Received November 5, 2020, accepted November 18, 2020, date of publication December 8, 2020, date of current version December 31, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3043187

Reverse Checking of Quantum Algorithm Execution

CHANG LIU, (Senior Member, IEEE)
School of Electrical Engineering and Computer Science, Ohio University, Athens, Athens, OH 45701, USA
e-mail: liuc@ohio.edu

ABSTRACT Verification of quantum computation is critical because undesirable interference and noise are major technical hurdles in quantum computing. We propose an approach for reverse checking of computation results that takes advantage of quantum teleportation and reversibility of unitary quantum gates. The main idea is to preserve the quantum state after the computation ends and before the result is measured, so that quantum teleportation can be performed to "save" the quantum state before the readout. After that, the computation is reversely performed in the reverse order of gate operations. The end result should match the original input. Any discrepancy would be proof of errors during computation or verification. The advantage of this approach is that the reverse computation circuit can be automatically generated and performed and that the error rate obtained reflects what happened during the actual computation. In addition, this approach leads to a potential way to reduce error rates in the future by discarding results from individual shots of the execution with detected errors.

INDEX TERMS Quantum computing.

I. INTRODUCTION

Because of the unobservable nature of qubits, the result of quantum computation is obtained through statistical measurement. In other words, even though quantum computation can be powerful and precise in nature, the result can only be associated with high probability, not certainty. To further complicate the matter, interference is a major technical obstacle in all current physical realizations of quantum computation. Even the best implementations of quantum computation today are still very noisy, so much so that Preskill referred to the immediate goal of the quantum computing community as the Noisy Intermediate-Scale Quantum (NISQ) technology [1]. Verification of quantum computation therefore is critical before quantum computers can be used to solve any real problems [2].

Current verification techniques are often realized through significantly more additional qubits and/or additional quantum circuit gates employed just for this purpose, which necessarily requires additional computational resources (often a lot more qubits) and/or additional computational time. We propose an on-the-fly checking technique that only minimally delays the availability of results because the checking is performed after the algorithm is complete. It only requires two times more qubits than what was needed for the original algorithm at any given time. This technique takes advantage of the fact that quantum computation is inherently reversible. All unitary gates in quantum circuits can be reversed. Therefore, after the original computation ends, the quantum circuit, including the data preparation circuit, is reversed. The output is used to compute the input backwards, which should all be $|0\rangle$ as is the case for most quantum algorithms before data preparation. Regardless, in all cases, the output should match the original classical input.

While the reversibility of unitary gates in quantum circuits is common knowledge, the application of this principle in error checking and possibly error rate reduction in the future is novel.

To make the content of this paper more accessible to software engineering practitioners, in the next section we introduce the basics of quantum computing in a self-contained way understandable to an average software engineer or a software engineering researcher, without the need to understand the underlying quantum physics. After that, reverse checking and an example on IBM Quantum Experience [3], [4] are presented. Related work and future work are discussed in the end.

The associate editor coordinating the review of this manuscript and approving it for publication was Wei Huang.

II. QUANTUM COMPUTATION: A PRIMER FOR SOFTWARE ENGINEERS

Quantum computation was built on a clear, well-defined computational model of qubits based on quantum mechanics theories [5]. To a software engineer, the key to understand the power of quantum computing is the mathematical model of qubits, not various physical realization methods of qubits, such as electrons in superconductors [6], photons [7], [8], trapped ions, anyons [9], or any other forms of particles exhibiting quantum behaviors. Quantum entanglement, the no-cloning theorem, the Heisenberg uncertainty principle, the quantum observer effect, and other quantum properties can all be deduced from this mathematical model. From that perspective, there is nothing weird or counter-intuitive about quantum computing. They all follow the rule of math.

A. QUBITS AND QUANTUM GATES

Qubits hold information in quantum computing. The quantum state of a single qubit Ψ, denoted in Dirac notation (a.k.a. bra–ket notation) as $|\Psi\rangle$, can be described as a function of pure states $|0\rangle$ and $|1\rangle$. It is commonly represented by a unit vector of two complex number coefficients. This column vector of size two is called a *spinor* and can be equivalently expressed as a point on the surface of a unit 2-sphere named Bloch sphere. The two coefficients represent amplitudes of waves as qubits possess wave–particle duality.

$$|\Psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle \equiv \begin{pmatrix}\alpha\\ \beta\end{pmatrix},$$

where $|\alpha|^2 + |\beta|^2 = 1$ Frequently used quantum states include:

$|0\rangle \equiv \begin{pmatrix}1\\ 0\end{pmatrix}$, a pure state,

$|1\rangle \equiv \begin{pmatrix}0\\ 1\end{pmatrix}$, the other pure state,

$|+\rangle \equiv \begin{pmatrix}\frac{1}{\sqrt{2}}\\ \frac{1}{\sqrt{2}}\end{pmatrix}$, an equal-chance superposition state, and

$|-\rangle \equiv \begin{pmatrix}\frac{1}{\sqrt{2}}\\ -\frac{1}{\sqrt{2}}\end{pmatrix}$, another equal-chance superposition state

Quantum states are not directly observable. Physics laws only allow measurements that would reveal either 0 or 1, the chances of which are determined by the angle or the plane of measurement and by the spinors of the quantum states. Measurements necessarily collapse the quantum states and turn qubits into classical bits.

A single-qubit quantum gate is a two-by-two matrix that operates on quantum states and effectively manipulate the spinors in Bloch spheres by rotating them around various axes. Intuitively, this is why all unitary quantum gates are reversible as they can be simply rotated back to the original locations.

