Oak Ridge National Laboratory

Experimentally Characterizing Quantum Processors Using Modeling and Simulation

Megan Lilly, Travis Humble

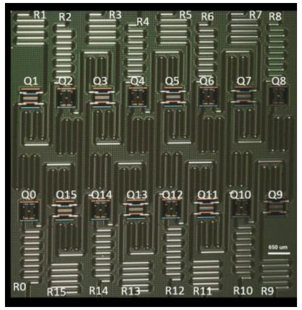
Oak Ridge National Laboratory and the Bredesen Center at the University of Tennessee, Knoxville

Fitting coarse-grained and application-focused noise models for NISQ devices

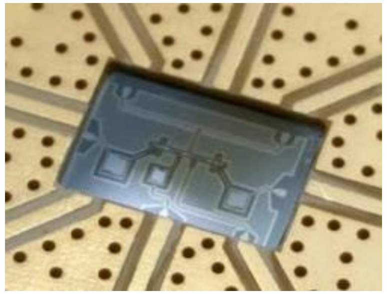
This work was supported by the Department of Energy Office of Science Early Career Research Program.

Current Quantum Processing Units

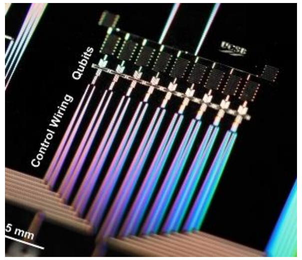
- QPU's are devices that implement the principles of digital quantum computing
  - Several different maturing technologies
  - Small register sizes (1-20)
  - Very high 1-qubit gate fidelities (0.999+)
  - Moderately high 2-qubit gate fidelities (0.99+)
  - Limited connectivity with good addressability
  - Low-depth sequences of reliable operations
  - Applications limited by gate noise, controllability
- Early stage vendors are offering access
  - D-Wave, IBM, IonQ, Google, Rigetti, Alibaba
  - Client-server interaction, "cloud" model
  - Very loose integration with modern computing



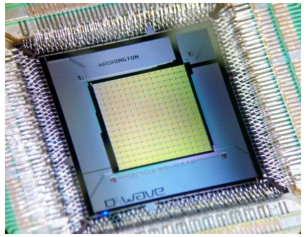
Superconducting chip from IBM



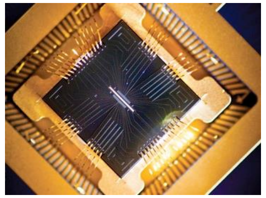
Superconducting chip from Rigetti



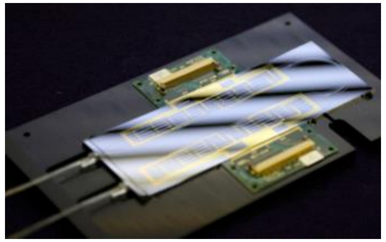
Superconducting chip from Google



Superconducting chip from D-Wave Systems

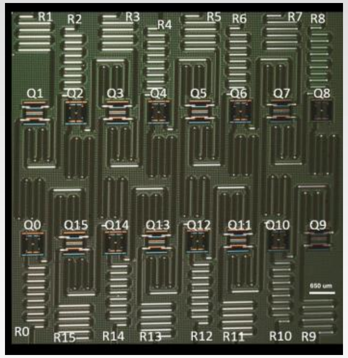
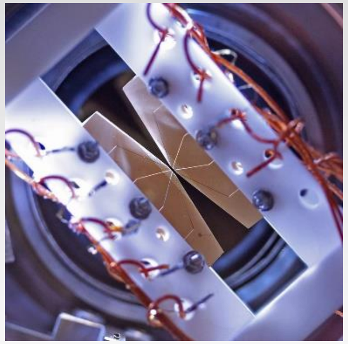
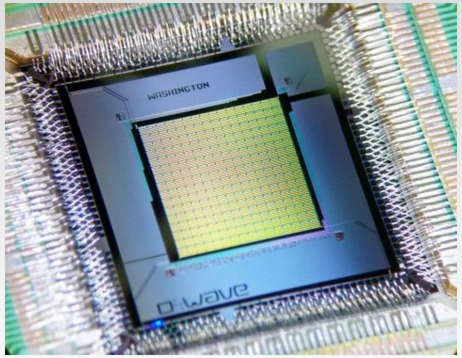


Ion trap chip from Sandia



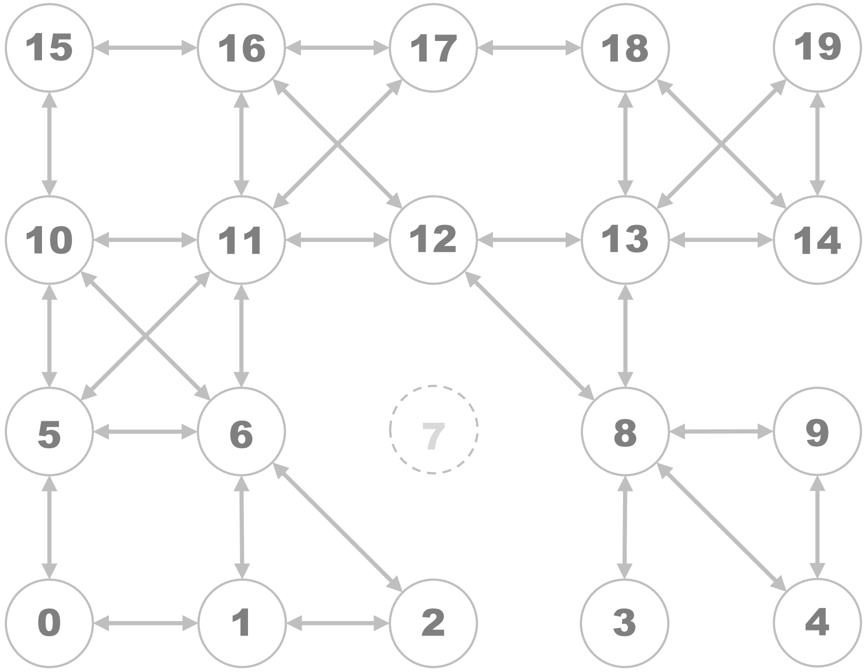
Linear optical chip from Univ. Bristol/QET Labs

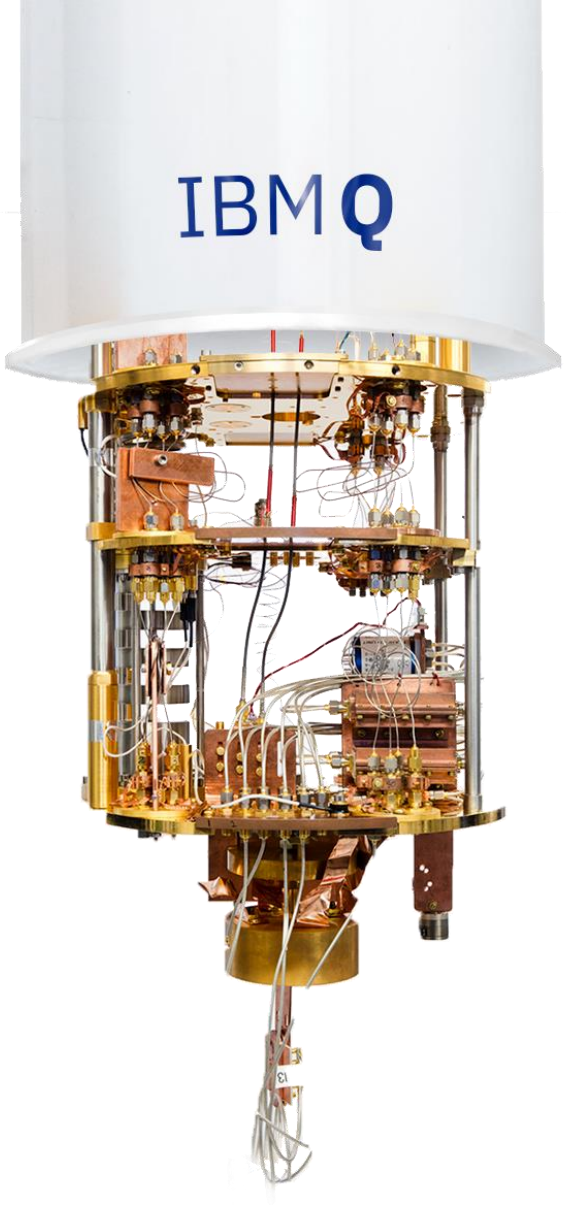
Measuring Quantum Computer Capabilities

Metrics	IBM	IonQ	D-Wave
			
Scale of qubits	5-50	5-79	2048
Initialization fidelity	95%	95%	99.9%
Gate set fidelity	99-95%	99-97%	N/A
Duty cycle	400	2,000	10^(-1)
Measurement fidelity	95%	95%	99.9%
Swap fidelity	98%	97%	N/A
Transport fidelity	N/A	N/A	N/A

