Quantum Low-Density Parity-Check Codes for Modular Architectures

Armands Strikis[1,2,*] and Lucas Berent[3,†]

[1] *Department of Materials, University of Oxford, Oxford OX1 3PH, United Kingdom*
[2] *Quantum Motion, 9 Sterling Way, London N7 9HJ, United Kingdom*
[3] *Chair for Design Automation, Technical University of Munich, Germany*

(Received 9 November 2022; accepted 4 April 2023; published 5 May 2023)

In efforts to scale the size of quantum computers, modularity plays a central role across most quantum computing technologies. In the light of fault tolerance, this necessitates designing quantum error-correcting codes that are compatible with the connectivity arising from the architectural layouts. In this paper, we aim to bridge this gap by giving a novel way to view and construct quantum low-density parity-check (LDPC) codes *tailored* for modular architectures. We demonstrate that if the intra- and intermodular qubit connectivity can be viewed as corresponding to some classical or quantum LDPC codes then their hypergraph product code fully respects the architectural connectivity constraints. Finally, we show that relaxed connectivity constraints that allow *twists* of connections between modules pave a way to construct codes with better parameters.

DOI: 10.1103/PRXQuantum.4.020321

I. INTRODUCTION

In classical computing it has become standard to design architectures that divide the necessary processing power into smaller components instead of only increasing the power of a single system [1,2]. A similar trend can be observed in recent proposals around scaling quantum computation. A multitude of quantum computing platforms have natural limitations, e.g., on how many qubits may be contained within a single ion trap or a superconducting chip, whereas each instance of such a platform is referred to as *a module* [3–7]. Scaling up existing systems is the main hurdle in current research. Therefore, *modular architectures* that consist of many similar modules will likely be necessary [8–13].

To execute large-scale quantum algorithms, fault-tolerant quantum computation (FTQC) is essential [14]. A crucial component of FTQC is the error-correcting code, which describes how to encode quantum information in a redundant way with the goal of lowering the error rates of computation [15,16]. Recent results have shown the existence of quantum low-density parity-check (QLDPC) codes with asymptotically good parameters [17–21]. This is a strong indication that QLDPC codes may play a key role in lowering the qubit overhead necessary for FTQC. It is known that well-performing QLDPC codes require "long-range" qubit connectivity if it is desired to embed the system in some finite-dimensional Euclidean space [22]. In fact, the asymptotic scaling of code parameters is upper bounded by the scaling of long-range qubit connectivity [23,24].

When specifically considering the practical setting of modular architectures of a quantum computer (with a finite number of qubits), we may expect that some degree of long-range interactions that scale with the code size is physically feasible [25–28]. These can be long-range interactions within each module or between the modules themselves. Because of this less-constrained connectivity, the question of whether it is useful and practical to favor QLDPC codes of finite size over the surface code—the current gold standard for many quantum computing platforms [29–36]—is important. To answer this question, a multitude of aspects of FTQC need to be considered, such as the implementation of fault-tolerant logic, decoding, and the code performance. Most of these questions are still open for QLDPC codes. Previous works have mentioned the compatibility of QLDPC codes and modular architectures, but without providing the exact details for the code construction or the partition of the qubits into modules [37]. Alternatively, past works have described in detail how to use a surface code for modular architectures [38], but do not consider general QLDPC codes.

In this paper, we explore aspects around fault tolerance for modular architectures with a focus on code

*armands.strikis@mansfield.ox.ac.uk
†lucas.berent@tum.de

constructions. In a formal and general way, we show how QLDPC codes *tailored* to modular architecture connectivity constraints can be constructed. This gives a correspondence between product constructions of QLDPC codes and modular architectures by viewing the intra- and intermodular connectivities as Tanner graphs or, equivalently, as chain complexes of classical or quantum LDPC codes. First, we give a formal perspective on the recently introduced *looped pipeline architecture* [39]. We prove that it can be slightly extended to produce a three-dimensional (3D) surface code. This immediately shows a valuable contribution of our formalism to practically highly relevant work around modular architectures. We then extend the construction to more general hypergraph product codes. Finally, we show that a broader class of product codes with potentially better code parameters can be constructed for architectures that allow *twisted* modular connectivity. With this work, we take a further step towards closing the gap between physical "low-level" system architecture questions and recent theoretical breakthroughs around asymptotically good quantum codes [17,18,20,21,40]. Furthermore, we want to emphasize the need for investigations around practical applications of general QLDPC codes [41].

The rest of this work is structured as follows. First, modular architectures and a formal viewpoint is given in Sec. II. Notation and fundamental background is presented in Sec. III. The looped pipeline architecture, the 3D surface code construction, and our first main theorem is discussed in Sec. IV. Stepwise generalizations are subsequently given in Sec. V, where the intramodular connectivity is generalized, in Sec. V B, where the intermodular connectivity is generalized, and in Sec. VI, where the allowed connections between the modules are generalized. Finally, we conclude with a short discussion and outlook in Sec. VII.

II. MODULAR ARCHITECTURES

In this paper we consider various qubit connectivity constraints that may arise in a quantum computer based on a modular architecture. Taking the constraints into account, we provide a novel recipe on how to construct quantum LDPC codes that respect a certain modular architecture.

Let us first concretely define what we mean by a *modular architecture*. Consider a quantum computer with a (finite) collection of qubits $\{q_N\}$. In a modular architecture, each qubit is assigned to one and only one module, where the (finite) collection of such modules is given as $\{M_k\}$; see Fig. 1. We assume that the modules are equivalent copies of each other. Therefore, we can partition the collection of qubits into disjoint sets $\{q_i\}_k$ such that $\{q_N\} = \bigcup_k \{q_i\}_k$, and we suppose that each module contains a finite and equal number of qubits n, i.e., $|\{q_i\}_k| = n$ for all k. To simplify the notation, we use the same canonically ordered index set for each module and hence drop the subscript k altogether. With this in mind, we can define the intramodular qubit connectivity in the usual sense as follows.

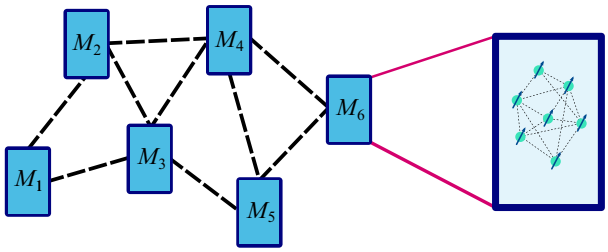

FIG. 1. Here, quantum computing architectures where modules $\{M_k\}$ have a defined sparse intermodular connectivity are considered. Each module contains a finite and equal number of qubits with the same connectivity constraints.

Definition 1: A qubit $q_i \in M_k$ is *connected* to a qubit $q_j \in M_k$ if the architecture allows us to directly implement two-qubit entangling operations between these qubits for all k.

Generally, we consider entangling gates such as controlled-not that are required for most syndrome circuits. However, entangling operations using measurements, e.g., in using photonic links, or otherwise are just as valid. We only require that these gates allow the construction of a syndrome extraction circuit required for quantum error correction.

In this work, we investigate cases where the qubit intramodular interactions are given by a connectivity graph that can be viewed as a *Tanner graph* of some classical or quantum code.

A (quantum) Tanner graph is a graph that has edges between nodes representing parity checks and data (qu)bits if and only if the (qu)bit is in the support of the parity check. See Fig. 2 for a Tanner graph of a seven-qubit Steane code and Sec. III for a more technical introduction to Tanner graphs. As an example, nearest-neighbor connectivity for a 1D chain of qubits corresponds to a Tanner graph of a classical repetition code.

In a modular architecture, each module may be connected to some number of other modules in a specific way. We define the intermodular connectivity as follows.

Definition 2: A module M_k is *connected* to a module M_j if the architecture allows us to directly implement two-qubit entangling operations between a qubit $q_i \in M_k$ and its *respective* qubit $q_i \in M_j$ for all i.

In Sec. VI A, this requirement will be slightly generalized to allow for *twists* of the connections between modules to construct better quantum codes (see Definition 3 below). Similarly to the intramodular connectivity case, we choose the graph defining the intermodular connectivity to correspond to a Tanner graph of a potentially different quantum or classical code. Finally, we say that the

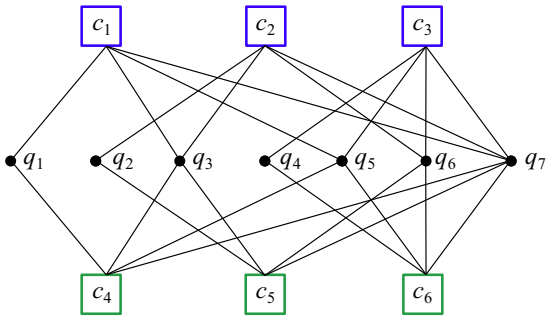

FIG. 2. Tripartite quantum Tanner graph of a seven-qubit Steane code. Set $\{q_i\}$ denotes the data qubits, while $\{c_j\}$ denotes the X parity checks (blue) and Z parity checks (green). Edges are drawn between them if and only if the data qubit is in the support of the check.

code *respects* the connectivity constraints if we can associate a physical qubit in our quantum system to every parity check and data qubit of the code such that the parity-check qubits are connected to the data qubits in their support.