The resultant new quantum state $|\Psi_1\rangle$ of a gate operation is the matrix product of the gate G and the original quantum state $|\Psi_0\rangle$. Quantum computation is in essence a sequence of quantum gate operations.

$$G\,|\Psi_0\rangle \equiv \begin{pmatrix}a & b\\ c & d\end{pmatrix}\begin{pmatrix}\alpha_0\\ \beta_0\end{pmatrix} = \begin{pmatrix}a\alpha_0 + b\beta_0\\ c\alpha_0 + d\beta_0\end{pmatrix} = \begin{pmatrix}\alpha_1\\ \beta_1\end{pmatrix} \equiv |\Psi_1\rangle$$

A Hadamard gate H turns a qubit from $|0\rangle$ or $|1\rangle$ to a superposition with equal chance of $|0\rangle$ or $|1\rangle$.

$$H = \begin{pmatrix}\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}}\\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}}\end{pmatrix} = \frac{1}{\sqrt{2}}\begin{pmatrix}1 & 1\\ 1 & -1\end{pmatrix}$$

It is easy to see that $H \cdot H = \begin{pmatrix}1 & 0\\ 0 & 1\end{pmatrix} = I$, and that

$$\begin{aligned}
H\,|0\rangle &= \frac{1}{\sqrt{2}}\begin{pmatrix}1 & 1\\ 1 & -1\end{pmatrix}\begin{pmatrix}1\\ 0\end{pmatrix} = \begin{pmatrix}\frac{1}{\sqrt{2}}\\ \frac{1}{\sqrt{2}}\end{pmatrix} = |+\rangle\\
&= \frac{|0\rangle + |1\rangle}{\sqrt{2}}\\
H\,|1\rangle &= \frac{1}{\sqrt{2}}\begin{pmatrix}1 & 1\\ 1 & -1\end{pmatrix}\begin{pmatrix}0\\ 1\end{pmatrix} = \begin{pmatrix}\frac{1}{\sqrt{2}}\\ -\frac{1}{\sqrt{2}}\end{pmatrix} = |-\rangle\\
&= \frac{|0\rangle - |1\rangle}{\sqrt{2}}\\
H\,|+\rangle &= \frac{1}{\sqrt{2}}\begin{pmatrix}1 & 1\\ 1 & -1\end{pmatrix}\begin{pmatrix}\frac{1}{\sqrt{2}}\\ \frac{1}{\sqrt{2}}\end{pmatrix}\\
&= \begin{pmatrix}1\\ 0\end{pmatrix} = |0\rangle = \frac{|+\rangle + |-\rangle}{\sqrt{2}}\\
H\,|-\rangle &= \frac{1}{\sqrt{2}}\begin{pmatrix}1 & 1\\ 1 & -1\end{pmatrix}\begin{pmatrix}\frac{1}{\sqrt{2}}\\ -\frac{1}{\sqrt{2}}\end{pmatrix} = \begin{pmatrix}0\\ 1\end{pmatrix} = |1\rangle\\
&= \frac{|+\rangle - |-\rangle}{\sqrt{2}}
\end{aligned}$$

Similarly, the quantum state $|\Psi\rangle$ of two qubits can be described as a function of $|00\rangle$, $|01\rangle$, $|10\rangle$, and $|11\rangle$, where $|00\rangle \equiv |0\rangle\,|0\rangle$, representing a two-qubit system in a state where both the first qubit and the second qubit are in pure state $|0\rangle$. The quantum state $|\Psi\rangle$ can also be represented by a column vector of four coefficients as in the following formula.

$$|\Psi\rangle = \alpha_{00}\,|00\rangle + \alpha_{01}\,|01\rangle + \alpha_{10}\,|10\rangle + \alpha_{11}\,|11\rangle \equiv \begin{pmatrix}\alpha_{00}\\ \alpha_{01}\\ \alpha_{10}\\ \alpha_{11}\end{pmatrix}$$

As an example,

$$\begin{aligned}
|+\rangle\,|0\rangle &= (\frac{1}{\sqrt{2}}\,|0\rangle + \frac{1}{\sqrt{2}}\,|1\rangle)(|0\rangle) = \frac{1}{\sqrt{2}}\,|00\rangle + \frac{1}{\sqrt{2}}\,|10\rangle\\
&\equiv \begin{pmatrix}\frac{1}{\sqrt{2}}\\ 0\\ \frac{1}{\sqrt{2}}\\ 0\end{pmatrix}
\end{aligned}$$

B. QUANTUM ENTANGLEMENT

A Controlled Not gate (a.k.a. *CX* gate), which is a double-qubit gate, can be combined with a H gate to entangle two input pure state qubits, as shown in Figure 1. The *CX* gate operates on two qubits. One is called the control qubit. The other is called the target qubit. When the control qubit is not $|1\rangle$, the target qubit stays the same; when the control qubit is not $|1\rangle$, the target qubit flips. Here's the mathematical definition of the *CX* gate:

$$CX = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

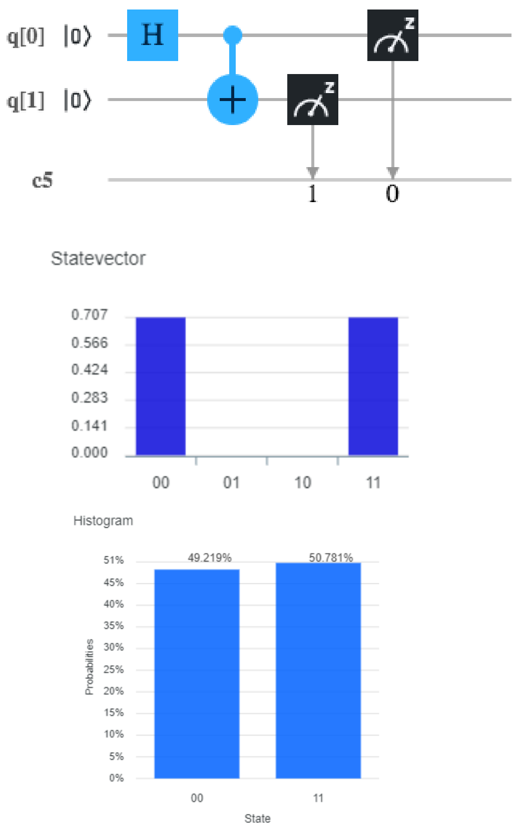


FIGURE 1. A quantum circuit that entangles two qubits. It consists of a Hadamard gate and a Controlled Not gate, represented by the plus sign with a line connected to the controlling qubit. The two black icons on the right side of the circuit represent qubit measurement.

