Error rates and resource overheads of encoded three-qubit gates

Ryuji Takagi, Theodore J. Yoder, and Isaac L. Chuang

Department of Physics, Massachusetts Institute of Technology, 77 Massachusetts Avenue, Cambridge, Massachusetts 02139, USA

(Received 4 July 2017; published 4 October 2017)

A non-Clifford gate is required for universal quantum computation, and, typically, this is the most error-prone and resource-intensive logical operation on an error-correcting code. Small, single-qubit rotations are popular choices for this non-Clifford gate, but certain three-qubit gates, such as Toffoli or controlled-controlled-Z (CCZ), are equivalent options that are also more suited for implementing some quantum algorithms, for instance, those with coherent classical subroutines. Here, we calculate error rates and resource overheads for implementing logical CCZ with pieceable fault tolerance, a nontransversal method for implementing logical gates. We provide a comparison with a nonlocal magic-state scheme on a concatenated code and a local magic-state scheme on the surface code. We find the pieceable fault-tolerance scheme particularly advantaged over magic states on concatenated codes and in certain regimes over magic states on the surface code. Our results suggest that pieceable fault tolerance is a promising candidate for fault tolerance in a near-future quantum computer.

I. INTRODUCTION

Quantum error-correcting codes are the most promising route to scalable quantum computation. However, some of their limitations are well known. For instance, a major problem is that a single code cannot support a full set of universal, transversal operations [1–3]. Often, and always for 2D designs [4], the missing gate is not in the normalizer of the Pauli group; that is, it is non-Clifford.

The techniques of gate teleportation [5] and magic states [6] can overcome the lack of a non-Clifford gate. Different magic states can be created to implement small Z rotations such as the T-gate or 3-qubit operations, like Toffoli or controlled-controlled-Z (CCZ). However, the process to create a magic state occurs postselectively and recursively, and leads to large resource overheads. Although improving consistently [7,8], approaching believed fundamental limits [9], large resource demands remain a serious obstacle for near-future architectures.

Certain other approaches exist in the literature for implementing a universal gate set while circumventing the use of magic states. A popular approach is gauge fixing [10–12], in which a subsystem code can implement complementary sets of transversal logical gates depending on the settings of the gauge qubits. Another approach [13–15] concatenates different codes with complementary transversal gate sets to achieve the same effect in one larger code. Recently, this approach was shown to lead to asymptotic thresholds around $\sim 10^{-3}$, albeit using more physical qubits than, for example, surface code magic-state distillation [16,17].

Any fault-tolerant, universal computing scheme operating without magic states is expected to be a promising candidate for near-future architectures where fairly accurate physical components are supplied, but space-time resources, like qubit count and circuit depth, are limited. The primary goal in this near-future regime is to achieve some desired target error rate after a finite-sized computation with small resource overheads. Such constraints imply that the logical error rates of encoded gates and the first-level pseudothreshold [18] (called just pseudothreshold hereafter) are more important measures than asymptotic threshold, which only becomes meaningful with access to huge amounts of resources.

To evaluate near-future fault-tolerant computation, we focus on another magicless alternative that allows for a logical implementation of 3-qubit gates, the pieceable fault-tolerance scheme [19]. In this approach, a logical gate is done nontransversally through the "round-robin" construction and made fault tolerant via partial error correction performed throughout the circuit. This construction has recently been used in [20] to perform fault-tolerant, universal computing on seven logical qubits, requiring only four ancillary qubits and 15 code qubits. The circuit volume metric, a space-time resource measure that counts all gates weighted by the number of qubits involved, was used in [19] to argue that pieceable fault tolerance reduces the logical gate overhead by nearly a factor of 2 over magic-state creation and injection. However, little was said about error rates of pieceable gates.

In this paper, we calculate these error rates and compare them to magic-state schemes for implementing 3-qubit non-Clifford gates. Our contenders are (1) a nonlocal magic-state scheme: magic states created postselectively on Steane's 7-qubit code (also known as the smallest color code); (2) a local magic-state scheme: surface code magic-state distillation; and (3) pieceable fault tolerance on the (a) 5-qubit [19], (b) 7-qubit [19], (c) 3×3 Bacon-Shor [21], and (d) 3×9 Bacon-Shor [21] codes. Our metrics are (I) error rate of the logical gate and (II) circuit volume. Among concatenated schemes (1) and (3), we can definitively declare pieceable 3×3 Bacon-Shor the winner in both metrics (I) and (II). When comparing to (2), the picture is more complicated and interesting. The pieceable 3×3 Bacon-Shor beats the surface code in error rate at low distance and in circuit volume when the physical error rate is sufficiently low compared with the desired target logical error. On the other hand, asymptotically in code distance, the surface code outperforms the pieceable 3×3 Bacon-Shor due to better scaling of the logical error rate and volume with distance.

II. METHODS

We first describe our method to evaluate the logical error rates. Evaluating the surface code scheme (2) draws on the extensive literature on the topic [22]. Our calculations of the logical error rates of schemes (1) and (3) at code distance

$d = 3$ are done by exact enumeration of all combinations of up to two faults in the circuit extended-rectangle (exREC) [23] under the standard depolarizing noise model (which serves as a model of average-case noise). In [23], a rigorous upper bound on the logical error rate under depolarizing noise is given. In contrast, we provide formulas giving a rigorous lower bound as well as a tighter rigorous upper bound. The lower and upper bounds on the logical error rate also determine lower and upper bounds on the pseudothreshold. Having both bounds allows us to definitively prove a separation between two different schemes when it exists. Our method also confers some advantages over a Monte Carlo simulation. First, we can rigorously verify our circuits are fault tolerant under the chosen noise model by checking that all single faults are correctable. Second, once the counting is complete, we can independently vary noise for each type of gate.

Our standard noise breakdown assigns single-qubit gates, 2-qubit gates, and 3-qubit gates each their own failure probabilities p_1, p_2, and p_3, respectively. In the circuit depolarization noise model, an r-qubit gate fails with one of the $4^r - 1$ r-qubit Pauli errors with probability $p_r/(4^r - 1)$. In principle, preparation and measurement could be treated separately as well, though we will assign them failure probabilities also equal to p_1. Bounds on the error rate can always be written as polynomials in p_1, p_2, p_3 as we discuss below.

Our ultimate goal in error-rate estimation is to find the probability the exREC is incorrect given that all ancillas pass verification. Denote this $P_{\text{fail}|\text{acc}} = \text{Pr}[\text{fail}|\text{acc}]$. Our counting gives the exact values of

$$P_{\text{fail,acc}}^{(2)} = \text{Pr}[\text{fail,acc}, \leqslant 2 \text{ faults}], \quad (1)$$

$$P_{\text{succ,acc}}^{(2)} = \text{Pr}[\neg\text{fail,acc}, \leqslant 2 \text{ faults}], \quad (2)$$

$$P_{\text{rej}}^{(2)} = \text{Pr}[\neg\text{acc}, \leqslant 2 \text{ faults}], \quad (3)$$

as polynomials in p_1, p_2, p_3 with degree equal to the number of potentially faulty components in the entire exREC. These exactly calculated quantities are enough to bound $P_{\text{fail,acc}} = \text{Pr}[\text{fail,acc}]$, $P_{\text{succ,acc}} = \text{Pr}[\neg\text{fail,acc}]$, and $P_{\text{acc}} = \text{Pr}[\text{acc}]$ as

$$P_{\text{fail,acc}}^{(2)} \leqslant P_{\text{fail,acc}}, \quad (4)$$

$$P_{\text{succ,acc}}^{(2)} \leqslant P_{\text{succ,acc}}, \quad (5)$$

$$P_{\text{acc}} \leqslant 1 - P_{\text{rej}}^{(2)}. \quad (6)$$

Thus,

$$\frac{P_{\text{fail,acc}}^{(2)}}{1 - P_{\text{rej}}^{(2)}} \leqslant P_{\text{fail}|\text{acc}} = 1 - P_{\text{succ}|\text{acc}} \leqslant 1 - \frac{P_{\text{succ,acc}}^{(2)}}{1 - P_{\text{rej}}^{(2)}}. \quad (7)$$

More details on the simulation, including the description on how to obtain these polynomials, can be found in Appendix B.

Next, we consider evaluating the resource overhead. There exist various resource measures such as qubit count, circuit volume, gate counts, and so on. The number of reusable physical qubits is often taken as a physical resource measure in the literature. However, it is not the best, especially when we would like to compare resource overheads between different codes, because there is ambiguity that comes with the level

of parallelization we assume. In this paper, we mainly focus on circuit volume, a space-time resource measure that counts all gates weighted by the number of qubits involved. Unlike physical qubit count, circuit volume takes into account the tradeoff between space and time resources. The circuit volume is a space-time metric in the same vein as the "quantum volume" [24], except for evaluating specific circuits rather than a universal quantum computer.

The circuit volume at a high concatenation level is easy to compute using the volume of the logical construction at the first level of encoding. Let $V_G^{(k)}$ be the volume for implementing circuit component G at the kth level of concatenation. Then there is a recursion relation between two concatenation levels, $V_G^{(k+1)} = \sum_{G'} N_G^{G'} V_{G'}^{(k)}$, where $N_G^{G'}$ is the number of the circuit component G' in the logical construction of component G. We can understand this as evolution of a vector of circuit volumes of each component via a transformation matrix determined by the logical gate constructions. Namely, we get

$$\mathbf{V}^{(\mathbf{k})} = A^k \mathbf{V}^{(\mathbf{0})}, \quad (8)$$

where A is the matrix $A_{ij} = N_{G_i}^{G_j}$, $\mathbf{V}^{(\mathbf{k})}_i = V_{G_i}^{(k)}$, and $V_G^{(0)}$ is the volume of an unencoded component. We set $\mathbf{V}^{(\mathbf{k})} = (V_3^{(k)}, V_2^{(k)}, V_1^{(k)}, V_{\text{prep}}^{(k)}, V_{\text{meas}}^{(k)})^T$, where the components refer to the circuit volume of 3-qubit gates, 2-qubit gates, single-qubit gates, $|0\rangle$ or $|+\rangle$ preparation, and measurement, respectively. Note that $(V_3^{(0)}, V_2^{(0)}, V_1^{(0)}, V_{\text{prep}}^{(0)}, V_{\text{meas}}^{(0)}) = (3, 2, 1, 1, 1)$.