In the following, we describe a way to create new codes that respect the overall architectural connectivity constraints as defined above. We only require that the intra- and intermodular connectivities are formulated as Tanner graphs of some codes. To keep this formulation as general as possible, we employ the language of chain complexes.

III. PRELIMINARIES

In this section, we discuss quantum codes and how to view them in terms of a ($\mathbb{F}_2$) homological perspective [31,40,42,43].

A. Classical and quantum codes

Since quantum LDPC codes have an interesting correspondence to classical codes, let us briefly discuss their classical analogue: binary linear codes.

A classical binary linear $[n,k]$ code is a subspace $\mathcal{C} \subseteq \mathbb{F}_2^n$. The set of codewords is called the *codespace* and corresponds to the k-dimensional subspace $\ker H$ of a binary matrix H called the *parity-check matrix*:

$$\mathcal{C} = \{x \in \mathbb{F}_2^n \mid Hx = 0\}.$$

It is useful to describe code $\mathcal{C}$ with its *Tanner graph*. This is a bipartite graph $\mathcal{T}(\mathcal{C})$ whose adjacency matrix is H.

Since stabilizer codes [44] play a central role in quantum error correction (QEC), let us recall some fundamental definitions. We consider an n-qubit Hilbert space $(\mathbb{C}^2)^{\otimes n} = \mathbb{C}^{2^n}$. Let $\mathcal{P}_n$ denote the (non-Abelian) group of n-qubit Pauli operators defined as

$$\mathcal{P}_n = \langle i, X_j, Z_j \mid j \in [n] \rangle = \left\{ \phi \bigotimes_{j=0}^{n} P_j \right\},$$

where $\phi \in \{\pm 1, \pm i\}$ and P_j is a single-qubit Pauli operator $P_j \in \{I, X, Y, Z\}$. The weight $\mathrm{wt}(P)$ of a Pauli operator P is the number of nonidentity components in the tensor product representation of P. A *stabilizer group* $\mathcal{S}$ is an Abelian subgroup of $\mathcal{P}_n$ such that $-I \notin \mathcal{S}$. Elements of $\mathcal{S}$ are called *stabilizers* and the group is generated by m independent *stabilizer generators* $\mathcal{S} = \langle S_1, \ldots, S_m \rangle$.

The main idea of the *stabilizer codes* is to use a common $+1$ eigenspace of all elements of a stabilizer group $\mathcal{S} \subset \mathcal{P}_n$ as the code space of a code $\mathcal{C}$. Therefore, an $[\![n,k,d]\!]$-quantum stabilizer code $\mathcal{C}$ is a 2^k-dimensional subspace of $(\mathbb{C}^2)^{\otimes n}$. Parameter d denotes the *minimal distance* of $\mathcal{C}$, given by the minimal weight of a Pauli operator that commutes with all stabilizers S_i but is not in the stabilizer group S.

Each n-qubit Pauli operator can be written as a binary vector. More formally, the quotient group $\mathcal{P}_n/\{\pm I^{\otimes n}, \pm iI^{\otimes n}\}$ is isomorphic (up to phases) to $\mathbb{F}_2^{2n}$ by the isomorphism that sends an n-qubit Pauli operator corresponding to a tensor product of X- and Z-Pauli operators to a binary vector representation $(x \mid z) \in \mathbb{F}_2^{2n}$. Thus, $P, Q \in \mathcal{P}_n$ commute if and only if, for their binary representations $P \cong (x \mid z), Q \cong (x' \mid z')$, it holds that

$$\langle x, z' \rangle + \langle z, x' \rangle = 0. \tag{1}$$

This representation can also be naturally applied to the m stabilizer generators $S_1, \ldots, S_m$ of a code $\mathcal{C}$, which yields an $m \times 2n$ matrix $H = (H_X \mid H_Z)$. Each row of H corresponds to the binary representation of a stabilizer generator S_i. As for classical codes, matrix H is the *parity-check matrix* of $\mathcal{C}$. By Eq. (1), $\ker H$ is exactly the set of vectors $(z \mid x) \in \mathbb{F}_2^{2n}$ such that their reordered form $(x \mid z)$ is the binary representation of a Pauli operator that commutes with all stabilizer generators $S_1, \ldots, S_m$.

An important subclass of stabilizer codes are Calderbank-Shor-Steane (CSS) codes, which is considered in this work if not stated otherwise. These are stabilizer codes where all nonidentity components of stabilizer generators are either all X or all Z. Hence, the commutativity relation [Eq. (1)] can be written as

$$H_X H_Z^T = 0, \tag{2}$$

or, equivalently, $C_Z^\perp \subseteq C_X$. Since the rows of a parity-check matrix correspond to the *checks* of the code, elements of H_X and H_Z are called X and Z checks, respectively. A CSS code is called a LDPC code if all checks have constant weight and each qubit is involved in a constant number of checks, i.e., if the parity-check matrix (matrices) is sparse.

Let us now introduce an alternative perspective on codes that was essential in recent results around asymptotically good quantum and locally testable classical codes [17–21], and has become standard.

B. Chain complexes

A *chain complex* of vector spaces $\{C_i\}$ together with linear maps

$$\partial_i : C_i \to C_{i-1}$$

with the condition that squared boundary maps vanish, i.e.,

$$\partial_i \partial_{i+1} = 0. \tag{3}$$

Equation (3) is equivalent to requiring that $\mathrm{Im}\,\partial_{i+1} \subseteq \ker \partial_i$. Elements in C_i are called *i-chains* and

$$Z_i(C) = \ker \partial_i \subset C_i, \tag{4}$$

$$B_i(C) = \mathrm{Im}\,\partial_{i+1} \subset C_i, \tag{5}$$

$$H_i(C) = Z_i(C)/B_i(C) \tag{6}$$

are the *i-cycles*, *i-boundaries*, and the *i*th *homology* of complex C, respectively. For instance, when considering chain complexes arising from simplicial complexes, 2-chains correspond to formal linear combinations of faces, 1-chains to formal sums of edges, and 0-chains to formal sums of vertices. Intuitively, 1-cycles are loops that start and end in the same vertex and boundaries are those cycles that are a boundary of a set of faces.

A classical binary linear code $\mathcal{C}$ can be viewed as a two-term chain complex:

$$C = C_1 \xrightarrow{\partial_1} C_0 \tag{7}$$

with $C_1 = \mathbb{F}_2^n$ and $\partial_i = H$ the parity-check matrix. Then code $\mathcal{C}$ is the space of 1-cycles, i.e.,

$$\mathcal{C} = Z_1(C) = \ker \partial_1,$$

and the space of 0-chains is the space of checks acting on $\mathcal{C}$. Note that $H_0(C) = 0$ if the checks are linearly independent. For classical codes, this representation does not yield any new insights and is hence rarely used. However, a quantum CSS code necessitates commutation relations between the bit-flip and phase-flip parity-check matrices H_X and H_Z [Eq. (2)]. Thus, there is a bijection between CSS codes and chain complexes. A CSS code corresponds to a three-term chain complex:

$$C = C_{i+1} \xrightarrow{\partial_{i+1}} C_i \xrightarrow{\partial_i} C_{i-1} \tag{8}$$

with $\partial_{i+1} = H_Z^T$ and $\partial_i = H_X$. Thus, qubits are associated with 1-chains and X and Z checks with 0- and 2-chains, respectively. A prototypical example is a toric code, where C_2, C_1, and C_0 are vector spaces of *faces*, *edges*, and *vertices*, obtained from a square cellulation of a torus. Conversely, given an arbitrary chain complex

$$\cdots \to C_{i+1} \xrightarrow{\partial_{i+1}} C_i \xrightarrow{\partial_i} C_{i-1} \to \cdots, \tag{9}$$

we can pick a dimension i to associate the space of qubits with and view the corresponding three-term chain complex as a CSS code. The code parameters are $n = \dim C_i$, $k = \dim H_i$ and the minimum weight d of a nontrivial representative of H_i.

