

Universal Fault-Tolerant Gates on Concatenated Stabilizer Codes

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

Massachusetts Institute of Technology, Cambridge, Massachusetts 02139, USA

(Received 12 March 2016; published 13 September 2016)

It is an oft-cited fact that no quantum code can support a set of fault-tolerant logical gates that is both universal and transversal. This no-go theorem is generally responsible for the interest in alternative universality constructions including magic state distillation. Widely overlooked, however, is the possibility of nontransversal, yet still fault-tolerant, gates that work directly on small quantum codes. Here, we demonstrate precisely the existence of such gates. In particular, we show how the limits of nontransversality can be overcome by performing rounds of intermediate error correction to create logical gates on stabilizer codes that use no ancillas other than those required for syndrome measurement. Moreover, the logical gates we construct, the most prominent examples being Toffoli and controlled-controlled-Z, often complete universal gate sets on their codes. We detail such universal constructions for the smallest quantum codes, the 5-qubit and 7-qubit codes, and then proceed to generalize the approach. One remarkable result of this generalization is that any nondegenerate stabilizer code with a complete set of fault-tolerant single-qubit Clifford gates has a universal set of fault-tolerant gates. Another is the interaction of logical qubits across *different* stabilizer codes, which, for instance, implies a broadly applicable method of code switching.

DOI: [10.1103/PhysRevX.6.031039](https://doi.org/10.1103/PhysRevX.6.031039)

Subject Areas: Quantum Information

I. INTRODUCTION

One of the crucial concepts in error-correcting codes is that of logical circuits—a set of circuits $\{C_i\}$ that give the ability to carry out a set of operations $\{U_i\}$ directly on encoded data, rather than performing the risky procedure of decoding, applying U_i , and reencoding. In fact, the latter procedure is forbidden if we insist on each C_i being a fault-tolerant logical circuit, for which the failure of any one component in C_i never leads to an uncorrectable error on the encoded data. If, in addition, the set $\{U_i\}$ is universal for the computational model in question (e.g., classical or quantum computation), the set of logical circuits $\{C_i\}$ is said to be universal, and the error-correcting code in question could, in principle, be used for all computational purposes without ever needing to decode, an essential ability for quantum computing especially, where decoherence remains the bane of all practical implementations of quantum algorithms.

In the quantum computational model, the one paradigm for designing fault-tolerant logical gates that is most preferred is transversality. A logical circuit is transversal if all physical qubits have interacted with at most one physical qubit from each code block, and if such a design preserves the code space, it is automatically fault tolerant. Indeed, many quantum codes, such as Steane's 7-qubit code [1], are highly regarded exactly because

they have important, and perhaps several important, transversal gates.

Unfortunately, it is a well-known theorem [2–4] that there is no quantum code with a universal set of transversal logical circuits. This fact means other methods must be used to perform universal, fault-tolerant quantum computation. The most common approach is that of magic states [5,6], encoded ancilla qubits that, combined with available transversal circuits (usually implementing Clifford operations), serve to complete a universal set of logical circuits (usually by implementing a T or Toffoli gate). Ideally, these magic states would be efficiently constructible themselves, but current so-called distillation procedures actually incur large overheads in terms of time and qubits [7,8].

It is, therefore, fortunate that other approaches to bypass the universal-transversal no-go theorem exist. Some involve code switching—the transversal circuits on two codes together might complete a universal set, encouraging development of a method to exchange data between the two code spaces. Simple procedures have been devised for conversions between specific codes, such as between the 5-qubit and 7-qubit [9] and between quantum Reed-Muller codes [10]. Another workaround for the no-go theorem uses triorthogonal subsystem codes [11], the smallest of which is 15-qubit, to implement a universal, transversal set of gates without code switching, but with additional error correction (EC) on the gauge qubits. Related gauge-fixing techniques are employed in topological color codes [12], where getting locally implementable non-Clifford gates requires jumping up a dimension to 3D [13]. Lastly, concatenated coding combines two codes with

Published by the American Physical Society under the terms of the Creative Commons Attribution 3.0 License. Further distribution of this work must maintain attribution to the author(s) and the published article's title, journal citation, and DOI.

complementary transversal gate sets into a single large code with universal, yet not transversal, gates [14].

Here, we complement the myriad previous approaches to universal fault tolerance by developing nontransversal, yet still fault-tolerant, circuits implementing logical gates on stabilizer codes. Our approach is based on an underappreciated trick first used by Knill, Laflamme, and Zurek [15] to implement a fault-tolerant controlled- S gate on the 7-qubit code. The trick revolves around breaking a nontransversal circuit into fault-tolerant pieces, with error correction performed in between to correct errors before they propagate too badly throughout the nontransversal circuit. We show that this trick for creating fault-tolerant logical gates can be significantly generalized to perform other logical gates on other codes and ultimately developed into a procedure we call pieceable fault tolerance.

As examples of pieceable fault tolerance, we create a logical controlled- Z (CZ) gate on the 5-qubit code (Sec. III) and logical controlled-controlled- Z (CCZ) gates on the 5-qubit (Sec. III) and 7-qubit (Sec. IV) codes, completing universal sets of gates on those codes. Also notable is that our circuits use no ancillas other than those required for the multiple rounds of error correction—in particular, we use no magic states. Indeed, in Sec. IV, we find that our construction for the 7-qubit code compares favorably against magic-state injection with regards to resources required. All our pieceable circuits are 1-fault-tolerant. That is, any *one* fault does not cause a logical error, as is consistent with both of these small codes being distance three. Through concatenation, these pieceable circuits also possess a fault-tolerance threshold in the usual sense [16,17].

In addition, we provide sufficient conditions for similar pieceable circuits to work on larger codes more generally (Sec. V). We find that nondegeneracy or, more specifically, not having weight-two stabilizers, is sufficient (albeit not necessary) for a code to have a pieceably 1-fault-tolerant gate. Therefore, any nondegenerate stabilizer code with a universal set of fault-tolerant *local* (i.e., single-qubit) Clifford logical gates can be promoted to fault-tolerant universality using our pieceable methods. Compare to magic states, where to achieve the same universality for distance-three codes, even with much more overhead, the entire set of logical Cliffords is required [5]. Moreover, we show nondegenerate CSS [1,18] codes just need *any* fault-tolerant local Clifford to achieve the same universality. Finally, we find that pieceable fault tolerance can also perform gates between different codes, and thus act as a quite general method of code switching.

Noteworthy ideas in the literature closely related to pieceable fault tolerance are the code-switching example of Hill *et al.* [9] and code deformations of Bombin and Martin-Delgado [19,20]. In the former, a circuit of Clifford gates with error correction performed after each gate is used to switch between the 5-qubit and 7-qubit codes. In the

latter, logical initialization, measurement, and Clifford gates are performed on surface codes by manipulating the geometry of the surface while correcting errors as they arise. Both techniques are similar to pieceable fault tolerance as the code undergoes transformations to several intermediate codes, each with distance large enough to correct any errors that may have arisen. However, both also do not generate logical universality on their own, something we show pieceable fault tolerance can indeed provide.

Achieving this universality also requires fundamentally new tools. For instance, we develop original circuit designs, called round-robin constructions, to perform our logical gates. Additionally, we create an adaptive procedure for error correction on the nonstabilizer intermediate codes we encounter. The profitable use of nonstabilizer codes is perhaps novel and interesting in and of itself. A fault-tolerance overview, the definition of pieceable fault tolerance, and a summary of these new tools are the goals of the next section.

II. DESIGN METHODOLOGY FOR FAULT-TOLERANT LOGICAL GATES

Quantum codes operate by the “fight entanglement with entanglement” [21] mantra—to protect sensitive data from a noisy, nosy environment, introduce additional degrees of freedom, and encode the data in globally entangled states. Local errors have no chance of rearranging the long-range entanglement to affect the data, and so our information is secure.

However, if we want to legitimately alter the data, this same security becomes a hassle. To perform quantum gates on the encoded data, we are forced to create circuits manipulating globally entangled states, and, moreover, these circuits must not corrupt the data, even if they themselves are faulty. In this section, we overview this design challenge, culminating in our solution, pieceably fault-tolerant logical gates.

A. Codes, logical operations, and error correction

An $[[n, k]]$ quantum code L using n physical qubits to encode k data qubits is essentially a collection of 2^k orthogonal n -qubit code states $\{|\bar{0}\rangle, |\bar{1}\rangle, \dots, |\bar{2^k-1}\rangle\}$. The code space \mathcal{C}_L is the span of the code states. In this sense, quantum codes are nothing more than a strange basis for a subspace of a larger Hilbert space.

However, to be practically useful the code states must also satisfy certain error-correcting conditions [22]. Not least of all, the code states should be far enough separated that errors localized to only a few physical qubits cannot undetectably change any code states. Also, it would be preferable if such error channels could be not only detected but reversed, and the code space restored to normal.

Stabilizer codes [23–25] were the first broad class of quantum codes that could be designed with such

error-correcting properties, and they are still the most common, and promising, type of quantum code today. Stabilizer codes are built on the foundation of the Pauli group. The Pauli operators on a single qubit are the familiar $\sigma_0 = I$, $\sigma_1 = X$, $\sigma_2 = Y$, $\sigma_3 = Z$. The Pauli group on n qubits \mathcal{P}_n is made of tensor products of these,

$$\mathcal{P}_n = \left\{ i^a \bigotimes_{j=1}^n \sigma_{h_j} : a, h_j \in \{0, 1, 2, 3\} \right\}. \quad (1)$$

As for notation, given a Pauli operator $p \in \mathcal{P}_n$, we denote the number of nonidentity members of the tensor product as the weight $|p|$ and the set of qubits on which p acts nonidentically as the support $\text{supp}(p)$.

A subgroup of \mathcal{P}_n , called the stabilizer group S , defines the code space \mathcal{C}_L of a stabilizer code as the $+1$ eigenspace of all Pauli operators in S . In order for this to be nontrivial, we must have $-I^{\otimes n} \notin S$, which is equivalent to two conditions: first, S is Abelian, and second, all $g \in S$ have signs ± 1 (no sign of $\pm i$). Because every Pauli operator has only ± 1 eigenspaces of equal size, we evidently need $n - k$ independent Pauli operators to generate S and reduce the 2^n -dimensional Hilbert space of n qubits to $2^k = 2^n / 2^{n-k}$ dimensional, the size of the code space. These generators are not unique, but, nevertheless, we label (a canonical choice of) them $\tilde{Z}_1, \tilde{Z}_2, \dots, \tilde{Z}_{n-k}$ and write

$$S = \langle \tilde{Z}_1, \tilde{Z}_2, \dots, \tilde{Z}_{n-k} \rangle, \quad (2)$$

where $\langle \cdot \rangle$ indicates a list of generators rather than a list of all group elements.

Yet, this cannot be all there is to a stabilizer code, because only the code space, and not the code states, has so far been defined. To complete the stabilizer code, choose real-signed Pauli operators $\tilde{Z}_1, \dots, \tilde{Z}_k$ and $\tilde{X}_1, \dots, \tilde{X}_k$ from \mathcal{P}_n that are independent from, yet commute with, both the stabilizer generators and each other. The only exception is for \tilde{Z}_i and \tilde{X}_i , which should anticommute for each i . Since any set of n independent Pauli operators from \mathcal{P}_n have a unique state in their simultaneous $+1$ eigenspace, the full group $\langle \tilde{Z}_1, \dots, \tilde{Z}_{n-k}, \tilde{Z}_1, \dots, \tilde{Z}_k \rangle$ defines a state $|\bar{0}\rangle$. The remaining encoded basis states are given by $|\bar{b}\rangle = \prod_j \tilde{X}_j^{b_j} |\bar{0}\rangle$ for all n -bit strings b . Evidently, the \tilde{X}_j operators act as encoded, or logical, X gates and the \tilde{Z}_j operators act as logical Z gates. Logical Y gates are given by $\tilde{Y}_j = i\tilde{X}_j\tilde{Z}_j$.

However, there is some freedom in choice of logical operator even now. Indeed, any member of the stabilizer coset $\tilde{Z}_i S$ acts exactly as \tilde{Z}_i does on the code space, and likewise with the cosets $\tilde{X}_i S$ and $\tilde{Y}_i S$ compared with \tilde{X}_i and \tilde{Y}_i . The union of these $3k$ cosets and of the stabilizer itself forms the normalizer:

$$\mathcal{N}(S) = \langle \tilde{Z}_1, \dots, \tilde{Z}_{n-k}, \tilde{Z}_1, \dots, \tilde{Z}_k, \tilde{X}_1, \dots, \tilde{X}_k \rangle. \quad (3)$$

The lowest weight element of $\mathcal{N}(S) \setminus S$ has weight equal to the code distance d , which can be conveniently thought of as the fewest number of qubits which need to be acted upon to undetectably alter the code states. The double bracket notation for quantum codes is often extended from $[[n, k]]$ to include the code distance as in $[[n, k, d]]$.

Nevertheless, logical versions of the Pauli operators are not enough to perform arbitrary operations on encoded data. We should define logical versions of other unitary gates, say U , as well. Since the Pauli group \mathcal{P}_k is a complete basis for the k -qubit unitaries, we can decompose $U = \sum_{p_j \in \mathcal{P}_k} a_j p_j$ for some $a_j \in \mathbb{R}$. A logical version of U , call it \bar{U} , should be similarly decomposed as $\bar{U} = \sum_{p_j \in \mathcal{P}_k} a_j \bar{p}_j s_j$, where $s_j \in S$ and \bar{p}_j is the logical version of p_j wherein each occurrence of X_i , Y_i , or Z_i is replaced by \tilde{X}_i , \tilde{Y}_i , or \tilde{Z}_i , respectively. The freedom of the stabilizer is present in the form of the arbitrary s_j . Slightly more generally, we define a logical operation as any operator \mathcal{U} that preserves the code space—that is, $\mathcal{U}(|\bar{\psi}\rangle) \in \mathcal{C}_L$ for all $|\bar{\psi}\rangle \in \mathcal{C}_L$.

Where does error correction fit in this picture of code states and code spaces? Well, one thing an (EC) operation should not do is change code states. That is, $\text{EC}(|\bar{\psi}\rangle) = |\bar{\psi}\rangle$ for all $|\bar{\psi}\rangle \in \mathcal{C}_L$. At the risk of overgeneralization, we actually take this condition as our definition—an error-correction operation is any operation that preserves all code states. Note that even the trivial $I^{\otimes n}$ is an error-correction operation under this definition. We define error correction this way because it makes our later definition of pieceable fault tolerance most general.

We call the usual notion of error correction complete error correction $\overline{\text{EC}}$. In this case, for any Pauli error E (note that all error channels can be decomposed into Pauli errors [26]) of weight less than $d/2$, $\overline{\text{EC}}(E|\psi\rangle) = |\psi\rangle$. There are several known methods for complete error correction on quantum codes, including methods due to Shor [27], Steane [28,29], and Knill [30]. We assume that every stabilizer code L comes equipped with a canonical method of complete error correction and denote it $\overline{\text{EC}}_L$.

B. Circuits, faults, and fault tolerance

It is important to distinguish between what a logical operation or error-correction does, which we define in the previous section as code space preserving or code state preserving, respectively, from its implementation, a circuit. Circuits are constructed from basic operations on physical qubits, including gates, $|0\rangle$ state preparation, and single-qubit measurement. The collection of allowed physical operations, or components, is denoted A_P . A circuit C then is nothing more than an ordered list of these physical operations, $C = \{c_m, c_{m-1}, \dots, c_1\}$. If the sequential action of these components, from right to left, performs a logical unitary gate, the circuit can be called a logical circuit, and if it performs error correction, it can be called

an error-correction circuit. To be most realistic, we could insist on dividing a circuit into several time steps and require each qubit to be acted upon during each time step, even if it is only by an identity operation. For most purposes, we treat circuits as operators and combine them with multiplication, $C_2 \cdot C_1$, with C_1 applied before C_2 .