III. LOGICAL CONSTRUCTIONS

Here, we describe the logical constructions used in the simulation. Explicit descriptions of the circuits at the gate level can be found in [25]. All of our constructions begin with a round of syndrome measurement and recovery (the leading error correction) and end with the same (the trailing error correction), in accordance with the exREC formalism [23]. The rest of the circuit may also include rounds of error correction, called intermediate, in accordance with piecewise fault tolerance [19].

For the 5-qubit code, we implement a logical CCZ gate by the round-robin construction [19] with three intermediate error corrections. The leading error correction and trailing error correction are done by Steane's error correction [26]. Since the 5-qubit code is non-CSS, a 10-qubit ancilla is needed to extract the entire syndrome simultaneously. We actually find that the circuit in [26] needs some modification for non-CSS codes, which we discuss in the Appendix A in detail. For intermediate error corrections, we use Shor-type error correction with cat states [27]. The size of the cat states is always 4 for measuring constant stabilizers (those that commute with the preceeding circuitry), but it varies for measuring nonconstant stabilizers because their weight changes as they go through the CCZ gates. For our circuit, we need to use 9-cat, 13-cat, 9-cat at maximum for the first, second, and third intermediate error correction, respectively.

For the 7-qubit code, we consider the construction that requires only one intermediate error correction [19]. All of the error corrections are done by Steane's error correction. Since the 7-qubit code is a CSS code, correction of Z-type errors can be done separately from that of X-type errors, and only

the encoded states $|\bar{0}\rangle$ and $|\bar{+}\rangle$ are needed. The state $|\bar{0}\rangle(|\bar{+}\rangle)$ is verified by applying CNOT gates transversally to another noisy $|\bar{0}\rangle(|\bar{+}\rangle)$ and measuring it transversally (a Steane ancilla factory [28]). If some error is detected, we discard the state and start again. For estimating the circuit volume, we consider a more resource-efficient state preparation method proposed by Goto [29]. Although we did not estimate the logical error rate using Goto's method, we suspect that the change in the logical error rate between different verification methods would be small, as indicated in [29]. Since intermediate Z-type error correction is not needed, we simply apply the X-type error correction in the middle and notify the trailing error correction about possible locations of Z-type errors as described in [19].

Logical CCZ on the Bacon-Shor code is implemented as proposed in [21]. On the 3×3 Bacon-Shor we need no intermediate correction, although we do use a non-Pauli recovery at the end. Furthermore, since the ancilla for the error correction is a tensor product of 3-cat states, there is no need for verification since, modulo its stabilizers, an error on a 3-cat is equivalent to a weight-1 error. In contrast, the 3×9 Bacon-Shor implements logical CCZ transversally, but it comes with a substantially larger overhead [21].

For the nonlocal magic-state scheme, we use magic-state injection on the 7-qubit code to implement a logical CCZ gate. The CCZ magic state is defined by the stabilizers $\langle X_1 CZ(2,3), X_2 CZ(1,3), X_3 CZ(1,2) \rangle$. The protocol consists of two parts, a state preparation circuit and a teleportation circuit. The state preparation starts with the $+1$ eigenstate of the second and the third stabilizer, $|\bar{0}\rangle|\bar{+}\rangle|\bar{+}\rangle$, and measures the first stabilizer [30]. Our circuit is a variant of the circuit in [31], which we modify to create the CCZ state instead. Two measurements of $X_1 CZ(2,3)$ are done with complete error correction in between. This makes the circuit fault tolerant (to one fault). If the two measurement results do not match, we discard the created state and start over again. If they match and they both show the result -1, we apply $\bar{Z}$ on the first code block to put it back to the desired magic state. If both show $+1$, we do not need to apply a correction. Like the pieceable 7-qubit case, all the error corrections are done using Steane's method [26].

IV. COMPARISON OF CONCATENATED SCHEMES

We compute the logical error rates and resource overheads of pieceably fault-tolerant CCZ gates on the 5-qubit code, 7-qubit code [19], 3×3 Bacon-Shor code, and 3×9 Bacon-Shor code [21], and compare them to a magic-state scheme on the 7-qubit code. Figure 1 shows the obtained logical error rates for these cases using two different settings of physical error rate, $p_1 = p_2 = p_3 = p$ and $10p_1 = p_2 = 0.1p_3 = p$. Lower and upper bounds on pseudothresholds are the crossing points of the "break-even" line and the upper and lower bounds for logical error rates. For both settings of physical error rate, the 3×3 Bacon-Shor code has a lower logical error rate than the magic-state scheme below pseudothreshold. For the 7-qubit code, whether the pieceable scheme has a lower rate than the magic-state scheme depends on the physical error-rate setting. The 5-qubit code has a large logical error rate due to a large number of pieces in the round-robin construction. Similarly, the 3×9 Bacon-Shor code has a higher logical error rate than
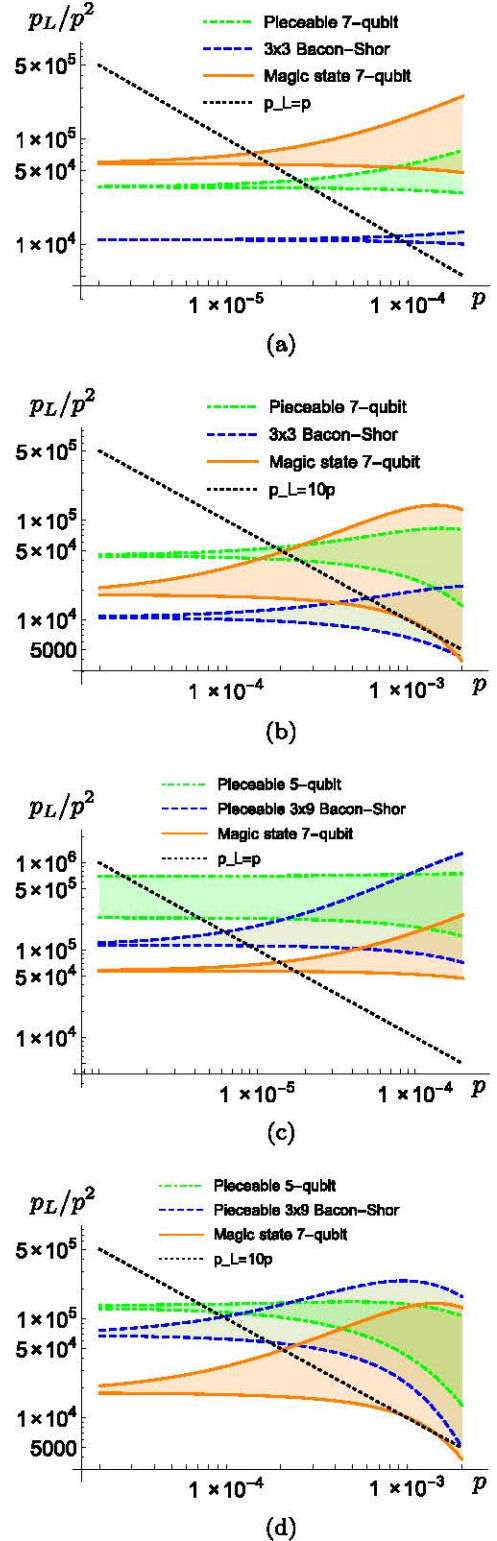

FIG. 1. Logical error rates of 3-qubit gate on (a, b) pieceable 7-qubit code (green, dot-dashed), pieceable 3×3 Bacon-Shor code (blue, dashed), (c, d) pieceable 5-qubit code (green, dot-dashed), pieceable 3×9 Bacon-Shor code (blue, dashed), and magic-state injection on 7-qubit code (orange, solid) with (a, c) $p_1 = p_2 = p_3 = p$ and (b, d) $10p_1 = p_2 = 0.1p_3 = p$, where p_i refers to the physical error rate of an i-qubit gate. Initialization of single-qubit states $|0\rangle$, $|+\rangle$ also fails with probability p_1. Dotted line is the "break-even" line where the logical error rate coincides with the physical error rate.

TABLE I. Resource overheads to implement logical CCZ. Volume refers to the circuit volume, which counts all gates weighted by the number of qubits involved. Qubits are the number of physical qubits, including data qubits and ancilla qubits, where ancilla qubits are assumed to be not reusable. Numbers for the 5-qubit code include all the resources for the adaptive measurements.

	Volume	Qubits	2-qubit gates	3-qubit gates
Pieceable 5-qubit	3841	364	445	46
Pieceable 7-qubit	771	93	162	21
3×3 Bacon-Shor	414	81	90	27
3×9 Bacon-Shor	1350	252	306	27
Magic state	1352	154	267	14
3×3 BS/Magic	0.31	0.59	0.34	1.9

the 3×3 Bacon-Shor code because the size of the logical code block is obviously much bigger. Moreover, the 3×9 Bacon-Shor needs to implement verification for 9-qubit cat states.

