Simulation-Based Fault Detection in Quantum Circuits via qATG: Modeling, Detection, and Experimental Analysis

Lai Yu-Hong
Department of Electrical Engineering
National Taiwan University
B10901125
Email: b10901125@ntu.edu.tw

***Abstract*—We present a comprehensive framework for fault detection in quantum circuits based on quantum automatic test generation (qATG). Our work focuses on simulating faults, generating fault-specific test circuits, and statistically evaluating detection performance using both Chi-square and KL-divergence-based bootstrap techniques. This report outlines the problem, implementation strategies, and experimental results in detail, highlighting the trade-offs and challenges encountered throughout the project.**

I. Introduction

This project focuses on the problem of quantum circuit testing within the context of fault detection and automatic test pattern generation (ATPG), also referred to as quantum Automatic Test Generation (qATG). The goal is to determine whether a given quantum circuit implementation deviates from its expected behavior due to the presence of faults, and to design test circuits capable of exposing such deviations.

we focus on four essential components of the testing workflow that are directly relevant to the implementation and experiments carried out in this project:

- **Fault Modeling:** This involves formally describing possible error mechanisms at the quantum gate level. A fault model specifies how a gate may behave incorrectly—such as by introducing a Pauli error or altering gate parameters—allowing us to simulate and reason about faults in a systematic way.
- **Fault Simulation:** Once a fault model is defined, we simulate how that fault would impact the measurement distribution of a quantum circuit. Since quantum outputs are probability distributions over bitstrings, we compare faulty vs. fault-free output distributions as part of this analysis.
- **Test Generation:** The goal of test generation is to design a quantum test circuit such that, when executed, it maximizes the ability to distinguish between correct and faulty circuit behavior. This is analogous to ATPG in classical testing, but must take quantum properties and fault models into account. We use the qATG tool to generate such test circuits automatically.
- **Fault Detection:** Finally, we must decide—based on measurement data—whether a fault is present. This is typically done by comparing the observed statistics of the test circuit's outputs against expected behavior under the fault-free model, using statistical divergence tests such as Chi-square and KL divergence with bootstrapping.

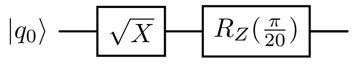
This report documents the process of applying this testing framework to multiple custom-designed quantum circuits, using fault injection, test circuit generation via qATG, and experimental evaluation of fault detectability under different statistical criteria. We emphasize implementation details and offer insights drawn from the results, aiming to contribute to the practical understanding of quantum circuit testing under realistic conditions.

II. Fault Modeling

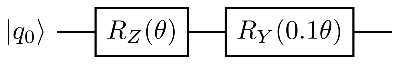
In this project, we tested three custom quantum fault models, each corresponding to a specific type of gate error. These models are not chosen arbitrarily—they simulate real physical imperfections commonly seen in superconducting and ion-trap systems, such as over-rotations, control drift, and crosstalk.

A. Description of Fault Models

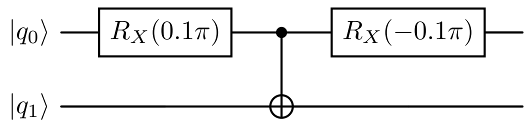
- **Fault Model 1 (SX Gate Fault):** This fault injects a small unwanted rotation around the Z-axis following a $\sqrt{X}$ (also known as SX) gate.



- **Fault Model 2 (RZ Gate Fault):** This model assumes that a parameterized $R_Z(\theta)$ gate suffers from over-rotation, but not around the same axis. Instead, a small rotation is added around the Y-axis, proportional to the original angle θ.



- **Fault Model 3 (CX Gate Fault):** This model simulates imperfect controllability in two-qubit gates. Specifically, it applies an undesired $R_X(\pm 0.1\pi)$ to the control qubit both before and after a standard CX gate.



B. qATG Test Generation

The faulty gates were implemented as subclasses of `qATGFault`, adhering to the qATG requirement that each fault be defined via an ideal and faulty version constructed as `QuantumCircuit` objects. A key implementation detail was the need to wrap these circuits using Qiskit's `Operator` and `UnitaryGate` classes to ensure compatibility with qATG's backend for gate replacement and test generation. Notably, even if two faulty behaviors conceptually share the same structure (e.g., same fault formula across qubits), qATG requires them to be defined as separate class instances when targeting different qubit indices.

