rigetti
quantum computing

Quantum Programming on near-term devices

MURI Review Invited Talk 11.29.2016

Will Zeng

Here's where we are

1994

(1) Invention of Quantum Algorithms

1992-4
First Quantum Algorithms w/ Exponential Speedup (Deutsch-Jozsa, Shor's Factoring, Discrete Log, ...)

1996
First Quantum Database Search Algorithm (Grover's)

2007
Quantum Linear Equation Solving (Harrow, Hassidim, Lloyd)

2008
High Threshold Error Correcting Codes Emerge
Quantum Algorithms for SVM's & Principal Component Analysis

2013

(3) Invention of Quantum/Classical Hybrid Algorithms

Low Overhead Error Correcting Codes
Practical Quantum Chemistry Algorithms (QVE)

2016
Low Overhead Error Correcting Codes
Practical Quantum Optimization Algorithms (QAOA)

1996
First Quantum Error Correction Codes

2001
First Superconducting Qubits

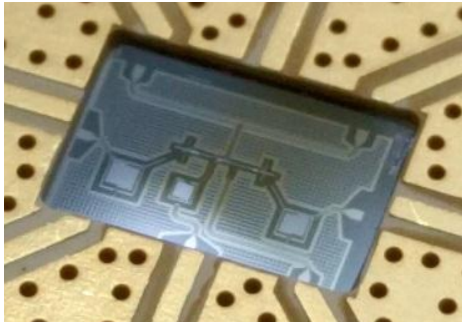
2007
Transmon Invented

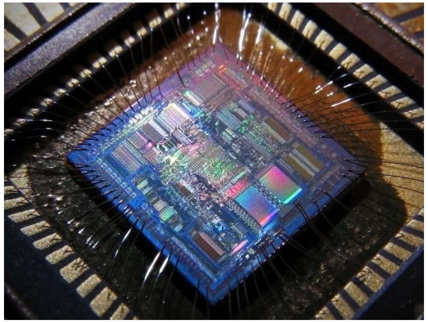
2012
Qubit coherence passes error-correcting threshold (Yale & IBM (CTR))
Nobel prize for quantum control of atoms & ions

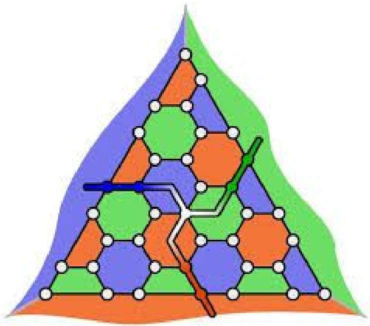
(2) Performant superconducting circuits: 1-9 qubits

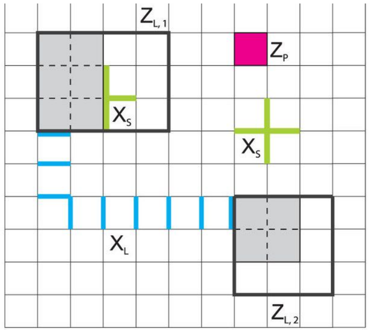
2015
First QEC Experiments on multi-qubit chips
Scalable quantum chemistry simulations on multi-qubit chips

TODAY









> Scalable chip-based quantum processors

Superconducting Microwave Circuits

> Build towards fault-tolerance

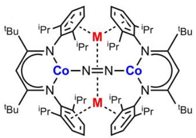
Quantum Simulation

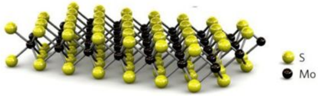
Variational Quantum Eigensolvers

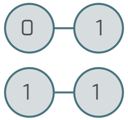
Quantum Optimization

QAOA

Catalysts	Complex Materials	MAX-CUT







Score +1

Score 0

N_2-activation, H_2O-O_2, etc

High-T_c, dichalcogenides, etc

Max-Cut cost operator

$$\hat{C}_{ij} = \frac{1}{2}\left(\mathbf{I} - \sigma_i^Z \sigma_j^Z\right)$$

> Focus on near-term applications

Quantum/Classical Hybrid Algorithms

> Access over the cloud

Quantum Computers as co-Processors

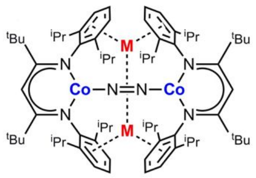
Outline

- What are near-term applications? Quantum/classical hybrid algorithms
- How to program hybrid algorithms? The Quil (Quantum Instruction Language) Stack
  - > Platform Overview
  - > Instruction Set & Programming Model
  - > Software Examples
- What is a near-term device?
  - > Superconducting circuits
  - > Noise in near-term devices
- Open Questions

Applications (*in silico* chemistry)

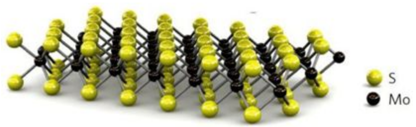
Nature's exploitation of quantum mechanics

Catalysts



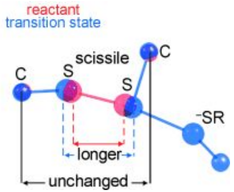
N_2-activation, H_2O-O_2, etc

Complex Materials



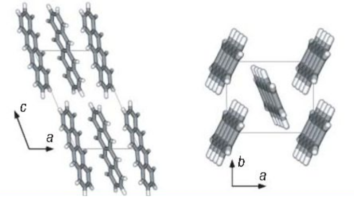
High-T_C, dichalcogenides, etc

Bond form/break



S_2-breaking, bioactivity, atmospheric, etc

Dynamics



Solar cell efficiency, molecular conductors

Model Complexity vs Computational Cost

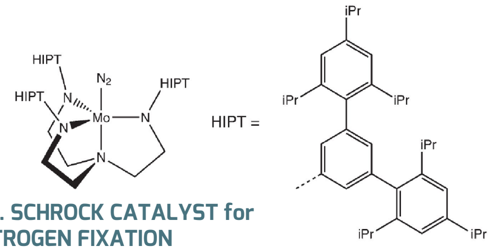
mean-field	DFT	Perturbation Theory	Many-Body (coupled-cluster)	RDM	Exact Diagonalization
$O(N^3)$	$O(N^3)$	$O(N^5)$	$O(N^7)$	$O(N^9)$	$O(2^N)$

