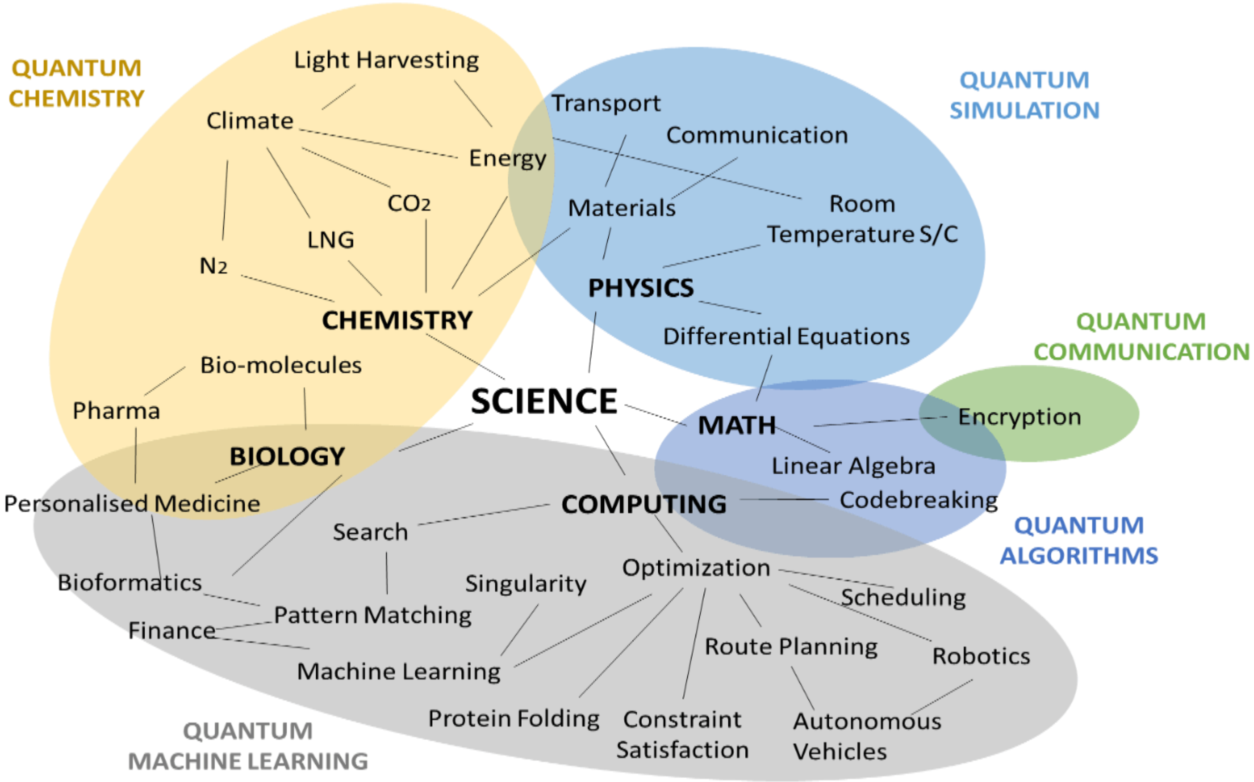
The Abdus Salam
International Centre
for Theoretical Physics

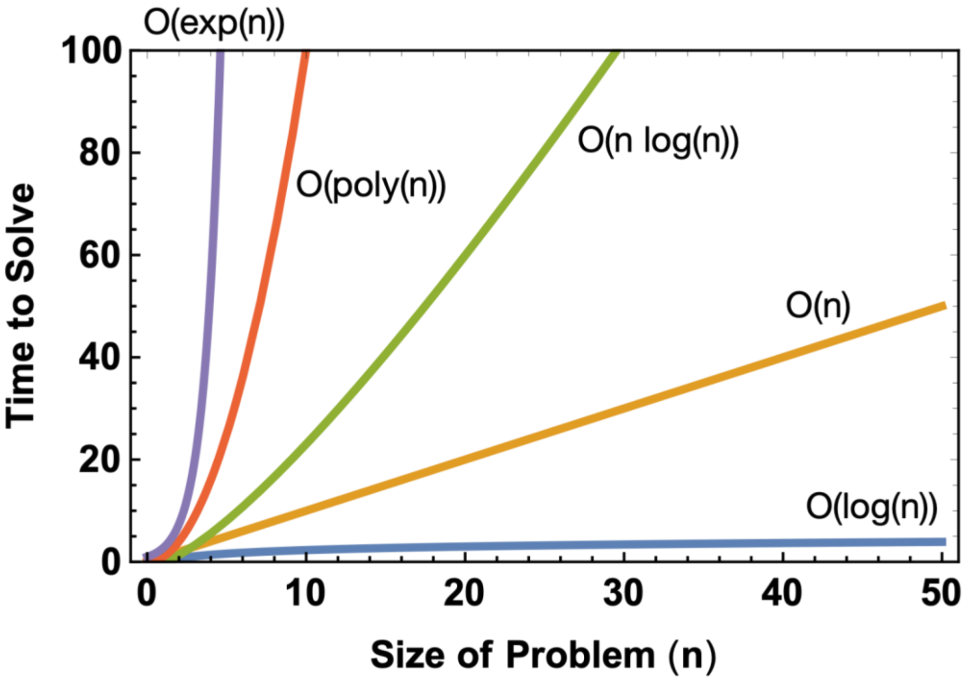
Quantum Computing Lab

Instructor: Mikhail Shalaginov, MIT

Why do we need quantum computers?



