

## Anyon Condensation and the Color Code

Markus S. Kesselring,<sup>1,\*</sup> Julio C. Magdalena de la Fuente,<sup>1</sup> Felix Thomsen<sup>1b</sup>,<sup>2</sup> Jens Eisert,<sup>1,3</sup>  
Stephen D. Bartlett,<sup>2</sup> and Benjamin J. Brown<sup>1b</sup><sup>2</sup>

<sup>1</sup>*Dahlem Center for Complex Quantum Systems, Freie Universität Berlin, Berlin 14195, Germany*

<sup>2</sup>*Centre for Engineered Quantum Systems, School of Physics, University of Sydney, Sydney,  
New South Wales 2006, Australia*

<sup>3</sup>*Helmholtz-Zentrum Berlin für Materialien und Energie, Berlin 14109, Germany*

 (Received 6 February 2023; accepted 8 February 2024; published 11 March 2024)

The manipulation of topologically ordered phases of matter to encode and process quantum information forms the cornerstone of many approaches to fault-tolerant quantum computing. Here we demonstrate that fault-tolerant logical operations in these approaches can be interpreted as instances of anyon condensation. We present a constructive theory for anyon condensation and, in tandem, illustrate our theory explicitly using the color-code model. We show that different condensation processes are associated with a general class of domain walls, which can exist in both spacelike and timelike directions. This class includes semitransparent domain walls that condense certain subsets of anyons. We use our theory to classify topological objects and design novel fault-tolerant logic gates for the color code. As a final example, we also argue that dynamical “Floquet codes” can be viewed as a series of condensation operations. We propose a general construction for realizing planar dynamically driven codes based on condensation operations on the color code. We use our construction to introduce a new Calderbank-Shor-Steane-type Floquet code that we call the Floquet color code.

DOI: [10.1103/PRXQuantum.5.010342](https://doi.org/10.1103/PRXQuantum.5.010342)

### I. INTRODUCTION

Topological quantum error-correcting codes [1–6] have provided the basis of many promising approaches to realize a fault-tolerant quantum computer. These codes are based on topological phases that robustly store quantum states in nonlocal degrees of freedom [7,8]. Additionally, there exist a number of distinct ways of performing logical operations on topologically protected quantum states, using unitary dynamics [2,3], measurement-based methods [9–15], or combinations thereof [16–18]. Performing these logical operations, however, is generally very resource intensive and so it remains a significant technical challenge to realize these current designs for quantum computing architectures. It is therefore important to develop novel methods of implementing robust operations that are available with topological phases in order to find more practical ways of performing the logical operations that we need for universal fault-tolerant quantum computing.

The *color code* [5] is a topological quantum error-correcting code with a rich structure that can be harnessed for topological quantum computing. Its study was first motivated by its multitude of available transversal logic gates. In addition to these local constant-depth unitary logical operations, which are inherently fault tolerant, the color code can also demonstrate various fault-tolerant measurement-based code deformations [10,12–15]. All together, these operations can be combined to give universal low-overhead fault-tolerant quantum computing [12, 13,19,20], with resource requirements that are favorable over those of the surface code.

The versatility of the color code for performing fault-tolerant logic gates can be attributed to the underlying symmetries among its quasiparticle excitations when viewed as a topological model [21–23]. Furthermore, the color code can be decomposed into copies of more elementary phases [23–27]. All together, these properties mean that the color code offers an excellent test bed both to design practical ways of performing fault-tolerant quantum computation as well as to investigate the fundamental phenomena of topological phases that enable robust logical operations.

In this work, we develop a theory of topological quantum computing in terms of *anyon condensation* [28–32] for stabilizer codes, where we use the color code as a guiding example. Anyon condensation implements a special type

\*markus.kesselring@fu-berlin.de

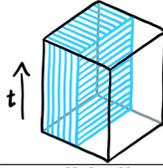
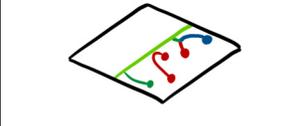
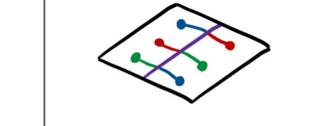
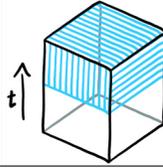
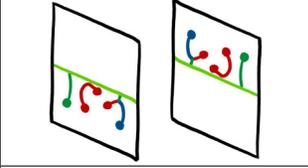
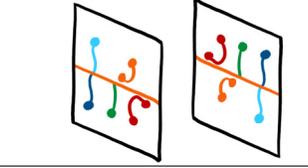
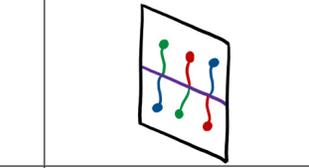
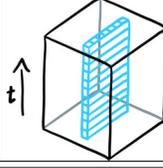
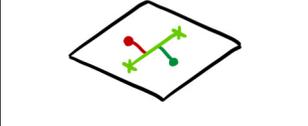
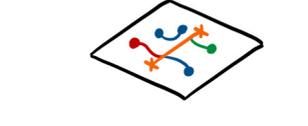
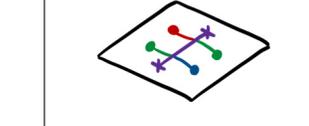
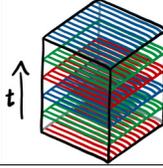
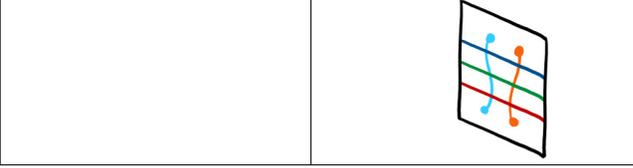
*Published by the American Physical Society under the terms of the Creative Commons Attribution 4.0 International license. Further distribution of this work must maintain attribution to the author(s) and the published article’s title, journal citation, and DOI.*

of topological phase transition by identifying a subset of anyons of the phase with the vacuum particle of a condensate. We find that anyon condensation offers a natural way of describing many aspects of quantum computation with topological stabilizer codes. In particular, it offers a concise and unified language for the logical operations available in the color code, as well as other code-deformation and code-switching schemes. Furthermore, the scheme we present to perform anyon condensation is constructive in microscopic lattice models, i.e., it allows us to construct topological stabilizer codes exemplifying and implementing the various features and operations described abstractly by anyon condensation. In Table I and the following

paragraphs, we give an overview of the results obtained from implementing anyon condensation in the color code in various ways.

As a first example, we consider condensing the anyons in a disk-shaped region of the color-code lattice. This allows us to recover the punctures with different types of boundaries that have been discovered earlier in the literature [5,23]. The boundaries that we produce can be viewed as a nonlocal degree of freedom, the state of which is determined by the type of anyons it has condensed, i.e., absorbed through local processes. Punctures provide the standard mechanism to encode quantum information in a topological code and, given these properties, we can

TABLE I. Anyon condensation on topologically ordered phases in various regions of space-time. We distinguish three types of condensation—maximal, partial, or trivial—depending on the number of anyons that are condensed. The different types of condensation are shown in the columns in the table and are discussed in Secs. III B 1–III B 3. We apply condensation to different regions of the space-time lattice, represented by the shaded regions in the figures of the leftmost column, over different sections of the work. In the first row, we condense anyons in the spatial bulk. This introduces features such as punctures, semipunctures, and invertible domain walls (Sec. IV). In the second row, we introduce temporal domain walls using anyon condensation over the entire bulk at an instant in time. This gives us protocols to initialize, manipulate, and read out logical qubits in the color code (Sec. V). In the third row, we condense anyons across a narrow subregion of the bulk, in order to obtain end points of domain walls (Sec. VI). Finally, we perform a dynamic condensation, where different anyons are condensed at different time steps. Using the partial condensation in the color code, we obtain a general construction for dynamically driven codes through this procedure (Sec. VII).

	Maximal (Sec. III B 1)	Partial (Sec. III B 2)	Trivial (Sec. III B 3)
Partial bulk (Sec. IV) 	Boundaries and punctures (Sec. IV A) 	Condensates and semipunctures (Sec. IV B) 	Invertible domain walls (Sec. IV C) 
Full bulk (Sec. V) 	Readout or initialization (Secs. V B and V C) 	Partial readout or initialization (Sec. V F) 	Transversal gates (Sec. V A) 
1D line (Sec. VI) 	Corners (Sec. VI B) 	Semitransparent domain walls (Sec. VI C) 	Twist defects (Sec. VI A) 
Dynamical (Sec. VII) 	Floquet codes (Sec. VII) 		

create and braid different punctures to robustly encode and manipulate quantum information.

In addition to known punctures, the theory of anyon condensation allows us to generalize the types of punctures that can be produced in the color code. Specifically, we discover what we coin a semipuncture, where only a single boson of the color code is condensed within a disk-shaped region. We regard these new objects as semipunctures in the sense that a pair of semipunctures, of the appropriate type, can be combined to give known types of punctures. Indeed, a standard puncture is obtained by condensing a larger subset of color-code bosons in some region. As we will show, new types of code deformation become available by manipulating these generalized punctures, leading to new approaches to fault-tolerant logic.

We also view bulk-condensation operations from a different perspective by examining how a topological phase transforms over time. In general, we observe a domain wall in a  $(2 + 1)$ -dimensional  $[(2 + 1)D]$  space-time picture as the system undergoes a phase transition between two distinct phases. In the example of the color code, we can condense a maximal subset of bosons, formally known as a Lagrangian subgroup of the anyon model, which transforms the phase onto the vacuum phase. Alternatively, we can also condense a smaller subset of bosons, such that we transform the system onto the toric-code phase. These operations all have important applications in fault-tolerant quantum computation, for logical state readout or, inversely, state preparation. As we will show, we find that the anyon-condensation perspective shows us how to address individual color-code logical qubits for readout operations, by condensing a smaller subsets of bosons to perform logical measurements. We touch on the interplay between spatial and temporal boundaries and how different boundary configurations relate to the fault tolerance of a given protocol. As an example, we show how the theory of anyon condensation can be used to design more general stability experiments [33] to evaluate the error-correction capabilities of the color code as it undergoes fault-tolerant logical operations.

Anyon condensation is also invaluable in the classification of twist defects. Twist defects are obtained by terminating a domain wall that connects a topological phase to itself. In earlier work [21–23], 72 twist defects have been shown to exist when the color-code phase is connected trivially to itself. More generally though, we find that additional domain walls can be obtained by merging a phase to itself via a nontrivial condensate [34,35]. This gives rise to a semitransparent domain wall where certain charges can pass through the domain, whereas others either condense or confine. We demonstrate the importance of these more general types of domain wall for fault-tolerant quantum computation by investigating how semitransparent domain

walls appear in lattice-surgery operations with the color code [20].

As a final example, we rederive and generalize recently proposed constructions of dynamically driven “Floquet” codes [36] from the perspective of color-code anyon condensation. We argue that the transformations used to read out the stabilizers of dynamically driven codes can be viewed as a sequence of condensation operations, where at each step we condense a different color-code anyon. Our construction enables us to discover more general classes of dynamically driven codes. We propose one such example that we call the Floquet color code; a Calderbank-Shor-Steane– (CSS) type Floquet code on the honeycomb lattice. We find that since our construction is based on the well-studied color code, we obtain a constructive way of designing the boundary stabilizers of dynamically driven codes [37], by appealing to the physics of the parent color-code theory. Furthermore, we present numerical results showing that the Floquet color code has a threshold that is very competitive with other known Floquet codes [38]. We remark that the Floquet color code has independently been discovered in other very recent work [39,40]. Furthermore, we note that a recent experiment has demonstrated an error detection measurement for the Floquet color code [41].

### A. A guide for the reader

We develop the theory of anyon condensation for Abelian-anyon models in the earlier sections of this work and we investigate various instances of anyon condensation with the color code, and its applications to fault-tolerant quantum computing, in the following sections of the paper. We have therefore written the latter sections of the paper in a self-contained way, assuming that the reader is familiar with the theory we present in the former sections. We summarize this structure in Table I, where the different examples of anyon condensation in the latter sections of the paper are presented in the rows of the table, with respect to the different types of anyon condensation that are represented by the columns of the table. Furthermore, we offer the following guide for the reader to navigate through the various sections of the paper, where we emphasize the dependence of the latter sections on requisite material from former sections.

In Sec. II, we review the requisite background for the color-code model and we identify its keys properties that enable it to give rise to a number of nontrivial condensation operations. We then give a general theory for anyon condensation in Sec. III. Specifically, we distinguish between maximal, partial, and trivial condensation. These different types of anyon condensation are distinguished by how excitations are transmitted across the domain wall that is produced by the condensation operation. Again, the different types of condensation are represented by the columns of Table I.

Assuming that the reader is familiar with the material presented in Secs. II and III, the microscopic examples of condensation in the following sections can be read independently. Indeed, Secs. IV and V are self-contained. Section IV investigates domain walls that spatially separate the color code from one of its condensates—the vacuum phase, the toric code, or the color code itself. This is represented by the first row of Table I. Then, in Sec. V, we investigate the different types of condensation over timelike domain walls, as shown in the second row of Table I.

Section VI builds on the ideas that we begin to develop in Secs. IV and V. In this section, we describe new topological features that are produced by interfacing two phases with an intermediate condensate. Specifically, at the microscopic level, we show that we can make nontrivial domain walls between the color code and itself to produce different types of twist defects, where the color-code phase is interfaced with a nontrivial condensate. This construction is represented in the third row of Table I.

Finally, we discuss our Floquet-code construction from the perspective of anyon condensation in Sec. VII. The picture we present for Floquet codes in terms of condensation operations is represented by the fourth and final row of Table I. This section can be read independently of Secs. IV–VI.

## II. PRELIMINARIES

The color code is a topologically ordered phase of matter that gives rise to anyonic quasiparticle excitations. We start this section by introducing the theory of Abelian-anyon models (Sec. II A) before turning our attention to the color code. We introduce the color-code anyons and a microscopic color-code lattice model (Sec. II B). We relate the color code to another well-known topological phase, the toric code (Sec. II C). In particular, we discuss first how the color code can be unfolded into two decoupled layers of toric codes (Sec. II D). Lastly, we introduce a space-time interpretation of topological error-correcting codes (Sec. II E).

### A. Anyons

Anyons are quasiparticles that exist in two spatial dimensions [2,42]. We denote the set of all anyons of a phase, and the data describing their behavior, by the anyon model  $\mathcal{C}$ . We label single anyons as lower-case letters  $a, b, c \in \mathcal{C}$ . The trivial anyon, or the *vacuum*, which is part of every anyon model, is denoted as  $1$ . Let us now discuss how fusion, exchange, and braiding of anyons is described.

*Fusion* is the process of bringing two anyons close together such that they behave as a third anyon within the same anyon model. We denote fusion by the “ $\times$ ” operation. Here, we concentrate on Abelian-anyon models,

where fusion outcomes are unique. The fusion rules of Abelian anyon models are of the form  $a \times b = c$  for  $a, b, c \in \mathcal{C}$ . Fusion with the vacuum anyon is trivial,  $a \times 1 = a$ . Furthermore, each anyon has an antiparticle with which it fuses to the vacuum. In the topological phases of interest here, namely, the color code and toric code, all anyons are their own antiparticles, such that  $a \times a = 1$ .

*Exchanging* two identical Abelian anyons results in a complex phase. If we exchange the position of a pair of  $a$  anyons, we denote the obtained phase as  $\theta_a$ . This phase is called *spin* and for qubit-stabilizer codes takes values  $\pm 1$ . Anyons for which the self-exchange results in a  $+1$  ( $-1$ ) phase are called *bosons* (*fermions*).

*Braiding* is the process of moving one anyon around another before returning it to its initial position. Let us say that we braid anyon  $a$  around a second stationary anyon  $b$ ; this process results in a phase called *monodromy*, denoted as  $M_{a,b}$ . For the color-code and toric-code phases, the monodromy can only take values  $M_{a,b} = \pm 1$ , allowing us to use a shorthand formulation in which we call braiding either trivial ( $M_{a,b} = +1$ ) or nontrivial ( $M_{a,b} = -1$ ).

In fact, it is worth pointing out that the self-consistency conditions governing Abelian-anyon models lead to a number of redundancies in the data. The identity  $M_{a,b} = \theta_a \theta_b / \theta_{a \times b}$ , for example, lets us determine the self-exchange statistics of any anyon by decomposing it into two different anyons, the spin and relative braid statistics of which are known.

We can define a microscopic local Hamiltonian, composed of commuting Pauli interaction terms acting on qubits such that the ground space is the common  $+1$  eigenspace of all the terms that give rise to a topological phase. We say that violated interaction terms occupy quasiparticle excitations. These excitations may behave like anyons. Unitary rotations create and transport anyons, allowing us to study their fusion, self-exchange, and braiding explicitly [43]. In the next two sections, we study two specific examples of Hamiltonian models, the color code and the toric code.

### B. The color code

In what follows, we introduce a lattice model realizing the color code [5], before turning our attention to the anyonic excitation that it hosts.

We employ the language of stabilizer codes [44] to describe the microscopic lattice models realizing topologically ordered phases. Stabilizer codes are defined by an Abelian subgroup of the Pauli group that we call the stabilizer group. Importantly, the common  $+1$  eigenspace of the elements of the stabilizer group specifies the code space of a code. The group therefore does not include  $-1$ , as this operator has only negative eigenvalues. We measure stabilizer operators to detect errors.

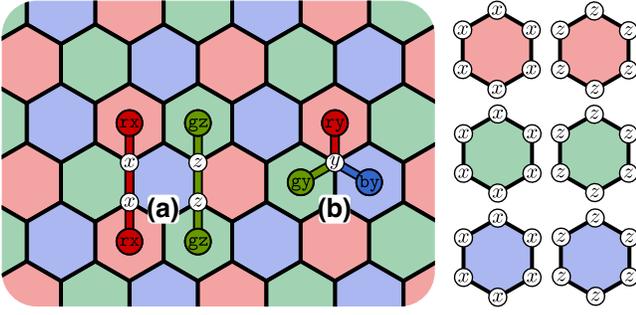


FIG. 1. The lattice model of a color code. A physical qubit is placed on every vertex. As shown on the right, each plaquette hosts two stabilizers, one acting in the  $X$  basis and one in the  $Z$  basis. (a) Pauli rotations on the two qubits of an edge create a pair of anyons on the plaquettes at the end points of said edge. In the displayed examples we show a pair of  $rx$  ( $gz$ ) anyons created at the end points of a string of Pauli- $X$  (Pauli- $Z$ ) rotations supported on a red (green) edge. (b) A single-qubit Pauli rotation creates a triplet of anyons. This process can also be understood as fusing two anyons into the third anyon of the triplet.

The color code can be defined on any trivalent lattice that is three-colorable with respect to its faces. We use the colors red, green, and blue. It is also helpful to assign colors to the edges of the lattice. The color of an edge is given by the color of the faces it connects. In this work, we focus on the hexagonal lattice, as shown in Fig. 1.

To specify the color code, we assign physical qubits to the vertices of the three-colorable lattice and stabilizers are associated to the plaquettes of the lattice. We index the plaquettes with the symbol  $p$ . Each plaquette hosts two stabilizer generators, one of which acts in the  $X$  basis on all qubits supported on the plaquette,  $s_X^p$ , and the other in the  $Z$  basis, denoted  $s_Z^p$ . We will refer to them as  $X$ -type or  $Z$ -type stabilizers, respectively. Note that by multiplying the two stabilizer generators on any plaquette, we obtain a  $Y$ -type stabilizer at plaquette  $p$ , i.e.,  $s_Y^p = s_Z^p s_X^p$ .

Given the stabilizers of a code, we can introduce a commuting Hamiltonian consisting of the negative sum of a set of stabilizer generators. For topological stabilizer codes, these Hamiltonian terms commute and can be chosen to be geometrically local. In the color code, we usually pick the plaquette terms  $s_X^p$  and  $s_Z^p$  acting in the Pauli- $X$  and Pauli- $Z$  basis on all qubits surrounding a plaquette  $p$ , as depicted in Fig. 1. This yields the following Hamiltonian:

$$H_{CC} = - \sum_p s_X^p - \sum_p s_Z^p. \quad (1)$$

The ground-state space of this Hamiltonian coincides with the code space of the stabilizer code. Excited eigenstates are reached when some plaquette terms are violated. These states differ from states in the ground-state space by Pauli

rotations on single qubits. We associate the violated plaquette terms with anyonic excitations and say that they are created by said Pauli rotations.

Let us now discuss color-code anyons and their properties. The color-code phase contains 16 anyons. Apart from the vacuum excitation, there are nine nontrivial bosons. Each boson has one of three color labels,  $r$ ,  $g$ , or  $b$ , as well as one of three Pauli labels,  $x$ ,  $y$ , or  $z$ . The labels are given by the color of the violated plaquette and the basis of the Pauli rotation that creates and moves the anyons. As an example, we say that a red plaquette the stabilizer(s) of which are violated by an applied Pauli- $X$  rotation is associated with an anyon labeled  $rx$  [see, e.g., Fig. 1(a)].

Throughout this work, we find it instructive to order the nine color-code bosons in a  $3 \times 3$  table [23], such that bosons in any given row (column) share their Pauli (color) label. This table is referred to as the color-code boson table:

$$\begin{array}{c|c|c} rx & gx & bx \\ \hline ry & gy & by \\ \hline rz & gz & bz \end{array}. \quad (2)$$

Let us review how the data of the color-code anyons are captured by the boson table in Eq. (2). All color-code anyons are their own antiparticles; hence two identical anyons fuse to the trivial anyon. For example, we obtain  $rx \times rx = 1$ . Bosons that lie in the same row or column fuse to the third boson in said row or column. Examples of such fusions are  $bx \times by = bz$  or  $gy \times by = ry$ . Braiding between two bosons from the same row or column is trivial, e.g.,  $M_{rx,ry} = M_{gz,bz} = +1$ , whereas bosons from differing rows and columns braid nontrivially, e.g.,  $M_{gx,rz} = -1$ .

Fusing two bosons that differ in both Pauli and color label results in one of six fermions:

$$f_1 = rx \times bz = ry \times gz = gx \times by, \quad (3)$$

$$f_2 = rz \times bx = ry \times gx = gz \times by, \quad (4)$$

$$f_3 = bz \times gy = gx \times rz = bx \times ry, \quad (5)$$

$$f_4 = rz \times gy = gx \times bz = rx \times by, \quad (6)$$

$$f_5 = rx \times gy = bx \times gz = by \times rz, \quad (7)$$

$$f_6 = bx \times gy = rx \times gz = ry \times bz. \quad (8)$$

Writing color-code fermions in terms of their composite bosons lets us infer all of their relevant data. For example, using rules that we have defined in Sec. II A), we find that fermions labeled with an even-number label braid trivially with fermions labeled by an odd number, e.g.,  $M_{f_1, f_4} = +1$ , whereas two distinct even (odd) fermions braid nontrivially, e.g.,  $M_{f_3, f_5} = -1$ .

To store quantum information in the color code, we can either place the code on a topologically nontrivial manifold

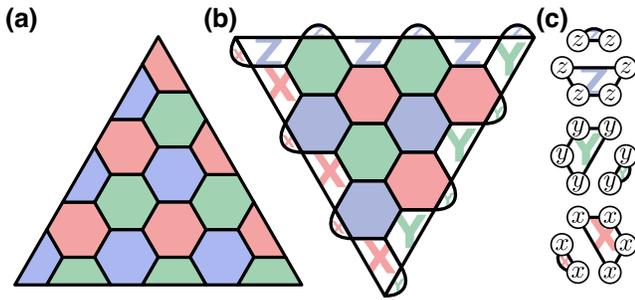


FIG. 2. The triangular color code with (a) three colored boundaries and (b) three Pauli-type boundaries, where the colored boundaries are defined in Ref. [5] and the Pauli boundaries are defined in Ref. [23]. We rederive all of these boundaries in Sec. IV A in terms of anyon condensation. Each code patch encodes one logical qubit with a code distance of  $d = 7$  and  $d = 6$ , respectively. Plaquettes with a single-colored letter host only one stabilizer generator acting in the basis indicated by the letter [see (c)].

or we can introduce boundaries. We define the different boundary types of the color code in terms of anyon condensation in Sec. IV A. The prototypical example of a quantum error-correcting code in the color-code phase is the triangular color code, depicted in Fig. 2(a). We associate a code distance  $d$  with the code, given by the weight of its least-weight logical operator. The code depicted in, e.g., Fig. 2(a) has a code distance  $d = 7$ . Importantly, we have a infinite family of codes that can be parametrized by their code distance. Assuming that we have access to a sensible decoder [24,45–55], we can correct all error configurations that affect fewer than some number of qubits that diverges in the code distance  $d$ . As we discuss in Sec. IV A, logical operators in the color code appear as strings along nontrivial paths. In codes with boundaries, such paths may connect distinct boundaries or enclose punctures. This means that by increasing the system size, i.e., by separating boundaries further apart or enlarging punctures, we can increase the code distance. This in turn means that we can tolerate more errors. Assuming that errors happen sufficiently rarely, we can decrease the probability of a logical error occurring arbitrarily close to zero by increasing the code distance.

### C. The toric code

In this section, we discuss the well known toric code phase; a topologically ordered phase that is closely related to the color code. The toric code, introduced by Kitaev in Ref. [2], is widely regarded as the prototypical example of a topological stabilizer code. Its associated phase is referred to as the toric code phase, or simply the toric code. Here, we will briefly review its anyonic excitations and introduce an example of a microscopic lattice model in the toric code phase.

There are four anyons in the anyon model of the low-energy theory of the toric code. The particle 1 represents the vacuum, or the trivial anyon. Particles  $e$  and  $m$  are bosonic anyons with  $\theta_e = \theta_m = +1$  and  $f$  is a fermion with  $\theta_f = -1$ . The anyons fuse as follows:  $e \times e = m \times m = f \times f = 1$  and  $e \times m = f$ . Any two nonidentical nontrivial anyons braid nontrivially, e.g.,  $M_{e,m} = -1$ .

In Fig. 3, we show a construction for the toric code that is particularly helpful for our discussion throughout this work. We start with a hexagonal lattice, place a physical qubit on each vertex, and color the plaquettes as in Fig. 3. Each plaquette hosts a stabilizer generator acting on the

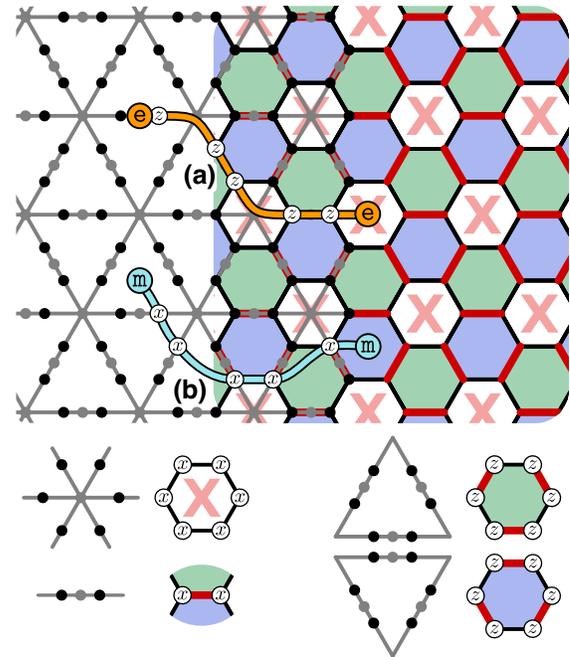


FIG. 3. The lattice model of the toric code. The left-hand side of the figure shows the triangular lattice with additional two-valent vertices, while the right-hand side shows an equivalent representation of the same model on a three-colorable hexagonal lattice. To show their equivalence, the lattice on the left overlays the lattice on the right in the middle of the figure. On the right-hand side of the figure, a physical qubit is placed on every vertex of the hexagonal lattice to the right of the figure. Red edges host a two-body  $XX$  stabilizer. As indicated by the colored letters, red plaquettes are stabilized by an  $X$ -type stabilizer. Blue and green plaquettes host a  $Z$ -type stabilizer. Note how the  $XX$  term on red edges surrounding a blue or a green plaquette multiply to the  $X$ -type stabilizer on said plaquette. Since they host both types of stabilizers, we color the green and blue plaquettes fully. The  $e$  and  $m$  excitations are created and moved by  $Z$  and  $X$  rotations, as is shown in (a) and (b). On the left-hand side, we show the same model defined on the standard toric code lattice, where qubits are placed on the lattice edges [2]. We mark qubits on the gray lattice with large black vertices. Pauli- $X$  (star) stabilizers are associated with the vertices of the gray lattice and Pauli- $Z$  (plaquette) stabilizers to the faces of the model. We give a key for the different types of stabilizer operators at the bottom of the figure.

surrounding qubits in a Pauli basis defined by the color of the plaquette. Green and blue plaquettes host weight-6  $Z$ -type stabilizers. Red plaquettes and red edges host weight-6 and weight-2  $X$ -type stabilizers, respectively. The same stabilizers are obtained when following Kitaev's original toric code construction [2] on a triangular lattice decorated with additional two-valent vertices on every edge, as shown in Fig. 3. In Kitaev's original description of the toric code model, qubits are placed on the edges of some arbitrary lattice. Then, Pauli- $X$  stabilizers are associated with the vertices of the lattice and Pauli- $Z$  stabilizers are associated with the faces of the lattice. Specifically, Pauli- $X$  vertex stabilizers (Pauli- $Z$  plaquette stabilizers) are the product of Pauli- $X$  (Pauli- $Z$ ) terms on qubits associated with edges adjacent to their respective lattice vertex (plaquette). See also the matching-code construction [56], which describes a construction for microscopic stabilizer models in the toric code phase based on Kitaev's honeycomb model [42], including the model described here. The excitations on red plaquettes and edges are  $e$  anyons. The blue and green plaquettes host the  $m$  anyon.