First Applications: Quantum Variational Eigensolvers

1. MOLECULAR DESCRIPTION

e.g. Electronic Structure Hamiltonian

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



e.g. SCHROCK CATALYST for NITROGEN FIXATION

2. MAP TO QUBIT REPRESENTATION

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

e.g. HYDROGEN

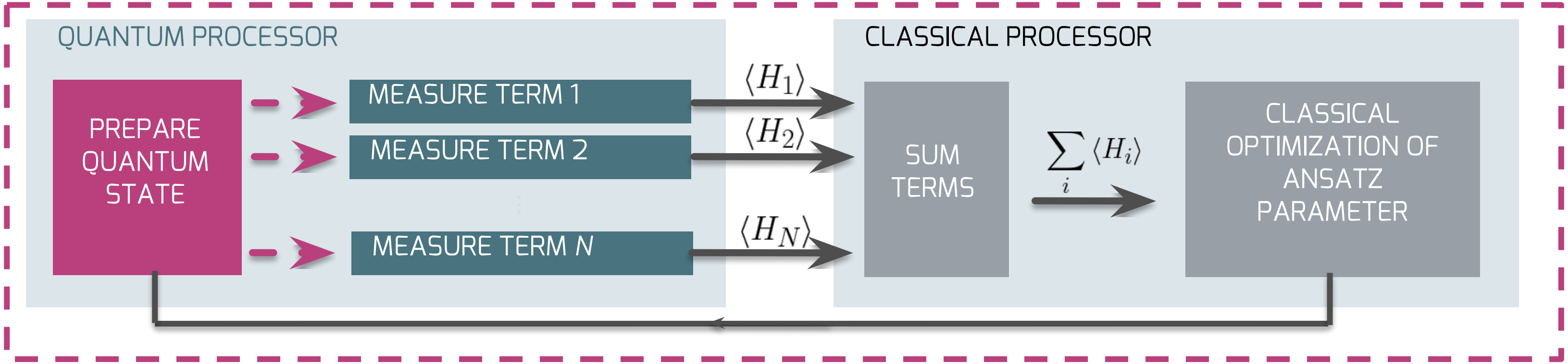
$$\begin{aligned} H = & f_0 \mathbb{1} + f_1 Z_0 + f_2 Z_1 + f_3 Z_2 + f_1 Z_0 Z_1 \\ & + f_4 Z_0 Z_2 + f_5 Z_1 Z_3 + f_6 X_0 Z_1 X_2 + f_6 Y_0 Z_1 Y_2 \\ & + f_7 Z_0 Z_1 Z_2 + f_4 Z_0 Z_2 Z_3 + f_3 Z_1 Z_2 Z_3 \\ & + f_6 X_0 Z_1 X_2 Z_3 + f_6 Y_0 Z_1 Y_2 Z_3 + f_7 Z_0 Z_1 Z_2 Z_3 \end{aligned}$$

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



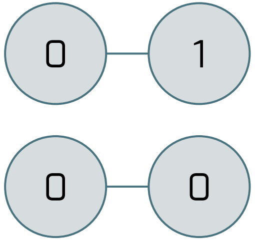
O'Malley, P. J. J., et al. (2015). Scalable Quantum Simulation of Molecular Energies. *arXiv:1512.06860*.
Wecker, D., et al. (2015). Progress towards practical quantum variational algorithms. *Physical Review A*, *92*(4), 042303.
McClean, J. R. et al. (2015). The theory of variational hybrid quantum-classical algorithms. *arXiv:1509.04279*.
Peruzzo, A., et al. (2014). A variational eigenvalue solver on a photonic quantum processor. *Nature communications*, *5*.

NP-Hard Optimization on Quantum Computers

First results on polynomial time approximate algorithms with quantum resources

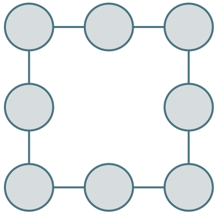
Exact Algorithms → Polynomial Approximation

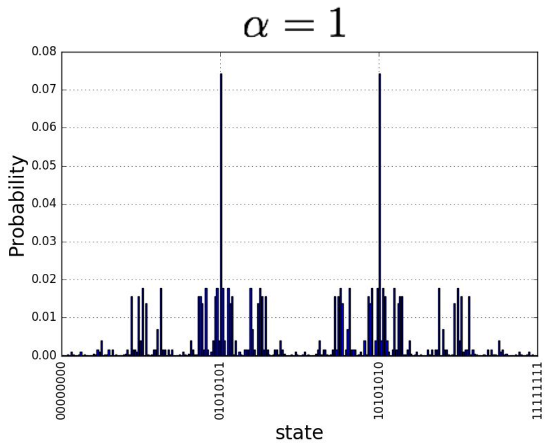
Classical Maximum Cut Problem

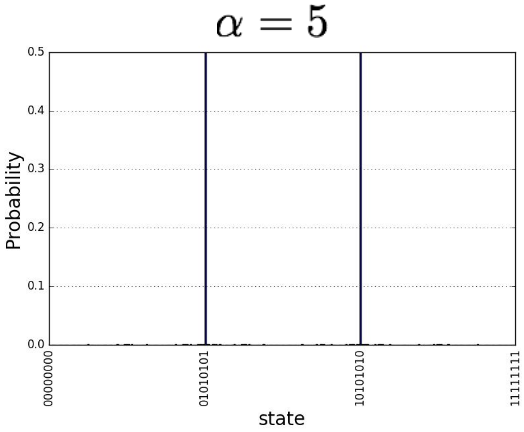


Score +1

Score 0







Classical Maximum Cut cost operator

$$C_{ij} = \frac{1}{2}(1 - Z_i Z_j)$$

$$Z_i \in \{+1, -1\}$$

Quantum Maximum Cut cost operator

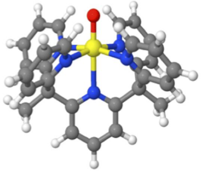
$$\hat{C}_{ij} = \frac{1}{2}\left(\mathbf{I} - \sigma_i^Z \sigma_j^Z\right)$$

Farhi et al. (2014). A Quantum Approximate Optimization Algorithm. *arXiv:/1411.4028.*
Farhi & Harrow (2015). Quantum Supremacy through the Quantum Approximate Optimization Algorithm. *arXiv: 1602.07674*

Optimizing Quantum Programs for real HW

Building more efficient encodings of second quantized operators

Define Molecular Structure



Electronic Structure Hamiltonian

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

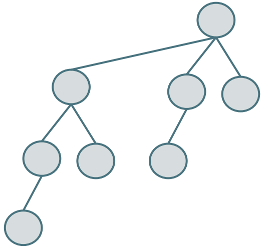
Second Quantized Hamiltonian

$$\hat{H} = \sum_i {}^1h_i^j \hat{a}_j^\dagger \hat{a}_i + \frac{1}{2}\sum_{ijkl} {}^2V_{kl}^{ij} \hat{a}_i^\dagger \hat{a}_j^\dagger \hat{a}_l \hat{a}_k$$

Bravyi-Kitaev (log(N)-mapping)

$$\hat{a}_j^\dagger = \frac{1}{2}\left(X_{U(j)} \otimes X_j \otimes Z_{P(j)} - iX_{U(j)} \otimes Y_j \otimes Z_{\rho(j)}\right)$$

