Systematic Approaches for Precise and Approximate Quantum State Runtime Assertion

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

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

***Abstract*—With the rapid growth of quantum computing technology, programmers need new tools for debugging quantum programs. Recent works show that assertions are a promising way for debugging quantum programs. However, there are two main drawbacks with the existing schemes. First, the existing schemes, including both statistical and dynamic assertions are only capable of asserting limited types of states, namely classical, superposition, and specific entanglement states. Second, the use cases of these assertions are limited, since the programmer has to know the exact/precise state to assert.**

In this work, we propose two systematic approaches for dynamic quantum state assertion and they can assert a much broader range of quantum states including both pure states and mixed states. We also introduce the idea of approximate quantum state assertion for the cases where the programmers only have limited knowledge of the quantum states. Approximate assertion is capable of checking membership in a set of states {$|\psi\rangle$, $|\phi\rangle$, ...}. While precise quantum state assertion can check a specific quantum state, approximate assertion enables a way to check whether the qubits of interest are in a super-set of some expected states, which is analogous to the well-known Bloom filter for membership checking in classical computing. Our experiments demonstrate that our systematic approaches can assert many more quantum states and can be used in various assertion locations for qubit state checking.

***Index Terms*—quantum computing, runtime assertion**

I. Introduction

Quantum computing has become an active research area due to its remarkable potential in chemistry simulation [25], cryptography [18], machine learning [9], and many more applications. The fast pace of development in quantum software platforms and programming languages has necessitated the development of program debugging tools. However, quantum computing features a number of unique properties including superposition and entanglement. These features increase the difficulty of writing correct quantum programs; even quantum experts may write them incorrectly. Li et al. [30] reported that bugs have been found in the example programs in IBM's OpenQASM project [16] and Rigetti's PyQuil project [38]. Previous works by Huang et al. [27], [28] have discussed several types of bugs in quantum programs. Based on their study, many bugs in the quantum programs are different from the ones in classical programs. Nevertheless, they showed that similar to debugging classical programs, assertions are very useful primitives for debugging quantum programs.

Assertion is an important technique for program debugging. It is a predicate that should always be evaluated to be true during program execution. Huang et al. proposed a statistical assertion scheme [28] which essentially creates breakpoints in the program and measures the qubits at breakpoints. Due to its destructive measurement, a statistical assertion cannot be implemented dynamically at runtime. To overcome this, Liu et al. proposed quantum circuits for dynamic runtime assertions [32], which do not interrupt program execution if there are no assertion errors. In addition to debugging quantum programs, it is shown that assertion can also be used to improve program reliability by filtering out erroneous results. However, the scope of their supported assertions is limited. The assertion primitives only support assertions for classical, superposition, and specific entanglement states. Li et al. proposed a runtime assertion scheme [30] based on projective measurements. The projection-based assertions can cover a broader range of quantum states. However, the projection-based assertions require architectural support for measuring the same qubits repeatedly and performing gate operations after measurement. The state-of-the-art publicly available quantum computers [41], [42] only provide measurements at the end of the program and cannot take advantage of the projection-based assertion scheme. Besides such limitations, another key issue is that all these dynamic assertions require precise state information before debugging. However, in practical cases, the programmer may not know the exact state to be asserted beforehand. If the programmer selects an incorrect assertion state, the quantum state would be destroyed after assertion. In short, there are four limitations of the prior assertion schemes: (a) destructive measurement stops program execution, (b) existing dynamic assertion schemes support limited assertion states, (c) repeated measurements require architectural support, and (d) asserting quantum states requires precise assertion state information.

In this paper, we focus on dynamic assertion as it overcomes the limitation of destructive measurement. We propose two systematic approaches: SWAP-based assertion and Non-destructive-discrimination(NDD)-based assertion, for assertion circuit design. Both approaches can assert a broad range of quantum states and both can be performed on the existing quantum computers (i.e., no need for additional hardware support). We show that the prior proposed dynamic assertion

circuits by Liu et al. [32] are actually special cases of our systematic approaches. We analyze the efficiency and effectiveness of both approaches and show that either design has its own advantages.

We further introduce the idea of approximate quantum state assertion. It can be used for cases when the programmer has limited knowledge of the quantum states. For example, the programmer expects that the quantum state needs to satisfy some requirements but does not have the knowledge of the exact quantum state. We propose approximate assertion as a membership check w.r.t. to a set of quantum states $\{|\psi\rangle$, $|\phi\rangle$, $|\theta\rangle$,...$\}$. An assertion failure means that the checked state is not in the set while no assertion error means that the checked state is within the set but not sure which state. When we construct the set to be a superset of the expected states, approximate quantum state assertion is somewhat analogous to the well-known Bloom filter [10] in classical computing. We also present the design methodology of approximate assertion circuits using both the SWAP-based and NDD-based approaches.

The major contributions of our paper are as follows:

- We propose two systematic approaches for dynamic assertion, which essentially generalize the prior (ad hoc) assertion schemes.
- We expand the assertion types from classical, superposition, and entanglement state to a wider range including arbitrary pure states and many mixed states.
- We analyze the efficiency and effectiveness of different assertion schemes.
- We introduce the idea of approximate assertion, which performs membership checks on a set of states.

The remainder of the paper is organized as follows. Section II presents the background of quantum computing and discusses the related work. Section III presents an overview of our proposed precise and approximate assertions and highlights how they overcome the limitations of the prior work. Sections IV and V detail SWAP-based and NDD-based assertion circuit designs, respectively. Section VI compares our proposed schemes with existing ones. Section VII presents our experimental methodology. Section VIII discusses the general applicability of the assertions. Sections IX and X showcase the use of precise and approximate assertions for debugging, respectively. Section XI summarizes the paper.

II. Background and Related Work

A. Quantum Computing

In quantum computing, information is encoded in quantum bits (qubits). A quantum program consists of a sequence of quantum gates operating on qubits. During program execution, qubits can be in pure or mixed states. Pure quantum states correspond to vectors in a Hilbert space, a complex vector space with an inner product, and can be represented with vectors with norm one. For example, a single-qubit pure state can be represented as $|\psi\rangle = a|0\rangle + b|1\rangle$ where $|a|^2 + |b|^2 = 1$.

Due to entanglement, qubits can be in mixed states [11], which are a mixture of pure states. A mixed state cannot be described with a vector. Instead, it is described with a density matrix $\rho = \sum_i P_i |\psi_i\rangle \langle\psi_i|$, where P_i are the probabilities of each pure state $|\psi_i\rangle$ and $\sum_i P_i = 1$. The pure state $|\psi\rangle$ can also be represented with a density matrix: $\rho = |\psi\rangle \langle\psi|$.

Multiple qubits are entangled if the state can not be represented with a tensor product of individual qubit states. In general, suppose we have a system containing two subsystems A and B, the state of the entire system AB is described by a state vector $|\psi_{AB}\rangle$ or a density matrix ρ_{AB}. When the state of the entire system can be represented as the tensor product of individual subsystems, i.e., $|\psi_{AB}\rangle = |\psi_A\rangle \otimes |\psi_B\rangle$, subsystem A is not entangled with subsystem B or system AB is separable. If system AB can't be represented as the tensor product of individual subsystems, subsystem A is entangled with subsystem B, and the system AB is not separable. The state of subsystem A can be derived from the partial trace of the whole system AB: $\rho_A \equiv tr_B(\rho_{AB})$.

Mixed states are not necessarily entangled states. When determining the mixedness of a system, the entanglement within the system doesn't matter. For example, we have a 3-qubit system with the state $|\psi_{123}\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle) \otimes |0\rangle$, which means that the first qubit is entangled with the second qubit while the third qubit is not entangled with the first two. The density matrix of the system is $\rho_{123} = \frac{1}{2}(|00\rangle \langle 00| + |11\rangle \langle 11|) \otimes |0\rangle \langle 0|$. If we consider the system of the first two qubits, since they are not entangled with the third qubit, the two-qubit system is in a pure state, which can be represented with a vector $|\psi_{12}\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$. Note that the first two qubits are entangled, but the two-qubit state is a pure state. However, if we consider the system consisting of the second and third qubits, since this two-qubit system is entangled with the first qubit, it is in the mixed state $\rho_{23} = tr_1(\rho_{123}) = \frac{1}{2}(|0\rangle \langle 0| + |1\rangle \langle 1|) \otimes |0\rangle$. In this case, although the second and the third qubits are not entangled, the state of these two qubits is a mixed state. The prior dynamic assertion schemes [32] can assert the special pure state $|\psi_{12}\rangle$ as it is a Bell pair, but they can't assert the mixed state ρ_{23} as well as some other pure states.

An n-qubit quantum operation/gate is represented by a $2^n \times 2^n$ unitary matrix U. A gate U operates on a pure state $|\psi\rangle$ results in a state $|\psi'\rangle = U|\psi\rangle$. A gate U operates on a mixed state ρ leads to a state $\rho' = U\rho U^\dagger$ [35].

The state vector $|\psi\rangle$ representing a pure state is a unit vector [35], i.e., $\langle\psi|\psi\rangle = 1$. Two pure states $|\psi\rangle$ and $|\phi\rangle$ are orthogonal if their inner product is zero [37], i.e., $\langle\psi|\phi\rangle = 0$. A set of states $|\psi_i\rangle$ is orthonormal when each state is represented by a unit vector and distinct states in the set are orthogonal [35]. In an n-qubit quantum system, a set of n orthonormal states $|\psi_i\rangle$ forms an orthonormal basis of the system. Any quantum state $|\pi\rangle$ can be represented as a superposition of these states, i.e., $|\pi\rangle = \sum_{i=0}^{n} \alpha_i |\psi_i\rangle$. An orthonormal basis $\{|\psi_i\rangle\}$ has following properties: (1)$\langle\psi_i|\psi_j\rangle = 0$, for all $i \neq j$, and $\langle\psi_i|\psi_j\rangle = 1$, for all $i = j$. (2)$\sum_{i=0}^{2^n-1} |\psi_i\rangle \langle\psi_i| = I$, where I is the identity matrix. We call the orthonormal basis, which only consists of the classical state $|0\rangle$ and $|1\rangle$, the computational basis. For example, the computational basis of a two qubit system is $|00\rangle$, $|01\rangle$, $|10\rangle$, $|11\rangle$. In the real

quantum devices, we usually measure the qubit states in the computational basis.

B. Related work

A few approaches have been proposed to provide assertions for quantum programs. Huang et al. [28] proposed a statistical approach by measuring the qubits multiple times and compare the probability distribution with that of the desired state. However, directly measuring the qubits may change the quantum state after measurement. Therefore, the statistical assertion cannot be applied at runtime. SWAP-test [13] is another candidate for assertions. However, when the ancilla qubit in the swap test is measured as 0, i.e., no assertion error, there is no guarantee that the qubit under test is in the desired state. For example, when the qubit under test and the desired state are orthogonal (e.g., $|0\rangle$ and $|1\rangle$), the ancilla qubit has a probability of 50% being 0. Therefore, it is not suitable for dynamic runtime assertions.

Liu et al. [32] proposed dynamic runtime assertion primitives for three different types of states: classical, superposition, and specific entanglement states. These assertion primitives are designed such that they can measure the qubits non-destructively through the ancilla qubits. The three types of states are represented by state vectors, which means they are pure states. However, asserting these three types of state are not sufficient, and might lead to an inefficient design. For example, the highly entangled states such as Bell states [6], GHZ states [23] and cluster states [12] are widely used in quantum teleportation [7], [14], quantum secret sharing [29], and super dense coding [34]. The prior work only describes assertions for entangled states with even or odd numbers of ones, leaving many types of entanglement states unaddressed. Moreover, we might want to assert only a few qubits in a large entangled system. These qubits are usually in a mixed state since they are likely to be entangled with the other qubits. The lack of a wide range of assertion types necessitates finding a more generic dynamic assertion scheme.

