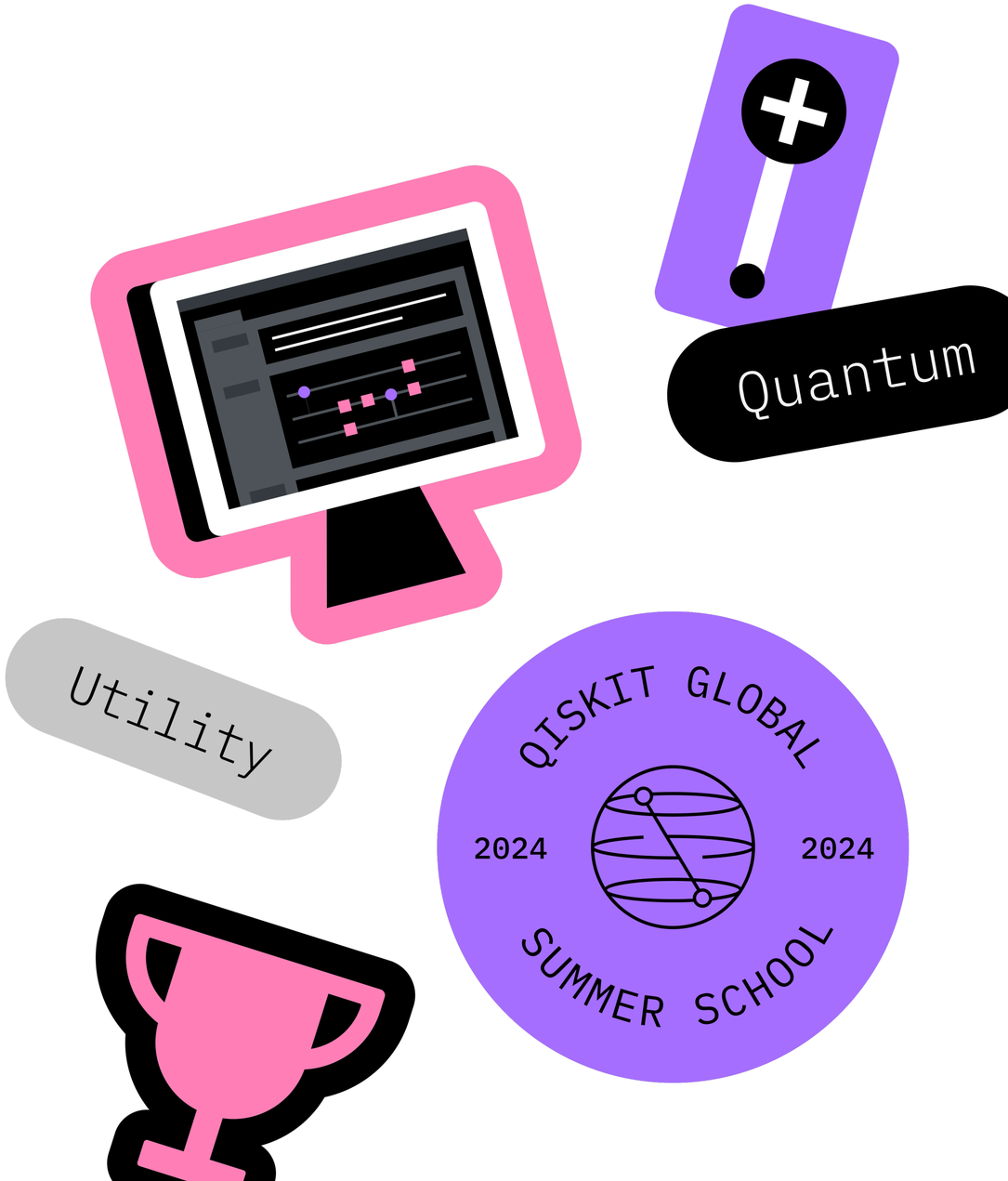
Qiskit Global Summer School 2024

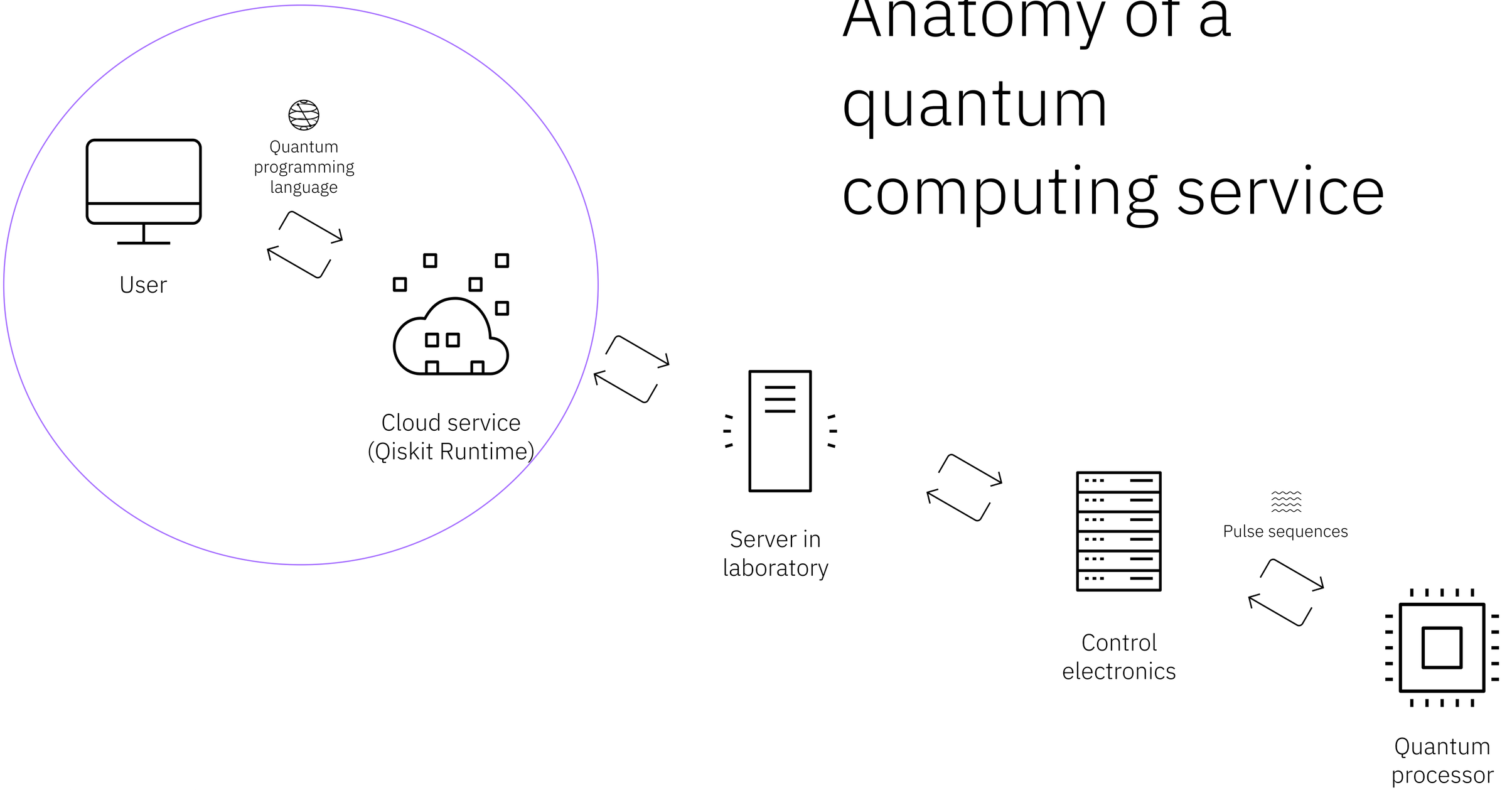
Lecture 1:
Intro to Qiskit 1.0



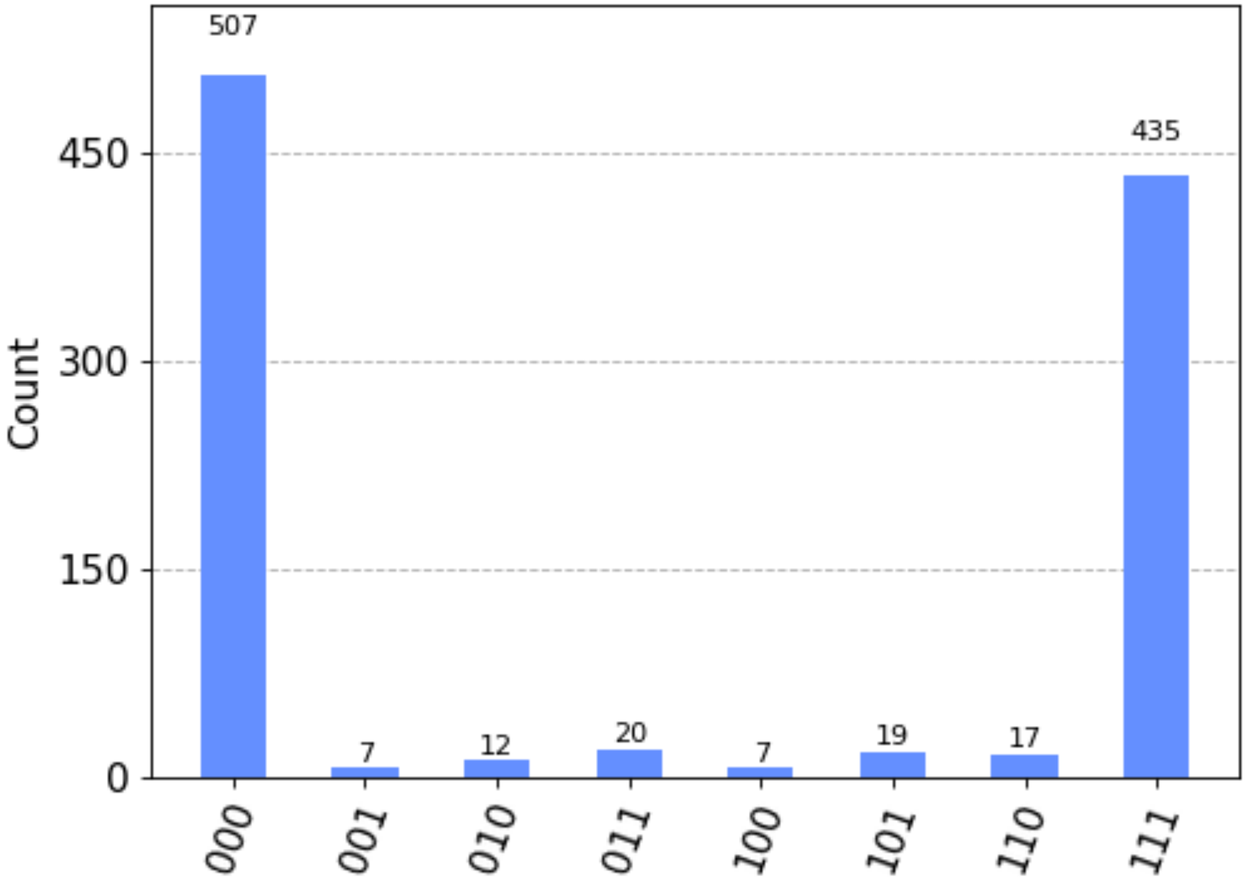
Abby Mitchell
Developer Advocacy Lead
IBM Quantum

IBM **Quantum**

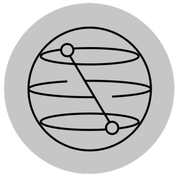
Anatomy of a quantum computing service



Example final result



What is Qiskit?