We now compare the resource overheads. Table I shows the resource overheads to implement a logical CCZ gate with these constructions. We assume that the ancillas are not reusable. Due to a finite ancilla verification rejection rate, the effective resource count is slightly higher than the values in the table. However, the rejection rate of the verification is $O(p)$, and the effective resource count is obtained by multiplying $(1 - n_{\rm rej}p)^{-1}$ where $n_{\rm rej}$ is the number of error locations that lead to rejection. Since we are interested in the region $p < 10^{-4}$, and the largest module involving verification is the magic-state preparation circuit, which has $n_{\rm rej} \approx 100$, increase in the resource due to verification is within 1%. Thus, it is safe to ignore the effects of verification. Besides using more 3-qubit gates, pieceable constructions on the 7-qubit and 3×3 Bacon-Shor code have smaller resource overheads compared to the magic-state scheme. In particular, they have a significant reduction in circuit volume. Figure 3 shows circuit volumes for the pieceable 7-qubit code, 3×3 Bacon-Shor code, and magic-state scheme. Transformation matrices A [see Eq. (8)] for these codes are given in Appendix D.

Combining the results for the logical error rates and circuit volume, we conclude that the pieceable construction on the 3×3 Bacon-Shor code beats the magic-state scheme on the 7-qubit code in both the criteria. The pieceable construction on the 7-qubit code also beats magic-state injection in circuit volume, and in logical error rate when $p_1 = p_2 = p_3$.

V. COMPARISON TO SURFACE CODES

Next, we compare logical error rate and resource overheads to a local magic-state scheme on surface codes. We find that the pieceable construction can have a significant advantage in circuit volume in a certain region in terms of physical error rate and target logical error rate.

A. Logical error rates

Surface codes are known to have a high asymptotic threshold, which is 0.1%–1% depending on assumptions and error model [22,32–35], and thus they have attracted attention as a candidate for a scalable quantum computer. However,
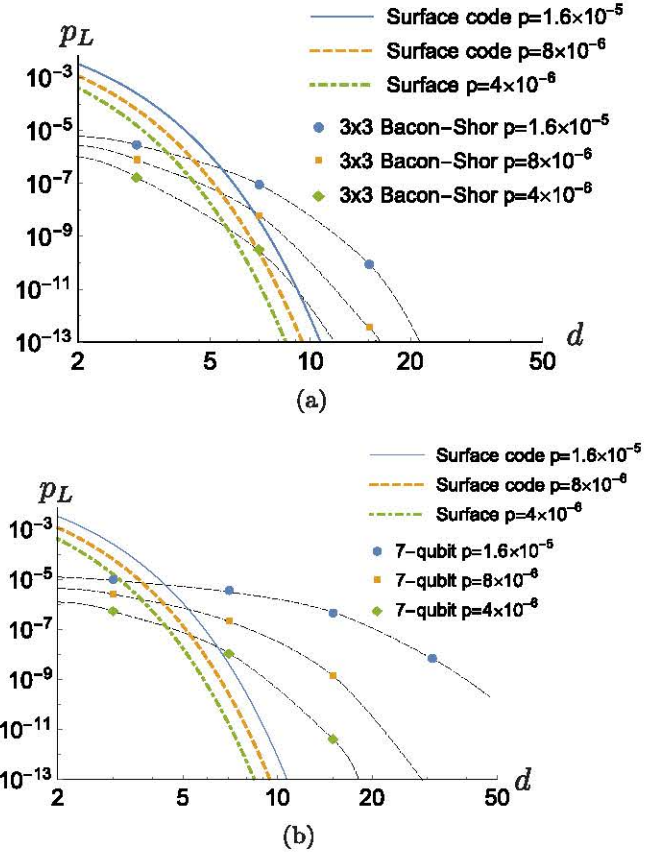

(a)

(b)

FIG. 2. Logical error rates for 3-qubit gate on the surface codes and (a) pieceable 3×3 Bacon-Shor code and (b) pieceable 7-qubit code in terms of code distance. Shown rates for pieceable codes are upper bounds obtained by concatenating the upper bounding function from Eq. (7). Black dashed curves are only a guide to the eye.

having a high asymptotic threshold does not automatically imply that the logical error rate is always low for reasonably sized codes.

Firstly, as can be seen in [22], in the low-distance regime the pseudothreshold of the surface code is much smaller than the asymptotic threshold. Thus, if the physical error rate is lower than the asymptotic threshold but not below the relevant pseudothreshold, encoding at low distance does not help to reduce the error rate.

Second, the logical error rate of a logical gate can be large even if the error rate for one surface code *cycle* is small, because a logical gate is made up of many cycles. Each cycle consists of measuring the complete error syndrome once via measurement qubits, one per stabilizer generator, as in [22]. Let $\bar{p}_{\rm cycle}$ be the logical error rate for the surface code per surface code cycle. Let C_G be the number of surface code cycles it takes to implement a logical version of gate G. Then, the logical error rate of gate G is $\bar{p}_G \approx C_G \bar{p}_{\rm cycle}$. Since $C_G \propto d$ and $\bar{p}_{\rm cycle} \propto p^{(d+1)/2}$ where d is the surface code distance, $\bar{p}_{\rm cycle}$ dominates for large distance. However, when d is small, the contribution to $\bar{p}_G$ from C_G is not negligible. In Appendix E we find a specific form of C_G for the logical Toffoli gate for two different implementations.

Figure 2 shows logical error rates of a 3-qubit gate on the surface code using a Toffoli state, and upper bounds of
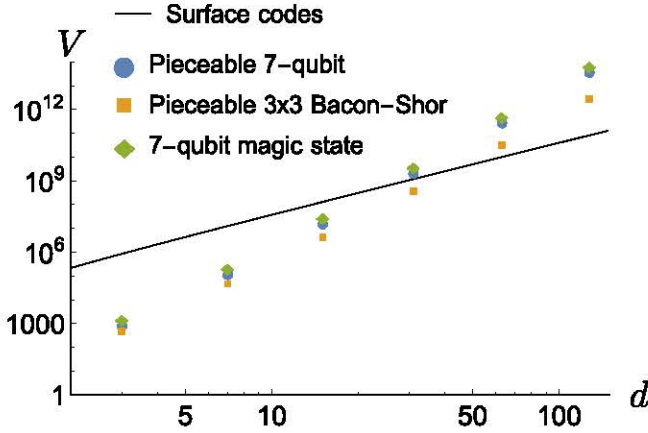
FIG. 3. Circuit volume for logical 3-qubit gate on pieceable 7-qubit code (circles), pieceable 3×3 Bacon-Shor code (squares), and magic-state scheme on 7-qubit code (diamonds) in terms of code distance. The dots correspond to every concatenation level in the range. Although it may be hard to see the data for the pieceable 7-qubit code because they are close to the data for the magic-state scheme, the pieceable 7-qubit has slightly lower volume than the magic-state scheme for every distance shown.

FIG. 4. The region where pieceable 3×3 Bacon-Shor code (orange and purple) and pieceable 7-qubit code (purple) use less volume than surface codes to implement 3-qubit gate to achieve fixed target logical error rate p_T with fixed physical error rate p. Dashed lines labeled by $\mathbf{ST_j}$ are upper bounds on the logical error rate at the jth concatenation level of the pieceable 7-qubit code (Steane code). The dotted line labeled by $\mathbf{SC(7)}$ is the logical error rate for the surface code with distance 7. The dot-dashed line labeled by $\mathbf{BS_3}$ is an upper bound of logical error rate at the third concatenation level of the 3×3 Bacon-Shor code. The solid line on the upper boundary is the $p_T = p$ line.

logical error rate of pieceable 3×3 Bacon-Shor code and pieceable 7-qubit code in terms of code distance with three different physical error rates. Upper bounds are obtained by concatenating the function upper bounding the actual rate in Eq. (7). Since the 3-qubit gate is the largest component among the components that appear in the logical construction of a 3-qubit gate, concatenating the upper bounding error function for the 3-qubit gate upper bounds its error rates at a higher concatenation level. However, because logical 3-qubit gates have an order of magnitude higher error rate than 2-qubit gates and the logical constructions of 3-qubit gates mostly consist of single gates and 2-qubit gates, this upper bound is highly pessimistic. A careful analysis taking into account error functions for other types of components and possibly even using a better decoding algorithm [36,37] at higher levels may greatly reduce estimates of logical error rates.

Nevertheless, in Fig. 2, we can see that surface codes have better scaling with distance than pieceable concatenated codes, which should be attributed to the high threshold. However, for small d, C_{Toffoli} has a significant contribution, and when $d = 3$ the logical error rate of the pieceable constructions is 2 orders of magnitude lower than that of the surface code.

B. Resource overheads

We also count the circuit volume for implementing logical Toffoli on surface codes. This allows us to compare the circuit volume between pieceable codes and the surface codes, shown in Fig. 3. Although surface codes have better scaling with distance, pieceable constructions have a significant advantage until three concatenations. This is especially true at distance 3, where the difference is 3 orders of magnitude.

