Fault-tolerant quantum error correction on near-term quantum processors using flag and bridge qubits

Lao, Lingling; Almudever, Carmen G.

DOI
10.1103/PhysRevA.101.032333

Publication date
2020

Document Version
Final published version

Published in
Physical Review A

Fault-tolerant quantum error correction on near-term quantum processors using flag and bridge qubits

Lingling Lao* and Carmen G. Almudever
QuTech, Delft University of Technology, Delft, The Netherlands

(Received 9 January 2020; accepted 3 March 2020; published 20 March 2020)

Fault-tolerant (FT) computation by using quantum error correction (QEC) is essential for realizing large-scale quantum algorithms. Devices are expected to have enough qubits to demonstrate aspects of fault tolerance in the near future. However, these near-term quantum processors will only contain a small amount of noisy qubits and allow limited qubit connectivity. Fault-tolerant schemes that not only have low qubit overhead but also comply with geometrical interaction constraints are therefore necessary. In this work, we combine flag fault tolerance with quantum circuit mapping, to enable an efficient *flag-bridge* approach to implement FT QEC on near-term devices. We further show an example of performing the Steane code error correction on two current superconducting processors and numerically analyze their performance with circuit level noise. The simulation results show that the QEC circuits that measure more stabilizers in parallel have lower logical error rates. We also observe that the Steane code can outperform the distance-3 surface code using flag-bridge error correction. In addition, we foresee potential applications of the flag-bridge approach such as FT computation using lattice surgery and code deformation techniques.

DOI: 10.1103/PhysRevA.101.032333

I. INTRODUCTION

Near-term quantum processors will consist of fifty to a few hundred noisy qubits and allow a limited number of faulty gates. They are also known as noisy intermediate-scale quantum (NISQ) [1] processors. For instance, Google, IBM, and Intel have respectively announced 72-qubit [2], 50-qubit [3], and 49-qubit [4] superconducting processors which have coherence times of ∼100 microseconds and two-qubit gate error rates near 0.1% [5]. Many efforts have been focusing on designing special quantum applications [6,7] and developing compilation techniques [8,9] such that one can solve practical problems and even demonstrate quantum supremacy on NISQ processors only using noisy bare qubits.

However, fault tolerance will be necessary to reliably implement large-scale quantum algorithms. This can be achieved through the use of active quantum error correction (QEC). The idea of QEC is to encode one logical qubit into many physical qubits and repeatedly perform syndrome extraction to detect and correct errors. Both the encoding and error detection procedure should be fault tolerant (FT). Furthermore, operations on these logical qubits need to be performed fault-tolerantly. Although the high qubit overhead of QEC makes it difficult to realize scalable FT computation in the near future, we can begin to learn how fault tolerance works in practice. The first step is to demonstrate fault-tolerant quantum error correction, that is, FT quantum memory.

General fault-tolerant quantum error correction protocols such as those from Shor [10], Steane [11], and Knill [12] can be applied to various stabilizer codes. However, these error correction schemes all require many ancilla qubits, which are scarce resources in near-term quantum processors. In order to perform FT QEC with low qubit overhead, a new error correction protocol has been proposed [13–16]. It replaces a non-FT syndrome extraction circuit by a circuit which can detect correlated (or hook) errors by adding only one or a few extra ancilla qubits, called *flag* qubits.

This flag QEC scheme provides an efficient way to demonstrate fault tolerance in small experiments. However, many orthodox flag circuits couple one qubit to many others, requiring high-degree qubit connectivity. It is difficult or even impossible to directly map available flag circuits onto near-term quantum processors which have geometrical interaction constraints such as the nearest-neighbor connectivity in superconducting processors [17–19]. One may need to apply extra operations such as SWAP gates to move qubits to be adjacent, increasing the circuit size in terms of depth and total gate number, or even circuit width. More importantly, the resulting circuit may not be fault tolerant, or may produce higher error rates when used.

In this work, we extend the set of available flag circuits to a variety of equivalent circuits that can perform the same stabilizer measurement fault-tolerantly. In these circuits, the flag qubits are also used as *bridges* to cope with the connectivity constraints, called *flag-bridge* qubits. Using these circuits, one can fault-tolerantly map a QEC code to a given processor with low overhead by choosing appropriate flag-bridge circuits. We also develop a simulation framework to automate the procedure of fault tolerance checking, decoder design (including a look-up-table decoder and a neural-network decoder) for given flag-bridge circuits of some low-distance QEC codes. This automation is desirable for demonstrating fault-tolerant quantum error correction in small experiments. Moreover, we present mapping examples of the Steane code on two different qubit processor topologies and analyze their fault tolerance

*Corresponding author: laolinglingrolls@gmail.com

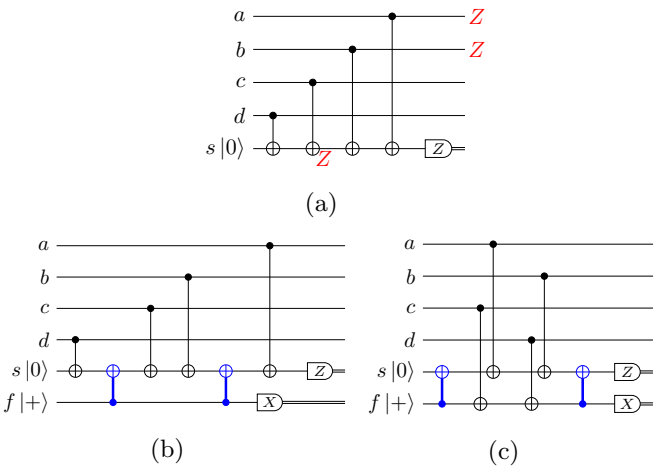


FIG. 1. The syndrome extraction circuits for the $Z_{a,b,c,d}$ operator, where s is the syndrome qubit and f is the flag qubit. (a) The circuit only using one syndrome ancilla may not be fault tolerant. For example, one fault (Z_s) on the second CNOT gate could lead to correlated weight-2 errors on data qubits (Z_a, Z_b), which may not be correctable. (b) and (c) The flag-based circuits can detect these hook errors [13–15].

numerically. In addition, we show the proposed flag-bridge approach can be applied to FT computation implemented by lattice surgery and code deformation techniques.

The rest of this paper is organized as follows. We first review the basics of flag-based quantum error correction in Sec. II. Then we introduce the proposed flag-bridge approach in Sec. III. Afterwards, the mapping of the Steane code onto two qubit processor topologies and corresponding numerical results are shown in Sec. IV. Moreover, we provide the potential applications of flag-bridge circuits in Sec. V. Finally, Sec. VI concludes the paper.

II. FLAG-BASED QUANTUM ERROR CORRECTION

In this section, we briefly introduce the flag-based error syndrome extraction for stabilizer codes. For more details, we refer the readers to [13–16,20].

Figure 1 shows the circuits for measuring a weight-4 Z-stabilizer (or check), similar circuits can be derived for measuring other Pauli operators. In all the circuits presented in this paper, a CNOT gate between a data qubit and an ancilla qubit is called an s-CNOT and a CNOT gate between two ancilla qubits is called an f-CNOT (thick blue lines). Generally, the syndrome for this Z-check can be extracted using the circuit with only one ancilla qubit [Fig. 1(a)]. However, this circuit is not fault tolerant because one single fault could cause two or more data errors. These correlated errors may lead to failures of some QEC codes. The surface code is an exception which can correct these hook errors if the two-qubit gates are performed in a specific order [21]. In order to perform fault-tolerant quantum error correction, one can use the flag circuits in Figs. 1(b) and 1(c) that only add one extra ancilla qubit. When there is no fault, each of these flag circuits behaves the same as the non-FT one. When there is a fault that can lead to hook errors, it will yield a nontrivial measurement outcome

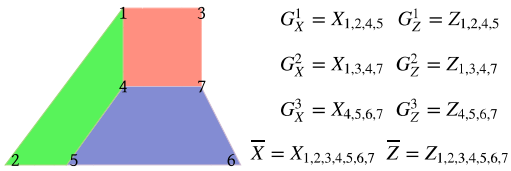


FIG. 2. (Left) The qubit layout of the $[\![7, 1, 3]\!]$ Steane code. Data qubits are on the vertices and each plaquette represents two stabilizers: one weight-4 X-stabilizer and one weight-4 Z-stabilizer. (Right) all the six stabilizer generators and logical X and Z operators.

of the flag qubit such that the hook errors are detected. For instance, if the same fault in Fig. 1(a) happens in the circuit of Fig. 1(b), then the measurement of qubit f will be 1 (raising a flag).