#### D. Unfolding the color code

The color code is equivalent to two decoupled layers of the toric code [24,26,57], meaning that the anyon model from two decoupled layers of toric codes is equivalent to the anyon model of the color code. One way of mapping the color-code anyon labels to the labels given by two layers of toric codes is

$$\begin{array}{c|c|c} & r & g & b \\ \hline x & e1 & ee & 1e \\ \hline y & em & ff & me \\ \hline z & 1m & mm & m1 \end{array}, \quad (9)$$

where  $1, e, m,$  and  $f$  are the toric code anyons and their position in the tuple  $ab$  represents on which of the two layers they live. We refer to Eq. (9) as the standard unfolding. There are, however, 72 valid ways to perform the unfolding, which can be obtained by applying one of the 72 anyon-permuting symmetries of the color code [21,23] to the standard mapping in Eq. (9). Furthermore, we can identify the six fermions of the color code as follows:

$$f_1 = f1, \quad f_3 = ef, \quad f_5 = mf, \quad (10)$$

$$f_2 = 1f, \quad f_4 = fe, \quad f_6 = fm. \quad (11)$$

#### E. Error correction in $(2 + 1)$ D space-time

In two-dimensional (2D) topological stabilizer codes, we detect errors by measuring stabilizer generators to obtain a list of violated stabilizers [1,2]. In reality, however, these stabilizer measurements may be imperfect and may give incorrect outcomes. To achieve tolerance to noise in the presence of measurement errors, we must repeat the

measurements multiple times [1,58]. This transforms the space in which the stabilizer violations live into a  $(2 + 1)$ D space-time.

As we have described, violated stabilizers can be associated with anyonic quasiparticles. This association carries over to the case where we consider the full space-time picture of the stabilizer readouts. In fact, from a condensed-matter perspective, this is very natural, as all 2D topologically ordered phases and processes happening therein are described in as a  $(2 + 1)$ D space-time. In what follows, we make this connection explicit.

We begin by replacing the stabilizer generators associated with plaquettes with *detection cells* [1,58,59]. For simplicity, we assume that all stabilizer readouts are performed in parallel. Each detection cell is associated with one position in space-time  $(s, t)$ , given by the location of a stabilizer generator  $s$  and a time step  $t$ . A detection cell compares the outcome of the measurement of stabilizer  $s$  during the  $t$ th round with the result obtained for  $s$  in the  $(t - 1)$ th round [see Fig. 4(a)]. Thus, they detect changes in measurement outcomes that might be caused by errors on the physical qubits they support or when a faulty measurement result is obtained.

Now, we associate anyonic quasiparticles with (single or sets of) violated stabilizer cells. A detection cell  $(s, t)$  detects errors occurring on physical qubits in the support of  $s$  between time steps  $t - 1$  and  $t$  [see Fig. 4(b)]. This is exactly analogous to the purely 2D viewpoint, as we show it in Fig. 1, where we interpret errors as creating sets of anyons with neutral total charge. Importantly, a detection cell  $(s, t)$  also detects measurement errors affecting

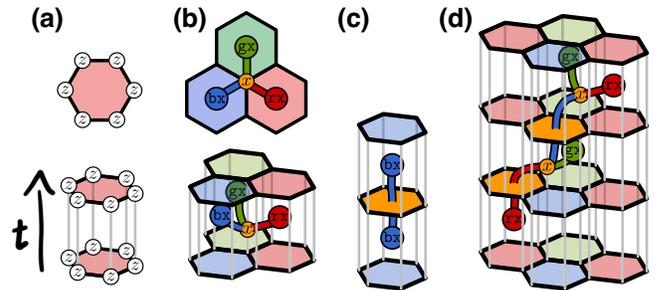


FIG. 4. When performing multiple rounds of stabilizer measurements on a 2D topological error-correcting code, one has to consider the  $(2 + 1)$ D space-time of the process. (a) Stabilizers (top) turn into detection cells (bottom) consisting of measurements in two consecutive time steps. A detection cell heralds a defect if the two measurements do not agree. (b) If an error (orange) occurs in between two consecutive rounds of measurements, surrounding detection cells light up. (c) Measurement errors (orange) light up the detection cells below and above. (d) General error configurations move anyons through the  $(2 + 1)$ D space-time. In the depicted process, two measurement errors and two errors on physical qubits create a total of four anyonic charges.

$s$  in round  $t - 1$  or in round  $t$ . Thus, a single measurement error occurring when measuring the stabilizer at  $s$  in round  $t$  thus violates two detection cells,  $(s, t - 1)$  and  $(s, t)$  [see Fig. 4(b)]. Such a process can be thought of as a pair creation in the time direction of an anyon associated with the violation of  $s$ . In general, strings are composed of both measurement errors as well as errors on physical data qubits and create pointlike anyons in space-time. As an example, we shown a world line of a color-code anyon in Fig. 4(d).

Thus, a 2D topological error-correcting code where measurements are repeated over time corresponds to a microscopic model realized in  $(2 + 1)$ D space-time. This allows us, on the one hand, to explicitly study topological processes taking place in time using topological stabilizer codes as toy models. On the other hand, we can draw on the wealth of results obtained in the mathematical study of topologically ordered  $(2 + 1)$ D phases of matter and apply them in real-world error-correcting schemes. With this in mind, let us turn our attention to anyon condensation.

At the end of Sec. II B, we have touched on the fault tolerance of the color code. Here, let us make this definition more precise while including errors affecting the measurements of stabilizer terms. As we have seen, errors in  $(2 + 1)$ D space-time can be interpreted as segments of strings with anyons at their end points. Crucially, due to measurement errors, these error strings can also travel along the temporal direction.

Logical errors occur when sufficiently many errors appear along nontrivial paths, connecting distinct boundaries [1,60]. Importantly, these can be spatial boundaries (see Sec. IV A) as well as temporal boundaries (see Sec. V B). This means that we need to include errors affecting measurement outcomes into our definition of the code distance  $d$ . The code distance  $d$  of a code is the lowest number of errors—including measurement errors—that results in a nondetectable and nontrivial logical error. In this work, we call a protocol fault tolerant if  $d$  grows extensively with the system size. In other words, by increasing the system size in the two spatial directions and in the temporal direction, we can reach an arbitrarily low logical error rate, assuming a suitable decoder and a physical error rate below threshold.

### III. ANYON CONDENSATION

In this section, we will walk through the theory of anyon condensation in Abelian-anyon models [28,61], focusing on the color-code phase as a concrete example. In particular, we show that anyons in the condensed phase of an Abelian anyon model can be identified with cosets of a *bosonic subgroup* of the anyon model (see Sec. III A). Based on properties of the condensed subgroup, we sort condensation into three types (see Sec. III B). We specifically describe domain walls in the context of anyon

condensation in Sec. III C. In later sections, we show that the theory of condensation that we develop here underlies the physics of topologically protected operations on the encoded logical information before and after some code transformation. To describe the details of code deformations, in Sec. III D we show how to implement anyon condensation at the microscopic level for topological stabilizer models, where the string operators of the anyons are Pauli operators. We find that in this class of models, condensation can be implemented by measuring low-weight string operators on a code state of what we call the *parent code*.

#### A. Condensation in Abelian-anyon models

Anyon condensation is a mechanism to relate certain anyon models to each other. Given a parent Abelian anyon theory  $\mathcal{C}$ , a  $\mathcal{C}$  *condensate*  $\mathcal{C}_{\mathcal{B}}$  is obtained by identifying a subgroup of bosons  $\mathcal{B} \subset \mathcal{C}$  with the trivial charge. An anyon in the condensed theory  $\mathcal{C}_{\mathcal{B}}$  is related to a coset of anyons  $a \mathcal{B} \subset \mathcal{C}$ .

In particular, we start by choosing a bosonic subgroup, meaning a subgroup of bosons that is closed under fusion and that contains only bosons with trivial mutual braid statistics. Next, we identify this subgroup  $\{b_1, b_2, \dots\} = \mathcal{B} \subset \mathcal{C}$  with the trivial charge,

$$b \equiv 1 \quad \forall b \in \mathcal{B}. \quad (12)$$

Hence, a pair of anyons  $a, b \in \mathcal{C}$  that differ by fusion with some elements of  $\mathcal{B}$  become identified in the condensate. In the remainder of the paper, we use the notation

$$a \simeq b \quad \text{iff} \quad a \mathcal{B} = b \mathcal{B}. \quad (13)$$

Let us look at the different ways in which anyons of the parent theory can be affected by condensation. The anyons of the parent theory  $\mathcal{C}$  fall into two classes:

- (a) *Deconfined*: Any anyon  $a$  that braids trivially with all elements in  $\mathcal{B}$  is *deconfined*:

$$a \equiv a \mathcal{B} \quad \text{iff} \quad M_{a,b} = 1 \quad \forall b \in \mathcal{B}. \quad (14)$$

The anyon  $a$  from the parent theory then defines the anyon  $a \mathcal{B}$  in the condensed theory.

- (b) *Confined*: If  $a \in \mathcal{C}$  braids nontrivially with at least one element in  $\mathcal{B}$ , the topological spin of  $a \mathcal{B}$  becomes ill defined and  $a$  therefore becomes *confined*. The object  $a \mathcal{B}$  is not an anyon in the condensed theory.

Anyons of a condensate  $\mathcal{C}_{\mathcal{B}}$ , defined by condensing a bosonic subgroup  $\mathcal{B} \subset \mathcal{C}$ , are in one-to-one correspondence with a subset of cosets  $\{a \mathcal{B} \mid M_{a,b} = 1 \quad \forall b \in \mathcal{B}\}$

$\mathcal{B}$ ). The modular data of the condensate is given by the parent theory:

$$\theta_{a\mathcal{B}} = \theta_a, \quad M_{a\mathcal{B},c\mathcal{B}} = M_{a,c}. \quad (15)$$

The constraint on  $\mathcal{B}$  being a set of bosons, all of which braid trivially with  $a$  and  $b$ , ensures that the topological numbers of  $a$  and  $b$  are the same. Furthermore, it is apparent that  $\mathcal{B}$  must be closed under fusion.

Throughout this work, we will make use of symmetries in a given parent theory to relate different condensates. A symmetry of anyon model  $\mathcal{C}$  is a permutation of the anyon labels that leaves the anyonic data invariant. The group formed by all these permutations is called the automorphism group of the anyon model,  $\text{Aut}(\mathcal{C})$ . In general, a symmetry in the parent maps a bosonic subgroup  $\mathcal{B}$  to a (potentially different) bosonic subgroup  $\mathcal{B}'$ . The fact that it is a symmetry of the parent indicates that  $\mathcal{C}_{\mathcal{B}}$  is in the same phase as  $\mathcal{C}_{\mathcal{B}'}$ . Symmetries that preserve  $\mathcal{B}$  turn into symmetries in the condensate.

## B. Types of condensation

In this section, we define different types of anyon condensation. We use the color-code anyon model to exemplify different condensation mechanisms.

### 1. Maximal condensation

We maximally condense a parent theory if we condense a *Lagrangian subgroup*  $\mathcal{L}$  [61,62] of bosons. A Lagrangian subgroup is a maximal bosonic subgroup, i.e., there exists no anyon of  $\mathcal{C}$  not included in  $\mathcal{L}$  that braids trivially with all elements in  $\mathcal{L}$ . When a Lagrangian subgroup is condensed, all nontrivial anyons get confined and the condensate is equivalent to the trivial phase.

The color code has six Lagrangian subgroups, each possessing three nontrivial bosons [23]. They can be associated with any one of the three color labels or one of the three Pauli labels:

$$\mathcal{L}_r^{CC} = \{1, rx, ry, rz\}, \quad (16a)$$

$$\mathcal{L}_g^{CC} = \{1, gx, gy, gz\}, \quad (16b)$$

$$\mathcal{L}_b^{CC} = \{1, bx, by, bz\}, \quad (16c)$$

$$\mathcal{L}_x^{CC} = \{1, rx, gx, bx\}, \quad (16d)$$

$$\mathcal{L}_y^{CC} = \{1, ry, gy, by\}, \quad (16e)$$

$$\mathcal{L}_z^{CC} = \{1, rz, gz, bz\}. \quad (16f)$$

The Lagrangian subgroups are expressed using boson tables (see Eq. (2)). Condensed anyons are marked by a black circle  $\bullet$  and confined charges with a  $\times$ . The top three rows show  $\mathcal{L}_r^{CC}$ ,  $\mathcal{L}_g^{CC}$ , and  $\mathcal{L}_b^{CC}$  and the bottom three

rows show  $\mathcal{L}_x^{CC}$ ,  $\mathcal{L}_y^{CC}$ , and  $\mathcal{L}_z^{CC}$ , respectively:

$$\begin{array}{ccc} \bullet \times \times & \times \bullet \times & \times \times \bullet \\ \bullet \times \times & \times \bullet \times & \times \times \bullet \\ \bullet \times \times & \times \bullet \times & \times \times \bullet \\ \bullet \bullet \bullet & \times \times \times & \times \times \times \\ \times \times \times & \bullet \bullet \bullet & \times \times \times \\ \times \times \times & \times \times \times & \bullet \bullet \bullet \end{array}.$$

### 2. Partial condensation

If the bosonic subgroup  $\mathcal{B}$  is not maximal, we have partial condensation. In this case, some of the anyons confine while some remain deconfined, depending on their braiding properties with the bosons in  $\mathcal{B}$ .

In the color code, we can choose any one of the nine nontrivial bosons to be condensed. In all cases, the resulting condensed phase is the toric code. We can depict this using the boson table in Eq. (2). The condensed anyon is marked with a black circle  $\bullet$ . We now have four deconfined color-code charges; those that braid trivially with  $\bullet$ . Two of the deconfined charges are identified with the electric charge anyon  $e$  of the toric code and marked with a  $\circ$ . The other two deconfined color-code anyons get identified with the magnetic flux  $m$  of the toric code and marked with  $\circ$ . The remaining four bosons are confined: we mark this with a  $\times$ . There are in total 18 ways to condense the color code to the toric code. For each of the nine choices of condensed boson, we have an additional binary choice of how to identify the deconfined charges with the  $e$  and  $m$  anyons of the toric code. Some examples are shown here:

$$\begin{array}{ccc} \bullet \circ \circ & \bullet \circ \circ & \circ \bullet \circ \\ \circ \times \times & \circ \times \times & \times \circ \times \\ \circ \times \times & \circ \times \times & \times \circ \times \\ \circ \bullet \circ & & \times \times \circ \\ \times \circ \times & \dots & \times \times \circ \\ \times \circ \times & & \circ \circ \bullet \end{array}.$$

The different ways of obtaining the toric code through condensation in the color code can be related to the color-code symmetries [21,23], which we will discuss in the context of trivial condensation.

### 3. Trivial condensation

*Trivial condensation* is where no nontrivial boson is condensed. The resulting phase after trivial condensation is the same as the initial phase. All types of trivial condensation are in one-to-one correspondence with symmetries of the parent theory, which are given by the automorphism group of the anyon model,  $\text{Aut}(\mathcal{C})$ . These are all the possible ways of relabeling the anyons such that the anyon data are preserved. We will discuss the consistency conditions that such a relabeling has to fulfill in more detail in Sec. III C.

The color-code symmetries can be read directly from the boson table, as discussed in detail in Ref. [23]. Permuting any rows or columns changes the anyon labels but leaves all the anyonic data invariant. Additionally, due to the duality between Pauli and color labels, reflections on the diagonals map the anyon model back to itself. This leads to the automorphism group  $(S_3 \times S_3) \times Z_2$ , which contains 72 elements. An example color-code symmetry is the following:

$$\begin{array}{c|c|c} \text{rx} & \text{gx} & \text{bx} \\ \text{ry} & \text{gy} & \text{by} \\ \text{rz} & \text{gz} & \text{bz} \end{array} \rightarrow \begin{array}{c|c|c} \text{gy} & \text{gx} & \text{gz} \\ \text{by} & \text{bx} & \text{bz} \\ \text{ry} & \text{rx} & \text{rz} \end{array}.$$

### C. Domain walls and anyon condensation

A domain wall is a one-dimensional (1D) subregion along which two phases interface. The types of interfaces that we consider in this work correspond to *gapped domain walls* between two (Abelian) topologically ordered phases,  $\mathcal{C}$  and  $\mathcal{C}'$  [34,63]. Anyon condensation proves to be a useful tool to study this class of domain walls. Furthermore, it is worth pointing out that this description is agnostic toward the space-time direction along which the phases are interfaced. This means that our following prescription will hold for domain walls that cut along any plane of the  $(2+1)\text{D}$  space-time.

We can interface two anyon models,  $\mathcal{C}$  and  $\mathcal{C}'$ , if and only if they share a common condensate  $\mathcal{C}_{\mathcal{B}} \simeq \mathcal{C}'_{\mathcal{B}'}$ . The domain wall can then be interpreted as a thin strip of the condensate between the two phases, as shown in Fig. 5. Anyons in  $\mathcal{B}$  ( $\mathcal{B}'$ ) condense at the domain wall. Anyons that braid nontrivially with any of the condensed anyons get confined to one side of the domain wall, meaning that they cannot move without creating additional excitations. Anyons that braid trivially with all bosons in  $\mathcal{B}$  ( $\mathcal{B}'$ ) are deconfined and can move through the domain wall. For this reason, in this context we call them “mobile.”

We can derive consistency conditions on how the anyonic data on one side of the domain wall relate to the other following Fig. 6. Let  $a$ ,  $b$ , and  $c$  be anyons in  $\mathcal{C}$  that, when moved through a domain wall, get mapped to

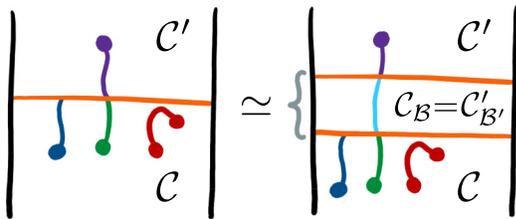


FIG. 5. Two phases  $\mathcal{C}$  and  $\mathcal{C}'$  can be interfaced via a domain wall if and only if they share a common condensate  $\mathcal{C}_{\mathcal{B}} = \mathcal{C}'_{\mathcal{B}'}$ . A domain wall between them can be interpreted as a thin strip of the condensate phase.

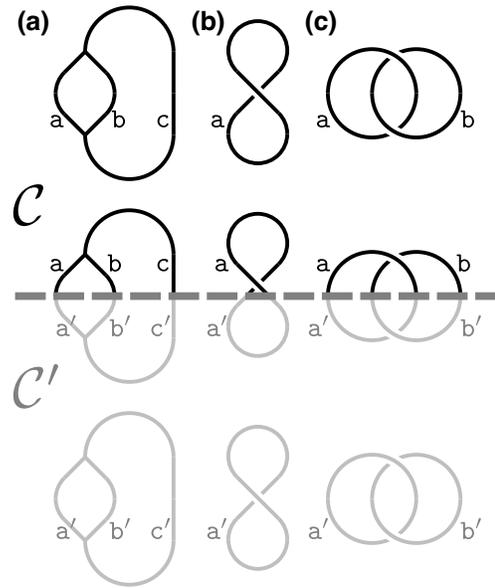


FIG. 6. We consider two topologically ordered phases,  $\mathcal{C}$  and  $\mathcal{C}'$ , and their anyons,  $a, b, c \in \mathcal{C}$  and  $a', b', c' \in \mathcal{C}'$ . A sequence of equivalent world lines of anyons undergoing (a) fusion, (b) self-exchange, and (c) braiding processes on either side of a domain wall that separates  $\mathcal{C}$  and  $\mathcal{C}'$ , is shown by the dashed line. We can deform the world lines smoothly to lie on either side of the domain wall. This allows us to derive consistency conditions relating the data describing the anyons on either side of the domain wall to each other. The second row shows the processes taking place partially on either side of the domain wall.

$a', b', c' \in \mathcal{C}'$ . World lines realizing the three processes of fusion, self-exchange, and braiding can be smoothly deformed to lie on either side of the domain wall. Hence, they must result in consistent outcomes. This implies that the anyonic data on either side of the domain wall must be the same, i.e., we have that

$$\begin{aligned} a \times b = c &\Rightarrow a' \times b' = c' \quad \text{with} \\ \theta_a = \theta_{a'} &\quad \text{and} \quad M_{a,b} = M_{a',b'}. \end{aligned} \quad (17)$$

Furthermore, we can obtain the conditions on anyons that condense at a boundary in this fashion. To this end, we equate  $\mathcal{C}'$  to the trivial phase, which we can think of as only hosting the trivial anyon. This means that all processes happening on this side of the domain wall must be trivial. From this, we obtain closure under fusion and the triviality of self-exchange and braiding of condensable anyons (see Eq. (15)).

Domain walls can be classified by the number of bosons they can condense. For the color code, this means that the domain walls fall into three classes: opaque, semitransparent, and invertible. All types will appear with different applications in Secs. IV–VI.

*Opaque domain walls* are obtained when a full Lagrangian subgroup of bosons can condense from both

sides. The corresponding condensate is the trivial phase and they have no mobile anyons.

We call a domain wall *semitransparent* if certain anyons remain mobile while others condense at the domain wall. Semitransparent domain walls within the same phase can be terminated within the bulk.

*Invertible domain walls* can only be realized within the same phase using trivial condensation. Anyons crossing the domain wall need to preserve their data; hence invertible domain walls are in one-to-one correspondence with the elements of the automorphism group of the anyon model,  $\text{Aut}(\mathcal{C})$ . The end points of invertible domain walls are called *twist defects* and behave similarly to non-Abelian anyons in terms of fusion and braiding [15,23,64–66].

#### D. Anyon condensation in stabilizer models

So far, we have discussed anyon condensation at an abstract level, in terms of anyons. In what follows, we turn our attention to microscopic realizations of topologically ordered phases using stabilizer models. In particular, we describe how to derive a stabilizer model for a condensate from a stabilizer model of a parent phase  $\mathcal{S}_{\mathcal{C}}$ . Microscopically, we achieve this by adding hopping terms of the condensed bosons to the stabilizer, effectively modeling the process of anyon condensation. While we only consider qubit stabilizers explicitly here, the procedure can be generalized to stabilizers on qudits [67], and even for non-Pauli models [32], if one knows the microscopic description of the string operators of the code. Finally, we show how to make the construction explicit by constructing the toric-code stabilizer model (see Sec. II C) from the color-code stabilizer model (see Sec. II A).

To bridge from the abstract notion of anyon condensation to microscopic models, let us reframe stabilizer operators of topological codes in terms of anyons and their string operators. Both the color code and the toric code, as introduced in Sec. II, are topological lattice models the code space of which corresponds to the ground space of a topologically ordered Hamiltonian. In any topological-lattice-model elementary excitations—anyons—are created at the end points of string operators that are supported on 1D subregions [see, e.g., Fig. 1(a)]. This means that any closed string operator does not create any excitation and thereby leaves the ground space invariant. In this reading, stabilizers correspond to contractible loops of the aforementioned string operators. Measuring a stabilizer is equivalent to performing an interferometric charge measurement [43].

We now make the abstract notion of anyon condensation from Sec. III explicit in microscopic lattice models. In a condensate, we require that string operators that transport condensed bosons  $\mathcal{B}$  do not change the state, as we identify them with the trivial charge. We achieve this by adding all open string operators transporting bosons in  $\mathcal{B}$  to the

stabilizer group. This can be achieved by, e.g., adding the set of shortest hopping terms as generators of the stabilizer group.

These string operators violate some of the original stabilizer terms. To recover a commuting stabilizer group that describes the condensate, the terms of the original stabilizer group that do not commute with the new hopping terms are removed. This step corresponds to removing confined charges from the anyon model of the condensed phase.

In the following, we give a recipe to construct the stabilizer group of a condensate from a parent theory  $\mathcal{C}$  and its stabilizer group  $\mathcal{S}_{\mathcal{C}}$  and a bosonic subgroup  $\mathcal{B} \subset \mathcal{C}$ :

- (1) Define the group  $\mathcal{S}_{\mathcal{B}}$  of open string operators for anyons in  $\mathcal{B}$ , generated hopping terms.
- (2) Remove the stabilizers from  $\mathcal{S}_{\mathcal{C}}$  that do not commute with  $\mathcal{S}_{\mathcal{B}}$ . These correspond to the closed string operators of the anyons that braid nontrivially with at least one anyon in  $\mathcal{B}$ . We denote the reduced stabilizer group as  $\tilde{\mathcal{S}}_{\mathcal{C}}$ .
- (3) The stabilizer group of the condensed phase  $\mathcal{C}_{\mathcal{B}}$  is given by  $\tilde{\mathcal{S}}_{\mathcal{C}} \cup \mathcal{S}_{\mathcal{B}}$ .

This construction shows how the toric code is a condensate of the color code. We can view the  $XX$  stabilizers on the red links, characteristic for the microscopic realization of the toric code introduced in Sec. II C, as the hopping terms of the  $rx$  anyon in the color code. Following the procedure laid out above, where  $\mathcal{S}_{\mathcal{C}}$  is the color code and  $\mathcal{S}_{\mathcal{B}}$  is generated by the  $XX$  terms on the red links of the color-code lattice, the stabilizer group  $\tilde{\mathcal{S}}_{\mathcal{C}} \cup \mathcal{S}_{\mathcal{B}}$  corresponds to closed string operators of the anyons that braid trivially with  $rx$ . Explicitly, the color-code anyons get mapped to the toric-code anyons as follows:  $rx \equiv 1$ ,  $ry \simeq rz \equiv e$ ,  $gx \simeq bx \equiv m$  and  $f_1 \simeq f_3 \equiv f$ . Condensation in the entire bulk results in a stabilizer model in the condensed phase.

#### IV. DOMAIN WALLS IN THE COLOR CODE

In the coming sections, we investigate explicit microscopic examples of anyon condensation in the color-code model. In this section, we first consider a condensed color-code phase that is spatially distinct from the color-code phase itself (see Fig. 7). We consider three types of anyon condensation in this setting, where the different types of condensation are discussed in Sec. III B. The anyons are condensed on a subregion of the lattice labeled  $R$ , where, for simplicity, we assume that  $R$  is a disk-shaped region and that the color code is embedded on a manifold with a topology equivalent to that of a sphere. Upon completing the condensation operation, we obtain a domain wall at the boundary of  $R$ , denoted  $\partial R$ . The application of maximal, partial, or trivial condensation transforms  $R$  to lie within

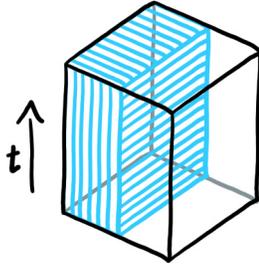


FIG. 7. In this section, we consider the condensation of anyons in given subregion (shaded in cyan). We keep the subregion constant in time in order to create spacelike domain walls between the parent phase and the condensed phase.

the trivial phase, the toric code phase, or the color-code phase, respectively.

In Sec. IV A, we investigate the condensation in region  $R$  to the trivial phase. This enables us to rederive qubit encodings with punctures [12,13,23] in terms of anyon condensation. In Sec. IV B, we consider the condensation of region  $R$  to obtain a toric-code phase. This allows us to introduce new types of domain walls between these distinct topological phases. We also introduce the notion of a semipuncture, which enables us to demonstrate new code deformations between different puncture encodings. To give a clearer perspective on semi punctures, we also reinterpret these objects in terms of the unfolded picture [23,24,26]. For completeness, in Sec. IV C, we consider the condensation of region  $R$  onto the color code itself. This enables us to incorporate known transparent domain walls and twist defects for the color code [21,23] into our theory of anyon condensation.

### A. Boundaries to the vacuum

Here, we describe the boundaries between the color code and the vacuum using the language of anyon condensation. We show how boundaries can be used to encode quantum information in a robust manner as logical qubits. Finally, we discuss the structure of the logical Pauli operators. We give a physical interpretation of them, both as unitary operators acting on the logical state as well as Hermitian operators used to perform measurements.

We obtain the trivial phase from the color-code phase by condensing a complete Lagrangian subgroup of the color-code bosons. Hence, when applying such a condensation to a region  $R$  of the system, we create a domain wall on  $\partial R$  that interfaces the color code with the trivial phase. Such a domain wall is referred to as a boundary [4]. As we have discussed in Sec. III B 1, there are six Lagrangian subgroups in the color-code anyon model, translating to six boundaries that terminate the color code in the spatial direction [23]. These six boundaries fall into two classes: colored boundaries and Pauli boundaries. Colored boundaries are obtained if we choose to condense all three bosons

with the same color label and we obtain Pauli boundaries if the three bosons that are condensed share a Pauli label. The colored boundaries correspond to the columns of the boson table in Eq. (2) and the Pauli boundaries correspond to the rows of the table.

