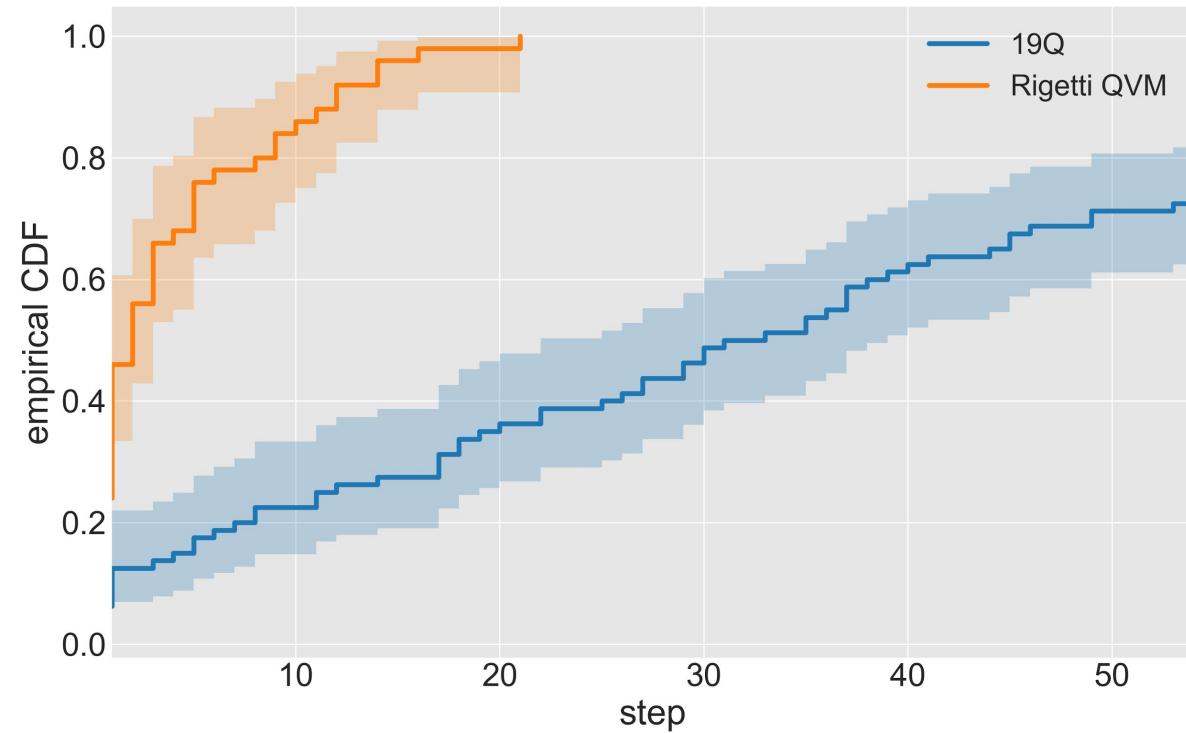
Empirical performance



- Success probability monotonically increases with number of steps.



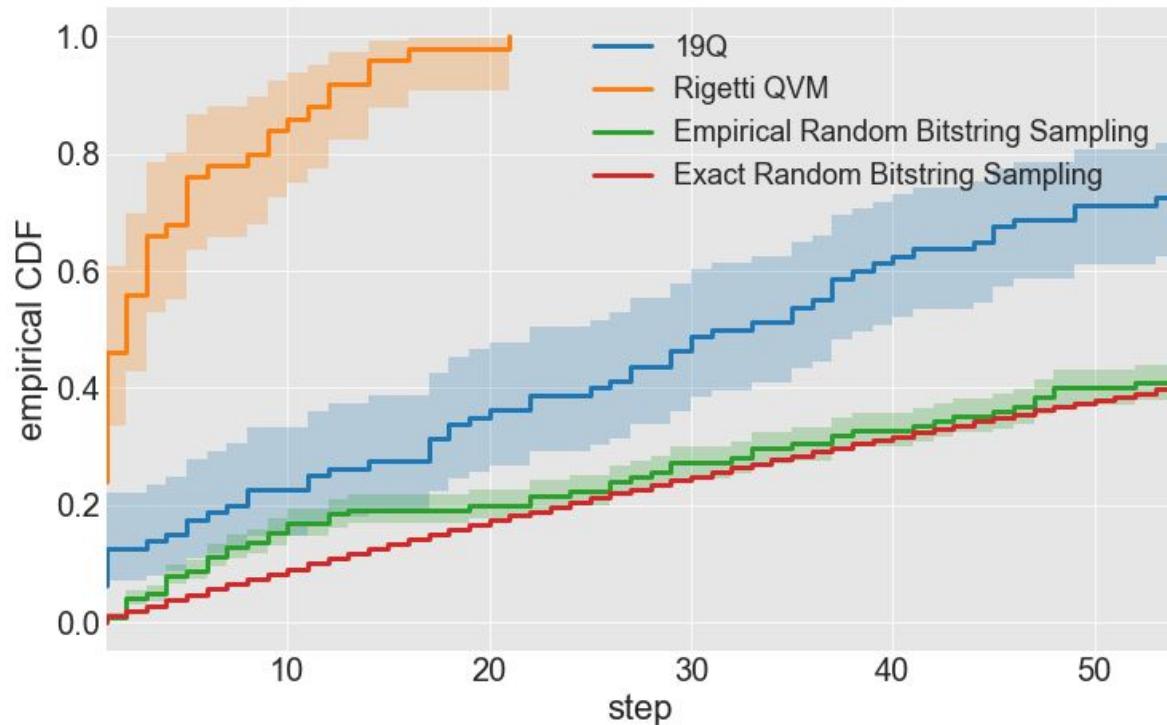
Empirical performance



- Success probability monotonically increases with number of steps.
- Noise in 19Q has a significant impact on performance.



Empirical performance



- Success probability monotonically increases with number of steps.
- Noise in 19Q has a significant impact on performance.
- Approach clearly outperforms random sampling.



Forest



Join our community Slack:

slack.rigetti.com



Find us on Github:

github.com/rigetticomputing

Sign-Up @ rigetti.com/forest

QPU access @ rigetti.com/qpu-request



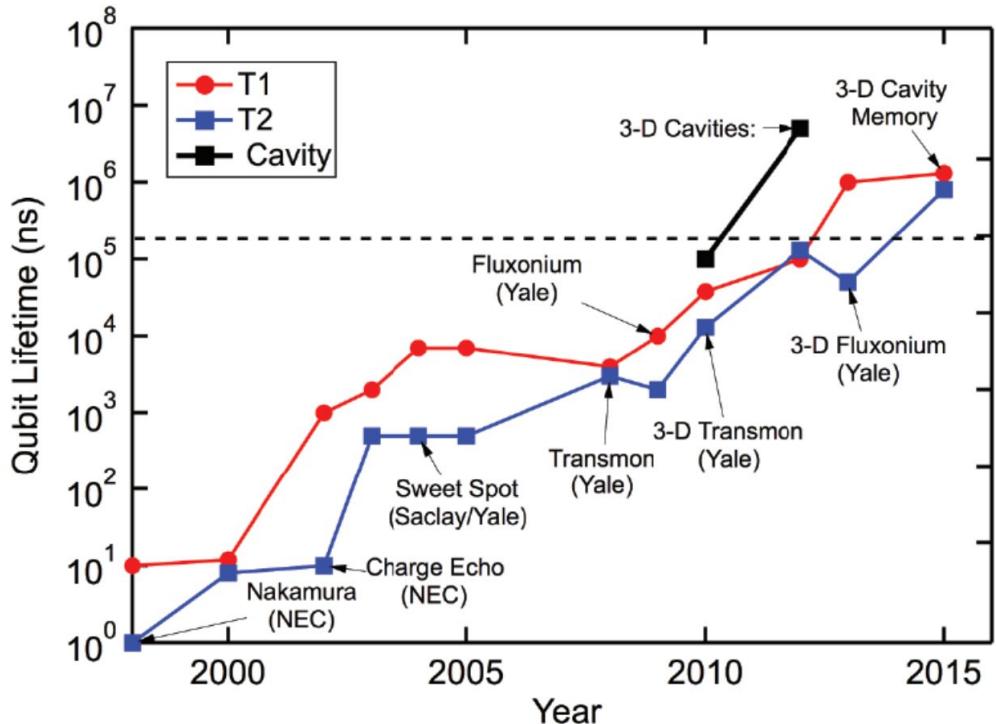
Thank you

More details in our pre-print arXiv: 1712.05771

Spare slides



15 years of exponential performance improvement



Trends in “modern” superconducting qubits:

- > All (or mostly) RF control
- > Dispersive readout
- > 3D cavity resonators



Quantum Fourier Transform (QFT)

Discrete Fourier Transform (DFT)

Fourier conjugates \mathbf{q}, \mathbf{p} (vectors)

Vector: $\mathbf{q} = (q_0, q_1, \dots, q_{N-1})$

N is a power of two

$$\text{DFT}[\mathbf{q}] = \mathbf{p}$$

$$\text{DFT}[\mathbf{p}] = \mathbf{q}$$

Quantum Fourier Transform

Fourier conjugates $|q\rangle, |p\rangle$ (state vectors)

State: $|q\rangle = q_0|0\rangle + q_1|1\rangle + \dots + q_{N-1}|N-1\rangle$
(Basis vectors explicit)

N is a power of two:

- $N=2^n$ with n qubits

$$\text{QFT}|q\rangle = |p\rangle$$

$$\text{QFT}|q\rangle = |p\rangle$$

Both:

$$p_k = \frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} q_j e^{2\pi i j k / N}$$

Quantum Fourier Transform (QFT)

Quantum Fourier Transform

- Is unitary ✓
- Is faster than DFT
- Is an important subroutine of other algorithms

$$p_k = \frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} q_j e^{2\pi i j k / N}$$

Quantum Fourier Transform (QFT)