Consider now the space consisting of pairs (physical error rate, target logical error rate) $\equiv (p, p_T)$. Combining volume and error-rate estimates, the region of this space where concatenated pieceable constructions require less circuit

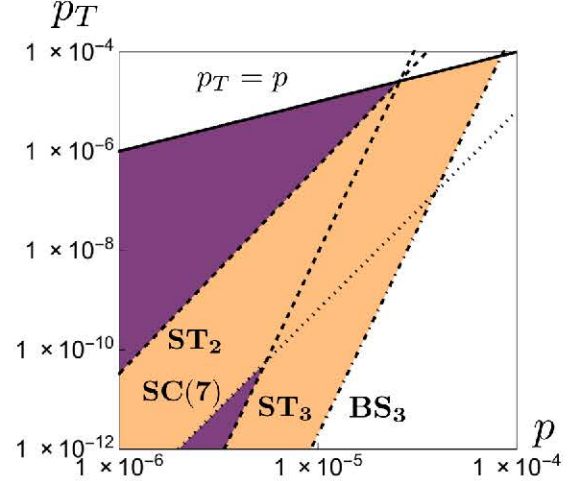volume for implementing Toffoli than the surface code can be obtained. Figure 4 shows this region for the pieceable 3×3 Bacon-Shor code and the 7-qubit code. It shows that in large range, the pieceable 3×3 Bacon-Shor code has an advantage in circuit volume over surface code, and the difference can be significant, as can be seen in Fig. 3. This region is actually determined by the upper bound of error rates at the third-level concatenation. This is because the surface code with distance 5 has already larger volume than 3×3 Bacon-Shor code with three concatenations, as can be seen in Fig. 3. For the 7-qubit code, Fig. 3 shows that a 3-qubit logical gate at two concatenations of the 7-qubit code has less circuit volume than the surface code of any size. Thus, whenever two concatenations are sufficient to achieve the target logical error rate, the 7-qubit code will be advantaged, as is represented by the region in Fig. 4. Figure 3 also shows that the volume for the 7-qubit code with three concatenations is slightly larger than that for the surface code with distance 7. Thus, the surface code is advantaged for the region where distance 7 is enough for the surface code, but three concatenations are needed for the 7-qubit code, which corresponds to the region between the upper purple region and the lower purple region in Fig. 4. The 7-qubit code again starts to have an advantage over the surface code for the region where the surface code needs distance 9, whereas the 7-qubit code needs to be concatenated only three times, which corresponds to the lower purple region in Fig. 4.

VI. CONCLUSIONS

In this paper, we calculated logical error rates and resource overheads of 3-qubit gates using pieceable fault-tolerant

constructions, a nonlocal magic-state scheme (on the 7-qubit code), and a local magic-state scheme (on the surface code). In comparison with the nonlocal magic-state scheme, we found that while pieceable constructions have comparable, or even lower logical error rate to the magic-state scheme, the required circuit volume can be as little as 30%. This suggests that the pieceable construction is a promising complement to schemes relying on magic states.

We also compared the pieceable construction to the surface codes and found that in quite a large region, in terms of physical error rates and target logical error rates, pieceable constructions can have significantly lower circuit volume than surface codes.

Although realizing physical components with a small physical error rate such that pieceable constructions have a great advantage is challenging, one should notice that surface codes also have as hard a challenge as this in terms of resource overheads. Just as surface codes are good candidates given access to large overheads, the pieceable construction appears to be a good candidate given access to small physical error rates.

Another difference between pieceable constructions and the surface code is locality, i.e., the constraint that the physical gates involved act only between qubits that are neighboring in some chosen low-dimensional layout. Although the locality property is desirable in many experimental setups, some systems allow nonlocal interaction, too [31]. Our result indicates that such a nonlocal techniques can lead to a significant reduction of resource use for quantum error-correcting codes.

APPENDIX A: STEANE'S ERROR CORRECTION FOR ARBITRARY STABILIZER CODES

Here, we describe the error correction used for the leading error correction and trailing error correction of the 5-qubit code. Since the 5-qubit code is not CSS, one might think Steane's error correction is inappropriate. However, in [26], Steane proposes a circuit to do just that for the 5-qubit code. Unfortunately, Steane's construction as written is not quite correct. We present the correct method that works for any stabilizer code. We will also see that this method gives a conceptually simple way to prepare the necessary ancilla state in line with Steane's original proposal [26].

Consider a $[\![n,k]\!]$ stabilizer code $\mathcal{C}$ with stabilizer

$$S = (S_x \mid S_z) \tag{A1}$$

and logical operators

$$N = (N_x \mid N_z), \tag{A2}$$

written in symplectic matrix form. That is, $S_x, S_z \in \mathbb{F}_2^{n-k} \times \mathbb{F}_2^n$ and $N_x, N_z \in \mathbb{F}_2^{2k} \times \mathbb{F}_2^n$. Also, if we define $\Lambda = \begin{pmatrix} 0 & I \\ I & 0 \end{pmatrix} \in \mathbb{F}_2^{2n} \times \mathbb{F}_2^{2n}$ using $k \times k$ block notation, then the canonical commutation relations are expressed by
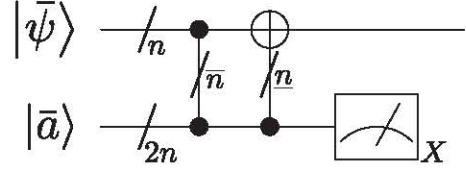

FIG. 5. Circuit for Steane's error correction on a non-CSS code. $|\bar{\psi}\rangle$ is an arbitrary encoded state, and $|\bar{a}\rangle$ is the $2n$-qubit ancilla state from Eq. (A3). The notation $\overline{n}$ and $\underline{n}$ means that the n CZ gates are transversally coupled to the top n qubits in the ancilla and that the n CNOT gates are transversally coupled to the bottom five qubits in the ancilla. Measurement is done transversally, from which the syndrome can be classically computed.

$S\Lambda S^T = S\Lambda N^T$ and $N\Lambda N^T = A$ for the $2k \times 2k$ matrix A with 1s on only the antidiagonal.

Following Steane, we propose the circuit in Fig. 5 to extract the syndrome of $\mathcal{C}$. The ancilla state used is twice the size of the code $\mathcal{C}$. The stabilizer of the ancilla state $|\bar{a}\rangle$ can be written

$$S_a = \begin{pmatrix} S_z & S_x & 0 & S_z \\ 0 & 0 & S_x & S_z \\ 0 & 0 & N_x & N_z \end{pmatrix}. \tag{A3}$$

We show that this ancilla state and the circuit in Fig. 5 successfully extract the syndrome without giving information about the logical operators by propagating the observables of the code $\mathcal{C}$ and the stabilizer S_a through the circuit. Begin with

$$\begin{pmatrix} aS_x & 0 & 0 & S_z & 0 & 0 \\ bN_x & 0 & 0 & N_z & 0 & 0 \\ 0 & S_z & S_x & 0 & 0 & S_z \\ 0 & 0 & 0 & 0 & S_x & S_z \\ 0 & 0 & 0 & 0 & N_x & N_z \end{pmatrix}, \tag{A4}$$

where the syndrome is $a \in \mathbb{F}_2^{n-k}$ and logical operator values are $b = \mathbb{F}_2^{2k}$. After the controlled-Z gates,

$$\begin{pmatrix} aS_x & 0 & 0 & S_z & S_x & 0 \\ bN_x & 0 & 0 & N_z & N_x & 0 \\ 0 & S_z & S_x & S_z & 0 & S_z \\ 0 & 0 & 0 & 0 & S_x & S_z \\ 0 & 0 & 0 & 0 & N_x & N_z \end{pmatrix}. \tag{A5}$$

After the controlled-X gates,

$$\begin{pmatrix} aS_x & 0 & 0 & S_z & S_x & S_z \\ bN_x & 0 & 0 & N_z & N_x & N_z \\ S_x & S_z & S_x & S_z & 0 & 0 \\ 0 & 0 & 0 & 0 & S_x & S_z \\ 0 & 0 & 0 & 0 & N_x & N_z \end{pmatrix}. \tag{A6}$$

This is equivalent to the stabilizer,

$$\begin{pmatrix} aS_x & 0 & 0 & S_z & 0 & 0 \\ bN_x & 0 & 0 & N_z & 0 & 0 \\ a0 & S_z & S_x & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & S_x & S_z \\ 0 & 0 & 0 & 0 & N_x & N_z \end{pmatrix}, \tag{A7}$$

and so we see that measuring all ancilla qubits in the X basis results in a bitstring $m \in \mathbb{F}_2^{2n}$ such that $S\Lambda m = a$.

We note that $|\bar{a}\rangle$ is simply related to a Bell pair $|\Phi\rangle = (|00\rangle + |11\rangle)/\sqrt{2}$ encoded in $\mathcal{C}$. If CX_{tb} denotes n CX gates
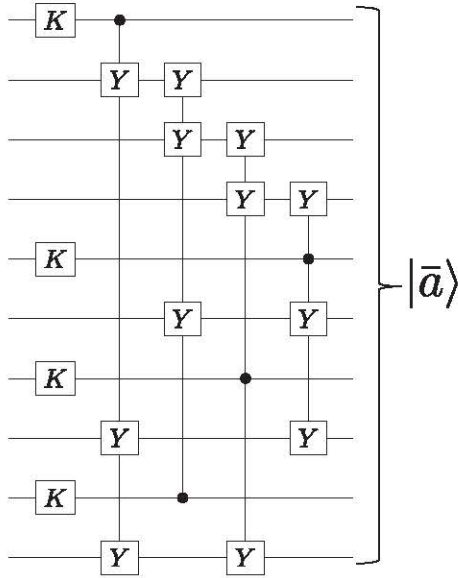
FIG. 6. Preparing the error-correction ancilla state for the 5-qubit code for use in Fig. 5. Input states are all prepared in $|0\rangle$.

transversally acting from the top n qubits of the ancilla to the bottom n and H_t denotes n H gates applied to the top n qubits, then $|\overline{a}\rangle = H_t \mathrm{CX}_{tb}|\overline{\Phi}\rangle$. Thus, we can think of $|\overline{a}\rangle$ as an encoded Bell pair that has been "transversally disentangled." Circuit identities can be used to rearrange Fig. 5 to Knill's error correction [38]. Also, if $\mathcal{C}$ is CSS, Fig. 5 reduces to Steane's error correction for CSS codes [28].