To a three-term chain complex we can associate a quantum Tanner graph by considering a simple mapping. A quantum Tanner graph G is a tripartite graph with a vertex set $V = P_Z \sqcup Q \sqcup P_X$, where vertices Q are associated with qubits and vertices $P_{X(Z)}$ are associated with $X(Z)$ parity checks. Then, for an arbitrary three-term chain complex C, we define a one-to-one mapping where the basis elements of the vector space C_{i+1} are mapped to vertices in P_Z, the basis elements of C_i to Q, and the basis elements of C_{i-1} to P_X. Finally, the edges between vertex partitions P_Z and Q are given by ∂_{i+1} such that an edge exists between vertices $p \in P_Z$ and $q \in Q$ if and only if $\overline{q} \in C_i$, which is the corresponding basis element of q, is in the span of $\partial_{i+1}\overline{p}$, where $\overline{p} \in C_{i+1}$ and corresponds to vertex p. Edges between P_X and Q are defined in an analogous manner using ∂_i. Note that there are no edges between any two vertices within the same partition or between partitions P_X and P_Z. Graphically, if the chain complex is drawn as an object with faces, edges, and vertices, then each individual face and edge is replaced by a vertex. The partition of vertices is naturally induced from this mapping and edges exist only between those pairs of vertices that correspond to adjacent faces and edges (or vertices and edges) in the original chain complex.

Conceptually, the same mapping to a graph can be applied to an arbitrary length chain complex. For example, a two-term chain complex would be mapped to a classical Tanner graph. Because of this bijection we refer to chain complexes, their boundaries, or their corresponding Tanner graphs (sometimes called connectivity graphs) interchangeably for the rest of the paper.

IV. LOOPED PIPELINE ARCHITECTURE

To form a better intuition about our construction of codes for modular architectures, we first recap basic ideas of a recent result around QEC on a looped pipeline architecture [39]. This constitutes the basis from which we build more involved and better quantum codes as we soften the constraints of the intra- and intermodular connectivities and generalize the construction.

The work by Cai *et al.* [39] considered a surface code layout (qubits are put on edges, Z checks on faces, and X checks on vertices) as depicted in Fig. 3(a) of a quantum chip where every data qubit and every ancilla qubit are replaced by a rectangular loop of a fixed number of

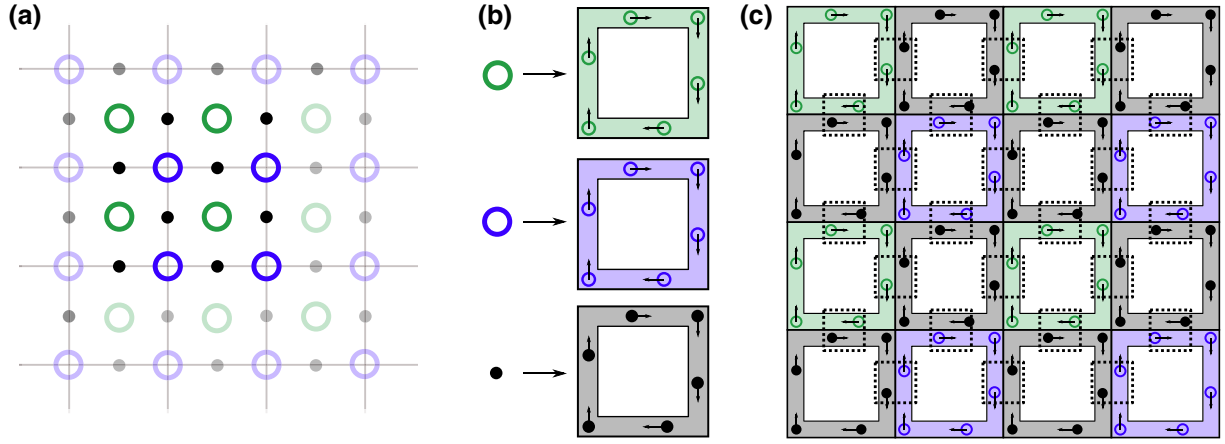

FIG. 3. A stack of 2D surface codes can be generated by replacing every data qubit (black), Z parity-check qubit (green), and X parity-check qubit (blue) of the regular surface code (a) by a loop of corresponding qubits (b). The qubits communicate (entangle) with each other once they enter the dashed regions depicted in (c).

qubits as sketched in Fig. 3(b). All types of qubits are moved along the loop in the clockwise direction at the same frequency. Once the qubits approach another qubit from a different loop, they interact to become entangled in a way that corresponds to the syndrome extraction circuit as illustrated in Fig. 3(c). After the ancilla qubits have gone around the full loop, they are measured to read out the syndrome. The authors showed that this forms a stack of 2D surface codes, where the stack size is given by the number of qubits within each loop.

In their work, the authors Cai *et al.* [39], and independently M. Fogarty [45], identified that the stack of 2D surface codes may be used to generate a 3D surface code, but did not provide an explicit construction. In the rest of the section we show how the looped pipeline architecture can be extended to implement a single 3D surface code (explained below) by assuming an additional connectivity within each qubit loop. Finally, we formalize and generalize this construction to the setting of modular architectures. This will allow us to construct quantum LDPC codes for more general connectivity constraints.

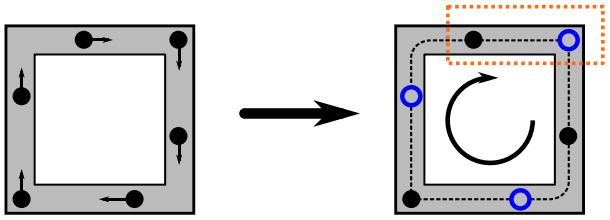

FIG. 4. We replace all loops with a loop that has both ancilla (blue) and data (black) qubits. Additionally, we allow for nearby qubits in the loop to communicate. For example, the nearby qubits entangle whenever both of them enter the orange dashed box.

A. 3D surface code

Consider a single data qubit loop from the looped pipeline architecture described above. Each qubit in this loop is part of a separate 2D surface code. If we extend the connectivity between the nearby qubits within each loop then we obtain a stack of surface codes linked together into a single block. This construction does not immediately produce a 3D surface code. For example, it links ancilla qubits to other ancilla qubits within the same loop.

Instead, to produce a valid 3D surface code, we additionally need to reidentify the qubits within each loop. In this regard, we form three different types of loops—face, edge, and vertex loops—where the naming will parallel the chain complexes. They are laid out in a similar pattern as previously with face loops replacing the Z ancilla loops, edge loops replacing the data qubit loops, and vertex loops replacing the X ancilla loops. Each of these loops may contain both data and ancilla qubits of the 3D surface code; hence, they must also contain measurement devices as described in Ref. [39] to extract the syndrome (see Fig. 4). We assume that the total number of qubits per loop is even. Then, depending on the type of loop and the position within the loop, each qubit can be given an assignment. These assignments can be found in Table I. The even or odd parity of the qubit corresponds to its index i within the loop, where the indexing is such that any qubit q_i of any loop gets to interact with qubits q_i of the neighboring loops. Note that we can exclude the even qubits in the face loop as they have no assignment.

By viewing the modular architecture in terms of chain complexes that describe codes, we can prove that such an assignment indeed produces a 3D surface code. In order to do so, we use tensor products of chain complexes.

Remark: The presented construction of a 3D surface code by shuttling the qubits around in loops will not have a

TABLE I. Qubit assignments of different loops to generate a 3D surface code. Even and odd assignments of qubits indicate their position in the qubit chain within the loop. Numbers in parentheses indicate the weight of the stabilizer.

Loop label	Odd qubits	Even qubits
Vertex	X stabilizer (6)	Data
Edge	Data	Z stabilizer (4)
Face	Z stabilizer (4)	⋯

threshold in general. Since the entangling and measurement operations are done on a one-by-one basis, the time it takes to extract the syndrome is proportional to the number of qubits within each loop. A slightly altered scheme was proposed by Cai *et al.* [39], where the qubits in adjacent loops circulate in opposite directions and all qubits on the same side of the loop are being entangled at the same time with their respective qubits in other loops. If this was supplied with a number of measurement devices that scale in proportion with the number of qubits in the loop, the whole syndrome extraction would require $O(1)$ operations (e.g., one operation constitutes moving qubits to a new side of the loop). However, from a physical perspective, one could argue that, in general, as the number of qubits per loop increases, so does the size of the loop that is needed. Therefore, the physical time it takes for a qubit to be shuttled at a finite speed to a new side of the loop (to perform a single operation) scales with the number of qubits per loop and, hence, the syndrome extraction time scales with the total number of qubits. In turn, errors accumulate faster than they can be corrected and the threshold for the code would not exist. This might be a general consequence for any nonconcatenated quantum code for which the shuttling is used to do long-range entangling gates locally (the same conclusion was reached by numerical analysis for a conceptually similar setup in Ref. [46]). Some ideas, like those presented in a recent work on hierarchical memories [47], may be used to overcome this challenge. Note that the aforementioned drawback only applies to this specific scheme based on shuttling. The main ideas in the remainder of this manuscript do not share the same consequences.

