

XYZ Ruby Code: Making a Case for a Three-Colored Graphical Calculus for Quantum Error Correction in Spacetime

Julio C. Magdalena de la Fuente^{1,*}, Josias Old^{2,3,†}, Alex Townsend-Teague,¹
Manuel Rispler^{2,3}, Jens Eisert^{1,4} and Markus Müller^{2,3}

¹*Dahlem Center for Complex Quantum Systems, Freie Universität Berlin, Berlin 14195, Germany*

²*Institute for Theoretical Nanoelectronics (PGI-2), Forschungszentrum Jülich, Jülich, Germany*

³*Institute for Quantum Information, RWTH Aachen University, Aachen, Germany*

⁴*Helmholtz-Zentrum Berlin für Materialien und Energie, Berlin 14109, Germany*



(Received 14 September 2024; revised 20 December 2024; accepted 30 January 2025; published 28 March 2025)

Analyzing and developing new quantum error-correcting (QEC) schemes is one of the most prominent tasks in quantum computing research. In such efforts, introducing time dynamics explicitly in both analysis and design of error-correcting protocols constitutes an important cornerstone. In this work, we present a graphical formalism based on tensor networks to capture the logical action and error-correcting capabilities of any Clifford circuit with Pauli measurements. We showcase the functioning of the formalism on new Floquet codes derived from topological subsystem codes, which we call *XYZ ruby codes*. Based on the projective symmetries of the building blocks of the tensor network we develop a framework of *Pauli flows*. Pauli flows allow for a graphical understanding of all quantities entering an error-correction analysis of a circuit, including different types of QEC experiments, such as memory and stability experiments. We lay out how to derive a well-defined decoding problem from the tensor-network representation of a protocol and its Pauli flows alone, independent of any stabilizer code or fixed circuit. Importantly, this framework applies to all Clifford protocols and encompasses both measurement-based and circuit-based approaches to fault tolerance. We apply our method to our new family of dynamical codes, which are in the same topological phase as the $2 + 1$ -dimensional color code, making them a promising candidate for low-overhead logical gates. In contrast to its static counterpart, the dynamical protocol applies a $\mathbb{Z}_3$ automorphism to the logical Pauli group every three time steps. We highlight some of its topological properties and comment on the anyon physics behind a planar layout. Lastly, we benchmark the performance of the XYZ ruby code on a torus by performing both memory and stability experiments and find competitive circuit-level noise thresholds of approximately equal to 0.18%, comparable with other Floquet codes and $2 + 1$ -dimensional color codes.

DOI: [10.1103/PRXQuantum.6.010360](https://doi.org/10.1103/PRXQuantum.6.010360)

I. INTRODUCTION

Reliable quantum computing can be pursued even in the presence of errors: To this aim, the field of *quantum error correction* (QEC) is concerned with designing quantum error-correcting codes, protocols for decoding, and fault-tolerant schemes for quantum computing [1]. With the

first QEC codes being proposed in the 1990s [2], only in recent years have steps been taken towards the experimental implementation of large-scale quantum error-correcting codes [3–12], with remarkable progress. Current QEC schemes, however, are still challenging to scale up with current technological capabilities, or require significant resource overhead. Therefore, it remains a fruitful and at the same time crucial endeavor to investigate novel schemes to design and analyze protocols for fault-tolerant storage and computation.

A stabilizer-based error-correcting code is defined to be the joint eigenspace of a group of commuting operators [13]. Hence, measuring these operators repeatedly gives rise to an error-correcting protocol where the measurement outcomes are used to infer if a physical error affected the system and potentially correct for that. Importantly, the order in which the stabilizer operators are measured affects

*Contact author: jmagdalena@zedat.fu-berlin.de,
jm@juliomagdalena.de

†Contact author: j.old@fz-juelich.de

‡These authors contributed equally.

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

neither the evolution of encoded logical information nor the possibility of inferring a suitable recovery given the measurement outcomes of the protocol.

Recently, novel schemes have been presented that rely on repeatedly measuring noncommuting operators in a specific order. While we should not generically expect such a scheme to preserve any information, one can carefully choose the operators that are being measured to design a fault-tolerant *dynamical error-correcting scheme* [14]. Removing the commutativity requirements allows one to circumvent many obstacles that appear for naïve stabilizer measurements. Importantly, the weight, i.e., the physical qubit support of the check operators that are measured throughout a protocol, can be drastically reduced. This constitutes an important avenue of research as high-weight operators usually come along with complicated quantum circuits that induce substantial circuit-level noise in many architectures. Additionally, the time ordering allows one to protect a qubit with a measurement sequence that without time ordering (i.e., as a subsystem code [15]) would not encode any logical qubits. Dynamical schemes where the same measurement sequence is repeated periodically over time have been coined *Floquet codes* [16,17] and after a first instance, the honeycomb code, presented in Ref. [14], the concept has attracted much attention.

For stabilizer-based QEC codes, topological stabilizer codes [18] have been promising candidates for reliable quantum information storage due to their intrinsic scalability with only local measurements. Similarly, in the context of dynamical codes, it has taken little time for constructions to arise based on models for topologically ordered phases [19–24]. It turns out that in topological protocols defined in $2 + 1$ spacetime dimensions, where time ordering is important, the effective logical degrees of freedom can be understood in terms of the data of an underlying topological phase, characterized by an *Abelian anyon theory* by considering associated 1-form symmetries [25]. Importantly, a spacetime perspective is of essence and the operators carrying the encoded information cannot be understood from a static, spatial perspective alone in order to argue about fault tolerance in the presence of circuit-level noise.

In this work, we contribute to a more thorough understanding of QEC in spacetime by extending existing graphical formalisms for QEC to a tensor-network-based formalism that captures all protocols defined from Clifford operations. On a higher level, our aims are twofold: We extend the common ZX calculus [26] and obtain a graphical representation of protocols that is closer to the actual components of a circuit involving Y measurements. Tracking Pauli operators through these networks yields a complete description of all algebraic quantities entering a QEC protocol and an associated decoding problem. Moreover, we showcase the usefulness of this framework by analyzing a class of dynamical error-correcting protocols

involving Y measurements. Specifically, we focus on topological codes and introduce a new family of topological Floquet codes. These codes can be constructed from any 2-colex [27], a trivalent and plaquette-tricolorable planar graph and is defined by a measurement sequence on the gauge generators of Bombin’s topological subsystem code [28]. Within this family, we mainly focus on the codes obtained from a hexagonal lattice, defined on a system of qubits composed of three qubits per vertex of the original lattice. We refer to this subfamily of protocols as *XYZ ruby codes*. As a topological code the associated subsystem code is related to an Abelian anyon theory. In this case, it is the three-fermion theory [29,30], which is a chiral subtheory of the color code anyon model. In the dynamical setting we find, however, that the actual logical operators that are protected and corrected for in the protocol are related to the color-code anyon model. As such, our code realizes the color-code phase in spacetime and hence offers a richer set of natively implementable fault-tolerant logical gates. For different schedules, this has already been observed in Ref. [22]. In contrast to other known schedules though, the qubits themselves are not in a state of the color-code phase at every time step of the protocol. They undergo an *instantaneous phase transition* during the measurement sequence to a phase that hosts more anyons. To the best of our knowledge, this is the first two-dimensional Floquet code where this happens.

In order to showcase the strength of the three-colored graphical calculus for QEC, we go so far as to design, simulate, and decode circuits for the XYZ ruby code. Concretely, we combine the spacetime tensor-network methods for QEC with the understanding of the anyon physics of the ruby code to devise fault-tolerant QEC experiments that benchmark different properties of the code. In particular, we perform memory experiments and the first stability experiments for a dynamical code, under different error models (phenomenological, EM3 and circuit-level noise). We find logical error rates and thresholds comparable to other Floquet codes encoding the same number of qubits.

A. Guide to the reader

In Sec. II, we give the necessary background on dynamical QEC protocols and two-dimensional topological codes. In Sec. III, we introduce the XYZ ruby code as a Floquet code and analyze the topological phases of the instantaneous states. These sections can be read independently of Sec. IV, which provides a pedagogical introduction into the three-colored graphical calculus for QEC, essential to understand the remainder of this work. The reader interested in a more rigorous treatment is referred to Appendix A where we also present our main technical mathematical result on how to understand the sequential application of Clifford maps (including measurement).

In Sec. V, we start by giving our perspective on a general “probing experiment” in spacetime. We then elaborate on the fault-tolerance properties of the protocols and construct and perform numerical experiments. We present the necessary considerations when designing planar codes, respectively, boundaries in spacetime, of a topological protocol defined in the bulk in Sec. VI and sketch a general construction for topological codes. Lastly, in Sec. VII, we comment on more general aspects of spacetime QEC, make connections to existing formalisms and comment on emerging perspectives and open questions arising from a tensor-network representation.

II. PRELIMINARIES

In this section, we introduce the important concepts that are required for our analysis of the XYZ ruby code. First, we give a high-level introduction to dynamical codes based on Pauli measurements. This class of codes has been introduced as instantaneous stabilizer group codes, or *ISG codes* in Ref. [17]. Second, we introduce graphical tools to analyze such codes in a spacetime tensor-network language inspired by the *ZX* calculus [26,31]. Importantly, we include tensors to represent *Y* measurements more directly in our framework. This renders the analysis of dynamical protocols involving measurements in the *Y* basis much easier and more efficient. Lastly, for readers more familiar with algebraic structures of topologically ordered systems, we give an overview on how certain algebraic features of topological stabilizer codes in two spatial dimensions can be described by an Abelian anyon model [32]. We will focus on the anyon models associated to toric and color codes in which most of the abstract mathematical language simplifies significantly.

Before we go into details, let us fix some notation. Consider a system of n qubits with a Hilbert space $(\mathbb{C}^2)^{\otimes n}$. For the rest of this work, we denote the Pauli matrices on qubit j by

$$Z_j = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}, \quad X_j = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \quad (1)$$

and $Y_j = iX_jZ_j = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}.$

These Pauli matrices together with a phase i generate the *n-qubit Pauli group*, denoted by $\mathcal{P}_n$. Any element $P \in \mathcal{P}_n$ can be given a *weight* $w : \mathcal{P}_n \rightarrow \mathbb{Z}_{\geq 0}$, which we define to be the number of tensor factors (qubits) on which P acts with a Pauli operator that is not proportional to the identity.

We call an Abelian subgroup of the Pauli group a *stabilizer group* and the common $+1$ eigenspace of all elements in that group the associated *stabilizer code*. Importantly, any state in the stabilizer code is stabilized by any Pauli

element in the stabilizer group. If we want the stabilized subspace to be at least one dimensional, we require additionally that $-\mathbb{1}$ is not contained in that group [13,33].

A. ISG and Floquet codes

Projectively measuring a (multiqubit) Pauli operator P projects a system of qubits onto the eigenspace associated to the measurement outcome. If the system was in a stabilizer state before the measurement, the postmeasurement state will again be a stabilizer state whose stabilizer group can be efficiently calculated from the initial stabilizer group $\mathcal{S}_0$ and the Pauli operator P [34]. We call the stabilizer group of the system immediately after a (Pauli) measurement the *instantaneous stabilizer group (ISG)*. This term has first been introduced in Ref. [14] together with the honeycomb code, the first Floquet code in the literature. The idea of *ISG* and *Floquet* codes is to use measurements to drive a quantum system between different ISGs while preserving the logical information. The goal of the following paragraph is to make this more specific.

We define an *ISG code* via a sequence (ordered list) of sets of Pauli operators

$$\mathcal{M} = [\mathcal{M}_0, \mathcal{M}_1, \mathcal{M}_2, \dots], \quad (2)$$

such that $\langle \mathcal{M}_i \rangle$ is a stabilizer group for all i [17]. We call elements in $\mathcal{M}_i$ *checks*. In principle, one can consider sequences with a (countably) infinite number of elements, although for this work it suffices to consider finite ones. Since $\langle \mathcal{M}_i \rangle$ is Abelian, measuring all checks in $\mathcal{M}_i$ projects the system unambiguously onto the subspace stabilized by $\{(-1)^{m_j} M_{i,j} \mid M_{i,j} \in \mathcal{M}_i\}$. The binary numbers $\{m_j \in \{0, 1\} \mid j = 1, 2, \dots, |\mathcal{M}_i|\}$ label the measurement outcomes of each of the measured Pauli operators. Note that the structure of the stabilized subspace is the same independent of the measurement outcome. In particular, only the signs of the stabilizer generators will differ. Hence, without restricting generality, we will later often assume that all measurement outcomes are 0 to simplify the analysis of the instantaneous code states. To avoid notational clutter, we denote the operation of “measuring all elements in $\mathcal{M}_i$ ” by m_i . If the system has initially been stabilized by a stabilizer group $\mathcal{S}_i$, measuring $\mathcal{M}_i$ drives the system into a new stabilizer group $\mathcal{S}_{i+1}$ that can be efficiently constructed from $\mathcal{S}_i$ and $\mathcal{M}_i$,

$$\mathcal{S}_i \xrightarrow{m_i} \mathcal{S}_{i+1}. \quad (3)$$

Starting from an initial stabilizer group $\mathcal{S}_0$, an ISG code hence defines a sequence of ISGs

$$\mathcal{S}_0 \xrightarrow{m_0} \mathcal{S}_1 \xrightarrow{m_1} \mathcal{S}_2 \xrightarrow{m_2} \dots \quad (4)$$

In most cases, we will assume $\mathcal{S}_0 = \{\mathbb{1}\}$, i.e., we start with an arbitrary state. To initialize a particular logical state,

however, one might want to fix S_0 to some other stabilizer group, for example $\langle \{Z_i\}_{i=1}^n \rangle$, stabilizing the product state vector $|0\rangle^{\otimes n}$.

A *Floquet* code is defined by a finite sequence $\mathcal{M}$. The length of $\mathcal{M}$ is referred to as the *period* $T \in \mathbb{Z}_{>0}$ of the Floquet code and the measurements are repeated periodically after the first round $m_0, m_1, \dots, m_{T-1}$, i.e., we define $\mathcal{M}_i = \mathcal{M}_{i \bmod T}$. The number of logical qubits encoded in the subspace stabilized by a stabilizer group $\mathcal{S}_i$ is captured by its *rank* r_i , the minimal number of independent generators. Specifically, the number of encoded logical qubits is given by $k_i = n - r_i$. Note that r_i is a nondecreasing function in i , where i is now associated to the *time steps* in the measurement protocol.

In this work, in order to encode a logical qubit in an ISG code, we require that there exists $\hat{i} \in \mathbb{Z}_{\geq 0}$ such that r_i is constant for all $i \geq \hat{i}$. Following Ref. [17], we say that the ISG code *establishes* $k = n - r_{\hat{i}}$ logical qubits after $\hat{i}$ measurements. This is closely connected to the fact that all pairs $(\mathcal{S}_i, \mathcal{S}_{i+1})$ for $i \geq \hat{i}$ form a *reversible pair* of stabilizer groups in the sense of Ref. [35]: Two stabilizer groups $(\mathcal{S}_i, \mathcal{S}_{i+1})$ form a reversible pair if any element $s \in \mathcal{S}_i$ that commutes with $\mathcal{S}_{i+1}$ is also contained in $\mathcal{S}_{i+1}$ and, vice versa, and element $s' \in \mathcal{S}_{i+1}$ that commutes with $\mathcal{S}_i$ is contained in $\mathcal{S}_i$. Importantly, this condition is not sufficient to have a fault-tolerant ISG code but only guarantees that the logical information is preserved by the measurement that switches from $\mathcal{S}_i$ to $\mathcal{S}_{i+1}$. To incorporate error-correcting abilities, one needs to specify how to decode on the set of measurement outcomes. In the next section we illustrate how we use graphical methods to analyze the fault-tolerance properties of a spacetime measurement circuit associated to an ISG code. For more details on formal definitions and basic properties of sequences of ISGs, we refer the interested reader to Refs. [17,35].

B. Topological stabilizer codes and their anyon models

In two-dimensional topological stabilizer codes *anyons* appear as local (pointlike) violations of stabilizers that cannot be created by any local operator [36]. They can be associated to topological (quasiparticle) excitations of an exactly solvable Hamiltonian

$$H = - \sum_i S_i, \quad (5)$$

defined by a choice $\{S_i\}$ of a (local) generating set of the stabilizer group. If $\langle \{S_i\} \rangle$ fulfills the error-detection condition [33] for *any local error*, the above Hamiltonian is topologically ordered [37] and we identify the (degenerate) ground space of that Hamiltonian with the code space of $\langle \{S_i\} \rangle$.

In two spatial dimensions, any such excitations can be created at the endpoints of Pauli operators supported

on stringlike regions [38–40]. The associated (stabilizer equivalence classes of) string operators can be labeled by an (Abelian) *fusion group* A and a complex phase $\theta : A \rightarrow U(1)$. The fusion group A relates to how the (equivalence classes of) string operators multiply, while the phase θ captures the commutativity of a string of type a with itself. Specifically, a string operator defined along a self-intersecting path is stabilizer equivalent to a string operator defined on a nonintersecting path with the same endpoints times a complex phase, given by θ . In fact, this data (A, θ) defines an *Abelian anyon model* [32], an Abelian braided fusion category. Interestingly, any Pauli topological stabilizer code realizes an anyon model that is equivalent to some (finite) number of copies of the toric-code anyon model [39,40].

Many properties of topological codes become apparent when interpreting them as ground spaces of the topologically ordered Hamiltonians introduced above. In particular, this perspective is helpful to construct logical operators as nontrivial anyon string operators, to describe native locality-preserving logical gates in terms of symmetries of the anyon model [41] and to construct boundaries and domain walls in terms of anyon condensation [19]. In this section, we introduce the toric-code and color-code anyon model and show the equivalence of the latter to two copies of the former explicitly.

1. Toric-code anyon model

The toric code is the most commonly known topological code for quantum error correction. It is constructed from the *quantum double model* of the Abelian group $\mathbb{Z}_2$ [42]. It can be defined on any planar graph by placing qubits on the edges of that graph. For each vertex v , we introduce a vertex stabilizer A_v acting with X on all qubits on the edges incident to that vertex. Similarly, we introduce a stabilizer B_p for each plaquette p that acts with Z on every qubit on the edges forming the boundary of that plaquette. Planarity of the underlying graph guarantees commutativity of both types of stabilizers. In Fig. 1, we illustrate how anyons appear at the endpoints of the string operators in the toric code defined on a triangular lattice.

The *toric code* hosts four types of anyons, commonly denoted by $A_{\text{TC}} = \{1, e, m, f\}$. Under fusion, they form the group $\mathbb{Z}_2 \times \mathbb{Z}_2$,

$$a \times 1 = 1 \times a = a, \quad a \times a = 1, \quad \forall a \in A_{\text{TC}}, \quad (6a)$$

$$e \times m = m \times e = f. \quad (6b)$$

Additionally, a braiding is defined via

$$\theta_1 = \theta_e = \theta_m = 1 \quad \text{and} \quad \theta_f = -1. \quad (7)$$

In analogy with usual particles, we call $1, e$, and m bosons (they have topological spin 1), and f a fermion (it has topological spin -1). On the lattice, the nontrivial spin of f is

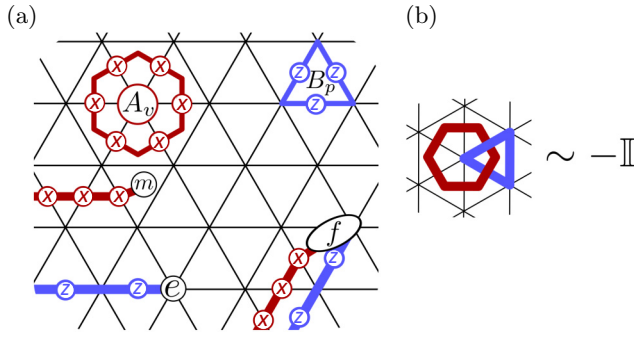


FIG. 1. A toric-code model can be defined from any planar graph. (a) The associated codespace is defined by placing a qubit on every edge and adding stabilizer generators on vertices and plaquettes. The plaquette stabilizers act with Pauli-Z operators around the qubits on the boundary of that plaquette and can be thought of as a (homologically trivial) loop (cycle) of Z operators on the direct lattice. The vertex stabilizers act with Pauli-X operators on all qubits adjacent to the vertex and can be understood as acting along a (homologically trivial) loop on the dual lattice (cocycle). Isolated violations of these stabilizers can be understood as anyons of the toric-code anyon model and are created at endpoints of X strings on the dual lattice (m anyons violating a plaquette stabilizer) or Z strings on the direct lattice (e anyons violating a vertex stabilizer). Combining these two types of string operators such that their endpoints are close to each other creates the fusion product of e and m , which is a fermion f . The fermionic statistics can be seen from the fact that the string operators of e and m anticommute if they intersect an odd number of times. (b) We depict a small example where two interlinked loops of string operators act like -1 on the code space.

captured by the fact that the string operators creating e and m anyons do not commute, see Fig. 1.

2. Color-code anyon model

The *color code* has been introduced as a family of topological stabilizer codes defined on any trivalent planar graph with tricolorable plaquettes [28]. In this construction, the qubits are placed on the vertices and two stabilizer generators are defined for each plaquette. One type, S_p^X , acts with X on all qubits surrounding plaquette p and the other type, S_p^Z , acts with Z on all qubits surrounding p . The trivalency and colorability ensures commutativity of all the stabilizers on any of these lattices.

In Fig. 2, we illustrate the color code defined on a hexagonal lattice. We label each of the violations according to the color and Pauli label of the operators creating it. For example, a violation of a single S_p^X (that is not simultaneously a violation of S_p^Z) for a red plaquette p is labeled by rx since it is localized at a red plaquette and can be created by a product of Pauli-Z operators. A single violation of that type, labeled by the color $c \in \{r, g, b\}$ and Pauli label $p \in \{x, y, z\}$, can only be created by a nonlocal operator. For example, when placed on an infinite plane, this could

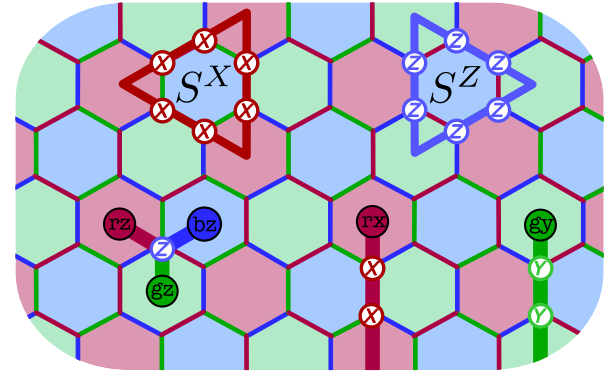


FIG. 2. The hexagonal color code is defined on a system of qubits played on vertices of a hexagonal lattice. The stabilizer group of the associated code has two types of generators per plaquette: One acting with a product of Pauli X on each qubit surrounding the plaquette, and one acting with Pauli Z on the same qubits. A state violating a set of these generators can be associated with a state with anyonic excitations at the plaquettes whose stabilizers are violated, labeled by the color of the plaquette and the Pauli operator that created them when applied to a codestate. A single Pauli error creates a triple of anyons, one at each plaquette touching the qubit. This gives rise to the anyon fusion rule $rx \times gx \times bx = 1$. An isolated anyon can be created at the endpoints of certain string operators. For example, the boson rx is created at the endpoints of a string of X operators acting on pairs of qubits along red edges, connecting red plaquettes. Similarly, one can create the other bosons in the color-code anyon model by acting with two-qubit Pauli operators of the form $P \otimes P$ on the qubits on edges of a given color.

be a half-infinite string operator connecting plaquettes of the same color c . Any color-code boson can be created by a product of string operators of definite Pauli and color type, which we label with cp . For now, we will call these string operators *generating string operators* and the associated anyons *generating anyons* [43]. Additionally, there are products of string operators labeled by different colors and Paulis that create an inequivalent anyon. We will see in the next paragraph that these will be the fermions in the color code.

In the color-code stabilizer model, we notice that any single-qubit Pauli creates a triple of violations on the plaquettes surrounding the qubit. On the level of the anyon model this means that any triple of such violations, with the same Pauli label corresponds to the trivial anyon,

$$rx \times gx \times bx = 1, \quad (8)$$

and analogously for the other Pauli labels. Similarly, the same holds for a triple of anyons with the same color but different Pauli labels.

It is instructive to order the generating anyons in a 3×3 “magic square,” where each row corresponds to a Pauli and

each column corresponds to a color label [41],

$$\begin{array}{c|c|c} \text{rx} & \text{gx} & \text{bx} \\ \hline \text{ry} & \text{gy} & \text{by} \\ \hline \text{rz} & \text{gz} & \text{bz} \end{array} \quad (9)$$

The square is aligned such that the fusion product of every row and of every column is 1. Phrased differently, the fusion product of any two anyons in the same row (column) is the third anyon in the same row (column). Products of two anyons that do not share a row or a column are anyons of the color code outside of this table. Taking the constraints above into account, we count six additional inequivalent nontrivial anyons. Looking at the string operators, we notice that each anyon in the square is a boson and the six not directly represented in the square are fermions, i.e., have topological spin -1 , see Fig. 2. We conclude that the color code has 16 inequivalent anyons: the trivial anyon 1, nine bosons in the table above, and six inequivalent products of these bosons that are fermions. In the following, we denote the anyon model of the color code with $\mathcal{C}_{\text{CC}}$.

As stated before, any two-dimensional topological Pauli stabilizer code is described by an anyon model that is equivalent to some number of layers of the toric code. In particular, this is also true for the color-code anyon model. Specifically, there exists an *unfolding* [44] of the color-code anyon model into two copies of the toric code. We demonstrate the unfolding by relabeling a set of generators for the color-code anyon model to a set of generators of the anyon model of two decoupled layers of the toric code. For this we label each anyon of two layers of the toric code by (a, b) , where $a, b \in \{1, e, m, f\}$ label the anyon restricted on layer one, respectively, two. For example, we can identify

$$\begin{array}{c|c|c} \text{rx} & \text{gx} & \text{bx} \\ \hline \text{ry} & \text{gy} & \text{by} \\ \hline \text{rz} & \text{gz} & \text{bz} \end{array} \leftrightarrow \begin{array}{c|c|c} (e, 1) & (e, e) & (1, e) \\ \hline (e, m) & (f, f) & (m, e) \\ \hline (1, m) & (m, m) & (m, 1) \end{array} \quad (10)$$

In this unfolding, the six fermions of the color code are labeled by

$$\mathfrak{f}_1 = (1, f), \quad \mathfrak{f}_2 = (f, e), \quad \mathfrak{f}_3 = (f, m), \quad (11a)$$

$$\mathfrak{f}_4 = (f, 1), \quad \mathfrak{f}_5 = (e, f), \quad \mathfrak{f}_6 = (m, f). \quad (11b)$$

Note that $\mathfrak{f}_1, \mathfrak{f}_2, \mathfrak{f}_3$ as well as $\mathfrak{f}_4, \mathfrak{f}_5, \mathfrak{f}_6$ form a closed subtheory of the color-code anyons. When viewed as an anyon model on its own, each of these subtheories are equivalent to the “three-fermion” anyon model [19,29,32]. The unfolding into two copies of the toric-code anyon model shows that the color-code anyon model can be identified

as the Abelian group $A_{\text{CC}} = \mathbb{Z}_2^{\times 4}$ together with a topological spin $\theta_{\text{CC}} = \theta_{\text{TC}} \otimes \theta_{\text{TC}}$ that acts like the product of toric-code spins on the two $\mathbb{Z}_2 \times \mathbb{Z}_2$ factors of A_{CC} .

The symmetries of an anyon model play an important role in the encoding and manipulation of logical information in topological codes. A symmetry of an anyon model A is a permutation of the anyons that leaves all the anyonic data invariant, i.e., preserves the fusion as well as the braiding properties. For the two-dimensional color code, the group formed by all symmetries, i.e., the *automorphism group* $\text{Aut}(A_{\text{CC}})$, can be understood in terms of symmetries of the boson table in Eq. (9) [41]. One can permute the rows and columns individually—permuting color or Pauli labels—and also mirror along the diagonal—exchanging color and Pauli label. This yields an automorphism group of $(S_3 \times S_3) \rtimes \mathbb{Z}_2$. For a more elaborate treatment of the color-code anyon model we refer the reader to Refs. [19,41].

III. THE XYZ RUBY CODE

In this section, we present two variants of the XYZ *ruby code*. Both are defined on the same lattice of qubits and checks of only weight 2. We first analyze a period-3 protocol and then a period-6 protocol, which can be thought of as a rewinding version of the period-3 protocol (in the sense of Ref. [22]). We find that the logicals of the protocol can be described by a color-code anyon model, although the instantaneous state is not always in the same phase as a color-code state.

The XYZ ruby code is defined on a system of qubits on the vertices of the ruby lattice shown in Fig. 3, tessellating a compact, oriented, two-dimensional manifold. Additionally, we require that we can three color the interior faces, as indicated in Fig. 3. Later, we refer to these hexagonal faces as *plaquettes* and the rings of 18 qubits around each hexagon as *plaques*. To each edge, we associate a two-body Pauli check: A XX check to every red edge, a YY check to every green edge and a ZZ check to every blue edge. The union of these checks generate the gauge group of Bombin’s subsystem code [45]. As we will see in this section, adding a schedule on the checks reveals richer code properties than just the ones of the subsystem code. We refer the reader interested in the relationship between the XYZ ruby code and the topological subsystem codes from Ref. [45] to Sec. VII A.

The (period-3) XYZ ruby code is defined by the following sequence of checks:

$$\mathcal{M} = [\textcircled{Z} \textcircled{Z}, \textcircled{X} \textcircled{X}, \textcircled{Y} \textcircled{Y}], \quad (12)$$

i.e., we define a protocol where we periodically measure the blue, red, and green edges in the respective bases.

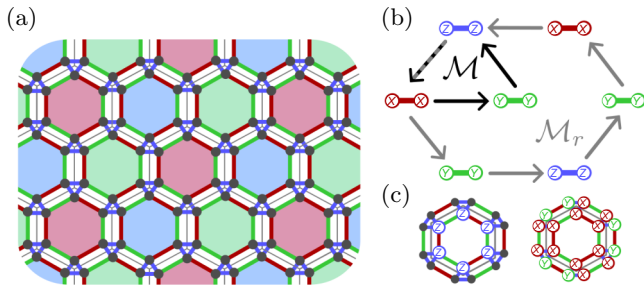


FIG. 3. (a) The XYZ ruby code is defined on qubits placed on the vertices of the ruby lattice with three-colorable plaquettes. The error-correcting protocol only involves weight-2 measurements between neighboring qubits. (b) The color of the edges indicates the Pauli basis of the measurement. Along a blue edge, a ZZ measurement is performed, along a green edge a YY measurement is performed and along a red edge a XX measurement is performed. In this work, we consider two different schedules, one with periodicity three (black arrows) and one with periodicity 6 (gray arrows). The period-6 schedule is a rewinding version of the period-3 schedule. (c) Both schedules are designed to read out the stabilizers of the associated subsystem code [28]. For each plaque(tte), the subsystem code has two independent stabilizer generators.

A. Establishing four logical qubits

Having defined the measurement sequence, we now track the ISGs through multiple rounds of measurements following the stabilizer formalism with measurements [13, 14]. Without loss of generality, we can assume the measurement outcomes are all +1, i.e., that the measurement projects onto the +1 eigenspace of all measurements in that round, see Refs. [17,31]. We start at time $t = -4$ with an arbitrary state, stabilized by $\mathcal{S}_{-4} = \{1\}$.

The first round of measurements adds all ZZ terms to the stabilizer group,

$$\mathcal{S}_{-3} = \langle \textcircled{Z} \text{---} \textcircled{Z} \rangle, \quad (13)$$

where the set of generators is understood to be the set of all ZZ operators acting on qubits connected by a blue edge. Note that at this time step the stabilized states are superpositions of codestates of three-qubit repetition codes on the triangles.

The next measurements are XX measurements on the red edges in Fig. 3. None of the ZZ operators measured in the previous time step commutes with any of the XX operators. However, there are extensively many elements in $\mathcal{S}_{-3}$ that do commute with XX on the red edges. They are Pauli- Z operators acting on qubits connected by cycles composed of alternating red and blue edges. This includes homologically trivial cycles generated by cycles around

each plaquette,

$$\text{each plaquette,} \quad (14)$$

and homologically nontrivial cycles around the handles of the ambient manifold,

$$\text{and homologically nontrivial cycles around the handles of the ambient manifold,} \quad (15)$$

The resulting stabilizer group at $t = -2$ is given by

$$\mathcal{S}_{-2} = \left\langle \begin{array}{c} \text{local generators (weight 2 and 12)} \\ \text{nontrivial loops around the torus} \end{array} \right\rangle. \quad (16)$$

Note that the generators in the first line are local and of constant weight (2 and 12), whereas the generators in the second line are supported on nontrivial loops around the torus and hence their support grows extensively with the system size. The path along which the nonlocal Z stabilizers act can be deformed (locally) by applying the local Z stabilizers.

The next measurements, YY along all green edges, again anticommute with the measurements of the previous time step. The product of XX edges within each hexagonal plaquette commutes with all of the YY measurements and hence remains in the stabilizer group. Additionally, the product of both trivial and nontrivial Z loops with the XX terms along their paths acts only in the Y basis and hence also commutes with the YY measurements. Taken together, the stabilizer group at $t = -1$ reads

$$\mathcal{S}_{-1} = \left\langle \begin{array}{c} \text{local generators (weight 2 and 12)} \\ \text{nontrivial loops around the torus} \end{array} \right\rangle. \quad (17)$$

At $t = 0$, we measure ZZ along the blue edges again. None of the YY generators above commute with all ZZ measurements. Moreover, none of the nonlocal stabilizers remain in the stabilizer group after the ZZ measurements. Both the weight-12 Y loops as well as the weight-6 X plaquettes can be multiplied with YY edges to commute with the just-measured ZZ operators. Combined, the XYZ ruby code establishes at $t = 0$, with the ISG

$$S_0 = \left\langle \begin{array}{c} \text{Diagram of stabilizer group } S_0 \end{array} \right\rangle. \quad (18)$$

Note that this stabilizer group was discussed in Ref. [46] as the group of local integrals of motion of the associated subsystem code Hamiltonian in one corner of its phase diagram. For a longer discussion on the connection of the XYZ ruby code to Bombin's subsystem code we refer to Sec. VII A. The code defined by this stabilizer group hosts four logical qubits per handle of the manifold on which we place the ruby lattice. To see this, we can count the number of independent generators of the stabilizer group and compare it to the Euler characteristic of the manifold. Alternatively, we can view the stabilizer group as obtained from a concatenation procedure. Specifically, the ZZ stabilizers define a three-qubit repetition code on each of the triangles. The effective qubits of the higher-level code sit on the vertices of the hexagonal lattice obtained from identifying each blue triangle with a qubit. On the effective qubits, the remaining stabilizers act like the stabilizers of the hexagonal color code [28], which encodes exactly four qubits per handle.

B. Reversible stabilizer groups and logical automorphism

In this section, we track the ISGs after having established four logical qubits. We will see that after $t = 0$, the number of logical qubits stays constant even though the ISGs change. To prove this, we construct logical operators, representing $\mathcal{N}(\mathcal{S}_t)/\mathcal{S}_t$, for any $t \geq 0$.