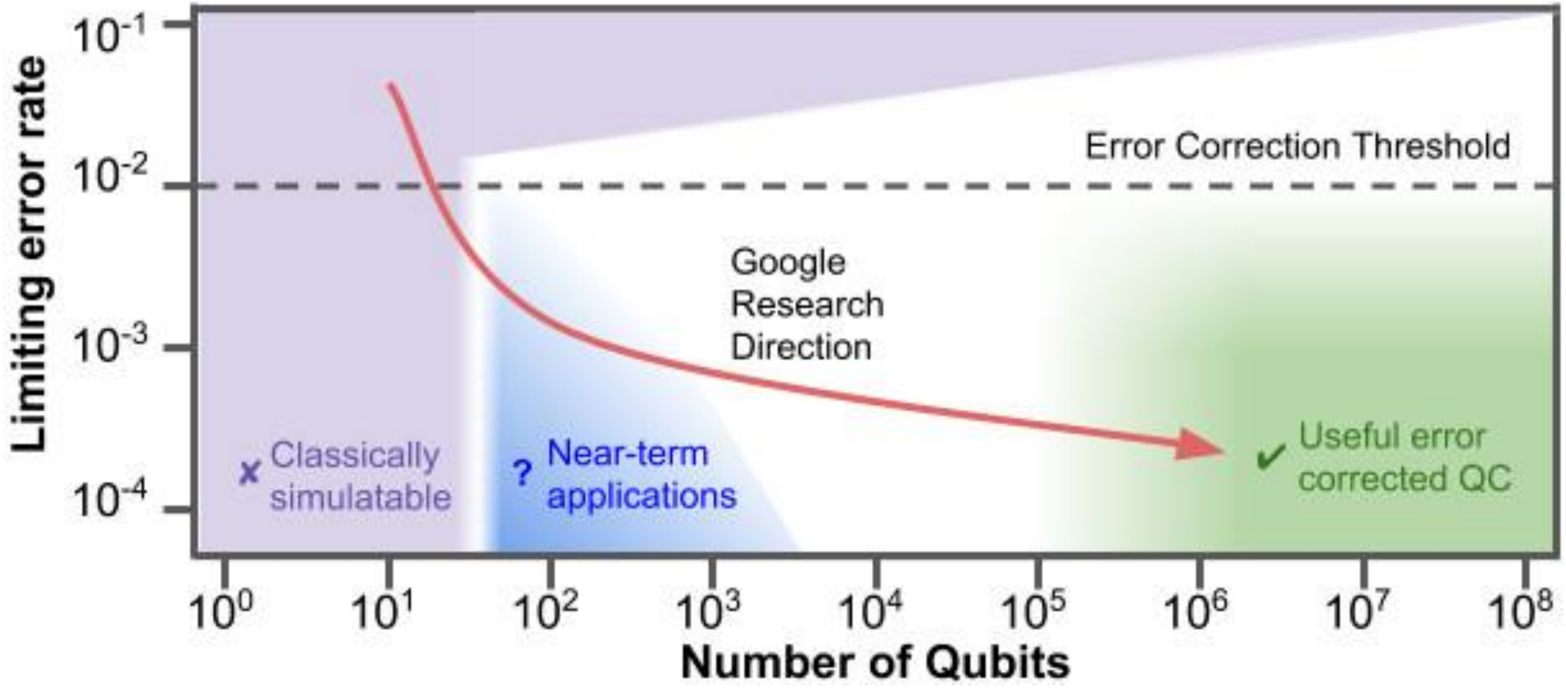
Still a lot of problems to solve

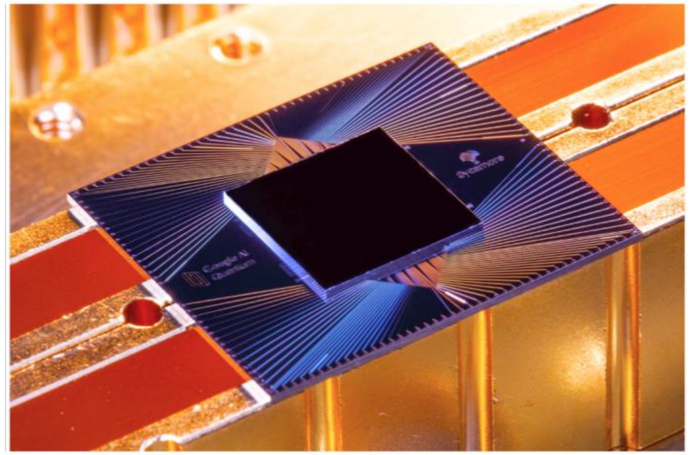


Problems of increased complexity

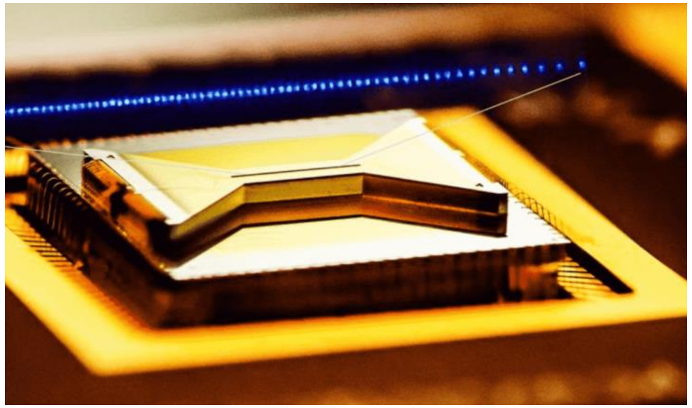