Li et al. [30] proposed assertion schemes based on projective-measurement. Their assertion schemes are capable of asserting a broader type of quantum states including mixed states. However, their approach inserts measurements at each assertion location and requires the system to operate gates after measurement. Such an assertion scheme requires architectural support for measuring the qubits repeatedly and operating gates after measurement. The state-of-the-art publicly available quantum computers [41], [42] only provide measurement at the end of the program and cannot take advantage of this projection-based assertion scheme.

III. Precise and Approximate Assertion

In this work, we propose systematic approaches for both precise and approximate assertions. We consider the assertions that require a quantum state vector $|\psi\rangle$ or a density matrix ρ as *precise* quantum state assertion as the state to be asserted for is precise. The precise assertions can be considered as $assertEquals(|\psi\rangle, |\phi\rangle)$. Approximate assertion, on the other hand, provides a membership check for a set of states.

Compared to prior works, our precise assertion can assert for a wider range of states. First, our precise assertion for pure states ensures that any changes in the states, including the coefficients, will raise an assertion error. In the prior entanglement state assertion [32], the state in the form of $a|00\rangle + b|11\rangle$ is asserted using parity checks. As a result, it won't raise an assertion error for an entangled state with incorrect coefficients. Furthermore, for the GHZ state $|GHZ\rangle = \frac{1}{\sqrt{2}}|000\rangle + \frac{1}{\sqrt{2}}|111\rangle$ that is widely used in quantum communication [19] and cryptography, the prior primitives for asserting entangled state can't check for it since not all the states have even number of ones. Our proposed precise assertion overcomes the limitations and can check all such cases. Second, we propose assertion schemes for mixed states to further increase the range of states that can be checked.

Approximate assertion checks whether a state $|\psi\rangle$ is within a set of states $\{|\phi\rangle, |\theta\rangle, ...\}$. An assertion error means that the state $|\psi\rangle$ is not in the set. It extends the usage of assertions when the programmer only has limited knowledge of the program. For example, in the Deutsch-Jozsa algorithm [17] we are given a black-box function $f(x)$. The function is guaranteed to be either constant (output is always 0 or always 1) or balanced (returns 0 for half of the inputs and 1 for the rest). The Deutsch-Jozsa algorithm determines whether the function $f(x)$ is constant or balanced. Since the $f(x)$ is a black-box function, the programmer can't predict the output of $f(x)$. If a program bug happens in the black-box function $f(x)$, which makes the function neither constant nor balanced, the existing dynamic assertion schemes can't assert for such type of bugs. However, since the constant output states together with the inputs form a set of states, we can resort to approximate assertion. If the function is neither constant nor balanced, the program will raise an assertion error. We will discuss this example in detail in Section X.

Our proposed mixed-state assertion and approximate assertion also enable a new trade-off between assertion accuracy and circuit complexity. For example, to check the GHZ state, we can use three different assertions as shown in Figure 1. With our SWAP-based circuit design, 10 CNOT gates are needed for the precise assertion for the GHZ state. We can use two ways to reduce the circuit cost. First, we assert for fewer qubits. Instead of asserting the three-qubit state, we can assert the last two qubits since our approach allows assertion for such a two-qubit mixed state. With our SWAP-based mixed-state assertion, the circuit requires 4 CNOT gates. Second, we can disregard the coefficients and assert for a set of states $\{|000\rangle, |111\rangle\}$ using approximate assertion. Our SWAP-based approximate assertion circuit requires 8 CNOT gates for such approximate assertion. We can expand the set of states to further reduce the circuit cost. When we extend the set to $\{|000\rangle, |011\rangle, |100\rangle, |111\rangle\}$, the CNOT gate count can be reduced to 4. We can also use the NDD-based approximate assertion circuit along with a different set of states $\{\frac{|000\rangle+|111\rangle}{\sqrt{2}}, \frac{|001\rangle+|110\rangle}{\sqrt{2}}, \frac{|011\rangle+|100\rangle}{\sqrt{2}}, \frac{|010\rangle+|101\rangle}{\sqrt{2}}\}$, the

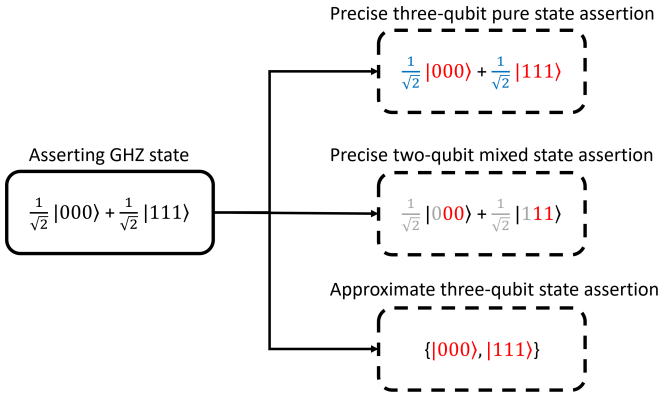


Fig. 1: Three different assertions for the GHZ state. The precise pure state assertion asserts for the exact three-qubit state, including coefficients and states. The precise two-qubit mixed state assertion asserts for the mixed state of the last two qubits. The approximate three-qubit state assertion asserts for a set of states $\{|000\rangle, |111\rangle\}$.

```
1 circuit.u2(0, np.pi, qr[0])
2 circuit.cx(qr[0], qr[1])
3 circuit.cx(qr[1], qr[2])
```

Fig. 2: The pseudo code for GHZ state preparation.

resulting circuit only requires 3 CNOT gates.

Next, let us use the GHZ example to illustrate the capabilities and the cost of different assertion schemes. Figure 2 shows the code for preparing the GHZ state. The code contains a single qubit $u2$ gate on qubit $qr[0]$ and two CNOT gates between qubit $qr[0]$, $qr[1]$, and $qr[2]$. Considering two types of bugs in the code. **Bug1**: In line 1, the programmer specifies incorrect $u2$ gate parameter order $circuit.u2(np.pi, 0, qr[0])$. This bug would lead to incorrect coefficients in the output state $\frac{1}{\sqrt{2}}|000\rangle - \frac{1}{\sqrt{2}}|111\rangle$. **Bug2**: The programmer reorders line 2 and 3. This bug will lead to incorrect entanglement in the output state $\frac{1}{\sqrt{2}}|000\rangle + \frac{1}{\sqrt{2}}|011\rangle$. Table I compares existing assertion schemes with our proposed ones in terms debugging capability and the circuit cost in the number of CNOT (CX) gates and single-qubit gates (SG), the number of ancilla qubits, and the number of measurements.

Assertion type	Bug1	Bug2	#CX	#SG	#ancilla	#measure
Stat [28]	False	True	N/A	N/A	N/A	N/A
Primitive [32]	N/A	N/A	N/A	N/A	N/A	N/A
Proq [30]	True	True	4	2	0	3
SWAP-based precise assertion	True	True	10	2	3	3
SWAP-based mixed state assertion	False	True	4	0	1	1
NDD-based approximate assertion	True	True	3	2	1	1

TABLE I: Assertion coverage and circuit cost for different assertion schemes. True/False means the assertion can/can't detect the corresponding bug.

The statistical assertion [28] measures the probability distribution of quantum states. Therefore, it is capable of detecting incorrect entanglement states (Bug2). However, it cannot capture the bugs with incorrect coefficients (Bug1) since the phase changes are not directly measurable in the computational basis. The prior assertion primitives [32] failed to provide assertion for the GHZ state as discussed before. The projection-based assertions (Proq) can capture these two bugs but it requires additional architectural support. In comparison, our proposed assertions are applicable to the state-of-the-art quantum computers and our approaches provide the flexibility for identifying the bug with different circuit costs.

In the next two sections, we will present two systematic assertion circuit designs: SWAP-based assertion circuit and NDD-based assertion circuit. We also propose logical OR based assertion design as an alternative for SWAP-based assertion design. For each circuit design approach, we will introduce the designs for precise and approximate assertion. For all of the designs, the ancilla qubit being $|0\rangle$ means no assertion error, being $|1\rangle$ means assertion error. The reason is that $|1\rangle$ has higher measurement error and may decay into $|0\rangle$.

IV. SWAP-Based Assertion Circuit

A. Motivation

To achieve $assertEquals(|\psi\rangle, |\phi\rangle)$, we can use a SWAP-based design as shown in Figure 3. We leverage the fact that any n-qubit pure state can be generated by applying a unitary gate U to the n-qubit ground state $|0\rangle^{\otimes n}$. For asserting an n-qubit pure state created by the program $|\psi\rangle = V_1...V_n |0\rangle^{\otimes n}$ equals to the desired state $|\phi\rangle = U|0\rangle^{\otimes n}$, we can apply the inverse gate U^{-1} to transfer the pure state $|\psi\rangle$ back to n-qubit state $|0\rangle^{\otimes n}$. Then we can measure the qubits to verify whether all the qubits are in $|0\rangle$ state. If the n-qubit state is not $|0\rangle^{\otimes n}$, the pure state $|\psi\rangle$ was not in the $|\phi\rangle$ state before the inverse transformation, and the circuit will raise an assertion error. However, the state-of-the-art quantum computers do not support operations after measurement which means the program cannot proceed after we measure the qubits. A solution to this problem is introducing SWAP gates and ancilla qubits. We can SWAP the qubits under test with ancilla qubits and measure the ancilla qubits, as shown in Figure 3. The first n-qubits are the qubits under test, and the pure state is generated by a sequence of gates $V_1...V_n$. We can apply the U^{-1} gate to convert it to $|0\rangle^{\otimes n}$ state. Since we can't measure it directly, we introduce SWAP gates to swap the qubits under test with the ancilla qubits. When we measure the ancilla qubits, all of the ancilla qubits should be in $|0\rangle$ state. In order to regenerate the desired state $|\phi\rangle$, we apply the unitary gate U to the ancilla qubits before swapping and then swap this prepared state to the qubits under test.

For example, the $|+\rangle$ state is generated by applying the Hadamard gate H to the $|0\rangle$ state. The Hadamard gate H is the inverse gate of itself. Therefore, if we want to assert for state $|+\rangle$, the unitary gate U in the assertion circuit should be the Hadamard gate. The assertion circuit is shown on the left side of Figure 4. The assertion circuit proposed by the prior work [32] is shown on the right side of Figure 4. We can prove these two circuits are equivalent, and the proof outline is in Appendix A.

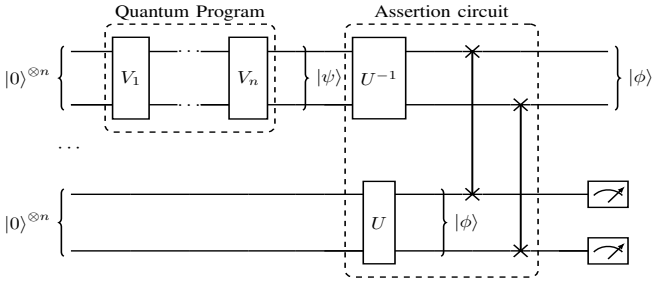


Fig. 3: The idea of SWAP based assertion circuits.