Flag-based quantum error correction can be applied to many codes such as the $[\![5, 1, 3]\!]$ code, Hamming codes, surface codes, color codes, etc. For example, fault-tolerant QEC for the smallest color code, the Steane code in Fig. 2, can be realized as follows: first measure each stabilizer generator one by one using flag circuits similar to those in Fig. 1; if a flag raises or a syndrome appears, then stop this round[1] and sequentially measure all the stabilizers using the non-FT syndrome extraction circuit. Note that if connectivity is fixed, we cannot necessarily change the syndrome measurement circuit all of a sudden. One can use only two ancilla qubits to perform FT QEC for the Steane code at the cost of using more timesteps. However, many quantum systems have very short coherence times [17,22,23]. Parallelizing stabilizer measurement will be beneficial to achieve lower logical error rates. Chao and Reichardt [14,16] have proposed several circuits to perform two or three parity checks in parallel for the $[\![7, 1, 3]\!]$ Steane code. The circuits they propose for measuring two and three Z-checks at the same time using only one flag qubit are shown in Fig. 3. If the measurement outcome of the shared flag qubit f is trivial, one single fault can only lead to at most one weight-1 Z error. If a flag raises, different Z errors (each of which is caused by a single fault) can be distinguished based on the observed syndromes. As shown in Fig. 3, more ancilla qubits are required to achieve this parallelism compared to the sequential stabilizer measurement circuits (Fig. 1). This implies there is a trade-off between the number of qubits required and the number of stabilizers that can be measured simultaneously.

Flag-based syndrome extraction is promising for demonstrating quantum error correction and fault tolerance in small quantum experiments because of its low qubit overhead. However, current or near-term quantum processors have many hardware limitations. One of the main constraints is the degree of qubit connectivity; that is, one qubit can only interact with a limited number of other qubits. It is challenging to map existing flag circuits onto connectivity-constrained quantum processors meanwhile maintaining the fault tolerance with low costs. For instance, the ancilla qubit s of the flag circuit

[1]A full round of error syndrome extraction is defined as measuring all the stabilizer generators of the code for one time.

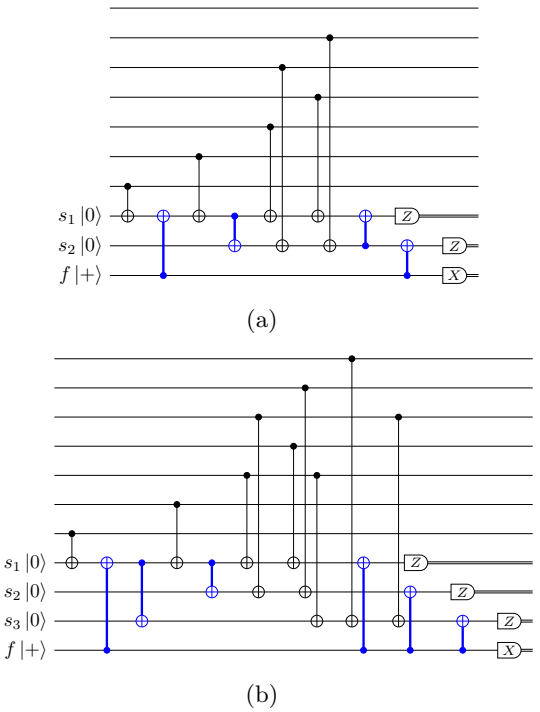


FIG. 3. Flag circuits [14] of the $[\![7,\,1,\,3]\!]$ Steane code for (a) measuring two weight-4 Z-checks in parallel using three ancillas and (b) measuring three weight-4 Z-checks in parallel using four ancillas. s_i is a syndrome qubit and f is the shared flag qubit.

in Fig. 1(b) needs to interact with five qubits, which cannot be supported in a grid topology where each qubit only has at most four neighbors such as the one in [18]. Besides, general circuit mapping techniques [9,24–29] that move qubits to be adjacent by applying SWAP gates will lead to high overhead in the circuit size. More importantly, it may result in higher logical error rates or even destroy the fault tolerance of the QEC circuits because of the error propagation through two-qubit gates. In this work, we propose a flag-bridge approach to solve this mapping problem, which will be explained in the next section.

III. FLAG-BRIDGE QUANTUM ERROR CORRECTION

In this section, we illustrate the proposed flag-bridge approach which allows fault-tolerant quantum error correction with low qubit overhead on connectivity-limited quantum processors.

A. Flag-bridge syndrome extraction circuits

We first provide a microscopic explanation of how a flag-based circuit can perform a specific stabilizer measurement using the stabilizer formalism [30]. Then we generalize this flag scheme such that one can extend available flag circuits to more equivalent ones that are different in terms of the total number of gates, circuit depth, and connectivity requirement. We will use the circuit in Fig. 1(c) as an example. A flag syndrome extraction circuit can be understood as a circuit that replaces the bare ancilla qubit by an "encoded" ancilla up to

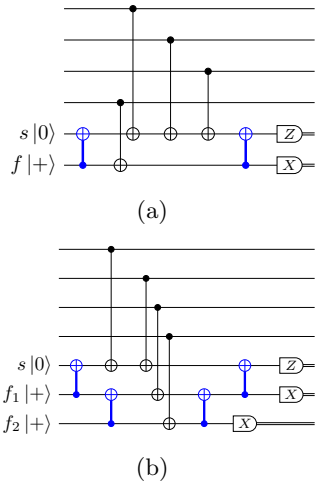


FIG. 4. Flag-bridge circuits for measuring one weight-4 Z-check using (a) two ancillas and (b) three ancillas.

gate commutation. As shown in Fig. 1(c), the first f-CNOT gate entangles ancilla qubit s and qubit f (the encoding circuit), encoding a logical ancilla in a $[\![2,\,1,\,1]\!]$ error detection code of which the stabilizer is

$$\langle X_s \otimes X_f \rangle$$

and logical operators are

$$\langle \overline{X} = X_s, \overline{Z} = Z_s \otimes Z_f \rangle.$$

This logical qubit is fixed in the $\overline{Z}$ basis. Then one can perform stabilizer measurement using this logical ancilla. Assume the four data qubits (a, b, c, d) are initially stabilized by $(-1)^y Z_{a,b,c,d}$; the four subsequent s-CNOT gates between data qubits and ancilla qubits will keep the stabilizers

$$\langle X_s \otimes X_f, (-1)^y Z_{4,5,6,7} \rangle$$

of all the qubits invariant, but they will gradually transform the logical operators into

$$\langle \overline{X} = X_s, \overline{Z} = Z_s \otimes Z_f \otimes (-1)^y Z_{4,5,6,7} \rangle.$$

More generally, since X_f and X_s have the same effect on the encoded ancilla state, one can perform each s-CNOT gate between the particular data qubit with any ancilla qubit. Specifically, in the encoded ancilla area, k_s and k_f s-CNOT gates can be applied on ancillas s and f respectively, where k_s and k_f are integers and $k_s + k_f = 4$. For example, the circuit shown in Fig. 4(a) also performs a weight-4 Z-stabilizer measurement equivalent to this circuit [Fig. 1(c)], where $k_s = 3$ and $k_f = 1$.

Afterwards, the last f-CNOT (the decoding circuit) disentangles these two ancillas, leading to the final stabilizer

$$\langle (-1)^y Z_{4,5,6,7} \rangle$$

and the logical operators of these ancillas,

$$\langle X_f, (-1)^y Z_s \rangle.$$

This means the readout y of measurement M_z on ancilla s indicates the measurement result of the stabilizer $Z_{a,b,c,d}$. Therefore, this circuit indeed measures a weight-4 Z-check.