BK parity heap

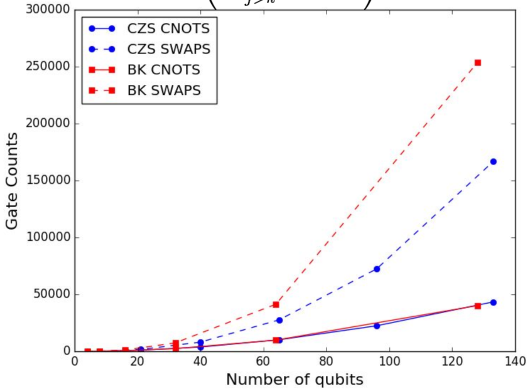


≠

Qubit connectivity graph

Hardware Optimal Mapping: Coulomb term scaling

$$\exp\left(-i\sum_{j>k} \hat{a}_j^\dagger \hat{a}_j \hat{a}_k^\dagger \hat{a}_k\right)$$



Outline

- What are near-term applications? Quantum/classical hybrid algorithms

- How to program hybrid algorithms? The Quil (Quantum Instruction Language) Stack
  > Platform Overview
  > Instruction Set & Programming Model
  > Software Examples

- What is a near-term device?
  > Superconducting circuits
  > Noise in near-term devices

- Open Questions

The Quantum Computing Stack

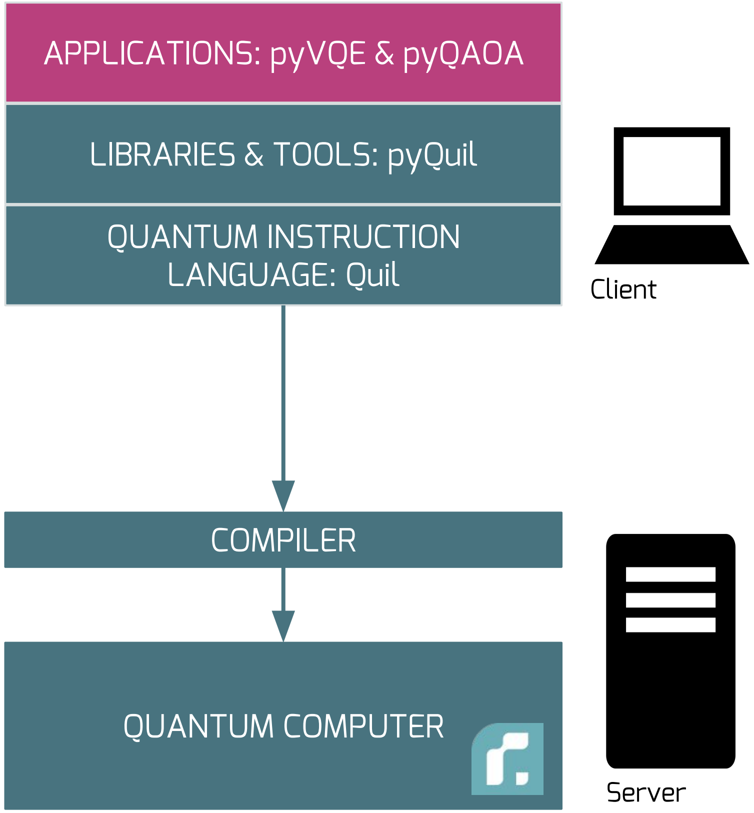
> Write applications...

> using tools...

> that build quantum programs...

> that compile onto quantum hardware...

> that execute on a quantum processor.



```
# Bell-state program
from pyquil.quil import Program
import pyquil.qvm as qvm

p = Program(H(0), CNOT(0,1))
qvm.run(p)
```

APPLICATIONS: pyQVE & pyQAOA

LIBRARIES & TOOLS: pyQuil

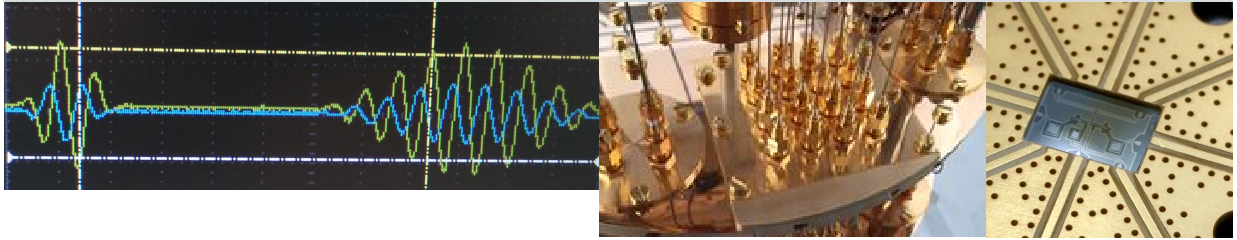
QUANTUM INSTRUCTION LANGUAGE: Quil

Client

```
HADAMARD 0
CNOT 0 1
MEASURE 0 [0]
```

COMPILER

QPU

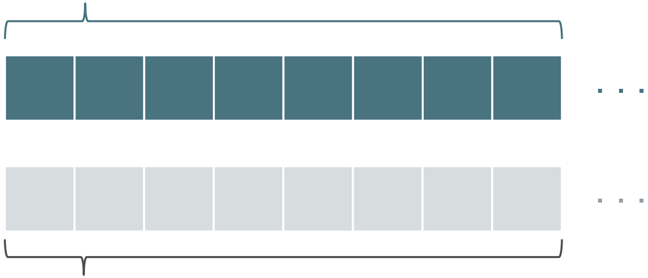


Server

The Programming Model

A Quantum-Classical Hybrid Model

1. *N* qubits



2. *M* classical octets (8-bits)
   = 8*M* total bits

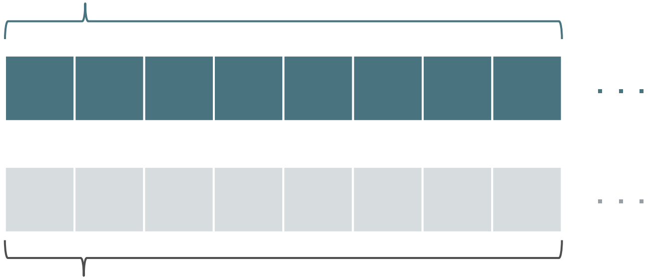
3. A fixed gate set, e.g.