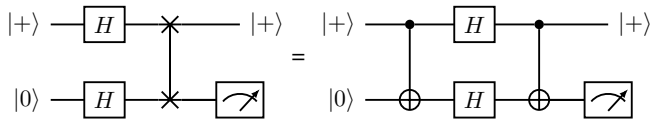


Fig. 4: Circuits for asserting $|+\rangle$ state: our circuit design is on the left side; the one in prior work [36] is on the right side.

Note that the prior work [36] has proved that the state preparation circuit U has lower cost than the original quantum program, i.e., $V_1...V_n$. Moreover, when used for debugging, both U and U^{-1} are synthesized from the assertion, therefore they are assumed to be bug free.

B. Precise Assertion for Pure States

The key to our precise assertion for pure states is to generate the U or U^{-1} gate from the pure state to assert for. To assert for an n-qubit pure state $|\psi_0\rangle$, from Figure 3, we know that $|\psi_0\rangle = U|0\rangle$ or $|0\rangle = U^{-1}|\psi_0\rangle$. Besides it, we also would like to signal an assertion error for other states. In other words, our goal is that by applying the unitary gate U^{-1}, we transform the "correct" state $|\psi_0\rangle$ to the zero state $|0\rangle^{\otimes n}$, and other "incorrect" states to the states in the computational basis, that contain at least one $|1\rangle$. This way, when we measure the qubits after U^{-1}, if all the qubits are in the $|0\rangle$ state, the circuit won't raise an assertion error. If any of the qubit is in $|1\rangle$ state, the pure state was not in the $|\psi_0\rangle$ state before the transformation, and the circuit will raise an assertion error. Figure 5 explains the idea. From the n-qubit pure state $|\psi_0\rangle$, we find an orthonormal basis $\{|\psi_i\rangle\}_{i\in[0,2^n-1]}$ that includes state $|\psi_0\rangle$ using the Gram-Schmidt process [22]. Then, based on the proposition in Appendix B, we generate U^{-1} to transform the states in this orthonormal basis to the states in the computational basis and $U^{-1} = |\psi_0\rangle\langle 0|^{\otimes n} + |\psi_1\rangle\langle 0|^{\otimes n-1}\langle 1| + ... + |\psi_{2^n-1}\rangle\langle 1|^{\otimes n}$.

For example, to assert for the Bell state $\frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$, we can find the orthonormal basis that includes the state as:$\{\frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$, $\frac{1}{\sqrt{2}}(|01\rangle + |10\rangle)$, $\frac{1}{\sqrt{2}}(|00\rangle - |11\rangle)$, $\frac{1}{\sqrt{2}}(-|01\rangle + |10\rangle)\}$. Then we can calculate the two-qubit unitary gate $U^{-1} = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)\langle 00| + \frac{1}{\sqrt{2}}(|01\rangle + |10\rangle)\langle 01| + \frac{1}{\sqrt{2}}(|00\rangle - |11\rangle)\langle 10| + \frac{1}{\sqrt{2}}(-|01\rangle + |10\rangle)\langle 11|$. With gate decomposition, we find this gate consists of a CNOT gate followed by a Hadamard gate on the control qubit.

On current quantum computers, we can only measure the qubits once at the end of the program. To overcome this limitation, we can swap the qubits with the ancilla qubits and measure the ancilla qubits. In order to reconstruct the qubit state $|\psi_0\rangle$, we need to prepare the state $|\psi_0\rangle$ by applying the gate U to the $|0\rangle^{\otimes n}$ state:$|\psi_0\rangle = U|0\rangle^{\otimes n}$. The general scheme of the SWAP based assertion circuit is illustrated in Figure 6. Since we are swapping the qubits with ancilla qubits in $|0\rangle$ state, the SWAP gates can be optimized with only two CNOT gates [31]. Depending on where we place the U gate and U^{-1} gate w.r.t. the SWAP gate, there are four different designs. For example, the design in Figure 3 and Figure 6 are two different designs with U gate at different places. We prefer to use the design in Figure 6 since the compiler has a better chance to optimize the gates with the gates before and after the assertion circuit.

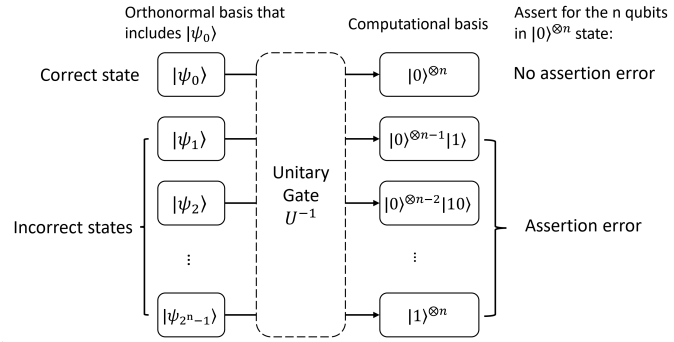


Fig. 5: The unitary gate transformation for pure states.

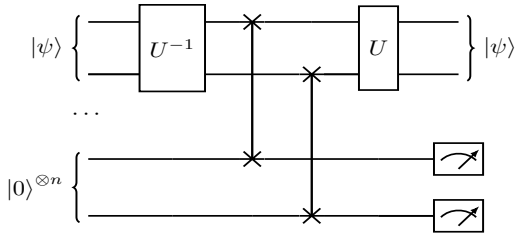


Fig. 6: General scheme of SWAP based assertion circuit for pure state assertions.

C. Precise Assertion for Mixed States

In this section, we generalize the assertion scheme to assert for mixed states. The main idea is that for a mixed state ρ_0 to be asserted for, we can generate an unitary gate U^{-1} to transform the state ρ_0 to the $|0\rangle$ states while keeping the entanglement with other qubits intact. A mixed state is a probability mixture of pure states, and can be defined with a density matrix, i.e., $\rho_0 = \sum_{i=0}^{t} P_i |\psi_i\rangle\langle\psi_i|$, where P_i are the probabilities of each pure state $|\psi_i\rangle$ and these pure states may not be orthogonal to each other.

In our design, the first step is to diagonalize the density matrix with singular value decomposition (SVD) and find orthonormal eigenstates. Since the density matrix is a square Hermitian matrix, it is always diagonalizable [26], i.e., $\rho_0 = D\Lambda D^{-1}$, where D is a unitary matrix, and $\Lambda = diag(\lambda'_0, \lambda'_1, ...\lambda'_{n-1})$ is a diagonal matrix. The columns vectors $|\psi'_0\rangle, |\psi'_1\rangle, ..., |\psi'_n\rangle$ of D form an orthonormal basis

and are eigenvectors of ρ_0 with eigenvalues $\lambda'_0, \lambda'_1, ..., \lambda'_{n-1}$, respectively. We use the orthonormal basis formed by the eigenvectors $|\psi'_i\rangle$. Since some of the eigenvalues might be zero, we reorder the eigenvalues and the eigenvectors such that the first t eigenvalues are non-zero, where t is the rank of the density matrix. Now the density matrix can be represented as $\rho_0 = \sum_{i=0}^{t-1} \lambda_i |\psi_i\rangle \langle\psi_i|$, where λ_i are the non-zero eigenvalues and $|\psi_i\rangle$ are the corresponding eigenvectors. The rank t of the density matrix should be less than or equal to the total number of basis, 2^n. The mixed state ρ_0 can fall into these t basis states $|\psi_i\rangle_{i\in[0,t-1]}$, but would never fall into the other $2^n - t$ states $|\psi_i\rangle_{i\in[t,2^n-1]}$. Therefore, these t states, from $|\psi_0\rangle$ to $|\psi_{t-1}\rangle$, are considered as the "correct" states and the assertion circuit should not raise an assertion error. However, the other $2^n - t$ states from $|\psi_t\rangle$ to $|\psi_{2^n-1}\rangle$ are considered as "incorrect" states and the assertion circuit should raise an assertion error for them. The unitary gate U^{-1} is generated to transform the states in the basis formed by $|\psi'_i\rangle$ to the computational basis. It is worth noting that by checking the "correct" states, we can guarantee the state under test is a probability mixture of t "correct" states when there is no assertion error. However, we can't guarantee the probability of each "correct" state is the same as the desired state. The prior work [30] also has the same limitation. In section IX-A2, we will show that our proposed mixed state assertion is capable of identifying program bugs.

In the second step, we discuss assertion circuit designs for different rank values, t:

1) $t = 2^m$ and $t \leq 2^{n-1}$, where m is an integer. Figure 7 shows the main idea of gate transformation and Figure 8 shows the assertion circuit. As shown in Figure 7, we can generate a unitary gate U^{-1}, which transforms the 2^m "correct" states to the states with the leading $n - m$ qubits in $|0\rangle$ state. Therefore, when we measure the leading $n - m$ qubits after unitary gate transformation, if the $n - m$ qubits are all in $|0\rangle$ state, the circuit won't raise an assertion error. If any of the leading $n - m$ qubits is in $|1\rangle$ state, the circuit will raise an assertion error.

As indicated in Figure 8, although the qubits under test are entangled with other qubits (therefore they are in a mixed state), after U^{-1}, the leading $n - m$ qubits are not entangled with any other qubits (as they are in $|0\rangle$ state). Therefore, they can be measured without affecting the existing entanglement.

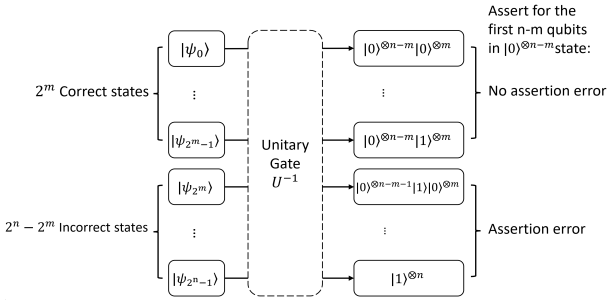


Fig. 7: The unitary gate transformation for mixed states when $t = 2^m$.

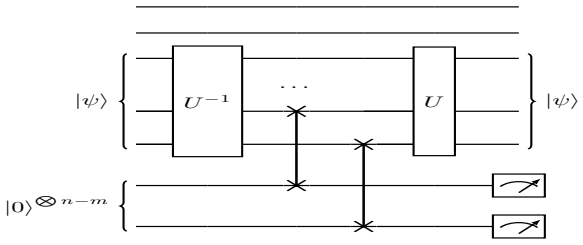


Fig. 8: General scheme of SWAP based assertion circuit for mixed state assertion when $t = 2^m$.

2) $2^m < t < 2^{m+1}$ and $t < 2^{n-1}$. In this case, we can take advantage of the assertion design for the case when $t = 2^m$ by finding two supersets of states. Either superset of states contains 2^{m+1} states and the t correct states are the intersection of these two supersets of states. By asserting for these two supersets of states, when there is no assertion error for both assertions, we can guarantee the state is within "correct" states. For example, we want to assert for a mixed state $\rho = 0.5\,|000\rangle\langle 000| + 0.25\,|001\rangle\langle 001| + 0.25\,|010\rangle\langle 010|$. Here, $t = 3$ and $2^1 < t < 2^{1+1}$, the "correct" states are $|000\rangle$, $|001\rangle$, and $|010\rangle$. We can first assert for the superset with 2^2 states:$\{|000\rangle$,$|001\rangle$,$|010\rangle$,$|011\rangle\}$. If there is no assertion error, we assert for another superset with 2^2 states:$\{|000\rangle$,$|001\rangle$,$|010\rangle$,$|100\rangle\}$. When both assertion don't raise an assertion error, the mixed state is a probability mixture of the "correct" states.