Besides, the measurement result of ancilla f implies the syndrome of the $[\![2,\ 1,\ 1]\!]$ code; that is, it can detect one single Z error that occurs on any ancilla and then raises a flag.

Once a flag circuit based on the above approach is generated, one can transform it into other equivalent ones that can perform the same stabilizer measurement by applying gate commutation, e.g., the circuit in Fig. 1(b). Note that the circuits generated by commuting gates may not be fault tolerant.

Moreover, one can use a larger "encoded" ancilla to measure a weight-n Z-check (similar circuits can be applied to other Pauli operators). This logical ancilla is encoded by m physical qubits denoted by a set $Q = \{1, 2, \ldots, m\}$, where one is syndrome qubit ($Q_s = \{1\}$) and the other $m-1$ are flag qubits ($Q_f = \{2, \ldots, m\}$). The underlying error detection code $[\![m,\ 1,\ 1]\!]$ of this logical ancilla has stabilizers

$$\langle X_j \otimes X_k \rangle, \quad j \in Q_s,\ k \in Q_f$$

and logical operators

$$\left\langle \overline{X} = X_j, \overline{Z} = \bigotimes_i Z_i \right\rangle, \quad i, j \in Q.$$

Similar to the two-ancilla flag circuits, this weight-n check can be distributed to all m ancillas; k_i s-CNOT gates will be applied on ancilla i, where $\sum_{i=1}^{m} k_i = n$. For example, the circuit in Fig. 4(b) measures one weight-4 Z-stabilizer using one syndrome qubit (s) and two flag qubits (f_1, f_2); each qubit only needs to interact with at most three others.

In addition, one can also measure p Z-checks in parallel by encoding p logical ancillas into m physical ancillas. The underlying $[\![m,\ p,\ 1]\!]$ code is stabilized by

$$\left\langle X_i \otimes \bigotimes_j X_j \right\rangle, \quad i \in Q_f,\ j \in Q_s.$$

Its p logical operators are

$$\langle \overline{X_k} = X_i, \overline{Z_k} = Z_i \otimes Z_j \rangle, \quad i, j \in Q,\ i < j,$$

where Q_s is the set of p syndrome qubits and Q_f is the set of $m-p$ flag qubits. After the encoding of ancilla qubits, one can simply assign all the s-CNOT gates for performing one check to a particular syndrome qubit. In this parallel syndrome extraction case, one can reduce the total number of s-CNOT gates by applying gate commutation when two or more checks are performed on the same data qubit(s). Figures 3(a) and 3(b) show the flag circuits to measure two and three Z checks of the Steane code in parallel by using ancillas encoded in a $[\![3,\ 2,\ 1]\!]$ code and in a $[\![4,\ 3,\ 1]\!]$ code, respectively, These circuits use fewer s-CNOT gates than required by commuting some CNOT gates out of the encoded area (generally 4 s-CNOT gates are needed for each weight-4 check).

Moreover, one can achieve this gate reduction by distributing some s-CNOT gates to flag qubits, which can even help to reduce the circuit depth. Figure 5(a) shows the example circuit that measures two checks of the Steane code in parallel but uses fewer timesteps than Fig. 3(a). Note that the s-CNOT distribution for parallel syndrome measurement needs to be designed carefully since one flag qubit is used for flagging multiple checks. This distribution also depends on the decoding procedure of the $[\![m,\ p,\ 1]\!]$ code. Figure 5(b) shows the example circuit that measures three checks using fewer timesteps than Fig. 3(b). Besides, the circuits in Fig. 5 require fewer degrees of qubit connectivity than the ones in Fig. 3.

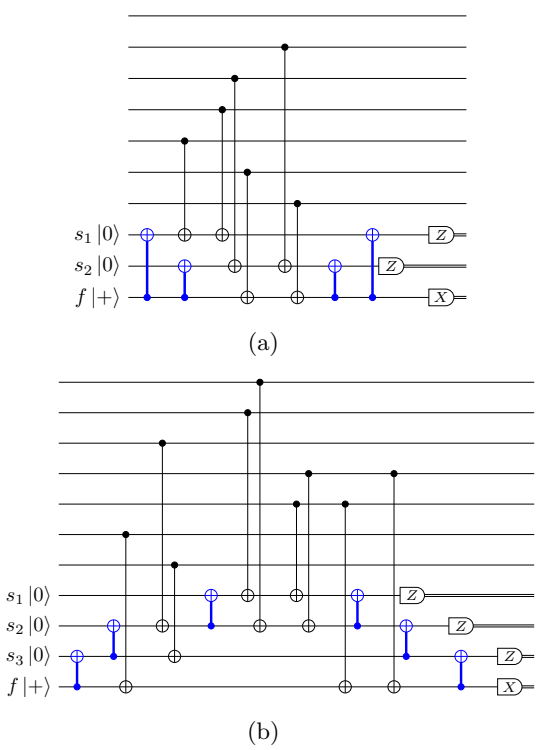


FIG. 5. Flag-bridge circuits of the $[\![7,\ 1,\ 3]\!]$ Steane code which measure (a) two and (b) three weight-4 Z-checks in parallel.

By employing the ideas of encoding ancillas, distributing s-CNOT, and commuting gates, we can generate more equivalent syndrome extraction circuits that have different connectivity requirements. Note that not all the equivalent circuits generated using this approach are fault tolerant. The fault tolerance can be checked based on the error correction protocol, which will be explained in the next section. For these FT circuits, ancillas are not only used as syndrome and flag qubits to detect errors, but also as bridges to allow the interaction between data qubits and the encoded ancilla block. Such a syndrome extraction circuit is called a flag-bridge circuit.

B. Fault-tolerant protocol for flag-bridge error correction

1. FT QEC condition

For distance-3 codes, a QEC circuit is fault-tolerant if it can either immediately correct all errors from a single fault or only leave a weight-1 error to the next cycle. A formal condition of FT flag-bridge quantum error correction for distance-3 codes, similar to the flag error correction in [15], can be defined as follows:

Consider a stabilizer code $\mathcal{S} = \langle g_1, g_2, \ldots, g_r \rangle$ and its QEC circuit $\mathcal{C}$ which is composed of the flag-bridge circuits for measuring the stabilizer generators, that is, $\mathcal{C} = \{c(g_1), c(g_2), \ldots, c(g_r)\}$, where $c(g_i)$ is the flag circuit of measuring stabilizer g_i. Note that the total number of flag-bridge circuits is smaller than r if several stabilizers are

measured simultaneously in one flag-bridge circuit. For all generators g, all pairs of elements $E, E' \in \mathcal{E}(g)$ satisfy $sf(E) \neq sf(E')$ or $E \sim E'$, where $\mathcal{E}(g)$ is the set of all errors caused by one fault and $sf(E)$ is the syndrome and flag string caused by E. We define $E \sim E'$ to mean that there is an element g in $\mathcal{S}$ such that $E' \propto gE$; that is, these errors are stabilizer equivalent.

Based on this criterion, we check the fault tolerance of each generated QEC circuit $\mathcal{C}$ through a brute-force simulation under circuit level noise, analogous to [13]. It is implemented by injecting each individual fault from a circuit-based error model on every single-qubit or two-qubit gate in a given QEC circuit and then collecting the final syndromes and flags. If there are two or more sets of errors which lead to the same syndrome-flag string but do not yield a stabilizer when multiplied, then this QEC circuit is not fault tolerant.

2. FT QEC procedure

A full cycle of fault-tolerant error correction for distance-3 codes using flag-bridge circuits can be performed as follows:

(1) For the first round of syndrome extraction, each circuit $c(g_i) \in \mathcal{C}$ is sequentially performed. If there are nontrivial flags (form a set $\mathcal{F}_i^1$) or nontrivial syndromes (form a set $\mathcal{S}_i^1$) of $c(g_i)$, then this round will be terminated and another full round for all circuits in $\mathcal{C}$ will be performed. All the syndromes $\in \mathcal{S}^2 = \bigcup_i \mathcal{S}_i^2$ and flags $\in \mathcal{F}^2 = \bigcup_i \mathcal{F}_i^2$ of the second round will be collected.

