

Quantum Software Testing

Concepts, Methods, and Challenges

Qichen Wang

*Graduate School and Faculty of Information Science and Electrical Engineering
Kyushu University*

September 28, 2026

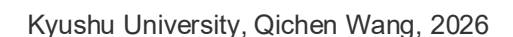


KYUSHU UNIVERSITY

Outline

- Background
- Motivation
- Related work
- Our work
- Challenges

Quantum software spans multiple abstraction levels, each exposing different inputs, outputs, and behavioral semantics.



Background: Program Execution Workflow

QISKIT CONSTRUCTION

Reusable Quantum Components

s1, s2: measurement-free circuits

```
circuit = s1.compose(s2)
```

Hybrid Wrapper

```
def sample(shots):
```

```
    qc = circuit.measure_all(inplace=False)
```

```
    job = sampler.run([qc], shots=shots)
```

```
    data = job.result()[0].data.meas
```

```
    return data.get_counts()
```

QUANTUM EXECUTION

One Composed Circuit

$\rho_0 \rightarrow \mathbf{S1} \rightarrow \rho_1 \rightarrow \mathbf{S2} \rightarrow \rho_2$

State passing

Measurement (Observation)

Quantum state $\rightarrow$ classical samples

Classical Interface

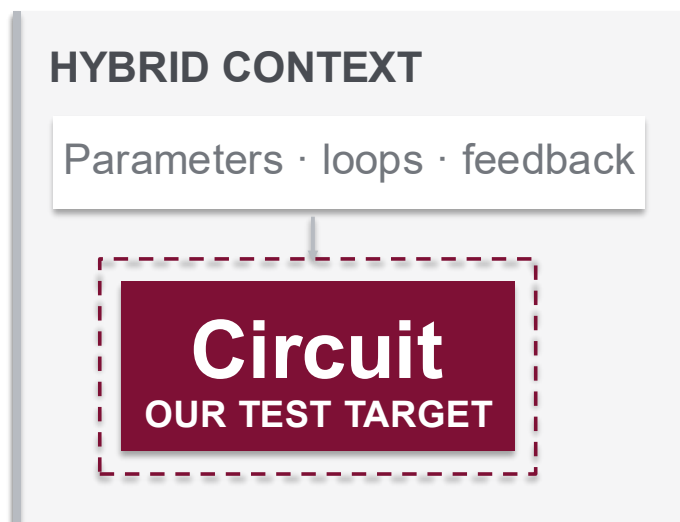
Samples $\rightarrow$ counts $\rightarrow$ **hybrid return**
(Dynamic Circuits)

Composition encodes **state passing**; terminal measurement yields **classical data**.

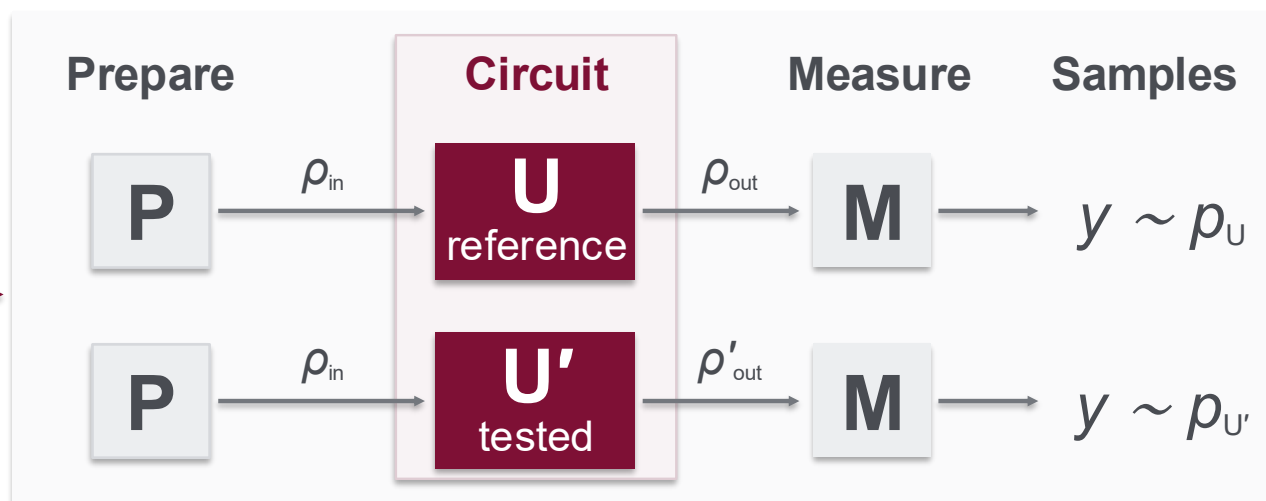
Background: Isolating Circuit Behavior for Testing