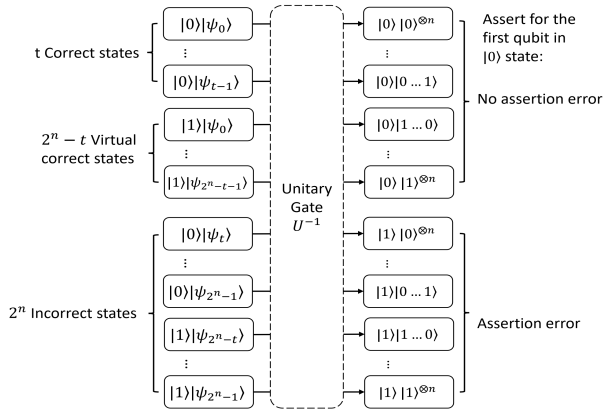


Fig. 9: The unitary gate transformation for mixed states when $2^{n-1} < t < 2^n$.

3) $2^{n-1} < t < 2^n$. As t increases, the number of "incorrect" states decreases and the number of qubits that we need to measure also decreases. When $t > 2^{n-1}$, the ratio of "incorrect" states is less than half and we do not have enough qubit to measure. Therefore, we need one extra ancilla qubits to enlarge the number of "incorrect" states. The ancilla qubit is initialized in $|0\rangle$ state. As shown in Figure 9, we pick some states as the "virtually correct" states. These "virtually correct" states won't raise an assertion error, but we also know they won't happen since the ancilla qubit is prepared in $|0\rangle$ state. The unitary gate transforms the union of "virtually correct" and "correct" states to the computational basis states with

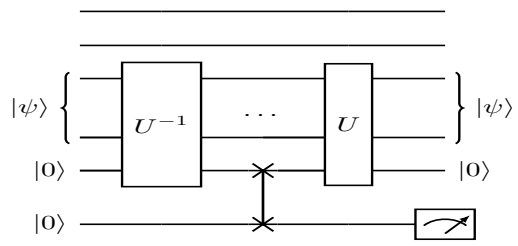


Fig. 10: General scheme of SWAP based assertion circuit for mixed state assertion when $2^{n-1} < t < 2^n$.

leading $|0\rangle$. Therefore, we can measure the leading qubit to assert for the union of "virtually correct" and "correct" states, which is functionally equivalent to asserting only the "correct" states.

Similarly, due to the measurement limitation, we need to swap the qubits with the ancilla qubits. When $t \leq 2^{n-1}$, the circuit design would be similar to the design in Figure 8. When $2^{n-1} < t < 2^n$ we need an extra qubit and the design is shown in Figure 10.

One corner case worth discussion is $t = 2^n$, where the mixed state is the probability mixture of all the 2^n basis states. In this case, all the basis states are marked as "correct" states, which means that we can't assert for this special mixed state.

D. Approximate Assertion

In Section III we discussed the idea of approximate assertion which asserts a set of states. We can see that asserting a set of states is actually similar to asserting mixed states. A mixed state is a probability mixture of pure states, which is analogous to a set of pure states. Therefore, we can reuse the same methodology as asserting mixed states. Given a set of quantum states $\{|\psi\rangle, |\phi\rangle, ...\}$, since the states may not be orthogonal, we can form a density matrix from these states with equal probability and use the same process to diagonalize it and calculate the orthonormal basis states. After finding the t orthonormal basis states, we can treat this problem as asserting for the mixed state composed of these t basis states. We mark these t basis states as "correct" states and follow the assertion process as described for mixed states.

E. Alternative Design: Logical OR Based Assertion Circuit

The circuit for asserting an n-qubit pure state requires n SWAP gates and n ancilla qubits. When the number n is large, the design becomes inefficient. Here, we present another design, which only requires one ancilla qubit. After the U^{-1} gate, the state of the qubits under test should be transformed to all $|0\rangle$ state. If any of them is in $|1\rangle$ state, the circuit should raise an assertion error. Therefore, we can apply logical OR gate to the qubits and store the result in one ancilla qubit. When the ancilla qubit is in $|0\rangle$ state, the circuit won't raise an assertion error. When the ancilla qubit is in $|1\rangle$ state, some of the qubits are in $|1\rangle$ state after U^{-1} gate and the circuit will raise an assertion error. The circuit design is shown in Figure 11. An n-qubit logical OR gate can be linearly decomposed into basic operators with one ancilla qubit [5], [24]

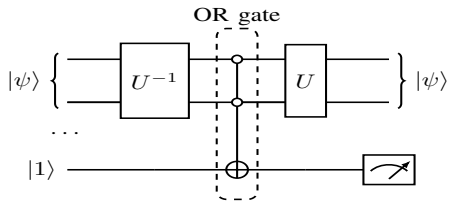


Fig. 11: Logical OR based assertion circuits.

Although both SWAP-based and logical OR-based designs can detect the incorrect states, their functionalities are slightly different. In the SWAP based assertion circuit, we prepare the desired state for the ancilla qubits and swap the qubits under test with the ancilla qubits. When the qubits under test are not in the desired state, after the assertion circuit, the state of the qubits will be "corrected" to the desired state. However, the logical OR based design doesn't have this property. When the ancilla qubit is in $|1\rangle$ state and an assertion error is raised, the qubits under test are not "corrected" by the assertion circuit. In the following discussion, we will find that the NDD based assertion circuit has the same property as this logical OR based design.

V. NDD-Based Assertion Circuit

Liu et.al. [32] proposed quantum assertion circuits for superposition states based on Non Destructive Discrimination (NDD). Here, we extend the idea to assert for pure and mixed states.

A. Precise Assertion for Pure States

The general scheme for asserting a pure state is shown in Figure 12. The design consists of a controlled-U gate and two Hadamard gates and is a phase-kickback circuit. It has the following properties: (i) when the quantum state $|\psi\rangle$ is the eigenstate of the unitary matrix U with the eigenvalue of 1, i.e., $|\psi\rangle = U|\psi\rangle$, the ancilla qubit's output state is $|0\rangle$. (ii) When the quantum state $|\psi\rangle$ is the eigenstate of the unitary matrix U with the eigenvalue of -1, the ancilla qubit's output state is $|1\rangle$. Based on these two properties, we can design the assertion circuit for asserting a pure state $|\psi_0\rangle$.

Similar to the SWAP based design, the first step of our design is to find an orthonormal basis $\{|\psi_i\rangle\}_{i\in[0,2^n-1]}$ that includes state $|\psi_0\rangle$. Since we want to distinguish between the "correct" state $|\psi_0\rangle$ and the other "incorrect" states $|\psi_i\rangle_{i\neq 0}$, we can find an unitary matrix U such that the "correct" state $|\psi_0\rangle$ is its eigenstate with the eigenvalue being 1 and the "incorrect" states $|\psi_i\rangle_{i\neq 0}$ are its eigenstates with eigenvalues being -1. Based on this property, when the qubits under test are in $|\psi_0\rangle$ state, we will get $|0\rangle$ state when measuring the ancilla qubit, i.e., the assertion circuit won't raise an assertion error. When the qubits under test are in any other state $|\psi_i\rangle_{i\neq 0}$, we will get $|1\rangle$ when measuring the ancilla qubit, i.e., the assertion circuit will raise an assertion error. Based on the spectral decomposition [20], the unitary matrix U can be calculated

as $U = |\psi_0\rangle\langle\psi_0| - \sum_{i=1}^{2^n-1} |\psi_i\rangle\langle\psi_i|$. The controlled-U gate can be generated by adding controls [43] to the unitary gate U.

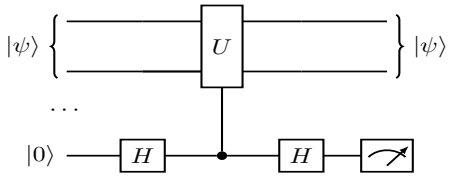


Fig. 12: General scheme of NDD-Based assertion circuits.

Except the assertion circuit shown in Figure 4, all the assertion circuits proposed in the prior work [32] can be categorized as NDD based assertion circuits. For example, in order to assert for classical state $|0\rangle$, we can find orthonormal basis $\{|0\rangle, |1\rangle\}$ that includes state $|0\rangle$. The matrix U can be calculated as $U = |0\rangle\langle 0| - |1\rangle\langle 1|$ = Z. Here Z is the matrix of the Pauli Z operator. Therefore, the controlled-U gate is indeed a controlled-Z gate. The corresponding assertion circuit is shown on the left side of Figure 13. The assertion circuit proposed by the prior work [32] is shown on the right side of Figure 13. It can be proven that these two circuits are equivalent based on the quantum circuit equivalence rules [21].

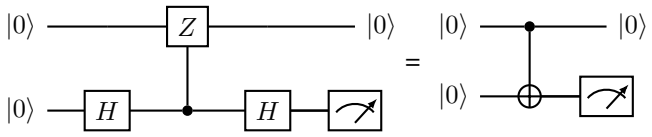


Fig. 13: Circuits for asserting $|0\rangle$ state: our circuit is on the left side; the one in prior work is on the right side.

B. Precise Assertion for Mixed States

The assertion scheme in Figure 12 can be generalized to assert for mixed states ρ_0. After performing SVD on the density matrix of the mixed state, we can find an orthonormal basis $\{|\psi_i\rangle\}_{i\in[0,2^n-1]}$ and represent the mixed state as a mixture of the basis states: $\rho_0 = \sum_{i=0}^{t-1} \lambda_i |\psi_i\rangle\langle\psi_i|$. The number of basis states t should be less than or equal to the total number of basis 2^n. Similar to the discussion in Section IV-C, the circuit should not raise an assertion error when the qubits under test are in the t "correct" states from $|\psi_0\rangle$ to $|\psi_{t-1}\rangle$. When the state belongs to the $2^n - t$ "incorrect" states from $|\psi_t\rangle$ to $|\psi_{2^n-1}\rangle$, the circuit should raise an assertion error. The assertion circuit structure is the same as the design for pure states shown in Figure 12. We can find a unitary matrix U for which the "correct" states are its eigenstate with the eigenvalue being 1, and the "incorrect" states are its eigenstates with the eigenvalues being -1. The unitary matrix U is calculated as $U = \sum_{i=0}^{t-1} |\psi_i\rangle\langle\psi_i| - \sum_{i=t}^{2^n-1} |\psi_i\rangle\langle\psi_i|$. The ancilla qubit will stay in $|0\rangle$, i.e., no assertion error, when the qubits under test is in a probability mixture of the "correct" states. The ancilla qubit will stay in $|1\rangle$, i.e., an assertion error, when the qubits under test is in a probability mixture containing "incorrect" states. Note that since the expected states are the eigenstates of U with eigenvalue of 1, the existing entanglement between the qubits under test and others are not affected.

C. Approximate Assertion

Similar to the discussion in Section IV-D, to assert a set of quantum states $\{|\psi\rangle, |\phi\rangle, ...\}$, we can form a density matrix and use the same process to diagonalize it and calculate the orthonormal basis states. After finding the t orthonormal basis states, we can treat this problem as asserting for the mixed state composed of these t basis states. We mark these t basis states as "correct" states and follow the assertion process as described for mixed states.

Based on the idea of parity check, the prior work [32] described the circuit for asserting entanglement state with even numbers of ones: $a|00\rangle + b|11\rangle$. Since coefficients a and b are not fixed, this can be considered as an approximate assertion for a set of states with even numbers of ones:$\{|00\rangle, |11\rangle\}$. We will show that our approximate assertion circuit and the circuit proposed in prior work are equivalent.