Quantum Processing Unit (QPU)

IBM Q "Tokyo"



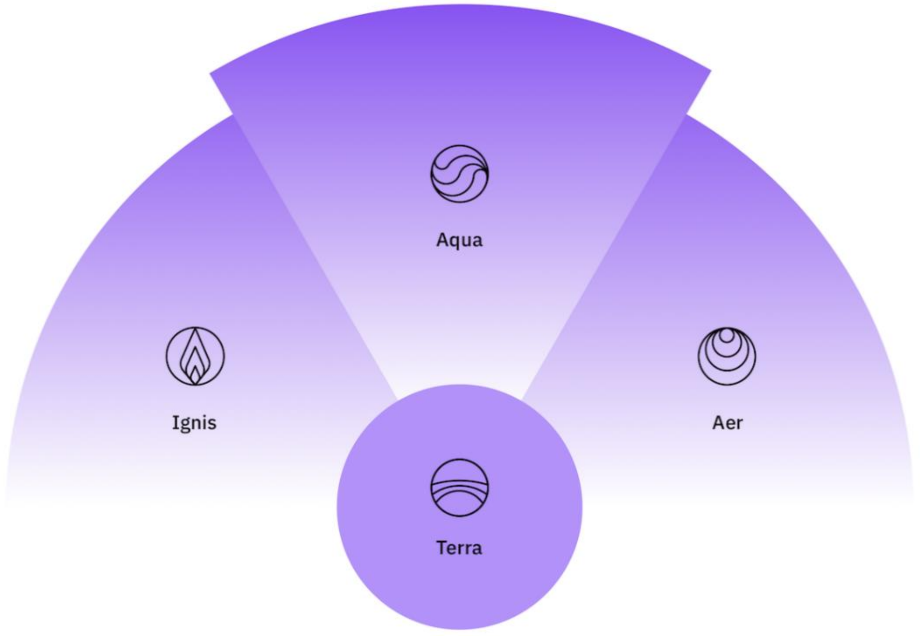


OAK RIDGE National Laboratory

Right Image: IBM

Interfacing with the QPU

- IBM's Qiskit framework
- Software support for quantum computing
  - Running experiments on QPUs (Terra)
  - Simulating quantum circuits (Aer)
  - Draw from quantum algorithm libraries (Aqua)
  - Study and mitigate quantum noise (Ignis)



Images: Qiskit Docs

Example Programs

Qiskit Code

```
# Create a Quantum Register with 2 qubits
q = QuantumRegister(2)
# Create a Classical Register with 2 bits
c = ClassicalRegister(2)
# Create a Quantum Circuit
qc = QuantumCircuit(q, c)

#Create Bell state and measure
qc.h(q[0])
qc.cx(q[0], q[1])
qc.measure(q, c)

# See a list of available devices
print("IBMQ backends: ", IBMQ.backends())

# Compile and run the quantum circuit on a device
backend = IBMQ.get_backend('tokyo')
job_bellstate = execute(qc, backend, shots=8192)
result = job_bellstate.result()
```

QASM Code

```
OPENQASM 2.0;
include "qelib1.inc";
qreg q[3];
creg c0[1];
creg c1[1];
creg c2[1];
h q[1];
cx q[1],q[2];
y q[0];
x q[0];
measure q[0] -> c0[0];
measure q[1] -> c1[0];
measure q[2] -> c2[0];
```

Methods of Characterizing QPUs

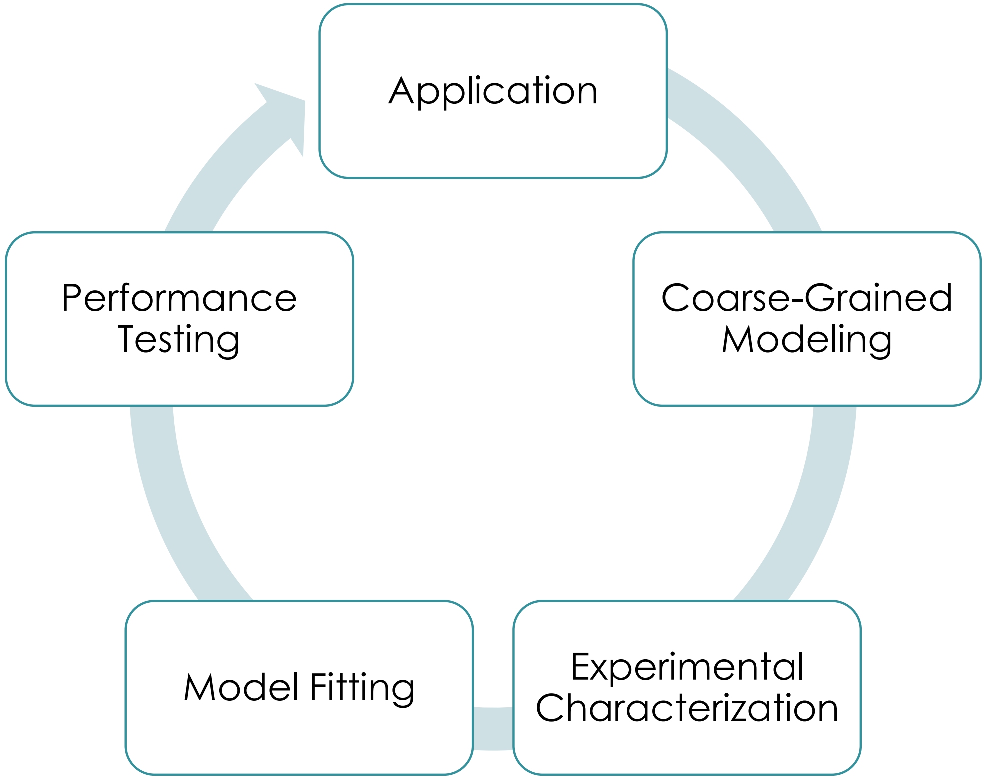
Fine-grain physics models

- Examples: quantum state tomography (QST), quantum process tomography (QPT), gate set tomography (GST), randomized benchmarking (RB)
- Very accurate and detailed description of the processor
- Computationally expensive
- Scales poorly with size of QPU

Coarse-grain circuit models

- Noisy circuit descriptions with reduced dimensionality
  - Empirical approach to inform descriptions of the processor
  - Varies depending on the experiment
- Approximate and effective description of the processor
- Computationally efficient
- Scales well with size of QPU

Experimental Characterization Workflow

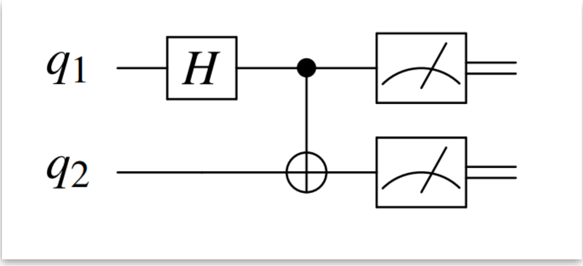


- The needs of the application determine what components we characterize.
- We run experiments on quantum hardware to perform these characterizations.
- We use simulation to test selected noise models.
- By comparing these simulations to the experimental results, we build noise models with the best fit to the experimental data.
- This process can be performed iteratively based on performance testing.