Fix the classical context; test the pure quantum part.

WHY ISOLATE?



BLACK-BOX COMPARISON



Same P and M. Compare repeated samples.

Isolate circuit behavior first.

FIND A DISTINGUISHING TEST

Choose (P, M) such that $p_U^{P,M} \neq p_{U'}^{P,M}$

Inputs must trigger differences; **measurements** must make them observable.

Motivation: Trigger, Error Propagation and Observation

Missing Z gate: $\mathbf{U} = \mathbf{Z}$ (reference) vs. $\mathbf{U}' = \mathbf{I}$ (faulty)

1 Not triggered

Input $|0\rangle$

$$\overset{\mathbf{U}}{|0\rangle} = \overset{\mathbf{U}'}{|0\rangle}$$

Z-basis measurement

Outcome order: 0, 1

$$(1, 0) = (1, 0)$$

Same distribution

2 Triggered, but hidden

Input $|+\rangle$

$$\overset{\mathbf{U}}{|-\rangle} \neq \overset{\mathbf{U}'}{|+\rangle}$$

Z-basis measurement

Outcome order: 0, 1

$$(\frac{1}{2}, \frac{1}{2}) = (\frac{1}{2}, \frac{1}{2})$$

Same distribution

3 Difference exposed

Input $|+\rangle$

$$\overset{\mathbf{U}}{|-\rangle} \neq \overset{\mathbf{U}'}{|+\rangle}$$

X-basis measurement

Outcome order: +, -

$$(0, 1) \neq (1, 0)$$

Different distributions

Change input only

Then change measurement

Joint probe design: input preparation $\mathbf{P}$ + measurement $\mathbf{M}$.

Related Work: From Inputs to Observations

Classical SE transfer

QuSBT (ICSE 2022) [1]

QuCAT (ASE 2023) [2]

Combinatorial coverage
Test-suite optimization

Bitstrings → **test suite**

LIMITATION

Only computational-basis
inputs are explored.

Quantum-feature inputs

QuraTest (ASE 2023) [3]

Ye et al.

Quantify magnitude, phase,
and entanglement

UCNOT / IQFT Template

Diversify inputs to trigger faults

LIMITATION

Input diversity does not
ensure visible differences.

Observations as tests

QOPS (ASE 2024) [4]

SB-QOPS (2026 preprint) [5]

Linear combinations within
one commuting Pauli family F

$$O = \sum_i c_i P_i$$

Expected vs. observed $\langle O \rangle$

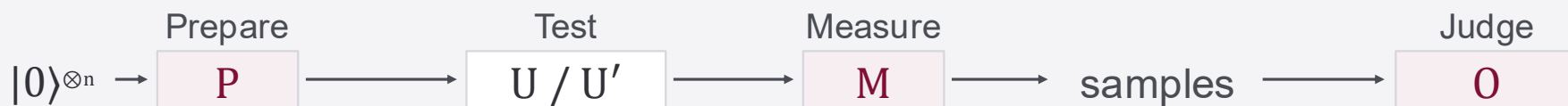
LIMITATION

Needs family specifications.
Does not search inputs.