At $t = 0$, after the second ZZ measurement, we can use the logical string operators of the color code to construct representatives of the logical operator classes $\mathcal{N}(\mathcal{S}_0)/\mathcal{S}_0$. Specifically, the color code admits a basis of logical operators that can be associated to anyonic string operators (cf. Sec. II B). For each handle there are four independent pairs of string operators that act like pairs of Pauli- X and Z operators on distinct encoded qubits, see Fig. 4. For example, we can take a noncontractible string of X operators along red edges to represent a logical X operator. We can represent the corresponding anticommuting logical Z operator by an operator acting with Z operators on green edges (connecting green plaquettes) of the effective color-code lattice along a noncontractible loop around the

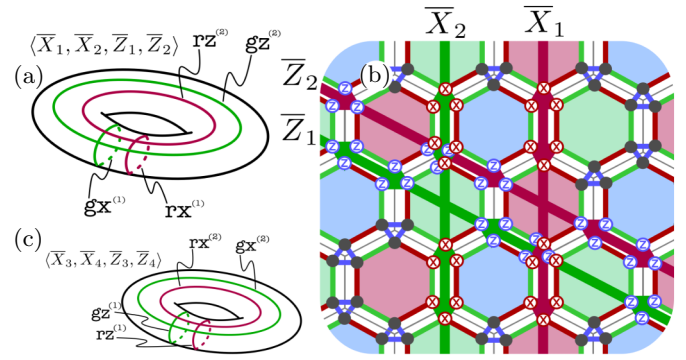


FIG. 4. When defined on a torus the XYZ ruby code encodes four logical qubits. In fact, the logical Pauli operators can be understood as color-code anyon string operators around homologically nontrivial cycles. On the torus, there are two inequivalent cycles, which we label with a superscript (1), respectively (2), in (a) and (c). In (a), we show one choice of generators for the logical Pauli operators on one pair of logical qubits. They are labeled by red and green X , respectively Z , bosonic string operators. The anyon exchange statistics ensures that they obey the necessary commutation relation as can be seen in (b) for a set of logical operators at $t = 0 \bmod 3$. We depict that a green Z string overlaps with one of the red X strings on the three qubits on a single blue triangle. The Pauli group on the other two logical qubits are generated by the same type of anyon string operators going around inequivalent cycles, see (c).

same handle but in the orthogonal direction. By construction, their support overlaps on an odd number of qubits and the two operators form an anticommuting pair of logical operators that square to one, i.e., generate a logical Pauli group. Exchanging red and blue in the construction above yields an additional pair of logical Pauli operators. Finally, we can exchange X and Z in both pairs of operators resulting in a total of four pairs of independent logical Pauli operators per handle of the ambient manifold. In fact, these are all of the logical Pauli operators for any color code on an orientable manifold (without boundaries). The physical representatives of the logicals of $\mathcal{S}_0$ are obtained by replacing each Pauli operator along the noncontractible loop with the logical Pauli operator of the three-qubit repetition code on the blue triangles. In Fig. 5, we show a subset of logical representatives, including the ones at $t = 0$.

At $t = 1$, we measure XX on every red edge. This updates the stabilizers to form another ISG,

$$S_1 = \left\langle \begin{array}{c} \text{Diagram of stabilizer group } S_1 \end{array} \right\rangle, \quad (19)$$

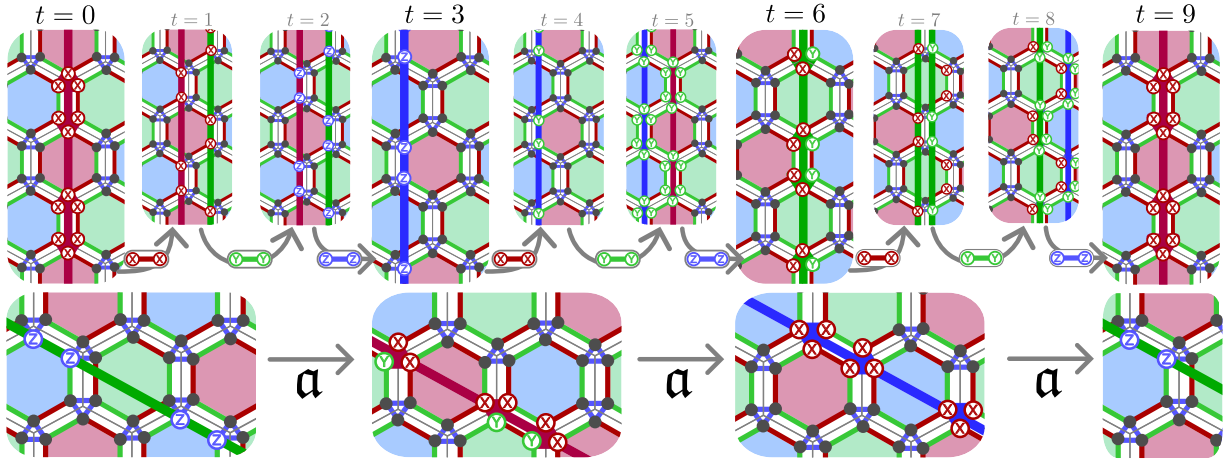


FIG. 5. The period-3 XYZ ruby code implements a logical $\mathbb{Z}_3$ automorphism α on the logical Pauli group. The figure shows the evolution of a pair of anticommuting logical operators throughout three cycles of the period-3 XYZ ruby code. At time t , for each logical operator we show a nontrivial representative that is shared among $\mathcal{N}(\mathcal{S}_t)$ and $\mathcal{N}(\mathcal{S}_{t+1})$. Note that to find a logical operator at $t = 2 \bmod 3$ that is preserved by the ZZ measurement the logical operator has to be multiplied by YY edges as well as by nonlocal Y strings in the stabilizer group $\mathcal{S}_2$. For the anticommuting logical operator we only show the representative at time steps $t = 0 \bmod 3$. Picking a basis of logical operators (in which the leftmost representatives are identified with $\bar{X}_1$ and $\bar{Z}_1$) for the times $t = 0 \bmod 3$ we can understand the action of the automorphism α on that pair of logical Pauli operators as $\bar{X}_1 \mapsto \bar{Z}_3 \bar{Z}_4 \mapsto \bar{X}_2 \bar{Z}_4 \mapsto \bar{X}_1$ and $\bar{Z}_1 \mapsto \bar{Z}_2 \bar{Z}_4 \mapsto \bar{X}_3 \bar{X}_4 \mapsto \bar{Z}_1$. Note that we show different representatives for the logical operators at $t = 0 \bmod 3$ compared to Fig. 4. How the logical operators transform from one time step to the next can easily be obtained from the logical flows associated to the logical Paulis, see, e.g., Fig. 10 for two exemplary logical flows.

Note that $\mathcal{S}_1$ needs nonlocal generators, which is not the case for $\mathcal{S}_0$. This is indicative for a phase transition between states in $\mathcal{S}_0$ and $\mathcal{S}_1$. We will later show that, indeed, $\mathcal{S}_1$ is in the same phase as three toric codes, see Sec. III C. $\mathcal{S}_1$ has a greater rank than $\mathcal{S}_{-2}$ (where we measured XX along red edges the last time) and the same rank as $\mathcal{S}_0$. When counting the number of independent generators to determine the rank one has to take into account global relations amongst the generators. For example, the product of all weight-6 X generators is the same as the product of all XX on red edges. As a consequence, the logical information encoded in the stabilized subspace of $\mathcal{S}_0$ is preserved in the subspace stabilized by $\mathcal{S}_1$. This can be shown by giving a representative logical Pauli operator that is shared among $\mathcal{N}(\mathcal{S}_0)$ and $\mathcal{N}(\mathcal{S}_1)$, as shown in Fig. 5 for one pair of logical operators. In fact, we can find such representatives by identifying corresponding logical operators of the lower-level repetition code on the blue triangles. All the logical X strings, for example, can be represented by $X^{\otimes 3}$ on each triangle, which commute trivially with the XX measurements. For the Z strings, we have to find logical Z representatives of the lower-level code that commute with the XX measurements. For that, note that all of the Z strings go along edges of a given color (in the higher-level color code). For each such edge, we can find exactly one red (XX) edge in the ruby lattice running perpendicular to it. It connects two physical qubits of two different triangles. The product of a Pauli Z on both of these qubits implements a logical $Z \otimes Z$ on the logical

qubits associated to the two triangles and commutes with the XX measurement along that red edge. We can combine these ZZ operators along a noncontractible path connecting plaquettes of the same color. This gives a set of logical operators for both $\mathcal{S}_0$ and $\mathcal{S}_1$. Together with the logicals composed of Pauli- X operators, they form a complete set of representatives of the logical Pauli group of both stabilizer groups. This shows that $\mathcal{S}_0$ and $\mathcal{S}_1$ form a *reversible pair* of stabilizer groups. Importantly, however, since $\mathcal{S}_1$ needs nonlocal generators and $\mathcal{S}_0$ does not, they do not form a *locally* reversible pair [47]. This manifests itself by the fact that one can view the ISGs as changing their topological phase during the measurement sequence. We will elaborate on this further in Sec. III C.

At $t = 2$, we measure YY along every green edge. This updates the stabilizer group to

$$\mathcal{S}_2 = \left\langle \begin{array}{c} \text{[Diagram of a green edge with a Y operator]} \\ \text{[Diagram of a hexagonal plaquette with Z operators]} \\ \text{[Diagram of a hexagonal plaquette with Z operators]} \\ \text{[Diagram of a hexagonal plaquette with Z operators]} \end{array} \right\rangle. \quad (20)$$

Again, we can find a logical representative shared amongst $\mathcal{N}(\mathcal{S}_1)$ and $\mathcal{N}(\mathcal{S}_2)$ for each logical coset. Starting from the representatives from before, inherited from the $t = 0$

round, we have to multiply XX stabilizers along the red edges to obtain the representatives that commute with the YY measurements. In Fig. 5 we show the transformation of a pair of anticommuting logical string operators throughout three measurement cycles. All other logical string operators are of the same form but act along paths connecting different colors or along a homologically inequivalent loop. Again, finding a logical representative shared among both $\mathcal{N}(\mathcal{S}_1)$ and $\mathcal{N}(\mathcal{S}_2)$ for each logical class shows that $\mathcal{S}_1$ and $\mathcal{S}_2$ form a reversible pair of stabilizer groups.

At $t = 3$, we measure ZZ again, which maps the ISG back to $\mathcal{S}_0$ and can find logical operators that are shared amongst $\mathcal{N}(\mathcal{S}_2)$ and $\mathcal{N}(\mathcal{S}_0)$ for each logical coset. This shows that the ISGs define a reversible sequence of stabilizer groups,

$$\begin{array}{ccc} & \mathcal{S}_0 & \\ m_0 \swarrow & & \searrow m_2 \\ \mathcal{S}_1 & \xrightarrow{m_1} & \mathcal{S}_2 \end{array} \quad (21)$$

After one period, the code gets mapped back to itself. As such, the measurement sequence defines a *logical automorphism* α_t on $\mathcal{N}(\mathcal{S}_t)/\mathcal{S}_t$ for each t , captured by the following diagram:

$$\begin{array}{ccc} & \alpha_t & \\ & \curvearrowright & \\ & \mathcal{N}(\mathcal{S}_t)/\mathcal{S}_t & \\ m_t \swarrow & & \searrow m_{t+2} \\ \mathcal{N}(\mathcal{S}_{t+1})/\mathcal{S}_{t+1} & \xrightarrow{m_{t+1}} & \mathcal{N}(\mathcal{S}_{t+2})/\mathcal{S}_{t+2} \\ \alpha_{t+1} \curvearrowleft & & \curvearrowright \alpha_{t+2} \end{array} \quad (22)$$

Each α_t itself is a Clifford unitary on the respective codespace that is implemented by a full period of the measurement cycle. To evaluate the logical automorphism we can pick any reference ISG and track how the logical operators evolve through a cycle of measurements. Let us pick $\mathcal{S}_0$, which describes a topological code with an anyon model equivalent to that of the two-dimensional color code. A single cycle of measurements implements an automorphism on the logical Pauli group of that code, which can equivalently be understood as an automorphism of its anyon model. Reference [41] classified all the automorphisms of the color-code anyon model. In particular, any automorphism is uniquely defined by the action it has on the color-code boson table, see Eq. (9). Tracking the logical string operators through one cycle of measurements we find that the XYZ ruby code implements a shift along the diagonal of that table. In particular, the automorphism has order 3, which can be seen in Fig. 5, where

we show the evolution of one pair of logical string operators through three cycles of measurements. Specifically, the figure shows the transformation

$$rx \rightarrow bz \rightarrow gy \rightarrow rx \rightarrow \dots \quad \text{and} \quad (23a)$$

$$gz \rightarrow ry \rightarrow bx \rightarrow gz \rightarrow \dots \quad (23b)$$

Graphically, we depict the automorphism applied on the color-code anyons within one period with

$$\begin{array}{c} \text{Diagram of color-code anyons (red, green, blue lines) with arrows indicating a cyclic shift.} \\ = \varphi \in \text{Aut}(A_{CC}). \end{array} \quad (24)$$

On a subset of logical Pauli operators it acts like

$$\bar{X}_1 \mapsto \bar{Z}_3 \bar{Z}_4 \mapsto \bar{X}_2 \bar{Z}_4 \mapsto \bar{X}_1 \mapsto \dots \quad \text{and} \quad (25a)$$

$$\bar{Z}_1 \mapsto \bar{Z}_2 \bar{X}_4 \mapsto \bar{X}_3 \bar{X}_4 \mapsto \bar{Z}_1 \mapsto \dots, \quad (25b)$$

as we show explicitly in Fig. 5. The explicit labeling of the above logical operators is an arbitrary choice, of course. However, the automorphism necessarily preserves the commutation relation amongst any pair of logicals, independent of the choice of basis. In a different basis, for example, we would find that some logical operators are “static” operators in the protocol. Consider a logical operator at $t = 0 \pmod 3$ that is associated to moving one of the fermions f_1 , f_2 , or f_3 as defined in Eq. (11) around a homologically nontrivial cycle. The fermions are products of bosons in the table above that lie within the same orbit of the automorphism. In fact, the automorphism simply exchanges the two constituent bosons and the composite (fermion) is left invariant by the automorphism. Indeed, we find that the associated logical operators are shared by all of the ISGs. We discuss this in the context of the associated subsystem code further in Sec. VII A. We note that an isomorphic automorphism was found in other schedules on the same set of checks in Ref. [22].

C. Instantaneous topological phase transitions

We have seen that the ISG $\mathcal{S}_0$ is in the same phase as the two-dimensional color code, which helped us significantly in finding the logical operators and their transformations throughout the measurement sequence of the XYZ ruby code. In this section, we will analyze the other two ISGs, $\mathcal{S}_1$ and $\mathcal{S}_2$, with respect to their topological properties. In particular, we consider the locally generated part of the ISGs as defining the topological phase of the stabilizer group and interpret the nonlocal stabilizers as additional constraints on the codespace of the topological code defined by the locally generated one. Interestingly, we

find that the code undergoes an *instantaneous phase transition* [48] from the color-code phase into three copies of the toric-code phase. A similar behavior was observed in Ref. [23], where the ISGs undergo a transition from an X -cube model to layers of the $2 + 1$ -dimensional toric code. To the best of our knowledge, the XYZ ruby code is the first two-dimensional code appearing in the literature that features such an instantaneous phase transition.

Similar to $\mathcal{S}_0$, we can view $\mathcal{S}_1$ and $\mathcal{S}_2$ as stabilizer groups of a concatenated code, where the XX , respectively YY , checks are stabilizers of a lower-level $[[2, 1, 1]]$ code. Viewed in this way, the other stabilizer generators act as higher-level stabilizers as a product of logical operators on these smaller codes. For both $\mathcal{S}_1$ and $\mathcal{S}_2$ the higher-level code is generated by weight-3 and weight-6 stabilizers, see Fig. 6. Additionally, we find that the local part of the stabilizer groups can be decomposed into three factors, each of which acts on a different subset of qubits. The three subsets can be associated to three colors, red, green, and blue, following the colorings of the edges of the underlying

hexagonal lattice, cf. Fig. 2. On each subset, the stabilizer group reduces to the stabilizer group of a toric code on a triangular lattice (see Fig. 6). Since there are no additional local stabilizers, we conclude that the topological order of states stabilized by $\mathcal{S}_1$ and $\mathcal{S}_2$ is that of three copies of the toric code. Specifically, the code undergoes an instantaneous phase transition from the color code to three copies of the toric code when the XX or YY checks are measured on a state stabilized by $\mathcal{S}_0$. Similarly, the reverse transition is induced by measuring the ZZ checks on $\mathcal{S}_1$ or $\mathcal{S}_2$. The additional nonlocal stabilizers can be identified with anyon string operators along nontrivial cycles and implement a nonlocal constraint on the codespace. In fact, the string operators act nontrivially on all three copies of effective toric codes and can be viewed as moving the electric charges (e anyons) of all three copies jointly around a nontrivial cycle. As such, the nonlocal stabilizers can be viewed as logical operators of the three toric codes and it being in the stabilizer group enforces a parity constraint on their logical qubits [49]. This explains how the logical dimension remains constant even though the number of anyons in the code changes periodically in time.

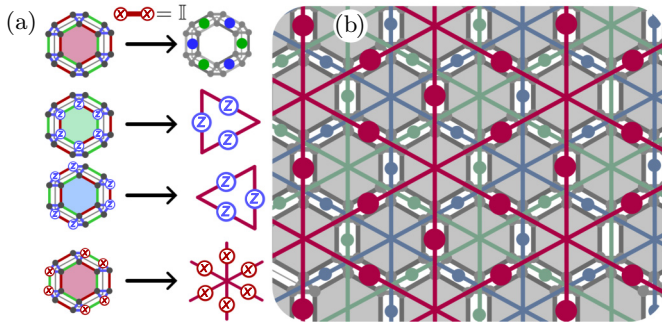


FIG. 6. At $t = 1 \bmod 3$ the ISG (after establishing) can be mapped to the one of three copies of toric codes on triangular lattices by a local isometry. (a) This can be seen explicitly by interpreting the XX stabilizers on the red edges as stabilizers of a lower-level $[[2, 1, 1]]$ repetition code. This defines an effective qubit on each red edge of the ruby lattice, which can also be viewed as associating one qubit to each edge of the underlying honeycomb lattice. The other stabilizer generators of $\mathcal{S}_1$ are mapped onto weight-3 Z and weight-6 X stabilizers on the effective qubits. The stabilizer generators can be grouped to act on three nonoverlapping subsets of qubits: the ones associated to the red, green, and blue edges of the underlying honeycomb lattice. We exemplify how the stabilizers act on the effective qubits by showing generators of $\mathcal{S}_1$ that effectively act only on the qubits associated to red edges. (b) On each of the three subsets, the effective stabilizers reduce to the ones of a toric code on a triangular lattice, cf. Fig. 1. We draw the lattices for the effective toric codes overlapping in three different colors. For better readability we highlight the lattice on which the effective red toric code is defined. The lattices defining the other two toric codes differ only by translation, with vertices placed in the center of plaquettes of a different color. The same mapping can be performed for $\mathcal{S}_2$, where the basis of the lower-level stabilizers changes from X to Y .

D. Rewinding schedule

In this section, we explain a *rewinding* schedule based on the schedule discussed so far in Sec. III. The idea of a rewinding schedule was originally introduced in Ref. [50] to define boundaries for the honeycomb code, a Floquet code that applies a nontrivial automorphism after one length-3 period. Reference [22] formalized this concept and applied it to other Floquet codes in two and three spatial dimensions to engineer transitions between desired ISGs. A rewinding schedule can in general help to construct boundaries, logical gates or other desired features of the ISG.

A Floquet code is considered *rewinding* if the measurement sequence $\mathcal{M}$ [see Eq. (2)] is symmetric with respect to time reversal, i.e., is of the form

$$[\mathcal{M}_0, \mathcal{M}_1, \dots, \mathcal{M}_{k-1}, \mathcal{M}_{k-1}, \dots, \mathcal{M}_1, \mathcal{M}_0] \quad (26)$$

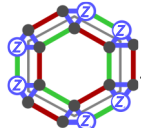
for some $k \in \mathbb{Z}_{>0}$ [22]. Note that in this definition both $\mathcal{M}_{k-1}$ and $\mathcal{M}_0$ are performed twice in a row. In either case, the second of the two measurements will not have any effect on the ISG or logical operators at that time, besides potentially changing signs of some stabilizers if an error occurs. Hence, we also refer to a schedule in which these measurements are not repeated as *rewinding*.

For the XYZ ruby code we analyze a rewinding schedule defined by the period-6 measurement sequence

$$\mathcal{M}_r = [\text{blue } Z, \text{red } X, \text{green } Y, \text{blue } Z, \text{green } Y, \text{red } X]. \quad (27)$$

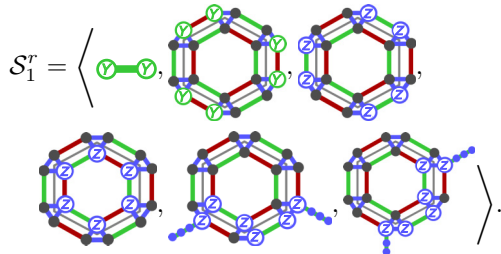
Note that in this rewinding schedule there are two types of Z measurements: One sandwiched by XX measurements and one sandwiched by YY measurements. We pick this rewinding schedule since the second ZZ measurement is needed to establish an ISG in the color-code phase, cf. Sec. III A.

In the following, we describe the ISGs in the rewinding schedule. Again, consider starting with the first round of ZZ measurements at $t = -3$. Since the measurement rounds at $t = -3, -2, -1$, and 0 agree with the nonrewinding schedule [see Eq. (12)], we find that the rewinding schedule also establishes four logical qubits in ISG $\mathcal{S}_0$ just after the second round of ZZ measurements. At $t = 1$, YY is measured along the green edges, which drives the system into a new ISG, which we denote by $\mathcal{S}_1^r$. First note that the Y stabilizers all remain in the stabilizer group and so do all the $Z^{\otimes 6}$ stabilizers around the inner plaquettes [see Eq. (18)]. Although the individual ZZ stabilizers anticommute with the YY measurement operators, certain combinations commute with them. Specifically, ZZ operators acting on qubits connected along a loop of green and blue edges remain in the stabilizer group after the YY measurement. These loops naturally fall into homology classes, so we sort the stabilizers associated to these loops into the same classes. The homologically trivial stabilizers are all generated from weight-6 operators forming a loop around a single plaquette,



$$(28)$$

Additionally, there is a stabilizer of extensive weight for each homologically nontrivial loop. We hence add one generator for each homology class obtained from a product of ZZ operators along a representative loop along green and blue edges. Taken together, the system is driven into a code stabilized by

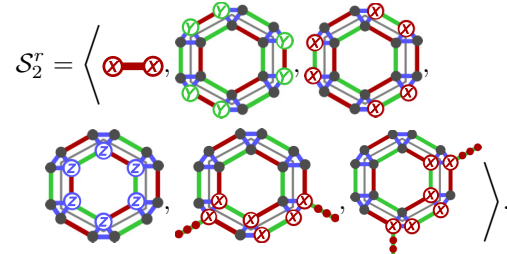


$$(29)$$

Note that $\mathcal{S}_1^r$ is related to $\mathcal{S}_1$ [Eq. 19] by a lattice rotation of $\pi/3$ together with a product of on-site unitary $(X + Y)/\sqrt{2}$, swapping X and Y bases.

At $t = 2$, XX operators along red edges are measured. With similar arguments as for the transition $\mathcal{S}_1 \rightarrow \mathcal{S}_2$ in the

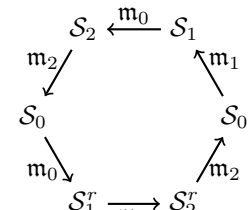
nonrewinding schedule we see that the XX measurement at $t = 2$ in the rewinding schedule drives the system into a code stabilized by



$$(30)$$

In particular, the $Y^{\otimes 6}$ generators and the $Z^{\otimes 6}$ generators acting on the inner plaquettes remain in the stabilizer group and the ZZ loops along green and blue edges have to be multiplied with YY stabilizers to commute with the XX measurement operators. We again find that rotating the lattice by $\pi/3$ and applying $(X + Y)/\sqrt{2}$ to every qubit maps $\mathcal{S}_2^r$ to $\mathcal{S}_2$ [Eq. (20)].

At $t = 3$, the ZZ measurements along the blue edges are repeated and we find that the system is driven back to a state stabilized by $\mathcal{S}_0$. From there, XX operators along red edges are measured, followed by YY measurements along the green edges, performing the transformation of the nonrewinding schedule until $t = 6 = 0 \pmod{6}$. To conclude, after establishing that the stabilizer groups undergo a periodic transformation,



$$(31)$$

we see that we encounter a length-6 rewinding schedule. Note that the same measurements are performed as in the period-3 schedule just in a different order, see Eq. (21). Since all measurements in the above diagrams are reversible we can infer from the mirror symmetry of the above diagram that the sequence implements a trivial automorphism on the encoded logical Pauli algebra [22,35]. We can track representatives of the logical operators through a full period of measurements in the rewinding schedule and verify that the cosets of logical operators are mapped back onto themselves, i.e., the rewinding schedule implements a trivial automorphism on the logical Pauli group. The explicit transformation of logical operators can be deduced from the logical flow as we will see later.

We find that after three time steps the code is also mapped back onto itself and the logical operators undergo the nontrivial automorphism of the period-3 schedule. Due

to the rewinding, the second half of the period implements the inverse of that automorphism rendering the action of the full period of measurements on the logical Pauli group trivial. For the rest of this work, we will investigate the XYZ ruby code with a rewinding schedule.

IV. THREE-COLORED GRAPHICAL CALCULUS FOR QEC

In this section, we introduce the concept of a three-colored graphical calculus for quantum error-correcting protocols defined by circuits composed of Pauli measurements and Clifford operations. In the following we write “circuit” for short since we only consider these types of circuits for now. In principle, one can analyze any such protocol with the usual ZX calculus, as shown for example in Refs. [26,31]. For protocols involving Y measurements, however, the formalism becomes less elegant and the mapping between a ZX diagram and circuit substantially more tedious. It is desirable to work with a representation that is closer to the “native” operations considered in an QEC protocol.

In the ZX calculus the elementary building blocks are two types of tensors, also called *spiders*, usually represented by circles of two different colors. We propose an extension to the graphical representation of circuits with spiders of a third color. Since tensors of three different colors—red, green, and blue—will play a central role, we call such a tensor network an *RGB tensor network* (RGB TN). Additionally, we extend the concept of *Pauli flow* [51] to that setting and rigorously define all algebraic quantities entering a quantum error-correction analysis from these flows alone. This gives rise to a purely graphical description of all the quantities needed to perform error correction on a circuit.

We would like to note that there have been other proposals to add a third color to the ZX calculus, specifically Refs. [52,53]. From our perspective, both approaches lack some properties that we find desirable for a representation of circuits, independent of their geometric structure. Reference [52] resorts to a two-dimensional ambient space when defining the spiders. In particular, the formalism only allows for left- and right-pointing legs attached to individual tensors. In light of the fact that we want to consider spacetimes of higher dimensions than two and we can interpret the time direction freely in these protocols [17], we want to avoid such choices. This is resolved in Ref. [53] with a graphical calculus that is defined independently of its ambient space but adds orientations to the edges. The authors’ goal was to obtain a calculus that is fully symmetric among the three colors. In order to achieve this they have to change one of the two tensors of the ZX calculus. In our context this is highly undesirable since simple circuit elements, such as the CNOT gate, are nicely expressed

in terms of original ZX tensors. We aim to fix both problems with minor additions to the Clifford ZX calculus by only adding a third type of tensor. To avoid the dependency on some ambient space, we will allow for edges attaching to that third tensor to be “flipped” without the need of adding orientations to all edges or tensors.

A. Building blocks

The idea behind the graphical calculus is to represent the circuit as a tensor network, which can be assembled using simple elementary building blocks. The goal of this section is to first introduce the tensors from which we build up the tensor network representing any circuit. All tensors will be considered over $\mathbb{C}^2$, i.e., any “wire” will be associated with a two-dimensional complex vector space. As such, it carries a computational basis, which we write as $\{|0\rangle, |1\rangle\}$ and identify it with the two orthogonal eigenstates of the Pauli-Z matrix.

The key building blocks will be three types of tensors. To each type, we associate a color: blue, red, or green. For each color, we define two tensors, labeled by $s \in \{0, 1\}$. In the following, we define each type of tensor in terms of its matrix elements in a computational basis. We define a *blue s spider* by

$$\begin{aligned} \text{Blue } s \text{ spider} &= \begin{cases} (-1)^{sa} & a = b = c = \dots = d \\ 0 & \text{else} \end{cases} \\ &= (-1)^{as} \delta_{a,b,c,\dots,d}, \end{aligned} \quad (32)$$

for any number of legs. This tensor implements a constraint on the vector space spanned by all the labelings of the legs, indicated by the three dots next to the tensor. Specifically, for $s = 0$ it is the usual δ tensor enforcing all labels to have the same value. Similarly, we define a *red s spider* by

$$\begin{aligned} \text{Red } s \text{ spider} &= \begin{cases} 1 & a + b + c + \dots + d = s \pmod{2} \\ 0 & \text{else} \end{cases} \\ &= \delta_{a \oplus b \oplus c \oplus \dots \oplus d, s}. \end{aligned} \quad (33)$$

It represents a constraint on the joint parity of all the legs of the tensor. Specifically, for $s = 0$ it projects onto the even sector whereas for $s = 1$ it projects onto the odd sector. Lastly, we introduce a third type of tensor, namely a *green s spider*, a complex tensor defined by

$$\text{Green } s \text{ spider} = i^{a+b+\dots+c} \delta_{a \oplus b \oplus c \dots \oplus d, s}. \quad (34)$$

The green tensor implements a similar constraint as the red tensor but additionally adds a phase generated by i ,

depending on the explicit configuration on the input legs of the tensor. For later convenience, we also want to capture tensors that act with a $-i$ on some legs. Hence, we introduce “flipping” of inputs of green tensors, which we denote with a black arrowtip on a leg attached to the tensor. Specifically, we define

$$\text{green spider with } a \text{ leg flipped} = i^{-a+b+c+\dots+d} \delta_{a \oplus b \oplus c \dots \oplus d, s} \quad (35)$$

to denote the tensor with the a leg flipped. Note that flipping all edges is the same as complex conjugating the tensor,

$$\text{green spider} = \text{green spider with all legs flipped}^* \quad (36)$$

For the rest of this work, if there is no explicit s label given, we consider the spider to be a $s = 0$ spider. Sometimes, we will refer to the $s = 1$ spider as a *signed* spider.

Lastly we note that the three spiders are related by basis transformations defined by the two-legged tensors

$$a \text{ --- } \text{brown rectangle} \text{ --- } b = \frac{1}{\sqrt{2}} (-1)^{ab}, \quad (37a)$$

$$a \text{ --- } \text{brown oval} \text{ --- } b = \delta_{a,b} i^a, \quad (37b)$$

$$a \text{ --- } \text{brown diamond} \text{ --- } b = \frac{1}{\sqrt{2}} i^{(1+2a)b}. \quad (37c)$$

For example, the blue spider can be obtained by conjugating either a red or green spider by the above tensors,

$$\text{blue spider} = \text{brown rectangle} \text{ --- } \text{red spider} = \text{brown rectangle} \text{ --- } \text{green spider} \quad (38)$$

In Appendix A, we give a more complete overview on RGB tensor networks and equivalences thereof. In particular, we show that relations amongst different tensor networks give rise to a “calculus” based on these networks, similar to the ZX calculus.

B. Graphical representation of circuit elements

In the following, we show how a generic Clifford circuit with Pauli measurements can be mapped to a tensor network of tensors that were just introduced.

One can think of each of the three colors as corresponding to one of the three Pauli labels, with red corresponding to X , green to Y , and blue to Z [54]. Specifically, their two-legged signed versions represent the Pauli matrices

$$\text{red spider with 1 leg flipped} = \text{--- } [X] \text{ ---}, \quad (39a)$$

$$\text{green spider with 1 leg flipped} = \text{--- } [Y] \text{ ---}, \quad (39b)$$

$$\text{blue spider with 1 leg flipped} = \text{--- } [Z] \text{ ---}. \quad (39c)$$

Together with usual Hadamard and S gate, represented by the tensors

$$\text{brown rectangle} = \text{--- } [H] \text{ ---}, \quad (40a)$$

$$\text{brown oval} = \text{--- } [S] \text{ ---}, \quad (40b)$$

the signed blue, green, and red spiders generate the single-qubit Clifford group. To complete the full Clifford group, we need a two-qubit gate, for example, the CNOT gate. The CNOT gate can be defined as a matrix with entries

$$\text{CNOT}_{ii',jj'} = \delta_{i,i'} \delta_{j',j \oplus i}, \quad (41)$$

which can be represented by the tensor network

$$\text{CNOT tensor network diagram} \quad (42)$$

This completes the representation of any Clifford unitary in the three-colored calculus.

Additionally, we can represent any multiqubit Pauli measurement nicely as an RGB tensor network. More concretely, we represent *projective measurements* in terms of the projectors onto a given measurement outcome. As such, the measurement outcome appears as a classical label of the network.

Let us start with a simple example of a two-body measurements. They constitute the elementary operations of the XYZ ruby code, defined in Sec. III. We represent the projector onto outcome $m \in \{0, 1\}$ of a $Z \otimes Z$ measurement via the following tensor network:

$$\text{Z} \otimes \text{Z} \text{ measurement tensor network} = \text{CNOT tensor network with outcome } m \quad (43)$$

Similarly, an XX measurement can be represented by swapping blue with red in the above diagram.

We can use only blue and red spiders in combination with Hadamard boxes to also include Y measurements, see, for example, Fig. 3 in Ref. [31]. However, this comes with a fairly big notational overhead where the components in the tensor network do not resemble the actual building blocks of a circuit implementing that measurement. Using the green spiders, we can represent measurements involving Pauli Y more directly. For example, consider a YY measurement. We find that

$$(44)$$

represents the projector onto the subspace associated to the YY measurement with outcome $m \in \{0, 1\}$. Note that the sign of the blue spider on the right represents the flipped measurement. This is an artefact on the details of how we defined the green tensor. In most cases we do not need to consider the flipped measurement explicitly since the respective stabilizer groups only differ by a sign.

In fact, we can represent any multiqubit Pauli measurement as a tensor network following a recipe presented in Appendix A.

C. Pauli flow and logical isomorphism

In this section, we introduce *Pauli flows* in terms of projective symmetries of the building blocks of our tensor network representation of circuits. We use the term “projective symmetry” because the operations we want to consider leave the individual tensors invariant up to a phase. Specifically, we obtain a graphical rule on how to track the propagation of Pauli operators through the network.

