

Universal weakly fault-tolerant quantum computation via code switching in the $[[8,3,2]]$ code

Shixin Wu,^{1,2,*} Dawei Zhong^{1,2,3}, Todd A. Brun^{1,2,3,4} and Daniel A. Lidar^{1,2,3,5,6}

¹Department of Electrical and Computer Engineering, *University of Southern California*, Los Angeles, California 90089, USA

²Center for Quantum Information Science and Technology, *University of Southern California*, Los Angeles, California 90089, USA

³Department of Physics and Astronomy, *University of Southern California*, Los Angeles, California 90089, USA

⁴Department of Computer Science, *University of Southern California*, Los Angeles, California 90089, USA

⁵Department of Chemistry, *University of Southern California*, Los Angeles, California 90089, USA

⁶Quantum Elements, Inc., 2829 Townsgate Road, Westlake Village, California 91361, USA



(Received 6 May 2026; accepted 20 August 2026; published 30 September 2026)

Code switching offers a route to universal, fault-tolerant quantum computation by circumventing the limitation implied by the Eastin-Knill theorem against a universal transversal gate set within a single quantum code. Here, we present a fault-tolerant code-switching protocol between two versions of the $[[8,3,2]]$ code. One version supports weakly fault-tolerant single-qubit Clifford gates, while the other supports a logical $\overline{\text{CCZ}}$ gate via transversal $T/T^\dagger$ together with logical $\overline{\text{CZ}}$, $\overline{\text{CNOT}}$, and $\overline{\text{SWAP}}$ gates. Because both codes have distance 2, the protocol operates in a postselected, error-detecting regime: Single faults lead to detectable outcomes, and accepted runs exhibit quadratic suppression of logical error rates. This yields a universal scheme for postselected fault-tolerant computation. We validate the protocol numerically through simulations of state preparation, interblock logical teleportation, code switching, and a three-logical-qubit implementation of Grover's search.

DOI: [10.1103/4gwn-6cnx](https://doi.org/10.1103/4gwn-6cnx)

I. INTRODUCTION

Quantum error correction (QEC) protects quantum information from noise by encoding logical states into a code subspace or subsystem [1–12]. Beyond storage, quantum computation also requires controlled processing of encoded logical information. This is accomplished by applying quantum circuits that transform logical states while preserving the code. Moreover, these circuits must be *fault-tolerant* (FT): Errors must not spread faster than the code's ability to correct them, or for purely error-detecting (distance-2) codes, at least to detect them. When this condition is met for a suitable choice of code and for physical error rates below a threshold, the logical error rate can be suppressed to an arbitrarily low value for an arbitrarily long computation [13–18]. Very recently, the first empirical evidence of subthreshold scaling has started to emerge [19–23].

One of the simplest routes to fault tolerance is through *transversal* gate implementations [24]. A transversal logical gate decomposes into physical gates that act independently on each qubit within a code block (and, for two-block gates, couple only corresponding qubits across blocks). This structure prevents a single physical error from spreading within a block and turning into a higher-weight error [17,25]. While scaling quantum computers to fully error-correcting fault tol-

erance remains challenging, a recent study demonstrated an approach to *weak* fault tolerance in implementing the logical Clifford set on the high-rate $[[n, n-2, 2]]$ family of codes without using any transversal logical gates [26]. In this weakly fault-tolerant setting, circuits are designed so that any error produced by a single faulty gate is detectable. Detection is performed by measuring the ancillas used in weakly fault-tolerant implementations of two-qubit entangling gates. Runs with nontrivial syndrome are discarded [17,26]. Although this approach is error detecting rather than error correcting, postselection yields a quadratic suppression of logical errors, consistent with the expected behavior of the $[[n, n-2, 2]]$ family of codes, and does not require real-time decoding and feedback.

Encoded universal quantum computation requires a set of logical gates that can approximate any unitary transformation [27]. A common minimal choice is $\{H, \text{CNOT}, T\}$, where H and T denote the Hadamard and $\pi/8$ gates, respectively [28]. However, the Eastin-Knill theorem implies that no QEC code admits a universal set of *transversal* logical gates [25]. A widely used work-around is magic-state injection and distillation [29]: Noisy “magic states” are injected as non-Clifford resources, and distillation (using only Clifford operations) converts many noisy copies into fewer copies with higher fidelity.

The associated space-time cost depends on the chosen distillation protocol, physical error rate, target output error probability, and the code distances used in its surface-code implementation. Reference [30] showed that this cost can be substantially reduced by carefully tailoring the code distances of the different surface-code patches. More recently, magic-state cultivation has been proposed as a low-overhead preparation method that gradually increases the size and reliability

*Contact author: shixinwu@usc.edu

Published by the American Physical Society under the terms of the [Creative Commons Attribution 4.0 International](https://creativecommons.org/licenses/by/4.0/) license. Further distribution of this work must maintain attribution to the author(s) and the published article's title, journal citation, and DOI.

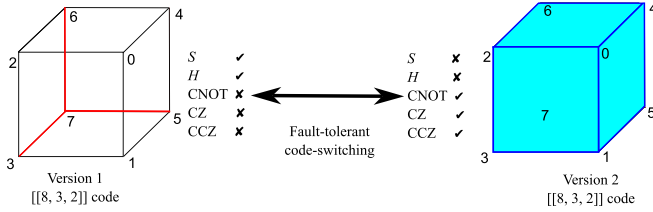


FIG. 1. Summary of this work. The two versions of the $[[8,3,2]]$ code, each of which can be geometrically represented by a cube. The three red edges of the $[[8,3,2]]_1$ code represent the fixed weight-2 gauge G^X operators, and the three blue faces of the $[[8,3,2]]_2$ code represent the fixed weight-4 gauge G^Z operators.

of a single encoded magic state and may reduce the need for subsequent distillation in favorable noise regimes [31,32].

An alternative route to a fault-tolerant universal gate set is *code switching* between codes that implement complementary logical gates via transversal physical gates [33]. Switching codes while preserving the logical information typically proceeds via *gauge fixing*, i.e., measuring suitable gauge operators and projecting onto eigenstates so that the measured eigenoperators become stabilizer generators of the target code [34–37].

Fault-tolerant code-switching protocols have been developed, for example, between the $[[7,1,3]]$ Steane code (the smallest two-dimensional color code) and the $[[15,1,3]]$ Reed-Muller code (the smallest three-dimensional color code) [33,37]. The same work also presented a protocol between the $[[7,1,3]]$ Steane code and a “morphed” $[[10,1,2]]$ code derived from the $[[15,1,3]]$ Reed-Muller code [38], reducing circuit depth and resource requirements, at the cost of reduced error-correction capability.

In this work, we further reduce the required resources by code switching between two versions of the $[[8,3,2]]$ code. Because our codes have distance 2, we adopt fault tolerance in the error-detecting (postselected) sense. One version of the $[[8,3,2]]$ code is the “smallest interesting color code,” with fault-tolerant implementations of the logical CCZ, CZ, and CNOT gates [39–43]. We call this the *Version 2* $[[8,3,2]]$ code, because it lacks fault-tolerant implementations of logical single-qubit gates. The *Version 1* $[[8,3,2]]$ code can instead use the weakly fault-tolerant circuits of Ref. [26] to implement the logical H , S , and $\sqrt{X}^\dagger$ gates. Together, the two codes provide an overcomplete universal set of logical gates while still achieving quadratic suppression of logical errors with postselection. This protocol is summarized in Fig. 1.

In comparison to the existing error-detecting scheme via code switching between the $[[7,1,3]]$ code and the $[[10,1,2]]$ code, this work reduces the number of physical qubits and increases the number of logical qubits, yielding a more efficient encoding. As a result, our scheme is well suited for near-term proof-of-principle demonstrations under realistic qubit constraints, such as running small-scale quantum algorithms (e.g., Grover’s algorithm [44,45]) within a single code block. Moreover, choosing distance-2 codes allows us to optimize the encoding rate while maintaining consistency with the implementation of single-qubit Clifford gates, which are fault tolerant in the error-detecting sense. Finally, because many

quantum algorithms make extensive use of the Toffoli gate [46–55], native logical ccz support can reduce the synthesis depth and non-Clifford overhead of Toffoli-intensive circuits relative to schemes in which ccz must be synthesized from other logical resources.

The remainder of the paper is organized as follows. We first review, in Sec. II, the weakly fault-tolerant logical gates available on the $[[n,n-2,2]]$ family. Then, in Sec. III, we introduce our two versions of the $[[8,3,2]]$ code. Additionally, we introduce the circuit that composes multiple blocks of $[[8,3,2]]$ codes for larger-scale computations in this section. In Sec. IV, we present a fault-tolerant code-switching protocol between them. Next, in Sec. V, we summarize fault-tolerant state-preparation routines for logical states relevant to applications. Finally, in Secs. VI and VII, we report numerical simulations of state preparation, interblock logical teleportation, code switching, and an encoded three-logical-qubit Grover search. Appendixes A–C present various technical details of our protocol, including fault-tolerant implementations of two-qubit rotation gates, encoding circuits used in the simulations of state preparation and Grover search, and extensions of the single interblock logical CNOT gate.

II. FAULT-TOLERANT LOGICAL GATES ON $[[n,n-2,2]]$ CODES

This section summarizes the constructions in Ref. [26] for implementing weakly fault-tolerant logical single-qubit Clifford gates (H , S , and $\sqrt{X}^\dagger$) on the $[[n,n-2,2]]$ code family with two-qubit entangling gates. Since these codes have distance $d = 2$, our notion of fault tolerance is the $d = 2$, postselected one: Any single faulty gate during the circuit must either leave the data in the codespace or produce an error that is detected by the stabilizer (and ancilla) measurements, so that the run can be discarded.

The $[[n,n-2,2]]$ code has stabilizer generators

$$S_X = X^{\otimes n}, \quad S_Z = Z^{\otimes n}. \quad (1)$$

A convenient choice of logical Pauli operators is

$$\bar{X}_j = X_j X_{n-2}, \quad \bar{Z}_j = Z_j Z_{n-1} \quad (j = 0, \dots, n-3), \quad (2)$$

with $\bar{Y}_j \equiv i\bar{X}_j\bar{Z}_j$ (global phases are irrelevant throughout).

Logical gates implemented *transversally* are naturally compatible with weak fault tolerance: A single faulty location can create at most a weight-1 error in each code block, which remains detectable for a distance-2 code. In contrast, a faulty entangling gate that acts *within* a block can produce a weight-2 error in the code block, potentially causing logical errors on a distance-2 code. Such gates can still be used in a weakly fault-tolerant manner, but only when embedded in ancilla-assisted measurement gadgets that ensure any single fault produces a detectable syndrome.

It is useful to distinguish three conceptual levels:

$$\text{logical level} \rightarrow \text{physical level} \rightarrow \text{FT physical level}. \quad (3)$$

At the *logical level*, we denote ideal encoded operations with a straight overline (e.g., $\bar{H}_j$). At the *physical level*, we use a wavy overline to denote the corresponding compilation into elementary physical operations; the particular two-qubit rotations used below are defined in Sec. II A. These physical-level

TABLE I. Conjugation action of the XX and ZZ rotation gates on selected two-qubit Pauli operators (e.g., XI denotes $X \otimes I$). Other Pauli mappings follow analogously.

	XI	ZI	YX	YZ	XX	ZZ
XX gate	XI	YX	$-ZI$	YZ	XX	ZZ
ZZ gate	YZ	ZI	YX	$-XI$	XX	ZZ

two-qubit entangling gates are not fault tolerant on a distance-2 code and must be replaced by fault-tolerant gadgets at the *FT physical level*; the gadgets we use are shown in Appendix A. We write products of unitaries in the usual right-to-left circuit order (the rightmost factor acts first).

A. Elementary Clifford rotations

Both the physical-level compilation and the FT gadgets use the following Clifford rotations:

$$U_{XX} := \frac{I + iXX}{\sqrt{2}}, \quad U_{YY} := \frac{I - iYY}{\sqrt{2}}, \quad (4a)$$

$$U_{ZZ} := \frac{I - iZZ}{\sqrt{2}}, \quad R_X := \frac{I - iX}{\sqrt{2}}, \quad (4b)$$

where XX , YY , and ZZ denote products of Pauli operators on two qubits (e.g., $XX = X \otimes X$). We refer to these as the XX , YY , ZZ , and R_X gates; note the opposite sign convention for the XX gate. We use $\widetilde{XX}(i, j)$ [resp. $\widetilde{ZZ}(i, j)$] to denote applying U_{XX} (resp. U_{ZZ}) to physical qubits i and j .

For later reference, Table I lists the conjugation action of the XX and ZZ rotations on a set of two-qubit Pauli operators used in the discussion.

B. Logical level $\rightarrow$ physical level

At the physical level, the logical single-qubit gates $\bar{S}_j$ (where S is the phase gate), $\sqrt{X}_j^\dagger$, and $\bar{H}_j$ can be expressed using $\widetilde{XX}$ and $\widetilde{ZZ}$ rotations involving the “shared” qubits $n-2$ and $n-1$:

$$\bar{S}_j = \widetilde{ZZ}(j, n-1), \quad \sqrt{X}_j^\dagger = \widetilde{XX}(j, n-2), \quad (5)$$

and

$$\begin{aligned} \bar{H}_j &= \widetilde{ZZ}(j, n-1) \times \widetilde{XX}(j, n-2) \\ &\times \widetilde{ZZ}(j, n-1) \times Y_j X_{n-2} Z_{n-1}. \end{aligned} \quad (6)$$

The final factor $Y_j X_{n-2} Z_{n-1}$ is a Pauli by-product correction (not merely a global phase). The sequence $\widetilde{ZZ} \times \widetilde{XX} \times \widetilde{ZZ}$ implements $\bar{H}_j$ up to a known logical $\bar{Y}_j$ correction, and with our choice of logical operators,

$$\bar{Y}_j \equiv Y_j X_{n-2} Z_{n-1} \quad (7)$$

up to an irrelevant phase.

The identities above follow from Table I together with the standard single-qubit Clifford conjugation rules

$$\begin{aligned} SX S^\dagger &= Y, \quad SY S^\dagger = -X, \quad SZ S^\dagger = Z, \\ \sqrt{X}^\dagger X \sqrt{X} &= X, \quad \sqrt{X}^\dagger Y \sqrt{X} = -Z, \quad \sqrt{X}^\dagger Z \sqrt{X} = Y, \end{aligned}$$

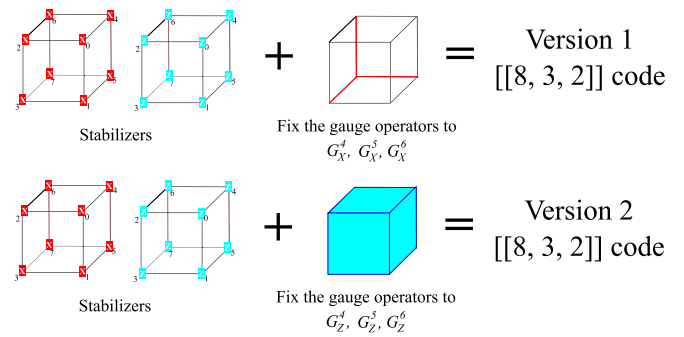


FIG. 2. Obtaining the two versions of the $[[8,3,2]]$ code by fixing the gauge operators of the $[[8,3,3,2]]$ subsystem code. Together with the transversal $X^{\otimes 8}$ and $Z^{\otimes 8}$ stabilizers of the $[[8,3,3,2]]$ subsystem code, fixing the gauge operators gives the different stabilizers that define the two stabilizer $[[8,3,2]]$ codes.

$$HXH = Z, \quad HYH = -Y, \quad HZH = X. \quad (8)$$

For example, $\widetilde{ZZ}(j, n-1)$ leaves $\bar{Z}_j$ invariant and maps $\bar{X}_j \mapsto \bar{Y}_j$ (up to phase), matching the action of $\bar{S}_j$. Similarly, $\widetilde{XX}(j, n-2)$ leaves $\bar{X}_j$ invariant and maps $\bar{Z}_j \mapsto \bar{Y}_j$, matching the action of $\sqrt{X}_j^\dagger$.

