Fault-tolerant gates via homological product codes

Tomas Jochym-O'Connor[*]
Walter Burke Institute for Theoretical Physics, Institute for Quantum Information & Matter, California Institute of Technology, Pasadena, CA 91125, USA

A method for the implementation of a universal set of fault-tolerant logical gates is presented using homological product codes. In particular, it is shown that one can fault-tolerantly map between different encoded representations of a given logical state, enabling the application of different classes of transversal gates belonging to the underlying quantum codes. This allows for the circumvention of no-go results pertaining to universal sets of transversal gates and provides a general scheme for fault-tolerant computation while keeping the stabilizer generators of the code sparse.

I. INTRODUCTION

Quantum error correction extends qubit coherence times through error mitigation and will be a requirement for any large-scale quantum computation. In this vein, tremendous research efforts have been placed on finding quantum error correcting codes that may be realizable in both the near and distant future. Among the leading candidates for experimental implementation are 2D topological stabilizer codes, such as the toric code [1, 2], which allow for the correction of errors by measuring small-weight local checks while protecting logical information in highly non-local degrees of freedom [3–7]. These codes are experimentally appealing due to their stabilizers being low-weight, thus minimizing the effect of noise during measurement. They can be generalized to higher spatial dimensions, again with the stabilizer generators being relatively low-weight, and provide theoretically simple implementations of different classes of fault-tolerant logic [8–13]. The motivation for quantum error correcting codes to have geometrically local stabilizers (in a given dimension) is to simplify experimental architectures, yet this may not necessarily be a hard requirement. However, the need for low-weight stabilizer checks is much stronger, as larger weight checks lead to more noise propagation, and generally lower threshold error rates. The theory of quantum low-density parity check (LDPC) codes has been developed to address such concerns and finding good codes with low-weight checks remains a very active area of research [14–19]. This work will not focus on the development of such codes, but rather will center on how to generally use such codes for the purposes of fault-tolerant computation.

Obtaining a universal set of fault-tolerant gates is complicated by the existence of no-go results for such constructions using only transversal gates [20, 21]. However, many alternative schemes have been developed to circumvent this restriction. They rely on the preparation of special ancillary states and gate teleportation [22–24], or tailored fault-tolerant constructions for certain classes of codes [8, 25–32]. This work extends the set of fault-tolerant alternatives, presenting a scheme for fault-tolerant logic on any CSS code while keeping the underlying stabilizer measurements low-weight.

We present a method for implementing fault-tolerant logical gates in a homological product code [16]. Namely, given the homological product of two quantum codes, we show how to map in a fault-tolerant manner between the encoded homological product logical state to a logical state specified by one of the two codes. Then, if the underlying codes have a set of transversal gates, such logical gates can be applied and the state can be re-encoded back into the full codespace fault-tolerantly. There are no restrictions on the underlying codes, other than having to be defined by a boundary operator $\delta : \mathcal{C} \to \mathcal{C}$ such that $\delta^2 = 0$ in the linear space $\mathcal{C}$. In particular, by using versions of the 2D and 3D color codes [9, 33] as the underlying codes in the construction, a universal set of fault-tolerant operations can be implemented.

The article is organized as follows: In Sec. II we review the theory of CSS codes defined by chain complexes and the construction of the homological product codes, carefully considering their underlying structure. In Sec. III we present the main result, a fault-tolerant method to implement a logical gate using homological product codes as well as discuss a simple decoding procedure. In Sec. IV we present examples of codes that exhibit a set of universal fault-tolerant gates, expanding on the notions of code padding and doubling. Finally, we conclude with some remarks and open questions in Sec. V.

II. HOMOLOGICAL PRODUCT CODES

A. Single sector theory

We begin by reviewing the connection between CSS codes and homology. Namely, as in Ref. [16], we focus on single sector theory in $\mathbb{Z}_2$, that is a chain complex defined by a linear space $\mathcal{C}$ and a linear boundary operator $\delta : \mathcal{C} \to \mathcal{C}$, such that $\delta^2 = 0$. We can then use such a boundary operator to define a CSS code [34, 35], that is a stabilizer code whose generators can be expressed as either X-type or Z-type.

Let $\mathcal{C}$ be a n-dimensional binary space $\mathbb{Z}_2^n$, then δ will be a $n \times n$ binary matrix. The (perhaps over-complete)

[*] tjoc@caltech.edu

generating set of X stabilizers will be given by the rows of δ, that is for a given row, a generator S_{X_i} will have X support on the qubits with a 1 in the given row. Similarly, the Z stabilizers will be defined by the columns of the matrix. Given $\delta^2 = 0$, we are thus assured commutativity of the stabilizers. The number of independent generators of both type will be $\mathrm{rank}(\delta)$, and as such the number of logical qubits of the code will be $k = n - 2\mathrm{rank}(\delta)$.

For the remainder of this section we shall focus on the reverse implication. That is, given a CSS code whose X and Z stabilizer spaces are of the same dimension, one can construct a boundary operator δ of a single-sector theory. The results presented are a fairly straightforward corollary of Lemma 3 from Ref. [16], yet we include them here for completeness and to review some important concepts, namely the canonical boundary operator.

Lemma 1. *Let $\mathcal{C}$ be a CSS code on n qubits, whose X and Z stabilizer spaces are each of cardinality 2^l. The code therefore encodes $k = n - 2l$ qubits. Then, there exists a invertible matrix U, and canonical boundary operator δ_0 defined as:*

$$\delta_0 = \begin{pmatrix} 0_k & 0 & 0 \\ 0 & 0_l & \mathbb{1}_l \\ 0 & 0_l & 0_l \end{pmatrix}, \tag{1}$$

such that $\delta = U\delta_0 U^{-1}$, where the rows (columns) of δ contain a set of generators of the X (Z) stabilizer group.