Therefore,

$$CX((H|0\rangle)|0\rangle)) = CX(|+\rangle|0\rangle))$$
$$= \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} \frac{1}{\sqrt{2}} \\ 0 \\ \frac{1}{\sqrt{2}} \\ 0 \end{pmatrix}$$

$$= \begin{pmatrix} \frac{1}{\sqrt{2}} \\ 0 \\ 0 \\ \frac{1}{\sqrt{2}} \end{pmatrix} = \frac{1}{\sqrt{2}}|00\rangle + \frac{1}{\sqrt{2}}|11\rangle$$
$$\approx 0.707\,|00\rangle + 0.707\,|11\rangle$$

This means equal chance for $|00\rangle$ and $|11\rangle$ and no chance for $|01\rangle$ and $|10\rangle$. In other words, the two quits are maximally entangled. This is consistent with the state vector and the histogram in Figure 1.

Entanglement can also be achieved by applying a controlled phase gate on two $|+\rangle$ superposition states. Because controlled phase gates are symmetry and do not distinguish their two inputs, they can be used to create massively entangled cluster states.

A controlled phase shift (by π radians) gate is:

$$CP = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \end{pmatrix}$$
$$|+\rangle|+\rangle = (\frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle)^2$$
$$= \frac{1}{2}|00\rangle + \frac{1}{2}|01\rangle + \frac{1}{2}|10\rangle + \frac{1}{2}|11\rangle$$
$$= \frac{1}{2}(|00\rangle + |01\rangle + |10\rangle + |11\rangle)$$

$$CP(|+\rangle|+\rangle) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \end{pmatrix} \begin{pmatrix} \frac{1}{2} \\ \frac{1}{2} \\ \frac{1}{2} \\ \frac{1}{2} \end{pmatrix} = \begin{pmatrix} \frac{1}{2} \\ \frac{1}{2} \\ \frac{1}{2} \\ -\frac{1}{2} \end{pmatrix}$$
$$\equiv \frac{1}{2}(|00\rangle + |01\rangle + |10\rangle - |11\rangle)$$
$$= \frac{1}{2}(|0\rangle|+\rangle + |1\rangle|-\rangle)$$

The vectors here are not normalized to unit vectors for simplicity.

This shows that the resulting state is an entangled state if the first qubit is measured in the standard basis and the second qubit is measured in the sign basis of $|+\rangle$ and $|-\rangle$.

C. QUANTUM TELEPORTATION

Quantum teleportation sends the complete information of a qubit to another one. Due to the quantum no-cloning theorem, the original qubit has to be destroyed through measurement for this to happen. Nevertheless, there is a well-known method to use quantum entanglement to achieve quantum teleportation. Suppose Alice has a qubit ($q0_0$ in Figure 2) to teleport to Bob. Alice and Bob need to each have one of an entangled pair of qubits ($q0_1$ and $q0_2$ in Figure 2).

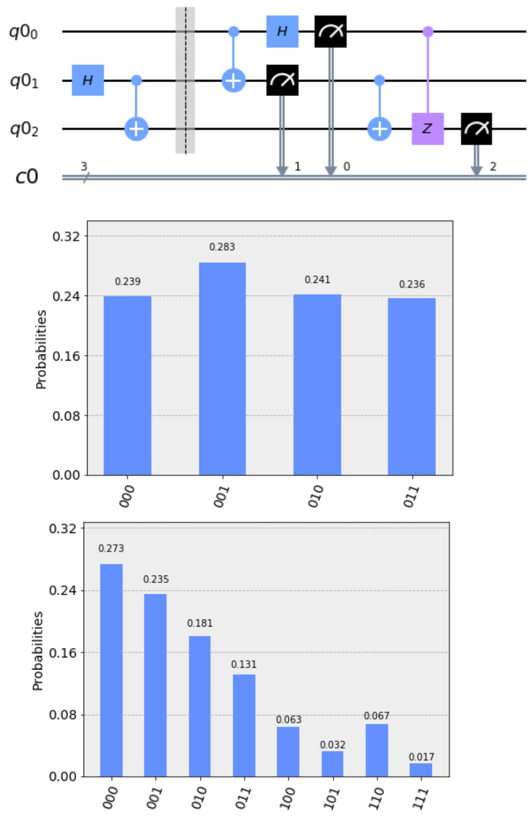


FIGURE 2. A quantum teleportation circuit [10] that teleports qubit $q0_0$ to $q0_2$ and the histograms of the execution results. The first histogram showed the result from an IBM Q simulator. The second histogram showed the result from a physical IBM Q quantum computer.

The H gate and the CX gate to the left of the dotted line are used to establish this entanglement. The CX gate is represented by a connection between two qubits. The control qubit is connected through a solid dot. The target qubit is connected through a dot with a plus sign.

Next, the second CX gate and the second H gate (which are to the right of the dotted line) are used to prepare for the teleportation. After that, Alice reads out her two qubits (and thus makes them collapse). The classical readout is sent to Bob so that Bob can use them as the control qubits to perform a Controlled Not operation and a Controlled Z operation on his qubit $q0_2$. The resulting qubit would be identical to Alice's original qubit $q0_0$. The first histogram in Figure 2 is from a noiseless simulator. Notice that $q0_2$ readout is always 0. The second histogram is from a physical IBM Q quantum computer and shows a significant noise level, as indicated by the four columns on the right with $q0_2$ being 1.