C. Physical level $\rightarrow$ FT physical level

The two-qubit entangling gates $\widetilde{XX}$ and $\widetilde{ZZ}$ are not automatically fault tolerant on a distance-2 code, because they may produce weight-2 errors on the data qubits that are logical errors in most cases. Because the probabilities for two-qubit entangling gates to produce weight-1 and weight-2 errors are both on the order of $O(p)$, where p is the error probability of the gate, logical errors cannot be quadratically suppressed. Reference [26] replaces each $\widetilde{XX}(j, n-2)$ and $\widetilde{ZZ}(j, n-1)$ by an ancilla-assisted, verification/postselection gadget that is fault tolerant in the error-detecting sense where all two-qubit entangling gates act on a data qubit and an ancilla and none on two data qubits. We reproduce these gadgets in Appendix A (Fig. 11) and use them as primitives in the remainder of this paper.

III. TWO VERSIONS OF THE $[[8,3,2]]$ CODE

In this section, we discuss two versions of the $[[8,3,2]]$ stabilizer code: the Version 1 $[[8,3,2]]_1$ code ($[[8,3,2]]_1$) and the Version 2 $[[8,3,2]]_2$ code ($[[8,3,2]]_2$), as well as their construction from the $[[8,6,2]]$ code. The constructions of the two versions are summarized in Fig. 2.

A. Subsystem $[[8,3,3,2]]$ code

The subsystem $[[8,3,3,2]]$ code [56] is constructed from the $n = 8$ $[[n, n-2, 2]]$ stabilizer code by “gauging out” three of its six logical operators, i.e., not using them to store logical data and not requiring that they be protected. The stabilizers of the $[[8,6,2]]$ code are $X^{\otimes 8}$ and $Z^{\otimes 8}$. We then choose the following logical operators, which differ from the canonical all-weight-2 choice in Eq. (2). This choice is crucial for the code-switching protocol to function and corresponds to

changing the basis of the codespace.

$$\begin{aligned}
\bar{X}_1 &= X_0 X_1 X_2 X_3, & \bar{Z}_1 &= Z_0 Z_4, \\
\bar{X}_2 &= X_0 X_1 X_4 X_5, & \bar{Z}_2 &= Z_0 Z_2, \\
\bar{X}_3 &= X_0 X_2 X_4 X_6, & \bar{Z}_3 &= Z_0 Z_1, \\
G_4^X &= X_3 X_7, & G_4^Z &= Z_0 Z_1 Z_2 Z_3, \\
G_5^X &= X_5 X_7, & G_5^Z &= Z_0 Z_1 Z_4 Z_5, \\
G_6^X &= X_6 X_7, & G_6^Z &= Z_0 Z_2 Z_4 Z_6.
\end{aligned} \tag{9}$$

Gauging out the last three logical qubits gives us the $[[8,3,2]]$ subsystem code. We refer to the last three logical qubits as “gauge qubits” and their corresponding logical operators $G_{i=4,5,6}^{X/Z}$ as “gauge operators.”

B. The $[[8,3,2]]_1$ stabilizer code

Version 1 of the $[[8,3,2]]$ stabilizer code is obtained by fixing the gauge qubits of the $[[8,3,3,2]]$ code into the logical $|\bar{+}\rangle$ state. This means that the logical X operators of the three gauge qubits now become stabilizers of the code. Thus, the new set of stabilizer generators is

$$X^{\otimes 8}, Z^{\otimes 8}, X_3 X_7, X_5 X_7, X_6 X_7. \tag{10}$$

The additional X stabilizer generators reduce the first three logical operators to weight 2:

$$\begin{aligned}
\bar{X}_1 &= X_0 X_1 X_2 X_3 \times X^{\otimes 8} \times X_5 X_7 = X_4 X_6, \\
\bar{X}_2 &= X_0 X_1 X_4 X_5 \times X^{\otimes 8} \times X_3 X_7 = X_2 X_6, \\
\bar{X}_3 &= X_0 X_2 X_4 X_6 \times X^{\otimes 8} \times X_3 X_5 X_6 X_7 = X_1 X_6.
\end{aligned} \tag{11}$$

Its logical Z operators remain unchanged:

$$\bar{Z}_1 = Z_0 Z_4, \quad \bar{Z}_2 = Z_0 Z_2, \quad \bar{Z}_3 = Z_0 Z_1. \tag{12}$$

Note that the logical operators are similar to those used in Ref. [26], in that all of them are weight 2 [instead of a mix of weight 4 and weight 2 as in Eq. (9)] and, within each Pauli type, the same shared qubit is used for all three logical qubits: qubit 6 for the logical $\bar{X}$ operators and qubit 0 for the logical $\bar{Z}$ operators. As a result, we can use the same construction as in Sec. II B to implement the logical $\bar{H}$, $\bar{S}$, and $\sqrt{\bar{X}}^\dagger$ gates for the Version 1 $[[8,3,2]]$ code:

$$\begin{aligned}
\bar{H}_1 &= \widetilde{ZZ}(4, 0) \times \widetilde{XX}(4, 6) \times \widetilde{ZZ}(4, 0) \times Y_4 X_6 Z_0, \\
\bar{H}_2 &= \widetilde{ZZ}(2, 0) \times \widetilde{XX}(2, 6) \times \widetilde{ZZ}(2, 0) \times Y_2 X_6 Z_0, \\
\bar{H}_3 &= \widetilde{ZZ}(1, 0) \times \widetilde{XX}(1, 6) \times \widetilde{ZZ}(1, 0) \times Y_1 X_6 Z_0, \\
\bar{S}_1 &= \widetilde{ZZ}(4, 0), & \sqrt{\bar{X}}_1^\dagger &= \widetilde{XX}(4, 6), \\
\bar{S}_2 &= \widetilde{ZZ}(2, 0), & \sqrt{\bar{X}}_2^\dagger &= \widetilde{XX}(2, 6), \\
\bar{S}_3 &= \widetilde{ZZ}(1, 0), & \sqrt{\bar{X}}_3^\dagger &= \widetilde{XX}(1, 6).
\end{aligned} \tag{13}$$

Then, we use the fault-tolerant implementation of Fig. 11 to replace $\widetilde{XX}$ and $\widetilde{ZZ}$. Finally, logical SWAPs among the three encoded qubits are induced by the corresponding permutations of the physical qubits $\{1, 2, 4\}$, since these are the qubits that distinguish the three logical-qubit labels.

On hardware platforms with dynamically reconfigurable or movable qubits, or with effective all-to-all connectivity, such as reconfigurable neutral-atom arrays, shuttling trapped-ion architectures, and small nonshuttling trapped-ion registers, these operations can often be absorbed into a software-level remapping of qubit identities rather than implemented as explicit logical gate sequences [57–60]. On architectures without such flexibility or connectivity, the same permutations must instead be compiled into explicit routing operations, typically SWAP networks on devices with restricted connectivity [61,62].

Although a fault-tolerant implementation of the logical intrablock CNOT gate was also proposed in Ref. [26], we opt not to use it in the Version 1 $[[8,3,2]]$ code due to its large circuit depth. Rather, cnots can be easily implemented in Version 2, as we show next.

C. The $[[8,3,2]]_2$ stabilizer code

Version 2 of the $[[8,3,2]]$ stabilizer code is obtained by fixing the gauge qubits of the $[[8,3,3,2]]$ code in the logical $|\bar{0}\rangle$ state. This means that the logical Z operators of the gauge qubits now become stabilizer generators for the code. Thus, the new set of stabilizer generators is

$$X^{\otimes 8}, Z^{\otimes 8}, Z_0 Z_1 Z_2 Z_3, Z_0 Z_1 Z_4 Z_5, Z_0 Z_2 Z_4 Z_6. \tag{14}$$

This recovers the $[[8,3,2]]$ code (version 2) that allows a fault-tolerant logical CCZ gate with transversal T and $T^\dagger$ gates [39]:

$$\overline{\text{CCZ}} = T_0 T_1^\dagger T_2^\dagger T_3^\dagger T_4^\dagger T_5^\dagger T_6^\dagger T_7^\dagger. \tag{15}$$

In addition, this code also allows fault-tolerant implementations of the logical CZ, SWAP, intrablock CNOT gates, and interblock CNOT gates [40–43]. The logical CZ gates are implemented with phase gates:

$$\begin{aligned}
\overline{\text{CZ}}(1, 2) &= S_0 S_2^\dagger S_4^\dagger S_6, \\
\overline{\text{CZ}}(1, 3) &= S_0 S_1^\dagger S_4^\dagger S_5, \\
\overline{\text{CZ}}(2, 3) &= S_0 S_1^\dagger S_2^\dagger S_3.
\end{aligned} \tag{16}$$

The logical SWAP gates are implemented by cyclically permuting the physical qubits:

$$\begin{aligned}
\overline{\text{SWAP}}(1, 2) &: 0 \rightarrow 4 \rightarrow 6 \rightarrow 2 \rightarrow 0, \quad 1 \rightarrow 5 \rightarrow 7 \rightarrow 3 \rightarrow 1, \\
\overline{\text{SWAP}}(1, 3) &: 2 \rightarrow 6 \rightarrow 7 \rightarrow 3 \rightarrow 2, \quad 0 \rightarrow 4 \rightarrow 5 \rightarrow 1 \rightarrow 0, \\
\overline{\text{SWAP}}(2, 3) &: 0 \rightarrow 2 \rightarrow 3 \rightarrow 1 \rightarrow 0, \quad 4 \rightarrow 6 \rightarrow 7 \rightarrow 5 \rightarrow 4.
\end{aligned} \tag{17}$$

Similarly, these physical SWAP operations can be treated as qubit permutations, explicit qubit motions, or software-level relabelings, depending on the connectivity and layout of the architecture.

Likewise, the following intrablock logical CNOT gates are induced by physical-qubit permutations:

$$\begin{aligned}
\overline{\text{CNOT}}(1, 2) &= 0 \leftrightarrow 4, \quad 1 \leftrightarrow 5, & \overline{\text{CNOT}}(2, 1) &= 0 \leftrightarrow 2, \quad 1 \leftrightarrow 3, \\
\overline{\text{CNOT}}(1, 3) &= 0 \leftrightarrow 4, \quad 2 \leftrightarrow 6, & \overline{\text{CNOT}}(3, 1) &= 0 \leftrightarrow 1, \quad 2 \leftrightarrow 3, \\
\overline{\text{CNOT}}(2, 3) &= 0 \leftrightarrow 2, \quad 4 \leftrightarrow 6, & \overline{\text{CNOT}}(3, 2) &= 0 \leftrightarrow 1, \quad 4 \leftrightarrow 5.
\end{aligned} \tag{18}$$

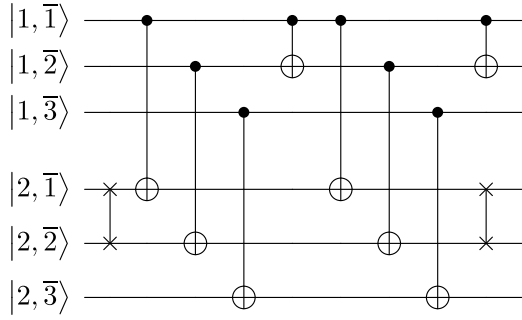


FIG. 3. The logical circuit for the single-pair interblock $\overline{\text{CNOT}}$ from logical qubit $\bar{1}$ of block 1 to logical qubit $\bar{1}$ of block 2 [40]. If the two logical $\overline{\text{SWAP}}$ gates are omitted, the circuit instead implements $\overline{\text{CNOT}}(|1, \bar{1}\rangle, |2, \bar{2}\rangle)$; the SWAPS simply SWAP the target from $|2, \bar{2}\rangle$ to $|2, \bar{1}\rangle$. Variants of this construction are given in Appendix C.

D. Block scalability

We consider scaling up with blocks of the $[[8,3,2]]$ code, which requires performing entangling gates between logical qubits in different blocks.

A parallel interblock logical $\overline{\text{CNOT}}$ gate means that the transversal physical CNOT gates are applied between corresponding qubits of two code blocks, so that the three corresponding logical qubits undergo logical $\overline{\text{CNOT}}$ gates in parallel. When the two blocks are in the same version of the $[[8,3,2]]$ code, this transversal CNOT preserves the Calderbank–Shor–Steane stabilizer structure and therefore defines a valid encoded operation [63]. For mixed versions, the situation is asymmetric. If the control block is $[[8,3,2]]_1$ and the target block is $[[8,3,2]]_2$, the fixed gauge- X operators $G_{4,5,6}^X$ of the control are copied to the target under transversal cnot, so the target block is taken out of the $[[8,3,2]]_2$ codespace. In the reverse direction, with $[[8,3,2]]_2$ as control and $[[8,3,2]]_1$ as target, no such fixed gauge- X operators are copied, so the codespace is preserved. Therefore a parallel interblock logical $\overline{\text{CNOT}}$ is available between mixed-version blocks only in the direction $[[8,3,2]]_2 \rightarrow [[8,3,2]]_1$.

The logical circuit for a single-pair interblock $\overline{\text{CNOT}}$ is shown in Fig. 3 [40]. It uses only logical $\overline{\text{SWAP}}$ gates, parallel interblock logical $\overline{\text{CNOT}}$ gates, and intrablock logical $\overline{\text{CNOT}}$ gates. Consequently, such a single-pair interblock logical $\overline{\text{CNOT}}$ is available whenever those three ingredients are available. They are always available between two Version 1 blocks or between two Version 2 blocks in principle, but in practice, the Version 2 realization uses much simpler circuits to implement the intrablock logical $\overline{\text{CNOT}}$ gates. Between mixed-version blocks, the same asymmetry as above remains, so a single-pair interblock logical $\overline{\text{CNOT}}$ is possible from $[[8,3,2]]_2$ to $[[8,3,2]]_1$ but not in the reverse direction.

An interblock logical $\overline{\text{CCZ}}$ gate (with participating logical qubits initially residing in different blocks) can be implemented after first switching the relevant blocks into a compatible version (typically Version 2). One then uses interblock logical $\overline{\text{SWAP}}$ operations implemented with interblock $\overline{\text{CNOT}}$ gates to bring the three participating logical qubits into a common block, applies the intrablock logical $\overline{\text{CCZ}}$ gate there, and finally reverses the SWAPS to return the logical qubits back to their original blocks.

IV. CODE SWITCHING BETWEEN TWO VERSIONS OF THE $[[8,3,2]]$ CODE

The two versions of the $[[8,3,2]]$ code are different gauge fixings of the parent $[[8,3,2,1]]$ subsystem code. A direct switch by applying Hadamard gates to the three gauge qubits would exchange the X -type and Z -type gauge operators, but we do not know a fault-tolerant implementation of those gauge-qubit Hadamards, because the corresponding gauge operators have different weights in the two versions. We therefore switch between the two codes by measuring the gauge operators of the target version and, if necessary, applying gauge corrections that return all three gauge outcomes to $+1$.

A. Switching from Version 2 to 1

To switch from Version 2 to Version 1, we measure G_4^X , G_5^X , and G_6^X . These measurements project the gauge qubits onto eigenspaces of the X -type gauge operators. For each outcome $M_i^X = -1$, we apply the corresponding operator G_i^Z ; we refer to this as a *gauge correction*.

For fault tolerance, we also measure a complementary check $X_0X_1X_2X_4$ that satisfies

$$G_4^X G_5^X G_6^X \times X_0X_1X_2X_4 = X^{\otimes 8}. \quad (19)$$

Hence, the product of the three gauge outcomes and the outcome of $X_0X_1X_2X_4$ is the precorrection $X^{\otimes 8}$ syndrome. If this parity is -1 , we discard the run before applying any gauge correction. We also verify the target-code stabilizer $X^{\otimes 8}$. The gauge outcomes alone determine the correction, but the complementary check and the $X^{\otimes 8}$ stabilizer verification are needed when the switched block will subsequently be used for logical gates.

