Quantum Circuits for Dynamic Runtime Assertions in Quantum Computation

Ji Liu
jliu45@ncsu.edu
North Carolina State University
Raleigh, North Carolina

Gregory T. Byrd
gbyrd@ncsu.edu
North Carolina State University
Raleigh, North Carolina

Huiyang Zhou
hzhou@ncsu.edu
North Carolina State University
Raleigh, North Carolina

Abstract

In this paper, we propose quantum circuits for runtime assertions, which can be used for both software debugging and error detection. Runtime assertion is challenging in quantum computing for two key reasons. First, a quantum bit (qubit) cannot be copied, which is known as the non-cloning theorem. Second, when a qubit is measured, its superposition state collapses into a classical state, losing the inherent parallel information. In this paper, we overcome these challenges with runtime computation through ancilla qubits, which are used to indirectly collect the information of the qubits of interest. We design quantum circuits to assert classical states, entanglement, and superposition states. Our experimental results show that they are effective in debugging as well as improving the success rate for various quantum algorithms on IBM Q quantum computers.

CCS Concepts. • **Hardware → Quantum technologies**.
Keywords. Quantum Computing; Runtime Assertion

ACM Reference Format:
Ji Liu, Gregory T. Byrd, and Huiyang Zhou. 2020. Quantum Circuits for Dynamic Runtime Assertions in Quantum Computation. In *Proceedings of the Twenty-Fifth International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS '20), March 16–20, 2020, Lausanne, Switzerland*. ACM, New York, NY, USA, 14 pages. https://doi.org/10.1145/3373376.3378488

1 Introduction

Quantum computing features unique advantages over classical computing and recent advances in quantum computer hardware raise high hopes to realize the remarkable potential of quantum computing. However, there are important hurdles to overcome to make quantum computing mainstream. The first is the difficulty of developing and debugging quantum programs. The second is that the quantum computers are highly susceptible to errors and quantum error correction incurs very high overhead. To tackle the first problem, prior work by Huang et al. [17] shows that many bugs in quantum programs can be detected using assertions. Assertions, especially dynamic ones, during quantum program execution are challenging for two key reasons. The first is the non-cloning theorem, which means that it is impossible to copy a quantum bit (qubit) with an arbitrary state. The second is that any measurement on a qubit in a superposition state will project it into a classical state. As a result, in a recent work by Huang et al. [18], statistical assertions, meaning statistical analysis on multiple measurement results, are proposed to debug quantum programs. The key limitation of this approach is that each measurement stops program execution and the assertions require ensembles of runs when the actual computation results are to be measured.

In this paper, we propose quantum circuits to overcome this limitation and to enable dynamic assertions for quantum programs. We also propose to leverage assertion for opportunistic error detection such that we can increase the success rate of the quantum computer without error correction. Our proposed quantum circuits for dynamic assertions are inspired from quantum error correction and nondestructive discrimination. As qubits cannot be copied and cannot be measured directly, our approach for dynamic assertions is to indirectly verify the desired condition to be checked. In comparison, quantum error correction shares the same constraints and the various previously proposed quantum error correction codes [13][25] introduce ancilla qubits and encode the information of the qubits to be protected in the ancilla qubits, which are checked and used to correct the qubits if they are corrupted. Similarly, we also introduce ancilla qubits for assertions but the difference is that we only need to check for assertions and our proposed quantum circuits for assertions are much simpler than those for error correction, which incurs very high overhead in the number of ancilla qubits and the associated quantum circuits. Nondestructive discrimination (NDD) is mainly used in secure quantum communication. Several NDD protocols [14, 15, 19] have been proposed to discriminate entangled states such as Bell states in quantum information processing. We can leverage these NDD protocols to assert the target quantum states

ASPLOS '20, March 16–20, 2020, Lausanne, Switzerland

ACM ISBN 978-1-4503-7102-5/20/03…$15.00
https://doi.org/10.1145/3373376.3378488

without disturbing the qubits under test while our proposed quantum circuits are simpler than the NDD protocols as we do not need to discriminate all the entangled states.

According to the previous work by Huang et al. [18], three types of possible assertions are essential for debugging quantum programs: classical assertions, superposition assertions, and entanglement assertions. Classical assertions check quantum variables with classical values to see whether they match the desired ones; superposition assertions check whether a quantum variable is in a desired superposition state; and entanglement assertions check whether the entangled quantum variables exhibit the desired correlation. In this paper, we propose circuits for these three types of assertions. In addition, we also enable superposition assertion with a desired phase, which cannot be asserted with the statistical approach. Besides debugging, we show how these assertion circuits can be used to improve success rate via post measurement selection, meaning that we ignore the measurement results which fail the assertion checks. In our evaluation, we perform experiments on an actual quantum computer, IBM Q, to show the effectiveness of our proposed schemes.

The key novelty of this work is (a) quantum circuits for dynamic assertions, which are used as primitives for quantum program debugging, (b) the use of the circuits for opportunistic error detection so as to improve success rates on noisy intermediate scale quantum (NISQ) systems, (c) an analysis of the impact from our proposed assertion circuits on the circuits under test, and (d) a detailed evaluation on a variety of quantum algorithms.

2 Background and Related Work

Qubits are the foundation of quantum computing. Executing a quantum program means performing a sequence of operations upon the qubits. A qubit can be in a classical state, i.e., the $|0\rangle$ state or $|1\rangle$ state, which can be viewed as the classical 0 or 1 states. Besides classical states, a qubit can be in a superposition state, which is a linear combination of classical states, i.e., $|\Psi\rangle = a|0\rangle + b|1\rangle$ where a and b are complex numbers and $|a|^2 + |b|^2 = 1$. When a qubit in the superposition state is measured, the superposition state is projected into a classical state with the probability of $|a|^2$ being state $|0\rangle$ and $|b|^2$ being state $|1\rangle$. Superposition states are the reason for quantum parallelism, as n qubits can be in a mixture of 2^n states while in classical computing an n-bit variable takes one of the 2^n states at a time.

The state of multiple qubits can be described as the tensor product between the individual qubit state vectors. For example, the state of the two qubit, $|\Psi\rangle = a|0\rangle + b|1\rangle$ and $|\delta\rangle = c|0\rangle + d|1\rangle$, can be described as $|\Psi\rangle \otimes |\delta\rangle = ac|00\rangle + ad|01\rangle + bc|10\rangle + bd|11\rangle$, where $|00\rangle$ is $|0\rangle \otimes |0\rangle$, $|01\rangle$ is $|0\rangle \otimes |1\rangle$, etc. Two or more qubits can be entangled, meaning that their measurements results will be correlated and their state cannot be expressed as the tensor product of individual qubits. One implication is that among the entangled qubits, if one of them is measured (i.e., projected to a classical state), the rest will also collapse into a compatible state, losing some or all of their superposition states.

A quantum program is a sequence of quantum operations (gates) performed upon a collection of qubits. There are single-qubit gates such as Hadamard (H) gate, phase (S) gate, and the Pauli gates (X, Y, and Z), and multi-qubit gates such as controlled-NOT (CNOT) gate. Barenco et al. [5] has proved that single-qubit gates and CNOT gates are universal for quantum computation. As we mainly use H gates and CNOT gates in this paper, we present their logic relationship in Figure 1.

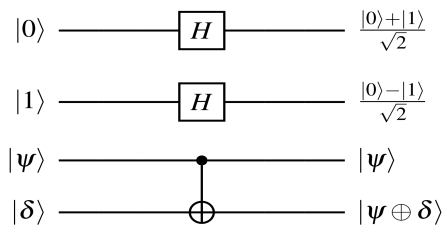

Figure 1. Logic functions of Hadamard gate and CNOT gate.

Both superposition and entanglement are used extensively in quantum programs, and they are the fundamental reason for the computational advantage of quantum computing over classical computing. However, they do not have correspondence in classical computing, which makes them hard to reason about. The development of quantum programs remains a difficult task and debugging them is also very challenging. In the prior work, Huang et al.[18] analyzed a set of quantum programs and identified that the following three types of assertions are needed in quantum programs: assertions for classical values, assertions for superposition states, and assertions for entangled states. They proposed a statistical approach to realize these assertions by measuring the qubits many times. The limitation is that each measurement collapses the superposition state and projects the entangled qubits. As a result, such measurements stop the execution of the quantum program. When the execution is performed and the results are measured, such intermediate assertions could not be enforced. Another limitation is that direct measurement results cannot reveal the phase information of qubits.

Quantum property testing studies [8][24] aim to design algorithms that can distinguish whether a large object has a certain property or not. Aharonov et al. [2] introduced a technique for testing whether a shared bipartite quantum state is the maximally entangled state. Harrow et al. [16] proposed a test that can distinguish the product states and the states that are far from the product states. The difference

between quantum property testings and our proposed assertion circuit is that quantum property testings usually handle very large objects and they do not require preserving the state after the test. For the purpose of assertion, our circuit should measure the states nondestructively.

