qATG: Automatic Test Generation for Quantum Circuits

Chen-Hung Wu, Cheng-Yun Hsieh, Jiun-Yun Li, and James Chien-Mo Li
Graduate Institute of Electronics Engineering, National Taiwan University, Taipei 10617, Taiwan
{r07943150, d08943012, jiunyun, cmil}@ntu.edu.tw

Abstract—**Researchers now use randomized benchmarking or quantum volume to test quantum circuits (QC) in the laboratory. However, these tests are long and their fault coverage is unclear. In this paper, we propose behavior fault models based on the function of quantum gates. These fault models are scalable because the number of faults is polynomial, not exponential, to the size of QC. We propose a novel test generation that uses gradient descent to generate test configuration with short length. We revise the chi-square statistical method to decide the number of test repetitions under the specified test escape and overkill. Experimental results on IBM Q systems show that our generated test configurations are effective, and our test lengths are 1,000X shorter than traditional test methods.**

Keywords—Quantum Circuit, Test Generation, Fault Model

I. Introduction

Quantum circuits (QC) are becoming an important computational technology in many useful applications, such as machine learning, optimization, and cryptography. Recently, QC have made significant advances: Google's 53 qubits Sycamore [1], Intel's 49 qubits Tangle Lake [2], and IBM's 50 qubits chip [3]. There are many potential technologies for QC: superconductor, quantum-dots, ion-traps, *etc*. Although the above QC are still in the laboratory, it is expected that QC will be commercially produced in the future.

If QC are going to be commercially produced, short and effective testing of QC is important to guarantee their quality. It has been shown that quantum gate fidelity must be greater than 99% to be practically useful [4]. However, testing QC is very different from testing traditional circuits for the following reasons. First, all quantum gates are reversible so traditional fault models, such as stuck-at fault and delay faults, are not suitable for QC. Second, unlike traditional circuits which use flip-flops or latches as memory, QC do not have state holding elements and scan design for testability (DfT). Instead, QC have many quantum states simultaneously, so we will need to consider all states to compute the *output probability distribution* (OPD). Third, QC are inherently probabilistic, not deterministic. QC may produce different outputs, even when the same test patterns are applied. We need to determine how many times we repeat a test pattern (called *test repetition* in this paper) to obtain test results given a confidence level. Fourth, there is no fixed circuit structure for QC so we will need to determine a sequence of quantum gates (called a *test configuration* in this paper) for the quantum circuit under test (CUT). Please note that, in this paper, we are testing *configurable quantum circuits*, which are also known as *quantum processors*.

The first two authors contributed equally to this work.

Conventionally, researchers use *randomized benchmarking* (RB) and *quantum volume* (QV) to test QC. However, these methods are designed to measure the performance of good circuits in the laboratory. Their test time is long and fault coverage is low. So far, there is no automatic test generation technique suitable for the production test of QC.

In this paper, we try to answer four questions about testing QC in commercially production. (1) What are the suitable fault models for QC? (2) How to automatically generate optimal test configurations to test QC? (3) What is the smallest number of times to repeat a test pattern to detect a fault? (4) What are the *fault coverage, test escape*, and *overkill* of our tests?

In this paper, we propose behavior fault models based on the behavior of quantum gates so that they represent realistic faults that can occur in QC. We also propose an automatic test generator which generates different test configurations for the given fault and QC. Our test configuration has higher fault coverage and shorter test lengths than existing test methods (RB and QV). We also propose to revise the chi-square statistical method to determine the test repetition, given an expected test escape and overkill. Finally, our generated tests are verified on the real QC, five-qubit IBM Q systems.

The advantages of our techniques are as follows. First, our tests are based on behavior fault models that can be used in many different QC technologies. Also, we can provide fault coverage quantitatively. Second, the number of faults is polynomial to the circuit size (not exponential) so the fault models are scalable to large QC. Third, we are able to calculate the minimum test repetition, so that our confidence level is quantified. Fourth, we generate test configuration to minimize the test repetition, so our tests are orders of magnitude shorter than existing test methods. It is suitable for the commercially production test environment.

This paper is organized as follows. Section II gives a brief introduction to the related past techniques for testing QC. Section III describes how we use conventional methods (RB and QV) to test QC. Section IV proposes our qATG to test QC. Section V provides experimental results, including simulation and real experiments on IBM Q. Finally, Section VI concludes this paper.

II. Background

In this section, we briefly introduce the concepts of QC, the previous techniques for QC testing, and chi-square testing.

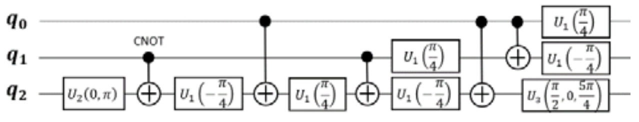
A. Quantum Circuit Concepts

A quantum state $|\psi\rangle$ of an n-qubits QC can be represented in the following form [5]:

$$|\psi\rangle = \sum_{i=0}^{2^n-1} c_i\,|i\rangle$$

, where $|i\rangle$ is the ith basis, ci is a complex number which represents the coefficient corresponding to $|i\rangle$. The measurement of QC results in exactly one observed state. The probability of observing $|i\rangle$ when measuring QC is $|c_i|^2$(note that $\sum_{i=0}^{2^n-1}|c_i|^2 = 1$). After measuring the QC, quantum states are collapsed into the observed state. Hence, we have to repeatedly operate the QC to obtain the OPD.

Figure 1 shows a diagram representing a quantum half adder circuit composed of *basic gates* on IBM Q. Basic gates are basic building blocks of a certain QC technology. Every horizontal line in the diagram represents a qubit and each object on the line represents a quantum gate. Below the quantum half adder circuit, Figure 1 shows the gate matrices of basic gates: *$U_1(\lambda)$*, *$U_2(\phi, \lambda)$*, *$U_3(\theta, \phi, \lambda)$*, and *CNOT*. A single-qubit gate is represented by a 2×2 matrix and a two-qubit gate is represented by a 4×4 matrix. Please note that Figure 1 shows ideal gate matrices without noise. In practice, due to environmental or material imperfection, realistic gate matrices can be interfered by *noise*, which make the realistic gate matrix different from the ideal gate matrix.



$$U_1(\lambda) = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\lambda} \end{pmatrix} \qquad U_2(\phi,\lambda) = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & -e^{i\lambda} \\ e^{i\phi} & e^{i\lambda+i\phi} \end{pmatrix}$$

$$U_3(\theta,\phi,\lambda) = \begin{pmatrix} \cos(\theta/2) & -e^{i\lambda}\sin(\theta/2) \\ e^{i\phi}\sin(\theta/2) & e^{i\lambda+i\phi}\cos(\theta/2) \end{pmatrix} \qquad CNOT = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

Figure 1: QC half adder and gate matrices

Unlike conventional circuits, which have fixed connection and functions, QC can be configured to perform different functions. Therefore, to test a QC, we first need to connect its quantum gates in different ways, called test configurations. For example, the half adder is a test configuration for a three-qubit QC. *Test length* (L_i) of the i_{th} test configuration is defined as the maximum *unit operations* in series, starting from the QC input to the QC output. Each object in the same column represents a unit operation. For example, test length of the half adder is ten. Although each gate operation takes a slightly different time, in this paper, the time of each unit operation (T_{op}) is assumed to be the same.