(2) If $\mathcal{F}_i^1$ is not empty, one can decode using $\mathcal{F}_i^1$ and $\mathcal{S}^2$ (and $\mathcal{F}^2$). If $\mathcal{F}_i^1$ is empty, but $\mathcal{S}_i^1$ is not empty, one can decode using $\mathcal{S}^2$ (and $\mathcal{F}^2$). Otherwise, no corrections are needed.

In this FT QEC procedure, we use flag-bridge circuits for both rounds of syndrome extraction because of the connectivity constraint, which is different from the ones proposed in [14–16], where non-FT syndrome extraction circuits that use only one ancilla are executed for the second round.

For example, each stabilizer generator g_i of the $[\![7, 1, 3]\!]$ Steane code (Fig. 2) can be measured fault tolerantly by using a flag-bridge circuit $c(g_i)$ that is the same as (for Z-checks) or similar to (for X-checks) the one in Fig. 1(c), where $g_i \in \{G_{X(Z)}^j\}$, $i \in \{1, 2, \ldots, 6\}$, $j \in \{1, 2, 3\}$. Then a full FT QEC cycle of the Steane code can be performed by first sequentially executing $c(g_i)$. If the measurement outcome of either the flag qubit f_t or the syndrome qubit s_t in $c(g_t)$ is nontrivial, then terminate the first round [that is, $c(g_{t+1}), \ldots, c(g_6)$ will not be executed] and perform the second round of syndrome measurement by sequentially running each flag-bridge circuit $c(g_i)$ regardless of any flags or syndromes. Only the flag (if f_t raises a flag) in the first round and all the flags and syndromes in the second round are collected and will be used for decoding. If neither a flag nor a syndrome is detected in the first round, then one will not perform the second round and no corrections will be applied.

3. Error decoders

Normally, error corrections of topological codes like surface codes have special structures for the measured syndromes so that one can use heuristic algorithms to find high-probability errors. These types of decoders such as the minimum weight perfect matching decoder [31] and the belief propagation decoder [32] can be applied to the same QEC code with different distances. However, the flag-bridge error correction circuits of a QEC code for a specific quantum platform are *ad hoc*. Different circuits may be chosen based on the qubit topology, leading to different error-syndrome patterns and in turn requiring different decoding strategies. It is difficult to design heuristic decoding algorithms that can be applied to various syndrome extraction circuits. Since flag-bridge circuits are likely to be used for low-distance codes in small experiments, a simple decoding solution is to create a look-up table (LUT) for each QEC circuit. A LUT decoder can find the most likely Pauli errors from a single fault that leads to the observed syndromes and flags. LUT decoders can be easily derived from the brute-force checking procedure [13].

Another type of decoders are the neural-network (NN) decoders [33–36]. They can provide high-speed decoding, be adaptable to different error models, and be more easily implemented on hardware. Moreover, a NN decoder can be developed by training the network using only input-output pairs without any knowledge of the QEC code, making it favorable for flag-bridge circuits. For example, the inputs of a NN decoder are the observed syndromes and its outputs can be the actual physical errors that have occurred. The implementation details of the LUT decoder and the NN decoder can be found in the Appendix.

In this work, we design a simulation framework to automate the procedure of fault tolerance checking, LUT generation, and NN decoder training for given flag-bridge syndrome extraction circuits of the Steane code. This automation is desirable for demonstrating fault-tolerant quantum error correction in near-term processors which may have different geometrical interaction constraints.

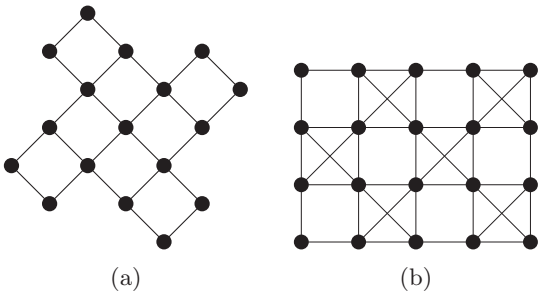
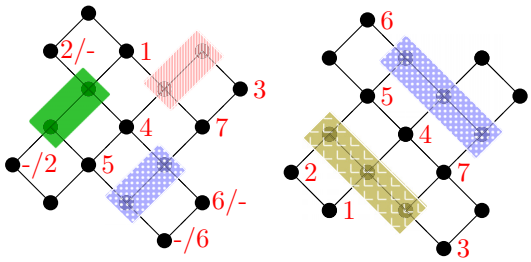


FIG. 6. (a) The Surface-17 topology and (b) the IBM-20 topology, where each node represents a qubit and each edge indicates the connectivity between two qubits.

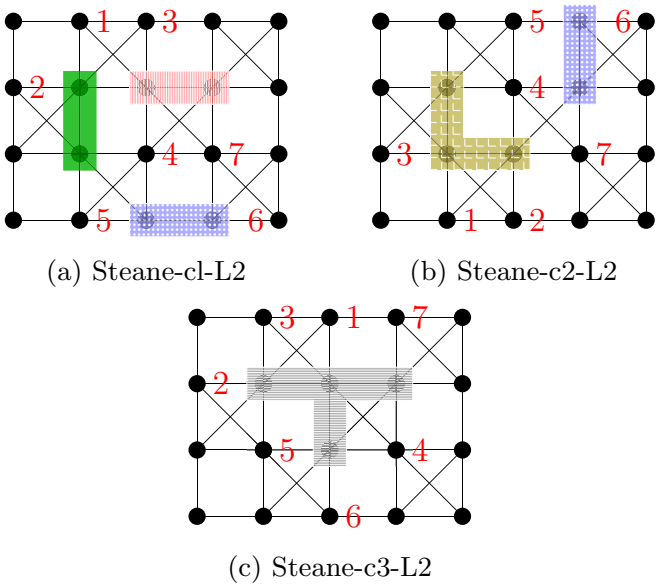
IV. STEANE CODE ERROR CORRECTION ON TWO PROCESSOR TOPOLOGIES

In this section, we show how to map the Steane code error correction onto two different processors with limited connectivity using the proposed flag-bridge circuits, namely, the Surface-17 transmon processor (Surface-17) [18] and the IBM Q Tokyo processor (IBM-20) [17] (Fig. 6). Furthermore, we numerically analyze each flag-bridge quantum error correction procedure under circuit level noise. This error model inserts depolarizing errors after each operation in a flag-bridge circuit as follows: (1) each single-qubit gate is followed by



(a) Steane-cl-L1 (b) Steane-c2-L1

FIG. 7. Mapping of the Steane code onto the Surface-17 topology, where the qubits labeled with numbers are data qubits and the qubits in the colored blocks are ancillas. (a) The mapping using the two-ancilla flag-bridge circuits in Figs. 1(c) ($G^1_{X(Z)}$ and $G^3_{X(Z)}$) and 4(a) ($G^2_{X(Z)}$) in which only one stabilizer is measured at a time. (b) The mapping using the three-ancilla flag-bridge circuits in Figs. 4(b) ($G^3_{X(Z)}$) and 5(a) ($G^1_{X(Z)}$ and $G^2_{X(Z)}$) that measure one and two stabilizers, respectively.



(a) Steane-cl-L2 (b) Steane-c2-L2

(c) Steane-c3-L2

FIG. 8. Mapping of the Steane code onto the IBM-20 topology (a) using the flag-bridge circuit in Fig. 1(c) to sequentially measure each stabilizer; (b) using the flag-bridge circuits in Figs. 1(c) ($G^3_{X(Z)}$) and 5(a) ($G^1_{X(Z)}$ and $G^2_{X(Z)}$) to measure one and two stabilizers, respectively; (c) using the four-ancilla flag-bridge circuit in Fig. 5(b) to measure three stabilizers simultaneously.

an X, Y, or Z with probability $p/3$; (2) each two-qubit gate is followed by an element of $\{I, X, Y, Z\}^{\otimes 2}\backslash\{II\}$ with probability $p/15$; (3) the preparation or measurement in the Z basis is flipped with probability p. The elementary Clifford operations used in this simulation are preparation and measurement in the Z basis, H, and CNOT gates. Other operations need to be further decomposed into these elementary operations. For example, each control-phase gate is replaced by two H gates and one CNOT gate.