First, we form a density matrix from these states with equal probability: $\rho = \frac{1}{2}|00\rangle\langle 00| + \frac{1}{2}|11\rangle\langle 11|$. After performing SVD on the density matrix ρ, we can find an orthonormal basis $\{|00\rangle, |01\rangle, |10\rangle, |11\rangle\}$. Then, we mark $|00\rangle$ and $|11\rangle$ as "correct" states and $|01\rangle$ and $|10\rangle$ as "incorrect" states. The unitary matrix U is calculated as $U = |00\rangle\langle 00| + |11\rangle\langle 11| - |01\rangle\langle 01| - |10\rangle\langle 10| = Z \otimes Z$. Here Z is the matrix of the Pauli Z operator. The controlled-U gate is indeed two controlled-Z gates. The corresponding assertion circuit is shown on the left side of Figure 14. The assertion circuit proposed in the prior work [32] is shown on the right side of Figure 14. It can be proven that these two circuits are equivalent.

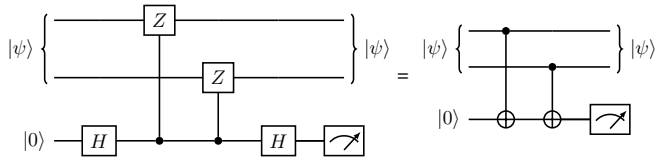


Fig. 14: Circuits for asserting $|\psi\rangle = a|00\rangle + b|11\rangle$: our circuit is on the left side; the one in prior work is on the right side.

VI. Design comparison

In this section, we compare the assertion coverage and circuit cost of different designs. We propose three different assertion circuit designs, namely the SWAP based assertion circuit, logical OR based assertion circuit, and NDD based assertion circuit. We compare them with the statistical assertion (Stat) [28], runtime assertion primitives (Primitive) [32], and projection-based assertion (Proq) [30].

A. Assertion Coverage

The statistical assertions and runtime assertion primitives support assertion for classical, superposition, and entanglement states. Nevertheless, the discussions of these three types of states are not complete. The superposition states with different relative phase have not been discussed in the statistical

assertion. Beyond that, some types of entanglement states have not been discussed either, for example, state $\frac{|00\rangle+e^{i\frac{\pi}{4}}|11\rangle}{\sqrt{2}}$. All these three types of quantum states can be represented by a state vector, which means that they are pure states. Proq proposed assertions for both pure states and mixed states but the assertion for a set of states has not been discussed. The coverages of different assertion designs are shown in Table II. Since our check for the mixed states cannot evaluate the probabilities, we mark the assertions for mixed states and for a set of states as "Part". The three designs proposed in this paper have the broadest coverage.

Assertion state type		Stat [28]	Primitive [32]	Proq [30]	SWAP based	logical OR based	NDD based
Pure states	Classical	ALL	ALL	ALL	ALL	ALL	ALL
	Superposition	Part	ALL	ALL	ALL	ALL	ALL
	Entanglement	Part	Part	ALL	ALL	ALL	ALL
	Other	N/A	N/A	ALL	ALL	ALL	ALL
Mixed states		N/A	N/A	Part	Part	Part	Part
Set of states		N/A	N/A	N/A	Part	Part	Part

TABLE II: Assertion coverage for different designs. ALL/Part/NA means the scheme supports all/part/none of the corresponding quantum states.

B. Assertion Circuit Cost

Since the runtime assertion circuits are inserted in the quantum programs, the circuit cost is an important factor. Intuitively, the SWAP-based assertion circuits may be costly since we are reverting the quantum state to the ground state and recomputing it and the cost of the matrix U may be similar to the quantum program itself. This, however, is not the case. For a quantum program, we don't know the answer and need to run the costly circuit to find the answer. In comparison, for asserting a known state, the circuit could be much smaller. As shown in the prior work [36], an n-qubit gate decomposition reaching the lower bound of $O(4^n)$ CNOT gates can lead to state preparation with $O(2^n)$ CNOT gates. Since the cost of the SWAP gates and the logical OR gates with one ancilla qubit both scale linearly $O(n)$, the resulting SWAP-based assertion circuit will have $O(2^n)$ CNOT gates which is much smaller than the original circuit. In addition, quantum machine learning [3], [39] has been used to prepare high fidelity quantum states with low depth circuits.

Given that the unitary gates in the designs need to be decomposed to the basis gates and different decomposition will result in different quantum gate counts, it is difficult to judge the circuit cost for an arbitrary state. Here, we present the circuit cost comparison for several common cases. We use four metrics: number of CNOT gates (#CX), number of single-qubit gates (#SG), number of ancilla qubits (#ancilla), number of measurements (#measure). The resulting cost of each circuit design is shown in Table III.

First, we discuss the cost for asserting arbitrary single-qubit pure states. The single-qubit state assertion is like the basic unit of assertions. The circuit cost for each design is shown in Table III. Even though the proq design has the least cost, it is not applicable to the recent quantum computers.When there is only one qubit under test, the logical OR gate in Figure 11 can be simplified to an open CNOT gate. The logical OR based assertion circuit consists of two single-qubit gates and a CNOT gate. Therefore, we choose logical OR based design for asserting single-qubit pure states.

Then, we discuss the cost for asserting arbitrary n-qubit separable state. The qubits in an n-qubit separable state are not entangled and the state can be expressed as the tensor product of individual single-qubit states:$|\psi\rangle = |\psi_0\rangle \otimes |\psi_1\rangle ... |\psi_{n-1}\rangle$. This kind of state often appears as the initial state or the output state of quantum programs. Since the qubits in this state are not entangled, asserting an n-qubit separable state is equivalent to asserting each qubit individually. For the logical OR based assertion circuit, the multi-qubit OR gate is indeed a multiple open-controlled Toffoli gate. There are multiple ways to decompose multi-controlled Toffoli gates [4], [33]. Hereby, we choose a simple design [35] which consists of 12n + 1 CNOT gates and 16n single-qubit gates. For the NDD based circuit, the number of CNOT gates in the circuit is not fixed, depending on the actual assertion state.

When asserting for the n-qubit entangled states with even numbers of ones, for example Bell state $\frac{|00\rangle+|11\rangle}{\sqrt{2}}$ and n-qubit GHZ state $\frac{|0\rangle^{\otimes n}+|1\rangle^{\otimes n}}{\sqrt{2}}$, the NDD based design has the least gate count among the assertion circuits we propose. The NDD based circuit only consists of n CNOT gates. When asserting for the entangled gates, the decomposition of unitary gate U in the SWAP based and logical OR based design will include at least one CNOT gates. Therefore, the gate count for asserting entanglement state should be greater than that of the separable state.

We will introduce the circuit design for asserting constant functions in Section X. In that case, the SWAP based design has the least number of gates.

Since each design performs the best for their special cases, none of the designs outperforms the rest for every situation. In our assertion function, the programmer can specify the assertion circuit design. The programmer can also let the assertion function to estimate the circuit cost of each design and select the design with the lowest cost.

VII. Methodology

Qiskit [2] is an open-source quantum computing framework. We augmented the Qiskit version 0.18.0 with the function to insert assertion circuits. We also provided several tools for finding orthonormal basis and calculating the number of basis states. The augmented version of Qiskit is publicly available [1]. The assertion function is in the form:

```
assert(circuit,qubitList,stateSet,design)
```

The assertion function takes four arguments. The first argument "circuit" specifies the quantum circuit. The second argument "qubitList" specifies a list of qubits under test. The "stateSet" argument represents a set of desired state variables. The state variable can be either a vector representing a pure state or a matrix representing a mixed state. If the size of the set is one, the assertion function will assert for that particular state. If the size of the set is greater than one, the assertion function will use approximate assertion to assert

	Primitive [32]			Proq [30]			SWAP based			Logical OR based			NDD based		
State	single	separable	even	single	separable	even	single	separable	even	single	separable	even	single	separable	even
#CX	2	2n	n	0	0	> 0	3	3n	$> 3n$	1	12n+1	$> 12n + 1$	2	State dependent	n
#SG	6	12n	0	2	2n	$\geq 2n$	2	2n	$\geq 2n$	2	16n	$\geq 16n$	6	State dependent	0
#ancilla	1	n	1	0	0	0	1	n	n	1	1	1	1	1	1
#measure	1	n	1	1	n	n	1	n	n	1	1	1	1	1	1

TABLE III: Circuit cost for different assertion circuit designs.

for the set of states. The "design" argument specifies the circuit design among the three different designs we proposed. If the "design" argument is set to NONE, the function will estimate the circuit cost of each design and select the design with the least CNOT gate count. We use the `UnitaryGate` function from Qiskit to generate gates from unitary matrices. We perform our experiments on a 15-qubit quantum computer *ibmq-melbourne* and the qasm simulator from Qiskit Aer. Each experiment is executed for 8192 shots.

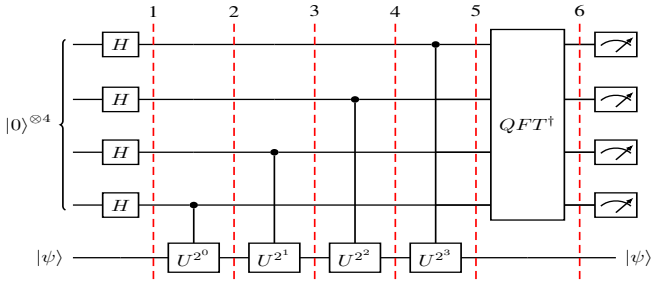


Fig. 15: Quantum phase estimation circuit.

VIII. General Applicability

Quantum programs usually operate on a finite number of qubits. These qubits only entangle with the other qubits within the system, thus the system stays in a pure state for every instruction. As a result, our systematic assertion scheme can essentially assert the state after every instruction. Using n-qubit QPE as an example, there are n+2 slots available for assertion as illustrated in Figure 15, which shows the 4-qubit QPE circuit. The corresponding code for the 4-qubit QPE algorithm is shown in Figure 16. In this example, all the assertions are precise assertions since we can calculate qubit state vectors beforehand. With the capability to assert for mixed states or a set of states, we also have the flexibility to choose different assertions, from single-qubit state assertion to multi-qubit state assertion, and from precise state assertion to approximate assertion.

While there is a variety of quantum programs, the quantum programs usually share common subroutines or program patterns. For example, the phase-kickback circuit is widely used in quantum algorithms such as Shor's algorithm [40], phase estimation algorithm [15], Deutsch algorithm [17], Bernstein-Vazirani algorithm [8], etc. In the next section, we will use the QPE circuit as a case study to introduce the debugging process for the phase-kickback subroutine. The debugging scheme is applicable to the other algorithms that contain the same subroutine. Similarly, many quantum programs also share the same program pattern. The prior paper [28] points out common program patterns such as superposition precondition, gates with recursion, and de-allocation of ancillary qubits. As another example, we illustrate debugging the gates with recursion in Appendix D.