QC is inherently probabilistic, not deterministic. We need to determine how many times we apply a test configuration (called test repetition) to obtain test results under a given confidence level. Test repetition (R_i) for the i_{th} test configuration is defined as the number of times that we repeatedly test a QC in the i_{th} test configuration. *Total test operation time* (TT_{op}) is equal to the summation of L_i times R_i times T_{op} for all *i*. Unlike conventional circuits which do not need initialization, QC needs *initialization* before applying each test configuration, and also *measurement* after applying each test configuration. *Total test initialization/measurement time* (TT_{im}) is equal to R_i times *initialization and measurement time* (T_{im}) for all *i*. The *total test time* (*TT*) to test a QC is therefore as follows:

$$TT = TT_{op} + TT_{im} = \sum_i L_i R_i T_{op} + \sum_i R_i T_{im} \qquad (1)$$

For example, if we repeat testing QC in Figure 1 for 5 times, L=10, R=5 and TT = 50 T_{op} + 5 T_{im}.

B. Previous QC Testing

In this section, we shortly introduce previous techniques to test QC. These test methods can be roughly classified into three types: (1) exhaustive testing of QC, (2) ATPG for QC, (3) testing for the entangled states of QC.

Exhaustive testing of a quantum gate systematically explored the response to a complete set of states in the system's *Hilbert space* [6]. The main advantage of their methods is that it needs no assumption (on fault model or errors) for the QC. However, this method has a fatal drawback — the test time of this method is exponential to the number of qubits.

The ATPG method in the previous work tried to detect *missing gate fault* (MGF) on a particular gate sequence. In [7], they showed that MGFs are easy to detect. They used an input state vector to detect all MGFs by adding one extra qubit and several *CNOT* gates. In [8], they used Boolean satisfiability and pseudo-Boolean optimization to determine the optimal input test pattern (input state vector) for testing MGFs on QC. Previous ATPG papers have a serious disadvantage — they used input state vectors to test a specific QC gate sequence. However, QC is typically initialized to the ground state so the input state vectors are actually fixed. Moreover, QC are configured into sequence of quantum gates. Therefore, there are no specific quantum gate sequences for a QC.

Since entangled states in QC are important, testing for the entangled state is another important research topic. In [9], they introduced a general framework to test maximally entangled states and measurement operations. In [10], they developed a scheme for testing the maximally entangled states of a two-qubit system. In [11], they tested entangled states by only one or two *CHSH games*. The work showed that entangled states can be tested in very short test lengths. However, a QC with good entangled states is not necessarily a fault-free QC. Hence, only testing entangled states is not enough. We still need to test the other faults in the QC. In conclusion, there is no suitable test technique for QC in the production test environment.

C. Chi-square Test

The chi-square test is based on the hypothesis testing [12]. The *null hypothesis* is the observed occurrence frequency (p_o) match expected occurrence frequency (p_N). The *alternative hypothesis* is the observed occurrence frequency does not match the expected occurrence frequency. Given the sample data, we can compute χ^2 by Equation (2), where R is the sample size and k is the number of categories.

$$\chi^2 = \sum_{j=1}^{k} \frac{(R\times p_{Nj} - R\times p_{oj})^2}{R\times p_{Nj}} \qquad (2)$$

If p_o matches p_N and we repeatedly sample many times, the distribution of χ^2 will match the *chi-square distribution*. Hence, to test our hypothesis, we can look up the χ^2 distribution table with *degree of freedom* (k-1) under a given confidence level. If the computed χ^2 is not bigger than the *critical value* χ_T, we say that the null hypothesis is accepted. Traditional chi-square testing cannot be use directly to test QC because it has only one confidence level. However, we need two confidence levels to control both test escape and overkill.

III. Testing by RB and QV

In this section, we explain how we test QC with two traditional standard verification techniques: randomized benchmarking (RB) and quantum volume (QV).

A. Testing by Randomized Benchmarking

RB is the most frequently used method for estimating the performance (such as error rate) of QC [13]. By repeatedly applying a sequence of random gates, RB can be used to determine the *error rate* of the CUT. Error is caused by *random noise* in the QC that cannot be avoided. The advantage of RB is its insensitivity to the initial state preparation and measurement error. RB is very useful to measure *time-independent* and *gate-independent* errors. Gate-independence means that *any type of gate on a qubit* has the same amount of errors. Time-independence means that the error does not change with time.

A very popular RB has been proposed to tolerate *weak* gate-dependent errors [14], which means *each gate on a qubit* has slightly different amount of errors. It robustly benchmarks the full set of *Clifford gates*. By applying this method, we can determine the average *error-rate per Clifford gate* (*EPC*).

The original usage of RB is to measure the error rate of each quantum gate. In this paper, we would like to use RB for testing whether a QC is defective. The test process can be separated into two stages. In the first stage, we determine the parameters and acceptable *EPC* range. In the second stage, we run the RB test on the CUT and determine whether the CUT passes or not.

Figure 2 is the flow of our modified RB test generation. First, we need to specify user-defined parameters, such as test lengths, test repetitions, *etc.* Given these parameters, we generate the RB circuits using the method provided by a popular open-source framework, *qiskit* [15]. After that, we *transpile* (like *technology mapping* in standard cell design) the generated test configuration into QC with the *basic gate set* (like standard cell library). To verify the overkill and test escape, we use Monte-Carlo simulation on both good QC and faulty QC.

We denote the *test escape* by α : the ratio of faulty CUTs that pass the test, as described in Equation (3). We denote *overkill* by β : the ratio of good CUTs that fail the test, as described in Equation (4).

$$\text{Test Escape } (\alpha) = \frac{number\ of\ faulty\ CUT\ passing\ the\ test}{number\ of\ faulty\ CUT} \quad (3)$$

$$\text{Overkill } (\beta) = \frac{number\ of\ good\ CUT\ failing\ the\ test}{number\ of\ good\ \ CUT} \quad (4)$$

Given the acceptable test escape (α) and *EPC* of a faulty QC, we can estimate the overkill (β) with the assumption of normal distribution. We can write the *EPC* of a good QC as $EPC_G \pm \Delta EPC_G$, where ΔEPC_G represents the standard deviation of *EPC*. We can also write *EPC* of QC of fault *f* as $EPC_f \pm \Delta EPC_f$. Thus, we can write the relation between test escape and *EPC threshold* (E_{th}) as follows:

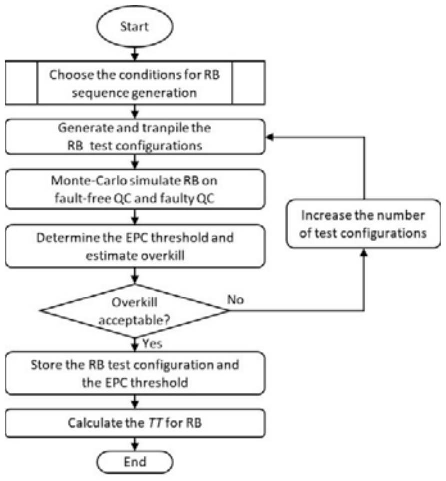
$$E_{th} = \min_f EPC_f - \sqrt{2}\Delta EPC_f \times erf^{-1}(2\alpha - 1) \quad (5)$$

, where *erf* is the *error function*, *i* is the index of the fault. If *EPC* of a CUT is larger than E_{th}, then we determine that the CUT fails the RB test. If the *EPC* of CUT is smaller than E_{th}, then we determine that the CUT passes the RB test.

Once the E_{th} is decided, we can estimate the overkill (β) using the following Equation:

$$\beta = \frac{1}{2}\left[1 + erf\left(\frac{E_{th} - EPC_G}{\sqrt{2}\Delta EPC_G}\right)\right] \quad (6)$$

If the overkill is acceptable, we store the RB test configurations and the E_{th}. Otherwise, we increase the number of test configurations and repeat the above process. Finally, we calculate the *TT* according to Equation (1).



Start
Choose the conditions for RB sequence generation
Generate and tranpile the RB test configurations
Monte-Carlo simulate RB on fault-free QC and faulty QC
Determine the EPC threshold and estimate overkill
Overkill acceptable?
No
Yes
Increase the number of test configurations
Store the RB test configuration and the EPC threshold
Calculate the TT for RB
End

Figure 2: flow of RB test generation

B. Testing by Quantum Volume

Another well-known method to verify the QC is to measure its *Quantum Volume* (QV) [16]. Given the same number of qubits and sequence length, QV measures whether the OPD meets the ideal expectation. QV can represent many important features, including fidelity of operation, connectivity, calibrated gate sets, *etc.* Thus, QV is a number (power of 2) that can be used to measure the computation power of QC. However, QV is not suitable to detect faults in the QC.

Figure 3 is the flow of QV testing. First, we generate the needed test configuration for measuring QV. Then, we transpile it into basic gates for CUT. After that, we test the CUT using QV by comparison of QV between CUT and fault-free QC. If the measured QV is different from that of fault-free QC, the CUT fails the test.

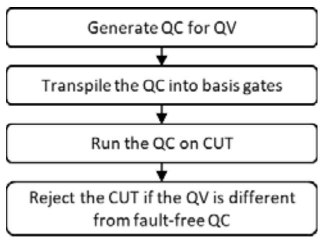

Generate QC for QV
Transpile the QC into basis gates
Run the QC on CUT
Reject the CUT if the QV is different from fault-free QC

Figure 3: flow of QV testing

IV. Proposed Techniques

A. qATG Test Generation Flow

Figure 4 is our proposed test generation flow of qATG. First, we generate a *fault model list* based on the *QC information*. The details of proposed behavior fault model will be discussed in Section IV.B. Second, we generate *test template* for a given *fault model*. Third, we repeatedly generate test configurations and run the fault simulation with a noise model. We will introduce the details in Section IV.C. After that, we use the chi-square test to estimate appropriate test repetitions in Section IV.D. Finally, we check whether the fault model list is empty. If so, the loop is terminated; otherwise, we go back to the second step.

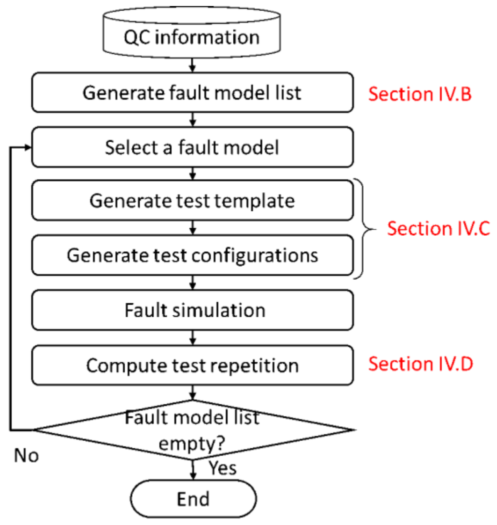


Figure 4: flow of qATG test generation

B. Fault Modeling and Injection

In this paper, we propose behavior fault models, including single-qubit gate fault models and two-qubit gate fault models. Our proposed fault models are based on two assumptions. First, *the same type of gates on a qubit* has the same fault. For the example in Figure 1, three U_1 gates on q_2 have the same faults. This is because we assume the faults are caused by permanent failure mechanisms, such as manufacturing defects. Second, *the same type of gates on different qubits* have their own faults. For the example in Figure 1, U_1 gates on q_0, q_1 and q_2 have their own faults. This is because each qubit receives different sources of control signals. Please note that RB assume that any types of gate on a qubit has the same fault. This is where our assumption is different from RB.

There are two methods to inject a fault into a quantum gate: (1) directly change the parameters of the gate under test, and (2) cascade an additional gate in serial with the gate under test. Hence, our proposed fault models have to consider the difference of behavior between a real device and computer simulation. In the computer simulation, we can use both methods. However, when testing the real IBM Q systems, we need to use the first method for single-qubit gate so that we avoid the extra noise introduced by the additional gate. For two-qubit gates, we still use the second method because the first method is hard to apply. The second method is reasonable since the noise of a two-qubit gate is much larger than that of a single-qubit gate. The noise of the additional gate will not significantly affect the test results.

For a single-qubit gate, we use a general quantum gate $U_3(\theta, \phi, \lambda)$ as an example, where θ, ϕ, λ are three user-defined parameters. Figure 1 shows the gate matrix of an ideal U_3 gate. We can model any behavior fault by the combination of three parameters. Because there are infinite number of combinations, in this paper, we propose to change one of three parameters at a time. In this way, the number of faults is *linear* to the number of quantum gates so that the number of faults is scalable to large QC in the future. Usually, these three parameters are controlled by three different physical sources, so our assumption makes sense. For each parameter, we consider three *fault categories*: (1) *ratio,* (2) *bias*, and (3) *truncation*. The ratio fault category models the resistive signal intensity decay that is *proportional* to our control. The bias fault category models the static charge or magnetic noise that is *additive*. The truncation fault category models the truncation of signal waveforms due to saturation. Therefore, for a U_3 gate, we have nine fault models: $f_{\theta r}$, $f_{\theta b}$, $f_{\theta t}$, $f_{\phi r}$, $f_{\phi b}$, $f_{\phi t}$, $f_{\lambda r}$, $f_{\lambda b}$, $f_{\lambda t}$. Figure 5(a) shows an original U_3 gate with parameters (θ, ϕ, λ). Figure 5(b) shows the corresponding U_3 faulty gate with parameters (θ', ϕ', λ'). The exact parameters of θ', ϕ', λ' can be defined by users according to the QC technology. For the other single-qubit gates, we can also handle them in the similar way. For example, an U_1 gate has only one parameter (λ). Hence, we have only three fault models: $f_{\lambda r}$, $f_{\lambda b}$, $f_{\lambda t}$.