Let us start with a simple example of the identity tensor represented by a naked wire. The Pauli matrices behave in the following way:

$$(45)$$

We find that, up to phases, they form an Abelian group $\hat{\mathcal{P}}_1 \simeq \mathbb{Z}_2 \times \mathbb{Z}_2$. Since these spiders can be freely moved around a wire and commute up to phases, we only need to keep track of which Pauli acts, not in which order and not which phase factor it carries. We represent this by highlighting the wire in one of three colors,

$$(46a)$$

$$(46b)$$

$$(46c)$$

Formally, the map depicted by “ $\mapsto$ ” can be understood as the map mapping the single-qubit Pauli group to $\mathbb{Z}_2 \times \mathbb{Z}_2$, whose three nontrivial elements we depict with three different colors. The commutative addition in $\mathbb{Z}_2 \times \mathbb{Z}_2$ is represented by the following rules, when “adding” two highlights,

$$(47a)$$

$$(47b)$$

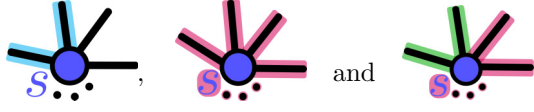
In the following, we derive the projective Pauli symmetries of the blue, green, and red spider and represent them graphically using the highlights introduced above. The blue spider obeys the following relations:

$$(48a)$$

$$(48b)$$

that can be understood as its projective Pauli symmetries. Every symmetry operation defines a valid highlight of the legs around a tensor, which we call *Pauli flows* of that tensor. Specifically, we highlight a leg in the color of the tensor with which the symmetry acts on that leg. For a more stringent mathematical treatment we also consider the trivial symmetry operation, acting with an identity tensor on each leg, as a trivial Pauli flow, where no leg is highlighted. Moreover, we see in Eq. (48) that the blue tensor acquires a sign when applying a signed red tensor on each of its legs if and only if $s = 1$. We indicate that by highlighting the (classical) s label of the tensor and say that the tensor is *charged* with respect to that flow. Together, the set of projective symmetries of the blue tensor, Eq. (48), defines the following Pauli flows of the blue

tensor:


(49)

We find that the tensors of the other colors have similar projective symmetries leading to these Pauli flows:


(50a)

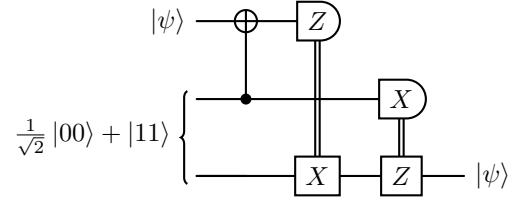

(50b)

It is helpful to think of all allowed Pauli flows around a tensor as being generated by generating flows of two types. The first type highlights an even number of legs in the same color as the tensor. The second type highlights all legs in one of the two complementary colors, as well as the s label. Adding these highlights gives rise to all Pauli flows of an individual tensor. Note that single-legged tensors play a special role, since they do not have pairs of legs. As a result, they only allow for the second type of nontrivial Pauli flow, highlighting the leg and the sign in one complementary color.

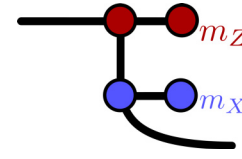
From the flows of a single tensor we can construct the flows of any tensor network composed of these tensors. We define a Pauli flow of a network as a highlight of legs in the network that is a Pauli flow—as defined above—when restricted to any of the constituents of the network. The notion of *charge* carries over directly to a set of tensors: Consider a set of (signed) RGB tensors T in an RGB network [55] and a given flow F . F highlights a subset of signs in the network, which we denote by S . The charge of T is defined as the binary sum of highlighted signs of the tensors in T , $c_T = \sum_{s \in S \cap T} s \pmod{2}$.

Reference [31] introduces a similar concept called *Pauli webs*. There, the authors work with a two-colored (ZX) calculus and two-colored flows. Implicitly, they include a third highlight as well by allowing a combination of the two. We take a more direct approach by explicitly working with a third color and adding the third type of tensor. This makes the flow diagrams more symmetric and easier to work with. Additionally, we introduce the rigorous notion of tensors being *charged* with respect to a flow, which—to the best of our knowledge—is a concept newly introduced in our work. It will be central to our perspective on spacetime quantum error correction (see Sec. IV D). For a rigorous treatment of Pauli flow, we refer to Appendix A 3.

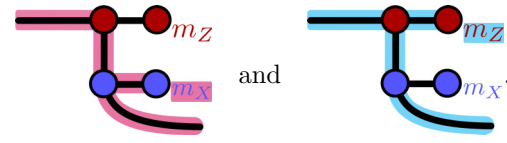
By mapping a given circuit to an RGB tensor network, we can understand its logical action in terms of Pauli flows. Let us illustrate this with a simple example: a single-qubit teleportation circuit is


(51)

In our tensor-network notation, omitting the adaptive Pauli correction, it can be represented by the diagram [56]


(52)

where the measurement outcomes m_Z and m_X are represented by signs of two of the tensors. The flows describe how Pauli operators propagate through the circuit (up to global phases). In particular, there are flows where the input and output wire of the circuit is highlighted. Specifically, the network has two independent flows


(53)

These flows indicate that the circuit maps

$$X_1 \mapsto (-1)^{m_X} X_3 \quad \text{and} \quad Z_1 \mapsto (-1)^{m_Z} Z_3, \quad (54)$$

which fully determines its action since it is a Clifford circuit that acts unitarily on the information encoded on the input qubit. Moreover, we recover that the circuit acts like $X^{m_Z} Z^{m_X}$ on the logical state, determining the adaptive correction applied in Eq. (51).

D. Error correction in spacetime from Pauli flows

In the previous paragraph we have sketched how the Pauli flows of the teleportation circuit (defined for three physical qubits) give rise to an effective logical (Clifford) circuit on a single qubit. The concept of a logical isomorphism is central to our analysis of dynamical error-correcting schemes. A circuit composed of Clifford operators and Pauli measurements implements an isomorphism from an input logical Pauli group to an output logical Pauli group. We call this isomorphism the *logical isomorphism*. Both the input logical Pauli group and the output logical

Pauli group will be defined in terms of an input stabilizer group $\mathcal{S}_{\text{in}}$ and an output stabilizer group $\mathcal{S}_{\text{out}}$.

In this subsection, we show how the Pauli flows of (a tensor-network representation of) a given circuit can be used to understand the logical isomorphism implemented by the circuit. In particular, we will rigorously define the input and output *stabilizer group*, the *logicals*, and *detectors* by their associated Pauli flows. When introducing errors, the notion of a *charged* tensor will be important to understand the effect of an (Pauli) fault on the syndrome and encoded logical information. Later in this work, we will use exactly these methods to benchmark the logical isomorphism implemented by the XYZ *ruby code*, see Sec. V.

Let us consider a circuit from n qubits to m qubits expressed as an RGB tensor network together with all its Pauli flows. The set of all flows forms a group, which we denote by F . Its identity is given by the trivial flow, not highlighting any edge in the network (nontrivially). We distinguish between three types of Pauli flows, forming subsets of F : *detector flows* D , *stabilizer flows* S , and *logical flows* L . In the following, we introduce these three types and explain how the Pauli flow gives rise to quantities entering the error-correction procedure and illustrate these quantities in a tensor network representing repeated measurements of the stabilizers of a repetition code. We will see that the notion of tensors being charged with respect to a given flow plays a central role. For a rigorous definition and treatment of the types of flows and associated quantities in the QEC protocol, we refer to Appendix A 3.

- (1) **Detector flows:** Pauli flows that only highlight internal legs nontrivially are called *detector flows*. In fact, the set of detector flows form a group D under adding the flows according to Eq. (47) and can be considered as a *symmetry group* of the network [57]. We will later use conserved quantities related to that symmetry for decoding.
- (2) **Stabilizer flows:** Pauli flows that can be generated by flows that may highlight either input or output legs nontrivially are called *stabilizer flows*. We denote the set of all such flows by S . From S , we can in fact derive a stabilizer group

$$\mathcal{S}_{\text{in}} \times \mathcal{S}_{\text{out}} \leq \mathcal{P}_{n+m} \quad (55)$$

that factors over input and output Pauli groups. Given a stabilizer flow s , the associated Pauli word is obtained by taking the highlight of s restricted on the input and output legs and interpreting it as a Pauli operator that is the product of the Pauli operators associated to each individual highlight on each open leg. We identify a trivial highlight with $\mathbb{1}$, a red highlight with X , a green highlight with Y , and a

blue highlight with Z . The sign in front of the Pauli word associated to flow s is obtained from the signs charged with respect to s .

- In Sec. V, we consider detectors in the bulk of an RGB tensor network and imagine cutting this network along a set of edges. In fact, all the detector flows that had nontrivial highlights on the cut edges will be promoted to nontrivial stabilizer flows of the cut network. This perspective helps us to understand slices of detector flows as instantaneous stabilizers.
- (3) **Logical flows:** A Pauli flow that highlights both input and output legs is called a *logical flow*. It captures how the network transforms (logical) Pauli operators on the input legs into logical Pauli operators on the output legs. Specifically, every logical flow can be associated with a Pauli operator supported on both input and output legs in the same way as discussed for the stabilizer flows. Importantly, every Pauli word will carry a sign given by the tensors charged with respect to the corresponding logical flow. The set of Pauli words obtained in this way again form a stabilizer group

$$\mathcal{S}_{\ell} \leq \mathcal{P}_{n+m} \quad (56)$$

that commutes with both $\mathcal{S}_{\text{in}}$ and $\mathcal{S}_{\text{out}}$. Importantly, this group has no element that is only supported on input or output legs that is not contained in $\mathcal{S}_{\text{in}}$ and $\mathcal{S}_{\text{out}}$. As such, it defines a logically entangled state between the codes of $\mathcal{S}_{\text{in}}$ and $\mathcal{S}_{\text{out}}$ and since it is also a Pauli group, it can be identified with a Clifford unitary by identifying one part of the entangled state with its dual and thereby interpreting a stabilizer state on a joint system as a linear map between the two systems. For details on this relation, we refer to Appendix A 3 b.

As with stabilizer flows, adding a detector flow does not change the Pauli word on the input and output legs assigned to a logical flow. In contrast, adding a stabilizer flow does change the Pauli word assigned to the logical flow. However, it will not change the logical coset of the Pauli word with respect to $\mathcal{S}_{\text{in}} \times \mathcal{S}_{\text{out}}$. Hence, we call two logical flows equivalent if and only if they differ by a detector or a stabilizer flow. One can check that this defines a valid equivalence relation, using the Abelian group structure of the Pauli flows of the network and the equivalence classes determine the logical isomorphism applied by the linear operator represented by that network.

Figure 7 summarizes the different types of Pauli flows and the associated algebraic structures. We give a more rigorous construction of all the stabilizer groups defined from Pauli flows in Appendix A 3 and prove that the stabilizer

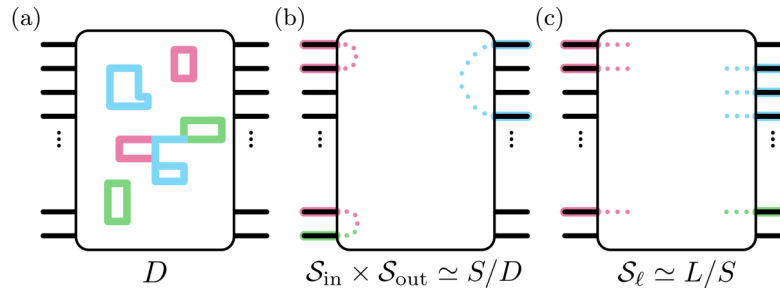
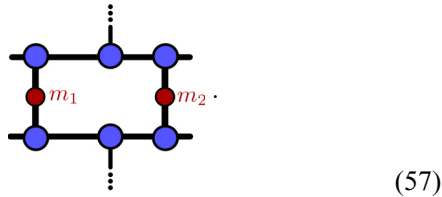


FIG. 7. Schematic depiction of the three different types of Pauli flows, defined by how they highlight the input and output legs. (a) Detector flows are purely internal and give rise to a symmetry group of the RGB tensor network. It determines the classical data that can be used for decoding, see Sec. V A and Appendix A 3. (b) Flows that are generated by flows that highlight only input or output legs nontrivially can be mapped to input and output stabilizer groups. (c) Flows highlighting both input and output legs give rise to a stabilizer group supported on both input and output qubits and together and can define the logical Clifford applied on the logical operators of the input and output logical stabilizer group.

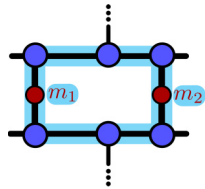
group obtained from Pauli flows on the input and output legs of the network fully describe the logical encodings on either side and the explicit isomorphism applied by the network. We want to remark that the main technical result presented in Appendix A 3 b applies more generally to any Clifford operations and their composition.

Let us illustrate the concepts introduced above via a guiding example based on a circuit implementing repeated measurements of the stabilizers of a repetition code. Consider n qubits on a line stabilized by the stabilizer group $\langle \{Z_i Z_j\}_{(i,j)} \rangle$, generated by $Z \otimes Z$ on neighboring qubits. The following tensor network represents two consecutive measurements of a stabilizer generator



(57)

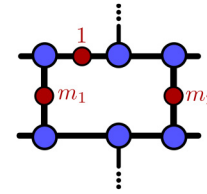
Time is understood to go from left to right and the dots indicate the rest of the circuit, performing the same measurements on other pairs. The labels on the red tensors correspond to the measurement outcomes of the two-body $Z \otimes Z$ measurements. The detector flows of this network are generated by internal loops of blue flows,



(58)

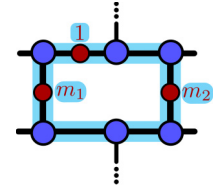
with respect to which two red tensors are charged. This tells us that without noise the measurement outcomes fulfill $m_1 + m_2 = 0 \pmod{2}$. This information can be used

for decoding. Consider a single-qubit X fault happening in between the two measurements. The tensor network including the fault reads



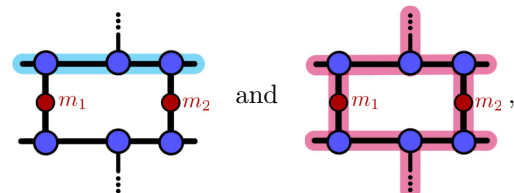
(59)

and inherits its detector flows from the fault-free network,



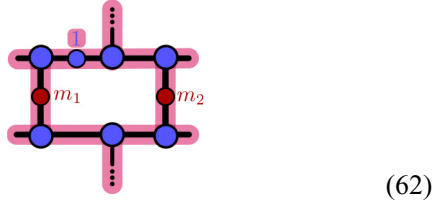
(60)

We see that the fault is charged with respect to the detector flow and the constraint now reads $m_1 + m_2 = 1 \pmod{2}$. As such, we can detect this single-qubit Pauli error by calculating the parity of the measurements charged with respect to a detector flow. Note that this network only has blue detector flows. Hence, Z faults, represented by blue tensors, can never be charged with respect to any detector flow in this network telling us that they are all undetectable. To assess the logical effect of an undetectable error, we have to consider the logical flows. In this simple example there are two independent logical flows, e.g., represented by

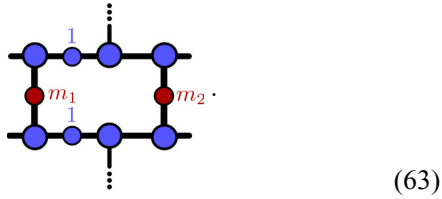


(61)

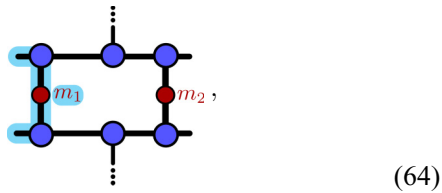
showing that a single logical qubit is preserved through the circuit. Note that when restricted to either input or output legs the highlights of the logical flow give rise to an anticommuting pair of logical operators. We find that the undetectable Z fault is charged with respect to the red flow,



and hence corresponds to a nontrivial logical error, i.e., has a *consequence* on an encoded state [58]. Note that this is not the case for every undetectable fault. For example, consider the two-qubit Z fault,



It is not charged with respect to any detector flow, i.e., is undetectable. At the same time, it is not charged with respect to any logical flow (since the charge is defined mod 2) and hence has trivial action on the encoded qubit. We say such a fault is *inconsequential*. Taking a step back, this is also expected since at the time when the fault happens the associated Pauli operator $Z_i Z_j$ is an element of the ISG. With respect to the chosen input (on the left) and output (on the right), the network also has stabilizer flows, e.g.,



which shows that any associated circuit projects any input state into a $+1$ eigenstate of the stabilizer $(-1)^{m_1} Z_i Z_j$ after the first measurement, up to signs of the stabilizers of the repetition code. By symmetry of the network we find the same stabilizer flows on the output legs. Knowing these stabilizer groups is mainly important when designing QEC experiments (see Sec. V A) and interfacing codes (see Sec. VII).

We would like to highlight that the preceding analysis of the effect of faults was independent of their physical origin. In particular, we could have placed faults on different edges in the network, possibly representing different types of physical error mechanisms, e.g., measurement errors,

and could follow the same arguments. This naturally leads to the definition of a *fault distance* as the smallest number of (order- p) generating faults that need to be combined to form a consequential undetectable fault. Importantly, this definition depends on the error model but is applicable to all cases where the noise acts as a stochastic Pauli channel. Note that the definition of fault distance has appeared similarly in Ref. [59], but we phrase it in terms of the language of RGB tensor networks and Pauli flows.

We also want to note that there is an efficient algorithm to obtain all Pauli flows of a network. If the network directly comes from a circuit, we can equivalently perform standard stabilizer tableau calculations rendering it unnecessary to describe the algorithm in this work. There may still be advantages gained in the simulation performance by considering the tensor network representing the whole spacetime evolution instead of time slices through the network, which is done in a more conventional simulation with stabilizer tableaus. We leave this analysis to future work.

V. FAULT TOLERANCE IN THE XYZ RUBY CODE

In this section, we investigate the fault-tolerance properties of the XYZ ruby code. To set the stage, we first introduce the notion of *probing* an error-correcting protocol. This will unify various types of error-correction experiments [60] and give a constructive framework to assess the quantities that can be probed in a given experiment. It will also make transparent how initialization and read-out layers have to be designed in order to probe a desired quantity. We note that our perspective is closely related to the one obtained from the detector error model [61,62] and Delfosse's spacetime code [63], which we comment on in Sec. VII C.

We exemplify our perspective by benchmarking various quantities in the XYZ ruby code on a torus. To do so, we first identify the Pauli flows of the tensor network of the XYZ ruby code with a rewinding schedule. We describe the detector, stabilizer, and logical flows and give some intuition on how they are affected by Pauli errors within the circuit. We find that, as expected, the tensor network is truly topological and there is a natural separation between local and nonlocal flows. This leads to two types of QEC experiments that we perform: memory and stability experiments. Both have physical interpretations in terms of the underlying topological phase. In both experiments, we find competitive thresholds and an exponential suppression of the logical error rate under various noise models for moderate system sizes using a belief propagation decoder enhanced with ordered and localized statistics decoding (BP + OSD/LSD). Finally, we describe the details of the implementation and the numerical experiments we perform to extract performance indicators.

A. Probing an error-correcting protocol

In the following, we introduce the concept of *probing* an error-correcting protocol expressed as an RGB tensor network. Since we live in a classical world, we will never be able to probe the full linear map implemented by an error-correcting protocol in a single experiment. In practice, “initialization” and “read-out” circuits have to be chosen, probing only matrix elements. When probing an RGB tensor network, both initialization and read-out layers can also be expressed as an RGB tensor network, usually of constant depth. They are networks that only have input, respectively output, legs and their (logical) action is fully described by their stabilizer flows.

We understand a logical probing experiment as the tensor network obtained when terminating the tensor network of the error-correcting protocol with an initialization and read-out layer. This tensor network does not have any open legs and hence represents a number. However, this number will not play an important role in the experiment. More importantly, all Pauli flows of the full experiment are detector flows. As such, they all give rise to a set of signs of tensors (mostly these will correspond to measurement outcomes) whose sum is 0 mod 2 in the absence of noise. In particular, they define a classical (linear) code. These will be the only quantities entering the probing experiment.

Every RGB tensor network is labeled by a set of signs of individual tensors. Some of them are fixed (to 0 or 1), others are open. One can think of the open ones as being determined by measurement outcomes. Let M be the $\mathbb{Z}_2$ -vector space spanned by all possible signs in the tensor network. For convenience, we can disregard the ones being fixed to 0 and assume we investigate a circuit where the remaining signs correspond to measurement outcomes. The set of detector flows of the experiment forms a group, which we denote by D . It defines a classical linear code $C_D \subseteq M$. For any probing experiment we pick a subgroup of *decoding flows* $D_d \leq D$. This subgroup also defines a linear classical code, $C_{D_d}^d$ and a quotient group D/D_d of *observable flows*. Importantly, $C_D \subseteq C_{D_d}^d$. In this context, the decoding problem can be phrased as follows: Given the syndrome information associated to D_d , respectively $C_{D_d}^d$, find a prediction for the information associated to D/D_d .

In practice, we define D and D_d in terms of their generators, which in turn defines parity-check matrices H_D and $H_{D_d}^d$ for both C_D and $C_{D_d}^d$ [64]. $H_{D_d}^d$ will be a submatrix of H_D (obtained by removing rows associated to the constraints D/D_d) and we denote the complementary submatrix as $H_{D_d}^o$. Given these definitions, the (numerical) experiment works as follows:

- (1) Sample the full erroneous circuit, resulting in a binary vector of $\mathbb{Z}_2$ values (e.g., measurement outcomes) $\mathbf{m}$.
- (2) From these infer

(a) *syndromes* $\mathbf{s} = H_D^d \mathbf{m}$ and

(b) *observations* $\mathbf{o} = H_{D_d}^o \mathbf{m}$.

- (3) Decode the syndrome $\mathbf{s}$, ideally using information about the error model [65], to obtain a binary vector $\mathbf{m}' \in M$ that can be interpreted as a proposed set of measurements that need to be flipped to “redo” the effect of the errors in the circuit. In that reading, $\mathbf{m}'$ has a 1 at the position of each measurement that needs to be flipped and 0 everywhere else.
- (4) From $\mathbf{m}'$, calculate a predicted value for the observations $\mathbf{o}' = \mathbf{o} \oplus H_{D_d}^o \mathbf{m}'$.
- (5) Declare a failure if $\mathbf{o}' \neq \mathbf{o}$.

Of course, only specific choices for D and D_d give rise to a meaningful experiment. Moreover, the choice of initialization and read-out layer highly influences the choices that are even possible.

In practice, however, there are clear guidelines in designing both the initialization and read-out layer as well as for the choice of D_d . For the main part of this section we focus on topological codes where there are two main classes of experiments that differ by the choice of observables. In both cases, the detector group D splits into a locally generated part D_{loc} and a part that cannot be created locally. We pick $D_d = D_{\text{loc}}$ and identify D/D_{loc} with the set of observables of the experiment.

1. Memory experiments

A memory experiment probes the capability of a circuit to preserve logical information over time [18]. The observables are defined by flows that connect the two time-like boundaries of the experiment in a nontrivial way, i.e., are nontrivial elements in D/D_{loc} . In particular, they are formed by logical flows of the circuit that are “closed off” by stabilizer flows of the initialization and read-out layer. Additionally, some of the stabilizer flows of the circuit are promoted to local boundary detector flows. Together with the local detector flows of the circuit itself they generate the group D_d and with that define the syndromes for decoding. We depict the different classes of Pauli flows in a memory experiment in Fig. 8.

For example, to probe a logical flow that has only blue highlights on the output legs, we want to choose an initialization and read-out layer that only have blue stabilizer flows. Moreover, it should allow for an attachment of the blue logical flow. This is achieved, for example, by a single-qubit initialization in the Z basis represented by single-legged red spiders.

Importantly, the choice of logical flow that can be attached to the initialization and read-out layer automatically rules out other logical flows being probed and boundary detector flows being formed. For example, probing a logical Z -type flow rules out a probe of an anticommuting

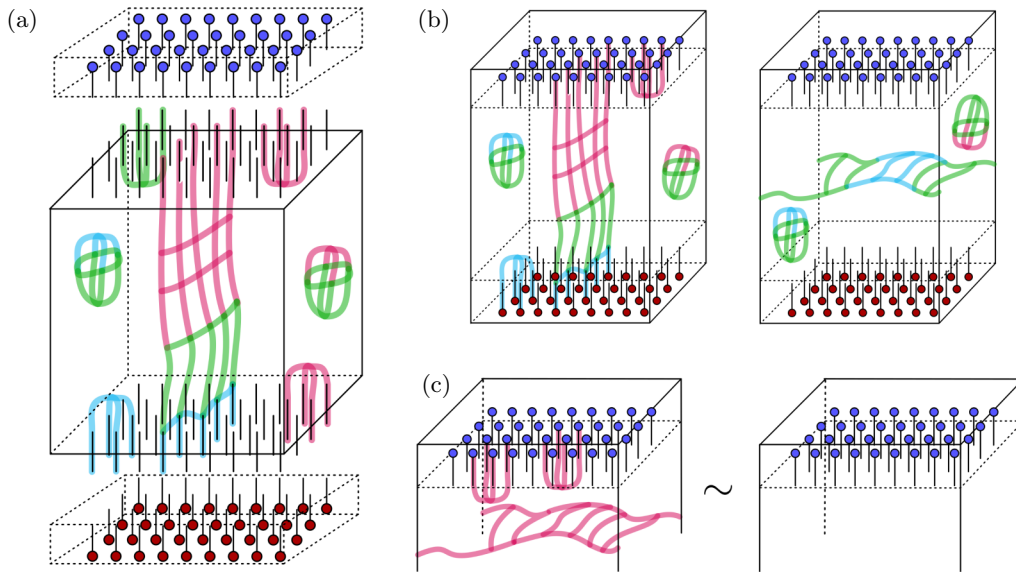


FIG. 8. Schematic depiction of the process of designing and performing an error-correction experiment to probe (nontrivial) observables in a protocol. We think of the left and the right of the cubes being identified, as well as the front and the back, as the code is defined on a torus, and time going upwards. (a) Given the bulk of a protocol and its flows, input and output layers need to be chosen to preserve the desired set of stabilizer and logical flows. (b) After closing off the input and output, the experiment is described by the detector flows of the resulting network. In topological protocols the homologically nontrivial flows are probed, using the local detector flows as input to the decoder. On the left, we depict the flow probed in a memory experiment, on the right the flow that is probed in a stability experiment. In general, they can be performed within the same experiment. (c) The choice of temporal boundaries determines which flows are probed in the experiment. A read-out layer can render a flow that was nontrivial in the bulk of the protocol trivial in the actual experiment. Here, we illustrate this for a red-colored flow that one would naively consider a nontrivial observable for a stability experiment.

X -type flow. Additionally, the local detector flows at the boundary are also constrained. As a consequence, close to the boundary, there will be Pauli errors that are not detectable in the experiment. However, since the logicals that would be flipped by that error are not probed by the experiment, any errors of that form are inconsequential and hence do not lower the distance observed in these experiments.

2. Stability experiments

The concept of a *stability experiment* has been introduced in Ref. [66] as a numerical experiment that probes how well a code can perform lattice-surgery-like operations. More specifically, it allows us to give a better estimate on how many times measurement rounds have to be repeated to suppress undetectable error chains in the time direction.

In this subsection, we focus on stability experiments in the context of $2+1$ -dimensional topological codes. The observables in a stability experiment are defined by nontrivial detector flows that connect spatially separated boundaries without connecting any of the temporal boundaries. Again, we understand nontrivial as being in a nontrivial equivalence class in D/D_{loc} . Note that these flows only exist for certain spatial boundary configurations.

Specifically, they are supported on a spacelike region in which the topological code has a property called *topological charge conservation* that guarantees a given parity on the stabilizers supported in that region [67]. For example, the stabilizers of the toric code (placed on, e.g., a surface without a boundary) multiply to the identity operator. In general, one can think of these types of nonlocal flows as giving rise to a global constraint on the stabilizers of an ISG. Any topological code has this property when placed on a closed, compact manifold, such as a torus or a surface with a single connected boundary component.

3. Interplay of stability and memory experiment

As stated above, it might seem that a stability experiment and memory experiment can be designed independently of one another but performed simultaneously. However, this is not true for topological codes. In order for it to be a nontrivial stability experiment we want the detector flow of the observable to be nontrivial in D/D_{loc} . In particular, this depends on the local detectors close to the boundaries, including the temporal ones. In order to perform a stability experiment we have to choose initialization and read-out layers that do not close off the local flows that could move the stability observable into the boundary and thereby make it trivial. This restricts the type of

stability experiments that can be performed simultaneously to a memory experiment. For example, we cannot probe an X -type memory observable together with an X -type stability observable, even if the circuit itself has both types of flows to begin with. Either the stability observable will be trivial or the X -type logical flow will be trivial in the experiment, i.e., can be written as a sum of local detector flows.

We illustrate this phenomenon in Fig. 8. Such trade-offs exist in any topological code and are related to the fact that all boundaries in the experiment have to be topologically protected boundaries in order to have a protocol with macroscopic fault distance. In $2 + 1$ dimensions, these are fully determined by so-called *Lagrangian subgroups*, which determine what type of anyon worldlines can be terminated at that boundary. These in turn define which detector flows exist close to the boundary (the ones detecting the confined anyons) and which logical flows can terminate at the boundary (the ones that correspond to the worldsheet of the logical operators associated to the anyons in the Lagrangian subgroup).

B. Pauli flows of the XYZ ruby code

In the following, we identify Pauli flows of the XYZ ruby code protocol defined by the rewinding schedule. As described in the section above generically for topological protocols, we partition the Pauli flows into *local* and *nonlocal* flows.

1. Decoding flows

We find detector flows of the rewinding protocol by first considering spacelike symmetries and then consecutively highlighting edges of the XYZ tensor network until we find a spatially and temporally confined Pauli flow, so we will use them as decoding flows in probing experiments. In the following, we will refer to them as *detectors*. Consider one unit cell of the ruby lattice with a few time steps before and after a ZZ measurement. Inserting a generator of the ISG,

$$S_0 = \left\langle \begin{array}{c} \text{Diagram showing a unit cell of the XYZ tensor network with highlighted edges and spiders.} \end{array} \right\rangle, \quad (65)$$

as faults at that time step have, by definition, trivial action on the circuit. In the three-colored graphical calculus, this corresponds to placing two-legged spiders for each qubit a generator acts on which leaves the RGB tensor network invariant. Mapping these to highlights [cf. Eqs. (46)] does not give a valid detector flow yet. For this, we have to construct highlights of adjacent edges, such that at each tensor, the highlights correspond to a Pauli flow, and nontrivial highlights form a spatially and temporally compact set. We can assume that the highlights emanating from the ISG

generator are part of a detector flow, because the previous measurement rounds have initialized this generator. Owing to the mirror symmetry of the network, we can expect that the corresponding measurements of the following rounds will contribute in forming a detector. We therefore also only need to highlight edges in one time direction, e.g., in time forward after the ZZ measurement, and the other half of the highlights are their mirror.

We start with the Z -plaquette generator and highlight the corresponding inner [68] timelike edges after the ZZ measurement at $t = 3$ in blue [cf. Fig. 9(a)]. At the green spider in the next time step $t = 4$, valid highlights of legs are all blue with an even number of red highlights, cf. Eq. (50a). Having the next XX measurement in mind we choose a pair of red highlights on the horizontal and vertical upwards outgoing legs of the spider. At the red spiders ($t = 5$), we choose to highlight the horizontal legs in red, leading to a (half) closed flow. A reflection about the ZZ measurement at $t = 3$ generates a closed Pauli flow in one unit cell from times $t = 1 \bmod 6$ to $t = 5 \bmod 6$ and therefore a detector flow.

Analogously, detectors can be formed by starting with blue highlights after the ZZ measurement at $t = 0$. After the next (and before the previous) XX measurement, the color of the detector flow should be green in order to be closed at the YY measurement in $t = 2$ and $t = 4$, see Fig. 9(b). These detectors exist for every unit cell, and we call them *rb* and *gb* *plaquette detectors*.

Also from an operational perspective these are reasonable detectors: measuring XX and YY once gives the eigenvalue of the Z plaquette as a product of the outcomes around the plaquette. This operator trivially commutes with the following ZZ measurement. In the rewinding schedule, now measuring YY and XX reads out the plaquette a second time without randomizing its eigenvalue.

The Y - and X -plaquette stabilizer generators give rise to a second set of detectors, shown in Figs. 9(c) and 9(d). These *plaquette detectors* start and end in a ZZ measurement and have again two different colors. Depending on the measurements around the middle ZZ step, we call them *bg* and *br* (right) for YY and XX measurements.

Another set of local detectors is purely spacelike and formed by the *triangles* of the ZZ measurements highlighted in blue. They correspond to the constraint that the value of the third ZZ measurement is already determined by the sum of the first two.

This completes the set of generating detectors for the rewinding schedule. In the period-3 schedule, we find analogous plaquette and plaque detectors. Note that they do not exhibit the mirror symmetry about the ZZ measurement due to the missing symmetry in the schedule.

2. Observable flows

Here, we show two types of flows that we will use as observables in memory and stability experiments.

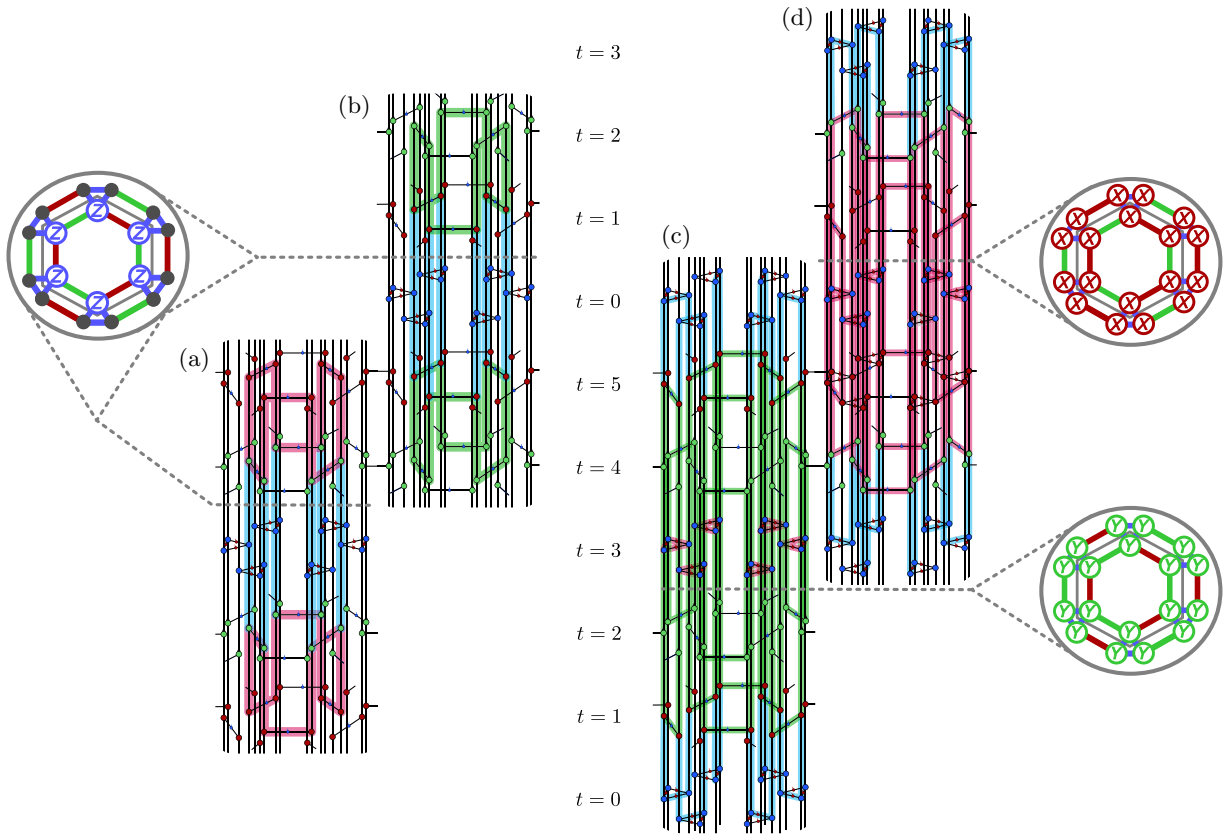


FIG. 9. Detectors of the XYZ ruby code in the rewinding schedule, with time going upwards. For each detector flow, there exists an equivalent one shifted by six time steps, corresponding to the period of the schedule and equivalent ones centered around every plaquette. Whenever a measurement spider is charged with respect to the flow (i.e., it is highlighted by a different color), it contributes to the set of measurements that are constrained to be deterministic in the error-free case. Cutting detector flows gives rise to instantaneous stabilizers. (a) rbr and (b) gbg plaquette detector flows. These flows combine 12 measurements that correspond to reading out the eigenvalue of the six-body Z plaquette twice by combining XX and YY measurements. (c) bgb and (d) brb plaque flows. They correspond to measuring a 18-body Y - or X -plaquette operator two times. Taken together, all detector flows densely cover the RGB tensor network.