B. Switching from Version 1 to 2

The reverse direction is analogous. We measure G_4^Z , G_5^Z , and G_6^Z , and for each outcome $M_i^Z = -1$ we apply G_i^X . We also measure a complementary check $Z_1Z_2Z_4Z_7$ chosen so that

$$G_4^Z G_5^Z G_6^Z \times Z_1Z_2Z_4Z_7 = Z^{\otimes 8}. \quad (20)$$

Therefore, the product of the four measurement outcomes gives the precorrection $Z^{\otimes 8}$ syndrome. If this parity is -1 , we discard the run before applying any gauge correction. We also verify the target-code stabilizer $Z^{\otimes 8}$. Again, the gauge outcomes determine the correction, while the complementary check and the $Z^{\otimes 8}$ stabilizer verification ensure that the switched block is in the correct sector before any Version 2 logical gate is applied.

C. Remarks on fault tolerance

Fault-tolerant code switching must do more than preserve the detectability of a single data error. It must also place the block in the correct gauge and stabilizer sector of the target code before any subsequent logical gate is applied.

This matters in both versions of the code. In the $[[8,3,2]]_2$ code, the transversal $T/T^\dagger$ pattern implements the logical $\overline{\text{CCZ}}$ gate only on the $+1$ eigenspace of the Z -type stabilizers, including G_4^Z , G_5^Z , and G_6^Z . In the $[[8,3,2]]_1$ code, the convenient weight-2 representatives of the logical $\bar{X}$ operators differ from

ALGORITHM 1. Fault-tolerant code switching from Version 1 to Version 2.

```

Measure the gauge operators
 $M_4^Z \leftarrow \text{Meas}(G_4^Z)$ ,  $M_5^Z \leftarrow \text{Meas}(G_5^Z)$ ,
 $M_6^Z \leftarrow \text{Meas}(G_6^Z)$ ;
Measure the complementary check
 $M_7^Z \leftarrow \text{Meas}(Z_1 Z_2 Z_4 Z_7)$ ;
Verify the  $Z^{\otimes 8}$  stabilizer
 $M_8^Z \leftarrow \text{Meas}(Z^{\otimes 8})$ ;
if  $\prod_{i=4,5,6,7} M_i^Z = +1$  and  $M_8^Z = +1$  then
    foreach  $i \in \{4, 5, 6\}$  such that  $M_i^Z = -1$  do
        Apply  $G_i^X$ 
else
    Discard current run;

```

the original weight-4 representatives by products of G_4^X , G_5^X , and G_6^X . A wrong gauge sign can therefore change the action of later logical gates, even if a later full target-code stabilizer measurement round, including the fixed gauge operators, would eventually reject the state.

The complementary check and the $X^{\otimes 8}/Z^{\otimes 8}$ stabilizer verification play different roles. The complementary check detects any single fault that flips the relevant $X^{\otimes 8}/Z^{\otimes 8}$ stabilizer syndrome before a gauge correction is applied, including a single ancilla fault during a gauge measurement. The $X^{\otimes 8}/Z^{\otimes 8}$ stabilizer verification detects faults that occur between successive gauge measurements and can therefore evade the precorrection parity while still leading to an incorrect gauge correction. For example, in the Version-1-to-Version-2 protocol, an X_0 fault after the G_4^Z measurement changes the remaining gauge outcomes to $(+1, -1, -1)$ while leaving $Z_1 Z_2 Z_4 Z_7$ unchanged. The precorrection parity, therefore, remains $+1$, so one would incorrectly apply G_5^X and G_6^X ; the subsequent $Z^{\otimes 8}$ verification would then reject the run, as it detects the X_0 error.

The two protocols are summarized in Algorithms 1 and 2, where $\text{Meas}(P)$ denotes the eigenvalue ± 1 returned by measuring the Pauli operator P .

As in syndrome extraction, weight-4 gauge measurements require flag qubits [64] to detect single ancilla faults that

ALGORITHM 2. Fault-tolerant code switching from Version 2 to Version 1.

```

Measure the gauge operators
 $M_4^X \leftarrow \text{Meas}(G_4^X)$ ,  $M_5^X \leftarrow \text{Meas}(G_5^X)$ ,
 $M_6^X \leftarrow \text{Meas}(G_6^X)$ ;
Measure the complementary check
 $M_7^X \leftarrow \text{Meas}(X_0 X_1 X_2 X_4)$ ;
Verify the  $X^{\otimes 8}$  stabilizer
 $M_8^X \leftarrow \text{Meas}(X^{\otimes 8})$ ;
if  $\prod_{i=4,5,6,7} M_i^X = +1$  and  $M_8^X = +1$  then
    foreach  $i \in \{4, 5, 6\}$  such that  $M_i^X = -1$  do
        Apply  $G_i^Z$ 
else
    Discard current run;

```

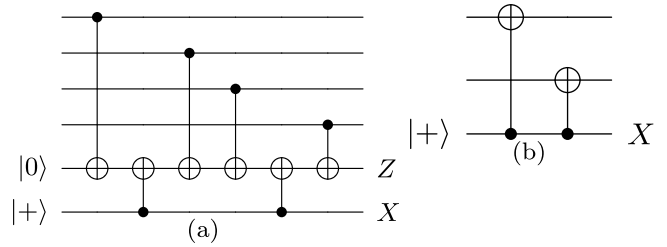


FIG. 4. (a) Flag-based measurement of a weight-4 Z -type gauge operator used in switching from Version 1 to Version 2. The upper four wires are the data qubits in the support of the measured operator, the fifth wire is the ancilla measured in the Z basis to decide whether G^X corrections are necessary, and the last wire is a flag ancilla measured in the X basis. A triggered flag identifies single-fault events that can propagate to weight-2 Z errors on the data qubits. (b) Fault-tolerant measurement of a weight-2 X -type gauge operator used in switching from Version 2 to Version 1. The upper two wires are the data qubits in the support of the measured operator; the third wire is the ancilla measured in the X basis to decide whether G^Z corrections are necessary. Because the measured operator has weight 2, no flag ancilla is required.

can propagate to weight-2 data errors. Therefore, switching from Version 1 to Version 2 requires flag qubits, because the measured G^Z operators have weight 4 (see Fig. 4). Switching from Version 2 to Version 1 does not, because the measured G^X operators have weight 2 and a single ancilla fault cannot propagate to a higher weight.

V. FAULT-TOLERANT ENCODING CIRCUITS

We require fault-tolerant preparation routines for several logical input states in both versions of the $[[8,3,2]]$ code. We use $|+20^1\rangle$ and $|+10^2\rangle$ to denote mixed logical-basis states, specifying only the numbers of $|+\rangle$ and $|0\rangle$ states and not their ordering. The detailed stabilizer reductions, circuit constructions, and explicit encoding circuits are given in Appendix B; here, we summarize the main results.

Most of the relevant encoded states reduce, after Gaussian elimination of their stabilizer generators, to standard entangled resource states [Greenberger-Horne-Zeilinger (GHZ) states, dual GHZ states, Bell pairs, or tensor products thereof], so their preparation can be built from standard fault-tolerant GHZ/Bell-state gadgets. In particular, both versions admit fault-tolerant preparation of $|000\rangle$ and Version 1 admits a direct fault-tolerant preparation of $|+++\rangle$. Without additional gauge fixing, Version 1 naturally supports states of the form $|+20^1\rangle$, while Version 2 naturally supports states of the form $|+10^2\rangle$. For Version 2, we use the preparation circuit from Ref. [42]; in the noisy-initialization Grover benchmark, we augment it with additional flagged G_5^Z and G_6^Z checks to monitor harmful gauge-sector faults.

When the opposite mixed-basis pairing is required, namely, $|+20^1\rangle$ in Version 2 or $|+10^2\rangle$ in Version 1, we do not perform a full code switching. Instead, we prepare a partially gauge-fixed intermediate state and measure the remaining single gauge operator, along with its complementary check, before immediately applying the stabilizer and gauge checks. This achieves the desired encoded state with lower overhead than a

full switching subroutine while still ensuring that the block is in the correct gauge and stabilizer sector for subsequent logical operations.

Appendix B collects the explicit initialization circuits drawn from these constructions. Their simulated performance is discussed in the next section.

VI. SIMULATION

In this section, we use numerical simulations to evaluate the logical error rates of our state-preparation circuits and code-switching protocol in the presence of faulty gates. A faulty gate is modeled as $\tilde{\mathcal{U}} = \mathcal{N} \circ \mathcal{U}$, where $\mathcal{U}(\rho) = U\rho U^\dagger$ is a noise-free unitary channel and $\mathcal{N}$ is the noise channel associated with the gate. Whenever noise is assigned to a single- or two-qubit gate, we use the depolarizing channels

$$\mathcal{N}_1(\rho) = (1-p)\rho + \frac{p}{3} \sum_{P \in \{X,Y,Z\}} P\rho P, \quad (21)$$

$$\mathcal{N}_2(\rho) = (1-p)\rho + \frac{p}{15} \sum_{P \in \mathcal{P}_2 \setminus \{I \otimes I\}} P\rho P, \quad (22)$$

where $\mathcal{P}_2 = \{I, X, Y, Z\}^{\otimes 2}$. Thus, each nonidentity single-qubit Pauli error is applied with probability $p/3$, and each nonidentity two-qubit Pauli error is applied with probability $p/15$. The same physical error probability p is used whenever both single- and two-qubit gate noise are included. The precise set of noisy gate locations is specified in each subsection: The state-preparation benchmark assigns noise only to CNOT gates, whereas the teleportation, code-switching, and Grover benchmarks assign noise to each physical one- and two-qubit gate.

Physical-qubit initialization, measurement, feedforward, and Pauli or gauge corrections are treated as ideal. For the Pauli-basis preparations and measurements used here, stochastic Pauli preparation and readout faults can be represented as Pauli errors immediately after preparation or immediately before measurement. We nevertheless do not sample these additional boundary error locations. Our numerical results therefore isolate gate-induced faults and should not be interpreted as hardware-level performance predictions that include state-preparation and measurement noise; including such faults could change the quantitative logical error and acceptance curves. The state-preparation, interblock logical teleportation, and code-switching benchmarks are performed using the Python package PECOS [65], a tool designed for efficient Monte Carlo simulations of quantum Clifford circuits. The precise set of noisy operations is specified in each subsection.

A. State preparation

In the presence of noise, each state-preparation run falls into one of three categories: successful runs, discarded runs, and failed runs. A run is discarded whenever postselection detects a fault, either through a nontrivial flag outcome or through the final noiseless stabilizer and gauge measurements. Among the accepted runs, the preparation is counted as successful if the output matches the target logical state up to stabilizers, and as a failure otherwise. For example, preparing the logical $|+++\rangle$ state for the $[[8,3,2]]_1$ code counts as a

failure if the output state is a $+1$ eigenstate of the stabilizers in Eq. (10) but not a $+1$ eigenstate of the logical operators $\bar{X}_1, \bar{X}_2$, and $\bar{X}_3$ defined in Eq. (11).

Figure 5 shows the logical error rate and acceptance rate for state-preparation circuits of Versions 1 and 2 of the $[[8,3,2]]$ code after 10^6 Monte Carlo samples. In these simulations, we apply depolarizing noise with physical error rate p to every CNOT gate and report both the postselected logical error rate

$$p_L = \frac{n_{\text{failure}}}{n_{\text{postselected}}} \quad (23)$$

and the acceptance rate

$$R = \frac{n_{\text{postselected}}}{n_{\text{total}}}, \quad (24)$$

i.e., the fraction of accepted runs among all runs.

The quantity p_L therefore measures the logical infidelity conditioned on acceptance, whereas R measures the postselection overhead. Because both versions of the $[[8,3,2]]$ code are error detecting rather than error correcting, the relevant signature of weak fault tolerance is that accepted runs exhibit $p_L = O(p^2)$. This is what we observe in the left panels of Fig. 5: For $p < 0.1$, the postselected logical error rate scales approximately quadratically with the physical error rate, indicating that only weight-2 (or higher) faults contribute to accepted-run logical failures in the state-preparation circuits described in Appendix B.

The right panels of Fig. 5 show the complementary behavior of the acceptance rate. At low physical error rates, R remains close to 1 for all preparations, and it decreases monotonically as p increases because single detectable faults are removed by postselection. Thus, the simulations exhibit the expected trade-off for an error-detecting protocol: Postselection suppresses the logical error rate of accepted runs to second order, but it does so at the cost of an acceptance penalty that appears already at first order in p . The spread among the acceptance curves is circuit dependent and reflects differences in circuit size and in the number and type of postselection checks. In Version 1, the four acceptance curves remain fairly close, indicating comparable postselection overhead across the four preparations. In Version 2, the spread is more pronounced: The $|+++\rangle$ preparation has the lowest acceptance, while $|00+\rangle$ has the highest acceptance over most of the plotted range. Acceptance and postselected logical fidelity therefore need not track each other monotonically: For example, in Version 2, the $|000\rangle$ preparation has a lower postselected logical error rate than $|00+\rangle$ over most of the plotted range, despite also having a lower acceptance rate. Lower acceptance therefore does not mean lower fidelity of the accepted states; rather, it means that the circuit detects and discards faulty runs more frequently.

B. Interblock logical teleportation

To benchmark entangling operations between distinct code blocks, we simulate the one-qubit teleportation circuits of Fig. 6 [66]. We consider same-version benchmarks with two Version 2 blocks and mixed-version benchmarks whose block assignments ensure that the interblock logical $\overline{\text{CNOT}}$ always has the allowed Version-2-to-Version-1 direction. In Fig. 6(a), the Version 2 source is the CNOT control and the Version 1