For a two-qubit gate, we consider only the *CNOT* gate in this paper, because *CNOT* is used in the IBM Q systems. Figure 5(d) shows an original good *CNOT* gate and Figure 5(e) shows its corresponding faulty gate with six parameters. Before the control input, we inject a $U_3(\theta', \phi', \lambda')$. After the target output, we inject another U_3 (θ'', ϕ'', λ''). These parameters can be defined by users according to the QC technology. In this paper, these values are set according to the noise level of the IBM Q systems.

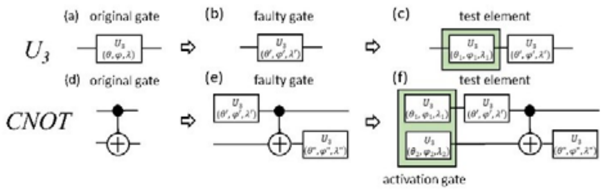


Figure 5: original gate, faulty gate, and test element for U_3 and $CNOT$

To generate a fault model list, we need the QC information, including *topology*, and *basic gate set*. Topology can be represented by a directed graph where *vertices* are qubits and *edges* are connections. Basic gate set is the set of quantum gates support by the QC technology. For single-qubit fault, suppose we totally have x adjustable parameters for all single-qubit gates in the basic gate set. We have $3x$ single-qubit fault models with the same faulty behavior, but not on any specified edge or vertex. Since each single-qubit gate has $3x$ fault models, so we totally have $3xn$ single-qubit faults in a QC. For two-qubit fault, suppose we have y fault models and m edges in QC topology. Then we have ym two-qubit gate faults. Totally, we have $3xn + ym$ faults in the QC.

Figure 6 shows the QC information for a 5-qubits QC. This *T-topology* QC has 5 (n) qubits and 8 (m) connections. First, because there are totally 6 (x) parameters in U_1, U_2, and U_3, we have 18 ($3x$) single-qubit fault models. Each fault model has one fault per qubit so there are 90 ($3xn$) single-qubit faults totally in this QC. Second, because we define 8 (y) two-qubit fault models in *CNOT*, so there are 64 (ym) two-qubit faults. Totally, there are 154 faults in this QC.

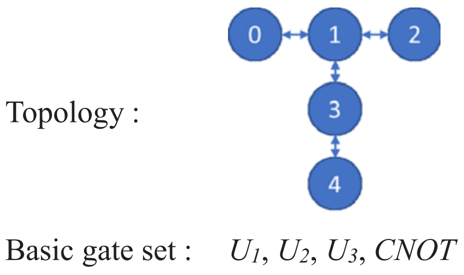


Figure 6: QC information example

C. *Generate Test Configuration*

In this section, we introduce the core of qATG, *test configuration*. Traditionally, to test a classical circuit, we need to apply a test pattern to *activate* the target fault and then to *propagate* its fault effect. However, to test a QC, we need to find test configurations, not test patterns. An optimal test configuration tries to maximize the difference of quantum states between the good and faulty QC. As a result, we want to maximize the difference of OPD between the good and the faulty QC.

In qATG, we propose a three-level hierarchical architecture: test element, test template, test configuration. A *test element* is composed of activation gate(s) and a faulty gate, as shown in Figure 5(c) and 5(f), where activation gates (green background) are U_3 before the faulty gate. A *test template* is composed of several test elements on a single qubit or double qubits. Finally, a *test configuration* is composed of several test templates to test the whole QC. Figure 7 has five test templates, each of which tests a qubit (in red dashed box). Each test template has four test elements (in blue dashed boxes), each of which activates and propagates the fault effects so that we can easily observe them at outputs.

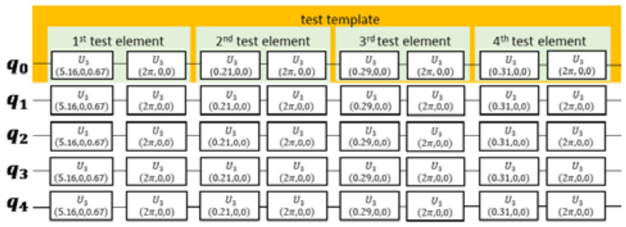


Figure 7: test configuration of $f_{\theta'}$ on U_3

Figure 8 is the flow of generating test configuration(s) for a fault model. First, we begin with an empty QC. The number of qubits in the empty QC is equal to the number of qubits that are affected by the given fault model. Second, we add a test element which can activate and propagate the fault effects of the give fault model. Third, to find the optimal parameters for the activation gates in the test element, we perform both good and fault simulation on this QC to search for optimal parameters using *gradient descent*. Fourth, we check whether two OPD is distinguishable and then decide whether we should go back to the second step or stop the iteration loop. Fifth, we replicate the generated test template into a test configuration.

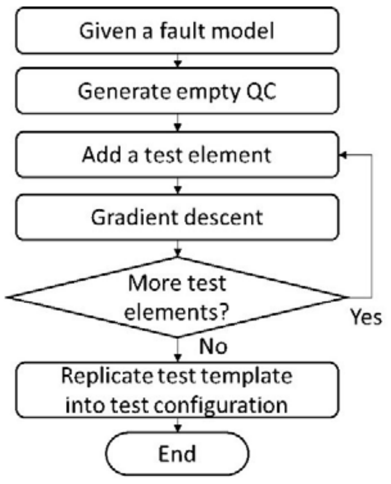


Figure 8: generating test configuration for a fault model

The most important concept in designing a test element is accumulating fault effect. However, direct cascade of many quantum gates cannot achieve this goal, because quantum gates do not have the *commutative* property. We use a single qubit to show that fault effect cannot be accumulated directly. A fault can actually be modeled as a rotation operation on the qubit. A single-qubit quantum state can be modeled as Bloch sphere. If we can accumulate the rotation effects in the same direction and axis, the difference of OPD will increase. We define the rotation operation of the good gate as R_O, and the rotation operation of faulty gate as R_F. Let R_T be the fault effect, which is a transformation from R_O to R_F. Then we have $R_F = R_T R_O$. For example, in Figure 9, R_O is a $\frac{\pi}{2}$ rotation around the z-axis and R_T is a $\frac{\pi}{2}$ rotation around the x-axis. Figure 9(a) are initial quantum states in the Bloch sphere of good and faulty QC. Figure 9(b) and 9(c) shows the quantum states after applying Ro and $R_T Ro$ once and twice, respectively. The difference of states are both $\frac{\pi}{2}$. Therefore, the difference of OPD does not increase by applying the same rotation twice.