Application Example

Bell state

$$|\Phi^+\rangle = \frac{1}{\sqrt{2}}(|0_1 0_2\rangle + \ |1_1 1_2\rangle)$$

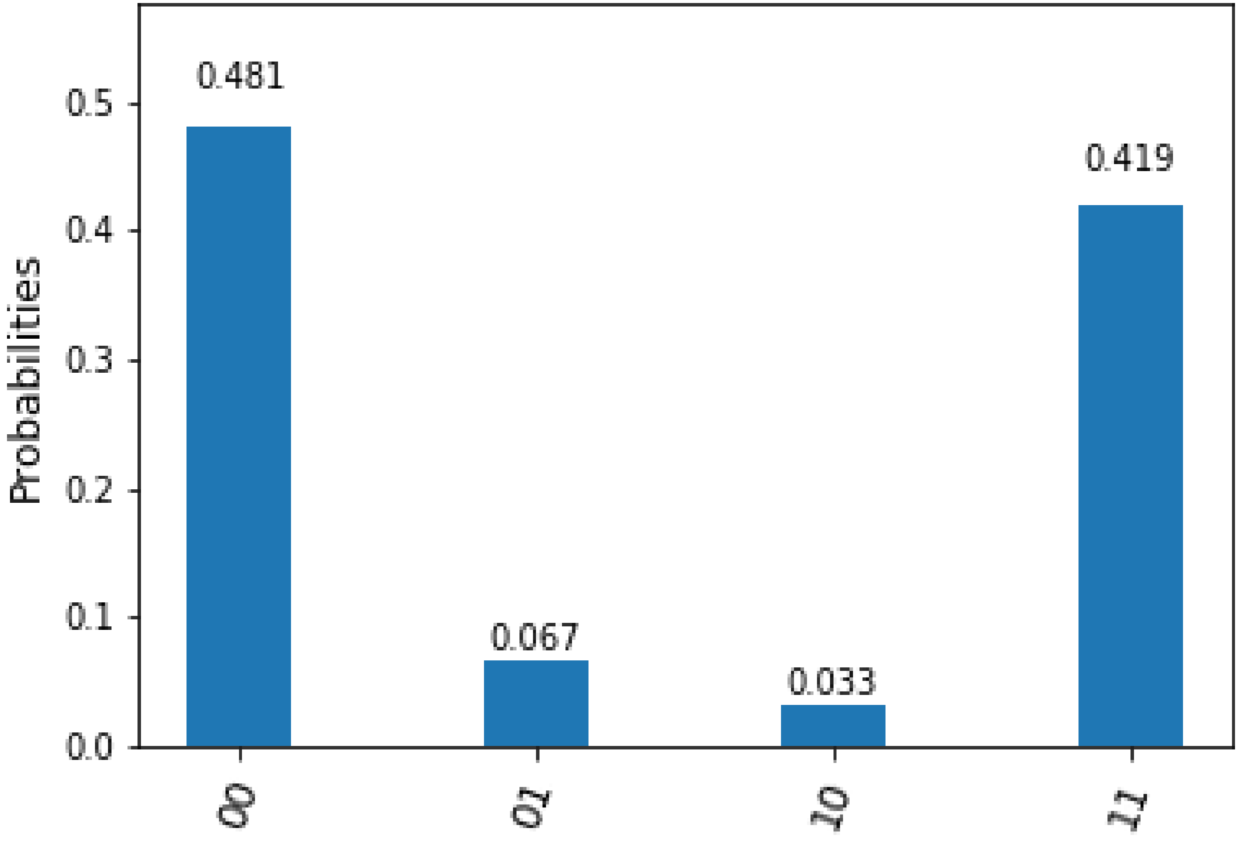


The Bell state represents an example of quantum superposition and entanglement in a simple circuit that can be implemented on existing hardware.

We characterize the pieces of this circuit – H, CNOT, and measurement – to find a noisy composite description of the hardware that closely matches experimental data.

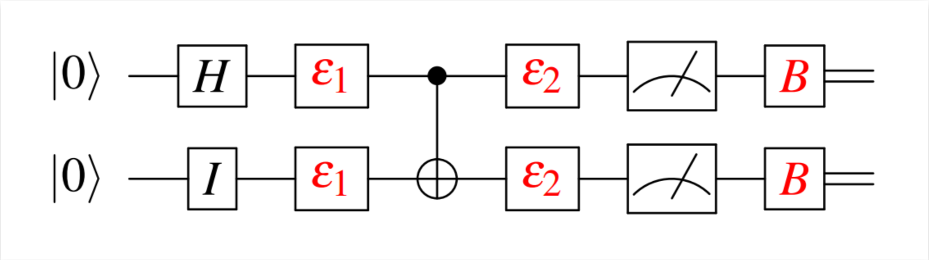
Example of Bell state data from "Tokyo"

Probabilities of bit string results out of 8192



OAK RIDGE National Laboratory

Noise Models



Using a bootstrapping approach, we can piece together a composite model for this circuit by considering smaller circuit examples.

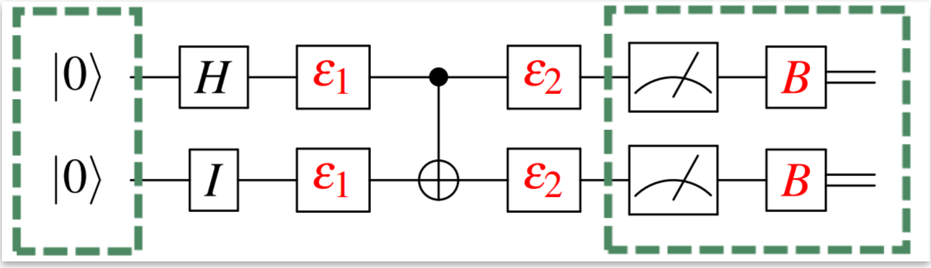
Error models:

- Depolarizing error
- Unitary rotations
- Symmetric/asymmetric bit flip

Components:

- Single qubit gates
- Two-qubit gates
- Readout

Noise Models



Starting with initialization and measurement, we characterize readout.

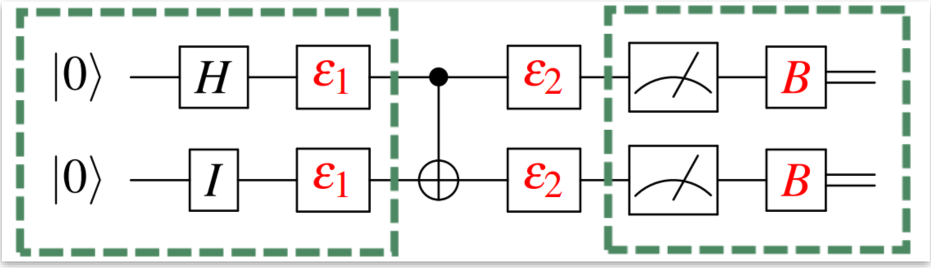
Error models:

- Depolarizing error
- Unitary rotations
- Symmetric/asymmetric bit flip

Components:

- Single qubit gates
- Two-qubit gates
- Readout

Noise Models

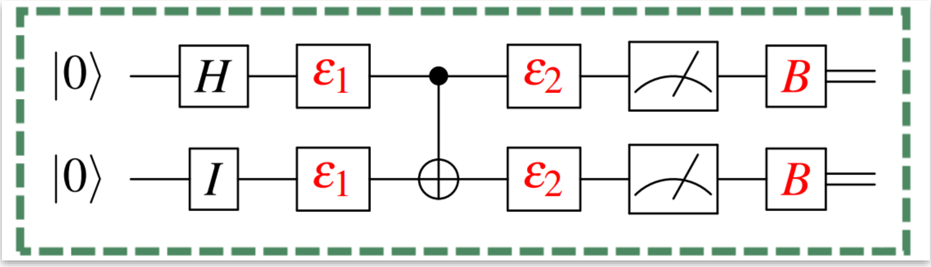


Using the description of readout noise we find from the previous example, we find a noise model that characterizes *H* gates.