A. Mapping

Many current and NISQ processors have geometrical connectivity constraints; that is, each qubit can only interact with a few neighbors. It is challenging or even impossible to directly perform existing flag-based quantum error correction without adding more operations and/or without losing fault tolerance. For example, the flag circuit which measures one weight-4 Z-stabilizer of the Steane code in Fig. 1(b) cannot be directly executed on the Surface-17 topology [Fig. 6(a)] but can be supported by the IBM-20 topology [Fig. 6(b)]. This is because qubit s needs to interact with five qubits but in Surface-17 each qubit has at most four neighbors. The flag circuit in Fig. 1(c) can be performed on both processor topologies. However, a full round of error syndrome extraction requires all the stabilizer generators of the Steane code to be measured. The full syndrome extraction using only these two flag circuits [Figs. 1(b) and 1(c)] can be directly performed on the IBM-20 topology [e.g., a mapping in Fig. 8(a)] but not on the Surface-17 topology.

As mentioned above, all the flag-bridge circuits shown in this paper are used to measure Z-stabilizers; similar circuits with the same ancillas can be derived for measuring X-stabilizers. Figures 7 and 8 show examples of mapping the Steane code error correction using the flag-bridge circuits onto the Surface-17 topology and the IBM-20 topology, respectively.

In these mapping figures, the qubits in each red (medium gray with vertical lines), blue (dark gray with dots), or green (light gray) block are the ancillas in each flag-bridge circuit and they are used to measure the corresponding X(or Z)-stabilizer in the same color plaquette in Fig. 2. The flag-bridge qubits in the yellow (gray with grids) block are used to measure the X(or Z)-stabilizers in both red and green plaquettes. The flag-bridge qubits in the gray block with horizontal lines measure the X(or Z)-stabilizers in all three plaquettes. The X- and Z-stabilizers are measured separately, more specifically, one first measures all the stabilizers in one type and then measures the other type. Furthermore, each of the flag-bridge circuits for the Steane code error correction needs to be executed sequentially. On the Surface-17 topology, one can measure all the stabilizers of the Steane code one by one when using the mapping in Fig. 7(a). Maximally two stabilizers can be measured in parallel in this topology, as shown in Fig. 7(b). In contrast, three $X(Z)$-stabilizers can be measured at the same time on the IBM-20 topology [Fig. 8(c)].

The circuit characterization of one full round of syndrome extraction for the Steane code when using different mappings is shown in Table I. This characterization includes the total number of ancilla qubits, the total number of operations and timesteps, and the number of f-CNOT and s-CNOT gates. To show the mapping overhead, the characterization of the QEC circuits (Steane-c1-nl, Steane-c2-nl, and Steane-c3-nl) of the Steane code without considering the geometrical interaction constraint is also described in Table I. In this case, if multiple flag-bridge circuits in previous sections can measure the same number of stabilizers, then the one that use fewest ancillas and fewest timesteps is chosen. Specifically, in Steane-c1-nl each stabilizer is measured sequentially by using the circuit in Fig. 1(c). In Steane-c2-nl, two same-type (X or Z) stabilizers are measured in parallel by using the circuit Fig. 5(a) and then the other same-type stabilizer is measured separately using the circuit in Fig. 1(c). In Steane-c3-nl all the three

TABLE I. Comparison of the quantum error correction circuits of the Steane code when different parallelism can be achieved and when different mappings are applied.

	Number of				
	Ancillas	Operations	f-CNOTs	s-CNOTs	Timesteps
Steane-c1-nl	2	72	12	24	48
Steane-c2-nl	3	62	12	20	36
Steane-c3-nl	4	54	12	18	26
Steane-c1-L1	6	72	12	24	50
Steane-c1-L2	6	72	12	24	48
Steane-c2-L1	6	72	16	20	40
Steane-c2-L2	5	62	12	20	36
Steane-c3-L2	4	54	12	18	26
SC d=3	8	48	0	24	8

same-type stabilizers are measured in parallel by using the circuit in Fig. 5(b). As shown in Table I, when there is no connectivity constraint, the FT QEC of the Steane code can be implemented by only using two ancillas; that is, these ancillas are reused by each stabilizer measurement circuit. However, when the qubit connectivity of a given quantum processor is limited, more ancilla qubits may be required to measure all the stabilizers. For instance, Steane-c1-L2 is implemented by using the same flag-bridge circuit as Steane-c1-nl but requires two times more ancillas. Furthermore, flag-bridge circuits that have more operations and/or more timesteps may be needed to comply with the connectivity constraint. Moreover, the circuits which can measure more stabilizers simultaneously require fewer operations and fewer timesteps. Besides, we also show these parameters of the rotated distance-3 surface code (SC d=3) for comparison. Though the distance-3 surface code uses more ancilla qubits, it always needs fewer operations and fewer timesteps than the Steane code.

B. Numerics

We further compare different mapping circuits in terms of their fault tolerance, which is analyzed by numerical simulation under circuit level noise. For each point in the numerics, 10^6 iterations of a full QEC cycle have been run and confidence intervals at 99.9% are plotted. Moreover, NN decoders are used for this comparison since it has better performance than LUT decoders [see Figs. 9(a) and 14]. As shown in Fig. 9, for the Steane code, the circuits that can measure more stabilizers in parallel have lower logical error rates, likely because they consist of fewer operations and require fewer timesteps. Moreover, when there are no idling errors [$p_I = 0$ in Fig. 9(a)] or a small probability of idling errors [$p_I = 0.01p$ in Fig. 9(b)], the Steane code can achieve similar performance to, or even outperform, the distance-3 surface code by parallelizing stabilizer measurements. This is because the circuit for the surface code error correction consists of

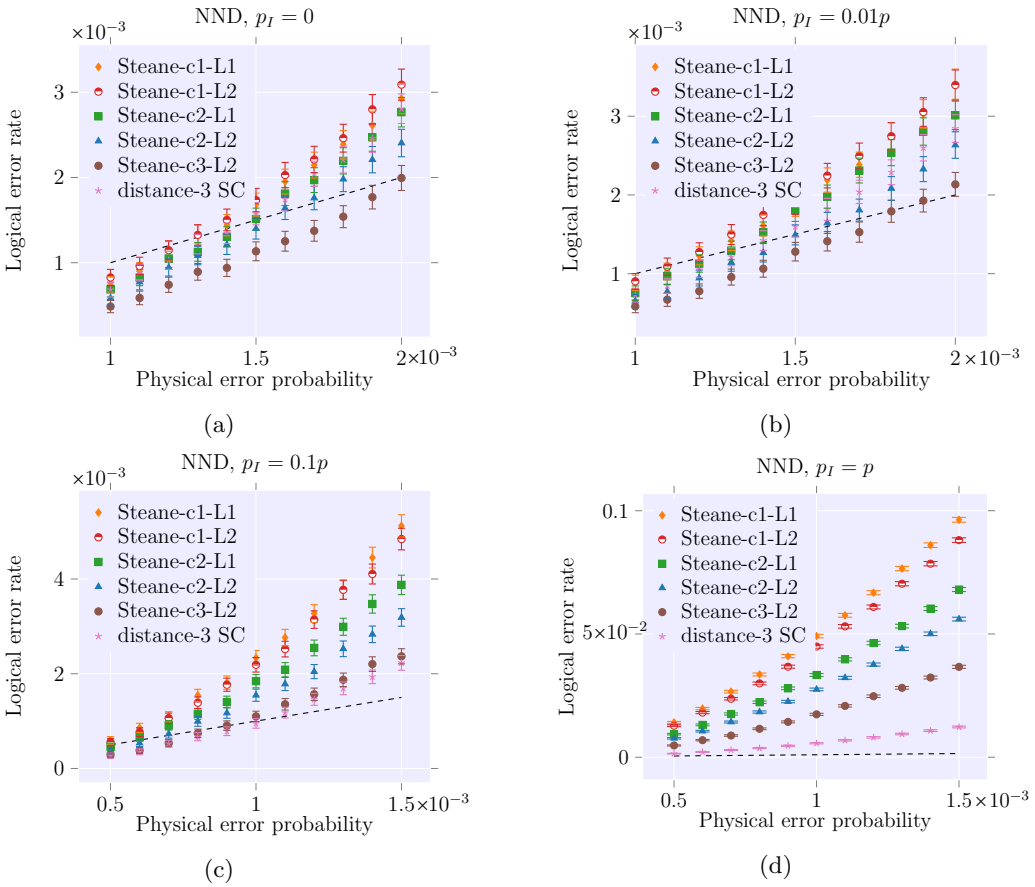


FIG. 9. Numerical simulation of the Steane code error correction based on different flag-bridge circuits using neural network decoders (NND). The circuit level noise ($p_1 = p_2 = p_M$) with ($p_I \neq 0$) or without idling errors ($p_I \neq 0$).