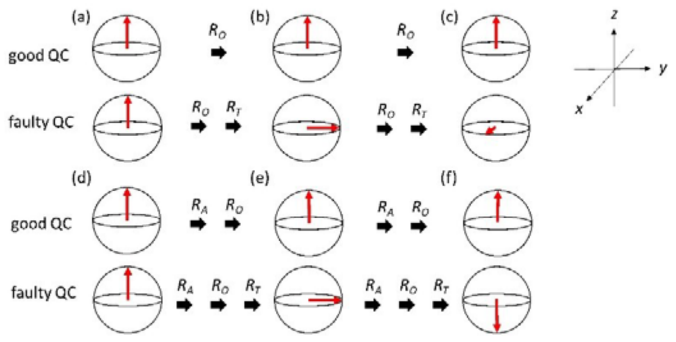


Figure 9: activation gate helps to accumulate fault effect

We need an activation gate to transform the quantum state so that the fault effect can be accumulated. We define the rotation of the activation gate as R_A. The rotation of a good test element is equal to $R_o R_A$; the rotation of a faulty test element is equal to $R_T R_o R_A$. In Figure 9, R_A is a $-\frac{\pi}{2}$ rotation around the z-axis. Figure 9(e) and 9(f) shows the quantum states after applying RoR_A and $R_T RoR_A$ once and twice, respectively. The difference between good and faulty are $\frac{\pi}{2}$ and π. Therefore, we can accumulate the fault effects by adding an activation gate.

Moreover, we cannot measure the imaginary part of quantum states; we can only measure the real part. By additional an activation gate, we can rotate the fault effects originally in the imaginary part into the real part so that we can measure. In this paper, we use U_3 as the activation gate because U_3 is a universal single-qubit gate with three adjustable parameters.

However, not all arbitrary U_3 gates can help to accumulate fault effect. We propose to use gradient descent to find the optimal parameters of U_3 in each test element. To quantitatively measure the difference of quantum states between good and faulty QC, we use *score* as a loss function in gradient descent. Score is the summation of squared differences between the faulty OPD and good OPD, defined as follows:

$$Score = \sum_{j=1}^{2^n} (p_{Nj} - p_{Aj})^2 \quad (7)$$

, where p_{Nj} is the faulty OPD of the j_{th} output state and p_{Aj} is the good OPD of the j_{th} output state.

Once a test template is generated, we can simply replicate the test template for each qubit to generate a test configuration. For a single-qubit fault model, we can replicate the test template for each qubit. As is shown in Figure 7, we can fit all test templates into one test configuration. For a two-qubit fault model, we need to replicate the test template for each edge in the QC topology. Take the topology in Figure 6 as an example, for a two-qubit fault model, there are at least six test configurations : {[0, 1], [3, 4]}, {[1, 0], [4, 3]}, {[1, 2]}, {[2, 1]}, {[1, 3]}, {[3, 1]}.

We now show an example to generate a test configuration in Figure 7. Continued from Section IV.B, we select the single-qubit fault model $f_{\theta'}$ on U_3 and set the ratio threshold to 0.9 ($\theta' = 0.9 \times \theta$). First, we generate an empty QC. Second, we add the 1st test element to the QC. Notice that we choose $U_3(2\pi, 0, 0)$ as the original gate because it has the maximum faulty effect in the fault model $f_{\theta'}$. Third, we apply gradient descent on the 1st test element and get parameters of the additional gate (θ_1, ϕ_1, λ_1) = (5.16, 0, 0.67). Fourth, add more test element(s) and apply gradient descent until the effect size is higher than a pre-defined upper limit (ω_{MAX}), or L reaches a pre-defined upper limit (L_{MAX}). In this step, we complete the generation of a test template for $f_{\theta'}$. Fifth, because the fault model of $f_{\theta'}$ on U_3 has 5 faults on q_0 to q_4. We generate a test configuration by replicating the test template to each qubit.

D. Revised Chi-square Test for QC

After finishing test configuration, we have OPD from good QC and faulty QC. We simulate both good QC and faulty QC with noise to obtain their OPD. Noise is injected by 10^5 Monte Carlo simulations using Qiskit. We propose to revise the chi-square test to determine: (1) test repetition, and (2) whether a CUT pass the test. For an n-qubits QC, the outcome of the QC is a random variable with probability p_j, where j is an integer from 0 to 2^n-1. The total probability sums up to one ($\sum_{j=0}^{2^n} p_j = 1$) and each p_j are independent. These properties are applicable to *chi-square of goodness of fit test*.

The traditional chi-square test only considers whether p_o belongs to p_N. Revised chi-square test determines whether p_o belongs to OPD of faulty QC (p_N) or OPD of good QC (p_A). If p_o belongs to p_N, the distribution of χ^2 after many sampling will approach the chi-square distribution. If p_o belongs to p_A, the distribution of χ^2 after many sampling will approach *noncentral chi-square distribution* [17].

In both distributions, the degree of freedom equals to 2^n-1. In noncentral chi-square distribution, the *noncentrality* (λ_N) parameter can be computed by Equation (8), where ω is *effect size* and R is *sample size* (R=test repetition in our case).

$$\lambda_N = R\omega^2 = R\sum_{j=1}^{2^n} \frac{(p_{Nj} - p_{Aj})^2}{p_{Nj}} \quad (8)$$

To determine test repetition R, first, we start from Equation (9), where χ_T can be looked up from the χ^2 distribution table with given α.

$$R = \left\lceil \frac{\chi_T}{\omega^2} \right\rceil \quad (9)$$

Because the chi-square test is *approximate testing*, the calculated R must have a low bound. We limit the lower bound of R by truncating the effect size to 0.8, which is large effect size suggested by [18]. Second, we define a *critical value* (c) as the boundary of pass and fail. Assume c_c is the chi-square value such that the cumulative distribution function of chi-square distribution is equal to 1-α. Assume c_n is the chi-square value such that the cumulative distribution function of noncentral chi-square distribution is equal to β. As shown in Figure 10, we keep increasing R until c_c equals to c_n. Because the noncentral chi-square distribution will right shift with R increasing, we can choose an appropriate R such that c_c equals to c_n. For example, assume p_N = [0.4, 0.3, 0.2, 0.1], p_A = [0.4, 0.2, 0.3, 0.1], and α = 0.99 (c_c =11.349), β = 0.999. Then ω = 0.289, χ_T = 11.345, so R= 136 by Equation (9) and c_n = 0.748. Because c_n< c_c, we keep increasing R until 468 such that c_n = 11.365. Finally, the needed test repetition is 468 with critical value c = 11.345. Please note that, for a test configuration that detects more than one fault, we calculate R for each fault. Then we pick the highest R as the test repetition of this test configuration.