The first are the logical flows of the QEC protocol that is implemented by the RGB TN observable flows. They are directly related to logical operators of the ISGs, cf. Fig. 4. In fact, using the intuition developed in the construction of detectors, we can find the measurements required to properly transform the logical operator from one time step to the next, cf. Fig. 5. In Fig. 10, we depict two (Pauli-)inequivalent logical flows, which we will refer to as rb (or $\mathcal{L}_X$) and as gb (or $\mathcal{L}_Y$) logical flows. They are supported along homologically nontrivial membranes as in any $2+1$ -dimensional protocol. When constructing a memory experiment, we will use thin layers that are compatible with one type of these and declare them as observables of the memory experiment.

The second type are detector flows that cannot be generated by above introduced local detector flows. We employ these global detector flows as observables in a stability experiment. From topological charge conservation, we expect the existence of four independent global

constraints, one for each independent anyon in the color code [42]. These give rise to two types of logical flows that are closely related to conservation laws in the (static) subsystem code. There, a global constraint is the product over all small string operators of one color around all vertices that evaluates to identity [cf. Eq. (5) of Ref. [45]]. To obtain those, we highlight the blue edges around all plaques of one color, e.g., red as shown in Fig. 11(a). Adding highlights backward in the time direction, there is a choice of consistent highlight to apply at the next (YY) step. We change the color to red such that at the next XX measurement, the highlights can be chosen to close the flow obtaining an rb observable flow. Choosing a second color of plaques for highlighting the blue edges at the ZZ -measurement time step, we get an independent observable flow. The third color, however, is the product of the flows of the other two colors, up to local triangle detectors. This gives a total two independent observable flows. A second set of two independent flows can be equivalently obtained

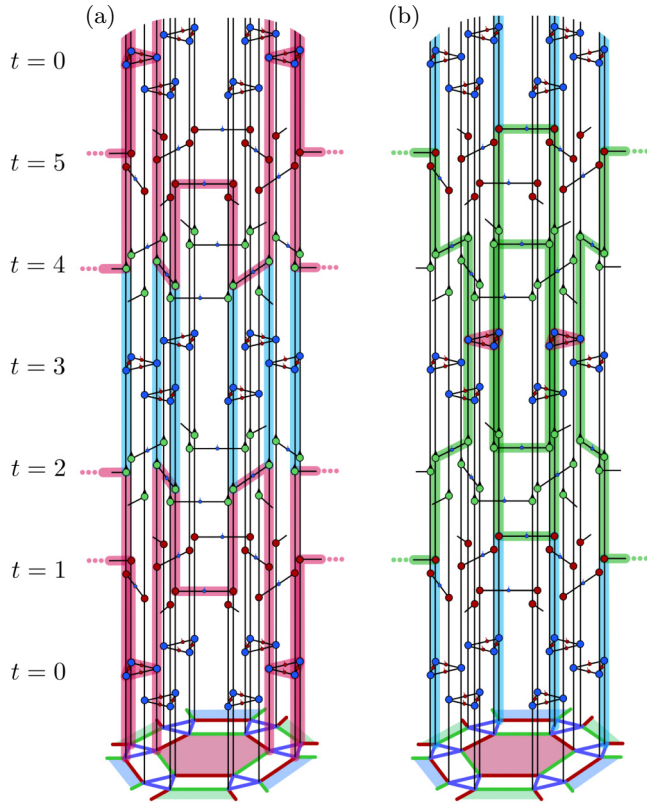


FIG. 10. We show parts of two logical flows in the bulk of the XYZ ruby code with the rewinding schedule. When cutting the network along a plane perpendicular to the time direction, i.e., at some time t_0 , the flows give rise to logical operators of the ISG at t_0 . As in any $2 + 1$ -dimensional topological protocol, the logical flows are membranelike, indicated with the colored dots on the left and the right of both flows. Each nontrivial logical flow is supported along one homologically nontrivial membrane. In (a), we show an rb ($\mathcal{L}_X$) logical flow that when cut after $t = 0$ gives rise to a rx string operator going through the red plaquette shown at the bottom, cf. $\bar{X}_1$ of Fig. 4. Cut after $t = 3$, the string operator has evolved to a bz operator through the blue plaquettes adjacent to the blue plaquette. Similarly, the gb ($\mathcal{L}_Y$) logical flow in (b) on the right can be associated with a $rx \times gz = bz$ (after $t = 0$) and a gy (after $t = 3$) string, as expected from the automorphism applied in these three time steps, cf. Eq. (24).

starting at the ZZ -measurement step at $t = 0 \bmod 6$, sandwiched by two XX measurements, which we show for highlight around green plaques in Fig. 11(b). Again, the product of two such flows of two colors yields the flow of the third color. We therefore get two more independent (gb) observable flows. It is apparent in the structure of both observable flows that they cannot be generated by products of local detectors, which only deform the observable flows in spacetime.

Already at that point, even before constructing the actual probing experiment, we can make statements on the fault tolerance of the protocol, based on decoding and observable flows as we show in the next subsection.

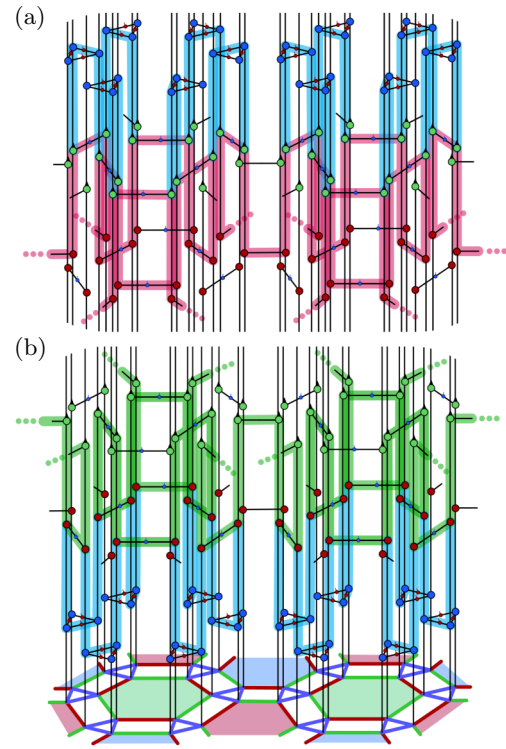


FIG. 11. Observable flows related to global constraints of the ISGs, i.e., products of ISG elements. There are two independent constraints related to which color of plaquettes are chosen in space (see also main text). Additionally, an offset in time gives two additional independent flows to get a total of four as expected by topological charge conservation in the color-code phase. In (a), we show the rb closing off blue highlights on ZZ edges around a red plaque. In (b), we show a time offset gb flow, starting with blue highlight around a green plaque.

C. Phenomenological fault tolerance

To investigate the fault tolerance of the protocol, we consider a simplified but complete error model based on the observations [69] that

- (1) a measurement error during measurement M_{PP} is equivalent to a Pauli error $P' \neq P$ just before and after a noiseless measurement,
- (2) any Pauli P is proportional to a product of two other Paulis, $P \propto P'P''$, and it is therefore enough to consider a basis of two Pauli operators after each measurement,
- (3) a Pauli fault P can be commuted through a measurement M_{PP} ,
- (4) after a measurement $M_{P_i P_j}$, the Pauli operator $P_i P_j$ is part of the ISG, such that the errors P_i and P_j are stabilizer equivalent $P_i \sim P_j$.

The last property is a consequence of the just-measured two-body operators on edges being inconsequential faults or fault equivalences [70].

This leaves a restricted error model where we place single-qubit X , Y , and Z faults after the XX , YY , and ZZ measurements respectively. Any other Pauli fault (of arbitrary weight involving any spatial or temporal location) can be generated by this set. We show some of these properties in the RGB TN in Appendix C, Fig. 23. Since any single-qubit fault on the other qubit(s) of the just-measured edge (triangle) is equivalent, we can also think of the red, green edges and blue triangles as the elementary faults.

All the elementary faults of this error model violate three detectors. This is similar to the action of elementary faults in a standard color code, where X - and Z - faults violate the three adjacent plaquette detectors, $Y \propto ZX$ and $c_Y = c_Z + c_X \pmod 2$ for X and Z faults inserted in the network at the same location. Here c_P is the charge of the corresponding P spiders with respect to the Pauli flows of the network, or equivalently, the syndrome. We find, however, that other properties are different from 2 + 1-dimensional color-code decoding graphs.

Firstly, the XYZ ruby code is not CSS and a Y elementary fault is independent of X and Z elementary faults. Secondly, the elements of the ISG of the current time step

represent inconsequential errors at that time step, i.e., any fault can be multiplied by an element of the ISG to get an equivalent fault with the same action on the codespace. As shown in Sec. III B, the elements of the ISG can be regarded as stabilizers of a color code for $\mathcal{S}_0$ and three copies of toric codes for $\mathcal{S}_1^{(r)}$ and $\mathcal{S}_2^{(r)}$. This phase transition is reflected in the behavior of faults at different time steps, for which we would like to develop a physical intuition.

To this end, suppose we place the elementary faults on a single qubit and investigate how these violate the detectors. In Fig. 12(a), we show detectors in time, represented as vertical shades in the color of the respective highlights of the flow. Horizontal lines represent single-qubit X , Y , and Z faults after the corresponding two-body measurements. Whenever crossing horizontal and vertical lines have different colors, the fault is charged with respect to the corresponding decoding flows and triggers the associated check in the decoding graph. Note that elementary X faults only violate g -type detectors (i.e., bgb plaques and gbg plaquettes) and elementary Y faults only violate r -type detectors (i.e., brb plaques and rbr plaquettes). The Z faults violate plaques only. To describe how syndromes of

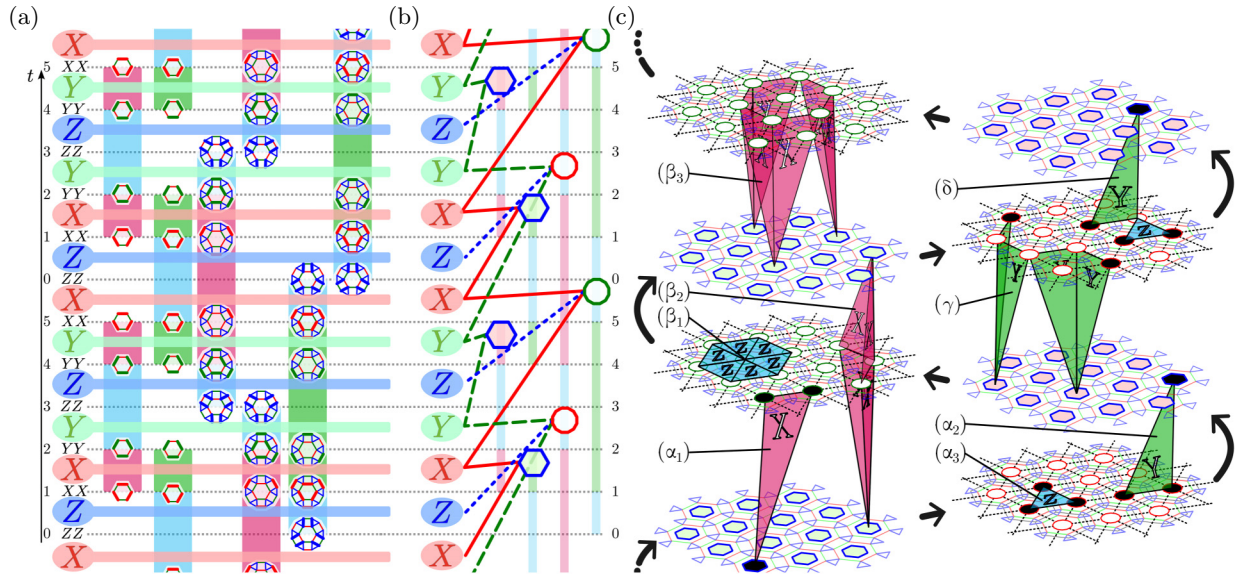


FIG. 12. (a) Action of elementary faults on detectors. We place single-qubit X , Y , and Z faults after XX , YY , and ZZ measurements, respectively, and represent them by horizontal shades. We draw the temporal extent of the detectors vertically and use a shade corresponding to the highlights of the Pauli flows. We also indicate the edges contributing to the value of the detector. Whenever the colors of single-qubit faults and detectors differ, the corresponding detector is violated. (b) This allows one to draw a time slice of the decoding graph, a bipartite graph with elementary faults connected to the detectors they trigger. We represent detectors by hexagons and dodecagons attached to the last time step of the corresponding shade. For the X and Y elementary faults, the decoding graph corresponds to two Tanner graphs of (classical) repetition codes. Z elementary faults violate plaques of a single time step. (c) For every time step where detectors close, we draw a plane associating them to their spatial location. Since elementary faults flip three detectors, we now represent them as triangles. We draw a subset of faults of the resulting spacetime decoding graph. Time follows the arrows, elementary faults are red (X), green (Y), and blue (Z) triangles. Elementary faults occurred on qubits that are supported on a red (green) edge cut by the short side of the X (Y) triangle, or on one of the three qubits covered by the Z triangle. A detector that is violated is filled. (α_{1-3}) Elementary faults violate three detectors. (β_{1-3}) We show some faults that do not flip any detectors. They are typically related to elements of the ISG ($\beta_{1,3}$), but can also involve faults of different time steps. (γ) Pure Y (and X) faults always violate two plaquette stabilizers on the endpoints of a string. (δ) Combination of Y and Z faults.

elementary faults combine, we draw decoding graphs that have vertices for faults and detectors and edges between fault E_i and detector D_j if E_i triggers D_j . In Fig. 12(b), we show a decoding graph abstracting away the spatial locations of faults and detectors, focusing on their action in time. The X and Y faults generate Tanner graphs of classical repetition codes, similar to how measurement errors trigger two temporally spaced detectors of a surface-code experiment. The Z faults violate plaques of a unique time step, in a similar way to data-qubit errors of a color-code experiment.

In Fig. 12(c), we extend that picture to include faults at different spatial locations. First, the temporal behavior induces a natural bipartition of the detectors into those violated by X and Z faults after $t = 0 \bmod 6$, and those by Y and Z faults after $t = 3 \bmod 6$. For every detector we place vertices on the ruby lattice in the center of the plaquette(s). Because the elementary faults always trigger three detectors, faults are extended to hyperedges, which we draw as triangles in Fig. 12(c). Whenever an odd number of a triangle's corners touch a detector, it is violated.

We draw subsets of fault configurations, starting with the elementary faults in Fig. 12(c) (α). The Y and X faults violate a single plaquette and two plaque detectors spanning over different time steps. The Z faults violate three adjacent plaques in a single time step. Superimposing a triangular lattice on the ruby lattice with triangles in the faces, this is equivalent to how Z faults behave in a regular color code on a hexagonal tiling.

In Fig. 12(c) (β), we show some other inconsequential errors. In (β_1), six Z faults in a loop surrounding six triangles, such that each involved plaquette detector is flipped twice, are undetectable. These fault configurations correspond to the stabilizer of the ISG at that time step, cf. Eq. (29), and have a resemblance to the smallest homologically trivial loops in hexagonal color codes. In (β_2) we show two X faults on one plaquette, both repeated at consecutive time steps as an inconsequential error that spans space and time. The weight-6 X fault shown in (β_3) is another fault related to ISG elements, this time a stabilizer of one of the three toric codes of S_1 , cf. Fig. 6.

The relation to the decoding problem of toric codes becomes apparent in Fig. 12(c) (γ). If we restrict the Y (X) faults to the subset of plaque detectors, they always trigger them in pairs at the endpoint of a string of faults. We believe that this can be used to design a matching-based decoder, similar to how matching decoders are adapted to color codes [71–73].

Having established that the distance of the XYZ ruby code is $\mathcal{O}(s)$ (where s is the one-dimensional system size) and considering that any single-qubit fault can be uniquely identified, we conjecture that the XYZ ruby code also exhibits a threshold. This is supported by the numerical results of the following sections. A mapping to a random

bond Ising model that incorporates the above-mentioned equivalences can provide a lower bound on a maximum-likelihood decoder but is left for future work (cf. e.g., Refs. [74,75]). We would like to note that we expect an interesting model, considering that the decoding graph (of the restricted noise model) is that of interleaved color-code and toric-code decoding graphs.

D. Circuits for memory and stability experiments and noise models

Knowing the colors of the highlights of decoding and observable flows, we can design memory and stability experiments by choosing compatible input and output layers corresponding to single-qubit initialization and measurement. Take, e.g., an initialization in the n -qubit product-state vector $|0\rangle^{\otimes n}$. This corresponds to a collection of single-legged red tensors as already mentioned above. These have blue Pauli flows and can therefore attach to any other blue Pauli flow. Both the $\mathcal{L}_X$ and $\mathcal{L}_Y$ memory logicals have blue flows around one of the ZZ measurements. Cutting a bulk protocol in between ZZ and YY or ZZ and XX measurement, respectively, therefore allows us to initialize the $\mathcal{L}_X$ and $\mathcal{L}_Y$ logicals. Crucially, these cuts allow us to *fault tolerantly* initialize the corresponding logical, because we can form enough detectors by closing stabilizer Pauli flows. In particular, the first ZZ measurements are deterministic, as are the Pauli flows that are half of the plaquette detectors (i.e., the product of subsequent XX and YY measurements, or vice versa). We show the memory experiment protocols graphically in Fig. 13.

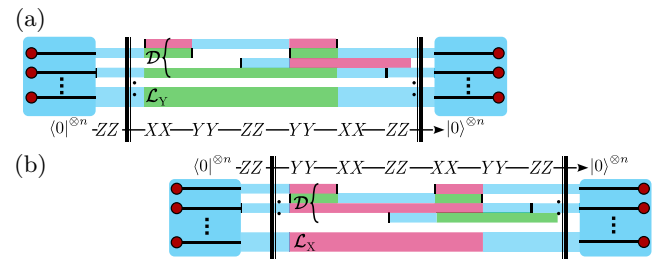


FIG. 13. Memory experiments. We show thin Z -basis initialization and measurement slices that sandwich the bulk protocol. The upper stripes of the (potentially repeated) bulk sequence indicated by $\mathcal{D}$ corresponds to decoding flows (cf. Fig. 12) and the lower to observable flows (cf. Fig. 10). In (a), we cut the network after measuring XX and before measuring ZZ to attach the initialization spiders. Similarly, we cut after the ZZ (and before an XX) measurement to attach the read-out spiders. The (blue) Pauli flows of the single-leg spiders match the blue highlight of the $\mathcal{L}_Y$ logical flows; this choice of cut therefore initializes and reads out these logical operators. Additionally, $\mathcal{L}_Y$ -plaquette stabilizer flows can be closed to generate detectors at the boundaries. In (b), cuts shifted by one half-cycle achieve the same for the $\mathcal{L}_X$ logical flow.

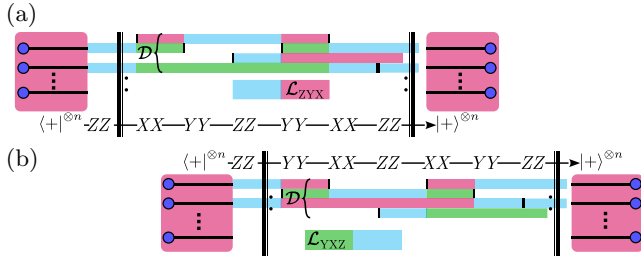


FIG. 14. Stability experiments. We show thin X -basis initialization and measurement slices that sandwich the bulk protocol. The upper stripes of the (potentially repeated) bulk sequence indicated by $\mathcal{D}$ corresponds to decoding flows (cf. Fig. 12) and the lower to observable flows (cf. Fig. 11). In (a), we cut the network after measuring XX and before measuring ZZ to attach the initialization spiders. Similarly, we cut after the ZZ (and before an XX measurement to attach the measurements). The (blue) Pauli flows of the single-leg spiders do not match the blue highlight of the ZYX -like stability flows, this choice of cut therefore leads to random initialization. Additionally, gbg -plaquette stabilizer flows do not close to generate detectors at the boundaries. In (b), cuts shifted by 1 half-cycle achieve the same for the $\mathcal{L}_{YXZ}$ stability flow.

For stability experiments the relevant observables are initially random, so we initialize in a different basis and declare the global constraints of Fig. 11 as observables, see Fig. 14.

We benchmark two different circuit implementations with different noise models. First, we assume that direct two-body Pauli measurements are available. When measuring the three edges of a triangle one after another, a cycle ($XX \rightarrow YY \rightarrow ZZ$) takes five steps. Unlike for the honeycomb code, some qubits will therefore be idling during the ZZ measurements. For this circuit, we employ two noise models. The first is a *phenomenological noise model* where we insert single-qubit depolarizing channels $\mathcal{D}^{\otimes 1}(p)$ in between the check measurements. Additionally, every measurement outcome is flipped with the same probability p . The second is the *EM3 noise model* [76] that is commonly used in the Floquet-code literature. Here, single-qubit depolarizing channels are replaced with two-qubit depolarizing channels $\mathcal{D}^{\otimes 2}(p)$ after a check measurement. This is motivated by the expected behavior of hardware noise on potential Majorana-based quantum computing platforms [77]. We expect the EM3 noise model to have a reduced distance due to two-qubit faults that already appear at first order. A detailed analysis of the noise channels of the elementary building blocks is deferred to Appendix C.

Additionally, we construct circuits with single-qubit initialization and measurement in the Z basis, single-qubit Clifford rotations on the data qubits, and entangling CNOT gates. We place one auxiliary qubit on each edge to facilitate the two-body measurements and schedule the gates

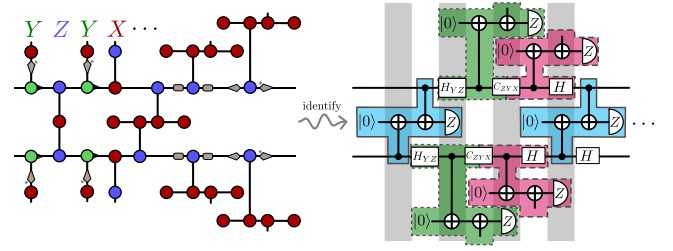


FIG. 15. Translating the XYZ ruby code tensor network to a circuit. We first use basis changes to express the network with blue and red spiders. We then split the red measurement spiders to Z -basis initialization, two CNOTs and a Z -basis measurement. The circuit can be scheduled such that in the bulk of the protocol no qubit is ever idle. Here, $C_{ZYX} = H_{YZ}H$ permutes the bases anti-cyclically. Note that any circuit obtained from rewrite rules performs the readout.

as shown in Fig. 15. With the exception of the first three layers in this circuit, no data or auxiliary qubits are idling. We use a uniform circuit-level noise model, with noisy operations specified in Table I.

E. Sampling, decoding, and notes on implementation

We construct the circuits in `stim` [61] and annotate the measurements that contribute to detectors and observables, as well as error channels. We generate a *detector error model* by evaluating with respect to which elementary faults defined by the error channels the detector and observable flows are charged [78]. The detector error model is a list of elementary faults with their probability and the detectors and logical observables this fault flips. We can directly sample detection events (syndromes) $\mathbf{s} \in \mathbb{F}_2^{n_d}$ and logical flips (observations) $\mathbf{o} \in \mathbb{F}_2^k$ [79]. This completes the first two steps of probing the error-correcting protocol, cf. Sec. V A. To decode this syndrome, we convert the detector error model into a *detector matrix* $\mathbf{M}_d \in \mathbb{F}_2^{n_d \times n_e}$, a *logical matrix* $\mathbf{M}_l \in \mathbb{F}_2^{k \times n_e}$ and a *prior vector*

TABLE I. Noise models used in benchmarking the circuits in this work. The noisy multi-Pauli measurement gate $M_{pp}(p)$ has two different realizations. In the phenomenological model, $M_{pp}(p)$ corresponds to a single-qubit depolarizing channel on data qubits of its support and a noisy measurement outcome. In the EM3 model, the single-qubit depolarizing channel is replaced by a two-qubit depolarizing channel on data qubits of its support and a noisy measurement outcome.

Noise model	Phenomenological, EM3	Circuit level
Noisy gateset	$\text{Init}_Z(p)$	$\text{Init}_Z(p)$
	$M_{pp}(p)$	$M_Z(p)$
		$C_1(p)$
		CNOT(p)
	Idle(p)	Idle(p)

$\mathbf{p} \in \mathbb{R}^{n_e}$. Here n_d , n_e , and k are the number of detectors, error mechanisms, and logical observables, respectively.

The task of a decoder is to find a set of error mechanisms $\mathbf{e} \in \mathbb{F}_2^{n_e}$ that fulfills $\mathbf{M}_d \mathbf{e} = \mathbf{s}$, i.e., a set of error mechanisms that produces the observed syndrome. To verify whether a correction would result in a logical error, the prediction $\mathbf{o}^\star = \mathbf{M}_l \mathbf{e}$ has to equal the observed logical flips, i.e., $\mathbf{o} \oplus \mathbf{o}^\star \stackrel{!}{=} \mathbf{0}$. Note that we decode directly for error mechanisms and not for flipped measurements, therefore adapting the protocol compared to step 4 and 5 of Sec. V A to make use of information on the error model. Using the faults or the measurement outcomes is equivalent for getting the predicted values of the observables. However, including information on the probability of faults can only improve the accuracy of decoding.

Based on our discussion above, the effective error model derived from the circuit is not trivially decomposable into errors with at most two syndromes. This renders a non-adapted matching-based decoder such as pymatching not suitable for this problem. We therefore employ the general purpose two-stage decoders *belief propagation + ordered statistics decoding* (BP + OSD) and the recently introduced *belief propagation + localized statistics decoding* (BP + LSD) [80–82]. The first stage uses belief propagation on the Tanner graph defined by the detector matrix $\mathbf{M}_d$ to approximate the single error mechanisms' marginal probabilities based on the observed syndrome $\mathbf{s}$ and the *a priori* probabilities of elementary error mechanisms $\mathbf{p}$. Typically—and in particular if the Tanner graph contains many small loops—BP is inconclusive in its output, returning an error guess that does not match the observed syndrome. However, it usually points to a subset of likely

erroneous qubits; OSD then brute forces the solution of the decoding problem on that subset via matrix inversion. LSD provides a speed-accuracy trade-off by doing inversion on localized cluster, which we exploit to generate more data for larger codes and smaller physical error rates. We give details on the implementation of BP + O(L)SD and the used parameters in Appendix D.

F. Results

In all following results, we take a conservative approach and report the logical failure probability as the probability that at least one of the four independent (logical) observables probed in the experiment is predicted wrongly by the decoder.

1. Memory experiment

In Fig. 16, we show logical error rates for the $\mathcal{L}_Y$ logical observable with increasing distance d and d cycles for the phenomenological, EM3 and circuit-level noise model, respectively. We find that for all three circuits and noise models, the logical error rate curves cross, indicating the existence of a threshold. Errors are obtained from a finite-size scaling analysis, which we detail in Appendix E. For the phenomenological noise model we find

$$p_{\text{th}}^{\text{ph}} \approx 0.28 \pm 0.02\%. \quad (66)$$

For the EM3 noise model we find

$$p_{\text{th}}^{\text{em3}} \approx 0.36 \pm 0.04\%. \quad (67)$$

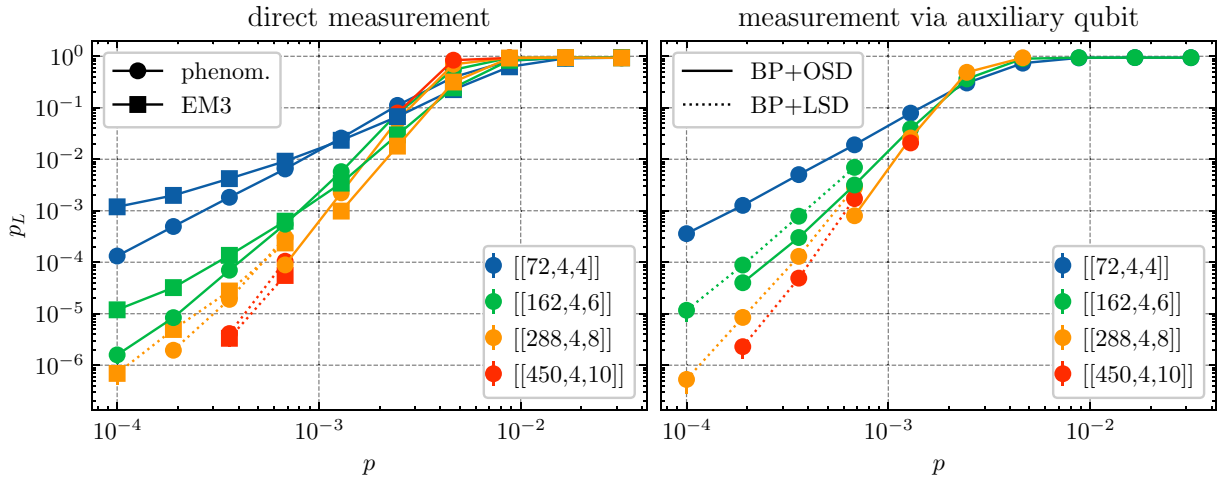


FIG. 16. Memory experiment for the $\mathcal{L}_Y$ logical observable using direct two-body measurements and measurements using an auxiliary qubit. We decode using BP + OSD (solid lines) and BP + LSD (dotted lines) for low physical error rate. BP + LSD has a higher logical error rate, but decodes orders of magnitude faster. For the direct measurements, we compare a phenomenological and an EM3 noise model with the latter showing a slightly higher threshold ($p_{\text{th}}^{\text{em3}} \approx 0.36 \pm 0.04\%$ vs $p_{\text{th}}^{\text{ph}} \approx 0.28 \pm 0.02\%$) but a worse scaling for small physical error probabilities p . Measuring using an auxiliary qubit and simulating full circuit-level noise shows a crossing at $p_{\text{th}}^{\text{cl}} \approx 0.18 \pm 0.01\%$. Here and in all other plots, error bars are standard Monte Carlo sampling errors and are smaller than the symbol.

For the circuit-level noise model we find

$$p_{\text{th}}^{\text{cl}} \approx 0.18 \pm 0.01\%. \quad (68)$$

The scaling of the logical error rate in the low p regime is $\propto p^s = p^{d/2}$ for the phenomenological and the circuit-level noise model, as expected. Because the distance of the codes on the torus is even, no failure occurs for faults with weight less than $d/2$. This confirms that these circuits and noise model are fault tolerant, as the (dynamic) fault distance coincides with the static distance of the codes defined by the ISGs. As expected, the fault distance is halved when using the EM3 noise model. This is reflected in a flattening of the curves at low physical error rate p .

The standard depolarizing threshold for d rounds of the (toric) honeycomb Floquet code is estimated in Refs. [76, 83] to lie within 0.2–0.3%, which is not significantly larger than our approximately equal to 0.18%. Note, however, that for the toric honeycomb code, they report the failure rate of one of the logical qubits to better compare to planar surface codes. Asymptotically, the threshold for failure of one or any of the logical qubits is the same. Due to the considered system sizes and the suboptimality of the decoder used, we consider our numbers a conservative estimate of the asymptotic threshold. We observe an improved threshold for the EM3 noise model, however only by a factor of approximately equal to 2 compared to a factor of approximately equal to 5 for the (toric) honeycomb Floquet code. One reason for the difference is that we measure the triangles in three separate two-body measurements.

Constructing and benchmarking fault-tolerant circuits for stabilizer readout of planar static color codes with triangular boundaries encoding a single logical qubit has seen progress in circuit-level noise thresholds from 0.1% [84] to

$0.143 \pm 0.001\%$ [71] and 0.2% [85]. These improvements arise, on the one hand, by constructing more involved circuits for syndrome measurements using cat states or flag constructions, and, on the other hand, designing better decoders. The circuits to measure the two-body stabilizers in our protocols are inherently simple and we achieve thresholds that compare well to values from the literature, despite reporting the probability that any of the four logical qubits fails. Additionally, a comparison is difficult because of our usage of BP + O(L)SD decoders, which is a fundamentally nonoptimal approach. Moreover, BP + O(L)SD has a reduced accuracy for large decoding problems, i.e., many qubits or a lot of rounds, see, e.g., Ref. [86]. This can be observed in Fig. 17, where we show the suppression of logical error rate with growing size in the subthreshold regime. The suppression is exponential for low error rates and small distances. In particular for the circuit-level noise, the suppression deviates from exponential for larger distances. It is left for future work to construct a decoder that is efficient and accurate enough to perform reasonable simulations for larger decoding problems. This can involve designing a new decoder based on the considerations in the previous section, or decoding with a sliding-window decoder [72,87–90].