- Depolarizing error
- Unitary rotations
- Symmetric/asymmetric bit flip

- Single qubit gates
- Two-qubit gates
- Readout

Noise Models

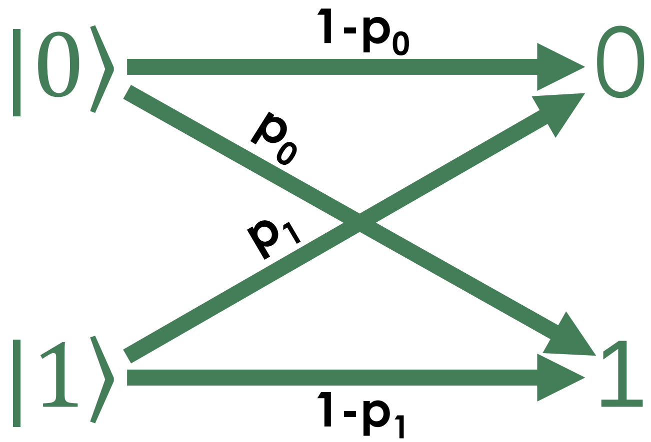


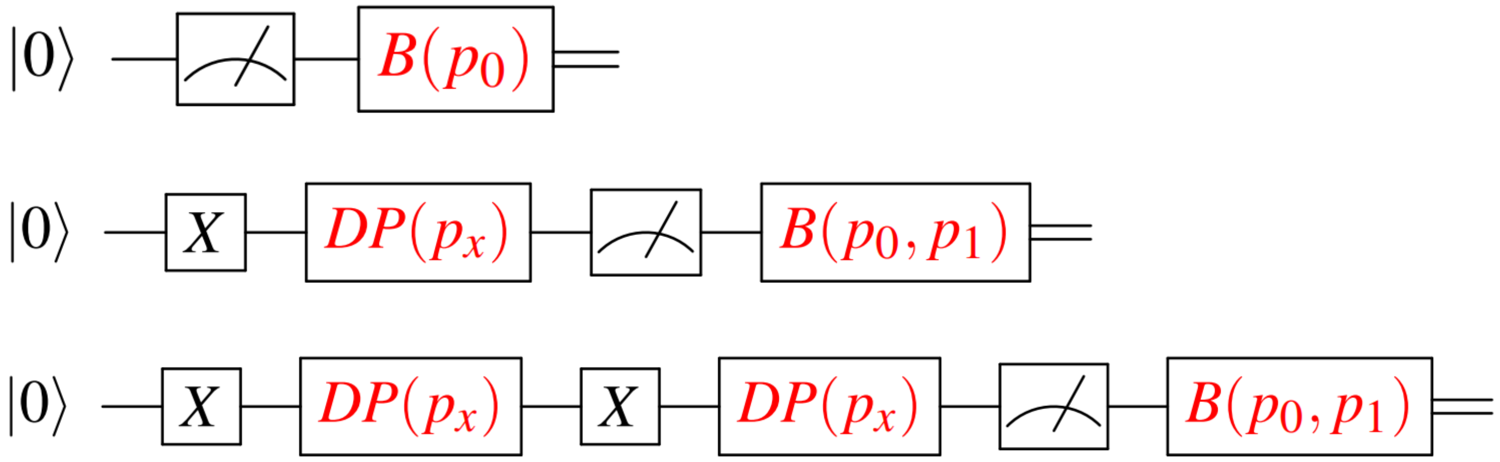
Finally, we characterize CNOT error using a full Bell state circuit.

- Depolarizing error
- Unitary rotations
- Symmetric/asymmetric bit flip

- Single qubit gates
- Two-qubit gates
- Readout

Asymmetric Readout (ARO) Parameters

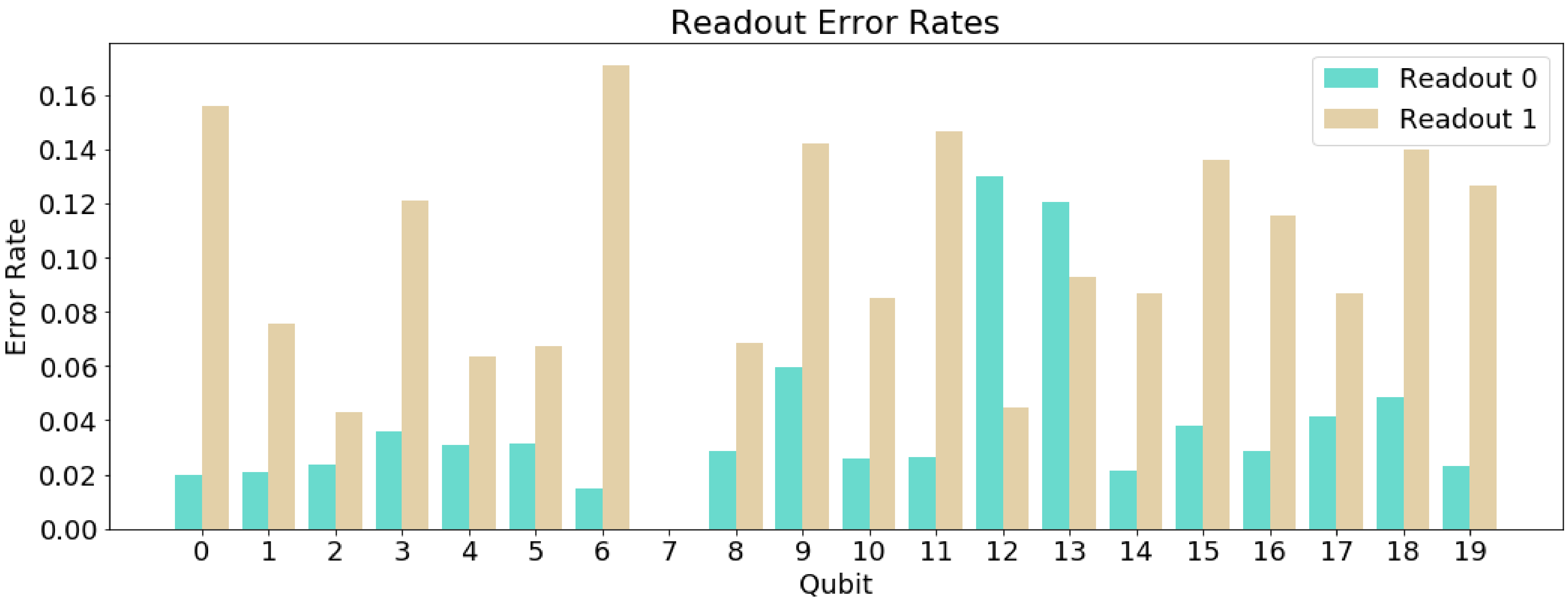




$$p^{obs}_{X\ gate}(0) = \frac{2p_X}{3}(1-p_0) + \left(1 - \frac{2p_X}{3}\right)p_1$$

$$p^{obs}_{XX\ gates}(0) = \left[\left(1 - \frac{2p_X}{3}\right)^2 + \left(\frac{2p_X}{3}\right)^2\right](1-p_0) + \left[\frac{4p_X}{3}\left(1 - \frac{2p_X}{3}\right)\right]p_1$$

OAK RIDGE National Laboratory



Readout Error Rates
Readout 0
Readout 1
Error Rate
0.00
0.02
0.04
0.06
0.08
0.10
0.12
0.14
0.16
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
Qubit

19-qubit averages: $p_0 = 0.0385$

$p_1 = 0.0984$


X Gate Error Rates
Error Rate
0.08
0.07
0.06
0.05
0.04
0.03
0.02
0.01
0.00
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19
Qubit

CNOT Depolarizing Parameters

For depolarizing noise, defined as a probability p_{DP} of a Pauli X, Y, or Z operation, we label results after the depolarizing channel with ij and after measurement with asymmetric readout error (ARO) as kl; we obtain a value for p_{DP} that makes $p^{obs}(kl) = \sum_{ij} p_{ij}(kl)$ true.

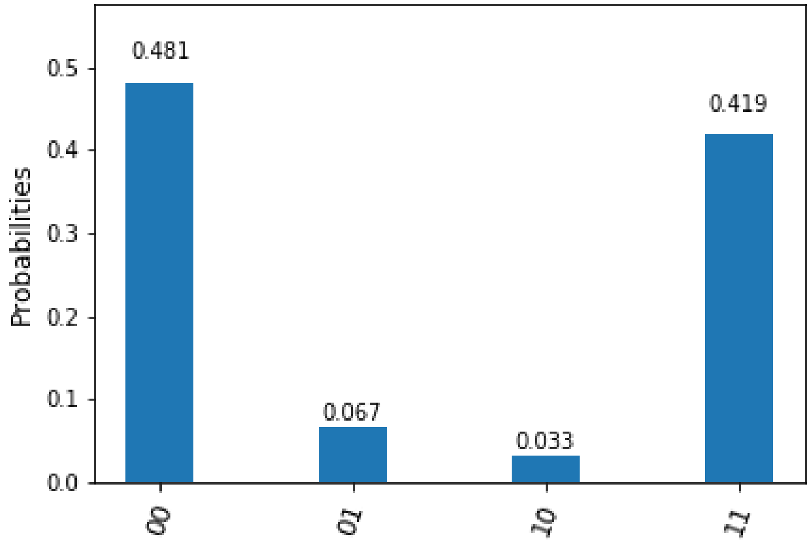
$$p(ij) = Tr\left[\Pi_{ij}\epsilon_{DP}\left(U_{BS}|0\rangle\langle 0|U_{BS}^{\dagger}\right)\right],\ p_{ij}(kl) = ARO\left(p(ij)\right)$$

$$p_{00}(kl) = p_{11}(kl) = ARO\left(\left[\frac{1}{2}(1-p_{DP})^2 + p_{DP}(1-p_{DP}) + \frac{3}{2}p_{DP}^2\right]\right)$$