{H(0), CNOT(0,1)...}

The Programming Model

A Quantum-Classical Hybrid Model

1. *N* qubits



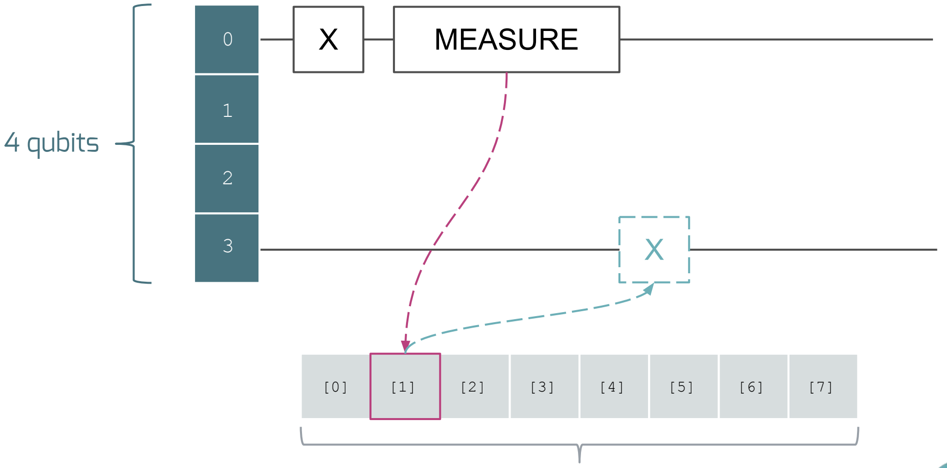
2. *M* classical octets (8-bits)
   = 8*M* total bits

3. A fixed gate set, e.g.

{H(0), CNOT(0,1)...}

Controlled by a Quantum Instruction Language (Quil)

```
X 0
MEASURE 0 [1]
JUMP-WHEN @THEN1 [1]
JUMP @END2
LABEL @THEN1
X 7
LABEL @END2
```



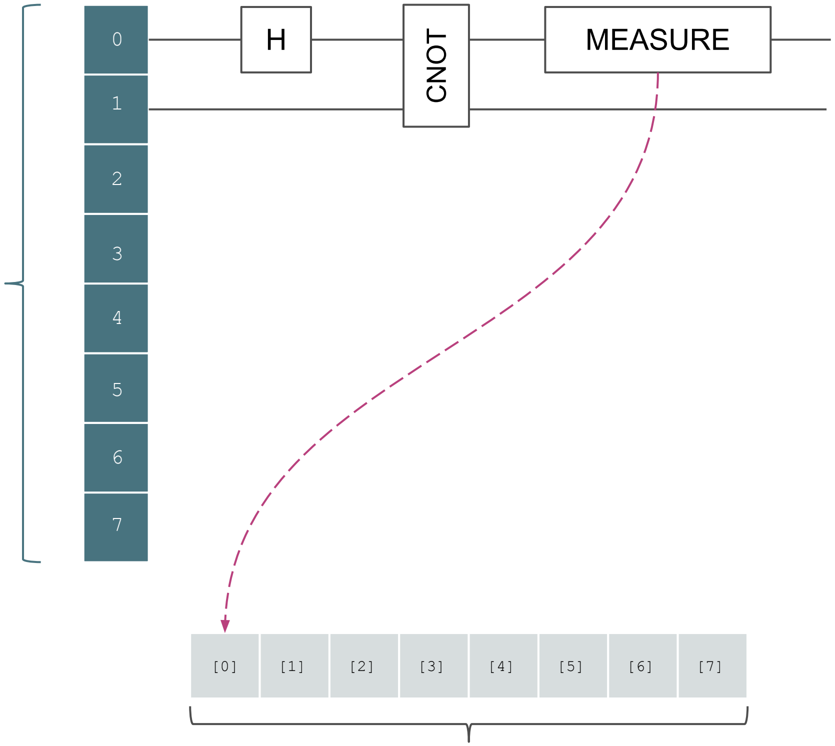
A classical octet (8-bits)

Quil: A Practical Quantum Instruction Language

Programming a Quantum Abstract Machine

```
# Bell-state program
HADAMARD 0
CNOT 0 1
MEASURE 0 [0]
```

1. 8 qubits



2. A fixed gate set, e.g. {H(0), CNOT(0,1)...}

3. 1 classical octet (8-bits)

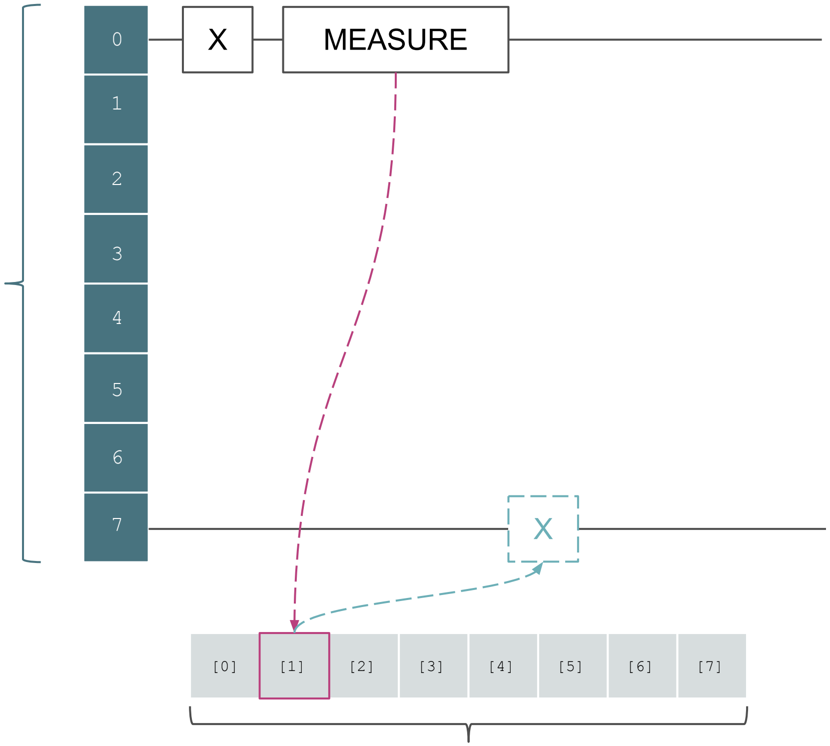
Quil: A Practical Quantum Instruction Language

Programming a Quantum Abstract Machine

```
#classical control

X 0
MEASURE 0 [1]
JUMP-WHEN @THEN1 [1]
JUMP @END2
LABEL @THEN1
X 7
LABEL @END2
```

1. 8 qubits



2. A fixed gate set, e.g. {H(0), CNOT(0,1)...}

3. 8 classical bits

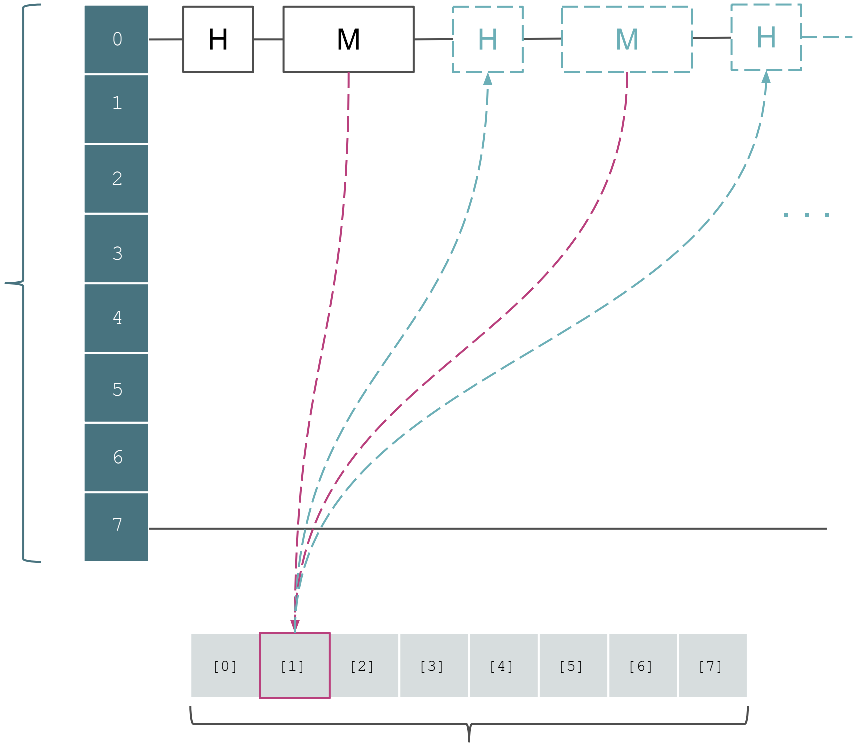
Quil: A Practical Quantum Instruction Language