Nondestructive discrimination (NDD) plays an important role in a number of quantum computational protocols such as entanglement concentration protocols [4] and secret quantum conversation [19]. It is used to discriminate entangled states without destroying the information. Gupta et al. [14] proposed nondestructive measurement scheme for Bell states. Mitali et al. [30] experimentally realised Gupta et al.'s scheme on a 5-qubit quantum computer. Manu et al. [22] describes an algorithm for arbitrary set of orthogonal quantum states nondestructive discrimination based on phase estimation. Satyajit et al. [28] defined a new set of highly entangled orthogonal states as Z-states and demonstrate their discrimination using IBM's 5-qubit quantum computer. Our dynamic assertion circuits are related to NDD as our circuits also nondestructively measure the qubits under test. The difference lies in what information we assert, which determines the complexity of the quantum circuits, and how we use them for. In a sense, we repurpose NDD for program debugging and error detection, although our proposed circuit is not the same as (usually simpler than) the NDD circuits.

3 Quantum Circuits for Dynamic Assertions

Our key approach to enabling dynamic assertion is to introduce additional quantum bits, aka ancilla qubits, to collect information about the qubits under test, and to measure the ancilla qubits rather than directly measuring the qubits under test. This way, we do not need to disrupt the program execution while the assertion is checked. However, care needs to be taken to ensure that measuring the ancilla qubits will not affect the original quantum circuit. Next, we describe our proposed circuits for each type of assertion. For all the circuits, a measurement of the ancilla qubit being $|1\rangle$ means an assertion error. This rule is helpful as it alleviates the decoherence and readout error in assertion circuits. When the assertion circuits are used on real quantum computers to improve success rates, we check to ensure no additional SWAP gates are inserted due to our assertion circuits.

3.1 Dynamic Assertion for Classical Values

To ensure that the qubits are initialized to the correct values or some intermediate classical results should satisfy some conditions such as ($|\Psi\rangle == |0\rangle$)), we can resort to assertions for classical values. We propose to introduce one ancilla qubit and a CNOT gate to achieve classical-value assertion for one qubit, as shown in Figure 2. In the figure, the qubit $|\Psi\rangle$ is to be checked for ($|\Psi\rangle == |0\rangle$). The ancilla qubit is initialized to $|0\rangle$ and measured after the CNOT gate. If we initialize the ancilla qubit to be $|1\rangle$, the same circuit asserts ($|\Psi\rangle == |1\rangle$).

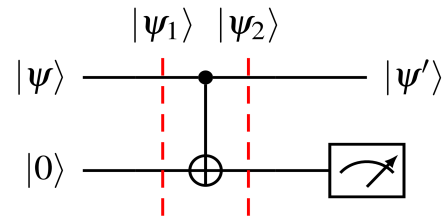

Figure 2. Circuit for asserting classical values ($|\Psi\rangle == |0\rangle$).

Proof. In Figure 2, the state $|\Psi_1\rangle = |\Psi\rangle \otimes |0\rangle$.

The state after the CNOT gate $|\Psi_2\rangle = |\Psi\rangle \otimes |\Psi \oplus 0\rangle = |\Psi\rangle \otimes |\Psi\rangle$.

If $|\Psi\rangle$ is in a classical state, either $|0\rangle$ or $|1\rangle$, then $|\Psi_1\rangle$ is either $|00\rangle$ or $|10\rangle$ and $|\Psi_2\rangle$ is either $|00\rangle$ or $|11\rangle$, correspondingly. As a result, when the ancilla qubit is measured, if it is $|0\rangle$, it means that $|\Psi\rangle$ must be $|0\rangle$; if it is $|1\rangle$, $|\Psi\rangle$ must be $|1\rangle$, i.e., an assertion error.

If the qubit $|\Psi\rangle$ is in a superposition state due to a bug or a runtime error, $|\Psi\rangle = a|0\rangle + b|1\rangle$. $|\Psi_1\rangle$ is $a|00\rangle + b|10\rangle$ and $|\Psi_2\rangle$ becomes $a|00\rangle + b|11\rangle$, which is an entangled state. Due to such entanglement, after the measurement of the ancilla qubit, if the measurement result is $|0\rangle$ (i.e., no assertion error), the qubit under test will be projected into the classical state $|0\rangle$, i.e., $|\Psi'\rangle = |0\rangle$. If the measurement result is $|1\rangle$ (i.e., an assertion error), it is projected into the classical state $|1\rangle$. It means that when we perform an assertion check ($|\Psi\rangle == |0\rangle$), if there is no assertion error, the proposed circuit may have automatically corrected the qubit if it is in a superposition state. If it cannot correct the qubit into the expected classical state, an assertion error occurs. Since the probability of the measurement result being $|0\rangle$ and $|1\rangle$ is $|a|^2$ and $|b|^2$, respectively, the probability distribution of assertion errors over repeated runs (and measurements) can be used to estimate a and b, if needed.

3.2 Dynamic Assertion for Entanglement

To assert that two or more qubits are in the entangled state of $a|00\rangle + b|11\rangle$ or $a|01\rangle + b|10\rangle$, we propose to leverage parity computation. Figure 3 shows the proposed circuit for computing the parity of two qubits. If checking whether the two qubits are entangled in the state of $a|00\rangle + b|11\rangle$, the ancilla qubit is initialized to $|0\rangle$. If asserting that the two qubits are in the state of $a|01\rangle + b|10\rangle$, the ancilla should be initialized to $|1\rangle$.

Proof. In Figure 3, if the input qubits are in the state of $a|00\rangle + b|11\rangle$, i.e., $|\Psi\rangle = a|00\rangle + b|11\rangle$.

Then, the state $|\Psi_1\rangle = (a|00\rangle + b|11\rangle) \otimes |0\rangle = a|000\rangle + b|110\rangle$.

The state $|\Psi_2\rangle = (a|000\rangle + b|111\rangle)$, i.e., the ancilla qubit is entangled as well after the CNOT gate.

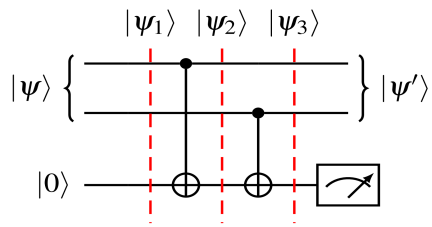

Figure 3. Circuit for asserting entanglement.

The state $|\Psi_3\rangle = a|000\rangle + b|110\rangle = (a|00\rangle + b|11\rangle) \otimes |0\rangle$, which means that the ancilla qubit is un-entangled from the two qubits under test and should be $|0\rangle$. The qubits state $|\Psi\rangle$ is unaffected for subsequent computations.

If the input qubits are not entangled as expected, i.e., $|\Psi\rangle = a|00\rangle + b|11\rangle + c|10\rangle + d|01\rangle$, then $|\Psi_1\rangle = a|000\rangle + b|110\rangle + c|100\rangle + d|010\rangle$, $|\Psi_2\rangle = a|000\rangle + b|111\rangle + c|101\rangle + d|010\rangle$, and $|\Psi_3\rangle = a|000\rangle + b|110\rangle + c|101\rangle + d|011\rangle$. When measuring the ancilla qubit, the result can be either $|0\rangle$ or $|1\rangle$. If $|0\rangle$, $|\Psi_3\rangle$ is projected to $a'|000\rangle + b'|110\rangle = (a'|00\rangle + b'|11\rangle) \otimes |0\rangle$, i.e., the input qubits are forced into the desired entangled state. If $|1\rangle$, $|\Psi_3\rangle$ is projected to $c'|101\rangle + d'|011\rangle = (c'|10\rangle + d'|01\rangle) \otimes |1\rangle$, i.e., another entangled state, while the assertion error is reported. The probability of measurement results being $|0\rangle$ or $|1\rangle$ can be used to compute the coefficients a, b, c, d, if needed. This parity circuit is also a subset of the circuits for Bell state NDD [15]. Note that our parity-based assertion circuit can be scaled to assert more than two qubits. The assertion circuit can assert multi-qubit entanglement states with odd or even numbers of ones. For example, the circuit in Figure 4 asserts for 3-qubit entanglement state with even number of ones: $|\psi\rangle = a\,|000\rangle + b\,|011\rangle + c\,|101\rangle + d\,|110\rangle$.

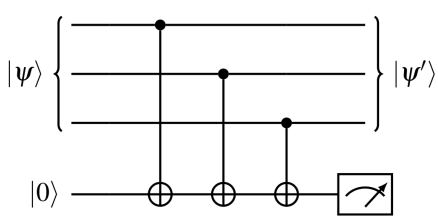

Figure 4. Circuit for asserting three qubits are entangled.

3.3 Dynamic Assertion for Superposition

Superposition is a linear combination of classical states. In Section 3.3.1, we propose our assertion circuit for uniform superposition state. In Section 3.3.2, we introduce assertion circuits for arbitrary superposition states.

3.3.1 Dynamic Assertion for Uniform Superposition State. In quantum computing, it is common to use Hadamard gates to set the input qubits in the uniform superposition state, $|+\rangle = \frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle$, in order to take advantage of quantum parallelism. To assert such operations are correctly performed, we propose the circuit as shown in Figure 5.

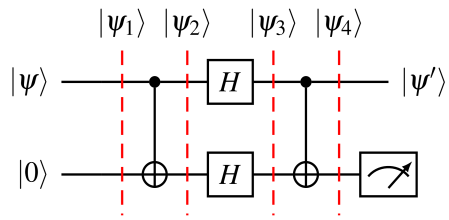

Figure 5. Circuit for asserting uniform superposition.