Proof. Given the existence of a CSS code, by the Gottesman-Knill theorem [36] there exists a unitary operator U composed solely of CNOT gates that maps $|\psi\rangle \otimes |0\rangle^{\otimes l} \otimes |+\rangle^{\otimes l}$ to the encoded stabilizer code, where $l = (n-k)/2$ and $|\psi\rangle$ is a k-qubit state. This statement can be expressed in terms of matrix manipulation on $\mathbb{Z}_2$, where δ_0 will represent the initial state of the stabilizers before the application of the encoding circuit, that is the rows of δ_0 represent the initial $|+\rangle$ states, and the columns the initial $|0\rangle$ states. A CNOT gate with qubit i as control, and qubit j as target can then be expressed according to the invertible matrix:

$$U_{i,j} = \mathbb{1} + v_i v_j^T, \tag{2}$$

where v_k is the standard basis column vector one non-zero entry at position k. The action of $U_{i,j}$ by conjugation maps column c_j to the sum of columns $c_i \oplus c_j$, and row r_i to the sum of rows $r_i \oplus r_j$. This is the exact action required from a CNOT, as it maps X_i to X_iX_j and Z_j to Z_iZ_j. Note, as required for a valid representation of a CNOT gate, $U_{i,j}^2 = \mathbb{1} \Rightarrow U_{i,j} = U_{i,j}^{-1}$, where again we are working modulo 2. Then, the encoding unitary U can be broken into its CNOT components and can be expressed as $U = U_{i_N,j_N} \cdots U_{i_2,j_2} U_{i_1,j_1}$. As such,

$$\delta = (U_{i_N,j_N} \cdots U_{i_2,j_2} U_{i_1,j_1})\delta_0 (U_{i_N,j_N} \cdots U_{i_2,j_2} U_{i_1,j_1})^{-1} \tag{3}$$

$$= (U_{i_N,j_N} \cdots U_{i_2,j_2} U_{i_1,j_1})\delta_0 (U_{i_1,j_1} U_{i_2,j_2} \cdots U_{i_N,j_N}), \tag{4}$$

will be a valid representation of the stabilizers of the code. Since U is a valid representation of the encoding circuit of the code, the rows (columns) of δ will remain a valid representation of the X (Z) stabilizers since they were so for δ_0. □

Perhaps as importantly for the purposes of this article, if the given CSS code has a generating set of stabilizers that are sparse, then the resulting constructed δ will be sparse. We define the *sparsity* of a code to be the smallest integer w such that there exists a set of generators of the code whose weights are at most w while any given qubit participates in at most w stabilizer checks.

Corollary 2. *Let $\mathcal{C}$ be a CSS code on n qubits with an equal number of X and Z stabilizers and sparsity w. Then, a boundary operator δ can be constructed such that no row or column will have more than w^2 non-zero entries.*

Proof. Given some sparse representative set of stabilizers $\{S_{X_i}, S_{Z_i}\}$, as in the proof of Lemma 1, a unitary encoder U can be chosen that maps $Z_{k+i} \to S_{Z_i}$ and $X_{k+l+i} \to S_{X_i}$. Consider the right action of U^{-1} in terms of its action on the canonical boundary operator:

$$\delta_0 U^{-1} = \begin{pmatrix} 0_{k\times n} \\ s_{X_1} \\ \vdots \\ s_{X_l} \\ 0_{l\times n} \end{pmatrix}, \tag{5}$$

where on the right side of the equality we have a matrix whose rows are either all-zero or a binary representation s_{X_i} of the stabilizer S_{X_i}. This follows from the fact that the right application of U^{-1} results in the propagation of the initial X stabilizers to their final generator form. Then, by the sparsity of the stabilizer generators, each row will be of weight at most w and each column will have weight at most w. Now consider the left application of U applied to $\delta_0 U^{-1}$, thus completing the conjugation, the resulting matrix $U\delta_0 U^{-1}$ will have rows that will be sums of the different rows of $\delta_0 U^{-1}$. Moreover, each row of $U\delta_0 U^{-1}$ will be a sum of at most w rows s_{X_i}, and will as such be of weight at most w^2. Finally, since a given row can map to at most w other rows, each non-zero entry within a column of $\delta_0 U^{-1}$ can map to at most w entries within that column. Therefore, since there were at most w non-zeros entries in a column of $\delta_0 U^{-1}$, there can be at most w^2 non-zeros in each column of $U\delta_0 U^{-1}$ □

B. Homological Product Construction

Given two complexes $(\mathcal{C}_1, \delta_1)$, $(\mathcal{C}_2, \delta_2)$ with their associated spaces and single sector boundary operators, we define a new operator as in Ref. [16],

$$\partial = \delta_1 \otimes \mathbb{1} + \mathbb{1} \otimes \delta_2, \tag{6}$$

acting on $\mathcal{C}_1 \otimes \mathcal{C}_2$. It follows from $\delta_i^2 = 0$ that $\partial^2 = 0$, again since we are working in $\mathbb{Z}_2$. Therefore, $(\mathcal{C}_1 \otimes \mathcal{C}_2, \partial)$ is a valid single sector complex, defining its own quantum CSS code.

We now restate some important properties of the homological product.

Lemma 3 ([16]). *Let $(\mathcal{C}_1, \delta_1)$, $(\mathcal{C}_2, \delta_2)$ be complexes defining codes with k_1, and k_2 logical operators, respectively. Let $\partial = \delta_1 \otimes \mathbb{1} + \mathbb{1} \otimes \delta_2$, then the resulting complex $(\mathcal{C}_1 \otimes \mathcal{C}_2, \partial)$ will encode $k = k_1 k_2$ logical qubits and*

$$\ker \partial = \ker \delta_1 \otimes \ker \delta_2 + \operatorname{im} \partial. \qquad (7)$$

Suppose that w_a is the sparsity of δ_a. Moreover, let d_a^X, d_a^Z be the X and Z distances for the corresponding codes. Then, the weight and distances of the new code can be bounded according to the parameters of the original code.