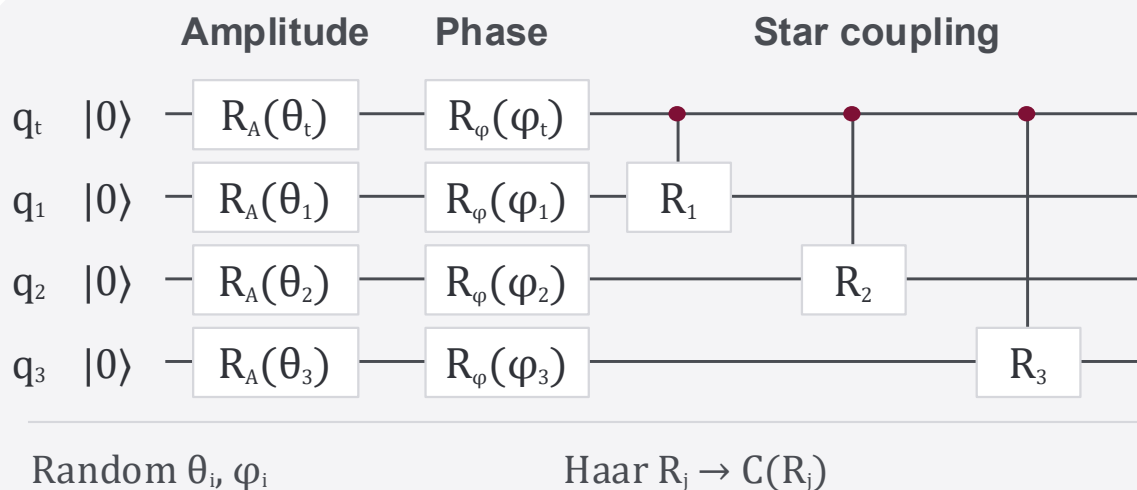
Our Approach: Joint Probes and Input Design

JOINT PROBE

$$\pi = (P, M, O)$$



P | Trigger differences



M | Expose differences

All-Z

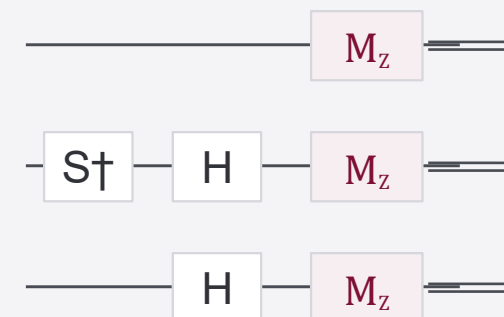
Basis probabilities

All-Y

Phase-sensitive

Hadamard + Z

All-X: interference



Our QOPS analysis: Z + Y sufficed.

For the analyzed faults and input ensemble.

Selected inputs + measurements

Targeted testing

All Pauli observables, fixed input

State tomography

+ Inputs spanning operator space

Process tomography

Select incomplete information that exposes differences—at low cost.

Our Approach: SWAP-Insensitive Semantic Probes

Make both ends invariant to permitted boundary SWAPs.

1 Input invariance

$$\Pi_{\text{in}} \rho_P \Pi_{\text{in}}^\dagger = \rho_P$$

$$\text{SWAP } |00\rangle = |00\rangle$$

2 Observation invariance

$$\Pi_{\text{out}}^\dagger M_k \Pi_{\text{out}} = M_k$$

$$\text{Count 1s: } 00 \rightarrow 0 \quad \{01, 10\} \rightarrow 1 \quad 11 \rightarrow 2$$