$$p_{01}(kl) = p_{10}(kl) = ARO\left(2p_{DP}(1-p_{DP})\right)$$

Example of CNOT Model Fitting

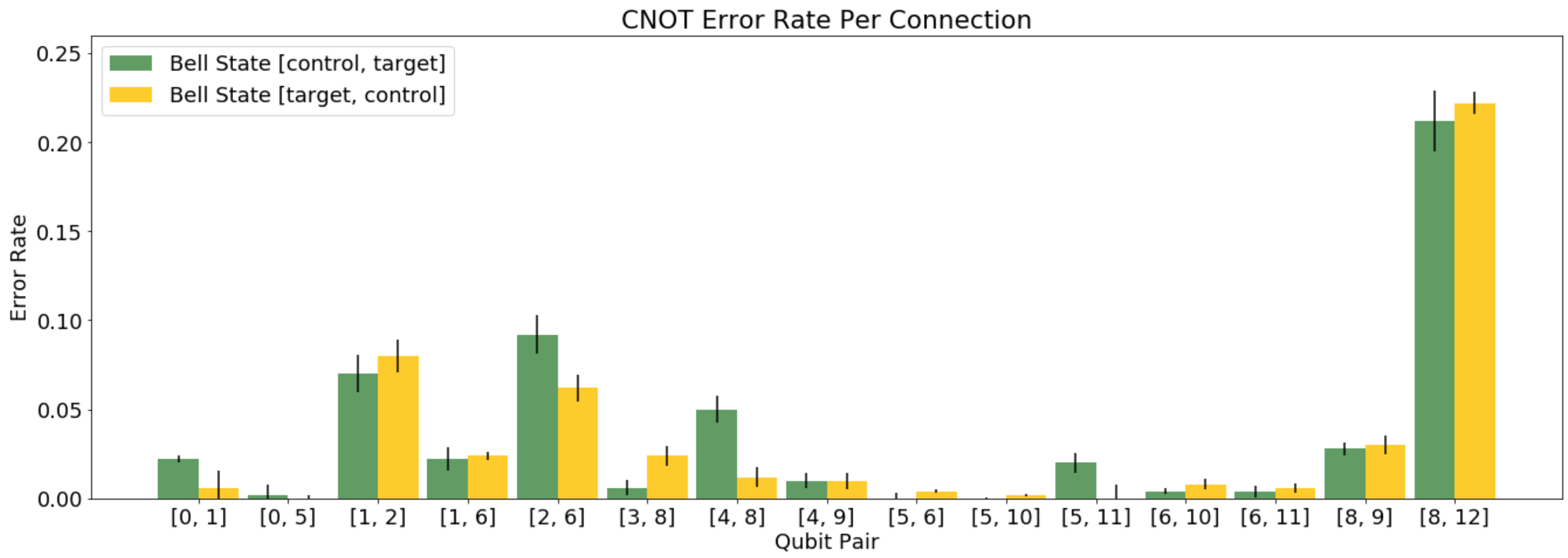
Experimental results for a single coupling

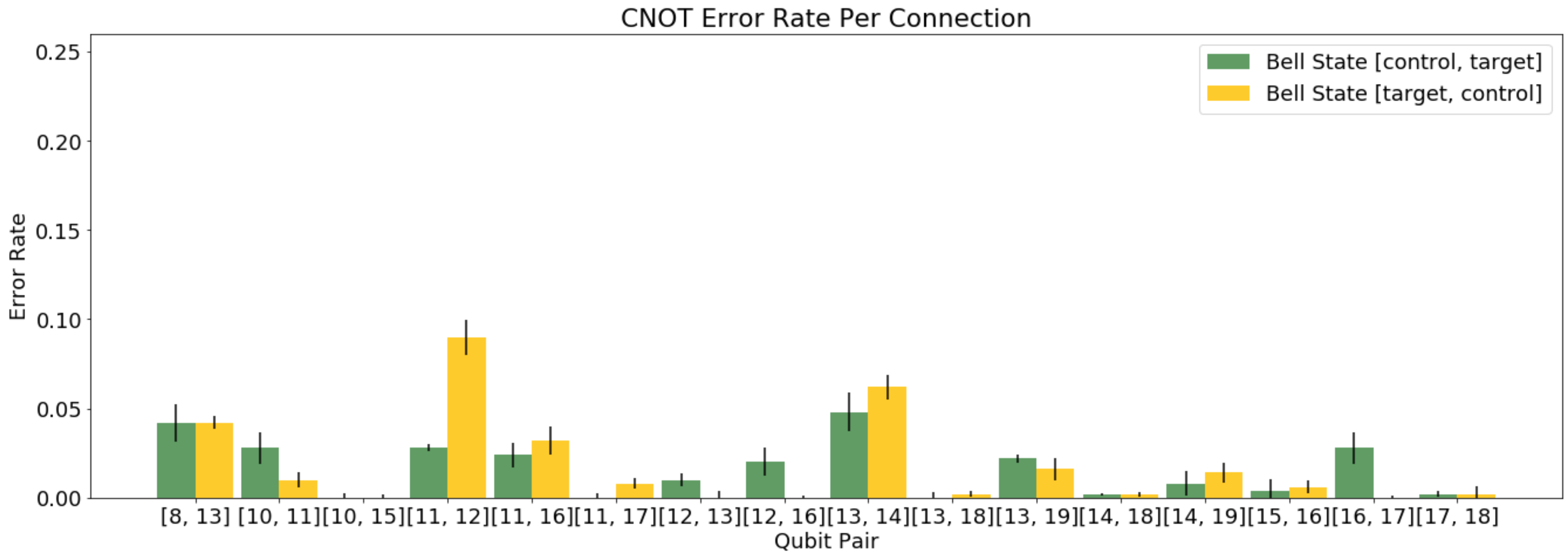


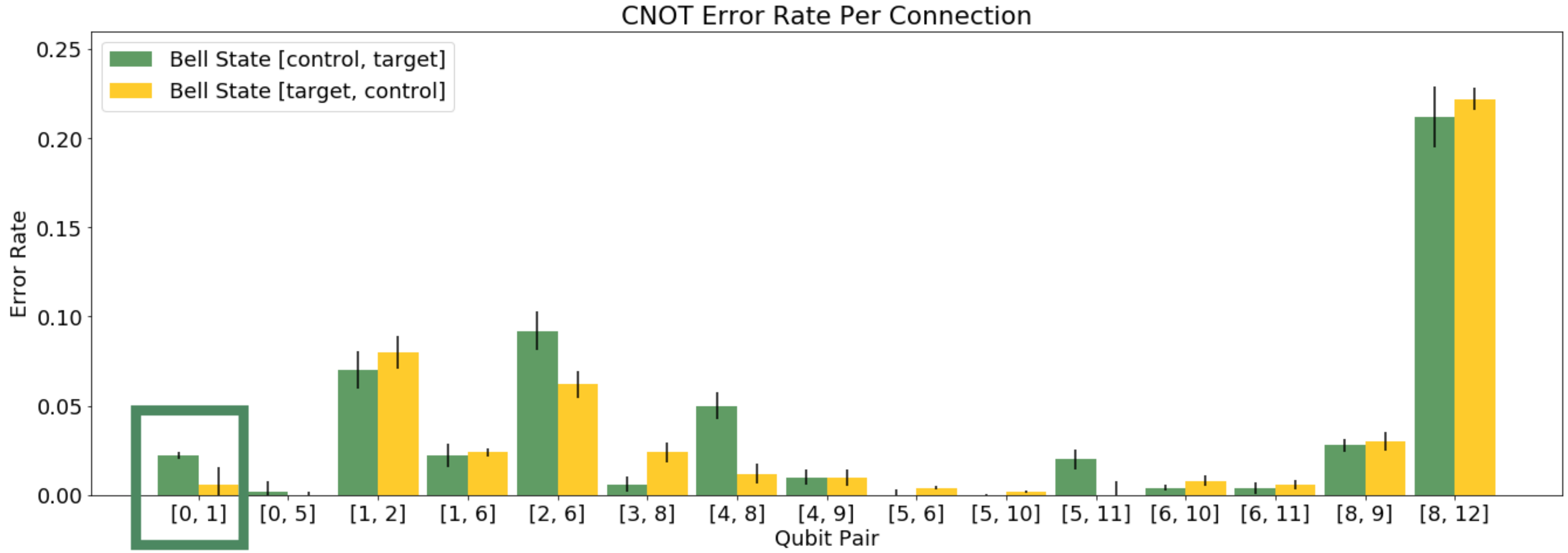
- Inject noise into simulations of quantum circuits and perform measurement
- Compare simulation results to experimental results using the expression for **model error**:

$$E_{model} = \sum_i \left(\frac{h_i^{exp}}{N} - \frac{h_i^{sim}}{N} \right)^2$$

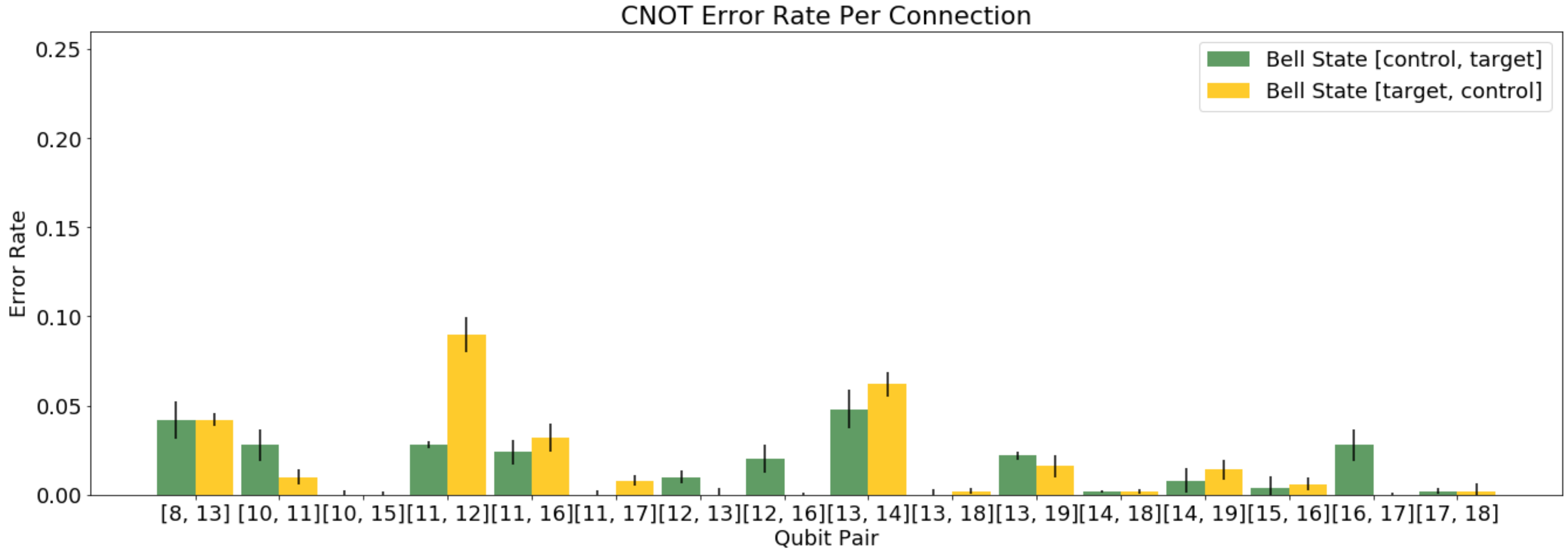
- Minimize this quantity to determine best fit


CNOT Error Rate Per Connection
Bell State [control, target]
Bell State [target, control]
Error Rate
0.25
0.20
0.15
0.10
0.05
0.00
[0, 1]
[0, 5]
[1, 2]
[1, 6]
[2, 6]
[3, 8]
[4, 8]
[4, 9]
[5, 6]
[5, 10]
[5, 11]
[6, 10]
[6, 11]
[8, 9]
[8, 12]
Qubit Pair


CNOT Error Rate Per Connection
Bell State [control, target]
Bell State [target, control]
Error Rate
0.25
0.20
0.15
0.10
0.05
0.00
[8, 13]
[10, 11]
[10, 15]
[11, 12]
[11, 16]
[11, 17]
[12, 13]
[12, 16]
[13, 14]
[13, 18]
[13, 19]
[14, 18]
[14, 19]
[15, 16]
[16, 17]
[17, 18]
Qubit Pair



Average over all 62:
$p_{DP} = 0.0255$



Numerical Simulations of Quantum Circuits

- Classical computer optimized for quantum circuit simulation on site at ORNL
  - TBs of RAM
  - Up to ~40 qubits
- QPU emulator
  - Write quantum circuits in AQASM language
  - Compile to simulator

Image: Atos

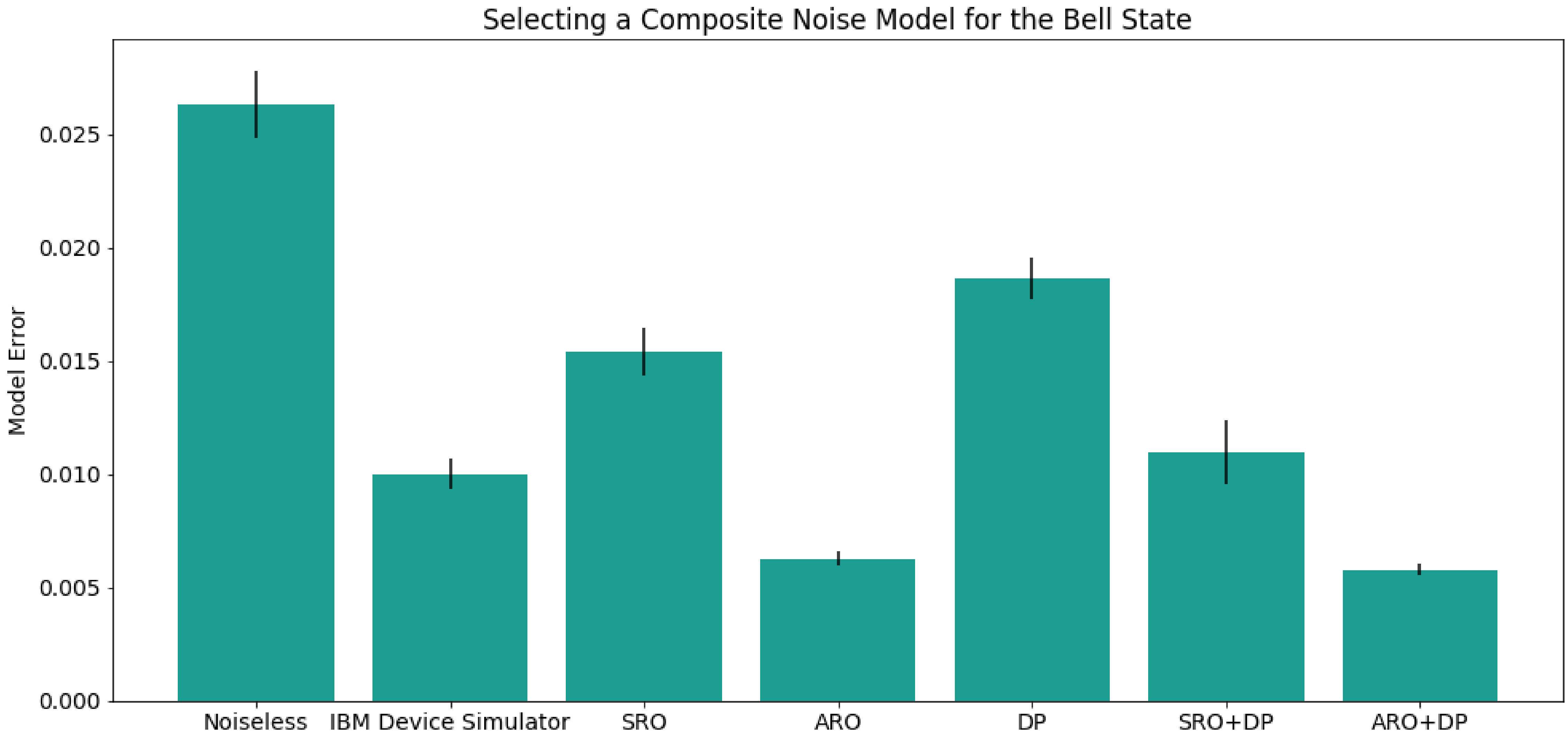
Interfacing with the QLM

- Python control “PyAQASM”
- Create AQASM circuit descriptions
- Execute on chosen simulator
  - Linear algebra
  - Stabilizer
  - MPS
  - Feynman path integral
  - Density matrix

AQASM file example

```
BEGIN
qubits 13
cbits 10

RY[1.7401524607843557] q[3]
PH[1.7150018525366089] q[3]
H q[4]
H q[5]
H q[6]
CNOT q[3],q[2]
CNOT q[3],q[1]
END
```