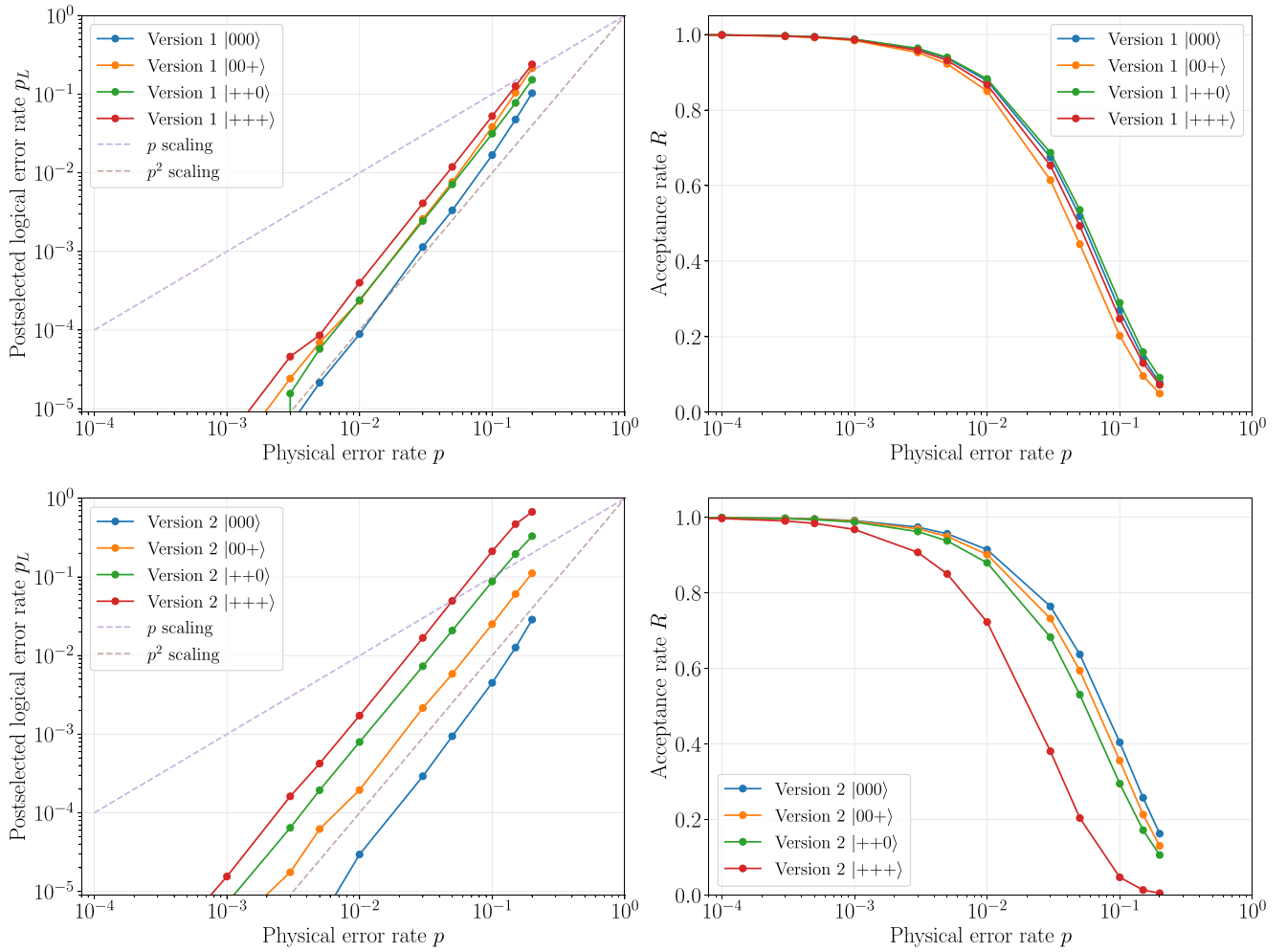


FIG. 5. Numerical simulation of fault-tolerant state preparation for the $[[8,3,2]]$ code; see Appendix B for the corresponding circuits. Top row: Version 1. Bottom row: Version 2. Left panels: postselected logical error rate p_L . Right panels: acceptance rate R . In these simulations, depolarizing noise with physical error rate p is applied to every CNOT gate. In the low-noise regime, the observed behavior is consistent with weak fault tolerance: p_L scales approximately quadratically with p , while R decreases as detectable faults are removed by postselection.

destination is the target, so the logical $|\bar{+}\rangle$ state is teleported from Version 2 to Version 1. In Fig. 6(b), the Version 2

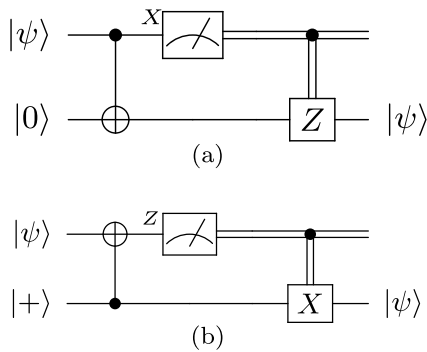


FIG. 6. One-qubit teleportation circuits used in the two-block benchmark. (a) Teleportation of $|\psi\rangle$ from the upper source qubit to the lower destination qubit initialized in $|0\rangle$. (b) Teleportation of $|\psi\rangle$ from the upper source qubit to the lower destination qubit initialized in $|+\rangle$. Double lines denote classical feedforward of the measurement outcome.

destination is the CNOT control and the Version 1 source is the target, so the logical $|\bar{0}\rangle$ state is teleported from Version 1 to Version 2 even though the logical $\overline{\text{CNOT}}$ itself still has the allowed Version-2-to-Version-1 direction. The single-pair interblock logical CNOT is implemented using the construction of Fig. 3. The encoded input states are prepared without faults, and the intrablock permutations are treated as software-level qubit relabelings. A depolarizing channel with error probability p is applied after each physical one- and two-qubit gate, while initialization, measurement, feedforward, Pauli corrections, and software-tracked permutations are treated as ideal. For each physical error rate, we ran 10^6 Monte Carlo samples.

We used the circuit in Fig. 6(a) to teleport a logical $|\bar{+}\rangle$ state from a Version 2 source block to a Version 1 destination block, and in Fig. 6(b) to teleport a logical $|\bar{0}\rangle$ state from a Version 1 source block to a Version 2 destination block. We also simulated the same-version Version-2-to-Version-2 realizations of both circuits. Here, we describe the logical $|\bar{+}\rangle$ teleportation; the other case is analogous. For each run, we prepare the logical $|++0\rangle$ state in the source block and the logical $|000\rangle$ state in the target block. Our goal is to

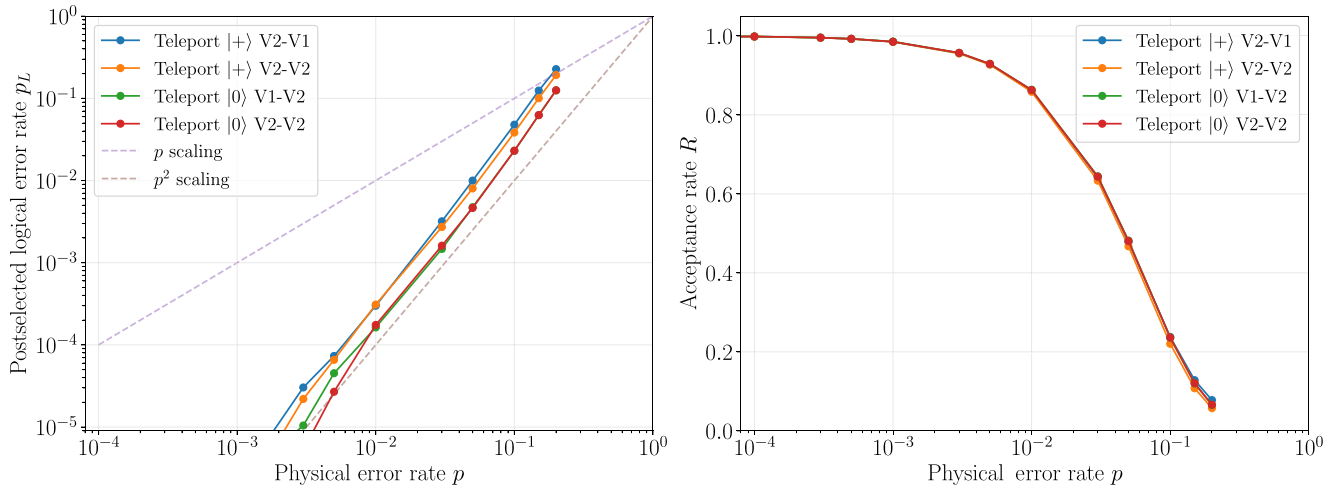


FIG. 7. Simulation of one-qubit logical teleportation between two $[[8,3,2]]$ code blocks. Same-version benchmarks use Version 2 source and destination blocks. In the mixed-version benchmarks, $|+\rangle$ is teleported from a Version 2 source to a Version 1 destination using Fig. 6(a), whereas $|0\rangle$ is teleported from a Version 1 source to a Version 2 destination using Fig. 6(b); in both cases, the interblock logical CNOT itself has the allowed Version-2-to-Version-1 direction. Left: postselected logical error rate p_L . Right: corresponding acceptance rate R . In the low-noise regime, all four benchmarks exhibit approximately quadratic scaling of p_L , while their acceptance rates are nearly identical.

teleport the first logical qubit to the second logical qubit, resulting in the logical state $|\overline{0} + \overline{0}\rangle$ in the target block. Therefore, we remove the logical $\overline{\text{SWAP}}$ gate in Fig. 3. After applying the logical $\overline{\text{cnot}}$ gate, we destructively measure each qubit of the source block in the X basis. Let $m_j \in \{-1, +1\}$ denote the X -basis outcome on physical qubit j , and first postselect on

$$\prod_{j=0}^7 m_j = +1, \quad (25)$$

which is the $X^{\otimes 8}$ stabilizer check. For Version 2, the corresponding logical X -eigenvalue outcomes are

$$\begin{aligned} b_1 &= m_0 m_1 m_2 m_3, \\ b_2 &= m_0 m_1 m_4 m_5, \\ b_3 &= m_0 m_2 m_4 m_6, \end{aligned} \quad (26)$$

which measure the Version 2 logical operators $\overline{X}_1$, $\overline{X}_2$, and $\overline{X}_3$. We use b_1 to decide whether to apply the logical $\overline{Z}_2$ gate on the target block. Finally, we postselect on the stabilizer measurement results on the target block.

An accepted run is counted as a logical failure when the decoded destination eigenvalue differs from that of the ideally teleported input state. Figure 7 shows the postselected logical error and acceptance rates p_L and R , as defined in Eqs. (23) and (24), for the same-version and mixed-version teleportation benchmarks. The two input states provide complementary state-specific tests of the teleportation channel. In the low-noise regime, all four state-and-configuration combinations exhibit approximately quadratic scaling of p_L with the physical error rate, consistent with weak fault tolerance. The logical error and acceptance curves for the mixed-version benchmarks are close to their same-version counterparts over the simulated range, indicating that the allowed mixed-version interblock operation does not introduce a substantial addi-

tional penalty. As expected for a postselected error-detecting protocol, the acceptance rate remains close to unity at low physical error rates and decreases as detectable faults become more frequent.

C. Code switching

We next benchmark the noisy switching step underlying the code-switching protocol of Sec. IV. Each shot is classified as successful, discarded, or failed. A shot is discarded when the switching procedure detects a fault, e.g., when a flag is triggered during a flagged weight-4 gauge measurement or when the output fails the target-code stabilizer or gauge checks. Among the accepted shots, the switching is counted as a failure if the final state no longer matches the target logical state. Thus, the logical error rate p_L characterizes accepted runs, while the acceptance rate R quantifies the postselection overhead.

To isolate the switching step itself, we start from a noiseless encoded source state and inject noise only during switching. In this benchmark, we do not include the complementary checks of Sec. IV; instead, after the code-switching circuit, we apply noiseless postselection on the target-code stabilizers and fixed gauge operators. In each direction, we use the representative logical state $|+++\rangle$, prepared with the corresponding noise-free encoding circuit from Appendix B. We then evaluate the output using the target-code stabilizers, fixed gauge operators, and logical operators. This benchmark therefore tests whether the switching step preserves this logical state under postselection. Because pure logical X -type faults act trivially on $|+++\rangle$, this benchmark should be interpreted as a state-specific diagnostic rather than a complete characterization of the logical channel.

In the benchmarked gauge-measurement step, the two switching directions have different overheads. As shown in Fig. 4, switching from Version 1 to Version 2 requires flagged

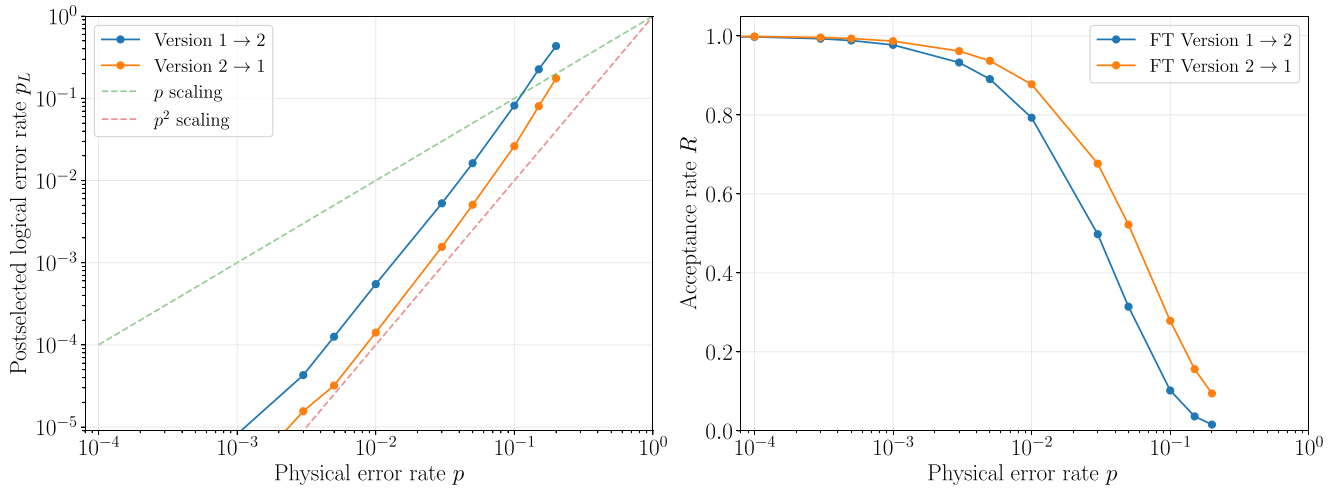


FIG. 8. Numerical simulation of the isolated switching-step benchmark. Left: postselected logical error rate for switching in both directions. Right: acceptance rates for the flagged Version 1 → 2 benchmark and the unflagged Version 2 → 1 benchmark. We apply depolarizing noise with physical error rate p after each single-qubit and two-qubit gate, and assume noise-free feedback and Pauli corrections. In this benchmark, switching from Version 1 to Version 2 uses flagged weight-4 Pauli-Z gauge measurements, whereas switching from Version 2 to Version 1 uses weight-2 Pauli-X gauge measurements without flag qubits. In the low-noise regime, the postselected logical error rates scale approximately as p^2 , consistent with weak fault tolerance.

measurements of three weight-4 Pauli-Z gauge operators, whereas switching from Version 2 to Version 1 uses only weight-2 Pauli-X gauge measurements and therefore needs no flag qubits. This asymmetry is reflected in the simulation results.

Figure 8 summarizes 10^6 Monte Carlo samples. We use depolarizing noise with physical error rate p after each single-qubit and two-qubit gate, and we again compute the postselected logical error rate p_L and the acceptance rate R using Eqs. (23) and (24). The left panel shows p_L for switching in both directions. In both cases, the postselected logical error rate scales approximately quadratically with p in the low-noise regime, consistent with weak fault tolerance. Moreover, the Version 2 → 1 curve lies below the Version 1 → 2 curve across the plotted range, as expected from the simpler unflagged weight-2 measurements used in the reverse direction.

The right panel of Fig. 8 shows the acceptance rates for the flagged Version 1 → 2 protocol and the unflagged Version 2 → 1 protocol. At the lowest plotted physical error rates, both acceptance rates are high, with the Version 2 → 1 curve closer to unity than the Version 1 → 2 curve, and both decrease monotonically as detectable single faults are removed by postselection. This first-order reduction in R is the expected cost of an error-detecting protocol, which discards a fraction of shots already at $O(p)$. The two acceptance curves therefore quantify the postselection overhead of the isolated switching-step benchmarks, after immediate postselection on the target-code fixed gauge operators and stabilizers. The Version 2 → 1 curve lies above the Version 1 → 2 curve across the plotted range, consistent with the simpler unflagged weight-2 measurements used in that direction. In the full switching protocol of Sec. IV, we additionally include the complementary checks to ensure that the target code is reached in the correct gauge and stabilizer sector before subsequent logical gates are applied.

VII. GROVER'S ALGORITHM WITH CODE SWITCHING

In this section, we use a three-logical-qubit instance of Grover's algorithm [44,45] as an algorithmic benchmark for our code-switching scheme. Grover search has also been employed as a small-scale benchmark for computation with error detection using the $[[n, n-2, 2]]$ and $[[8, 3, 2]]$ codes [67–69]. The algorithm searches for a set of marked strings $M \subseteq \{0, 1\}^n$ in an unstructured database of size $N = 2^n$, where the oracle operator O is defined by

$$O|x\rangle = \begin{cases} -|x\rangle, & x \in M, \\ |x\rangle, & \text{otherwise,} \end{cases} \quad (27)$$

for any n -bit string x . A classical algorithm requires $O(N/|M|)$ oracle queries, whereas Grover's algorithm achieves a quadratic speedup using $O(\sqrt{N/|M|})$ oracle queries [70,71].