To create a puncture microscopically, we condense anyons by adding hopping terms to the stabilizer group, as described in Sec. III D. To create a puncture with a colored boundary, we consider all the edges of the chosen color that lie within  $R$  and perform Bell-pair measurements on the pairs of physical qubits supported on these edges [13] [see Fig. 8(a)]. To create a puncture with one of the three Pauli boundaries, the prescription from Sec. III D dictates that we should add the two-qubit Pauli rotations

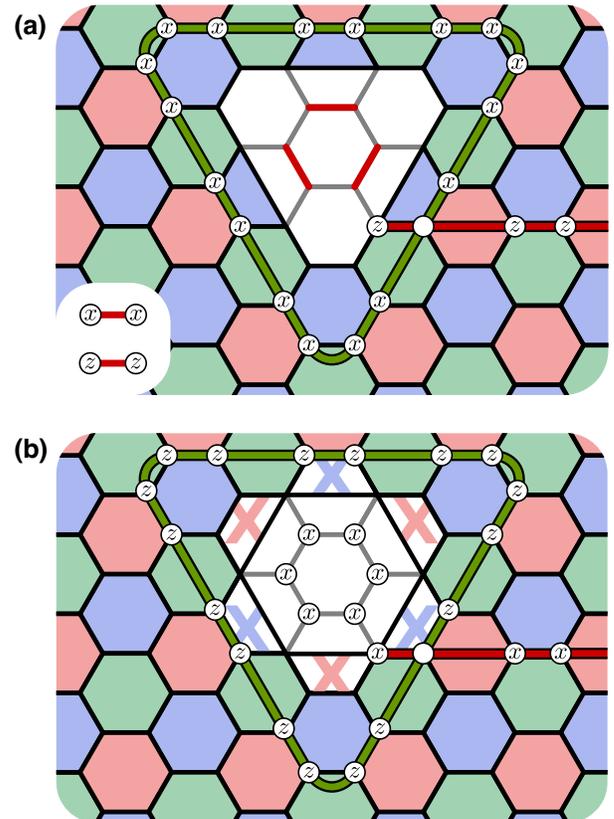


FIG. 8. Punctures with different types of boundaries can be used to encode logical qubits. (a) A puncture with a colored boundary. It is introduced when all three red bosons get condensed within a disk-shaped region  $R$ . Microscopically, condensation is achieved by adding two-body  $XX$  and  $ZZ$  stabilizers on red edges. These edges are highlighted in the center of the puncture. (b) A puncture terminated by a Pauli- $X$  boundary. We produce the puncture by adding single-qubit  $X$  terms supported on qubits in the interior of  $R$  to the stabilizer. Some of the logical operators for these logical qubits are shown, where we assume that the string exiting the figure on the right terminates on a second boundary of appropriate type. Empty circles are used on qubits where the logical operators have common support, where the logical operators act on these qubits in different bases.

in the appropriate Pauli basis on all the edges within region  $R$ .

Alternatively, we can condense all charges with a common Pauli label using single-qubit Pauli measurements in the chosen basis on all qubits within  $R$ . These single-qubit Pauli rotations act as simultaneous hopping terms of all three bosons with the chosen Pauli label. This can be seen in Fig. 1(b), where we apply a single-qubit  $Y$  rotation to decompose an  $ry$  anyon into a  $gy$  and a  $by$  anyon while simultaneously moving the charge in the process. Importantly, the two bosons  $gy$  and  $by$  have a joint charge equivalent to the  $ry$  boson; hence we can regard the single-qubit Pauli- $Y$  rotation as moving an  $ry$  charge. Likewise, we can interpret the same single-qubit rotation as a hopping operator for the green or the blue bosons. The same argument holds for bosons with a Pauli- $X$  or - $Z$  label and single-qubit Pauli- $X$  or - $Z$  rotations. The creation of a Pauli- $X$  puncture using single-qubit  $X$  measurements is shown in Fig. 8(b).

Let us now look at the physics of how a puncture can be used to encode and manipulate logical qubits. To do so, we give a physical interpretation of logical Pauli operators. As the boundary of a puncture can condense four anyons (including the trivial charge  $1$ ), the puncture can be in one of four states, corresponding to the four condensed bosons (including  $1$ ). A red puncture can, e.g., contain one of the following charges:  $\{1, rx, ry, rz\}$ . Similarly, a Pauli- $X$  puncture as an example of a Pauli puncture contains one of these four bosons:  $\{1, rx, gx, bx\}$ . Hence, a single puncture constitutes a four-dimensional Hilbert space that we can use to store quantum information in a robust manner. However, the dimension of the Hilbert space associated with a single puncture only describes the dimension of the logical subspace in the asymptotic limit of a large number of punctures. This is because we require that the charges that describe the internal states of multiple punctures respect global charge conservation [2]. In general then, the ground-state degeneracy scales like  $2^\Upsilon$ , where  $\Upsilon = 2(\#\text{punctures} - C)$ , in which  $C$  is some small correction that depends on the boundary types of the different punctures. We find that for generic configurations of punctures, we have  $C = 2$ . This correction can be lower for special cases where there are punctures with only one or two types of boundary.

In general, we might prefer to encode qubits over small subsets of punctures, as this enables us to perform logical operations on the encoded information. One simple encoding using punctures consists of a pair of punctures with the same type of boundary. This puncture configuration encodes two logical qubits. In Fig. 9(a), we show a pair of red punctures and its logical operators, while in Fig. 9(b) we show a pair of Pauli- $X$  punctures. Alternatively, a triple of colored punctures, one of each color, can also be used to encode logical qubits fault tolerantly. This puncture configuration also encodes two logical qubits, as

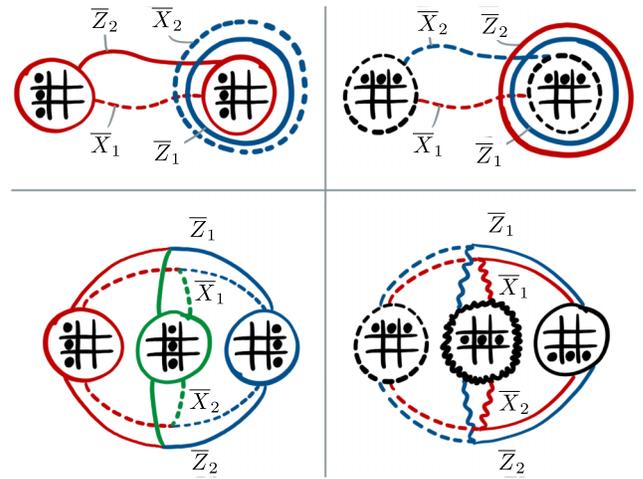


FIG. 9. Four different configurations of punctures are shown. The type of boundary is indicated by the bosons that are condensed inside the puncture, marked on the boson table shown inside each puncture. Each configuration encodes two logical qubits, where the logical operators are depicted in the figure. String operators act on edges of their respective color and the basis is given by the line style: dashed lines correspond to the  $X$  basis, wavy lines to  $Y$ , and solid lines to  $Z$ .

shown in Fig. 9(c). Similarly, three Pauli punctures, again one of each type, encode two logical qubits [see Fig. 9(d)]. Note how the left and the right halves of the figure are related by a duality of the color code between the color and the Pauli labels.

Labeling different logical states by the anyon type that occupies the puncture allows us to identify the logical Pauli operators. Importantly, Pauli operators are both unitary and Hermitian, meaning that we can interpret them as changing a state or being used as a measurement. Viewing the logical operators as unitaries, we require their microscopic realizations to either change the occupation of a puncture or to apply a relative phase depending on the condensed charge within a puncture. We achieve this by applying string operators transporting anyonic charges between punctures or by wrapping an anyonic string operator around a puncture to the encoded state.

Specifically, we can write down an overcomplete set of logical operators as string operators that move a charge from any one puncture to another, provided that the two punctures can condense a common charge that we wish to move. We can convince ourselves that we can find a set of strings that do not cross, such that all of these hopping operators commute. These string operators that hop charges between punctures anticommute with loop-like string operators that wrap around a puncture. Naturally, these looplike string operators correspond to hopping operators for charges that are confined by the puncture that is enclosed by the loop. When viewed as a Pauli-measurement operator, this set of looplike operators can be

physically interpreted as operators that measure the charge content of a given puncture.

The weight of the least-weight logical operator, i.e., the code distance  $d$  in the quantum error-correction literature, is proportional to the circumference of the punctures and their relative separation. A large code distance is obtained by choosing large punctures that are well separated.

So far, we have viewed the logical Pauli operators as unitary operators. Now, let us consider them as Hermitian measurements. The naive way to perform a logical Pauli measurement is to measure the string operators described above. This prescription has two problems for experimental realization, however. First, the number of qubits participating in the parity measurement grows with the circumference or separation of the punctures, leading to a parity measurement on extensively many qubits. A naive implementation of this measurement is not fault tolerant. Second, a single error either on a physical qubit or in the measurement apparatus might change the outcome. In order to obtain a fault-tolerant readout consisting of geometrically local few-qubit measurements, we can follow a procedure similar to the one introduced in Ref. [1]. The language of anyon condensation lends itself nicely to describe fault-tolerant protocols for readout. This is the topic of Sec. VB.

### B. Domain walls to partial condensates

Condensation of a single boson in a region  $R$  transforms the color code into the toric code. The boundary  $\partial R$  constitutes a domain wall between the two phases. As discussed in Sec. III B 2, there are 18 possible ways to interface the two phases. They differ in which anyons are condensed, confined, or remain mobile when approaching the domain wall and to which anyons of the condensed phase the mobile anyons are mapped.

In Fig. 10, we show an example where the  $rx$  boson gets condensed in the top half of the lattice. The four bosons  $gy$ ,  $gz$ ,  $by$ , and  $bz$  that all braid nontrivially with  $rx$  become confined. The two remaining red bosons,  $ry$  and  $rz$ , as well as the two remaining bosons with a Pauli- $X$  label,  $gx$  and  $bx$ , braid trivially with  $rx$ . Hence, they remain mobile and can pass through the domain wall. Upon crossing the domain wall, the mobile charges are mapped to one of the two toric-code bosons. In the example shown, we map  $ry$  and  $rz$  to  $e$  and  $gx$  and  $bx$  to  $m$ .

Microscopically, we introduce this domain wall by following the prescription given in Sec. III D to condense a single boson. Concretely, we designate a boson with a specific color and Pauli label to be condensed. In the example shown in Fig. 10, we choose to condense  $rx$ . To condense the boson, we add two-body hopping terms to the stabilizer. Their support and the basis in which they act are given by the labels of the boson. To condense  $rx$ , we add  $XX$  terms on all red edges in  $R$  to the stabilizer

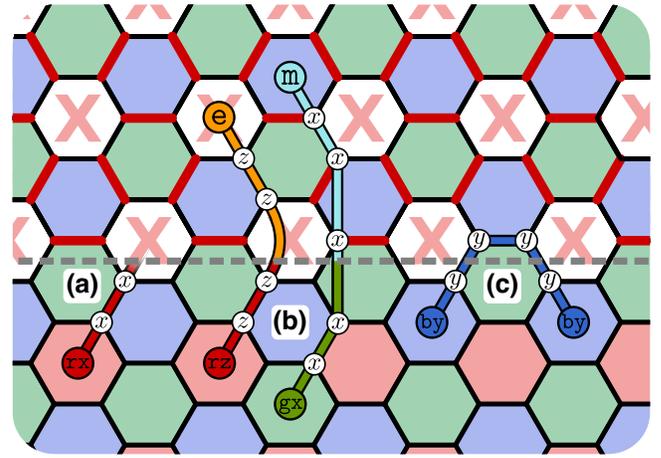


FIG. 10. A domain wall (gray dashed line) between the toric code (top) and the color code (bottom). Fully colored-in plaquettes host stabilizers acting in both  $S_X^p$  and  $S_Z^p$  stabilizers. Red plaquettes marked with an  $X$  host only the  $X$ -basis stabilizers;  $S_X^p$ . The red edges in the top half of the figure host  $XX$  stabilizers. The color-code anyons either (a) condense or (c) confine at the domain wall. Otherwise, they (b) remain deconfined as they pass through the domain wall.

group. Finally, we update the stabilizer by removing terms that do not commute with the introduced hopping terms. In this example, we remove the  $Z$ -basis stabilizers on red plaquettes.

Let us now examine the properties of the partially condensed region from the perspective of unfolding. By choosing a suitable unfolding, where we separate the color code into two disjoint copies of the toric code, we can identify the feature obtained by partial condensation in  $R$  as a puncture on one of the two copies of the toric-code. Hence, we dub such a feature a “semipuncture.”

With this observation, we discover the value of viewing the color-code phase from the perspective of the boson table. Let us stress that we obtain a puncture on one copy of the toric code only assuming that we choose a suitable unfolding map. As we have discussed in Sec. II D, there are 72 different choices of unfolding map onto copies of the toric code. However, under the same unfolding map, there are certain semipunctures that are obtained by condensing other choices of boson, which do not immediately divide into a puncture on a single copy of the toric-code. Rather, we require the use of additional domain walls to describe all of the semipunctures for any fixed unfolding map.

We therefore find the unfolded picture to be somewhat unsatisfying, because different color-code semipunctures manifest themselves differently in the toric-code picture. On the other hand, in the color-code picture, all of the bosons are equivalent, up to symmetries among relabeling of their color and Pauli labels. We then argue that the color code and its corresponding boson table offer a clearer way to describe these generalized topological features that

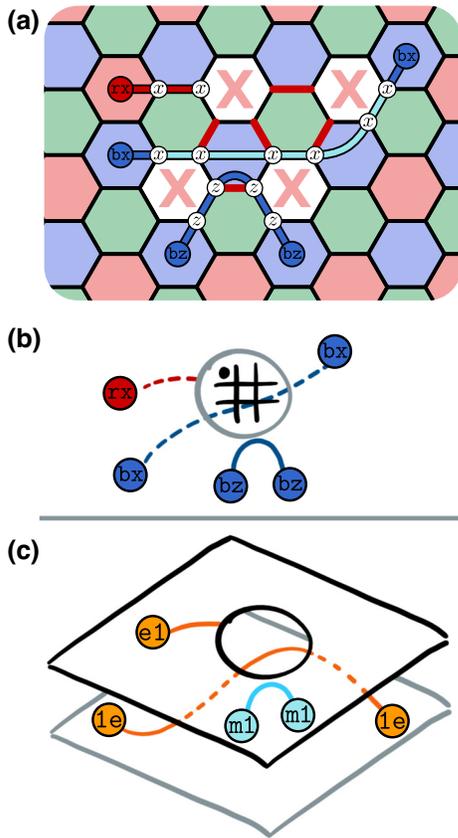


FIG. 11. An  $rx$  semipuncture shown in both the color-code picture and the unfolded toric-code picture for a suitable choice of unfolding. (a) The microscopic model, together with different types of creation operators in the vicinity of the semipuncture. Specifically, the  $rx$  anyon condenses at the semipuncture, the  $bx$  anyon deconfines and remains mobile, and  $bz$  is confined, i.e., cannot enter the semipuncture. (b) A macroscopic representation of the same semipuncture in the color-code picture. We explicitly label the semipuncture with a boson table that marks the condensed boson in the center of the semipuncture. (c) The semipuncture in an unfolded picture. We choose the unfolding map shown in Eq. (9) to reveal the semipuncture picture.

we have introduced here, as the boson table symmetrizes the classification of all of the different semipunctures we can produce. We depict these contrasting descriptions in Fig. 11.

Naturally, like punctures, we can use semipunctures to encode logical information. We show examples of logical encodings using semipunctures in Fig. 12. We can describe the physics of the associated logical operators equivalently to that of the logical operators encountered in Sec. IV A. They are string operators that connect semipunctures or string operators that wrap around the semipuncture. The discovery of semipunctures also opens the door for the design of new types of code deformations. As an example, we show that we can transform between different configurations of punctures using semipunctures to mediate the transition. As an example, in Fig. 13 we show that we

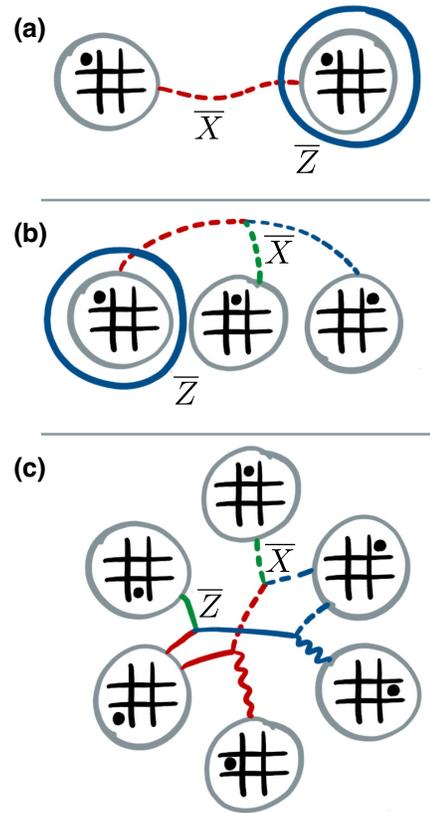


FIG. 12. Encoding logical qubits using semipunctures. We indicate the type of semipuncture by marking the condensed boson on the shown boson table. The logical string operators are presented, where the colors of the lines indicate on which color of edges they act. The dashed, wavy, and solid lines correspond to operators acting in the  $X$ ,  $Y$ , and  $Z$  basis, respectively. (a) and (b) encode one logical qubit each, whereas (c) encodes two logical qubits. The second set of logical operators in (c) are supported to the exterior of the semipuncture configuration we have presented.

can transform between the logical-qubit encoding shown in Fig. 9(b) onto the encoding in Fig. 9(c), where we make use of the six semipunctures at an intermediate step. We add that we have already encountered the six-semipuncture encoding in Fig. 12(c). In addition to this example, we find that semipunctures also emerge when performing a read-out addressing only some of the logical qubits encoded in a color code. We discuss this example in Sec. V F.

The examples that we present show us that the discovery of semipunctures may be helpful to find new fault-tolerant logical operations. These may be helpful to, e.g., reduce the resource overhead of color-code-based quantum computation. Furthermore, considering that they can be combined with corners to obtain mixed boundary semipunctures [68,69], we have presented a significant landscape to design and explore code-deformation protocols in the future.

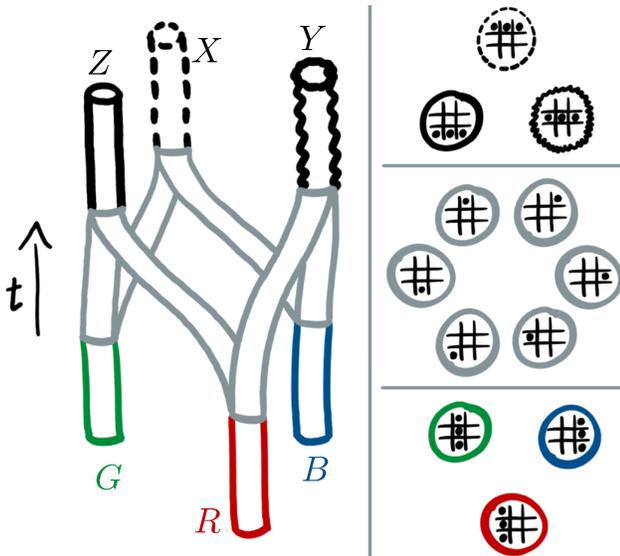


FIG. 13. A code-deformation protocol that is mediated by semipunctures. We find that we can switch between a puncture encoding where each of the three punctures has a differently colored boundary onto an encoding where three punctures all have distinct types of Pauli boundary. We achieve this by splitting each of the colored punctures into two semipunctures. Next, the semipunctures are moved and fused to form three Pauli-type punctures. The left-hand part of the figure shows the space-time diagram of the protocol, while the right-hand part shows the (semi)puncture configuration at intermediate time steps.

### C. Invertible domain walls

When condensing a set of anyons, we implicitly or explicitly make use of the symmetries of the underlying topological phases. This is also true in the trivial case, where no nontrivial anyons are condensed. This leaves us interfacing the parent phase with itself while applying an anyon-permuting symmetry. These symmetries are the automorphisms of the anyon model  $\text{Aut}(\mathcal{C})$ . In the case of the color code, there are 72 such automorphisms, as we have briefly summarized in Sec. III B 1. To each automorphism, we can associate one domain wall (for a detailed discussion on automorphisms of the color code and the associated spatial domain walls, see Ref. [23]).

Anyons are distinguished by their fusion and braiding properties. Consider applying a symmetry given by the automorphism  $\text{Aut}(\mathcal{C})$ , i.e., trivial condensation, in a simply connected closed region  $R$ . This creates an invertible domain wall along the boundary  $\partial R$ . We say that the domain wall acts as  $\text{Aut}(\mathcal{C})$  on the anyon crossing it.

However, note that there is no way of detecting the domain wall using only operations based on moving the color-code anyons, i.e., fusion and braiding. This is because, by definition, the relative behavior of the anyons is independent of the presence of the domain wall (see Fig. 6).

Domain walls can, however, be terminated. In doing so, we create so-called “twist defects” at the end points. This is the topic of Sec. VI A. Alternatively, we can consider applying an automorphism on an entire code patch, hence applying a transversal logical operation on the encoded qubits. This is the topic of Sec. V A.

## V. TEMPORAL DOMAIN WALLS

In this section, we discuss temporal domain walls. They are commonplace in topological quantum computation, as they describe initialization or injection of a code state, the application of a locality-preserving gate, or the readout step at the end of a computation. Temporal domain walls are introduced by changing the stabilizer group over time (see Fig. 14). In particular, here we argue that anyon condensation is a well-suited framework, not only to describe temporal domain walls but also to construct them microscopically as topologically protected deformations of the stabilizer group. In this sense, our treatment of temporal domain walls in this section is exactly analogous to that of spatial domain walls discussed in Sec. IV.

On a high level, quantum information is processed by changing logical operators over time. Since logical Pauli operators in 2D topological error-correcting codes are anyonic string operators, keeping track of how anyons get mapped when passing through a domain wall gives us a physical interpretation for the action of different topological operations on logical qubits.

As we show in Sec. V A, we can relate locality-preserving gates with invertible domain walls. This enables us to include transversal gates [5,70] in our theory of anyon condensation. We also find that initialization and readout can be interpreted in terms of anyon condensation. In Sec. V B, we go through the details of fault-tolerant state readout [12,13] in terms of the anyon-condensation picture that we have introduced. We develop this discussion further in Sec. V C, where we describe at the microscopic level details of a readout operation in terms of error-correction in space-time. In Sec. V D, we elaborate on the

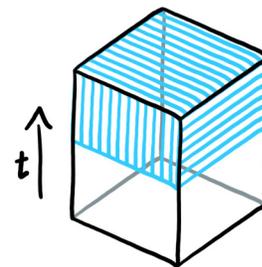


FIG. 14. In this section, we create temporal domain walls between the parent phase and the condensed phase. We achieve this by condensing anyons in the whole spatial bulk after (or before) a given point in time. The region of the space-time diagram in which the condensation takes place is shaded in cyan.

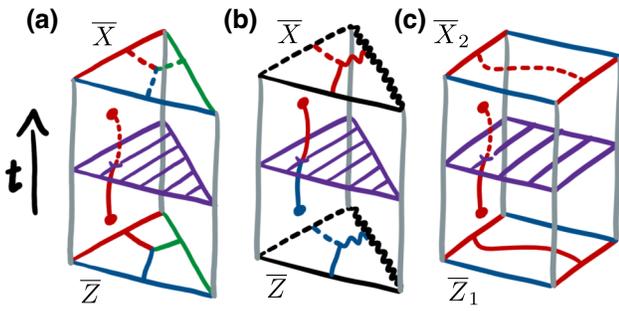


FIG. 15. Transversal gates in the triangular color code correspond to invertible domain walls in the time direction. The action of the domain walls (purple) is depicted by the effect of an anyon crossing it. Dashed (solid) world lines crossing the domain wall correspond to anyons with a Pauli- $X$  ( $-Z$ ) label. (a) In the triangular color code with colored boundaries, a domain wall that turns the Pauli- $Z$  label into the Pauli- $X$  label changes the logical  $\bar{Z}$  operator into  $\bar{X}$ . (b) For the triangular color code with Pauli boundaries, the domain wall needs to permute the color labels of the anyons in order to act as a logical Clifford gate. (c) In a rectangular color code where two opposite boundaries are red and two opposite boundaries are blue, a transversal Hadamard implements the logical gate  $\bar{H}_1 \circ \bar{H}_2 \circ \bar{\text{SWAP}}_{1,2}$ . The dashed, wavy, and solid lines correspond to the Pauli- $X$ ,  $-Y$ , and  $-Z$  basis, respectively.

theory of temporal boundaries further by showing how temporal boundaries interact with spatial boundaries during certain known color-code state-preparation procedures [12,13,19].

In Sec. VE, we describe new stability experiments for the color code that exemplify nicely the interplay between different types of spatial and temporal boundaries. This builds on recent work introducing the notion of a stability experiment [33]. Finally, in Sec. VF, we introduce partial initialization and partial readout for the color code. This gives us new readout protocols based on semitransparent temporal domain walls. These operations enable us to address specific logical qubits encoded on some region of the color-code lattice, without interacting with other qubits encoded on the same region.

### A. Invertible domain walls and Clifford gates

A temporal invertible domain wall corresponds to a symmetry of the anyon model,  $\text{Aut}(\mathcal{C})$ . This means that all anyons can traverse the domain wall and in doing so get mapped to potentially different anyons. As the logical Pauli operators are associated with anyon strings, they get permuted. Thus, the introduction of an invertible temporal domain wall acts as a logical Clifford gate. We show examples in Fig. 15.

In the triangular color code, all single-qubit Clifford gates are transversal [5]. This leads to an apparent mismatch between the number of different invertible color-code domain walls, 72 [21,23], and the size of the Clifford

group acting on one qubit, six if we ignore phases. We assume the applied symmetry to not change the boundaries of the code in order to preserve the code space. In the example of the triangular color code with colored boundaries, a symmetry that permutes any of the color labels would also change the boundary type of some of the boundaries. Similarly, any symmetry that applies the duality transformation exchanging the Pauli and the color labels of the anyonic charges of the color code transforms colored boundaries into Pauli boundaries. Thus, we exclude them here. This leaves us with the  $|\mathcal{S}_3| = 6$  symmetries, which solely permute the Pauli labels. These are exactly the elements of the Clifford group without phases. Applying any of these gates transversally on all physical qubits applies the equivalent logical gate [5].

Interestingly, this argument holds true for any boundary configuration that contains only colored boundaries—or, equivalently, any configuration only containing Pauli boundaries. Meaning that any such code contains exactly six gates that can be applied transversally as an invertible temporal domain wall. In the case of the square color code with RBRB boundaries [see Figs. 15(c) and 24], a generating set for the transversal gates that can be realized is as follows. A Hadamard gate  $H$  applied transversally to all of the physical qubits exchanges the logical operators  $\bar{Z}_1 \leftrightarrow \bar{X}_2$  and  $\bar{X}_1 \leftrightarrow \bar{Z}_2$ , corresponding to a logical swap gate followed by a Hadamard gate on each logical qubit,  $\bar{H}_1 \circ \bar{H}_2 \circ \bar{\text{SWAP}}_{1,2}$ . This transversal gate is shown in Fig. 15(c). The gate exchanging the Pauli- $Y$  and  $-Z$  basis when applied transversally to all physical qubits is, up to phases, equivalent to the application of  $\bar{H}_1 \circ \bar{\text{CNOT}}_{1,2} \circ \bar{H}_1$ . The other three nontrivial single-qubit Pauli-permuting gates can be generated from the two given examples. On the logical level, they can all be composed of controlled-NOT (CNOT) gates and Hadamard gates.

Similarly, through the duality between the color labels and the Pauli labels, we can argue that any color code that is terminated exclusively by Pauli boundaries also has six transversal Clifford gates. It may be interesting to relax the assumption that the boundaries before and after the transversal gates have to match. This might also combine in a nontrivial way with the code deformations that we have introduced in Sec. IV B, which smoothly transform between different boundary types.

Lastly, we point out that invertible temporal domain walls have applications beyond the implementation of logical gates. Certain Floquet codes [36–38,71–73] can be interpreted as so-called automorphism codes, where invertible domain walls on different subregions are periodically introduced [74]. In particular, in this reading of the honeycomb code [36], every time step introduces a domain wall around  $\frac{1}{3}$  of the plaquettes, such that after three steps an automorphism has been applied to the whole code. In this work, we argue that anyon condensation is a well-suited tool to study and construct Floquet codes (see Sec. VII).

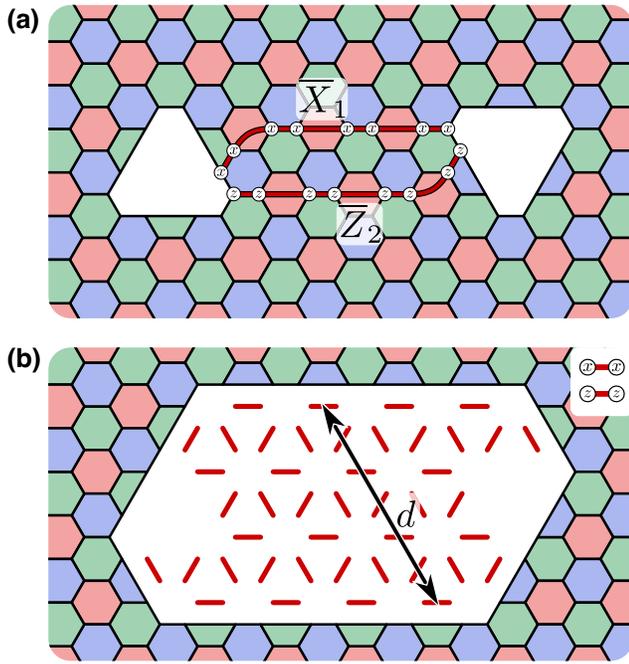


FIG. 16. Reading out logical qubits by condensing anyons. Two red punctures, as shown in (a), encode two logical qubits. A red string connecting the two punctures constitutes a logical operator. To read out the logical operators  $\bar{X}_1$  and  $\bar{Z}_2$ , we can measure the  $XX$  and  $ZZ$  parity on all red links in a region  $R$  encompassing the two punctures, as depicted in (b). This process condenses all red anyons and creates one large red puncture in  $R$ . The inverse process initializes the two logical qubits in an eigenstate of  $\bar{X}_1$  and  $\bar{Z}_2$ .