For parameterized gates, such as Fault 2, explicit management of symbolic parameters via Qiskit's `Parameter` object was necessary. Failure to do so caused qATG to raise exceptions due to incompatible gate types or unresolved parameter bindings during internal compilation.

Once the faults were registered, qATG was used to generate distinguishing test configurations under the following conditions:

- One- or two-qubit circuit templates matching the arity of the faulty gate.
- Initialization in computational basis states (`ket(0)` for 1-qubit, `ket(00)` for 2-qubit).
- A minimal fidelity threshold of 0.1 to ensure perceptible state deviation.

qATG Output and Observations: Across all three faults, qATG successfully generated test circuits that exposed the injected errors. However, several notable patterns emerged:

- **Gate Diversity**: For the three main fault types, the synthesized test circuits predominantly employ their respective native gates combined with general single-qubit U gates. For example, the fault involving SX gates is mainly constructed using SX and U gates, while the fault with RZ gates uses mostly RZ and U gates. This indicates that qATG favors leveraging the intrinsic gates associated with each fault, supplemented by the flexible U gate to efficiently generate expressive and minimal test sequences.
- **State Fidelity**:
  - For Fault 1 (SX gate augmented with R_Z), the optimal test configuration achieved a fidelity of approximately **0.0949**, indicating it was just below the threshold and relatively hard to detect. Despite the added R_Z rotation, the faulted SX gate retained partial alignment with the original, contributing to lower sensitivity.
  - Fault 2, which perturbs a standard $RZ(\theta)$ gate by appending a small $RY(0.1\theta)$ rotation (with $\theta = \frac{\pi}{2}$), resulted in a significantly lower fidelity of approximately **0.0545**. Despite the subtle nature of this orthogonal deviation, it was readily detectable by qATG's synthesis, indicating high sensitivity to out-of-plane perturbations.
  - Fault 3, involving a perturbation of the CNOT control behavior, yielded fidelities consistently **below 0.05**, with some configurations achieving values under **0.02**. This implies that the fault introduced significant entanglement errors, making it easier for qATG to isolate deviations.
- **Circuit Structure**: The generated circuits, particularly for Fault 1, often included repeated layers of $\sqrt{X}$ and U gates, with nested inverses that resembled echo-like structures. For Fault 3, the test sequences frequently relied on controlled operations and entanglement-driven probes, suggesting qATG optimizes for circuit constructions that emphasize fault propagation paths.
- **Execution Time**: Circuit synthesis time scaled with both qubit count and the subtlety of the fault. Faults that were closer to the identity (like Fault 1) required longer synthesis due to the difficulty of reaching the fidelity threshold, while stronger faults like Fault 3 were identified more quickly.
- **Export Artifacts**: qATG's exported QASM files did not retain metadata about the injected faults, so manual bookkeeping remains necessary to associate test circuits with their corresponding fault classes.

Reflection on Use of qATG: From an implementation standpoint, the most delicate part was conforming to qATG's internal gate replacement logic, which is strict about gate types and qubit indexing. The framework provides powerful automated synthesis, but at the cost of some verbosity in setting up fault classes, especially for multi-qubit gates or parameterized faults. Nevertheless, the generated test circuits proved effective at capturing behavioral drift, making qATG a valuable tool for fault diagnosis and quantum circuit validation under realistic noise models.

III. Fault Simulation

A. Simulation Framework Overview

To investigate the impact of gate-level faults on quantum circuits, we developed a simulation framework that systematically injects faults by replacing specific gates in a quantum circuit with faulty counterparts defined by fault models. The modified circuits are then executed on Qiskit's `AerSimulator` to observe the resulting output distributions.

The overall procedure involves:

- **Gate Scanning and Replacement:** The input quantum circuit is deep-copied to preserve the original. Each gate is inspected sequentially, and if it matches the fault model's target gate on the specified qubits, it is replaced with the corresponding faulty gate generated by the fault model.

- **Automatic Measurement:** To ensure that the circuit outputs classical measurement results, all qubits are measured if the circuit lacks explicit measurement operations or classical registers.
- **Simulation Execution:** The resulting faulty circuit is run on the `AerSimulator` backend, with a configurable number of shots (repeated executions) to statistically approximate the output distribution.

This approach allows for flexible plugging-in of any fault model and seamless comparison of the output distributions between fault-free and faulty circuits.