Figure 9 shows the case $M = \{101, 011\}$ for $n = 3$. In the ideal noiseless circuit, one Grover iteration maps the initial

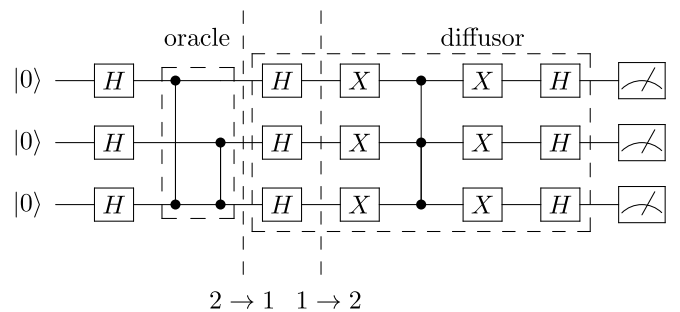


FIG. 9. Logical Grover circuit for the two marked strings 011 and 101 among eight candidates. The vertical dashed lines indicate the two code-switching locations used in the encoded implementation. The initial $\bar{H}^{\otimes 3}$ layer is absorbed into preparation of $|+++\rangle$ in Version 2, and the final $\bar{H}^{\otimes 3}$ layer is absorbed into destructive X-basis readout.

ALGORITHM 3. Fault-tolerant implementation of the encoded three-logical-qubit Grover circuit.

Preparation: Prepare $|++\rangle$ in Version 2 using the fault-tolerant preparation circuit of Fig. 15(d), including the flagged G_5^Z and G_6^Z checks.

Oracle: Apply $\overline{\text{CZ}}(1, 3)\overline{\text{CZ}}(2, 3)$ using Eq. (16) in Version 2.

Code switch from Version 2 to Version 1: Apply Algorithm 2.

Logical Hadamards: Apply $\overline{H}_1\overline{H}_2\overline{H}_3$ in Version 1 with Eq. (13) using the weakly fault-tolerant gadgets of Fig. 11.

Code switch from Version 1 to Version 2: Apply Algorithm 1.

Diffusor: Apply $\overline{X}_1\overline{X}_2\overline{X}_3$, then the transversal logical $\overline{\text{CCZ}}$ using Eq. (15) in Version 2, and then another round of $\overline{X}_1\overline{X}_2\overline{X}_3$.

Readout: Measure all data qubits in the physical X basis, obtaining $m_0, \dots, m_7 \in \{\pm 1\}$. Postselect on $\prod_{j=0}^7 m_j = +1$ and decode using Eq. (26).

uniform superposition to $(|101\rangle + |011\rangle)/\sqrt{2}$, so no other output bit string occurs upon measurement. In the simulations below, any other output bit string is counted as an output failure.

Grover's algorithm is a useful small-scale demonstration of our protocol because its logical circuit uses both a logical $\overline{H}$ gate and a logical $\overline{\text{CCZ}}$ gate (see Fig. 9). Within the fault-tolerant gate sets available in a single version of the $[[8,3,2]]$ code considered here, implementing both requires switching between the two versions. The corresponding encoded implementation is summarized in Algorithm 3. The algorithm is stated in terms of the fault-tolerant subroutines introduced in the preceding sections, while their constituent physical circuits are not repeated here.

To evaluate the algorithm's performance with code switching, we simulated two cases using Qiskit AerSimulator [72]: the unencoded three-qubit circuit and an encoded implementation that includes the fault-tolerant preparation circuit for the Version 2 state $|++\rangle$ together with the flagged G_5^Z and G_6^Z checks. For this benchmark, we use the same depolarizing noise model as in Sec. VI. Measurement, feedforward, and gauge corrections are treated as ideal. For each physical error rate between 10^{-4} and 10^{-2} , we sampled 10^6 shots.

In the encoded simulations, the final logical $\overline{H}^{\otimes 3}$ layer is absorbed into destructive X -basis measurements of all data qubits. We use Eqs. (25) and (26) to similarly postselect and decode the measurement results. Equivalently, the decoded logical bits are $s_i \in \{0, 1\}$. A shot is counted as successful if the decoded bit string $s_1s_2s_3$ is 101 or 011. The unencoded benchmark is evaluated analogously after absorbing the final Hadamards into X -basis readout. Postselection is applied to all flag, complementary check, and stabilizer measurements, as well as to the ancilla checks in the weakly fault-tolerant $\overline{X}\overline{X}$ and $\overline{Z}\overline{Z}$ gadgets used to implement the logical Hadamards. It is also applied to the auxiliary measurements immediately after the state-preparation circuit in Fig. 15(d) together with the additional flagged G_5^Z and G_6^Z checks.

In these simulations, error detection is performed at the boundaries already required by the fault-tolerant subroutines, including state preparation, the ancilla-assisted logical-Hadamard gadgets, and code switching. We do not

insert additional full QED cycles elsewhere to retain the low resource overhead of the present demonstration. This choice is appropriate for the relatively shallow Grover circuit considered here, for which the simulations already exhibit quadratic suppression of the accepted-run output failure probability. In deeper circuits, however, delaying error detection allows multiple individually detectable errors to persist and potentially combine into an undetectable logical error. Additional intermediate QED cycles could reduce this accumulation, at the cost of extra ancillas, entangling gates, and measurements, as well as additional postselection losses. Optimizing the placement and frequency of such cycles therefore involves a trade-off between accepted-run logical fidelity and total resource and sampling overhead, which we leave for future work.

Accordingly, the left panel of Fig. 10 reports the output failure probability of the unencoded circuit and the postselected decoded-output failure probability of the encoded implementation. The unencoded circuit exhibits approximately linear scaling with the physical error rate, while the encoded implementation exhibits approximately quadratic scaling in the low-noise regime. The encoded result includes the noisy fault-tolerant preparation of the Version 2 input state as well as the Grover and code-switching steps.

The right panel of Fig. 10 shows the acceptance rate for the encoded implementation. It remains close to unity at low physical error rate and then decreases monotonically as detectable faults are removed by postselection. This first-order acceptance penalty is the expected cost of the postselected subroutines, including state preparation, the weakly fault-tolerant $\overline{X}\overline{X}$ and $\overline{Z}\overline{Z}$ gadgets used to implement the logical Hadamards, and code switching: The accepted-run output failure is suppressed to second order, but only a fraction R of shots are retained.

VIII. CONCLUSION

In this paper, we developed a fault-tolerant, error-detecting protocol for switching between two versions of the $[[8,3,2]]$ code and used it to assemble a universal logical gate set. Numerical simulations of state preparation and code switching show quadratic suppression of postselected logical error rates, and the three-logical-qubit Grover benchmark shows quadratic suppression of the postselected decoded-output failure probability, consistent with weak fault tolerance. While the present shallow benchmark uses only the error-detection steps associated with its constituent fault-tolerant subroutines, deeper computations may benefit from additional intermediate QED cycles, whose placement must balance improved error suppression against increased circuit and postselection overhead.

We showed that both versions arise from the same $[[8,3,3,2]]$ subsystem code, obtained by gauging out three logical qubits of the $[[8,6,2]]$ code after a noncanonical choice of logical basis. Fixing the gauge qubits in $|0_g\rangle$ yields Version 2, namely, the $[[8,3,2]]$ code with logical $\overline{\text{CCZ}}$, $\overline{\text{CZ}}$, $\overline{\text{CNOT}}$, and $\overline{\text{SWAP}}$ gates. Fixing them in $|+_g\rangle$ yields Version 1, in which the resulting weight-2 representatives of the logical operators enable weakly fault-tolerant implementations of $\overline{H}$, $\overline{S}$, and

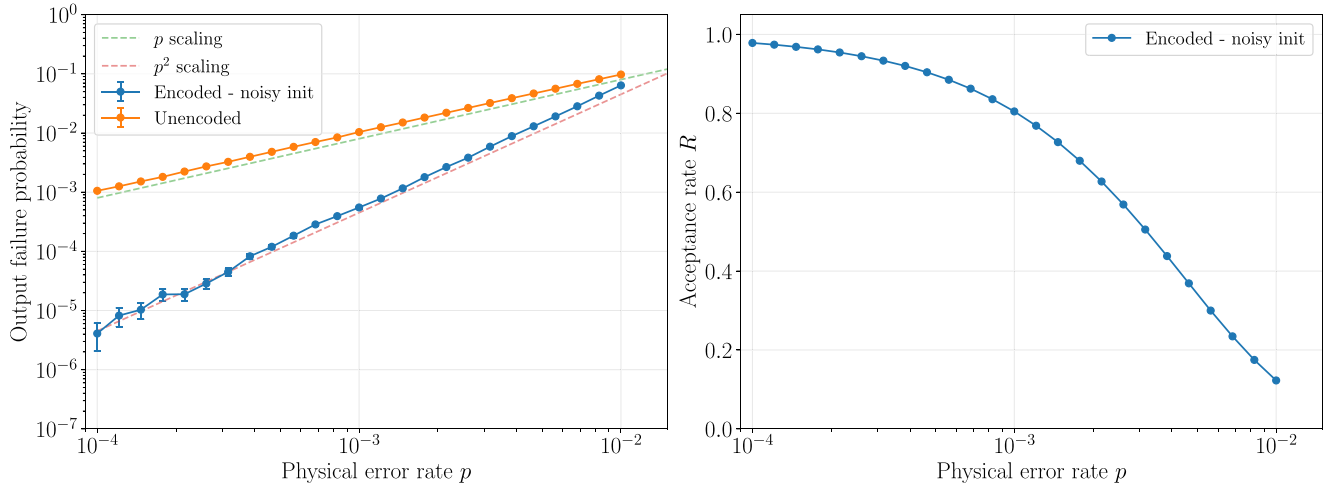


FIG. 10. Simulation results for Grover's algorithm with physical error rate p . Left: output failure probability for the unencoded three-qubit circuit and postselected decoded-output failure probability for the encoded implementation, which includes fault-tolerant preparation of the Version 2 input state $|\overline{++}\overline{++}\rangle$ together with the additional flagged G_5^Z and G_6^Z checks. Right: acceptance rate for the encoded implementation. In the low-noise regime, the unencoded circuit exhibits approximately linear scaling with p , whereas the encoded implementation exhibits approximately quadratic scaling.

$\sqrt{X}^\dagger$ using the $\widetilde{XX}$ and $\widetilde{ZZ}$ rotation gadgets. We also showed that single interblock logical CNOT gates are available between same-version blocks and, for mixed-version blocks, only in the direction $[[8, 3, 2]]_2 \rightarrow [[8, 3, 2]]_1$.

Our construction suggests a broader family viewpoint. The code $[[8, 3, 2]]_2$ belongs to the $[[2^D, D, 2]]$ hypercube family [38, 73], with $D = 3$, while the parent $[[8, 6, 2]]$ code belongs to the $[[n, n - 2, 2]]$ family. Our use of a noncanonical logical basis raises the question of how to choose logical bases systematically for $[[2^D, 2^D - 2, 2]]$ codes so that $2^D - 2 - D$ logical qubits may be gauged out while preserving useful transversal structure. Fixing those gauge qubits in $|0_g\rangle$ recovers the D -dimensional hypercube code, while fixing them in $|+_g\rangle$ suggests a companion family generalizing our Version 1 construction. More generally, D -dimensional hypercube codes support transversal multi-controlled-phase gates of increasing order [40, 74], providing depth-1 access to higher levels of the Clifford hierarchy. It would therefore be interesting to systematically characterize the corresponding Version 1 family and to determine its logical Clifford gate set and its fault-tolerant implementation.

A broader direction is to investigate whether the present approach to fault-tolerant universality can be extended to higher-distance code families. Conventional code-switching constructions, such as those based on the $[[7, 1, 3]]$ and $[[15, 1, 3]]$ codes, obtain complementary logical gate sets primarily from transversal operations in the participating codes [35, 37]. In contrast, our construction combines transversal gates in one code version with explicitly fault-tolerant, but nontransversal, logical-gate gadgets in the other. Allowing this mixture of transversal operations and suitably designed fault-tolerant gadgets could substantially enlarge the class of codes that support universal computation through code switching, since neither code would be required to supply its entire complementary gate set transversally.

Concatenating the present blocks with an outer error-correcting code is another possible route toward a larger construction. Such a scheme would require the inner code-switching and logical-gate gadgets to preserve the effective distance of the concatenated code.

The central requirement for any such generalization is that the fault-tolerant gadgets scale together with the distance of the codes. In particular, the gadgets must not reduce the effective protection below that provided by the code blocks themselves. Otherwise, they would become the dominant bottleneck of the full computation. For a distance- d error-correcting construction, this means that the gadgets should be able to correct the same errors that are correctable by the code, so that the leading logical failure probability remains consistent with the code distance. In a postselected error-detecting construction, faults below the relevant distance should likewise be either benign or detected before acceptance. The present distance-2 construction satisfies this matching condition: The codes detect any single-qubit Pauli error, and the weakly fault-tolerant gadgets are designed so that any such error is either harmless or rejected. Together, they yield first-order suppression of the logical errors. Extending this compatibility between code distance and gadget fault tolerance is therefore a necessary first step toward higher-distance versions of the protocol.

ACKNOWLEDGMENTS

The authors acknowledge Christopher Gerhard, Anirudh Lanka, and Juan Garcia Nila for useful discussions. This material is based upon work supported by, or in part by, the U.S. Army Research Laboratory and the U.S. Army Research Office under contract/grant number W911NF2310255 and by NSF Grant FET-2316713.

DATA AVAILABILITY

The data that support the findings of this article are openly available in Ref. [75].

APPENDIX A: FAULT-TOLERANT IMPLEMENTATIONS OF TWO-QUBIT ROTATION GATES

Figure 11 is the fault-tolerant implementation of the $\widetilde{XX}$ and $\widetilde{ZZ}$ gates taken from Ref. [26]. Both implementations have two versions. In the first version, ancillas are prepared in a Bell state and end in a $|++\rangle$ state, while in the second version, ancillas are prepared in the $|++\rangle$ state and end in a Bell state. Hence, alternating between two versions avoids the need to reprepare the ancillas. After each circuit, Pauli operations are applied according to Table II.

The logical Hadamard gate can be implemented by using the decomposition in Eq. (13). To avoid repreparing the ancillas between successive two-qubit-rotation gadgets, one may choose the gadget versions so that the ancilla output of one gadget is the required ancilla input of the next: use

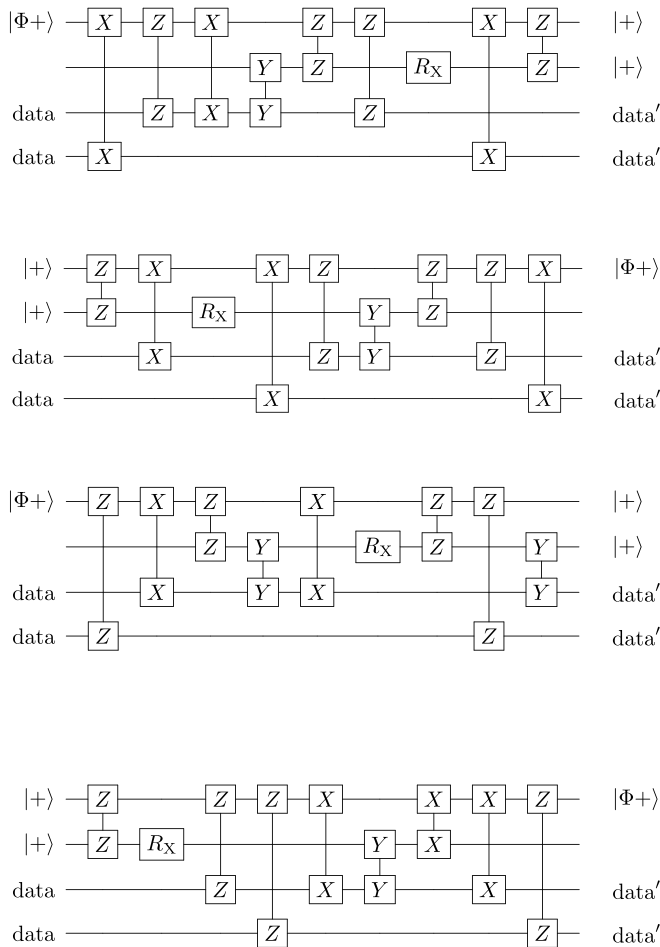


FIG. 11. Weakly fault-tolerant gadgets for the XX (top two panels) and ZZ rotations (bottom two panels). The top and third panels consume a Bell pair and return the ancillas in $|++\rangle$; the second and bottom panels consume $|++\rangle$ and return a Bell pair. Alternating the two versions allows successive XX and ZZ gadgets to reuse ancillas without an explicit repreparation step.