B. Tensor product of chain complexes

Quantum codes can be constructed from products of chain complexes that describe other codes [48,49]. Let us discuss the construction in the following. The *double complex* $C \boxtimes D$ is defined as

$$(C \boxtimes D)_{p,q} = C_p \otimes D_q \tag{10}$$

with *vertical* boundary maps $\partial_i^v = \partial_i^C \otimes \mathrm{id}^D$ and *horizontal* boundary maps $\partial_i^h = \mathrm{id}^C \otimes \partial_i^D$ such that $\partial_i^v \partial_{i+1}^v = 0$, $\partial_i^h \partial_{i+1}^h = 0$, and $\partial_i^v \partial_j^h = \partial_j^h \partial_i^v$. An example double complex where C, D are 2-term complexes is visualized in

$$\begin{array}{ccc} C_1 \otimes D_1 & \xrightarrow{id^C \otimes \partial_1^D} & C_1 \otimes D_0 \\ \downarrow{\scriptstyle \partial_1^C \otimes id^D} & & \downarrow{\scriptstyle \partial_1^C \otimes id^D} \\ C_0 \otimes D_1 & \xrightarrow{id^C \otimes \partial_1^D} & C_0 \otimes D_0 \end{array}$$

FIG. 5. A commuting diagram representing a double chain complex of two 2-term chain complexes C, D.

Fig. 5. The *total complex* arises when we collect vector spaces of equal dimensions, i.e., "summing over the diagonals" in the double complex as

$$\mathrm{Tot}(C \boxtimes D)_n = \bigoplus_{p+q=n} C_p \otimes D_q = E_n, \tag{11}$$

where the boundary maps are $\partial^E = \partial^v \oplus \partial^h$. Then, the *tensor product complex* $C \otimes D$ is defined as $\mathrm{Tot}(C \boxtimes D)$. For the example given in Fig. 5, the tensor product complex is

$$C_1 \otimes D_1 \xrightarrow{\partial_2} C_0 \otimes D_1 \oplus C_1 \otimes D_0 \xrightarrow{\partial_1} C_0 \otimes D_0,$$

where

$$\partial_2 = \begin{pmatrix} \partial_1^C \otimes \mathrm{id}^D \\ \mathrm{id}^C \otimes \partial_1^D \end{pmatrix},$$
$$\partial_1 = (\mathrm{id}^C \otimes \partial_1^D \mid \partial_1^C \otimes \mathrm{id}^D).$$

As the homology of a chain complex is related to the parameters of the corresponding code, the Künneth formula is central. It gives a method to compute the homology of a double complex, from the homology of the vertical and horizontal complexes:

$$H_n(C \otimes D) \cong \bigoplus_{p+q=n} H_p(C) \otimes H_q(D). \tag{12}$$

C. 3D surface code in the chain complex formalism

While the previous construction of a 3D surface code may seem arbitrary at first, we can naturally describe it in the language of chain complexes. This description allows us to further generalize the construction of codes for modular architectures and gives a strong intuition for which codes may or may not be constructed given the connectivity constraints.

First, consider a single loop of qubits with nearest-neighbor connectivity between them. It is natural to view the loop as a classical repetition code where half of the qubits are assigned to be data qubits and the other half ancilla qubits, as depicted in Fig. 4. As mentioned in Sec. III, we may use a two-term chain complex $C = C_1 \xrightarrow{H_X} C_0$ to describe the repetition code, where the data and ancilla qubits are elements (chains) of C_1 and C_0,

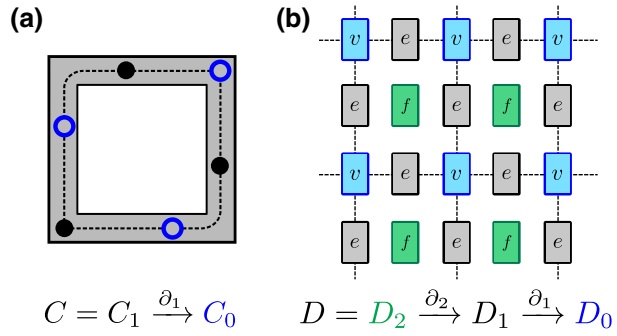

FIG. 6. A single loop corresponds to a two-term chain complex and a layout of loops to a three-term chain complex representing a repetition code (a) and a surface code (b), respectively.

respectively, and H_X is the parity-check matrix of the code; see Fig. 6(a).

Moreover, the 2D layout of the loops can be considered as a two-term chain complex $D = D_2 \xrightarrow{H_Z^T} D_1 \xrightarrow{H_X} D_0$ representing a 2D surface code. In this language, each loop is labeled as an element of either D_2, D_1, or D_0, which represent Z checks (faces), data qubits (edges), or X checks (vertices), respectively; see Fig. 6(b).

Using such a layout of loops, we have effectively created a new code $\mathcal{E}$. As outlined in Eq. (11), $\mathcal{E}$ is represented by a chain complex

$$E = C \otimes D, \tag{13}$$

where C corresponds to the code within the loop and D corresponds to the code describing the layout of the loops. In more detail, we associate two vector spaces C_i and D_j with each qubit in our system. In this example, C_i describes whether the qubit in the repetition code is an ancilla ($i = 0$) or data ($i = 1$) qubit, while D_j describes whether the qubit belongs to the face ($j = 2$), edge ($j = 1$), or vertex ($j = 0$) loop. See Fig. 7 for a schematic explanation of qubits in the edge loop. Furthermore, we assign each qubit to a vector space E_k, with $k = i + j$, which form the sequence of vector spaces for a new four-term chain complex

$$E = E_3 \xrightarrow{\partial_3} E_2 \xrightarrow{\partial_2} E_1 \xrightarrow{\partial_1} E_0, \tag{14}$$

where

$$\begin{aligned} E_3 &= C_1 \otimes D_2, \\ E_2 &= C_0 \otimes D_2 \oplus C_1 \otimes D_1, \\ E_1 &= C_0 \otimes D_1 \oplus C_1 \otimes D_0, \\ E_0 &= C_0 \otimes D_0. \end{aligned}$$

This chain complex E describes a 3D surface code, as per the tensor product of a repetition code and a surface code [49,50]. Here, for example, we can identify the data qubits with chains in E_1, and, hence, Z (X) stabilizers with chains in E_2 (E_0). Then, parity-check matrices are given as boundary operators $\partial_2 = H_Z^T$ and $\partial_1 = H_X$. Note that in this construction one of the boundaries of the surface is periodic and, also, that we are free to identify the data qubits with chains either in E_2 or E_1. This freedom corresponds to the choice of having the logical X (Z) operators to be planar (string)-like on the 3D surface or the other way around.

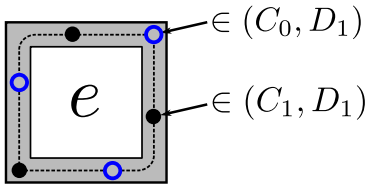

FIG. 7. Our observations allow us to readily identify which vector spaces each qubit belongs to. The ancilla qubits of the repetition code are elements (chains) of C_0, while the data qubits are chains of C_1. The loop itself has an assignment as an edge loop; therefore, every qubit in it belongs to D_1.

As an example, consider an $L \times L$ surface code as the layout of the loops. It has parameters $[\![2(L^2 - L) + 1, 1, L]\!]$. The respective chain complex D has homology $H_1(D) \cong \mathbb{Z}_2$ and $H_0(D)$ is trivial. Similarly, consider a length-L classical repetition code inside the loop with parameters $[L, 1, L]$. The respective chain complex $\mathcal{C}$ has homology $H_1(C) \cong \mathbb{Z}_2$ and $H_0(C) \cong \mathbb{Z}_2$ since one of the checks is linearly dependent. Then, for the chain complex of a 3D surface code $E = C \otimes D$, by identifying the elements of E_1 as data qubits, we find that $n = \dim E_1 = \dim(C_1 \otimes D_0 \oplus C_0 \otimes D_1) = L(L^2 - L) + L(2(L^2 - L) + 1)$, where D_0 and C_0 are vector spaces associated with X parity checks. The number of encoded qubits is given by the dimension of the first homology $H_1(E)$. We can compute it using the Künneth formula [Eq. (12)],

$$\begin{aligned} k &= \dim H_1(E) \\ &= \dim(H_0(C) \otimes H_1(D) \oplus H_1(C) \otimes H_0(D)) \\ &= 1. \end{aligned}$$

The distance of the code is still L; hence, the resulting 3D surface code has parameters $[\![3L(L^2 - L) + L, 1, L]\!]$. As an example, for $L = 20$, the parameters are $[\![22\,820, 1, 20]\!]$.

Note that in this construction qubits that were previously data qubits may be reassigned to parity-check qubits and vice versa. More importantly, the intra- and intermodular connectivity requirements of $\mathcal{E}$ correspond to the connectivity requirements given by codes $\mathcal{C}$ and $\mathcal{D}$. We can prove this for general tensor products of chain complexes.