Steane's original proposal for non-CSS error correction [26] omitted the S_z on the right side of the first row of Eq. (A3). Doing the same calculation as above shows that this will not succeed in measuring the syndrome. Steane's proposal suggested that the ancilla state would always be CSS for any code. This, unfortunately, is not true. Indeed, the 7-qubit code from [39] has an ancilla that is not even local-Clifford (LC) equivalent to a CSS state.

However, there are non-CSS codes for which S_a is LC equivalent to a CSS code. The 5-qubit code with stabilizer

$$S_5 = \begin{pmatrix} 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \end{pmatrix} \quad (A8)$$

is one of these. Indeed, S_a can be written using only Y-type and Z-type generators. This allows us to prepare the ancilla using Fig. 6 and verify the ancilla against single-circuit faults using Fig. 7, which are both standard constructions for CSS states [28,40].

APPENDIX B: DETAILS OF THE SIMULATION FOR LOGICAL ERROR ESTIMATION

Here, we describe some techniques used in the estimation of logical error rates. For reasons of simulation efficiency, only errors originating from at most two faults are considered, but all such errors are counted. For Clifford circuits, propagating the Pauli errors resulting from circuit depolarizing noise can be done simply using the Gottesman-Knill theorem [41].
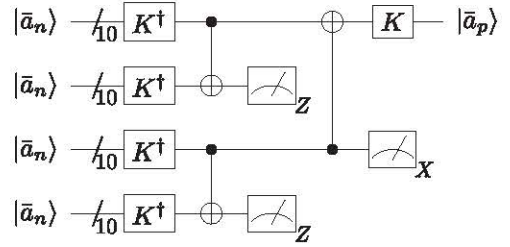

FIG. 7. Verification circuit for the ancilla state prepared by the circuit in Fig. 6. $|\overline{a}_n\rangle$ is a noisy ancilla state that needs to be verified, and $|\overline{a}_p\rangle$ is a purified one.

However, some of our circuits are built from non-Clifford CCZ gates. In this case, a tracked error is modified to include controlled-Z (CZ) terms. A Pauli error that propagates through m CCZ gates picks up at most m CZ terms (some may cancel). Upon measurement (e.g., in the error-correction circuits), the CZ terms must be broken down into a sum of Paulis, only some of which flip measurement bits to cause a signal. We treat each term as a different error element with the probability equal to the square of the amplitude of the term.

There is a subtlety in breaking down CZ errors. As a sum of Pauli terms, a CZ error is written $(II + ZI + IZ - ZZ)/2$. If there are multiple CZ errors, this Pauli sum has every possible combination of I and Z on the qubits on which CZ errors are applied, each with a plus or minus sign. Thus, m CZ errors applied on different qubit pairs are decomposed into a Pauli sum with 4^m terms. If we treated each term as different error element at this point, each term would be assigned the probability square of the amplitude. However, some terms may be equivalent to other terms up to stabilizers. Such terms should interfere coherently. In the simulation of pieceable CCZ on 3×3 Bacon-Shor code, all the terms in the Pauli sum are rewritten in an unambiguous way up to stabilizers, and terms interfere before assigning them a probability.

Now that we recognize the subtle issue as the coherent addition of the Pauli terms, we argue that it does not affect the logical error rate except for that of the 3×3 Bacon-Shor code case. First, note that the coherent addition can only happen when the number of qubits in one block on which CZ errors are applied is more than or equal to the weight of the stabilizers. This is because if the stabilizers have a higher weight, multiplying a stabilizer necessarily gives extra Paulis on the qubits that are not affected by CZ errors. It prevents the term multiplied by a stabilizer being the same as another term in the Pauli sum. In a pieceable CCZ circuit on the 5-qubit code, CZ errors occur only on the three qubits, the support of logical Z. Since stabilizers are weight 4, the coherent addition will not happen for the above reason. For the pieceable CCZ circuit on the 7-qubit code, we argue that although the coherent addition may happen, it will not affect logical error rate. Since stabilizers for the 7-qubit code are weight 4, the coherent addition could happen only when two X errors go through in the same block in the first piece. However, these X errors cannot be corrected because the 7-qubit code is a perfect CSS code. Thus, all the error elements where the coherent addition could happen end up with logical errors regardless, and it does not matter whether we accurately interfere the terms.

For the pieceable CCZ circuit on the 3×9 Bacon-Shor code, the situation is similar to the 7-qubit code case; the coherent addition could happen but will not affect the logical error rate. Since CCZs on the 3×9 Bacon-Shor code are transversal, the number of qubits in the same code block on which CZ errors are applied is at most 2. Since weight-2 Z-gauge operators are aligned along a row, the coherent addition could only happen when two CZ errors are applied on the two qubits in the same row in some code block. However, all the terms in the Pauli sum for the CZ errors in that block are Z-type errors whose weight is less than or equal to 2 and whose support is in the same row. Since weight-1 errors can be corrected by the standard error correction, and weight-2 errors in the same row are equivalent to the identities up to stabilizers for the Z-gauge Bacon-Shor code, the terms in the Pauli sum are all correctable when the concerned coherent addition could happen. Thus, it will not affect the logical error rate.

Considering CZ errors as a Pauli sum is inefficient—m CZ terms lead to 4^m Pauli addends. However, in the simulation, we do not actually break down all the CZ errors. Under certain cases, we definitely know that the final error correction will succeed to correct the CZ error. One such case is when the CZ error is applied over different code blocks, and those code blocks do not have any Z errors. The other case is that the CZ error is applied in one code block, there are no Z errors in the block, and an intermediate error correction notifies the correct locations that the CZ error is applied over.

Also, we can reduce the number of CZ errors by removing harmless CZ errors before the measurements in the final error correction take place. A harmless error is one that does not affect encoded states. When errors are only Paulis, like in the circuits that only consist of Clifford gates, such errors are just stabilizers. The following theorem generalizes the condition for the harmless errors to the non-Pauli case.

Theorem 1: Let E be an error operator, $S = \langle g_1, \ldots, g_{n-k} \rangle$ be the stabilizers, $\langle g_{n-k+1}, \ldots, g_{n+k} \rangle$ be the logical operators of the code, and $|\bar{\psi}\rangle$ be an encoded state. If $g_i^{\dagger} E^{\dagger} g_i E \in S$ for all $i = 1, \ldots, n + k$, then $E|\bar{\psi}\rangle = |\bar{\psi}\rangle$ up to global phase.

Proof. By the assumption, there exists a stabilizer s_l such that $g_i E = E g_i s_l, \forall i$. For $i = 1, \ldots, n - k$, since

$$g_i E |\bar{\psi}\rangle = E g_i s_l |\bar{\psi}\rangle = E |\bar{\psi}\rangle, \tag{B1}$$

E preserves the codeword space. Now for $i = n - k + 1 \ldots n + k$, let $|g_i^{(\pm)}\rangle$ be the eigenstate of the logical operator g_i with eigenvalue ± 1, then

$$g_i E |g_i^{(\pm)}\rangle = E g_i s_l |g_i^{(\pm)}\rangle = \pm E |g_i^{(\pm)}\rangle. \tag{B2}$$

Thus, E also preserves the logical space.

This theorem allows us to ignore the CZ errors that satisfy the above condition, which greatly reduces the computational task.

When intermediate error corrections are present, CZ errors need to be broken down according to the Pauli sum in the intermediate error corrections and need to be propagated until the error correction at the end. If the number of intermediate corrections is zero or 1, it is rather easy to deal with, because the number of error elements due to the CZ errors that need to be propagated until the end is limited. Actually, except for the pieceable 5-qubit code, all the CZ errors that do not satisfy the

condition of Theorem 1 were broken down upon measurement and tracked to see if they end with a logical error.

For the 5-qubit code, to reduce the computational demand, we take the rule where we declare an error to be a logical error as soon as some CZ errors are measured in an intermediate error correction. Although this strategy would cause some overestimation of the logical error rate, we argue that the probability that CZ errors are measured in an intermediate error correction is rather small. CZ errors are measured in an intermediate error correction in the following two cases. The first case is that an X- or Y-type error is caught by a CCZ gate in the adaptive nonconstant-stabilizer measurement. It is described in [19] that the adaptive nonconstant-stabilizer measurement is only triggered when some constant stabilizer measurements click due to an X- or Y-type error only for a single code block. The adaptive measurement may contain CCZ gates connected between the ancilla block and the code blocks whose constant stabilizers did *not* click. Thus, an X- or Y-type error is caught by a CCZ gate in the adaptive measurement only when an X or Y error triggers the adaptive measurement; the constant measurement in different code blocks fail with X- or Y-type error, and it goes to a CCZ gate in the adaptive measurement. The second case is that a CZ error is caught by a CNOT gate in the adaptive measurement. Note that CZ error only happens when an X- or Y-type error propagates through the CCZ gates in the code blocks. CZ errors are then present in code blocks other than the one in which the X- or Y-type error exists. Also, CNOT gates in the adaptive measurement could be applied only to the code block whose constant stabilizers click. Thus, a CZ error is caught by a CNOT gate in the adaptive measurement only when an X- or Y-type error generates CZ errors in different code blocks, a later CCZ gate fails to cancel the first X or Y-type error and generate another X or Y-type error in the other code block that will make the constant measurement click, and the CZ error goes into the CNOT gate in the adaptive measurement. These two cases are realized in very restricted situations, so the contribution to the total logical error rate from these cases would be rather small.