TABLE II. Pauli recovery operations to apply after each weakly fault-tolerant $\widetilde{XX}$ or $\widetilde{ZZ}$ gate. The row is determined by the initial state of the ancilla qubits, and the column by the rotation gate being implemented.

	$\widetilde{ZZ}$ rotation	$\widetilde{XX}$ rotation
$ \Phi_+\rangle$	$ZIZI$	$IZYI$
$ ++\rangle$	$XIXI$	$YIYI$

a Bell-to- $|++\rangle$ version of $\widetilde{ZZ}$, followed by a $|++\rangle$ -to-Bell version of $\widetilde{XX}$, followed by a Bell-to- $|++\rangle$ version of $\widetilde{ZZ}$.

To characterize the encoded Hadamard gate, we performed Monte Carlo sampling with two-qubit depolarizing noise applied to every physical two-qubit gate in the circuit; measurements were assumed ideal. For each sampled fault pattern, the induced Pauli error is propagated through the Clifford circuit, the run is postselected on the trivial stabilizer syndrome and on the ancillas returning to the expected final state, and the resulting logical action is compared with the ideal encoded Hadamard on the chosen logical qubit. One could also choose the opposite versions of the $\widetilde{XX}$ and $\widetilde{ZZ}$ gadgets, but then the final ancillas would need to be checked in the Bell basis rather than in the X basis.

The resulting postselected logical error rates and postselection success rates for the three logical qubits are shown in Fig. 12. In the low-noise regime, all three implementations exhibit approximately quadratic suppression of the logical error rate with the physical two-qubit error rate, consistent with weak fault tolerance.

APPENDIX B: FAULT-TOLERANT ENCODING CIRCUITS FOR BOTH VERSIONS OF THE $[[8,3,2]]$ CODE

In this Appendix, we collect the explicit fault-tolerant state-preparation circuits used throughout the paper. We consider the logical-basis states $|000\rangle$ and $|+++\rangle$ in both versions of the $[[8,3,2]]$ code, as well as mixed logical-basis states. For the latter, we use the shorthand $|+^2 0^1\rangle$ and $|+^1 0^2\rangle$ to indicate only the numbers of logical $|+\rangle$ and $|0\rangle$ states, not their ordering. Without additional gauge fixing, Version 1 naturally supports states of the form $|+^2 0^1\rangle$, while Version 2 naturally supports states of the form $|+^1 0^2\rangle$. The opposite pairing is obtained more efficiently by preparing a partially gauge-fixed intermediate state and then performing only the final gauge-fixing step, rather than a full code switch.

1. GHZ and dual-GHZ resource states

Most of the encoding circuits below reduce, after Gaussian elimination of the stabilizer generators, to standard GHZ states, dual GHZ states, Bell pairs, or tensor products thereof. An n -qubit GHZ state [76] in the computational basis is

$$|\text{GHZ}_n\rangle = \frac{1}{\sqrt{2}}(|0\rangle^{\otimes n} + |1\rangle^{\otimes n}), \quad (\text{B1})$$

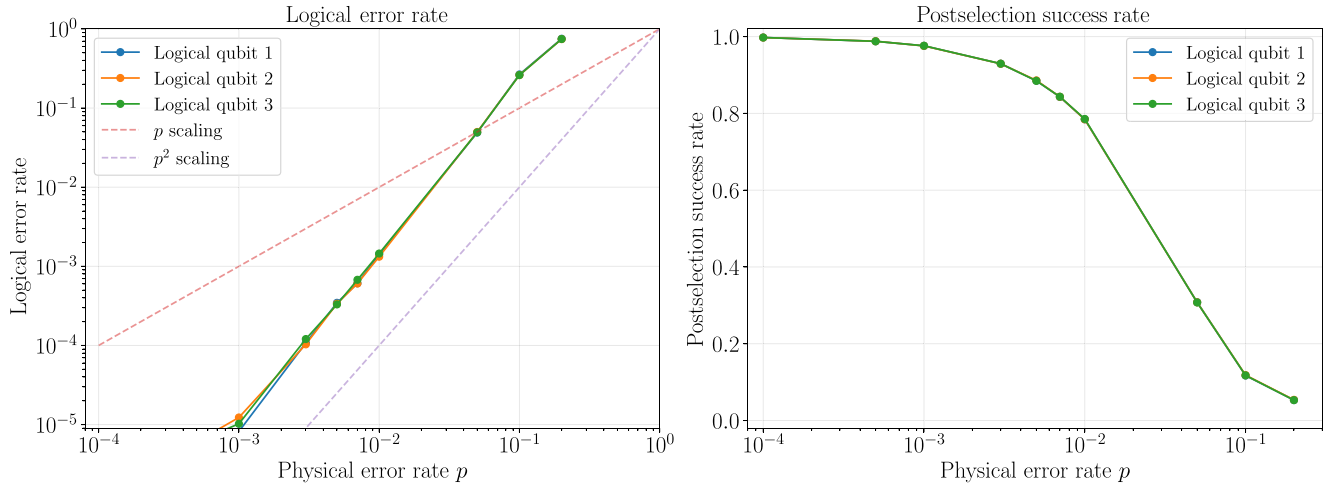


FIG. 12. Simulation results for the encoded Hadamard gate on the three logical qubits. In the low-noise regime, the postselected logical error rate is approximately quadratic in the physical two-qubit error rate, consistent with weak fault tolerance. The right panel shows the corresponding postselection success rate.

with $+1$ eigenoperators $X^{\otimes n}$ and $Z_0 Z_j$ for $j = 1, \dots, n-1$. The corresponding dual GHZ state in the X basis is

$$|\text{Dual GHZ}_n\rangle = \frac{1}{\sqrt{2}}(|+\rangle^{\otimes n} + |-\rangle^{\otimes n}), \quad (\text{B2})$$

with $+1$ eigenoperators $Z^{\otimes n}$ and $X_0 X_j$ for $j = 1, \dots, n-1$.

The fault-tolerant preparation circuit for $|\text{GHZ}_n\rangle$ is shown in Fig. 13. The dual GHZ state is prepared by the same construction after interchanging the X and Z bases, i.e., by replacing $|0\rangle \leftrightarrow |+\rangle$, reversing the CNOT directions, and measuring in the X basis instead of the Z basis.

2. Version 1

The Version 1 preparation circuits are collected in Fig. 14.

Logical $|\overline{000}\rangle$ [Fig. 14(a)]. This preparation fixes all three logical qubits of the parent $[[8,3,3,2]]$ subsystem code to $|\overline{0}\rangle$ and all three gauge qubits to $|+_g\rangle$. Since all relevant logical and gauge operators are weight 2, we can reuse the fault-tolerant preparation of logical $|+^j 0^{n-j-2}\rangle$ states in the $[[n, n-2, 2]]$ code from Ref. [64], after changing basis. In particular, the required state is equivalent to the logical $|\overline{+^3 0^3}\rangle$ state of the $[[8,6,2]]$ stabilizer code with logical operator basis

$$X_k X_7, \quad Z_0 Z_k, \quad k = 1, \dots, 6. \quad (\text{B3})$$

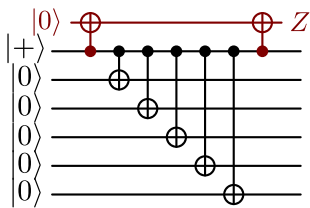


FIG. 13. Fault-tolerant GHZ state encoding circuit [64]. The black part is the encoding circuit, and the red part checks for error propagation.

Because $j = 3$ is odd, this state factorizes into a four-qubit dual GHZ state and a four-qubit GHZ state:

$$\begin{aligned} |\overline{+^3 0^3}\rangle &= \frac{1}{\sqrt{2}}(|+3 +5 +6 +7\rangle + |-3 -5 -6 -7\rangle) \\ &\otimes \frac{1}{\sqrt{2}}(|0_0 0_1 0_2 0_4\rangle + |1_0 1_1 1_2 1_4\rangle). \end{aligned} \quad (\text{B4})$$

This gives the Version 1 logical $|\overline{000}\rangle$ state.

Logical $|\overline{+++}\rangle$ [Fig. 14(d)]. Here, all three logical qubits and all three gauge qubits are fixed to $|\overline{+}\rangle$ and $|+_g\rangle$, respectively. The $+1$ eigenoperators reduce as

$$\begin{aligned} \overline{X}_1 &= X_0 X_1 X_2 X_3 \rightarrow X_6 X_4, \\ \overline{X}_2 &= X_0 X_1 X_4 X_5 \rightarrow X_6 X_2, \\ \overline{X}_3 &= X_0 X_2 X_4 X_6 \rightarrow X_6 X_1, \\ G_4^X &= X_7 X_3 \rightarrow X_3 X_6, \\ G_5^X &= X_7 X_5 \rightarrow X_5 X_6, \\ G_6^X &= X_7 X_6 \rightarrow X_7 X_6, \\ S^X &= X^{\otimes 8} \rightarrow X_0 X_6, \\ S^Z &= Z^{\otimes 8} \rightarrow Z^{\otimes 8}. \end{aligned} \quad (\text{B5})$$

These reduced generators are those of the eight-qubit dual GHZ state, so the preparation reduces directly to the dual-GHZ gadget.

Logical $|\overline{+++0}\rangle$ [Fig. 14(c)]. This is the native mixed-basis state in Version 1. Using $|\overline{+++0}\rangle$ as a representative, the Gaussian-eliminated generators are identical to those of $|\overline{+++}\rangle$ except that $X^{\otimes 8}$ is replaced by $X_0 X_1$. Together with the additional operator $Z_0 Z_1$, this isolates a Bell pair on qubits 0 and 1 and a six-qubit dual GHZ state on the remaining qubits:

$$\begin{aligned} |\overline{+++0}\rangle &= \frac{1}{\sqrt{2}}(|+3 +5 +6 +7 +2 +4\rangle + |-3 -5 -6 -7 -2 -4\rangle) \\ &\otimes \frac{1}{\sqrt{2}}(|0_0 0_1\rangle + |1_0 1_1\rangle). \end{aligned} \quad (\text{B6})$$

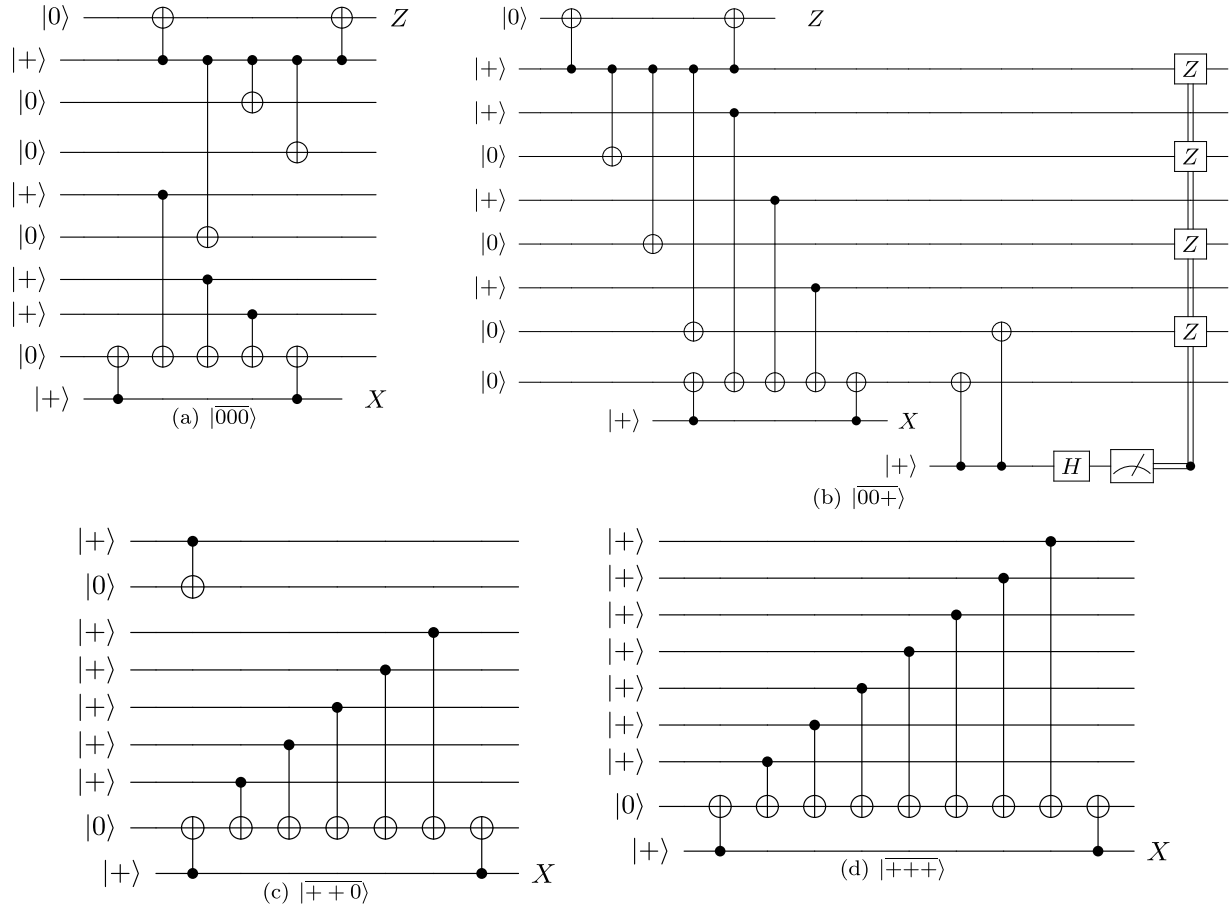


FIG. 14. Fault-tolerant state-preparation circuits for the Version 1 $[[8,3,2]]$ code: (a) $|000\rangle$, (b) $|00+\rangle$, (c) $|++0\rangle$, and (d) $|+++ \rangle$. Terminal X and Z labels on ancilla lines denote measurements in the corresponding basis. Boxes labeled X or Z connected by classical double lines denote feedforward Pauli corrections conditioned on the measured ancilla outcome.

The Bell state is prepared fault tolerantly without ancillas, and the six-qubit dual GHZ state is prepared fault tolerantly with one ancilla. Other orderings of $|+^{201}\rangle$ are obtained analogously.