The fundamental problem we face in quantum error correction is that the physical components of any circuit can be faulty and introduce errors to the qubits they act upon. The standard model of faulty components, used in essentially every discussion of fault tolerance [15,17,25,31], restricts the way that a component $c \in A_p$ can fail by saying that any faulty version of c can be modeled as c followed by an arbitrary error channel acting on the qubits in the support of c . A correlated error between different qubits therefore arises only when those qubits are explicitly coupled by a physical component. It is an interesting and important fact that these analoglike error channels can be “digitized” [26]. That is, from the perspective of an error-correcting code, any error channel can be viewed as the random application of an (unintended) Pauli operator. In turn, these Pauli operators can be decomposed into single-qubit Pauli errors. For example, we already discussed that any complete error correction of a distance d quantum code can correct $t < d/2$ single-qubit errors.

The solution to the problem of faulty components is to design fault-tolerant circuits. Such circuits are created with a quantum code L with a complete error-correction operator \overline{EC}_L in mind. We say a logical circuit C is fault tolerant (with respect to L) if any single faulty component in the combined circuit $C \cdot \overline{EC}_L$ creates only errors correctable by an ideal version of \overline{EC}_L performed afterward. Whether the circuit is fault tolerant or not might be said to be judged by this final ideal \overline{EC}_L . Since errors arising from faulty components in the middle of $C \cdot \overline{EC}_L$ must be propagated to the end of the circuit before being corrected, it is crucial that components of C do not couple too many qubits within a single code block so that the spread of errors is limited. Also, in correspondence with the exREC formalism [17], it is important to include the leading complete error correction, because input errors to C arise from faults in the previous round of error correction.

The definitions of fault tolerance and of faulty components suggest the following sufficient design methodology for logical gates—a logical circuit is evidently 1-fault-tolerant if all qubits interact (either directly or indirectly) with fewer than t qubits in any other code block and fewer than $t - 1$ qubits in their own blocks and $t < d/2$. In this case, called t transversality [14], the standard model of faulty components implies that any one code block cannot accumulate more than t single-qubit errors assuming only a single faulty component. For $d = 3$ codes, the essential concept is 1-transversality, a property that most current logical circuit designs possess.

However, it can be proved that transversality is not enough—for no quantum code can we create a quantum universal set of logical gates with transversal circuits alone [3,4]. This fact has prompted other techniques of logical circuit design to complete a universal logical gate set. Some of the largest breakthroughs have been universality with transversal Clifford gates and magic ancillas [5], with a transversal gate set but with a gauge qubit reset [11], with nontransversal constructions on large codes built as concatenations of smaller codes with complementary transversal gate sets [14], and with code-switching techniques [9,10] to switch between codes with complementary transversal gate sets.

C. Pieceable fault tolerance

We now define our method to provide universality without transversality. We are motivated by a simple idea: even if an entire logical circuit is not fault tolerant, parts of the circuit may be. If we perform error correction partway through the circuit, perhaps we can quell the propagation of errors before they become uncorrectable. This idea was used before in the fault-tolerant code-switching technique of Hill *et al.* [9], and here we provide a generalization, one that succeeds in making even some non-Clifford circuits fault tolerant.

To make this idea precise, begin with a logical circuit C for some logical gate for a quantum code L . Decompose this logical circuit into pieces $C = C_m \cdot C_{m-1} \dots C_1$. After each partial logical gate C_i , there is still a code space and code states, and if L was a stabilizer code there is still a stabilizer and logical Pauli operators (though each may be non-Pauli if C_i is non-Clifford). Thus, we can insert code state preserving operations, error corrections EC_i , after each piece without altering the logical effect of C . We now have a modified circuit:

$$\tilde{C} = EC_m \cdot C_m \cdot EC_{m-1} \cdot C_{m-1} \dots EC_1 \cdot C_1. \quad (4)$$

If such a modified circuit can be found and \tilde{C} is fault tolerant in the traditional sense, then we say that C is pieceably fault tolerant (in m pieces). We also make a distinction between any ancilla qubits used during the circuit pieces C_i , called functional ancillas, and those used during EC_i , called error-correction ancillas. All of our constructions use no functional ancillas, although pieceable constructions with more pieces will obviously use more error-correction ancillas than a nonpieceable (i.e., $m = 1$) logical gate.

A few things should be clarified about the definition. First, a traditionally fault-tolerant circuit C is evidently pieceably fault tolerant in one piece because $\tilde{C} = EC_1 \cdot C = C$ is traditionally fault tolerant when we take EC_1 to be the trivial error correction $I^{\otimes n}$.

Second, we include the final error correction EC_m even though the code space has already returned to that of L

because, for full generality, we may want to perform a round of error correction before we have our circuit's fault tolerance judged by the ideal \overline{EC}_L , as the exREC definition says. For instance, we see examples where the intermediate error corrections share classical syndrome data with EC_m so that the set of errors EC_m corrects is different than that \overline{EC}_L would. If EC_m is not needed, however, it can always be set to $I^{\otimes n}$ and remain in line with the definition.

Third, whenever the stabilizer becomes non-Pauli, intermediate error corrections will be trickier than simply using sufficiently large CAT states $(|0\rangle^w + |1\rangle^w)/\sqrt{2}$ to measure all stabilizers as per [27]. In those cases, we must be extra careful to construct circuits that correct all errors we need to correct without introducing errors that cannot be corrected in the next round. The quantum error-correcting conditions [26], although providing a necessary condition for such error-correcting circuitry to exist, do not provide a sufficient one.

Finally, note that we do not require an intermediate error-correction EC_j to correct all errors resulting from one fault in the previous circuitry. Although sufficient to guarantee pieceable fault tolerance, such a condition is not necessary. Indeed, to simplify the circuitry of EC_j and mitigate the number of errors it might introduce itself, we often intentionally let errors slip by an intermediate error correction as long as they are correctable in some subsequent round (often the final round).

At this point, it may not actually be clear that pieceable fault tolerance is going to do anything interesting. Nevertheless, our spirits might be buoyed by the example in Hill *et al.* [9]. After all, being able to switch from the 5-qubit to 7-qubit codes in a pieceably fault-tolerant fashion immediately endows the 5-qubit code with a pieceably fault-tolerant CZ gate, because the 7-qubit code has a transversal one.

In Sec. III, we construct a significantly simpler CZ gate for the 5-qubit code by a pieceably fault-tolerant construction in two pieces. Our CZ construction also suggests a similar design for a pieceably fault-tolerant logical CCZ gate on the 5-qubit code. Notably, this completes a universal set of gates for the 5-qubit code, which already has a transversal set of single-qubit Cliffords. In Sec. IV, we develop a similar CCZ construction for the 7-qubit code. Moreover, in Sec. V, we show that all nondegenerate stabilizer codes have a pieceably fault-tolerant logical gate equivalent to CCZ under local Clifford gates. Actually, there is more that is possible with pieceable fault tolerance. For any two *different* nondegenerate stabilizer codes, we can perform some logical gate locally Clifford equivalent to a CZ gate between them. This, for instance, allows pieceably fault-tolerant SWAP gates between different codes (when they have appropriate fault-tolerant local Cliffords), providing perhaps the broadest method of code switching yet known, at least when no functional ancillas are used. Finally, each of these statements generalizes to C^hZ and

C^hX gates, where h is an integer specifying the number of control qubits (e.g., C^1Z is a CZ gate and C^2Z is a CCZ, and we denote what is usually called CNOT by CX and Toffoli by $C^2X = CCX$).

D. Additional pieceable concepts

We take the time now to introduce some terminology that applies to pieceably fault-tolerant circuits in general. We use these concepts, too, for our specific constructions.

Let us first try to envision what the code stabilizer might look like after several pieces $C_{1r} = C_r \cdot C_{r-1} \dots C_1$ of the logical circuit. We assume C_{1r} is unitary for this discussion. Basic stabilizer theory tells us that, if the stabilizer group began as $S \subseteq \mathcal{P}_n$, then after C_{1r} , we will have $S_r = \{C_{1r} s C_{1r}^\dagger : s \in S\}$. If C_{1r} is non-Clifford, then at least some members of S_r may be non-Pauli. How does error correction work for such a stabilizer?

First, note that while members of S_r might not be Pauli, each is at least both Hermitian and unitary, because all $s \in S$ were. Each also still has two eigenspaces, with eigenvalues ± 1 , that are equally sized. The circuit in Fig. 1(a) still serves to measure elements of S_r . So, still, we can measure a generating set of S_r . However, as already mentioned in Sec. II C, we must be careful about what errors a failure in these measurements might introduce to the next piece of the circuit.

So we first ask, what form can these errors take? We first need to recognize that some errors are contagious, in the sense that they propagate other errors within some piece C_i . We define the set of contagious errors of a pieceable circuit C as

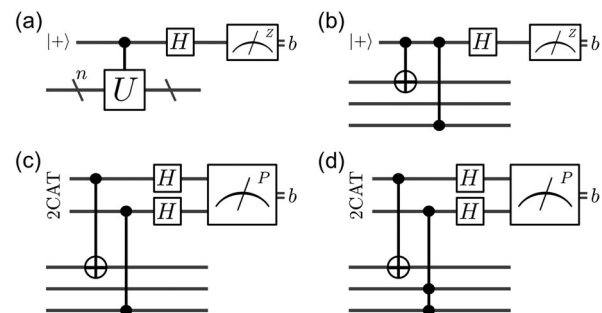


FIG. 1. (a) The method to measure a Hermitian and unitary n -qubit operator U [26]. When the result of the Z-basis measurement is $b = \pm 1$, the state of the lower qubit is projected into the ± 1 eigenspace of U . (b) The method applied to measuring a Pauli XIZ on three code qubits. (c) The method made fault tolerant by using a verified 2-qubit CAT state, $2CAT = (|00\rangle + |11\rangle)/\sqrt{2}$ [27]. The measurement is now of the parity of the ancilla qubits. (d) The CAT state method for measuring a non-Pauli operator $X_1CZ(2,3)$. Strictly speaking, this measurement is not fault tolerant, because the failure of the CCZ gate can cause two errors on the code qubits (which is generally uncorrectable if the code has distance 3). This is why we are so careful about measuring non-Pauli operators.

$$\mathcal{E}_C = \{E \in \mathcal{P}_n : \exists i \text{ s.t. } [E, C_i] \neq 0\}. \quad (5)$$

All other errors, however, those that commute with all pieces C_i , are errors that may be allowed to pass an intermediate error correction (or be introduced by faults in one). Also, we can never prevent error correction from introducing single-qubit errors to the next piece, and so single-qubit errors must also be allowable errors.

So our goal at intermediate error correction should be to correct all contagious errors and do so in such a way that only noncontagious and single-qubit errors are ever possibly allowed to enter the next piece of the circuit. An immensely helpful tool in this regard is the constant stabilizer of the circuit C , which we define as

$$S_C = \{s \in S : \forall i, [s, C_i] = 0\}. \quad (6)$$

The benefit of S_C , which we note is a subgroup of the whole stabilizer S , is that all its members remain Pauli throughout the entirety of the pieceable circuit C . The upshot is that we can always measure elements of S_C with the CAT state method, Fig. 1(c), and be assured that this process will not introduce more than a single-qubit error to the code qubits. The other stabilizers, those we call nonconstant because they do not commute with some piece of C , are not guaranteed to have the same property, failing specifically when they are non-Pauli.

This suggests an adaptive error-correction procedure to be used whenever we have both constant and non-Pauli nonconstant stabilizers. We require that constant stabilizers can detect all contagious errors. The procedure begins by measuring the constant stabilizers. If an error is not detected, then we know that no contagious errors have occurred. We can, therefore, let whatever errors have occurred (if any) pass through uncorrected. If an error has been detected, then we know a fault has occurred, and, barring a second fault, future stabilizer measurements will be faultless. Thus, we can now measure the nonconstant stabilizers, deduce what error has occurred, and correct it.

Notice that the constant stabilizers have been used to guarantee that the measurement of the nonconstant stabilizers proceeds reliably, without any faults. We call this a reliability guarantee on the nonconstant stabilizers. This is necessary because faulty measurement of nonconstant stabilizers can introduce contagious errors to the next piece of the circuit, which may allow errors to spread out of hand. Consider, for instance, failure of the CCZ gate in Fig. 1(d).

A second guarantee can be given for a nonconstant stabilizer as well, which, while not strictly necessary, is convenient when we want to think about errors as Paulis. To define this guarantee, imagine that we have a set of Pauli errors $\mathcal{E} \subseteq \mathcal{P}_n$ entering an intermediate error-correction stage. Rather than applying a non-Pauli recovery we want to apply Pauli recovery operations only. Measurement of a stabilizer generator g eliminates the possibility of some

Pauli errors, but to measure g requires knowing that each possible Pauli error remaining either commutes or anti-commutes with g . For generators of the constant stabilizer $g \in S_C$, this is trivial, but for a nonconstant stabilizer $t \in S \setminus S_C$, there is no reason *a priori* to assume the commutation guarantee for t is true.

The way we solve this problem is with a four-step procedure: (1) measure the constant stabilizers, (2) partially correct some Pauli errors so that commutation guarantees exist on nonconstant stabilizers, (3) measure the non-constant stabilizers, and (4) complete the recovery. Notice that we had already planned to use steps (1), (3), and (4) to assure a reliability guarantee. Step (2) has been added so that recovery, though now split into two stages, (2) and (4), consists entirely of Pauli operators. We call this our procedure for adaptive, reliable stabilizer error correction or PARSEC, and it plays a prominent role in our non-Clifford pieceably fault-tolerant constructions.

We note that PARSEC generally has these four steps, but because it is adaptive some may be skipped in any one instance, and there are other variations depending on the code. The PARSEC variants we find most useful, in general, are detailed herein as Tables I and II.

As for pieceable constructions, one type of circuit is our main innovation in logical circuits and makes our systematic study of pieceable fault tolerance possible. We call it the round-robin design. Given an h -qubit gate U and disjoint sets of qubits $\Lambda_1, \Lambda_2, \dots, \Lambda_h$, the round-robin circuit of U on $\{\Lambda_i\}$ is

TABLE I. PARSEC. A procedure for doing intermediate error correction in the nonstabilizer codes encountered partway through the round-robin construction of logical CCZ and, in Sec. V, the constructions of Theorem V.1. An “error-correcting constant stabilizer” is one that can distinguish between all contagious errors modulo noncontagious errors, defined more formally in Sec. V.

Input: (1) $h + 1$ code blocks partway through a round-robin circuit.
(2) Each block has an error-correcting constant stabilizer.
Output: (1) $h + 1$ code blocks free of contagious errors.
(2) The locations of noncontagious errors if there are multiple per block.
(i) Measure the constant stabilizers of all blocks
(ii) If two or more blocks have triggered: (<i>faulty $C^h Z$</i>)
(a) For each triggered block:
(b) Apply X to the affected qubit
(c) Note possible spread of Z errors to other blocks
(iii) If one block triggered: (<i>single error in that block</i>)
(a) Measure nonconstant stabilizers of that block
(b) Correct the error
(c) Note possible spread of Z errors to other blocks
(iv) If no blocks triggered: (<i>at most noncontagious errors</i>)
(a) Do nothing

TABLE II. CSS PARSEC. An error-correction procedure achieving the same goals as PARSEC, but without ever having to measure the nonconstant stabilizers of any block. The CSS in the procedure's name stems from the fact that any normalizer of a CSS code that consists entirely of X 's and I 's or of Z 's and I 's will have a constant stabilizer satisfying the third input condition (but not necessarily the second, though nondegeneracy of the code is sufficient to do so).