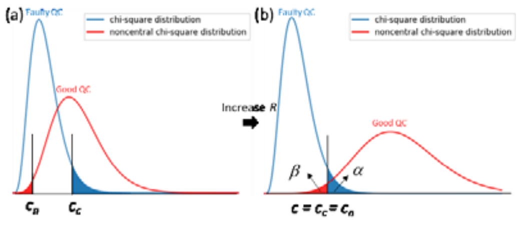


Figure 10: revised chi-square test: (a) start (b) end

To test whether the CUT is faulty with target fault, we can use Equation (2) to get χ and compare that with c. In this case, p_N is faulty distribution of target fault and k is 2^n. If $\chi \leq c$, the CUT is faulty. If $\chi > c$, the CUT pass this test. Continued from the previous example, if we test a QC 468 times and then get the results [187, 140, 97, 44]. The corresponding χ is 0.292, which means this QC fail the test because 0.292 ≤ 11.345. If we test another QC 468 times and then get the results [187, 97, 140, 44]. The corresponding χ is 36.585, which means this QC pass the test because 36.585 > 11.345.

V. Experimental Results

A. *IBM Q and Qiksit Setups*

The experimental setups for this paper can be summarized as Table 1. The main tool we use for simulating QC and accessing IBM Q systems is open-source software *Qiskit*. The Qiskit version we use is version 0.15.0. In Qiskit, there are three frameworks we have used in this work. We use *Qiskit Aer* to simulate the QC and build noise models for QC simulation. We use the *Qiskit Ignis* framework to generate RB and QV sequences. The last framework we used is *Qiskit IBMQ provider*. This framework connects to the cloud-based IBM Q systems. We submit jobs to the real QC device via this framework.

Table 1: summary of experimental setups

Qiskit version	qiskit: 0.15.0 qiskit-aer: 0.4.0 qiskit-ignis: 0.2.0 qiskit-ibmq-provider: 0.4.6
Noise model for QC simulation	gate error (depolarizing_error): 0.001 for single-qubit gates 0.01 for two-qubit gates readout error: 1.5% flipping probability for each qubit
Real QC devices (5 qubits)	ibmq_ourense ibmq_vigo ibmq_london ibmq_burlington ibmq_essex
Experiment date	April/May 2020

The noise model we use for QC simulation includes two parts: *gate errors* and *readout errors*. Gate errors indicate the error that occurs when a quantum gate is applied to QC. The gate error we insert into QC simulator is *depolarizing error*, where the error rate p = 0.001 for single-qubit gate and p = 0.01 for two-qubit gate. The readout error rate we use in this work is 1.5%, which is the probability that a certain output can be inverted.

The real QC device we use to verify our work is the open-access IBM Quantum systems. We use all the 5-qubit QC devices of T-topology for our experiments [19]: *ibmq_ourense*, *ibmq_vigo*, *ibmq_london*, *ibmq_burlington*, *ibmq_essex*. Because sometimes IBM calibrates the QC devices, we run similar experiments all at once to avoid the influence of calibration. Our data were run in April/May 2020.

B. *Test Generators Setup*

This section will introduce the parameters for the test generators, including RB and our qATG. For both methods, we set α = 0.01 and β = 0.001 to demonstrate our results. The fault models parameters are as following: ratio fault with ratio 0.9, bias fault with bias 0.1π, truncation fault with threshold 1.5π. The *CNOT* fault model parameters are ($[\theta', \phi', \lambda', \theta'', \phi'', \lambda'']$):

f_1 = [0.05π, 0.05π, 0.05π, 0.05π, 0.05π, 0.05π],
f_2 = [0.05π, 0.05π, -0.05π, 0.05π, 0.05π, -0.05π],
f_3 = [0.05π, -0.05π, 0.05π, 0.05π, -0.05π, 0.05π],
f_4 = [0.05π, -0.05π, -0.05π, 0.05π, -0.05π, -0.05π],
f_5 = [-0.05π, 0.05π, 0.05π, -0.05π, 0.05π, 0.05π],
f_6 = [-0.05π, 0.05π, -0.05π, -0.05π, 0.05π, -0.05π],
f_7 = [-0.05π, -0.05π, 0.05π, -0.05π, -0.05π, 0.05π],
f_8 = [-0.05π, -0.05π, -0.05π, -0.05π, -0.05π, -0.05π].

We follow the RB method proposed in [14]. To run RB tests, we need to decide three parameters: a list of test lengths, number of random sequences for each test length (N_{seed}), and test repetition for each random sequence (R).

We explain how to decide the parameters for RB tests. Since the parameters for testing QC are different from those for estimating *EPC*, parameters are decided by the fault model and noise model. We specified test lengths as follows: for single-qubit RB, we choose test lengths: 1, 20, 50, 100, 150, 200, 300, 400, 500, 600, 700, 800, 900, 1000, 1100, 1200, 1300, 1400, 1500, 1600, 1700, 1800, 1900, 2000. For two-qubit RB, we choose test lengths: 1, 10, 20, 50, 75, 100, 125, 150, 175, 200. Test lengths for single-qubit RB is larger than those for two-qubit RB because the *EPC* for the former is usually smaller than that for the latter. Single-qubit RB has only single-qubit gates but two-qubit RB has both single and two-qubit gates. Note that, compared with conventional RB, we increase L and N_{seed} to make gate-dependent faults measureable. We perform experiments to decide optimal parameters R and N_{seed}. Figure 11 shows the relation between ΔEPC and the test repetition R. Results show that N_{seed} is a dominant factor for ΔEPC. Because [20] recommended at least 100 samples for analysis, we fix test repetition to 128 for each sequence for the rest of experiments.

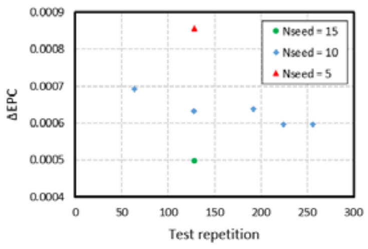


Figure 11: relation between ΔEPC and R

For our proposed qATG, there are two parameters needed to be determined: L_{MAX} and ω_{MAX}. These parameters can be set according to the expected noise of CUT. In Figure 12, we cascade many *CNOT* gates on the same qubit. Simulation results show that the probability of correct output on a two-qubit QC is below 30% after 40 *CNOT* gates. Hence, we set L_{MAX} to 40 according to the noise level of IBM Q. After L_{MAX} is decided, we generate a test template with L_{MAX}. We simulate the test template with different error rate. Figure 13 shows the relation between effect size and error rate. The vertical axis is effect size between ideal QC (without noise) and good QC (with noise). The horizontal axis is the corresponding error rate of *CNOT*. Because the error rate of *CNOT* on IBM Q ranges from 0 to 0.05, the maximum effect size for noise is 1.7. Therefore, ω_{MAX} for qATG is set to 5 to insure effect size of the fault is at least 3 times larger than that of noise.