Here is how quantum teleportation worked mathematically. The three-qubit system starts as $|\Psi_0\rangle = |000\rangle$. After the entanglement of two qubits, it becomes

$$|\Psi_1\rangle = \frac{1}{\sqrt{2}}|000\rangle + \frac{1}{\sqrt{2}}|011\rangle .$$

The second CX gate does not change anything because $q0_0$ is $|0\rangle$. The second H gate turns $q0_0$ to a superposition. The new state of the system becomes

$$\begin{aligned}|\Psi_2\rangle &= (\frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle)(\frac{1}{\sqrt{2}}|00\rangle + \frac{1}{\sqrt{2}}|11\rangle)\\ &= \frac{1}{2}(|000\rangle + |011\rangle + |100\rangle + |111\rangle).\end{aligned}$$

The readout on $q0_1$ is then used to control a Not operation on $q0_2$, turning the state into

$$|\Psi_3\rangle = \frac{1}{2}(|000\rangle + |01\underline{0}\rangle + |100\rangle + |11\underline{0}\rangle).$$

The two underlined digits are flipped as a result. Because $q0_0$ is 0, the last Z gate does not do anything in this case. In the end, $q0_2$ becomes 0, matching the original $q0_0$. This explains how the teleportation worked when $q0_0$ started as $|0\rangle$. For more general cases, refer to the calculation in [5] (p. 27).

III. REVERSE CHECKING

We propose an approach to reversely check the result of quantum computation. Next, we explain this approach in two cases.

A. REVERSE CHECKING OF A ONE-QUBIT SYSTEM

For arbitrary computation on a one-qubit system, after the computation is completed, a quantum teleportation can be applied to preserve the quantum state after the readout. The readout from the teleportation operation can then be used as the result of the computation. There is only a delay of two gate operations (a CX gate and a H gate) in obtaining this result. After that all gate operations in the original computation can be reversed one by one. The end result should be the same as the original input, which is typically $|0\rangle$. If it matches, the computation result is checked and can be trusted with more confidence. Note that it is still possible for interference to cancel out and leave a matching result. If not, an error in either the original computation or the verification has occurred for sure. An error rate can be computed to indicate the noise level.

To test this approach, we performed a three-gate random unitary operation on a single qubit on IBM Q. The three gates were randomly selected from the following gates through Python code: X, Y, Z, H, and T.

Since $H \cdot H = I$, applying H again reverses it. The X, Y, Z gates are *Pauli*-X, -Y, -Z gates and are rotations through π radians around the x-, y-, and z-axes, respectively. Therefore, applying them again also reverses the gate operations. The T gate, a.k.a. the $\frac{\pi}{8}$ gate, is a $\frac{\pi}{4}$ rotation around the z-axis. Its inverse $T^\dagger$ is a $-\frac{\pi}{4}$ (45 degree) rotation around the z-axis. It is obvious that applying T eight times would rotate the quantum state 360 degrees and back to the original. T and $T^\dagger$ gates are commonly used in fault-tolerant quantum computation.

$$T = \begin{pmatrix}1 & 0\\ 0 & e^{\frac{i\pi}{4}}\end{pmatrix},$$

$$T^\dagger = \begin{pmatrix}1 & 0\\ 0 & e^{-\frac{i\pi}{4}}\end{pmatrix},$$

$$
\begin{aligned}
T \cdot T^{\dagger} &= \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} = I_2 \\
T^{\otimes 4} &= \begin{pmatrix} 1 & 0 \\ 0 & e^{\frac{i\pi}{4}} \end{pmatrix}^{\otimes 4} = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\pi} \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} = Z \\
T^{\otimes 8} &= \begin{pmatrix} 1 & 0 \\ 0 & e^{\frac{i\pi}{4}} \end{pmatrix}^{\otimes 8} = \begin{pmatrix} 1 & 0 \\ 0 & e^{i2\pi} \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & e^{0} \end{pmatrix} \\
&= \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} = I_2
\end{aligned}
$$

In the example execution shown in Figure 3, Z, H, T were selected by a random number generator in a Python library. Based on the calibration profile of the IBM Q Burlington computer as shown in Figure 4 and Table 1, the average readout error rate was about 4.93%; the average single-qubit error rate was about 0.0533%; the average CNOT error rate was about 1.37%. In the example quantum circuit in Figure 3, there were a total of three measurement gates, three CNOT gates, and nine single-qubit gates. The error rate of the entire computation according to the calibration profile should be $1-(1-0.0493)^3 * (1-0.0137)^3 * (1-0.000533)^9 = 0.179509201$, or about 17.95%.