Qiskit SDK is an open-source SDK for working with quantum computers at the level of extended quantum circuits, operators, and primitives.

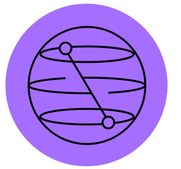
https://github.com/Qiskit/qiskit
https://docs.quantum-computing.ibm.com/

version 1.0 released Feb 2024 🎉

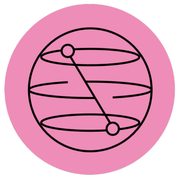
Qiskit Runtime is a cloud-based service for executing quantum computations on IBM Quantum hardware.

https://github.com/Qiskit/qiskit-ibm-runtime
https://docs.quantum-computing.ibm.com/

Runtime Primitives version 2 released March 2024 🎉



Qiskit Patterns is a framework for breaking down domain-specific problems into stages.



The **Qiskit Ecosystem** is a collection of software and projects that build on or extend Qiskit.

http://qiskit.github.io/ecosystem

Getting set up with Qiskit

```
pip install 'qiskit[visualization]'
pip install qiskit-ibm-runtime
```

To access IBM hardware, set up your credentials through one of:

- IBM Quantum Platform
- IBM Cloud

https://docs.quantum-computing.ibm.com/start

Qiskit Patterns

1. Map problem to quantum circuits and operators

2. Optimize circuits for target hardware

3. Execute on target hardware

4. Postprocess results

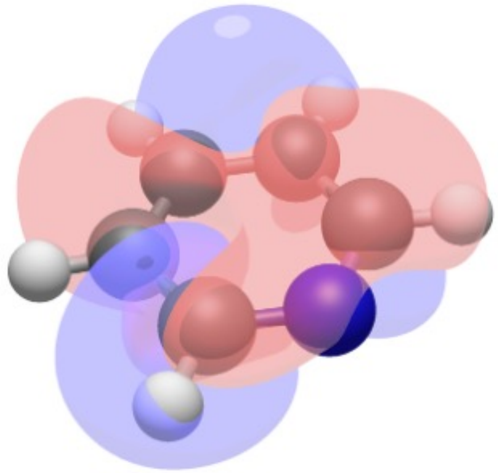
Qiskit Patterns

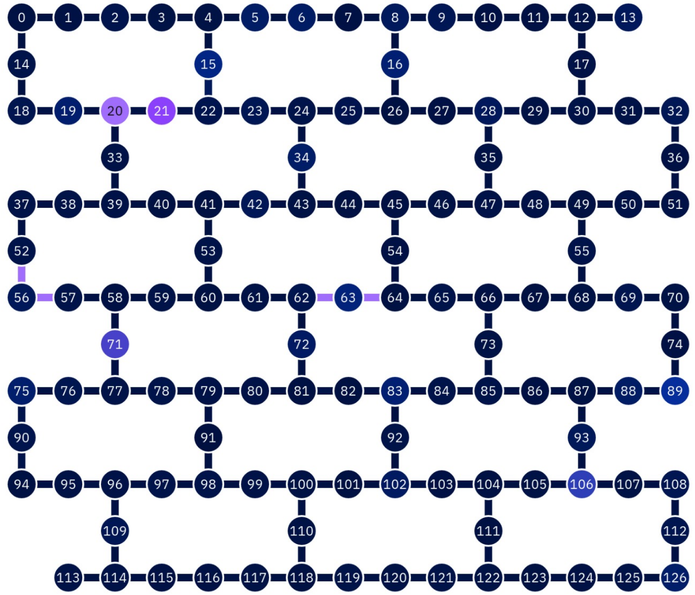
1. Map problem to quantum circuits and operators