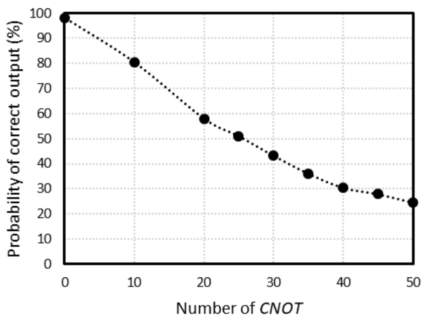


Figure 12: impact of noise on IBM Q

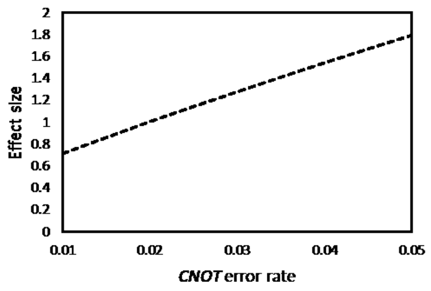


Figure 13: relation between effect size and error rate

C. Simulation Results

The fault coverage and total test time (*TT*) for three different methods (RB, QV, qATG) are summarized in Table 2. For single-qubit faults, RB and QV have 67% and 0% fault coverage, respectively. Our proposed qATG has 100% fault coverage. Please note that none of the fault that we insert into the QC is detected by QV because the fault we insert is not large enough to change the QV value. For single-qubit gates, *TT* of RB is over 23M T_{op} plus 27 K T_{im}. *TT* of QV is over 480K T_{op} plus 19 K T_{im}. Please note that QV only has one test so it only has one *TT* value. *TT* of qATG is only 5K T_{op} plus 211 T_{im}. Our qATG has much lower *TT* and higher fault coverage for the single-qubit gate.

For two-qubit faults, RB also has 100% fault coverage because only one kind of two-qubit gate (*CNOT*) in the system. The randomly generated RB tests have many *CNOT* gates. Therefore, the two-qubit faults can be detected easily. However, *TT* of our qATG is still shorter than that of RB. We can conclude from the Table 2 that qATG has lower *TT* and higher fault coverage than conventional RB and QV.

Table 2: fault coverage and *TT* for three different test methods

	RB		QV		Our qATG	
Fault model	Fault coverage	*TT*	Fault coverage	*TT*	Fault coverage	*TT*
Single qubit (90 faults)	67%	23,482,874 T_{op} + 27,648 T_{im}	0%	480,000 T_{op} + 19,200 T_{im}	100%	5,188 T_{op} + 211 T_{im}
Two qubits (64 faults)	100%	9,363,786 T_{op} + 24,960 T_{im}	0%		100%	18,144 T_{op} + 864 T_{im}

Figure 14 shows the relation between test escape and *TT* for qATG and RB. The blue line (left axes) shows the data for qATG. The red line (right axes) shows the data for RB test. The higher the required test quality, the higher the *TT*. T_{im} and T_{op} for both methods have a similar trend. We can conclude that *TT* of qATG are orders of magnitude lower than RB test for all test escape in the range from 0.001 to 0.2.

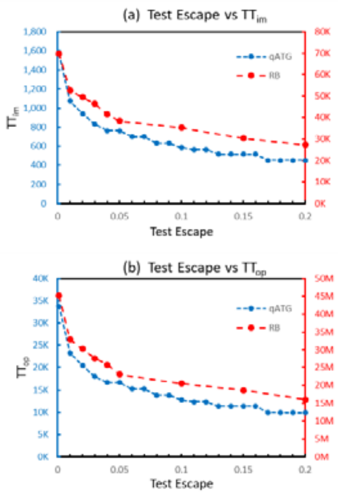


Figure 14: relation between test escape and *TT*

D. *Testing Real QC*

Table 3 shows results on IBM Q real QC for 1,000 times. Due to the limited availability of real QC, we only choose test configurations for a single-qubit fault and a two-qubit fault on the real device. The table shows that the test escape is low for both methods. The single-qubit fault we choose is $f_{\theta r}$ on U_3 gates and the two-qubit fault is f_1 on *CNOT* gates. The test configuration for $f_{\theta r}$ on U_3 gates generated by our qATG is shown in Figure 7. The real test escape is smaller than our target α because we set a lower bound on the test repetition to make the statistical process effective (see the chi-square section). Because the test escapes of the two methods are both low and very close, we believe that the cause of test escape comes from the variation of the real device rather than the test effectiveness of two methods.

Table 3: test escape for testing faulty QC on IBM Q

$\alpha = 0.01$	Method	Fail	Pass (test escape)
Test configurations for $f_{\theta r}$ @ U_3	RB	999 (99.9%)	1 (0.1%)
	qATG	1,000 (100%)	0 (0%)
Test configurations for f_1 @ *CNOT*	RB	1,000 (100%)	0 (0%)
	qATG	998 (99.8%)	2 (0.2%)

Table 4 shows the overkill for testing good QC on IBM Q for 1,000 times. For the single-qubit fault, RB has 141 overkills and qATG has none. For the two-qubit fault, RB has 708 overkills and qATG has 12. Overkill for RB testing is high because the variation of intrinsic error rate on IBM Q is in the same order as the error rate induced by fault. Moreover, qATG analyzes the test results by comparing the OPD of CUT with OPD of faulty QC, while RB uses error rate to distinguish good and faulty QC. Hence, the variation of intrinsic error rate does not affect qATG so much as RB. Please note that the overkill of *CNOT* gate is much larger than that of U_3 because the variation of the error rate for *CNOT* is larger than that of U_3. From the table, we can conclude that our method has lower overkill compared to the RB test.

Table 4: overkill for testing fault-free QC on IBM Q

$\beta = 0.001$	Method	Pass	Fail (overkill)
Test configurations for $f_{\theta r}$ @ U_3	RB	859 (85.9%)	141 (14.1%)
	qATG	1,000 (100%)	0 (0%)
Test configurations for f_1 @ *CNOT*	RB	292 (29.2%)	708 (70.8%)
	qATG	988 (98.8%)	12 (1.2%)

Table 5 shows test escape for qATG on many faulty QCs with different fault models. We test each fault model for 1,000 times on IBM Q. *L* and *R* are shown for each fault models. Most *R* are the same as the lower bound. This shows that our test configuration is effective to distinguish the good QC from the good QC. Results show that the test escapes range from 0 to 0.9% for different fault models.