Input: (1) $h + 1$ code blocks partway through a round-robin circuit.
(2) Each block has an error-correcting constant stabilizer.
(3) Each block's constant stabilizer also distinguishes between contagious errors and errors on idle qubits.
Output: (1) $h + 1$ code blocks free of contagious errors.
(2) The locations of noncontagious errors if there are multiple per block.
(i) Measure the constant stabilizers of all blocks
(ii) If any block has triggered:
(A) For each triggered block:
(a) If error is contagious, apply X to affected qubit and note spread of Z errors to other blocks
(b) If error is on an idle qubit, do nothing
(iii) If no blocks triggered:
(B) Do nothing

$$\prod_{j_1 \in \Lambda_1} \prod_{j_2 \in \Lambda_2} \dots \prod_{j_h \in \Lambda_h} U(j_1, j_2, \dots, j_h). \quad (7)$$

We also make this definition assuming that U is such that all gates in the product mutually commute, so that ordering the products is unnecessary. Then, all of our new pieceably fault-tolerant logical constructions are, in fact, of the round-robin variety with different choices of U and the sets Λ_i . When it comes to round-robin circuits, we refer to qubits in $\Lambda = \bigcup_{j=1}^h \Lambda_j$ as active qubits and any nonancilla qubits not in Λ as idle.

III. 5-QUBIT CODE

Having overviewed the design of logical circuits in theory and defined our particular method, we now get our hands dirty and do some actual circuit designing, beginning with the 5-qubit code. We show how to make fault-tolerant CZ and CCZ gates on the 5-qubit code. More than these are possible in the formalism of pieceable fault tolerance, but we save the most general proofs until Sec. V. Our goal here is to present concrete examples of pieceably fault-tolerant logic on a small quantum code. This includes examples of contagious errors, constant stabilizers, PARSEC, and round-robin circuits, concepts which we introduce only very broadly in the previous section.

A. Transversal gates

The 5-qubit quantum code, a $[[5, 1, 3]]$ stabilizer code, was one of the earliest quantum codes to be discovered [32,33] and is the smallest quantum code of distance 3.

The stabilizer and normalizer of the 5-qubit code can be written as

$$S_5 = \left\langle \begin{array}{c} ZZXIX \\ XZZXI \\ IXZZX \\ XIXZZ \end{array} \right\rangle, \quad (8)$$

$$\bar{Z}_5 = -XIZIX, \quad (9)$$

$$\bar{X}_5 = -YIXIY. \quad (10)$$

Notice that we deviate from the standard presentation of $\bar{Z}_5 = ZZZZZ$ and $\bar{X}_5 = XXXXX$ to versions that are equivalent under multiplication by stabilizers, and also have the lowest possible weight.

It is well known that the 5-qubit code supports a universal, transversal set of single-qubit Clifford operations. To begin with, for $s_x, s_y, s_z \in \{-, +\}$, all eight members of the family of single-qubit octahedral gates,

$$K_{s_x s_y s_z} = e^{i(\pi/3\sqrt{3})(s_x X + s_y Y + s_z Z)}, \quad (11)$$

are transversal on the 5-qubit code in the obvious way: $\bar{K}_{s_x s_y s_z} = K_{s_x s_y s_z}^{\otimes 5}$. A special case that we name and use often is $K_{+++} = SH := K$, where $S = \text{diag}(1, i)$ and $H = (X + Z)/\sqrt{2}$. The set of transversal single-qubit Clifford operations can, therefore, be completed by implementing a logical Hadamard gate as $\bar{H} = P_\pi H^{\otimes 5}$, where P_π is an appropriate permutation of the five qubits, one example of which is shown in Fig. 2(b). A permutation is transversal if it is implemented, not by physical SWAP gates, but by simply relabeling the qubit lines. It is known, at least for stabilizer codes, that such permutations alone cannot bypass the universality-transversality no-go [4].

Gottesman [34] gave a way to complete the set of fault-tolerant Clifford operations on the 5-qubit code. He noted that a 3-qubit Clifford gate T_3 is transversal on the 5-qubit code. Supplementing T_3 with measurement and single-qubit Clifford gates provides a fault-tolerant implementation of CX. There are a couple downsides to this approach. First, T_3 is a complicated gate. To implement T_3 on physical qubits will likely require compiling into simple

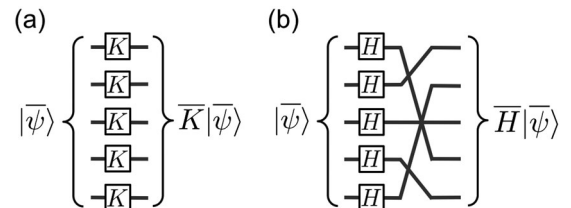


FIG. 2. Transversal implementations of (a) the octahedral gate $K = SH$ and (b) the Hadamard gate H on the 5-qubit code.

1- and 2-qubit Clifford gates, as experiments will not have direct access to T_3 . Second, implementing logical CX with T_3 requires the use of five functional ancilla qubits making up the third 5-qubit code block that is eventually measured and discarded.

B. Pieceably fault-tolerant CZ

Our first goal is to develop a pieceably fault-tolerant logical circuit for CZ on the 5-qubit code that uses no functional ancilla qubits. In fact, our pieceably fault-tolerant circuit consists of only two pieces. Moreover, we are able to easily generalize this design to pieceably fault-tolerant logical CCZ (in four pieces) on the 5-qubit code.

Both pieces of the pieceably fault-tolerant circuit for logical CZ are shown in Fig. 3. This is our first example of a useful round-robin construction, accompanied by a prologue and epilogue of local Clifford gates. The story begins with a local unitary transformation $K_1 Y_3 K_5$ on both code blocks. These Clifford gates take the 5-qubit code to an equivalent $[[5, 1, 3]]$ code, which we refer to as the $5'$ -qubit code, with

$$S_{5'} = \left\langle \begin{array}{l} -YZXIZ \\ -ZZXXI \\ -IXZZZ \\ -ZIXZY \end{array} \right\rangle, \quad (12)$$

$$\bar{Z}_{5'} = ZIZIZ, \quad (13)$$

$$\bar{X}_{5'} = XIXIX. \quad (14)$$

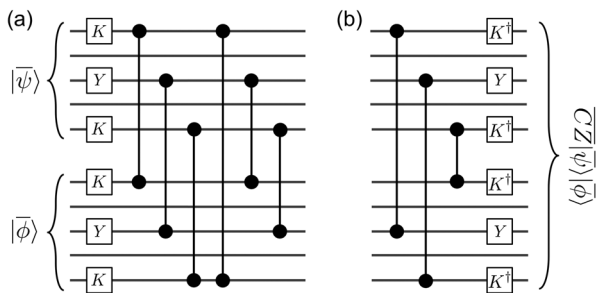


FIG. 3. Circuits C_a (a) and C_b (b) such that $C_{CZ} = C_b \cdot C_a$ is a pieceably fault-tolerant implementation of CZ on the 5-qubit code. The intermediate error correction, done after (a), is described in the text, while the final error correction after (b) can be taken as two copies of the canonical 5-qubit error correction, which is the canonical error correction for the tensor product of two 5-qubit codes. In the notation of the definition of pieceable fault tolerance, $\tilde{C}_{CZ} = C_b \cdot EC_1 \cdot C_a$. To do a logical CX from the top code block to the bottom change all physical CZs to physical CXs from qubits in the top to those in the bottom. The intermediate error correction will still succeed in correcting all single faults if all stabilizers of the corresponding intermediate code are measured.

Label qubits in the first code block j_A and qubits in the second code block k_B . Now it is simple to check that the round-robin circuit of CZ gates,

$$\prod_{j,k \in \{1,3,5\}} CZ(j_A, k_B), \quad (15)$$

which contains nine CZ gates total, preserves the combined stabilizer of both blocks $\{s_1 \otimes s_2 : s_1, s_2 \in S_{5'}\}$. This is most easily seen by noticing that in each stabilizer generator there are an even number of X 's and Y 's on the active qubits, qubits 1,3,5 of each block. This round-robin circuit also effects the appropriate transformation on the normalizers of code blocks A and B ,

$$\bar{Z}_A \otimes \bar{I}_B \rightarrow \bar{Z}_A \otimes \bar{I}_B, \quad \bar{I}_A \otimes \bar{Z}_B \rightarrow \bar{I}_A \otimes \bar{Z}_B, \quad (16)$$

$$\bar{X}_A \otimes \bar{I}_B \rightarrow \bar{X}_A \otimes \bar{Z}_B, \quad \bar{I}_A \otimes \bar{X}_B \rightarrow \bar{Z}_A \otimes \bar{X}_B, \quad (17)$$

due to the odd number of X 's on active qubits in $\bar{X}_{5'}$.

Breaking the round-robin circuit Eq. (15) after any six CZs that touch no single qubit more than twice is sufficient to guarantee pieceable fault tolerance. One example of such a decomposition is shown in Fig. 3. To use the language of Sec. II B, the pieces should be 2-transversal. The intuitive explanation of why this works is that, while the 5-qubit code can only correct single-qubit errors when errors are randomly distributed across all code qubits, if errors are restricted to two known locations, then all errors on those two qubits are correctable. By sharing information about the stabilizers they have measured, code blocks A and B can both correct individual, uncorrelated errors if they exist, and inform each other about possible propagation of errors from one block to the other, a propagation that will affect at most two known qubits in the other block.

Anthropomorphizing aside, we can actually check that all errors introduced by components in Fig. 3(a) are correctable by measuring the stabilizers of the appropriate code after Fig. 3(a). The code in question is a $[[10, 2, 3]]$ stabilizer code with stabilizer generators

$$S_{10} = \left\langle \begin{array}{ll} -YZXIZIIZIZ & -ZIZIIZIXZY \\ -ZZZXIIIIII & -IIIIIXZZZ \\ -IXZZZIIIII & -IIIIZZZXI \\ -ZIXZYZIIZ & -ZIIIZYZXIZ \end{array} \right\rangle. \quad (18)$$

Note that the first and last rows of generators, those with Pauli weight on both blocks, are nonconstant stabilizers, and the remaining four generators are constant stabilizers, since they commute with the round-robin circuit, Eq. (15). We also refer to the generators on the left as belonging to block A and those on the right as belonging to B , since they are the transformations of the generators in Eq. (12) for their respective blocks.

Assume only one component in Fig. 3(a) has failed, and propagate the resulting Pauli errors to the end of the subcircuit, where we measure the stabilizer generators in Eq. (18) in a fault-tolerant fashion. A generator is said to trigger if we measure that stabilizer and find a -1 eigenstate.

There are three cases from the measurement: (1) constant stabilizers trigger in both blocks, (2) constant stabilizers from just one block trigger, or (3) no constant stabilizers trigger. The same three cases appear in later arguments as well. They are convenient since the constant stabilizers of a block *not* triggering implies no contagious errors in that block, which proves that the nonconstant stabilizers of the *other* block have triggered (or not) depending only on errors within their native block.

We proceed to argue that errors within case (1) can be corrected. Here, we know that a CZ gate has failed with contagious errors (X 's or Y 's) on both nodes. These may have also propagated Z errors through a subsequent CZ. Fortunately, the constant stabilizers, when restricted to the active qubits, represent a bit-flip redundancy code $\langle ZZI, IZZ \rangle$ on each block, so we can pinpoint exactly the qubit lines with contagious errors, diagnose precisely which CZ failed, and learn where Z errors have propagated. At most, two errors are present in each block, with known locations. Now notice that we can obtain the eigenvalues of any original $5'$ -qubit stabilizer [i.e., Eq. (12)] on either block using the measured syndromes, the eigenvalues of the Paulis in Eq. (18), and our knowledge of the locations of contagious errors. Thus, the error-correction problem now becomes two independent corrections of at most two located errors per block of the distance-three, $5'$ -qubit code.

In case (2), if the constant stabilizers of, say, block A triggered, we know that there is no contagious error in block B , and moreover only a single error in block A . Thus, the syndrome of all block A 's stabilizers, both constant and nonconstant, will determine the error in A in the usual $5'$ -qubit code fashion for single error correction. If that error was contagious, we will also know now the two possible locations of Z errors in block B . As before in case (1), by calculating the eigenvalues of the original $5'$ -qubit code generators for block B from our measured syndromes plus knowledge of the error in block A , we can correct the two errors in the usual $5'$ -qubit code fashion for correcting two located errors.

Finally, case (3) implies that no contagious errors have occurred, and also that there is at most a single-qubit error in each block. Therefore, the syndrome will independently correct errors in both blocks in usual $5'$ -qubit code fashion.

It is interesting to consider that the intermediate error-correction procedure we have just described actually uses only 84 nontrivial syndromes out of a total $2^8 - 1 = 255$ possible. The remaining syndromes do not correspond to errors that can arise in the standard model of faulty components from just one component failing, but they

will allow the correction of some errors arising from two faulty components.

It should be noted that the final error correction for our logical CZ construction can be taken to be a canonical error correction, consisting of two copies of the canonical error correction for the 5-qubit code. This works because the final piece of the round-robin circuit, Fig. 3(b), is 1-transversal.

C. \bar{T} will always require functional ancillas

We have described how to obtain the entirety of the Clifford group fault tolerantly on the 5-qubit code, without any ancillas other than those needed for error correction. A final logical gate is needed, however, to complete a universal set of fault-tolerant gates. From the perspective of universality, there are many choices we could make—any gate outside the Clifford group will do [4]. Nevertheless, from the perspective of fault tolerance, some gates are indeed forbidden from being implemented without overhead in the form of functional ancillas. The $T = \text{diag}(1, e^{i\pi/4})$ gate is one of these.

To show that a fault-tolerant T gate on the 5-qubit code cannot be implemented without functional ancillas, even in a pieceably fault-tolerant fashion, begin by noting that a logical T cannot be implemented with only single-qubit gates and qubit swaps—that is, T is not a transversal gate on the 5-qubit code. This follows from the second theorem of Zeng *et al.* [4], where they show that no stabilizer code has a universal, transversal set of gates even on one encoded qubit. Since T is not transversal, any logical T construction with no ancillas must use a multiqubit gate between some number of the five code qubits (see, e.g., Fig. 4). Look at the chronologically last such multiqubit gate. After that gate, if the circuit really implements a logical T gate, the code space will be locally unitary

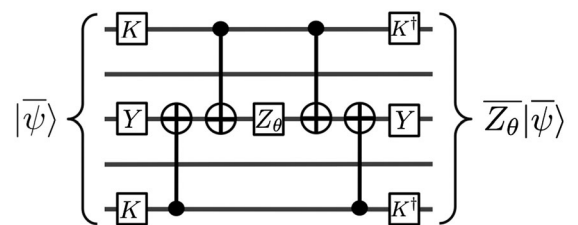


FIG. 4. A logical $Z_\theta = \exp(i\theta Z/2)$ gate for the 5-qubit code. The case $\theta = \pi/4$ corresponds to a T gate. This construct, like any logical T -gate circuit using no functional ancillas, is not fault tolerant nor even pieceably fault tolerant. However, this does not mean such designs are entirely useless. For instance, the above circuit provides a fault-tolerant circuit for T in the $[[75, 1, 3]]$ code formed by concatenating the $[[15, 1, 3]]$ Reed-Muller code [15,35] within the $5'$ -qubit code [the code resulting from applying $K_1 Y_3 K_5$ to the 5-qubit code; see Eq. (12)], because the Reed-Muller code has both transversal T and transversal CX logical constructs. This same idea of concatenating codes to produce a larger code with a greater, and sometimes even universal, set of fault-tolerant logical gates is used to great effect in Ref. [14].

equivalent to the code space of the 5-qubit code. Thus, because the 5-qubit code is a perfect code, we know that no multiqubit errors will be correctable (as long as all single-qubit errors must simultaneously be correctable, which is always the case). We conclude that the last multiqubit gate will always have the potential to introduce uncorrectable errors if we use no functional ancillas to implement T on the 5-qubit code.