2. Optimize circuits for target hardware

3. Execute on target hardware

4. Postprocess results





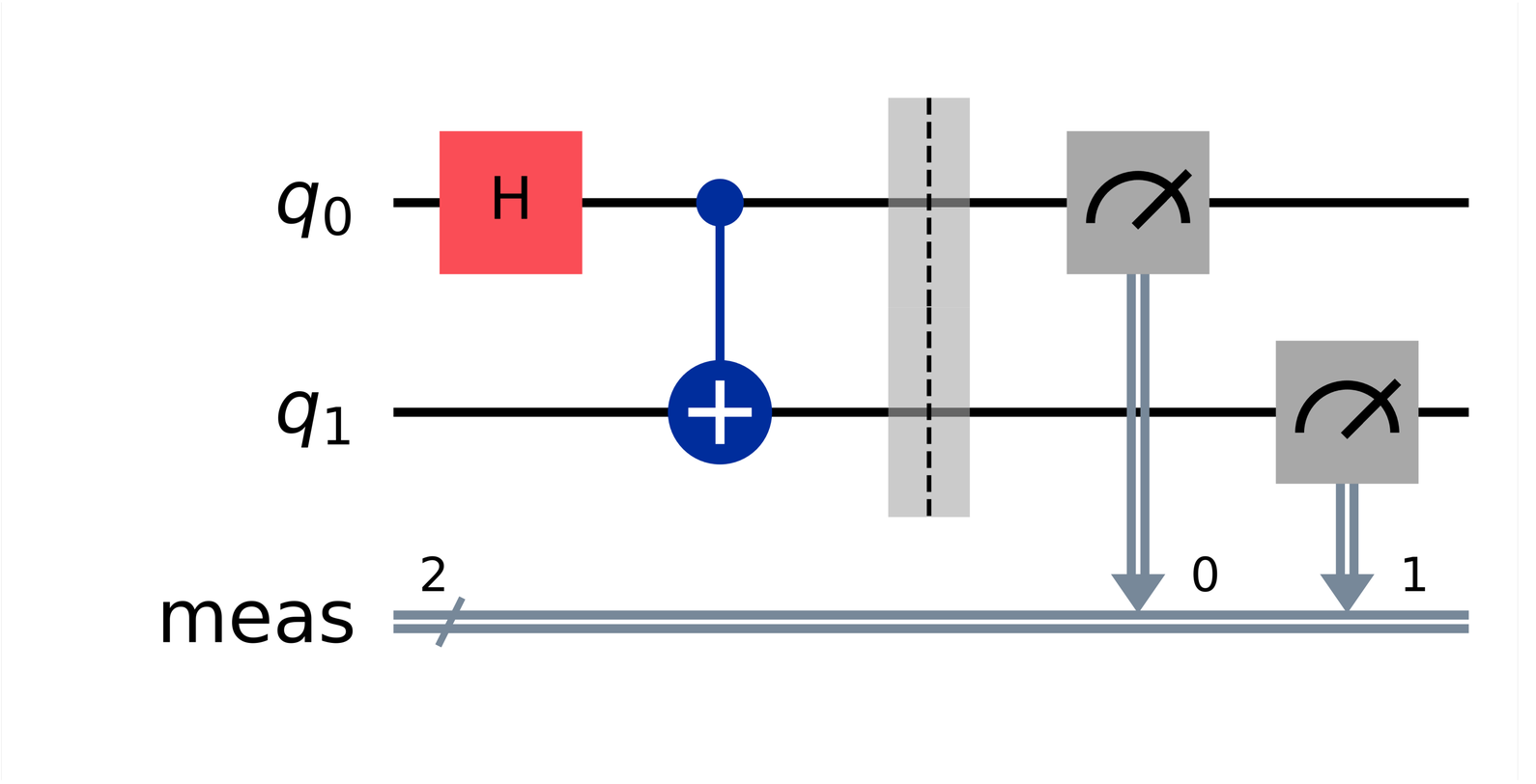
To map a problem:

- Start with numerical representation
- Map to quantum computing representation
- Choose quantum algorithm

Build a circuit with Qiskit SDK

The foundation of quantum programs are **quantum circuits.**

They consist of operations—including gates, measurement, and reset—that manipulate qubits.



Build a circuit with Qiskit SDK

To build a circuit:

- Initialize a register of qubits
- Add the qubits to a circuit
- Perform operations on those qubits

```
from qiskit import QuantumCircuit, QuantumRegister

qubits = QuantumRegister(2, name="q")
circuit = QuantumCircuit(qubits)

q0, q1 = qubits
circuit.h(q0)
circuit.cx(q0, q1)
circuit.measure_all()

circuit.draw("mpl")
```

Qiskit SDK includes a library of standard gates and circuits.

Standard gates

Hadamard, Pauli rotation gates, CNOT, Quantum Fourier Transform, etc.

Variational ansatzes

Parameterized quantum circuits for chemistry and combinatorial optimization, including hardware efficient ansatzes

Qiskit Patterns

1. Map problem to quantum circuits and operators

2. Optimize circuits for target hardware

3. Execute on target hardware

4. Postprocess results

Optimize for hardware – techniques

Required: *transpile* your abstract circuit into a circuit that can run on target hardware.

Optional: *verify* your circuit with simulation.

Optimize for hardware – techniques

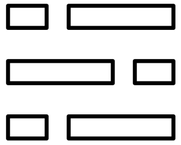
Required: *transpile* your abstract circuit into a circuit that can run on target hardware.

Optional: *verify* your circuit with simulation.

Real quantum devices are subject to constraints

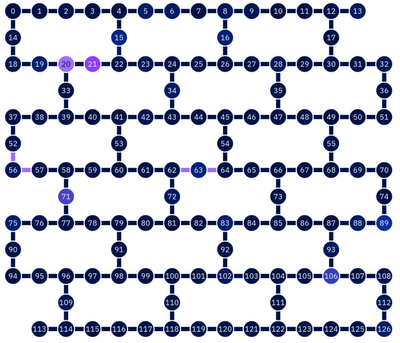
Basis gate set

Only a limited set of gates can be executed directly on the hardware. Other gates must be rewritten in terms of these basis gates.



Qubit connectivity

Only certain pairs of qubits can be directly interacted with each other.

