Qiskit Quantum Lab: Hands-On Guide to Quantum Programming

Suman Kumar Roy · Follow

7 min read · Aug 14, 2023

Listen Share

Welcome to the forefront of computing's next frontier: quantum computing. In this realm, conventional rules no longer apply, and the potential for solving complex problems surpasses the limits of classical computing. At the heart of this quantum revolution lies Qiskit programming.

Quantum computing harnesses the principles of quantum mechanics to process information in ways that were once deemed impossible. Unlike classical bits, quantum bits or qubits can exist in multiple states simultaneously, a phenomenon known as superposition. Additionally, qubits can become entangled, sharing correlations that enable potent computations. Qiskit, developed by IBM, is an open-source platform that democratizes quantum programming. It empowers novices

and experts to engage with quantum concepts through a user-friendly Python interface. This guide will escort you into Qiskit programming, from fundamental principles to crafting quantum circuits and implementing advanced algorithms.

Processor

The different types of processors are named based on their specific technology traits, which are categorized by family and revision. The family name (such as Falcon) relates to the maximum circuits that can be constructed on the chip, which is influenced by the number of qubits and the connectivity graph. Revisions (such as r1) are design variations within a specific family, which can have a positive impact on performance but also come with certain tradeoffs. Segments of the chip are made up of sub-sections and are specific to a certain family. For example, segment H on a Falcon consists of seven qubits arranged as depicted in the illustration below. If implemented, segment H on a Hummingbird would likely look completely different.

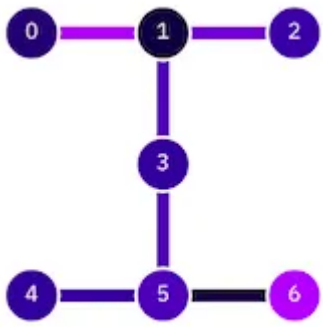


Illustration of segment H on a Falcon processor [Link]

Qiskit's processor is a physical manifestation of the abstract qubits that lie at the heart of quantum computing theory. These qubits, which exist in a superposition of states, allowing for parallel computation, are implemented using a variety of physical systems — superconducting circuits, trapped ions, and more. The processor is a piece of sophisticated hardware that leverages these quantum systems to perform calculations that classical computers can only dream of.

However, working with quantum processors comes with a unique set of challenges. Unlike classical bits, qubits are highly sensitive to their environment, leading to a phenomenon known as quantum noise. This noise can introduce errors and limit the reliability of computations. To combat this, Qiskit integrates error correction

techniques, including quantum error correction codes, to mitigate the impact of noise and enhance the stability of computations.

Ospery

IBM introduces the game-changing 'IBM Osprey,' a quantum processor boasting an unprecedented 433 qubits. This surpasses the 127 qubits of the previous IBM Eagle processor. Osprey's potential to handle intricate quantum computations far outstrips classical computers, exemplified by the astronomical number of classical bits required to represent its states — exceeding the total atoms in the known universe.