IX. Debugging Case Study: Quantum Phase Estimation

In this section, we use quantum phase estimation (QPE) to showcase that our proposed circuits significantly improve the effectiveness of assertions. In subsection IX-A, we use several case studies to demonstrate the debugging process. In subsection IX-B, we run the experiments on a 15-qubit real quantum device to show our work is practical for near-term quantum computers.

```
1  #n−qubit quantum phase estimation
2  qr = QuantumRegister(4)
3  #ancilla qubit
4  ar = QuantumRegister(1)
5  cr = ClassicalRegister(4)
6  circuit = QuantumCircuit(qr, ar, cr)
7
8  # Precalculated state vectors for each assertion
9  stateVectorList = [V1, V2, ..., V6]
10
11 # Initialize the n−qubits to |+> state
12 circuit.h(qr[0:4])
13
14 # Assertion slot 1, we choose the logical OR based design:
15 qubitList = [qr[0],qr[1]...,qr[3], ar[0]]
16 assert(circuit, qubitList, set(V1), "ORbased")
17
18 # Controlled U^(2^3) gate
19 for j in range(4):
20     circuit.cu3(0, 0, 2^j × π/8, qr[j], ar[0])
21     #Assertion slot 2 to slot 5, the function choose best design
22     assert(circuit, qubitList, set(Vj+2), NONE)
23
24 # 4−qubit inverse QFT
25 iQFT(circuit, qr, 4)
26
27 # Assertion slot 6:
28 assert(circuit, qubitList, set(V6), NONE)
29
30 circuit.measure(qr, cr)
```

Fig. 16: The pseudo code for 4-qubit QPE with assertions.

A. Assertions for Program Debugging

Given a unitary operator U and its eigenstate $|\psi\rangle$, $U|\psi\rangle = e^{i\theta}|\psi\rangle$, QPE estimates the phase θ of the corresponding eigenvalue $e^{i\theta}$. The circuit consists of the superposition precondition (line 12), a phase-kickback subroutine (line 19-20) and an inverse QFT subroutine (line 25). Let's consider

the 4-qubit QPE program for which the unitary operator U is $u3(0,0,\frac{\pi}{8})$ gate operator with the eigenstates $|0\rangle$ and $|1\rangle$. The state $|\psi\rangle$ is in the superposition of the eigenstates: $|\psi\rangle = \frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle$. Assuming the following two types of bugs at line 20. **Bug1:** the programmer didn't include the loop index at line 20: $circuit.cu3(0,0,\frac{\pi}{8})$. Due to this mistake in rotation angles, slot3, 4, and 5 in Figure 15 will have incorrect quantum states. **Bug2:** the programmer missed the word "c" when coding for "cu3" gate, which makes all the controlled-u3 gates become single qubit gate u3. This is a bug due to choosing incorrect gates, and slots 2 to 5 will have incorrect quantum states. These two bugs can't be captured by the prior primitives since they do not support the assertion for the states from slots 2 to 5.

1) Pure State Assertion: We can assert for all the five qubits. For example, the qubit state at slot 5 in Figure 15 is a five qubit pure state:$|\phi_5\rangle = \frac{1}{\sqrt{2}}|++++\rangle|0\rangle + \frac{1}{\sqrt{2}}|\theta_4\rangle|1\rangle$, where $|\theta_4\rangle = (|0\rangle + e^{i\pi}|1\rangle)(|0\rangle + e^{\frac{i\pi}{2}}|1\rangle)(|0\rangle + e^{\frac{i\pi}{4}}|1\rangle)(|0\rangle + e^{\frac{i\pi}{8}}|1\rangle)$. For bug1, the assertion at slot 1, and 2 will not raise an assertion error, but the assertion at other slots will. Therefore, we can check the gates between slot 2 and slot 3 to locate the bug. For bug2, only the assertion at slot 1 will not raise an assertion error. Therefore, we can check the gates between slot 1 and slot 2 to pinpoint the bug. We verified the results on both a noise-free simulator and a noisy simulator to simulate the ideal and real cases.

2) Mixed State Assertion: The assertion at slot 5 asserts for a five-qubit pure state:$|\phi_5\rangle$. The circuit for asserting five qubits can be costly. For example, the SWAP based assertion circuit requires 26 CNOT gates. In this case, our mixed state assertion offers flexibility to assert for fewer qubits.

The first four qubits' state is a mixed state, and we can calculate it by taking the partial trace: $\rho_{1234} = tr_5(|\phi_5\rangle\langle\phi_5|)$ = $\frac{1}{2}|++++\rangle\langle++++| + \frac{1}{2}|\theta_4\rangle\langle\theta_4|$. Since the state $|\theta_4\rangle$ is orthogonal to state $|++++\rangle$, they are eigenvectors of the mixed state. We can diagonalize the density matrix ρ_{1234} to find the orthonormal basis that includes $|\theta_4\rangle$ and $|++++\rangle$ as basis states. We mark state $|++++\rangle$ and $|\theta_4\rangle$ as "correct" states, which also means that the number of decomposed basis states is 2. The circuit for asserting this four-qubit mixed state requires 20 CNOTs, less costly than asserting all five qubits.

Similar to our pure state discussion, we assume two types of bugs at line 20. When the loop index is not included (Bug1), the five qubit state at slot 5 would be a different state $|\phi_5'\rangle = \frac{1}{\sqrt{2}}|++++\rangle|0\rangle + \frac{1}{\sqrt{2}}|\theta_4'\rangle|1\rangle$, where $|\theta_4'\rangle = (|0\rangle + e^{\frac{i\pi}{8}}|1\rangle)(|0\rangle + e^{\frac{i\pi}{8}}|1\rangle)(|0\rangle + e^{\frac{i\pi}{8}}|1\rangle)(|0\rangle + e^{\frac{i\pi}{8}}|1\rangle)$. The four qubit mixed state will be $\rho_{1234}' = \frac{1}{2}|++++\rangle\langle++++| + \frac{1}{2}|\theta_4'\rangle\langle\theta_4'|$. When we represent the state with the basis states, we can find that besides basis state $|++++\rangle$ and $|\theta_4\rangle$, the state has other components, which means the mixed state has "incorrect" basis states. Our mixed state assertion circuit will raise an assertion error as a result. For bug2, the first four qubits are not entangled with the fifth qubit, and the state is $|++++\rangle$. Since it is the "correct" basis state, our mixed state assertion would not report an assertion error.

We can also assert mixed states with even fewer qubits to reduce the circuit cost. There's a tradeoff in using mixed state assertions: the fewer qubits we assert, the lower assertion circuit cost, but the higher chance to miss program bugs.

3) Approximate Assertion: We can use approximate assertion to reduce the circuit cost. For example, the assertion at slot 5 asserts for pure state $|\phi_5\rangle = \frac{1}{\sqrt{2}}|++++\rangle|0\rangle + \frac{1}{\sqrt{2}}|\theta_4\rangle|1\rangle$. We can use approximate assertion to assert for a set of two states: $\{|++++\rangle|0\rangle, |\theta_4\rangle|1\rangle\}$. Both of the bugs will lead to states outside this set, therefore, we can capture both bugs with less circuit cost using approximate assertion.

B. Experiment on Real Quantum Computer

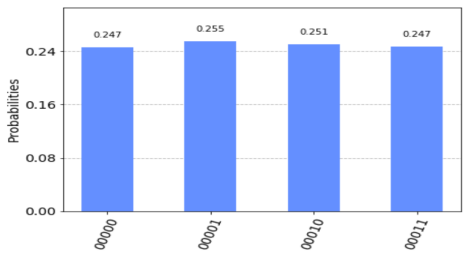
We run our experiment on a 15-qubit quantum computer *ibmq-melbourne*. When running the assertion circuit on real quantum computers, we actually check for the errors caused by both system noise and program bugs. To compare with the prior work [32], we assert for single-qubit pure state $|\psi\rangle$ at slot 6. We set the gate $cu3(2^j \times \frac{\pi}{8}, 0, 0)$ and the eigenstate $|\psi\rangle = \frac{1}{\sqrt{2}}|0\rangle + \frac{i}{\sqrt{2}}|1\rangle$. The swap-based design only requires two CNOTs and two single-qubit gates. On the other hand, the assertion primitives in the prior work requires two CNOTs and six single-qubit gates. Hereby, we reduced the number of single-qubit gates by four. When we don't introduce any bug, due to the system noise, 36% of the output states have assertion errors. When we insert a bug (changing the controlled-u3 gate to $circuit.cu3(0, 2^j \times \frac{\pi}{8}, 0, qr[j], ar[0])$), 45% of the output states have assertion errors. Therefore, we can detect the program bug based on the increment in the number of assertion errors. In comparison, the circuit using the assertion primitives in the prior work [32] has 42% assertion errors and 50% assertion errors after inserted a bug.

The prior work highlights that the assertion circuit can also be used to improve the success rate. The success rate of the original circuit is 19%. After filtering out the erroneous results using the assertion primitives, the success rate improves to 33%. In comparison, our lower-cost assertion circuit further improves the success rate to 36%.

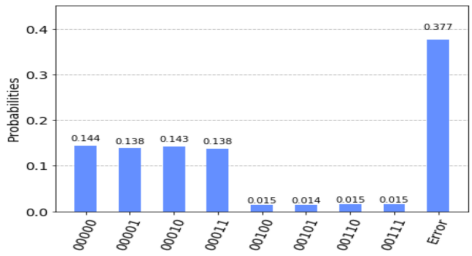
X. Debugging Case Study: Programs with Limited Knowledge of States

In this section, we use the Deutsch-Jozsa algorithm to show the effectiveness of approximate assertion for debugging programs with limited knowledge of states. None of the existing works supports this kind of assertion.

In the Deutsch-Jozsa algorithm [17], we are given a black-box function $f(x)$, which takes n-bit binary values as input and produces either 0 or 1 as output for each value. The function is guaranteed to be constant (output is always 0 or always 1) or balanced (returns 0 for half of the inputs and 1 for the rest of the inputs). When we try to debug the program, we can't predict the output of the black-box function $f(x)$. If a program bug happens in the black-box function $f(x)$, which makes the function neither constant nor balanced, the existing dynamic assertion schemes can't detect such a bug.



(a) Constant function



(b) Inconstant function

Fig. 17: The results for asserting the constant function and inconstant function.

We can assert the function in two steps: 1) initialize all the input qubits in $|+\rangle$ state, which is the superposition of all possible inputs, and 2) assert for the joint output state of the function: $|\psi\rangle = |x\rangle\,|f(x)\rangle$. When the function is constant, the joint output state is either $|x\rangle\,|0\rangle$ or $|x\rangle\,|1\rangle$, and these states form a set of states. Similarly, the balanced functions form another set of states. We can use approximate assertion to assert for the joint state $|\psi\rangle$ is in the constant state set, balanced state set, or the combined constant and balanced set.

For example, we have a hidden function $f(x)$ with two input qubits. We set the input state x in state $|{+}{+}\rangle$, which is the superposition of all possible inputs: $|{+}{+}\rangle = \frac{1}{2}(|00\rangle + |01\rangle + |10\rangle + |11\rangle)$. If the function is a constant function, the output qubit is either all $|0\rangle$ or all $|1\rangle$ for any possible input. The joint output state $|x\rangle\,|f(x)\rangle$ has two possibilities: $\frac{1}{2}(|000\rangle + |010\rangle + |100\rangle + |110\rangle)$ or $\frac{1}{2}(|001\rangle + |011\rangle + |101\rangle + |111\rangle)$. These two possible states are shown in the first two rows in Table IV. We can use the approximate assertion to assert for this set of constant output states. If the function is a balanced function, the possible output states are shown in Table IV from row 3 to row 8. We can use the approximate assertion function to assert for constant state set, balanced state set, or both of them.