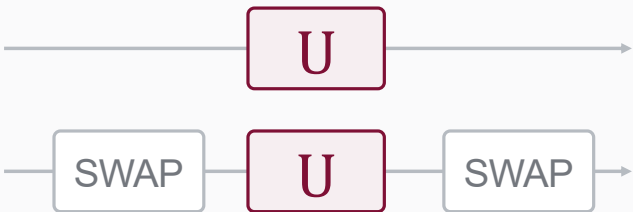
ILLUSTRATIVE CONSTRUCTION

$$U = H \otimes I$$

$$U' = I \otimes I$$

$$p(w), \quad w = 0, 1, 2$$

Implementation



Output from $|00\rangle$

$$(|00\rangle + |10\rangle)/\sqrt{2}$$

$$(|00\rangle + |01\rangle)/\sqrt{2}$$



$$(1/2, 1/2, 0)$$

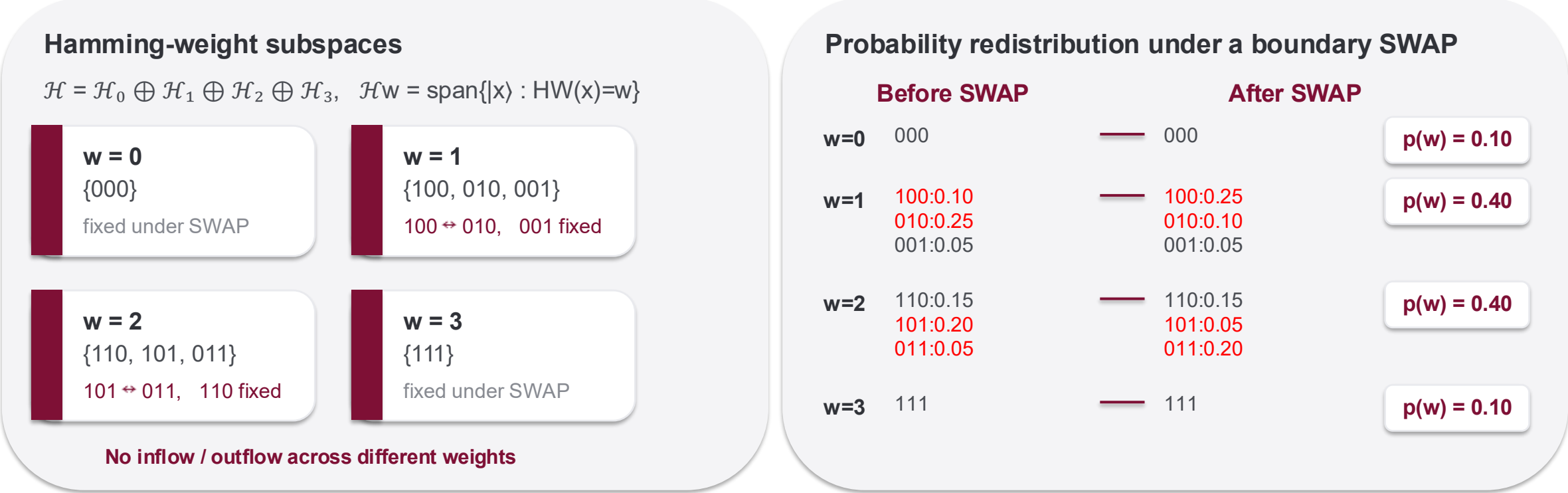
$$(1/2, 1/2, 0)$$

Missing H: $p(w) = (1, 0, 0) \neq (1/2, 1/2, 0)$

Filter order-only differences—not a full-equivalence test.

Our Approach: Why the Semantic Probe Sees Through SWAPs

A boundary SWAP preserves Hamming weight: probability may move within each weight subspace, but not across different weights.



Semantic probe observation: $p(w) = \sum_{\text{HW}(x)=w} p(x)$. It ignores order-only redistribution inside each $\mathcal{H}_w$, but still detects faults that move probability across different weight subspaces.

Open Challenges

Current challenges span both circuit-level fault detection and testing at higher software abstraction levels.

1 Circuit-level challenges

- **Triggering at a distance**
How can inputs effectively activate a fault when the fault location is far from the prepared state?
- **Propagation to observation**
Once a fault is triggered, how can its effect survive and propagate to the measurement position?
- **Benchmark obsolescence**
Mutation-based benchmarks are becoming less representative; they need to be updated.
- **Richer semantic probes**
Can we design more probes that encode useful semantics beyond plain equivalence?

2 Higher-level challenges

- **Lowering across abstractions**
How should testing tasks at subroutine, hybrid-program, or API levels be reduced to circuit-level evidence?
- **Lack of ground truth**
Circuit-level testing often lacks reliable ground truth or high-quality expected outputs.
- **Program specification (PS)**
How can we express and obtain usable program specifications, and check them efficiently?
- **Beyond black-box equivalence**
How can black-box equivalence testing be extended to specification-based program testing?

Future progress requires better triggering/observability mechanisms, updated benchmarks, richer semantic probes, and a path from circuit testing toward specification-aware higher-level testing.



Thank You

Q&A