B. Implementation Details

The fault injection process is carefully implemented to preserve circuit semantics:

- **Gate Matching:** Replacement occurs only when both the gate type and the exact qubit indices (including their order) match those specified by the fault model. This ensures precise fault localization and avoids unintended substitutions.
- **Gate Replacement Handling:** The faulty gate can be either a single gate or a composite circuit. When the faulty gate is composite, each constituent operation is mapped to the corresponding qubits in the original circuit to maintain correct connectivity.
- **Measurement Addition:** If the circuit does not contain classical registers or measurements, a global measurement operation is appended to all qubits. This guarantees observable output for statistical analysis.
- **Simulation Backend:** The `AerSimulator` is used due to its high performance and fidelity for circuit execution, supporting large numbers of shots for reliable statistics.

C. Simulation Results and Observations

Each quantum circuit (qc1, qc2) was simulated both in its ideal and faulty versions using 100,000 shots to observe the effect of fault injection on output distribution.

- **Fault 1 (SX + RZ):** Minor phase errors slightly alter interference patterns, resulting in small deviations in output frequency (e.g., more `00` and `11` outcomes).
- **Fault 2 (RZ + RY):** The added Y-rotation shifts the state amplitude, leading to more dominant `10` outputs in specific circuits.
- **Fault 3 (RX-CX-RX):** The most disruptive fault, strongly modifying entanglement and increasing `11` probability significantly.

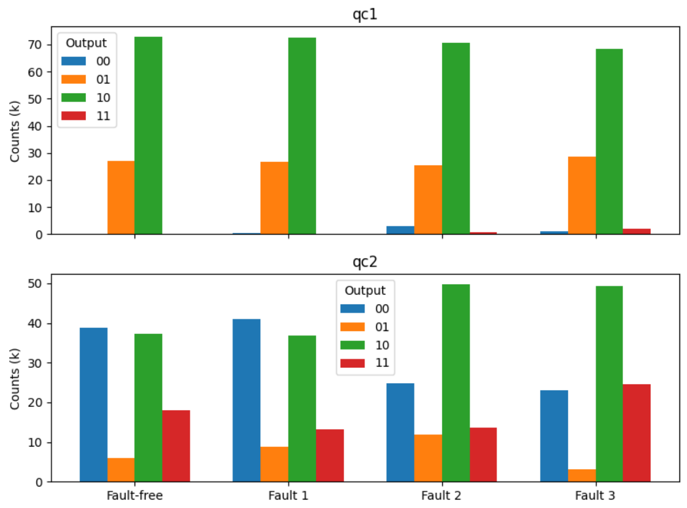


Fig. 1: Output Distributions for Sample Circuits under Each Fault Model

The output statistics provide a clear fingerprint of different fault types, laying the groundwork for systematic fault detection techniques.

IV. Fault Detection

This section documents two statistical methods for detecting faults in quantum circuits based on output distributions, each with different tradeoffs in sensitivity and computational overhead.

Method 1: Chi-square Test

The first method uses the classic Chi-square statistic to compare the observed output distribution from a possibly faulty circuit to the expected distribution obtained from a fault-free simulation.

Test Statistic:

$$\chi^2 = \sum_i \frac{(O_i - E_i)^2}{E_i}$$

where O_i and E_i represent the observed and expected counts for each bitstring outcome, respectively.

Implementation Notes: To ensure numerical stability, both the observed and expected distributions are Laplace-smoothed with a small constant ($\varepsilon = 10^{-6}$) to avoid division by zero. After smoothing, both distributions are scaled to match the total shot count (typically 100,000) before applying the test.

The SciPy implementation is used to obtain a p-value from the Chi-square statistic. A fault is declared if the p-value falls below a chosen threshold `max_escape`. Optionally, an overkill guard is implemented using a secondary threshold `max_overkill` to prevent false positives when testing fault-free circuits.

Advantages:

- Lightweight and fast; no retraining or resampling required.
- Effective at catching large distributional shifts.

Limitations:

- Sensitive to noise in low-probability outcomes.
- May overreact to minor variations, especially under high shot counts.

Method 2: KL Divergence with Bootstrap Threshold

This method leverages the KL-divergence between output distributions, with the detection threshold empirically estimated through bootstrapping.

KL Divergence Formula:

$$D_{\mathrm{KL}}(P \parallel Q) = \sum_i p_i \log \frac{p_i}{q_i}$$

where P is the observed probability distribution from the possibly faulty circuit, and Q is the expected distribution from the fault-free baseline.