Lemma 4 ([16]). *The sparsity of ∂ is upper bounded by $w_1 + w_2$. The X and Z distance of the new code satisfy the following bounds:*

$$\max\{d_1^\alpha, d_2^\alpha\} \le d^\alpha \le d_1^\alpha d_2^\alpha, \qquad \alpha = X, Z. \qquad (8)$$

C. Encoding the homological product code

In this subsection, we review some facts about the encoding circuit for ∂ [16]. As eluded to in Section II A, there are invertible matrices U_a such that $\delta_a = U_a \delta_{a,0} U_a^{-1}$, where $\delta_{a,0}$ are the canonical boundary operators for δ_a. The matrices U_a are binary representatives of the encoding circuit for the given code, and as such, by taking their tensor product we obtain the encoding operation for ∂. That is:

$$\begin{aligned}\partial &= (U_1 \otimes U_2)(\delta_{1,0} \otimes \mathbb{1} + \mathbb{1} \otimes \delta_{2,0})(U_1^{-1} \otimes U_2^{-1}) \qquad (9)\\ &:= (U_1 \otimes U_2)\partial_0 (U_1 \otimes U_2)^{-1}, \qquad (10)\end{aligned}$$

where we have defined ∂_0 to be the canonical boundary operator for ∂. We can express ∂_0 in matrix form as follows:

$$\begin{aligned}\partial_0 &= (\delta_{1,0} \otimes \mathbb{1} + \mathbb{1} \otimes \delta_{2,0}) \qquad (11)\\ &= \begin{pmatrix} \mathbb{1}_{k_1} \otimes \delta_{2,0} & 0 & 0 \\ 0 & \mathbb{1}_{l_1} \otimes \delta_{2,0} & \mathbb{1}_{l_1} \otimes \mathbb{1}_{n_2} \\ 0 & 0 & \mathbb{1}_{l_1} \otimes \delta_{2,0} \end{pmatrix}, \qquad (12)\end{aligned}$$

where k_i are the number of logical qubits and $l_i = (n_i - k_i)/2$ is the number of X/Z stabilizers of the given code code.

It is worth further exploring the form of ∂_0, as this will be informative of how the logical information is encoded in the code. Each row and column of ∂_0 will be of weight at most 2. Moreover, if a given row has 2 non-zeros entries, say at positions q_i and q_j, then any column with a non-zero entry at q_i will also have a non-zero entry at position q_j in order to satisfy commutativity. As such, these rows and columns represent an initial entangled Bell pair between qubits q_i and q_j since they will be stabilized by the operators $X_{q_i} X_{q_j}$ and $Z_{q_i} Z_{q_j}$.

The initial state can be pictorially represented by Fig. 1, where along a fixed row and column, the states in $\mathcal{C}_1$ and $\mathcal{C}_2$ are fixed, respectively. Then, the tensor product binary operators $U_1 \otimes \mathbb{1}$ and $\mathbb{1} \otimes U_2$ will have geometric meaning in this picture. Note for the remainder of this work, we will denote W_i as the physical encoding

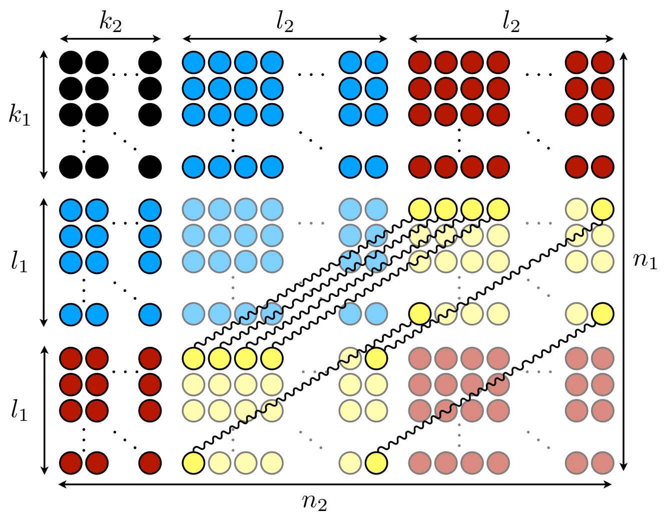

FIG. 1. Initial state of the homological product code prior to encoding [16]. Each circle represents a qubit, with the black qubits representing those holding the logical information to be encoded. Blue qubits are prepared in the $|0\rangle$ state, while red qubits are prepared in $|+\rangle$. The yellow qubits joined by an oscillating edge are prepared in a Bell pair $(|00\rangle + |11\rangle)/\sqrt{2}$.

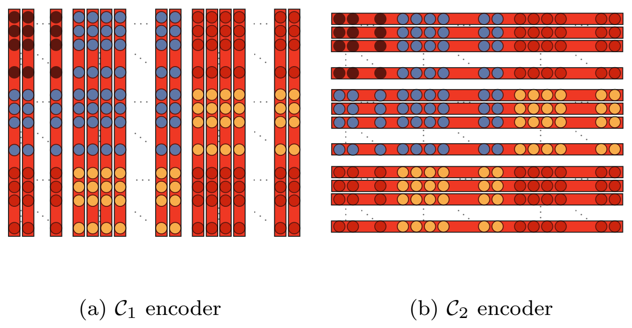

FIG. 2. Schematic representation of the support of the encoding circuits for each of the codes underlying the homological product code, defined by the boundary operator $\partial = \delta_1 \otimes \mathbb{1} + \mathbb{1} \otimes \delta_2$. The overall encoding circuit has a binary representation of the operator $U_1 \otimes U_2$, which acts on the initial state represented in Fig. 1. That is, the physical encoding unitary W_1 for the code $\mathcal{C}_1$ will act on every column on qubits from Fig. 1, and conversely the physical encoding unitary W_2 will act on every row.

unitaries composed of CNOT gates acting on the quantum states whose binary representation is given by U_i. Thus W_1 will couple qubits within vertical bands, while W_2 will couple qubits within horizontal bands, as represented in Fig. 2.

III. FAULT-TOLERANT LOGICAL GATES

A. Partial decoding of the homological product code

The key idea for expanding the set of available fault-tolerant logical gates will be for the two underlying codes composing the homological product to have different complimentary sets of transversal gates. Then, we can achieve the application of these logical gates by only decoding one of the two underlying codes, while remaining protected by the other. This is reminiscent of the scheme for implementing fault-tolerant gates using two concatenated codes [26], with the added advantage that the stabilizers remain low-weight in the case of the homological product.

The main result is that, while we decode one of the codes, we still remain fully protected by the other code. While errors may potentially propagate during the application of the decoding process, they can still be corrected as long as the number of faults is less than half the distance of the underlying code protecting the information (that is the code that remains encoded at all times). The main Theorem is a variant of Lemma 4, yet is proved using the concept of error bands.