Proof. In Figure 5, the state $|\Psi\rangle = a|0\rangle + b|1\rangle$. If it is in the uniform superposition state, i.e., $|\Psi\rangle = |+\rangle$ or $a = b = \frac{1}{\sqrt{2}}$, the state $|\Psi_1\rangle = (a|0\rangle + b|1\rangle) \otimes |0\rangle = a|00\rangle + b|10\rangle$, and $|\Psi_2\rangle = a|00\rangle + b|11\rangle$.

The state $|\Psi_3\rangle = a\frac{|0\rangle+|1\rangle}{\sqrt{2}} \otimes \frac{|0\rangle+|1\rangle}{\sqrt{2}} + b\frac{|0\rangle-|1\rangle}{\sqrt{2}} \otimes \frac{|0\rangle-|1\rangle}{\sqrt{2}} = \frac{1}{2}[a(|00\rangle + |01\rangle + |10\rangle + |11\rangle) + b(|00\rangle - |01\rangle - |10\rangle + |11\rangle)]$.

The state $|\Psi_4\rangle = \frac{1}{2}[a(|00\rangle+|01\rangle+|11\rangle+|10\rangle)+b(|00\rangle-|01\rangle-|11\rangle+|10\rangle)] = \frac{1}{2}[(a+b)|00\rangle + (a-b)|01\rangle + (a+b)|10\rangle + (a-b)|11\rangle] = \frac{1}{2}\{|0\rangle \otimes [(a+b)\,|0\rangle + (a-b)\,|1\rangle] + |1\rangle \otimes [(a+b)\,|0\rangle + (a-b)\,|1\rangle]\} = \frac{1}{2}\{(|0\rangle + |1\rangle) \otimes [(a+b)\,|0\rangle + (a-b)\,|1\rangle]\} = \frac{1}{\sqrt{2}}\{|+\rangle \otimes [(a+b)\,|0\rangle + (a-b)\,|1\rangle]\}$. Therefore, after the assertion circuit, the qubit under test is always in the $|+\rangle$ state and the ancilla qubit is un-entangled from it. The subsequent computation is not affected by the measurement of the ancilla qubit.

If $|\Psi\rangle = |+\rangle$ or $a = b = \frac{1}{\sqrt{2}}$, then $|\Psi_4\rangle = |+\rangle \otimes |0\rangle$. This means that the ancilla qubit should always be $|0\rangle$, and it is un-entangled from the qubit under test.

If $|\Psi\rangle = |-\rangle$ or $a = \frac{1}{\sqrt{2}}$ and $b = -\frac{1}{\sqrt{2}}$, then $|\Psi_4\rangle = |+\rangle \otimes |1\rangle$. This means that the ancilla qubit should always be $|1\rangle$, and it is un-entangled from the qubit under test.

If $|\Psi\rangle! = |+\rangle$ or $|-\rangle$, the ancilla qubit and the qubit under test are un-entangled and we can derive the probability of the measurement result on the ancilla qubit being $|0\rangle$ or $|1\rangle$. The probability of the measurement result being $|0\rangle$ can be computed as $|a+b|^2/(|a+b|^2 + |a-b|^2) = |a+b|^2/2$. If both a and b are real, then the probability becomes $(a^2 + 2ab + b^2)/2 = (1 + 2ab)/2$. Similarly, we can derive the probability of the measurement result on the ancilla qubit being $|1\rangle$ as $|a-b|^2/(|a+b|^2 + |a-b|^2) = |a-b|^2/2$, which becomes $(a^2 - 2ab + b^2)/2 = (1 - 2ab)/2$ if both a and b are real. The probabilities of the measurement result of the ancilla qubit being $|0\rangle$ or $|1\rangle$ can be used to compute the magnitude of the original coefficients a and b. In the case of $|\Psi\rangle$ being in a classical state, i.e., $a = 0$ and $b = 1$ or $a = 1$ and $b = 0$, the measurement result on the ancilla qubit has the equal probability of 50% being $|0\rangle$ or $|1\rangle$.

3.3.2 Dynamic Assertion for Arbitrary Superposition State. In quantum computing, it is also common for a qubit

to stay in an arbitrary superposition state in the middle of computation. This state can be represented as a point on the Bloch sphere. Given an orthonormal basis $|0\rangle$ and $|1\rangle$, an arbitrary superposition state $|\Psi\rangle$ of a qubit can be written as a superposition of the basis vectors $|0\rangle$ and $|1\rangle$: $|\Psi\rangle = cos(\frac{\theta}{2})|0\rangle + e^{i\varphi}sin(\frac{\theta}{2})|1\rangle$. Here, we use the terms θ and φ instead of coefficients a and b in the previous section. Manu and Kumar [22] described an algorithm for NDD of arbitrary orthogonal quantum states. Here, we adapt this algorithm to construct circuits for asserting the superposition state $|\Phi\rangle = cos(\frac{\theta}{2})|0\rangle + e^{i\varphi}sin(\frac{\theta}{2})|1\rangle$.

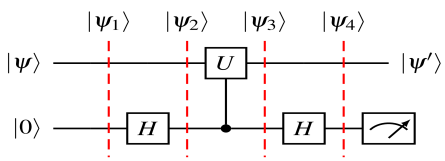

Figure 6. Circuit for asserting arbitrary superposition

Figure 6 shows the circuit to assert a qubit with an arbitrary superposition state. It consists of a controlled-U gate and two Hadamard gates. The U gate is constructed from the superposition state $|\Phi\rangle$ we want to assert.

$$U = V \times M \times V^{-1} \tag{1}$$

V is the matrix formed by column vectors, $V = [|\Phi\rangle|\Phi'\rangle]$. We assign eigenvalue 1 for $|\Phi\rangle$ and -1 for $|\Phi'\rangle$. The corresponding matrix M is

$$M = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \tag{2}$$

For the superposition state $|\Phi\rangle = cos(\frac{\theta}{2})|0\rangle + e^{i\varphi}sin(\frac{\theta}{2})|1\rangle$ that we want to assert, its orthogonal state is its antipodal point on Bloch sphere: $|\Phi'\rangle = sin(\frac{\theta}{2})|0\rangle - e^{i\varphi}cos(\frac{\theta}{2})|1\rangle$. The V matrix is:

$$V = \begin{bmatrix} cos(\frac{\theta}{2}) & sin(\frac{\theta}{2}) \\ e^{i\varphi}sin(\frac{\theta}{2}) & -e^{i\varphi}cos(\frac{\theta}{2}) \end{bmatrix} \tag{3}$$

and V^{-1} is

$$V^{-1} = \begin{bmatrix} cos(\frac{\theta}{2}) & e^{-i\varphi}sin(\frac{\theta}{2}) \\ sin(\frac{\theta}{2}) & -e^{-i\varphi}cos(\frac{\theta}{2}) \end{bmatrix} \tag{4}$$

then

$$U = \begin{bmatrix} cos(\theta) & e^{-i\varphi}sin(\theta) \\ e^{i\varphi}sin(\theta) & -cos(\theta) \end{bmatrix} \tag{5}$$

Proof. In Figure 6, $|\Psi\rangle = a|0\rangle + b|1\rangle$, the state $|\Psi_1\rangle = |\Psi\rangle \otimes |0\rangle$ and $|\Psi_2\rangle = |\Psi\rangle \otimes \frac{|0\rangle+|1\rangle}{\sqrt{2}}$.

After the controlled-U gate, the state $|\Psi_3\rangle = \frac{1}{\sqrt{2}}[|\Psi\rangle \otimes |0\rangle + U|\Psi\rangle \otimes |1\rangle]$.

If the input state $|\Psi\rangle$ is the superposition state $|\Phi\rangle$ that we want to assert, it is an eigenvector of the unitary matrix U and the corresponding eigenvalue is 1, i.e., $U|\Phi\rangle = |\Phi\rangle$. Therefore, $|\Psi_3\rangle = \frac{1}{\sqrt{2}}[|\Phi\rangle \otimes |0\rangle + |\Phi\rangle \otimes |1\rangle] = |\Phi\rangle \otimes \frac{|0\rangle+|1\rangle}{\sqrt{2}}$. Afte another H gate, the state $|\Psi_4\rangle = |\Phi\rangle \otimes |0\rangle$, which means that the ancilla qubit is un-entangled from the qubit under test and should be $|0\rangle$.

If the state $|\Psi\rangle$ is in the orthogonal state, i.e., $|\Phi'\rangle$, as its corresponding eigenvalue is -1, the state $|\Psi_3\rangle = \frac{1}{\sqrt{2}}[|\Phi'\rangle \otimes |0\rangle - |\Phi'\rangle \otimes |1\rangle] = |\Phi'\rangle \otimes \frac{|0\rangle-|1\rangle}{\sqrt{2}}$. The state after the H gate $|\Psi_4\rangle = |\Phi'\rangle \otimes |1\rangle$, which means that the ancilla qubit is un-entangled from the qubit under test and should be $|1\rangle$.