Quantum Fourier Transform

- Is unitary ✓
- Is faster than DFT
- Is an important subroutine of other algorithms

$$p_k = \frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} q_j e^{2\pi i j k / N}$$

$$|q\rangle = q_0|0\rangle + q_1|1\rangle + \dots + q_N|N-1\rangle$$

Decimal labels

$$= q_{0\dots 00}|0\dots 00\rangle + q_{0\dots 01}|0\dots 01\rangle + \dots + q_{11\dots 1}|11\dots 1\rangle$$

Binary labels

Quantum Fourier Transform (QFT)

Quantum Fourier Transform

- Is unitary ✓
- Is faster than DFT
- Is an important subroutine of other algorithms

$$p_k = \frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} q_j e^{2\pi i j k / N}$$

$$\begin{aligned}|q\rangle &= q_0|0\rangle + q_1|1\rangle + \dots + q_N|N-1\rangle && \text{Decimal labels} \\&= q_{0\dots 00}|0\dots 00\rangle + q_{0\dots 01}|0\dots 01\rangle + \dots + q_{11\dots 1}|11\dots 1\rangle && \text{Binary labels}\end{aligned}$$

Fact:

$$\text{QFT}|j_n, \dots, j_1\rangle = \frac{1}{\sqrt{2^n}} \left(|0\rangle + e^{i\varphi_n} |1\rangle \right) \otimes \dots \otimes \left(|0\rangle + e^{i\varphi_1} |1\rangle \right)$$

$$\varphi_n \equiv 2\pi(j_1/2^n + j_2/2^{n-1} + \dots + j_n/2)$$

Quantum Fourier Transform (QFT)

Concrete example: $|q\rangle = |j_3 j_2 j_1\rangle = |101\rangle$

$$\varphi_n \equiv 2\pi(j_1/2^n + j_2/2^{n-1} + \dots + j_n/2)$$

Quantum Fourier Transform (QFT)

Concrete example: $|q\rangle = |j_3 j_2 j_1\rangle = |101\rangle$

$$\varphi_n \equiv 2\pi(j_1/2^n + j_2/2^{n-1} + \dots + j_n/2)$$

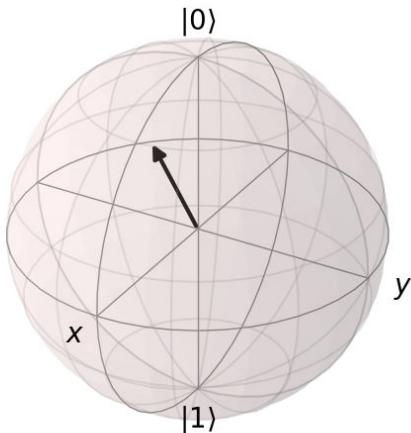
$$\text{QFT}|101\rangle = \frac{1}{\sqrt{2^n}} \left(|0\rangle + e^{i2\pi(5/8)}|1\rangle \right) \otimes \left(|0\rangle + e^{i2\pi(1/4)}|1\rangle \right) \otimes \left(|0\rangle + e^{i2\pi(1/2)}|1\rangle \right)$$

Quantum Fourier Transform (QFT)

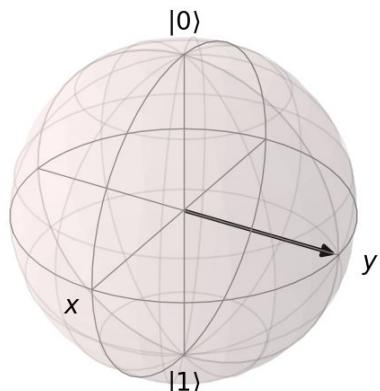
Concrete example: $|q\rangle = |j_3 j_2 j_1\rangle = |101\rangle$

$$\varphi_n \equiv 2\pi(j_1/2^n + j_2/2^{n-1} + \dots + j_n/2)$$

$$\text{QFT}|101\rangle = \frac{1}{\sqrt{2^n}} \left(|0\rangle + e^{i2\pi(5/8)}|1\rangle \right) \otimes \left(|0\rangle + e^{i2\pi(1/4)}|1\rangle \right) \otimes \left(|0\rangle + e^{i2\pi(1/2)}|1\rangle \right)$$

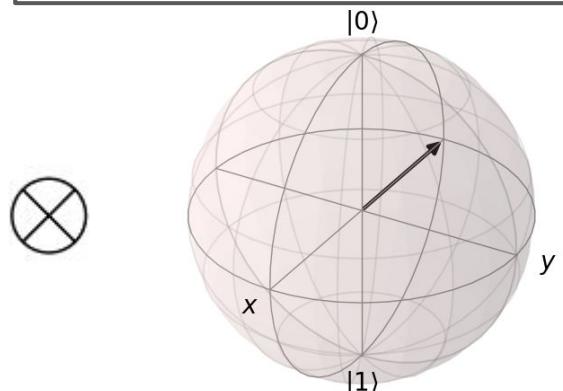


$$R_z(\varphi_3 = 5\pi/4)$$



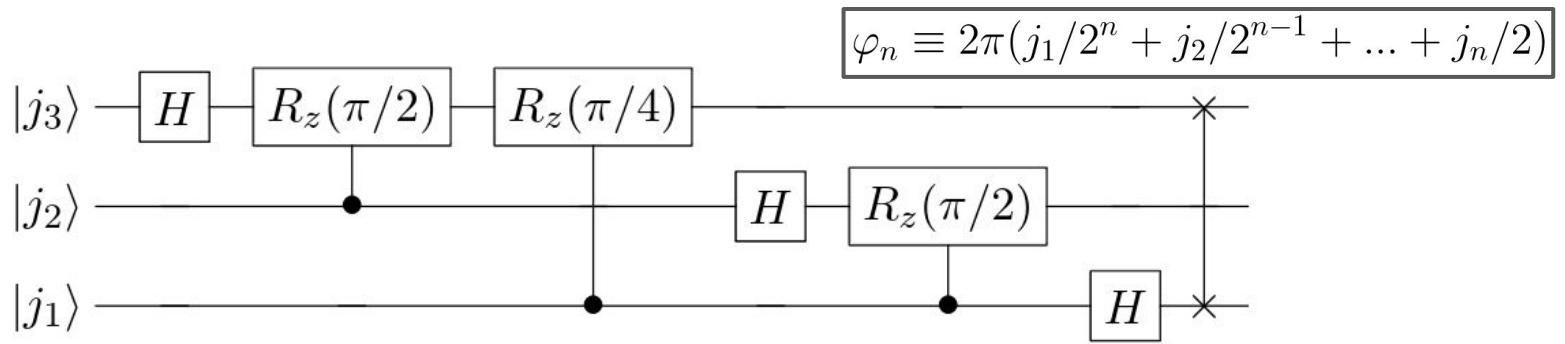
$$R_z(\varphi_2 = \pi/2)$$

$$|\psi\rangle = \cos(\theta/2)|0\rangle + e^{i\varphi}\sin(\theta/2)|1\rangle$$

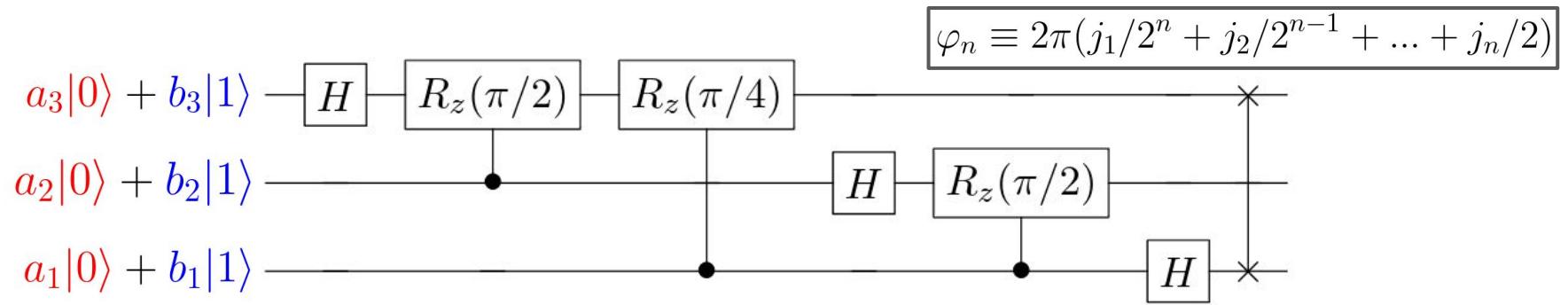


$$R_z(\varphi_1 = \pi)$$

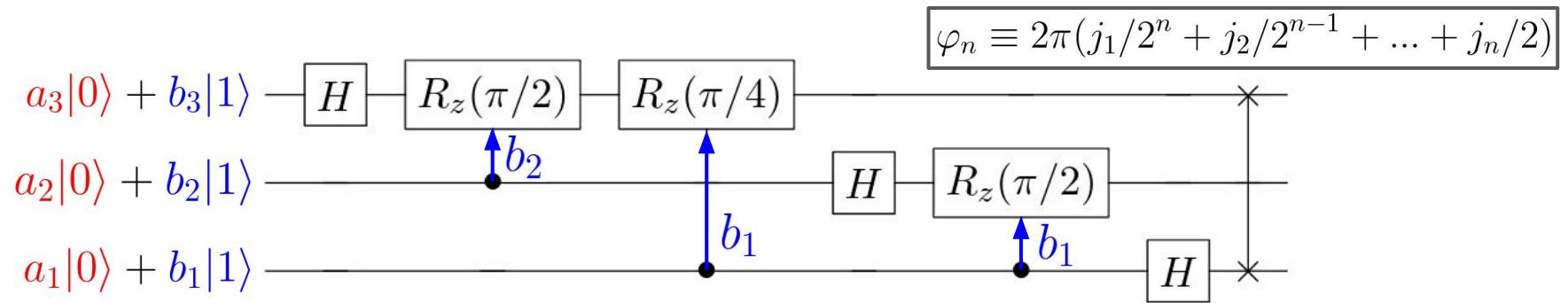
Quantum Fourier Transform (QFT)