Recall the homological product is defined by the complex $(\mathcal{C}_1 \otimes \mathcal{C}_2, \partial)$, where $\mathcal{C}_i$ are binary spaces. The complex defines a code with $n_1 n_2$ physical qubits. We can label the individual qubits of the quantum code defined by the complex $(\mathcal{C}_1 \otimes \mathcal{C}_2, \partial)$ using the following notation: $|\psi_{i,j}\rangle$. Then, the associated Pauli operator supported on the qubit $|\psi_{i,j}\rangle$ will be denoted by $P_{i,j}$. We will call the *error band* E^1_a to be all possible Pauli operators of the form $\prod_j P_{a,j}$, that is a product of any Pauli operators $P_{i,j}$ with $i = a$. Conversely, the error band E^2_a will be all possible Pauli operators of the form $\prod_j P_{j,a}$. Note that W_1 will only couple qubits within a fixed error bands E^1_a, while conversely W_2 will only couple qubits within fixed error bands E^2_a.

Theorem 5. *Let $(\mathcal{C}_1, \delta_1)$, $(\mathcal{C}_2, \delta_2)$ be complexes and let $(\mathcal{C}_1 \otimes \mathcal{C}_2, \partial)$ be the homological product code constructed from these codes with $\partial = \delta_1 \otimes \mathbb{1} + \mathbb{1} \otimes \delta_2$. Then any error E supported on fewer than d_2 error bands E^1_a cannot support a logical operator. Similarly, any error E supported on fewer than d_1 error bands E^2_a cannot support a logical operator.*

Proof. Consider an error E supported on fewer than d_2 error bands E^1_a, let the affected bands be denoted by the set $\{e_1, \cdots, e_l\}$, that is $E \subseteq E^1_{e_1} \cup \cdots \cup E^1_{e_l}$, where $l < d_2$. Since any error can be expressed in the Pauli basis, if we show any Pauli error supported on the above set cannot support a logical operator, E cannot support a logical operator. As such, without loss of generality, suppose E is a Pauli error. Consider the modified error $E' = W_1^\dagger E W_1$, where W_1 is the encoding unitary for the code $\mathcal{C}_1$. Then, if we can show that E' cannot support a logical operator on the new codespace after applying $W_1^\dagger$, then by unitary equivalence it cannot on the homological product codespace.

Any logical operator must commute with all of the stabilizers of the code. The modified codespace is given by the boundary operator $\partial_{1,0} = (U_1^{-1} \otimes \mathbb{1})\partial(U_1 \otimes \mathbb{1}) = \delta_{1,0} \otimes \mathbb{1} + \mathbb{1} \otimes \delta_2$, that is it will correspond to k_1 logical codeblocks encoded in the code $(\mathcal{C}_2, \delta_2)$ along with accompanying encoded ancilla state[1]. This can be view visually by considering the initial unencoded state in Fig. 1 followed by the encoding operation of Fig. 2b. Therefore, in order for E' to support a logical error, it will have to support a logical operator on one of the first k_1 error bands E^2_a. Without loss of generality, consider the first error band E^2_1, that is the first row of qubits in Figs. 1-2, and the support of E' on that band, $E'_1 = E' \cap E^2_1 \subseteq (E^1_{e_1} \cup \cdots \cup E^1_{e_l}) \cap E^2_1$. Therefore, the weight of the Pauli operator given by E'_1 is limited by the number of initial error bands E^1_a on which the error was supported, that is: $\mathrm{wt}(E'_1) \leq l < d_2$, and as such since any logical operator supported on the band E^2_1 must be of weight at least d_2, the error E'_1 cannot support a logical error. Since this will be true for all encoded logical bands supported on E^2_a, with $a \leq k_1$, the error E' cannot support a non-trivial logical error.

To conclude, since E' cannot support a logical error on the code specified by the boundary operator $\partial_{1,0}$, then $E = W_1 E' W_1^\dagger$ cannot support a logical operator on the code specified by $\partial = (U_1 \otimes \mathbb{1})\partial_{1,0}(U_1^{-1} \otimes \mathbb{1})$. □

Equipped with Theorem 5, we propose the following scheme to implement a fault-tolerant logical gate. Suppose the logical gate G_1 can be implemented transversally on the single-qubit partition of the code induced by the complex $(\mathcal{C}_1, \delta_1)$, that is it can be implemented by applying gates that are each individually supported on single qubits of the code. Then, in order to apply the logical gate G_1 on the logical state of the homological code $(\mathcal{C}_1 \otimes \mathcal{C}_2, \partial)$, we begin by unencoding the code $(\mathcal{C}_2, \delta_2)$, that is we apply the unitary $\mathbb{1} \otimes U^\dagger_{2,E}$. At this point, the first k_2 blocks of n_1 qubits remain encoded in the code $(\mathcal{C}_1, \delta_1)$, while the the remaining blocks are in an encoded ancillary state. Therefore, we can apply the transversal implementation of the logical gate G_1 on any of the k_2 logical states we desire. We complete the logical gate application by then reencoding into the code $(\mathcal{C}_1 \otimes \mathcal{C}_2, \partial)$ by applying $\mathbb{1} \otimes U_2$.

[1] An encoded ancillary state is a fixed state that contains no non-trivial logical information, yet still may be partially encoded.

The proposed scheme is fault-tolerant in that, it will be able to correct against up to $\lfloor (d_1-1)/2 \rfloor$ faults throughout the process. Any error $E_{a,b}$ that occurs during the application of either $\mathbb{1} \otimes U_2^{-1}$, $\mathbb{1} \otimes U_2$, or the transversal gate can spread to a high-weight error, yet such an error will remain within a single error band E_a^2. This follows as the application of $\mathbb{1} \otimes U_2$ only ever couples qubits within the same error band E_a^2. Similarly, a transversal gate with respect to the code $(\mathcal{C}_1, \delta_1)$ will not couple different error bands E_a^2, by definition. Therefore, any single fault $E_{a,b}$ can result in an error that is contained within the error band E_a^2. Since any logical error must be supported on at least d_1 such error bands, any error affecting less than half of such error bands must remain correctible by the Knill-Laflamme condition [37].