If the state $|\Psi\rangle$ is in an arbitrary superposition state due to a bug or error, it can be expressed as $|\Psi\rangle = a'|\Phi\rangle + b'|\Phi'\rangle$ where a' and b' are complex numbers and $|a'|^2 + |b'|^2 = 1$ as the states $|\Phi\rangle$ and $|\Phi'\rangle$ form a complete basis. Based on the linearity principle, the state after assertion circuit $|\Psi_4\rangle = a'|\Phi\rangle \otimes |0\rangle + b'|\Phi'\rangle \otimes |1\rangle$. After the measurement of the ancilla qubit, if the measurement result is $|0\rangle$ (i.e., no assertion error), the qubit state $|\Psi\rangle$ is projected to $|\Phi\rangle$. If the measurement result is $|1\rangle$ (i.e., assertion error), the state is projected to $|\Phi'\rangle$. It means that this circuit can automatically correct the qubit state if there is no assertion error. And it will report an assertion error when the qubit state is not corrected. For the purpose of debugging, we can also estimate a' and b' based on the probability distribution of $|0\rangle$ and $|1\rangle$.

As a special case, to assert the uniform superposition state, i.e., $\theta = \frac{\pi}{2}$ and $\varphi = 0$, the U gate becomes

$$U = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \tag{6}$$

It is actually a NOT gate, and the controlled-U gate becomes a CNOT gate. The resulting assertion circuit is shown in Figure 7.

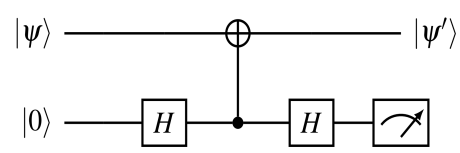

Figure 7. Circuit for asserting uniform superposition

Although the circuits in Figure 5 and Figure 7 both assert uniform superposition, the difference is that the circuit in Figure 5 always "corrects" the qubit under test regardless of the measurement outcome of the ancilla qubit. In comparison, the circuit in Figure 7 only corrects the qubit under test when the ancilla qubit is $|0\rangle$. However, when running on current quantum machines, as the CNOT gate has relatively high error rates, the latter design results in lower error rates as it requires only one CNOT gate.

3.4 Impact of Errors in Assertion Circuits

Note that the derivations in previous sections are based on the assumption that there are no errors in the assertion

circuits. This assumption is valid for quantum program debugging. On the other hand, for error detection, an error in the assertion circuits may propagate into the qubits under test due to the CNOT gates used for assertions. We use two different approaches to study this impact.

The first approach is an error model. Instead of a detailed study of quantum noise channels [21], we only consider the error probability of the gates and qubits. In this model we consider three types of errors:

1. Single qubit coherent error: The time of a qubit retaining its information is called *coherence time* and the process of losing the information is called decoherence process [29]. There are two kinds of coherence time T_1 and T_2. T_1 associates with the amplitude damping channel as it denotes the process where the high-energy state $|1\rangle$ decays to the low-energy state $|0\rangle$. In the amplitude damping channel, the qubit retains its state with a probability of $p_1(t) = e^{-t/T_1}$, where t is the time of operation that depends on the gate time. T_2 associates with the phase damping channel as it denotes the process of phase change. In the phase damping channel, the qubit retains its state with a probability of $p_2(t) = e^{-t/T_2}$. So the coherence error rate $\epsilon_{coherent}$ of a quantum gate can be expressed as

$$p(t + \Delta t) = p(t)(1 - \epsilon_{coherent}) \tag{7}$$

where Δt is the gate time. As $p(t+\Delta t) = p_1(t+\Delta t)p_2(t+\Delta t)$ and $p(t) = p_1(t)p_2(t)$,

$$\epsilon_{coherent} = 1 - e^{-\Delta t(1/T1+1/T2)}. \tag{8}$$

In current IBM quantum computers [11], the coherent time varies from 10 to 100 micro-seconds. The gate time varies from 100 to 1000 nano-seconds.

2. Gate error: Here, we consider the depolarizing error ϵ_{gate}, which can be depicted by the randomized benchmarking [20]. Randomized benchmarking measures the average gate errors by running sequences of randomly selected Clifford gates followed by the reverse gates that would return the qubits to the initial state. This method is useful as it measures the depolarization probability and does not rely on accurate state preparation and measurement. In current IBM quantum computers, the single qubit gate's error rates are approximately 10^{-3} and two-qubit CNOT gate's error rate is 10^{-2} [11].

3. State preparation and Measurement error: State preparation error ϵ_{state} happens when preparing the qubit. Measurement error $\epsilon_{measure}$ happens at the measurement of the qubit. In IBM quantum computer, the measurement error rates are between 10^{-3} and 10^{-2} [11]. The state preparation error rates of qubits are not reported.

For purpose of deriving the success probability of the system, we use the following assumptions similar to [27]:

- An error in a gate or measurement will cause the whole program to fail.
- The probability of each error are independent of each other and only depends on the corresponding qubit and gate. In other words, we ignore the crosstalk errors.

Based on the assumptions above, the success probability $P_{success}$ of a quantum circuit can be represented as:

$$P_{success} = \prod_i (1 - \epsilon_i) \tag{9}$$

where i denotes all the errors including state preparation error ϵ_{state}, gate error ϵ_{gate}, coherent error $\epsilon_{coherent}$ and measurement error $\epsilon_{measure}$.

As a result, for our assertion circuit, its error rate can be computed as

$$\epsilon_{assert} = 1 - P_{success} = 1 - \prod_i (1 - \epsilon_i) \tag{10}$$

We use our 3-qubit entanglement assertion circuit as an example to demonstrate the extra errors caused by our assertion logic. For this entanglement assertion circuit we append three CNOT gates to the circuit under test. We assume the state preparation error ϵ_{state} is 0.01, CNOT gate time is 500ns, $T_1 = T_2 = 100\mu s$, CNOT gate error ϵ_{gate} is 0.01, measurement error $\epsilon_{measure}$ is 0.01, the error rate ϵ_{assert} introduced by the assertion circuit is:

$$\epsilon_{assert} = 1 - [(e^{-500\times(10^{-5}+10^{-5})})(1 - 0.01)^3]^3 = 0.113 \tag{11}$$

We can see that based on the simple error model, the extra error introduced by the assertion circuit is small. The other assertion circuits has fewer numbers of quantum gates than the 3-qubit entanglement assertion circuit, thus they should have fewer errors. In comparison, the circuit under test is expected to have many more gates and more qubits, which would have much higher error rates compared to the assertion circuits.

The second approach is based on quantum state tomography [31]. Quantum state tomography reconstructs the quantum state and provides its density matrix ρ^E by performing sequences of measurements in different bases. Fidelity quantifies the difference between experimental density matrix ρ^E and ideal density matrix ρ^I[26], and it is defined by:

$$F(\rho^E, \rho^I) = Tr\left[\sqrt{\sqrt{\rho^I}\rho^E\sqrt{\rho^I}}\right] \tag{12}$$

Based on the prior work [10], a quantum error detection code with fidelity score higher than 80% is considered as high fidelity. Our experiments show that the fidelity of our assertion circuits well above 80% (see Section 6.1), which indicates that our assertion circuit has little impact on the circuits under test.

Another related issue to be considered is the actual CNOT gate implementation. Given the limited connectivity among qubits in quantum computers, we check to ensure that there are no additional SWAP gates being introduced as these additional gates increase the circuit depth and are susceptible to higher error rates.

4 Methodology

We implement our assertion circuits on Qiskit [3] which is an open-source framework for quantum computing. We augmented Qiskit version 0.13.0 with the function to insert assertion circuits and check ancilla bits for assertion. The adapted version of Qiskit is publicly available [1]. With our tool, the programmer is able to insert dynamic assertion circuits for classical, entanglement, and superposition states. It checks the ancilla bits' results for assertion errors and it can also filter out the erroneous results when running on real quantum computers. We offer three kinds of assertion functions:

- classical_assertion(circuit, qubitList, value)
  The classical_assertion() function takes three arguments specifying the quantum circuit under test, the list of qubits for assertion, and a particular classical value to assert for.
- entanglement_assertion(circuit, qubitList, flag)
  The entanglement_assertion() function takes three arguments specifying the quantum circuit under test, the list of qubits for assertion and the type of entanglement. The flag being 0 denotes that the circuit asserts for state in the form of $a\,|00\rangle + b\,|11\rangle$. The flag being 1 denotes that the circuit asserts for state in the form of $a\,|01\rangle + b\,|10\rangle$.
- superposition_assertion(circuit, qubitList, phaseDict, flag)
  The superposition_assertion() function takes four arguments specifying the quantum circuit under test, the list of qubits for assertion, the quantum state dictionary for the qubits, and a flag. The flag being 0 denotes that the uniform entanglement assertion circuit described in Section 3.3.1 is in use. The flag being 1 denotes that the circuit in Section 3.3.2 is used.

Qiskit does not have full support for an arbitrary controlled-U gate. For the controlled-U gate discussed in Section 3.3.2, we use the two qubit KAK decomposition function [12] in Qiskit to decompose our proposed controlled-U gate into a set of single qubit and CNOT gates.

We perform our experiments for quantum program debugging on the simulator Aer from Qiskit. We also perform experiments on an IBM Q (ibmq-20-tokyo) quantum computer to check the effectiveness of using assertions to filter out erroneous results. The connectivity map of ibmq-20-tokyo quantum computer is shown in Figure 8. For each benchmark mentioned in Section 6, its comparison experiments are executed within a time window of 30 minutes to guarantee that there is not much change in the quantum computer's environment and error characteristics.