Theorem 1. *Let C and D be a two- or three-term chain complexes representing classical or quantum codes $\mathcal{C}, \mathcal{D}$,*

respectively. Let their boundaries ∂^C and ∂^D define the intra- and intermodular connectivities, respectively. Then a quantum code $\mathcal{E}$ corresponding to the chain complex $E = C \otimes D$ respects the connectivity constraints of the architecture.

Proof. The chain complex E (corresponding to $\mathcal{E}$) is at least a three-term chain complex given that both C and D are at least two-term chain complexes. Therefore, we can identify chains of E_i with data qubits, E_{i+1} with Z parity checks, and E_{i-1} with X parity checks. The required qubit connectivity of $\mathcal{E}$ is defined by its parity-check matrices, which are given by the boundary operators

$$\partial^E = \partial^h \oplus \partial^v = \mathrm{id}^C \otimes \partial^D \oplus \partial^C \otimes \mathrm{id}^D. \tag{15}$$

Note that i-chains of C label the qubit c within each module and that i-chains of D label each module d. We can ensure that $\mathcal{E}$ respects the connectivity constraints of the architecture if both terms of Eq. (15) match the given qubit connectivity. We do so by looking at both of the terms separately.

The basis elements are pairs of chains $(c, d) \in C \times D$ on which the first term acts as $\partial^h : (c, d) \to (\mathrm{id}^C(c), \partial^D(d))$ for all c, d. By linear extension, this defines a map on $C \otimes D$. It is equally stated that *each* qubit c in a module d is connected to its *respective* qubits c in adjacent modules given by ∂^D for all d. This matches Definition 2 for the intermodular connectivity. Similarly, the second term in Eq. (15) defines a map that acts on basis elements as $\partial^v : (c, d) \to (\partial^C(c), \mathrm{id}^D(d))$ for all c, d. This is equally stated as in *each* module d a qubit c is connected to qubits $\partial^C(c)$ for all c. This matches Definition 1 for intramodular connectivity. Therefore, the required qubit connectivity of $\mathcal{E}$ is given by some additive combination of terms that define the intra- and intermodular connectivities, respectively. ■

Furthermore, it is clear that Theorem 1 still applies whenever boundaries ∂^C and ∂^D define any subgraphs of intra- and intermodular connectivities, respectively. If the subgraphs are proper then some connectivity that is allowed by the architecture is not required. Note that our qubit assignment in the chain complex formalism completely matches the assignments given in Table I if we identify the data qubits with the vector space E_1 and ignore qubits in E_3. Since the qubit assignments and the qubit interactions match between both perspectives, the stabilizers of the code match as well. This proves that our previous construction in Sec. IV A produced a 3D surface code.

The correspondence between modular architectures and quantum codes obtained from product constructions is very natural and gives an intuitive way of designing codes that obey architectural connectivity constraints. In the next sections we propose generalizations of this idea in a step-by-step fashion. First, we generalize the intramodular connectivity by considering a less local code within each module. Then we similarly generalize the intermodular connectivity. Finally, we elaborate on more general product code constructions by allowing *twists* between intermodular connections.

V. HYPERGRAPH PRODUCT CODES

In the previous section it was shown that the proposed formalization of the looped pipeline architecture enables us to obtain a 3D surface code that can be viewed in a rigorous way as the tensor product of two chain complexes $C \otimes D$, where C corresponds to the loop structure and D to the (gridlike) layout.

In this section, we further elaborate on the tensor product code construction and extend it to the setting of more general intramodular and intermodular connectivities.

A. Generalized intramodular connectivity

The first generalization of the proposed formalization of the looped pipeline architecture is to replace the loops of qubits with modules admitting a more general intramodular connectivity. Similarly to viewing the loops of qubits as repetition codes, we view this connectivity as a chain complex corresponding to some code $\mathcal{C}$, for instance, a simple classical linear block LDPC code. Note that in general this implies that a higher degree of intramodular connectivity is needed.

Formally, we consider a tensor product $E = C \otimes D$ of a chain complex C corresponding to some classical or quantum code and a three-term chain complex D corresponding to a surface code, describing the overall layout of modules. This yields a four- or five-term chain complex E that can be viewed as a surface code layout of modules, where each module is replaced by an arbitrary code given by the chain complex C. In general, this does not yield a nice 3D geometry as in the more simple 3D surface code case.

As an explicit example, we consider intramodular connectivity that corresponds to a classical linear code. We obtain a code by generating a random sparse parity-check matrix with dimensions 51×60. Through exhaustive search over all codewords we find the code parameters $[60, 9, 20]$. Its maximum row or column weight is 8, but, on average, each check has a support of approximately 5 bits. See Ref. [51] for the full parity-check matrix. The respective chain complex C has homology dimension $\dim(H_1(C)) = 9$ since it encodes 9 bits and $H_0(C)$ is trivial as all parity checks are linearly independent. The homological properties of the chain complex D associated with the surface code describing the intermodular layout are given in the previous example. Note that $H_2(D)$ is trivial and, for this example, we choose a surface code with length $L = 20$. Then, for the resulting chain complex $E = C \otimes D$, by identifying the elements of E_2 as data qubits, we find that

$n = \dim E_2 = \dim(C_1 \otimes D_1 \oplus C_0 \otimes D_2) = 65\,040$, where D_2 and C_0 are vector spaces associated with Z and X parity checks, respectively. The number of encoded qubits is given by the dimension of the second homology $H_2(E)$. We can compute it using the Künneth formula [Eq. (12)],

$$\begin{aligned} k &= \dim H_2(E) \\ &= \dim(H_1(C) \otimes H_1(D) \oplus H_0(C) \otimes H_2(D)) \\ &= 9. \end{aligned}$$

Generally, finding the distance of the code is not trivial, and thus, usually, Monte Carlo simulations or similar approaches are employed. For general hypergraph products of arbitrary length chain complexes, Zeng and Pryadko [49] proposed methods to compute upper and lower bounds on the distance of the hypergraph product complex from the distances of the individual complexes in the product. Moreover, for the special case where one of the chain complexes in the hypergraph product is a one-term complex, given by a binary check matrix, Zeng and Pryadko showed that their result allows one to compute the distance exactly. Recall that the homological distance d_i is the minimum Hamming weight of a nontrivial representative in the ith homology group.

Theorem 2 (Ref. [49]). *Let A be an m chain complex with distances d_i for $0 \leq i \leq m$ and let B be a two-term chain complex. Then,*

$$d_i(A \otimes B) = \min(d_{i-1}(A)d_1(B), d_i(A)d_0(B)).$$

Applying Theorem 2 to the previous example, we find that the Z distance of the code is

$$d_2(E) = \min(d_1(D)d_1(C), d_2(D)d_0(C)) = 400,$$

where, by convention, $d_i(A) = \infty$ if $H_i(A)$ is trivial. Furthermore, we find the X distance of the code using cohomology:

$$d^2(E) = \min(d^1(D)d^1(C), d^2(D)d^0(C)) = 20.$$

Therefore, code $E = C \otimes D$ has parameters $[\![65\,040, 9, 20]\!]$. Given that this code is constructed as a tensor product of a quantum and a classical code, and moreover, that the Z distance is $d^2 = 400$, we choose to compare it to the traditional 3D surface code with distance 20 as it matches both the X and Z distance parameters. In comparison to the 3D surface code (with full parameters given in Sec. IV C) one can see that our example code encodes 9 times more qubits at the cost of increasing the overall number of physical qubits by a factor of about 3. We would expect such favorable trade-offs for architectures with higher qubit connectivity.

This idea can be generalized to more connected modules, by considering a connectivity inside each module that corresponds to a quantum code, i.e., a three-term chain complex. Note that this may imply an even higher degree of intramodular connectivity and a higher number of qubits associated with each module. The latter implies that there are more connections between modules as per Definition 2. Because of the generality of the chain complex formalism, this construction is equivalent to the previous case, with the only difference that the resulting product is a five-term chain complex.