We use the constant state set as an example to illustrate our assertion circuit designs. The first design is SWAP based assertion circuit. Our assertion function calculates that the unitary gate U in the SWAP based design consists of two Hadamard gates on the first two qubits $U = H \otimes H \otimes I$. The resulting assertion circuit is shown in Appendix C. It consists of four CNOT gates and four single-qubit gates. The second design is logical OR based assertion circuit. The unitary gate U is the same as the gate in the SWAP based design. The assertion circuit consists of six CNOT gates and twelve single-qubit gates. The third design is NDD based assertion circuit. After decomposing the controlled U gate, the assertion circuit consists of fourteen CNOT gates and twenty single-qubit gates. Among these three circuits, the SWAP based assertion circuit has the least number of gates. Therefore, we prefer to use the SWAP based assertion circuit for asserting n-qubit constant function.

If the function $f(x)$ is not a constant function due to program bugs, our assertion circuit will raise assertion errors. For example, when the output state is $\frac{1}{2}(|000\rangle + |010\rangle + |100\rangle + |111\rangle)$, $f(x)$ is zero for three input states and 1 for one input state. The function is neither constant nor balanced. We run the program with assertion for constant set on Qiskit simulator,

First two qubit state		$\|00\rangle$	$\|01\rangle$	$\|10\rangle$	$\|11\rangle$
Third qubit state	Constant	$\|0\rangle$	$\|0\rangle$	$\|0\rangle$	$\|0\rangle$
		$\|1\rangle$	$\|1\rangle$	$\|1\rangle$	$\|1\rangle$
	Balanced	$\|0\rangle$	$\|0\rangle$	$\|1\rangle$	$\|1\rangle$
		$\|0\rangle$	$\|1\rangle$	$\|0\rangle$	$\|1\rangle$
		$\|1\rangle$	$\|0\rangle$	$\|0\rangle$	$\|1\rangle$
		$\|0\rangle$	$\|1\rangle$	$\|1\rangle$	$\|0\rangle$
		$\|1\rangle$	$\|0\rangle$	$\|1\rangle$	$\|0\rangle$
		$\|1\rangle$	$\|1\rangle$	$\|0\rangle$	$\|0\rangle$

TABLE IV: Set of constant and balanced output states.

and the result is shown in Figure 17b. In comparison, the result when $f(x)$ is a constant function is shown in Figure 17a. In the figure, the first two qubits are the ancilla qubits. When the function is not a constant function, the ancilla qubit has a chance to be $|1\rangle$ state, and the circuit will raise assertion error. The reason that it does not raise an assertion error 100% of the time is that the state $\frac{1}{2}(|000\rangle + |010\rangle + |100\rangle + |111\rangle)$ is not orthogonal to the constant states meaning that it still has constant components.

XI. Conclusions

In this paper, we proposed two systematic approaches, namely SWAP-based and NDD-based designs for quantum state runtime assertion. The systematic approaches are capable of asserting a broader range of quantum states than existing works. We present detailed analysis of the efficiency and effectiveness of different assertion schemes. We also introduce the idea of approximate assertion, which performs membership check on a set of states. We showcase that both the precise and approximate assertions can help with debugging quantum programs.

Acknowledgment

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

References

[1] https://github.com/revilooliver/Systematic-Approaches-for-Precise-and-Approximate-Quantum-State-Runtime-Assertion, 2020.

[2] H. Abraham, I. Y. Akhalwaya, G. Aleksandrowicz, T. Alexander, G. Alexandrowics, E. Arbel, A. Asfaw, C. Azaustre, AzizNgoueya, P. Barkoutsos, G. Barron, L. Bello, Y. Ben-Haim, D. Bevenius, L. S. Bishop, S. Bolos, S. Bosch, S. Bravyi, D. Bucher, F. Cabrera, P. Calpin, L. Capelluto, J. Carballo, G. Carrascal, A. Chen, C.-F. Chen, R. Chen, J. M. Chow, C. Claus, C. Clauss, A. J. Cross, A. W. Cross, S. Cross, J. Cruz-Benito, C. Culver, A. D. Córcoles-Gonzales, S. Dague, T. E. Dandachi, M. Dartiailh, DavideFrr, A. R. Davila, D. Ding, J. Doi, E. Drechsler, Drew, E. Dumitrescu, K. Dumon, I. Duran, K. EL-Safty, E. Eastman, P. Eendebak, D. Egger, M. Everitt, P. M. Fernández, A. H. Ferrera, A. Frisch, A. Fuhrer, M. GEORGE, J. Gacon, Gadi,

B. G. Gago, J. M. Gambetta, A. Gammanpila, L. Garcia, S. Garion, J. Gomez-Mosquera, S. de la Puente González, J. Gorzinski, I. Gould, D. Greenberg, D. Grinko, W. Guan, J. A. Gunnels, M. Haglund, I. Haide, I. Hamamura, V. Havlicek, J. Hellmers, Ł. Herok, S. Hillmich, H. Horii, C. Howington, S. Hu, W. Hu, H. Imai, T. Imamichi, K. Ishizaki, R. Iten, T. Itoko, A. Javadi, A. Javadi-Abhari, Jessica, K. Johns, T. Kachmann, N. Kanazawa, Kang-Bae, A. Karazeev, P. Kassebaum, S. King, Knabberjoe, A. Kovyrshin, R. Krishnakumar, V. Krishnan, K. Krsulich, G. Kus, R. LaRose, R. Lambert, J. Latone, S. Lawrence, D. Liu, P. Liu, Y. Maeng, A. Malyshev, J. Marecek, M. Marques, D. Mathews, A. Matsuo, D. T. McClure, C. McGarry, D. McKay, D. McPherson, S. Meesala, M. Mevissen, A. Mezzacapo, R. Midha, Z. Minev, A. Mitchell, N. Moll, M. D. Mooring, R. Morales, N. Moran, P. Murali, J. Müggenburg, D. Nadlinger, K. Nakanishi, G. Nannicini, P. Nation, Y. Naveh, P. Neuweiler, P. Niroula, H. Norlen, L. J. O'Riordan, O. Ogunbayo, P. Ollitrault, S. Oud, D. Padilha, H. Paik, S. Perriello, A. Phan, F. Piro, M. Pistoia, A. Pozas-iKerstjens, V. Prutyanov, D. Puzzuoli, J. Pérez, Quintiii, R. Raymond, R. M.-C. Redondo, M. Reuter, J. Rice, D. M. Rodríguez, RohithKarur, M. Rossmannek, M. Ryu, T. SAPV, SamFerracin, M. Sandberg, H. Sargsyan, N. Sathaye, B. Schmitt, C. Schnabel, Z. Schoenfeld, T. L. Scholten, E. Schoute, J. Schwarm, I. F. Sertage, K. Setia, N. Shammah, Y. Shi, A. Silva, A. Simonetto, N. Singstock, Y. Siraichi, I. Sitdikov, S. Sivarajah, M. B. Sletfjerding, J. A. Smolin, M. Soeken, I. O. Sokolov, SooluThomas, D. Steenken, M. Stypulkoski, J. Suen, K. J. Sung, H. Takahashi, I. Tavernelli, C. Taylor, P. Taylour, S. Thomas, M. Tillet, M. Tod, E. de la Torre, K. Trabing, M. Treinish, TrishaPe, W. Turner, Y. Vaknin, C. R. Valcarce, F. Varchon, A. C. Vazquez, D. Vogt-Lee, C. Vuillot, J. Weaver, R. Wieczorek, J. A. Wildstrom, R. Wille, E. Winston, J. J. Woehr, S. Woerner, R. Woo, C. J. Wood, R. Wood, S. Wood, J. Wootton, D. Yeralin, R. Young, J. Yu, C. Zachow, L. Zdanski, C. Zoufal, Zoufalc, a matsuo, azulehner, bcamorrison, brandhsn, chlorophyll zz, dan1pal, dime10, drholmie, elfrocampeador, enavarro51, faisaldebouni, fanizzamarco, gadial, gruu, kanejess, klinvill, kurarrr, lerongil, ma5x, merav aharoni, michelle4654, ordmoj, sethmerkel, strickroman, sumitpuri, tigerjack, toural, vvilpas, welien, willhbang, yang.luh, yelojakit, and yotamvakninibm, "Qiskit: An open-source framework for quantum computing," 2019.

[3] J. M. Arrazola, T. R. Bromley, J. Izaac, C. R. Myers, K. Brádler, and N. Killoran, "Machine learning method for state preparation and gate synthesis on photonic quantum computers," *Quantum Science and Technology*, vol. 4, no. 2, p. 024004, 2019.

[4] J. M. Baker, C. Duckering, A. Hoover, and F. T. Chong, "Decomposing quantum generalized toffoli with an arbitrary number of ancilla," *arXiv preprint arXiv:1904.01671*, 2019.

[5] A. Barenco, C. H. Bennett, R. Cleve, D. P. DiVincenzo, N. Margolus, P. Shor, T. Sleator, J. A. Smolin, and H. Weinfurter, "Elementary gates for quantum computation," *Physical review A*, vol. 52, no. 5, p. 3457, 1995.

[6] J. S. Bell, "On the einstein podolsky rosen paradox," *Physics Physique Fizika*, vol. 1, no. 3, p. 195, 1964.

[7] C. H. Bennett, G. Brassard, C. Crépeau, R. Jozsa, A. Peres, and W. K. Wootters, "Teleporting an unknown quantum state via dual classical and einstein-podolsky-rosen channels," *Physical review letters*, vol. 70, no. 13, p. 1895, 1993.

[8] E. Bernstein and U. Vazirani, "Quantum complexity theory," *SIAM Journal on computing*, vol. 26, no. 5, pp. 1411–1473, 1997.

[9] J. Biamonte, P. Wittek, N. Pancotti, P. Rebentrost, N. Wiebe, and S. Lloyd, "Quantum machine learning," *Nature*, vol. 549, no. 7671, pp. 195–202, 2017.

[10] B. H. Bloom, "Space/time trade-offs in hash coding with allowable errors," *Commun. ACM*, vol. 13, no. 7, pp. 422–426, 1970. [Online]. Available: https://doi.org/10.1145/362686.362692

[11] K. Blum, *Density matrix theory and applications*. Springer Science & Business Media, 2012, vol. 64.

[12] H. J. Briegel and R. Raussendorf, "Persistent entanglement in arrays of interacting particles," *Physical Review Letters*, vol. 86, no. 5, p. 910, 2001.

[13] H. Buhrman, R. Cleve, J. Watrous, and R. de Wolf, "Quantum fingerprinting," *Physical Review Letters*, vol. 87, no. 16, Sep 2001. [Online]. Available: http://dx.doi.org/10.1103/PhysRevLett.87.167902

[14] S. Choudhury, S. Muralidharan, and P. K. Panigrahi, "Quantum teleportation and state sharing using a genuinely entangled six-qubit state," *Journal of Physics A: Mathematical and Theoretical*, vol. 42, no. 11, p. 115303, 2009.

[15] R. Cleve, A. Ekert, C. Macchiavello, and M. Mosca, "Quantum algorithms revisited," *Proceedings of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences*, vol. 454, no. 1969, pp. 339–354, 1998.

[16] I. Corporation, "Gate and operation specification for quantum circuits," https://github.com/Qiskit/openqasm, 2019, [Online; accessed 3-9-2020].

[17] D. Deutsch and R. Jozsa, "Rapid solution of problems by quantum computation," *Proceedings of the Royal Society of London. Series A: Mathematical and Physical Sciences*, vol. 439, no. 1907, pp. 553–558, 1992.

[18] M. Ekerå and J. Håstad, "Quantum algorithms for computing short discrete logarithms and factoring rsa integers," in *International Workshop on Post-Quantum Cryptography*. Springer, 2017, pp. 347–363.