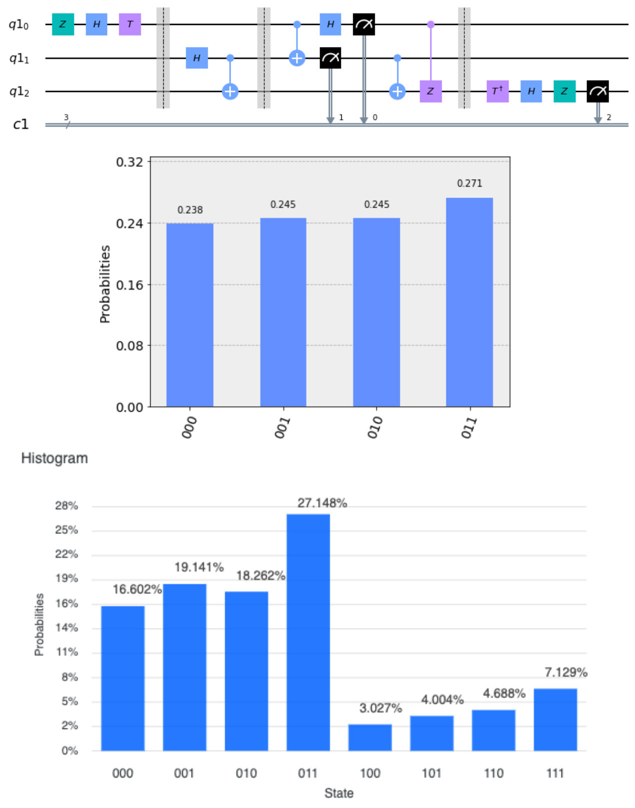


FIGURE 3. A randomly generated three-gate quantum circuit with the on-the-fly reverse checking circuit. The result on the top was from a noiseless IBM Q simulator. The result on the bottom was from a physical IBM Q Quantum Computer named *IBM Q Burlington*.

At the end of the three-gate computation, $q1_0$ was teleported to $q1_2$. Qubit $q1_0$ was then measured to obtain the result of the computation. We then reversed the computation

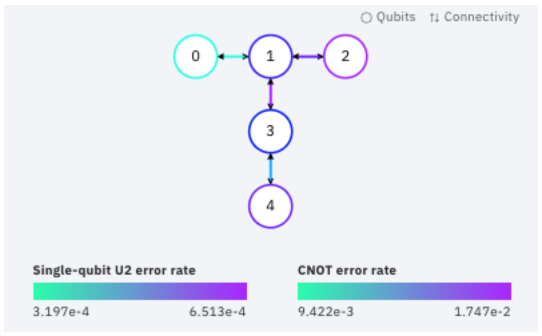


FIGURE 4. IBM Q Burlington architecture, a screenshot from IBM Q.

to perform the reverse of the three gates in reverse order as $T^\dagger, H, Z$. After that, we checked the result, as shown in the two histograms in Figure 3. The histogram on the left was the result from an IBM Q simulator, which was noiseless. It was clear that $q1_2$ was always 0 in the end, matching the initial value of $q1_0$. The error rate was 0. Therefore, there were no noise or errors detected.

B. REVERSE CHECKING OF A N-QUBIT SYSTEM

For an arbitrary algorithm using n qubits, one can do the same for each of the qubits in the system simultaneously after the computation ends but before the result were measured. If an algorithm involves non-reversible measurement operations, our approach will have to be modified to remain effective. Specifically, before any measurement operation, a quantum teleportation must be performed to ‘‘save’’ the full state of the qubit so that it can be used in the reverse computation.

Figure 5 shows an example two-qubit algorithm using the CX, H, T, and S gates. The addition of the CX gate here is important because it is a two-qubit gate, different from the one-qubit gates used in the previous example. CX is

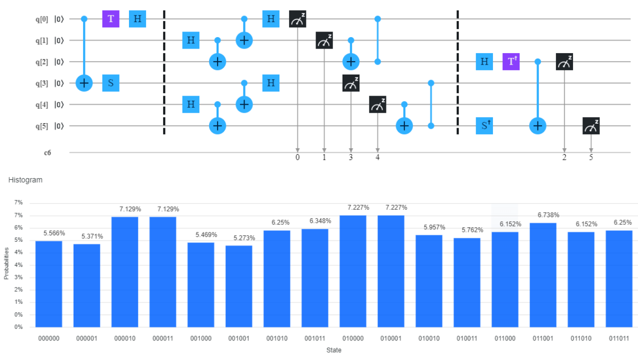


FIGURE 5. A six-qubit quantum circuit for two-qubit computation and four-qubit on-the-fly reverse checking circuit. q[0] and q[1] were for the original computation. q[0] was teleported to q[3] and q[1] to q[5] for reverse checking. The result from an IBM Q simulator showed that q[2] and q[5] were both 0 in the end. q[0] is the one that is furthest to the right on the state. The vertical lines with two solid dots on both ends are an alternative representation for Controlled Z gates.

TABLE 1. **Calibration profile of the IBM Q Burlinton 5-Qubit quantum computer (from IBM Q).**

Qubit	Energy relaxation time T1 (μs)	Dephasing time T2 (μs)	Frequency (GHz)	Readout error rate	Single-qubit U2 error rate	CNOT error rate
Q0	98.33551265	46.82808058	4.641200919	5.50000e-2	3.19722e-4	cx0_1: 9.422e-3
Q1	66.80352570	84.08716319	4.719992563	6.05e-2	5.76695e-4	cx1_0: 9.422e-3 cx1_2: 1.594e-2 cx1_3: 1.747e-2
Q2	77.96055113	88.27547132	4.761962111	4.04999e-2	6.51329e-4	cx2_1: 1.594e-2
Q3	93.81392649	86.04122557	4.687003753	5.04999e-2	5.11595e-4	cx3_1: 1.747e-2 cx3_4: 1.208e-2
Q4	57.44885729	40.31605568	4.923978085	4.00000e-2	6.05968e-4	cx4_3: 1.208e-2
Ave.	78.87247466	69.10959927	4.746827487	4.93e-2	5.33062e-4	1.37e-2

reversible by applying it again.

$$CX \cdot CX = \begin{pmatrix}1&0&0&0\\0&1&0&0\\0&0&0&1\\0&0&1&0\end{pmatrix}\begin{pmatrix}1&0&0&0\\0&1&0&0\\0&0&0&1\\0&0&1&0\end{pmatrix} = \begin{pmatrix}1&0&0&0\\0&1&0&0\\0&0&1&0\\0&0&0&1\end{pmatrix} = I_4$$

The S gate is a $\frac{\pi}{2}$ phase shift gate, equivalent to T^2. S can be reversed by the $S^\dagger$ gate, which is a $-\frac{\pi}{2}$ phase shift gate.

$$S = \begin{pmatrix}1&0\\0&i\end{pmatrix}, S^\dagger = \begin{pmatrix}1&0\\0&-i\end{pmatrix}, S \cdot S^\dagger = \begin{pmatrix}1&0\\0&1\end{pmatrix} = I_2$$

CX, H, T, and S gates form a universal gate set, which means any other two-qubit gates can be approximated to arbitrary accuracy through a combination of these gates [5] (p. 188).