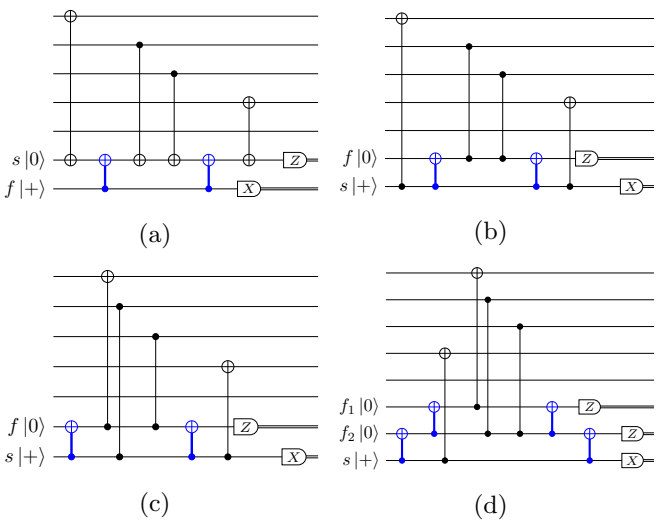


FIG. 10. Fault-tolerant circuits for performing an $XZZX$-check: (a), (b), (c) using two ancillas but requiring different connectivity; (d) using three ancillas, similar circuits can be generated by re-distributing the s-CNOT gates for each weight-4 check to different ancillas, as mentioned in Sec. III.

more s-CNOT gates than the QEC circuits that can measure several stabilizers in parallel for the Steane code. When idling errors are significant, we observe that the circuit with fewer timesteps results in lower logical error rates [as shown in Figs. 9(c) and 9(d) for $p_I = 0.1p$ and $p_I = p$ respectively].

V. OTHER APPLICATIONS OF THE FLAG-BRIDGE CIRCUITS

In this section, we foresee some possible applications of the flag-bridge circuits including both fault-tolerant quantum error correction and fault-tolerant quantum computation.

A. Flag-bridge QEC for the five-qubit code

Analogous to the flag circuits, the flag-bridge circuits can also be applied to other distance-3 error correction codes such as the $[\![8,\,3,\,3]\!]$, $[\![10,\,4,\,3]\!]$, $[\![11,\,5,\,3]\!]$, $[\![5,\,1,\,3]\!]$ codes, Hamming codes $[\![2^r-1,\,2^r-1-2r,\,3]\!]$, etc. In this section, we consider the $[\![5,\,1,\,3]\!]$ code as an example. This code has four stabilizers, which are cyclic permutations of $XZZXI$. Figure 10 shows the flag-bridge circuits that can measure an $XZZX$-stabilizer fault-tolerantly. Each stabilizer of the five-qubit code can be measured using these circuits up to data qubit permutation. Similar circuits using three ancillas to measure one stabilizer are also proposed in [13]. All these circuits have different connectivity requirements. By selecting and combining some of them, one can map the five-qubit code error correction onto different qubit topologies. Figure 11 shows the mapping of the five-qubit code to the Surface-17 processor topology using the two-ancilla flag-bridge circuits and the IBM Q Melbourne (IBM-16) processor topology using the three-ancilla flag-bridge circuits.

B. Flag-bridge circuits for FT computation

The geometrical interaction constraint in near-term quantum processors has also limited the fault-tolerant implementation of logical operations. For instance, a fault-tolerant CNOT gate in planar surface codes and color codes in principle can be implemented transversally in a three-dimensional (3D) structure, that is, performing pair-wise CNOT gates between data qubits of the two lattices. However, this transversal CNOT is not realizable in near-term quantum technologies because of the local qubit connectivity limitation in a 2D architecture. Measurement-based protocols such as lattice surgery [37,38] and code deformation [39,40] have been proposed to comply with the 2D local interaction constraint. Figures 12 and 13 show the qubit layouts for performing lattice-surgery-based operations on the distance-3 surface code and the distance-3 color code (the Steane code), respectively. The details of implementing logical operations by lattice surgery can be found in [37,38].

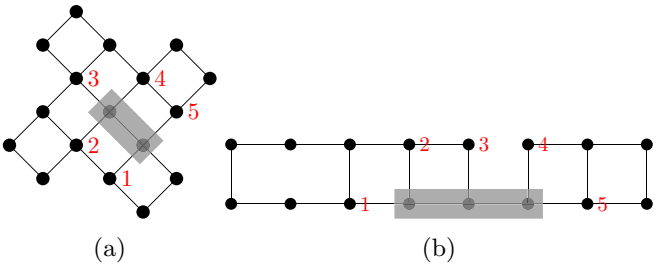


FIG. 11. Mapping of the five-qubit code onto (a) the Surface-17 topology by using the two-ancilla flag-bridge circuits in Figs. 10, and (b) the IBM-16 topology using the three-ancilla circuit in Fig. 10(d).

As shown in Figure 12, the merge operations can be directly performed on a 2D grid topology. As mentioned previously, the stabilizer measurement of surface codes can be realized by only using the one-ancilla circuit similar to Fig. 1(a). However, one ancilla qubit [the circled one in Fig. 12(b)] is used by two stabilizers from different lattices during the split operation. One may have to measure these two stabilizers sequentially, which leads to more timesteps and in turn may result in higher logical error rates. To preserve parallelism of the stabilizer measurement, we propose to use

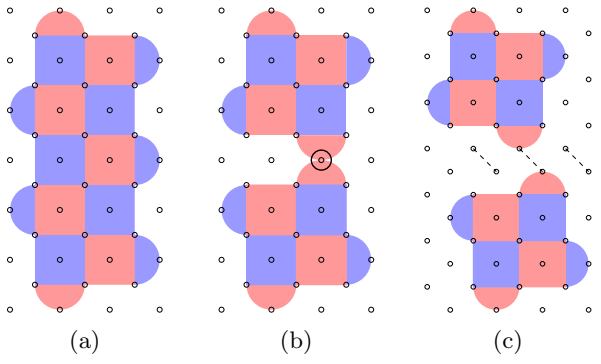


FIG. 12. Mapping lattice surgery-based operations for the distance-3 surface code using flag-bridge circuits. Each red (light gray) plaquette represents a weight-4 or weight-2 X-stabilizer. Each blue (dark gray) plaquette represents a weight-4 or weight-2 Z-stabilizer. Data qubits are on the vertices and ancilla qubits are on the plaquettes. (a) and (b) Initial layouts for performing a merge and a split operation using lattice surgery, respectively. (c) The layout after mapping using the two-ancilla flag-bridge circuits.