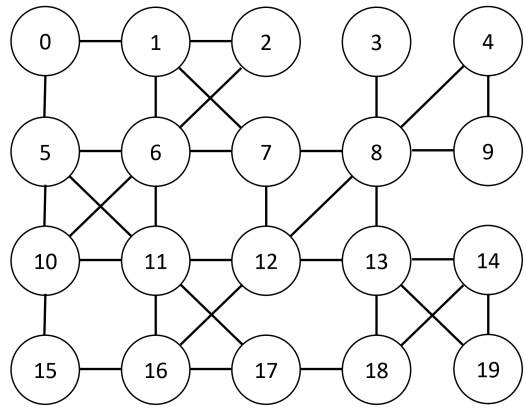

Figure 8. Connectivity map of ibmq-20-tokyo

Note that our proposed assertion circuit is aimed for detecting bugs/errors while the program is running. In the ideal case, when an assertion error is detected, the program should stop or restart. Due to the limitation of current systems, all the measurement are taken at the end. So these assertion qubits are used as post measurement selection on actual quantum computers.

5 Dynamic Runtime Assertion for Program Debugging

Previous research on statistical assertion [18] studies common types of bugs and proposes the statistical assertion approach for debugging. The usage of our dynamic assertion is the same as the statistical approach for debugging. Here, we use the Quantum Phase Estimation (QPE) algorithm as an example to illustrate how our proposed assertion primitives are used.

QPE is an important algorithm in quantum computing and is used as a building block in algorithms such as Shor's factoring algorithm. It estimates the phase (eigenvalue) of a unitary operator U. Given a unitary operator U and a quantum state $|\psi\rangle$ such that $U|\psi\rangle = e^{i2\pi\theta}|\psi\rangle$, the algorithm estimates the value for θ. Figure 9 shows the circuit of QPE based on inverse Quantum Fourier Transform (QFT).

Figure 10 shows the code for n-qubit QPE and we have inserted assertions at different stages. A quantum algorithm begins with initializing all the qubits to the preconditions in the initialization stage. Usually, the qubits are initialized to a classical value or a uniform superposition of values. In the quantum information process protocols, the qubits are often initialized to entanglement states such as Bell states. Our dynamic assertion can assert the preconditions for classical,

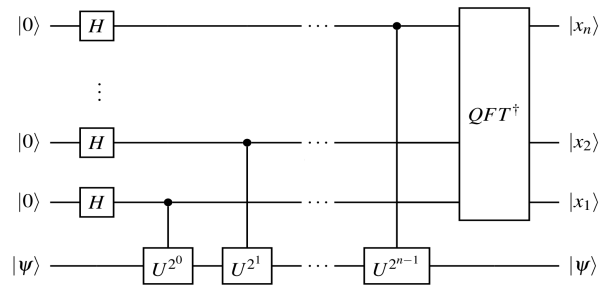

Figure 9. Quantum Phase Estimation circuit based on inverse QFT

superposition and entanglement states. As shown in line 17, we assert for the uniform superposition state using the "superposition_assertion()" function with the flag being 0. We also assert at line 23 the ancilla qubit which is in a non-uniform superposition state $|\psi\rangle$. In this case, the flag is set to 1.

In QPE, after applying inverse QFT, the output states of the qubits are in classical state. We assert for the expected classical states using the "classical_assertion()" function in line 35.

Besides the desired output, we can also assert for deallocated ancilla qubits used in the algorithm. Generally speaking, the programmer would only measure the qubits that carry program output and ignore the ancilla qubits. The ancilla qubits in the algorithm should be deallocated and stay in classic or superposition state. We argue that this ancilla qubit still carries useful information that indicates whether there is a bug in the program. Line 38 shows the superposition assertion for deallocated ancilla qubit, which ensures that this qubit remain the same as its initialized state.

6 Dynamic Runtime Assertion for Improving Success Rates

In this section, we will first show the reliability of our proposed assertion circuits on real quantum computers. Then we will provide several case studies on QFT, Quantum Phase Estimation and Bernstein Vazirani benchmark to show the effectiveness of our approach in increasing the success rate on the real quantum computers. In our experiment, each trial is executed on a 20-qubit quantum computer, ibmq_20_tokyo, for 8192 shots. We use the function from IBM Qiskit-Ignis to calculate fidelity.

6.1 Reliability of Proposed Assertion Circuits

Because of the faulty gates in real quantum computers, an error may occur in the assertion circuits and propagate into the qubits under test. Also, the qubits that we add for assertion may have measurement errors when reading out the value. It may, for example, be the case that the results are

```
1  #n qubits for quantum phase estimation circuit
2  q = QuantumRegister(n)
3  #one ancilla qubit
4  a = QuantumRegister(1)
5  c = ClassicalRegister(n)
6  circuit = QuantumCircuit(q,a,c)
7
8  # Initialize the qubits to uniform superposition
9  for i in range(n):
10     circuit.h(q[i])
11
12 # Initialize the qubit list for superposition assertion
13 qubitList = [q[0], q[1] ..., q[n−1]]
14 # Initialize the state dictionary for superposition assertion
15 phaseDict = {q[0]:[pi/2, 0], ..., q[n]:[pi/2, 0]}
16 # Superposition assertion for initialization states
17 superposition_assertion(circuit, qubitList, phaseDict, 0)
18
19 # Initialize φ and λ for the ancilla qubit
20 circuit.u1(phi,lambda, a[0])
21
22 # Superposition assertion for ancilla qubit initialization state
23 superposition_assertion(circuit, [a[0]], {a[0]:[phi,lambda]}, 1)
24
25 # Controlled U^(2^n) gate
26 for j in range(n):
27     controlled_U(circuit, a[0], q[j], j)
28
29 # n−qubit inverse QFT
30 iQFT(circuit, q, n)
31
32 # Initialize the qubit list for classical assertion
33 qubitList2 = [q[0], q[1] ..., q[n−1]]
34 # Classical assertion for output states
35 classical_assertion(circuit, qubitList2, value)
36
37 # Superposition assertion for ancilla qubit deallocated state
38 superposition_assertion(circuit, [a[0]], {a[0]:[phi,lambda]}, 1)
39
40 circuit.measure(q, c)
```

Figure 10. The code for n-qubit QPE and assertions.

correct, but assertion raises an error (false positive). We need the false positive probability as well as fidelity of the circuit to quantify the difference between experimental output state and the ideal output state.

Table 1 shows the results for our experiment. "Un1" stands for the circuit for uniform superposition assertion in Figure 5. "Un2" stands for the circuit for uniform superposition assertion in Figure 7. "Arb" is the arbitrary superposition assertion circuit, for which $\theta = \frac{\pi}{2}$ and $\varphi = \frac{\pi}{2}$. For these circuits, the qubits under test are set to have no assertion error.

From the table, we can see that all the classical and superposition assertion circuits have false positive possibility

	Classical	Entanglement		Superposition		
Type	1bit	2bits	3bits	Uni1	Uni2	Arb
Probability(%)	3.1%	2.6%	7.8%	2.7%	2.1%	3.8%
Fidelity(%)	95.0%	93.8%	88.5%	92.7%	93.3%	82.1%

Table 1. Probability of false positive case and fidelity for all assertion circuits

lower than 5%. And all the assertion circuits have fidelity higher than 80%. The entanglement assertion circuits have higher false positive possibility as a result of more CNOT gates in the circuit and the entanglement of multiple qubits. We can also find that "Uni2" has lower false positive probability and higher fidelity than "Uni1", which is in line with our expectation.

6.2 Quantum Fourier Transform: Asserting for Classical and Superposition States

In this section, we use QFT (Quantum Fourier Transform) as a case study, where the code of the QFT function is from the IBM Qiskit-Terra[3]. We compare the 4-qubit QFT program with and without assertion supports and add different types of assertion supports. Our experiment shows the usefulness of output state assertions and intermediate state assertions.

First, we evaluate the effectiveness of output state assertions. We set the input qubits in the uniform superposition state such that the expected output should be $|0000\rangle$. The quantum circuit based on code in Figure 11 is shown in Figure 12. By inspecting the quantum circuit, we can see that, although the circuit depth (i.e., number of gates) for the four input qubits is the same, q_3 is measured last. Therefore, rather than asserting for $|q_3q_2q_1q_0\rangle = |0000\rangle$, which requires four extra qubits, we choose to add the circuitry to assert $'|q_3\rangle == |0\rangle'$. We use the function "classical_assert()" to assert $q_3's$ output state for a classical value 0. The qubits for QFT are mapped to actual qubit No.5, 6, 10 and 11 on the 20-qubit quantum computer, and the ancilla qubit for assertion is mapped to qubit No.0. As shown in the connectivity map, i.e., Figure 8, our classical assertion only requires one extra CNOT gate, and no swap gate is involved. Note that although the CNOT gate used for assertion itself is subject to error, we expect it has much lower error rate compared to the QFT circuit of interest due to the disparity in circuit depth.

The measurement results are reported in Table 2 and Table 3. Without the assertion circuit (Table 2), the machine has a 72.0%(=5900/8192) success rate for the 4-qubit QFT computation. Among the erroneous ones, 14.0% has an error in q_3. After filtering out the measurements with q_3 being measured as $|1\rangle$, the success rate becomes 5811/(5811 + 219 + 879) = 84.1% (an improvement of 16.8%) as shown in Table 3.