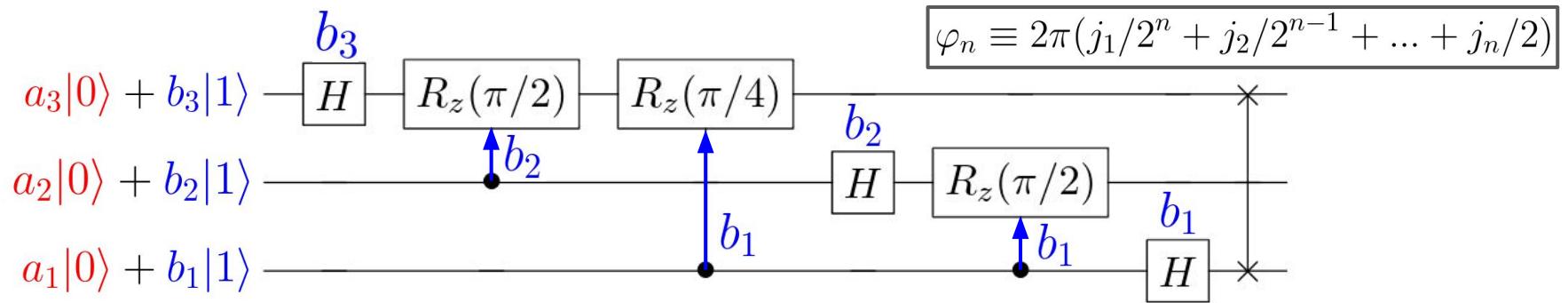
Quantum Fourier Transform (QFT)



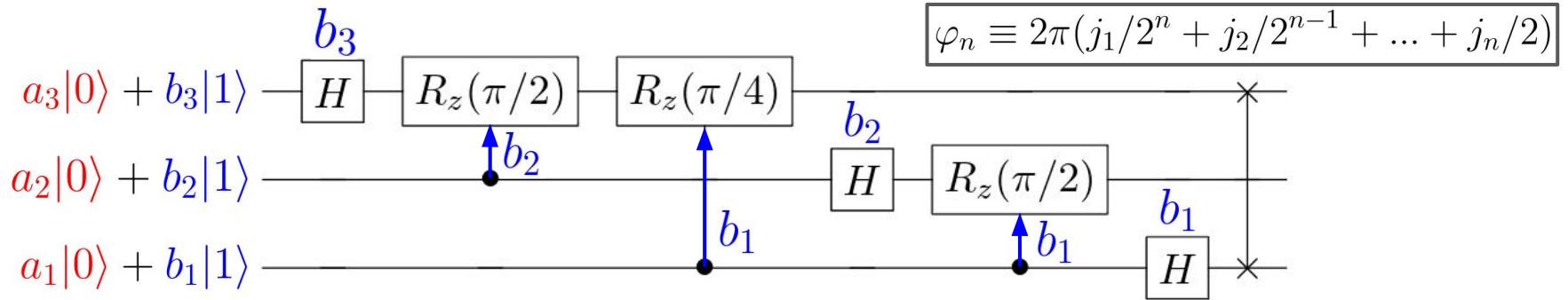
Quantum Fourier Transform (QFT)



Quantum Fourier Transform (QFT)



Quantum Fourier Transform (QFT)



$$p_k = \frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} q_j e^{2\pi i j k / N}$$

Fast Fourier Transform: $\Theta(n2^n)$
Quantum Fourier Transform: $\Theta(n^2)$

Quantum Fourier Transform (QFT)

Quantum Fourier Transform

- Is unitary ✓
- Is faster than Fast Fourier Transform
- Is an important subroutine of other algorithms

Fast Fourier Transform: $\Theta(n2^n)$

Quantum Fourier Transform: $\Theta(n^2)$

$$p_k = \frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} q_j e^{2\pi i j k / N}$$

Caveats:

- Can't directly read out p_k
→ use as subroutine
- State preparation of $|q\rangle$ is inefficient
→ restricted to simple initial states

Quantum Fourier Transform (QFT)

Quantum Fourier Transform

- Is unitary ✓
- Is faster than Fast Fourier Transform
- Is an important subroutine of other algorithms

$$p_k = \frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} q_j e^{2\pi i j k / N}$$

Fast Fourier Transform: $\Theta(n2^n)$

Quantum Fourier Transform: $\Theta(n^2)$

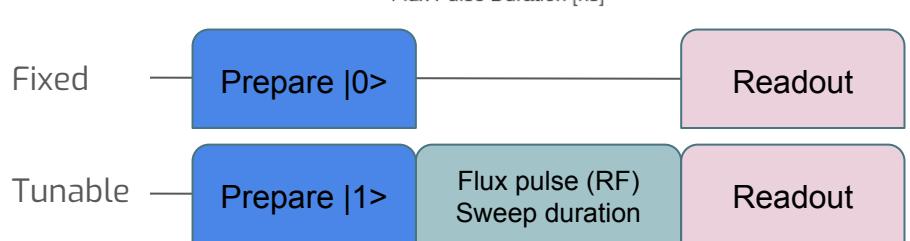
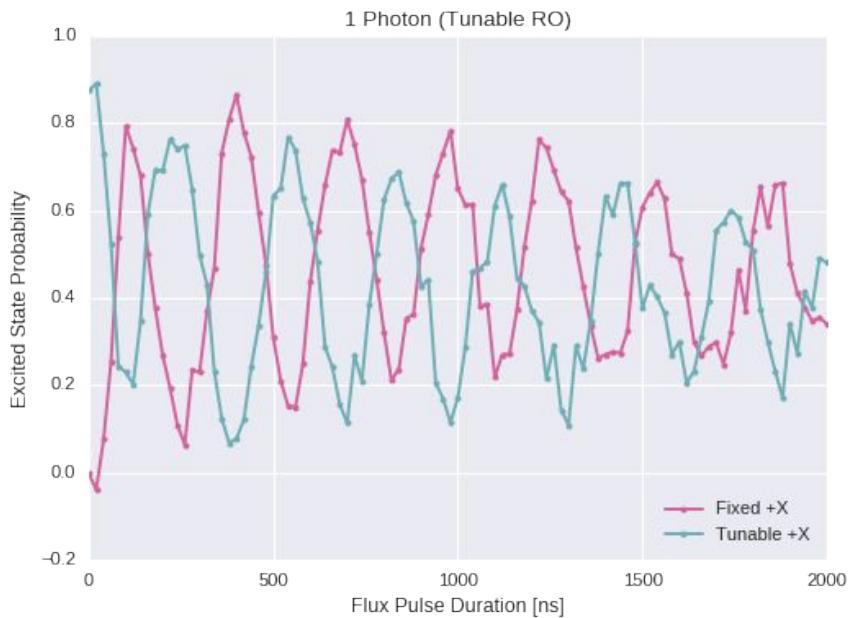
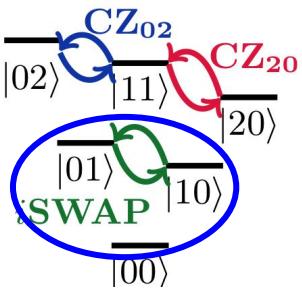
Caveats:

- Can't directly read out p_k
→ use as subroutine
- State preparation of $|q\rangle$ is inefficient
→ restricted to simple initial states

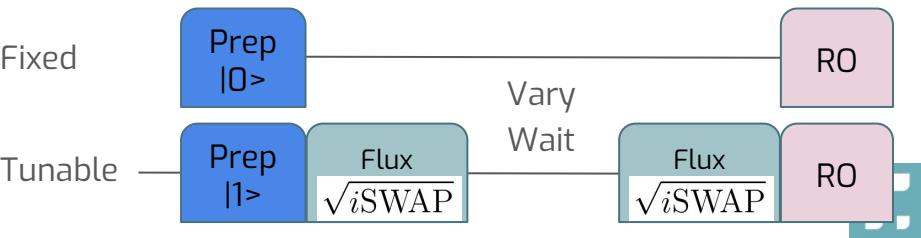
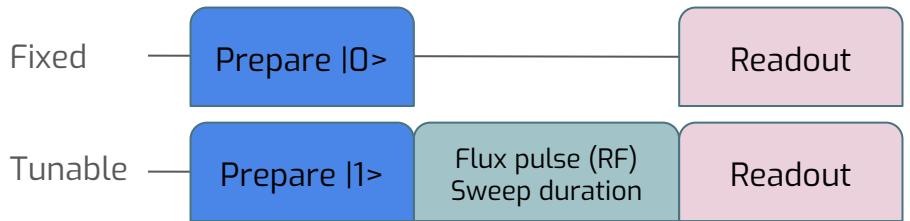
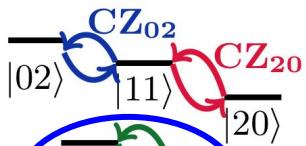
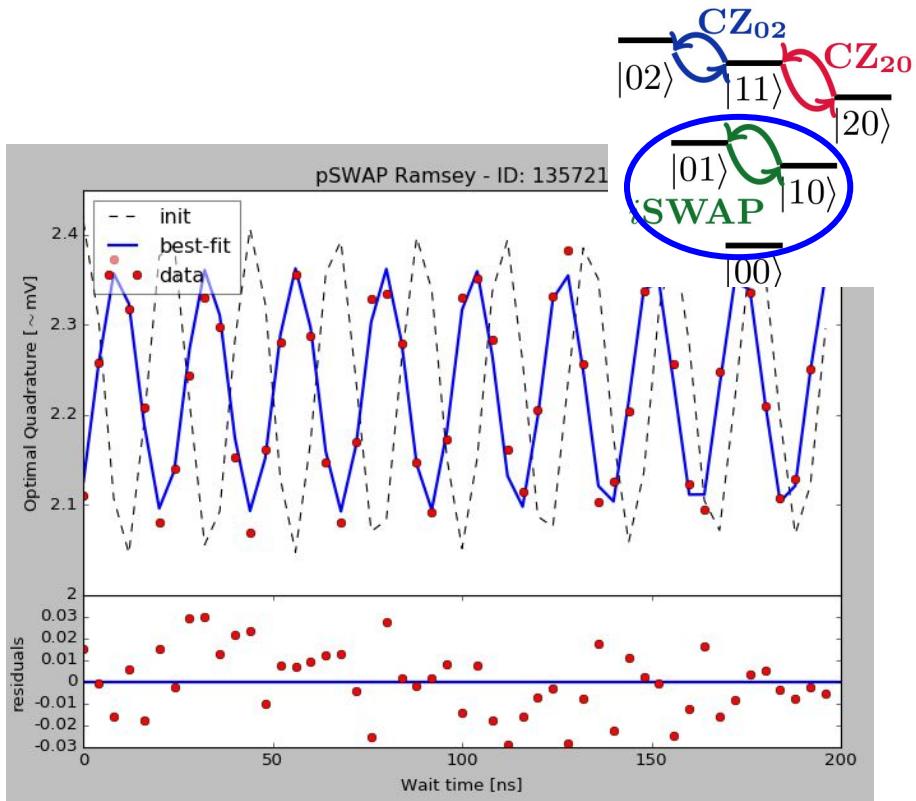
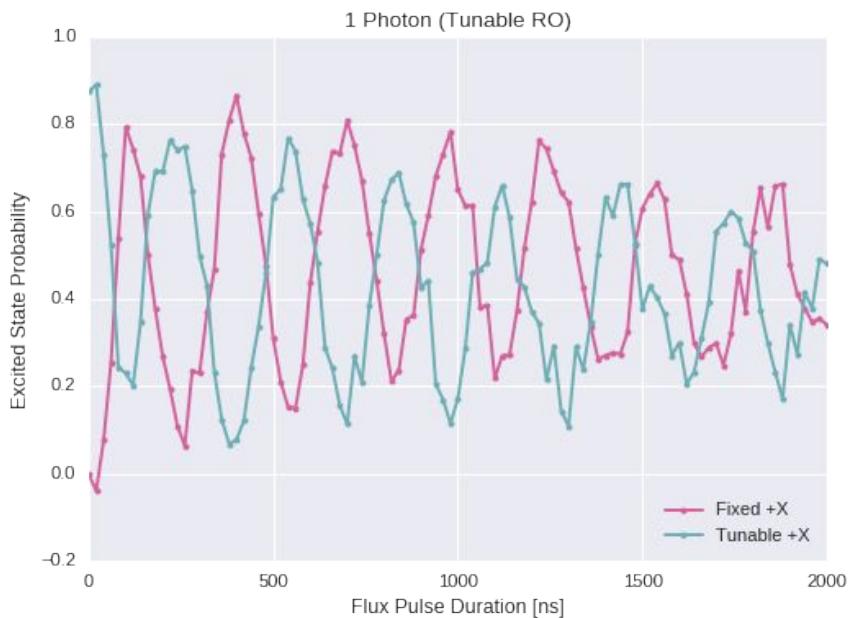
QFT descendants:

- Phase estimation
- Order finding
- Prime number factorization

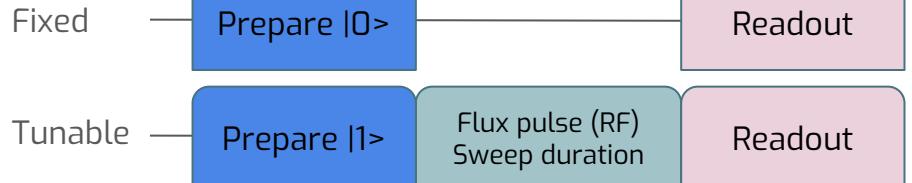
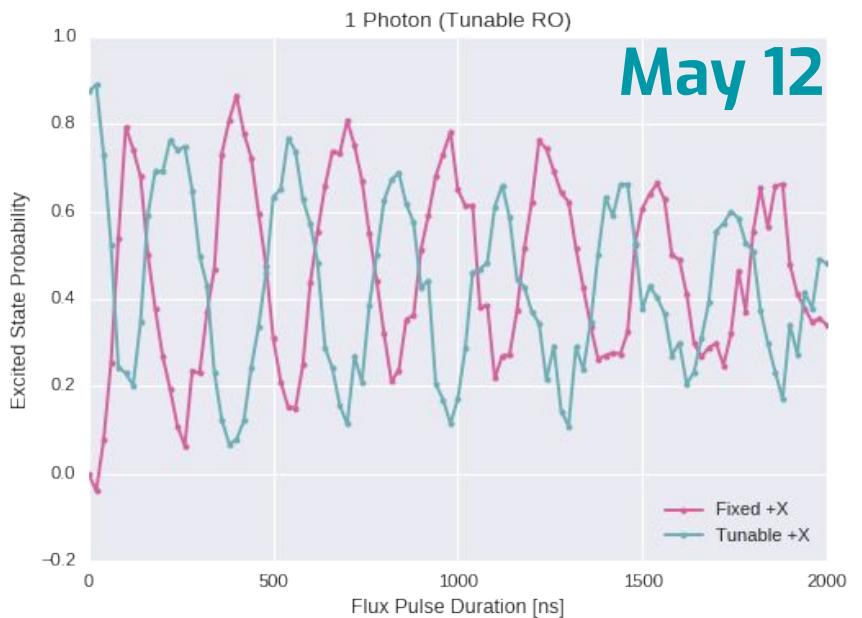
Proof of principle



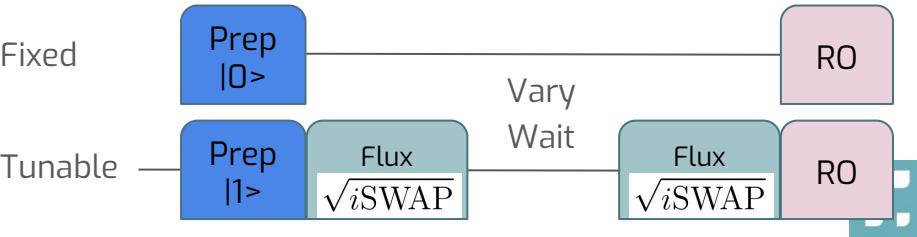
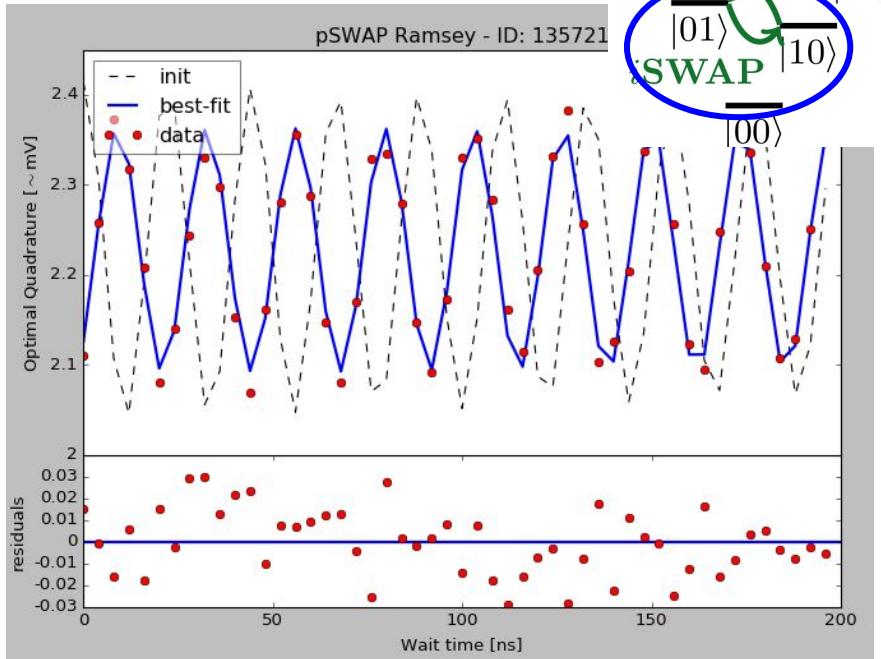
Proof of principle



Proof of principle



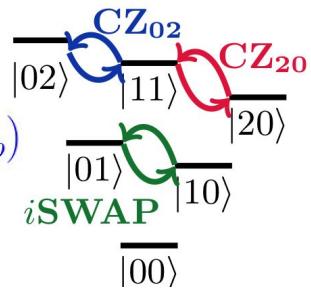
May 16



Theory

$$\Phi(t) = \bar{\Phi} + \tilde{\Phi} \cos(\omega_p t + \theta_p)$$

$$\omega_T(t) \approx \overline{\omega_T}(\tilde{\Phi}) + \widetilde{\omega_T}(\tilde{\Phi}) \cos(2\omega_p t + 2\theta_p)$$



Resonant frequencies

$$f[\cos \phi_{\text{ext}}(t)] \simeq \bar{f} + \tilde{f} \cos[2(\omega_p t + \theta_p)], \quad (34)$$

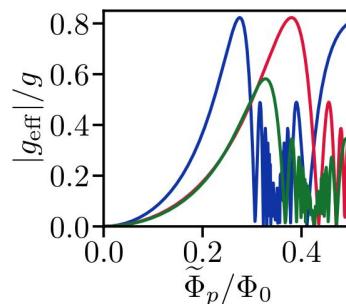
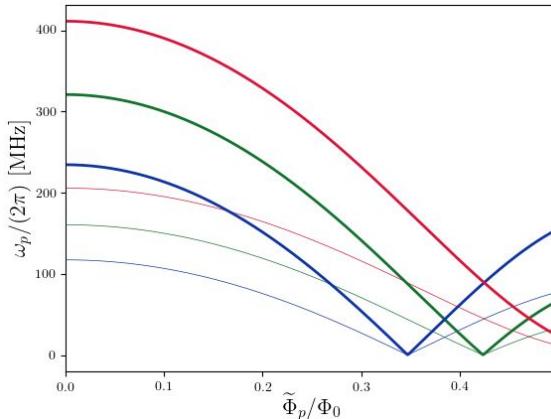
$$\begin{aligned} \bar{f} &= f^{(0)}[J_0(\tilde{\phi}_p)] + J_2^2(\tilde{\phi}_p)f^{(2)}[J_0(\tilde{\phi}_p)] \\ &+ J_2^2(\tilde{\phi}_p)J_4(\tilde{\phi}_p)f^{(3)}[J_0(\tilde{\phi}_p)] + \frac{1}{4}J_2^4(\tilde{\phi}_p)f^{(4)}[J_0(\tilde{\phi}_p)] \end{aligned} \quad (35)$$

$$\begin{aligned} \tilde{f} &= -2J_2(\tilde{\phi}_p)\{f^{(1)}[J_0(\tilde{\phi}_p)] + J_4(\tilde{\phi}_p)f^{(2)}[J_0(\tilde{\phi}_p)] \\ &+ \frac{1}{2}J_2^2(\tilde{\phi}_p)f^{(3)}[J_0(\tilde{\phi}_p)] + \frac{2}{3}J_2^2(\tilde{\phi}_p)J_4(\tilde{\phi}_p)f^{(4)}[J_0(\tilde{\phi}_p)]\}, \end{aligned} \quad (36)$$