### B. Initialization and readout

In this section, we study the readout, and implicitly the initialization, of logical qubits encoded in color codes. These processes are naturally described as maximal anyon condensation. We discuss the relationship between the logical operators that we address in the process and the Lagrangian subgroups that are condensed. The condensation of anyons in time introduces temporal domain walls. We defer a detailed discussion of temporal boundaries to Sec. VC.

We begin by revisiting the encoding scheme encountered in Sec. IVA, where two red punctures are encoding two logical qubits. Specifically, we show how to simultaneously read out logical operators  $\bar{X}_1$  and  $\bar{Z}_2$  [see Fig. 16(a)]. Both of these logical operators,  $\bar{X}_1$  and  $\bar{Z}_2$ , can be interpreted as string operators that transport red anyons between the punctures. Hence, they are composed of two-body hopping terms on red edges [see Fig. 16(a)].

To read out the logical operators, we begin by defining a region  $R$  on which  $\bar{X}_1$  and  $\bar{Z}_2$  can be fully supported. Then, we measure all red hopping terms in  $R$ , i.e., the  $XX$  and  $ZZ$  parities on all red edges. These measurements can be combined to infer the values of  $\bar{X}_1$  and  $\bar{Z}_2$ . Choosing a suitable

region  $R$  allows us to correct for errors on the value of the logicals (for details, see Sec. VD). Likewise, this readout scheme is used to perform measurement-based logical gates in lattice-surgery protocols presented in Ref. [20] and in Sec. VID of this work.

The above example can be understood in terms of anyon condensation. In particular, we condense all red anyons  $\mathcal{L}_R$  when measuring the red hopping terms. In fact, it is straightforward to generalize the concept of condensing anyons to read out logical qubits to topological stabilizer codes. The logical Pauli operators  $\{\bar{L}_i\}$  that can be read out simultaneously have to be composed of hopping terms of bosons contained in the same Lagrangian subgroup  $\mathcal{L}$ . This guarantees that we read out a commuting set of logical operators, as their corresponding anyons braid trivially, by the definition of a Lagrangian subgroup.

Let us look at how we read out logical Pauli operators at the physical level. To this end, we consider two different stabilizer groups,  $\mathcal{S}_C$ , being the stabilizer group of a topological stabilizer code  $C$ , and  $\mathcal{S}_M$ , which is generated by the microscopic hopping terms of the anyons in  $\mathcal{L}$  (see Sec. IIID). Note how all logical operators in question are contained in this stabilizer,  $\bar{L}_i \in \mathcal{S}_M$ . Thus, condensing the Lagrangian subgroup  $\mathcal{L}$  by measuring the generators of  $\mathcal{S}_M$  measures the eigenvalues of  $\bar{L}_i$ . Due to the construction in terms of anyon condensation, we guarantee the geometric locality and thus the bounded weight of the operators that we measure.

To initialize logical qubits in a certain eigenstate of their logical Pauli operators, we follow an equivalent procedure, reversing the time direction. Concretely, this means that we change from an initialization stabilizer group  $\mathcal{S}_I$  to the stabilizer group  $\mathcal{S}_C$  of the code. We construct  $\mathcal{S}_I$  to contain the hopping terms composing  $\bar{L}_i$ , thus initializing the system in eigenstates of  $\bar{L}_i$ . Note that the sign that the logical Pauli  $\bar{L}_i$  carries depends on the choice of signs for the generators of  $\mathcal{S}_I$ .

The described initialization and readout protocols condense a full Lagrangian subgroup of anyons. This introduces boundaries between the code and the vacuum, which lie perpendicular to the time direction. In the color-code readout example from Fig. 16, for instance, we introduce a red temporal boundary between the color code and the vacuum phase. We can depict this using the space-time picture (see Fig. 17). The microscopic details of temporal color-code boundaries are discussed in Sec. VC.

### C. Microscopics of temporal boundaries

It is helpful to view fault-tolerant logical operations in the complementary three-dimensional (3D) space-time picture. In this picture, we identify spatial domain walls as lying perpendicular to a spatial direction. On the other hand, we identify domain walls that lie perpendicular to the time direction with temporal domain walls. In fact, we find that the space-time picture reveals a duality between the

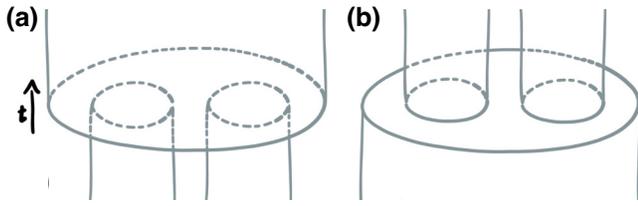


FIG. 17. The space-time diagrams corresponding to the readout process depicted in Fig. 16. (a) During readout, the two punctures turn into one large puncture. (b) Initialization is achieved by flipping the time direction, i.e., by starting with one large red puncture that then gets deformed into two smaller red punctures. In both cases, the horizontal boundaries are temporal boundaries.

spatial domain walls that we have introduced in Sec. IV A and the temporal domain walls. Here, we will describe this duality in the microscopic picture, focusing on boundaries.

We conduct our investigation into the behavior of different types of charges, identified by different types of detection cells, by focusing on two examples. In one example, we discuss a temporal boundary where we read out logical qubits using single-qubit Pauli- $X$  measurements and a second example where we read out the color code with Bell measurements on the green edges. In short, we study a temporal Pauli boundary as well as a temporal color boundary. We remark, though, that focusing on these two cases is without loss of generality, due to the Pauli-label and color-label exchange symmetries of the color code. Likewise, while we concentrate on the readout step by investigating the detection cells obtained by deforming  $\mathcal{S}_{CC}$  onto  $\mathcal{S}_M$ , we can reproduce the same discussion at the initialization step by, instead, deforming the stabilizer group  $\mathcal{S}_I$  onto  $\mathcal{S}_{CC}$ . This case differs only in the sense that the direction of time is reversed, as discussed in Sec. V B.

The characteristic features of spatial boundaries are their ability to condense a Lagrangian subgroup of bosonic excitations of the underlying anyon model. In the space-time picture, we replace the notion of stabilizers for identifying pointlike charges with that of a detection cell (see Sec. II E). Moreover, for each distinct anyon that we can measure in the 2D picture, we have a corresponding detection cell in the space-time picture. We therefore find a one-to-one correspondence between our description for bosonic charges in the space-time picture with the more conventional 2D idealization of the color code. Adopting this perspective, we find that we can view the charges in space-time to either condense or confine at a temporal boundary between the color code and the vacuum phase. Moreover, we find that we can obtain a temporal boundary that corresponds to any of the six Lagrangian subgroups of bosonic charges in the color-code model. Following the prescription given in Sec. III D, we realize the found temporal boundaries explicitly in microscopic lattice models.

Let us begin our discussion by considering the temporal boundary at which we read out the color code with Pauli- $X$  measurements. We will identify the condensed charges before looking at confined charges. A signature of a boundary being able to condense a given type of charge,  $a$ , is that an individual charge of that type can be created locally at the boundary, seemingly violating the fusion rules in the bulk of the system. Indeed, the fusion rules of an anyon model are modified close to a boundary in general. As such, identification of a configuration of errors that give rise to a single charge close to a boundary is indicative of the fact that a boundary condenses (the antiparticle of) that given charge type.

In Fig. 18(a), we show an error configuration in which a single  $rx$  charge is created at the temporal boundary where we measure all qubits in the Pauli- $X$  basis. Specifically, we show a measurement error on the final reading of the six-body Pauli- $Z$  stabilizer. Its only corresponding detection cell indicates the detection of a  $rx$  boson, where we recall the convention that we have adopted in which Pauli- $Z$  stabilizers detect bosons with Pauli- $X$  labels. Indeed, this is the final detection cell of this type that we measure in this readout process, as we cannot infer the values of Pauli- $Z$  stabilizers from the choice of readout operation. Identification of a single  $rx$  charge signifies that  $rx$  charges are condensed at this boundary. We show the charge configuration at a spatial boundary in Fig. 18(b) to emphasize the analogy between this temporal boundary and a spatial boundary. In both cases, we can find similar error configurations in which detector cells identify individual charges of any color, provided that they have a Pauli- $X$  label. For a temporal  $X$  boundary, they are introduced by measurement errors to Pauli- $Z$  stabilizers of the appropriate color. For a spatial boundary, they are created by an  $XX$  error supported on an edge of the appropriate color.

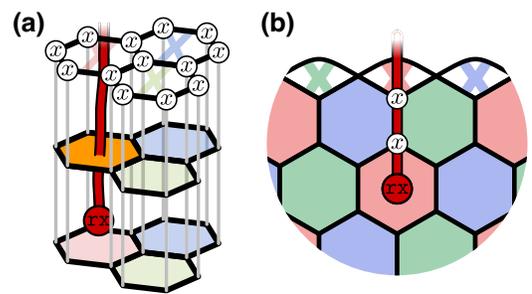


FIG. 18. Condensed charges at Pauli- $X$  type boundaries in the temporal and spatial direction. (a) The final reading of the stabilizers of the color code before we make a transversal Pauli- $X$  measurement on all the physical qubits, thereby condensing all charges with a Pauli- $X$  label. A measurement error (marked in orange) on the final reading of a Pauli- $Z$  stabilizer before the code is read out gives rise to a detector cell that identifies a single  $rx$  charge in the space-time picture. (b) An analogous charge configuration at a spatial Pauli- $X$  boundary. We show an error configuration that gives rise to a single  $rx$  charge.

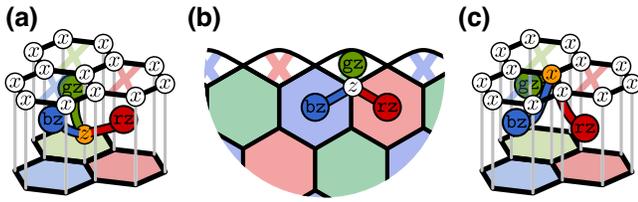


FIG. 19. Configurations of confined charges at a Pauli- $X$  boundary. (a) A physical Pauli- $Z$  error close to the temporal domain wall created by measuring all physical qubits in the Pauli- $X$  basis creates a configuration of confined charges. (b) We compare the confined charges of the temporal Pauli- $X$  boundary to an analogous configuration of charges created at a spatial Pauli- $X$  boundary of the color code. We show the error and the charge configuration on the 2D lattice. (c) A single measurement error, marked by the orange qubit, that occurs on one of the physical-qubit measurements at the readout step creates a configuration of confined charges equivalent to that shown in (a).

While certain charges are condensed at a temporal boundary, we find that other types of charges are confined. Let us posit that since charges with Pauli- $X$  labels are condensed at Pauli- $X$  boundaries, all charges with other labels must be confined. In Fig. 19(a), we show a single Pauli- $Z$  error that gives rise to a configuration of color-code charges that is allowed by the bulk-fusion rules that are confined at a Pauli- $X$  boundary. We contrast this configuration in the space-time picture with an analogous configuration of defects at a spatial boundary in the 2D picture in Fig. 19(b). In the space-time picture of Fig. 19(a), these detection cells are completed by taking the last six-body reading of the Pauli- $X$ -type stabilizers and comparing them with the values of the same stabilizers that are inferred from the single-qubit measurements made at the readout step. Indeed, we can find no error configuration that locally annihilates any one of the individual charges of this configuration such that the fusion rules of this error configuration are violated. We find the same configuration of charges if we have a measurement error at the final readout step [see Fig. 19(c)].

We can attribute the confinement of charges during this condensation procedure to a fault-tolerant readout step. Measurement errors at this final readout step also give rise to confined subsets of charges. In Fig. 19(c), we show a measurement error that occurs on one of the physical qubits during the readout step. We find that this configuration of charges identified by the detection cells is identical to that of Fig. 19(b), i.e., a physical error that produces a configuration of charges that is consistent with the fusion rules of the bulk phase. We can compare the detection of confined charges to the detection of condensed charges at this temporal boundary, to find that it is not *a priori* obvious that we should expect to detect charges that respect the color-code fusion rules close to a temporal boundary. The confining effect during the readout enables us

to employ standard decoding methods to identify errors during readout.

With this example, let us finally note that the distinction between the confined charges with a Pauli- $Y$  label and those with a Pauli- $Z$  label is not well defined at the Pauli- $X$  boundary. At the microscopic level, this is due to the fact that at the final time step where we infer the values of Pauli- $X$  stabilizer detection cells at readout, we do not make a Pauli- $Z$  detection-cell measurement. Macroscopically too, this is a generic feature of a boundary that condenses charges with a Pauli- $X$  label. This is due to the fact that we can locally create individual charges with a Pauli- $X$  label of any color and arbitrarily fuse them with the confined charges that are in the vicinity of this boundary. As such, charges with a Pauli- $Y$  label and a Pauli- $Z$  label are indistinguishable when they are close to a Pauli- $X$  boundary. This is another example of how anyon condensation leads to identification of distinct confined anyons, as described in Sec. III A.

Let us next look at the temporal boundary created by making Bell measurements on the lattice edges for some choice of color. Without loss of generality, we choose to make Bell measurements on the green edges. Like the temporal domain we have already discussed that gave rise to a Pauli- $X$ -type boundary, here we find that the temporal domain wall that we produce is analogous to that of a green color-code boundary.

Again, we begin by looking at the condensed charges in this example. In Fig. 20(a), we show a measurement error during a reading of a six-body Pauli- $X$  stabilizer. Given that we cannot infer the values of the green stabilizers from the Bell measurements that we measure during readout, this is the final reading of this specific detection cell. This means that the detection cells give rise to a syndrome configuration showing a single  $gz$  charge. We show the charge configuration next to a green boundary of

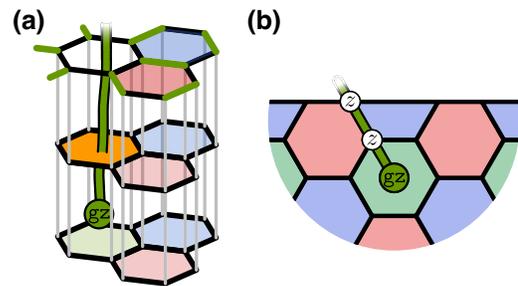


FIG. 20. A condensed charge at green boundaries. (a) A single measurement error on the final reading of a green six-body stabilizer immediately before the readout step is identified by a single detection cell only. This is consistent with the behavior of a temporal boundary that condenses green charges in the space-time picture. (b) A configuration of physical errors that creates a single green charge at a spatial boundary of the color code, as shown in two dimensions.

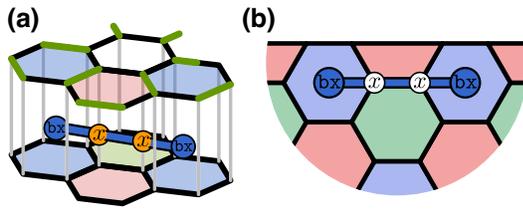


FIG. 21. Confined charges at green boundaries. (a) Errors supported on a blue edge create charges at two blue detection cells. This is an example of a configuration for confined charges at a green boundary. (b) An analogous configuration of charges to that of (a), shown at a green spatial boundary in two dimensions.

the color code in two dimensions in Fig. 20(b). One can check that we could have equivalently made a single  $gx$  or  $gy$  charge with different configurations of measurement errors at the final reading of the green stabilizer. We therefore observe physics that is consistent with that of a green boundary at the temporal domain wall that we have created in the space-time picture, where we have performed a measurement that condenses the green charges to read out the logical qubit.

Lastly, let us look at the confined charges at the green temporal boundary. In Fig. 21(a), we show a two-qubit error supported on a blue edge. The figure shows detection cells identifying a pair of  $bx$  charges. We compare the charge configuration found at the temporal boundary to the more familiar green spatial boundary shown in two dimensions in Fig. 21(b).

We can also look at charges at the temporal boundary created by making Bell measurements at the green edges. In Fig. 22(a), we show a single measurement error that occurs during the Bell-measurement readout step. We note that one can find a physical error on a single qubit that occurs just before the readout, which is equivalent to this

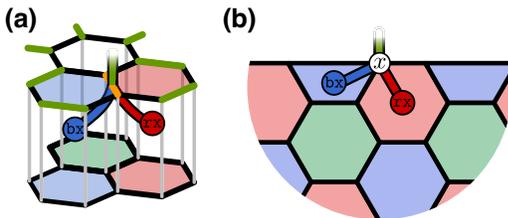


FIG. 22. Charges at green boundaries. (a) A measurement error on a two-body measurement that occurs during readout creates charges at two detection cells, a blue cell and a red cell. This is an allowed charge configuration at a green boundary, where individual green charges can be created locally. (b) An analogous configuration of charges to that of (a) shown at a green spatial boundary in two dimensions. We note that the net charge in both of the displayed charge configurations is that of a green anyon. This is consistent with the behavior of a boundary that condenses green charges where individual green charges can be created locally.

measurement error in the sense that the two local errors give rise to an equivalent charge configuration. Here, we focus on a measurement error. The figure shows detection cells identifying a pair of charges, an  $rx$  charge and a  $bx$  charge. While this charge configuration is inconsistent with the fusion rules that are allowed in the bulk of the color code, this configuration is in fact allowed at a boundary that condenses green charges. Indeed, as we can create green charges locally at a green boundary, the coloring of the confined red- and blue-labeled charges with the same Pauli label becomes ill defined. As such, the creation of any even parity of charges that take any color other than green is allowed near to a green boundary. We have observed similar physics at the Pauli- $X$  boundary discussed earlier in this section, except that whereas in the previous example the Pauli labels have become ill defined, here, in this example, the color labels become ambiguous. We compare the charge configuration found at the temporal boundary to the more familiar green spatial boundary shown in two dimensions in Fig. 21(b).

In fact, the last example of a blue and a red charge close to a green boundary has an unusual quirk that we finally point at here. Specifically, we can take two different perspectives on this charge configuration. At a microscopic level, we can view it as a pair of confined charges. Alternatively, from a more macroscopic perspective, we can view the charge as a single green charge. The fusion rules of the color code are such that the union of a red and a blue charge with the same Pauli label fuse to give a green charge. As such, we can view the charge configurations shown in Fig. 22 as demonstrating a net charge that has a green label from a global perspective. Of course, as we have mentioned, this is consistent with the physics of a green boundary, where individual green charges can be created locally.

#### D. Interplay between temporal and spatial boundaries and fault tolerance

To investigate the fault-tolerance properties of a quantum computation in topological stabilizer codes, the space-time picture is essential. In this section, we begin by using the established framework of anyon condensation to describe error detection at boundaries. We then analyze the fault tolerance of a space-time computational scheme by studying the configuration of its boundaries.

In Sec. VC, we have developed a microscopic theory for the temporal boundaries in the space-time picture. This allows us to view fault-tolerant quantum computational protocols with the color code as space-time volumes enclosed by one of six boundaries in both the space and time directions, where the boundary types correspond to the Lagrangian subgroups of the color code. We can analyze these volumes to determine processes that give rise to logical errors.

At any given time step of a computation, the logical state of the encoded qubits is determined by the parity of charges condensed at the boundaries (or other condensation objects, such as twist defects, as discussed in Sec. VI). A nontrivial logical Pauli error corresponds to an anyon string supported on a topologically nontrivial path. This means that it cannot be deformed continuously to a point. Note that one has to take into account how anyon strings interact with boundaries and other types of defects when deforming them through space-time. Fault tolerance is achieved when the support of any nontrivial logical error grows with the system size.

To ensure fault tolerance, we have to pick a suitable boundary configuration in our computation such that all of the nontrivial logical errors have a macroscopic length. To illustrate this, let us look at different temporal boundaries to initialize the triangular color code (see Fig. 23). We find that the  $X$  boundary is a valid temporal boundary to initialize this code in a logical  $\bar{X}$  eigenstate. This is because it confines all bosons with a Pauli- $Z$  or Pauli- $Y$  label. This leaves only the three spatial colored boundaries for the logical  $\bar{Z}$ -error ( $\bar{Y}$ -error) strings to terminate [see Fig. 23(c)].

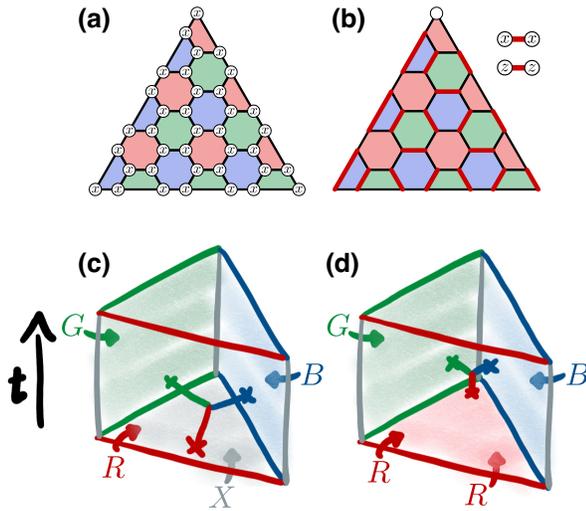


FIG. 23. A triangular color code with colored boundaries can be initialized (and read out) fault tolerantly using one of the three temporal Pauli boundaries. In (a), we show as an example how initializing a  $\bar{X}$  eigenstate is achieved by preparing all physical qubits in an eigenstate of the single-qubit  $X$  operator, creating a temporal  $X$  boundary. If a colored boundary is used, as shown in (b) with the example of a red temporal boundary, the process is not fault tolerant and corresponds to the state injection of the single-qubit state on the qubit in the top corner of the code. To see why this is the case, we can check the support of the logical operators in the space-time picture of the process. In (c), we can see that the weight  $d$  is maintained throughout the process. On the other hand, in (d), we can find a constant-size logical operator on one of the corners.

If we use a red boundary, on the other hand, we can now find a logical error with constant support. It spans between the spatial green and blue boundaries as well as the temporal red boundary [see Fig. 23(d)]. Such configurations can easily be spotted by keeping the space-time diagram of a computational operation in mind. Finally, we would like to point out that even this non-fault-tolerant protocol has its uses in quantum computational schemes. In Ref. [19], a similar protocol is proposed to inject logical non-Pauli eigenstates from single qubits into patches of color code. The equivalent readout protocol teleports the state of the logical qubit onto a single physical qubit.

As a last example, consider the square color code in Fig. 24. In Sec. VB, we have discussed how the logical qubits can be initialized and/or read out using anyon condensation. The basis in which the logical qubits are initialized and/or read out is determined by the strings that can condense at the boundary that is introduced by the condensation. For example, the red temporal boundary in Fig. 24 reads out (initializes) the parity of  $\bar{X}_1$  and  $\bar{Z}_2$ . Importantly, a subset of the logical Paulis are red anyon strings connecting the two red boundaries. Let us turn our attention to the case when the temporal boundary has a different color than all the logical strings that can condense at any spatial

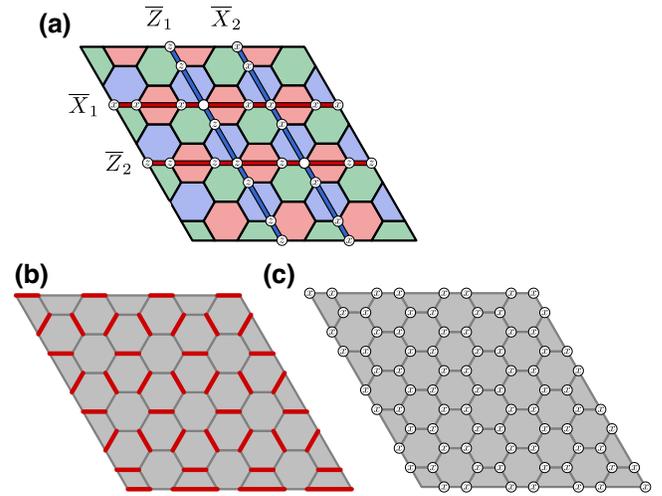


FIG. 24. (a) We show a rectangular patch of color code encoding two logical qubits and its logical operators. This code can be read out or initialized in five different ways in a fault-tolerant manner. (b) An example where all red edges get initialized and/or read out with Bell-pair stabilizers, i.e.,  $XX$  and  $ZZ$ . This corresponds to initializing and/or measuring  $\bar{X}_1$  and  $\bar{Z}_2$ . Choosing instead the blue-colored temporal boundary (not shown) would initialize and/or measure  $\bar{Z}_1$  and  $\bar{X}_2$ . (c) This figure corresponds to interfacing to the vacuum with the Pauli- $X$  boundary by initializing and/or measuring every single qubit in the  $X$  basis. This prepares and/or reads out  $\bar{X}_1$  and  $\bar{X}_2$ . Similarly, choosing the temporal  $Z$  boundary corresponds to the initialization and/or readout of  $\bar{Z}_1$  and  $\bar{Z}_2$  (not shown). Lastly, using the temporal Pauli- $Y$  boundary, we initialize and/or read out the product of  $\bar{Z}_1\bar{X}_2$  and  $\bar{X}_1\bar{Z}_2$  (not shown).

boundary, e.g., a green boundary for the code in Fig. 24. In fact, the logical Pauli- $Y$ s can be represented by green string operators connecting opposing corners. Hence, one might assume that we can use a green temporal boundary to initialize. However, this process is not fault tolerant. Again, we can see this by looking at the boundary configuration in the space-time picture. Here, at all four corners of the code, a green temporal boundary is interfaced with a red and a blue spatial boundary. This means that we can find a operator of constant support that changes the parity of the charges condensed at these three boundaries with constant support, similar to the one shown in Fig. 23(d). Nonetheless, this protocol can be used as an injection or teleportation scheme between the rectangular color code and a  $[[4, 2, 2]]$  code, where the qubits in the corners are the physical qubits of this small code.

The two cases in which we have described an injection or a teleportation between a small code and a color code can be interpreted as a code-switching [75] protocol. Indeed, here we see that we switch between a 2D and a zero-dimensional (0D) code. Alternatively, by condensing anyons in the bulk but not along a boundary of the code, we can switch between a 2D and a 1D code. We can also extend this to the third spatial dimension and interpret the gauge-fixing protocol presented in Ref. [75] as a condensation procedure. Specifically we can obtain a 2D color code from a 3D color code by condensing its charges in the bulk, up to its boundary.

So far, we have discussed the fault tolerance in initialization and readout protocols considering relatively simple examples in which the logical qubits have been encoded in the spatial boundary configuration. In general, all condensation objects have to be considered, such as corners and twists, which are the subject of Sec. VI.

The combination of anyon condensation with the space-time picture results in some no-go [76–80] theorems for stabilizer-based topological quantum computation. In any topological stabilizer code, nontrivial anyon strings define the logical Pauli operators. From this, we can deduce that temporal domain walls can only be used to initialize logical Pauli eigenstates, apply logical Clifford operations, or perform Pauli-basis readouts. To promote this set of topologically protected operations to an universal one, one needs to make use of topological codes in higher spatial dimensions [75]. Alternatively, one could include some operation that is not topologically protected, such as the described state injection, to inject non-Pauli eigenstates. Combined with state-distillation protocols [81] or other appropriate means, this allows us obtain a universal set of fault-tolerant logical operations.

### E. Stability experiments in topological codes

In the following, we discuss memory and stability experiments in topological error-correcting codes. We find that

the theory of spatial and temporal boundaries that we have introduced is well suited to explain stability experiments as well as to devise variants thereof. After reviewing memory and stability experiments for general topological codes, we turn to the color code and show how it can be used to perform a combined memory-stability experiment.

To test the performance of an error-correction code as a quantum memory, we can perform a memory experiment. These experiments consist of three parts. First, we begin by fault tolerantly initializing the logical qubit(s) of the code in a certain state. Next, we let the code idle for a given period of time while measuring its stabilizer generators. Finally, we measure the logical qubit(s) and verify whether or not they have remained in the initialized state. For topological codes, such experiments check if we can tolerate the errors affecting the physical qubits and the gadgets used to perform the stabilizer measurements.

In addition to strings of errors introducing unwanted transformations to logical qubits, logical errors may also occur during a computation due to strings of faults that align in the timelike direction in the space-time picture. This can be a problem, e.g., when we perform gates by code deformations. In recent work [33], a simple experiment has been proposed to check the performance of the toric code against timelike logical errors. The experiment is called the *stability experiment* and consists of a patch of toric code that does not encode any logical qubits. However, it is initialized and read out in a manner that allows us to check for occurrences of strings of noncorrectable errors in the measurements of the stabilizer generators. As the direction of these measurement errors is in the time direction, the experiment effectively checks for temporal logical errors. These are errors connecting distinct temporal boundaries. Considering the full space-time of the experiment, we can see that it is equivalent to a memory experiment “rotated by  $90^\circ$ ,” i.e., where one of the spatial directions is exchanged with the temporal direction. The following discussion of said experiments in terms of spatial and temporal boundaries allows us to construct stability experiments for any topological stabilizer code. We turn to the color code as a concrete example. Interestingly, we find that we can use the color code to perform a stability and a memory experiment simultaneously.