The impact of the errors introduced from the assertion circuit (i.e., the CNOT gate and the measurement) is small: 1.5% false positive and 2.7% false negative measurements,

```
# Quantum fourier transform function
def qft(circ, q, n):
    for j in range(n):
        for k in range(j):
            circ.cu1(np.pi/float(2**(j-k)), q[j], q[k])
        circ.h(q[j])
q = QuantumRegister(4)
c = ClassicalRegister(4)
circuit = QuantumCircuit(q,c)

# Set up the 4-qubit input
for j in range(4):
    circuit.h(q[j])

#4-qubit QFT
qft(circuit, q, 4)

# Initialize the qubit list for classical assertion
qubitList = [q[1]]
# Classical assertion for q[1]
classical_assertion(circuit, qubitList, 0)

circuit.measure(q,c)
```

Figure 11. The code for 4-qubit QFT and result assertion on q[3].

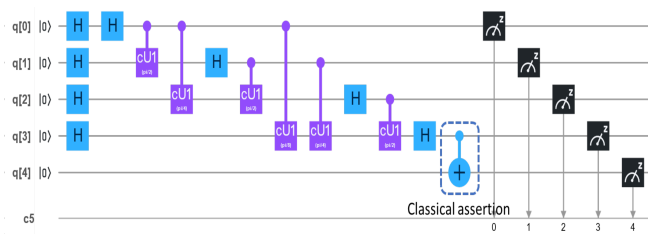

Figure 12. QFT circuit based on the code from Figure 11

$q_3q_2q_1q_0$	Counts	%(=counts/8192)	Meaning
0000	5900	72.0%	Correct result
0001~0111	1142	14.0%	Incorrect result with correct q_3
1xxx	1150	14.0%	Incorrect result with incorrect q_3.

Table 2. The results of QFT without assertion on IBM Q

$q_3q_2q_1q_0$	Counts	%(=counts/8192)	Meaning
00000	5811	70.9%	Correct result
10000	124	1.5%	Correct result with assertion error (false positive)
01xxx	219	2.7%	Incorrect result without assertion error (false negative)
11xxx	1033	12.6%	Incorrect result with assertion error
10001~10111	126	1.5%	Incorrect results with correct q_3 but assertion error
00001~00111	879	10.7%	Incorrect results with correct q_3 and without assertion error

Table 3. The results of QFT with classical assertion on IBM Q

compared to the errors in the QFT circuit. We also tried with asserting two qubits '$|q_3q_2\rangle == |00\rangle$' in the QFT circuit and the success rate is further increased to 86.2%.

To improve statistical significance, we repeated the same experiment five times on different dates. The min, median, and max improvement on the success rate by asserting q_3 are 15.4%, 19.6%, 29.0%, respectively.

In order to observe the effect of decoherence (in which a $|1\rangle$ state devolves into $|0\rangle$), we change the input state to produce an expected output state as $|1111\rangle$. After measurement we find that the success rate of QFT without assertion drops to 49.3% due to decoherence error, and the success rate of QFT with assertion is 60.8% (an improvement of 23.3%).

```
q = QuantumRegister(4)
c = ClassicalRegister(4)
circuit = QuantumCircuit(q,c)
# Set up the 4−qubit input 0100
circuit.x(q[2])

# 4−qubit QFT
qft(circuit, q, 4)

# Initialize the qubit list for superposition assertion
qubitList = [q[1]]
# Initialize the state dictionary for superposition assertion
phaseDict = {q[1]:[pi/2,pi/2]}
# Superposition assertion for q[1]'s intermediate state
superposition_assertion(circuit, qubitList, phaseDict, 1)

for i in range(4):
    circuit.u1(−pi/(2**(2−i)), q[i]) #change the output phase
    circuit.h(q[i]) #change the output state to |0>
circuit.measure(q,c)
```

Figure 13. The code for 4-qubit QFT and intermediate assertion on q[1].

Second, we evaluate the effectiveness of intermediate state assertions. We set the input qubit state as $|0100\rangle$ therefore the output states of QFT are in superposition. The ideal output states are: '$|q_0\rangle = \frac{1}{\sqrt{2}}|0\rangle + \frac{1+i}{2}|1\rangle$, $|q_1\rangle = \frac{1}{\sqrt{2}}|0\rangle + \frac{i}{\sqrt{2}}|1\rangle$, $|q_2\rangle = \frac{1}{\sqrt{2}}|0\rangle - \frac{1}{\sqrt{2}}|1\rangle$, $|q_3\rangle = \frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle$'. We add superposition assertion to assert '$|q_1\rangle == \frac{1}{\sqrt{2}}|0\rangle + \frac{i}{\sqrt{2}}|1\rangle$' in the QFT's output states. Since the measurement of superposition states are probabilistic, we add phase changing gates and Hadamard gates at the output of QFT to change the qubit states to classical states $|0000\rangle$. As shown in Figure 13, the output of QFT becomes intermediate state of the circuit and the final output is $|0000\rangle$.

Table 4 and Table 5 show the measurement results. As shown in Table 4, the success rate without assertion is 43.4% (=3556/8192). After we enable assertion, the success rate becomes 4961/(4961 + 187 + 1815) = 71.2% (an improvement of 64%). We repeated the same expreiment five times on different dates, The min, median and max improvement on the success rate by assertion q_1 are 36.4%, 52.8%, 66%, respectively.

$q_3q_2q_1q_0$	Counts	%(=counts/8192)	Meaning
0000	3556	43.4%	Correct result
xx0x	2890	35.3%	Incorrect result with correct q_1
xx1x	1746	21.3%	Incorrect result with incorrect q_1.

Table 4. The results of QFT without assertion on IBM Q

$q_3q_2q_1q_0$	Counts	%(=counts/8192)	Meaning
00000	4961	60.6%	Correct result
10000	184	2.2%	Correct result with assertion error (false positive)
0xx1x	187	2.3%	Incorrect result without assertion error (false negative)
1xx1x	929	11.3%	Incorrect result with assertion error
1xx0x	300	3.7%	Incorrect results with correct q_1 but assertion error
0xx0x	1815	22.2%	Incorrect results with correct q_1 and without assertion error

Table 5. The results of QFT with superposition assertion on IBM Q

6.3 Quantum Phase Estimation: Asserting for Classical States

As introduced in Section 5, QPE algorithm is used to estimate the phase of a unitary operator U. In our experiment, we implement the modified Lloyd QPE algorithm [23] and enable output state assertions. We change the phase of the unitary operator to produce different output states. The results of 4-qubit QPE are listed in Table 6. We always assert the most significant output bit.

Output states	Without assertion	With assertion	Improvement
0000	73.2%	84.9%	16.0%
0001	53.3%	68.1%	12.8%
0011	47.3%	53.2%	12.5%
0111	44.0%	53.1%	20.7%
1000	69.2%	79.6%	15.0%
1100	64.0%	71.7%	12.0%
1110	56.3%	66.9%	18.8%
1111	47.9%	58.0%	21.1%

Table 6. The results of 4-qubit QPE algorithm with output state classical assertion on IBM Q

Based on the results we can find that the success rate drops as the number of ones in the output state increases. This is because of two kinds of errors. The decoherence error decays the high-energy state $|1\rangle$ into the low-energy state $|0\rangle$. The measurement error has a higher error rate when measuring state $|1\rangle$. When we design our assertion circuits, we always set the rule such that the ancilla qubit being $|0\rangle$ means no

assertion error. This rule is set to alleviate decoherence error and measurement error in the assertion circuit since these errors will introduce false positive cases. We also find that the success rate of "0011" is lower than "1100" while they have the same number of ones in the output state. This is due to the property difference of the actual qubits as different qubits will have different gate and measurement error rates. Also some optimizations are performed when the compiler unrolls the circuit to basic gate sets. We find that the circuit generated for output "0011" requires one less U3 gate than the one for output "1100". This also explains that "0011" has higher error rate than "1100".

6.4 Quantum Entanglement Swapping Protocol: Asserting for Bell Pair States

Quantum entanglement swapping protocol [6] is a protocol that swaps the entanglement between two repeater stations. It is an important component for transferring information to distant places. Assume we have three parties, Alice, Bob and Charlie. We consider two pairs of entangled qubits, $A - C_1$, $B - C_2$, where A and B denote the qubits of Alice and Bob, respectively, and C_1, C_2 are qubits belongs to Charlie. The qubits are entangled by the Bell channel, $\frac{|00\rangle+|11\rangle}{\sqrt{2}}$. After this swapping process, the qubits of Alice and Bob, A and B get entangled. The qubits of Charlie(C_1 and C_2) also get entangled. We can add entanglement assertion for the Bell pair initialization stage and output stage of the swapping protocol.

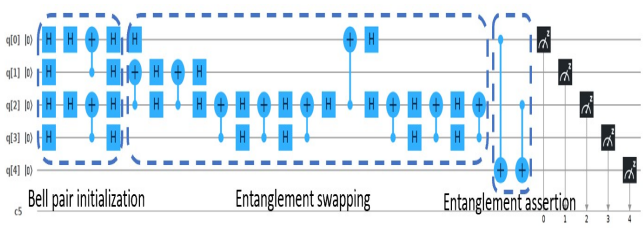

Figure 14. Entanglement swapping protocol with entanglement assertion

If $A - B$, $C_1 - C_2$ are entangled after the swapping process, after measurement, the output should be '$|0000\rangle$', '$|0101\rangle$', '$|1010\rangle$' and '$|1111\rangle$'. In our experiment, the success rate of the swapping protocol is 32.0%. After we enable entanglement assertion for Bell pair initialization stage of q_0 and q_1, the success rate becomes 29.6%. Because the depth of the initialization stage is low, error is not likely to happen in the initialization stage. So the entanglement assertion we enabled at the initialization stage does not improve the success rate. In contrast, the success rate after asserting the output stage q_0 and q_2 is 56.2%, much improved compared to the no assertion case.