Programming a Quantum Abstract Machine

```
#classical loops

LABEL @START3
H 0
MEASURE 0 [1]
JUMP-WHEN @END4 [1]
JUMP @START3
LABEL @END4
```

1. 8 qubits



2. A fixed gate set, e.g. {H(0), CNOT(0,1)...}

3. 8 classical bits

Quil: A Practical Quantum Instruction Language

Programming a Quantum Abstract Machine

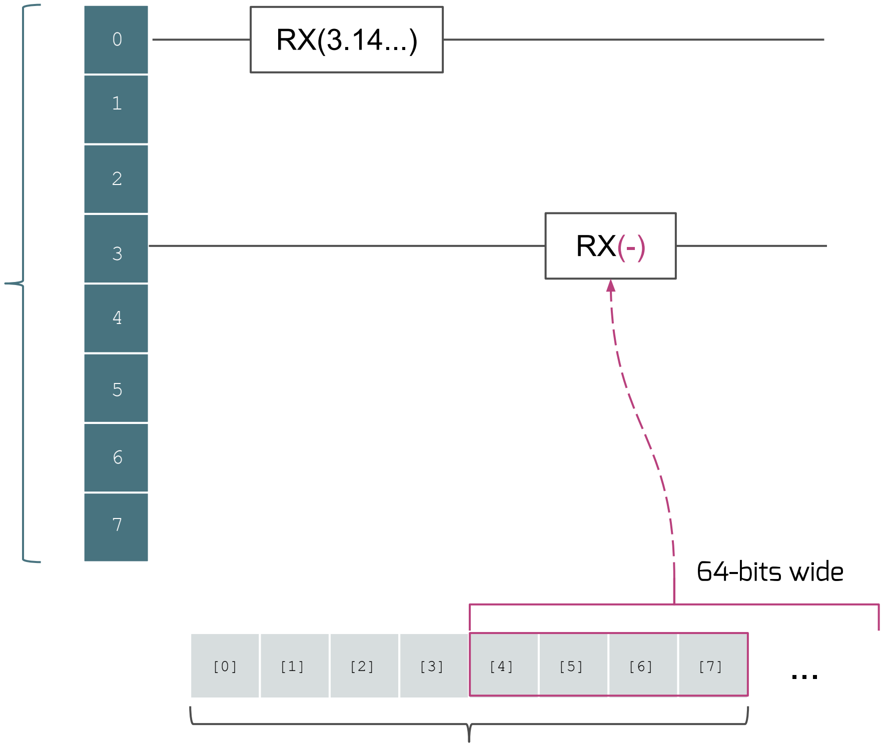
```
 # defining gates
DEFGATE HADAMARD:
  1/sqrt(2), 1/sqrt(2)
  1/sqrt(2),-1/sqrt(2)

DEFGATE RX(%theta):
  cos(%theta/2),-i*sin(%theta/2)
 -i*sin(%theta/2),cos(%theta/2)

# parameterized gates

RX(3.141592653589793) 0
RX([4-67]) 3
```

1. 8 qubits



2. A fixed gate set, e.g. {H(0), CNOT(0,1)...}

3. 1 classical octet (8-bits)

pyQuil: Python library for writing Quil

```
from pyquil.quil import Program
import pyquil.qvm as qvm_module
from pyquil.gates import *
qvm = qvm_module.Connection()

p = Program(H(0), CNOT(0, 1)) # create an entangled state
    <pyquil.pyquil.Program object at 0x101ebfb50>
qvm.wavefunction(p)
    [(0.7071067811865475+0j), 0j, 0j, (0.7071067811865475+0j)]

p = Program(H(0)) # create a superposition state
p.measure(0, 0)   # measure the qubit into [0]

# create loop
prob_inf_loop = Program().quil_while(0, p)
qvm.run(p, [0]) # ask for the result of register [0] back
    [[0]]
```

Quil: A Practical Quantum Instruction Language

For more info and Quil information see our arXiv white paper

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***

Outline

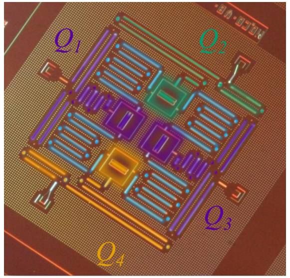
- What are near-term applications? Quantum/classical hybrid algorithms

- How to program hybrid algorithms? The Quil (Quantum Instruction Language) Stack
  > Platform Overview
  > Instruction Set & Programming Model
  > Software Examples

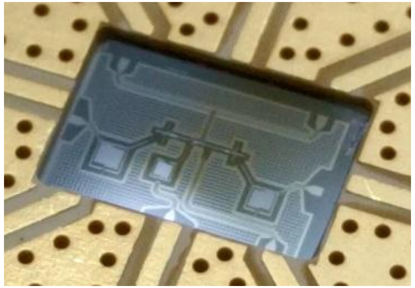
- What is a near-term device?
  > Superconducting circuits
  > Noise in near-term devices

- Open Questions

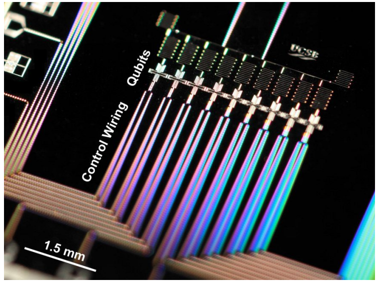
Microwave Quantum Integrated Circuits



IBM



Rigetti



Google

SUPERCONDUCTING QUBITS

> Long-lived quantum coherence
> Engineered circuit Hamiltonians
> Controlled with digital & RF electronics

+

INTEGRATED CIRCUITS

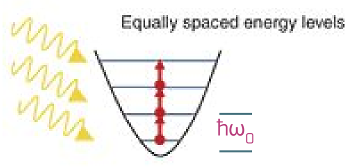
> Leverage existing semi fab & packaging
> Advanced tools for design & validation
> Promise of low cost and high reliability