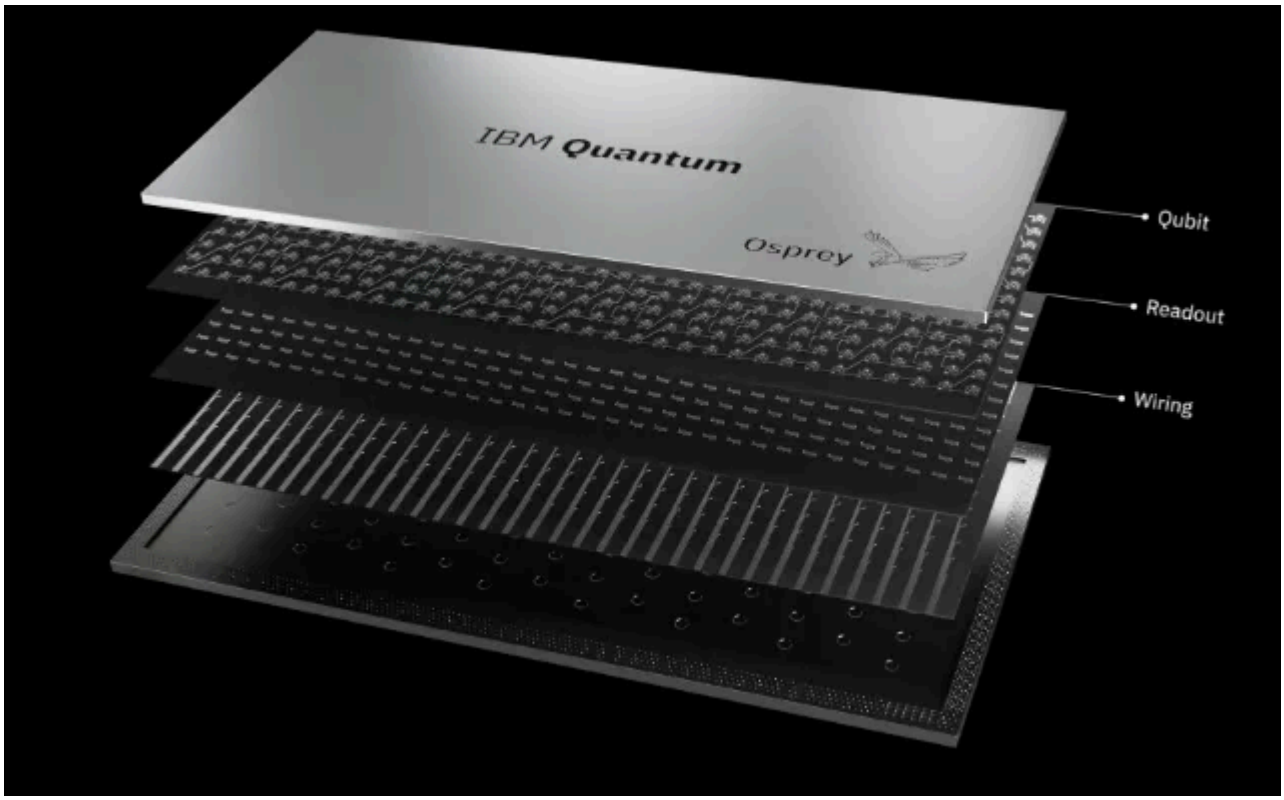
Quantum Software Advancements: IBM addresses quantum noise with enhanced quantum software. A beta update to Qiskit Runtime lets users trade computation speed for reduced errors through a simple API option. This simplification paves the way for easier quantum integration into workflows, accelerating the development of quantum applications.

IBM Quantum System Two: Scaling towards a goal of 4,000+ qubits by 2025, IBM reveals the modular IBM Quantum System Two. This flexible system combines multiple processors into a single unit with communication links. Expected by the end of 2023, it's a cornerstone of quantum-centric supercomputing, leveraging hybrid cloud middleware for seamless integration of quantum and classical workflows.

Quantum Security Advances: IBM's quantum safe technology takes center stage. With contributions to NIST's standardization goals, IBM safeguards systems against future quantum decryption threats. Collaborating with Vodafone, IBM explores applying quantum-safe cryptography to technology infrastructure.

Expanding Quantum Network: IBM Quantum Network expands with new partners. Bosch, a German conglomerate, joins to explore quantum use cases. Recent additions include Vodafone, Crédit Mutuel Alliance Fédérale, and uptownBasel, amplifying a community of 200+ organizations and 450,000 users with access to over 20 cloud-based quantum computers.

IBM's 433-qubits Osprey quantum processor more than triples the 127 qubits on the IBM Eagle processor unveiled in 2021 [Link]



Ospery chip [Link]

Qiskit Programming

Qiskit is a freely available software development kit (SDK) that enables users to work with quantum computers on the level of circuits, algorithms, and application modules. To use Qiskit, users typically follow four main steps.