2. Stability experiment

We also perform stability experiments and show results in Fig. 18. In the phenomenological noise model, we simulate sizes 2 and 3. For size $s = 2$, we simulate 2, 4, 6, 10, and 16 measurement cycles. In all noise models, the logical error rate scales $\propto p^{n_{\text{cycles}}/2}$ for low physical error rates p . This implies that this timelike distance of the stability experiment coincides with the spacelike fault distance for

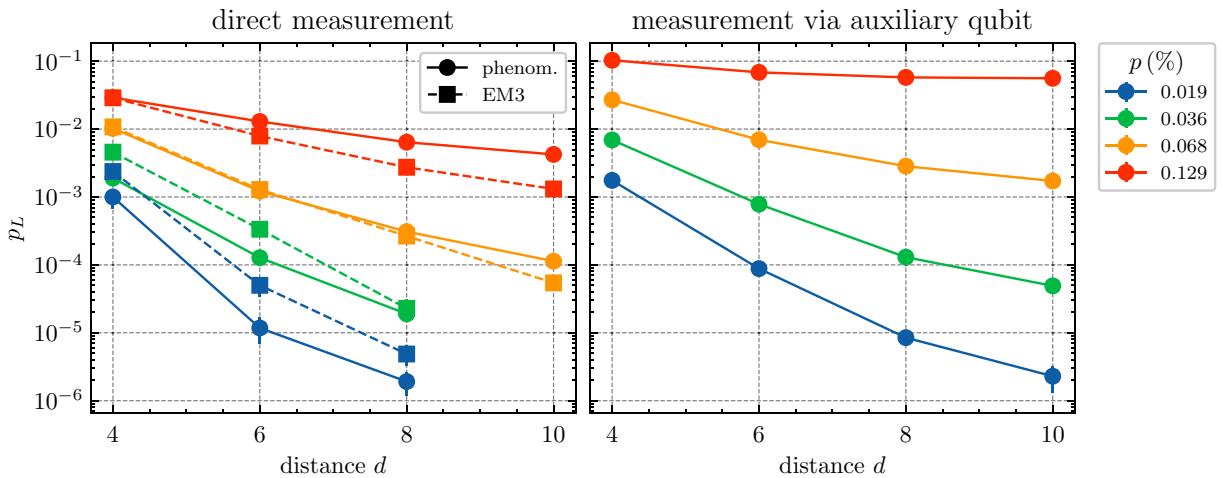


FIG. 17. Cuts for the $\mathcal{L}_Y$ memory experiment decoded with BP + LSD at different physical error rates p showing the suppression of logical error rate with code distance. Well below threshold, we observe an exponential suppression for small distances. We attribute the flattening of the curves, in particular, for circuits with auxiliary qubits, to nonoptimal decoding of zeroth-order BP + LSD, see the main text and Appendix D.

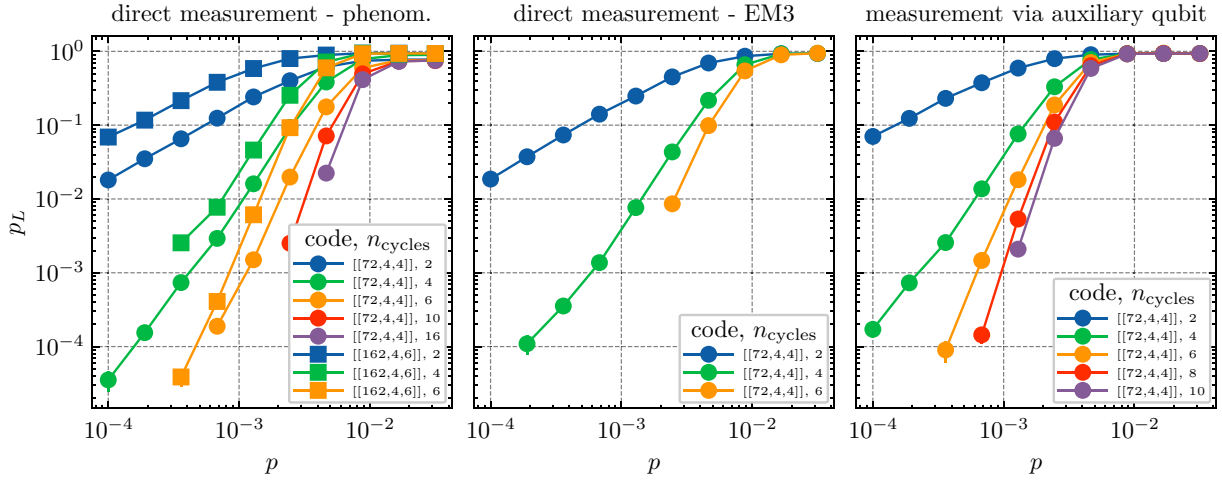


FIG. 18. Stability experiment of the $\mathcal{L}_Y$ stability observable for direct measurements using phenomenological and EM3 noise model, as well as measurements using an auxiliary qubit. In all noise models, the logical error rate scales $\propto p^{n_{\text{cycles}}/2}$ for low physical error rates p . This implies that this timelike distance of the stability experiment coincides with the spacelike fault distance for $n_{\text{cycles}} = 2s$.

$n_{\text{cycles}} = 2s$. This means concretely, that we can read off these experiments how many rounds of measurements are needed to reach a desired distance for the global observable. Or, more practically, how many rounds are needed to obtain a target logical error rate given a physical error rate. The global observables are typically measured in space-like parts of dynamical protocols, like moving a logical qubit or performing lattice surgery for entangling operations like logical CNOTs [91]. The stability experiments also allow us to explore how different error rates for data qubit and measurement errors influence the performance of the error-correction protocol, which we consider an interesting avenue for future research.

VI. TOWARDS PLANAR CODES

So far, we have considered topological codes with logical qubits coming from the nontrivial topology of the ambient manifold on which the model is defined. It is also possible to use *stable boundaries* to encode logical information robustly in a topological code. From the anyon perspective a stable boundary type is defined by a Lagrangian subgroup of anyons L , determining which anyons can condense at the boundary [92,93]. On the level of the stabilizer group and logical operators, the Lagrangian subgroup determines which string operators can be terminated at the boundary without leaving the codespace. In general, devising models for boundaries of a well-defined type is a nontrivial task. Specifically, for ISG codes, if a bulk measurement sequence transforms an anyon string operator s at time t into a string operator s' at time $t + 1$, a boundary where s can terminate at t has to transform into a boundary where s' can terminate at $t + 1$. At the same time, the measurement is not allowed to read-out the associated string operators. Moreover, a Floquet

code can apply a nontrivial automorphism to the logical operators after multiple rounds of measurements. As a consequence, (discrete) time-translation symmetry has to be broken at the boundary into a coarser symmetry. This leads to an enlargement of the periodicity of the schedule close to the boundary. If the automorphism applied after one cycle has finite order o , the period at the boundary has to be enlarged from T to oT .

For example, the honeycomb code is in a toric-code phase throughout the whole evolution. As such, it can have two boundary types, usually referred to as *rough* and *smooth*, condensing e and m anyons, respectively. Since it applies a $\mathbb{Z}_2$ automorphism after one length-3 cycle, exchanging the electric and the magnetic anyons of the underlying toric code, the boundaries have to be transformed into each after one cycle of measurements, which leads to a period doubling in the presence of boundaries [50].

Additionally to transforming according to the bulk evolution of the ISGs, a spacetime boundary has to be fault tolerant. In the case of topological protocols, this means that there are no constant-sized fault configurations that lead to a logical error. So far, boundary constructions have been understood well for some specific examples, with the most general framework that uses dynamical anyon condensation to understand $2 + 1$ -dimensional Floquet codes [19,20,50].

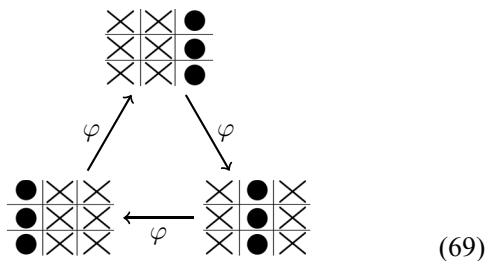
In the following, we will present two equivalent perspective on boundaries in the XYZ ruby code. First, we give a macroscopic description of how the ISGs have to transform in the XYZ ruby code in order to preserve the encoded logical information. Specifically, we pick $\mathcal{S}_0$ as our reference ISG and use the automorphism applied after one cycle of XX - YY - ZZ measurements to argue about how each of the six possible boundaries have to transform.

Second, we sketch an explicit method to construct boundaries for topological protocols using the spacetime tensor-network representation. A subset of boundaries obtained in this way for the XYZ ruby code can be found in Appendix B and are by construction fault tolerant. We want to note that this construction, in principle, works for any topological Clifford protocol, that is, for any protocol whose bulk is defined by local quantum operations and has a macroscopic fault distance with only local detector flows.

A. Tracking Lagrangian subgroups

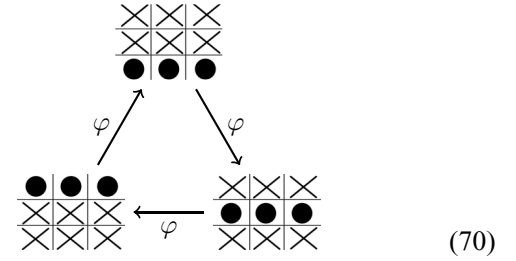
For our first analysis, we pick $\mathcal{S}_0$ as a reference ISG and assume we devised a schedule that has established a topological boundary at $t = 0$. Since $\mathcal{S}_0$ is in the phase of the two-dimensional color code (see Sec. III A) there are only six possible topological boundaries [41], three different “color” boundaries, where bosons of a given color in $\{\mathfrak{r}, \mathfrak{b}, \mathfrak{g}\}$ can condense, and three different “Pauli” boundaries, where bosons of a given Pauli label in $\{\mathfrak{x}, \mathfrak{y}, \mathfrak{z}\}$ can condense.

Assume the ISG at $t = 0$ realizes a boundary of type a described by a Lagrangian subgroup L_a . After three rounds of measurements, the anyon permuting automorphism φ from Eq. (24) is applied. This does not change the bulk stabilizer group [cf. Eq. (21)] but acts nontrivially on the stabilizer generators close to the boundary. On the macroscopic level, this changes the boundary type. To obtain the new boundary type we apply the automorphism to the Lagrangian subgroup and get a possibly changed boundary described by $L_{a'} = \varphi(L_a)$, where the function is understood to be applied elementwise. Since φ preserves the braiding data of the anyon model, $L_{a'}$ is again a Lagrangian subgroup but potentially of different type. We find that there is no boundary that is invariant under φ . There are, however, two transitive subsets of boundaries: Color boundaries get mapped to color boundaries and Pauli boundaries get mapped to Pauli boundaries. For example, φ maps a $\mathfrak{r}$ boundary to a $\mathfrak{b}$ boundary and the $\mathfrak{b}$ boundary to a $\mathfrak{g}$ boundary. Following the notation introduced in Ref. [41] for the different boundary types as rows, respectively, columns, of the color-code boson table [Eq. (9)] we can, for example, track the transformation of the color boundaries in the following way:



(69)

Similarly, the Pauli boundaries need to transform in the following way:



(70)

In the rewinding schedule, the automorphism is applied after three rounds—the first half the period—and then inverted throughout the second half.

B. Constructing boundaries in spacetime

Given the rather intricate transitions that the instantaneous boundaries need to undergo throughout one cycle of the rewinding schedule, a systematic construction of boundaries for topological protocols might seem out of reach. We find that this can be achieved using the RGB tensor-network representation of the protocol together with the Pauli flows. Specifically, we can construct a boundary of a given type by cutting the RGB network describing the bulk of the protocol along an arbitrary plane and consider the stabilizer state defined by the Pauli flows of the bulk. In this perspective, a boundary corresponds to terminating the tensor network along the cut with a (thin) RGB tensor network that attaches to the cut legs but has no additional open legs. The network that is attached along the cut has to be chosen such that it preserves the desired set of (logical) flows. Additionally, the stabilizers mapping between the logicals representatives have to be “read out” along the boundary to form the detector flows close to the boundary. In Appendix B, we present three distinct types of boundaries that can be combined to form a triangular code [94]. For the instantaneous color codes (at $t = 0 \pmod{3}$), these boundaries correspond to the three Pauli types discussed in the previous section. A more elaborate treatment of the construction is left for future work.

VII. REMARKS

In this section, we want to remark on some aspects of the XYZ ruby code and spacetime fault tolerance not touched on in the main part of this work. First, we compare the Floquet code to its associated subsystem code, introduced by Bombin in Ref. [28], and other topological subsystem codes. Secondly, we comment on how fault tolerance beyond topological codes can fit into the framework of Pauli flows and highlight some potentially interesting aspects of protocols defined via quantum operations of constant weight. Moreover, we comment on how the concept of Pauli flows of a protocol and the design principles

for a probing experiment discussed in Sec. V A are related to the recently introduced spacetime code construction for Clifford circuits [63]. Lastly, we comment on how rewrite rules of RGB tensor networks naturally lead to an equivalence class of protocols and highlight some interesting equivalences between protocols defined in different spatial dimensions. We argue that Pauli flows could play a central role in a topological classification of these equivalence classes and how such a classification, also beyond the Clifford case, could help in the construction of composable fault-tolerant protocols.

A. Comparison to topological subsystem codes

The group generated by all check operators of a Floquet code can be thought of as a gauge group of a subsystem code. We call this subsystem code the *associated subsystem code* [17]. One can understand the bare logical operators of a two-dimensional topological subsystem code as generating a 1-form symmetry associated to an anyon model [30].

The associated subsystem code of the XYZ ruby code is Bombin’s topological subsystem code constructed from a hexagonal lattice [28]. Its anyon model is the three-fermion model [28,29], a chiral anyon model. It attracted some attention in the community of topological phases since there cannot exist a gapped, commuting projector Hamiltonian on finite-dimensional degrees of freedom—and with that no stabilizer code—whose ground states represent a chiral anyon theory [36,40,95,96].

In contrast to the anyon model of the subsystem code, we find that the XYZ ruby code as a three-dimensional protocol, in fact, is described by the color-code anyon model, which is equivalent to two copies of the three-fermion anyon model and nonchiral. One can think of the logical qubits of the subsystem code as “static” and the additional logicals as “dynamical” qubits of the XYZ ruby code. For the subsystem code these dynamically generated qubits are gauge qubits. These gauge qubits, however, have logical operators of extensive size. We have chosen the measurement schedule such that the extensive gauge qubits are never fixed which renders them logical qubits of the protocol. In fact, any measurement schedule on the generators of the gauge group that reads out the stabilizers of the subsystem code that we investigated gives a protocol associated to the color-code anyon model [97].

In that sense, the anyon model of the subsystem code does not characterize the topological phase of the measurement protocol, as already popularized by the honeycomb code and its CSS counterpart [14,19,98], where the anyon theory of the associated subsystem code, respectively of its stabilizer group, is a proper subtheory of the toric-code anyon model. It would be interesting to investigate the phases arising in protocols derived from other subsystem codes representing chiral anyon theories, for example, chiral subsystem codes from Ref. [30].

From the QEC perspective it might also be interesting to investigate how different measurement schedules on the same set of checks affect the properties of the arising protocol. For example, the instantaneous topological phase transition present in our schedule seems like a fairly unique feature compared to other schedules, compare, e.g., with Ref. [22]. It is an open question if this has implications on the error-correcting capabilities of the schedule.

B. Fault tolerance in nontopological protocols

In the sections above we have mainly discussed topological protocols as they are central to our analysis. However, both memory and stability experiments can be thought of in the context of nontopological protocols. Here, we define topological protocols as error-correcting protocols that admit a complete set of local detector flows. That means we consider a family of QEC circuits to be topological if they give rise to a family of locally generated detector flows such that the fault distance—with respect to that set of detectors—grows with the number of physical qubits. In these cases, there is a natural partition of detector flows of any probing experiment into local and nonlocal flows, as described in the subsections above. Moreover, for topological protocols there has been established a thorough understanding of how to understand fault tolerance with respect to local noise [18,99–101].

One might ask in which sense fault tolerance can be understood generically in the case of nontopological protocols. i.e., given an experiment with its flows, can one make rigorous statements about the fault tolerance with respect to a given error model? And more importantly, can we argue about fault tolerance based on the structure of the detector flows of the protocol, say, e.g., their weight distribution and a guarantee on the fault distance? The natural generalization of “locality” of detector flows is obtained by ignoring their geometric locality but considering only the scaling of the number of edges that are nontrivially highlighted by the flows within the a family of protocols. If one can choose a set of detector flows of bounded weight to define syndromes and correct all observables (with extensive weight), we propose to call the corresponding protocol “spacetime LDPC” [102]. As far as we know there is no good reason to believe that this notion of spacetime LDPC is similarly important to fault tolerance as it is believed to be for an instantaneous stabilizer code. Importantly, quantum operations of constant weight can lead to protocols that are not LDPC. An interesting example in that regard is a recently proposed (modified) schedule for the Bacon-Shor code that is believed to have a threshold but does not fall into that category, see Ref. [103]. Even though most detector flows used for decoding have low weight, the fractal structure of the protocol creates some high-weight detector flows and hence a decoding graph with unbounded check weight. In light of the fact that the

previously considered (unmodified) schedule associated to the Bacon-Shor code only has detectors of extensive weight and is not thresholded we consider the question if there has to exist a minimum amount of constant-weight detector flows in order to be fault tolerant an interesting question for further research. It would also be interesting to investigate the structure of the detector flows in protocols based on fracton codes [104,105] to inspire new and in particular efficient decoding strategies for type-II fracton models. The more exotic structure of the resulting decoding graph could reveal insights into the origin of fault tolerance beyond more well-behaved topological codes. The example above indicates that detector flows do not need to be of strictly bounded weight in order to achieve fault tolerance with a protocol based on bounded-weight quantum operations.

C. Connection of Pauli flow to outcome and spacetime codes

In Ref. [63], Delfosse and Paetzniak introduce the *outcome code* of a Clifford circuit. It is a classical affine linear code on the space spanned by all measurement outcomes. In fact, we recover similar constraints on the set of signs in the network based on detector flows. In our formulation, we end up with a linear code because we include signs of tensors that might flip measurements also without noise (e.g., Pauli unitaries). The way that the outcome code is translated to form a linear code in Ref. [63] is equivalent to identifying these operations and flipping the measurements accordingly, i.e., inverting the affine translation with which a linear code is translated to obtain the outcome code. In Appendix A 3 d, we give a rigorous derivation of the linear code obtained from detector flows.

Additionally, the authors of Ref. [63] introduce the *spacetime code*, a quantum code derived from the circuit and the outcome code. One can understand the stabilizers of the spacetime code directly from the tensor-network picture. The spacetime code is defined with respect to a time direction, i.e., an interpretation of the tensor network as a sequential application of stabilizer quantum instruments on a set of qubits. Given that interpretation, we can identify each detector flow with a stabilizer of the spacetime code. Specifically, each stabilizer is the product of Pauli operators associated to the highlights of the detector flow on the edges associated to qubit worldlines [106].

For a fixed circuit and error model, the spacetime code gives a faithful description of the behavior of faults in the QEC protocol. The commutation relation of a fault with a stabilizer of the spacetime code are exactly given by the charge of the fault with respect to the associated detector flow.

More generally, Pauli flows carry more information. Since they are preserved by all rewrite rules of the network, they are independent of the interpretation of the

network. For example, consider two circuits A and B that implement the same stabilizer measurement. In the RGB tensor-network formulation any such pair of equivalent circuits is related by a rewrite rule presented in Appendix A 2. Pauli flows are a preserved quantity in the sense that we can update the flows locally with each rewrite without computing the full detector flow and the associated constraints on the signs explicitly, see Appendix A 4. In that way, there is a direct translation between these quantities in the RGB tensor-network formulation. Note that this includes mappings between protocols based on static stabilizer codes, dynamical Floquet-style readout schemes as well as measurement-based computation schemes. It would be interesting to see if one can extend the formalism in Ref. [63] to construct direct mappings between the outcome and spacetime codes of circuits related to the same underlying tensor network.

In a fault-tolerant architecture it makes sense to think about logical blocks, specific building blocks from which any logical operation is built out of. We think of them as being expressed in terms of protocols on the physical level that have an effective logical action. When composing logical Clifford blocks it is natural to understand the Pauli flows of the combined block in terms of the Pauli flows of the individual blocks. In the spacetime code, it is not that obvious how to combine the spacetime codes of two logical blocks (see also Lemma 3). We see the reason for that in the fact that there is no direct analog of stabilizer and logical flows in the spacetime code. They can be mapped to a certain subset of logical operators of the spacetime code but there are many more logical operators that have no direct interpretation in the composition of logical blocks as they act purely internal to the network, respectively, circuit. It would be interesting to see a more direct way to incorporate the composition of logical blocks into the spacetime code formalism.

D. Equivalence classes of protocols and associated invariants

We want to finish on a more general thought arising from considering Pauli flows in RGB tensor networks.

One can use certain rewrite rules of RGB tensor networks, presented in Appendix A 2, to define a sensible equivalence relation between RGB tensor networks. Under that equivalence relation we find that many protocols are equivalent even though they live in different spatial dimensions. A well-studied example of that is the equivalence between the fault-tolerant identity gate in *measurement-based quantum computation* (MBQC) with the three-dimensional cluster state and the $2 + 1$ -dimensional protocol based on the surface code on a square lattice as illustrated in Ref. [40].

One can make this equivalence much more explicit and rigorous and concrete by viewing a MBQC scheme as

a full protocol where qubits are initialized in a product state, a constant depth Clifford circuit is applied to prepare the resource state and single-qubit measurements are performed. If the preparation circuit is local, and constant depth, in some dimension D , the TN of that protocol is equivalent to a TN of a protocol over time composed of operations that are local in $D - 1$ dimensions by treating one spatial direction as the time direction and reinterpreting qubit worldlines accordingly. Vice versa, any protocol defined by a tensor network that is local in some spacetime dimension D can be rewritten and reinterpreted as a constant-shot (MBQC) protocol. A similar connection was made in Ref. [101] and also applies to other three-dimensional stabilizer states such as states of the three-dimensional toric code [107], 3D color code [27], or topological single-shot protocols [108,109] that can, in principle, be identified with $2 + 1$ -dimensional protocols.

The striking feature of Pauli flows is that one can think of them as being preserved throughout the process, giving rise to an invariant of an equivalence class of protocols. A direct consequence of that is that the logical isomorphism applied by a protocol as well as the input and output stabilizer groups are preserved by all rewrites. Finding an (algebraic) invariant that captures the fault-tolerance properties of a protocol, possibly with respect to some “unit volume” would constitute a big step in the understanding of fault tolerance beyond individual (classes of) protocols and in defining a “phase” of a stabilizer tensor network.

These invariants would also be important when combining logical blocks. Given one block with invariants A and one block with invariants B , can one make statements about the invariant of the composed block? So far, we only know how to explain this for Clifford protocols in terms of Pauli flows and techniques presented in Appendix A. A more general algebraic treatment of the Clifford case could inspire new perspectives on the understanding of fault tolerance for non-Clifford operations.

VIII. CONCLUSION AND OUTLOOK

In this work, we have established a Pauli flow formalism, which unifies an error-correction analysis for protocols based on Clifford circuits and Pauli measurements. The formulation in terms of a tensor network leads to a well-defined equivalence between different protocols and circuits that share their fault-tolerance properties and also provides a systematic procedure to design various implementations of the same logical operation with a clear handle on the error-correcting properties. The concept of Pauli flow emerges from combining projective symmetries of the elementary building blocks of an RGB tensor network consistently to obtain symmetries of the full network. Defining the notion of tensors being charged with respect

to such a symmetry allows us to tie together the design and fault-tolerance analysis of a protocol directly in spacetime. Specifically, we can rigorously derive a unified perspective on properties of an error-correcting protocol from first principles within the tensor-network representation. This goes beyond the specific representation of a protocol in terms of the RGB tensors and applies to all Clifford protocols, independent of the underlying paradigm of quantum computation. With this, we tie together existing notions on error correction in spacetime [31,61,63] that have taken different approaches, where either a family of a topological protocol or explicit circuits take center stage. We do this by proving very general algebraic properties of stabilizer tensor networks that are not tied to specific implementations or even error-correcting operations. Although a tensor-network representation is very natural for topological protocols our analysis on how to think about QEC in spacetime does not require any (spatial) locality. Hence, we believe it is helpful to investigate any active QEC protocol through the lens of a tensor-network representation of the full spacetime evolution of the system. Formulating the tensor network in terms of RGB tensors allows for additional flexibility as local rewrite rules can be used to turn non-fault-tolerant schemes into fault-tolerant ones. This makes this method appealing for both code design and fault-tolerant circuit compilation.

To make this case, we have constructed a new dynamical error-correcting code based on repeated measurements of two-body operators, which we call the XYZ ruby code. We have analyzed the dynamic protocol using “traditional” methods and have shown how the graphical calculus allows us to think of the protocol directly in spacetime and devise the algebraic object required for error correction and fault-tolerance properties. We have further used the RGB tensor networks to construct fault-tolerant protocols and circuits to probe the memory and stability qualities of the XYZ ruby code under different noise models. We have explained how to generate and decode a decoding graph from the detector flows and have developed a thorough physical understanding of faults and their action on the detectors. We have performed numerical experiments, decoded using general-purpose decoders, and have found that the XYZ ruby codes show a threshold in the vicinity of $0.18 \pm 0.01\%$ for full circuit-level noise. We believe that this could be further improved upon by designing decoders specifically tailored for the code using, e.g., restriction or matching-based approaches. Even as such, the thresholds are comparable to and competitive with other Floquet and non-Floquet codes in the literature. In particular, static color codes require contrived syndrome readout constructions using, e.g., flag qubits, whereas our dynamical version relies on two-body nearest-neighbor measurements only.

The locality and performance of XYZ ruby codes places them as promising candidates for near-term fault tolerance.

Additionally, the XYZ ruby code allows for more transversal gates than Floquet codes realizing toric codes [22], because it is in the topological phase of a color code. Logical gates implemented transversally are naturally fault tolerant and introduce a very limited amount of noise in the protocol [13,27,110]. Similarly, the small weight of measurements in Floquet codes limits the spreading of faults. With properly introduced boundaries, one can design logical gates based on lattice surgery protocols, which rely on fault-tolerant joint logical measurements between different code patches [91,111,112]. For the honeycomb code, one way to perform these joint measurements was shown in Ref. [50]. We have sketched how boundaries could be introduced in the XYZ ruby code. Important next steps include the microscopic realization of these boundaries and design of measurement sequences that join multiple copies of the code along those boundaries to perform said joint logical measurements.

More generally, Pauli flows offer a versatile tool to design interfaces between logical blocks in spacetime without resorting to a specific code on which the protocol could be based on. In particular, Pauli flows allow one to systematically join microscopically different blocks with little overhead. For topological protocols these interfaces can be described abstractly as topological defects in a topological spacetime. Analyzing how other types of defects, such as twist defects, can be described locally in terms of Pauli flows constitutes an interesting avenue for further research. We believe that this connects to formulations of fault-tolerant logical gates within a measurement-based computation scheme based on Walker-Wang models enriched with external symmetry defects [113,114]. We hope that, once we understand how to construct these defects explicitly within the bulk of a tensor network, we can use this insight to design similar defects in more general classes of protocols that involve nonlocal operations. This would extend our understanding of how to implement fault-tolerant logical operations to protocols based on QLDPC codes that offer better distance and encoding rates.

To achieve computational universality, fault-tolerant logical non-Clifford operations are needed. Towards a more general understanding of those, we think it is important to investigate non-Pauli symmetries in our tensor-network formulation.

We hope that Pauli flows in RGB tensor networks and the new competitive dynamical QEC code will contribute to inspire further research and drive the progress towards scalable fault-tolerant quantum computing.

ACKNOWLEDGMENTS

We want to thank M. Kesselring for comments and ideas on constructions of Floquet codes in the color-code phase. J.M. wants to thank A. Bauer for fruitful discussions

on tensor-network representations of topological protocols and D. Williamson, T. Ellison, J. Conrad, T. Scruby, S. Roberts, and T. Hillmann for stimulating conversations. Moreover, J.M. wants to thank R. Raussendorf for bringing up the usage of Pauli symmetries in Ref. [115] to our attention. We gratefully acknowledge support by the European Union’s Horizon Europe research and innovation programme under Grant Agreement No. 101114305 (“MILLENION-SGA1” EU Project). This research is also part of the Munich Quantum Valley (K-8), which is supported by the Bavarian state government with funds from the Hightech Agenda Bayern Plus. We additionally acknowledge support by the BMBF project MUNIQCATOMS (Grant No. 13N16070). We also acknowledge support for the research that was sponsored by IARPA and the Army Research Office, under the Entangled Logical Qubits program through Cooperative Agreement Number W911NF-23-2-0216. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of IARPA, the Army Research Office, or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation herein. J.M. and J.E. acknowledge additional support from the DFG (CRC 183), the ERC (DebuQC), the BMBF (QSolid, RealistiQ), the Quantum Flagship (PasQuans2, Millenion), and Berlin Quantum. M.M., J.O., and M.R. acknowledge support by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany’s Excellence Strategy “Cluster of Excellence Matter and Light for Quantum Computing (ML4Q) EXC 2004/1” 390534769 and the ERC Starting Grant QNets through Grant No. 804247. The authors gratefully acknowledge the computing time provided to them at the NHR Center NHR4CES at RWTH Aachen University (Project No. p0020074). This is funded by the Federal Ministry of Education and Research and the state governments participating on the basis of the resolutions of the GWK for national high-performance computing at universities.

J.M. and J.O. led this project. They jointly analyzed the XYZ ruby code as a Floquet code and as an RGB tensor network and devised the probing experiments. J.M. had the idea to formalize Pauli flows based on projective symmetries and devised the proofs in the Appendix. A.T.T., J.M., and J.O. jointly worked out the details of the formalism. A.T.T. contributed to the design of the QEC experiments, the boundaries presented and gave important feedback on the draft at many stages. J.O. performed all the numerical experiments, designed the circuits, and implemented the code, with feedback from M.R. J.E. and M.M. contributed to shaping the direction of the project and contributed in the writing of the paper.

APPENDIX A: RGB TENSOR NETWORKS AND PAULI FLOW

In this section, we give some additional properties and relations for *RGB tensor networks* (RGB TNs). We will use

the definitions introduced in Sec. IV. Equation (38) shows how the blue tensor can be seen as the red, respectively green, tensor in a different local basis. Using unitarity of the basis transformations,

$$\text{---} \square \square \text{---} = \text{---} \bigcirc \bigcirc^* \text{---} = \text{---} \diamond \diamond^* \text{---} = \text{---} \quad (\text{A1})$$

we find similar relations for red and green tensors,

(A2)

1. Representing any Pauli measurement as an RGB TN

Consider a system of n qubits. Projective measurements of an operator O project the system into a definite eigenspace of O , associated with the image of a measurement outcome labeled by a projector P_m . Specifically, starting with $|\psi\rangle$, the postmeasurement state, knowing the outcome m , is proportional to $P_m |\psi\rangle$. Within our graphical formalism we represent any Pauli measurement with a binary outcome $m \in \{0, 1\}$ with the associated projector. For a generic n -qubit Pauli operator $P_1 \otimes P_2 \otimes \cdots \otimes P_n$ this projector reads

$$P_m = \frac{1}{2} (1 + (-1)^m P_1 \otimes P_2 \otimes \cdots \otimes P_n). \quad (\text{A3})$$

In the following, we describe a recipe of how to construct the graphical representation of that projector with an RGB TN.

- (1) Each qubit comes with its own worldline, which we represent by a line,

$$n \left[\begin{array}{c} \text{---} \\ \text{---} \\ \vdots \\ \text{---} \end{array} \right] \quad (\text{A4})$$

We identify the left-hand side with the “input” and the right-hand side with the “output.” This can be motivated by interpreting that diagram as representing an identity operator on $(\mathbb{C}^2)^n$, mapping from the left to the right.

- (2) Introduce a red m tensor for the measurement,

(A5)

- (3) Place an $s = 0$ tensor on each qubit wire taking part in the measurement. The color of the tensor is determined by the Pauli type of the measurement operator on that qubit (X , red; Y , green; Z , blue). For the green tensor we have to “flip” the right-pointing output to represent the correct measurement. For example, for a $X \otimes Y \otimes \cdots \otimes Z$ measurement, we place the following tensors:

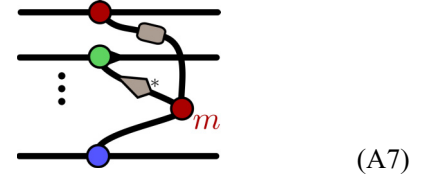
Diagram (A6) illustrates a multi-qubit system. It consists of four horizontal black lines representing qubits. The top qubit has a red circle on it, the second has a green circle, and the bottom has a blue circle. A vertical ellipsis between the second and third lines indicates more qubits. To the right of the lines, there is a red circle labeled m . The label (A6) is at the bottom right.

By flipping the input of the green tensors where the wire attaches from the right we have chosen to interpret the resulting diagram as a map from the open wires on the left to open wires on the right. Specifically, the projector of a measurement of a Pauli operator that includes a Y operator is not generically symmetric (as a linear operator) so the diagram cannot be mirror symmetric either.

- (4) Finally, we connect the newly introduced tensors to the measurement tensor. Depending on the colors

of the tensors that are connected, we place an additional basis transformation on the connecting wire. A blue tensor is connected to the measurement tensor with a simple wire. A red tensor is connected via the basis transforming tensor that maps it to the blue tensor (in this case, the Hadamard box). Similarly, the green tensor is connected via the basis transforming tensor that maps it to the blue tensor. For

the $X \otimes Y \otimes \dots \otimes Z$ example above, this yields



(A7)

Single-qubit measurements can be viewed as projections onto eigenstates of single-qubit Pauli operators, i.e., initializations of single-qubit Pauli eigenstates. As such it is helpful to directly represent these as single-legged tensors,

$$|s\rangle \sim \text{red circle with a line}, \quad |(-1)^s\rangle \sim \text{blue circle with a line} \quad \text{and} \quad |(-1)^s i\rangle \sim \text{red circle with a line and a dot}.$$

(A8)

2. Putting the calculus onto the three colors

In the main text, we have focused on constructing an RGB TN given an explicit circuit, i.e., sequence of gates and then using this representation to analyze the QEC properties of that circuit. However, we find that RGB TNs have *local rewrite rules* that allow one to locally rewrite a given RGB TN without changing the global action, i.e., the linear operator it represents. These types of rewrite rules induce an equivalence relation amongst RGB tensor networks. These equivalences are a core part of the ZX calculus [26], where the central objects are equivalence classes of diagrams and not individual diagrams. One could say that these equivalences constitute the “calculus” part of the ZX calculus. Reference [31] showed that (the bulk of) a fault-tolerant measurement-based quantum computation protocol, the CSS Floquet honeycomb code [19,98] and the two-dimensional surface code are equivalent in exactly that sense. In Ref. [17], these rewrite rules were used to “Floquetify” the two-dimensional color code, i.e., construct a Floquet code that only uses one- and two-body measurements that is equivalent to the protocol where the stabilizer generators of the hexagonal color code are periodically measured. These equivalences are

also useful when considering individual circuit components. For example, Ref. [116] used these equivalences to improve read-out circuits for the square surface code. All these examples show that such equivalences are a desirable feature for a graphical representation of any circuit.

Since the ZX calculus is complete [26], we could express our green tensor in terms of ZX tensors and use the ZX rewrite rules. We argue, however, that working with more direct relations involving only green tensors themselves is more helpful when actually working with RGB networks. In the following, we present these relations for all three types of tensors [117]. Let us start with the blue tensors. A straightforward tensor-network calculation shows that they obey the following relations:

(A9)

Using the basis transformations from Eq. (A2) and unitarity, Eq. (A1), this implies that the tensors of the other two colors fulfill the following relations:

(A10)

Note that when splitting and fusing the green tensors, the orientation of the input and output is important. These relations hold for any number of legs.