Where are we now? What is next?

Era of NISQ (Noisy Intermediate-Scale Quantum)

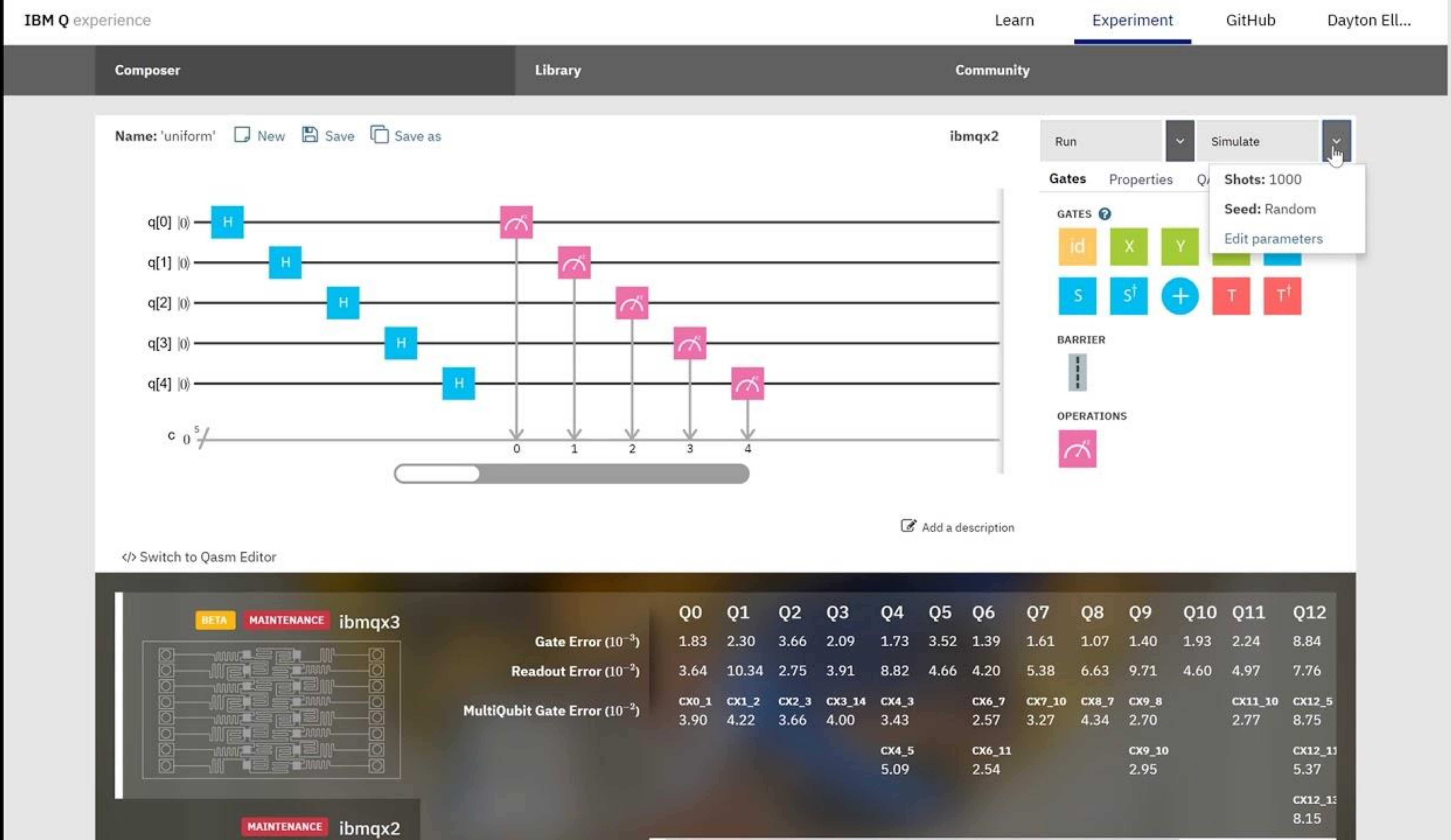




Sycamore chip with
53 superconducting qubits
(Google)