Effective couplings

$$\begin{aligned} g_{11}^{(n)} &= \bar{g}_{11} J_n \left(\frac{\tilde{\omega}_{T_{01}}}{2\omega_p} \right) \\ &- \frac{1}{2} \tilde{g}_{11} \left[J_{n-1} \left(\frac{\tilde{\omega}_{T_{01}}}{2\omega_p} \right) + J_{n+1} \left(\frac{\tilde{\omega}_{T_{01}}}{2\omega_p} \right) \right], \end{aligned} \quad (41)$$

$$\begin{aligned} g_{21}^{(n)} &= \bar{g}_{21} J_n \left(\frac{\tilde{\omega}_{T_{01}}}{2\omega_p} \right) \\ &- \frac{1}{2} \tilde{g}_{21} \left[J_{n-1} \left(\frac{\tilde{\omega}_{T_{01}}}{2\omega_p} \right) + J_{n+1} \left(\frac{\tilde{\omega}_{T_{01}}}{2\omega_p} \right) \right], \end{aligned} \quad (42)$$



Why do we need to schedule?

- Quil has **no** notion of time or synchronization.
- But time and synchronization are very important.
- What are our options?

**Give up;
Admit the physicists are better**

"Program" with buttons and wires.

**Include *ad hoc*
synchronization instructions**

Extend Quil to "know" about time.

**Compile Quil into some
temporal representation**

Add machine-specific directives.

Pros:

- Maximal control

Cons:

- Difficult to reason about
- Nixes the idea of an abstraction
- Difficult to automate
- Have to think about hardware

Pros:

- Directly addresses the issue
- Still an abstract framework

Cons:

- Extremely complicated!
- Difficult to reason about
- Not easily extensible
- Hard to implement
- Loses the "essence"

Pros:

- Remains abstract
- Adds control as necessary
- Extensible!
- Keeps Quil "clean"

Cons:

- Compilation is more difficult
- Performance characterization is machine-specific

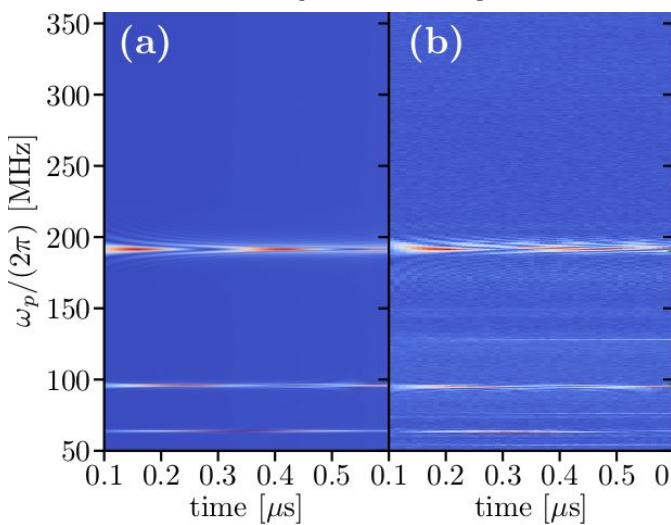


Parametric entangling gates

iSWAP

Theory

Experiment



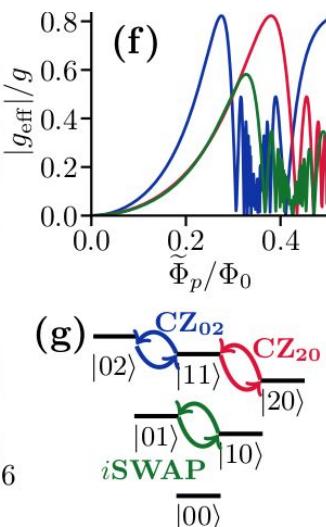
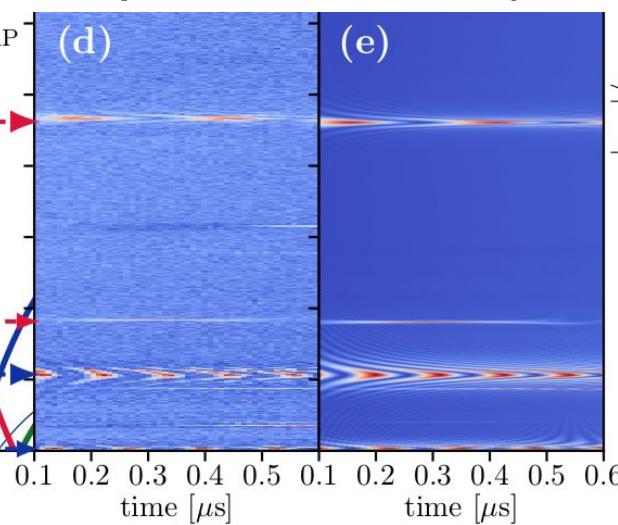
Theory

Experiment

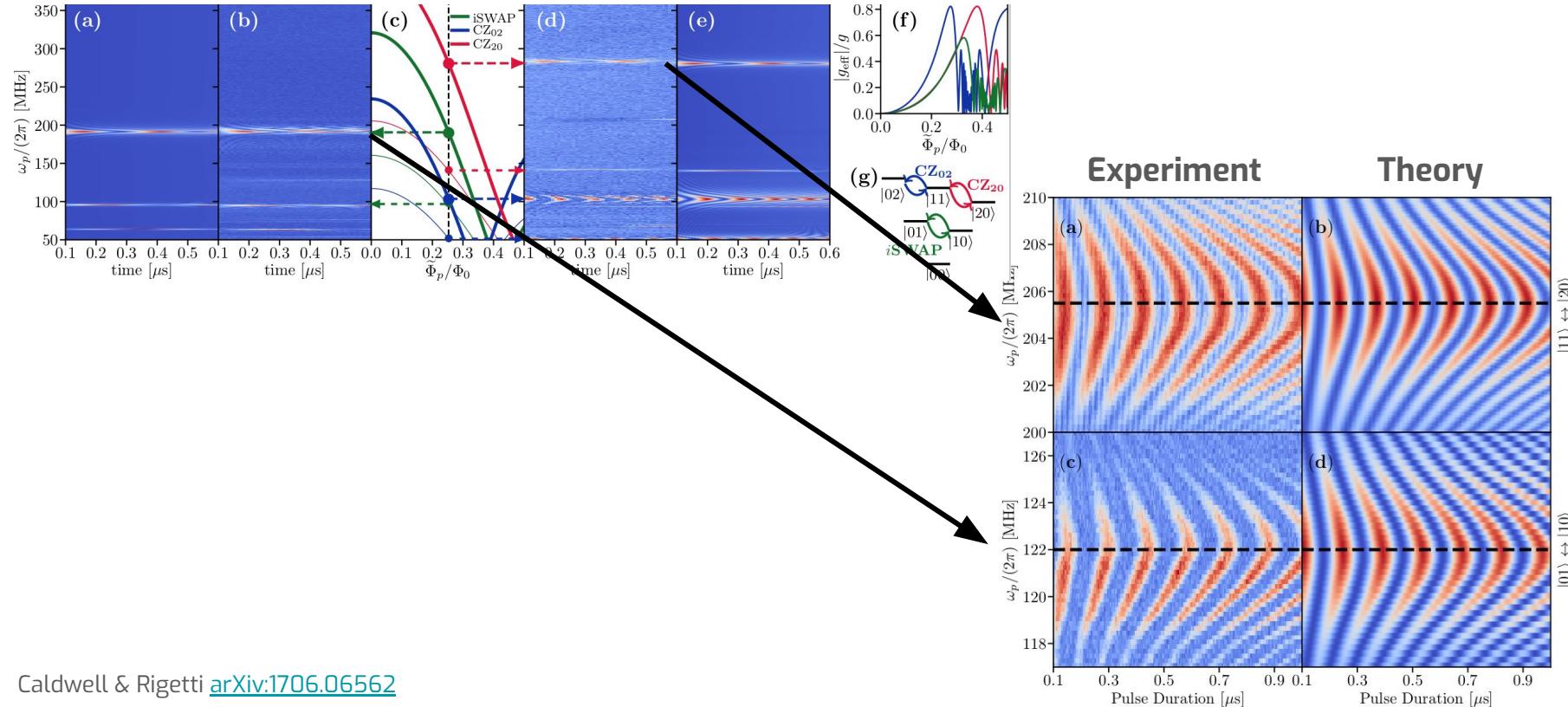
CZ₀₂ & CZ₂₀

Theory

Theory



Parametric entangling gates



Parametric entangling gates

TABLE II. Characteristics of the two-qubit CZ gates performed between neighboring qubit pairs (Q_0, Q_1) , (Q_1, Q_2) , and (Q_2, Q_3) . g represents the qubit-qubit coupling, $\Delta_{11 \leftrightarrow 02}$ the detuning between $|11\rangle$ and $|02\rangle$, $\Delta_{11 \leftrightarrow 20}$ the detuning between $|11\rangle$ and $|20\rangle$, ω_m the modulation frequency, $\delta\omega$ the effective detuning of the tunable qubit under modulation, $T_{2,\text{eff}}^*$ the effective coherence time of the tunable qubit under modulation, τ the duration of the CZ gate, and \mathcal{F}_{QPT} the two-qubit gate fidelity measured by quantum process tomography. The symbol † denotes the transitions used for the gate.