In its simplest form, a stability experiment constitutes a cylinder in space-time with one type of spatial boundary wrapping around the cylinder and a second type of temporal boundary capping it off at the top and bottom. We call the Lagrangian subgroups describing the temporal boundaries  $\mathcal{L}_1$  and the spatial boundary  $\mathcal{L}_2$ . A space-time sketch of the experiment is shown in Fig. 25. We are interested in the anyonic charges that can condense at the top and bottom but not at the sides of the cylinder. These are charges that are not contained in the Lagrangian subgroup describing the spatial boundary  $\mathcal{L}_2$  but can condense at the top or bottom boundary or at the corner between the

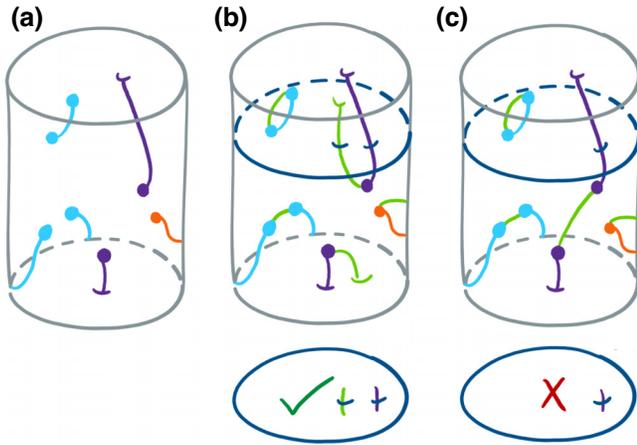


FIG. 25. The space-time diagram of a simple stability experiment. The temporal boundaries at the top and the bottom are of one type and the spatial boundaries on the sides are of a different type. (a) Errors occurring during the experiment correspond to string operators creating anyons in space-time. We depict different types of anyons using different colors. The stability experiment is concerned with anyons that can condense only at the top or bottom boundary (drawn in purple) or at the corners (cyan). Errors creating anyons that can condense on a spatial boundary (orange) are not relevant to the outcome of the experiment. (b) A successful experiment, as the errors together with the applied corrections (green) lead to an even number of string operators of the considered anyons through any time slice (dark blue). This implies that the evaluated conserved quantities are indeed conserved. (c) A failed experiment. Here, the errors together with the applied correction lead to a temporal logical error. We see this as a violation of the conserved quantity, since an odd number of string operators cross a given time slice.

two boundaries. More precisely, we consider anyons in  $\mathcal{L}_1 \times \mathcal{L}_2 \setminus \mathcal{L}_2$ . Depending on the boundaries used, a different number of anyons fulfill the required property. The stability experiment then consists in checking whether an even or odd number of string operators corresponding to the anyons in question cross a given time slice. This can be inferred from the parity of a set of stabilizer measurements. In Ref. [33], the product of this set of stabilizer measurements is called a *conserved quantity* [2,82].

In the color code, a stability experiment can be performed using any combination of spatial and temporal boundaries, as long as two different boundaries are used. This is true since for any pair of different boundaries described by the Lagrangian subgroups  $\mathcal{L}_1$  and  $\mathcal{L}_2$ , respectively, there exists at least one boson that is in  $\mathcal{L}_1$  but not in  $\mathcal{L}_2$ . To maximize the significance of a performed experiment, we want to maximize the number of conserved quantities, i.e., maximize the number of bosons in  $\mathcal{L}_1$  but not in  $\mathcal{L}_2$ . This is achieved by picking either two colored boundaries or two distinct Pauli boundaries. If we suppose that initializing single-qubit states and performing single-qubit measurements is simpler than preparing and

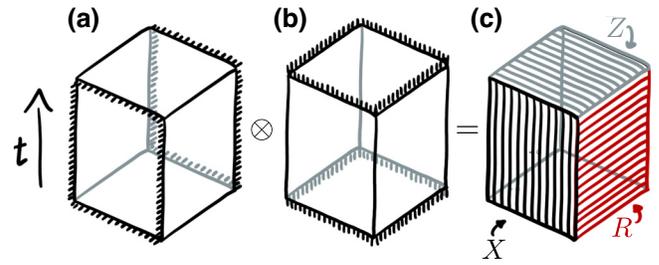


FIG. 26. (a) A surface-code memory experiment. (b) A surface-code stability experiment. If we overlay the two, we obtain a combined stability and memory experiment. As shown in (c), this can be realized within a single color code. It looks like a cube in space-time, where opposite boundaries are of the same type. A suitable boundary configuration has, e.g., two pairs of Pauli boundaries and one pair of colored boundaries.

measuring Bell pairs, using Pauli boundaries is experimentally simpler.

As mentioned above, the goals of stability and memory experiments are similar. In both, we check for the presence of a logical error by validating the parity of a space-time slice perpendicular to the direction of the logical operator. The difference is the direction in which the logical operator runs, a spatial direction for the memory experiment and a temporal direction for the stability experiment. If we overlay two code patches in the toric-code phase, one used to carry out a stability experiment and a second one to perform a memory experiment, we obtain a code patch in the color-code phase. This is sketched in Fig. 26. Such a code is capable of simultaneously checking for the presence of temporal and spatial logical errors within the same experiment, using certain anyon parities. The corresponding space-time diagram is shown in Fig. 26. A range of color-code boundary conditions are suitable to carry out such a combined memory-stability experiment. We might, e.g., initialize and terminate the experiment using temporal Pauli-Z boundaries and terminate the code in the spatial directions using red and Pauli-X boundaries on opposite sides. The microscopic lattice model of this experiment, together with instructions for initialization, readout, and evaluation, are given in Fig. 27 and its caption.

## F. Partial initialization and readout

A temporal domain wall introduced by partial condensation condenses a subset of bosons that do not form a Lagrangian subgroup. This implies that the logical Pauli operators that correspond to the condensed anyons get initialized or read out, depending on the orientation of the domain wall. However, since some anyons remain mobile through the introduced semitransparent domain wall, some logical degrees of freedom remain encoded throughout this process. We hence call the effect of semitransparent temporal domain walls *partial initialization* and *partial readout*.

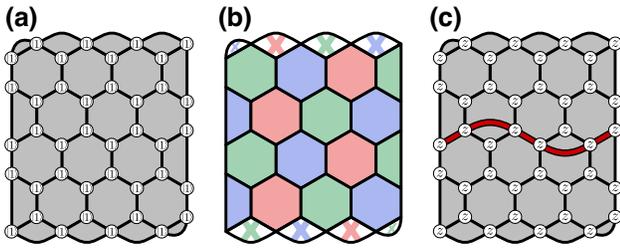


FIG. 27. The combined memory-stability experiment using the color code. (a) We initialize the code using a Pauli-Z boundary by preparing all physical qubits in the  $|1\rangle$  state. (b) The stabilizers of a rectangular patch of color code with two red and two Pauli-X boundaries are measured for  $d$  rounds. (c) We terminate the experiment by performing single-qubit Pauli-Z measurements on all physical qubits. After error correction, a successful experiment reveals an even parity in all blue and green  $X$ -type stabilizers and a  $+1$  outcome for the depicted red string operator.

Note that a semitransparent domain wall can also act non-trivially on the mobile anyons and hence implement logical gates on the associated logical degrees of freedom.

Let us now illustrate partial condensation on exemplary instances of the color code. First, consider two qubits encoded in two punctures in the color code (see Fig. 28). This is equivalent to the example shown in Sec. VB in

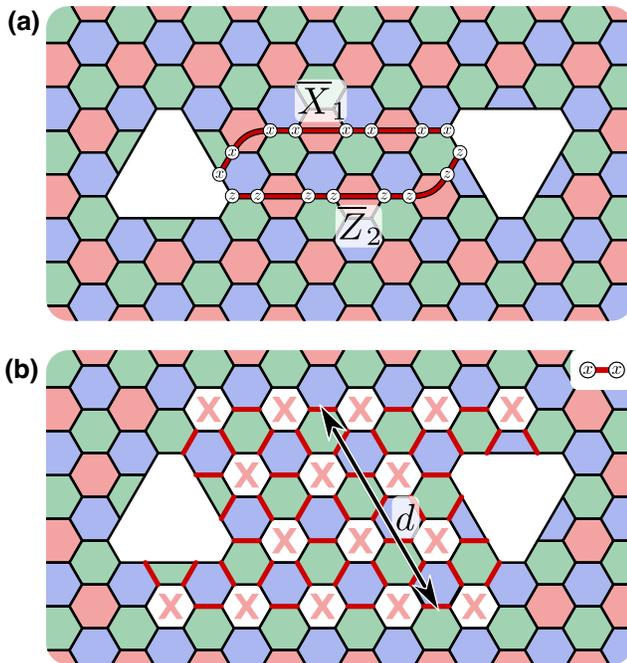


FIG. 28. Two red punctures as shown here encode two logical qubits. (a) The logical operators  $\bar{X}_1$  and  $\bar{Z}_2$ . They can be chosen to have the exact same support. However, it is still possible to read out one but not the other. (b) To measure, e.g.,  $\bar{X}_1$ , we measure the  $XX$  terms on the red edges in a region  $R_L$ . The width  $d$  of the region determines how many fault readouts can be tolerated.

Fig. 16, where we condense a full Lagrangian subgroup to read out two logical qubits at once. We now perform a partial condensation to obtain the value of one logical degree of freedom while leaving another one encoded. For example, we can measure the eigenstates of  $\bar{X}_1$  while leaving the second qubit encoded. We achieve this by condensing  $rx$  in region  $R$ . The second logical qubit remains encoded, as its logical operators commute with all the  $rx$  hopping terms while not being a combination thereof. Similarly, reversing the process lets us encode a second qubit in the punctures. The condensation of different bosons leads to the partial readout of different logical degrees of freedom. For instance, if we condense  $ry$ , we measure the value of  $\bar{X}_1\bar{Z}_2$ . The logical degree of freedom that remains encoded is a logical parity qubit with  $\bar{X} \equiv \bar{X}_1 \simeq \bar{Z}_2$  and  $\bar{Z} \equiv \bar{Z}_1\bar{X}_2$ .

As a second example, let us return to the rectangular color-code patch (see Fig. 29). First, let us consider the partial condensation of the  $rx$  anyon. To condense  $rx$ , we measure the  $XX$  hopping terms on all red edges. From these measurements, we infer the value of the logical operator  $\bar{X}_1$ . The second logical qubit remains encoded in the obtained code. We identify this to be a surface code (see Fig. 3). Similarly, condensation of  $rz$  by measuring red  $ZZ$  terms measures  $\bar{Z}_2$ , leaving the first qubit encoded in the surface code that we produce. If we choose to condense  $ry$ , we measure the product  $\bar{X}_1\bar{Z}_2$ . The surface code now encodes a parity qubit with the logical operators

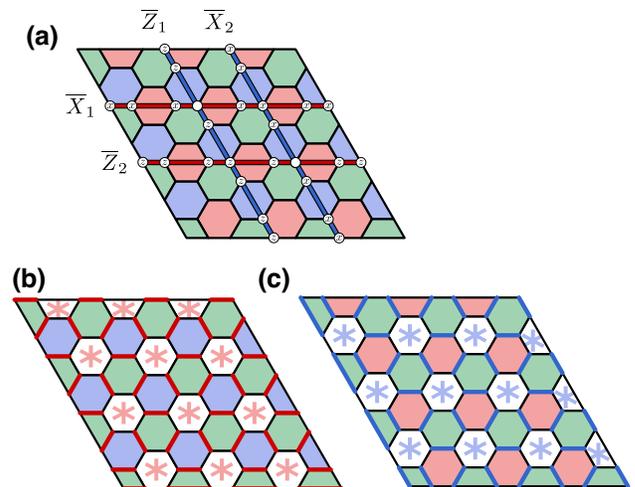


FIG. 29. (a) A color code encoding two logical qubits and its logical operators. A color-code boson gets condensed to obtain a toric code, encoding one logical qubit. Depending on which boson is chosen to be condensed, one logical operator is being read out. The asterisks are placeholders for the Pauli label of the condensed boson. (b) The readout of  $\bar{X}_1$ ,  $\bar{Z}_2$ , or  $\bar{X}_1\bar{Z}_2$  if the chosen boson is  $rx$ ,  $rz$ , or  $ry$ , respectively. (c) Similarly, we condense  $bx$ ,  $bz$ , or  $by$  to read out  $\bar{X}_2$ ,  $\bar{Z}_1$ , or  $\bar{Z}_1\bar{X}_2$ . Condensation of a green boson results in a partial teleportation of one of the logical qubits into a  $[[4, 1, 2]]$  code supported on the four corner qubits.

$\bar{X} \equiv \bar{X}_1 \simeq \bar{Z}_2$  and  $\bar{Z} \equiv \bar{Z}_1 \bar{X}_2$ . These new logical operators are composed of hopping terms of the deconfined bosons  $e \equiv rx \simeq rz$  and  $m \equiv gy \simeq by$ , respectively. Condensation of one of the three blue bosons,  $bx$ ,  $by$ , or  $bz$ , lets us infer the value of the logical Pauli operators  $\bar{X}_2$ ,  $\bar{Z}_1 \bar{X}_2$ , or  $\bar{Z}_1$ , respectively. Condensation of a green boson leads leaves one logical qubit encoded in the obtained surface code. The other qubit, however, is not read out but is now encoded in a  $[[4, 1, 2]]$  code that is supported on the four corner qubits.

The fault tolerance of a computation in the color code involving partial initialization (readout) can be assessed with the same methods as described in Sec. VD. Together with spatial domain walls, this completes the space-time picture of computations in the color code and provides a unified tool to design and study topologically protected computational protocols in 2D topological stabilizer codes.

## VI. TERMINATING THE COLOR-CODE DOMAIN WALLS

In this section, we terminate the domain walls realizable in the color code and study the features emerging at the end points. To terminate a domain wall, we can condense color-code anyons in a 1D spatial open region. This is depicted as a  $(1 + 1)$ -dimensional object in  $(2 + 1)$ D space-time in Fig. 30. Depending on the type of condensation we apply, we obtain different types of domain walls between the color code and itself. While we start the discussion by terminating invertible domain walls, we extend the theory here to include opaque and semitransparent domain walls as well.

In what follows, in Sec. VIA we review the theory of twist defects in the color code [23] and describe how they are manifest in the space-time picture. In Sec. VIB, we investigate the corners of the color code [23], i.e., twist defects that divide two distinct boundary types. We reinterpret corners in terms of the condensates of their adjacent boundaries, to incorporate corners into our theory of anyon condensation with the color code. In Sec. VIC,

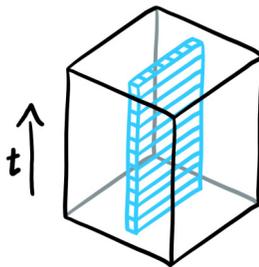


FIG. 30. In this section, we create domain walls between the parent phase and itself. We create them by condensing a 1D spatial region, as depicted by the  $(1 + 1)$ -dimensional cyan region in the  $(2 + 1)$ D space-time figure.

we use the theory of anyon condensation to classify a new type of domain wall that we refer to as semitransparent domain walls. This elaborates on the theory that has been briefly introduced in Ref. [20], where semitransparent domain walls have been employed in fault-tolerant quantum-computing processes with the color code. Finally, in Sec. VID, we discuss the physics of lattice surgery of the color code [19,20], in terms of the semitransparent domain walls that we have discussed throughout this section using the language of anyon condensation.

### A. Invertible domain walls and twist defects

Anyon models have associated domain walls that transform the anyons onto other anyon types as the domain walls are crossed [34,64,65]. These domain walls can be terminated, where we call their end points twist defects or simply “twists.” For the color code, the 72 distinct twist defects are described macroscopically and microscopically in Refs. [21,23]. Here, we will briefly recall some important results that we make use of in the following sections. In particular, we show three examples of invertible domain walls in the lattice model. Furthermore, we talk about the twist defects that terminate at invertible domain walls and how they can be used to store logical qubits. Finally, we discuss how domain walls and twist defects manifest in the space-time picture of topological phases of matter.

Let us start by showing microscopic realizations of three different invertible domain walls and their twist defects (see Fig. 31). In each case, the stabilizers along the domain wall are changed. In the case of the color-permuting domain wall presented in Fig. 31(a), the support of the stabilizers is changed according to the new lattice geometry. This change is such that the tricolorability of the faces is violated by the addition of twist defects to the lattice. This inconsistency in the coloring leads to a permutation of the color label of anyons moving around the twist defect. The domain wall that we show in Fig. 31(b) does not change the lattice geometry but, instead, we change the basis in which the stabilizers along the domain wall act. This leads to a permutation of the Pauli labels of anyons crossing it. Finally, Fig. 31(c) shows a domain wall implementing the duality symmetry between the color and the Pauli label of the color-code anyons. The three domain walls that we show generate all 72 invertible domain walls in the color code [23].

The twist defects at the end points of the domain wall can condense certain color-code anyons. As such, we can find string operators that transport appropriate choices of anyonic excitations between twist defects. These string-like operators correspond to logical operators from the perspective of quantum error correction. By arranging configurations of twist defects on the lattice such that all the twist defects are sufficiently well separated, we encode logical qubits robustly. Twist defects that condense many

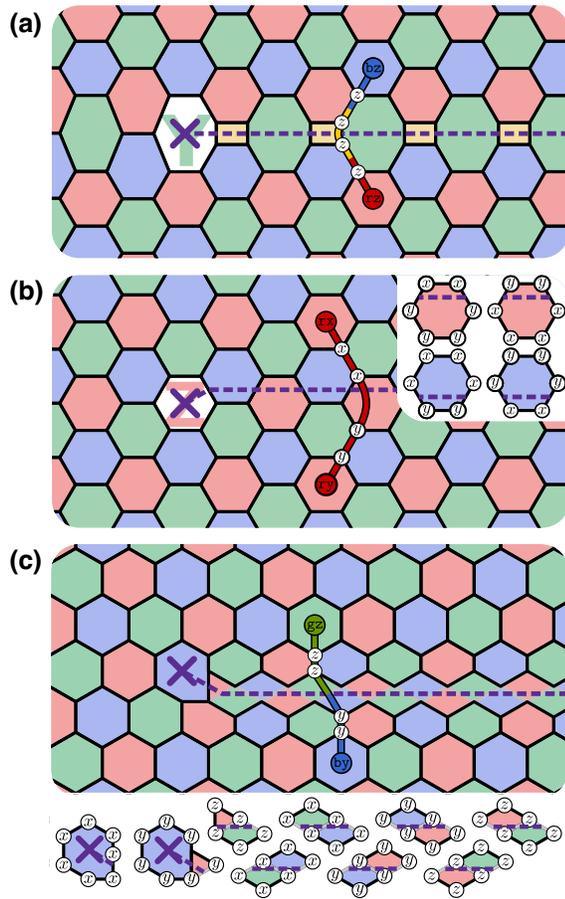


FIG. 31. Three invertible domain walls (purple dashed line) in the color code and the twist defects terminating them (purple cross). In each case, we show one example of an anyon crossing the domain wall. (a) An example of a color-permuting domain wall. Here, the red and blue color labels of anyons crossing the domain wall get exchanged. The green plaquette at the end point hosts only one stabilizer acting in the  $Y$  basis on the seven qubits in its support. (b) A Pauli-label-permuting domain wall that exchanges the Pauli- $X$  and Pauli- $Y$  labels. (c) The duality domain wall that exchanges the color label with the Pauli label.

charges thus increase the size of the logical Hilbert space more than twists that only condense fewer anyons. Formally, we capture this by assigning a quantum dimension to each type of twist defect [64,83–85]. For details on twist defects in the color code, see Ref. [23].

In a space-time picture, we keep track of the position of the twist defect and the position of the physical defect line over time. In doing so, we obtain a 2D membrane for the domain wall that is terminated by the 1D world line of the twist defect, as shown in Fig. 32. As the  $(2+1)$ D topological space-time that we consider is isotropic from a macroscopic perspective, we can deform the world lines of the twist defects arbitrarily and the processes they undergo will be equivalent up to continuous deformations of the world lines of the twist defects. On the other hand, the

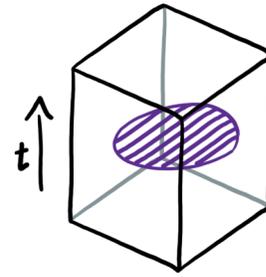


FIG. 32. In space-time, domain walls correspond to 2D membranes and their 1D boundaries are twist defects. We show a pair of twist defects being created and moved apart before they are brought back together and annihilated.

orientation of domain walls affects the microscopic details of the implementation of a defect braiding process in a physical system. These microscopic processes are distinct from an error-correction point of view.

The interactions between twist defects and other topological features are also discussed in Refs. [15,23]. Notably, it is interesting to examine the interactions that can occur as twist defects approach boundaries. The process that occurs depends on two degrees of freedom; first, by the automorphism associated with the domain wall, which gives rise to the twist defect at its end point, an element in  $\text{Aut}(\mathcal{L})$ , and, second, by the Lagrangian subgroup that specifies its bosonic condensate  $\mathcal{L}$ .

We distinguish two cases on how these degrees of freedom—or, equivalently, how the twist defect and the boundary—interplay with each other. In the first case, the associated symmetry leaves the Lagrangian subgroup of the boundary invariant, i.e.,  $\text{Aut}(\mathcal{L}) = \mathcal{L}$ . In this case, an individual twist effectively vanishes at the boundary. We can regard this as twist condensation, named to reflect the analogy between this process and anyon condensation, where an anyon is absorbed, or “vanishes” at the boundary. In contrast, if the associated anyon symmetry nontrivially alters the elements of the Lagrangian subgroup associated with the boundary, such that  $\text{Aut}(\mathcal{L}) = \mathcal{L}' \neq \mathcal{L}$ , then the presence of the twist nontrivially changes the physics of the boundary. Specifically, we find that the twist is confined at the boundary. This object has been termed a “corner” [15,23], as they are often found at the corners of topological codes. We discuss corners in Sec. VIB.

## B. Corners between boundaries

Corners are points on a boundary at which the boundary type changes. They are important features of topological error-correcting codes proposed for quantum computation, such as the surface code [1] and the triangular color code [5] depicted in Fig. 2(a). In the following, we offer two constructive interpretations that can give rise to equivalent corners. First, we view them as confined twist defects.

This allows us to make statements about the computational power of different types of corners. Second, we use the language of anyon condensation to introduce two distinct boundaries such that a corner is prepared between them. This allows for a general procedure to construct corners in topological error-correcting codes.

Earlier in this paper, we have already encountered corners (see, e.g., Fig. 23), which features the triangular color code. This code is terminated by three distinct color boundaries and the three points at which the boundary type changes are corners. We say that an anyon can condense at a corner if it is contained in the union of the two Lagrangian subgroups that describe the boundaries interfacing at the corner. Note that the union of two distinct Lagrangian subgroups, each consisting of anyons of a given color label, contains a generating set of all color-code anyons. As such, a corner interfacing two colored boundaries can condense all 16 color-code anyons. Indeed, the same holds true for corners that interface two distinct Pauli boundaries. We discuss corners that interface a colored boundary with a Pauli boundary later in this section.

Let us now interpret corners as confined twist defects (see Sec. VI A). To this end, we start with a uniform boundary between the color code and the vacuum. Next, we introduce a pair of twist defects and move them close to the boundary. This may change the type of boundary, as anyons now need to cross a domain wall before reaching the boundary. If the anyon-permuting symmetry applied when crossing the domain wall changes the Lagrangian subgroup corresponding to the boundary, then the confined twist defects correspond to a nontrivial corner. If, however, the automorphism corresponding to the twist defect leaves the Lagrangian subgroup invariant, we say that the twist defect condenses and no nontrivial corner is introduced. We show an example of a confining twist in Fig. 33(a). Here, a color-Pauli-duality twist transforms a red boundary into a Pauli- $X$  boundary. Importantly, this interpretation shows us that we can pull the corners out far away from the boundaries, where they can be braided as bulk twist defects to perform logical gates [15]. Similarly, this interpretation allows us to modify the lattice realization of topological codes, which can increase the number of encoded qubits. This is done in Refs. [23,86], where the corners are moved as twists into the center of the lattice. Finally, the insight that we can identify corners with twists can be of use in designing lattice-surgery protocols in certain topological error-correcting codes. In, e.g., Ref. [87], a twist-based modification of the lattice-surgery protocol is presented that allows all three logical Pauli operators in a patch of surface code to be addressed.

A second interpretation of corners is found by noting that they can be created by condensing different Lagrangian subgroups in adjacent regions. This is shown in Fig. 33(b). Here, we see three neighboring regions in

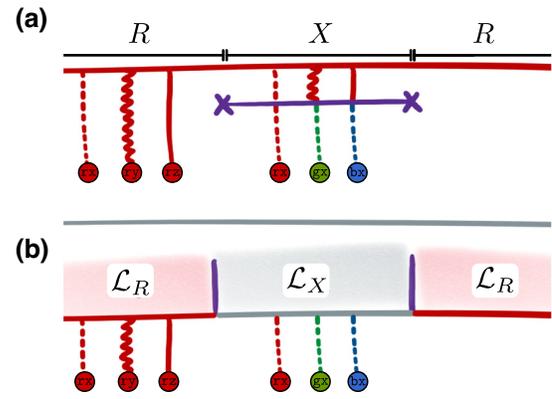


FIG. 33. Corners are the points at which the boundary changes type. They can be interpreted as condensed twist defects. (a) A domain wall close to the boundary, to demonstrate how an invertible domain wall can change the boundary type. (b) This figure explains how to construct boundaries microscopically using anyon condensation. In the red and gray regions, we condense the red and the Pauli- $X$  Lagrangian subgroups  $\mathcal{L}_R$  and  $\mathcal{L}_X$ , respectively.

which, from left to right, we condense the Lagrangian subgroups  $\mathcal{L}_R$ ,  $\mathcal{L}_X$ , and  $\mathcal{L}_R$ . We therefore create a red, a Pauli- $X$ , and a red boundary, respectively. The corners appear at triple points, where the boundaries of these two condensed regions meet the color-code phase. This perspective shows us that corners can be interpreted as end points of opaque domain walls, as shown in Table I. An opaque domain wall is essentially a narrow puncture, which, if it consists of two distinct boundaries, features nontrivial corners at its end points.

Let us now use the second interpretation that we have presented for corners to generalize this class of topological features. Our new construction generalizes the notion of corners that we have already encountered, as the two Lagrangian subgroups describing the neighboring boundaries have nonzero overlap. This means that there exists a nontrivial anyon  $a$  for which  $a \in \mathcal{L}$  and  $a \in \mathcal{L}'$ . We dub these corners “semicorners” by, again, appealing to the unfolded picture of the color code. In an appropriate unfolded picture, we find that color-code semicorners appear as nontrivial corners on only one of the two toric code layers. As opposed to the corners between two distinctly colored boundaries (or two Pauli boundaries), semicorners cannot condense all of the species of color-code anyons. The corner between a red colored boundary and, e.g., a  $X$ -Pauli boundary condenses the following anyons:  $\{1, rx, ry, rz, gx, bx, f_2, f_3\}$ .

As an example of their utility, we can use semicorners to transform color codes with colored boundaries into color codes with Pauli boundaries. We show this process in Fig. 34, where semicorners are depicted as gray lines. As a middle stage of this transformation, we encounter a color code that hosts all six distinct color-code boundaries. In the

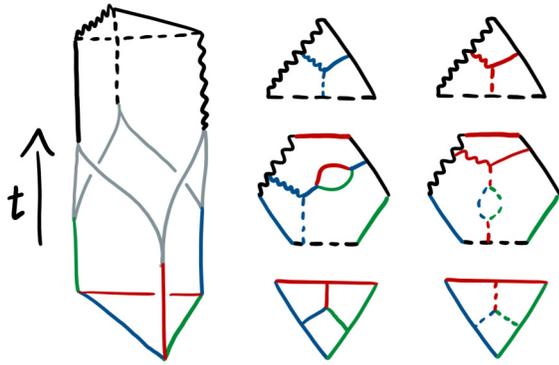


FIG. 34. The space-time picture of a triangular color code undergoing a transformation that changes its boundary type. We show the space-time process on the left while depicting three instances of the 2D code at different times to the right of the figure. In separate pictures of the lattice, we show the various deformations for both the logical  $Z$  and logical  $X$  operators over time. We start at the bottom with a triangular color code with colored boundaries. We then split each of its corners into two semicorners and move them apart. When they meet a semicorner originating from a distinct corner, they fuse together to form a corner interfacing two Pauli boundaries. The dotted, wavy, and solid lines correspond, in turn, to the Pauli- $X$ ,  $-Y$ , and  $-Z$  basis.

following, we unfold this code (see Fig. 35). Interestingly, we obtain the surface code with a twist [86].

### C. Semitransparent domain walls

We have seen examples of topological features that condense charges, confine charges, and also allow deconfined charges to remain mobile. In general, we can find topological features that allow all three of these processes to occur

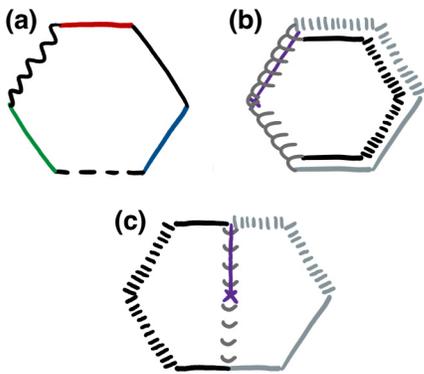


FIG. 35. (a) The hexagonal color code that is terminated by all six different boundaries, hosting six semicorners. (b) The same code unfolded as two stacked toric code layers. The different boundaries correspond to stacked toric code boundaries or a combination of folds and toric-code domain walls, drawn in purple. (c) To obtain one single layer of toric code, we unfold the top layer to the left. This leaves us with one central twist defect. This code is known as the twisted surface code [86].

over the charges of some anyon model. A semitransparent domain wall permits all three of these processes to occur. We obtain a semitransparent domain wall if we perform partial condensation along a 1D subregion of the lattice. More generally, we can obtain and classify all of the semitransparent domain walls by composing them with other domain walls.

In what follows, we classify the semitransparent domain walls of the color code using an abstraction based on the color-code boson table in Eq. (2). This allows us to divide all 162 semitransparent domain walls into eight classes. We can therefore explore how they can be used to store and manipulate logical qubits. We will also describe the physics of a semitransparent domain wall for the color code by viewing it in the unfolded picture. We note that the exposition given in this subsection expands on the discussion given in the appendix of Ref. [20]. Before we present our general classification of semitransparent domain walls, let us first show an explicit microscopic example.