Another situation arises with two or more intermediate corrections. The pieceable construction on the 5-qubit code has multiple intermediate corrections, and they detect X errors and notify possible error locations to the final error correction so that the final error correction can correct up to weight-2 located errors. However, multiple faults can cause two intermediate error corrections to incorrectly notify more than two locations to the final error correction. We declare those elements to be logical errors.

APPENDIX C: DETAILS ON THE ERROR POLYNOMIALS

Here, we describe how to obtain Eqs. (1)–(3) from the exact counting. We first consider Eq. (1), the probability that one or two faults occur and that pattern is accepted by all the verification modules through the propagation but ends up with a logical error. Due to the fault-tolerant property, a single fault never causes a logical error. Thus, it suffices to consider the cases when two faults occur. In the simulation, each combination of two fault patterns is assigned a probability $(\frac{p_r}{4^r - 1})(\frac{p_s}{4^s - 1})$ if the faulty components are an r-qubit gate and an s-qubit gate. We propagate all the errors until the end and

sum up the probabilities of the errors that lead to logical errors. During the propagation, these errors may encounter verification processes. If they are accepted by the verification, we keep propagating them. Otherwise, we stop propagating them so that they do not contribute to the logical error rate. Let $Q_{\text{fail,acc}}$ denote the estimated logical error rate. Since each physical error rate is either $p_1, p_2,$ or p_3, it looks like

$$Q_{\text{fail,acc}} = \sum_{r=1}^{3} \sum_{s \geqslant r}^{3} F_{rs}^{(2)} p_r p_s. \tag{C1}$$

Let n_r be the total number of r-qubit gates. Since we assume that different components fail independently, Eq. (1) is obtained as

$$P_{\text{fail,acc}}^{(2)} = \left[\Pi_{t=1}^{3} (1-p_t)^{n_t} \right] \left(\sum_{r=1}^{3} \sum_{s \geqslant r}^{3} F_{rs}^{(2)} \frac{p_r}{1-p_r} \frac{p_s}{1-p_s} \right). \tag{C2}$$

Similarly, $Q_{\text{succ,acc}}$, the sum of the assigned probability of the patterns that are accepted by all the verification modules and do not cause a logical error, looks like

$$Q_{\text{succ,acc}} = \sum_{r=1}^{3} S_r^{(1)} p_r + \sum_{r=1}^{3} \sum_{s \geqslant r}^{3} S_{rs}^{(2)} p_r p_s \tag{C3}$$

and Eq. (2) is obtained as

$$P_{\text{succ,acc}}^{(2)} = \left[\Pi_{t=1}^{3} (1-p_t)^{n_t} \right] \left(1 + \sum_{r=1}^{3} \frac{S_r^{(1)} p_r}{1-p_r} \right.$$
$$\left. + \sum_{r=1}^{3} \sum_{s \geqslant r}^{3} \frac{S_{rs}^{(2)} p_r p_s}{(1-p_r)(1-p_s)} \right). \tag{C4}$$

The patterns that are not counted in either $Q_{\text{fail,acc}}$ or $Q_{\text{succ,acc}}$ are rejected in some verification module. Thus, we obtain Eq. (3) as

$$P_{\text{rej}}^{(2)} = \left[\Pi_{t=1}^{3} (1-p_t)^{n_t} \right] \left(\sum_{r=1}^{3} \frac{A_r^{(1)} p_r}{1-p_r} \right.$$
$$\left. + \sum_{r=1}^{3} \sum_{s > r}^{3} \frac{A_{rs}^{(2)} p_r p_s}{(1-p_r)(1-p_s)} \right), \tag{C5}$$

where

$$A_r^{(1)} = n_r - S_r^{(1)} \tag{C6}$$

$$A_{rs}^{(2)} = \begin{cases} n_r n_s - F_{rs}^{(2)} - S_{rs}^{(2)} & (r \neq s) \\ \binom{n_r}{2} - F_{rr}^{(2)} - S_{rr}^{(2)} & (r = s) \end{cases}. \tag{C7}$$

Special care is required for the 5-qubit code because n_r cannot be definitely determined because of the adaptive measurements. Note that at most two adaptive measurements are triggered when one or two faults occur. Thus, taking n_r, which includes the two largest adaptive measurements, which are the ones with 13-cat and 9-cat, the lower bound in Eq. (7)

still holds. Instead of Eq. (C4), we take

$$P_{\text{succ,acc}}^{(2)} = \left[\Pi_{t=1}^{3} (1-p_t)^{n_t'} \right] \left(1 + \sum_{r=1}^{3} \frac{S_r^{(1)} p_r}{1-p_r} \right)$$
$$+ \left[\Pi_{t=1}^{3} (1-p_t)^{n_t} \right] \left(\sum_{r=1}^{3} \sum_{s \geqslant r}^{3} \frac{S_{rs}^{(2)} p_r p_s}{(1-p_r)(1-p_s)} \right), \tag{C8}$$

where n_r' is the number of fault locations not including adaptive measurements.

For the 5-qubit code we also use

$$P_{\text{acc}} = \Pi_j P_{\text{acc},j} = \Pi_j (1 - P_{\text{rej},j}) < \Pi_{j'} \left(1 - P_{\text{rej},j'}^{(2)} \right), \tag{C9}$$

where j is taken over all the verification modules and j' is taken over all the verification modules except adaptive measurements.

The following are the obtained values for the parameters for each construction:

(a) 3×3 Bacon-Shor

$$n_1 = 252, n_2 = 180, n_3 = 27 \tag{C10}$$

$$S_r^{(1)} = (252, 180, 27) \tag{C11}$$

$$F_{rs}^{(2)} = \begin{pmatrix} 4216.8 & 4271.9 & 783.5 \\ & 1194.5 & 461.5 \\ & & 34.9 \end{pmatrix} \tag{C12}$$

$$S_{rs}^{(2)} = \begin{pmatrix} 27\,409.2 & 41\,088.1 & 6020.5 \\ & 14\,915.5 & 4398.5 \\ & & 316.1 \end{pmatrix} \tag{C13}$$

(b) Pieceable 7-qubit

$$n_1 = 648, n_2 = 480, n_3 = 21 \tag{C14}$$

$$S_r^{(1)} = (383, 224, 21) \tag{C15}$$

$$F_{rs}^{(2)} = \begin{pmatrix} 13\,258.4 & 12\,722.6 & 3581.4 \\ & 3077.3 & 1855.3 \\ & & 176.7 \end{pmatrix} \tag{C16}$$

$$S_{rs}^{(2)} = \begin{pmatrix} 56\,460.9 & 68\,953.7 & 4461.6 \\ & 20\,748.8 & 2848.7 \\ & & 33.3 \end{pmatrix} \tag{C17}$$

(c) 3×9 Bacon-Shor

$$n_1 = 2736, n_2 = 864, n_3 = 27 \tag{C18}$$

$$S_r^{(1)} = (1524, 566.4, 27) \tag{C19}$$

$$F_{rs}^{(2)} = \begin{pmatrix} 52\,074 & 43\,098.4 & 7049.2 \\ & 8663.0 & 2968.1 \\ & & 183.3 \end{pmatrix} \tag{C20}$$

$$S_{rs}^{(2)} = \begin{pmatrix} 1\,013\,640 & 748\,636 & 34\,098.8 \\ & 138\,296 & 12\,324.7 \\ & & 167.7 \end{pmatrix} \tag{C21}$$

(d) Pieceable 5-qubit

$$n_1 = 3365, n_2 = 1228, n_3 = 41 \tag{C22}$$

$$n_1' = 2967, n_2' = 1152, n_3' = 27 \tag{C23}$$

$$S_r^{(1)} = (1475, 457.6., 27) \tag{C24}$$

$$F_{rs}^{(2)} = \begin{pmatrix} 113\,030.0 & 85\,261.6 & 14\,679.2 \\ & 16\,067.4 & 5551.4 \\ & & 332.5 \end{pmatrix} \tag{C25}$$

$$S_{rs}^{(2)} = \begin{pmatrix} 639\,301.0 & 482\,043.0 & 20\,392.4 \\ & 90\,716.0 & 7554.7 \\ & & 59.3 \end{pmatrix} \tag{C26}$$

(e) 7-qubit with magic state

$$n_1 = 1138, n_2 = 743, n_3 = 14 \tag{C27}$$

$$S_r^{(1)} = (612, 324.3, 6.9) \tag{C28}$$

$$F_{rs}^{(2)} = \begin{pmatrix} 25\,436.5 & 24\,565.9 & 1078.5 \\ & 6232.4 & 521.1 \\ & & 26.9 \end{pmatrix} \tag{C29}$$

$$S_{rs}^{(2)} = \begin{pmatrix} 154\,650 & 166\,625 & 3921.4 \\ & 44\,308.3 & 2178.5 \\ & & 18.6 \end{pmatrix} \tag{C30}$$

APPENDIX D: TRANSFORMATION MATRIX FOR VOLUME CALCULATION

As explained in the main text, the circuit volume for concatenated codes at higher concatenation level is described by a transformation matrix A, where $A_{ij} = N_{G_i}^{G_j}$. We show the matrices for a pieceable 3×3 Bacon-Shor code, pieceable 7-qubit code, and 7-qubit with magic state, which are denoted by A_{pBS}, A_{p7}, and A_{m7}, respectively. We take the following order for gates: $\mathbf{G} = \{$3-qubit gate, 2-qubit gate, single-qubit gate, $|0\rangle$ and $|+\rangle$ preparation, X-basis and Z-basis measurement$\}$. For preparation of $|\bar{0}\rangle$ and $|\bar{+}\rangle$ on 7-qubit code, we use the method proposed by Goto [29], which requires just one additional ancilla:

$$A_{pBS} = \begin{pmatrix} 27 & 90 & 45 & 54 & 54 \\ 0 & 69 & 30 & 36 & 36 \\ 0 & 30 & 24 & 18 & 18 \\ 0 & 6 & 3 & 9 & 0 \\ 0 & 0 & 0 & 0 & 9 \end{pmatrix}$$

$$A_{p7} = \begin{pmatrix} 21 & 162 & 240 & 72 & 72 \\ 0 & 79 & 104 & 32 & 32 \\ 0 & 36 & 59 & 16 & 16 \\ 0 & 11 & 22 & 8 & 1 \\ 0 & 0 & 0 & 0 & 7 \end{pmatrix}$$

$$A_{m7} = \begin{pmatrix} 14 & 267 & 504 & 136 & 136 \\ 0 & 79 & 104 & 32 & 32 \\ 0 & 36 & 59 & 16 & 16 \\ 0 & 11 & 22 & 8 & 1 \\ 0 & 0 & 0 & 0 & 7 \end{pmatrix}$$

APPENDIX E: DETAILED RESOURCE ANALYSIS FOR THE SURFACE CODE

We describe the detailed resource analysis to implement a logical Toffoli gate on the surface code. There are mainly two ways to do it: synthesizing a Toffoli gate using Clifford gates and T gates, and injecting a logical Toffoli state by gate teleportation.