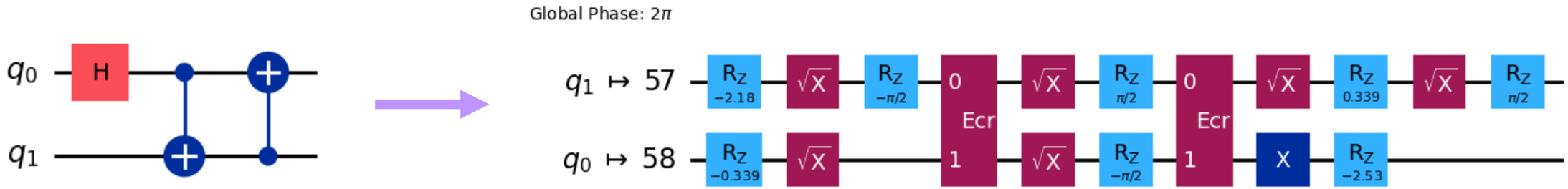


Errors

Each operation has a chance of error, so circuit optimizations can greatly affect performance.

Challenge: run abstract circuit on a specific quantum device.

Solution: *transpilation* — convert abstract circuit into an ISA (instruction set architecture) circuit.



Transpilation terms

Term	Definition	Orchestra analogy
pass	a standalone circuit or metadata transformation	an instrument
pass manager	a list of transpiler passes grouped into a logical unit	an instrument section
staged pass manager	a list of pass managers, with each one representing a discrete stage of a transpilation pipeline	the conductor

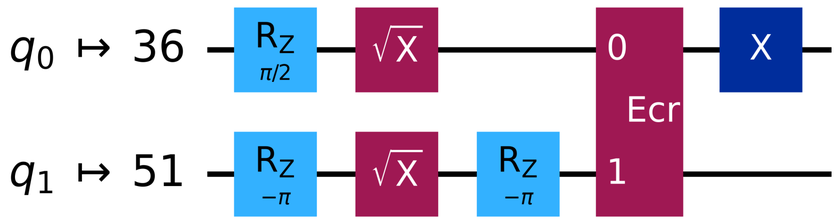
Transpile a circuit with Qiskit SDK

Steps:

- Choose which device backend you want to target
- Create a preset staged pass manager with your desired optimization level
- Run the staged pass manager on the circuit

```python
from qiskit import QuantumCircuit, QuantumRegister
from qiskit.transpiler.preset_passmanagers import generate_preset_pass_manager
from qiskit_ibm_runtime import QiskitRuntimeService

qubits = QuantumRegister(2, name="q")
circuit = QuantumCircuit(qubits)

q0, q1 = qubits
circuit.h(q0)
circuit.cx(q0, q1)

service = QiskitRuntimeService()
backend = service.backend("ibm_brisbane")
pass_manager = generate_preset_pass_manager(optimization_level=3, backend=backend)
isa_circuit = pass_manager.run(circuit)

isa_circuit.draw("mpl", idle_wires=False)
```

Global Phase: 3π/4



Transpiler stages

1. Initialization

The circuit is prepared for transpilation, e.g., multi-qubit gates are decomposed into two-qubit gates.

2. Layout

The abstract qubits of the circuit are mapped to physical qubits on the device.

3. Routing

Swap gates are inserted to enable interactions between qubits that are not physically connected.

4. Translation

The gates of the circuit are translated to the basis gate set of the device.

5. Optimization

The circuit is rewritten to minimize its depth (# of operations) to decrease the effect of errors.

6. Scheduling

Delay instructions are added to align the circuit with the hardware's timing.

Additional transpilation tools

- Qiskit transpiler service
- Qiskit Ecosystem, e.g., circuit-knitting-toolbox
- Write your own transpiler plugins

Optimize for hardware – techniques

Required: *transpile* your abstract circuit into a circuit that can run on target hardware.

Optional: *verify* your circuit with simulation.

Simulation tools

Qiskit SDK reference primitives

Exact simulation, but small circuits only and no noise simulation.

Qiskit Runtime local testing

Provides “fake” backends to model each quantum machine.

Qiskit Aer

Ecosystem project for simulation, including

- larger circuits
- stabilizer circuits
- noise models

```
# Run the sampler job locally using FakeManilaV2
fake_manila = FakeManilaV2()
pm = generate_preset_pass_manager(backend=fake_manila, optimization_level=1)
isa_qc = pm.run(qc)

# You can use a fixed seed to get fixed results.
options = {"simulator": {"seed_simulator": 42}}
sampler = Sampler(backend=fake_manila, options=options)

result = sampler.run([isa_qc]).result()
```

Problem: the runtime cost of simulating quantum circuits scales exponentially with the number of qubits.

~50+ qubits cannot be simulated.

Techniques for large circuits:

1. Test smaller versions of circuit
2. Modify circuit so that it becomes classically simulatable: stabilizer circuit aka Clifford circuit

Qiskit Patterns

1. Map problem to quantum circuits and operators

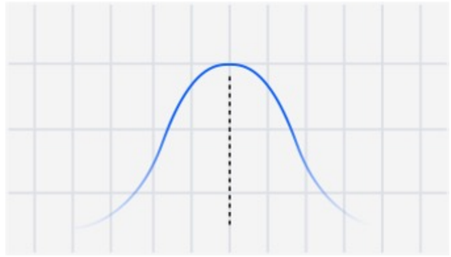
2. Optimize circuits for target hardware

3. Execute on target hardware

4. Postprocess results

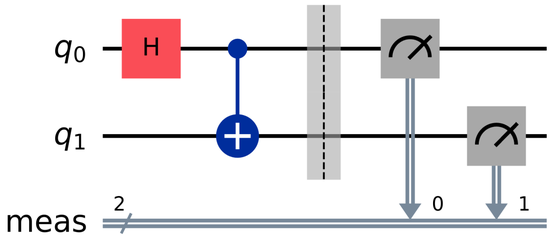
Primitives encapsulate the output of a quantum circuit

Sampler primitive

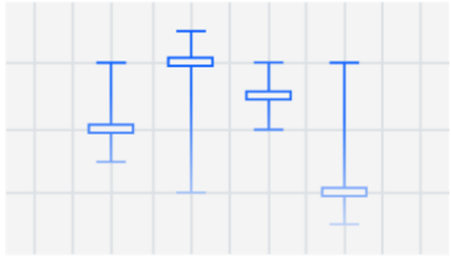


Output is mapping of bitstrings to counts, e.g., {'0': 12, '1': 9}

Circuit should include measurements.