- 1. Build: Create quantum circuits that represent the problem that is being considered.
- 2. Compile: Prepare the circuits for a specific quantum service, such as a quantum system or classical simulator.

- 3. Run: Execute the compiled circuits on the designated quantum service(s), which can be located either in the cloud or locally.
- 4. Analyze: Calculate summary statistics and visualize the experimental results.

Import Packages

The basic elements needed for your program are imported as follows

```
from qiskit import QuantumCircuit
from qiskit_aer import AerSimulator
from qiskit.visualization import plot_histogram
```

In more detail, the imports are *QuantumCircuit*, which can be thought of as the instructions of the quantum system. It holds all your quantum operations. *AerSimulator* is the Aer high-performance circuit simulator, and *plot_histogram* creates histograms.

Initialize Variables

Here, you are initializing with 2 qubits in the zero state; with 2 classical bits set to zero; and *circuit* is the quantum circuit.

```
circuit = QuantumCircuit(2, 2)
```

Add Gates

You can add gates (operations) to manipulate the registers of your circuit. The gates are added to the circuit one by one to form the Bell state below.

$$|\psi\rangle = (|00\rangle + |11\rangle)/\sqrt{2}.$$

Bell State

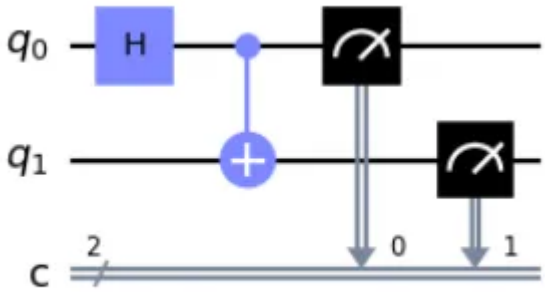
```
circuit.h(0)
circuit.cx(0, 1)
circuit.measure([0, 1], [0, 1])
```

Applied a Hadamard gate (H) on qubit 0, which puts it into a **superposition state**. Then, a controlled-Not operation (CNOT) on control qubit 0 and target qubit 1, puts the qubits in an **entangled state**. If you pass the entire quantum and classical registers to *measure*, the i-th qubit's measurement result will be stored in the ith classical bit.

Visualize the Circuit

You can use *qiskit.circuit.QuantumCircuit.draw()* to view the circuit that you have designed in the various forms used in many textbooks and research articles.

```
circuit.draw("mpl")
```



In this circuit, the qubits are ordered with qubit zero at the top and qubit one at the bottom. The circuit is read left-to-right, meaning that gates that are applied earlier in the circuit show up farther to the left.

Simulate the Experiment

Qiskit Aer is a top-performing simulator framework designed for quantum circuits.

included.