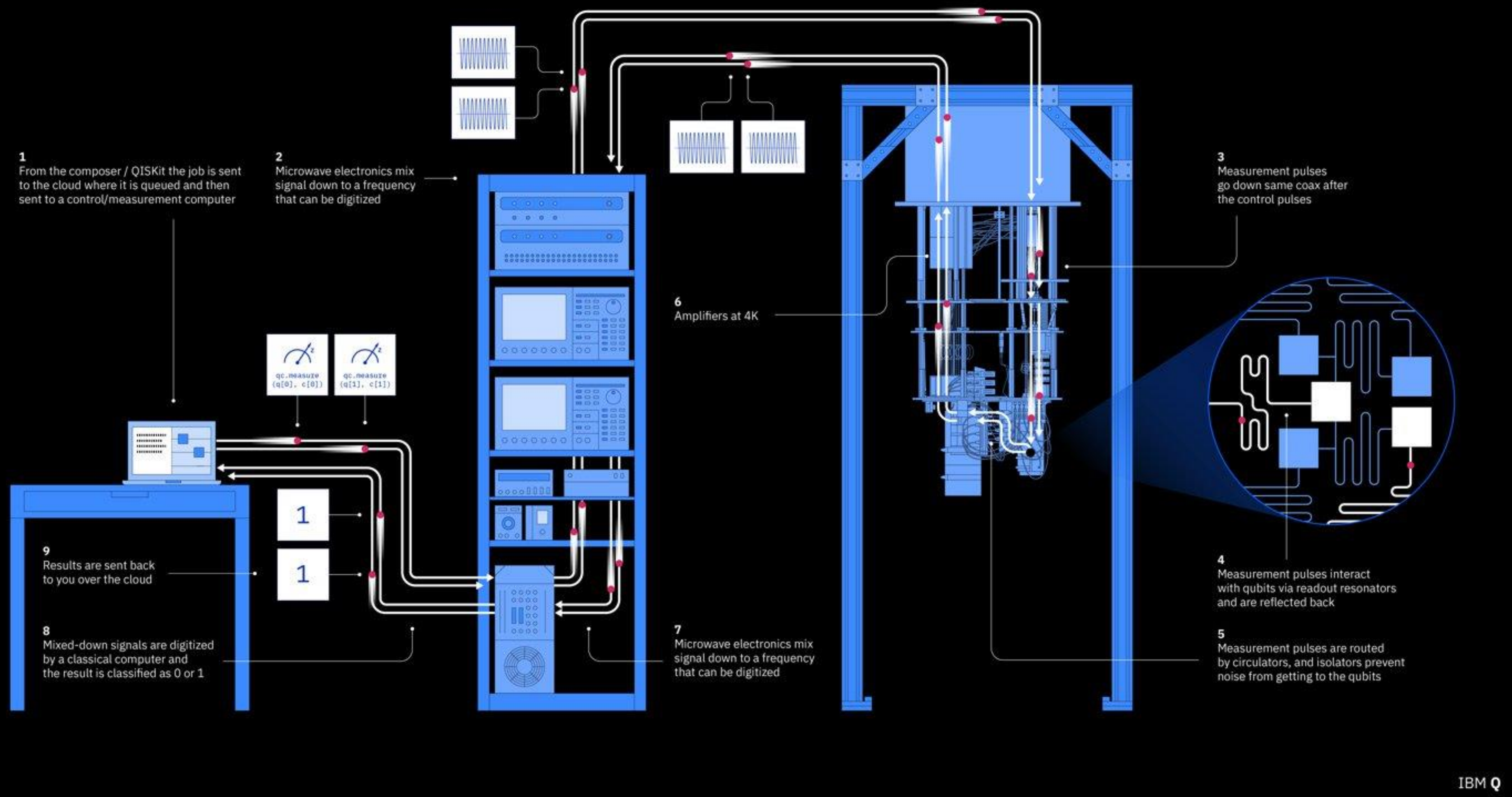


trapped ion/atom arrays
of ~100 qubits
(Harvard, U of Maryland)