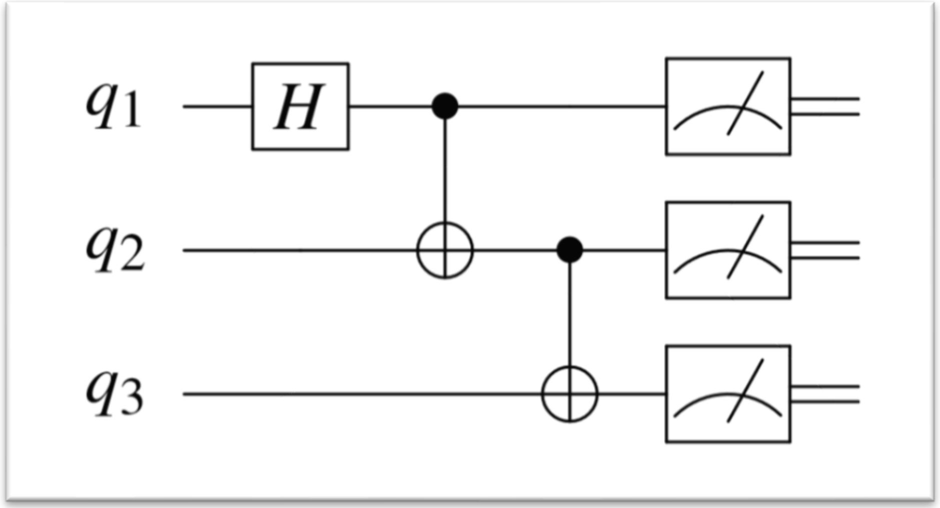


Selecting a Composite Noise Model for the Bell State
Model Error
0.025
0.020
0.015
0.010
0.005
0.000
Noiseless
IBM Device Simulator
SRO
ARO
DP
SRO+DP
ARO+DP

GHZ States

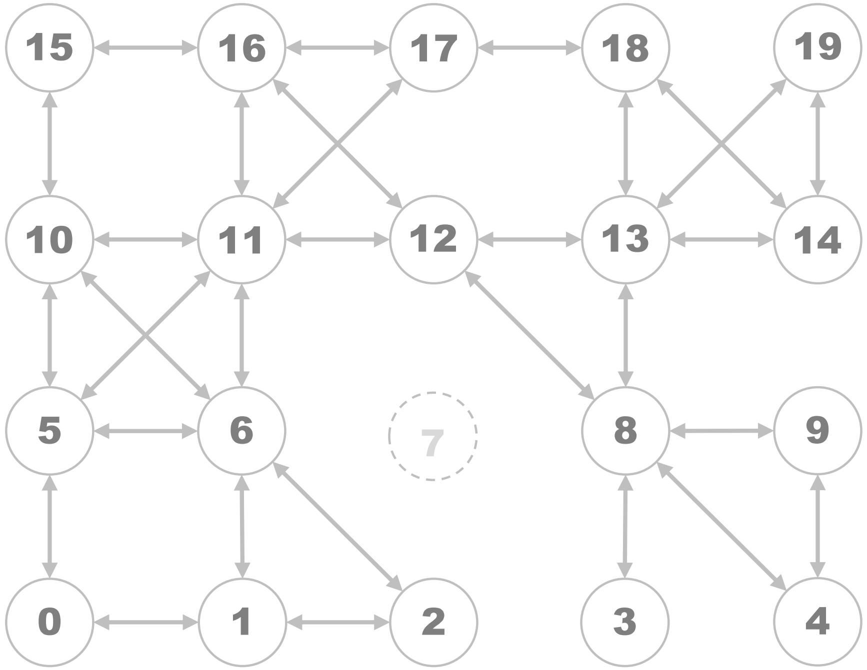
n-qubit GHZ for *n* = {2, 3, …, 20}

$$|GHZ(n)\rangle = \frac{|0\rangle^{\otimes n} + |1\rangle^{\otimes n}}{\sqrt{2}}$$



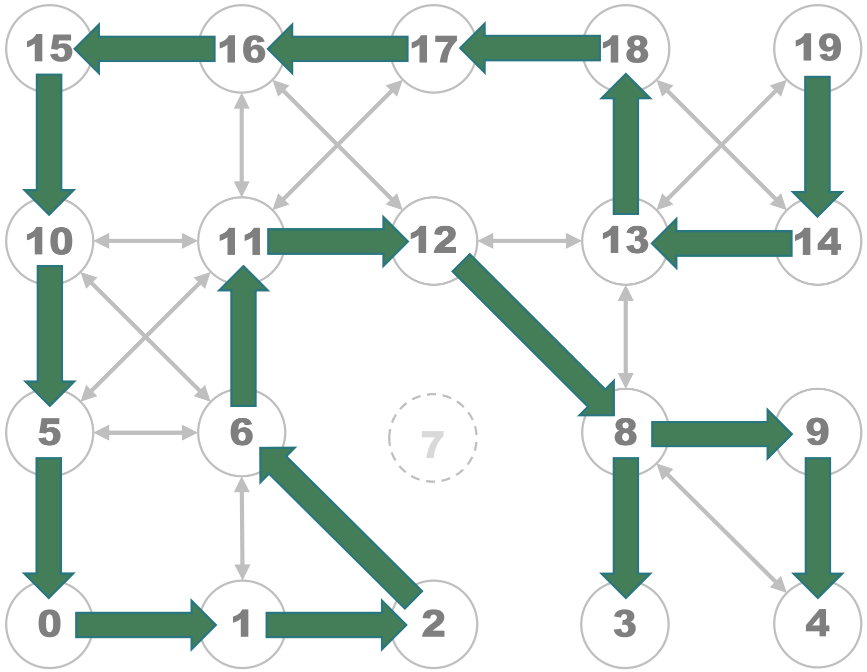
GHZ on Tokyo

"Tokyo" layout at time of data collection.



GHZ on Tokyo

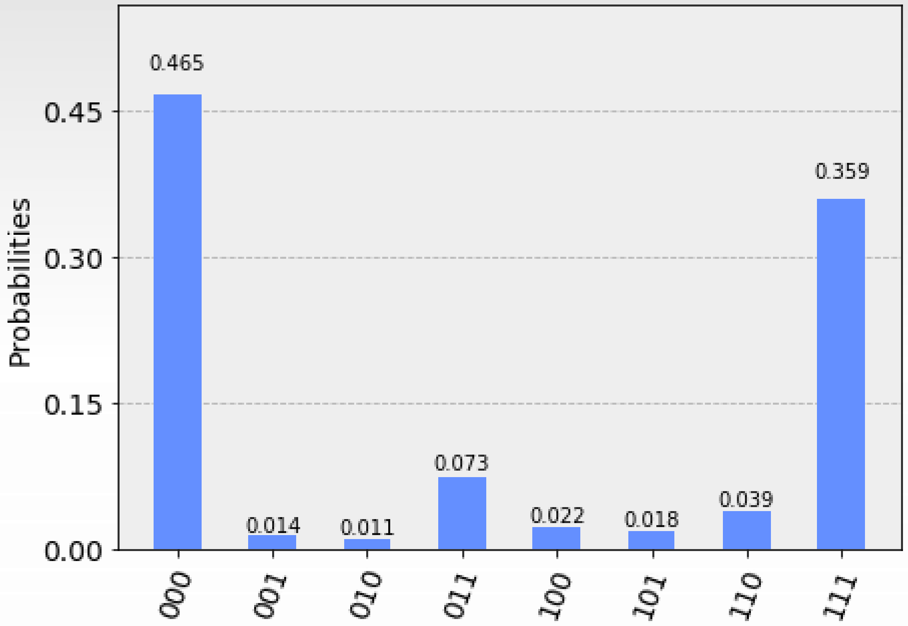
For $n = \{2, 3, \ldots, 20\}$, we map the GHZ circuits onto the chip as shown (arrow from control to target).