We introduce a semitransparent domain wall to the color code by applying a partial condensation to a 1D subregion of the lattice. As discussed in Sec. III B 2, one color-code boson is chosen to be identified with the trivial charge in order to perform a partial condensation. In Fig. 36(a), we show a domain wall where the  $\tau_x$  anyon is chosen to be condensed on the color-code lattice. One can check that this domain wall condenses the  $\tau_x$  charge no matter on which side it approaches the domain wall. For a general semitransparent domain wall, however, this is not the case and anyons approaching the domain wall from opposite sides might behave differently. We can construct one such example microscopically by composing two distinct bosons to be condensed along adjacent 1D regions. In Fig. 36(b), we show a horizontal semitransparent domain wall that we construct by condensing  $\tau_x$  above the domain wall and  $\sigma_y$  below the domain wall. We follow the procedure laid out in Sec. III D to obtain a valid stabilizer realization.

We obtain a clearer understanding of the physics of semitransparent domain walls of the color code by unfolding them into two layers of the toric code [24,26]. In Fig. 37(a), we show an example of an unfolded semitransparent domain wall. Here, the top layer hosts a narrow puncture, while the bottom layer is connected across the domain wall. Anyons on the bottom layer can pass through the domain wall, i.e., they remain mobile, while anyons on the top layer either condense or confine according to the chosen boundary type of this toric code layer. While this is the simplest example of an unfolded semitransparent color-code domain wall, the general case can be obtained from it readily. More precisely, we redundantly obtain all possible semitransparent domain walls in the color code from this simple example by adding transparent domain walls to either side of the displayed semitransparent domain

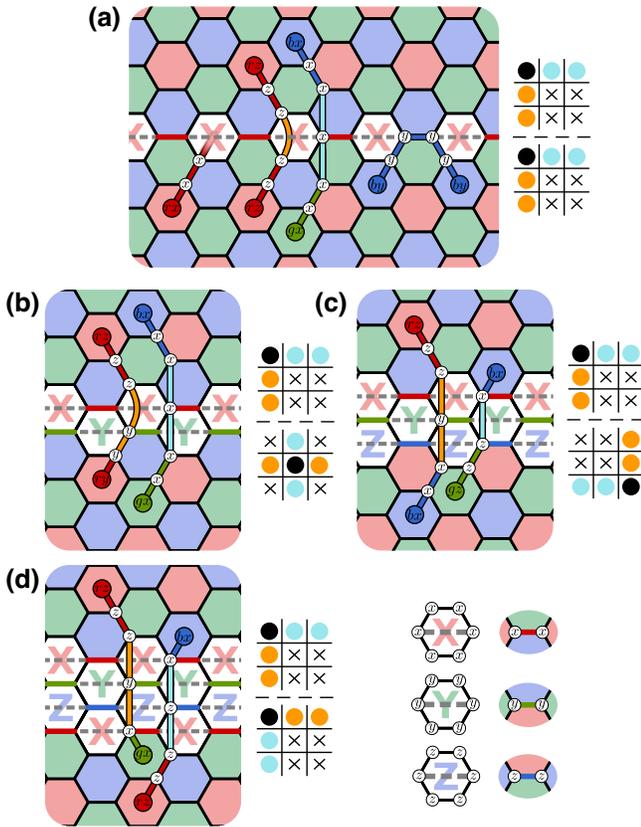


FIG. 36. Four different semitransparent domain walls realized microscopically in the color code. Each colored plaquette hosts an  $X$ - and a  $Z$ -type stabilizer. All exceptions and additional stabilizers are given in the bottom right of the figure. In each case, we present the lattice realization of the domain wall and show different anyons crossing the domain wall. In (a), we additionally show the  $rx$  charge condensing and a  $by$  charge being confined to one side of the domain wall. Next to the lattice realization of the domain wall, we draw the effect the domain wall has on all color-code anyons in terms of the color-code boson table, as discussed in the main text. (b), (c) and (d) show other examples of semitransparent domain walls together with their corresponding boson tables. Deconfined excitations are also shown on each lattice.

wall. One such example is shown in Fig. 37(b), where we additionally introduce a layer-swapping domain wall.

Clearly, one can conceive of many different types of semitransparent domain wall for the color code. In order to catalogue the semitransparent domain walls of the color code, we will once again make use of the boson table in Eq. (2). Specifically, we take two copies of the boson table, one corresponding to the “top side” of the domain wall and the other corresponding to the “bottom side.” Examples are shown on the right of the domain walls in Fig. 36. We make use of the notation introduced in Sec. III B 2 to denote anyons that condense, those that confine, and those that remain deconfined. On either side, one of the nine bosons is condensed, marked by a  $\bullet$  label. The other eight bosons

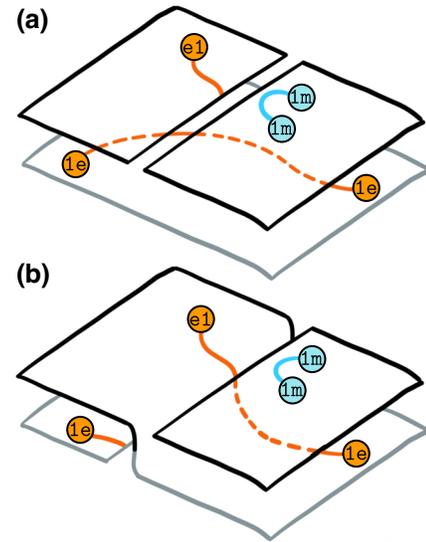


FIG. 37. We show two semitransparent color-code domain walls after unfolding. (a) In the simplest case, the domain wall corresponds to a narrow puncture on one layer (top), while the other layer (bottom) is continued. This means that charges on the bottom layer remain mobile, while charges on the top layer condense or confine. From this simple example, we can construct all semitransparent domain walls in the color code by adding a transparent domain wall. One such example is shown in (b), where a layer-swapping domain wall is introduced.

then get marked by  $\times$ ,  $\bullet$ , or  $\circ$ , depending whether they confine or deconfine at the domain wall. Charges marked with  $\times$  remain confined to their corresponding side of the domain wall. The remaining anyons are deconfined and remain mobile, i.e., can be moved across the domain wall. We label these remaining anyons  $\bullet$  or  $\circ$ , corresponding to electric and magnetic charges, respectively. Upon transmission through the domain wall, anyons marked with the  $\bullet$  ( $\circ$ ) labels of the top side are mapped onto anyons marked with the  $\bullet$  ( $\circ$ ) labels of the bottom.

This characterization using two boson tables suffices to find the total number of semitransparent domain walls. First of all, we can choose which of the nine color-code bosons we condense,  $\bullet$ , on both the top side and bottom side of the domain wall, arbitrarily. This fixes the confined charges on each grid. We have one final degree of freedom, namely, how the mobile charges from the top get mapped to the mobile charges on the bottom. Without loss of generality, let us arbitrarily fix the  $\bullet$  and  $\circ$  labels on the top grid. There are now two possible choices to configure the  $\bullet$  and  $\circ$  labels on bottom grid. Given these rules, let us now count the semitransparent domain walls. Given the nine choices of condensing anyons on the top- and bottom-side grid, we obtain  $9 \times 9 = 81$  semitransparent domain walls. Then, together with the binary choice for how to configure the  $\bullet$  and  $\circ$  labels on the bottom grid, we arrive at  $81 \times 2 = 162$  semitransparent domain walls.

Let us now consider the end points of these semitransparent domain walls. We call them “semitwists.” We will also briefly discuss how they can be characterized and used to store quantum information in a robust manner. To interpret semitwists, we can follow the unfolding procedure above to obtain two decoupled layers of toric code. See Ref. [69], where objects that can be interpreted as semitwists on a single layer of the toric code are discussed. In the case of the color code, we find that for any domain wall, there exists an unfolding in which it is composed of a narrow puncture (with possibly two distinct boundaries) on one layer of toric code and an invertible domain wall (possibly the trivial one) on the other. Hence, the semitwists can be regarded as corners “on top of” twists.

Both corners as well as twist defects can condense certain charges. This can be used to encode logical information in pairs of semitwist defects. The associated logical operators are either string operators transporting a charge from one semitwist to another or strings wrapping around a pair of semitwists. Examples of the logical operators associated with semitwists are shown in Fig. 38.

Each of the 162 semitwists can be associated with one of eight classes. The classes are obtained by checking if the condensing anyons on the top side and the bottom side share both their color and Pauli label (class 1), just the Pauli label (class 2), just the color label (class 3), or neither

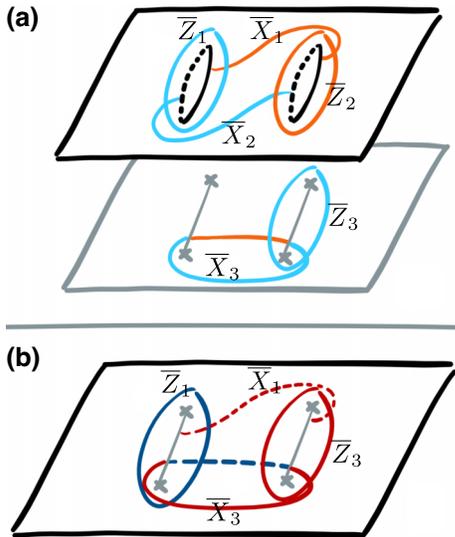


FIG. 38. Two pairs of semitwists are shown, together with some of the logical operators they support. (a) The unfolded version, where the semitransparent domain walls correspond to punctures on the top layer and an invertible toric-code domain wall on the bottom layer. (b) The same configuration in the color code. Here, we omit the logical operators  $\bar{X}_2$  and  $\bar{Z}_2$  for clarity. In the toric code, we represent logical operators corresponding to electric (magnetic) anyons  $e$  ( $m$ ) with orange (blue) lines. In the color code, we draw the color corresponding to the color label (red or blue, here) and the Pauli- $X$  ( $-Z$ ) label corresponds to the dashed (solid) lines.

TABLE II. The 162 semitransparent domain walls in the color code can be separated into eight classes. Here, we show an example of one member of the class as well as the number of distinct domain walls in each class.

	1	2	3	4
A	● ○ ○	● ○ ○	● ○ ○	● ○ ○
	○ × ×	○ × ×	○ × ×	○ × ×
	○ × ×	○ × ×	○ × ×	○ × ×
	9	18	18	36
B	● ○ ○	● ○ ○	● ○ ○	● ○ ○
	○ × ×	○ × ×	○ × ×	○ × ×
	○ × ×	○ × ×	○ × ×	○ × ×
	9	18	18	36

of the two labels (class 4). Within each class, we introduce a subclass  $A$  or  $B$  depending on how the mobile anyons get mapped when crossing the domain wall. If “rows get mapped to rows” and “columns get mapped to columns,” we are in subclass  $A$ , whereas if “rows get mapped to columns” and vice versa, we are in subclass  $B$ . Table II shows an example of a semitransparent domain wall in the boson-table notation for each of the eight classes as well as the number of elements in each of the classes.

Figures 36(a)–36(d) correspond to the classes 1A, 4B, 4A, and 1B, respectively. Examples of members of classes 2 and 3 are obtained by adding a color- or Pauli-permuting invertible domain wall to either side of one of the examples shown. This is shown in Fig. 39, where a semitransparent domain wall in class 1A gets transformed to a class 3A (2A) domain wall by complementing it with a Pauli- (color-) permuting invertible domain wall.

Semitransparent domain walls have previously been described abstractly in Ref. [34]. In general, a domain wall is described by two “tunneling maps” describing how the bulk excitations get transformed when approaching the domain wall from either side. In this picture, nontransparent domain walls correspond to noninvertible tunneling maps. In the Appendix, we describe how to explicitly calculate the tunneling map in the sector of trivial wall excitations for phases equivalent to stacks of toric code, based on Ref. [63]. Calculation of the full tunneling map for arbitrary phases goes beyond the scope of this work but will be covered in future work (see Ref. [88]).

Let us conclude this section by remarking that semitransparent domain walls appear in color-code lattice-surgery protocols [19,20]. In fact, lattice surgery with the color code makes use of all the domain walls presented in

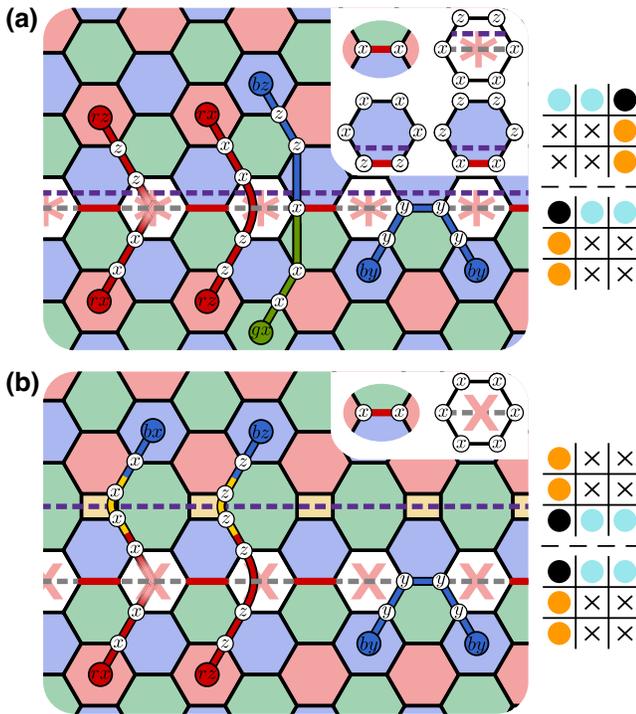


FIG. 39. Semitransparent domain walls change type when combined with an invertible color-code domain wall. Here, we show a domain wall in class  $1A$  (as depicted in Fig. 36(a), marked by the gray dashed line), and how it transforms when an invertible domain wall (dashed purple line) is introduced next to it. In (a) a Pauli-label-permuting domain wall, equivalent to the one shown in Fig. 31(b), is introduced next to it, transforming it to a domain wall in class  $3A$ . In (b) we introduce a color-permuting domain wall, as depicted in Fig. 31(a), to obtain a class  $2A$  domain wall.

this work so far—opaque, semitransparent, and invertible ones in both the temporal and spatial orientations. This is the subject of Sec. VID.

#### D. Lattice surgery

Lattice surgery is a protocol to make fault-tolerant joint measurements of Pauli observables between multiple logical qubits [14,15,23,89]. These fault-tolerant operations are carried out by merging and subsequently splitting disjoint code patches, which we achieve by changing the measured stabilizer terms. A sufficiently large set of lattice-surgery operations can implement the Clifford group by measurement. Together with the preparation of noisy magic states and distillation protocols, we recover a universal set of fault-tolerant logic gates.

Lattice-surgery methods give rise to resource-efficient proposals for implementing fault-tolerant logical gates in topological error-correcting codes [14,19,20,89]. In Ref. [20], the color code has been shown to have an advantage over other topological codes in terms of the resource cost of its implementations. Achieving this advantage requires the use of all different types of color-code

boundaries and domain walls. In what follows, we aim to tie together the above discussion of anyon condensation in the color code in order to understand the role of these boundaries and domain walls appearing in overhead-efficient color-code lattice-surgery-based quantum computation.

Let us begin by considering a simple example of a lattice-surgery operation that captures many elements of the physics of more complex merging and splitting operations. In its simplest form, lattice surgery merges two disjoint color-code lattices along their adjacent boundaries before they are subsequently split [19]. This is shown in Fig. 40. In the initial configuration, we start with two disjoint triangular color codes. Next, the two codes are merged. To achieve this merging operation, we measure stabilizers between the two red boundaries of the two code patches to create one large code patch. Finally, we split the two codes again by measuring the initial stabilizers to return to the original code space.

Looking at the stabilizers measured during the lattice-surgery protocol depicted in Figs. 40(c) and 40(f), we can

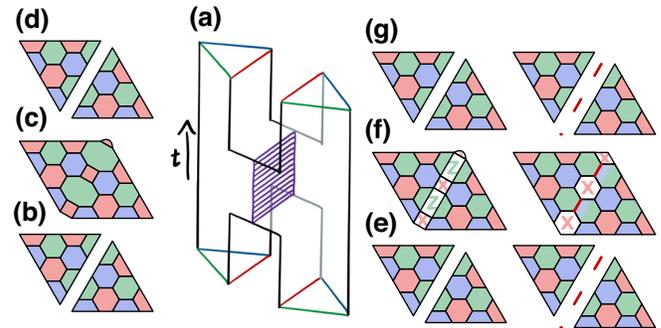


FIG. 40. Lattice surgery between two triangular color codes. (a) The space-time diagram of the operation, where two distinct triangular color-code patches are merged and then split again. The purple region depicts a domain wall. (b)–(d) Time slices of the process. In (b), we depict the initial configuration, where we start with two logical qubits encoded in two disjoint triangular color codes. In (c), we measure additional stabilizers supported on qubits on the red boundaries of both original triangular color codes. In doing so, we obtain two bits of classical information from reading out the  $\bar{X}_1\bar{X}_2$  and  $\bar{Z}_1\bar{Z}_2$  logical parities. Merging the codes as shown introduces a trivial domain wall in the purple region. In (d), we revert to the initial configuration by measuring the initial stabilizer generators to split the big code patch. (e)–(g) Two possible ways of reading out only  $\bar{X}_1\bar{X}_2$  shown to the left and right of these panels, where time steps progress sequentially from (e) through to (g). In this case, a semitransparent domain wall is introduced in the purple region. Two different microscopic realizations are shown. The left one does not require the use of additional auxiliary qubits [for the microscopic details of the stabilizers, see Fig. 41(2)]. The one on the right does require auxiliary qubits but features only stabilizer measurements of weight 6 or lower. Note that the obtained semitransparent domain wall in this case is the one introduced earlier in Fig. 36(a).

see that the product of the red  $X$ -type ( $Z$ -type) stabilizers in the seam is the product of two logical operators of the original code patches, namely,  $\bar{X}_1\bar{X}_2$  ( $\bar{Z}_1\bar{Z}_2$ ). Hence, by measuring these additional stabilizers, we obtain the logical readings for the logical operators  $\bar{X}_1\bar{X}_2$  ( $\bar{Z}_1\bar{Z}_2$ ). Finally, we split the code patch to obtain the two initial triangular color codes again. Their logical qubits are now prepared in a Bell state, with the explicit state depending on the outcome of the logical parity measurements.

Domain walls play a central role in lattice-surgery operations. Let us study the domain walls that we obtain along the seam when performing lattice surgery in our example of two triangular color codes. In the process depicted in the left of Fig. 40, we have converted an opaque domain wall [Fig. 40(b)], which consisted of two red boundaries and did not let any anyons pass from one code to the other, into a fully transparent domain wall [Fig. 40(c)], effectively joining the two codes together.

Performing a different measurement leads, in general, to a different domain wall. For instance, consider the case depicted on the right of Fig. 40. Here, we measure only  $\bar{X}_1\bar{X}_2$ . This requires us to find a set of commuting stabilizers that multiply to  $\bar{X}_1\bar{X}_2$  while leading to a logical code patch that still encodes one logical qubit. We show a valid solution in Fig. 40(f). Note that this measurement results in the semitransparent domain wall shown in Fig. 36. Similar protocols have been considered in Refs. [19,90].

Let us generalize the observed behavior and investigate the connection between the different types of domain walls and the types of Pauli word(s) being measured. Microscopic examples for the case of two code patches are given in Fig. 41. In this encoding, we show two code blocks, where both the top and bottom code block encode two logical qubits over a red boundary with blue boundaries on either side. The red boundary of the upper patch supports the logical operators  $\bar{X}_1$  and  $\bar{Z}_2$  and the red boundary of the lower patch supports  $\bar{X}_3$  and  $\bar{Z}_4$ .

- (0) The trivial case, in which no measurement is performed, results in an opaque domain wall. This occurs when logical qubits are left idling.
- (1) Invertible domain walls are obtained if two commuting Pauli words that address two different logical degrees of freedom on each boundary are measured. In the example presented in Fig. 41(1), we measure  $\bar{X}_1\bar{X}_3$  and  $\bar{Z}_2\bar{Z}_4$ .
- (2) As discussed before, a semitransparent domain wall is obtained if only one of the two degrees of freedom supported on each boundary of a code patch is addressed. As an example, a measurement of  $\bar{X}_1\bar{X}_3$ , as in Fig. 41(2), results in a semitransparent domain wall.
- (3) If one of the boundaries involved in the lattice surgery has support on an even number of qubits, as is the case for the rectangular color code, it supports

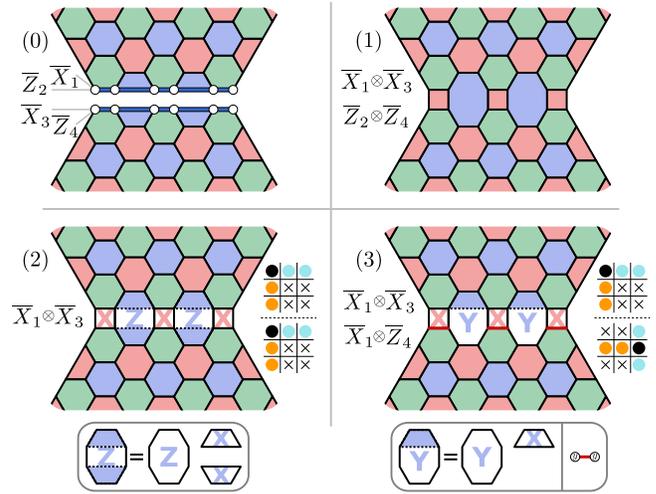


FIG. 41. The four cases of possible measurements for two-patch color-code lattice surgery. The logical operators supported on the interfacing boundaries of the code patches are shown in the top left. (0) The trivial case, in which no parity measurement is performed, leads to an opaque domain wall between the two patches. (1) An invertible domain wall is obtained if two different logical operators on each boundary are addressed. A semitransparent domain wall is obtained when only one of the logical operators is addressed in both code patches (2) or in just one of the code patches (3).

two commuting logical operators. In this case, it is possible to perform two commuting measurements involving the same logical operator on the other code patch. For instance, as in Fig. 41(3), we might measure  $\bar{X}_1\bar{X}_3$  and  $\bar{X}_1\bar{Z}_4$ . Such a measurement also leads to a semitransparent domain wall.

Having discussed the different cases arising in two-patch color-code lattice surgery, let us now move on to measurements of Pauli code words with support on a larger number of patches. General lattice-surgery operations are discussed in more detail in Ref. [20]. We consider the example shown in Fig. 42. Here, we aim to measure the two Pauli words  $W_1 = \bar{X}_1\bar{X}_2\bar{X}_3\bar{1}_4\bar{1}_5\bar{1}_6$  and  $W_2 = \bar{Z}_1\bar{Z}_2\bar{X}_3\bar{1}_4\bar{Z}_5\bar{1}_6$ , which are encoded on five distinct patches of color code. Note that the two Pauli words commute:  $[W_1, W_2] = 0$ . This implies that we are able to measure them at the same time using color-code-based lattice surgery [20]. In the example, we find triangular code patches (see Fig. 2) as well as a rectangular code patch (see Fig. 24). Note how the rectangular patch encodes two logical qubits, indexed 4 and 5, leading to Pauli words of length  $n = 6$ . To perform the measurement, we introduce an auxiliary patch in the center, such that it neighbors a boundary of each of the five code patches surrounding it. Furthermore, we choose the boundaries interfacing with code patches that are acted on nontrivially by both  $W_1$  and  $W_2$  to be red, and of Pauli type if they are acted

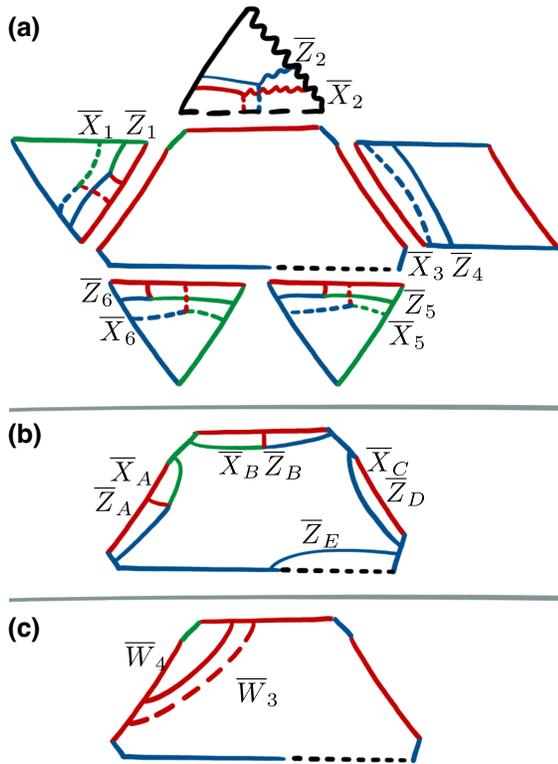


FIG. 42. An example of two Pauli words being measured in parallel using color-code lattice surgery. In the example shown, we measure  $W_1 = \bar{X}_1 \bar{X}_2 \bar{X}_3 \bar{I}_4 \bar{I}_5 \bar{I}_6$  and  $W_2 = \bar{Z}_1 \bar{Z}_2 \bar{X}_3 \bar{I}_4 \bar{Z}_5 \bar{I}_6$ . The boundary configuration on the data patches [shown in (a)] and the logical Pauli words that we are reading out,  $W_1$  and  $W_2$ , determine the boundary configuration on the auxiliary patch. For instance, the boundary interfacing with the fifth qubit is a Pauli- $X$  boundary, as it is only involved in one of the two Pauli-word measurements that we perform. The auxiliary patch is initialized and read out using a red temporal boundary. This ensures that it is not initialized in an eigenstate of the logical operators depicted in (b) and we therefore avoid obtaining information about individual logical Pauli observables during the merging step. During readout we obtain information about the products  $W_1 W_3$  and  $W_2 W_4$ , where  $W_3 = \bar{X}_A \bar{X}_B \bar{X}_C$  and  $W_4 = \bar{Z}_A \bar{Z}_B \bar{Z}_D \bar{Z}_E$ , shown in (c), are the products of the logical operators in (b). These are then measured during the readout step. During the splitting step where we measure the red edge terms, we measure the values of  $W_3$  and  $W_4$ , thereby enabling us to recover the values of the Pauli words  $W_1$  and  $W_2$ .

on nontrivially only by one of  $W_1$  and  $W_2$ , such as the Pauli- $X$  boundary interfacing with code patch number 5. The exception is the boundary interfacing with the code patch encoding qubit number 6, which only gets acted on trivially by both  $W_1$  and  $W_2$ .

Let us study the auxiliary central patch in its own right. It encodes five logical qubits, which we label based on the logical operators that are supported on the boundaries interfacing the code patches [see Fig. 42(b)]. In the lattice-surgery protocol, we measure the parity of the depicted logical operators on the auxiliary code patch and

the logical operators on the surrounding code patches. In order to measure only  $W_1$  and  $W_2$ , and no additional information, we initialize the auxiliary code patch using a red temporal boundary, ensuring that it is not initialized in an eigenstate of the depicted logical operators. As shown in Fig. 42(c), the products of the logical operators in (b) are red strings, namely,  $W_3 = \bar{X}_A \bar{X}_B \bar{X}_C$  and  $W_4 = \bar{Z}_A \bar{Z}_B \bar{Z}_D \bar{Z}_E$ . Thus, initializing with a red temporal boundary initializes the auxiliary code patch in an eigenstate of  $W_3$  and  $W_4$ . Now, by performing the merging step in the lattice-surgery protocol, we measure the products  $W_1 W_3$  and  $W_2 W_4$ . Importantly, the measurements that we have performed do not commute with  $W_3$  and  $W_4$ . Thus, in order to infer  $W_1$  and  $W_2$ , the values of  $W_3$  and  $W_4$  need to be obtained during the splitting step [14]. This is achieved by reading out the auxiliary code patch using a red temporal boundary.

In the example we have examined, we encounter all types of domain walls. The opaque domain wall between the auxiliary code patch and the triangular code patch encoding logical qubit number 6 is maintained throughout the protocol. The domain walls between the auxiliary patch and the triangular code patches numbers 1 and 2 are invertible or fully transparent. Note that the domain wall to the patch number 2 is nontrivial, as it joins a red boundary and a Pauli- $X$  boundary together. The two domain walls between the auxiliary patch and the patch encoding qubits 3 and 4 as well as the one encoding qubit number 5 are semitransparent. Here, the former is of type (3) in the above discussion, while the latter is of type (2).

With this example, we hope to have elucidated how color-code lattice surgery can be used to efficiently perform logical gates on a wide range of logical encodings by making use of its rich set of domain walls. This should aid the design of further schemes using unconventional encodings, such as the thin color code proposed in Ref. [20], which might be preferable in architectures with biased noise.

To conclude this part of the work, let us summarize how to translate from a circuit describing a quantum computation to a fault-tolerant implementation using the color code. We make our summary using the example presented in Fig. 43, where we make use of all of the features discussed in Secs. IV–VI. The operations that we present give rise to a universal set of fault-tolerant logical operations for topological stabilizer codes, where non-Clifford gates are performed with magic state distillation. As we have now elaborated, all of these operations are described, both macroscopically and microscopically, using the unifying language of anyon condensation.