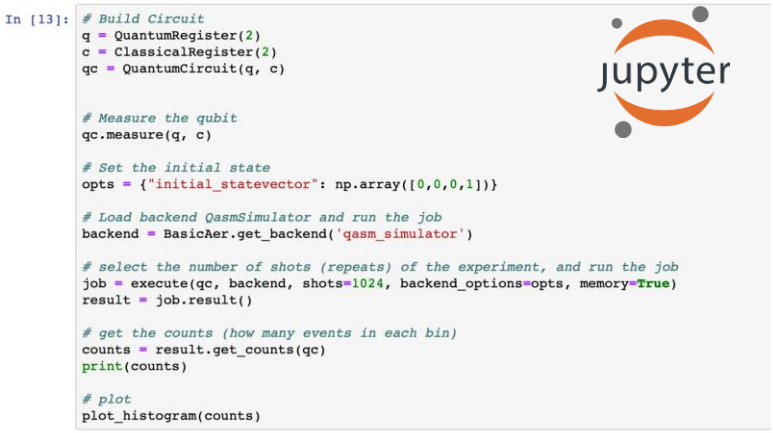


	Q0	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10	Q11	Q12
Gate Error (10^{-3})	1.83	2.30	3.66	2.09	1.73	3.52	1.39	1.61	1.07	1.40	1.93	2.24	8.84
Readout Error (10^{-2})	3.64	10.34	2.75	3.91	8.82	4.66	4.20	5.38	6.63	9.71	4.60	4.97	7.76
MultiQubit Gate Error (10^{-2})	CX0_1 3.90	CX1_2 4.22	CX2_3 3.66	CX3_14 4.00	CX4_3 3.43		CX6_7 2.57	CX7_10 3.27	CX8_7 4.34	CX9_8 2.70		CX11_10 2.77	CX12_5 8.75
					CX4_5 5.09		CX6_11 2.54			CX9_10 2.95			CX12_11 5.37
													CX12_13 8.15

Big picture of Quantum Computing process on IBM machines



Lab Goals

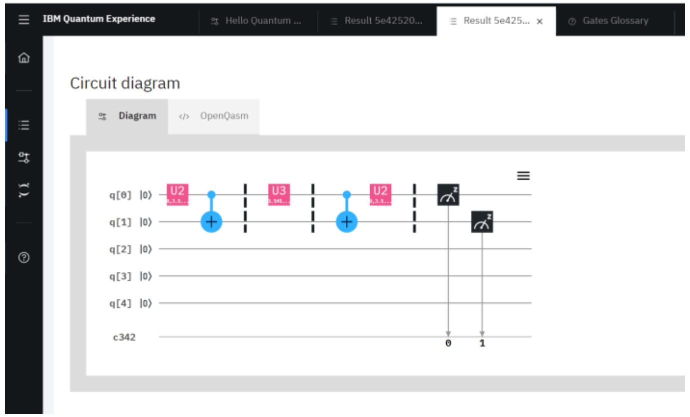


```
In [13]: # Build Circuit
q = QuantumRegister(2)
c = ClassicalRegister(2)
qc = QuantumCircuit(q, c)

# Measure the qubit
qc.measure(q, c)

# Set the initial state
opts = {"initial_statevector": np.array([0,0,0,1])}

# Load backend QasmSimulator and run the job
backend = BasicAer.get_backend('qasm_simulator')

# select the number of shots (repeats) of the experiment, and run the job
job = execute(qc, backend, shots=1024, backend_options=opts, memory=True)
result = job.result()

# get the counts (how many events in each bin)
counts = result.get_counts(qc)
print(counts)

# plot
plot_histogram(counts)
```