The same argument works for any nontransversal single-qubit gate and any code that is unable to correct all errors on all subsets of multiple code qubits. For instance, the 7-qubit code also needs functional ancillas for a logical T gate, because the classical codes underlying its CSS construction are perfect.

When functional ancilla qubits are used, a pieceably fault-tolerant logical circuit for T does exist for the 5-qubit code by our construction of a pieceably fault-tolerant CCZ in the next section and the universality that it bestows. But it is also interesting to note other constructions in the literature. First, there are the magic-state constructions due to Shor [27] and Bravyi and Kitaev [5]. Alternatively, code-switching techniques, first from the 5-qubit to 7-qubit code [9], and then from the 7-qubit to 15-qubit code [11], also give a fault-tolerant method of implementing T , since it becomes transversal once we have transferred to the 15-qubit code.

D. Pieceably fault-tolerant CCZ

In this section, we describe a pieceably fault-tolerant implementation of the CCZ gate on the 5-qubit code, thereby completing a fault-tolerant universal set of gates for the smallest quantum error-correcting code. Unlike fault-tolerant T -gate implementations, this construction requires no functional ancilla qubits. It is, however, a pieceably fault-tolerant design of four pieces, so there are four rounds

of error correction to contend with. The error-correction procedures themselves are novel, of the adaptive variety termed PARSEC in Sec. II D, since, unlike the logical CZ case of Sec. III B, intermediate in the logical CCZ circuit the code space is no longer that of a stabilizer code.

The pieceably fault-tolerant CCZ circuit is shown in Fig. 5. The general idea is no different from the design of the fault-tolerant CZ. First, transform all three code blocks into the 5'-qubit code in Eq. (12) by applying the single-qubit unitaries $K_1 Y_3 K_5$. Second, apply the round-robin CCZ circuit,

$$\prod_{j,k,l \in \{1,3,5\}} \text{CCZ}(j_A, k_B, l_C), \quad (19)$$

containing $3^3 = 27$ CCZs total. Finally, transform each block back to the standard 5-qubit code with $K_1^\dagger Y_3 K_5^\dagger$.

Checking that Eq. (19) implements a logical CCZ on the 5'-qubit code is only slightly more difficult than checking that Eq. (15) implements logical CZ. The $\text{CCZ} = \text{diag}(1, 1, 1, 1, 1, 1, 1, -1)$ gate is in the third level of the Clifford hierarchy [36]. Moreover, it treats all three qubits symmetrically, commutes with Pauli Z 's, and acts on Pauli X 's like

$$\text{CCZ}(XII)\text{CCZ} = XII \times \text{CZ}(2, 3). \quad (20)$$

We must verify the same action of Eq. (19) on the 5'-qubit codes' logical operators $\bar{Z}_A, \bar{Z}_B, \bar{Z}_C, \bar{X}_A, \bar{X}_B, \bar{X}_C$ and preservation of the stabilizer.

The latter is simple. Since each stabilizer generator has an even number of X or Y operators on the active qubits (again, qubits 1,3,5 of each block), any CZs introduced to the stabilizer by Eq. (20) will cancel by the end of the round-robin circuit, Eq. (19).

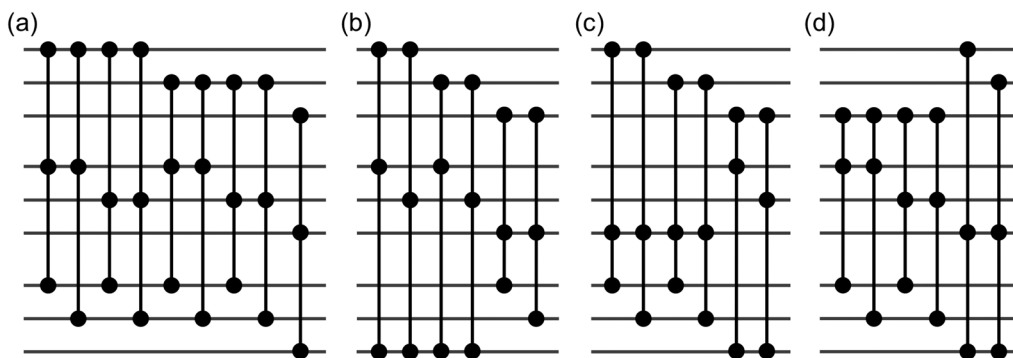


FIG. 5. The round-robin circuit for CCZ between three code blocks each with three active qubits (e.g., qubits 1, 3, 5 for a 5'-qubit code block and qubits 5, 6, 7 for a 7-qubit code block). Idle code qubits are omitted. The circuit is broken into four pieces (a)–(d), each of which is 2-transversal, and therefore an appropriate error correction performed after each piece ensures pieceably fault tolerance (see text). This circuit design is quite versatile. For instance, this idea performs a CCZ on the 5'-qubit code (Sec. III D), on the 7-qubit code (Sec. IV B), and between 5'- and 7-qubit code blocks, among other possibilities (Sec. V). On some codes (see Sec. V for explicit conditions), including the 5'- and 7-qubit codes, replacing the CCZs with CCXs or CCYs will also implement logical CCX or logical CCY, respectively.

As for the transformation of the logical operators, we can note that \bar{Z}_Q for code blocks $Q = A, B, C$ trivially does not change. Since \bar{X}_Q is odd weight and is supported by Pauli X 's on only the active qubits, we find

$$\bar{X}_A \bar{I}_B \bar{I}_C \rightarrow \bar{X}_A \bar{I}_B \bar{I}_C \prod_{j,k \in \{1,3,5\}} \text{CZ}(j_B, k_C), \quad (21)$$

and likewise for \bar{X}_B and \bar{X}_C . However, we argue in Sec. III B above that the latter product of CZs, the round-robin CZ circuit from Eq. (15), is exactly a logical CZ on the $5'$ -qubit code (in this case, on blocks B and C), and so Eq. (21) is the logical version of Eq. (20), as required.

We now show that the round-robin CCZ circuit of Eq. (19) is pieceably fault tolerant. The decomposition proceeds similarly to that of the round-robin CZ circuit. Group the CCZs of Eq. (19) into pieces such that, in any piece, no single qubit interacts with more than two qubits of another block (i.e., each piece is 2-transversal). One possibility, with four pieces, is shown in Fig. 5. We perform error correction after each piece, which we describe shortly.

However, let us first note the form of the stabilizer at any of the three intermediate error corrections. There are four stabilizer generators corresponding to each block, for a total of 12. For instance, the four from block A at any intermediate error correction will have the form

$$S_A = \left\langle \begin{array}{l} -YZXIZ \times \text{CZ}_{A1} \\ -ZZZXI \\ -IXZZZ \\ -ZIXZY \times \text{CZ}_{A4} \end{array} \right\rangle, \quad (22)$$

where it is understood that these generators contain $I^{\otimes 5}$ on blocks B and C and CZ_{A1} and CZ_{A4} are products of CZs between blocks B and C . For instance, after the first piece of Fig. 5, $\text{CZ}_{A4} = \text{CZ}(1_B, 1_C)\text{CZ}(1_B, 3_C)\text{CZ}(3_B, 1_C) \times \text{CZ}(3_B, 3_C)\text{CZ}(5_B, 5_C)$. The stabilizer generators for blocks B and C are of a similar form. Again, two stabilizer generators per block, six total, have changed (the nonconstant stabilizers) and two per block have not (the constant stabilizers).

Of course, the nonconstant stabilizers have not changed when we just look at the qubits on their native block. We use this fact, because if we have ensured at most Z errors in the other blocks (this is the meaning of commutation guarantees in this case), then the nonconstant stabilizers in block A (for instance) will act just as the original stabilizer generators in terms of detecting errors on A .

Assume that a single faulty component in the previous circuitry, say $\text{EC}_k \cdot C_k \dots \text{EC}_1 \cdot C_1$, will, if anything, introduce either noncontagious errors or single-qubit errors to the $(k+1)$ th piece, C_{k+1} . It can be verified that the intermediate error-correction procedure we describe next actually does ensure this—only noncontagious errors will

pass, and, assuming no earlier fault in the circuit, a failure in the error correction itself will introduce at most a single-qubit error. The latter property is because we measure only nonconstant stabilizers when we have a reliability guarantee (see Sec. II D).

We implement error correction EC_{k+1} using PARSEC, Table I. But why does it work for the 5-qubit code? We go through the procedure step by step. Begin by measuring the constant stabilizers of each block (six total). Since these are Paulis, we can measure them in standard fashion (e.g., via a CAT state ancilla introduced by Shor [27] with repeats and majority voting). Notice that any contagious errors will be detected by these constant stabilizer measurements. Indeed, if constant stabilizers from two or three different blocks have triggered [case (1)], we know for sure that a CCZ has failed, since this is the only way just one failure could introduce errors detectable by the constant stabilizers into two different blocks. We also know that the errors are contagious. These errors are immediately correctable modulo noncontagious errors—we know where the errors have occurred from the constant stabilizers' syndrome, and can apply X to the afflicted qubits, leaving at most Z errors. If constant stabilizers have just triggered in one block [case (2)], then we know that at worst noncontagious errors are present in the other blocks, and that the triggered block has at most a single-qubit error. If no constant stabilizers have triggered [case (3)], then at worst noncontagious errors have occurred.

In case (1), having found and corrected contagious errors, we know where Z errors might possibly be. By the way the pieces are constructed (i.e., 2-transversally), at most two Z errors will be present in each block. Now we could just measure all nonconstant stabilizers [see, for example, Fig. 1(d)], with reliability and commutation guarantees from having detected and corrected the X errors, and this would give enough information to correct the Z errors. But luckily, the Z errors will not propagate further, so we can also just leave them until the final error correction, when all stabilizers are Pauli again and so simple to measure. This means we have to inform the final error correction of the locations of these Z errors, and this is why that information is listed as an “output” in Algorithm 1 (Table I). If the final error correction gets no information from the intermediates, it should revert to canonical form, correcting one arbitrary error per block.

In case (2), if the constant stabilizers of block Q have triggered, we have, assuming just one faulty component, reliability and commutation guarantees on the nonconstant stabilizers of block Q . Measuring them (and this time, but only this time, the nonconstant stabilizers must be measured), we can deduce what and where the error in block Q is. Correct the error in block Q . If the error had occurred on an idle qubit, we are done. If the error was on an active qubit, however, then we know where the possible Z errors

may be in other blocks. Again, inform the final error correction.

Though case (3) might appear the most straightforward, it is actually the most interesting. In this case, we could go ahead and measure all nonconstant stabilizers of all blocks, and therefore find the locations of, at most, one Z error per block. However, there is a problem with this approach—we have no reliability guarantee. Having detected no error, the measurement of the nonconstant stabilizers during EC_{k+1} could itself contain the first fault of the circuit, and it might result in a contagious error entering the next piece of the circuit C_{k+2} . This is a problem, because to analyze cases (1) and (2) we assume one faulty component in $EC_k \cdot C_k \dots EC_1 \cdot C_1$ causes either a noncontagious error or a single-qubit error to enter C_{k+1} , and the same assumption must hold for $k \rightarrow k+1$. We therefore cannot measure the nonconstant stabilizers in case (3).

The solution to this problem is, of course, that we do not actually need to correct the Z errors, as we have already seen. They will not propagate further, and so can just be corrected at the final error correction. Since there is only at most one per block, we do not even need to give the final error correction special instruction.

Because the CCZ and CZ gates we design for the 5-qubit code can recover from only one faulty component, we envision concatenating [17] the circuits k times to achieve tolerance to at most $2^k - 1$ faults. Relevant to this process, the CCZ circuit we design in this section has the appealing property of requiring only physical components (be it gates, measurements, or state preparations) that can be directly implemented logically on the 5-qubit code. Therefore, concatenating the circuit of Fig. 5 can be done without first compiling the CCZ gates into 1- and 2-qubit gates. Only at the physical level, and only in architectures without native CCZ gates, must such a compilation be done.

Next, we mention how the same PARSEC procedure is general enough to work on the 7-qubit code as well. But it can also be made simpler, by virtue of the 7-qubit code being CSS.

IV. 7-QUBIT CODE

A. Transversal gates

The 7-qubit code [1] is the most frequently studied of the small stabilizer codes, trumping the 5-qubit code in this metric despite its larger size. One advantage of the 7-qubit code is its complete set of transversal Clifford gates—all three of H , S , and CX are transversal in the simplest way, as 7 copies of the physical gate applied qubitwise (e.g., $\bar{H} = H^{\otimes 7}$). It is also a code grounded in classical coding theory, being the smallest member of the CSS code family.

Here, we briefly recall the stabilizer and normalizer of this $[[7, 1, 3]]$ code. Normalizers, like those of the 5-qubit code previously, are reduced to minimum weight:

$$S_7 = \left\langle \begin{array}{cc} XXXXIII & ZZZZIII \\ XXIIXXI & ZZIIZZI \\ XIXIXIX & ZIZIZIZ \end{array} \right\rangle, \quad (23)$$

$$\bar{Z}_7 = IIIIIZZ, \quad (24)$$

$$\bar{X}_7 = IIIIXXX. \quad (25)$$

Transversality of H , S , and CX can easily be verified.

B. Pieceably fault-tolerant CCZ

In this section, we complete a universal set of gates on the 7-qubit code. Initially, we present a round-robin design mirroring that for the 5-qubit code and fitting within our general framework for pieceable gate design in Sec. V. However, the 7-qubit code has other properties that lead to appreciable optimization, which we discuss and use to perform a resource comparison against magic-state injection in the following section.

A logical CCZ circuit for the 7-qubit code circuit is the round-robin CCZ circuit,

$$\prod_{j,k,l \in \{5,6,7\}} CCZ(j_A, k_B, l_C), \quad (26)$$

with error correction after each subcircuit of CCZ s. These circuit pieces, like those in the case of the 5-qubit code, should be made 2-transversal. Qubits 5,6,7 of each block are active qubits and the remainder are idle qubits.

Checking that Eq. (26) implements a logical CCZ follows the same logic as in the 5-qubit case. The stabilizer generators will all be preserved because the first three each have an even number of X 's on the active qubits and the last three commute with Eq. (26). The normalizer \bar{Z}_7 will also be preserved for the same commutation reason. However, \bar{X}_7 will change,

$$\bar{X}_A \bar{I}_B \bar{I}_C \rightarrow \bar{X}_A \bar{I}_B \bar{I}_C \prod_{j,k \in \{5,6,7\}} CZ(j_B, k_C), \quad (27)$$

with the analogous transformation on the normalizers of blocks B and C . The thing to note is that the round-robin product of CZ s in Eq. (27) is also, just like for the 5-qubit code, a logical CZ on the 7-qubit code. And so Eq. (27) is the correct logical transformation consistent with a logical CCZ .

The pieces of Eq. (26) consist of one group of nine CCZ s and three groups of six, just like the 5-qubit case (see Fig. 5). The 7-qubit code also has some stabilizer generators that become non-Pauli as the round-robin CCZ circuit progresses: namely, the second and third, the non-constant stabilizers, and some generators that do not change at all, the first and the last three. Although the first is a constant stabilizer, it is trivially so (all Paulis on active

qubits are I 's), so only the last three, those of Z type, need be concerned when we say to measure constant stabilizers. Our job now is to argue that the pieces in Fig. 5 can all be error corrected without introducing uncorrectable errors into the next piece.

The key to intermediate error correction in the 7-qubit case is that all contagious X errors can be corrected by measuring only the constant stabilizers, while noncontagious Z errors commute through the circuit to the end. Because the 7-qubit code is CSS it can detect and correct one X error per block using only measurement of the first three stabilizers in Eq. (23). Conveniently and not coincidentally, at most one X or Y error will occur per block as well, resulting from failure of a physical CCZ. Having learned the locations of X or Y errors, the intermediate error correction can then tell the final error correction where to look for Z errors, at most two of them per block in known locations. The final error correction measures all stabilizers (which are Pauli once we return to the tensor product of three 7-qubit code spaces) and applies recovery using its syndrome plus the information from any intermediate error corrections.