B. Beyond planar surface layouts

In a similar fashion to the discussion on higher intramodular connectivity, we can generalize the layout of the modules. This can be done by considering layouts of modules corresponding to a general code given by a chain complex D. We examine three-term chain complexes describing quantum codes, but the same reasoning is applicable to classical codes. The 2D grid layout (corresponding to a surface code) has the advantage of planarity, which implies a sparse nearest-neighbor connectivity. However, the same planarity clearly limits the code parameters of the derived product codes. Additionally, some architectures (e.g., based on photonic links between modules) might not even be subject to nearest-neighbor communication constraints and hence would have no benefit from a planar layout of modules. In such cases, as long as it holds that modules communicate with a few other modules each, it is not crucial that they are located close to each other physically and we can use less geometrically local codes to describe the layout of the modules. We consider a modular layout given by a chain complex

$$D = D_F \xrightarrow{\partial_2} D_E \xrightarrow{\partial_1} D_V.$$

For illustrative purpose, we call the vector spaces of the chain complex the spaces of *faces*, *edges*, and *vertices*. Here D can be made to correspond to the given intermodular connectivity by associating modules with each i-chain of D. That is, the spaces D_F, D_E, D_V are associated with a set of modules $\{m_1, \ldots, m_{|V|+|E|+|F|}\}$ and the modules are connected corresponding to the boundary maps of D. In other words, the differentials ∂_2, ∂_1 are the incidence matrices associating faces and edges, and edges and vertices, respectively. If C denotes the chain complex describing the intramodular connectivity then, by Theorem 1, the tensor product complex $C \otimes D$ represents a quantum code that respects the connectivity of the overall architecture. This idea is illustrated in Fig. 8.

We are now able to formally describe codes arising from modular architectures using the following recipe. We take an arbitrary CSS code representing the intramodular connectivity (depending on the desired degree of connectivity,

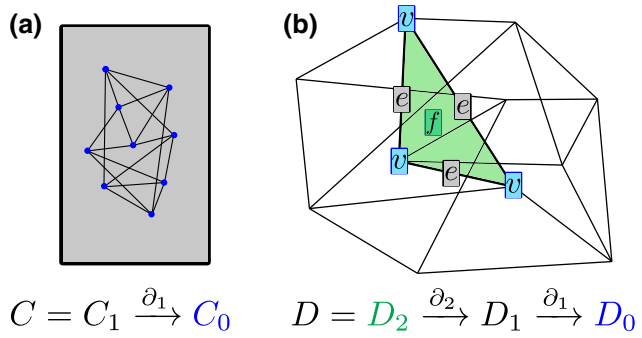

FIG. 8. (a) We represent the intramodular qubit connectivity with a chain complex C, where the qubits sit on all i-chains. (b) Similarly, we represent the intermodular connectivity with a chain complex D, where modules are placed on every i-chain. Some elements of the chain complex are highlighted. Code $E = C \otimes D$ fully respects the connectivity constraints of the architecture (Theorem 1).

number of qubits, and code parameters) and another CSS code that represents the intermodular connectivity. Then, using the tensor product chain complex, we construct a new code satisfying the architectural connectivity constraints. This can already give moderately good codes for a specific modular architecture, depending on the chosen seed codes and the allowed degree of connectivity between modules.

C. From codes to modular architectures

Let us briefly note a slightly different view on the construction presented above. An architecture might allow an "all-to-all" connectivity between modules, or a connectivity that is not constrained other than requiring that each module may only be connected to a constant number of other modules. Then, we can ask: given these "weak" constraints, how should we arrange the modules in order to generate a good code tailored for the architecture? This can be done by choosing a code defined by a chain complex C and mapping it to the respective connectivity graph as introduced in Sec. III B. This graph in turn, defines the overall modular layout as depicted in Fig. 9.

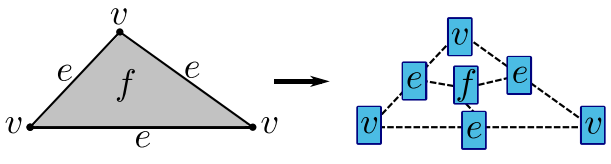

FIG. 9. We can take any chain complex (on the left) to describe the intermodular connectivity of our architecture by replacing every element of the chain complex by a module (on the right).

VI. BALANCED PRODUCT CONSTRUCTIONS

In the previous section we showed how modular architectures can be viewed as tensor products of chain complexes (that describe codes). This perspective corresponds to having intramodular connectivity—the layout of qubits within each module—and intermodular connectivity—the layout of modules themselves—being determined by such codes. In order to further generalize the previous constructions, we consider modular architectures where the intermodular connections are not constrained to be between the respective qubits only. Instead, these connections can be tweaked (*twisted*) in some way that will be dictated by the product construction. Hence, we redefine the intermodular connectivity as follows.

Definition 3: A module M_k is *connected* to a module M_j if the architecture allows us to directly implement two-qubit entangling operations between a qubit $q_i \in M_k$ and its *respective* qubit $q_l \in M_j$ for all i, l.

Note that many quantum computing platforms that consider linking modules together with photonic links or similar already allow this degree of freedom. For this setting, we consider more general products than standard hypergraph products. Specifically, we show that the newly defined intermodular connectivity allows us to construct codes that can be described in the language of *balanced product codes* [19]. As before, these codes fully respect the architectural connectivity constraints. Before proceeding to the proof, let us shortly introduce the notion of balanced products of chain complexes.

A. Balanced product chain complexes

Breuckmann and Eberhardt (BE) introduced *balanced product codes*, which are analogously constructed to the balanced (or mixed) product of topological spaces [19]. This and related constructions can be used to construct *asymptotically good* quantum codes [17–21].

In order to define the balanced product of chain complexes, we need to discuss the balanced product of vector spaces. Let V, W be vector spaces with linear right and left actions, respectively, of a finite group G. The balanced product is defined as the quotient

$$V \otimes_G W = V \otimes W / \langle vg \otimes w - v \otimes gw \rangle,$$

where $v \in V, w \in W$, and $g \in G$. If V and W have bases X and Y, and G maps basis vectors to basis vectors then the basis of $V \otimes_G W$ is given by $X \times_G Y = X \times Y / \sim$. The equivalence relation "$\sim$" is defined as $(x, y) \sim (xg, g^{-1}y)$ for all $x \in X, y \in Y$, and $g \in G$. We can now extend this notion to chain complexes. Let C and D be chain complexes where C has a linear right action and D has a linear left action of a group G. The *balanced product double*

complex $C \boxtimes_G D$ is defined via

$$\left(C \underset{G}{\boxtimes} D\right)_{p,q} = C_p \otimes_G D_q$$

with horizontal and vertical differentials defined analogously to the double complex [Eq. (10)] that act on the quotients $C_p \otimes_G D_q$ of vector spaces C_p and D_q. The *balanced product complex* is the corresponding total complex:

$$C \otimes_G D = \mathrm{Tot}\left(C \underset{G}{\boxtimes} D\right).$$

We limit the discussion to cases where the vector spaces C_i, D_i are based and the action of G restricts to an action on these bases [19]. If G is a finite group of odd order, BE [19] gave a version of the Künneth formula that can be applied to the balanced product complex:

$$H_n\left(C \otimes_G D\right) \cong \bigoplus_{p+q=n} H_p(C) \otimes_G H_q(D). \tag{16}$$

We want to emphasize that, for certain cases, the balanced product is equivalent to related constructions such as the *lifted product* [41,52] and the *fiber bundle construction* [53].

B. Architecture-tailored codes from balanced products

To prove that the more general intermodular connectivity including twists allows us to construct better quantum codes than those constructed as hypergraph products, we consider cases where we can cast the balanced product $C \otimes_G D$ as a fiber bundle complex $B \otimes_\varphi D$. In this complex, B denotes the base, D denotes the fiber, and φ denotes the connection that describes the *twists* of the fiber along the base. Using the homological language, the connection φ represents an automorphism on the fiber D that alters the horizontal differentials of the double complex $B \boxtimes D$. Similarly to other products, the fiber bundle complex can be used to describe a quantum error-correcting code once we identify the data qubits and the parity checks correspondingly. Such codes are called *fiber bundle codes* [53]. In our modular architecture setting, it is crucial to correctly identify the base and fiber chain complex to ensure that connections are twisted between modules only. This idea is depicted in Fig. 10. Note that, twisting connections in the "other direction" (i.e., the connections are twisted between layers of respective qubits across all modules) generally results in a low connection count in each module. For many quantum platforms, the intramodular connections are considered to be faster to implement and less noisy, and, therefore, preferential over the qubit connections between distinct modules [11,54,55].

BE showed that, when C is a two-term complex and H is Abelian and acts freely on the bases of each C_i, then there exists a connection φ such that $C \otimes_G D = B \otimes_\varphi D$, where $B_i = C_i/\langle cg - c \rangle$ [19]. Therefore, a wide range of codes constructed from balanced products can be recast into the language of fiber bundle codes. Here we restrict ourselves to these cases and cast the following theorem in terms of fiber bundle codes.

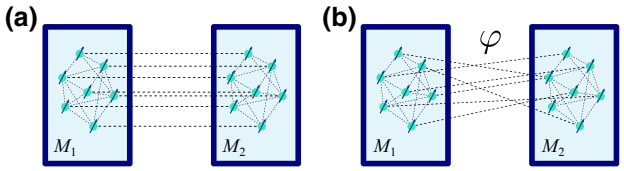

FIG. 10. Allowing twists of intermodule connections allows us to create better codes and yields a more general formulation.