## VII. DYNAMICALLY DRIVEN CODES

Dynamically driven “Floquet” codes [36] are a generalization of subsystem codes where a time-ordered

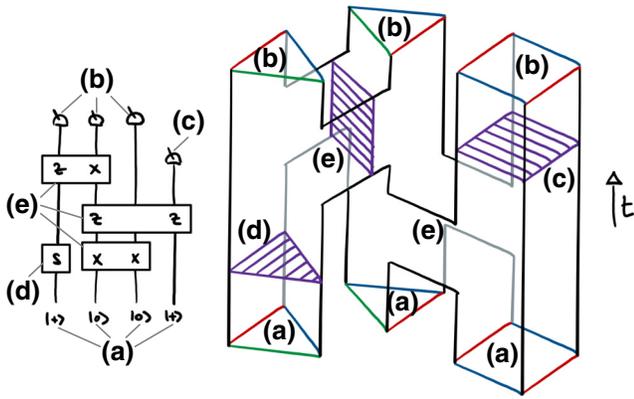


FIG. 43. A circuit (left) is translated to a fault tolerant computation in the color code (right). All fault-tolerant logic gates are described with the condensation processes that we present in Secs. IV–VI. The spatial boundaries dictate the number of encoded qubits per code patch. Logical qubits are initialized (a) and read out (b) with temporal boundaries. A partial readout (c) corresponds to a domain wall obtained by partial condensation. Transversal gates (d) are applied using invertible domain walls corresponding to trivial condensation. To perform parity measurements between logical qubits, we deform the spatial boundaries of the code patches and introduce domain walls between them in lattice-surgery protocols (e).

sequence of gauge-operator measurements, or “checks,” is specified to measure the syndrome data. This generalizes subsystem codes [91] that are described by a generating set of all check measurements with no explicit time ordering. The honeycomb code [36] is an example of a Floquet code that has received considerable attention [37,38,71–73] following its recent discovery, due to its practical implementation. Specifically, the honeycomb code has weight-6 stabilizers that are inferred using only weight-2 parity measurements. Furthermore, the honeycomb code can be realized on a planar lattice with boundaries [37].

The honeycomb code demonstrates the significance of specifying the order in which check operators are measured [36]. If we describe the honeycomb code as a subsystem code, where its gauge group is generated by its full set of check measurements, we arrive at a subsystem code that encodes no logical qubits (for an introduction to subsystem codes, see Ref. [91]). Nevertheless, by specifying a constrained sequence of measurements where only a subset of the generators of the gauge group are measured at each time step, we find that each round of check measurements projects the system onto a new instantaneous code, such that logical qubits with an arbitrarily high distance are encoded in the system.

Here, we find that anyon condensation gives us a complementary perspective of dynamically driven codes. To this end, we present a general construction for new types of such codes that we call dynamically condensed color codes, among which the honeycomb code is included. We obtain this generalization from the observation that all of

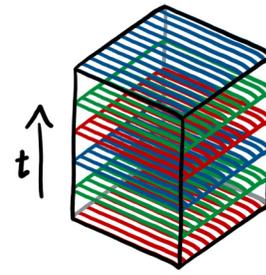


FIG. 44. In this section, we derive dynamically driven codes by condensing the confined charges of different instances of the partially condensed color code in an ongoing sequence. The figure shows condensation operations in the space-time picture as colored planes that lie orthogonal to the timelike direction.

the instantaneous codes of the honeycomb code are examples of partially condensed color codes. By regarding the color code as a parent theory from which the instantaneous toric code states of a dynamically driven code can be derived, we can identify new transitions between different instances of the toric code via sets of weight-2 projective measurements that act on the edges of the color-code lattice. Specifically, we find that we can reproduce the transformations of dynamically driven codes by condensing color-code anyons that have been confined by a previous condensation operation. Using other choices of edge measurements to make checks on the color-code lattice gives us additional freedom to design new dynamically driven codes. A sketch of our framework is shown in Fig. 44.

We use our construction to introduce one specific example of a Floquet code that is of *Calderbank-Shor-Steane* (CSS) type. We find that by measuring only weight-2 parity checks on the edges of a three-colorable lattice, we can infer the values of both Pauli- $X$ -type and Pauli- $Z$ -type stabilizers for each plaquette. We can use the outcomes of these stabilizers to detect the occurrence of errors. As the stabilizers that we measure are precisely those of the color code, we term the code the *Floquet color code*. We note that this code defined with periodic boundary conditions has been discovered independently in Ref. [39]. Let us remark that one might regard this choice of name as a misnomer. Although we measure check operators that infer the values of color-code stabilizers, the Floquet color code emulates the ground space of the toric-code phase.

The Floquet color code represents a generalization beyond other examples of known dynamically driven codes. When expressed as a subsystem code, where the check operators are the generators of a gauge group, the Floquet color code has no geometrically local stabilizer operators. In contrast, the honeycomb code, for example, maintains a constant set of stabilizer operators the cardinality of which is extensive in the system size. This is related to the fact that the Floquet color code emulates the toric-code phase. Although we measure all of the color-code

stabilizers over a period of the Floquet color code, we never produce a simultaneous eigenstate of all of the color-code stabilizers at a single instant of the period of the Floquet color code. Specifically, this is because each time we perform a set of check measurements, we kick our system out of eigenstates of stabilizers that do not commute with the check operators that are measured.

One might be troubled that our new dynamically driven code does not maintain a constant stabilizer group. However, we present numerical results showing that our code demonstrates a threshold comparable to the honeycomb code [38]. Our example therefore shows that no terms in the centralizer of a subsystem code need to remain sacred in order to encode and protect logical quantum information. We note that the observation that the Floquet color code does not have a constant stabilizer group has been made in independent work [39,40]. Another Floquet code with this property, the automorphism code, is also presented in Ref. [74].

Two-dimensional codes can be realized on a planar qubit array by introducing boundaries. We therefore require suitable transformations for the boundary stabilizers of dynamically driven codes as we make deformations between instantaneous code states. We find that regarding the color code as a parent theory for dynamically condensed color codes reveals a general rule to find appropriate boundary conditions as we perform the code deformations of a dynamically driven code. We complete this section by explaining boundary transformations from the perspective of bulk-condensed color codes, before offering some concluding remarks on dynamically driven codes.

### A. Dynamically condensed color codes

In what follows, we will describe the instantaneous stabilizer group for dynamically condensed color codes. We will explain how the instantaneous stabilizer group is transformed as we make different choices of weight-2 edge measurements on the three-colorable lattice.

Dynamically driven codes are described by a series of instantaneous stabilizer groups. Each round of check measurements that are made in the sequence projects the system onto a new instantaneous stabilizer group. Each of the instantaneous stabilizer groups of dynamically condensed color codes  $\mathcal{S}_a$  are obtained by condensing boson  $a$  of the color code. The honeycomb code [36], for example, moves, up to a local basis change, through a series of three instantaneous stabilizer groups,  $\mathcal{S}_{rx}$ ,  $\mathcal{S}_{gy}$ , and  $\mathcal{S}_{bz}$ . In practice, we find that we can transform between any two anyon-condensed color codes,  $\mathcal{S}_a$  and  $\mathcal{S}_b$ , provided that  $a$  and  $b$  correspond to bosons of the color code that share neither the same color nor Pauli label.

The deconfined charges of the condensed color code give rise to logical operators of an instantaneous stabilizer group. Specifically, we regard physical string operators

that transport deconfined charges over large nontrivial cycles of the lattice as the extensive logical operators. In the case of  $\mathcal{S}_{rx}$ , the logical operators are generated by the string operators for  $ry$ ,  $rz$ ,  $gx$ , and  $bx$  charges.

The edge terms for an instantaneous stabilizer group are stabilizers for the condensed code. They correspond to the string operators that transport the condensed charge. For  $\mathcal{S}_{rx}$ , this is a weight-2 string operator that transports the  $rx$  charges in the parent color-code model. Indeed, we can generate longer string operators that transport these charges by taking the products of red Pauli- $X$  edge terms that are included in  $\mathcal{S}_{rx}$ .

We can check that the deconfined charges of the condensed code are identified by fusion with a condensed charge at the microscopic level (see Sec. III A). We find that logical operators that are identified by condensation are equivalent up to multiplication by edge operators. We depict this equivalence for the case of the condensed code  $\mathcal{S}_{rx}$  in Figs. 45(a) and 45(b), where we show the equivalence between the  $ry$  and  $rz$  operators, as well as equivalence between the  $gx$  and  $bx$  operators, by multiplication with the edge operators that transport the condensed  $rx$  charges. We can therefore regard string operators  $ry$  and  $rz$  equivalently as Pauli- $Z$  logical operators and  $gx$  and  $bx$  equivalently define Pauli- $X$  logical operators. Likewise, as discussed previously in Sec. III D, we can identify the deconfined red bosons with, say, an electric charge of the toric code model,  $ry \simeq rz \equiv e$ , and, similarly, the two deconfined bosons that have a Pauli- $X$  label with a magnetic flux of the toric code;  $gx \simeq bx \equiv m$ .

Let us now look at how we transform between different instantaneous stabilizer groups in dynamically condensed color codes. We will concentrate on a single transformation to explain how the logical operators are modified under a transformation process but we note that we can compile a long sequence of transformations of this type. Additionally, we will show how we detect error events from the weight-2 check measurements.

As we will see, we can transform between any condensed color codes  $\mathcal{S}_a$  and  $\mathcal{S}_b$  provided that  $a$  and  $b$  are bosons that share neither their color label nor their Pauli label. In other words, we require  $b$  to be a confined charge in  $\mathcal{S}_a$ . It is this observation that gives us a generalized construction for dynamically driven codes. As such, without loss of generality, we concentrate on one projection from the initial instantaneous code  $\mathcal{S}_{rx}$  onto  $\mathcal{S}_{gy}$ . We complete this transformation by measuring all of the Pauli- $Y$  edge operators on green edges. It will be helpful to tabulate the bosonic charges of  $\mathcal{S}_{rx}$  as follows:

	r	g	b
x	●	●	●
y	●	×	×
z	●	×	×

(18)

Measuring the green Pauli- $Y$  edge operators to condense the confined  $g_Y$  charges then maps us onto a code with the following bosonic charges:

$$\begin{array}{c|cc}
 & r & g & b \\
 \hline
 x & \times & \text{cyan} & \times \\
 \hline
 y & \text{orange} & \text{black} & \text{orange} \\
 \hline
 z & \times & \text{cyan} & \times
 \end{array} \quad (19)$$

In the former,  $\mathcal{S}_{rx}$ , the  $rx$  charges are condensed such that, up to exchange of  $e$  and  $m$  labels, we can regard the deconfined  $ry$  and  $rz$  anyons equivalently as electric charges  $e$  of the toric code ( $ry \simeq rz \equiv e$ ) and, similarly,  $gx$  and  $bx$  can both be regarded as deconfined magnetic charges  $m$  of the toric code ( $gx \simeq bx \equiv m$ ). All other charges are confined. In the latter, the  $g_Y$  anyons are condensed and we have  $gx \simeq gz \equiv m$  and  $ry \simeq by \equiv e$ . In Fig. 45, we show the microscopic details of the logical operators as the transformation is made.

Importantly, both  $\mathcal{S}_{rx}$  and  $\mathcal{S}_{gy}$  share a pair of logical operators. These are string operators for  $gx$  and  $ry$  charges. This can be read directly by comparing the two boson tables in Eqs. (18) and (19), where we use different colors to correspond to different species of toric-code bosons. One can readily check that both of these boson tables have a common pair of deconfined bosons, which correspond to these anticommuting logical operators and braid nontrivially with one another. Importantly,

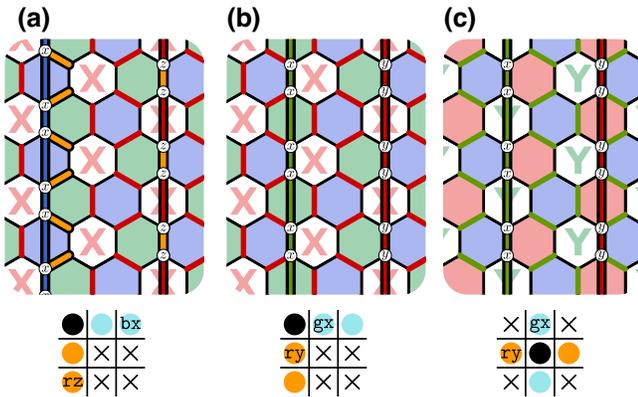


FIG. 45. Logical operators of dynamically condensed color codes. The logical operators corresponding to string operators for  $bx$  ( $rz$ ) charges [shown in (a)], are equivalent to the string operators for  $gx$  ( $ry$ ) charges [shown in (b)]. They differ by red  $XX$  edge terms, the hopping terms of the condensed  $rx$  anyon, which are stabilizer terms. We highlight the red edge checks that we use to identify this equivalence in (a). The logical operators obtained in (b) commute with all of the elements of both instantaneous stabilizer groups  $\mathcal{S}_{rx}$  and  $\mathcal{S}_{gy}$ . Therefore, these logical operators are maintained as we transform between  $\mathcal{S}_{rx}$  and  $\mathcal{S}_{gy}$ . We show the stabilizers of the  $\mathcal{S}_{gy}$  instantaneous stabilizer group together with the same logical operators corresponding to  $gx$  and  $ry$  string terms in (c).

this means that logical operators are maintained throughout the transformation and, as such, logical information is preserved. It follows that measuring the  $g_Y$  edge operators does not reveal any logical information from the corresponding stringlike logical operators of the deconfined charges. For a microscopic illustration, see Fig. 45.

Let us further emphasize at this point that the transformation determines the charge labeling convention for the new code, as this will be important in Sec. VII C 2, where we consider the transformation of boundary stabilizers for dynamically driven codes. Specifically, we find that the particle identification across the transformation can be read directly from the boson tables of the condensed color codes before and after the transformation. By identifying  $ry$  and its corresponding logical operators with, say, the electric charge  $e$  in  $\mathcal{S}_{rx}$ , it follows that the same logical operators must also transport  $e$  charges in the transformed code  $\mathcal{S}_{gy}$ . As such, the  $ry$  charge of the transformed code, together with charges that are identified with it under the condensation operation, must also be identified with the  $e$  particle. Likewise, by identifying  $gx$  with the magnetic particle  $m$  in the original code  $\mathcal{S}_{rx}$ , it follows that  $gx$ , and its corresponding string terms that give rise to logical operators, must also be identified with the  $m$  particle in the transformed code  $\mathcal{S}_{gy}$ . We reflect this identification of charges across the transformation using a consistent coloring convention for the two types of deconfined charges in Eqs. (18) and (19). Specifically, we use different colors to correspond to different species of toric-code bosons. In both boson tables in Eqs. (18) and (19),  $ry$  is colored orange in both cases, denoting an  $e$  charge, and  $gx$  is colored blue, indicating the identification of this particle with  $m$ .

Let us recall that the above argument for a specific transformation holds, in general, for any transformation in which a charge that was previously deconfined becomes condensed. If we were to measure the hopping terms of a deconfined charge, such as the red Pauli- $Y$  or Pauli- $Z$  edge terms, or the green or blue edge terms with a Pauli- $X$  label, we would perform a readout. As we have discussed in Secs. IV A and IV B, condensing a deconfined charge results in the readout of logical information. To summarize, it is essential that the ongoing condensation operation condenses a confined charge of  $\mathcal{S}_a$  in our construction for dynamically condensed color codes, in order to maintain coherent logical information encoded in the dynamically driven code.

In addition to transforming the logical operators, measuring the green Pauli- $Y$  edge terms also serves to both infer some stabilizer data and to reinitialize new stabilizers. Measurement of the green edge terms also means that certain stabilizers of  $\mathcal{S}_{rx}$  are removed from the system. We will discuss this in more detail for some specific examples in Sec. VII B but let us provide an overview of the mechanics of stabilizer readout with dynamically condensed color

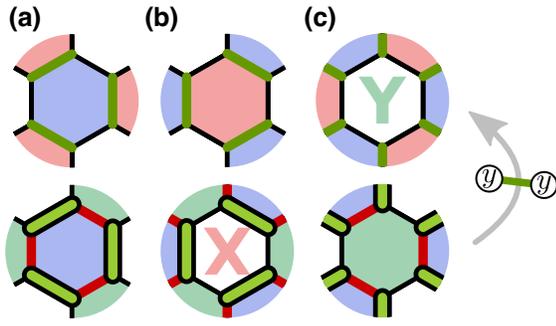


FIG. 46. An (a) blue, (b) red, and (c) green plaquette of the  $\mathcal{S}_{rx}$  and  $\mathcal{S}_{gy}$  instantaneous stabilizer groups shown at the bottom and top, respectively. The  $\mathcal{S}_{rx}$  instantaneous stabilizer group is overlaid with the green  $YY$  edge checks that are used to make the transformation. The following takes place during the transformation. The blue stabilizer (a) begins in an eigenstate of the Pauli- $Y$  stabilizer. We therefore read its known eigenstate by measuring the green  $YY$  edge checks. The red plaquette (b) is not in an eigenstate of the Pauli- $Y$  stabilizer; we therefore project the system onto a random eigenstate of this stabilizer. Both the Pauli- $X$  and Pauli- $Z$  stabilizers on the green plaquette (c) anticommute with the green  $YY$  edge checks. We therefore kick the system out of eigenstates of these stabilizers, leaving the green plaquettes in an eigenstate of the Pauli- $Y$  plaquette stabilizer only.

codes here. In what follows, we explain how the stabilizer group is transformed under a single condensation process (see also Fig. 46).

We are considering the transformation in which we project the stabilizer group  $\mathcal{S}_{rx}$  onto  $\mathcal{S}_{gy}$ . Here, we perform green edge measurements that commute with stabilizers on the red and blue plaquettes. As such, the system remains in an eigenstate of all stabilizer generators on red and blue plaquettes. We can also use the green edge measurements to infer the values of the  $S_f^{xy}$  and  $S_f^{by}$  operators. Given that we have begun in an eigenstate of  $S_f^{by}$ , we learn its value for a second time. Comparison of its new value to its original value allows us to identify errors that have occurred during the interim period [see Fig. 46(a)]. This gives rise to a detection cell, introduced in Sec. II E and discussed further in Sec. VII B 1. In contrast, the stabilizer group  $\mathcal{S}_{rx}$  did not include  $S_f^{xy}$  terms [see Fig. 46(b)]. Measuring the green Pauli- $Y$  edge terms therefore initializes the system in eigenstates of these stabilizers and in turn the  $S_f^{xz}$  stabilizer, given that we maintain an eigenstate of the  $S_f^{rx}$  stabilizer throughout the transformation. However, given that the outcome of this inferred stabilizer measurement is random, these measurements provide no new syndrome data with respect to the red plaquettes.

The transformation also removes stabilizers from the system. The green Pauli- $Y$  edge measurements anticommute with the  $S_f^{gx}$  and  $S_f^{gz}$  stabilizers. These stabilizers are therefore not included in  $\mathcal{S}_{gy}$ . Indeed, the new stabilizer

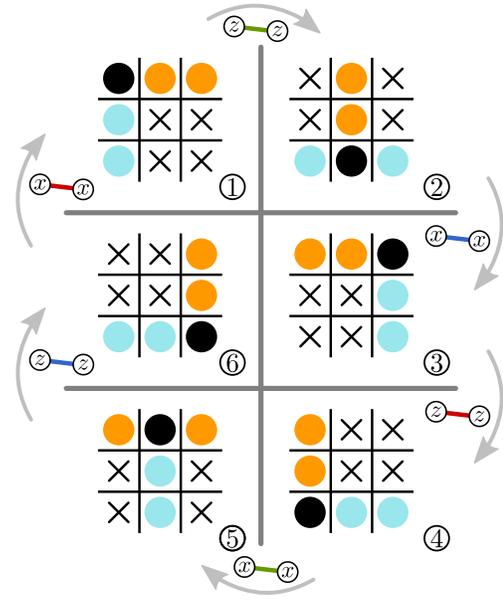


FIG. 47. The boson tables of the different condensed color codes that are produced at each stage of the Floquet-color-code measurement sequence.

group only includes  $S_f^{gy}$  on the green lattice faces. We depict this in Fig. 46(c).

## B. The Floquet color code

With the discovery that we can transform between any pair of anyon-condensed color codes from  $\mathcal{S}_a$  to  $\mathcal{S}_b$  provided that bosons  $a$  and  $b$  share neither color nor Pauli labels, we find a new degree of freedom that we can use to design new dynamically driven codes that follow different sequences of check measurements. To this end, we introduce the Floquet color code, a specific example of a dynamically condensed color code (see also Refs. [39,40], where this code has recently been introduced in independent work on a lattice with periodic boundary conditions). The Floquet color code follows a sequence of six instantaneous stabilizer groups,

$$\begin{aligned} \dots &\rightarrow \mathcal{S}_{rx} \rightarrow \mathcal{S}_{gz} \rightarrow \mathcal{S}_{bx} \rightarrow \mathcal{S}_{rz} \rightarrow \mathcal{S}_{gx} \rightarrow \mathcal{S}_{bz} \\ &\rightarrow \dots \end{aligned} \quad (20)$$

(see Fig. 47), where we use boson tables to show how the electric and magnetic charges are transformed as the Floquet color code undergoes code deformations.

Let us now describe quantum error correction using the Floquet color code before presenting the numerical results from our threshold simulation. As we have discussed, dynamically condensed color codes maintain canonical pairs of anticommuting logical operators provided that the transformations that we use respect the rules that are detailed in Sec. VII A) One can check that our sequence

respects these results, so let us concentrate on stabilizer measurements.

### 1. Detection cells

Fault-tolerant error correction requires that we identify all types of errors over time. In addition to measuring the occurrence of physical errors that act on the qubits of the system, we must also identify errors where the measurement apparatus returns the incorrect results. In Sec. VII A, we have described how we transform between different instantaneous stabilizer groups as we perform check measurements. However, we use more general objects to identify the occurrence of errors in practice. We therefore define detection cells (see also Sec. II E) that we use to identify measurement errors, as well as physical errors.

Detection cells compare the value of some Pauli check that we measure to the value of the same check that was measured at some earlier time. If the final reading deviates from the initial reading, then we declare that an error event has been detected. These events can be thought of analogously with pointlike anyonic excitations in the space-time picture. Error events can be caused by physical errors that occur between its initial and final reading or by measurement errors that change the value of either reading of the check measurement.

Let us first consider the example of a detection cell corresponding to a Pauli- $X$  stabilizer on a red plaquette [see Fig. 48(left)], although we remark that no generality is lost here, as our periodic measurement sequence is invariant under cyclic permutations of the arbitrary color labels or exchange of the Pauli- $X$  and Pauli- $Z$  labels. We initialize the red Pauli- $X$  stabilizer at the earliest time, shown at the bottom of the figure, at which we measure the green Pauli- $X$  edge checks. We compare the stabilizer measurement to the measurement of the same stabilizer, performed at the final time at the top of the figure, at which we infer its value from measuring the blue Pauli- $X$  edge checks. Assuming that no errors occur, we expect the reading of this stabilizer to be the same at both the first time and the last time.

It is important that all the check measurements that are made in the interim period between the initial and final measurements of the detection cell commute with the red Pauli- $X$  stabilizer. This will mean that both measurements of the stabilizer will have the same value, provided that no errors occur. In between the initial and final readout of the red Pauli- $X$  stabilizer, we measure blue and green Pauli- $Z$  edge checks and red Pauli- $X$  edge checks. One can readily verify that all of these check measurements commute with the red Pauli- $X$  stabilizer. As such, we obtain a detection cell that identifies errors that occur in between the initial and final readout of the stabilizer of interest, as well as any measurement errors that may occur during either reading of the stabilizer.

Let us now look at how all of the detection cells are supported. We find that all qubits support four detection

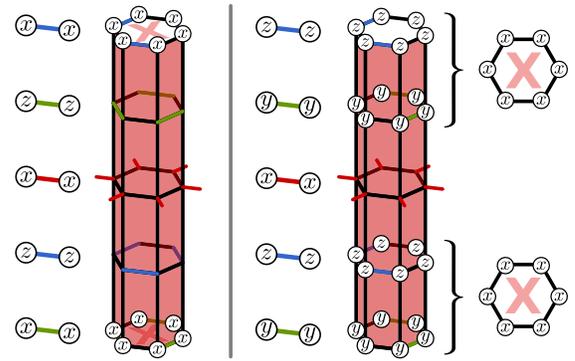


FIG. 48. Detection cells for the Floquet color code (left) and the honeycomb code (right). For the Floquet color code, we measure the red Pauli- $X$  stabilizer at the initial and final times in the figure. At the initial time, we infer its value by measuring the green Pauli- $X$  edge checks and we compare its value to the reading of the same stabilizer at the final time in the figure, where we infer its value from measuring the blue Pauli- $X$  edge checks. One can check that all of the check measurements that are made in between the two readings of the red Pauli- $X$  stabilizer commute with the stabilizer of interest. As such, we obtain a detection cell by comparing the first and last readings of the stabilizer. In the case of the honeycomb code, we infer the value of the red Pauli- $X$  stabilizer by taking the product of all of the check measurements during the first two time steps, both the green Pauli- $Y$  checks and the blue Pauli- $Z$  checks, and we compare the value of this stabilizer to the value inferred from the green and blue check measurements that are collected at the final two time steps. The Pauli- $X$  stabilizer commutes with the red Pauli- $X$  edge checks that are performed in between the two readings of the stabilizer.

cells at any given instant; two Pauli- $X$ -type cells and two Pauli- $Z$ -type cells. In Fig. 49, we show red, green, and blue detection cells in space-time, with the Pauli- $X$ -type cells on the left of the figure and the Pauli- $Z$ -type cells on the right. Note how the left and right diagrams are equivalent up to conjugation by a Hadamard and a shift by three time steps. Let us thus, without loss of generality, concentrate on the Pauli- $X$  detection cells. As we have already discussed, the red Pauli- $X$  detection cells are initialized when we measure the green Pauli- $X$  edge checks and are read out a second time when we measure the blue Pauli- $X$  edge checks. Similarly, the detection cells corresponding to green (blue) stabilizers are initialized when we measure the blue (red) edge checks and they are read out at the final instant at which we measure the red (green) edge checks. By construction, every qubit supports one stabilizer of each color. We can therefore see in the diagram that at any given instant, every qubit supports two Pauli- $X$ -type detection cells.

An unusual feature of the Floquet color code is that we do not maintain a constant group of stabilizers. Rather, stabilizers are constantly reinitialized and later checked to obtain detection cells that identify error events. For example, the red Pauli- $X$  stabilizers do not commute with

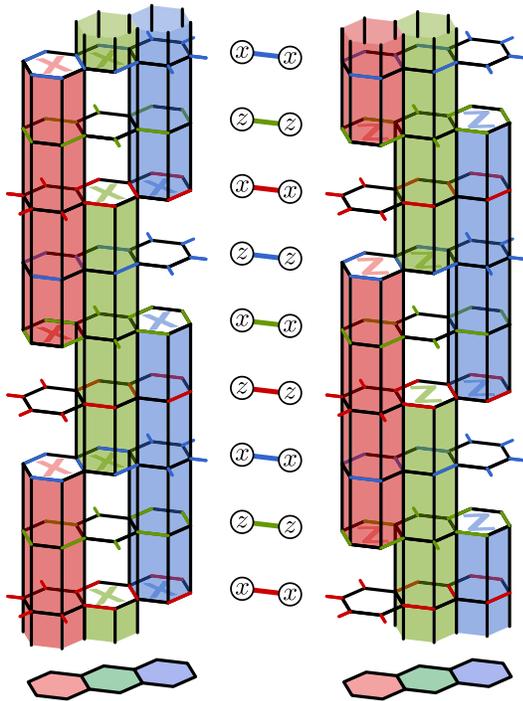


FIG. 49. The space-time diagram showing detection cells corresponding to Pauli- $X$ -type stabilizers and Pauli- $Z$ -type stabilizers shown, respectively, on the left and right. Time runs upward, and is marked by the points in time at which the edge measurements are made. We show the edge measurements in the center of the figure. Red Pauli- $X$ -type (Pauli- $Z$ -type) cells are initialized when the green Pauli- $X$  (Pauli- $Z$ ) edge checks are measured and they are read out a second time when we measure the blue Pauli- $X$  (Pauli- $Z$ ) edge checks, thereby completing the detection cell. Likewise, green (blue) detection cells are initialized when we measure the blue (red) edge checks and are read out a second time when we measure the red (green) edge checks. As the stabilizers of the Floquet color code do not commute with all the edge checks, we observe short intervals in which a qubit does not support some given cell. For example, the red Pauli- $X$ -type detection cell is not supported over the interval in which the Pauli- $Z$  edge checks are measured. As such, we find a temporal gap between detection cells of the same type.

red Pauli- $Z$  edge-check measurements; we cannot maintain a red Pauli- $X$  detection cell as we measure Pauli- $Z$  edge checks. We must therefore reinitialize the red Pauli- $X$  stabilizer after we measure the red Pauli- $Z$  edge checks. Likewise, we do not maintain a green (blue) Pauli- $X$  detection cell over the interval in which we measure green (blue) Pauli- $Z$  edge checks. This is represented in the figure by the temporal gap between two detection cells. Nevertheless, we see that every qubit always supports two detection cells at any given time.