Notice, no non-Pauli stabilizers are ever measured. Yet, we still manage to correct any single component failure in the circuit, Fig. 5. Not having to measure nonconstant stabilizers is a feature of all our round-robin constructions on CSS codes. Indeed, this modified PARSEC we call CSS PARSEC, outlined in Table II.

C. Resource comparison against magic states

The most popular method for performing logical non-Clifford gates on the 7-qubit code is through magic-state injection [5,26]. While many other codes require an extensive distillation procedure to create magic states, the 7-qubit code offers a postselective strategy [17,37], which is less resource intensive. So it is the latter that we compare against in this section, using three metrics: ancillas consumed, multiqubit gate counts, and circuit volume (time steps times active qubits). All the circuits used for this comparison are drawn in Appendix D.

Specifically, the postselection protocol we compare against is a highly optimized version similar to that of Goto [37] (whose circuit produces H -type magic states for implementing T gates). With our circuit, CCZ states for implementing a teleported CCZ gate are produced also after just one measurement of the appropriate non-Pauli operators and one syndrome measurement. This circuit is less resource intensive than the postselected procedure used in [17] that measures non-Pauli operators multiple times and also less intensive than decomposing a logical CCZ into four T gates [38] that are each magically injected (e.g., using Ref. [37]). For ease of resource counting, we assume that any postselections always succeed so our metrics do not have functional dependence on any physical error rates. This assumption generally helps the magic-state method

more, because of the large circuit volume that must be recomputed upon failed postselection.

As for the pieceable construction, while the circuit in the previous section for logical CCZ uses 27 physical CCZs, there is actually a simpler design that uses 21 physical CCZs and needs just one intermediate error correction. We briefly describe the two main ideas of this optimization. The first idea is to exploit the fact that CZ is transversal on the 7-qubit code to show that

$$\prod_{j,k \in \{5,6,7\}} \text{CCZ}(j_A, k_B, k_C) \quad (28)$$

implements logical CCZ. Unfortunately, as written Eq. (28) needs to be broken into four pieces to be made pieceably fault tolerant. This can be traced to the fact that qubits 5,6,7 from block A are each coupled to seven qubits from the other blocks. In accordance with the fault-tolerance principle of keeping pieces at most 2-transversal, we see $\lceil 7/2 \rceil = 4$ pieces are required for pieceable fault tolerance.

To reduce the number of pieces, the second idea is to use the Z stabilizers of block A to spread the nodes of the 21 CCZs around to all seven of the code qubits of block A . Indeed, we can do so such that each code qubit in any block is involved in no more than three CCZ gates. Thus, 2-transversality implies just $\lceil 3/2 \rceil = 2$ pieces are required. See Fig. 7 in Appendix D for one particular fault-tolerant piecing.

We use the two-piece, 21 CCZ pieceably fault-tolerant circuit derived above in our resource comparison. Error correction, both intermediate and final, are done with Steane states created by postselection that is assumed to always succeed.

Table III shows the final resource comparisons. We find that using a pieceable CCZ gate over magic-state injection nearly *halves* the required resources for logical CCZ. These results also give a reason to expect a lower logical error rate for a pieceable CCZ, though we leave the exact determination of this quantity to future work.

V. CONDITIONS FOR PIECEABLE FAULT TOLERANCE IN STABILIZER CODES

The previous two sections, Sec. III about constructing pieceably fault-tolerant circuits for the 5-qubit code and Sec. IV about similar constructions for the 7-qubit code, left

TABLE III. Resource comparison for implementing $\overline{\text{CCZ}}$ on the 7-qubit code using either magic-state injection via postselection or pieceable fault tolerance.

	CX count	CCZ count	Ancillas	Volume
Magic states	312	22	132	1518
Pieceable	162	21	72	771
Percent improvement	48.1%	4.5%	45.5%	49.2%

one burning theoretical question open: for what codes do there exist pieceably fault-tolerant constructions for interesting gates?

In this section, we explore this question in detail. We find several remarkable results, including the fact that all nondegenerate stabilizer codes with distance at least three have a pieceably fault-tolerant gate that is locally Clifford equivalent to C^hZ . For codes already endowed with transversal local Clifford gates, this is enough for universality. Our general constructions also work between blocks in different codes as well, implying a method of code switching with pieceable circuits.

Because we find round-robin circuits so useful in Secs. III and IV, we continue studying them in this section, only in more generality. In the first part of this section, we have two main concerns. Our first is to make logical gates from round-robin circuits. Only after do we state further conditions so that the logical circuits are also pieceably fault tolerant, which comes alongside a general method for piecing round-robin circuits into few pieces. In the second part of this section, our concern is the identification of well-known codes and families of codes that can attain universal sets of logical gates through pieceable fault tolerance.

A. Pieceable constructions in general

To find logical uses for round-robin circuits, we define a particular $h + 1$ qubit gate, a generalization of C^hZ . Given a trit string $\{j_0, j_1, \dots, j_h\}$, with $j_m \in \{1, 2, 3\}$, let

$$\Gamma(\sigma_{j_0}, \sigma_{j_1}, \dots, \sigma_{j_h}) = A^\dagger (C^hZ) A, \quad (29)$$

where $A = \bigotimes_{m=0}^h K^{3-j_m}$. Recall that $(\sigma_1, \sigma_2, \sigma_3) = (X, Y, Z)$ and $K = SH$. Notice that $\Gamma(Z, Z, \dots, Z) = C^hZ$. Likewise, $\Gamma(X, Z, \dots, Z)$ is a C^hX with target the zeroth qubit, and $\Gamma(X, X, \dots, X)$ is a symmetric gate analogous to C^hZ but in the X basis. We also abuse notation slightly to write Γ gates in terms of Pauli operators, like $\Gamma(p)$, where $p \in \mathcal{P}_{h+1}$ and $|p| = h + 1$. So $\Gamma(Z^{\otimes h+1}) = C^hZ$ as well.

With round-robin circuits we can at least implement logical versions of all Γ gates on all stabilizer codes. We save the argument for fault tolerance of the construction for later.

Theorem V.1.—Consider any $h + 1$ stabilizer codes $L_0, L_1, L_2, \dots, L_h$ each encoding a single qubit. For all h and all $p \in \mathcal{P}_{h+1}$ with $|p| = h + 1$, there is a round-robin circuit, perhaps conjugated by local Clifford gates, that implements logical $\Gamma(p)$.

Proof.—Choose a logical p_j operator for code j and denote it \bar{p}_j . For instance, if $p_j = Z$ then \bar{p}_j indicates a logical Z operator of code j . With these choices of logical operators made, build the following circuit.

- (1) By applying local Clifford gates, change every \bar{p}_j into a form consisting of only Pauli Z 's and I 's with positive sign. This of course changes the stabilizer

and other normalizers of the codes accordingly. We call this the Z form of the code with respect to \bar{p}_j .

- (2) Apply the round-robin circuit of C^hZ on the sets $\text{supp}(\bar{p}_0), \text{supp}(\bar{p}_1), \dots, \text{supp}(\bar{p}_h)$.
- (3) Reverse the Clifford gates of step (1) to return the codes to their original form.

The only nontrivial part of proving this “prologue, round-robin, epilogue” construction works is in showing that a round-robin C^hZ circuit acts as logical $\Gamma(p)$ on codes in Z form. To that purpose, from now on we just assume the $h + 1$ codes are already in Z form.

How does $\Gamma(p)$ act on Paulis? It should be clear, generalizing Eq. (16) for the CZ and Eq. (20) for the CCZ, that $\Gamma(p)q_j\Gamma(p)$ is q_j if $q_j \in \{I, p_j\}$ and $q_j\Gamma(p_0, \dots, p_{j-1}, p_{j+1}, \dots, p_h)$ otherwise (where it is to be understood that the h -qubit Γ gate here applies to all qubits except qubit j). We have to argue the same statements hold logically after the round-robin circuit of C^hZ 's and, moreover, that the stabilizer is preserved.

Tackle the latter point first. A stabilizer of the j th code $s \in S_j$, since it commutes with \bar{p}_j and \bar{p}_j is in a form consisting of all Z 's or I 's, must have an even number of X 's and Y 's on $\text{supp}(\bar{p}_j)$. Thus, the round-robin CZ circuit will preserve all stabilizers s . Likewise, \bar{p}_j is also preserved.

Now, to argue that the other normalizers transform correctly, we argue by induction on h . Start with $h = 1$, where the round-robin circuit is of CZ gates. A normalizer $l \in \mathcal{N}(S_0) \setminus S_0$ of the zeroth code that is not a logical p_0 operator will necessarily anticommute with \bar{p}_0 . Therefore, l has an odd number of X 's and Y 's on $\text{supp}(p_0)$, and so $l \rightarrow l \otimes p_1$ under the action of the round-robin CZ circuit. The symmetric argument holds for the first code block.

Now, the inductive step with $h + 2$ code blocks. The same fact stands that for code j if $l \in \mathcal{N}(S_j) \setminus S_j$ and $l \notin \bar{p}_j S_j$, then $\{l, \bar{p}_j\} = 0$, and so l has an odd number of X 's and Y 's on $\text{supp}(\bar{p}_j)$. However, now this means that

$$l \rightarrow l \prod_{\substack{q_i \in \text{supp}(\bar{p}_i) \\ i \neq j}} C^hZ(q_0, \dots, q_{j-1}, q_{j+1}, \dots, q_{h+1}).$$

But by induction, the product of C^hZ gates is logical, $\Gamma(p_0, \dots, p_{j-1}, p_{j+1}, \dots, p_{h+1})$. This is exactly the effect we want. ■

To show that the circuits presented in the proof of Theorem V.1 are pieceably fault tolerant requires specification of where and how intermediate error correction should be done. We put additional conditions on a stabilizer code to guarantee that such intermediate error correction can be done. The idea behind these conditions is simply a generalization of the procedure used in the 5-qubit and 7-qubit cases. At an intermediate error correction, we first measure the constant stabilizers, which remain Pauli throughout the round-robin circuit, and correct errors that

we can correct before measuring any non-Pauli stabilizers [using some measurement circuit like Fig. 1(d)]. Our conditions amount to the existence of an appropriate group of constant stabilizers for *each* code involved.

Because each code can be different, it makes sense to place conditions on the individual codes rather than all $h+1$ codes together. To do this, we should define constant stabilizers for the individual codes. We could consider taking from the stabilizer S of code j only the stabilizers that commute with the round-robin circuit C . But recall from the proof of Theorem V.1 that the round-robin circuit itself depends on a normalizer $p \in \mathcal{N}(S) \setminus S$ of code j —it is wired to all qubits in the support of p . Therefore, in the end, the constant stabilizer of code j depends only on which normalizer p we choose. We therefore propose the constant stabilizer of a normalizer be defined as

$$S_C(p) = \{g \in S : \forall i \in \text{supp}(p), [g_i, p_i] = 0\}. \quad (30)$$

Notice that $S_C(p)$ is a subgroup of S , and, in fact, $S_C(p)$ is a proper subgroup if and only if $d \geq 2$, where d is the code distance. Examples for the 5-qubit code include $S_C(\text{ZZZZZ}) = \{\text{IIIII}\}$ and $S_C(\text{XIZIX}) = \langle \text{IXZZX}, \text{XZZXI} \rangle$. We emphasize that this definition of the constant stabilizer by way of a normalizer, Eq. (30), is really a special case of the more general definition, Eq. (6), when the pieceable circuit in question is a round-robin circuit like in Theorem V.1.

The same logic that changed the constant stabilizer from depending on the round-robin circuit to depending on a normalizer applies to the contagious errors. Accordingly, define the contagious errors of a normalizer p (on an n -qubit code block) to be

$$\mathcal{E}_C(p) = \{E \in \mathcal{P}_n : |E| = 1, [E, p] \neq 0\}. \quad (31)$$

We include the additional condition $|E| = 1$, because after a single fault contagious errors can never affect more than one qubit per code block in our round-robin constructions.

It is important to study how $S_C(p)$ and $\mathcal{E}_C(p)$ interact. After all, we see in the 5-qubit and 7-qubit code examples that the constant stabilizer detecting contagious errors is key to PARSEC. So we say that $S_C(p)$ is error detecting if all errors in $\mathcal{E}_C(p)$ anticommute with some member of $S_C(p)$. Likewise, we say $S_C(p)$ is error correcting if it can distinguish between all contagious errors, at least modulo noncontagious errors. These definitions can be rephrased in terms of the distance of a certain classical code related to the stabilizer code and normalizer in question (see Appendix A). This relation to classical coding should not be unexpected—after the local Clifford transformation of the codes to Z form in Theorem V.1, the stabilizers with only Z 's on the active qubits are the constant ones, and the contagious errors are X 's (and Y 's, but these are just X 's modulo a noncontagious Z).

The following theorem establishes pieceable fault tolerance for the round-robin constructions of Theorem V.1, along an upper bound on the number of pieces required

Theorem V.2.—Consider $h+1$ stabilizer codes $L_0, L_1, L_2, \dots, L_h$ such that

- (1) Each L_j encodes a single qubit and has code distance $d_j \geq 3$.
- (2) For all j , L_j has a logical p_j operator with an error-correcting constant stabilizer and weight, say, w_j .

Then, there is a pieceably fault-tolerant circuit for logical $\Gamma(p)$ using $\prod_{j=0}^h m_j / \min_j m_j$ pieces, where $m_j = \lceil w_j / (d_j - 1) \rceil$.

Proof.—We first establish that a decomposition $C = C_m \cdot C_{m-1} \dots C_1$ of the round-robin circuit C is good, in the sense that intermediate error correction after each C_k will succeed in correcting all contagious errors, if for all $k = 1, 2, \dots, m$ and all $i, j \in \{0, 1, \dots, h\}$ all qubits in Λ_i are connected to at most $d_j - 1$ qubits in Λ_j by gates in C_k . Once we come up with an intermediate error-correction procedure that works given this connectivity condition, we set about piecing the circuit to achieve it.

All intermediate error corrections proceed according to PARSEC, Table I. We go through this general procedure now, and indicate where we use the assumptions of the theorem. First, measure all constant stabilizers of all the code blocks. There are now three cases to consider.

- (1) If two or more blocks have triggered, we know that a $C^h Z$ has failed, because there is no other way to get a contagious error in more than one block. Since the constant stabilizers are error correcting, and we know the errors are indeed contagious, we can locate the affected qubits and apply X , converting them to at most noncontagious Z errors on the active qubits. However, we must also note which qubits the affected qubits had interacted with during the last piece of the circuit—those qubits might have picked up Z errors. Luckily, there are at most $d_j - 1$ such qubits in block j by our piecing of the circuit. Make note of those locations and inform the final error correction.
- (2) If only one block (say, j) has triggered, we do not yet know whether the constant stabilizers have detected a contagious error on the active qubits or an error on the idle qubits, but we do know there is at most a single-qubit error in this block. To find out if it is contagious, measure the nonconstant stabilizers of block j . Fortunately, we can do so [with a circuit like Fig. 1(d)] because we have ensured a reliability guarantee on this nonconstant stabilizer measurement by having detected that some fault had already occurred. We also have commutation guarantees for the nonconstant stabilizers of block j , since no contagious errors have been detected on the other blocks. The nonconstant stabilizers of block j have

the form of a Pauli on the qubits of block j and $C^{h-1}Z$ gates on the other blocks. The commutation guarantees assure us that only the Pauli part will be involved in error detection. Since this Pauli part is the same as it was in the original code L_j , we have reduced the problem to correcting a single error with code L_j . Code L_j can do this correction modulo its own stabilizers. However, the nonconstant stabilizers have since changed, so we must ensure that the error correction does not happen modulo the nonconstant stabilizers. The fact that the constant stabilizer of block j is error correcting ensures this, however. More precisely, there can be no nonconstant stabilizers of code L_j with weight two [see Appendix A, Lemma A.1(4)]. After correcting the error, if it was contagious, make note of where it could have spread to other code blocks, and inform the final error correction.