Key Challenges for Scalability

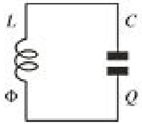
ELECTROMAGNETIC ISOLATION | RF & DC WIRING | FUNCTIONAL PROCESSOR DESIGN | RELIABLE CIRCUIT PARAMETERS | CONTROL COMPONENTS

Superconducting Qubits

Electrical circuits as a two-level quantum (an)-harmonic oscillator



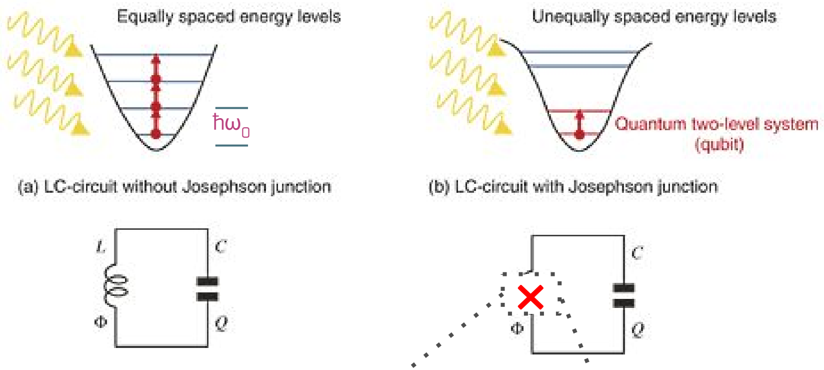
(a) LC-circuit without Josephson junction



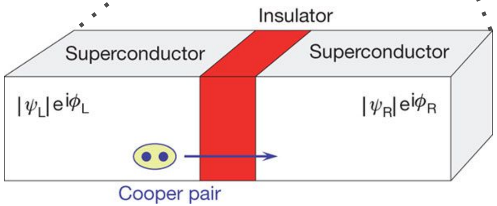
$$H = Q^2/2C + \Phi^2/2L = \hbar\omega_0(\underline{a}^\dagger \underline{a} + 1/2)$$

Superconducting Qubits

Electrical circuits as a two-level quantum (an)-harmonic oscillator



$H = Q^2/2C + \Phi^2/2L = \hbar\omega_0(a^\dagger a + 1/2)$



$\varphi = \varphi_R - \varphi_L$

$I = I_c \sin(\varphi) \rightarrow \Phi = L_J I$ with $\mathbf{L_J = \Phi_0/2\pi I_c \cos(\varphi)}$

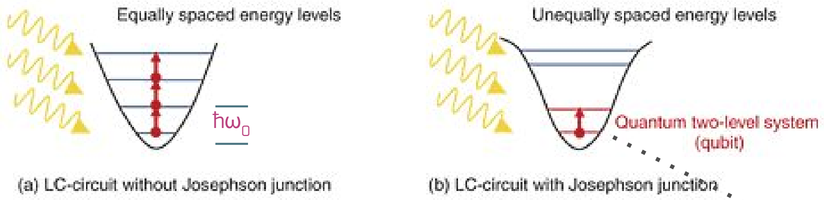
→ Nonlinear dissipationless inductor

→ Anharmonic energy spectrum

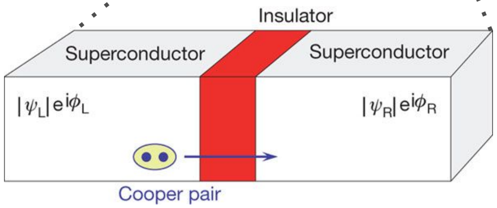
→ Possibility to engineer a two-level system

Superconducting Qubits

Electrical circuits as a two-level quantum (an)-harmonic oscillator

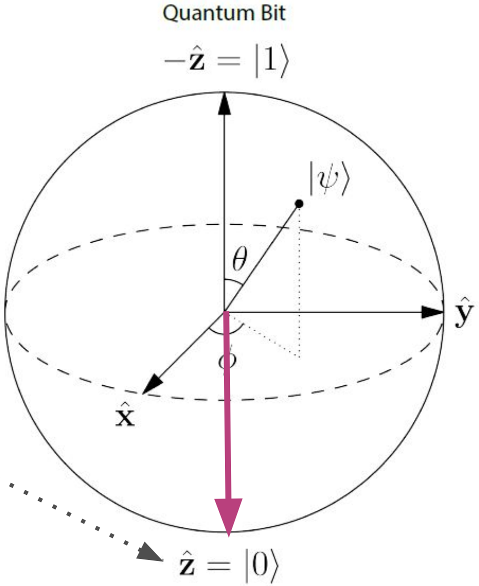


$H = Q^2/2C + \Phi^2/2L = \hbar\omega_0(a^\dagger a + 1/2)$



$\varphi = \varphi_R - \varphi_L$

$I = I_C \sin(\varphi) \rightarrow \Phi = L_J I$ with $L_J = \Phi_0 / 2\pi I_C \cos(\varphi)$



System: Two-dimensional complex vector-space

State: $|x\rangle = \alpha|0\rangle + \beta|1\rangle$

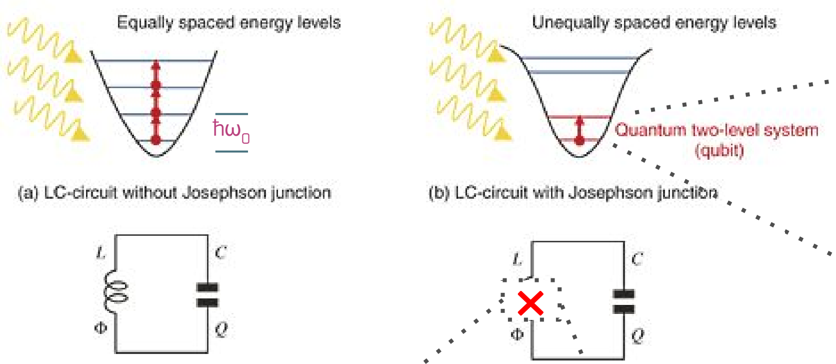
where α, β are complex numbers s.t. $|\alpha|$^2+$|\beta|$^2 = 1

Measurement: $\mathrm{Prob}(0) = \langle 0|x|0\rangle = |\alpha|^2$

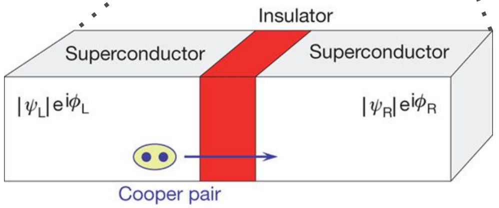
$\mathrm{Prob}(1) = \langle 1|x|1\rangle = |\beta|^2$