The result of the example two-qubit algorithm in Figure 5 was preserved with two quantum teleportation operations (q[0] to q[2] and q[3] to q[5]). Then reverse checking was performed by applying the H, $S^\dagger$, $T^\dagger$, and CX gates on the corresponding qubits (q[2] and q[5]) in reverse order. The result of the 6-qubit system should have $2^6 = 64$ states. The actual histogram from a IBM Q simulator indicated that 48 of those states, where q[2] or q[5] were not 0, were never reached. It showed that all 16 remaining states had significant chances. This confirmed that the noiseless simulator did perform the computation without introducing any noise. Unfortunately, at the time of the writing, we only had access to 5-qubit physical quantum computers and were not able to perform this experiment on a physical quantum computer with at least six qubits. We did run simulator-based experiments of reverse checking of three-qubit and four-qubit algorithms, as listed in the Appendix.

IV. RELATED WORK

In the strictest sense, verification is to prove the result of an algorithm to be correct. This is difficult to do for quantum algorithms because they are specifically designed to solve hard problems not feasible to solve on classical computers. This means that one cannot simply run a classical computer side-by-side with a quantum computer working on the same problem and use the classical output to verify the quantum output, because the classical computer would not be able to keep up as the size of the input grows.

Efforts have been made to use different approaches to mitigate this problem. Verification of quantum algorithms typically require one or more separate computers, either quantum or classic, connected through classical or quantum communication channels [2], [11], [12]. Some used a smaller quantum computer for the verification. Some used interactive proof systems where there were a verifier and a prover.

In [13], the verification of the algorithm was performed on a classical supercomputer simulator capable of simulating the evolution of the full quantum state for smaller numbers of qubits (up to 43 qubits). For larger numbers of qubits, Google data centers were used to collectively simulate the quantum state using a hybrid algorithm. Positive outcomes for smaller inputs helped build confidence to trust the quantum algorithm when it was applied to larger inputs (53 qubits) that could not be effectively verified on classical computers.

Mahadev [12] proposed a measurement protocol for a classical verifier to use a quantum prover as a trusted measurement device to perform the verification. The goal was to interactively verify the result of an efficient quantum computation.

In our approach, we focus on undesirable interference and noise detection and do not perform verification in its strictest sense. In fact, our approach does not have the capability to detect any errors in the logic of a quantum algorithm. What it can detect effectively is noise levels and errors introduced by the quantum computer, not in the algorithm design. In other words, we check algorithm execution results, but not the algorithm design itself. What we do check is as critical in quantum computing because today's quantum computers all suffer from noise issues.

If verification in general is considered to be result checking in the large (i.e. checking the result for the entire computation), quantum error correction can be considered result checking in the small. Quantum error correction (e.g. quantum stabilizer codes) is a way to deal with noise and maintain the correct quantum state for individual qubits [14]. This is often done on the hardware design level and takes place in a black box from the perspective of quantum software

```
from qiskit import(
  QuantumCircuit,
  execute,
  Aer)
from qiskit.visualization import plot_histogram

simulator = Aer.get_backend('qasm_simulator')

qubitNumber = 3
circuit = QuantumCircuit(qubitNumber*3, qubitNumber*3)

circuit.cx(0, 3)
circuit.t(0)
circuit.s(3)
circuit.z(6)
circuit.cx(3, 6)
circuit.h(0)
circuit.barrier([0,1,2,3,4,5,6,7,8]);

for i in range(0, qubitNumber):
  circuit.h(3*i+1)
  circuit.cx(3*i+1, 3*i+2)
  circuit.barrier([3*i+0,3*i+1,3*i+2]);

for i in range(0, qubitNumber):
  circuit.cx(3*i, 3*i+1)
  circuit.h(3*i)

circuit.barrier([0,1,2,3,4,5,6,7,8]);
for i in range(0, qubitNumber):
  circuit.measure([3*i+0,3*i+1], [3*i+0,3*i+1])
  circuit.cx(3*i+1, 3*i+2)
  circuit.cz(3*i,3*i+2)

circuit.barrier([0,1,2,3,4,5,6,7,8]);
circuit.h(2)
circuit.cx(5, 8)
circuit.z(8)
circuit.sdg(5)
circuit.tdg(2)
circuit.cx(2, 5)
circuit.measure([2,5,8], [2,5,8])

job = execute(circuit, simulator, shots=1000)
result = job.result()
counts = result.get_counts(circuit)
circuit.draw(fold=-1, output="text")
plot_histogram(counts, color='midnightblue', title="State", figsize= (30,10))
```

Listing. 1. **Python code for reverse checking of a three-qubit quantum algorithm.**

programmers. Our approach does not check or correct individual qubits as in most quantum error correction work. We look for errors and measure error rates for the entire algorithm on the quantum program level. Our approach can be used in combination with those qubit-level error correction mechanisms, which will be treated as part of the invisible internals of qubits.

Shor showed how fault tolerant quantum computing can be performed [15]. Harper and Flammia [16] discussed fault-tolerant logical gates in the IBM Quantum experience in the context of the 4, 2, 2-concatenated toric code [17], [18], a type of quantum code on a lattice with boundary [19].