- (3) If no blocks have triggered, at most there is a single-qubit noncontagious error per block. These errors will not spread, and they can be corrected by the final error correction.

The final error correction at the end of the round-robin circuit is special in a different way than the intermediate error corrections. Since the code space is once again stabilizer, we can measure a complete generating set. Normally, this would allow us to correct at least one error per block (since $d_j \geq 3$ for all j), and it will do so if case (3) occurred in the intermediate error corrections. However, if case (1) or (2) occurred in an intermediate error correction, then we instead switch to correcting at most $d_j - 1$ noncontagious errors in block j , using the information about the locations of those errors, which was sent to the final error correction by the intermediate error correction that detected them.

One might wonder if something like PARSEC is really necessary for intermediate error correction. In Appendix B, however, we show that, provided we use lowest weight normalizers as our \bar{p}_j , intermediate in the round-robin circuit the code space is nonstabilizer. Therefore, *some* procedure different from the conventional stabilizer error correction is required.

Now we can group the physical C^hZ gates of the round-robin circuit C according to the connectivity condition—in any one piece of the circuit, a qubit is connected to at most $d_j - 1$ qubits of block j . First, take the w_j qubits of L_j and partition them into sets of size at most $d_j - 1$. Thus, $\Lambda_j = \lambda_{j1} \cup \lambda_{j2} \cup \dots \cup \lambda_{jm_j}$, where $m_j = \lceil w_j / (d_j - 1) \rceil$. Note that $C = R_M \cdot R_{M-1} \dots R_1$, where R_j are round-robin circuits of C^hZ on sets $\lambda_{0i_0}, \lambda_{1i_1}, \dots, \lambda_{hi_h}$ for some i_k indices dependent on j and $M = m_0 m_1 \dots m_h$. Effectively, each code block is now made of m_j “composite qubits” and C can be viewed as a round-robin circuit of “composite gates” (themselves round-robin circuits) on these composite

qubits. The connectivity rule for pieces on the composite objects is that within a piece no composite qubit may be involved in more than one composite gate. Since the smallest set of composite qubits has size $\min_j m_j$, this is how many composite gates we can fit into one piece. Thus, $\prod_{j=0}^h m_j / \min_j m_j$, as we wanted to show. ■

Whether the number of pieces used by Theorem V.2 is optimal is a question that is maybe best answered for specific codes. See, for example, the improvements over Theorem V.2 that we achieve for the 7-qubit code in Sec. IV C. In addition, even ordering the C^hZ gates differently within a piece changes the set of possible errors at an intermediate point. In general, any code will have unused syndromes at an intermediate error correction (such as we note for the 5-qubit code CZ), which may be exploitable through clever, code-specific circuit design.

It is interesting that, generically, the number of pieces required to implement a pieceably 1-fault-tolerant Γ gate will not depend on the distance of the code (although the circuit size will). For instance, Theorem V.2 implies that, if we can use lowest weight normalizers to wire up the round-robin circuit (i.e., $w_j = d_j$ for all j), then implementing a $h + 1$ qubit Γ gate uses only 2^h pieces, independent of any d_j . Intuitively, this is because a large distance d code, although requiring d^{h+1} physical Γ gates, also allows us to squeeze more gates into a single piece, while still guaranteeing correctable errors.

Finally, we note that the fact that each code L_j encodes a single qubit is not necessary in the proofs of Theorems V.1 and V.2. The same arguments go through for doing Γ gates between any $h + 1$ encoded qubits, taking (your choice of) one encoded qubit from each $k \geq 1$ code block. Generally, however, because we are not considering gates between encoded qubits in the same block anyway, we prefer to simplify to the $k = 1$ case.

B. Universality through pieceable means

We now use the theorems from the previous section to construct universal sets of fault-tolerant logical gates on nondegenerate stabilizer codes. To do so, we have to prove facts about the constant stabilizers of stabilizer codes. The main technical result, proven in Appendix A, is the following lemma.

Lemma V.3.—Given a nondegenerate $[[n, 1, d]]$ stabilizer code with $d \geq 3$ and stabilizer S , if $p \in \mathcal{N}(S) \setminus S$ has $|p| = d$, then $S_C(p)$ is error correcting.

It follows that, since all stabilizer codes have a normalizer with weight d , all nondegenerate stabilizer codes have a normalizer with an error-correcting constant stabilizer. Therefore, using Theorem V.2, we have that any nondegenerate stabilizer code with $d \geq 3$ has pieceably fault-tolerant circuits for at least one of $\Gamma(ZZZ), \Gamma(XXX), \Gamma(YYY)$. With appropriate single-qubit gates, along with state preparation and measurement circuits we know exist

for any stabilizer code, any one of these can form a universal set. In fact, for each of the three Γ gates, adding a single-qubit Clifford gate that does not commute with the gate will suffice for universality. For example, $\Gamma(ZZZ)$ plus H is universal [39,40], as is $\Gamma(YYY)$ plus SHS . The octahedral gates of Eq. (11) commute with none of these three Γ gates, and so always complete a universal set.

The upshot of the brief argument of the previous paragraph is the following corollaries.

Corollary V.4.—Any nondegenerate stabilizer code with a complete fault-tolerant set of logical local Clifford gates has a universal set of fault-tolerant logical gates.

Corollary V.5.—Any nondegenerate linear stabilizer code with distance $d \geq 3$ has a fault-tolerant and universal set of logical gates.

Obviously, the assumptions of Corollary V.4 are a bit stronger than necessary given our argument above, but with this statement we have removed reference to constant stabilizers. Corollary V.5 follows from the fact that GF(4)-linear codes (like the 5-qubit code) always have transversal octahedral gates, such as K .

We have a similar logical progression for CSS codes. In this case, from Appendix A we have the lemma

Lemma V.6.—Any nondegenerate $[[n, 1, d]]$ CSS code with $d \geq 3$ and stabilizer S has two normalizers $p_1, p_2 \in \mathcal{N}(S) \setminus S$ with $S_C(p_1)$ and $S_C(p_2)$ both error correcting and $\{p_1, p_2\} = 0$.

The corresponding corollary is

Corollary V.7.—Any nondegenerate CSS code with a fault-tolerant logical local Clifford gate has a universal set of fault-tolerant logical gates.

This corollary is true because by Lemma V.6 nondegenerate CSS codes have pieceably fault-tolerant circuits for at least two of $\Gamma(ZZZ), \Gamma(XXX), \Gamma(YYY)$. Every single-qubit Clifford gate fails to commute with at least two of these three Γ gates, and, as we note above Corollary V.4, noncommutation suffices for universality. The conclusion of Corollary V.7 would similarly hold true for any code satisfying the conclusion of Lemma V.6.

Finally, we have a code-switching corollary.

Corollary V.8.—For any two nondegenerate stabilizer codes, each of which either (a) is CSS with a logical local Clifford gate or (b) possesses a complete fault-tolerant set of logical local Cliffords, there exists a logical SWAP between them.

This is simply a result of having CX from one code to the other, in both directions. It is perhaps more interesting to note the implication of a scheme for doing universal computation with any nondegenerate CSS code L through code switching, employing an ancilla code block. Using Lemma V.6, we agree to encode L such that p_1 is a logical Z operator and p_2 is logical X . Then, code switching is possible between L and any code indicated in Corollary V.8, including codes like the 7-qubit and 15-qubit, which together provide a universal set of transversal gates. The

same argument also works for any code satisfying the conclusion of Lemma V.6, and can also work to interact qubits encoded in the same block by switching some to ancillary blocks first. These possibilities demonstrate some of the ability of pieceable fault tolerance once functional ancillas are allowed.

Degenerate codes are conspicuously absent from our results in this section. However, it should be noted that Theorem V.2 does not require nondegeneracy, it only requires error-correcting constant stabilizers (condition 2). Nondegeneracy simply makes it easy to prove the existence of such a constant stabilizer (Lemma V.3), but the converse is not true. In fact, some degenerate codes do indeed have error-correcting constant stabilizers, such as Shor's 9-qubit code [41] for some logical Z operators. Moreover, we can even show (Appendix C) that it is possible for pieceable constructs to exist despite violating the second condition of Theorem V.2. Finally, it is also worth noting that because we are dealing with 1-fault-tolerance, an essentially distance-three concept, that really nondegenerate in the lemmas and corollaries of this section means no weight-two stabilizers. This provides a strengthening of Lemmas V.3 and V.6 (proofs follow from Lemma A.1 in Appendix A) from requiring a nondegenerate code to requiring a code without weight-two stabilizers, though, again, considering only 1-fault-tolerance, these are essentially the same idea regardless.

VI. CONCLUSION

Although we develop fault-tolerant universal sets of logical gates for many stabilizer codes, there is more design space to explore. For instance, it is natural to ask questions such as, what other logical gates can be implemented in a pieceably fault-tolerant manner, with or without functional ancillas, or, what are the fewest functional ancillas necessary to implement a given gate pieceably fault tolerantly in a given code? Some interesting target gates for these questions might be the controlled-Hadamard and controlled-phase gates, which are powerful gates from a universality perspective as well [42], or important single-qubit gates such as T [43]. Similarly, universal gates between logical qubits encoded in the same code block are also worth studying.

Another direction to explore is in finding more codes that support universal sets of pieceable fault-tolerant gates. All of our results, namely, Corollaries V.4, V.5, and V.7, provide sufficient conditions for pieceable universality, but not necessarily necessary ones. We already raised the question of degenerate codes, and if there is always some pieceable Γ gates that can be constructed on them, like for Shor's 9-qubit code. Of course, once functional ancillas are allowed, magic states can also be used, so perhaps this question would make the most sense with an ancilla restriction.

Moreover, we restrict ourselves to concatenated codes, an assumption it would be interesting, and important, to lift. When we think about nonconcatenated code families, such as surface codes, the poor scaling of the depth of our pieceable circuits with respect to code distance d (and, correspondingly, with code size) coupled with the fact that noncontagious errors may build up over the circuit (as they are not always corrected at intermediate error corrections) suggests no threshold would exist.

A related extension is to consider more robust pieceable fault tolerance in higher distance codes. We build logical circuits that can recover from *single* faults, regardless of the distance d of the code, and through concatenation we achieve tolerance to an arbitrary number of faults. Still, if the distance of the base code were $d = 2t + 1$, it would be nice to design logical circuits that could, even before any concatenation, recover from t faults, or, in other words, are t -fault-tolerant. For instance, our circuits on distance d , nondegenerate CSS codes can be pieced so that they are t -fault-tolerant, since in that case, using CSS PARSEC, t contagious errors are intermediately correctable even without measuring the nonconstant stabilizers. However, can PARSEC, and, in particular, the concept of a reliability guarantee for measuring nonconstant stabilizers, generalize to provide t -fault-tolerant pieceable gates on other codes?

Finally, there is the question of efficiency. A complete comparison of pieceable fault tolerance against other known means of achieving logical universality on several codes with respect to fault-tolerance threshold, ancilla qubits required, and circuit depth is warranted. Also, we expect that along with rigorous threshold calculations will come an inevitable wave of micro-optimizations to pieceable fault tolerance as we lay out here. For instance, changing from a CAT state syndrome measurement scheme for intermediate error correction to something like a Steane [28,29] or Knill [30] scheme (except with the added difficulty of non-Pauli measurements) is likely to have threshold increasing effects. We also cannot ignore the possibility of macro-optimizations—for instance, replacing the round-robin circuit with a simpler, but still pieceably fault tolerant, design customized for a particular code.

ACKNOWLEDGMENTS

The authors would like to thank Andrew Cross, Graeme Smith, and John Smolin for insightful discussions about this work. Special thanks go to Cody Jones for suggesting the circuit in Eq. (28) and David Poulin for pointing out Ref. [15]. We gratefully acknowledge funding from NSF RQCC Project No. 1111337 and ARO quantum algorithms program. T.J.Y. is thankful for support from a National Defense Science and

Engineering Graduate (NDSEG) fellowship. R.T. acknowledges the support of the Takenaka scholarship and Frank fellowship.

APPENDIX A: CLASSICAL CODES WITHIN QUANTUM ONES

There is a useful description of the constant stabilizer $S_C(p)$ [see Eq. (30)] that comes along with an (efficient) algorithm for finding it. Recall that the stabilizer S of an $[[n, k, d]]$ quantum code can be written as a bifurcated binary matrix [25,44],

$$S = [A|B], \quad (\text{A1})$$

with $n - k$ rows and $2n$ columns. The j th row corresponds to stabilizer generator \tilde{Z}_j . The k th Pauli of \tilde{Z}_j is I, X, Y, Z if and only if $(A_{jk}, B_{jk}) = (0, 0), (1, 0), (1, 1), (0, 1)$. The signs of the stabilizer generators, either ± 1 , can be stored separately, but are unnecessary for our purposes. The normalizer l can likewise be written as $[a|b]$, for (row) vectors a and b of length n .

Now consider two transformations of this representation. First, we apply local Clifford gates to put p into Z form, $[\vec{0}|b']$. This causes S to change correspondingly as well. Then, restrict S and p to the “window” $W = \text{supp}(p)$ by removing all columns from their binary matrix representations that correspond to qubits outside W . Now $p_W = [\vec{0}|\vec{1}]$ for length $|p|$ vectors of 0’s and 1’s, and $S_W = [A'|B']$. Since we can swap qubits within W (corresponding to switching columns of S_W) and multiply stabilizer generators (corresponding to adding rows of S_W modulo 2) without changing p_W , we can perform Gaussian elimination so that

$$S_W = \left[\begin{array}{cc|cc} I_r & D & B_1 & B_2 \\ 0 & 0 & B_3 & B_4 \end{array} \right]. \quad (\text{A2})$$

Letting r be the rank of A' , the heights of the blocks are r and $n - k - r$ and the widths are $r, |p| - r, r, |p| - r$. So I_r is the $r \times r$ identity matrix. Note $r \leq |p| - 1$, because the first r rows of S_W represent stabilizers that must commute with p , and the fact that I_r has only one 1 per row implies D must have an odd number of 1’s per row, and, thus, its width cannot be zero.

We remark offhand that, for codes encoding one qubit, if p has minimum weight within its coset (i.e., $|p| = \min\{|p'| : p' \in pS\}$), then a stronger fact is true: namely, $r = |p| - 1$. This is so because all errors restricted to the support of p , consisting of only Pauli Z ’s and I ’s, and with weight $< |p|$ are detectable. Exactly half of these detectable errors must also be correctable, because, for any such error E , only E and pE will have the same syndrome. Thus,

$$\frac{1}{2} \left[\binom{|p|}{1} + \binom{|p|}{2} + \cdots + \binom{|p|}{|p|-1} \right] = 2^{|p|-1} - 1 \quad (\text{A3})$$

such errors are correctable. Only the first r stabilizers in S_W have the possibility to trigger on these errors. Thus, to have enough unique syndromes to correct them all, we must have $r \geq |p| - 1$.

Continuing our discussion of the constant stabilizer $S_C(p)$, we now notice that $S_C(p)$ is generated by the stabilizers corresponding to the last $n - k - r$ rows of S_W . We observe that those same rows $H = [B_3 B_4]$ represent the parity check matrix of a classical code. We call this the classical code $C(p)$ induced by p on quantum code L . Not all rows of H are necessarily linearly independent, but the essential parity checks can be recovered by removing linearly dependent rows. It is also worth mentioning that the first r rows of S_W correspond to a minimal generating set for the nonconstant stabilizers.