Superconducting Qubits

Electrical circuits as a two-level quantum (an)-harmonic oscillator

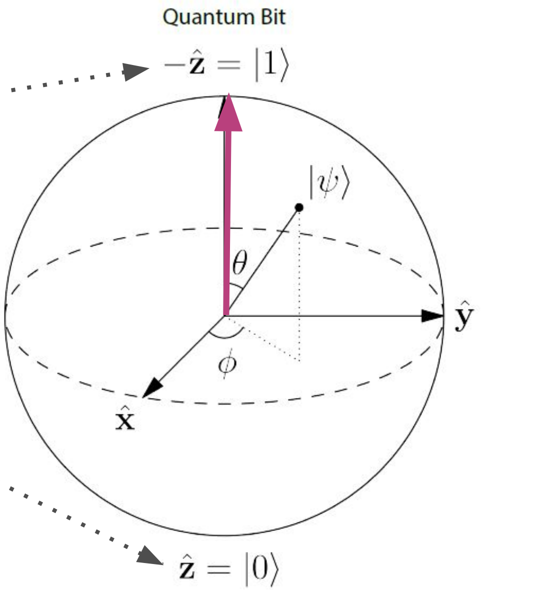


$H = Q^2/2C + \Phi^2/2L = \hbar\omega_0(a^\dagger a + 1/2)$



$\varphi = \varphi_R - \varphi_L$

$I = I_C\sin(\varphi) \rightarrow \Phi = L_J I$ with $L_J = \Phi_0/2\pi I_C\cos(\varphi)$



System: Two-dimensional complex vector-space

State: $|x\rangle = \alpha|0\rangle + \beta|1\rangle$

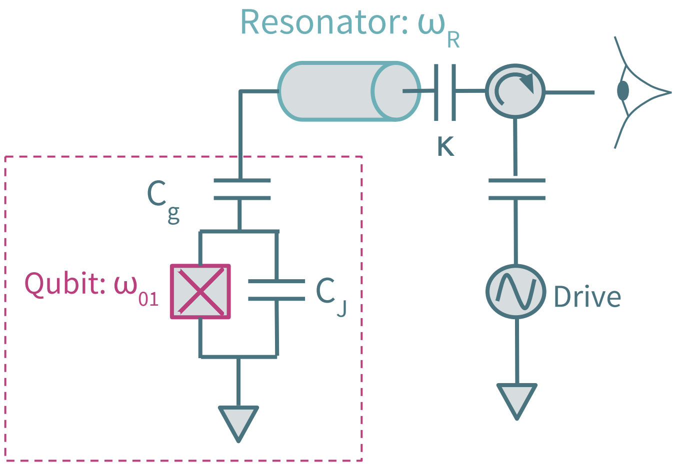
where α, β are complex numbers s.t. $|\alpha|$^2+$|\beta|$^2 = 1

Measurement: $\text{Prob}(0) = \langle 0|x|0\rangle = |\alpha|^2$

$\text{Prob}(1) = \langle 1|x|1\rangle = |\beta|^2$

Qubit Measurement

→ Readout by probing a linear resonator (ω_r) coupled to the qubit

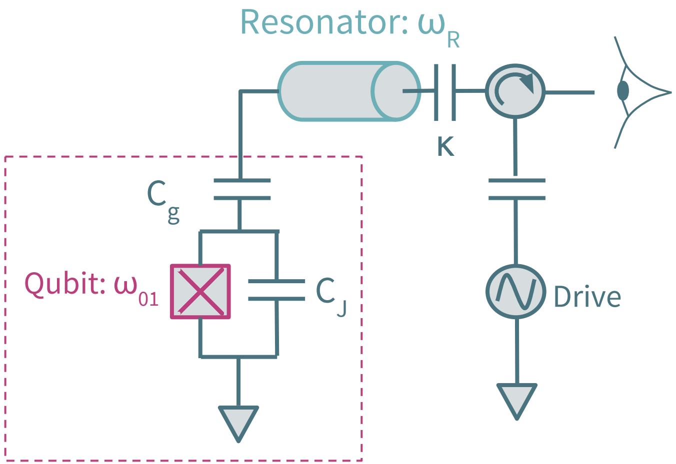


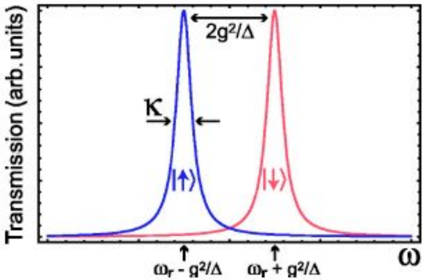
Dispersive Jaynes-Cummings Hamiltonian:

$$\hat{H} = \underbrace{\hbar\omega_r' \hat{a}_r{}^\dagger \hat{a}_r}_{\text{Resonator}} + \underbrace{\frac{\hbar\omega_{01}'}{2}\hat{\sigma}_z}_{\text{Qubit}} + \underbrace{\frac{\chi}{2}\hat{\sigma}_z \hat{a}_r{}^\dagger \hat{a}_r}_{\text{Coupling}}$$

Qubit Measurement

→ Readout by probing a linear resonator (ω_r) coupled to the qubit





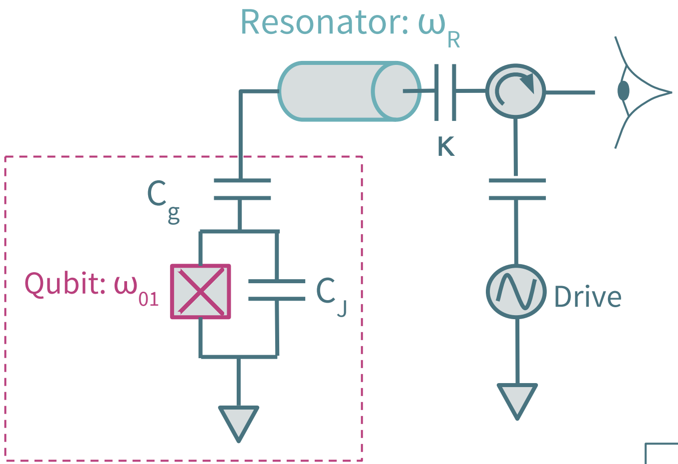
$$\hat{H}_r = \hbar(\omega_r' + \frac{\chi}{2}\hat{\sigma}_z)\hat{a}_r{}^\dagger \hat{a}_r$$

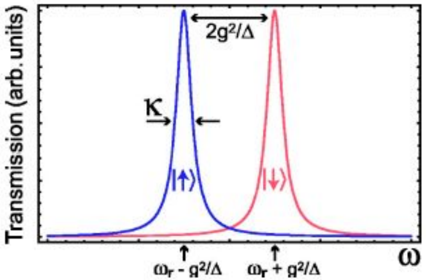
Dispersive Jaynes-Cummings Hamiltonian:

$$\hat{H} = \underbrace{\hbar\omega_r' \hat{a}_r{}^\dagger \hat{a}_r}_{\text{Resonator}} + \underbrace{\frac{\hbar\omega_{01}'}{2}\hat{\sigma}_z}_{\text{Qubit}} + \underbrace{\frac{\chi}{2}\hat{\sigma}_z \hat{a}_r{}^\dagger \hat{a}_r}_{\text{Coupling}}$$