Theorem 3. *Let D be a two-term and C a two- or three-term chain complex. Let their boundaries ∂^C and ∂^D define the intra- and intermodular connectivities as given in Definitions 1 and 3, respectively. Then, a fiber bundle code $\mathcal{E}$ corresponding to the chain complex $E = D \otimes_\varphi C$ respects the connectivity constraints of the architecture.*

Proof. The resulting chain complex E (corresponding to $\mathcal{E}$) is at least a three-term chain complex given that both C and D are at least two-term chain complexes. Therefore, we can identify chains of E_i with data qubits, E_{i+1} with Z parity checks, and E_{i-1} with X parity checks. The required qubit connectivity of $\mathcal{E}$ is defined by the parity-check matrices H^E, which are given by the boundary operators

$$\partial^E = \partial^h \oplus \partial^v = \partial_\varphi \oplus \mathrm{id}^D \otimes \partial^C, \tag{17}$$

where

$$\partial_\varphi(d_1 \otimes c) = \sum_{d_0 \in \partial^D d_1} d_0 \otimes \varphi(d_1, d_0)(c)$$

with $d_i \in D_i$ and c any i-chain of C. Here, φ denotes the connection and hence $\varphi(d_1, d_0)$ describes a specific element of the automorphism group acting on fiber C.

Note that i-chains of C label qubit c within each module and that i-chains of D label each module d. We can ensure that $\mathcal{E}$ respects the connectivity constraints of the architecture if both terms of Eq. (17) match the given qubit connectivity. We do so by looking at both of the terms separately. The first term in Eq. (17) indicates that each qubit c in a module d_1 is connected to qubits labeled $\varphi(d_1, d_0)c = c'$ in adjacent modules $d_0 \in \partial^D(d_1)$ for all d. This connectivity requirement is fully satisfied by our new definition of intermodular connectivity (Definition 3). For this exact reason, we consider the base of the fiber bundle to represent the code describing the intermodular connectivity as we want to *twist* connections only between the

modules. The basis elements of the second term in Eq. (17) are pairs of cells $(d, c) \in D \times C$ on which the boundary operator acts as $\partial^v : (d, c) \to (\mathrm{id}^D(d), \partial^C(c))$ for all d, c. By linear extension, this defines a map on $D \otimes C$. This is equally stated as in *each* module d, a qubit c is connected to qubits $\partial^C(c)$ for all c. This matches our Definition 1 for intramodular connectivity. Therefore, the required qubit connectivity of $\mathcal{E}$ is given by some additive combination of terms that define the intra- and intermodular connectivities, respectively. ■

Theorem 3 thus establishes a correspondence between modular architectures (including intermodular connectivity) and quantum codes that can be described using balanced product construction. While the proof considers balanced product codes that are equivalent to fiber bundle codes, we expect that a wider range of these codes respect the connectivity constraints—depending on the chosen group and group action.

As a simple example, we construct a balanced product code from two classical codes represented by chain complexes C and D. Complex C describes the intramodular connectivity and corresponds to a $d = 15$ cyclic repetition code as presented in Sec. IV. Code D describes the intermodular connectivity and is obtained by generating a random sparse parity-check matrix with dimensions 255×450 and a cyclic symmetry of order 15. It encodes $\dim(H_1(D)) = 195$ bits. Since both codes share the cyclic symmetry, we take the balanced product over group $\mathbb{Z}_{15}$. The product can be recast as a fiber bundle code and therefore satisfies Theorem 2. The resulting code $D \otimes_{\mathbb{Z}_{15}} C$ has $n = 705$ qubits and encodes at least $k = \dim(H_1(D/\mathbb{Z}_{15})) = 195/15 = 13$ logical qubits, where the calculation follows from the Künneth formula for fiber bundle codes [53]. The X and Z parity checks have approximate average weights of 10 and 6, respectively. An exhaustive probabilistic distance search with *QDistRnd* [56] showed that the code distance is at most 15; we believe that this bound is saturated. Hence, the balanced product code has expected parameters $[\![705, 13, 15]\!]$ and all qubits (including check qubits) can be partitioned into 47 modules with 30 qubits each. In comparison, encoding 13 qubits into rotated surface codes with the same distance requires 2925 data qubits. The parity-check matrix used for code D and parity-check matrices H_X and H_Z for the balanced product code can be found online [51].

VII. CONCLUSION

We have proposed a novel correspondence between two concepts from distinct, rapidly evolving domains: QLDPC product code constructions and modular quantum computing architectures. Using tools from homological algebra that have been used in constructions of product codes, we give a novel way to view modular architectures as chain complexes and show that valid quantum codes that respect the given architectural constraints can be constructed from products of such chain complexes. Our results constitute an essential further step towards closing the gap between recent theoretical breakthroughs around asymptotically good quantum codes and practical applications of QLDPC codes. In particular, due to the formalization of the looped pipeline architecture, we show practical relevance of our work.

As a direct extension of this work, it may be possible to generalize the considered constructions by allowing modules to have different intermodular connectivities. This renders the construction more complex and product constructions such as those used in this work might *a priori* not be applicable. Moreover, a further generalization of our approach could be considered by investigating layouts of modular architectures. That is, by considering layouts of layouts of modules. It may be possible that such constructions can also be described using product constructions as described in this work, but we leave an exact formulation open for future work. On a more practical note, it would be valuable to find small instances of QLDPC codes that can readily be realized on currently available architectures in order to draw comparisons to recent experimental breakthroughs around surface code realizations.

In general, many open questions around practical aspects of QLDPC codes remain. An important example is how to do fault-tolerant logic on QLDPC codes. Investigating further areas of their potential and more practically relevant regimes around QLDPC codes is a crucial area of research towards scalable fault-tolerant quantum computing.

ACKNOWLEDGMENTS

The authors would like to thank Nikolas P. Breuckmann for insightful discussions throughout this project and comments on the first draft. Furthermore, the authors would like to thank Simon Benjamin, Zhenyu Cai, and Michael Fogarty for helpful discussions at the initial stages of this project. A.S. acknowledges support from the European Union's Horizon 2020 research and innovation programme under Grant Agreement No. 951852 (QLSI). L.B. acknowledges funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation program (Grant Agreement No. 101001318). This work is part of the Munich Quantum Valley, which is supported by the Bavarian state government with funds from the Hightech Agenda Bayern Plus, and is supported by the BMWK on the basis of a decision by the German Bundestag through project QuaST, as well as by the BMK, BMDW, and the State of Upper Austria in the frame of the COMET program (managed by the FFG). This research was funded in part by the UKRI grant number EP/R513295/1. For the purpose of Open Access, the

[1] R. Buyya, *High Performance Cluster Computing: Programming and Applications*, Vol. 2 (Prentice Hall, 1999).

[2] D. Singh and C. K. Reddy, A survey on platforms for big data analytics, J. Big Data **2**, 1 (2015).

[3] C. Monroe, R. Raussendorf, A. Ruthven, K. R. Brown, P. Maunz, L.-M. Duan, and J. Kim, Large-scale modular quantum-computer architecture with atomic memory and photonic interconnects, Phys. Rev. A **89**, 022317 (2014).

[4] S. Bartolucci, P. Birchall, H. Bombin, H. Cable, C. Dawson, M. Gimeno-Segovia, E. Johnston, K. Kieling, N. Nickerson, and M. Pant *et al.*, Fusion-based quantum computation, arXiv preprint arXiv:2101.09310 (2021).

[5] A. Gold, J. Paquette, A. Stockklauser, M. J. Reagor, M. S. Alam, A. Bestwick, N. Didier, A. Nersisyan, F. Oruc, and A. Razavi *et al.*, Entanglement across separate silicon dies in a modular superconducting qubit device, Npj Quantum Inf. **7**, 1 (2021).

[6] D. Hucul, I. V. Inlek, G. Vittorini, C. Crocker, S. Debnath, S. M. Clark, and C. Monroe, Modular entanglement of atomic qubits using photons and phonons, Nat. Phys. **11**, 37 (2015).

[7] B. Buonacorsi, Z. Cai, E. B. Ramirez, K. S. Willick, S. M. Walker, J. Li, B. D. Shaw, X. Xu, S. C. Benjamin, and J. Baugh, Network architecture for a topological quantum computer in silicon, Quantum Sci. Technol. **4**, 025003 (2019).

[8] S. Bravyi, O. Dial, J. M. Gambetta, D. Gil, and Z. Nazario, The future of quantum computing with superconducting qubits, arxiv:2209.06841 (2022).

[9] K. Brown, Modular architectures for quantum computers, Bull. Am. Phys. Soc. **2022**, F37.0062 (2022).

[10] H. Bombin, I. H. Kim, D. Litinski, N. Nickerson, M. Pant, F. Pastawski, S. Roberts, and T. Rudolph, Interleaving: Modular architectures for fault-tolerant photonic quantum computing, arXiv preprint arXiv:2103.08612 (2021).