Logical $|00+\rangle$ [Fig. 14(b)]. This is the non-native mixed-basis state in Version 1, so we prepare it through a partially gauge-fixed intermediate state. We first fix G_4^X , G_5^X , and G_6^Z . For the representative state $|00+\rangle$, the $+1$ eigenoperators reduce to

$$\begin{aligned}
 \bar{Z}_1 &= Z_0 Z_4 \rightarrow Z_0 Z_4, \\
 \bar{Z}_2 &= Z_0 Z_2 \rightarrow Z_0 Z_2, \\
 \bar{X}_3 &= X_0 X_2 X_4 X_6 \rightarrow X_1 X_7, \\
 G_4^X &= X_7 X_3 \rightarrow X_7 X_3, \\
 G_5^X &= X_7 X_5 \rightarrow X_7 X_5, \\
 G_6^Z &= Z_0 Z_2 Z_4 Z_6 \rightarrow Z_0 Z_6, \\
 S^X &= X^{\otimes 8} \rightarrow X_0 X_2 X_4 X_6, \\
 S^Z &= Z^{\otimes 8} \rightarrow Z_1 Z_3 Z_5 Z_7.
 \end{aligned} \tag{B7}$$

These reduced generators describe a four-qubit GHZ state on the even qubits $\{0, 2, 4, 6\}$ and a four-qubit dual GHZ state on

the odd qubits $\{1, 3, 5, 7\}$. After preparing this intermediate state, we measure the single remaining gauge operator $G_6^X = X_7 X_6$ together with its complementary check $X_0 X_1 X_2 X_3 X_4 X_5$, whose product gives the precorrection $X^{\otimes 8}$ syndrome. Because only one gauge operator remains to be measured, there are no later gauge measurements whose outcomes can be corrupted by an intermediate fault, so an additional dedicated $X^{\otimes 8}$ stabilizer measurement is not needed inside this subroutine. Any residual data error is handled by the stabilizer and gauge checks immediately after state preparation.

3. Version 2

The Version 2 preparation circuits are collected in Fig. 15. *Logical $|000\rangle$ [Fig. 15(a)].* This preparation fixes all three logical qubits and all three gauge qubits of the parent $[[8,3,3,2]]$ subsystem code to $|0\rangle$ and $|0_g\rangle$, respectively. The $+1$ eigenoperators reduce as

$$\begin{aligned}
 \bar{Z}_1 &= Z_0 Z_4 \rightarrow Z_0 Z_4, \\
 \bar{Z}_2 &= Z_0 Z_2 \rightarrow Z_0 Z_2, \\
 \bar{Z}_3 &= Z_0 Z_1 \rightarrow Z_0 Z_1, \\
 G_4^Z &= Z_0 Z_1 Z_2 Z_3 \rightarrow Z_0 Z_3,
 \end{aligned}$$

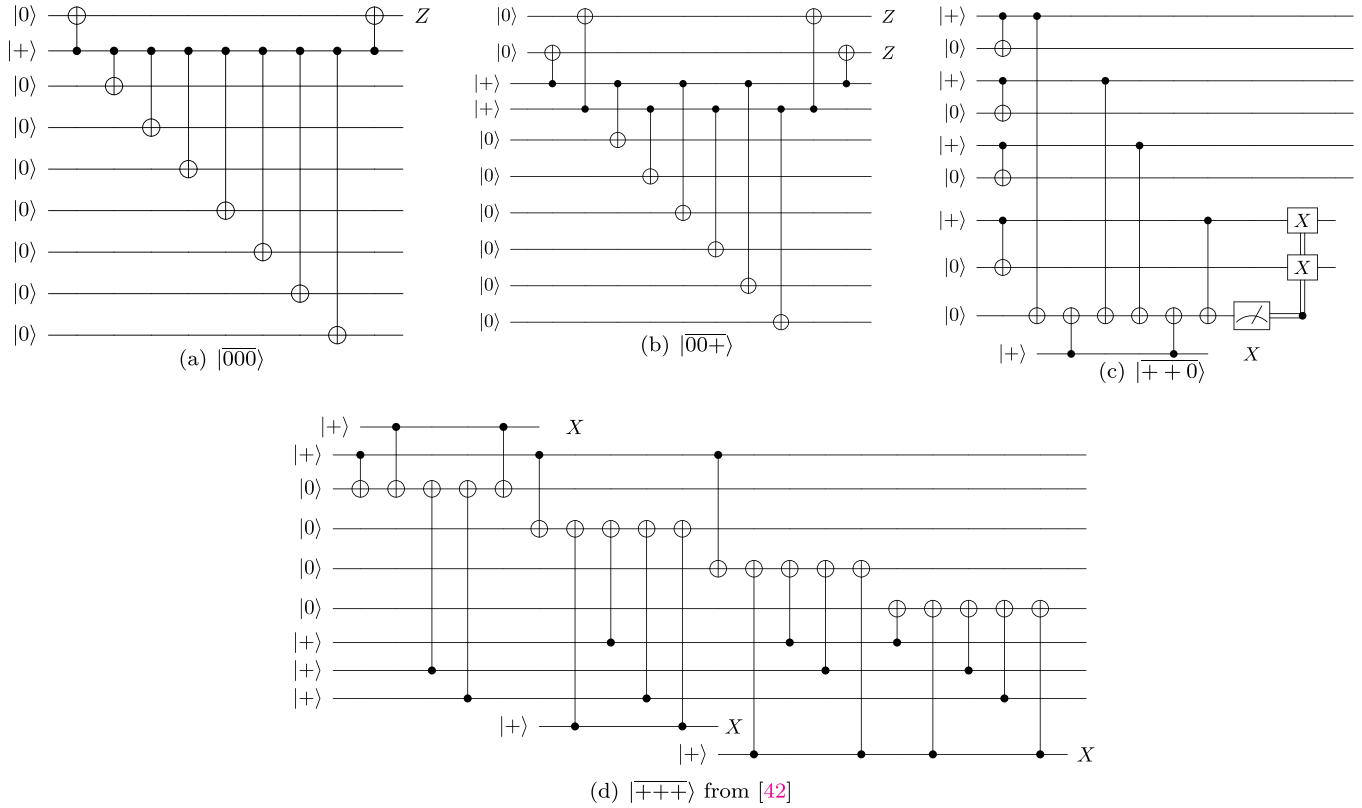


FIG. 15. Fault-tolerant state-preparation circuits for the Version 2 $[[8,3,2]]$ code: (a) $|\overline{000}\rangle$, (b) $|\overline{00+}\rangle$, (c) $|\overline{++0}\rangle$, and (d) $|\overline{+++}\rangle$ (from Ref. [42]). Terminal X and Z labels on ancilla lines denote measurements in the corresponding basis. Boxes labeled X or Z connected by classical double lines denote feedforward Pauli corrections conditioned on the measured ancilla outcome.

$$\begin{aligned}
 G_5^Z &= Z_0 Z_1 Z_4 Z_5 \rightarrow Z_0 Z_5, \\
 G_6^Z &= Z_0 Z_2 Z_4 Z_6 \rightarrow Z_0 Z_6, \\
 S^X &= X^{\otimes 8} \rightarrow X^{\otimes 8}, \\
 S^Z &= Z^{\otimes 8} \rightarrow Z_0 Z_7.
 \end{aligned} \tag{B8}$$

These are the generators of the eight-qubit GHZ state, so the logical $|\overline{000}\rangle$ preparation reduces to the GHZ gadget.

Logical $|\overline{+++}\rangle$ [Fig. 15(d)]. For Version 2, we use the fault-tolerant preparation circuit for $|\overline{+++}\rangle$ introduced in Ref. [42].

Logical $|\overline{00+}\rangle$ [Fig. 15(b)]. This is the native mixed-basis state in Version 2. Using $|\overline{00+}\rangle$ as a representative, the $+1$ eigenoperators reduce to

$$\begin{aligned}
 \bar{Z}_1 &= Z_0 Z_4 \rightarrow Z_0 Z_4, \\
 \bar{Z}_2 &= Z_0 Z_2 \rightarrow Z_0 Z_2, \\
 \bar{X}_3 &= X_0 X_2 X_4 X_6 \rightarrow X_0 X_2 X_4 X_6, \\
 G_4^Z &= Z_0 Z_1 Z_2 Z_3 \rightarrow Z_1 Z_3, \\
 G_5^Z &= Z_0 Z_1 Z_4 Z_5 \rightarrow Z_1 Z_5, \\
 G_6^Z &= Z_0 Z_2 Z_4 Z_6 \rightarrow Z_0 Z_6, \\
 S^X &= X^{\otimes 8} \rightarrow X_1 X_3 X_5 X_7, \\
 S^Z &= Z^{\otimes 8} \rightarrow Z_1 Z_7.
 \end{aligned} \tag{B9}$$

These reduced generators describe two independent four-qubit GHZ states, one on the even qubits $\{0, 2, 4, 6\}$ and one on the odd qubits $\{1, 3, 5, 7\}$. The logical state is therefore prepared by fault tolerantly preparing those two GHZ states.

Logical $|\overline{++0}\rangle$ [Fig. 15(c)]. This is the non-native mixed-basis state in Version 2, so again we use a partially gauge-fixed intermediate state. We first fix G_4^Z , G_5^Z , and G_6^X . For the representative state $|\overline{++0}\rangle$, the reduced $+1$ eigenoperators are

$$\begin{aligned}
 \bar{X}_1 &= X_0 X_1 X_2 X_3 \rightarrow X_4 X_5, \\
 \bar{X}_2 &= X_0 X_1 X_4 X_5 \rightarrow X_0 X_1, \\
 \bar{Z}_3 &= Z_0 Z_1 \rightarrow Z_0 Z_1, \\
 G_4^Z &= Z_0 Z_1 Z_2 Z_3 \rightarrow Z_2 Z_3, \\
 G_5^Z &= Z_0 Z_1 Z_4 Z_5 \rightarrow Z_4 Z_5, \\
 G_6^X &= X_7 X_6 \rightarrow X_7 X_6, \\
 S^X &= X^{\otimes 8} \rightarrow X_2 X_3, \\
 S^Z &= Z^{\otimes 8} \rightarrow Z_7 Z_6.
 \end{aligned} \tag{B10}$$

These generators define four Bell states on the qubit pairs $\{0, 1\}$, $\{2, 3\}$, $\{4, 5\}$, and $\{6, 7\}$, so the intermediate state is prepared fault tolerantly without ancillas. We then measure the single remaining gauge operator $G_6^Z = Z_0 Z_2 Z_4 Z_6$ together with its complementary check $Z_1 Z_3 Z_5 Z_7$, whose product gives the precorrection $Z^{\otimes 8}$ syndrome. As in the Version 1

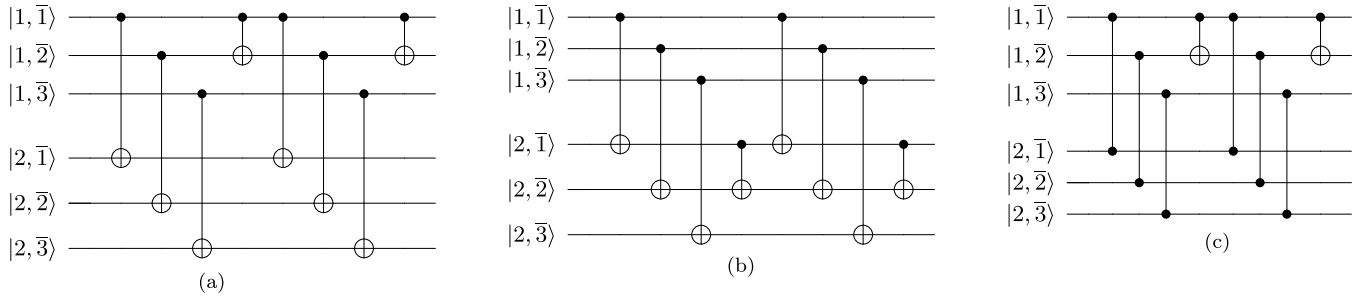


FIG. 16. Interblock logical gates between blocks 1 and 2. (a) Smallest interblock logical $\overline{\text{CNOT}}$, (b) Alternative interblock logical $\overline{\text{CNOT}}$, and (c) Interblock logical $\overline{\text{CZ}}$.

construction above, no additional dedicated $Z^{\otimes 8}$ stabilizer measurement is required inside this single-step gauge-fixing subroutine, because there are no later gauge measurements whose outcomes can be corrupted by an intermediate fault. Any residual data error is handled by the stabilizer and gauge checks applied immediately after state preparation.

4. Remarks on circuit simplifications

Single-qubit errors may propagate to higher-weight errors through an encoding circuit. If these propagated errors can be detected by measuring the ancillas and stabilizers in the following subroutines, the encoding circuit is fault tolerant. Figure 5 has shown that the encoding circuits in this section are fault tolerant when the corresponding stabilizers and gauge operators are measured immediately after the encoding circuits.

Measuring the $X^{\otimes 8}$ and $Z^{\otimes 8}$ stabilizers is sufficient to verify that the state lies in the stabilizer codespace. However, as discussed in Sec. IV, this is not always sufficient to guarantee the correct gauge sector for subsequent logical operations, because incorrectly applied gauge operators can act as logical faults. Some encoding circuits therefore do not require any gauge measurement, whereas others do. We discuss the different scenarios below. Note that fault-tolerant measurement of weight-4 gauge operators requires flag qubits [64], as shown in Fig. 4.

All the native encoding circuits with GHZ states. They do not need to measure any gauge operator immediately after, because all the propagated errors act as stabilizers.

All the non-native encoding circuits. A single Pauli fault on the gauge-measurement ancilla can flip the reported gauge outcome and thereby cause an incorrect gauge correction. For this reason, the remaining gauge operator is measured together with the complementary check described above. Their product gives the precorrection $X^{\otimes 8}$ or $Z^{\otimes 8}$ syndrome, so a single outcome-flipping fault is rejected before any gauge correction is applied. The full stabilizer and gauge checks immediately following state preparation detect residual data errors.

The logical $|+++\rangle$ state in Version 2. In this fault-tolerant preparation circuit, the propagated errors X_0X_3 , X_4X_5 , X_4X_6 ,

and X_4X_7 are not detectable by measuring only the $Z^{\otimes 8}$ stabilizer. Moreover,

$$\begin{aligned} X_0X_3 &= \overline{X}_1\overline{X}_2\overline{X}_3 G_5^X G_6^X, \\ X_4X_5 &= X^{\otimes 8}\overline{X}_1 G_6^X, \\ X_4X_6 &= X^{\otimes 8}\overline{X}_1 G_5^X, \\ X_4X_7 &= X^{\otimes 8}\overline{X}_1 G_5^X G_6^X. \end{aligned} \quad (\text{B11})$$

Therefore, on the logical state $|+++\rangle$, for which $\overline{X}_1 = \overline{X}_2 = \overline{X}_3 = +1$ and $X^{\otimes 8} = +1$, these propagated errors are equivalent to the gauge operators G_5^X , G_6^X , or $G_5^X G_6^X$. As in Sec. IV, incorrectly applied gauge operators can cause logical errors under subsequent Version 2 logical gates, so a standalone verification of this preparation circuit requires measuring both G_5^Z and G_6^Z . This verification strategy is used in the Grover algorithm simulation in Sec. VII.

APPENDIX C: EXTENSIONS OF THE SINGLE INTERBLOCK LOGICAL $\overline{\text{CNOT}}$ GATE

The logical $\overline{\text{SWAP}}$ gates in Fig. 3 are used only to move the target from $|2, \overline{2}\rangle$ to $|2, \overline{1}\rangle$; they are not required for the core construction. Removing them yields Fig. 16(a), which implements a single interblock logical $\overline{\text{CNOT}}$ from logical qubit $\overline{1}$ of block 1 to logical qubit $\overline{2}$ of block 2.

More generally, two layers of parallel interblock logical $\overline{\text{CNOT}}$ gates, separated by an intrablock logical $\overline{\text{CNOT}}(\overline{i} \rightarrow \overline{j})$ on one block and followed by the same intrablock gate again, cancel the unwanted pairwise couplings and leave only the desired single interblock logical $\overline{\text{CNOT}}$ from logical qubit $\overline{i}$ in one block to logical qubit $\overline{j}$ in the other. In Fig. 16(a), the intrablock gate is applied on block 1, whereas Fig. 16(b) shows the equivalent construction with the intrablock gate applied on block 2. The latter form is useful when only one of the two blocks admits a convenient fault-tolerant intrablock logical $\overline{\text{CNOT}}$ implementation.