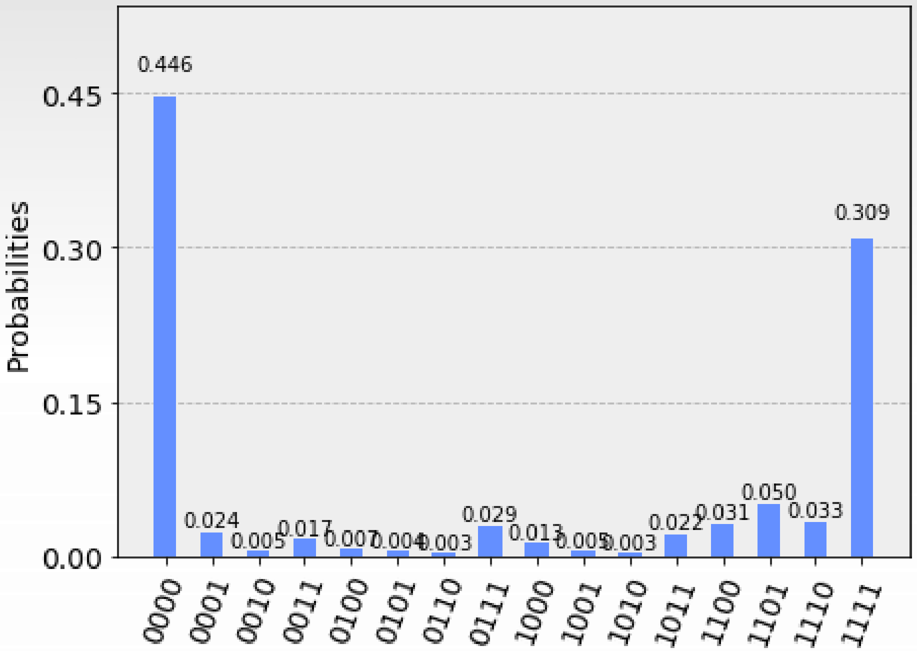
OAK RIDGE National Laboratory

Examples of Experimental Results from QPU

3-qubit GHZ results

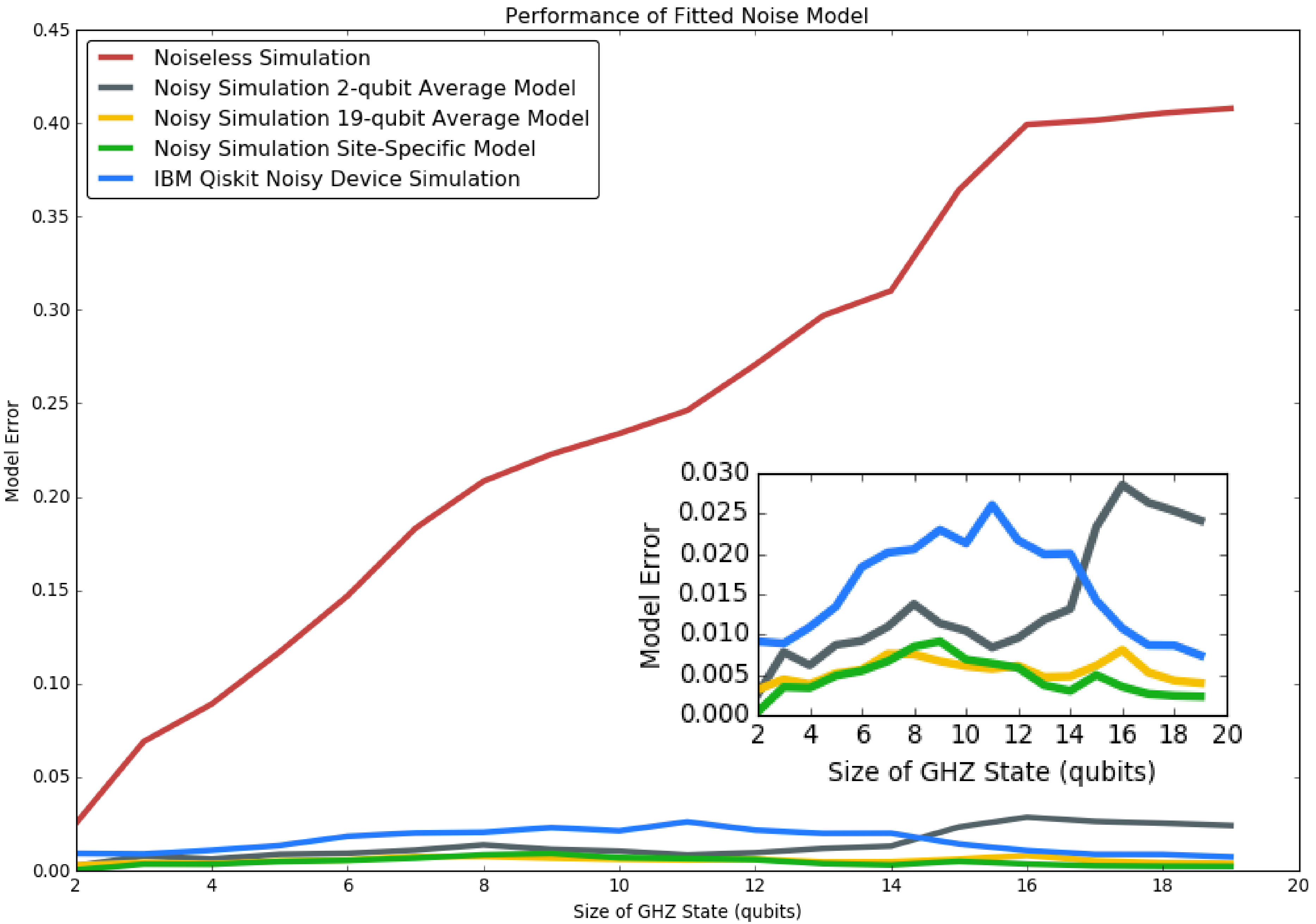


4-qubit GHZ results



GHZ on Tokyo

For GHZ states of increasing size, model error in noisy simulation increases far less dramatically than noiseless and remains under 3% even for the largest circuits.



Oak Ridge National Laboratory

Conclusions

- Coarse-grained, application-focused noise models can be used to predict the performance of NISQ devices.
- Development of these noise models requires few computational resources.
  - Needs as few as 4 characterization circuits
  - Yields as few as 3 noise parameters
- Coarse-graining is an iterative process driven by required accuracy.
- Future work will include more refined noise models, exploration of other applications, and comparisons to other characterization methods.

Thank you

Oak Ridge National Laboratory

Bonus Slides

Aer “Basic Device Noise Model”

- Input RB error rates from daily calibration and device properties from selected backend
- One- and two-qubit gate errors
  - Determine thermal relaxation error from T1, T2, and gate times
  - Add a depolarizing probability parameter such that the error rates of DP+TR=RB
- Readout error
  - Use reported readout error from RB protocol as symmetric bit flip channel

Code example

```
'''Qiskit-provided basic device noise model'''
# Choose a real device to simulate
device = IBMQ.get_backend('tokyo')
properties = device.properties()
coupling_map = device.configuration().coupling_map

# Generate an Aer noise model for device
noise_model = noise.device.basic_device_noise_model(properties)
basis_gates = noise_model.basis_gates

#Define registers
q = QuantumRegister(20)
c = ClassicalRegister(20)
#Circuit list
bell = QuantumCircuit(q, c)
bell.h(q[0])
bell.cx(q[0],q[1])
bell.barrier()
bell.measure(q, c)

pm = PassManager()
pm.append(Unroller(['u1', 'u2', 'u3', 'cx', 'id']))

# Perform noisy simulation
backend = Aer.get_backend('qasm_simulator')
job_sim = execute(circuits, backend, shots=8192,
                  pass_manager = pm,
                  coupling_map=coupling_map,
                  noise_model=noise_model,
                  basis_gates=basis_gates)
```

Aer "Basic Device Noise Model"

- Input RB error rates from daily calibration and device properties from selected backend
- One- and two-qubit gate errors
  - Determine thermal relaxation error from T1, T2, and gate times
  - Add a depolarizing probability parameter such that the error rates of **DP=RB**
- Readout error
  - Use reported readout error from RB protocol as symmetric bit flip channel

Code example

```
'''Qiskit-provided basic device noise model'''
# Choose a real device to simulate
device = IBMQ.get_backend('tokyo')
properties = device.properties()
coupling_map = device.configuration().coupling_map

# Generate an Aer noise model for device
noise_model = noise.device.basic_device_noise_model(properties)
basis_gates = noise_model.basis_gates

#Define registers
q = QuantumRegister(20)
c = ClassicalRegister(20)
#Circuit list
bell = QuantumCircuit(q, c)
bell.h(q[0])
bell.cx(q[0],q[1])
bell.barrier()
bell.measure(q, c)

pm = PassManager()
pm.append(Unroller(['u1', 'u2', 'u3', 'cx', 'id']))

# Perform noisy simulation
backend = Aer.get_backend('qasm_simulator')
job_sim = execute(circuits, backend, shots=8192,
                  pass_manager = pm,
                  coupling_map=coupling_map,
                  noise_model=noise_model,
                  basis_gates=basis_gates)
```