[19] A. Farouk, J. Batle, M. Elhoseny, M. Naseri, M. Lone, A. Fedorov, M. Alkhambashi, S. H. Ahmed, and M. Abdel-Aty, "Robust general n user authentication scheme in a centralized quantum communication network via generalized ghz states," *Frontiers of Physics*, vol. 13, no. 2, p. 130306, 2018.

[20] J. N. Franklin, *Matrix theory*. Courier Corporation, 2012.

[21] J. C. Garcia-Escartin and P. Chamorro-Posada, "Equivalent quantum circuits," *arXiv preprint arXiv:1110.2998*, 2011.

[22] G. H. Golub and C. F. Van Loan, *Matrix computations*. JHU press, 2012, vol. 3.

[23] D. M. Greenberger, M. A. Horne, and A. Zeilinger, "Going beyond bell's theorem," in *Bell's theorem, quantum theory and conceptions of the universe*. Springer, 1989, pp. 69–72.

[24] Y. He, M.-X. Luo, E. Zhang, H.-K. Wang, and X.-F. Wang, "Decompositions of n-qubit toffoli gates with linear circuit complexity," *International Journal of Theoretical Physics*, vol. 56, no. 7, pp. 2350–2361, 2017.

[25] C. Hempel, C. Maier, J. Romero, J. McClean, T. Monz, H. Shen, P. Jurcevic, B. P. Lanyon, P. Love, R. Babbush *et al.*, "Quantum chemistry calculations on a trapped-ion quantum simulator," *Physical Review X*, vol. 8, no. 3, p. 031022, 2018.

[26] R. A. Horn and C. R. Johnson, "Norms for vectors and matrices," *Matrix analysis*, pp. 313–386, 1990.

[27] Y. Huang and M. Martonosi, "Qdb: From quantum algorithms towards correct quantum programs," *arXiv preprint arXiv:1811.05447*, 2018.

[28] Y. Huang and M. Martonosi, "Statistical assertions for validating patterns and finding bugs in quantum programs," in *Proceedings of the 46th International Symposium on Computer Architecture*, 2019, pp. 541–553.

[29] S. Jain, S. Muralidharan, and P. K. Panigrahi, "Secure quantum conversation through non-destructive discrimination of highly entangled multipartite states," *EPL (Europhysics Letters)*, vol. 87, no. 6, p. 60008, 2009.

[30] G. Li, L. Zhou, N. Yu, Y. Ding, M. Ying, and Y. Xie, "Proq: Projection-based runtime assertions for debugging on a quantum computer," *arXiv preprint arXiv:1911.12855*, 2019.

[31] J. Liu, L. Bello, and H. Zhou, "Relaxed peephole optimization: A novel compiler optimization for quantum circuits," in *Proceedings of the 2021 International Symposium on Code Generation and Optimization*, 2021.

[32] J. Liu, G. T. Byrd, and H. Zhou, "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*, 2020, pp. 1017–1030.

[33] D. M. Miller, R. Wille, and Z. Sasanian, "Elementary quantum gate realizations for multiple-control toffoli gates," in *2011 41st IEEE International Symposium on Multiple-Valued Logic*. IEEE, 2011, pp. 288–293.

[34] S. Muralidharan and P. K. Panigrahi, "Perfect teleportation, quantum-state sharing, and superdense coding through a genuinely entangled five-qubit state," *Physical Review A*, vol. 77, no. 3, p. 032321, 2008.

[35] M. A. Nielsen and I. Chuang, "Quantum computation and quantum information," 2002.

[36] M. Plesch and Č. Brukner, "Quantum-state preparation with universal gate decompositions," *Physical Review A*, vol. 83, no. 3, p. 032302, 2011.

[37] E. Rieffel and W. Polak, "An introduction to quantum computing for non-physicists," *ACM Computing Surveys (CSUR)*, vol. 32, no. 3, pp. 300–335, 2000.

[38] Rigetti, "A Python library for quantum programming using Quil," https://github.com/rigetti/pyquil, 2019, [Online; accessed 3-9-2020].

[39] K. K. Sabapathy, H. Qi, J. Izaac, and C. Weedbrook, "Production of photonic universal quantum gates enhanced by machine learning," *Physical Review A*, vol. 100, no. 1, p. 012326, 2019.
[40] P. W. Shor, "Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer," *SIAM review*, vol. 41, no. 2, pp. 303–332, 1999.
[41] I. Q. team, "IBM Q 16 Melbourne V1.1.0 backend specification V1.1.0," https://quantum-computing.ibm.com, 2019, [Online; accessed 3-9-2020].
[42] R. F. team, "Forest SDK," https://www.rigetti.com/forest, 2019, [Online; accessed 3-9-2020].
[43] X.-Q. Zhou, T. C. Ralph, P. Kalasuwan, M. Zhang, A. Peruzzo, B. P. Lanyon, and J. L. O'brien, "Adding control to arbitrary unknown quantum operations," *Nature communications*, vol. 2, no. 1, pp. 1–8, 2011.

APPENDIX A
ASSERTION CIRCUIT EQUIVALENCE

The prior work [21] has proved the two circuits shown in Figure 18 are equivalent.

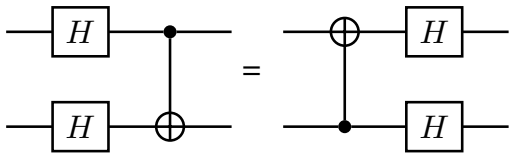


Fig. 18: H gates mirroring through CNOT inversion

Based on the circuit equivalence, we can transform our assertion circuit step by step as shown in Figure 19. In the last step, since the first CNOT gate is controlling over the $|0\rangle$ state, it can be substituted with a quantum wire, the resulting circuit is equivalent to the assertion circuit proposed by prior work [32].

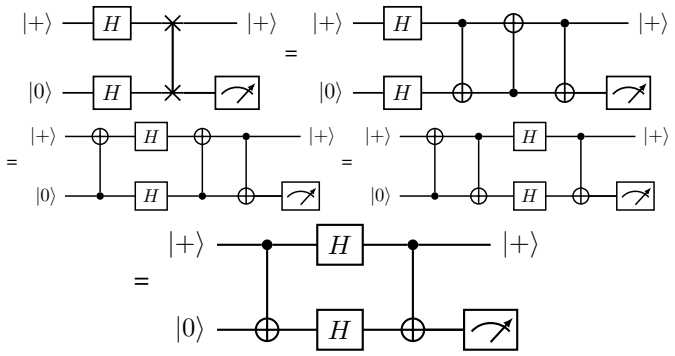


Fig. 19: Circuit transformation of the assertion circuit for $|+\rangle$ state

APPENDIX B
ORTHONORMAL BASIS TRANSFORMATION

Proposition: In an n-qubit quantum system, given two orthonormal quantum bases $\{|\psi_i\rangle\}_{i\in[0,2^n-1]}$ and $\{|\psi_i'\rangle\}_{i\in[0,2^n-1]}$, we can always find a unitary gate U that transforms each basis state $|\psi_i'\rangle$ to the corresponding state $|\psi_i\rangle$ in the other basis. In other words, $|\psi_i\rangle = U|\psi_i'\rangle$ for every i in $[0, 2^n - 1]$. The unitary gate U can be calculated as $U = \sum_{i=0}^{2^n-1} |\psi_i\rangle \langle\psi_i'|$.

Proof: An orthonormal basis $\{|\psi_i\rangle\}$ have following properties: (1)$\langle\psi_i|\psi_j\rangle = 1$ for all $i = j$, and $\langle\psi_i|\psi_j\rangle = 0$ for all $i \neq j$. (2)$\sum_{i=0}^{2^n-1} |\psi_i\rangle \langle\psi_i| = I$, here I is the identity matrix. First, we need to prove $|\psi\rangle = U|\psi_i'\rangle$ for every $i \in [0, 2^n-1]$. Based on the first property, we have $U|\psi_i'\rangle = \sum_{j=0}^{2^n-1} |\psi_j\rangle\langle\psi_j'|\psi_i'\rangle = |\psi_i\rangle \langle\psi_i'|\psi_i'\rangle + \sum_{i\neq j} |\psi_j\rangle\langle\psi_j'|\psi_i'\rangle = |\psi_i\rangle$. Then, we need to prove that the matrix U is a unitary matrix such that we can find a corresponding gate. Based on the first and the second property, we have $UU^\dagger = \sum_{i=0}^{2^n-1} |\psi_i\rangle \langle\psi_i'| |\psi_i'\rangle \langle\psi_i| = \sum_{i=0}^{2^n-1} |\psi_i\rangle \langle\psi_i| = I$. Similarly, we can prove $U^\dagger U = I$. Therefore, U is a unitary matrix.

The computational basis is indeed an orthonormal basis. In our assertion circuit, we transform an orthonormal basis that includes the assertion state $|\psi\rangle$ to the computational basis. Note that we can change the order of basis states in computational basis to form a new correspondence, the unitary gate U will change accordingly.

APPENDIX C
SWAP BASED CIRCUIT FOR ASSERTING CONSTANT SET

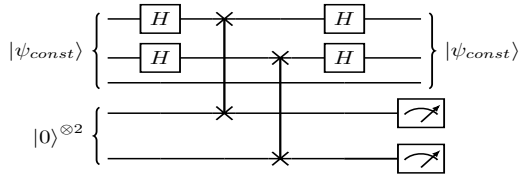


Fig. 20: SWAP based circuit for asserting constant set

APPENDIX D
DEBUGGING CASE STUDY: CONTROLLED ADDER SUBROUTINE

In quantum computing, it is a common practice to use recursion of quantum operations. Bugs may happen when coding for such recursive patterns. In Figure 21, we show the pseudo code for controlled adder using QFT. The program computes $qr = a + qr$, where a is a constant integer with length "width" and qr is an integer encoded in Fourier space with "width" numbers of qubits.

```
1  def controlled_adder(num_ctrl, qr_ctrl0, qr_ctrl1, width, a, qr):
2      for i in range(width−1, −1, −1):
3          for j in range(i, −1, −1):
4              if (a >> j) & 1: //shift out bits in constant a
5                  angle = np.pi/ pow(2, i−j) //rotation angle
6                  if num_ctrl is 0:
7                      circuit.rz(qr[i], angle)
8                  elif num_ctrl is 1:
9                      circuit.crz(qr_ctrl0, qr[i], angle)
10                 elif num_ctrl is 2:
11                     circuit.ccrz(qr_ctrl0, qr_ctrl1, qr[i], angle)
12     return circuit
```

Fig. 21: The pseudo code for controlled adder using QFT

Since quantum algorithms need varying control and target qubits, the same subroutine might occur multiple times with different numbers of controlled qubits. For example, line 7, 9, and 11 in Figure 21 are the codes for rz gate with different control qubits. Now, let's consider a bug happens in line 11, where the programmer used variable j instead of i. This bug leads to an incorrect entangled state and it can't be asserted by the prior primitives. We can insert precise pure state assertions after each rz, crz or $ccrz$ gate to capture this bug. Since i and j are the same for the first rz gate and they start to differ from the second rz gate, asserting the state after the second rz gate should be sufficient to capture the bug. The bug also leads to changes in the mixed states of a subset of qubits, thus it is also detectable with mixed state assertions. Similarly, the incorrect entanglement will lead to states out of the expected set of states and the bug is detectable with approximate assertions as well. We confirmed such analysis with experiments using a quantum simulator.