Qubit Measurement

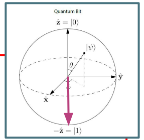
→ Readout by probing a linear resonator (ω_r) coupled to the qubit

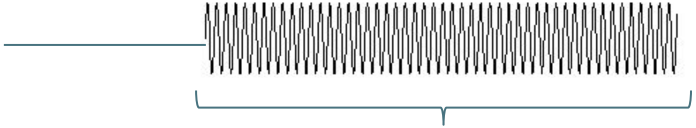




$$\hat{H}_r = \hbar(\omega_r' + \frac{\chi}{2}\hat{\sigma}_z)\hat{a}_r{}^\dagger\hat{a}_r$$

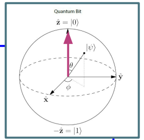
```
MEASURE 0 [0]
```

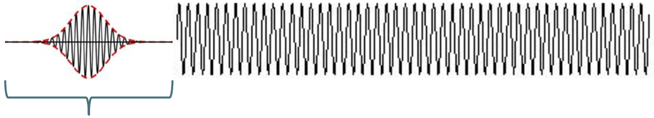




Measurement Pulse ~ 1us

```
X 0
MEASURE 0 [0]
```



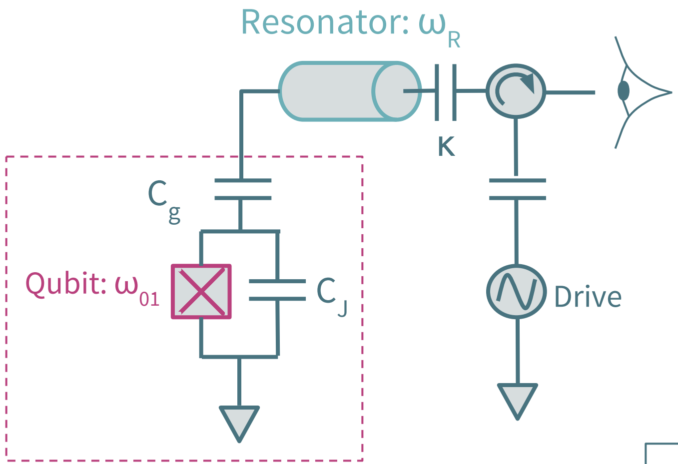


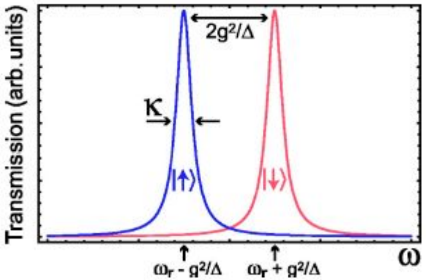
Qubit Pulse 10-100's ns

Qubit Measurement

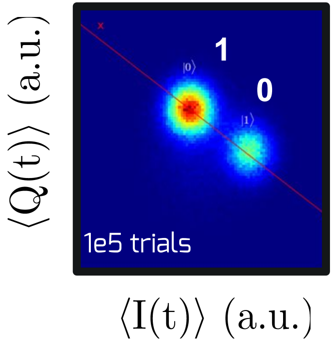
See [JPA TTS talk]

→ Readout by probing a linear resonator (ω_r) coupled to the qubit





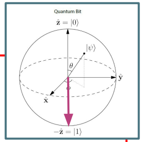
Measurement histograms

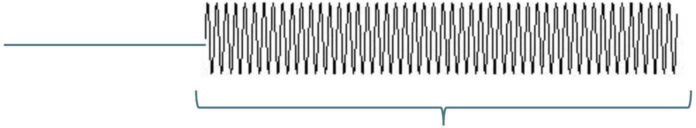


Fidelity = 99% with single shot

$$\hat{H}_r = \hbar(\omega_r' + \frac{\chi}{2}\hat{\sigma_z})\hat{a_r}^\dagger\hat{a_r}$$

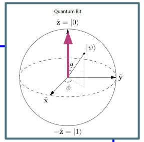
```
MEASURE 0 [0]
```

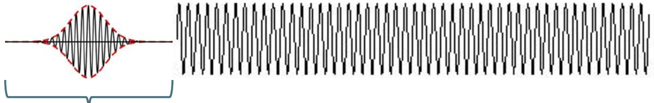




Measurement Pulse ~ 1us

```
X 0
MEASURE 0 [0]
```

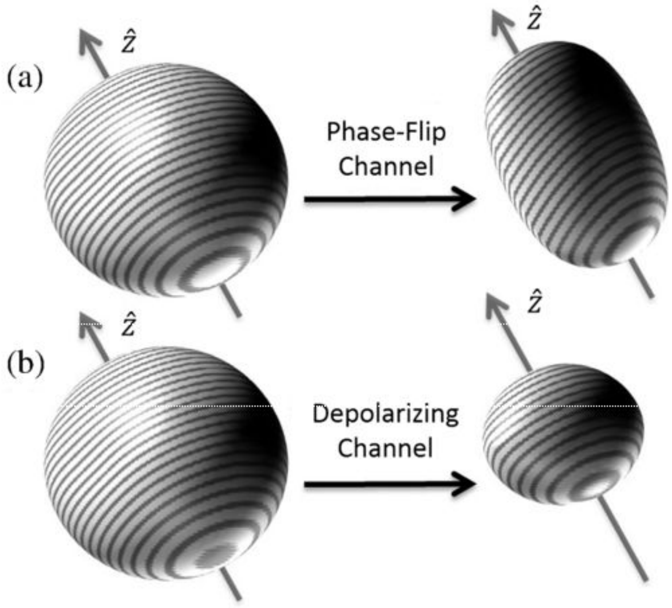




Qubit Pulse 10-100’s ns

Noise in Near Term Devices

- Coherent vs. Incoherent Noise
- Markovian Noise
  > Preparation errors
  > Measurement errors
  > Gate errors (the Pauli Channel)
  > Relaxation (T1) and Dephasing (T2)
- Non-Markovian Noise aka everything else
- Metrics (fidelity, diamond-norm)
- Procedures: Randomized Benchmarking, Gate Set Tomography



Important Open Questions

> Near-term Benchmarks & Applications
>> Quantum Supremacy[1] at
>> TQF ≅ 100 x 50 = 5k
>> Quantum Simulation
>> Quantum Optimization

> How to program for sampling supremacy?

> How to program under noise?

> How do we debug quantum-classical hybrids?

TOTAL QUANTUM FACTOR

$$\mathrm{TQF} = t_c \, / \, t_g \;\; \times \#Q$$

t_c = coherence time
t_g = gate time

#Q = number of qubits

[1] Boxio et al. Characterizing Quantum Supremacy in Near-Term Devices arXiv: 1608.00263

[2] Bremner et al. Achieving quantum supremacy with sparse and noisy commuting quantum computations arXiv: 1610.01808