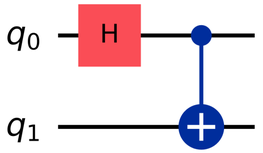


Estimator primitive



Output is the *expectation value* of an *observable*, e.g., the net spin of a system.

Circuit should not include measurements.



To run a circuit on quantum hardware:

1. Initialize the Qiskit Runtime service
2. Choose a hardware backend
3. Initialize a Qiskit Runtime primitive with your chosen backend
4. Invoke the primitive with your circuit

```python
import numpy as np
from qiskit.circuit.library import IQP
from qiskit.quantum_info import random_hermitian
from qiskit_ibm_runtime import QiskitRuntimeService, SamplerV2 as Sampler

service = QiskitRuntimeService()

backend = service.least_busy(operational=True, simulator=False, min_num_qubits=127)

n_qubits = 127

mat = np.real(random_hermitian(n_qubits, seed=1234))
circuit = IQP(mat)
circuit.measure_all()

pm = generate_preset_pass_manager(backend=backend, optimization_level=1)
isa_circuit = pm.run(circuit)

sampler = Sampler(backend)
job = sampler.run([isa_circuit])
result = job.result()
```

The input to the Estimator primitive is a list of *primitive unified blocs (PUBs).* Each PUB consists of:

- A single circuit without measurements
- One or more observables
- (Optional) One or more parameter values

```python
import numpy as np
from qiskit.circuit.library import IQP
from qiskit.transpiler.preset_passmanagers import generate_preset_pass_manager
from qiskit.quantum_info import SparsePauliOp, random_hermitian
from qiskit_ibm_runtime import EstimatorV2 as Estimator, QiskitRuntimeService

service = QiskitRuntimeService()
backend = service.least_busy(operational=True, simulator=False, min_num_qubits=127)
estimator = Estimator(backend)

n_qubits = 127

mat = np.real(random_hermitian(n_qubits, seed=1234))
circuit = IQP(mat)
observable = SparsePauliOp("Z" * n_qubits)

pm = generate_preset_pass_manager(backend=backend, optimization_level=1)
isa_circuit = pm.run(circuit)
isa_observable = observable.apply_layout(isa_circuit.layout)

job = estimator.run([(isa_circuit, isa_observable)])
result = job.result()

print(f" > Expectation value: {result[0].data.evs}")
print(f" > Metadata: {result[0].metadata}")
```

You can customize Qiskit Runtime behavior

Shots

The number of measurements.

More shots reduce statistical error but increase running time.

Error suppression

Use dynamical decoupling to reduce decoherence during execution.

Caveat: requires delays between gate executions, so it’s not always possible.

Error mitigation

Reduce the effects of device noise after execution.

- Twirled Readout Error eXtinction (TREX) measurement twirling
- Zero Noise Extrapolation (ZNE)

Downside: computational overhead

Execution modes

- Single job
- Batch: multiple concurrent jobs
- Session: iterative workload

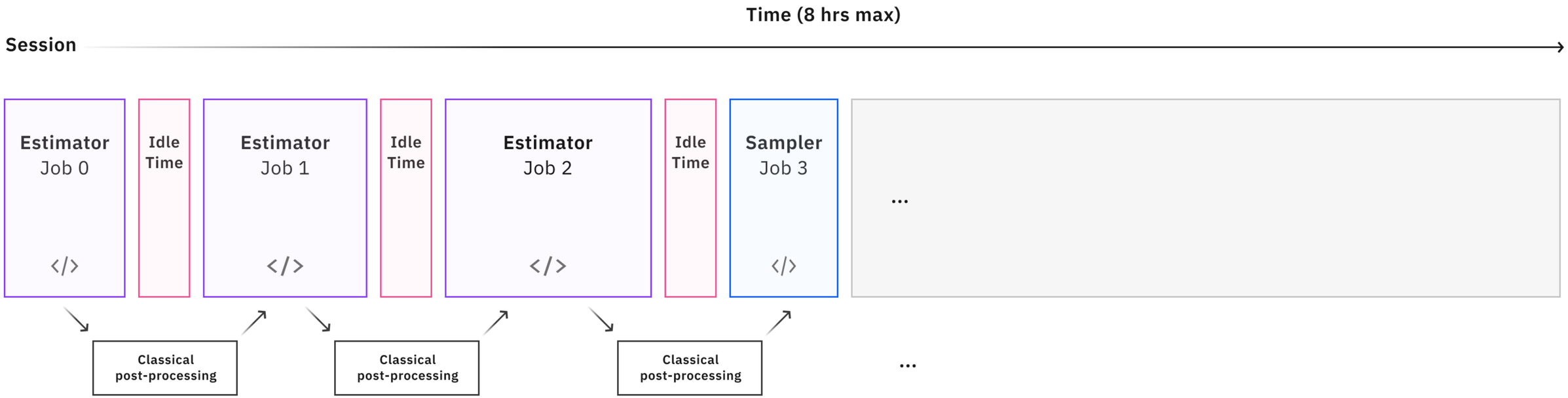
Benefits of batch mode versus a single job with multiple circuits:

- better concurrency of classical processing
- get individual results sooner

Batch mode

```python
from qiskit_ibm_runtime import SamplerV2 as Sampler, Batch

max_circuits = 100
all_partitioned_circuits = []
for i in range(0, len(circuits), max_circuits):
    all_partitioned_circuits.append(circuits[i : i + max_circuits])
jobs = []
start_idx = 0

with Batch(backend=backend):
    sampler = Sampler()
    for partitioned_circuits in all_partitioned_circuits:
        job = sampler.run(partitioned_circuits)
        jobs.append(job)
```

Sessions allow iterative workloads, like VQE.



Sessions

```python
from qiskit_ibm_runtime import EstimatorV2 as Estimator
from qiskit_ibm_runtime import QiskitRuntimeService, Session

# Initialize Qiskit Runtime service
service = QiskitRuntimeService()
backend = service.backend("ibm_brisbane")

with Session(backend=backend):
    estimator = Estimator()
    # invoke the Estimator as usual
```

Qiskit Patterns

1. Map problem to quantum circuits and operators

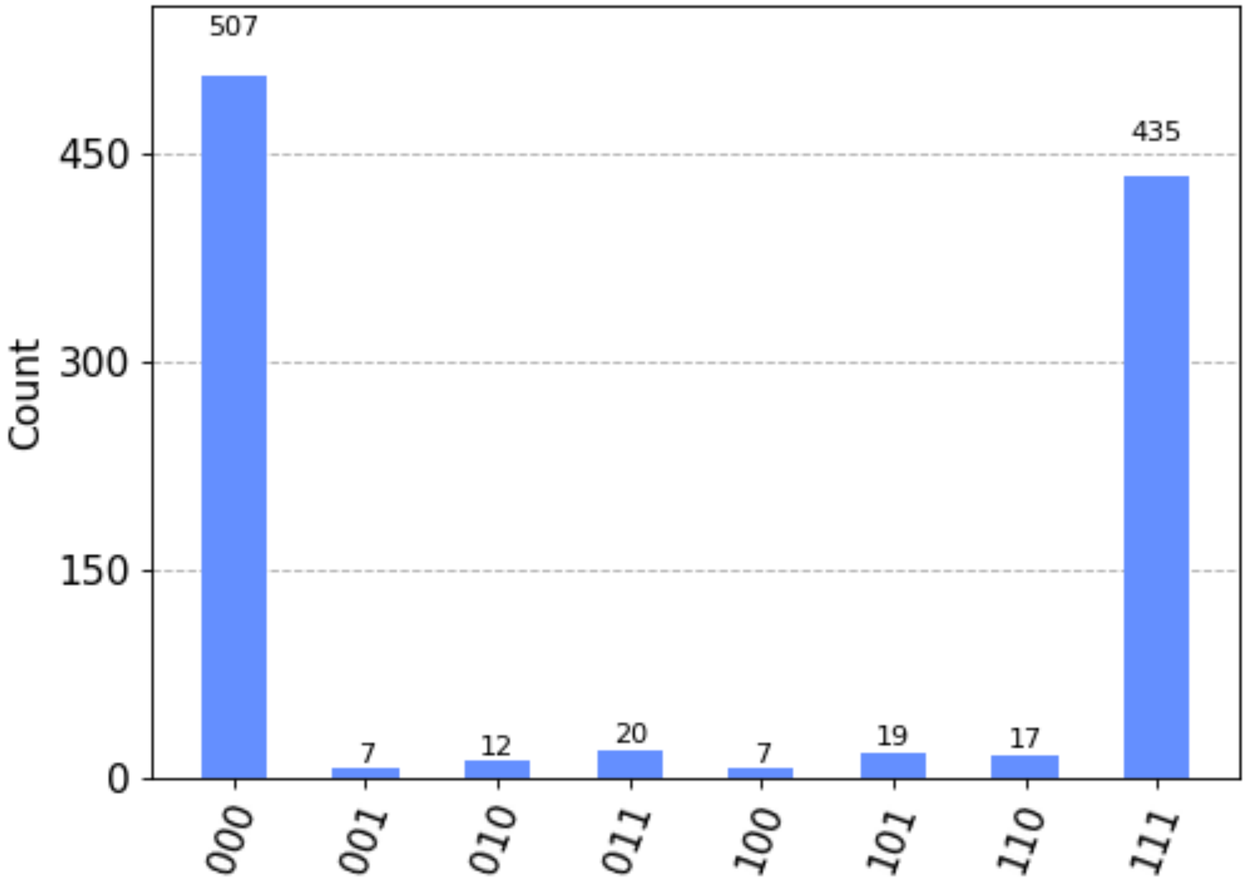
2. Optimize circuits for target hardware