By symmetry, given a transversal gate G_2 on the single-qubit partition of the code $(\mathcal{C}_2, \delta_2)$, a fault-tolerant implementation of G_2 can be achieve by applying $W_1^\dagger \otimes \mathbb{1}$, followed by the transversal gate, and a reencoding $W_1 \otimes \mathbb{1}$. Such a fault-tolerant gate will be able to correct against up to $\lfloor (d_2-1)/2 \rfloor$ faults.

B. Correcting errors

In the last section, we showed how even when decoding one of the two codes, we can always protect against at least $\lfloor (d_i-1)/2 \rfloor$ faults. However, the encoding/decoding operations will themselves have d_i time steps, and as such, errors will accumulate within a given error band (assuming independent non-Markovian error processes). Moreover, at a given time step, there are d_i different locations where an error can occur, and as such the probability of an error occuring within a particular error band throughout the process will scale roughly as $p_e d_i^2$, where p_e is the physical error rate. This is undesirable from the perspective of fault-tolerance, as we hope that for a given family of codes, by growing the distance, the probability of incurring a logical error decreases exponentially below some threshold value. Yet, if the underlying error rate is growing quadratically with the distance, this yields a pseudo-threshold for each code, rather than a global threshold for a code family.

In this section, we present a simple decoding algorithm for the homological product code, based on the decoding algorithm of the individual codes composing the homological product. While the presented scheme will likely be far from ideal in many settings, it will serve as a proof of principle decoder as well as provide a means to correct against errors during the implementation of the fault-tolerant logical gates. This will help alleviate the concern errors accumulating within an error band due to the encoding/decoding having a macroscopic number of individual time steps. However, the global scheme will still not necessarily have a threshold against independent noise on all qubits, and such a threshold would have to be determined on a case-by-case basis.

Consider the homological product code as specified in the previous subsection, with a boundary operator $\partial = \delta_1 \otimes \mathbb{1} + \mathbb{1} \otimes \delta_2$. Moreover, suppose we have recovery operators $\mathcal{R}_i$ for each code that returns the code to the codespace and moreover corrects with certainty when the weight of the error is below half the distance of the respective code. We present the following Corollary to Theorem 5, which follows directly from the proof of that result.

Corollary 6 (of Theorem 5)**.** *Let $(\mathcal{C}_1, \delta_1)$, $(\mathcal{C}_2, \delta_2)$ be complexes and let $(\mathcal{C}_1 \otimes \mathcal{C}_2, \partial)$ be the homological product code constructed from these codes with $\partial = \delta_1 \otimes \mathbb{1} + \mathbb{1} \otimes \delta_2$. Then, for any error E supported on fewer than $\lfloor (d_j-1)/2 \rfloor$ error bands E_a^i, the error $E' = W_i^\dagger E W_i$ is correctible using the recovery operator $\mathcal{R}_j$, where $i,\ j \in \{1,2\}$ such that $i \neq j$.*

The above Corollary states that given a correctible error, as stated by Theorem 5, conjugating that error by either decoding operator $W_i^\dagger$ will result in a correctible error on the remaining encoded states in the code $\mathcal{C}_j$.

We propose the following decoding algorithm. Given an encoded state in the traditional homological product code, we measure the syndromes of the code as specified by the row (X type) and columns (Z type) of the boundary operator ∂. Now, given these measurement outcomes, we can map them onto syndrome outcomes for either of the two code, using the following procedure. Without loss of generality, suppose we would like to map them onto the syndromes of the code $\mathcal{C}_2$. We know the modified boundary operator $\partial_{1,0} = (U_1^{-1} \otimes \mathbb{1})\partial(U_1 \otimes \mathbb{1}) = \delta_{1,0} \otimes \mathbb{1} + \mathbb{1} \otimes \delta_2$ corresponds to k_1 logical states that are encoded in the code $\mathcal{C}_2$. Specifically, the first $k_1 n_2$ rows and columns of $\partial_{1,0}$ will correspond to the stabilizers of the code $\mathcal{C}_2$, see Eq. 12 for an example, replacing $\delta_{2,0}$ with δ_2. Suppose we measured a given stabilizer S_l of the original stabilizer code such that $ES_l = (-1)^{b_l} S_l E$, that is $b_l \in \{0,1\}$ records the stabilizer measurement outcome. Then since the encoding circuit is Clifford, and S is Pauli, we can efficiently classically compute the form of the transformed syndrome $S_l' = W_1^\dagger S_l W_1$. Moreover, S_l' will keep the same commutation relation with the transformed error E', that is $E' S_l' = (-1)^{b_l} W_1^\dagger S_l E W_1 = (-1)^{b_l} S_l' E'$. Therefore, we can use the transformed stabilizers S_l' to determine the syndrome of E' in the code $\mathcal{C}_2$. To find the appropriate recovery Pauli operator Q' we use the known decoder of $\mathcal{C}_2$, and transform Q' back into a recovery operator for the original code by classically computing $Q = W_1 Q' W_1^\dagger$, which is again efficient since W_1 is a Clifford circuit.

We can then generalize the above method to address a build up of errors throughout the fault-tolerant process presented in Sec. III A. Suppose, without loss of generality, we want to implement the logical gate G_2 which is transversal for the code $\mathcal{C}_2$. Then, we would start with unencoding $\mathcal{C}_1$ by applying $W_1^\dagger = V_1^\dagger \cdots V_t^\dagger$, where V_i are the CNOT gates used in constructing the encoding unitary W_1. After the application of each $V_i^\dagger$, the code will be partially unencoded and the resulting boundary operator will be of the form $\partial_{1,i} = \delta_{1,i} \otimes \mathbb{1} + \mathbb{1} \otimes \delta_2$, where $\delta_{1,i} = (V_i \cdots V_1)\delta_{1,0}(V_1^\dagger \cdots V_i^\dagger)$. If we assume that throughout the application of each V_i operator the generators of the code remain sparse, then we can measure these generators after each application V_i in order to address the errors that occurred during the application of that gate. The errors are then corrected by classically mapping the stabilizer generators $\partial_{1,i}$ onto those of $\partial_{1,0}$, and using, as outlined above, the decoder of $\mathcal{C}_2$ to correct for the resulting errors.