Implementation Details: Both distributions are normalized to form valid probability vectors using shot counts. To prevent log-zero errors, missing keys are assigned a small probability floor (10^{-9}). The KL divergence is computed using 'scipy.stats.entropy'.

Since KL-divergence lacks a known closed-form distribution under quantum sampling noise, the fault-free circuit is simulated n times (default: 100) to empirically generate a reference distribution of divergences. The detection threshold τ is then defined as the $100(1-\alpha)\%$ percentile of this bootstrapped distribution (e.g., $\alpha = 0.01 \Rightarrow \tau = 99\%$ percentile).

Detection Rule: A fault is flagged if the observed KL divergence exceeds the bootstrapped threshold:

$$D_{\mathrm{KL}}(P \parallel Q) > \tau$$

Advantages:

- Non-parametric; adapts to the actual noise characteristics of the quantum hardware.
- Tunable false positive rate via the bootstrap percentile (α).
- More robust against false alarms compared to Chi-square.

Limitations:

- Requires 100+ simulations per detection, increasing runtime.
- Threshold is sensitive to shot count and bootstrap size.
- KL-divergence is asymmetric; direction matters.

Comparison of Detection Accuracy

Detection Rates (%) on Simulated Circuits:

Circuit	Fault	Chi-square	KL+Bootstrap
qc1	none	16.67	3.33
	myfault_1	100.00	100.00
	myfault_2	100.00	100.00
	myfault_3	100.00	100.00
qc2	none	3.33	0.00
	myfault_1	100.00	100.00
	myfault_2	100.00	100.00
	myfault_3	100.00	100.00

Discussion: Both detection methods achieve perfect accuracy in identifying all real faults across two test circuits. However, differences emerge in their handling of fault-free cases:

- The Chi-square test exhibits a moderate false positive rate (e.g., 16.67% on `qc1`).
- The KL+Bootstrap method shows improved precision, with nearly zero false alarms.
- These results confirm that while the Chi-square test is fast and sufficient for detecting obvious faults, the KL approach offers a stricter and more statistically grounded alternative for subtle deviations.

Summary

The choice of fault detection method depends on the desired tradeoff between computational cost and false positive tolerance. The Chi-square test is suitable for rapid diagnostics, while KL+Bootstrap offers finer control and reliability, particularly in low-noise or production-quality quantum settings.

V. Testing on Quantum Simulators

This section presents the testing process for several quantum circuits using the qATG framework and custom fault detection methods. Three fault models were implemented following the abstract fault interface, and test configurations were generated to target each fault. The generated tests were then applied to a set of unknown circuits to assess fault detectability.

A. Test Configuration and Discriminative Power

Three fault models (`myfault_1`, `myfault_2`, `myfault_3`) were created as subclasses of the abstract fault model, conforming to the inheritance structure defined in Problem 1. For each fault, a test configuration (circuit) was automatically generated by qATG to maximize its detectability.

To evaluate the ability of these test circuits to discriminate between fault types, we injected different faults into each test circuit and measured the detection rate using both the Chi-square method and KL+Bootstrap divergence. Results are presented in Table I.

Test Circuit (Target)	Injected Fault	Chi-square (%)	KL+Bootstrap (%)
test1 (myfault_1)	none	10.00	0.00
	myfault_1	100.00	100.00
	myfault_2	3.33	0.00
	myfault_3	10.00	6.67
test2 (myfault_2)	none	0.00	0.00
	myfault_1	13.33	0.00
	myfault_2	100.00	100.00
	myfault_3	3.33	0.00
test3 (myfault_3)	none	46.67	3.33
	myfault_1	20.00	6.67
	myfault_2	16.67	6.67
	myfault_3	100.00	100.00

TABLE I: Detection Rates of qATG Test Circuits on Injected Faults

Insights

- Each test circuit demonstrates high sensitivity (100%) to its targeted fault model.
- Detection of off-target faults is low, validating the discriminative design of the test circuits.
- The Chi-square test shows higher false positives (e.g., detecting fault-free runs), while KL+Bootstrap is stricter.
- These results confirm that qATG successfully generates circuits tailored to specific fault characteristics.

B. Testing on Backend Circuits

The three generated test configurations were applied to four unknown quantum circuits located in the `CUT` directory. The goal was to determine whether any of the defined faults could be consistently detected in these backends. The Chi-square method was used as the detection criterion.