[11] J. Ramette, J. Sinclair, N. P. Breuckmann, and V. Vuletić, Fault-tolerant connection of error-corrected qubits with noisy links, arXiv preprint arXiv:2302.01296 (2023).

[12] J. Niu, L. Zhang, Y. Liu, J. Qiu, W. Huang, J. Huang, H. Jia, J. Liu, Z. Tao, and W. Wei *et al.*, Low-loss interconnects for modular superconducting quantum processors, Nat. Electron. **6**, 235 (2023).

[13] S. Bartolucci, P. Birchall, H. Bombin, H. Cable, C. Dawson, M. Gimeno-Segovia, E. Johnston, K. Kieling, N. Nickerson, M. Pant, Fernando Pastawski, Terry Rudolph, and Chris Sparrow, Fusion-based quantum computation, Nat. Commun. **14**, 912 (2023).

[14] J. Preskill, Fault-tolerant quantum computation (World Scientific, 1998), p. 213.

[15] D. A. Lidar and T. A. Brun, *Quantum error correction* (Cambridge University Press, 2013).

[16] B. M. Terhal, Quantum error correction for quantum memories, Rev. Mod. Phys. **87**, 3075 (2015).

[17] P. Panteleev and G. Kalachev, in *Proceedings of the 54th Annual ACM SIGACT Symposium on Theory of Computing* (2022), p. 375.

[18] I. Dinur, M.-H. Hsieh, T.-C. Lin, and T. Vidick, Good quantum ldpc codes with linear time decoders, arXiv preprint arXiv:2206.07750 (2022).

[19] N. P. Breuckmann and J. N. Eberhardt, Balanced product quantum codes, IEEE Trans. Inf. Theory **67**, 6653 (2021).

[20] A. Leverrier and G. Zémor, Quantum Tanner codes, arXiv preprint arXiv:2202.13641 (2022).

[21] T.-C. Lin and M.-H. Hsieh, Good quantum LDPC codes with linear time decoder from lossless expanders, arXiv preprint arXiv:2203.03581 (2022).

[22] M. A. Tremblay, N. Delfosse, and M. E. Beverland, Constant-Overhead Quantum Error Correction with Thin Planar Connectivity, Phys. Rev. Lett. **129**, 050504 (2022).

[23] N. Baspin and A. Krishna, Connectivity constrains quantum codes, Quantum **6**, 711 (2022).

[24] N. Baspin and A. Krishna, Quantifying Nonlocality: How Outperforming Local Quantum Codes is Expensive, Phys. Rev. Lett. **129**, 050505 (2022).

[25] F. M. Gambetta, C. Zhang, M. Hennrich, I. Lesanovsky, and W. Li, Long-Range Multibody Interactions and Three-Body Antiblockade in a Trapped Rydberg Ion Chain, Phys. Rev. Lett. **125**, 133602 (2020).

[26] K. A. Landsman, Y. Wu, P. H. Leung, D. Zhu, N. M. Linke, K. R. Brown, L. Duan, and C. Monroe, Two-qubit entangling gates within arbitrarily long chains of trapped ions, Phys. Rev. A **100**, 022332 (2019).

[27] H. Häffner, C. F. Roos, and R. Blatt, Quantum computing with trapped ions, Phys. Rep. **469**, 155 (2008).

[28] S. H. Choe and R. Koenig, Long-range data transmission in a fault-tolerant quantum bus architecture, arXiv preprint arxiv:2209.09774 (2022).

[29] E. Dennis, A. Kitaev, A. Landahl, and J. Preskill, Topological quantum memory, J. Math. Phys. **43**, 4452 (2002).

[30] A. Y. Kitaev, Fault-tolerant quantum computation by anyons, Ann. Phys. (N. Y) **303**, 2 (2003).

[31] S. B. Bravyi and A. Y. Kitaev, Quantum codes on a lattice with boundary, arXiv preprint arXiv:quant-ph/9811052 (1998).

[32] S. Krinner, N. Lacroix, A. Remm, A. Di Paolo, E. Genois, C. Leroux, C. Hellings, S. Lazar, F. Swiadek, and J. Herrmann *et al.*, Realizing repeated quantum error correction in a distance-three surface code, Nature **605**, 669 (2022).

[33] C. K. Andersen, A. Remm, S. Lazar, S. Krinner, N. Lacroix, G. J. Norris, M. Gabureac, C. Eichler, and A. Wallraff, Repeated quantum error detection in a surface code, Nat. Phys. **16**, 8755 (2020).

[34] J. Marques, B. Varbanov, M. Moreira, H. Ali, N. Muthusubramanian, C. Zachariadis, F. Battistel, M. Beekman, N. Haider, and W. Vlothuizen *et al.*, Logical-qubit operations in an error-detecting surface code, Nat. Phys. **18**, 80 (2022).

[35] Z. Chen, K. J. Satzinger, J. Atalaya, A. N. Korotkov, A. Dunsworth, D. Sank, C. Quintana, M. McEwen, R. Barends, and P. V. Klimov *et al.*, Exponential suppression

of bit or phase errors with cyclic error correction, Nature **595**, 383 (2021).

[36] R. Acharya, I. Aleiner, R. Allen, T. I. Andersen, M. Ansmann, F. Arute, K. Arya, A. Asfaw, J. Atalaya, and R. Babbush *et al.*, Suppressing quantum errors by scaling a surface code logical qubit, arXiv preprint arXiv:2207.06431 (2022).

[37] E. T. Campbell, B. M. Terhal, and C. Vuillot, Roads towards fault-tolerant universal quantum computation, Nature **549**, 172 (2017).

[38] N. H. Nickerson, Y. Li, and S. C. Benjamin, Topological quantum computing with a very noisy network and local error rates approaching one percent, Nat. Commun. **4**, 1 (2013).

[39] Z. Cai, A. Siegel, and S. Benjamin, Looped pipelines enabling effective 3D qubit lattices in a strictly 2D device, arXiv preprint arXiv:2203.13123 (2022).

[40] N. P. Breuckmann and J. N. Eberhardt, Quantum Low-Density Parity-Check Codes, PRX Quantum **2**, 040101 (2021).

[41] P. Panteleev and G. Kalachev, Degenerate quantum LDPC codes with good finite length performance, Quantum **5**, 585 (2021).

[42] H. Bombin and M. Martin-Delgado, Topological quantum error correction with optimal encoding rate, Phys. Rev. A **73**, 062303 (2006).

[43] M. H. Freedman, D. A. Meyer, and F. Luo, Z2-systolic freedom and quantum codes, in *Mathematics of quantum computation* (2002), p. 303.

[44] D. Gottesman, *Stabilizer codes and quantum error correction* (1997).

[45] M. Fogarty, personal communication (2022-04-01).

[46] N. Delfosse, M. E. Beverland, and M. A. Tremblay, Bounds on stabilizer measurement circuits and obstructions to local implementations of quantum LDPC codes, arXiv preprint arXiv:2109.14599 (2021).

[47] C. A. Pattison, A. Krishna, and J. Preskill, Hierarchical memories: Simulating quantum LDPC codes with local gates (2023), arXiv:2303.04798 [quant-ph].

[48] J.-P. Tillich and G. Zémor, Quantum LDPC codes with positive rate and minimum distance proportional to the square root of the blocklength, IEEE Trans. Inf. Theory **60**, 1193 (2013).

[49] W. Zeng and L. P. Pryadko, Higher-Dimensional Quantum Hypergraph-Product Codes with Finite Rates, Phys. Rev. Lett. **122**, 230501 (2019).

[50] O. Higgott and N. P. Breuckmann, Improved single-shot decoding of higher dimensional hypergraph product codes, arXiv preprint arXiv:2206.03122 (2022).

[51] https://github.com/PurePhys/QuantumPCMs (2022).

[52] P. Panteleev and G. Kalachev, Quantum LDPC codes with almost linear minimum distance, IEEE Trans. Inf. Theory **68**, 213 (2021).

[53] M. B. Hastings, J. Haah, and R. O'Donnell, in *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing* (2021), p. 1276.

[54] E. T. Campbell, Distributed quantum-information processing with minimal local resources, Phys. Rev. A **76**, 040302 (2007).

[55] L. J. Stephenson, D. P. Nadlinger, B. C. Nichol, S. An, P. Drmota, T. G. Ballance, K. Thirumalai, J. F. Goodwin, D. M. Lucas, and C. J. Ballance, High-Rate, High-Fidelity Entanglement of Qubits Across an Elementary Quantum Network, Phys. Rev. Lett. **124**, 110501 (2020).

[56] L. P. Pryadko, V. A. Shabashov, and V. K. Kozin, QDistRnd: A gap package for computing the distance of quantum error-correcting codes, J. Open Source Software **7**, 4120 (2022).