Attempts have been made to enhance quantum state preservation through additional error-correcting qubits using surface code architectures, e.g. parity measurement on a 5-qubit plaquette of four data qubits plus one syndrome qubit [20].

V. FUTURE WORK

In the future, we are certainly eager to perform reverse checking of algorithms with two or more qubits on physical quantum computers. Separately, the current experiment was run on IBM Q Experience, 1024 shots at a time. The error rate was calculated for all 1024 shots. The obtained error rate was informative, but our approach currently do not help reduce the

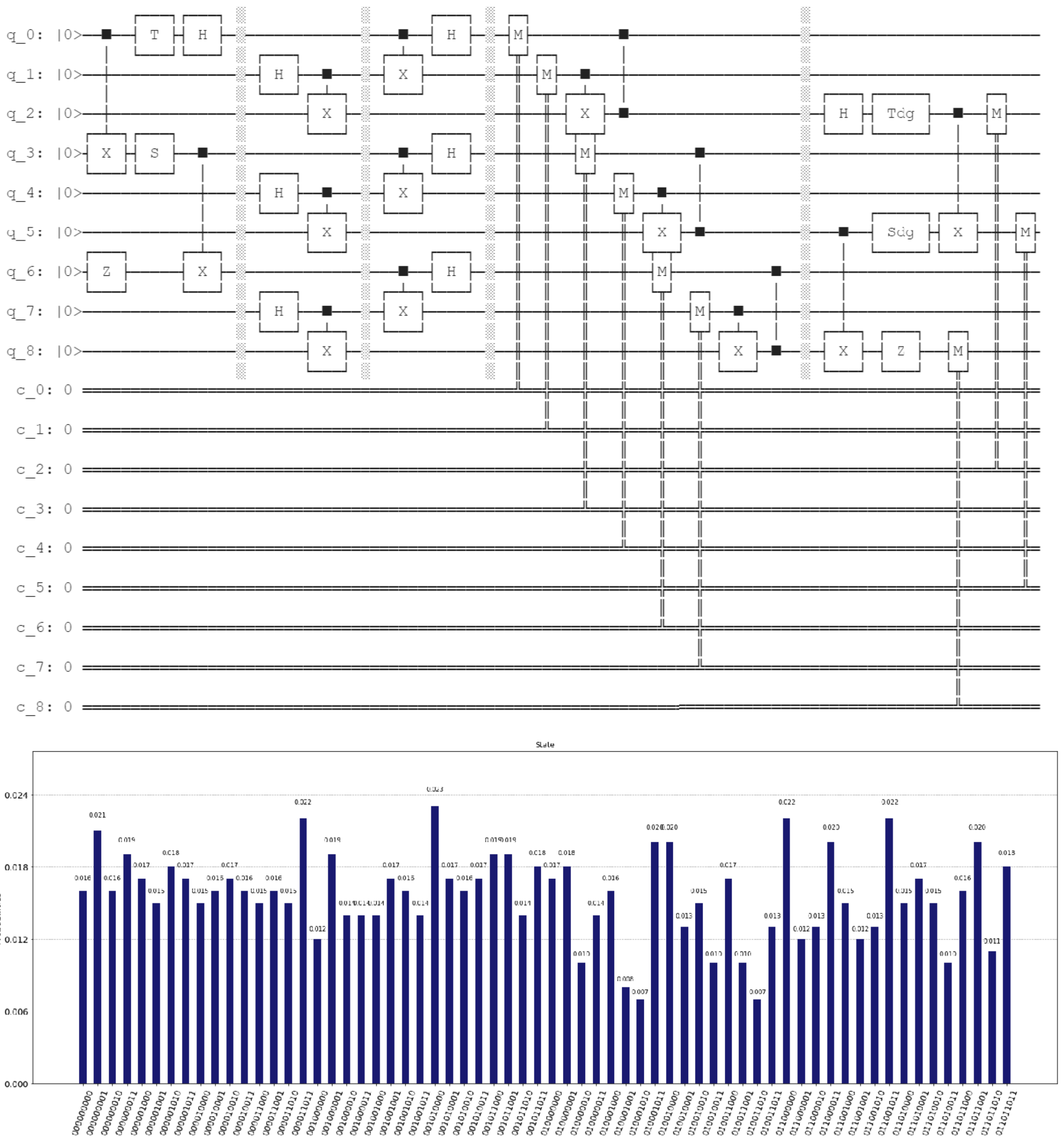


FIGURE 6. A nine-qubit quantum circuit for three-qubit computation and six-qubit on-the-fly reverse checking circuit. Qubits q_0, q_3, and q_6 were for the original computation. Qubit q_0 was teleported to q_2, q_3 to q_5, and q_6 to q_8 for reverse checking. The simulation result using the IBM Qiskit Python Package showed that q_2, q_5, and q_8 were all 0 in the end. Out of $2^9 = 512$ states, only $\frac{1}{2^3}$ of that, or 64 states, were possible, as shown in the histogram. Qubit q_0 is the one that is furthest to the right on the state.

error rate. In the future, one potential way to improve this is to run the reverse verification one shot at a time. If the reverse computation outcome matches the original input, the original computation result will be accepted; if the verification outcome does not match the input, the computation result can be discarded. This way, instead of learning and having to accept an error rate of 18.848% as in our example in Section III, we can discard 18.848% of the result with detected errors and use only the remaining ones as the result of the entire computation. While it is still possible for errors to remain, the error rate could be dramatically reduced, which we plan to work on next to demonstrate.

VI. SUMMARY

In summary, we presented a unique approach for one-the-fly checking of quantum algorithm results based on quantum teleportation and reverse quantum gate operations. An experiment on IBM Q showed that this approach measured actual error rates reliably. In the future, we may be able to improve error rates by refining this method.