A test is considered:

- **Failed** if the detection rate is 100% (fault was always detected),
- **Passed** if the detection rate is less than 100% (fault not reliably detected).

Quantum Backend	**Config. 1**	**Config. 2**	**Config. 3**
Backend 1	**Failed**	**Failed**	**Failed**
Backend 2	**Passed**	**Passed**	**Failed**
Backend 3	**Failed**	**Failed**	**Failed**
Backend 4	**Passed**	**Passed**	**Passed**

TABLE II: Fault Detection Status on Backends (Chi-square). **Passed** = detection rate $< 100\%$, **Failed** = detection rate $= 100\%$

C. Discussion

Effectiveness of Targeted Tests

The effectiveness of the test configurations was validated prior to backend application. Their ability to detect their respective faults with high specificity provided a solid foundation for backend diagnosis.

Backend Analysis

- **Backend 1 and 3** failed all tests, suggesting strong fault exposure and lack of masking.
- **Backend 2** passed the first two tests but failed the third, implying fault `myfault_3` is the most disruptive in this circuit.
- **Backend 4** passed all tests, indicating that either no relevant faults are active or their effects are unobservable in measurement outcomes.

Implications for Circuit Diagnosis

- Discriminative tests help narrow down which fault classes are active, especially when tests are passed selectively.
- Chi-square detection may raise false alarms on circuits with high measurement variance, as seen in minor detections in fault-free configurations.
- KL+Bootstrap, while stricter, may overlook subtle but real statistical shifts, especially in small sample sizes.

Takeaways

- qATG-generated circuits can serve as reliable, fault-specific probes across varied quantum architectures.
- Test outcome diversity across backends highlights the diagnostic potential of automated fault testing.
- Future testing frameworks may benefit from hybrid decision metrics combining sensitivity (Chi-square) with specificity (KL+Bootstrap).

VI. Conclusion and Future Work

A. Summary

This project explored the process of fault modeling, test generation, and statistical detection within quantum circuits, using the qATG framework.

We began by modeling quantum faults in a structured and extensible manner, ensuring compatibility with qATG's requirements for fault type and qubit configuration. Faulty and fault-free versions of circuits were simulated, generating statistical data that enabled comparison and detection.

Two statistical methods were applied to distinguish faulty behavior:

- **Chi-square test**: A simple and sensitive method that showed high detection power but occasionally over-detected due to statistical variance.
- **KL-divergence with bootstrap resampling**: A more principled and noise-tolerant method that provided robust confidence estimates, albeit with slightly reduced sensitivity.

Test circuits generated by qATG were highly effective in detecting their respective target faults. Empirical evaluation confirmed that each test configuration showed strong specificity and detection power when applied to its corresponding fault model.

When applied to unknown quantum backend circuits, these test configurations helped reveal which faults, if any, were active. Variations in outcomes across different backends demonstrated the practical value of automated test generation and statistical evaluation in fault diagnosis.

B. Challenges Encountered

Several challenges arose during the implementation and experimentation:

- **Rapid evolution of Qiskit**: Frequent updates to the Qiskit library introduced compatibility issues, deprecations, and inconsistencies in simulation behavior.
- **Fault model complexity**: Ensuring that each fault conformed to the correct gate type and number of qubits required careful, manual attention.
- **Randomness and circuit noise**: Even in simulations, detection results exhibited significant variance across runs, necessitating repeated trials and robust statistical tools.

C. Future Directions

This project lays the groundwork for a more scalable and systematic approach to quantum fault testing. Future enhancements may include:

- Developing a standardized **library of reusable fault models** along with a repository of validated test circuits for benchmarking.
- Investigating **machine learning approaches** for both test generation and fault classification, which may provide better generalization across unseen circuit structures.
- Integrating noise-aware statistical modeling and adaptive sampling strategies to better handle circuit randomness.

Closing Remarks

Through the integration of abstract fault modeling, automated test generation, and statistical detection, this project demonstrates a practical and extensible framework for quantum circuit testing. As quantum hardware matures, such systematic approaches will become increasingly important for debugging, validating, and certifying quantum computations.

Project Repository

All code and experiments were implemented using Python and Qiskit, and are available at the project repository:

https://github.com/LaiYuHong/Quantum-Testing-via-qATG

References

[1] NTU-LaDS-II qATG. Quantum Automatic Test Generator. https://github.com/NTU-LaDS-II/qATG

[2] Qiskit: An Open-source Framework for Quantum Computing. https://qiskit.org