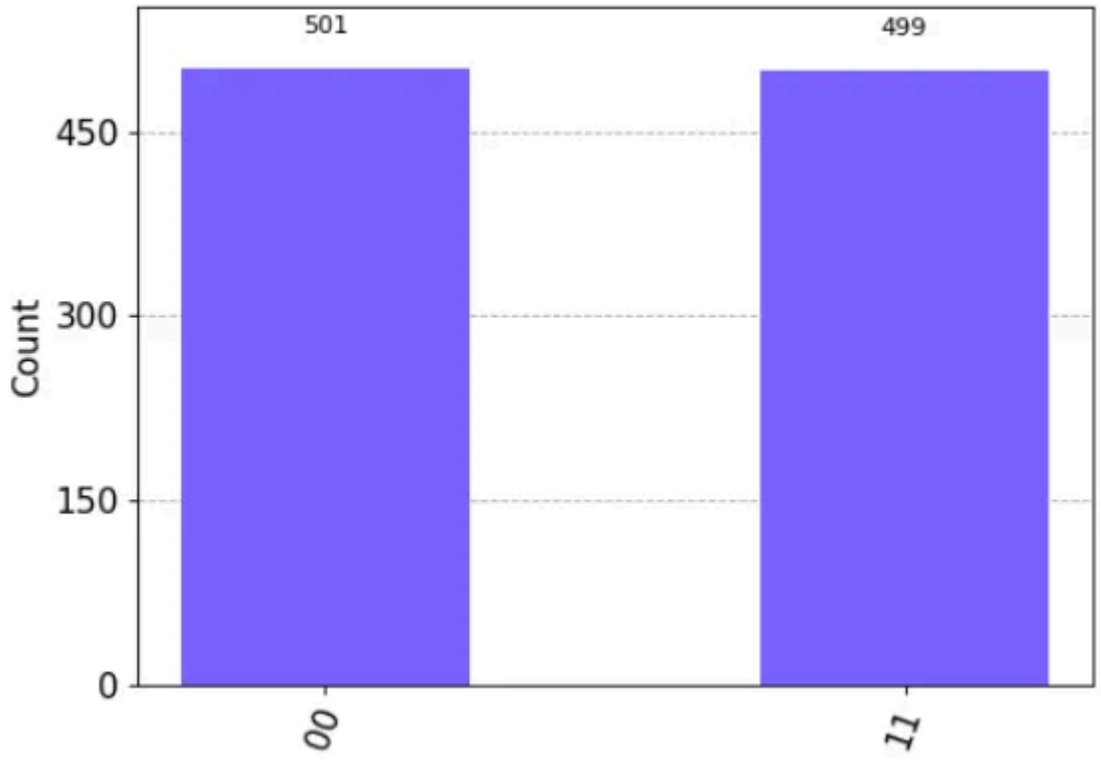
```
from qiskit import QuantumCircuit, transpile
from qiskit.providers.basicaer import QasmSimulatorPy

simulator = AerSimulator()
```

```
compiled_circuit = transpile(circuit, simulator)
job = simulator.run(compiled_circuit, shots=1000)
result = job.result()
counts = result.get_counts(circuit)
print("\nTotal count for 00 and 11 are:",counts)
```

In order to simulate this circuit, you can utilize the *AerSimulator*. Running this circuit will produce either the bit string 00 or 11, with the output bit string being 00 about half of the time. You can specify the number of times the circuit is run by using the shots argument in the *execute* method. By default, the number of shots for the simulation is set to 1024, but you can set it to 1000 if preferred. Once you have obtained a *result* object, you can retrieve the counts using the *get_counts(circuit)* method. This will provide you with the collective outcomes of the experiment that you ran.

Visualize the Results

Qiskit provides many visualizations, including the function *plot_histogram*, to view your results.



The observed probabilities *Pr(00)* and *Pr(11)* are computed by taking the respective counts and dividing them by the total number of shots.

Our voyage through Qiskit programming has illuminated the awe-inspiring universe of quantum computing. Qubits, gates, and circuits have transcended theory, becoming tools for innovation. Qiskit's Python interface facilitated our quantum odyssey, enabling exploration, simulation, and real-world execution on quantum processors.

From foundational concepts to processors boasting hundreds of qubits, our journey has showcased quantum's ascent. Qiskit's evolution parallels this progress, promising ever more robust tools and insights for harnessing quantum's potential.

Farewell to this guide, but not to the quantum revolution. With Qiskit, you stand ready to embrace the quantum future and leave an indelible mark on the ever-evolving canvas of quantum computing.

Refernces:

IBM Unveils 400 Qubit-Plus Quantum Processor and Next-Generation IBM Quantum System Two

IBM kicked off the IBM Quantum Summit 2022, announcing new breakthrough advancements in quantum hardware and software...

newsroom.ibm.com

Introduction to Qiskit - Qiskit 0.44.0 documentation

When using Qiskit a user workflow nominally consists of following four high-level steps: Build: Design a quantum...

qiskit.org

#Quantum30 #Day_14 QuantumComputingIndia

Quantum Computing Qiskit