6.5 Bernstein Vazirani: Asserting for Uniform Superposition States

In this section we use Bernstein Vazirani algorithm [7] as an example to evaluate the effectiveness of deallocated ancilla qubit assertion. In the Bernstein Vazirani algorithm, a black box oracle implements the the function $f_c(x) = x \cdot c$. The algorithm finds hidden string c with a single evaluation of the function. In the oracle, hidden string c is encoded with a set of CNOT gates. An ancilla qubit is used to encode the hidden string, and it is in the uniform superposition after deallocation. Therefore, we can apply superposition assertion on this ancilla qubit. Figure 15 shows the code.

```
q = QuantumRegister(2)
a = QuantumRegister(1) # one ancilla qubit
c = ClassicalRegister(2)
circuit = QuantumCircuit(q,a,c)
for i in range(2):
    circuit.h(q[i]) #set up the preconditions for 2 qubits
circuit.x(a[0])
circuit.h(a[0]) #set up the precondition for ancilla qubit

# Apply the oracle for hidden string 10
for j in range(2):
    if (2 & (1 << j)):
        circuit.cx(q[j], a[0])
    else:
        circuit.iden(q[j])
for k in range(2):
        circuit.h(q[k])

qubitList = [a[0]] #list of qubits for assertion
phaseDict = {a[0]:[pi/2, 0]}
superposition_assertion(circuit, qubitList, phaseDict, 1)
circuit.measure(q,c)
```

Figure 15. The code for 2-qubit Bernstein Vazirani algorithm and assertion on ancilla qubit.

In our experiment, we change the number of qubits in the Bernstein Vazirani algorithm, and results are shown in Table 7. The hidden strings for 2, 3 and 4-qubit BV algorithm are "10","110" and "1110" respectively. As the number of qubits increases, the success rate of the system decreases; however, our proposed assertion circuit consistently improve the success rate after asserting the ancilla qubit.

Number of qubits	Without assertion	With assertion	Improvement
2	77.9%	87.4%	12.2%
3	72.3%	80.0%	10.7%
4	51.1%	57.9%	13.1%

Table 7. The results of Bernstein Vazirani algorithm with ancilla qubit assertion on IBM Q

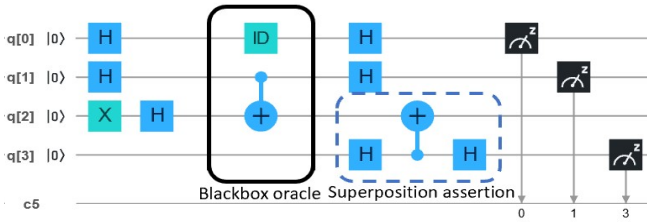

Figure 16. Bernstein Vazirani circuit based on code from Figure 15

6.6 Other Benchmarks

We also enable our assertion circuit on other benchmarks. Table 8 shows the success rate of the benchmarks and the improvement of success rate after assertion. For the Toffolli gate, our input state is $|110\rangle$ and the output state is $|111\rangle$, we enable classical assertion for '$|q_2\rangle == |1\rangle$'. For the 1-qubit adder, the output state is $|10\rangle$. We enable classical assertion for '$|q_0\rangle == |0\rangle$'. The Deutsch-Jozsa algorithm [9] determines whether a hidden oracle function is constant or balanced. We assert '$|q_4 == |0\rangle$' which indicates the hidden function is constant.

benchmark	Without assertion	With assertion	Improvement
Tofolli gate	61.9%	70.0%	13.1%
1-bit adder	63.3%	81.8%	29.2%
4-bit Deutsch–Jozsa	74.3%	80.0%	7.7%

Table 8. The results of classical assertions on IBM Q

7 Conclusion

In this paper, we propose quantum circuits to enable dynamic assertions for classical values, entanglement, and superposition. This enables a dynamic debugging primitive, driven by a programmer's understanding of the correct behavior of the quantum program. We show that besides generating assertion errors, the assertion logic may also force the qubits under test to be into the desired state. Besides debugging, our proposed assertion logic can also be used in noisy intermediate scale quantum (NISQ) systems to filter out erroneous results, as demonstrated on a 20-qubit IBM Q quantum computer. Our proposed assertion circuits have been implemented as functions in the open-source Qiskit tool.

Acknowledgments

We thank the anonymous reviewers for their valuable comments. This work is funded in part by NSF grants 1717550 and 1908406.

References

[1] 2020. https://doi.org/10.5281/zenodo.3597507. [Online; accessed 14-1-2020].

[2] Dorit Aharonov, Aram W Harrow, Zeph Landau, Daniel Nagaj, Mario Szegedy, and Umesh Vazirani. 2014. Local tests of global entanglement and a counterexample to the generalized area law. In *2014 IEEE 55th Annual Symposium on Foundations of Computer Science*. IEEE, 246–255.

[3] Gadi Aleksandrowicz, Thomas Alexander, Panagiotis Barkoutsos, Luciano Bello, Yael Ben-Haim, David Bucher, Francisco Jose Cabrera-Hernández, Jorge Carballo-Franquis, Adrian Chen, Chun-Fu Chen, Jerry M. Chow, Antonio D. Córcoles-Gonzales, Abigail J. Cross, Andrew Cross, Juan Cruz-Benito, Chris Culver, Salvador De La Puente González, Enrique De La Torre, Delton Ding, Eugene Dumitrescu, Ivan Duran, Pieter Eendebak, Mark Everitt, Ismael Faro Sertage, Albert Frisch, Andreas Fuhrer, Jay Gambetta, Borja Godoy Gago, Juan Gomez-Mosquera, Donny Greenberg, Ikko Hamamura, Vojtech Havlicek, Joe Hellmers, Łukasz Herok, Hiroshi Horii, Shaohan Hu, Takashi Imamichi, Toshinari Itoko, Ali Javadi-Abhari, Naoki Kanazawa, Anton Karazeev, Kevin Krsulich, Peng Liu, Yang Luh, Yunho Maeng, Manoel Marques, Francisco Jose Martín-Fernández, Douglas T. McClure, David McKay, Srujan Meesala, Antonio Mezzacapo, Nikolaj Moll, Diego Moreda Rodríguez, Giacomo Nannicini, Paul Nation, Pauline Ollitrault, Lee James O'Riordan, Hanhee Paik, Jesús Pérez, Anna Phan, Marco Pistoia, Viktor Prutyanov, Max Reuter, Julia Rice, Abdón Rodríguez Davila, Raymond Harry Putra Rudy, Mingi Ryu, Ninad Sathaye, Chris Schnabel, Eddie Schoute, Kanav Setia, Yunong Shi, Adenilton Silva, Yukio Siraichi, Seyon Sivarajah, John A. Smolin, Mathias Soeken, Hitomi Takahashi, Ivano Tavernelli, Charles Taylor, Pete Taylour, Kenso Trabing, Matthew Treinish, Wes Turner, Desiree Vogt-Lee, Christophe Vuillot, Jonathan A. Wildstrom, Jessica Wilson, Erick Winston, Christopher Wood, Stephen Wood, Stefan Wörner, Ismail Yunus Akhalwaya, and Christa Zoufal. 2019. Qiskit: An Open-source Framework for Quantum Computing. https://doi.org/10.5281/zenodo.2562110

[4] Anindita Banerjee, Chitra Shukla, and Anirban Pathak. 2015. Maximal entanglement concentration for a set of (n+1)-qubit states. *Quantum Information Processing* 14, 12 (2015), 4523–4536.

[5] Adriano Barenco, Charles H Bennett, Richard Cleve, David P DiVincenzo, Norman Margolus, Peter Shor, Tycho Sleator, John A Smolin, and Harald Weinfurter. 1995. Elementary gates for quantum computation. *Physical review A* 52, 5 (1995), 3457.

[6] Bikash K Behera, Swarnadeep Seth, Antariksha Das, and Prasanta K Panigrahi. 2019. Demonstration of entanglement purification and swapping protocol to design quantum repeater in IBM quantum computer. *Quantum Information Processing* 18, 4 (2019), 108.

[7] Ethan Bernstein and Umesh Vazirani. 1997. Quantum complexity theory. *SIAM Journal on computing* 26, 5 (1997), 1411–1473.

[8] Dagmar Bruß and Chiara Macchiavello. 1999. Optimal state estimation for d-dimensional quantum systems. *Physics Letters A* 253, 5-6 (1999), 249–251.

[9] Richard Cleve, Artur Ekert, Chiara Macchiavello, and Michele Mosca. 1998. Quantum algorithms revisited. *Proceedings of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences* 454, 1969 (1998), 339–354.

[10] Antonio D Córcoles, Easwar Magesan, Srikanth J Srinivasan, Andrew W Cross, Matthias Steffen, Jay M Gambetta, and Jerry M Chow. 2015. Demonstration of a quantum error detection code using a square lattice of four superconducting qubits. *Nature communications* 6 (2015), 6979.