A final remark on the stabilizer generators of $\partial_{1,i}$. As stated above, if we were to measure them after every application of V_i, $V_i^\dagger$, we would like them to remain sparse. In general, while the initial and final boundary operators $\partial_{1,0}$ and ∂ are certainly sparse, there will be no guarantee that the intermediary matrices remain sparse as well. However, for many common codes, such as topological codes, this can be achieved by choosing an appropriate encoding unitary. Roughly speaking, the idea is to build up the non-local logical operators of a topological code by growing the code at its boundary in a systematic manner [38].

IV. UNIVERSAL CONSTRUCTIONS

In this section, we explore an explicit example of a family of codes for implementing a universal set of fault-tolerant operations using the construction from Section III A. We will focus on the Clifford + T universal gate set [39]. Let the code $\mathcal{C}_1$ be the 2D color code with distance d_1 encoding a single logical qubit. The code has parameters $[[c_1 d_1^2, 1, d_1]]$, for a constant c_1, and can implement any Clifford gate transversally [9, 33]. The code $\mathcal{C}_2$ will be composed of the gauge-fixed 3D color code. That is, a particular choice of the 3D color code where volume cell terms are of both X and Z type, while the face terms are only of Z type. The resulting code has a transversal implementation of a $T = \text{diag}(e^{-i\pi/8}, e^{i\pi/8})$, a non-Clifford gate and code parameters $[[c_2 d_2^3, 1, d_2]]$, for a constant c_2 [8, 9]. However, due to the asymmetry in the number of X and Z stabilizer generators, arising from having to fix the face terms to be Z-type, the resulting code cannot be directly used in the single-sector homological product code construction. We present two alternative code constructions of $\mathcal{C}_2$, one based on code padding, and one using two complementary copies and re-encoding them in the $[[4, 2, 2]]$ repetition code.

A. Code padding

Suppose we have a CSS code $\mathcal{C}$ that we want to use in a universal fault-tolerant implementation of a homological product code, and moreover assume without loss of generality there are more Z generators than X generators. In order to use the code $\mathcal{C}$ in a homological product code construction, as presented, we must have the same number of independent X and Z stabilizers. A rather simple alternative code that we can use is to pad the original code with extra ancillary qubits in the $|+\rangle$ state, thus adding an extra set of single-qubit X generators. The resulting code will have the same distance as the original code, where all non-trivial logical Pauli operators can be supported on the original qubits of the code. For example, in the case of the smallest gauge-fixed 3D color code, the 15-qubit Reed-Muller code, we can pad the code with an extra 6 $|+\rangle$ qubits, resulting in a $[[21, 1, 3]]$ code. While this extra padding of qubits does not change the base code other than trivially alternating the number of underlying stabilizer generators, these generators will play a role in the homological product, via the initial entanglement present in the unencoded state, represented by ∂_0.

Therefore, the universal scheme for implementing a set of fault-tolerant logical operations that can correct an arbitrary single qubit error will use the Steane and padded Reed-Muller codes, which are the smallest distance 3 versions of the 2D and padded 3D color codes, respectively. The overall scheme will have coding parameters $[[147, 1, 3^*]]$, where 3^* corresponds to the minimum fault-tolerant distance of the overall scheme, not necessarily the distance of the homological code itself. The stabilizer measurements will be of weight at most 15, see Appendix A. This is a large improvement over requiring measuring stabilizers of weight 28 in the concatenated scheme [26], which leads to punitive threshold values and qubit overheads [40–42].

B. Code doubling

The process of code doubling was first presented in Refs. [43, 44] for converting between Majorana fermion codes and stabilizer codes. We will outline the general logical procedure for any CSS code here, yet it can be generalized for arbitrary stabilizer code rather similarly.

Consider a CSS code $\mathcal{A}^{(1)}$ of n physical qubits with stabilizer generators $S_{X_i}^{(1)} = \prod_{j \in \mathcal{X}_i} X_j^{(1)}$, where $\mathcal{X}_i$ is a list of qubits in the support of $S_{X_i}^{(1)}$ and the use of the superscript (1) will become clear shortly. Similarly, the Z stabilizers are given by $S_{Z_i}^{(1)} = \prod_{j \in \mathcal{Z}_i} Z_j^{(1)}$. Consider now a rotated version of $\mathcal{A}^{(1)}$ where each of the X stabilizers are replaced by Z stabilizers and vice-versa, call this code $\mathcal{A}^{(2)}$. More explicitly, the X and Z stabilizers of $\mathcal{A}^{(2)}$ are given by: $S_{X_i}^{(2)} = \prod_{j \in \mathcal{Z}_i} X_j^{(2)}$, $S_{Z_i}^{(2)} = \prod_{j \in \mathcal{X}_i} Z_j^{(2)}$. Therefore, the different superscripts represent different blocks of n qubits.

These two codes are then re-encoded into the $[[4, 2, 2]]$ repetition code, whose encoding circuit is given in Fig. 3. The third block of qubits will be initially prepared as $|0\rangle^{\otimes n}$, while the fourth block will be prepared as $|+\rangle^{\otimes n}$. Consider how the stabilizers are transformed

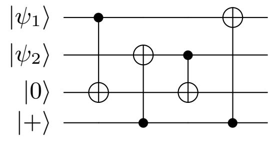

FIG. 3. Encoding circuit for the 4-qubit repetition code. The stabilizers generators of the code are: $X^{\otimes 4}$, $Z^{\otimes 4}$.

under the action of the circuit in Fig. 3:

$$S_{X_i}^{(1)} = \prod_{j\in\mathcal{X}_i} X_j^{(1)} \longrightarrow \prod_{j\in\mathcal{X}_i} X_j^{(1)} X_j^{(3)} \tag{13}$$

$$S_{X_i}^{(2)} = \prod_{j\in\mathcal{Z}_i} X_j^{(2)} \longrightarrow \prod_{j\in\mathcal{Z}_i} X_j^{(2)} X_j^{(3)} \tag{14}$$

$$X_j^{(4)} \longrightarrow X_j^{(1)} X_j^{(2)} X_j^{(3)} X_j^{(4)} \tag{15}$$

$$S_{Z_i}^{(1)} = \prod_{j\in\mathcal{Z}_i} Z_j^{(1)} \longrightarrow \prod_{j\in\mathcal{Z}_i} Z_j^{(1)} Z_j^{(4)} \tag{16}$$

$$S_{Z_i}^{(2)} = \prod_{j\in\mathcal{X}_i} Z_j^{(2)} \longrightarrow \prod_{j\in\mathcal{X}_i} Z_j^{(2)} Z_j^{(4)} \tag{17}$$

$$Z_j^{(3)} \longrightarrow Z_j^{(1)} Z_j^{(2)} Z_j^{(3)} Z_j^{(4)}. \tag{18}$$

Note that we can combine the mapped stabilizers with those of the repetition code in order to recognize a complete symmetry between the X and Z stabilizers of the code. While it may immediately follow from the presented construction, one can show that the above construction is equivalent to concatenating the $[[4,2,2]]$ code with the code $\mathcal{A}^{(1)}$ and its rotated compliment $\mathcal{A}^{(2)}$. Moreover, the distance of the new code will be twice that of the original code. Therefore, this concatenated code provides a code that can be used in the homological product code construction.