Consider the first method, in the context of the Toffoli implementation proposed by Jones [42] using four T gates. The T gates are implemented by $|T\rangle$ state and gate teleportation, where the $|T\rangle$ state is purified by a distillation protocol. We use the 15-1 protocol [6,22], which reduces error rates of $|T\rangle$ from $O(p)$ to $O(p^3)$, because it requires the smallest circuit volume compared to others [9,43,44]. Since the region of the physical error rate that pieceable construction helps to reduce the error rate is $p < 10^{-4}$ as can be seen in Fig. 1, the logical error rate of the magic state distilled once is $<10^{-12}$. Although the reduction in error rate may not be sufficiently low depending on the goal logical error rate, one distillation already gives large overheads. Thus, we consider the circuit volume for one distillation as a lower bound and proceed with the discussion.

It may come as a surprise that other distillation protocols with better conversion rates between a noisy magic state and a purified magic state have larger circuit volumes. This is because the Hadamard gate and phase gate are not transversal on the surface code. For implementing the Hadamard gate or phase gate fault-tolerantly, some nontrivial techniques, such as state injection, lattice surgery [45], code deformation [46], or surface folding [47], are required. These take many surface code steps, which affect the circuit volume. Even though the conversion rate between the noisy T state and purified T state is high, if it requires many costly Clifford gates, the circuit volume will be large. Especially in the case when only one distillation is required, a poor conversion rate does not hurt the circuit volume that much.

Let us analyze the number of surface code cycles and circuit volume for each gate that are necessary to implement the logical Toffoli gate. Let C_G and V_G be the surface code cycles and circuit volume it takes to implement G. We discuss circuit volume in units of [qubit $\times$ cycle] and then convert it to [qubit $\times$ step] using the fact that one surface code cycle consists of six steps [22]. Also, let d be the surface code distance, and $n = (2d - 1)^2$ be the number of physical qubits on a surface. The necessary components here are $\{|\bar{0}\rangle$ and $|\bar{+}\rangle$ preparation, CNOT, Hadamard, Phase$\}$.

For logical state preparation, we initialize a surface with physical $|0\rangle$ for $|\bar{0}\rangle$ preparation, and $|+\rangle$ for $|\bar{+}\rangle$ preparation. After d rounds of error correction, an appropriate recovery can be determined to prepare the desired logical state fault-tolerantly. Thus, we find $C_{\text{prep}} = d$, $V_{\text{prep}} = nd$.

The CNOT gate can be transversally implemented if we allow nonlocality or a three-dimensionally layered architecture. However, since one of the striking features of surface codes is local interactions in a two-dimensional architecture, we use lattice surgery to implement the CNOT gate [45]. First, prepare a surface with the $|\bar{+}\rangle$ state between the control surface and the target surfaces. The control surface and the intermediate surface are merged while obtaining measurement syndromes. This corresponds to $\bar{Z}\bar{Z}$ measurement. After that, the surface
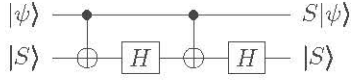
FIG. 8. Circuit identity used for implementing the S gate. Here $|S\rangle = S|+\rangle = (|0\rangle + i|1\rangle)/\sqrt{2}$.

is split into two original surfaces and the intermediate surface is merged to the target surface, which corresponds to $\bar{X}\bar{X}$ measurement. It ends with splitting it into the two original surfaces. Since merger and splitting each take d rounds of error correction to stabilize the surface,

$$C_{\text{CNOT}} = C_{\text{prep}} + 4d = 5d \qquad \text{(E1)}$$

and

$$\begin{aligned} V_{\text{CNOT}} &= V_{\text{prep}} + [3n + 2(2d-1)](C_{\text{prep}} + 4d) \\ &= 6d - 44d^2 + 64d^3. \qquad \text{(E2)} \end{aligned}$$

The Hadamard gate is also implemented by the lattice surgery. In the lattice surgery technique, first Hadamard gates are applied transversally. To correct the orientation of the boundary, additional qubits are merged to the boundary and some qubits are split out so that it restores the original boundary orientation. The protocol ends with moving the surface back to the original position. It takes d cycles to stabilize the original surface after applying transversal H, d cycles for lattice merger, d cycles for lattice splitting, and d cycles for SWAP operations to move the lattice back to the original position. Thus, $C_H = 4d$. For circuit volume, we need a bigger surface to carry out merger and split by one more column and row of qubits. Thus, $V_H = (2d)^2 C_H = 16d^3$.

For implementing the phase gate, we use the circuit in Fig. 8. A good thing about this circuit is that the ancilla state $|S\rangle = S|+\rangle = (|0\rangle + i|1\rangle)/\sqrt{2}$ is preserved. Thus, once a purified $|S\rangle$ state is prepared at the beginning of the computation, it can be reused whenever a phase gate needs to be applied. After averaging over a whole computation, the volume used for the distillation process at the beginning will be negligible per one logical gate construction. Note that if only local interactions are allowed, it may take additional circuit volume when the qubit to which the phase gate should be applied is far from the stored $|S\rangle$ state. Thus, our estimation should be considered as a lower bound of the actual circuit volume under the setting in which only local interactions are allowed. It gives $C_S = 2C_{\text{CNOT}} + 2C_H = 18d$ and $V_S = 2V_{\text{CNOT}} + 2V_H + 2nC_H = 20d - 120d^2 + 192d^3$. Combining these building blocks, we find the number of cycles and volume required to implement a T gate and a Toffoli gate.

For distilling a T state, $|T\rangle = T|+\rangle$, we use the circuit in [22] which takes 15 $|T\rangle$ states and output 1 $|T\rangle$ with lower error rate. It takes seven surface code cycles for CNOTs and two steps for transversal T and measurements, which is 1/4 surface code cycle. Ignoring the last 1/4 cycles, we get $C_{|T\rangle} = 7C_{\text{CNOT}} = 35d$. With some parallelization, we get $V_{|T\rangle} = 16V_{\text{prep}} + V_{\text{CNOT}} + 7(5V_{\text{CNOT}} + 6nC_{\text{CNOT}}) + \frac{1}{4}(16nd) = 446d - 2504d^2 + 3224d^3$.

For implementing a T gate, we use the usual gate teleportation technique [48]. The S gate correction is applied with probability 1/2. We get $C_T = C_{\text{CNOT}} + \frac{1}{2}C_S = 14d$ and

$V_T = V_{|T\rangle} + V_{\text{CNOT}} + \frac{1}{2}V_S = 462d - 2608d^2 + 3384d^3$. Since the surface code is CSS, we can transversally make measurements on all the data qubits and extract the eigenvalue for the measurement operator. Thus, measurement is done with only one time step, which is 1/8 of one surface code cycle, and we ignore the volume due to the measurement.

Toffoli gate synthesis in [42] consists of two steps. In the first part, one constructs the Toffoli* gate, which is a Toffoli gate followed by a controlled-$S^\dagger$ gate, where four T gates and two H gates are used. Also, note that one logical ancilla block is used. The second part takes the Toffoli* gate to the usual Toffoli gate with help of one additional ancilla block. By construction of the synthesis circuit, we get

$$C_{\text{Toffoli}^*} = 2C_H + 4C_{\text{CNOT}} + C_T = 42d \qquad \text{(E3)}$$

and

$$\begin{aligned} V_{\text{Toffoli}^*} &= V_{\text{prep}} + 6nC_H + 2V_H + 8V_{\text{CNOT}} + 4V_T \\ &= 1921d - 10884d^2 + 14180d^3. \qquad \text{(E4)} \end{aligned}$$

The second part of the circuit gives

$$C_{\text{Toffoli}} = C_{\text{Toffoli}^*} + C_S + C_{\text{CNOT}} + C_H + C_{|T\rangle} = 104d \qquad \text{(E5)}$$

and

$$\begin{aligned} V_{\text{Toffoli}} &= V_{\text{Toffoli}^*} + V_{\text{prep}} + nC_{\text{Toffoli}^*} + V_S + V_{\text{CNOT}} \\ &\quad + V_H + 3n(C_S + C_{\text{CNOT}} + C_H) \\ &\quad + n(C_{\text{Toffoli}^*} + C_{\text{CNOT}} + C_H) \\ &= 2118d - 11732d^2 + 15136d^3, \qquad \text{(E6)} \end{aligned}$$

where the unit for the volume is [qubit $\times$ cycle]. We included $C_{|T\rangle}$ in C_{Toffoli} because cycles in the distillation circuit also contribute to an increase in the final logical error rate. Note that it includes the ancilla qubits for keeping the $|S\rangle$ state that is kept during the whole computation.