Qubit pair index	$g/2\pi$ (MHz)	$\Delta_{11 \leftrightarrow 02}/2\pi$ (MHz)	$\Delta_{11 \leftrightarrow 20}/2\pi$ (MHz)	$\omega_m/2\pi$ (MHz)	$\delta\omega/2\pi$ (MHz)	$T_{2,\text{eff}}^*$ (μs)	τ (ns)	\mathcal{F}_{QPT} %
$Q_0 - Q_1$	3.8	69.2 †	315.0	83.3	281	3.8	278	95
$Q_1 - Q_2$	4.2	187.3 †	180.1	82.9	338	3.0	353	93
$Q_2 - Q_3$	4.2	855.1	1240.3 †	199.9	257	5.2	395	91

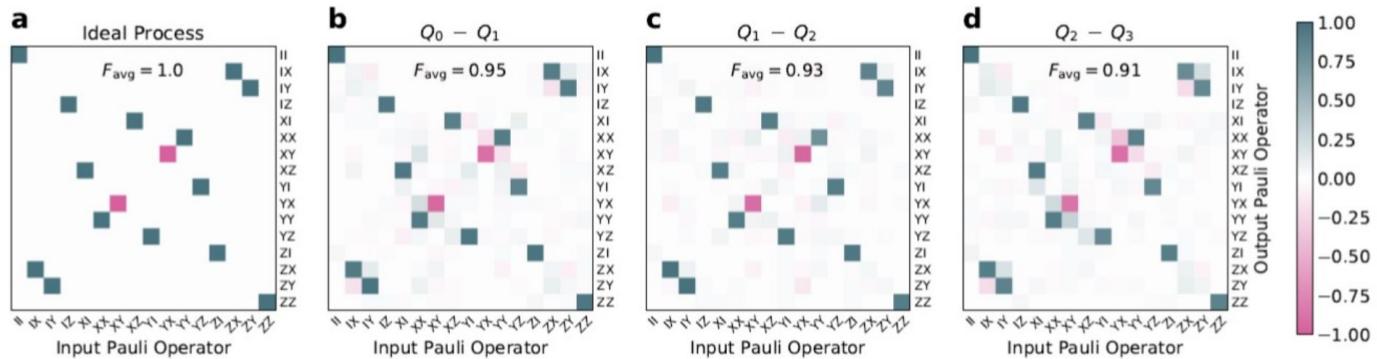


FIG. 3. **Quantum process tomography.** Process matrices of **a**, the ideal process, and CZ gates between **b**, $Q_0 - Q_1$, **c**, $Q_1 - Q_2$, and **d**, $Q_2 - Q_3$. The achieved average fidelities are measured to be 95%, 93%, and 91%, respectively.



Forest 1.0

Analytical modeling of parametrically-modulated transmon qubits

Nicolas Didier, Eyob A. Sete, Marcus P. da Silva, and Chad Rigetti Computing, 775 Heinz Avenue, Berkeley, CA 94710
(Dated: June 21, 2017)

Scaling up quantum machines requires developing appropriate models to understand their complex quantum dynamics. We focus on superconducting quantum transmons for which full numerical simulations are already challenging at the basis thus highly desirable to develop accurate methods of modeling qubit networks via numerical computations. Using systematic perturbation theory to large order regime, we derive precise analytic expression of the transmon parameters. We turn to the case of parametrically-modulated transmons to study recently-implemented activated entangling gates.

I. INTRODUCTION

Scaling up quantum machines is a challenging enterprise that requires accurate modeling of complex quantum dynamics. Precise understanding is crucial to design, manipulate, optimize, and verify the machine. In the field of superconducting quantum computers, transmons [1, 2] are currently widely used as qubits [3–16] or quantum devices [17–19]. Transmons are weakly nonlinear oscillators based on the Cooper pair box, a Josephson junction shunted by a capacitance. The transmon regime corresponds to a large Josephson energy compared to the charging energy—it is a compromise between a large anharmonicity and a weak sensitivity to charge noise. The coherence and gate times of transmons in quantum computing experiments have been steadily improving over the last several years, and transmons are now one of the leading candidates to an architecture that can satisfy stringent requirements of fault-tolerant quantum computing [20].

Although analytical expressions for the behaviour of non-interacting transmons are well understood, the accurate description for the bidirectionally interacting transmons requires the diagonalization of coupled systems (i.e., the charge basis description) of the transmons with charge dipole interactions. Numerical diagonalization of these systems quickly becomes intractable because a large number of basis states are necessary to obtain high accuracy [21] for non-interacting transmons. A more efficient approach is to use analytical expressions of transmon energies and states. Exact diagonalization of the Cooper-pair box Hamiltonian is achieved with Mathieu functions [21, 22], but manipulating them can be cumbersome. For example, calculating the Fourier transform of Mathieu functions, necessary to describe capacitive couplings, leads to rather complex expressions. An alternative is to consider controlled approximations, such as the approximate diagonalization via standard perturbation theory, which is widely used in quantum mechanics [23]. For transmons, the natural small parameter is the ratio of the charging energy of the Cooper-pair box to the Josephson energy of the junction, as this parameter is typically below 2%.

In this paper, we apply our model interacting terms with respect to numerical analytical expressions are crosstalk in the dispersive regime. We also use the transmon to realize similar to other proposal [31]. Our theory has been predict and simulate ISWAP on 2-qubit [13] and 8-qubit

II. FIXED-FREQUENCY

The circuit of a fixed-frequency Josephson junction shunt in Fig. 1 and is governed

$$\dot{H}_F = 4E_J \cos(\varphi_F)$$

FIG. 1. Circuit of a fixed-frequency transmon. The circuit consists of a Josephson junction with energy E_J , shunted by a capacitor C_{cr} . The phase of the loop is denoted by φ_F .

arXiv:1706.06562v1 [quant-ph] 20 Jun 2017

Parametrically-Activated Entangling Gates Using Transmon Qubits

S. Caldwell,* N. Didier,* C. A. Ryan,* E. A. Sete,* A. Hudson, P. Karalekas, M. P. da Silva, R. Sinclair, E. Acala, N. Alidoust, J. Angeles, A. Bestwick, M. Bli C. Bui, L. Capelutto, R. Chilcott, J. Cordova, G. Crossman, M. Curtis, S. Des D. Gershovich, S. Hong, K. Kuang, M. Lenihan, T. Manning, J. Marshall, Y. Moh J. Otterbach, A. Papageorge, J.-P. Paquette, M. Pelstring, A. Polloreno, G. Pra R. Renzas, N. Rubin, D. Russell, M. Rust, D. Scarabelli, M. Scheer, M. Selvanay M. Suska, N. Tezak, T.-W. To, M. Vahidpour, N. Vodrahalli, T. Whyland, K. Yad Rigetti Computing, 775 Heinz Avenue, Berkeley, CA 94710
(Dated: June 21, 2017)

We propose and implement a family of entangling qubit operations activated via flux pulses. By parametrically modulating the frequency of the tunable transmon selectively, we realize exchange of excitation with its statically coupled nearest-neighbor transmon. This direct exchange of excitation is utilized for mediator qubits or resonator modes and it allows for the full utilization of a scalable architecture. Moreover, we are able to activate three highly-selective two-qubit entangling gates that enable universal quantum computation: an iSWAP and a controlled-Z rotation. This selectivity is enabled by resonance between the drive frequencies and the amplitude and is helpful in isolating specific cross talk between qubits in a multi-qubit register. We report average process fidelities of $F = 0.95\%$ for a single iSWAP 175ms and 270ms controlled-Z operations.

One of the main challenges in building a scalable superconducting quantum processor architecture is the construction of a reliable two-qubit gate. There are two main approaches to achieving this goal using transmons [1]. The first approach utilizes fixed-frequency qubits with static coupling where the two-qubit operations are activated by applying transverse microwave drives [2–8]. While the fixed-frequency qubits may have long coherence times, this architecture is susceptible to crosstalk. Moreover, the activation of the two-qubit gate requires satisfying stringent constraints on qubit frequencies and anharmonicities [5, 6, 8]. Because of these issues, scaling to many qubits can be challenging. The second approach uses frequency-tunable transmons and two-qubit gates are activated by tuning qubits into and out of resonance with a particular transition [9–12]. However, scalability comes at the expense of additional decoupling channels, thus significantly limiting coherence times [13], mostly due to magnetic flux noise. Such gates are furthermore sensitive to frequency crowding—allowing unwanted crossings with neighboring two-qubit levels during gate operations limits the achievable connectivity of the architecture.

An alternative to both of the approaches above relies on parametrically modulating couplings or energy levels at a frequency corresponding to the detuning between particular energy levels of interest [13–22]. This enables an entangling gate between a qubit and a single resonator [17, 18], a qubit and many resonator modes [22], two transmon qubits coupled by a tunable mediating qubit [12, 21], or two tunable transmons coupled to a mediating resonator [19, 20].