The error-correcting properties of $C(p)$ are directly related to the error-correcting properties of $S_C(p)$. Indeed, $S_C(p)$ is error detecting if and only if $C(p)$ has classical code distance $d_{C(p)} \geq 2$. Likewise, $S_C(p)$ is error correcting if and only if $C(p)$ has classical code distance $d_C \geq 3$. The classical errors detected or corrected, respectively, are exactly the contagious errors from $\mathcal{E}_C(p)$, Eq. (31).

We now show the following.

Lemma A.1.—Let L be an $\llbracket n, 1, d \rrbracket$ stabilizer code with stabilizer S and $p \in \mathcal{N}(S) \setminus S$.

- (1) If $d \geq 2$ and $|p| < 2d - 1$, then $C(p)$ exists and has distance $d_{C(p)} \geq 2$.
- (2) If p is minimum weight within its coset, there are no nonconstant stabilizers of weight two, and $C(p)$ exists with $d_{C(p)} \geq 2$, then $d_{C(p)} \geq 3$.
- (3) If $d \geq 3$, $|p| = d$, and L is nondegenerate, then $d_{C(p)} \geq 3$.
- (4) If $d_{C(p)} \geq 3$, then there are no nonconstant stabilizers of weight two.
- (5) If L is nondegenerate CSS and l consists only of Pauli Z 's and I 's or only of X 's and I 's, then $d_{C(p)} \geq d$.

Given the correspondence between the error-correcting properties of $S_C(p)$ and $C(p)$ discussed above, it should be clear that part (3) of Lemma A.1 results directly in the proof of Lemma V.3. Part (5) of Lemma A.1 implies Lemma V.6 after one notices that, for any CSS code, there is always a normalizer in Z form (only Z 's and I 's) and another in X form (only X 's and I 's), and these belong to different cosets of the stabilizer [45]. Part (4) of Lemma A.1 is used in the proof of Theorem V.2.

Proof of Lemma A.1.—Note that (3) is merely a consequence of (1) and (2). To prove the first two, we actually show something stronger: if $d \geq 2$ and for all

$l \in \mathcal{N}(S) \setminus S$ we have $d_{C(l)} \leq 2$, then there is a weight-two stabilizer. The argument here proceeds recursively—either we find a weight-two stabilizer or we find a normalizer with smaller weight than we began with. At the bottom of the recursion (i.e., $|p| < 2d - 1$ or p minimum weight within its coset), we will get (1) and (2).

We work with the windowed stabilizer, Eq. (A2), windowed on $l \in \mathcal{N}(S) \setminus S$. If $H = [B_3 B_4]$ does not exist (i.e., $r = n - k$), then we choose a column from $B = [B_1 B_2]$, say, column k , and define $\mathcal{B} = \{j : B_{jk} = 1\}$. Then $e = \prod_{j \in \mathcal{B}} Z_j X_k$, whose weight notice is $|e| \leq |l|$, commutes with all stabilizers but anticommutes with l . Therefore, $e \in \mathcal{N}(S) \setminus S$ and $el \in \mathcal{N}(S) \setminus S$. So, $|e| \geq d$ and $|l| - |e| + 1 \geq d$ together imply $d \leq |e| \leq |l| - d + 1$. Thus, because $d \geq 2$, $|e| \leq |l| - 1$, and we find a lower weight normalizer.

If $H = [B_3 B_4]$ does exist, then it either has a column of 0's (distance $d_{C(l)} = 1$) or has two identical columns (distance $d_{C(l)} = 2$). In the former case, let the column of 0's be k and define $\mathcal{B}' = \{j : B_{jk} = 1\}$ and $e = \prod_{j \in \mathcal{B}'} Z_j X_k$. The argument proceeds as before, so we omit it. But in the latter case, say, the identical columns are k_1 and k_2 . Define $\mathcal{B}_{12} = \{j : B_{jk_1} \neq B_{jk_2}\}$ and $e = \prod_{j \in \mathcal{B}_{12}} Z_j X_{k_1} X_{k_2}$. Note $|e| \leq |l|$ and additionally e commutes with all stabilizers and l , so $e \in lS$ or $e \in S$. In the former case, if $|e| = |l|$, then $el \in S$ has weight two. In the latter case, $el \in lS$ and $|el| \leq |l|$. If $|el| = |l|$, then $\mathcal{B}_{12} \subseteq \{k_1, k_2\}$, implying $|e| = 2$.

The assumptions of (1) mean that if we started this recursive process with $l = p$, we would only be able to conclude that H has no column of 0's and, thus, $d_{C(p)} \geq 2$. Likewise, if we started with the assumptions of (2) instead, the conclusion would be $d_{C(p)} \geq 3$.

Part (4) is simple to prove in contrapositive. If there is a nonconstant stabilizer of weight two, it must have the form $X_{k_1} X_{k_2}$, $X_{k_1} Y_{k_2}$, $Y_{k_1} X_{k_2}$, or $Y_{k_1} Y_{k_2}$ on two qubits $k_1, k_2 \in \text{supp}(p)$. Since X and Y are the same from the perspective of commutation with constant stabilizers, we can consider just the first case. In that case, the constant stabilizers cannot distinguish between X_{k_1} and X_{k_2} , which means $C(p)$ is not error correcting.

Finally, we finish with the last part, part (5). This one is comparatively simple. Recall the stabilizer of a CSS code can always be written like [18,46]

$$S = \left[\begin{array}{c|c} H_1 & 0 \\ \hline 0 & H_2 \end{array} \right]. \quad (\text{A4})$$

Thus, if p is in Z form (consists only of Z 's and I 's), then H_2 is the check matrix of the classical code $C(p)$. But since the quantum code has distance d and is nondegenerate, this classical code must have distance at least d as well. Likewise, if p is in X form, then H_1 is the check matrix of $C(p)$, which must also have distance at least d . ■

APPENDIX B: NONSTABILIZER INTERMEDIATE CODES

Here, we show that the codes we encounter intermediate in the round-robin constructions of Theorem V.1 are in fact nonstabilizer codes. This implies that doing complete error correction, rather than just correcting contagious errors as we do with PARSEC, will likely require the discovery of new tools. Only a few examples are known of doing error correction on nonstabilizer codes, such as CWS codes [47].

In particular, we prove the following:

Lemma B.1.—Assume that we use normalizers with weight minimal within their coset to construct logical Γ gates with $h \geq 2$ in Theorem V.1. At any point in the round-robin circuit, if the code space is not equal to the code space at the beginning of the round-robin circuit, the code space is nonstabilizer.

Proof.—Assume we have converted each of the $h + 1$ code blocks into Z form. We take $h \geq 2$ because our conclusion is obviously false for round-robin CZ constructions. Then the stabilizer of any one code block, before any gates of the round-robin are applied, takes the form of $S_j = \langle t_{j1}, \dots, t_{jr}, s_{j1}, \dots, s_{jm} \rangle$, where $m = n - 1$. We write nonconstant stabilizer generators as t and constant stabilizer generators as s . We assume that we have found a maximal set of constant stabilizers (e.g., using the methods of Appendix A). Also, let \bar{p}_j denote the Z -form normalizer of code j , the one with minimal weight.

Let S denote the complete stabilizer of all $h + 1$ codes before any round-robin gates have been applied. The transformation of S after some number of gates, a subcircuit C , is denoted S' , and the transformation of S_j as S'_j . Notice that $t'_{jk} = t_{jk}G_{jk}$, where G_{jk} is a product of $C^{h-1}Z$ gates on the qubits in code blocks other than block j . However, this is the extent of the transformation of the stabilizer—the constant stabilizers are preserved: $s'_{jk} = s_{jk}$.

There are three properties of G_{jk} to note: (1) $G_{jk}^2 = I$, (2) $G_{jk} = (1/M_{jk})\sum_m g_{jk}^m$, where g_{jk}^m are Pauli operators (with ± 1 signs) consisting of only Z 's and I 's supported on $\bigcup_j \text{supp}(\bar{p}_j)$ and $M_{jk} > 0$ is normalization, (3) $[G_{jk}, s_{uv}] = 0$ for all j, k, u, v .

Let P' denote the projector onto the code space after C . Explicitly,

$$P' = \frac{1}{N_S} \sum_{g \in S'} g = \frac{1}{N_S} \sum_{g \in S} CgC^\dagger, \quad (\text{B1})$$

where N_S is just normalization, so that $P'^2 = P'$. We want to show that there is no subgroup T of the Pauli group such that

$$P' \propto \sum_{g \in T} g. \quad (\text{B2})$$

To prove P' is not of this stabilizer form, it is sufficient to show that two terms in the Pauli decomposition of P' have

coefficients that differ in magnitude. Since the constant stabilizers are part of the Pauli decomposition of P' and their coefficients have unit magnitude, we only need to find a Pauli in the decomposition with a coefficient of magnitude less than one.

If all G_{jk} are identity, we are in the original code space. Therefore, one of G_{jk} is not identity. Now, G_{jk} cannot be a single Pauli term $G_{jk} = g_{jk}^0$. This is because g_{jk}^0 , as a nonidentity Pauli consisting of Z 's and I 's, has some -1 eigenstate $|b\rangle$ with b a weight-one bit string. However, for G_{jk} , as a product of $C^{h-1}Z$ gates with $h \geq 2$, all eigenstates of this form have eigenvalue $+1$. Since G_{jk} consists of at least two different terms, then $1/M_{jk}$, the normalization of G_{jk} , must be at most $1/\sqrt{2}$.

We now argue that, for all m , $t_{jk}g_{jk}^m$ appears only once in the Pauli decomposition of P' , namely, as part of the decomposition of $t_{jk}G_{jk}$. If no other $g_{jk}^l \neq g_{jk}^m$ exists such that $g_{jk}^l g_{jk}^m$ is a stabilizer, this is certainly true. After all, t_{jk} as a nonconstant stabilizer can only appear on block j from one combination of stabilizer generators, namely, from $t_{jk}G_{jk}$ itself. Indeed, it cannot be the case that $g_{jk}^l g_{jk}^m$ is a stabilizer, as this would imply [since both g_{jk}^l and g_{jk}^m are supported only in $\bigcup_j \text{supp}(\bar{p}_j)$ and only by Z 's] a constant stabilizer exists in at least one block $i \neq j$ that could be multiplied by the normalizer \bar{p}_i to get a lower weight normalizer. ■

This lemma can be used to verify that an intermediate code space is nonstabilizer. In particular, it says, in the case of using lowest weight normalizers for the construction, that if any nonconstant stabilizer t_{jk} takes the intermediate form $t_{jk}G_{jk}$, where G_{jk} is a product of $C^{h-1}Z$ gates not equal to identity, then the code space is nonstabilizer.

APPENDIX C: DEMONSTRATING PIECEABILITY BEYOND THEOREM V.2

In this Appendix, we address the question of whether condition (2) of Theorem V.2 is necessary for a fault-tolerant pieceable circuit implementing logical $\Gamma(p)$ to exist. The answer is “no.” To show this, we provide a simple example implementing CZ on Shor's 9-qubit code. Thus, codes and gates $\Gamma(p)$ satisfying the conditions of Theorem V.2, such as the 5-qubit and 7-qubit codes in the main text, are put in context as simply convenient examples of pieceable fault tolerance, rather than the extent of its applicability.

Recall that Shor's 9-qubit code [41] is the concatenation of two redundancy codes, one L_Z correcting phase-flip errors and one L_X correcting bit flips. We take L_Z as the outer code, thus forming the stabilizer as

$$S_9 = \left\langle \begin{array}{ccc} X_1 X_2 X_3 X_4 X_5 X_6 & Z_1 Z_2 & Z_2 Z_3 \\ X_4 X_5 X_6 X_7 X_8 X_9 & Z_4 Z_5 & Z_5 Z_6 \\ & Z_7 Z_8 & Z_8 Z_9 \end{array} \right\rangle. \quad (\text{C1})$$

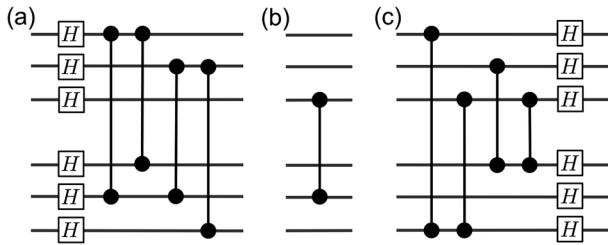


FIG. 6. A pieceable implementation of CZ on Shor's 9-qubit code using three pieces (a)–(c). Only active qubits, namely, 1, 2, 3 from each code block, are shown. Since the circuit is Clifford, intermediate error correction consists of standard Pauli measurements.

Logical operators are taken as $\bar{Z}_9 = X^{\otimes 9}$ and $\bar{X}_9 = Z^{\otimes 9}$, defined this way according to convention [26]. Shor's code is CSS, so it possesses a transversal $\Gamma(ZX) = CX$ gate.

Using a computer and the methods of Appendix A, it is easy to check that, while Shor's code does possess logical X operators with error-correcting constant stabilizers (such as $Z_1Z_4Z_7$), no logical Z operator has an error-correcting constant stabilizer. This makes a pieceable $\Gamma(XX)$ gate [or $\Gamma(XXX)$ for that matter] implementable by Theorem V.2, but not a $\Gamma(ZZ) = CZ$ gate. Gauge fixing can implement a logical Hadamard on Shor's code [48], and thus implement CZ from CX [not to mention completing a universal gate set when placed alongside pieceable $\Gamma(XXX)$]. But the question remains whether gauge fixing is necessary for a logical CZ, or if pieceable alone is enough.

It turns out CZ is implementable in pieceable fashion, as shown in Fig. 6. Only the active qubits are shown, which can be taken as qubits 1, 2, 3 from each block. This pieceable circuit is clearly the round-robin circuit that implements logical CZ according to Theorem V.1. However, it is not and cannot be pieced as specified by Theorem V.2 while remaining fault tolerant. In fact, even the order of CZ gates within a piece is important to the success of the intermediate error corrections. By exhaustive computer search over permutations of the CZ gates and placement of the intermediate error corrections, three is the smallest number of pieces making this round-robin circuit on Shor's code fault tolerant.

APPENDIX D: CIRCUITS FOR RESOURCE COMPARISON

In Sec. IV C, we count ancilla qubits, multqubit gates, and circuit volume for two implementations of logical CCZ on the 7-qubit code—magic-state injection via postselection and pieceable fault tolerance. Here, we draw the circuits we use in both cases. Figure 7 shows the pieceable design, while Figs. 8–12 show the components of the magic state design. The captions explain their circuit volumes. The number of ancilla qubits and number of CX and CCZ gates involved are also easily countable from the diagrams.

Of course, there are potential improvements and trade-offs in both circuit designs. For instance, reducing the size of the CAT states in the magic-state preparation from 4

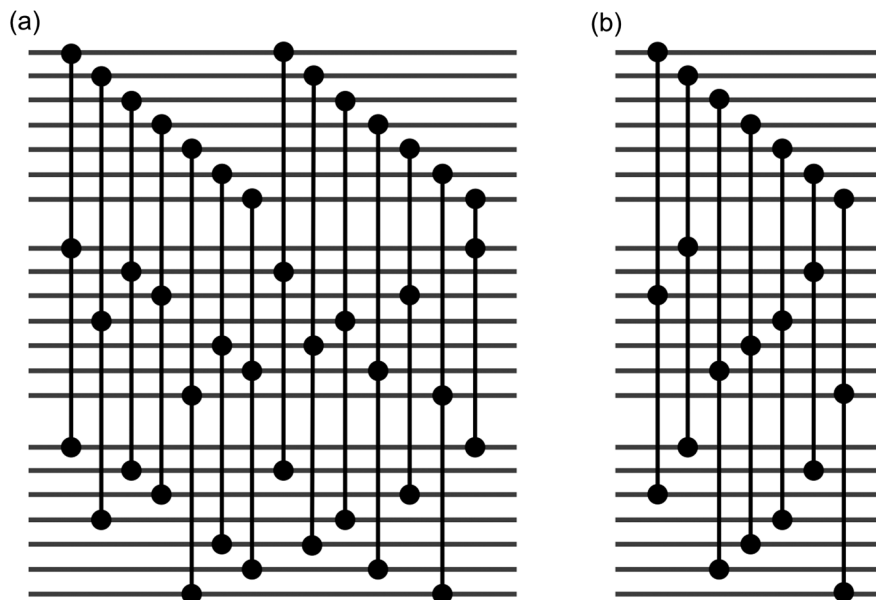


FIG. 7. A pieceable circuit for logical CCZ on the 7-qubit code that uses just 21 physical CCZ gates and two pieces (a), (b). Intermediate error correction consists of Steane syndrome measurement of the Z stabilizers on each block. The final error correction is full Steane syndrome measurement on all blocks. We assume that recovery need not be applied directly. Instead, X errors detected intermediately can be dealt with by adaptively changing the following CCZ gates to be $|0\rangle$ controlled on the affected qubits. The first piece has volume $2 \times 21 = 42$ and the second just 21. Using the circuit volumes of the measurement circuits in Fig. 10, we find the total volume of the pieceable CCZ to be $3 \times 21 + 3 \times 81 + 3 \times 155 = 771$.