Another way to implement a logical Toffoli gate on the surface code is to use the Toffoli state. To locally prepare the Toffoli state, we use the protocol that takes eight $|H\rangle$ states and outputs one Toffoli state [49]. In the preparation circuit, there are two $Y(\pi/4)$ gates and four $Y(-\pi/4)$ gates, which are rotations with respect to the Y axis. These gates are implemented using the $|H\rangle$ state with a Y basis measurement, controlled-Y gate, and $Y(\pm\pi/2)$ gate. To implement these gates on the surface code, we use phase gates and Hadamard gates to rotate them to X basis measurement, CNOT gate, and phase gate. We then obtain

$$\begin{aligned} C_{Y(\pi/4)} &= C_S + C_{\text{CNOT}} + C_S + (2C_H + C_S)/2 \\ &= 54d \qquad \text{(E7)} \end{aligned}$$

and

$$\begin{aligned} V_{Y(\pi/4)} &= V_{\text{prep}} + V_S + V_{\text{CNOT}} + 2V_S + (2V_H + V_S)/2 \\ &= 77d - 468d^2 + 756d^3. \qquad \text{(E8)} \end{aligned}$$

Using these, we obtain

$$\begin{aligned} C_{|\text{Toffoli}\rangle} &= 7C_{\text{CNOT}} + (15/2)C_S + 3C_H \\ &= 182d \qquad \text{(E9)} \end{aligned}$$
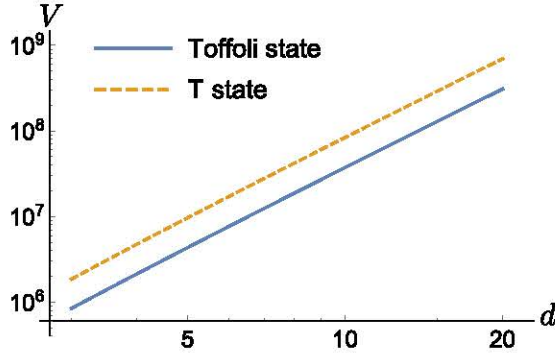
FIG. 9. Circuit volume for two different implementations of the Toffoli gate. Dashed: gate synthesis using a T gate. Solid: Toffoli state scheme.

and

$$V_{|\text{Toffoli}\rangle} = 4V_{\text{prep}} + 3(2V_{\text{CNOT}} + 2V_{Y(\pi/4)} + 2nC_{Y(\pi/4)})$$
$$+ V_{\text{CNOT}} + 2nC_{\text{CNOT}}$$
$$= 842d - 4468d^2 + 6336d^3, \tag{E10}$$

where $|\text{Toffoli}\rangle$ refers to the Toffoli state. Cycles and volume for the teleportation circuit, which we write C_{tele} and V_{tele}, are

$$C_{\text{tele}} = C_{\text{CNOT}} + 1/2(3C_{\text{CNOT}} + 2C_H)$$
$$= 16.5d, \tag{E11}$$

$$V_{\text{tele}} = 3V_{\text{CNOT}} + \{4nC_H + 3(V_{\text{CNOT}} + nC_{\text{CNOT}})\}/2$$
$$= 42.5d - 260d^2 + 350d^3. \tag{E12}$$

Combining all of them, we get

$$C_{\text{Toffoli}} = 198.5d \tag{E13}$$

and

$$V_{\text{Toffoli}} = 7076d - 37824d^2 + 53488d^3, \tag{E14}$$

where the unit for the volume is [qubit $\times$ cycle]. Figure 9 shows circuit volume with unit [qubit $\times$ step] in terms of code distance for both ways of implementation. We can see that the scheme with the Toffoli state has a lower circuit volume. This is the reason why the scheme with the Toffoli state is discussed in the main text.

[1] B. Zeng, A. Cross, and I. L. Chuang, IEEE Trans. Inf. Theory **57**, 6272 (2011).
[2] X. Chen, H. Chung, A. W. Cross, B. Zeng, and I. L. Chuang, Phys. Rev. A **78**, 012353 (2008).
[3] B. Eastin and E. Knill, Phys. Rev. Lett. **102**, 110502 (2009).
[4] S. Bravyi and R. König, Phys. Rev. Lett. **110**, 170503 (2013).
[5] D. Gottesman and I. L. Chuang, Nature (London) **402**, 390 (1999).
[6] S. Bravyi and A. Kitaev, Phys. Rev. A **71**, 022316 (2005).
[7] C. Jones, Phys. Rev. A **87**, 042305 (2013).
[8] E. T. Campbell and M. Howard, Phys. Rev. A **95**, 022316 (2017).
[9] S. Bravyi and J. Haah, Phys. Rev. A **86**, 052329 (2012).
[10] A. Paetznick and B. W. Reichardt, Phys. Rev. Lett. **111**, 090505 (2013).
[11] J. T. Anderson, G. Duclos-Cianci, and D. Poulin, Phys. Rev. Lett. **113**, 080501 (2014).
[12] H. Bombín, New J. Phys. **17**, 083002 (2015).
[13] T. Jochym-O'Connor and R. Laflamme, Phys. Rev. Lett. **112**, 010505 (2014).
[14] E. Nikahd, M. Sedighi, and M. S. Zamani, arXiv:1605.07007.
[15] E. Nikahd, M. S. Zamani, and M. Sedighi, arXiv:1610.03309.
[16] C. Chamberland, T. Jochym-O'Connor, and R. Laflamme, Phys. Rev. Lett. **117**, 010501 (2016).
[17] C. Chamberland, T. Jochym-O'Connor, and R. Laflamme, Phys. Rev. A **95**, 022313 (2017).
[18] K. M. Svore, A. W. Cross, I. L. Chuang, and A. V. Aho, Quantum Inf. Comput. **6**, 193 (2006).
[19] T. J. Yoder, R. Takagi, and I. L. Chuang, Phys. Rev. X **6**, 031039 (2016).
[20] R. Chao and B. W. Reichardt, arXiv:1705.05365.
[21] T. J. Yoder, arXiv:1705.01686.
[22] A. G. Fowler, M. Mariantoni, J. M. Martinis, and A. N. Cleland, Phys. Rev. A **86**, 032324 (2012).

[23] P. Aliferis, D. Gottesman, and J. Preskill, Quantum Inf. Comput. **6**, 97 (2006).
[24] L. S. Bishop, S. Bravyi, A. Cross, J. M. Gambetta, and J. Smolin (2017), https://dal.objectstorage.open.softlayer.com/v1/AUTH_039c3bf6e6e54d76b8e66152e2f87877/community-documents/quatnum-volumehp08co1vbo0cc8fr.pdf.
[25] R. Takagi, T. Yoder, and I. Chuang, Circuit database, https://github.com/RyujiPat/error_resource_TYC.
[26] A. M. Steane, Phys. Rev. Lett. **78**, 2252 (1997).
[27] P. W. Shor, in *Proceedings of the 37th Annual Symposium on Foundations of Computer Science*, FOCS '96 (IEEE Computer Society, Washington, DC, 1996), p. 56.
[28] A. Steane, Fortschr. Phys. **46**, 443 (1998).
[29] H. Goto, Sci. Rep. **6**, 19578 (2016).
[30] X. Zhou, D. W. Leung, and I. L. Chuang, Phys. Rev. A **62**, 052316 (2000).
[31] C. Monroe, R. Raussendorf, A. Ruthven, K. R. Brown, P. Maunz, L.-M. Duan, and J. Kim, Phys. Rev. A **89**, 022317 (2014).
[32] A. G. Fowler, A. M. Stephens, and P. Groszkowski, Phys. Rev. A **80**, 052312 (2009).
[33] D. S. Wang, A. G. Fowler, A. M. Stephens, and L. C. L. Hollenberg, Quantum Inf. Comput. **10**, 456 (2010).
[34] D. S. Wang, A. G. Fowler, and L. C. L. Hollenberg, Phys. Rev. A **83**, 020302(R) (2011).
[35] J. R. Wootton and D. Loss, Phys. Rev. Lett. **109**, 160503 (2012).
[36] D. Poulin, Phys. Rev. A **74**, 052333 (2006).
[37] J. Fern, Phys. Rev. A **77**, 010301(R) (2008).
[38] E. Knill, Nature (London) **434**, 39 (2005).
[39] T. J. Yoder and I. H. Kim, Quantum **1**, 2 (2017).
[40] A. W. Cross, D. P. Divincenzo, and B. M. Terhal, Quantum Inf. Comput. **9**, 541 (2009).
[41] D. Gottesman, in *Group22: Proceedings of the XXII International Colloquium on Group Theoretical Methods in Physics* (International Press, Cambridge, MA, 1999), pp. 32–43.

[42] C. Jones, Phys. Rev. A **87**, 022328 (2013).

[43] A. M. Meier, B. Eastin, and E. Knill, Quantum Inf. Comput. **13**, 195 (2013).

[44] B. W. Reichardt, Quant. Inf. Process. **4**, 251 (2005).

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

[46] H. Bombin and M. A. Martin-Delgado, J. Phys. A **42**, 095302 (2009).

[47] J. E. Moussa, Phys. Rev. A **94**, 042316 (2016).

[48] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information* (Cambridge University Press, Cambridge, MA, 2010).

[49] B. Eastin, Phys. Rev. A **87**, 032321 (2013).