Building on these earlier results, we implement two en-

abling gates, iSWAP and a controlled-Z operation, via frequency modulation of the same tunable transmon

arXiv:1706.06570v2 [quant-ph] 13 Jul 2017

Demonstration of Universal Parametric Entangling Gates on a Multi-Qubit Lattice

M. Reagor,* C. B. Osborn, N. Tezak, A. Staley, G. Prawiroatmodjo, M. Scheer, N. Alidoust, E. A. Sete, N. Didier, M. P. da Silva, E. Acala, J. Angeles, A. Bestwick, M. Block, B. Bloom, A. Bradley, C. Bui, S. Caldwell, L. Capelutto, R. Chilcott, J. Cordova, G. Crossman, M. Curtis, S. Deshpande, T. El Bouyadi, D. Gershovich, S. Hong, A. Hudson, P. Karalekas, K. Kuang, R. Manenti, T. Manning, J. Marshall, Y. Mohan, W. O’Brien, J. Otterbach, A. Papageorge, J.-P. Paquette, M. Pelstring, A. Polloreno, V. Rawat, C. A. Ryan, R. Renzas, N. Rubin, D. Russell, M. Rust, D. Scarabelli, M. Selvanayagan, R. Sinclair, R. Smith, M. Suska, T.-W. To, M. Vahidpour, N. Vodrahalli, T. Whyland, K. Yadav, W. Zeng, and C. Rigetti Rigetti Computing, 775 Heinz Avenue, Berkeley, CA 94710
(Dated: July 14, 2017)

We show that parametric coupling techniques can be used to generate selective entangling interactions for multi-qubit processors. By inducing coherent population exchange between adjacent qubits under frequency modulation, we implement a universal gateset for a linear array of four superconducting qubits. An average process fidelity of $F = 93\%$ is estimated for three two-qubit gates via quantum process tomography. We establish the suitability of these techniques for computation by preparing a four-qubit maximally entangled state and comparing the estimated process fidelity against the expected performance of the individual entangling gates. In addition, we prepare a eight-qubit register in all possible bitstring permutations and monitor the fidelity of a four-qubit gate across one pair of these qubits. Across all such permutations, an average fidelity of $F = 91.6 \pm 2.6\%$ is observed. These results thus offer a path to a scalable architecture with high selectivity and low cross talk.

All practical quantum computing architectures must address the challenges of gate implementation at scale. Superconducting quantum processors designed with static circuit parameters can achieve high coherence times [1, 2]. For these schemes, however, entangling gates have come at the expense of always-on qubit-qubit coupling [3] and frequency crowding [4]. Processors based on tunable Josephson qubits, meanwhile, can achieve minimal residual coupling and fast qubit-qubit operations [5–6]; yet, these systems are prone to come flux noise decoherence [7, 8] and computational basis leakage [9–12]. Moreover, the difficulty increases with both fixed-frequency and tunable qubit designs compounded as the system size grows. Parallel architectures [13, 14], however, promise to alleviate many of the fundamental challenges of scaling quantum computers. By using modular techniques akin to those used in logic quantum processors [15–17], these schemes allow for frequency-selective entangling gates between otherwise static, weakly-interacting qubits.

Several proposals for parametric logic gates have been experimentally verified in the last decade. Parametric entangling gates have been demonstrated between two flux qubits via frequency modulation of an ancillary qubit [13, 14]; between two transmon qubits via AC Stark modulation of the computational basis [17] and of the non-computational basis [18] with estimated gate fidelity of $F = 81\%$ [18]; between two fixed-frequency transmon qubits via frequency modulation of a tunable bus resonator with $F = 98\%$ [19]; between high quality factor resonators via frequency modulation of one tunable transmon [20–22] with $F = [60–80]\%$ [22]; and finally, between a fixed-frequency and tunable transmon

via frequency modulation of the same tunable transmon with $F = 95\%$ [23, 24]. Yet, despite these significant advances, there has yet to be an experimental assessment of the feasibility of parametric architectures with a multi-qubit system.

Here, we implement universal entangling gates via parametric control on a superconducting processor with eight qubits. We leverage the results of Refs. [23, 24] to show how the multiple degrees of freedom for parametric drives can be used to resolve on-chip, multi-qubit frequency-crowding issues. For a four-qubit subarray of the processor, we compare the action of parametric CZ gates to the ideal CZ gate using quantum process tomography (QPT) [25–27], estimating average gate fidelities [28, 29] of $F = 95\%$, 93%, and 91%. Next, the scalability of parametric entanglement is established by comparing the performance of individual gates to the observed fidelity of a four-qubit maximally entangled state. Further, we directly quantify the effect of the remaining six qubits of the processor on the operation of a single two-qubit CZ gate. To do so, we prepare the 64 classical states of the ancilla qubit register and, for each preparation, conduct two-qubit QPT. Tracing out the measurement outcomes of the ancilla results in an average estimated fidelity of $F = 91.6 \pm 2.6\%$ to the ideal process of CZ. Our error analysis suggests that scaling to larger processors through parametric modulation is readily achievable.

Figure 1a shows an optical image of the processor used in our experiment. The multi-qubit lattice consists of alternating tunable and fixed-frequency transmons, each capacitively coupled to its two nearest neighbors to form a ring topology. This processor is fabricated on a high re-



What is a quantum computer?

Machine that natively executes unitary operations on quantum systems

- Generalizes universal classical computer
- Benefits from inherent size of Hilbert spaces
- Better performance on notable hard problems

Classical state	Quantum state
011	$a_{000} 000\rangle +$
	$a_{001} 001\rangle +$
	$a_{010} 010\rangle +$
	$a_{011} 011\rangle +$
	$a_{100} 100\rangle +$
	$a_{101} 101\rangle +$
	$a_{110} 110\rangle +$
	$a_{111} 111\rangle$

$$|j_3 j_2 j_1\rangle \equiv |j_3\rangle \otimes |j_2\rangle \otimes |j_1\rangle$$



What is a quantum computer?

Machine that natively executes unitary operations on quantum systems

- Generalizes universal classical computer
- Benefits from inherent size of Hilbert spaces
- Better performance on notable hard problems

+1 qubit = 2x compute or memory

- Addressable problem size
- Energy efficiency

Classical state	Quantum state
011	$a_{000} 000\rangle +$
	$a_{001} 001\rangle +$
	$a_{010} 010\rangle +$
	$a_{011} 011\rangle +$
	$a_{100} 100\rangle +$
	$a_{101} 101\rangle +$
	$a_{110} 110\rangle +$
	$a_{111} 111\rangle$

$$|j_3 j_2 j_1\rangle \equiv |j_3\rangle \otimes |j_2\rangle \otimes |j_1\rangle$$



What is a quantum computer?

Machine that natively executes unitary operations on quantum systems

- Generalizes universal classical computer
- Benefits from inherent size of Hilbert spaces
- Better performance on notable hard problems

+1 qubit = 2x compute or memory

- Addressable problem size
- Energy efficiency

Interesting properties

- Fully reversible
- No copying an arbitrary state
- Non-deterministic state readout

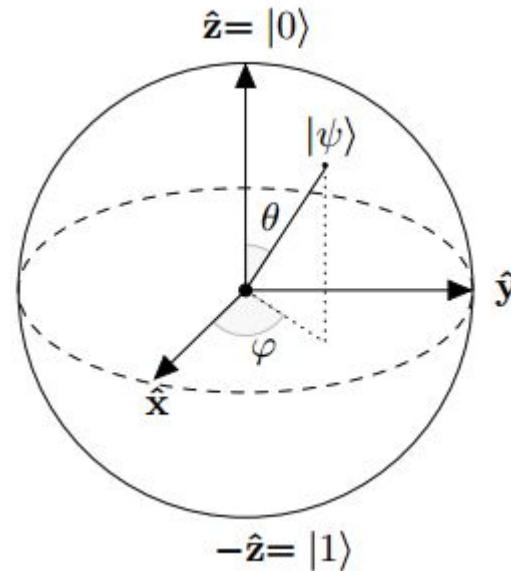
Classical state	Quantum state
011	$a_{000} 000\rangle +$
	$a_{001} 001\rangle +$
	$a_{010} 010\rangle +$
	$a_{011} 011\rangle +$
	$a_{100} 100\rangle +$
	$a_{101} 101\rangle +$
	$a_{110} 110\rangle +$
	$a_{111} 111\rangle$

$$|j_3 j_2 j_1\rangle \equiv |j_3\rangle \otimes |j_2\rangle \otimes |j_1\rangle$$



One-qubit quantum state

Lives on the surface of the Bloch sphere



$$|\psi\rangle = \cos(\theta/2)|0\rangle + e^{i\varphi}\sin(\theta/2)|1\rangle$$

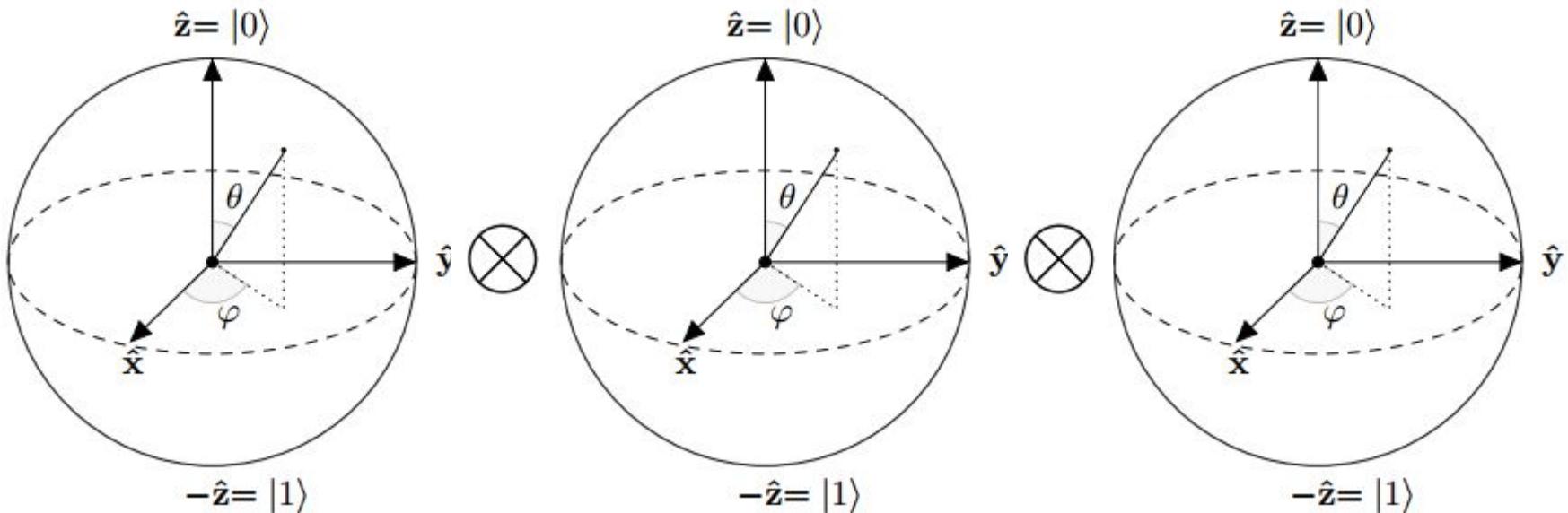


Multi-qubit state

Arbitrary state $|\psi\rangle$

$|\psi\rangle =$

$$a_{000}|000\rangle + \\ a_{001}|001\rangle + \\ a_{010}|010\rangle + \\ a_{011}|011\rangle + \\ a_{100}|100\rangle + \\ a_{101}|101\rangle + \\ a_{110}|110\rangle + \\ a_{111}|111\rangle$$



