Debugging quantum logic and circuit defects

Locate a wrong gate, phase or control by reducing a circuit to an explainable failing case.

Undergraduate 22 min lesson Dr. Charith Jayasekara

By the end, you'll be able to

- Use a minimal phase-sensitive failure.
- Test every CNOT basis mapping.
- Separate display conventions from gate semantics.

Begin with a clear question

Debugging becomes easier when you can point to the first operation where reality differs from the intended state. Start with the smallest failing input and an exact local reference. Then inspect amplitudes after each coherent step, followed by measurement mapping and result processing.

Build the reasoning

A missing Z in HZH changes the operation from X to identity. On zero, the intended output is one and the defective output is zero. In other bugs, only relative phase differs until a final basis change. Reversing a CX control and target is a separate fault; prepare basis inputs where exactly one wire is one so the two directions behave differently.



Follow time from left to right. Qiskit statevector labels put q0 on the right; diagrams put q0 on the top wire.

Worked example

The code first confirms the phase-sensitive HZH identity on zero. It then checks all four CNOT basis mappings under native Qiskit labels q1q0. The mapping is specified independently instead of reading an expected result from the same circuit builder. These cases isolate gate logic before any noisy samples enter the test.

$$HZH = X, \qquad \mathrm{CX}_{0\to 1}|q_1 q_0\rangle = |q_1 \oplus q_0,\ q_0\rangle.$$

```python
from qiskit import QuantumCircuit
from qiskit.quantum_info import Statevector

circuit = QuantumCircuit(1)
circuit.h(0); circuit.z(0); circuit.h(0)
assert Statevector.from_instruction(circuit).equiv(Statevector.from_label("1"))
cx = QuantumCircuit(2)
cx.cx(0, 1)
expected = {"00": "00", "01": "11", "10": "10", "11": "01"}
for before, after in expected.items():
    assert Statevector.from_label(before).evolve(cx).equiv(Statevector.from_label(after))
```

Practical lab

If these exact checks pass but measured data disagree, inspect classical-register destinations, result keys and compilation layout. If only hardware differs, review the target and noise assumptions before changing correct algorithm logic. Keep the failing input as a regression test. A diagram, native bitstring and q0-first display should all describe the same prepared state after their stated conversions.

Run the corresponding notebook from the software practice collection:

- Download Quantum software tests that catch plausible wrong programs

Keep a short execution record: predicted result, observed result, environment, inputs and the check that would reject a plausible defect. Explain a failure before changing the expected value.

Chapter summary

- Reduce a defect to a small deterministic case.
- Specify expected mappings independently.
- Check measurement and layout separately from coherent gate logic.

Exercises

1. Apply CX(1,0) to native label 01. Compare with CX(0,1).
2. Change one part of the worked program to create a plausible fault. State which assertion should fail and why.

> Worked answers
>
> 1. CX(1,0) leaves 01 unchanged because q1 is zero. CX(0,1) produces 11 because q0 is one.
> 2. A useful answer identifies a specific changed gate, parameter, mapping or model and links it to an independent expected behaviour. Keep the intended specification fixed while changing the implementation; save the failing case as a regression test.

Check your understanding

PAUSE & THINK

Which input distinguishes CX(0,1) from CX(1,0) in native Qiskit labels?

- [ ] 01
- [ ] 00 alone
- [ ] An empty circuit description

Answer: 01. Exactly one control candidate is one, so the two directions differ.

Sources

IBM's bit-ordering guide, Statevector API and Operator API describe the SDK conventions and equivalence tools. The notebook bundle records its pinned environment and local execution evidence.