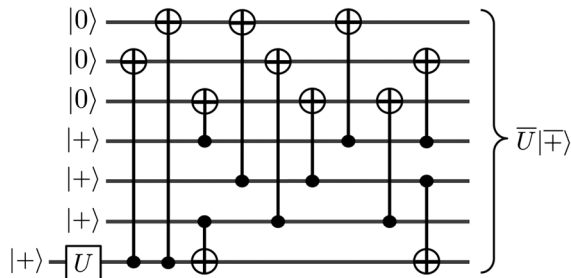


FIG. 8. The non-fault-tolerant circuit, based off the design in Ref. [37], for encoding a single-qubit state into the 7-qubit code. Pushing state preparations as far right as possible, we find that this construction has circuit volume 34. To make $|\overline{CCZ}\rangle$ (assuming no faults), we can use three such circuits on three physical qubits in the state $|\overline{CCZ}\rangle = CCZ|+\rangle^{\otimes 3}$.

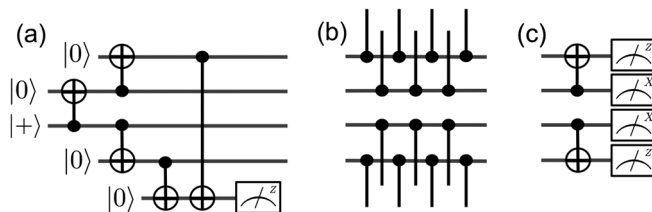


FIG. 9. (a) Preparing and verifying $4CAT = (|0000\rangle + |1111\rangle)/\sqrt{2}$ (the measurement is postselected on finding $|0\rangle$). The circuit volume is 26, and we must wait for the postselection to succeed before proceeding. (b) The circuit coupling the 4CAT to the $|\overline{CCZ}\rangle$ blocks in Fig. 11. All outgoing lines are to unique qubits, so this takes 4 time steps for a total circuit volume of 16. (c) The CAT decoding [49] and measurement step. Successful postselection occurs upon seeing $|0\rangle$ from both Z-basis measurements. The sum modulo two of the X-basis measurements gives the eigenvalue of the measured operator. The volume of this step is 8. Total volume of (a)–(c) is 50.

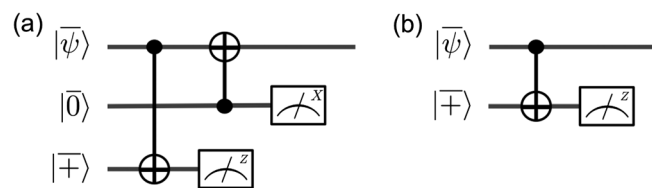


FIG. 10. (a) Measurement of the stabilizers of the 7-qubit code block $|\bar{\psi}\rangle$ as put forward by Steane [28]. Gates and measurements are drawn to represent transversal versions. Both $|\bar{0}\rangle$ and $|\bar{+}\rangle$ are verified logical states that we assume are created by Goto’s postselection [37] circuit with just 1 qubit for verification (so 8 qubits total), which has circuit volume 53. The total circuit volume of the complete Steane stabilizer measurement is $53 \times 2 + 49 = 155$. (b) The measurement of just the Z stabilizers on the 7-qubit code using a verified $|\bar{+}\rangle$ state. Total circuit volume is 81.

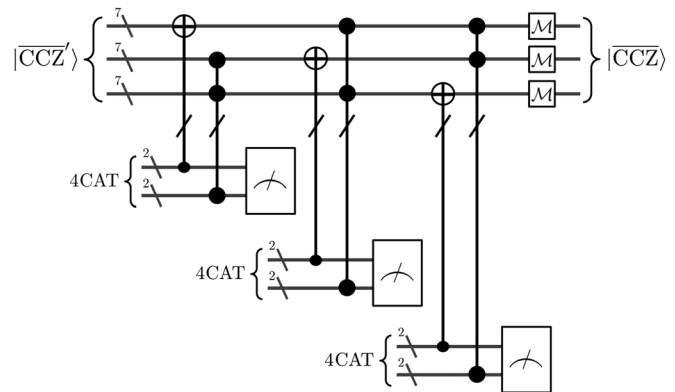


FIG. 11. The postselected verification of the potentially faulty $|\overline{CCZ}'\rangle$ state that is output from the non-fault-tolerant preparation in Fig. 8. The slashed CX and CCZ gates are controlled on the 4CATs as shown in Fig. 9(b) and targeted as transversal X and transversal CZ on the encoded blocks. The 4CAT inputs are prepared as in Fig. 9(a) and the measurement boxes on the 4CATs represent Fig. 9(c). We use \mathcal{M} to represent measurement of all 7-qubit code stabilizers using Steane states, Fig. 10(a). In this case, a successful postselection occurs upon a trivial syndrome from all \mathcal{M} and from the CAT measurements. The total circuit volume is $34 \times 3 + 50 \times 3 + 4 \times 21 \times 3 + 155 \times 3 = 969$.

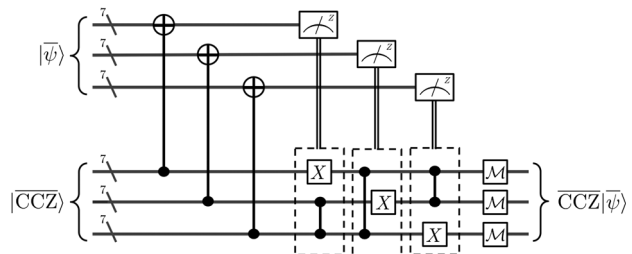


FIG. 12. Teleporting a CCZ gate into code blocks using a $|\overline{CCZ}\rangle$ state verified using Fig. 11. All gates and measurements are transversal. Again, \mathcal{M} is the measurement of all 7-qubit stabilizers using Steane states, Fig. 10(a). Assuming the classically controlled operations are never done, the total circuit volume, the complete volume of magic-state injection with postselected preparation, is $969 + 42 \times 2 + 155 \times 3 = 1518$.

qubits to 2 is possible but will lead to increased circuit volume, as coupling to the $|\overline{CCZ}\rangle$ register will take more additional volume than is removed by preparing a smaller CAT. However, such optimizations are unlikely to erase the nearly 50% improvement (see Table III) that a pieceable CCZ offers.

[1] A. M. Steane, *Error Correcting Codes in Quantum Theory*, *Phys. Rev. Lett.* **77**, 793 (1996).
 [2] X. Chen, H. Chung, A. W. Cross, B. Zeng, and I. L. Chuang, *Subsystem Stabilizer Codes Cannot Have a Universal Set of*

- Transversal Gates for Even One Encoded Qudit*, *Phys. Rev. A* **78**, 012353 (2008).
- [3] B. Eastin and E. Knill, *Restrictions on Transversal Encoded Quantum Gate Sets*, *Phys. Rev. Lett.* **102**, 110502 (2009).
- [4] B. Zeng, A. Cross, and I. L. Chuang, *Transversality versus Universality for Additive Quantum Codes*, *IEEE Trans. Inf. Theory* **57**, 6272 (2011).
- [5] S. Bravyi and A. Kitaev, *Universal Quantum Computation with Ideal Clifford Gates and Noisy Ancillas*, *Phys. Rev. A* **71**, 022316 (2005).
- [6] S. Bravyi and J. Haah, *Magic-State Distillation with Low Overhead*, *Phys. Rev. A* **86**, 052329 (2012).
- [7] A. G. Fowler, M. Mariantoni, J. M. Martinis, and A. N. Cleland, *Surface Codes: Towards Practical Large-Scale Quantum Computation*, *Phys. Rev. A* **86**, 032324 (2012).
- [8] A. G. Fowler, S. J. Devitt, and C. Jones, *Surface Code Implementation of Block Code State Distillation*, *Sci. Rep.* **3**, 1939 (2013).
- [9] C. D. Hill, A. G. Fowler, D. S. Wang, and L. C. L. Hollenberg, *Fault-Tolerant Quantum Error Correction Code Conversion*, *Quantum Inf. Comput.* **13**, 439 (2013).
- [10] 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).
- [11] A. Paetznick and B. W. Reichardt, *Universal Fault-Tolerant Quantum Computation with Only Transversal Gates and Error Correction*, *Phys. Rev. Lett.* **111**, 090505 (2013).
- [12] H. Bombín, *Gauge Color Codes: Optimal Transversal Gates and Gauge Fixing in Topological Stabilizer Codes*, *New J. Phys.* **17**, 083002 (2015).
- [13] H. Bombín, *Dimensional Jump in Quantum Error Correction*, *New J. Phys.* **18**, 043038 (2016).
- [14] T. Jochym-O'Connor and R. Laflamme, *Using Concatenated Quantum Codes for Universal Fault-Tolerant Quantum Gates*, *Phys. Rev. Lett.* **112**, 010505 (2014).
- [15] E. Knill, R. Laflamme, and W. Zurek, *Threshold Accuracy for Quantum Computation*, [arXiv:quant-ph/9610011](https://arxiv.org/abs/quant-ph/9610011).
- [16] E. Knill and R. Laflamme, *Concatenated Quantum Codes*, [arXiv:quant-ph/9608012](https://arxiv.org/abs/quant-ph/9608012).
- [17] P. Aliferis, D. Gottesman, and J. Preskill, *Quantum Accuracy Threshold for Concatenated Distance-3 Codes*, *Quantum Inf. Comput.* **6**, 97 (2006).
- [18] A. R. Calderbank and P. W. Shor, *Good Quantum Error-Correcting Codes Exist*, *Phys. Rev. A* **54**, 1098 (1996).
- [19] H. Bombin and M. A. Martin-Delgado, *Quantum Measurements and Gates by Code Deformation*, *J. Phys. A* **42**, 095302 (2009).
- [20] H. Bombin, *Clifford Gates by Code Deformation*, *New J. Phys.* **13**, 043005 (2011).
- [21] J. Preskill, *Reliable Quantum Computers*, *Proc. R. Soc. A* **454**, 385 (1998).
- [22] E. Knill and R. Laflamme, *Theory of Quantum Error-Correcting Codes*, *Phys. Rev. A* **55**, 900 (1997).
- [23] D. Gottesman, *Class of Quantum Error-Correcting Codes Saturating the Quantum Hamming Bound*, *Phys. Rev. A* **54**, 1862 (1996).
- [24] 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).
- [25] D. Gottesman, *Stabilizer Codes and Quantum Error Correction*, [arXiv:quant-ph/9705052](https://arxiv.org/abs/quant-ph/9705052).
- [26] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information* (Cambridge University Press, Cambridge, England, 2010).
- [27] P. W. Shor, in *Proceedings of the 37th Annual Symposium on Foundations of Computer Science, 1996* (IEEE, New York, 1996), pp. 56–65.
- [28] A. M. Steane, *Active Stabilization, Quantum Computation, and Quantum State Synthesis*, *Phys. Rev. Lett.* **78**, 2252 (1997).
- [29] A. M. Steane, *Space, Time, Parallelism and Noise Requirements for Reliable Quantum Computing*, *Fortschr. Phys.* **46**, 443 (1998).
- [30] E. Knill, *Quantum Computing with Realistically Noisy Devices*, *Nature (London)* **434**, 39 (2005).
- [31] C. Zalka, *Threshold Estimate for Fault Tolerant Quantum Computation*, [arXiv:quant-ph/9612028](https://arxiv.org/abs/quant-ph/9612028).
- [32] C. H. Bennett, D. P. DiVincenzo, J. A. Smolin, and W. K. Wootters, *Mixed-State Entanglement and Quantum Error Correction*, *Phys. Rev. A* **54**, 3824 (1996).
- [33] R. Laflamme, C. Miquel, J. P. Paz, and W. H. Zurek, *Perfect Quantum Error Correcting Code*, *Phys. Rev. Lett.* **77**, 198 (1996).
- [34] D. Gottesman, *Theory of Fault-Tolerant Quantum Computation*, *Phys. Rev. A* **57**, 127 (1998).
- [35] A. M. Steane, *Quantum Reed-Muller Codes*, *IEEE Trans. Inf. Theory* **45**, 1701 (1999).
- [36] D. Gottesman and I. L. Chuang, *Demonstrating the Viability of Universal Quantum Computation Using Teleportation and Single-Qubit Operations*, *Nature (London)* **402**, 390 (1999).
- [37] H. Goto, *Minimizing Resource Overheads for Fault-Tolerant Preparation of Encoded States of the Steane Code*, *Sci. Rep.* **6**, 19578 (2016).
- [38] C. Jones, *Low-Overhead Constructions for the Fault-Tolerant Toffoli Gate*, *Phys. Rev. A* **87**, 022328 (2013).
- [39] Y. Shi, *Both Toffoli and Controlled-NOT Need Little Help to Do Universal Quantum Computing*, *Quantum Inf. Comput.* **3**, 84 (2003).
- [40] D. Aharonov, *A Simple Proof That Toffoli and Hadamard are Quantum Universal*, [arXiv:quant-ph/0301040](https://arxiv.org/abs/quant-ph/0301040).
- [41] P. W. Shor, *Scheme for Reducing Decoherence in Quantum Computer Memory*, *Phys. Rev. A* **52**, R2493 (1995).
- [42] A. Y. Kitaev, *Quantum Computations: Algorithms and Error Correction*, *Russ. Math. Surv.* **52**, 1191 (1997).
- [43] P. O. Boykin, T. Mor, M. Pulver, V. Roychowdhury, and F. Vatan, *A New Universal and Fault-Tolerant Quantum Basis*, *Inf. Proc. Lett.* **75**, 101 (2000).
- [44] S. Aaronson and D. Gottesman, *Improved Simulation of Stabilizer Circuits*, *Phys. Rev. A* **70**, 052328 (2004).
- [45] M. M. Wilde, *Logical Operators of Quantum Codes*, *Phys. Rev. A* **79**, 062322 (2009).
- [46] A. Steane, *Multiple-Particle Interference and Quantum Error Correction*, *Proc. R. Soc. A* **452**, 2551 (1996).
- [47] Y. Li, I. Dumer, M. Grassl, and L. P. Pryadko, *Structured Error Recovery for Code-Word-Stabilized Quantum Codes*, *Phys. Rev. A* **81**, 052337 (2010).
- [48] P. Aliferis and A. W. Cross, *Subsystem Fault Tolerance with the Bacon-Shor Code*, *Phys. Rev. Lett.* **98**, 220502 (2007).
- [49] D. P. DiVincenzo and P. Aliferis, *Effective Fault-Tolerant Quantum Computation with Slow Measurements*, *Phys. Rev. Lett.* **98**, 020501 (2007).