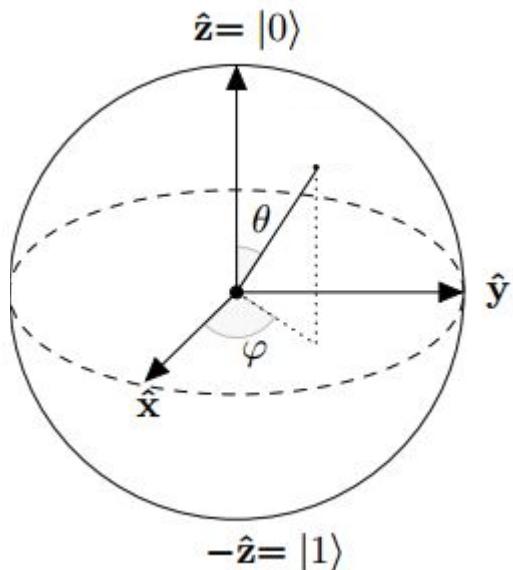
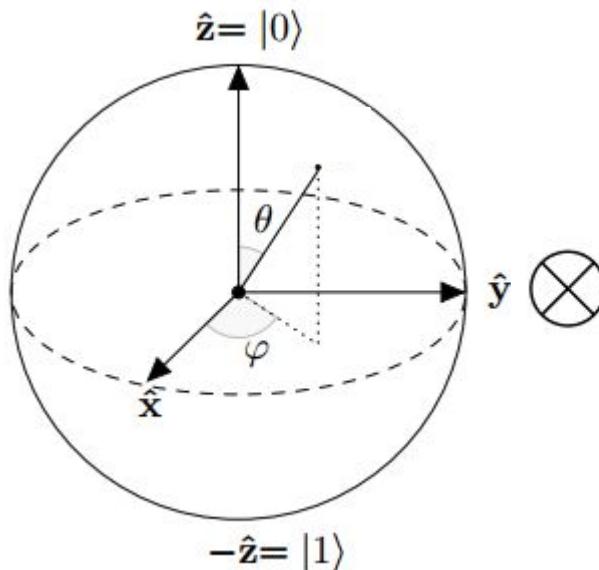
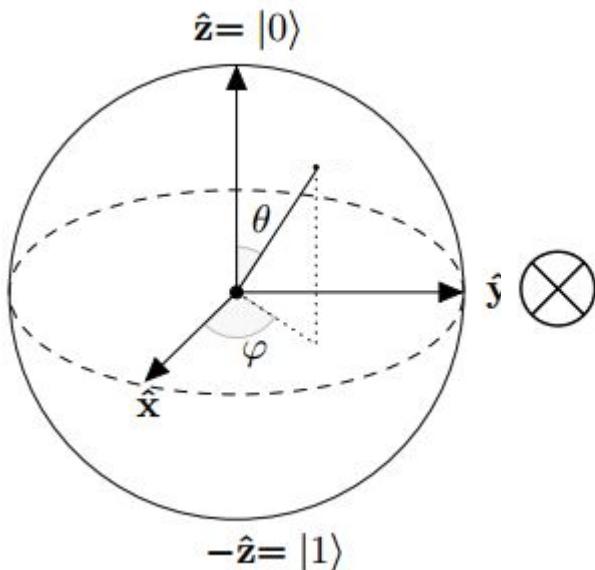
Larger Hilbert space is tensor product of smaller ones



Multi-qubit state

Arbitrary state $|\psi\rangle$

$$|\psi\rangle = a_{000}|000\rangle + a_{001}|001\rangle + a_{010}|010\rangle + a_{011}|011\rangle + a_{100}|100\rangle + a_{101}|101\rangle + a_{110}|110\rangle + a_{111}|111\rangle$$



Basis states:

$$|j_3 j_2 j_1\rangle \equiv |j_3\rangle \otimes |j_2\rangle \otimes |j_1\rangle$$

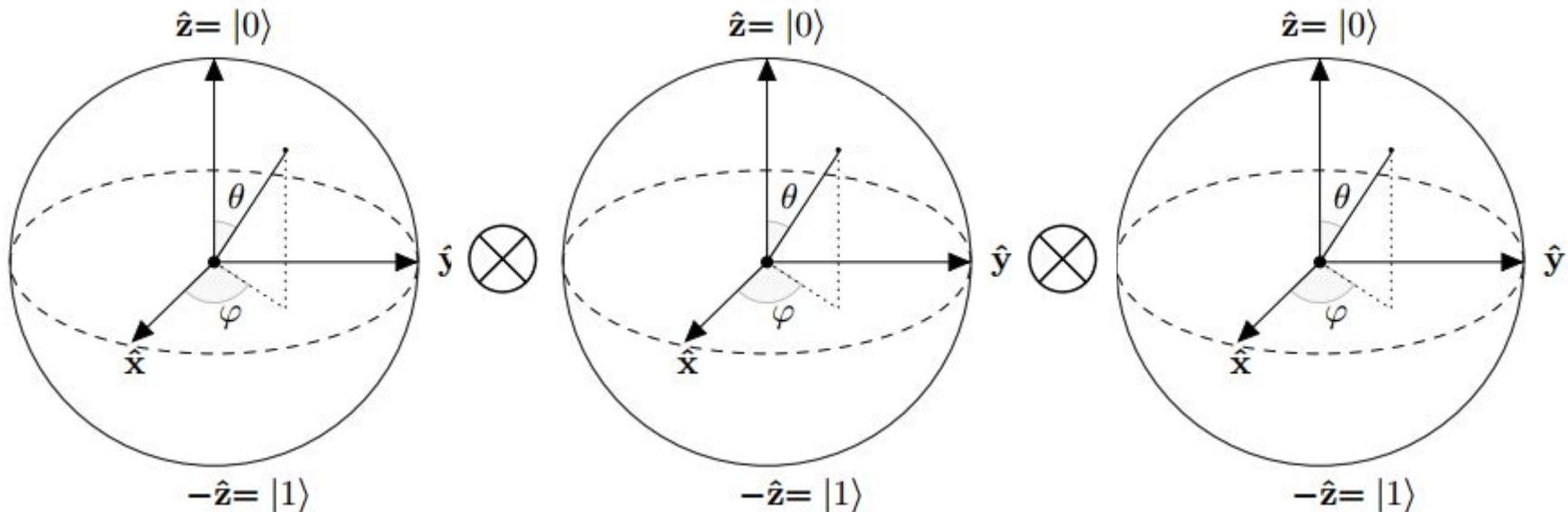


Multi-qubit state

Arbitrary state $|\psi\rangle$

$|\psi\rangle =$

$$a_{000}|000\rangle + \\ a_{001}|001\rangle + \\ a_{010}|010\rangle + \\ a_{011}|011\rangle + \\ a_{100}|100\rangle + \\ a_{101}|101\rangle + \\ a_{110}|110\rangle + \\ a_{111}|111\rangle$$



Basis states:

$$|j_3 j_2 j_1\rangle \equiv |j_3\rangle \otimes |j_2\rangle \otimes |j_1\rangle$$

General states:

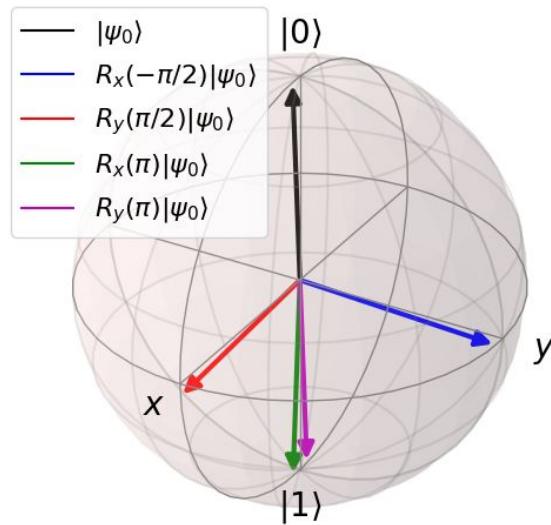
$$|\psi\rangle \not\equiv |\psi_3\rangle \otimes |\psi_2\rangle \otimes |\psi_1\rangle \quad (\text{entanglement})$$



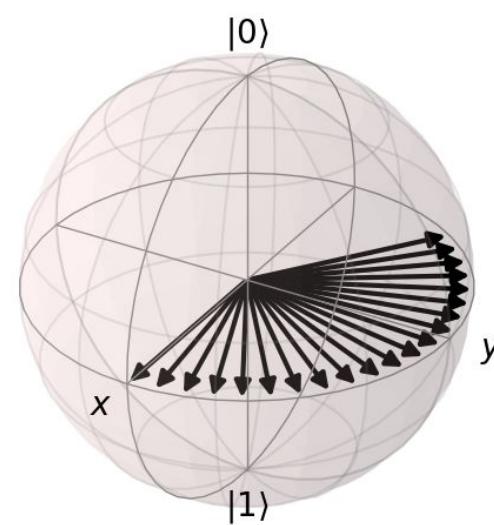
Controlling a quantum state

"Machine that natively executes unitary operations on quantum systems"

Unitaries are rotations



X and Y rotations by driving



Z rotations by waiting

- Qubit frequency f_q
- Qubit precesses in xy (complex) plane at f_q
- X and Y rotations driven with external fields