Choosing the code $\mathcal{A}^{(1)}$ to be the gauge-fixed 3D color code along with its rotated compliment $\mathcal{A}^{(2)}$, we can use these codes in conjunction with the 2D color code for the purposes of universal fault-tolerant computation via homological product codes. To perform any logical Clifford gate, it will be sufficient to decode the $[[4,2,2]]$ repetition code, followed by the decoding of $\mathcal{A}^{(i)}$, for either or both $i \in \{1,2\}$, depending on which codeblock one would like to apply the desired Clifford gate transversally.

In order to implement the T gate fault-tolerantly, one would first decode the 2D color code, as specified in Sec. III A. At this point, one could not directly apply the non-Clifford gate transversally, as the two encoded codeblocks will still be further encoded in the $[[4,2,2]]$ code. However, one can then decode the $[[4,2,2]]$ code by applying the circuit of Fig. 3 in reverse. This will preserve the protection guaranteed by Theorem 5 as each block of 4-qubits will belong to the same error band, allowing the application of a transversal T gate bookended by fault-tolerant operations.

A final note regarding code doubling: since the stabilizers are symmetrized, the code will gain a transversal Hadamard gate. The logical result of the transversal Hadamard will be to implement logical Hadamard followed by logical SWAP between the two logical qubits. As such, for this particular operation, code doubling allows for a rapid implementation of this logical gate without having to decode one of the codes in the homological product.

V. CONCLUSION

This work introduces a method for implementing a set of logical gates using homological product codes, applicable to any set of CSS codes. Namely, we show that if the underlying codes composing the homological product have complementary classes of transversal gates, then this scheme can be used to implement a fault-tolerant universal gate set. Moreover, if the underlying codes have stabilizer generators that are sparse, the construction will remain sparse, allowing for the implementation of a fault-tolerant gate set that does not require measurement of high-weight operators. This method is particularly interesting for the theory of quantum LDPC codes, where the hope would be to construct codes with good parameters and sets of transversal gates. A recent result exploring the connection between homological product codes and single-shot error correction highlights a potential avenue for constructing codes with interesting transversal gates [45], yet new constructions remain elusive.

The presented scheme relies on decoding one of the two codes composing the homological product, applying the transversal gate, and re-encoding. The encoding/decoding process may indeed spread errors in a dramatic way, yet due to the protection of the other code, the global operation remains fault-tolerant. If the encoder/decoder of each code preserves the sparsity of the code after each gate, then modified stabilizers may be measured during the encoding/decoding process, allowing for increased protection. It remains an interesting open question if whether there exists a family of codes that would allow for a fault-tolerance threshold error rate using the presented universal method.

ACKNOWLEDGMENTS

We thank Benjamin J. Brown, and Sam Roberts for insightful discussions on intermediary correction of errors during the fault-tolerant operations. We also thank Christopher Chamberland and John Preskill for comments during the development of this work. We acknowledge the support from the Walter Burke Institute for Theoretical Physics in the form of the Sherman Fairchild Fellowship as well as support from the Institute for Quantum Information and Matter (IQIM), an NSF Physics Frontiers Center (NFS Grant PHY-1733907).