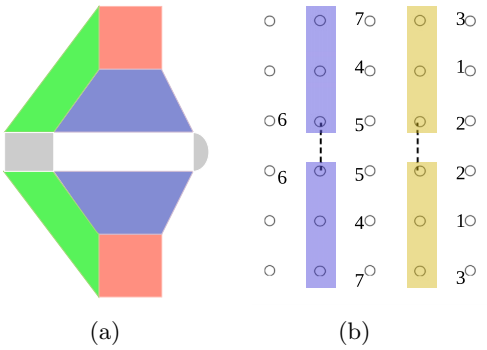


FIG. 13. Mapping lattice surgery-based operations for the Steane code using flag-bridge circuits. (a) Initial layout, where the gray plaquettes between two lattices only contain one type of stabilizers, depending on which joint measurement needs to be performed. Data qubits are on the vertices and ancilla qubits are on the plaquettes. (b) Mapping to a grid topology similar to Fig. 7(b) (Ancillas in each left block and right block are used to measure one and two stabilizers in parallel, respectively.).

the qubit layout in Fig. 12(c). By using this layout, one can measure all the stabilizers in parallel when splitting lattices since they no longer share ancillas. One can also perform the merge operation by replacing the original syndrome extraction circuit using one ancilla with the proposed flag-bridge circuits using two ancillas [Fig. 1(c)] where ancillas are connected by dash lines in Fig. 12(c). Similar mapping can be applied to other code-deformation-based operations on surface codes.

Furthermore, lattice-surgery-based operations for the Steane code in Fig. 13(a) cannot be directly realized in a 2D grid topology. Similar to the mapping in Fig. 7(b), one can map these operations fault-tolerantly using the three-ancilla flag-bridge circuits as shown in Fig. 13(b). Compared to the distance-3 surface code, the Steane code can achieve Clifford gates transversally. Moreover, it requires fewer qubits for both FT error correction and FT computation, which may be preferable for demonstrating fault tolerance in small experiments.

VI. DISCUSSION AND CONCLUSION

We have shown that flag circuits can be interpreted as replacing bare ancillas by encoded ancillas in an error detection code. Based on this formulation, we proposed a flag-bridge approach to perform fault-tolerant quantum error correction for distance-3 codes on connectivity-constrained near-term quantum processors with low overhead. Furthermore, we mapped the Steane code error correction onto two current qubit topologies using the flag-bridge circuits. The numerical simulation results show that QEC circuits that can measure more stabilizers in parallel to achieve lower logical error rates, providing insights for fabricating processors with more connectivity. In addition, we have observed that the Steane code that uses fewer qubits even outperforms the distance-3 surface code when qubit idling errors are negligible. Since the Steane code also allows transversal Clifford gates, it may be a better candidate than the distance-3 surface code for demonstrating fault tolerance in small experiments. However, because the numerics in this work were carried out with Pauli errors, it will be interesting to test these circuits using more realistic error models.

As mentioned previously, general circuit compilation techniques [9,24–29] that enable two-qubit gates between nonadjacent qubits by adding extra operations such as SWAP gates or by performing gate reordering cannot be directly used for mapping QEC codes. This is because errors can propagate through two-qubit gates, and adding more two-qubit gates and reordering gate sequences may destroy the fault tolerance of a QEC circuit. For example, the two-qubit gates in a FT syndrome extraction circuit of surface codes need to be performed in a specific order [21], changing this order will lead to failures of FT error correction. For ease of implementation, the mapping procedure in this work was hand optimized to ensure the fault tolerance of QEC circuits. Future work will focus on automating the fault-tolerant mapping of flag-bridge quantum error correction onto given processors that have limited connectivity. One solution is to first generate a large number of FT QEC circuits with different connectivity requirements for a given code based on the proposed flag-bridge approach in Sec. III and then find the ones that can be supported on the underlying quantum hardware by a brute-force searching algorithm. Another possible approach is to apply circuit optimization algorithms that can mitigate errors, as in [41–43]. Moreover, we have shown that the flag-bridge circuits can be applied to the five-qubit code and lattice-surgery-based operations for the surface codes and the Steane code. The flag circuits can be used for arbitrary code distances [15], future work should also investigate the extensibility and scalability of flag-bridge circuits to higher distance codes and fault-tolerant computation.

ACKNOWLEDGMENTS

The authors would like to thank Ben Criger for enlightening discussions on this project and feedback on the manuscript. We also thank Yang Wang and Xiaotong Ni for useful discussions on the implementation of error decoders. L.L.L. acknowledges funding from the China Scholarship

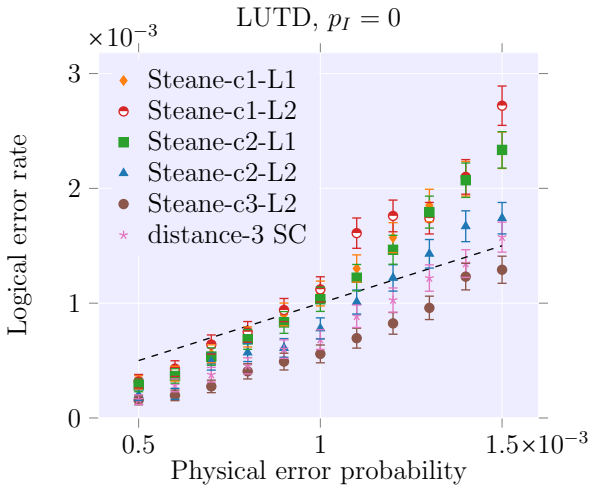


FIG. 14. Performance of the LUT decoder for the Steane code under circuit level noise without idling errors.

TABLE II. The implementation details of the NN decoder.

Loss function	Hidden layers	Activation function: Output layer	Activation function: hidden layer	Optimizer	Learning rate	Batch size	PER	Samples
cross-entropy	3	sigmoid	ReLU (tanh)	Adam (Nadam)	0.002	50	∼0.01	10^5

Council. C.G.A. acknowledges support from the Intel Corporation.

APPENDIX: IMPLEMENTATION OF LUT AND NN DECODERS

Based on the FT QEC procedure for distance-3 codes in Sec. III, decoding is only needed when two rounds of syndrome extraction (SE) are performed (the first round has nontrivial syndromes or flags). If there are only nontrivial syndromes (no flags) in the first round, then the decoders will only decode using the measurement results in the second round. If there is any nontrivial flag in the first round, then the decoders will decode using these flags and the measurement results in the second round. For the measurement information in the second round, the simple LUT decoder only considers the results of syndrome qubits, which is enough for correcting all the errors caused by one fault. In contrast, the NN decoder also takes the flags of the second round into account. This means the NN decoder could potentially correct some errors caused by more faults, outperforming the LUT decoder.

The LUT decoder. As mentioned previously, we use a brute-force search to check the fault tolerance of flag-bridge circuits. After this search, all the errors from one single fault and the corresponding syndrome-flag (SF) string are collected. For FT flag-bridge circuits, these error-SF pairs can be directly used to design a LUT decoder. Two look-up tables need to be created. One is used for the case where only syndromes are observed in the first round of SE with a size 2^{m_s}, where m_s is the total number of syndrome qubits in the QEC circuit $\mathcal{C} = \{c(g_1), c(g_2), \ldots, c(g_r)\}$. Note that if the same ancilla qubits are reused in different c_g, they are still considered as different syndrome qubits, and similarly for flag qubits. The other table is to decode for the case where flags are raised in the first round of SE, which has a size of $\sum_i 2^{m_{f_i}} 2^{m_s}$, where m_{f_i} is the total number of flag qubits in c_{g_i}. The LUT decoder is designed to correct all single faults, but not to correct the most likely two faults correspond to measured syndromes. The performance of different flag-bridge circuits for the Steane code using LUT decoders is shown in Fig. 14. As can be seen, the QEC circuits that can achieve more parallelism of stabilizer measurement have lower logical error rates.

The NN decoder. Decoding can be seen as a classification problem; that is, given the observed syndromes, the decoder identifies the error or the logical coset of the error that has occurred. It has been shown that neural networks are versatile tools for decoding topological quantum error correction codes [33–36]. The inputs x_i for a neural network decoder are the syndromes (and flags for flag QEC). In this paper, two rounds of syndromes and flags will be collected when using the flag-bridge error correction for distance-3 codes. Therefore, the size of input layer will be $2 \times m$, where m is the total number of syndrome and flag-bridge qubits. In this work, the outputs y_i are the suggested physical errors which can result in the given syndromes and flags. For a CSS code with n data qubits, the size of output layer is set to be $2 \times n$, which can describe whether a X and/or a Z error has occurred on each data qubit. The neural network will find an approximate function $f : x \to y$ to describe the input-output relation from the set of training data $\{(x_i, y_i)\}$. Note that for large-distance codes it is more efficient to use logical errors as outputs and a simple decoder (e.g., LUT decoder) is required to generate the logical error information.