Using Eq. (A2) and the (projective) symmetries of the blue tensor, shown in Eq. (48), we obtain similar symmetries for the tensor of the other two colors,

$$\text{Red tensor with } s \text{ legs} = \text{Red tensor with } s \text{ legs} = (-1)^s \text{Red tensor with } s \text{ legs} \quad \text{and} \quad (\text{A11a})$$

$$\text{Green tensor with } s \text{ legs} = \text{Green tensor with } s \text{ legs} = (-1)^s \text{Green tensor with } s \text{ legs}. \quad (\text{A11b})$$

Additionally, they fulfill a *generalized bialgebra rule* for the Z and X tensors [118],

$$\text{Network of red and blue tensors} = \text{Simplified network} \quad (\text{A12})$$

Applying the basis transformation rules (A2) we can derive similar rules involving the green (Y) tensor. Importantly, this particular rule is not fully symmetric with respect to permuting the colors.

There is a second set of rewrite rules of the form

$$\text{Loop rules for red, blue, and green tensors} \quad (\text{A13})$$

which allow one to remove or add loops into the network. Often, we do not care about constants appearing via such contractions and think about a normalized version of each network. Note that for the first rule shown above we can use Eqs. (38) and (A2) to obtain a similar relation involving a green tensor. This will, however, have some basis transformations on contracted legs so the rule is not fully symmetric among the three colors.

Another straightforward calculation shows how we can graphically see that single-qubit Pauli measurements initialize a qubit in a given Pauli eigenstate,

$$\text{Measurement rules for red, blue, and green tensors} \quad (\text{A14})$$

Note that in this section we do not give a minimal nor a complete set of graphical rules but we solely aim to gather the most useful rules when considering analyzing and working with active QEC schemes.

a. Example: Teleportation circuit

Here, we show how a naïve representation of the teleportation circuit shown in Eq. (51) is rewrite equivalent to

Including a preparation circuit of the Bell pair on the left we start with the following network representing the teleportation circuit, which we can rewrite in two steps in to the diagram above:

$$\text{Teleportation circuit diagram} \quad (\text{A15})$$

where we first used the splitting rules and then used that for each type of tensor, its two-legged version with $s = 0$ corresponds to the identity matrix, and, can just be removed. The diagram on the right is exactly the same as the one in Eq. (A15) by construction as a tensor network, which is independent on how the lines are drawn on the paper.

3. Physical meaning of Pauli flows: A logical isomorphism protected by a classical code

In the main text, we introduced a set of colorings of edges of an RGB tensor network that represents how Pauli operators can propagate through the network. Here, we want to make the meaning of these highlights more precise in terms of their associated constraints and stabilizer groups on the input and output qubits.

First, we want to complete the rules of highlighting introduced in the main text [see Eq. (50a)] to all components of an RGB network. Specifically, we have not yet defined a highlight for the basis transformations, defined in Eq. (37). Calculating how these tensors map between different Pauli types,

$$\text{---} \square \text{---} = \text{---} \text{---} \square \text{---} = \text{---} \text{---} \square \text{---}, \quad (\text{A17a})$$

$$\text{---} \text{---} \square \text{---} = \text{---} \text{---} \square \text{---} = \text{---} \text{---} \square \text{---}, \quad (\text{A17b})$$

$$\text{---} \triangle \text{---} = \text{---} \triangle \text{---} = \text{---} \triangle \text{---}, \quad (\text{A17c})$$

we obtain the following generators of “valid” highlights for these elements:

$$\begin{array}{cc} \text{---} \square \text{---} & \text{---} \square \text{---} \\ \text{---} \text{---} \square \text{---} & \text{---} \text{---} \square \text{---} \\ \text{---} \triangle \text{---} & \text{---} \triangle \text{---} \end{array}. \quad (\text{A18})$$

a. Setting the stage

Consider an n -qubit Pauli group $\mathcal{P}_n$, $n \in \mathbb{N}$, and a bipartition of n into n_A and $n_B = n - n_A$ qubits. We can write $\mathcal{P}_n = \mathcal{P}_{n_A} \otimes \mathcal{P}_{n_B}$. In the following, we show that a stabilizer group $S \leq \mathcal{P}_n$ of rank n defines a unique isomorphism between a logical Pauli group embedded into $\mathcal{P}_{n_A}$ and $\mathcal{P}_{n_B}$. If clear from context, we denote a single-qubit Pauli operator $P \in \mathcal{P}_1$ on qubit i with P_i . We denote the center of a group $G \leq \mathcal{P}_n$ by $Z(G) = \{x \in G \mid xg = gx \ \forall g \in G\}$ and its centralizer over the Pauli group by $C(G) = \{p \in \mathcal{P}_n \mid pg = gp \ \forall g \in G\}$. Note that we omit the subscript in the centralizer whenever we consider centralizers over the Pauli group. Sometimes we will refer to a “real Pauli group.” With that, we denote the subgroup of the Pauli group that only has real entries, generated by all X and Z operators. Note that Y is not directly included in that group but only $\pm iY$.

Lemma 1. Consider the space of n qubits with a bipartition $\mathbb{C}^{2^n} = A \otimes B$ with $A \simeq \mathbb{C}^{2^{n_A}}$ and $B \simeq \mathbb{C}^{2^{n_B}}$. Let $S \leq \mathcal{P}_n$ be a rank- n stabilizer group on $A \otimes B$. Let $S_A = \{a \mid a \otimes$

$\mathbb{1} \in S\}$ be a group isomorphic to the maximal subgroup of S solely supported on A and analogously S_B , the maximal subgroup of S supported on B . Then S together with a bipartition (A, B) defines a group isomorphism

$$\varphi_S(A, B) : N(S_A)/S_A \rightarrow N(S_B)/S_B, \quad (\text{A19})$$

where S gives rise to a complete set of representatives of both $N(S_A)$ and $N(S_B)$. Vice versa, any such isomorphism defines a stabilizer group on $A \otimes B$ of maximal rank.

We remark that this holds for any bipartition and holds due to the maximality of the rank of S .

Proof. The main object in our proof will be the group generated from truncated stabilizers, such as $S|_A = \langle \{a \mid a \otimes b \in S\} \rangle$ and $S|_B = \langle \{b \mid a \otimes b \in S\} \rangle$. By including all operators generated from the truncated stabilizer group we might add $-\mathbb{1}$ to the group due to noncommutativity of restricted stabilizers. Let $S_A = \{a \mid a \otimes \mathbb{1} \in S\}$ be a group of operators on A that is isomorphic to the maximal subgroup of S supported on A and similarly S_B for B . Since S is a stabilizer group, S_A is in the center of $S|_A$. Hence, we can decompose $S|_A$ as [119]

$$S|_A \simeq S_A \times S|_A/S_A. \quad (\text{A20})$$

In fact, S_A is the center of $S|_A$ up to operators proportional to the identity, $\langle i\mathbb{1} \rangle$, due to the maximality of the rank of S . We can see this by taking an element $a' \in Z(S|_A) = \{x \in S|_A \mid xs = sx \ \forall s \in S|_A\}$. It commutes with $a \in S|_A$, hence, defines an element $a' \otimes \mathbb{1}$ that commutes with S and hence, due to its maximality, must be included in S . By definition of S_A , $a' \in S_A$.

Hence, $Z(S|_A/S_A) \subseteq \langle -\mathbb{1}, S_A \rangle$. In fact, this inclusion is an equality if and only if $S|_A$ includes anticommuting Pauli operators, i.e., S_A has nonmaximal rank (on A).

$S|_A/S_A$ admits a generating set of pairwise anticommuting operators [120]. As such, it is isomorphic to the group of logical Pauli operators with real entries, i.e., $S|_A \simeq N(S_A) \cap \langle \{X_i, Z_i\}_{i=0}^n \rangle$. The same applies for $S|_B$, the stabilizer group truncated to the qubits in B .

a. Construction of isomorphism. Let $S_A \otimes S_B = \{a \otimes b \mid a \in S_A, b \in S_B\}$. Consider the quotient group $S/(S_A \otimes S_B) = \{xS_A \otimes yS_B \mid x \otimes y \in S\}$. Note that we can write any element in $S/(S_A \otimes S_B)$ as $xS_A \otimes yS_B$ for $x \in S|_A$ and $y \in S|_B$. From this we define the map

$$\begin{aligned} \varphi_S(A, B) : S|_A/S_A &\rightarrow S|_B/S_B, \\ xS_A &\mapsto yS_B \quad \text{for } xS_A \otimes yS_B \in S/(S_A \otimes S_B). \end{aligned} \quad (\text{A21})$$

In the following, we show that this is a well defined group isomorphism.

First, we show that it is well defined, i.e., there exists no two cosets $yS_B \neq y'S_B$ such that $xS_A \otimes yS_B, xS_A \otimes y'S_B \in S/(S_A \otimes S_B)$ for some $x \in S|_A$. Assume $\exists y, y' \in S|_B$ such that $yS_B \neq y'S_B$. Since $S_A \otimes S_B$ is central in S the cosets $S/(S_A \otimes S_B)$ admit a group structure. This implies that $S_A \otimes yy'^{-1}S_B \in S/(S_A \otimes S_B)$, i.e., $\mathbb{1} \otimes yy'^{-1} \in S$. By definition of S_B , $yy'^{-1} \in S_B$ and we obtain $yS_B = y'S_B$, contradicting our assumption. This shows that φ is well defined.

Let us tweak the argument slightly to see that φ is in fact injective. Assume it was not. This means $\exists x \neq x' \in S|_A$: $xS_A \otimes yS_B, x'S_A \otimes yS_B \in S/S_A \otimes S_B$. Again by the group structure of the cosets, this implies $xx'^{-1}S_A \otimes S_B \in S/S_A \otimes S_B$ and $xx'^{-1} \in S_A$. Which contradicts the assumption $xS_A \neq x'S_A$. Surjectivity of φ can be shown similarly. By construction via a quotient group, for any $yS_B \in S|_B/S_B$ there exists a coset $xS_A \otimes yS_B \in S/S_A \otimes S_B$ for some $x \in S|_A$ and with that $yS_B \in \text{Im}(\varphi_S)$. This shows that φ is a bijection.

It remains to be shown that φ_S is a group homomorphism. This is straightforward to calculate using the centrality of S_A and S_B in $S|_A$ and $S|_B$. Let $x \otimes y, x' \otimes y' \in S$. Then

$$\varphi_S(xS_A)\varphi_S(x'S_A) = yS_By'S_B = yy'S_B = \varphi_S(xx'S_A), \quad (\text{A22})$$

where we used centrality of S_B and the group structure on $S/S_A \otimes S_B$.

We can extend φ_S to act on a full logical Pauli group $N(S_A)/S_A$ by letting it act like the identity on all phases $\langle i \rangle$, the factors by which $S|_A$ differs from $N(S_A)$, i.e., we define $\varphi(i^m xS_A) = i^m \varphi(xS_A)$ for any $m \in \mathbb{Z}, x \in S|_A$. This completes the proof that φ is an isomorphism.

b. Construction of stabilizer group. To construct a maximal rank stabilizer group from a given isomorphism, we reverse the construction above. Consider isomorphism φ' between logical Pauli groups, on given codes, defined by stabilizer groups S_A and S_B of rank $n_A - k$ and $n_B - k$, respectively. Consider the subgroup of the domain of φ' that is generated by cosets that act like X and Z operators, i.e., it does not contain the logical $-\mathbb{1}$ and any imaginary logical Pauli and denote it with $\tilde{P}_{\varphi'}$. We use φ' to construct cosets $xS_A \otimes \varphi'(xS_A)$, for $xS_A \in \tilde{P}_{\varphi'}$. Take a complete set of representatives $R_{\varphi'} = \{x_i \otimes y_i\}_{i=1}^{2k}$ for these cosets. Since φ' is a group isomorphism, they generate a stabilizer group $S_{AB} = \langle R_{\varphi'} \rangle$ on $A \otimes B$ that is in the centralizer of both $S_A \otimes \mathbb{1}$ and $\mathbb{1} \otimes S_B$. We define the stabilizer group associated to φ' as

$$S_{\varphi'} = \langle S_A \otimes \mathbb{1}, S_B \otimes \mathbb{1}, S_{AB} \rangle \simeq S_A \times S_B \times S_{AB}. \quad (\text{A23})$$

It remains to be shown that it is of maximal rank. Since every factor above is independent, the ranks add up and we obtain

$$\text{rank}(S_{\varphi'}) = \text{rank}(S_A) + \text{rank}(S_B) + \text{rank}(S_{AB}). \quad (\text{A24})$$

Let n_A be the number of qubits in A and n_B be the number of qubits in B . We can use the fact that the isomorphism is on k logical qubits to obtain $\text{rank}(S_A) = n_A - k, \text{rank}(S_B) = n_B - k, \text{rank}(S_{AB}) = 2k$. Plugging this into Eq. (A24) yields $\text{rank}(S_{\varphi'}) = n_A - k + n_B - k + 2k = n_A + n_B$, which is the maximal rank of a stabilizer group on a system of $n_A + n_B$ qubits. ■

Note that this can be generalized straightforwardly to Pauli stabilizer groups on qudits. In that case, one should take the adjoint on the B factor to not overcomplicate things in the interpretation of the resulting stabilizer group. Besides that, the proof goes through in the same way, as long the qudit dimension is finite.

c. An algebraic perspective. Before getting into more details on how to identify an RGB tensor network with an isomorphism above, we want to comment on how to understand the above isomorphism analogously on the level of a full “logical algebra,” the algebra of matrices acting within the logical subspace. This is not too important for any logical Clifford operation but should be considered when generalizing logical operations beyond Clifford operations in a unified manner. Once a non-Clifford operation is applied, the full evolution of the system cannot be described by any Pauli group isomorphism anymore but should be described by a $*$ -isomorphism on a logical algebra. In the following, we want to sketch how the Clifford case and a logical group isomorphism fits into this algebraic perspective.

Consider a Pauli stabilizer group $S \leq \mathcal{P}_n$ and its normalizer group $N(S) = \{p \in \mathcal{P}_n | pS = Sp\}$. Both S and $N(S)$ are sets of linear operators on a finite-dimensional vector space that are closed under multiplication. As such, they span a $*$ -algebra where the matrix adjoint † , transposing and complex conjugating, plays the role of the $*$ -operation. Let $\mathcal{S} = \text{span}_{\mathbb{C}}(s \in S)$ be the commutative $*$ -algebra spanned by the stabilizer group. Since for Pauli stabilizer groups, the normalizer equals the centralizer, $N(S) = C(S) = \{p \in \mathcal{P}_n | ps = sp \forall s \in S\}$, it spans the commutant

$$\mathcal{C}(S) = \text{span}_{\mathbb{C}}(p \in C(S)). \quad (\text{A25})$$

Note that any element in that algebra acts the same within one “syndrome sector,” a subspace that transforms irreducibly under the action of $\mathcal{S}$ [121]. Within the trivial

syndrome sector, the space of states [122] in the image of

$$P_0 = \frac{1}{\dim(\mathcal{S})} \sum_{s \in \mathcal{S}} s \in \mathcal{S}, \quad (\text{A26})$$

the logical action of elements in $\mathcal{C}(\mathcal{S})$ is formally characterized by $\mathcal{A}_\mathcal{S} = \text{span}_\mathbb{C}(C(\mathcal{S})/\mathcal{S})$, which again is a $*$ -algebra. In fact, since $C(\mathcal{S})/\mathcal{S}$ is isomorphic to a Pauli group, it spans the full matrix algebra on the logical subspace, $\mathcal{A}_\mathcal{S} \simeq \mathcal{M}_{2^k}$, where $k = \log_4(\dim(\mathcal{A}))$ denotes the number of logical qubits on which $\mathcal{A}$ acts.

The isomorphism $\varphi_\mathcal{S}(A, B)$ obtained in Lemma 1 can be lifted to a $*$ -isomorphism

$$\Phi_\mathcal{S}(A, B) : \mathcal{A}_{\mathcal{S}_A} \rightarrow \mathcal{A}_{\mathcal{S}_B} \quad (\text{A27})$$

on two (formal) logical algebras. Importantly, a logical Clifford isomorphism is an isomorphism of that kind that happens to be fully described by an isomorphism on a particular basis of $\mathcal{A}_{\mathcal{S}_A}$ and $\mathcal{A}_{\mathcal{S}_B}$, the Pauli basis. A non-Clifford operation will not be described by such a group isomorphism but is intrinsically an algebra isomorphism that can map single Paulis to sums of Paulis.

b. The Logical isomorphism implemented by an RGB tensor network

In this section, we want to show that any RGB *tensor network* (TN) implements a logical isomorphism on some logical Pauli algebra. As a byproduct, we show in the proof of Lemma 3, how to think of the composition of (logical) Clifford blocks in a unified manner, independent of a specific encoding. This applies to any Pauli stabilizer tensor network and gives a complementary perspective on how to view the stabilizer formalism including measurements and may give insight onto how to better understand architectures for quantum computations on the logical level.

Proposition 1. Let T be an RGB tensor network composed of red, green, and blue tensors, introduced in Eqs. (32)–(34), as well as the basis transformations defined in Eq. (37), with $n \in \mathbb{Z}_{\geq 0}$ open legs. Consider a bipartition of the open legs into two subsets A and B . Equivalently, we denote the vector space associated to these legs with A and B . T can be interpreted as a linear map $A \rightarrow B$, considered as the vector spaces assigned to the input and output legs, and can act via conjugation as a map $B(A) \rightarrow B(B)$ mapping linear operators on A to linear operators in B . As such, it implements an isomorphism

$$\varphi_T(A, B) : N(\mathcal{S}_A)/_{\mathcal{S}_A} \rightarrow N(\mathcal{S}_B)/_{\mathcal{S}_B}, \quad (\text{A28})$$

for some stabilizer groups $\mathcal{S}_A$ on A and $\mathcal{S}_B$ on B .

Given Lemma 1, this reduces to showing that any RGB TN admits a stabilizer group of maximal rank, i.e., when it is interpreted as a state, it is a unique stabilizer state. We proceed step by step and first show that this structure holds for any of the constituents of an RGB TN and then, in Lemma 3, show how to construct the stabilizer group of a contracted network, knowing the stabilizer groups of the constituents. This lemma is independent of the specific RGB TN representation and applies to all stabilizer tensor networks and with that to any linear map composed of Clifford operations.

In the following, the red, green, and blue spider, defined in Eqs. (33), (34), and (32), as well as the basis transformations, defined in Eq. (37), are considered “elementary building blocks” of an RGB TN.

Lemma 2. Let t be an elementary building block of an RGB TN with n legs. As a state, it is stabilized by a stabilizer group of rank n .

Proof. Let us go through each building block. We will map the projective symmetries of the tensors to a stabilizer group of maximal rank. Each of the symmetries of the tensors considered in this work are Pauli symmetries, i.e., they can be viewed as multiplying the tensor with a tensor product of Paulis on the vector space associated to the open legs of a tensor.

We find that the Pauli operators associated to the symmetries of the basis transformations in Eq. (A17) indeed commute and generate a rank-2 stabilizer group. For example, the first tensor in Eq. (A17), admits two generators $X \otimes Z$ and $Y \otimes Y^T$. Since they have two legs, this is a maximal stabilizer group and fully defines the tensor.

For the red, green, and blue spiders, we consider the symmetries defined in Eqs. (48) and (A11). We find that their symmetry groups are related by permuting the color (i.e., Pauli type), such that it suffices to consider a single color. Take the blue tensor with n open legs. Its symmetries are shown in Eq. (48). It has one X -type symmetry with which we associate the stabilizer $(-1)^s X^{\otimes n}$, where s is the value of the sign of the tensor, and a Z -type symmetry for each pair of legs (i, j) associated to the stabilizer $Z_i Z_j$. A simple counting argument shows that the group of these Pauli operators is indeed a stabilizer group of rank n . One can view this stabilizer group as the stabilizer group of a GHZ-like state. Similarly, for the other colors there are two types of stabilizer generators, one that acts on all input legs and might carry a sign of -1 and a set of operators that act on pairs of legs with a different Pauli type.

This completes the proof for each of the elementary building blocks. ■

Given that the elementary building blocks are defined by a stabilizer group of maximal rank, we want to proceed by showing how to obtain the stabilizer group of a tensor network composed of these building blocks. Specifically, the

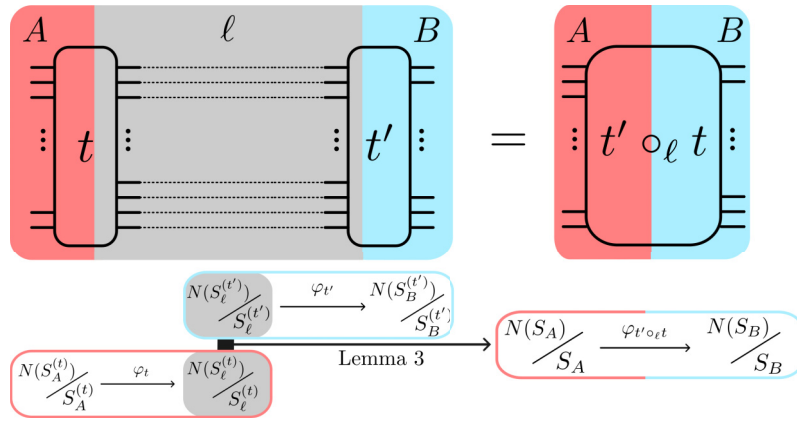


FIG. 19. An alternative perspective on Lemma 3 and the construction in our proof of it. Given two linear Clifford maps $t : A \rightarrow \ell$ and $t' : \ell \rightarrow B$ each of which acts like an isomorphism φ_t and $\varphi_{t'}$ on an encoded logical Pauli group (possibly a different one for t and t'), we can construct the isomorphism applied by the contracted network, a linear map from $A \rightarrow B$ explicitly. This is done by finding a subgroup of Pauli operators on which the isomorphism can be composed and deriving a resulting stabilizer group on $A \otimes B$ from it. Using the equivalence in Lemma 1, this stabilizer group can be associated to a logical isomorphism of the contracted network. Note that, in a sense, the global stabilizer group on $A \otimes B$ is more fundamental since it fully describes the entire network. Lemma 1 tells us that for any bipartition (A, B) we can infer the induced logical isomorphism from a logical Pauli group on A to a logical Pauli group on B . For a detailed description of the construction of the composite isomorphism we refer to the proof of Lemma 3.

following lemma and its proof give an explicit construction of the resulting stabilizer group when contracting two stabilizer TNs along an arbitrary subset of legs.

Lemma 3. Let t and t' be two tensor networks with n and n' open legs that admit stabilizer groups $S^{(t)}$ and $S^{(t')}$, both with maximal rank (n and n'). Consider an ordered subset $\ell = \{\ell_1, \ell_2, \dots, \ell_c\}$ of c legs of t , with $c \leq \min(n, n')$, and identify them with another ordered subset of legs ℓ' of t' of the same size. We can define a unique contraction of t and t' by joining ℓ with ℓ' according to their order and denote the contracted network with $t' \circ_\ell t$.

The contracted network $t' \circ_\ell t$, admits a stabilizer group of maximal rank that is uniquely determined from the stabilizer groups of t and t' , respectively.

We illustrate the “joining operation” for two stabilizer tensor networks and an alternative interpretation of Lemma 3 in Fig. 19. The idea behind the construction of the isomorphism, respectively, the stabilizer group, of the contracted network is that t' effectively acts like measuring the stabilizer group $S_\ell^{(t')}$ on a code stabilized by $S_\ell^{(t)}$ [123]. How to treat the action of such a measurement on a set of logical operators is part of the usual stabilizer formalism and captured well by many references, e.g., Refs. [13, 17, 124]. Our proof builds on that to find the right set of operators along which a composition of the isomorphism of φ_t and $\varphi_{t'}$ make sense and shows that this indeed leads to a composite isomorphism, respectively, a contracted tensor network with a full-rank stabilizer group.

Proof. Let A be the vector space associated to the set of open legs of t in the complement of ℓ and B the vector

space associated to the set of open legs of t' in the complement of ℓ' . In this proof, we aim to construct a stabilizer group of maximal rank on $A \otimes B$ from $S^{(t)}$ and $S^{(t')}$. We denote their maximal subgroups supported on the contracted legs as $S_\ell^{(t)}$ and $S_\ell^{(t')}$ and their restricted subgroups by $S^{(t)}|_\ell$ and $S^{(t')}|_\ell$ (cf. notation from the proof of Lemma 1). Since there is a unique 1-1 mapping from ℓ to ℓ' , we use ℓ in all expressions. Additionally, we use the same symbol ℓ for the vector space associated to the contracted legs. The vector spaces considered in this proof are A , ℓ , and B .

Recall from Lemma 1 that we can think of $S^{(t)}|_\ell$ and $S^{(t')}|_\ell$ as generating a complete set of (real) logical operators including corresponding stabilizer groups $S_\ell^{(t)}$ and $S_\ell^{(t')}$. Moreover, by Lemma 1 the problem of identifying a stabilizer group of maximal rank on $A \otimes B$ is equivalent to constructing an isomorphism between the logical Pauli group of a (stabilizer) code in A and the logical Pauli group of a (stabilizer) code in B , both defined by a stabilizer group S_A and S_B . In this proof, we will start with the isomorphisms φ_t and $\varphi_{t'}$, associated to t and t' with the given bipartition of the space associated to the open legs, $A \otimes \ell$ and $\ell \otimes B$. We denote the the number of logical qubits on which φ_t acts by k and the number of logical qubits on which $\varphi_{t'}$ acts by k' .

Consider the maximal joint subgroup of (potentially trivial) logical operators (generated by logical X and Z representatives),

$$P_\ell = S^{(t)}|_\ell \cap S^{(t')}|_\ell. \quad (\text{A29})$$

Each element in P_ℓ can be identified with a coset of $S_\ell^{(t)}$ and $S_\ell^{(t')}$ by the unique homomorphism that given a normal subgroup $H \leq G$ implements the map $g \mapsto gH$. Moreover, P_ℓ forms a subgroup of both $S^{(t)}|_\ell$ and $S^{(t')}|_\ell$. Taken together, P_ℓ defines both a subgroup of cosets with respect to $S_\ell^{(t)}$ and $S_\ell^{(t')}$.

We are now set to construct the maximal stabilizer group of $t' \circ_\ell t$. We start with the (probably nonmaximal) stabilizer group $S = S_A^{(t)} \otimes S_B^{(t')} = \{s_a \otimes s_b \mid s_a \in S_A^{(t)}, s_b \in S_B^{(t')}\}$ and add commuting terms to get a stabilizer group of maximal rank. This is done by composing the isomorphisms of t and t' along a well-defined intersection. We can summarize the construction as follows: For each $p \in P_\ell$, we add a representative of $\varphi_t^{-1}(pS_\ell^{(t)}) \otimes \varphi_{t'}(pS_\ell^{(t')})$ to S . This gives a systematic mapping of elements in P_ℓ , operators acting on ℓ , to stabilizers on $A \otimes B$ such that they encode the isomorphism applied by the contracted network, $t' \circ_\ell t$. Let us elaborate on the individual steps of the construction. It can be understood with the following commutative diagram:

$$\begin{array}{ccccc}
 N(S_A^{(t)}) & \xleftarrow{\quad} & S^{(t)}|_\ell \cap S^{(t')}|_\ell & \xrightarrow{\quad} & N(S_B^{(t')}) \\
 \downarrow \cdot S_A^{(t)} & & \swarrow \cdot S_\ell^{(t)} \quad \searrow \cdot S_\ell^{(t')} & & \downarrow \cdot S_B^{(t')} \\
 N(S_A^{(t)}) / S_A^{(t)} & \xrightarrow{\varphi_t} & N(S_\ell^{(t)}) / S_\ell^{(t)} & \xrightarrow{\varphi_{t'}} & N(S_B^{(t')}) / S_B^{(t')}
 \end{array}, \quad (A30)$$

where in the top row we have sets composed of physical operators on A , ℓ , and B , respectively, and on the bottom row we have sets of cosets over different stabilizer groups. One should interpret this diagram as defining the maps associated to the arrows in the top row [125]. Each step in the construction outlined above can be understood as an arrow in this diagram. Specifically, we go from top to bottom by identifying an operator with a coset with respect to a given stabilizer group. The “inverse” operation, going from bottom to top is “taking a representative of a coset.” The horizontal arrows on the bottom represent isomorphisms so admit proper inverses. In the procedure described above the stabilizer added to S is obtained by taking an element p in the intersection set in the top center and mapping it to an operator p_A on A by going in clockwise direction in the left loop and then to an operator p_B on B by applying maps in counter clockwise direction in the right loop. The operator added to S is given by $p_A \otimes p_B$. This is done for each element in P_ℓ , the intersection set.

It remains to be shown that S is now a stabilizer group of maximal rank. Let n_A be the number of qubits in A and n_B be the number of qubits in B . We know that $S_A^{(t)}$ has rank $n_A - k$ and $S_B^{(t')}$ has rank $n_B - k'$. Hence, we need to show that the procedure of adding elements to S described above increases the rank by exactly $k + k'$. Since the construction starts with a stabilizer group that contains the stabilizers in

$S_A^{(t)}$ and $S_B^{(t')}$ we only need to consider elements in P_ℓ in nontrivial logical cosets with respect to $S_\ell^{(t)}$ and $S_\ell^{(t')}$. Additionally, due to the group structure on the cosets, we only need to consider a suitable generating set for the group of cosets. For each nontrivial generating coset for which there is at least one $p \in P_\ell$, the rank is increased by one.

To prove that this construction leads to a stabilizer group of maximal rank, we employ techniques introduced in Ref. [124] for code deformations. Consider the subsystem code with the gauge group

$$G = \langle S_\ell^{(t)}, S_\ell^{(t')} \rangle \quad (A31)$$

generated by the two stabilizer groups on ℓ given by t and t' . We denote the center of that gauge group by $S_G = Z(G)$. Up to signs, it corresponds to the stabilizer group associated to G . We can interpret the sets in the construction of S in light of that subsystem code. Let

$$\mathcal{L}_G = C(S_G)/G \quad (A32)$$

be the group of inequivalent sets of logical Pauli operators of this subsystem code and $\kappa = \log_4(|\mathcal{L}_G|/4)$ the number of logical qubits in that subsystem code.

We can identify both $S_\ell^{(t)}$ and $S_\ell^{(t')}$ as stabilizing a (partly) gauge-fixed subspace since they are Abelian subgroups of G . Each $p \in P_\ell$ that is a nontrivial logical with respect to G , is hence also a nontrivial logical with respect to $S_\ell^{(t)}$ and $S_\ell^{(t')}$. Recalling that P_ℓ is exactly the real subgroup of $C(S_G)$, we find that it includes all nontrivial logicals and hence the rank of S is increased by at least 2κ when following the construction described above. Additionally, P_ℓ includes logical operators of the gauge qubits of the code, operators that are nontrivial logicals in $S_\ell^{(t)}$ but not in $S_\ell^{(t')}$ and vice versa. In fact, P_ℓ contains exactly the gauge logical operators that are not fixed in $S_\ell^{(t)}$ but in $S_\ell^{(t')}$ and vice versa. Note that here, P_ℓ only contains one operator per logical (gauge) qubit. In $S_\ell^{(t)}$ there are $k - \kappa$ gauge qubits unfixed and in $S_\ell^{(t')}$ there are $k' - \kappa$ gauge qubits unfixed. Following the construction above we find that for each of these unfixed gauge qubits the rank of S is increased by 1 when following the construction above.

Taken together, we find that the procedure above increases the rank of S by $2\kappa + k - \kappa + k' - \kappa = k + k'$. This leads to a full rank of S of $n - k + m - k + k + k' = n + m$, which is the maximal rank and hence completes our proof. ■

This leaves us with all the necessary ingredients to prove the main proposition of this section.

Proof of Proposition 1. Given Lemma 1 and Lemma 3 the theorem follows by induction. Any RGB TN can be

obtained by a sequence of contractions of the form in Lemma 3 and with that admits a stabilizer group of maximal rank. By Lemma 1, this leads to the isomorphism in Proposition 1. ■

In the proof of Lemma 3 we used an auxiliary subsystem code to prove that the stabilizer group constructed had maximal rank. It is defined as being generated by the stabilizer groups that determine the codes on which t and t' act on like an isomorphism. We found that all nontrivial logical operators of the subsystem code survive the composition. They determine the size of the logical Pauli group on which the contracted network acts isomorphically. The other logicals that survive the composition are gauge logicals of the subsystem code and correspond to logical qubits of either t or t' and are “measured” within t' or t , respectively.

c. Basics of Pauli flows

Given this fact, we proceed in defining the concept of a Pauli flow based on the projective symmetries of the RGB tensors and associated colorings defined in the main text and here, and elaborate on their physical meaning. It is not surprising that they fully characterize the network since its logical action in Proposition 1 was obtained from the symmetries leading to Pauli flows. Additionally to the logical action, we first show that detector flows give rise to a linear code in the space of measurement outcomes. Based on this “redundancy” within the network, we construct a group homomorphism between (cosets of) Pauli flows and a stabilizer group on the Hilbert space associated to the input and output legs that captures the logical isomorphism applied by the circuit.

Definition 1 (Highlights and Pauli flows). Let T be an RGB tensor network with a set of tensors V and edges E connecting them. Each of the red, green, or blue tensors in the network might carry a sign, labeled s , that has a *value* $\hat{s} \in \mathbb{Z}_2 = \{0, 1\}$. We denote the set of all signs in T with $M(T)$. Given any subset $m \subseteq M(T)$ we can define its value by adding the values of its elements,

$$\hat{m} := \sum_{j \in m} \hat{j} \pmod{2}. \quad (\text{A33})$$

We define a Pauli *highlight* to be a coloring of edges and signs where each edge and sign is highlighted in one of four colors {trivial, red, green, blue}. A Pauli flow $F(T)$ is a highlight that for each tensor individually is *valid*, i.e., either it is trivial (all edges connected to v as well as the sign highlighted trivially) or is of the form of one of the highlights in Eqs. (49) and (50a).

Definition 2 (Detector flows). We call a Pauli flow in which only internal edges in an RGB tensor network are

highlighted nontrivially a *detector flow*. We denote the set of detector flows of a given RGB TN by $D(T)$.

For clarity, here we review in a more formal setting the necessary definitions made informally in the main text, and derive the main theorem from these.

Note that we distinguish between the sign of a tensor as an abstract object and its value. In some cases these values might be fixed (e.g., when applying a Pauli operator on an edge in the network) or obtained in an experiment (e.g., for measurement outcomes). In most cases one can think of a sign as being analogous to its value (see Definition 5) but this distinction simplifies upcoming definitions.

Lemma 4. The set of all Pauli flows $F(T)$ of an RGB tensor network T forms an Abelian group under the group operation inherited from the group operation of the highlights on each edge, defined in Eq. (47). We denote this group operation by $\oplus_F$. The set of detector flows $D(T)$ forms a subgroup of $F(T)$.