[1] A. Y. Kitaev, Annals of Physics **303**, 2 (2003).
[2] A. G. Fowler, M. Mariantoni, J. M. Martinis, and A. N. Cleland, Phys. Rev. A **86**, 032324 (2012).
[3] R. Barends, J. Kelly, A. Megrant, A. Veitia, D. Sank, E. Jeffrey, T. C. White, J. Mutus, A. G. Fowler, B. Campbell, *et al.*, Nature **508**, 500 (2014).
[4] J. M. Chow, J. M. Gambetta, E. Magesan, D. W. Abraham, A. W. Cross, B. R. Johnson, N. A. Masluk, C. A. Ryan, J. A. Smolin, S. J. Srinivasan, *et al.*, Nature communications **5** (2014).
[5] D. Nigg, M. Mueller, E. A. Martinez, P. Schindler, M. Hennrich, T. Monz, M. A. Martin-Delgado, and R. Blatt, Science , 1253742 (2014).
[6] J. Kelly, R. Barends, A. G. Fowler, A. Megrant, E. Jeffrey, T. C. White, D. Sank, J. Y. Mutus, B. Campbell, Y. Chen, *et al.*, Nature **519**, 66 (2015).
[7] M. Takita, A. D. Córcoles, E. Magesan, B. Abdo, M. Brink, A. Cross, J. M. Chow, and J. M. Gambetta, Phy. Rev. Lett. **117**, 210505 (2016).
[8] H. Bombín, New J. Phys. **17**, 083002 (2015).
[9] A. Kubica and M. E. Beverland, Phys. Rev. A **91**, 032330 (2015).
[10] S. Bravyi and R. König, Phys. Rev. Lett. **110**, 170503 (2013).
[11] F. Pastawski and B. Yoshida, Phys. Rev. A **91**, 012305 (2015).
[12] A. Kubica, B. Yoshida, and F. Pastawski, New Journal of Physics **17**, 083026 (2015).
[13] T. Jochym-O'Connor, A. Kubica, and T. J. Yoder, Phys. Rev. X **8**, 021047 (2018).
[14] D. MacKay, G. Mitchison, and P. McFadden, IEEE Transactions on Information Theory **50**, 2315 (2004).
[15] A. A. Kovalev and L. P. Pryadko, in *2012 IEEE International Symposium on Information Theory Proceedings* (IEEE, 2012) pp. 348–352.
[16] S. Bravyi and M. B. Hastings, in *Proceedings of the forty-sixth annual ACM symposium on Theory of computing* (ACM, 2014) pp. 273–282.
[17] M. H. Freedman and M. B. Hastings, Quant. Inf. Comput. **14**, 144 (2014).
[18] J.-P. Tillich and G. Zémor, IEEE Transactions on Information Theory **60**, 1193 (2014).
[19] M. B. Hastings, Quantum Information & Computation **17**, 1307 (2017).
[20] B. Eastin and E. Knill, Phys. Rev. Lett. **102**, 110502 (2009).
[21] B. Zeng, A. W. Cross, and I. L. Chuang, IEEE Transactions on Information Theory **57**, 6272 (2011).
[22] P. W. Shor, Proceedings of the 37th Annual Symposium on Foundations of Computer Science , 56 (1996).
[23] E. Knill, R. Laflamme, and W. H. Zurek, arXiv: quant-ph/9610011 (1996).
[24] S. Bravyi and A. Kitaev, Phys. Rev. A **71**, 022316 (2005).
[25] A. Paetznick and B. W. Reichardt, Phys. Rev. Lett. **111**, 090505 (2013).
[26] T. Jochym-O'Connor and R. Laflamme, Phys. Rev. Lett. **112**, 010505 (2014).
[27] J. T. Anderson, G. Duclos-Cianci, and D. Poulin, Phys. Rev. Lett. **113**, 080501 (2014).
[28] H. Bombín, New J. Phys. **18**, 043038 (2015).
[29] S. Bravyi and A. Cross, arXiv:1509.03239 (2015).
[30] T. Jochym-O'Connor and S. D. Bartlett, Phys. Rev. A **93**, 022323 (2016).
[31] C. Jones, P. Brooks, and J. Harrington, Phys. Rev. A **93**, 052332 (2016).
[32] T. J. Yoder, R. Takagi, and I. L. Chuang, Phys. Rev. X **6**, 031039 (2016).
[33] H. Bombín and M. A. Martin-Delgado, Phys. Rev. Lett. **97**, 180501 (2006).
[34] A. R. Calderbank and P. W. Shor, Phys. Rev. A **54**, 1098 (1996).
[35] A. W. Steane, Proc. Roy. Soc. Lond. **452**, 2551 (1996).
[36] D. Gottesman, *Stabilizer Codes and Quantum Error Correction*, Ph.D. thesis, California Institute of Technology (1997).
[37] E. Knill and R. Laflamme, Phys. Rev. A **55**, 900 (1997).
[38] B. J. Brown, W. Son, C. V. Kraus, R. Fazio, and V. Vedral, New J. Phys. **13**, 065010 (2011).
[39] A. Barenco, C. H. Bennett, R. Cleve, D. P. DiVincenzo, N. Margolus, P. Shor, T. Sleator, J. A. Smolin, and H. Weinfurter, Phys. Rev. A **52**, 3457 (1995).
[40] C. Chamberland, T. Jochym-O'Connor, and R. Laflamme, Phys. Rev. Lett. **117**, 010501 (2016).
[41] C. Chamberland, T. Jochym-O'Connor, and R. Laflamme, Phys. Rev. A **95**, 022313 (2017).
[42] C. Chamberland and T. Jochym-OConnor, Quantum Sci. Technol. **2**, 035008 (2017).
[43] A. Kitaev, Annals of Physics **321**, 2 (2006).
[44] S. Bravyi, B. M. Terhal, and B. Leemhuis, New J. Phys. **12**, 083039 (2010).
[45] E. T. Campbell, arXiv:1805.09271 (2018).

Appendix A: Examples of boundary operators

Boundary operator for the 7-qubit Steane code, each row and column have weight 4:

$$\delta_7 = \begin{pmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 1 & 1 & 0 \end{pmatrix}. \tag{A1}$$

Note that for the Steane code, every non-trivial element of the stabilizer group is represented in the rows and columns, this will not hold in general for other codes.

A boundary operator for the padded 15-qubit Reed-

Muller code, composed of 21 qubits:

$$\delta_{15p} = \left(\begin{array}{cc}
011010011001011 & 111000 \\
110000110011110 & 110010 \\
101001010101101 & 101100 \\
000011111111000 & 100100 \\
100110010110011 & 011001 \\
001100111100110 & 010010 \\
010101011010101 & 001001 \\
111111110000000 & 000000 \\
100101101001011 & 000000 \\
001111000011110 & 000010 \\
010110100101101 & 000100 \\
111100001111000 & 000100 \\
011001100110011 & 000001 \\
110011001100110 & 000010 \\
101010101010101 & 000001 \\
\hline
000000000000000 & 000000 \\
000000000000000 & 000000 \\
000000000000000 & 000000 \\
000000000000000 & 000000 \\
000000000000000 & 000000 \\
000000000000000 & 000000
\end{array}\right). \tag{A2}$$

We have visually split the matrix into two sets, the first 15 qubits and 6 ancillary qubits. The first 15 qubits are where the logical information is stored, while the extra 6 qubits represent the padded ancilla qubits that are prepared in the $|+\rangle$ state. Note that none of the Z stabilizers, represented by the columns, have support on the last 6 qubits. It is fairly straightforward to check that $\text{rank}(\delta_{15p}) = 10$. One can recover independent generators for the rows that correspond to the 15-qubit Reed-Muller code X stabilizers on the first 15 qubits, and individual single-qubit X generators on the last 6 qubits. One can also recover the independent 15-qubit Reed-Muller code Z generators by considering the columns as well as a representative of the independent 6 gauge face generators in the last 6 columns, these correspond to fixing the gauge in the Z basis.

The homological product boundary operator $\partial = \delta_7 + \mathbb{1} + \mathbb{1} \otimes \delta_{15p}$ will have sparsity 15, that is every row and column in ∂ will be of weight at most 15. This corresponds to the maximum weight operator one would have to measure for implementing the universal scheme on the homological product of these two codes, an improvement over the universal concatenated model [26] which would require measuring operators of weight 28. More importantly, in using higher distance versions of each of the codes, the concatenated model would require measuring operators whose weight will grow linearly with system size, as opposed to that of the homological construction which will remain constant-sized.