1. Learn basics of coding in Python (Jupyter Notebook)

2. Build quantum logic circuits quantum using Qiskit framework

3. Use actual quantum nodes for running the developed algorithms



Initial instructions

1. Log in to your Linux Ubuntu account

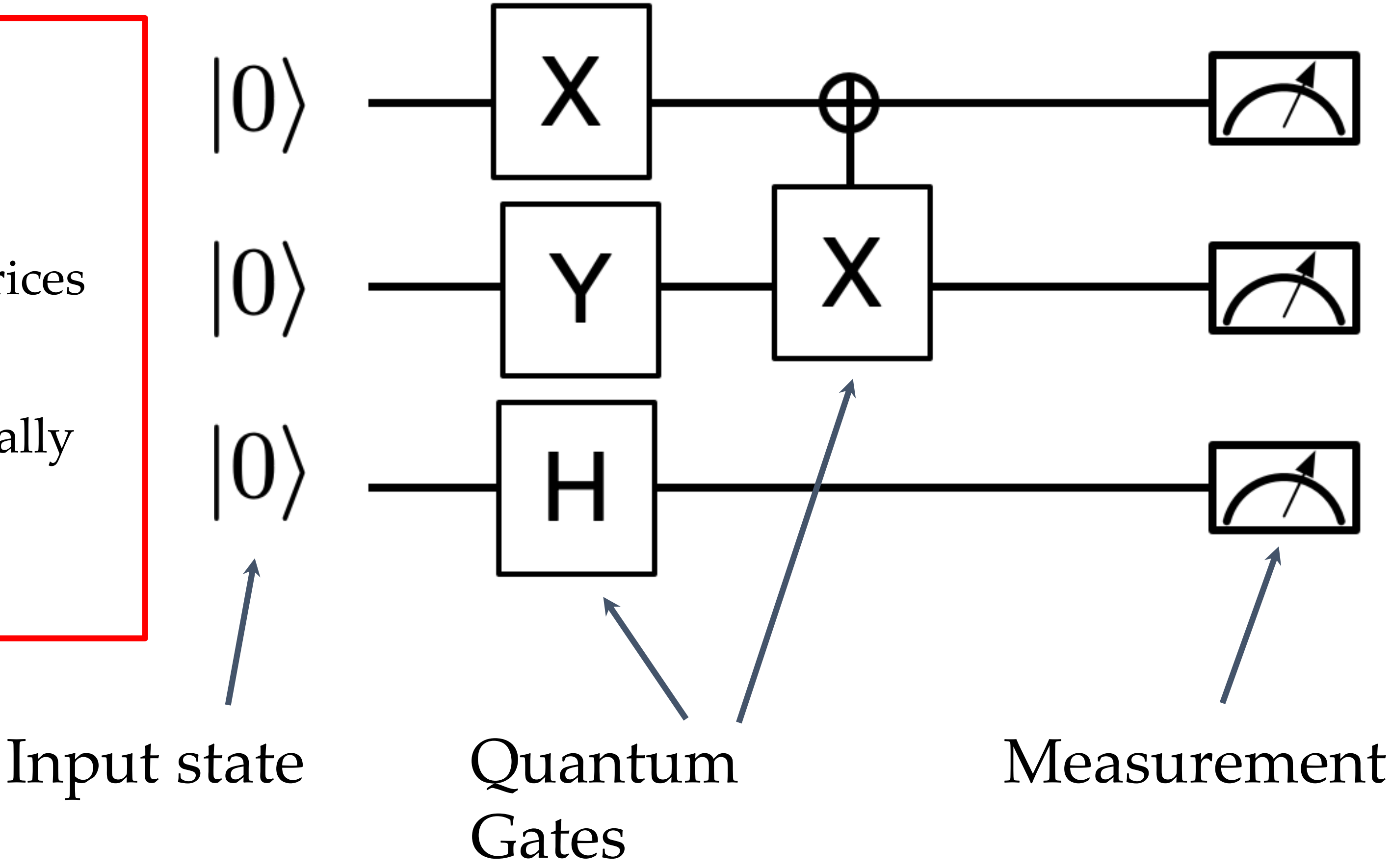
2. Download and unzip the repository from GitHub:
   - in a browser type: https://github.com/shalm/ictp2020-quantum-computing
   - Clone or download; then Download ZIP
   - extract the content of the package

3. Open odt file “Setting up & Running Qiskit” and follow the instructions to setup Qiskit framework

```
pip install qiskit-terra[visualization]
```

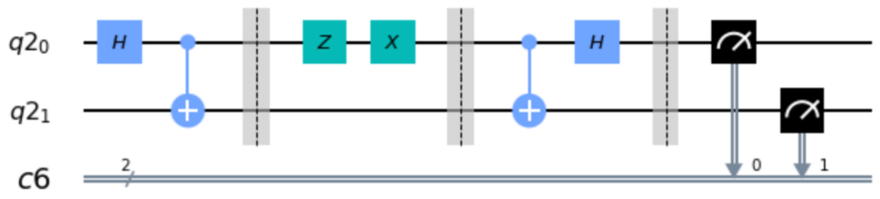
Quantum Logical Circuit

- **<u>INPUT:</u>** n qubits a 2^n dimensional (normalised) complex vector.
- **<u>OPERATIONS:</u>** Gates are unitary matrices in 2^n dimensions.
- **<u>OUTPUT:</u>** Measurements probabilistically collapse the state → $| \ |^2$ of amplitudes



All single-qubit operators are compiled down to physical gates: U_1, U_2, U_3

designed circuit

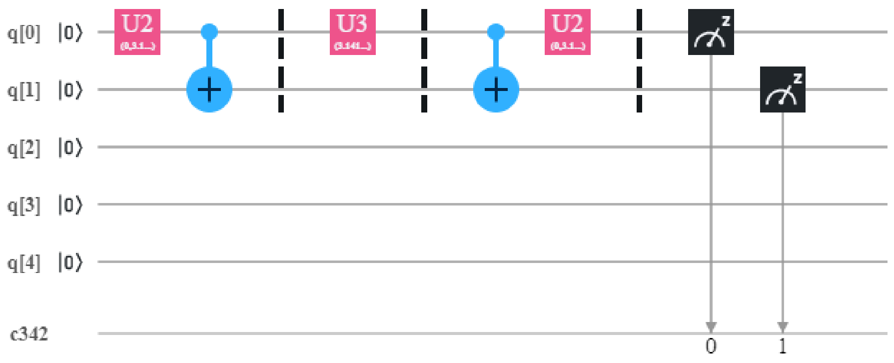


$$U_1(\lambda) = U_3(0,0,\lambda) = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\lambda} \end{pmatrix}$$

$$U_2(\phi,\lambda) = U_3(\pi/2,\phi,\lambda) = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & -e^{i\lambda} \\ e^{i\phi} & e^{i\lambda+i\phi} \end{pmatrix}$$