The same cancellation idea also gives a single interblock logical $\overline{\text{CZ}}$: Replacing the two layers of parallel interblock $\overline{\text{CNOT}}$ gates by layers of parallel interblock $\overline{\text{CZ}}$ gates yields the construction shown in Fig. 16(c).

[1] P. W. Shor, Scheme for reducing decoherence in quantum computer memory, *Phys. Rev. A* **52**, R2493 (1995).

[2] A. M. Steane, Error correcting codes in quantum theory, *Phys. Rev. Lett.* **77**, 793 (1996).

- [3] D. Gottesman, Class of quantum error-correcting codes saturating the quantum Hamming bound, *Phys. Rev. A* **54**, 1862 (1996).
- [4] A. R. Calderbank, E. M. Rains, P. W. Shor, and N. J. A. Sloane, Quantum error correction and orthogonal geometry, *Phys. Rev. Lett.* **78**, 405 (1997).
- [5] E. Knill and R. Laflamme, Theory of quantum error-correcting codes, *Phys. Rev. A* **55**, 900 (1997).
- [6] E. Knill, R. Laflamme, and L. Viola, Theory of quantum error correction for general noise, *Phys. Rev. Lett.* **84**, 2525 (2000).
- [7] D. Bacon, Operator quantum error-correcting subsystems for self-correcting quantum memories, *Phys. Rev. A* **73**, 012340 (2006).
- [8] D. Kribs, R. Laflamme, and D. Poulin, Unified and generalized approach to quantum error correction, *Phys. Rev. Lett.* **94**, 180501 (2005).
- [9] D. Poulin, Stabilizer formalism for operator quantum error correction, *Phys. Rev. Lett.* **95**, 230504 (2005).
- [10] R. L. Kosut, A. Shabani, and D. A. Lidar, Robust quantum error correction via convex optimization, *Phys. Rev. Lett.* **100**, 020502 (2008).
- [11] F. Gaitan, *Quantum Error Correction and Fault Tolerant Quantum Computing* (Taylor & Francis Group, Boca Raton, 2008).
- [12] *Quantum Error Correction*, edited by D. Lidar and T. Brun (Cambridge University Press, Cambridge, UK, 2013).
- [13] D. P. DiVincenzo and P. W. Shor, Fault-tolerant error correction with efficient quantum codes, *Phys. Rev. Lett.* **77**, 3260 (1996).
- [14] D. Aharonov and M. Ben-Or, Fault-tolerant quantum computation with constant error rate, *SIAM J. Comput.* **38**, 1207 (2008).
- [15] P. Aliferis, D. Gottesman, and J. Preskill, Quantum accuracy threshold for concatenated distance-3 codes, *Quantum Inf. Comput.* **6**, 97 (2006).
- [16] B. W. Reichardt, Fault-tolerance threshold for a distance-three quantum code, in *Automata, Languages and Programming: 33rd International Colloquium, ICALP 2006, Venice, Italy, July 10–14, 2006, Proceedings, Part I*, edited by M. Bugliesi, B. Preneel, V. Sassone, and I. Wegener (Springer, Berlin, 2006), pp. 50–61.
- [17] E. Knill, Quantum computing with realistically noisy devices, *Nature (London)* **434**, 39 (2005).
- [18] E. T. Campbell, B. M. Terhal, and C. Vuillot, Roads towards fault-tolerant universal quantum computation, *Nature (London)* **549**, 172 (2017).
- [19] R. Acharya *et al.*, Quantum error correction below the surface code threshold, *Nature (London)* **638**, 920 (2025).
- [20] A. Eickbusch *et al.*, Demonstration of dynamic surface codes, *Nat. Phys.* **21**, 1994 (2025).
- [21] N. Lacroix *et al.*, Scaling and logic in the colour code on a superconducting quantum processor, *Nature (London)* **645**, 614 (2025).
- [22] D. Bluvstein *et al.*, A fault-tolerant neutral-atom architecture for universal quantum computation, *Nature (London)* **649**, 39 (2026).
- [23] A. Vezvaei, C. Benito, M. Morford-Oberst, A. Bermudez, and D. A. Lidar, Surface code scaling on heavy-hex superconducting quantum processors, *Nat. Commun.* **17**, 9201 (2026).
- [24] P. W. Shor, Fault-tolerant quantum computation, in *Proceedings of the 37th Annual Symposium on Foundations of Computer Science* (IEEE Computer Society Press, Los Alamitos, CA, 1996), pp. 56–65.
- [25] B. Eastin and E. Knill, Restrictions on transversal encoded quantum gate sets, *Phys. Rev. Lett.* **102**, 110502 (2009).
- [26] C. Gerhard and T. A. Brun, Weakly fault-tolerant computation in a quantum error-detecting code, *Phys. Rev. A* **114**, 012429 (2026).
- [27] A. Y. Kitaev, Quantum computations: Algorithms and error correction, *Russ. Math. Surv.* **52**, 1191 (1997).
- [28] P. Boykin, T. Mor, M. Pulver, V. Roychowdhury, and F. Vatan, On universal and fault-tolerant quantum computing: A novel basis and a new constructive proof of universality for Shor’s basis, in *40th Annual Symposium on Foundations of Computer Science* (IEEE Computer Society Press, Los Alamitos, CA, 1999), p. 486.
- [29] S. Bravyi and A. Kitaev, Universal quantum computation with ideal Clifford gates and noisy ancillas, *Phys. Rev. A* **71**, 022316 (2005).
- [30] D. Litinski, Magic state distillation: Not as costly as you think, *Quantum* **3**, 205 (2019).
- [31] C. Gidney, N. Shutty, and C. Jones, Magic state cultivation: Growing T states as cheap as CNOT gates, [arXiv:2409.17595](https://arxiv.org/abs/2409.17595).
- [32] Y. Vaknin, S. Jacoby, A. Grimsmo, and A. Retzker, Efficient magic state cultivation on the surface code, [arXiv:2502.01743](https://arxiv.org/abs/2502.01743).
- [33] J. T. Anderson, G. Duclos-Cianci, and D. Poulin, Fault-tolerant conversion between the Steane and Reed-Muller quantum codes, *Phys. Rev. Lett.* **113**, 080501 (2014).
- [34] H. Bombín, Gauge color codes: Optimal transversal gates and gauge fixing in topological stabilizer codes, *New J. Phys.* **17**, 083002 (2015).
- [35] A. Kubica and M. E. Beverland, Universal transversal gates with color codes: A simplified approach, *Phys. Rev. A* **91**, 032330 (2015).
- [36] D.-X. Quan, L.-L. Zhu, C.-X. Pei, and B. C. Sanders, Fault-tolerant conversion between adjacent Reed-Muller quantum codes based on gauge fixing, *J. Phys. A: Math. Theor.* **51**, 115305 (2018).
- [37] F. Butt, S. Heußen, M. Rispler, and M. Müller, Fault-tolerant code-switching protocols for near-term quantum processors, *PRX Quantum* **5**, 020345 (2024).
- [38] M. Vasmer and A. Kubica, Morphing quantum codes, *PRX Quantum* **3**, 030319 (2022).
- [39] E. T. Campbell, The smallest interesting colour code (2016), <https://earlrcampbell.com/2016/09/26/the-smallest-interesting-colour-code/>.
- [40] D. Hangleiter, M. Kalinowski, D. Bluvstein, M. Cain, N. Maskara, X. Gao, A. Kubica, M. D. Lukin, and M. J. Gullans, Fault-tolerant compiling of classically hard instantaneous quantum polynomial circuits on hypercubes, *PRX Quantum* **6**, 020338 (2025).
- [41] Y. Wang *et al.*, Fault-tolerant one-bit addition with the smallest interesting color code, *Sci. Adv.* **10**, eado9024 (2024).
- [42] D. Honciuc Menendez, A. Ray, and M. Vasmer, Implementing fault-tolerant non-Clifford gates using the $[[8,3,2]]$ color code, *Phys. Rev. A* **109**, 062438 (2024).
- [43] D. Bluvstein *et al.*, Logical quantum processor based on reconfigurable atom arrays, *Nature (London)* **626**, 58 (2024).
- [44] L. K. Grover, A fast quantum mechanical algorithm for database search, in *Proceedings of the 28th Annual ACM Symposium on Theory of Computing*, STOC ’96 (ACM, New York, NY, 1996), pp. 212–219.

- [45] L. K. Grover, Quantum computers can search arbitrarily large databases by a single query, *Phys. Rev. Lett.* **79**, 4709 (1997).
- [46] C. Gidney and M. Ekerå, How to factor 2048 bit RSA integers in 8 hours using 20 million noisy qubits, *Quantum* **5**, 433 (2021).
- [47] M. Roetteler, M. Naehrig, K. M. Svore, and K. Lauter, Quantum resource estimates for computing elliptic curve discrete logarithms, in *Advances in Cryptology—ASIACRYPT 2017*, edited by T. Takagi and T. Peyrin, Lecture Notes in Computer Science, Vol. 10625 (Springer, Cham, 2017), pp. 241–270.
- [48] D. Litinski, How to compute a 256-bit elliptic curve private key with only 50 million Toffoli gates, [arXiv:2306.08585](https://arxiv.org/abs/2306.08585).
- [49] R. Anand, A. Maitra, and S. Mukhopadhyay, Grover on Simon, *Quantum Inf. Process.* **19**, 340 (2020).
- [50] J. Lee, D. W. Berry, C. Gidney, W. J. Huggins, J. R. McClean, N. Wiebe, and R. Babbush, Even more efficient quantum computations of chemistry through tensor hypercontraction, *PRX Quantum* **2**, 030305 (2021).
- [51] M. S. Zini, A. Delgado, R. dos Reis, P. A. M. Casares, J. E. Mueller, A.-C. Voigt, and J. M. Arrazola, Quantum simulation of battery materials using ionic pseudopotentials, *Quantum* **7**, 1049 (2023).
- [52] A. J. Bay-Smidt, N. Glaser, M. D. Fabian, E. T. Campbell, N. S. Blunt, and G. C. Solomon, Quantum simulation of nanographenes and Trotter error cancellation, [arXiv:2605.00745](https://arxiv.org/abs/2605.00745).
- [53] E. T. Campbell, Early fault-tolerant simulations of the Hubbard model, *Quantum Sci. Technol.* **7**, 015007 (2022).
- [54] Y. R. Sanders, D. W. Berry, P. C. Costa, L. W. Tessler, N. Wiebe, C. Gidney, H. Neven, and R. Babbush, Compilation of fault-tolerant quantum heuristics for combinatorial optimization, *PRX Quantum* **1**, 020312 (2020).
- [55] J. F. Doriguello, G. Giapitzakis, A. Luongo, and A. Morolia, On the practicality of quantum sieving algorithms for the shortest vector problem, in *Post-Quantum Cryptography: 17th International Conference, PQCrypto 2026*, edited by M. Bardet and R. Niederhagen, Lecture Notes in Computer Science, Vol. 16491 (Springer, Cham, 2026), pp. 3–36.
- [56] P. Singkanipa, Z. Xia, and D. A. Lidar, Families of $d = 2$ 2D subsystem stabilizer codes for universal Hamiltonian quantum computation with two-body interactions, *Quantum* **9**, 1821 (2025).
- [57] D. B. Tan, D. Bluvstein, M. D. Lukin, and J. Cong, Compiling quantum circuits for dynamically field-programmable neutral atoms array processors, *Quantum* **8**, 1281 (2024).
- [58] D. Bluvstein, H. Levine, G. Semeghini, T. T. Wang, S. Ebadi, M. Kalinowski, A. Keesling, N. Maskara, H. Pichler, M. Greiner, V. Vuletić, and M. D. Lukin, A quantum processor based on coherent transport of entangled atom arrays, *Nature (London)* **604**, 451 (2022).
- [59] J. M. Pino, J. M. Dreiling, C. Figgatt, J. P. Gaebler, S. A. Moses, M. S. Allman, C. H. Baldwin, M. Foss-Feig, D. Hayes, K. Mayer, C. Ryan-Anderson, and B. Neyenhuis, Demonstration of the trapped-ion quantum CCD computer architecture, *Nature (London)* **592**, 209 (2021).
- [60] P. Schindler, D. Nigg, T. Monz, J. T. Barreiro, E. Martinez, S. X. Wang, S. Quint, M. F. Brandl, V. Nebendahl, C. F. Roos, M. Chwalla, M. Hennrich, and R. Blatt, A quantum information processor with trapped ions, *New J. Phys.* **15**, 123012 (2013).
- [61] G. Li, Y. Ding, and Y. Xie, Tackling the qubit mapping problem for NISQ-era quantum devices, in *Proceedings of the Twenty-Fourth International Conference on Architectural Support for Programming Languages and Operating Systems* (Association for Computing Machinery, New York, NY, 2019), pp. 1001–1014.
- [62] A. Cowtan, S. Dilkes, R. Duncan, A. Krajenbrink, W. Simmons, and S. Sivarajah, On the qubit routing problem, in *14th Conference on the Theory of Quantum Computation, Communication and Cryptography (TQC 2019)*, Leibniz International Proceedings in Informatics (LIPIcs), Vol. 135 (Schloss Dagstuhl—Leibniz-Zentrum für Informatik, Dagstuhl, Germany, 2019), pp. 5:1–5:32.
- [63] D. Gottesman, Stabilizer codes and quantum error correction, Ph.D. thesis, California Institute of Technology, Pasadena, CA, 1997.
- [64] R. Chao and B. W. Reichardt, Quantum error correction with only two extra qubits, *Phys. Rev. Lett.* **121**, 050502 (2018).
- [65] C. Ryan-Anderson, PECOS: Performance estimator of codes on surfaces, GitHub, 2019, <https://github.com/PECOS-packages/PECOS>.
- [66] X. Zhou, D. W. Leung, and I. L. Chuang, Methodology for quantum logic gate construction, *Phys. Rev. A* **62**, 052316 (2000).
- [67] B. Pokharel and D. Lidar, Better-than-classical Grover search via quantum error detection and suppression, *npj Quantum Inf.* **10**, 23 (2024).
- [68] T. Ginsberg and V. Patel, Quantum error detection for early term fault-tolerant quantum algorithms, [arXiv:2503.10790](https://arxiv.org/abs/2503.10790).
- [69] F. Butt, I. Pogorelov, R. Freund, A. Steiner, M. Meyer, T. Monz, and M. Müller, Demonstration of measurement-free universal logical quantum computation, *Nat. Commun.* **17**, 995 (2026).
- [70] M. Boyer, G. Brassard, P. Hoyer, and A. Tapp, Tight bounds on quantum searching, *Fortschr. Phys.* **46**, 493 (1998).
- [71] E. Biham, O. Biham, D. Biron, M. Grassl, and D. A. Lidar, Grover’s quantum search algorithm for an arbitrary initial amplitude distribution, *Phys. Rev. A* **60**, 2742 (1999).
- [72] Qiskit Aer contributors, Qiskit Aer: A high performance simulator for quantum circuits, GitHub repository, 2024, <https://github.com/Qiskit/qiskit-aer>.
- [73] A. Kubica, B. Yoshida, and F. Pastawski, Unfolding the color code, *New J. Phys.* **17**, 083026 (2015).
- [74] M. A. Webster, B. J. Brown, and S. D. Bartlett, The XP stabiliser formalism: A generalisation of the Pauli stabiliser formalism with arbitrary phases, *Quantum* **6**, 815 (2022).
- [75] S. Wu and D. Zhong, Universal weakly fault-tolerant quantum computation via code switching in the $[[8, 3, 2]]$ code [Dataset], Zenodo, version 2, 2026, doi: [10.5281/zenodo.22006449](https://doi.org/10.5281/zenodo.22006449).
- [76] D. M. Greenberger, M. A. Horne, and A. Zeilinger, Going beyond Bell’s theorem, in *Bell’s Theorem, Quantum Theory and Conceptions of the Universe*, edited by M. Kafatos (Springer Netherlands, Dordrecht, 1989), pp. 69–72.