In this work, a simple NN decoder using the TENSORFLOW library [44] is developed to analyze the fault tolerance of different flag-bridge circuits. We use the "sigmoid" activation function for the output layer, and 10^5 syndrome-error pairs at a physical error rate (PER) around 0.01 are sampled for each training; more details of the designed NN decoder are described in Table II. Since the focus of this work is to evaluate the flag-bridge quantum error correction, we leave the performance and speed optimization of NN decoders for future work.

[1] J. Preskill, Quantum **2**, 79 (2018).
[2] J. Kelly, A Preview of Bristlecone, Google's New Quantum Processor, Google AI Blog (2018), https://ai.googleblog.com/2018/03/a-preview-of-bristlecone-googles-new.html.
[3] W. Knight, IBM Raises the Bar with a 50-Qubit Quantum Computer, MIT Technology Review (2017), https://www.technologyreview.com/s/609451/ibm-raises-the-bar-with-a-50-qubit-quantum-computer/.
[4] J. Hsu, CES 2018: Intel's 49-Qubit Chip Shoots for Quantum Supremacy, IEEE Spectrum, general technology blog (2018), https://spectrum.ieee.org/tech-talk/computing/hardware/intels-49qubit-chip-aims-for-quantum-supremacy.
[5] R. Barends, J. Kelly, A. Megrant, A. Veitia, D. Sank, E. Jeffrey, T. C. White, J. Mutus, A. G. Fowler, B. Campbell *et al.*, Nature (London) **508**, 500 (2014).

[6] C. Hempel, C. Maier, J. Romero, J. McClean, T. Monz, H. Shen, P. Jurcevic, B. P. Lanyon, P. Love, R. Babbush *et al.*, Phys. Rev. X **8**, 031022 (2018).

[7] C. Kokail, C. Maier, R. van Bijnen, T. Brydges, M. Joshi, P. Jurcevic, C. Muschik, P. Silvi, R. Blatt, C. Roos *et al.*, Nature (London) **569**, 355 (2019).

[8] X. Fu, M. Rol, C. Bultink, J. Van Someren, N. Khammassi, I. Ashraf, R. Vermeulen, J. De Sterke, W. Vlothuizen, R. Schouten *et al.*, in *Proceedings of the 50th Annual IEEE/ACM International Symposium on Microarchitecture, October 14–18, 2017, Cambridge, MA* (ACM, New York, 2017), pp. 813–825.

[9] Y. Shi, N. Leung, P. Gokhale, Z. Rossi, D. I. Schuster, H. Hoffmann, and F. T. Chong, in *Proceedings of the Twenty-Fourth International Conference on Architectural Support for Programming Languages and Operating Systems, April 13–17, 2019, Providence, RI* (ACM, New York, 2019), pp. 1031–1044.

[10] P. W. Shor, in *Proceedings of 37th Conference on Foundations of Computer Science, October 14–16, 1996 Burlington, Vermont* (IEEE Computer Society, Los Alamitos, CA, 1996), pp. 56–65.

[11] A. M. Steane, Phys. Rev. Lett. **78**, 2252 (1997).

[12] E. Knill, Phys. Rev. A **71**, 042322 (2005).

[13] T. J. Yoder and I. H. Kim, Quantum **1**, 2 (2017).

[14] R. Chao and B. W. Reichardt, Phys. Rev. Lett. **121**, 050502 (2018).

[15] C. Chamberland and M. E. Beverland, Quantum **2**, 53 (2018).

[16] B. W. Reichardt, arXiv:1804.06995.

[17] IBM, Quantum Experience, https://www.research.ibm.com/ibm-q.

[18] R. Versluis, S. Poletto, N. Khammassi, B. Tarasinski, N. Haider, D. J. Michalak, A. Bruno, K. Bertels, and L. DiCarlo, Phys. Rev. Appl. **8**, 034021 (2017).

[19] Rigetti, Regetti forest, https://www.rigetti.com/forest.

[20] D. P. DiVincenzo and P. Aliferis, Phys. Rev. Lett. **98**, 020501 (2007).

[21] A. G. Fowler, M. Mariantoni, J. M. Martinis, and A. N. Cleland, Phys. Rev. A **86**, 032324 (2012).

[22] D. Riste, S. Poletto, M.-Z. Huang, A. Bruno, V. Vesterinen, O.-P. Saira, and L. DiCarlo, Nat. Commun. **6**, 6983 (2015).

[23] J. Kelly, R. Barends, A. G. Fowler, A. Megrant, E. Jeffrey, T. C. White, D. Sank, J. Y. Mutus, B. Campbell, Y. Chen *et al.*, Nature (London) **519**, 66 (2015).

[24] F. T. Chong, D. Franklin, and M. Martonosi, Nature (London) **549**, 180 (2017).

[25] D. Venturelli, M. Do, E. Rieffel, and J. Frank, Quantum Sci. Technol. **3**, 025004 (2018).

[26] L. Lao, B. van Wee, I. Ashraf, J. van Someren, N. Khammassi, K. Bertels, and C. G. Almudever, Quantum Sci. Technol. **4**, 015005 (2019).

[27] G. Li, Y. Ding, and Y. Xie, in *Proceedings of the Twenty-Fourth International Conference on Architectural Support for Programming Languages and Operating Systems, April 13–17, 2019, Providence, RI* (ACM, New York, 2019), pp. 1001–1014.

[28] S. S. Tannu and M. K. Qureshi, in *Proceedings of the Twenty-Fourth International Conference on Architectural Support for Programming Languages and Operating Systems, April 13–17, 2019, Providence, RI* (ACM, New York, 2019), pp. 987–999.

[29] L. Lao, D. M. Manzano, H. van Someren, I. Ashraf, and C. G. Almudever, arXiv:1908.04226.

[30] D. Gottesman, arXiv:quant-ph/9807006.

[31] A. G. Fowler, Quantum Inf. Comput. **15**, 145 (2015).

[32] G. Duclos-Cianci and D. Poulin, Phys. Rev. Lett. **104**, 050504 (2010).

[33] S. Varsamopoulos, B. Criger, and K. Bertels, Quantum Sci. Technol. **3**, 015004 (2017).

[34] X. Ni, arXiv:1809.06640.

[35] S. Krastanov and L. Jiang, Sci. Rep. **7**, 11003 (2017).

[36] P. Baireuther, T. E. O'Brien, B. Tarasinski, and C. W. Beenakker, Quantum **2**, 48 (2018).

[37] C. Horsman, A. G. Fowler, S. Devitt, and R. Van Meter, New J. Phys. **14**, 123011 (2012).

[38] A. J. Landahl and C. Ryan-Anderson, arXiv:1407.5103.

[39] H. Bombín and M. A. Martin-Delgado, J. Phys. A: Math. Theor. **42**, 095302 (2009).

[40] C. Vuillot, L. Lao, B. Criger, C. G. Almudéver, K. Bertels, and B. M. Terhal, New J. Phys. **21**, 033028 (2019).

[41] Y. Nam, N. J. Ross, Y. Su, A. M. Childs, and D. Maslov, npj Quantum Inf. **4**, 1 (2018).

[42] K. Sharma, S. Khatri, M. Cerezo, and P. Coles, New J. Phys. (2020), doi: 10.1088/1367-2630/ab784c.

[43] S. Khatri, R. LaRose, A. Poremba, L. Cincio, A. T. Sornborger, and P. J. Coles, Quantum **3**, 140 (2019).

[44] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard *et al.*, in *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI '16), November 2–4, 2016, Savannah, GA* (The USENIX Association, Berkeley, CA, 2016), pp. 265–283.