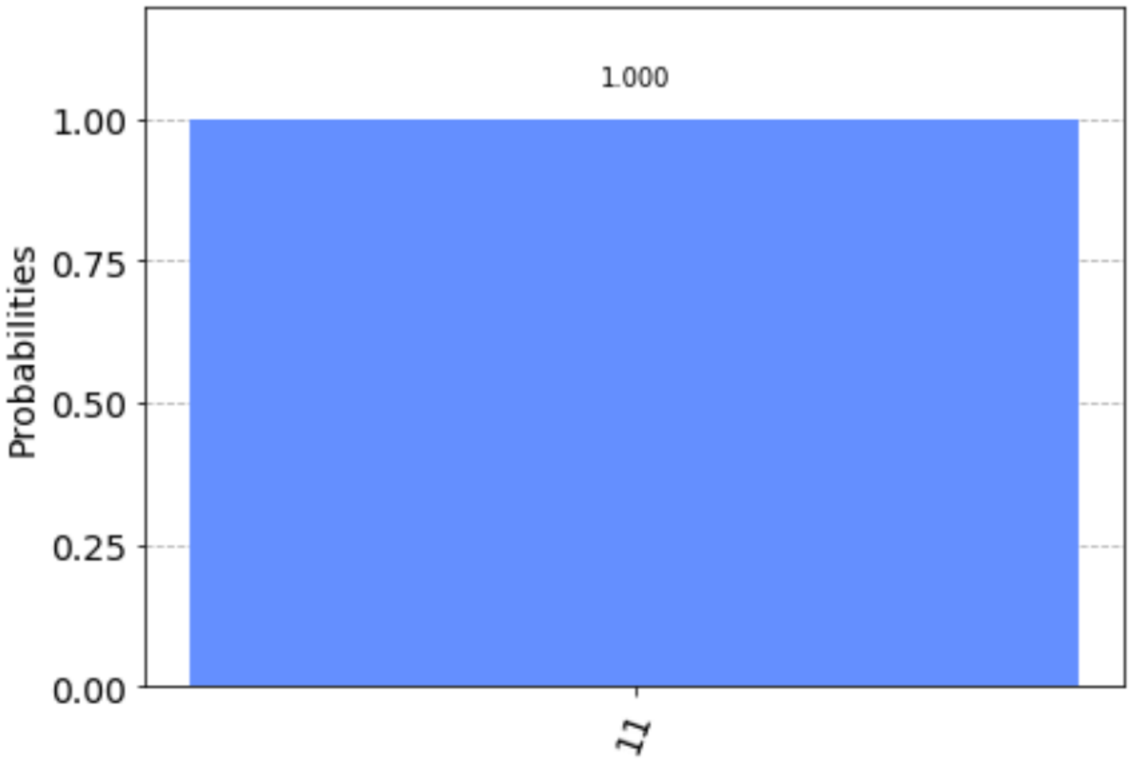
$$H = U_2(0,\pi)$$

compiled circuit

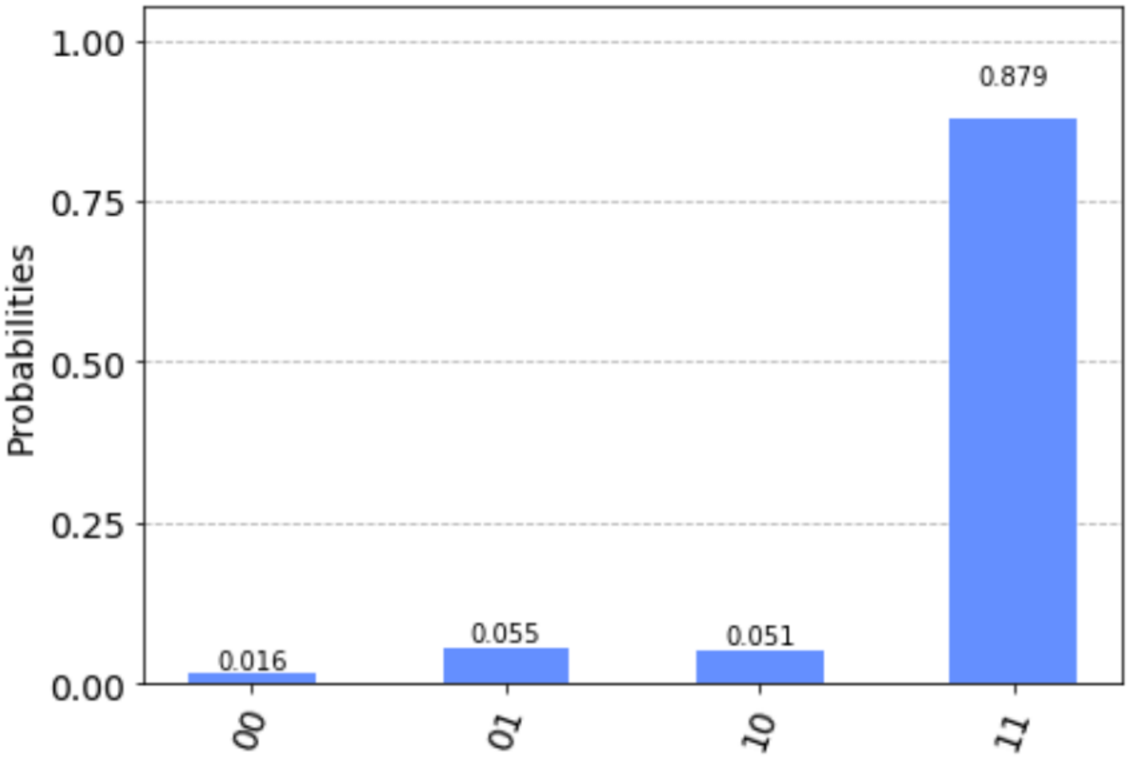


$$U_3(\theta,\phi,\lambda) = \begin{pmatrix} \cos(\theta/2) & -e^{i\lambda}\sin(\theta/2) \\ e^{i\phi}\sin(\theta/2) & e^{i\lambda+i\phi}\cos(\theta/2) \end{pmatrix}$$

Superdense coding: Results

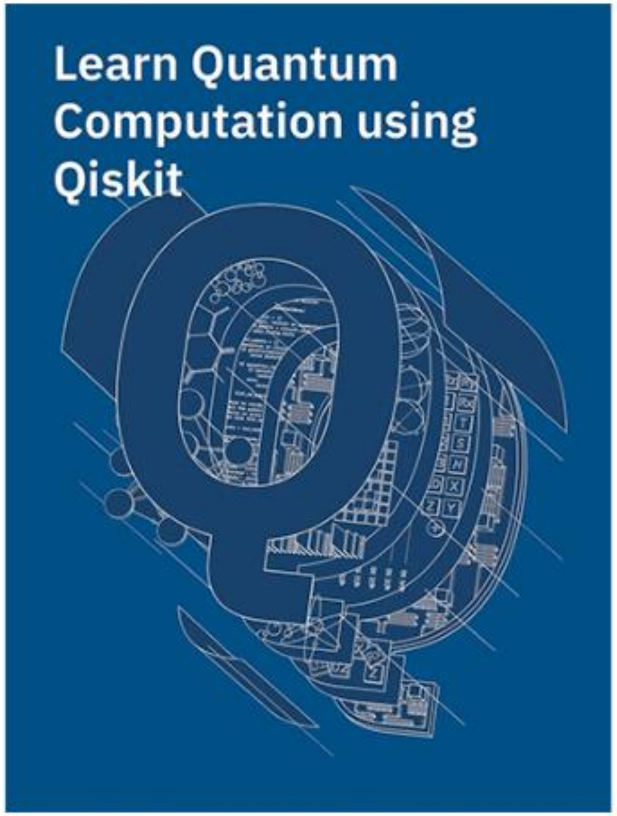


qasm. simulator



Ibmq_ourence: 256 shots

Further resources

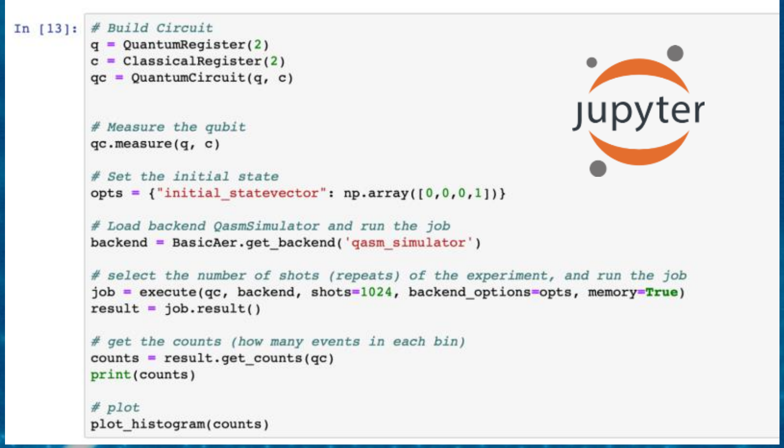


Search on GitHub

https://qiskit.org/textbook/preface.html

https://github.com/Qiskit/qiskit-community-tutorials

The Abdus Salam
International Centre
for Theoretical Physics



```
In [13]: # Build Circuit
         q = QuantumRegister(2)
         c = ClassicalRegister(2)
         qc = QuantumCircuit(q, c)

         # Measure the qubit
         qc.measure(q, c)

         # Set the initial state
         opts = {"initial_statevector": np.array([0,0,0,1])}

         # Load backend QasmSimulator and run the job
         backend = BasicAer.get_backend('qasm_simulator')

         # select the number of shots (repeats) of the experiment, and run the job
         job = execute(qc, backend, shots=1024, backend_options=opts, memory=True)
         result = job.result()

         # get the counts (how many events in each bin)
         counts = result.get_counts(qc)
         print(counts)

         # plot
         plot_histogram(counts)
```

Quantum Computing Lab

Instructor: Mikhail Shalaginov, MIT

Thank you!