APPENDIX

The Python code in Code Listing 1 using the IBM Qiskit package was used in the experiment. The number of qubits is controlled by the variable qubitNumber. It can be set to four or another bigger number instead of three, as long as one has the computational resource to perform the simulation or the actual computation. The result of the three-qubit experiment is presented in Figure 6. We have also performed reverse checking of a four-qubit algorithm. That result is not included due to space constraints. The experiments were performed on Google Colaboratory at https://colab.research.google.com/.

ACKNOWLEDGMENT

Krerkkiat Chusap assisted in performing an experiment for the reverse checking of a three-qubit algorithm.

REFERENCES

[1] J. Preskill, "Quantum computing in the NISQ era and beyond," *Quantum*, vol. 2, p. 79, Aug. 2018.

[2] A. Gheorghiu, T. Kapourniotis, and E. Kashefi, "Verification of quantum computation: An overview of existing approaches," *Theory Comput. Syst.*, vol. 63, no. 4, pp. 715–808, May 2019.

[3] S. J. Devitt, "Performing quantum computing experiments in the cloud," *Phys. Rev. A, Gen. Phys.*, vol. 94, no. 3, Sep. 2016, Art. no. 032329.

[4] Y. Wang, Y. Li, Z.-Q. Yin, and B. Zeng, "16-qubit IBM universal quantum computer can be fully entangled," *NPJ Quantum Inf.*, vol. 4, no. 1, p. 46, Dec. 2018.

[5] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information*, 10th ed. Cambridge, U.K.: Cambridge Univ. Press, 2010.

[6] P. Krantz, M. Kjaergaard, F. Yan, T. P. Orlando, S. Gustavsson, and W. D. Oliver, "A quantum engineer's guide to superconducting qubits," *Appl. Phys. Rev.*, vol. 6, no. 2, Jun. 2019, Art. no. 021318.

[7] H. J. Briegel, "Versatile cluster entangled light," *Science*, vol. 354, no. 6311, pp. 416–417, Oct. 2016.

[8] H. J. Briegel, D. E. Browne, W. Dür, R. Raussendorf, and M. V. D. Nest, "Measurement-based quantum computation," *Nature Phys.*, vol. 5, no. 1, p. 19, 2009.

[9] C. Nayak, S. H. Simon, A. Stern, M. Freedman, and S. D. Sarma, "Non-abelian anyons and topological quantum computation," *Rev. Mod. Phys.*, vol. 80, no. 3, p. 1083, 2008.

[10] A. Asfaw, L. Bello, Y. Ben-Haim, S. Bravyi, L. Capelluto, A. C. Vazquez, J. Gambetta, S. Garion, L. Gil, S. D. L. P. Gonzalez, D. McKay, Z. Minev, P. Nation, A. Phan, A. Rattew, J. Shabani, J. Smolin, K. Temme, M. Tod, and J. Wootton, *Learn Quantum Computation Using Qiskit*. 2019. [Online]. Available: https://qiskit.org/textbook/

[11] S. Barz, J. F. Fitzsimons, E. Kashefi, and P. Walther, "Experimental verification of quantum computation," *Nature Phys.*, vol. 9, no. 11, pp. 727–731, 2013.

[12] U. Mahadev, "Classical verification of quantum computations," in *Proc. IEEE 59th Annu. Symp. Found. Comput. Sci. (FOCS)*, Oct. 2018, pp. 259–267.

[13] F. Arute *et al.*, "Quantum supremacy using a programmable superconducting processor," *Nature*, vol. 574, no. 7779, pp. 505–510, 2019.

[14] J. Chiaverini, D. Leibfried, T. Schaetz, M. D. Barrett, R. B. Blakestad, J. Britton, W. M. Itano, J. D. Jost, E. Knill, C. Langer, R. Ozeri, and D. J. Wineland, "Realization of quantum error correction," *Nature*, vol. 432, no. 7017, pp. 602–605, 2004.

[15] P. W. Shor, "Fault-tolerant quantum computation," in *Proc. 37th Conf. Found. Comput. Sci.*, Oct. 1996, pp. 56–65.

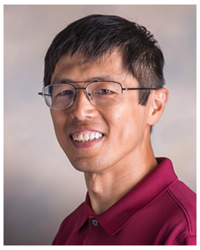
[16] R. Harper and S. T. Flammia, "Fault-tolerant logical gates in the IBM quantum experience," *Phys. Rev. Lett.*, vol. 122, no. 8, Feb. 2019, Art. no. 080504.

[17] A. Y. Kitaev, "Fault-tolerant quantum computation by anyons," *Ann. Phys.*, vol. 303, no. 1, pp. 2–30, Jan. 2003.

[18] B. Criger and B. Terhal, "Noise thresholds for the [[4, 2, 2]]-concatenated toric code," *Quantum Inf. Comput.*, vol. 16, nos. 15–16, pp. 1261–1281, 2016.

[19] S. B. Bravyi and A. Y. Kitaev, "Quantum codes on a lattice with boundary," 1998, *arXiv:quant-ph/9811052*. [Online]. Available: https://arxiv.org/abs/quant-ph/9811052

[20] M. Takita, A. D. Córcoles, E. Magesan, B. Abdo, M. Brink, A. Cross, J. M. Chow, and J. M. Gambetta, "Demonstration of weight-four parity measurements in the surface code architecture," *Phys. Rev. Lett.*, vol. 117, no. 21, Nov. 2016, Art. no. 210505.



CHANG LIU (Senior Member, IEEE) received the Ph.D. degree in information and computer science from the University of California at Irvine, USA, in 2002.

He is currently a Professor of Computer Science with Ohio University. His research interest includes software engineering and its applications in various domains, including learning and medical applications.