Since the group operation $\oplus_F$ is Abelian, we refer to it as an addition.

Proof. First, we define an “addition” on highlights by lifting the addition defined for each edge individually to highlights on the network by applying the rules of Eq. (47) to each edge individually. Additionally, we impose the same rules of combining highlights of signs. Clearly, this fulfills all the conditions of a group multiplication where the trivial highlight on all edges and signs plays the role of the (unique) unit and the group operation is by definition commutative and any flow is its own inverse. This renders the Pauli flows a group isomorphic to $\mathbb{Z}_2^{\times d_F}$ for some integer d_F .

Given this notion of summing with the obvious choice of identity, it remains to be shown that Pauli flows are closed under that operation. Looking at Eqs. (49) and (50a) we can see that the sum of two valid highlights again is a valid highlight and hence the product of two Pauli flows f_1 and f_2 is again a Pauli flow $f_1 \oplus_F f_2$.

If two flows highlight an edge e trivially, then their sum clearly also highlights e trivially. From this it follows directly that the set of detector flows is closed under $\oplus_F$, i.e., a subgroup of $F(T)$. ■

The abstract group formed by $F(T)$ can be viewed as a subgroup of $\mathbb{Z}_2^{\times 2|E|}$, where $|E|$ is the number of edges in T . Additionally, we can include the highlight of the signs carried by the tensors into the definition of Pauli flows by incorporating additional $\mathbb{Z}_2$ variables for each tensor that can be charged with respect to a flow.

Definition 3 (Pauli looping of a flow). For any Pauli flow $f \in F(T)$ on an RGB tensor network T we associate the following two-step transformation to the network:

- (1) Apply the symmetries associated to the flow, for each tensor in the network.
- (2) Combine the remaining Pauli tensors on each contracted leg to a global phase.

We call any such transformation (*Pauli looping*) of the flow f .

The first step can be understood as iteratively moving and transforming Pauli operators through the network as we show for an exemplary detector flow in a TN representing a repetition code circuit in Fig. 20. Note that the last step can always be performed by the definition of the Pauli flow.

Definition 4 (Sign of a flow). Let $f \in F(T)$. Applying the looping of f results in a network that differs by the starting network only by a ± 1 prefactor c_f and Pauli tensors on the open legs. Consider a fixed orientation of the green tensors that remain on the open legs, e.g., pointing into the network. For this orientation, we define the *sign of the flow* f as $\text{sgn}(f) = (1 + c_f)/2$. It takes values in $\{0, 1\}$.

For the purposes discussed in this paper the choice of fixed “reference” orientation on the open legs does not matter, as long as it is fixed.

d. Detector flows define a classical linear code

In this section, we prove that the group of detector flows gives rise to a classical linear code in the space of all possible sign configurations of the network. This proof works for any RGB TN, including those not necessarily coming directly from a circuit [126], but for example

from a measurement-based protocol, where the tensor network represents the amplitudes associated to measurement outcomes obtained from single-qubit measurements on a “resource state” represented as an RGB TN.

Lemma 5. Let d be a detector flow. It holds that $\text{sgn}(d) = 0$.

Proof. By definition of a detector flow, it highlights all open legs trivially. As such, there is no choice for a reference orientation on the open legs when defining its sign $\text{sgn}(d)$. By definition of $\text{sgn}(d)$, the network before and after applying the Pauli looping of d is exactly the same, up to a prefactor $(-1)^{\text{sgn}(d)}$. This prefactor has to equal 1 for the network to give nonzero values, i.e., $\text{sgn}(d) = 0$. If this is not fulfilled the network evaluates to zero, which says that the probability of observing a particular sign configuration of elementary tensors that leads to $\text{sgn}(d) = 1$ is zero. ■

We illustrate the proof for a detector flow of a repetition code circuit in Fig. 20.

Lemma 6. Let f_1 and f_2 be two Pauli flows of the same RGB TN. In that case it holds that $\text{sgn}(f_1) \oplus \text{sgn}(f_2) = \text{sgn}(f_1 \oplus_F f_2)$, where $\oplus_F$ denotes the addition of Pauli flows and $\oplus$ addition modulo 2. Phrased differently, sgn is group homomorphism.

Definition 5 (Sign space). Let $M(T)$ be the set of all signs of an RGB tensor network. We define the *sign space* $\tilde{M}_T$ as the formal $\mathbb{Z}_2$ span over $M(T)$, $\text{span}_{\mathbb{Z}_2}(M(T))$. This

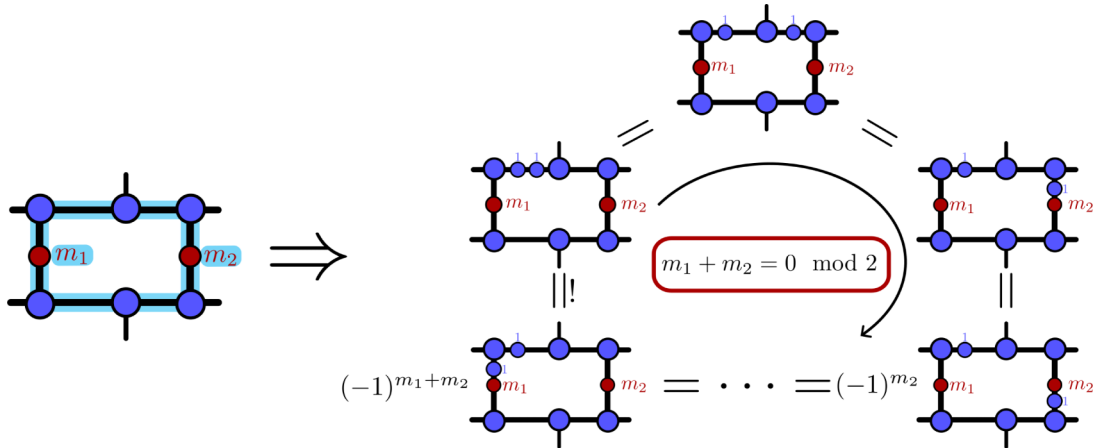


FIG. 20. We show an exemplary detector flow associated to two repeated ZZ measurements as happening, for example, in a repetition code circuit. Any such detector flow defines a constraint on the values of the signs charged with respect to that flow. In the case shown, $m_1 + m_2 = 0 \pmod{2}$, which can be seen explicitly by constructing a sequence of transformations that leave the contracted network invariant (following in the arrow in the figure). We first add two (blue) $s = 1$ tensors on one edge of the (blue) highlight and iteratively push one of the tensor through the network using the symmetries of the individual tensors. For the shown detector flow, the network acquires a phase of $(-1)^{m_1+m_2}$ through that sequence, which has to equal 1 if the network gives a nonzero value.

equips $\hat{M}_T$ with a canonical basis in which each basis element corresponds to a specific sign label. We denote the basis element associated to sign $s \in M(T)$ by $\mathbf{e}_s \in \hat{M}_T$.

Each Pauli flow f defines a projection $p_f : \hat{M}_T \rightarrow \hat{M}_T$,

$$\mathbf{m} = \sum_{s \in M} m_s \mathbf{e}_s \mapsto p_f(\mathbf{m}) = \sum_{s \in s(f)} m_s \mathbf{e}_s. \quad (\text{A34})$$

We call the image of p_f the *subspace charged with respect to f* .

Theorem 1. Let $D(T)$ be the group of detector flows of an RGB tensor network with a generating set of flows G_D . It defines a linear code $C_D \subseteq \hat{M}_T$, defined as the kernel of a parity-check matrix H_D whose rank (as a linear operator) equals the rank of $D(T)$ (as a finite Abelian group).

Physically, the space C_D corresponds to the space spanned by all sign values that fulfill Lemma 5 for every detector flow of the network. This space is equivalent to the *outcome code* introduced in Ref. [63] up to some subtleties in how the space of all signs is set up. Specifically, we include all signs in the network, not only measurement outcomes. This includes Pauli operators that would potentially flip some stabilizers and thereby render the resulting outcome code (in the language of Ref. [63]) an *affine* linear code. Since we include the signs attached to tensors that represent Pauli operators, we obtain a linear code without any further modifications.

We use the code C_D to infer errors occurring while performing a QEC protocol. Specifically, we associate a vector in $\hat{M}_T$ to the recorded signs during the QEC protocol and obtain a syndrome by applying H_D to it.

Proof. Let $d \in D$ be a detector flow and define the vector $\mathbf{h}_d = \sum_{s \in s(d)} \mathbf{e}_s \in \text{Im}(p_d)$ lying in the subspace charged with respect to d . In fact, we can define a (linear) subspace $C_d = \ker(\mathbf{h}_d^T)$, where $\mathbf{h}_d^T : \hat{M} \rightarrow \mathbb{Z}_2$ is understood as a linear map that acts via taking the dot product with the vector $\mathbf{h}_d$. The kernel of C_d is the subspace spanned by sign values that fulfill $\text{sgn}(d) = 0$.

Now consider two independent detector flows d_1 and d_2 with the associated subspaces C_{d_1} and C_{d_2} . By linearity of the dot product and transposition and Lemma 6 it follows that

$$C_{d_1} \cap C_{d_2} = C_{d_1} \cap C_{d_2} \cap C_{d_1 \oplus_F d_2}. \quad (\text{A35})$$

Hence, it suffices to consider a generating set of detector flows, G_D , when defining the space

$$C_D = \bigcap_{d \in D(T)} C_d = \bigcap_{g \in G_D} C_g = \ker(H_D), \quad (\text{A36})$$

where H_D is the matrix whose rows are $\{\mathbf{h}_g^T | g \in G_D\}$. ■

e. The logical isomorphism from Pauli flows

Given an RGB tensor network T and its Pauli flow group $F(T)$ as well as its detector subgroup $D(T)$, we can construct a quotient group of $F(T)/D(T) \simeq \mathcal{S}$ and an isomorphism to a stabilizer group on the input and output legs of T that we associate to the network. This stabilizer group is exactly the stabilizer group that captures the logical isomorphism applied by T in Proposition 1.

Consider an RGB tensor network with $n + m$ open edges. We interpret the first n legs as “input” legs and the remaining m legs as “output” legs. In the following, we define a group homomorphism from $F(T)$ onto $\hat{\mathcal{P}}_{n+m} = \mathcal{P}_{n+m}/\langle i \rangle$, the *quasi-Pauli group* on the vector spaces associated to the open legs in the network.

Definition 6. Let $f \in F(T)$ be a flow of an RGB tensor network T . Consider its restriction onto the input and output legs and let $\text{col}_f(i)$ be the color of the highlight of edge $i \in \{1, 2, \dots, n + m\}$. Let $\rho : \{\text{trivial, red, green, blue}\} \rightarrow \hat{\mathcal{P}}_1$ be the inverse map of the map defined in Eq. (46). We define $\hat{p} : F(T) \rightarrow \hat{\mathcal{P}}_{n+m}$

$$\hat{p}(f) = \bigotimes_{i=1}^{n+m} \rho(\text{col}_f(i)), \quad (\text{A37})$$

to map onto Pauli operators modulo phases on the input and output spaces, where the labeling of the open edges is chosen that the first n tensor factors correspond to the “input” space and the remaining m tensor factors to the “output” space.

Lemma 7. $\hat{p}$ is a group homomorphism.

Proof. First, we note that $F(T)$, as a group, is isomorphic to a subgroup of $\hat{\mathcal{P}}_E$, the quasi-Pauli-group on all edges of T , contracted or open, which itself is isomorphic to $\mathbb{Z}_2^{\times 2|E|}$.

Having established the isomorphism between a highlight and a quasi-Pauli-group, it remains to be shown that $\hat{p}(f_1)\hat{p}(f_2) = \hat{p}(f_1 \oplus_F f_2)$. Noting that the group law on the highlights directly resembles how their images under ρ multiply, i.e., ρ is a group homomorphism onto $\hat{\mathcal{P}}_1$ and $\hat{p}(f)$ is a simple tensor product over images of ρ we find that this is indeed the case. ■

Corollary 1. $\ker(\hat{p}) = D(T)$.

Proof. By definition, elements in $D(T)$ have trivial highlights on the input and output edges. This shows that $D(T) \subseteq \ker(\hat{p})$. Moreover, ρ is injective, i.e., has trivial kernel. Hence, $\ker(\hat{p}) \subseteq D(T)$. ■

As D is the kernel of $\hat{p}$, we can unambiguously define $\hat{p}$ to act on $F(T)/D(T)$ by taking an arbitrary representative

in $fD(T)$ as input into $\hat{p}$. Since $D(T)$ is a normal subgroup ($F(T)$ is Abelian), the induced map $\hat{p}_D : F(T)/D(T) \rightarrow \hat{\mathcal{P}}_{n+m}$ is still a group homomorphism. Moreover, it is clearly injective.

Definition 7. We define $\iota : \hat{\mathcal{P}} \rightarrow \mathcal{P}$:

$$[1] \mapsto 1, \quad [X] \mapsto X, \quad [Y] \mapsto Y, \quad [Z] \mapsto Z, \quad (\text{A38})$$

that maps each coset to its representative, with +1 sign. This map can be straightforwardly defined on $\hat{\mathcal{P}}_{n+m}$ by taking its tensor product $\iota^{\otimes(n+m)}$. We will sometimes omit the superscript if the domain and with that the superscript is clear from context.

The map ι allows us to map images from $\hat{p}_D$ to actual Pauli operators in input and output spaces. It is defined such that the image in the Pauli group is well behaved, in the sense of the following key lemma.

Lemma 8. $\tilde{p} = \iota \circ \hat{p}_D$ is an injective group homomorphism.

Proof. Since $\hat{p}_D$ and ι are both injective, $\tilde{p}$ is injective. It remains to be shown that $\tilde{p}$ is a group homomorphism. Specifically, we need to show that two Pauli operators in $\text{Im}(\tilde{p})$ commute since it could only fail to respect the (Abelian) group multiplication up to signs that were modded out in the definition of $\hat{p}$.

We can see this directly from Proposition 1. In fact, as common in the stabilizer formalism, any Pauli *-homomorphism $\mathcal{P}_n \rightarrow \mathcal{P}_m$ can be understood as a group homomorphism from cosets with respect to an “input” stabilizer group to cosets with respect to an “output” stabilizer group. As such, it can be understood as a collection of pairs (p_i, p_j) , for some $p_i \in \mathcal{P}_n, p_j \in \mathcal{P}_m$, indicating that (a coset containing) p_i gets mapped to (a coset containing) p_j . Importantly, not every operator in $\mathcal{P}_n$, respectively $\mathcal{P}_m$, appears in these pairs. Moreover, there can be many p_i ’s for the same p_j and many p_j ’s for the same p_i , which can be understood as stabilizer equivalences on $\mathcal{P}_n$, respectively, $\mathcal{P}_m$ induced by some sort of projection (measurement) in the circuit.

Following the construction of $\tilde{p}$ we find that the image of $\tilde{p}$ is exactly the minimal set of these pairs of Pauli operators that defines the homomorphism but without any signs. To show that omitting these signs does not alter the group structure, we need to show that all elements in $\text{Im}(\tilde{p})$ commute. To show this, consider two different elements $p_i \otimes p_j, p'_i \otimes p'_j \in \text{Im}(\tilde{p})$. By the homomorphism it is guaranteed that $[p_i, p'_i] = [p_j, p'_j]$, where $[a, b] = ab - ba$ is the algebra commutator. Hence, $[p_i \otimes p_j, p'_i \otimes p'_j] = 0$ for all $p_i \otimes p_j, p'_i \otimes p'_j \in \text{Im}(\tilde{p})$, which completes the proof. ■

We have just constructed an isomorphism of (cosets of) flows to the stabilizer groups associated to the RGB tensor

network that capture the logical encoding and nontrivial Clifford applied by it, namely $\text{Im}(\tilde{p})$. There is still some freedom, however, in choosing the signs of elements in $\text{Im}(\tilde{p})$ consistently. On the level of the logical action $\tilde{p}$ determines the Clifford action up to Paulis since these are exactly the operators that might change the sign of a Pauli operator by conjugation.

We can resolve this ambiguity within the Pauli flow formalism though, by including the signs of the flows explicitly. Note that this is the first time in this section where we actually need to consider how a given flow extends into the network, not only its restriction on the open legs.

Theorem 2. The map $p : F(T)/D(T) \rightarrow \mathcal{P}_{n+m}$ defined as

$$fD(T) \mapsto (-1)^{\text{sgn}(f)} \tilde{p}(f), \quad (\text{A39})$$

where f is an arbitrary element in $fD(T)$, is well defined and $\mathcal{S} = \text{Im}(p)$ is a stabilizer group.

Proof. p only differs from $\tilde{p}$ (which is well defined) by adding a factor $(-1)^{\text{sgn}(f)}$. By Lemma 5, this factor is independent of the choice of representative in fD and hence p is well defined.

Lemma 8 says that $\text{Im}(\tilde{p})$ is a stabilizer group. By Lemma 6, the added factor $(-1)^{\text{sgn}(f)}$ also multiplies like the elements in $F(T)$, following the multiplication of the flows. Combined, the additional sign factor does not alter the group structure and by Lemma 5 the identity coset is (the only coset that) gets mapped to the identity operator, $p(D(T)) = \tilde{p}(D(T)) = 1$, rendering it a stabilizer group. In particular, $-1 \notin \text{Im}(p)$. ■

Often, we might abuse the notation slightly and use p for a map that directly maps flows, rather than cosets of flows, to Pauli operators. Since the step of identifying a given flow with a coset is unambiguous.

Definition 8. As a subgroup of $\mathcal{P}_{n+m} = \mathcal{P}_n \otimes \mathcal{P}_m$, we can decompose $\mathcal{S} = \text{Im}(p)$ into three parts

$$\mathcal{S} = \mathcal{S}_{\text{in}} \times \mathcal{S}_{\text{out}} \times \mathcal{S}_L, \quad (\text{A40})$$

where $\mathcal{S}_{\text{in}}$ is the maximal subgroup only generated by Paulis of the form $P \otimes 1_m$ (only acting on the input space), $\mathcal{S}_{\text{out}}$ the maximal subgroup generated by Paulis of the form $1_n \otimes P'$ and $\mathcal{S}_L$ the subgroup that cannot be generated by operators that act trivially on one side of the tensor product.

We call $\mathcal{S}_{\text{in}}$ the *input stabilizer group*, $\mathcal{S}_{\text{out}}$ the *output stabilizer group* and $\mathcal{S}_L$ the *logical stabilizer group*. Analogously, we call their preimages under p *input and output stabilizer flows* and *logical flows*.

In fact, $\mathcal{S}$ is exactly the stabilizer group obtained when following the procedure described in Sec. A3 b based on the symmetries of the elementary building block of an RGB TN. As constructed, Pauli flows capture exactly these symmetries and the signs of the flows track how Pauli operators acquire a sign when propagating through the network. We can also track the composition described in the proof of Lemma 3 graphically by matching up Pauli flows. This makes this tool very useful when analyzing arbitrary Clifford protocols.

4. Comments on Pauli flows when applying rewrite rules to RGB network

We have seen that Pauli flows capture the full action of a given RGB network and give rise to the error-correcting capabilities of the associated protocol via its

detector flows. Since a given tensor network can be used to construct a whole equivalence class of concrete protocols implementing the same logical action via the rewrite rules discussed in Sec. A2, it is natural to ask how Pauli flows change under this equivalence relation. We will see that the flows of two equivalent networks t and t' are in a sense isomorphic: Given t and its flows, the flows of t' are uniquely determined by the sequence of rewrites we need to map t to t' .

To understand this more concretely, let us look at the individual rewrite rules. In fact, we can use the stabilizer group of each individual constituent to argue about how the flows change. There are different types of rewrite rules. The simplest one being the splitting rules presented in Eqs. (A9) and (A10). For these it is easy to see that the flows after the splitting operation are uniquely determined by the flows before. For example,

$$\text{Diagram (A41): Two rewrite rules for a blue tensor. The first rule shows a blue tensor with 'a' incoming flows splitting into two blue tensors with 'x' and 'y' incoming flows. The second rule shows a blue tensor with 'a' incoming flows being replaced by two blue tensors with 'x' and 'y' incoming flows, with additional pink lines representing other flows.$$
(A41)

where $x + y = a \pmod 2$. Similar relations are obtained when splitting and fusing red and green tensors. Note that the number of signs charged with respect to a given flow can change when applying the rewrite rules. As a consequence, the isomorphism class of the linear code obtained from the detector flows (see Theorem 1) is not an invariant under rewritings. This is the first indication that QEC performance can differ within one family of equivalent protocols. One should be aware, though, that the performance of a protocol highly depends on the error model and less about the details of the classical code on the signs in the network. Here we find a 1-1 correspondence of flows before and after the local rewriting. The same applies when canceling loops between differently colored tensors, e.g.,

$$\text{Diagram (A42): Two rewrite rules for canceling loops. The first rule shows a blue loop being removed, leaving a single blue line. The second rule shows a red loop being removed, leaving a single red line.$$
(A42)

and when applying the generalized bialgebra rule, e.g.,

$$\text{Diagram (A43): A generalized bialgebra rule showing a complex network of red and blue tensors being simplified into a single blue tensor with multiple incoming flows.$$
(A43)

The blue flows are mapped analogously by symmetry of the diagram.

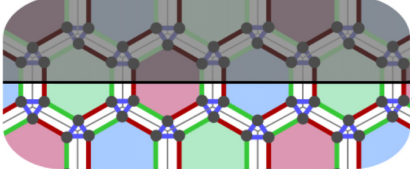
For the rules discussed so far there is a 1-1 mapping between the flows before and after rewriting, which makes the statement “flows are preserved” hold exactly. There is a remaining rule that is important to include, namely the rule that allows one to remove loops among tensors of the same color. Each of these loops can carry a detector flow, which has to be added when “spawning” such a loop or removed when canceling such a loop. Given the explicit rewrite, we can still uniquely determine the flows after the transformation. Namely, we add or remove one generator from a set of generating flows,

$$\text{Diagram (A44): Two rewrite rules for removing loops. The first rule shows a blue loop being removed, leaving a single blue line. The second rule shows a red loop being removed, leaving a single red line.$$
(A44)

This holds analogously for the red and the green tensor but for the flows of a different color. Since the resulting map from the group of all flows before to the flows after the rewrite is invertible (even though the flows are not mapped onto each other 1-1), we say that the flows give rise to an invariant under the rewrite rules. We leave a more rigorous mathematical study of the mathematical structure of that invariant to future work. Since adding a loop is the only way how to increase the rank of the group of flows $F(T)$ this is the rewrite rule that adds nontrivial constraints (detectors) on the signs of the network.

APPENDIX B: DYNAMICAL BOUNDARIES IN THE XYZ RUBY CODE

In this section, we present three distinct boundaries to the XYZ ruby code with the rewinding schedule. Each boundary is given in terms of its representation as an RGB tensor network and is obtained from cutting the bulk network along a plane that is parallel to the time direction. This cut can be represented on the spatial lattice of qubits. In Fig. 21, we present boundaries defined along the following cut:



(B1)

where we think of disregarding the network associated to the half-space depicted in gray, above the black line. After cutting, we obtain a (three-dimensional) network with open legs at the boundary. The boundaries along this cut are defined by a specific two-dimensional stabilizer tensor network with which we terminate the open legs along the cut. The stabilizer group of the network with which we terminate the open legs determines the Pauli flows close to the boundary (cf. Lemma 3) and with that the QEC properties, such as the emerging logical flows and the fault distance.

Given the (topological) algebraic data associated to the nonlocal flows, respectively, the logical operators they represent along the spacetime boundary, we can impose conditions on the stabilizer state with which we terminate the boundary. In Fig. 21, we present three distinct boundaries associated to the Pauli boundaries of the underlying color-code phase. From the perspective of the ISGs the

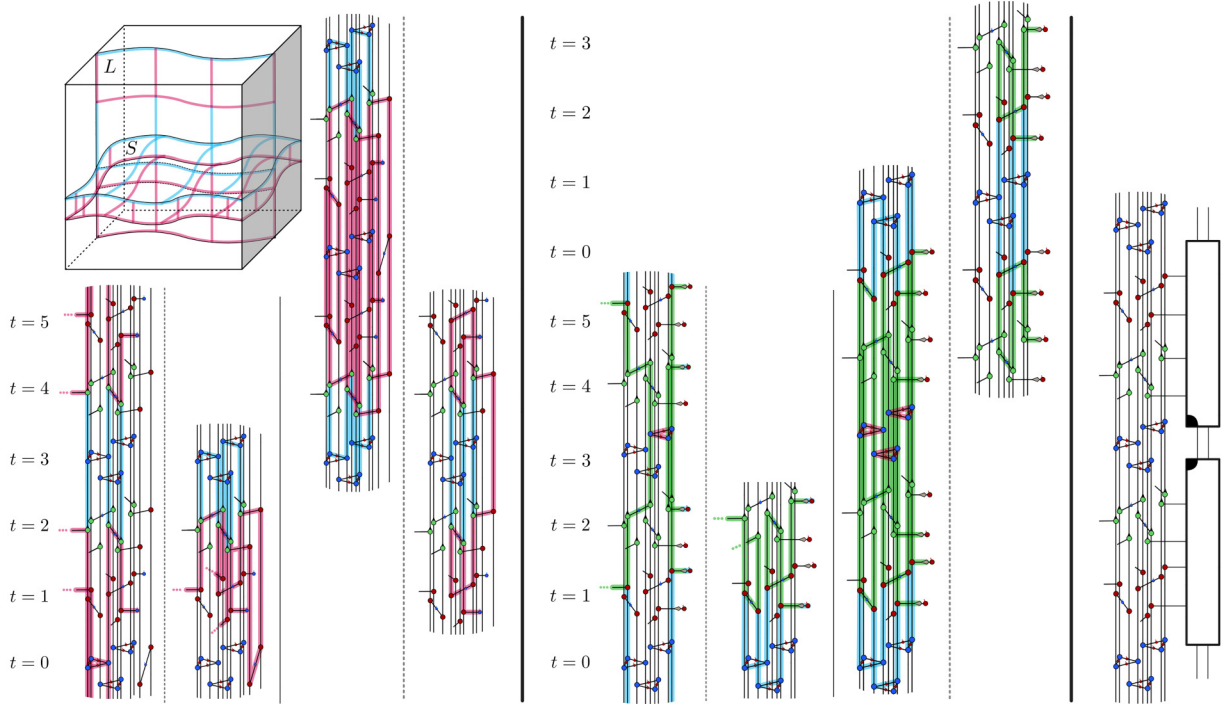
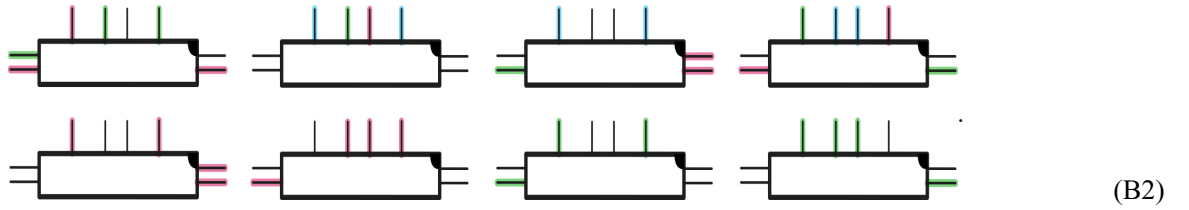


FIG. 21. Three dynamical boundaries for the XYZ ruby code in the rewinding schedule (cf. Sec. III). The time steps are expressed modulo 6. We construct the boundaries by cutting the network along a plane in spacetime and terminating the open legs by attaching a specific two-dimensional tensor network. The stabilizers of the attached tensor network determine, which flows are preserved in that procedure. There are choices for which the preserved flows give rise to a topological boundary. Here we show three inequivalent ways to terminate the boundary topologically, two explicitly resolved in terms of an RGB TN and the third one implicitly via an elementary cell of the boundary tensor network, which is defined by its stabilizer flows in Eq. (B2). At each boundary a different set of nonlocal flows can end, illustrated by L and S on the top left. Additionally, the local (detector) flows that deform them in the bulk can end at the boundary. For example, for the boundary on the left we show representatives of two inequivalent logical flows (probed in memory and stability experiments) that end at the boundary. They are both deformed by the same set of local (detector) flows, shown next to them. Analogously, the other boundaries shown allow for inequivalent logical flows to end. Importantly, the logical operators associated to the nonlocal flows that can end at two different boundaries of the three above anticommute. In fact, the boundaries correspond to the three Pauli boundaries of the underlying color-code phase and can be used to define a code on a triangle encoding a single logical qubit, as discussed in Ref. VI. The color boundaries can be constructed similarly but one might want to consider a different cut to simplify the construction.

boundary type changes according to the discussion in Sec. VI. In spacetime, however, each boundary should be regarded as a topological boundary of a unique type. This can be seen by looking at the nonlocal flows in the system with boundaries. We find that when introducing a boundary only a subset of the nonlocal flows survive. In particular, the nonlocal flows that are preserved along the boundary are associated to a Lagrangian subgroup of the underlying color-code anyon theory. As a consequence the nonlocal flows admit a topological symmetry in the bulk of the network, i.e., they can be freely deformed by a subgroup of local detector flows. The detector flows that deform the nonlocal flows close to the boundaries are also preserved at the boundary in our construction as can be seen in Fig. 21.

Since the microscopic realization and associated circuits have to be constructed separately using rewrite rules of the RGB TN representing the boundary in spacetime, we only show an explicit realization of the boundaries for two of the three types. The third boundary is defined by the elementary tensor from which we construct the two-dimensional TN with which we terminate the boundary. It is designed such that the following holds: If we interpret the nonlocal flows that can end at the first boundary as X -type logicals and the ones that can end at the second boundary as Z -type logicals then the third boundary allows their product, Y -like logicals, to end. The boundary itself is defined by an eight-legged tensor with the following stabilizer flows:



The associated stabilizers (cf. construction in Sec. A 3 e) uniquely determine the network up to signs of the stabilizers.

We leave the detailed construction of the other three types of color boundaries and corners between them to future work.

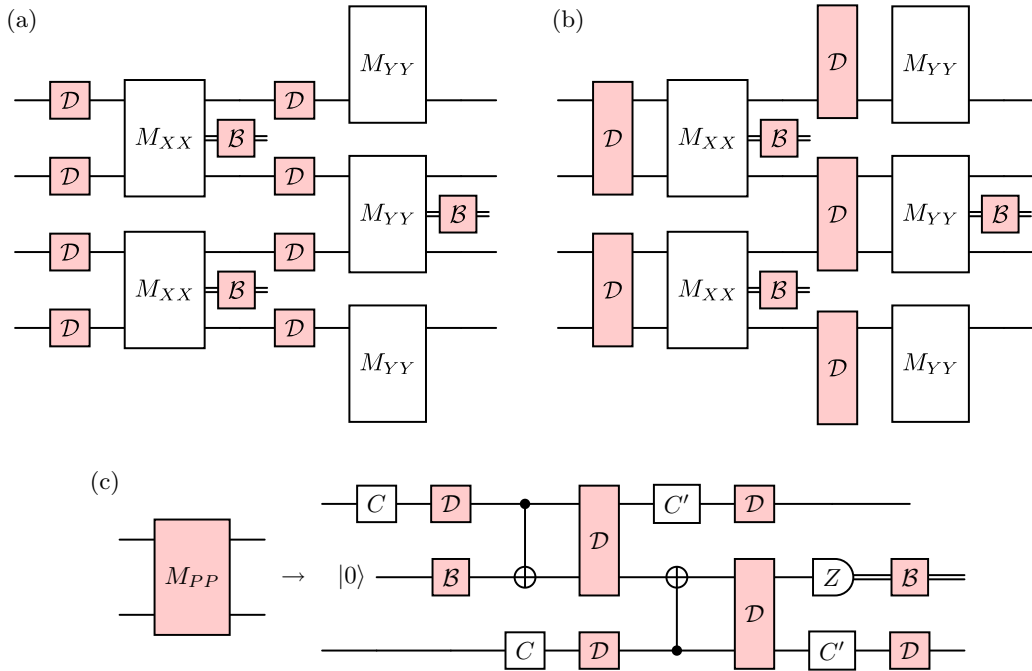


FIG. 22. Noise models for direct measurements. (a) Phenomenological noise model with single-qubit depolarizing noise in between every noisy two-body measurement. (b) EM3 noise model with correlated two-qubit depolarizing noise before the noisy two-body measurement. (c) Circuit implementation of the (noisy) two-body Pauli measurement.

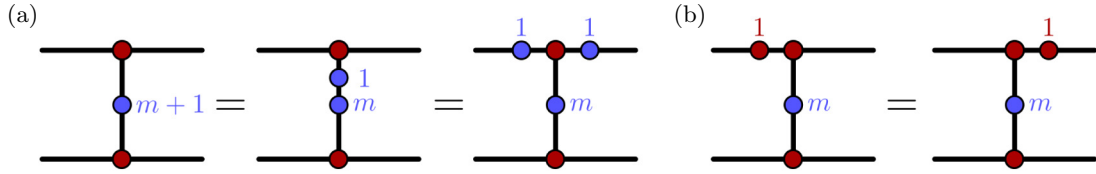


FIG. 23. Equivalences of error mechanism to derive the effective noise model of single qubit X , Y , and Z faults after XX , YY , and ZZ measurements. (a) A measurement error, here of an XX measurement, is represented as a flip of the measurement spider sign. Using the rewrite rules Eq. (A9), we can pull out the flip of the measurement as a two-legged $s = 1$ spider. This can be further rewritten to a blue spider just before, and after the measurement. (b) Pauli faults in the same basis as the measurement can be pushed through the measurement using the splitting rule for the red spider [Eq. (A10)].