[11] IBM Corporation. 2019. IBM Q Experience. https://quantum-computing.ibm.com/. [Online; accessed 14-8-2019].

[12] Andrew W Cross, Lev S Bishop, Sarah Sheldon, Paul D Nation, and Jay M Gambetta. 2018. Validating quantum computers using randomized model circuits. *arXiv preprint arXiv:1811.12926* (2018).

[13] Daniel Gottesman. 2010. An introduction to quantum error correction and fault-tolerant quantum computation. In *Quantum information science and its contributions to mathematics, Proceedings of Symposia in Applied Mathematics*, Vol. 68. 13–58.

[14] Manu Gupta and Prasanta K Panigrahi. 2005. Deterministic Bell State Discrimination. *arXiv preprint quant-ph/0504183* (2005).

[15] M Gupta, A Pathak, R Srikanth, and K Panigrahi. 2007. General circuits for indirecting and distributing measurement in quantum computation. *International Journal of Quantum Information* 5, 04 (2007), 627–640.

[16] Aram W Harrow and Ashley Montanaro. 2013. Testing product states, quantum Merlin-Arthur games and tensor optimization. *Journal of the ACM (JACM)* 60, 1 (2013), 3.

[17] Yipeng Huang and Margaret Martonosi. 2018. QDB: from quantum algorithms towards correct quantum programs. *arXiv preprint arXiv:1811.05447* (2018).

[18] Yipeng Huang and Margaret Martonosi. 2019. Statistical assertions for validating patterns and finding bugs in quantum programs. In *Proceedings of the 46th International Symposium on Computer Architecture*. ACM, 541–553.

[19] Sakshi Jain, Sreraman Muralidharan, and Prasanta K Panigrahi. 2009. Secure quantum conversation through non-destructive discrimination of highly entangled multipartite states. *EPL (Europhysics Letters)* 87, 6 (2009), 60008.

[20] Emanuel Knill, Dietrich Leibfried, Rolf Reichle, Joe Britton, R Brad Blakestad, John D Jost, Chris Langer, Roee Ozeri, Signe Seidelin, and David J Wineland. 2008. Randomized benchmarking of quantum gates. *Physical Review A* 77, 1 (2008), 012307.

[21] Daniel A Lidar and Todd A Brun. 2013. *Quantum error correction*. Cambridge university press.

[22] VS Manu and Anil Kumar. 2011. Non-Destructive Discrimination of arbitrary set of orthogonal quantum states by NMR using Quantum Phase Estimation. In *AIP Conference Proceedings*, Vol. 1384. AIP, 229–240.

[23] Hamed Mohammadbagherpoor, Young-Hyun Oh, Anand Singh, Xianqing Yu, and Andy J Rindos. 2019. Experimental Challenges of Implementing Quantum Phase Estimation Algorithms on IBM Quantum Computer. *arXiv preprint arXiv:1903.07605* (2019).

[24] Ashley Montanaro and Ronald de Wolf. 2013. A survey of quantum property testing. *arXiv preprint arXiv:1310.2035* (2013).

[25] Michael A Nielsen and Isaac Chuang. 2002. Quantum computation and quantum information.

[26] Michael A Nielsen and Isaac L Chuang. 2010. *Quantum Computation and Quantum Information: Introduction and overview; 2. Introduction t quantum mechanics; 3. Introduction to computer science; Part II. Quantum Computation: 4. Quantum circuits; 5. The quantum Fourier transform and its application; 6. Quantum search algorithms; 7. Quantum computers: physical realization; Part III. Quantum Information: 8. Quantum noise and quantum operations; 9. Distance measures for quantum information; 10. Quantum error-correction; 11. Entropy and information; 12. Quantum information theory; Appendices; References; Index*. Cambridge university press.

[27] Shin Nishio, Yulu Pan, Takahiko Satoh, Hideharu Amano, and Rodney Van Meter. 2019. Extracting Success from IBM's 20-Qubit Machines Using Error-Aware Compilation. *arXiv preprint arXiv:1903.10963* (2019).

[28] Saipriya Satyajit, Karthik Srinivasan, Bikash K Behera, and Prasanta K Panigrahi. 2018. Nondestructive discrimination of a new family of highly entangled states in IBM quantum computer. *Quantum Information Processing* 17, 9 (2018), 212.

[29] Maximilian A Schlosshauer. 2007. *Decoherence: and the quantum-to-classical transition*. Springer Science & Business Media.

[30] Mitali Sisodia, Abhishek Shukla, and Anirban Pathak. 2017. Experimental realization of nondestructive discrimination of Bell states using a five-qubit quantum computer. *Physics Letters A* 381, 46 (2017), 3860–3874.

[31] RT Thew, Kae Nemoto, Andrew G White, and William J Munro. 2002. Qudit quantum-state tomography. *Physical Review A* 66, 1 (2002), 012303.

A Artifact Appendix

A.1 Abstract

Our artifact provides the experiments for all our evaluated benchmarks, along with experiments to validate the quantum circuit fidelity in our paper.

We also provide source code for all of our benchmarks and our function for inserting assertion circuits.

A.2 Artifact check-list (meta-information)

- **Algorithm:** quantum mechanics
- **Hardware:** We recommend to run the experiments on a 20-qubit IBM Q machine to verify results.
- **Execution:** Run the corresponding jupyter notebooks
- **Metrics:** Fidelity: Defined by Eq.12
  Probability of false-positive case: % The count of the false-positive case over the total count of the trials.
  Success rate: % The count of correct output state over the total count of the trials
- **Output:** Fidelity and probability of false positive case are printed by running the jupyter notebook for fidelity test. Success rate for each benchmark is printed by running the corresponding jupyter notebook.
- **Experiments:** We use functions from Qiskit to measure the fidelity of our assertion circuit. We calculate the probability of false positive case and success rate based on the output data of the IBMQ backends.
- **How much disk space required (approximately)?:** 2GB
- **How much time is needed to prepare workflow (approximately)?:** A couple of minutes
- **How much time is needed to complete experiments (approximately)?:** Dozens of minutes, depending on the number of jobs submitted to the machine.
- **Publicly available?:** Yes.

A.3 Description

A.3.1 How delivered. Our benchmark, source code, and jupyter notebooks for experiments are available on Github: https://github.com/revilooliver/Quantum-Circuits-for-Dynamic-Runtime-Assertions-in-Quantum-Computation.git. The DOI of our artifact is https://doi.org/10.5281/zenodo.3597507.

A.3.2 Hardware dependencies. We recommend running the experiments with a 20-qubit IBM Q machine (In our paper, we used *ibmq_20_tokyo*). The experiments can also run on the publically available 14-qubit machine *ibmq_16_melbourne*. Due to the hardware property difference, the results of fidelity and success rate on different machines may differ.

A.3.3 Software dependencies. Python 3.5+, Qiskit 0.13.0, Jupyter notebook. Qiskit requires Ubuntu 16.04 or later, macOS 10.12.6 or later and Windows 7 or later

A.3.4 Data sets. Quantum computing benchmarks mentioned in our paper.

A.4 Installation

To install Qiksit, please refer to:

https://qiskit.org/documentation/install.html

You can clone our jupyter notebooks and benchmarks from Github:

$ git clone https://github.com/revilooliver/Quantum-Circuits-for-Dynamic-Runtime-Assertions-in-Quantum-Computation.git

After clone, copy and paste the assertion.py file under the compiler folder "... \qiksit \compiler" in the Qiskit installation directory.

A.5 Experiment workflow

To run the experiments for the fidelity of our assertion circuit, run the jupyter notebooks under the folder named "fidelity". Tests should take less than 10 mins for the 20-qubit quantum computers, depending on the number of jobs submitted to the machine.

To run the experiments for the success rate of different benchmarks, run the jupyter notebooks under the folder named "benchmark". Test could take dozens of minutes, depending on the number of jobs submitted to the machine.

A.6 Evaluation and expected result

The fidelity and success rates are printed after running the corresponding jupyter notebook. The expected results are reported in our paper.

A.7 Experiment customization

The jupyter notebooks are all customizable to run different assertions with different benchmarks. To insert different assertion circuits, first import the corresponding assertion function:

from qiskit.compiler.assertion import classical_assertion

Then call the functions as described in Section 4. The assertion function will insert the assertion circuits to the circuit under test.

A.8 Notes

Due to the hardware property difference of different backends, the results of fidelity and success rate may differ. When running the experiments on backends with limited connectivity, the inserted assertion circuit may introduce too many extra swap gates and therefore hurt the success rate of the circuit under test. In our paper, we ran our experiments on *ibmq_20_tokyo* quantum computer which offers the best connectivity among all the 20-qubit machines. However, this quantum computer has retired. Among the currently available 20-qubit machines, *ibmq_boeblingen* has the lowest noise level, so we recommend to reproduce the experiments on boeblinegen machine.

The transpiler from Qiskit uses the stochastic swap pass, so the number of swap gates (each swap gate consists of three CNOT gates) inserted for the logical-to-physical mapping may vary for the reproductions of the same experiment. We recommend use the circuit.count_ops() function in Qiskit to check minimum number of CNOT gates is inserted after logical-to-physical mapping.

A.9 Methodology

Submission, reviewing and badging methodology:

- http://cTuning.org/ae/submission-20190109.html
- http://cTuning.org/ae/reviewing-20190109.html
- https://www.acm.org/publications/policies/artifact-review-badging