Table 5: test escape for different fault models on IBM Q (each 1,000 times)

Fault model	$f_{\theta r}$ @ U_3	$f_{\theta b}$ @ U_3	$f_{\theta t}$ @ U_3	$f_{\phi r}$ @ U_3	$f_{\phi b}$ @ U_3	$f_{\phi t}$ @ U_3	$f_{\lambda r}$ @ U_3	$f_{\lambda b}$ @ U_3	$f_{\lambda t}$ @ U_3
L, R	*L=8, R=11*	*L=42, R=11*	*L=42, R=11*	*L=8, R=11*	*L=42, R=11*	*L=42, R=11*	*L=4, R=11*	*L=10, R=11*	*L=10, R=11*
Test escape	0%	0%	0%	0%	0%	0.3%	0.3%	0%	0%
Fault model	$f_{\phi r}$ @ U_2	$f_{\phi b}$ @ U_2	$f_{\phi t}$ @ U_2	$f_{\lambda r}$ @ U_2	$f_{\lambda b}$ @ U_2	$f_{\lambda t}$ @ U_2	$f_{\lambda r}$ @ U_1	$f_{\lambda b}$ @ U_1	$f_{\lambda t}$ @ U_1
L, R	*L=42, R=11*	*L=8, R=11*	*L=42, R=17*	*L=16, R=11*	*L=8, R=11*	*L=4, R=11*	*L=42, R=11*	*L=42, R=18*	*L=10, R=11*
Test escape	0%	0.2%	0%	0%	0%	0%	0%	0.2%	0%
Fault model	f_1 @ *CNOT*	f_2 @ *CNOT*	f_3 @ *CNOT*	f_4 @ *CNOT*	f_5@ *CNOT*	f_6 @ *CNOT*	f_7 @ *CNOT*	f_8 @ *CNOT*	
L, R	*L=8, R=11*	*L=8, R=11*	*L=8, R=11*	*L=8, R=11*	*L=8, R=11*	*L=8, R=11*	*L=8, R=11*	*L=8, R=11*	
Test escape	0.2%	0.9%	0.3%	0.5%	0.1%	0.9%	0.5%	0.3%	

Test escape variation is larger than expected because the test escape should be independent of fault model. Hence, we rerun some fault models which has unexpectedly high test escape (f_2 and f_6) for 1,000 times. New test escape become 0.1% and 0.3%, respectively. These numbers are very different from the first experiment. Hence, we conclude that the test escapes difference mainly come from the variation of IBM Q. In spite of the above variation problem, all test escapes are less than the given value α=1%. Experimental data shows that our qATG can effectively detect 26 fault models on the real QC.

We run all the test configurations on IBM Q (fault-free) for 1,000 times. Totally 84 times of test fails. Hence, the overkill of running our test on IBM Q has an overkill of about 8.4%. Because the variation of intrinsic error rate of IBM Q can be large, the overkill is inevitable. Experimental results show that the overkill of our qATG is acceptable given the large variation on real QC.

VI. Conclusion

In summary, this paper proposes 26 behavior fault models based on the function of quantum gates. The number of faults is polynomial, $3xn+ym$, to the size of QC. We propose a qATG algorithm that uses gradient descent to generate test configurations with short length. We revise the chi-square statistical method to decide whether the CUT pass or fail the test. The test repetition can be determined under the specified test escape and overkill. Experimental results on IBM Q systems show that test escape of our qATG is lower than 1% and overkill is less than 8.4%. Our test lengths are 1,000X shorter than two traditional RB and QV test methods. If we keep the fault model and basic gate set unchanged, the *TT* of qATG is linear to QC size, $O(n+m)$, where n is the number of qubits and m is the number of connections.

Acknowledgment

We acknowledge the IBM Q systems for this work. The views expressed are those of the authors and do not reflect the official policy or position of IBM or the IBM Q team. This work is sponsored by MOST, project number: MOST 109-2221-E-002-150-.

References

[1] Frank Arute, et al. "Quantum supremacy using a programmable superconducting processor." *Nature* 574.7779 (2019): 505-510.

[2] Jeremy Hsu, "CES 2018: Intel's 49-Qubit Chip Shoots for Quantum Supremacy." *IEEE Spectrum*, Jan 09, 2018

[3] Will Knight, "IBM Raises the Bar with a 50-Qubit Quantum Computer." *MIT Technology Review*, Nov 10, 2017

[4] Emanuel Knill. "Quantum computing with realistically noisy devices." *Nature* 434.7029 (2005): 39.

[5] Michael A. Nielsen, and Isaac Chuang. "Quantum computation and quantum information." (2002): 558-559.

[6] Isaac L. Chuang, and Michael A. Nielsen. "Prescription for experimental determination of the dynamics of a quantum black box," *Journal of Modern Optics* 44.11-12 (1997): 2455-2467.

[7] John P. Hayes, Ilia Polian, and Bernd Becker. "Testing for missing-gate faults in reversible circuits," IEEE *13th Asian Test Symposium*, 2004.

[8] Robert Wille, Hongyan Zhang, and Rolf Drechsler, "ATPG for reversible circuits using simulation, Boolean satisfiability, and pseudo Boolean optimization," IEEE *Computer Society Annual Symposium on VLSI*, 2011.

[9] Matthew McKague, Tzyh Haur Yang, and Valerio Scarani, "Robust self-testing of the singlet," *Journal of Physics A: Mathematical and Theoretical* 45.45 (2012): 455304.

[10] Ivan Šupić, *et al.*, "Self-testing protocols based on the chained Bell inequalities," *New Journal of Physics* 18.3 (2016): 035013.

[11] Rui Chao, *et al.*, "Test for a large amount of entanglement, using few measurements," *Quantum* 2 (2018): 92.

[12] Karl Pearson. "X. On the criterion that a given system of deviations from the probable in the case of a correlated system of variables is such that it can be reasonably supposed to have arisen from random sampling." *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science* 50.302 (1900): 157-175.

[13] Emanuel Knill, *et al.*, "Randomized benchmarking of quantum gates," *Physical Review A* 77.1 (2008): 012307.

[14] Easwar Magesan, Jay M. Gambetta, and Joseph Emerson. "Characterizing quantum gates via randomized benchmarking." *Physical Review A* 85.4 (2012): 042311.

[15] Héctor Abraham, et al. "Qiskit: An Open-source Framework for Quantum Computing(2019)."

[16] Andrew W. Cross, et al. "Validating quantum computers using randomized model circuits." *Physical Review A* 100.3 (2019): 032328.

[17] Andrew F. Siegel. "The noncentral chi-squared distribution with zero degrees of freedom and testing for uniformity." *Biometrika* 66.2 (1979): 381-386.

[18] Jacob Cohen. *Statistical power analysis for the behavioral sciences*. Academic press, 2013.

[19] 5-qubit backend: IBM Q team, "IBM Q Burlington backend specification V1.1.4, IBM Q Essex backend specification V1.0.1, IBM Q London backend specification V1.1.0, IBM Q Ourense backend specification V1.0.1, IBM Q Vigo backend specification V1.0.2," (2020). Retrieved from https://quantum-computing.ibm.com.

[20] Norm O'Rourke, and Larry Hatcher. A step-by-step approach to using SAS for factor analysis and structural equation modeling. Sas Institute, 2013.