3. Execute on target hardware

4. Postprocess results

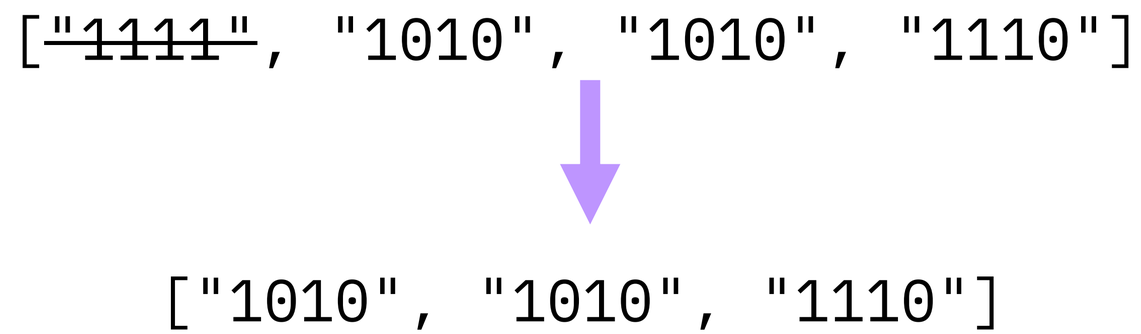
Postprocess technique: visualize results

With Sampler output, use `plot_histogram()` from `qiskit.visualizations`



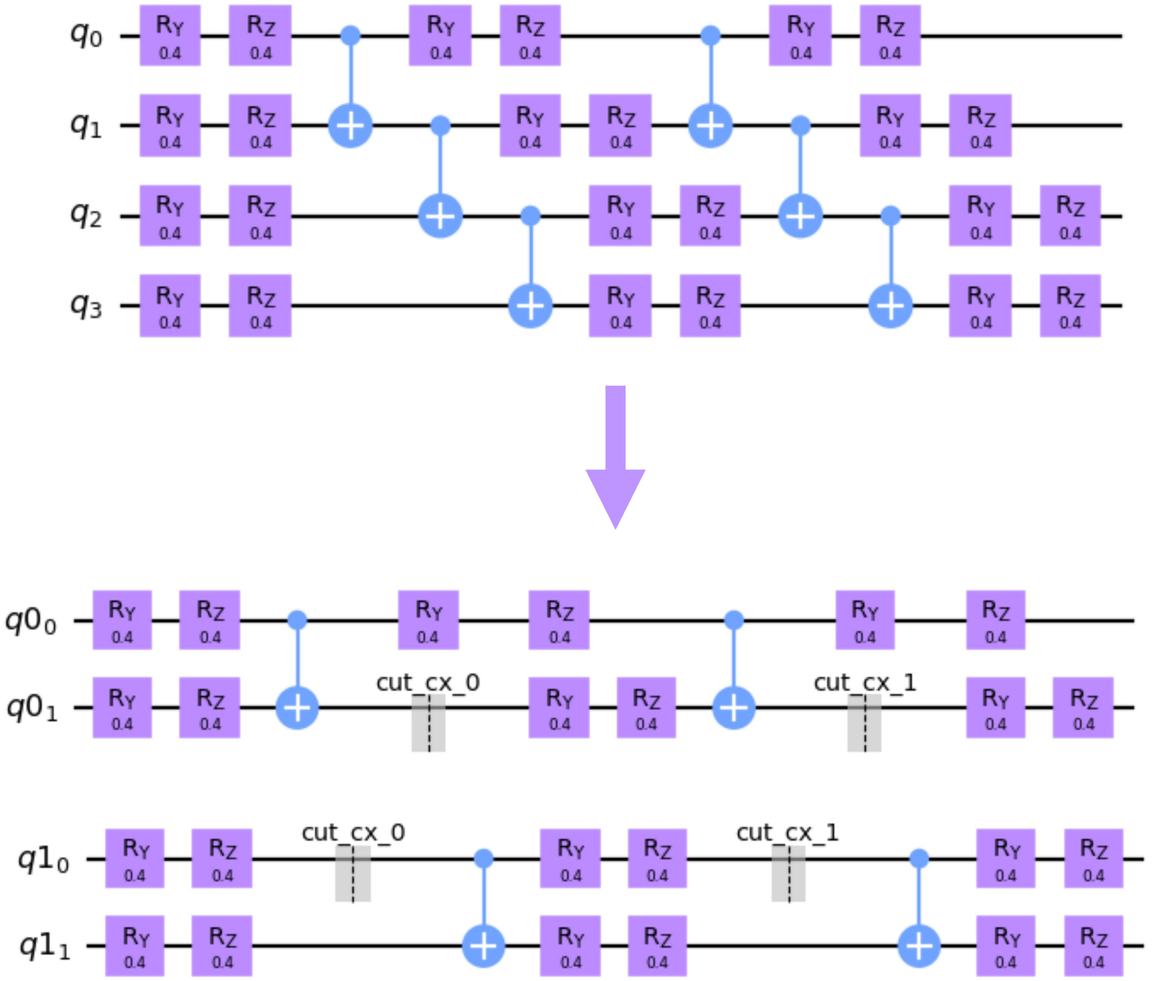
Postprocess technique: post-selection

Discard outputs known to be incorrect based on prior knowledge, e.g., if you know outputs must match a certain pattern



Postprocess technique: circuit knitting

1. During the optimize for hardware step, decompose the problem into smaller circuits, i.e., “circuit cutting”.
2. Execute smaller circuits.
3. “Knit” the results into a reconstruction of the original circuit’s outcome.



Qiskit Patterns

1. Map problem to quantum circuits and operators

2. Optimize circuits for target hardware

3. Execute on target hardware

4. Postprocess results

docs.quantum.ibm.com

Documentation for Qiskit and IBM Quantum services.

learning.quantum.ibm.com

Courses on quantum computing and how to use IBM Quantum services to solve real-world problems.

www.youtube.com/qiskit

Lectures, tips & tricks, tutorials, community updates, and exclusive Qiskit content!