APPENDIX C: FAULT TOLERANCE OF ELEMENTARY CIRCUIT-BUILDING BLOCKS

Here, we show fault-tolerance properties of the three noise and circuit models we use. Figure 22 shows the (a) phenomenological, (b) EM3 noise model for direct measurements, and (c) circuit-level noise model. We use a uniform noise model with parameter p and depolarizing noise channels

$$\mathcal{D}^{\otimes 1}(\rho) = (1 - p)\rho + \frac{p}{3} \sum_{E \in \{X, Y, Z\}} E\rho E \quad (\text{C1})$$

and

$$\mathcal{D}^{\otimes 2}(\rho) = (1 - p)\rho + \frac{p}{15} \sum_{EE' \in \{I, X, Y, Z\}^{\otimes 2} \setminus II} EE' \rho E' E. \quad (\text{C2})$$

Qubit initializations and measurements are followed by bit-flip channels,

$$\mathcal{B}(\rho) = (1 - p)\rho + pX\rho X. \quad (\text{C3})$$

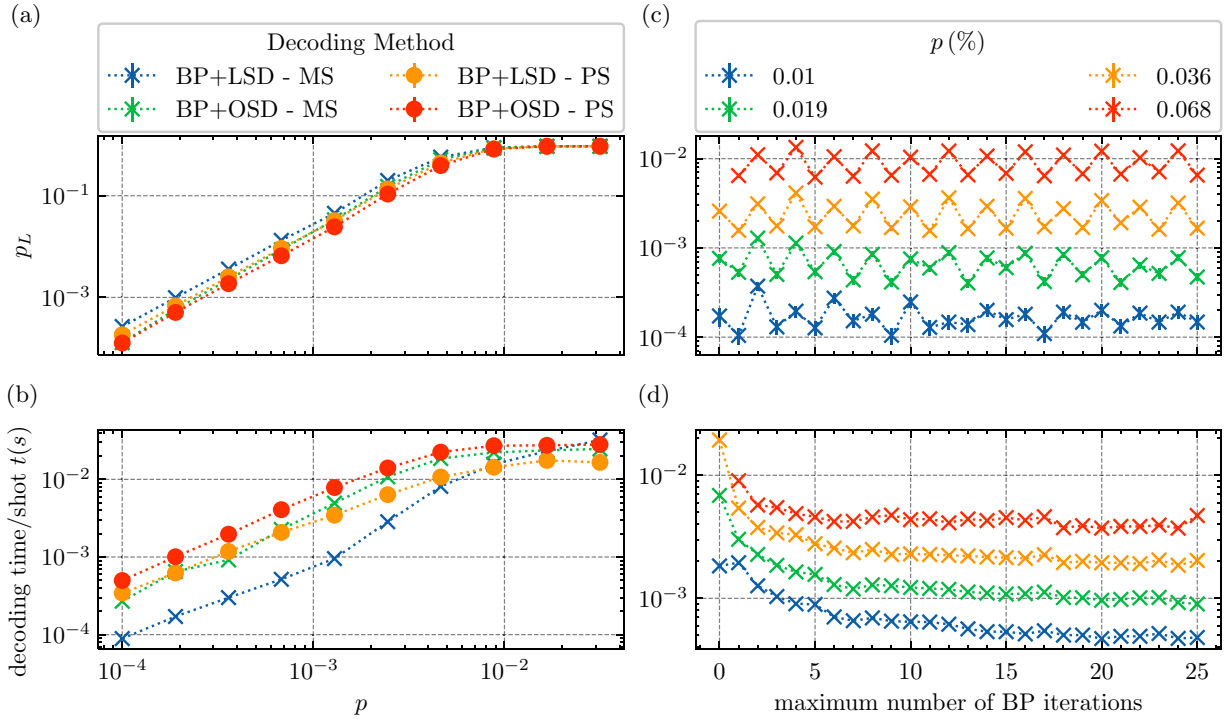


FIG. 24. Comparison of different decoding methods for $\mathcal{L}_Y$ memory experiments for the distance 4 XYZ ruby code using the phenomenological noise model and four rounds. In (a), we fix the OSD order to 20 and the BP iterations to 19. In general, BP + OSD with the *product-sum* (PS) algorithm gives the lowest logical error rate at the cost of the highest decoding time per shot, shown in (b). BP + LSD with the *minimum-sum* (MS) algorithm offers almost an order of magnitude faster decoding per shot, at the cost of a lower accuracy. (c) A different number of BP iterations shows oscillatory behavior in the logical error rate using OSD postprocessing. An odd number of iterations gives lower logical error rates. In (d) we show the overall decoding time per shot, which decreases with increasing number of BP iterations.

For the phenomenological noise model, any two-qubit error is an event $\mathcal{O}(p^2)$, whereas for the EM3 noise model, two-qubit Pauli errors are already first order $\mathcal{O}(p)$. This explains the halved fault distance of the direct measurement circuit with EM3 noise.

The implementation of the two-qubit Pauli measurement with an auxiliary qubit, single-qubit rotations and CNOTs keeps detrimental two-qubit faults at order $\mathcal{O}(p^2)$. This is due to the fact that all two-qubit errors on the data qubits that are $\mathcal{O}(p)$ are stabilizer equivalent to a single-qubit error. These are two-qubit depolarizing errors after the first CNOT, where the Pauli on the auxiliary qubit has part Z, i.e., a fault of the form PZ with $P \in \{X, Y, Z\}$. This part can propagate to the lower data qubit, such that (before the single-qubit rotation), the effective Pauli error on the data qubits is PZ with $P \in \{X, Y, Z\}$. Before rotating, any of the circuits has implemented a ZZ measurement, which is then part of the stabilizer group. The effective Pauli error is therefore stabilizer equivalent to a single-qubit error, $PZ \sim PI$. This shows that the chosen implementation of two-body measurements is fault tolerant. Figure 23 shows

equivalences to derive the effective error model we use for the argumentation on fault tolerance in the main text.

APPENDIX D: BELIEF PROPAGATION AND POSTPROCESSING PARAMETERS

We decode the syndrome by converting the detector error model of the noisy circuits to a detector matrix, a logical matrix and a prior vector, see the main text. We use the open source library `bposd` [81]. It offers a variety of belief propagation and postprocessing methods. As postprocessing methods, we use the well-established ordered statistics decoding (OSD) [80], and the recently introduced *localized statistics decoding* (LSD), that offers a better speed-accuracy trade-off enabling us to perform Monte Carlo sampling for larger distances, at the cost of a higher logical error rate. For the OSD postprocessing and an *OSD order* set to 20, we observe a similar logical error rate across all available BP methods, with the *product-sum* method giving a good trade-off between decoding time and logical error rate, see Fig. 24(a). The *maximum number of*

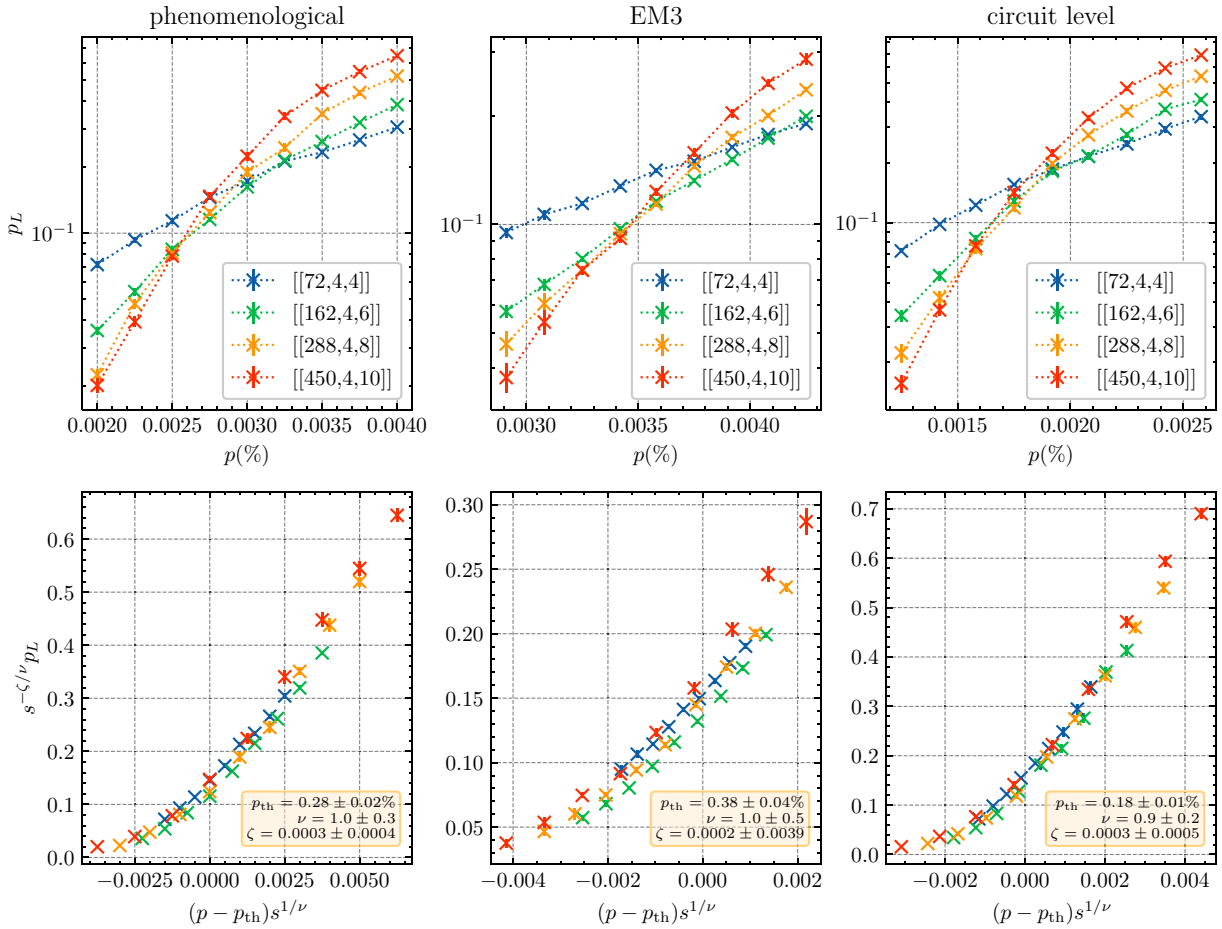


FIG. 25. Finite-size scaling analysis of thresholds, for $\mathcal{L}_Y$ memory experiments with d cycles and the three circuit and noise model defined in Sec. V D. We make the ansatz $p_L(p) = s^{\zeta/\nu} f(s^{1/\nu}(p - p_{th}))$, where s is the linear system size $s = 2d$. The `pyfssa` package then determines the critical exponents and threshold that gives the best data collapse.

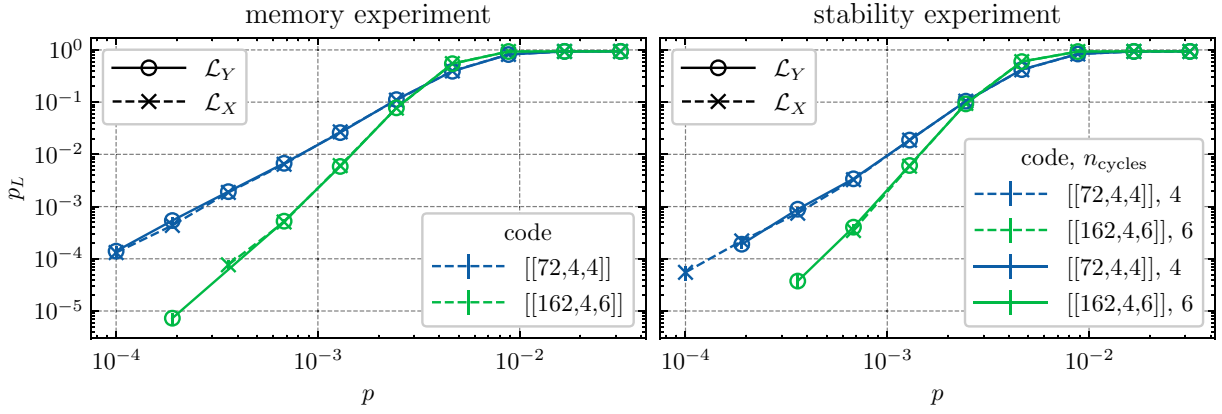


FIG. 26. Memory (left) and stability (right) experiments for $\mathcal{L}_X$ observables compared to $\mathcal{L}_Y$ observables shown in the main text in Sec. V F show that within error bars, the performance coincides.

BP iterations shows oscillatory behavior: An odd number shows significantly lower logical error rates compared to an even number of maximum iterations. We do not investigate this behavior any further here and fix the number of maximum iterations to 19 in all of our simulations.

APPENDIX E: FINITE-SIZE SCALING ANALYSIS OF THRESHOLDS

We perform finite-size scaling analysis on the logical error rate of $\mathcal{L}_Y$ memory experiments using the open-source package `pyfssa` [127]. We make the ansatz for a scaling in the form of $p_L(p) = s^{\zeta/\nu} f(s^{1/\nu}(p - p_{\text{th}}))$, where s is the linear system size $s = 2d$ and $f(x)$ a linear dimensionless function. The `pyfssa` package then determines the critical exponents and threshold that gives the best data collapse. The results are shown in Fig. 25. We extract critical exponents of approximately equal to 1 thresholds for the phenomenological noise model of

$$p_{\text{th}}^{\text{ph}} \approx 0.28 \pm 0.02\%. \quad (\text{E1})$$

For the EM3 noise model we find

$$p_{\text{th}}^{\text{em3}} \approx 0.38 \pm 0.04\%, \quad (\text{E2})$$

and for the circuit-level noise model

$$p_{\text{th}}^{\text{cl}} \approx 0.18 \pm 0.01\%. \quad (\text{E3})$$

Note that because BP + postprocessing is not an MLD decoder, these numbers do not represent fundamental (i.e., optimal) thresholds of the code. The collapse is evidently not perfect, which we attribute to the suboptimality of the decoder. We, however, still use this method to get a reasonable estimate, including uncertainty, for the crossing point of the logical error curves.

APPENDIX F: $\mathcal{L}_X$ EXPERIMENTS

In Fig. 26 we show that the $\mathcal{L}_X$ memory and stability observables perform within error bars the same as the $\mathcal{L}_Y$ observables shown in the main text Sec. V F. This is expected due to the symmetry of the code and protocols.

-
- [1] B. M. Terhal, Quantum error correction for quantum memories, *Rev. Mod. Phys.* **87**, 307 (2015).
 - [2] A. M. Steane, Error correcting codes in quantum theory, *Phys. Rev. Lett.* **77**, 793 (1996).
 - [3] M. Takita, A. W. Cross, A. D. Córcoles, J. M. Chow, and J. M. Gambetta, Experimental demonstration of fault-tolerant state preparation with superconducting qubits, *Phys. Rev. Lett.* **119**, 180501 (2017).
 - [4] P. Campagne-Ibarcq, A. Eickbusch, S. Touzard, E. Zalys-Geller, N. E. Frattini, V. V. Sivak, P. Reinhold, S. Puri, S. Shankar, R. J. Schoelkopf, L. Frunzio, M. Mirrahimi, and M. H. Devoret, Quantum error correction of a qubit encoded in grid states of an oscillator, *Nature* **584**, 368 (2020).
 - [5] S. Krinner, N. Lacroix, A. Remm, A. Di Paolo, E. Genois, C. Leroux, C. Hellings, S. Lazar, F. Swiadek, J. Herrmann, G. J. Norris, C. K. Andersen, M. Müller, A. Blais, C. Eichler, and A. Wallraff, Realizing repeated quantum error correction in a distance-three surface code, *Nature* **605**, 669 (2022).
 - [6] Y. Zhao, *et al.*, Realization of an error-correcting surface code with superconducting qubits, *Phys. Rev. Lett.* **129**, 030501 (2022).
 - [7] G. Q. Ai, Suppressing quantum errors by scaling a surface code logical qubit, *Nature* **614**, 676 (2023).
 - [8] L. Postler, S. Heußen, I. Pogorelov, M. Risppler, T. Feldker, M. Meth, C. D. Marciniak, R. Stricker, M. Ringbauer, R. Blatt, P. Schindler, M. Müller, and T. Monz, Demonstration of fault-tolerant universal quantum gate operations, *Nature* **605**, 675 (2022).
 - [9] C. Ryan-Anderson, *et al.*, Implementing fault-tolerant entangling gates on the five-qubit code and the color code, *ArXiv:2208.01863*.

- [10] D. Bluvstein, *et al.*, Logical quantum processor based on reconfigurable atom arrays, [Nature](#) **626**, 58 (2024).
- [11] S. Huang, K. R. Brown, and M. Cetina, Comparing Shor and Steane error correction using the Bacon-Shor code, [Sci. Adv.](#) **10**, eadp2008 (2024).
- [12] L. Postler, F. Butt, I. Pogorelov, C. D. Marciniak, S. Heußen, R. Blatt, P. Schindler, M. Rispler, M. Müller, and T. Monz, Demonstration of fault-tolerant Steane quantum error correction, [PRX Quantum](#) **5**, 030326 (2024).
- [13] D. Gottesman, *Stabilizer Codes and Quantum Error Correction* (California Institute of Technology, Pasadena, California, 1997).
- [14] M. B. Hastings and J. Haah, Dynamically generated logical qubits, [Quantum](#) **5**, 564 (2021).
- [15] D. Kribs, R. Laflamme, and D. Poulin, Unified and generalized approach to quantum error correction, [Phys. Rev. Lett.](#) **94**, 180501 (2005).
- [16] C. Vuillot, Planar Floquet codes, [ArXiv:2110.05348](#).
- [17] A. Townsend-Teague, J. Magdalena de la Fuente, and M. Kesselring, Floquetifying the colour code, [Electron. Proc. Theor. Comput. Sci.](#) **384**, 265 (2023).
- [18] E. Dennis, A. Kitaev, A. Landahl, and J. Preskill, Topological quantum memory, [J. Math. Phys.](#) **43**, 4452 (2002).
- [19] M. S. Kesselring, J. C. M. de la Fuente, F. Thomsen, J. Eisert, S. D. Bartlett, and B. J. Brown, Anyon condensation and the color code, [PRX Quantum](#) **5**, 010342 (2024).
- [20] M. Davydova, N. Tantivasadakarn, S. Balasubramanian, and D. Aasen, Quantum computation from dynamic automorphism codes, [Quantum](#) **8**, 1448 (2024).
- [21] T. D. Ellison, J. Sullivan, and A. Dua, Floquet codes with a twist, [ArXiv:2306.08027](#).
- [22] A. Dua, N. Tantivasadakarn, J. Sullivan, and T. D. Ellison, Engineering 3D Floquet codes by rewinding, [PRX Quantum](#) **5**, 020305 (2024).
- [23] Z. Zhang, D. Aasen, and S. Vijay, X -cube Floquet code: A dynamical quantum error correcting code with a subextensive number of logical qubits, [Phys. Rev. B](#) **108**, 205116 (2023).
- [24] D. Aasen, Z. Wang, and M. B. Hastings, Adiabatic paths of Hamiltonians, symmetries of topological order, and automorphism codes, [Phys. Rev. B](#) **106**, 085122 (2022).
- [25] A. Bauer, Topological error correcting processes from fixed-point path integrals, [Quantum](#) **8**, 1288 (2024).
- [26] J. van de Wetering, ZX calculus for the working quantum computer scientist, [ArXiv:2012.13966](#).
- [27] H. Bombín and M. A. Martin-Delgado, Exact topological quantum order in $d = 3$ and beyond: Branyons and brane-net condensates, [Phys. Rev. B](#) **75**, 075103 (2007).
- [28] H. Bombín and M. A. Martin-Delgado, Topological quantum distillation, [Phys. Rev. Lett.](#) **97**, 180501 (2006).
- [29] S. Roberts and D. J. Williamson, 3-fermion topological quantum computation, [PRX Quantum](#) **5**, 010315 (2024).
- [30] T. D. Ellison, Y.-A. Chen, A. Dua, W. Shirley, N. Tantivasadakarn, and D. J. Williamson, Pauli topological subsystem codes from Abelian anyon theories, [Quantum](#) **7**, 1137 (2023).
- [31] H. Bombín, D. Litinski, N. Nickerson, F. Pastawski, and S. Roberts, Unifying flavors of fault tolerance with the ZX calculus, [Quantum](#) **8**, 1379 (2024).
- [32] L. Wang and Z. Wang, In and around Abelian anyon models, [J. Phys. A](#) **53**, 505203 (2020).
- [33] M. A. Nielsen and I. L. Chuang, Quantum computation and quantum information, [Phys. Today](#) **54**, 60 (2001).
- [34] D. Gottesman, The Heisenberg representation of quantum computers, [ArXiv:quant-ph/9807006](#).
- [35] D. Aasen, J. Haah, Z. Li, and R. S. K. Mong, Measurement quantum cellular automata and anomalies in Floquet codes, [ArXiv:2304.01277](#).
- [36] A. Kitaev, Anyons in an exactly solved model and beyond, [Ann. Phys.](#) **321**, 2 (2006).
- [37] S. Bravyi and M. B. Hastings, A short proof of stability of topological order under local perturbations, [Commun. Math. Phys.](#) **307**, 609 (2011).
- [38] S. Bravyi and B. Terhal, A no-go theorem for a two-dimensional self-correcting quantum memory based on stabilizer codes, [New J. Phys.](#) **11**, 043029 (2009).
- [39] H. Bombín, Structure of 2D topological stabilizer codes, [Commun. Math. Phys.](#) **327**, 387 (2014).
- [40] H. Bombín, G. Duclos-Cianci, and D. Poulin, Universal topological phase of two-dimensional stabilizer codes, [New J. Phys.](#) **14**, 073048 (2012).
- [41] M. S. Kesselring, F. Pastawski, J. Eisert, and B. J. Brown, The boundaries and twist defects of the color code and their applications to topological quantum computation, [Quantum](#) **2**, 101 (2018).
- [42] A. Y. Kitaev, Fault-tolerant quantum computation by anyons, [Ann. Phys.](#) **303**, 2 (2003).
- [43] This choice of generator is not unique and not even minimal but makes many aspects of the analysis of the anyon model easier.
- [44] A. Kubica, B. Yoshida, and F. Pastawski, Unfolding the color code, [New J. Phys.](#) **17**, 083026 (2015).
- [45] H. Bombín, Topological subsystem codes, [Phys. Rev. A](#) **81**, 032301 (2010).
- [46] M. Kargarian, H. Bombín, and M. A. Martin-Delgado, Topological color codes and two-body quantum lattice Hamiltonians, [New J. Phys.](#) **12**, 025018 (2010).
- [47] Given a pair of locally reversible stabilizer groups, there exists a locality preserving unitary mapping between them, see Ref. [35]. Here, we have two reversible stabilizer groups where one is locally generated and the other one is not. Any unitary mapping between them cannot be locality preserving, contradicting local reversibility.
- [48] We want to note that the same phenomenon in the 3D X -cube Floquet code was coined “splitting” in Ref. [22].
- [49] For one choice of logical basis we can view the nonlocal string operators as logical operators $\bar{Z}_i \bar{Z}_j \bar{Z}_k$ for some triples of logical qubits (i, j, k) .
- [50] J. Haah and M. B. Hastings, Boundaries for the honeycomb code, [Quantum](#) **6**, 693 (2022).
- [51] A similar concept has been defined as “stabilizer flow” in Ref. [116], “Pauli webs” in Ref. [31] or identified with stabilizers of tensors in Refs. [115, 128].
- [52] R. Yeung, Diagrammatic design and study of ansätze for quantum machine learning, [ArXiv:2011.11073](#).
- [53] A. Lang and B. Coecke, Trichromatic open digraphs for understanding qubits, [Electron. Proc. Theor. Comput. Sci.](#) **95**, 193 (2012).
- [54] Note the different color convention cf. ZX calculus [26].

- [55] Note that these tensors do not have to be neighbors in the network but an arbitrary subset of tensors.
- [56] In Appendix A 2, we show the steps needed to see that this diagram represents the teleportation circuit starting from a diagram that is closer to the circuit presented above.
- [57] Note that this is not a symmetry group of operators but a group of sequences of transformations of the network that can be related to inserting certain tensors and propagating them through. For more details, see Appendix A 3 d. This is related to the notion of symmetric circuits in Ref. [101] but not equivalent in that not every symmetry in the tensor networks of Ref. [101] corresponds to a detector flow.
- [58] Concretely, the circuit with the fault maps a logical X operator to a logical $-X$ operator, which can also be understood as acting with a logical Z operator.
- [59] H. Bombin, C. Dawson, R. V. Mishmash, N. Nickerson, F. Pastawski, and S. Roberts, Logical blocks for fault-tolerant topological quantum computation, *PRX Quantum* **4**, 020303 (2023).
- [60] Note that we mostly think of numerical experiments and simulations, where the true dynamics of the system is known. However, the same formal structure applies to real-life experiments.
- [61] C. Gidney, Stim: A fast stabilizer circuit simulator, *Quantum* **5**, 497 (2021).
- [62] P.-J. H. S. Derks, A. Townsend-Teague, A. G. Burchards, and J. Eisert, Designing fault-tolerant circuits using detector error models, *ArXiv:2407.13826*.
- [63] N. Delfosse and A. Paetzniak, Spacetime codes of Clifford circuits, *ArXiv:2304.05943*.
- [64] Each generator of a subgroup of detectors D , respectively D_d , can be associated to a $\mathbb{Z}_2$ vector in M . The parity-check matrix H_D , respectively $H_{D_d}^d$, is the matrix whose rows are the $\mathbb{Z}_2$ vectors associated to the generators of D , respectively D_d . This construction leads to $C_D = \ker(H_D)$ and $C_{D_d}^d = \ker(H_{D_d}^d)$.
- [65] For example, the detector error model in `stim` matches the syndrome onto a decoding graph obtained from the error model. Note that this graph can be easily constructed from the tensor-network representation of the probing experiment and the noise model. For details on this, see Sec. V D.
- [66] C. Gidney, Stability experiments: The overlooked dual of memory experiments, *Quantum* **6**, 786 (2022).
- [67] B. J. Brown, Conservation laws and quantum error correction: Towards a generalised matching decoder, *IEEE Bits Inf. Theory Mag.* **2**, 5 (2023).
- [68] When we talk about the 18-site plaque, we use the notion of *inner* to refer to edges related to the six qubits (or edges) forming the hexagon of the plaquette and *outer* regarding the 12 qubits (or edges) around the outside of the plaque.
- [69] A similar approach was used to argue about fault tolerance in the honeycomb code [14].
- [70] In the literature, minimum fault configurations that do not violate any detectors are also called elementary equivalences, see, e.g., Ref. [74].
- [71] A. M. Stephens, Efficient fault-tolerant decoding of topological color codes, *ArXiv:1402.3037*.
- [72] C. Gidney and C. Jones, New circuits and an open source decoder for the color code, *ArXiv:2312.08813*.
- [73] S.-H. Lee, A. Li, and S. D. Bartlett, Color code decoder with improved scaling for correcting circuit-level noise, *Quantum* **9**, 1609 (2025).
- [74] R. S. Andrist, H. G. Katzgraber, H. Bombin, and M. A. Martin-Delgado, Tricolored lattice gauge theory with randomness: Fault tolerance in topological color codes, *New J. Phys.* **13**, 083006 (2011).
- [75] A. A. Kovalev and L. P. Pryadko, Fault tolerance of quantum low-density parity check codes with sublinear distance scaling, *Phys. Rev. A* **87**, 020304 (2013).
- [76] C. Gidney, M. Newman, A. Fowler, and M. Broughton, A fault-tolerant honeycomb memory, *Quantum* **5**, 605 (2021).
- [77] C. Knapp, M. Beverland, D. I. Pikulin, and T. Karzig, Modeling noise and error correction for Majorana-based quantum computing, *Quantum* **2**, 88 (2018).
- [78] This is done in `stim` using `circuit.detector_error_model()`.
- [79] Using `stim's CompiledDetectorSampler()`.
- [80] P. Panteleev and G. Kalachev, Degenerate quantum LDPC codes with good finite length performance, *Quantum* **5**, 585 (2021).
- [81] J. Roffe, LDPC: Python tools for low density parity check codes, 2022, <https://pypi.org/project/ldpc/>.
- [82] T. Hillmann, L. Berent, A. O. Quintavalle, J. Eisert, R. Wille, and J. Roffe, Localized statistics decoding: A parallel decoding algorithm for quantum low-density parity-check codes, *ArXiv:2406.18655*.
- [83] C. Gidney, M. Newman, and M. McEwen, Benchmarking the planar honeycomb code, *Quantum* **6**, 813 (2022).
- [84] D. S. Wang, A. G. Fowler, C. D. Hill, and L. C. Hollenberg, Graphical algorithms and threshold error rates for the 2D color code, *Quantum Inf. Comput.* **10**, 780 (2010).
- [85] C. Chamberland, A. Kubica, T. J. Yoder, and G. Zhu, Triangular color codes on trivalent graphs with flag qubits, *New J. Phys.* **22**, 023019 (2020).
- [86] O. Higgott and N. P. Breuckmann, Improved single-shot decoding of higher-dimensional hypergraph-product codes, *PRX Quantum* **4**, 020332 (2023).
- [87] X. Tan, F. Zhang, R. Chao, Y. Shi, and J. Chen, Scalable surface-code decoders with parallelization in time, *PRX Quantum* **4**, 040344 (2023).
- [88] L. Skoric, D. E. Browne, K. M. Barnes, N. I. Gillespie, and E. T. Campbell, Parallel window decoding enables scalable fault tolerant quantum computation, *Nat. Commun.* **14**, 7040 (2023).
- [89] K. Meinerz, C.-Y. Park, and S. Trebst, Scalable neural decoder for topological surface codes, *Phys. Rev. Lett.* **128**, 080505 (2022).
- [90] K. Sahay and B. J. Brown, Decoder for the triangular color code by matching on a Möbius strip, *PRX Quantum* **3**, 010310 (2022).
- [91] D. Horsman, A. G. Fowler, S. Devitt, and R. Van Meter, Surface code quantum computing by lattice surgery, *New J. Phys.* **14**, 123011 (2012).
- [92] A. Kapustin and N. Saulina, Topological boundary conditions in Abelian Chern–Simons theory, *Nucl. Phys. B* **845**, 393 (2011).
- [93] M. Levin, Protected edge modes without symmetry, *Phys. Rev. X* **3**, 021009 (2013).

- [94] Note that in order for the resulting protocol to have fault tolerant *corners*, the interfaces between the boundaries (one-dimensional objects in spacetime) have to be designed accordingly.
- [95] J. Haah, Classification of translation invariant topological Pauli stabilizer codes for prime dimensional qudits on two-dimensional lattices, *J. Math. Phys.* **62**, 012201 (2021).
- [96] A. Kapustin and L. Fidkowski, Local commuting projector Hamiltonians and the quantum Hall effect, *Commun. Math. Phys.* **373**, 763 (2019).
- [97] Note that we have not rigorously defined yet how to make sense of an “anyon model” or an anomalous 1-form symmetry of a three-dimensional topological protocol. For the tensor networks constructed in Ref. [101], the anyon model describing the protocol follows from the construction via (twisted) Abelian gauge theories. We aim to provide a constructive framework to identify the anyon model represented by more generic (families of) topological protocols in future work.
- [98] M. Davydova, N. Tantivasadakarn, and S. Balasubramanian, Floquet codes without parent subsystem codes, *PRX Quantum* **4**, 020341 (2023).
- [99] S. Roberts, B. Yoshida, A. Kubica, and S. D. Bartlett, Symmetry-protected topological order at nonzero temperature, *Phys. Rev. A* **96**, 022306 (2017).
- [100] J. K. Iverson and J. Preskill, Coherence in logical quantum channels, *New J. Phys.* **22**, 073066 (2020).
- [101] A. Bauer, Low-overhead non-Clifford topological fault-tolerant circuits for all non-chiral Abelian topological phases, *ArXiv:2403.12119*.
- [102] Note that Ref. [63] introduces the notion of an LDPC spacetime code. Although it refers to a stabilizer code derived from detector flows, the notion of LDPC is the same: What we call a spacetime LDPC protocol will give rise to a LDPC spacetime code.
- [103] C. Gidney and D. Bacon, Less bacon more threshold, *ArXiv:2305.12046*.
- [104] J. Haah, Local stabilizer codes in three dimensions without string logical operators, *Phys. Rev. A* **83**, 042330 (2011).
- [105] B. J. Brown and D. J. Williamson, Parallelized quantum error correction with fracton topological codes, *Phys. Rev. Res.* **2**, 013303 (2020).
- [106] For a rigorous mapping, we refer to the map ρ and $\tilde{p}$ in Appendix A 3 e.
- [107] A. Hamma, P. Zanardi, and X.-G. Wen, String and membrane condensation on three-dimensional lattices, *Phys. Rev. B* **72**, 035307 (2005).
- [108] H. Bombin, Single-shot fault-tolerant quantum error correction, *Phys. Rev. X* **5**, 031043 (2015).
- [109] J. C. Bridgeman, A. Kubica, and M. Vasmer, Lifting topological codes: Three-dimensional subsystem codes from two-dimensional anyon models, *PRX Quantum* **5**, 020310 (2024).
- [110] H. Zhou, C. Zhao, M. Cain, D. Bluvstein, C. Duckering, H.-Y. Hu, S.-T. Wang, A. Kubica, and M. D. Lukin, Algorithmic fault tolerance for fast quantum computing, *ArXiv:2406.17653*.
- [111] L. Z. Cohen, I. H. Kim, S. D. Bartlett, and B. J. Brown, Low-overhead fault-tolerant quantum computing using long-range connectivity, *Sci. Adv.* **8**, eabn1717 (2022).
- [112] F. Thomsen, M. S. Kesselring, S. D. Bartlett, and B. J. Brown, Low-overhead quantum computing with the color code, *Phys. Rev. Res.* **6**, 043125 (2024).
- [113] K. Walker and Z. Wang, (3+1)-TQFTs and topological insulators, *Front. Phys.* **7**, 150 (2011).
- [114] D. J. Williamson, 1-form spts & measurement-based quantum computation, 2021, <https://www.youtube.com/watch?V=gT9BP-a90Jo>, talk at Harvard CMSA.
- [115] R. Raussendorf, C. Okay, D.-S. Wang, D. T. Stephen, and H. P. Nautrup, Computationally universal phase of quantum matter, *Phys. Rev. Lett.* **122**, 090501 (2019).
- [116] M. McEwen, D. Bacon, and C. Gidney, Relaxing hardware requirements for surface code circuits using time-dynamics, *Quantum* **7**, 1172 (2023).
- [117] The relations on the red and blue tensors are special cases of the ZX-calculus rules. For completeness, we still include them here.
- [118] Note that this is just a usual ZX rule [26].
- [119] D. Gorenstein, *Finite Groups* (AMS Chelsea Publishing, American Mathematical Society, Providence, Rhode Island, 2007), Vol. 301.
- [120] This can be proven using the fact that it is isomorphic to a group generated by products of X and Z operators and there is no Pauli operator besides potentially $-\mathbb{1}$ that commutes with all. This allows us to iteratively construct a set of mutually anticommuting Pauli operators.
- [121] These subspaces can be identified with the images of minimal central idempotents in $\mathcal{S}$.
- [122] Note that this can directly be generalized to density matrices by considering the set of square matrices containing a representation of the algebra above where the algebra acts via conjugation, not via left multiplication as for state vectors.
- [123] Equivalently, t can be interpreted as measuring $S_\ell^{(t)}$ on a code stabilized by $S_\ell^{(t')}$. The symmetry becomes apparent in the proof.
- [124] C. Vuillot, L. Lao, B. Criger, C. García Almudéver, K. Bertels, and B. M. Terhal, Code deformation and lattice surgery are gauge fixing, *New J. Phys.* **21**, 033028 (2019).
- [125] Note that the definition of the maps in the top row are only defined up to choosing a representative of a coset $pS_A^{(t)}$. This gives a different element added to the stabilizer group for that specific p . The full stabilizer group constructed will be independent of that choice, however, since $S_A^{(t)} \otimes \mathbb{1}$ and $\mathbb{1} \otimes S_B^{(t')}$ are included in $\mathcal{S}$ and is hence uniquely given by the diagram and the associated construction.
- [126] Note, however, that any RGB TN is equivalent to a network that can directly be translated back to a circuit using the rewrite rules discussed in Sec. A 2, potentially acting on additional qubits.
- [127] A. Sorge, Pyfssa 0.7.6, 2015, <https://doi.org/10.5281/zenodo.35293>.
- [128] C. Cao and B. Lackey, Quantum Lego: Building quantum error correction codes from tensor networks, *PRX Quantum* **3**, 020332 (2022).