## 2. The error syndrome of the Floquet color code

Let us now look at how the detection cells respond to errors. As the Floquet color code is a CSS code, we

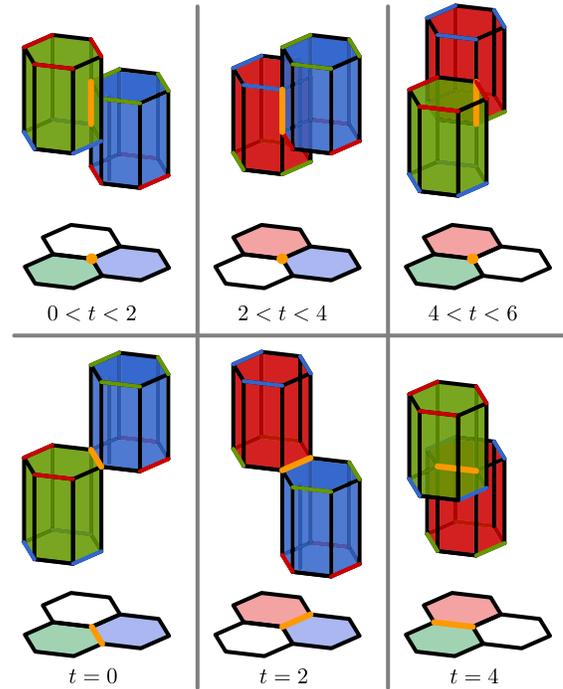


FIG. 50. The error syndrome of physical errors (top row) and measurement errors (bottom row). Let us consider the physical error shown in the top left: a qubit supports a green and a blue Pauli- $Z$ -type detection cell after the blue Pauli- $Z$  edge checks are measured, up until the instant at which the following green Pauli- $Z$  edge checks are measured. Therefore, a physical error that occurs on a qubit during the interval  $0 < t < 2$  will produce two detection events on the green and the blue detection cell that it supports. To understand how measurement errors manifest, let us focus on the case of a measurement error on a red edge (bottom left): the red edge checks read out the green detection cell and simultaneously reinitialize the stabilizer used for a blue detection cell. Therefore, a measurement error on a red edge check will violate a green and a blue detection cell.

concentrate only on bit-flip errors but remark that an equivalent discussion will hold for Pauli- $Z$ -type errors acting on Pauli- $X$ -type detection cells. Pauli- $Y$  errors can be regarded as the product of a Pauli- $X$ -type and Pauli- $Z$ -type error. In Fig. 50, we show the occurrence of physical errors at different time intervals over a period, as well as a measurement error on different types of edge check. We will concentrate our discussion on errors that create a pair of detection events on red and blue detection cells but remark that an equivalent discussion will hold for any pair of colors, up to a cyclic permutation of color labels. We show examples of all types of errors in Fig. 50.

Let us first look at a bit-flip error. A qubit supports a red and a blue Pauli- $Z$ -type detection cell after the instant at which the red Pauli- $Z$ -type checks are measured, up to the instant at which the following blue Pauli- $Z$ -type checks are measured. A bit-flip error that occurs in this interval

will therefore violate the two detection cells that the qubit is supporting at this time, thereby identifying a pair of error events. We show the two violated detection cells in Fig. 50 (top middle).

Measurement errors also create a pair of detection events in the error syndrome. Each edge check that is measured performs two tasks; it contributes to the read out of a stabilizer, thereby completing a detection cell, and it is also used to reinitialize a stabilizer to produce a new detection cell. A measurement error on a given edge check will create detection events on both of its associated detection cells, as shown in Fig. 50 (bottom middle). The figure shows a single measurement error on a green edge check that reads out a blue detection cell and reinitializes a red detection cell.

Given that all types of errors, both bit flips on physical qubits and measurement errors, create detection events in pairs in the space-time bulk, we have a conservation law among error-detection events [82,92] that enables us to employ standard decoding methods such as minimum-weight perfect matching [1,58,82] or union find [48]. In what follows, we describe simulations to evaluate the threshold using a minimum-weight perfect-matching decoder implemented using the PyMatching software program [93].

### 3. Numerical simulations for the Floquet color code and comparison with the honeycomb code

To assess its performance, we simulate fault-tolerant quantum error correction with the Floquet color code under circuit-level noise (for details, see the caption of Fig. 51). For the standard depolarizing circuit-noise model used in Ref. [38], we obtain a threshold of approximately 0.3% using a matching decoder. The decoder matches detection events from the Pauli- $Z$ -type detection cells independent of the syndrome information that is measured by the Pauli- $X$ -type detection cells. To contrast with other work, Ref. [38] reports a threshold of 0.2–0.3% with the honeycomb code under the same noise model using a minimum-weight perfect-matching decoder that accounts for correlations.

Using a basic decoding algorithm, we have shown that the Floquet color code has a threshold that is competitive with the honeycomb code. This may be surprising, given that we are simulating a code that does not have a constant stabilizer group. Rather, local stabilizers are constantly being removed from the system and reinitialized with different transformations between instantaneous stabilizer groups. In spite of this, the Floquet color code obtains syndrome data at the same rate as the honeycomb code.

In what follows, we discuss three factors to explain the similar thresholds obtained for the honeycomb code and the Floquet color code. These factors are:

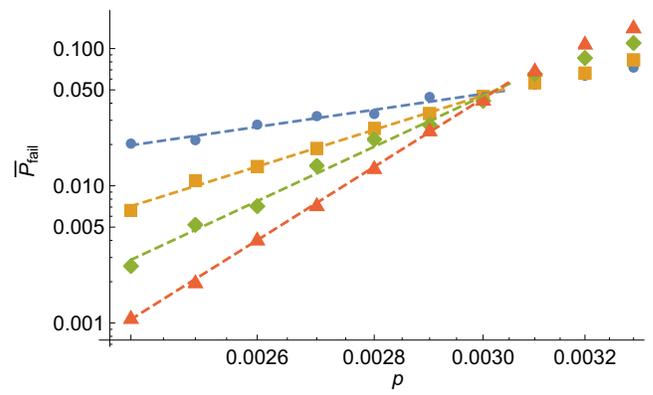


FIG. 51. The threshold plot for the Floquet color code with periodic boundary conditions using the standard depolarizing noise model [38]. Data for codes of  $L \times L$  red hexagons with  $L = 4, 8, 12,$  and  $16$  are shown in blue, yellow, green, and red, respectively. We obtain a threshold of approximately 0.3%, which is approximately the same as that obtained with the honeycomb code for the equivalent noise model [38]. The simulations have been conducted using the STIM software program [59] and decoding has been performed using a minimum-weight perfect-matching decoder, implemented using PyMatching [93], that has concentrated only on stabilizer measurements obtained from Pauli- $Z$  edge checks. Further improvements might be obtained by exploiting correlations between the detection events that are measured by the Pauli- $X$ -type and Pauli- $Z$ -type stabilizers under the circuit-level depolarizing noise model.

- (i) the rate at which detection cells are evaluated
- (ii) the stabilizer operators of the two Floquet codes
- (iii) similar syndrome data structures for decoding

To review, up to a local change of basis, two periods of the honeycomb code check measurements are as follows:

$$\begin{aligned} \dots \rightarrow \mathcal{S}_{rx} \rightarrow \mathcal{S}_{gy} \rightarrow \mathcal{S}_{bz} \rightarrow \mathcal{S}_{rx} \rightarrow \mathcal{S}_{gy} \rightarrow \mathcal{S}_{bz} \\ \rightarrow \dots \end{aligned} \quad (21)$$

We also show a corresponding detection cell for the honeycomb code in Fig. 48(right).

Let us summarize the difference in how detection cells are measured for each of the Floquet codes in regard to point (i).

The Floquet color code measures two distinct stabilizers per plaquette, whereas the honeycomb code only measures one stabilizer per plaquette. On the other hand, the honeycomb code obtains new detection cells for each of its corresponding stabilizers at double the rate of each of the stabilizers for the Floquet color code. Overall, detection events are measured at an equivalent rate for each Floquet code.

In regard to point (ii), both codes have very similar structures in the sense that detection cells correspond to weight-6 stabilizers that are obtained over five rounds of weight-2 check measurements. Also, given that both codes

produce syndrome data at the same rate, as discussed in point (i), it is perhaps unsurprising that the Floquet color code and the honeycomb code demonstrate comparable threshold error rates.

Let us finally discuss decoding using the minimum-weight perfect-matching decoder for the two Floquet codes [point (iii)]. To summarize the remaining discussion in this section, we find that both decoders have similar decoding graphs in the sense that errors give rise to pairs of events at detection cells in very similar configurations in space-time. We can exploit this structure for both codes to design a minimum-weight perfect-matching algorithm.

Let us discuss how we can obtain a matching decoder for the honeycomb code. We find that the honeycomb code demonstrates a parity-conservation law among its detection events [82,92] over two subsets of its detection cells. We obtain these two conservation laws by dividing the cells according to the time at which the cell is initialized. One subset includes all the detection cells that start by check measurements made at odd time steps, while the other subset begin their initialization by checks made at even time steps. We leave it to the reader to verify this fact. Nevertheless, the detection cells of one of the aforementioned symmetries for the honeycomb code share the same structure as those of, say, the Pauli- $X$ -type detection cells of the Floquet color code (see Fig. 49). Given a detection-event parity symmetry, one can exploit its corresponding conservation law to design a matching decoder [82,92].

Likewise, we obtain two subsets of detection-event parity conservation laws with the Floquet color code. This has already been mentioned implicitly in Sec. VII B 2. Using that the code is a CSS code, we can trivially separate the decoding problem for the Pauli- $X$ -type detection cells and the Pauli- $Z$ -type detection cells. Indeed, as we have discussed in detail, errors give rise to defects in pairs in the Floquet color code if we subdivide the results from the detection cells in this way. Given that the Pauli- $X$ -type detection cells are initialized at odd time steps and Pauli- $Z$ -type detection cells are initialized at even time steps, we can readily see an equivalence in the structure of the decoding graph for the two different Floquet codes.

### C. The boundaries of Floquet codes

Dynamically driven codes with a local group of stabilizer generators can be realized on a planar array of qubits by encoding logical information on a lattice with boundaries [36,37]. In order to design a dynamically driven code with boundaries, we must also find suitable transformations between the boundary-stabilizer operators as we perform deformations between instantaneous stabilizer groups. Here, we appeal to the physics of the underlying parent phase to find a systematic way of obtaining

suitable boundary transformations for examples of dynamically driven codes. In what follows, we will show how the boundaries of dynamically condensed color codes can be derived using the structure of the parent anyon theory of the color-code model. We will go on to demonstrate our general theory by producing microscopic boundary conditions for the Floquet color code.

#### 1. Boundaries of condensed color codes

The toric-code phase that is realized by dynamically condensed color codes has a well-understood boundary theory [61]. We encode logical qubits by introducing “rough” and “smooth” boundaries to the lattice [1, 94], where a rough boundary condenses electric charges, labeled  $e$ , and a smooth boundary condenses magnetic charges, labeled  $m$ . We must therefore look for boundaries for dynamically driven codes, together with their associated transformations, that reproduce the behavior of the rough and smooth boundaries of the toric code. We find that we can derive the boundaries of dynamically condensed color codes from the parent color-code model. Let us review the boundaries of the color code before discussing how the boundaries of dynamically driven codes are obtained from the parent theory. We pay particular attention to the color-code boson table to help to elucidate our construction.

We describe six different boundaries for the color code in Sec. III B 1. Three boundaries are associated with color labels of the color-code bosons and three boundaries are associated with Pauli labels. We call these color boundaries and Pauli boundaries, respectively. The boundary label denotes the types of boson that the boundary condenses.

We recall that the color boundaries correspond to the columns of the boson table of the color code and the Pauli boundaries correspond to the rows of the boson table. Indeed, the table is defined such that all the bosons in a column share a common color label and all of the bosons in a row share a common Pauli label. Specifically, the boson table is designed such that the rows and columns represent distinct Lagrangian subgroups of the color-code anyon model (see Sec. III B 1).

Let us look once again at the charges of the boson table that remain deconfined as we condense a single charge. Upon condensing a single color-code boson, up to a symmetry in exchange of the  $e$  and  $m$  labels, we identify the charges that share a row with the condensed charge with the electric charges of the toric code and we identify the charges that share a column with the condensed charge with the magnetic charges of the toric code. Correspondingly, we require suitable boundaries to condense these charge types at appropriate locations.

We posit that we obtain appropriate rough and smooth boundary terms that, respectively, absorb  $e$  and  $m$  charges

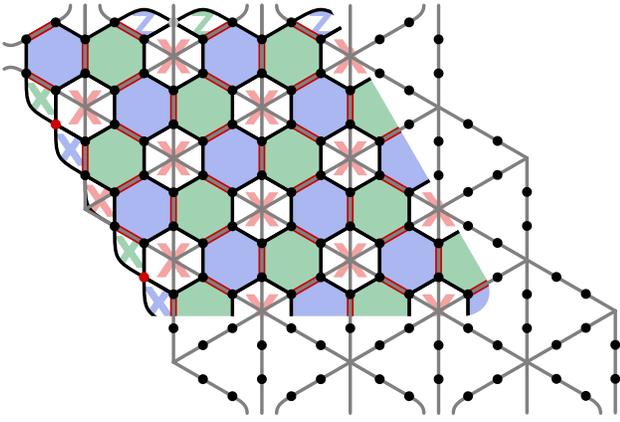


FIG. 52. The boundaries of the Floquet color code at any point in time can be related to smooth and rough boundaries of the corresponding toric code. We demonstrate this here by drawing the toric code in gray on top of the Floquet color code with  $\mathcal{S}_{rx}$  being condensed.

for dynamically condensed color codes by choosing appropriate boundaries in the corresponding color-code theory. Without loss of generality, let us take the color code where the red Pauli- $X$  charge is condensed,  $\mathcal{S}_{rx}$  as an example. In this condensed color code, we have that the deconfined electric charges of the condensed theory correspond to the  $ry$  and  $rz$  charges of the parent theory. Both of these charges have a common red color label and are therefore both absorbed, uniquely, by a red color boundary. As such, red color boundaries of a color code become rough boundaries that absorb electric charges of the condensed theory  $\mathcal{S}_{rx}$ . Similarly, the magnetic fluxes of the condensed theory correspond to the  $gx$  and  $bx$  charges of the parent theory. Both of these charges are uniquely absorbed by a Pauli- $X$  boundary, as they share an  $X$  Pauli label. We therefore find that the Pauli- $X$  boundaries of a color code become the smooth boundaries that absorb the magnetic charges of  $\mathcal{S}_{rx}$ . Indeed, in Fig. 52, we show a condensed color code  $\mathcal{S}_{rx}$  where the parent theory had two distinct red boundaries and two distinct Pauli boundaries. The figure shows the corresponding toric-code lattice overlaid, with qubits on the edges. We observe that the Pauli- $X$  boundaries produce smooth boundaries in the condensed theory, whereas the red boundaries produce rough boundaries.

## 2. Boundary transformations

We have argued that the lattice geometry at the boundary of the instantaneous stabilizer groups of dynamically condensed color codes can be determined by the corresponding excitations that are absorbed at the boundary for the parent color-code anyon theory. We can go on and follow our rule to its conclusion to learn how the boundaries must transform as we perform deformations between different instantaneous stabilizer groups of dynamically

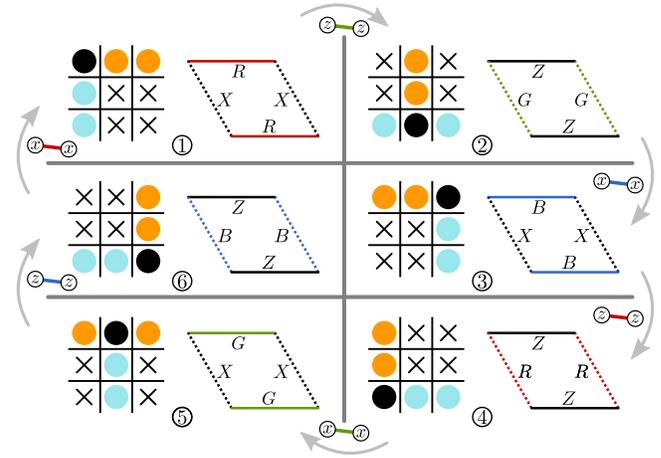


FIG. 53. The boundaries of the Floquet color code as it changes in time. The rough boundaries are always located on the left and right boundaries of the code and the smooth boundaries on the top and bottom boundaries. The corresponding boundary of the parent color code, however, changes over time, as indicated by the capital letters, where  $R$ ,  $G$ , and  $B$  are the red-, green-, and blue-colored boundaries, and  $X$  and  $Z$  denote a parent color-code theory with an  $X$ -type and  $Z$ -type Pauli boundary, respectively.

condensed color codes (see also Ref. [37]). As an explicit example, in Fig. 53, we show the boundary transformations through a single period of the Floquet color code in terms of the boundaries of its corresponding parent theory together with their corresponding boson tables. We explain this choice of boundary transformations throughout this section.

Without loss of generality, we will follow the details of the transformation between the first and second step of the period of the Floquet color code as an example, where we transform from  $\mathcal{S}_{rx}$  onto  $\mathcal{S}_{gz}$ —see steps 1 and 2 in Fig. 53—but we note that the following discussion will hold between any pair of condensed color codes,  $\mathcal{S}_a$  and  $\mathcal{S}_b$ , with bosons  $a$  and  $b$  condensed, provided that  $a$  and  $b$  share neither a color label nor a Pauli label.

Let us posit again that rough boundaries must remain rough after a Floquet-code transformation and likewise a smooth boundary must remain smooth. More specifically, this means that the rough boundaries must condense the bosons of the parent theory that are identified with electric charges both before and after the transformation and, likewise, the smooth boundaries must condense the magnetic charges throughout the transformation. We must therefore follow how the electric and magnetic charges are maintained throughout the transformation at the level of the parent color-code anyon theory (see Sec. VII A). To restate our result briefly, if we choose a convention for  $\mathcal{S}_{rx}$  where we have identified red parent bosons with electric charges and Pauli- $X$  parent bosons with magnetic charges, then it follows that, after the transformation from  $\mathcal{S}_{rx}$  onto  $\mathcal{S}_{gz}$ ,

Pauli-Z parent bosons correspond to electric charges and green parent bosons correspond to magnetic charges. This convention is laid out explicitly in the boson tables shown in Fig. 53.

We find that constraining the electric and magnetic charge labels as we transform  $\mathcal{S}_{rx}$  onto  $\mathcal{S}_{gz}$  also constrains the boundary transformation. With the convention used above, we have that the red (Pauli-Z) parent bosons of  $\mathcal{S}_{rx}$  ( $\mathcal{S}_{gz}$ ) correspond to electric charges and that the Pauli-X (green) parent bosons of  $\mathcal{S}_{rx}$  ( $\mathcal{S}_{gz}$ ) correspond to magnetic charges. This fixes the boundaries of the code. We specifically have that the red (Pauli-Z) boundaries correspond to the rough boundaries of  $\mathcal{S}_{rx}$  ( $\mathcal{S}_{gz}$ ) and that the Pauli-X (green) boundaries correspond to the smooth boundaries of  $\mathcal{S}_{rx}$  ( $\mathcal{S}_{gz}$ ). It follows that, at the level of the parent theory, red boundaries must transform onto Pauli-Z boundaries as we transform from  $\mathcal{S}_{rx}$  onto  $\mathcal{S}_{gz}$  and, likewise, Pauli-X boundaries must transform onto green boundaries in the parent theory under the transformation of interest. This is the result shown in Fig. 53. In Sec. VII C 3, we give microscopic details demonstrating the boundary transformation that we have described here at a macroscopic level.

To conclude here, let us summarize the prescription that we have given to determine boundary transformations for dynamically condensed color codes. As discussed in Ref. VII A, the electric and magnetic charge labels are constrained throughout a transformation. The transformation can be displayed clearly in boson tables once a charge-label convention is specified at the initial step. Furthermore, boundary-stabilizer terms can be obtained at the level of the parent theory by reading the respective rows and columns of the boson table that support the deconfined charges of the condensed anyon theory (see Sec. VII C 1). Finally, we have argued that constraints among the charge labels under the transformation, together with a rule for finding check operators from the parent theory at the boundary of an instantaneous code, can determine how we transform the boundaries of dynamically condensed color codes.

### 3. Microscopic details at the boundary of the Floquet color code

Having given an overview of how the boundaries of the dynamically condensed color codes transform, we can also present the microscopic details for how the boundaries transform throughout a period of the Floquet color code. We show these details in Fig. 54. In the figure, we concentrate on a smooth boundary that oscillates between color labels and Pauli-X labels. The case of a rough boundary transformation is obtained by replacing Pauli-X and Pauli-Z labels. Details are given in the figure caption.

We also show the detection cells at the boundary of the Floquet color code in Fig. 55. Specifically, we show a boundary-detection cell corresponding to a Pauli-Z

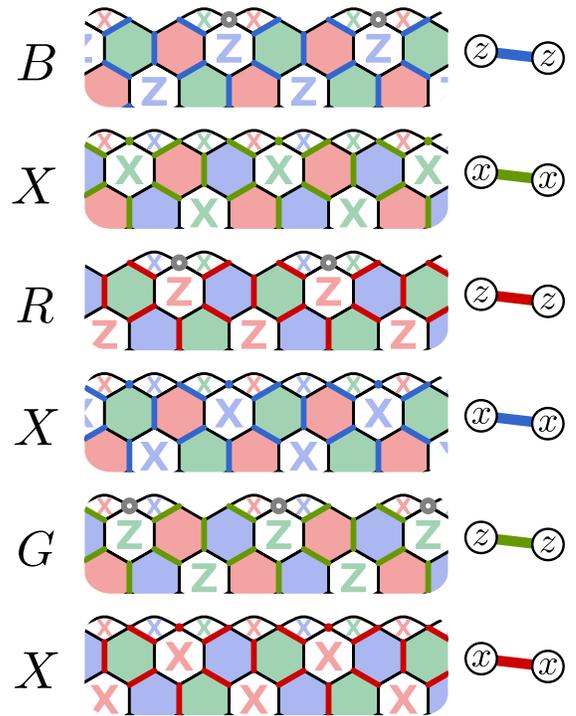


FIG. 54. Microscopic realizations of the smooth boundaries of the Floquet color code as the code transforms between different instantaneous stabilizer groups over a single period of the code. The letters on the left indicate the boundary of the parent color-code phase. We measure weight-2 edge checks at each round, where the measurements are given by the key illustrated to the right of the figure. We also perform single-qubit measurements at certain time steps, shown by colored spots. Furthermore, certain qubits are not measured at a given step. These are marked by gray circles.

stabilizer (left), a weight-6 boundary Pauli-X stabilizer (middle), and a weight-3 Pauli-X stabilizer (right). In each figure, one can check that all of the intermediate measurements in the schedule commute with the initial and final inferences of the stabilizer at the first and last time steps. One can also check that the detectors at the boundary are consistent with the behavior of detectors at their respective rough and smooth boundaries, as we expect. We can check this by going through an analysis equivalent to the one we have used in Sec. VII B 2.

We show a period of the Floquet color code with boundaries in Fig. 56. We also show the support of its logical operators. One may worry that a Floquet code may drift over an array of qubits as transformations are performed. Using the lattice geometry and the measurement pattern that we have specified, we find that the footprint of our code remains static.

### D. Remarks on Floquet codes

We have presented a generalized construction for dynamically driven codes called dynamically condensed

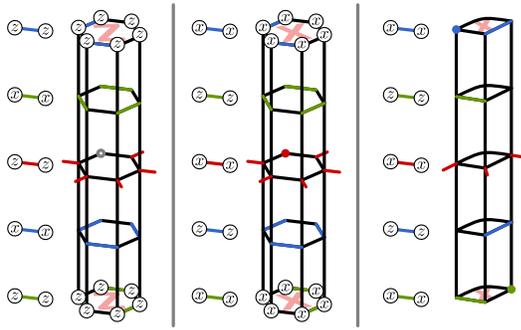


FIG. 55. Detection cells at the smooth boundary of the Floquet color code. These measurements correspond to a single period with corresponding stabilizers as discussed in Fig. 54. We show detection cells for a red Pauli-Z stabilizer (left) and a red Pauli-X stabilizer (middle), as well as the detection cell for a weight-3 Pauli-X stabilizer (right).

color codes. Our construction contains the honeycomb code. Moreover, it has enabled us to present a new dynamically driven code that we call the Floquet color code. We have also given a general boundary theory that allows us to write down a valid stabilizer group at the code boundary. Let us now discuss various aspects of dynamically driven codes that may suggest new research directions.

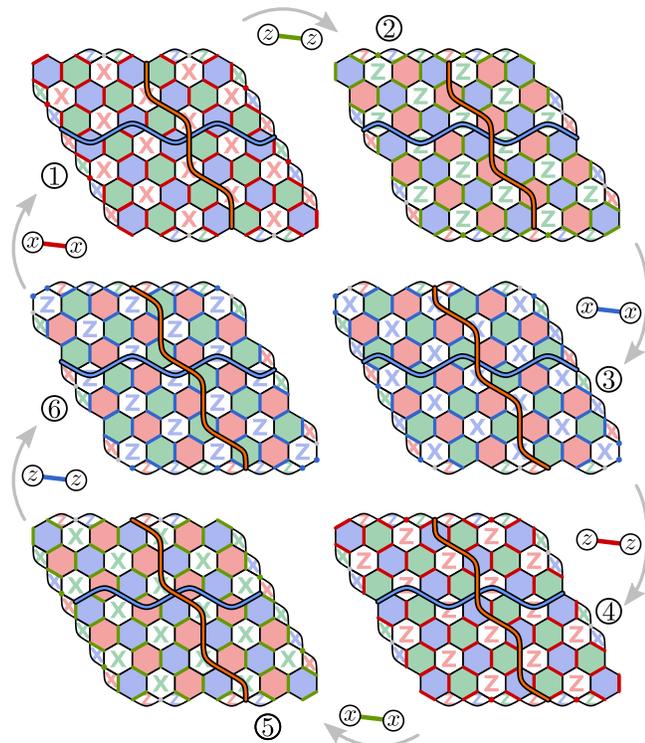


FIG. 56. The Floquet color code as it evolves throughout the six steps of the measurement scheme. We show the logical  $X$  ( $Z$ ) operator in cyan (orange).

### 1. Topological phases for Floquet codes

From the perspective of topological order, the Floquet color code presented in this work and its close cousins such as the honeycomb code are in the toric-code phase at any given time step. However, from one time step to the next, its local representation, given by the instantaneous stabilizer group, changes. In a space-time picture, this corresponds to an invertible domain wall within the toric-code phase appearing periodically in time.

The dynamically condensed color code presented here inherits features from the well-studied parent phase. In our construction, we benefit from exploiting details of the color code, such as its symmetries and boundaries [21,23]. For example, we have used the color-code boundaries to construct boundaries for dynamically driven codes systematically. Naturally, we inherit beneficial properties of the parent stabilizer group, such as the geometric locality and the bounded weight of the stabilizer checks.

Continuing in this spirit, in future work it will be interesting to show how we can produce other types of topological defects in dynamically driven codes, such as twist defects, that can be used to perform fault-tolerant gates. In addition to finding how they manifest in the microscopic details of an instantaneous stabilizer group, it is also important to find how they transform as they undergo a full period of check measurements for a Floquet code. We may discover an intuitive way of designing these objects by appealing to the details of the parent color-code theory.

On the other hand, we note that not all sequences of condensation lead to a valid code. In certain sequences, the measured two-body checks do not let us infer enough syndrome information to perform error correction. It will be interesting to understand the conditions on the validity of a measurement sequence in terms of the macroscopic parent anyon theory and its microscopic stabilizer realization.

In recent work (see Ref. [74]), a general construction of automorphism codes based on string-net models has been presented. The authors use an automorphism of a topological order and the associated domain wall in the microscopic description to define a Floquet code that implements the chosen automorphism in the time direction over a cycle of measurements. Given a phase and a representation of an automorphism, their construction gives rise to a unique Floquet code.

Our construction offers a different perspective on dynamically driven codes. At any time step, we condense a different boson in a parent theory to describe the instantaneous stabilizer group. This allows us to construct more measurement sequences to drive codes in the condensed phase. We leave the connection of dynamically condensed to automorphism codes from Ref. [74] for future work. We expect that the automorphism formalism has to be extended to more general (invertible) bimodule categories in order to capture dynamically condensed codes.

The dynamically condensed color codes that we have studied provide us with an intuitive model that shows us that we can design dynamically driven codes by condensing bosons of a parent color-code theory. Taking the perspective of a parent anyon theory with a nontrivial condensate may illuminate a way for us to design a much broader landscape of dynamically condensed codes. A first step would be to find other topological stabilizer codes corresponding to richer topological phases with nontrivial condensates and to perform a study similar to the one we have conducted for the color code. Going beyond stabilizer codes, we could consider taking a non-Abelian parent phase with a nontrivial condensate to construct a *non-Abelian dynamically driven code*. One possibility to enforce nontrivial condensates is by doubling an anyon theory. One could take some number of copies of an anyon model with a gappable boundary and dynamically condense bosons in there.

The concept of condensing topological excitations can also be applied to higher-dimensional models. In Ref. [95], a network of (partly) condensing defects within a  $(3 + 1)$ -dimensional topological theory has given rise to various fracton models. The use of similar defects interleaved (periodically) in time can define dynamical condensation in three dimensions. Moreover, it would be interesting to see how known 3D subsystem codes, such as the gauge color code [96], relate to the concept of topological condensation. Gauge fixing could be related to picking a particular description of a condensate. Changing the gauge would then correspond to an invertible domain wall within the condensate. It may be valuable to find a unifying theory between gauge fixing and dynamically driven code-measurement cycles, as such a theory may provide us with new ways of performing fault-tolerant logic gates with other types of topological codes.

## 2. A unifying framework for quantum error-correcting codes

Dynamically driven codes have provided us with a new way to read out syndrome data from topological phases using a certain sequence of check measurements. The fact that we can obtain syndrome data using only weight-2 measurements is particularly appealing from a practical perspective. A generalization of this idea may lead us to ways of modifying more general quantum codes with a constant encoding rate in the number of physical qubits [97]. As such, it is valuable to find a unifying picture to describe Floquet codes together with other types of codes evenhandedly. We might expect any such framework to include different types of circuits that read out stabilizer operators. Here, let us discuss other syndrome-readout circuits, and check measurement sequences for other subsystem codes. We do this with the aim of identifying

some similarities between Floquet codes and subsystem codes, to attempt to demystify the physics of these new models.

It is a challenging exercise to point to the differences between dynamically driven codes and more conventional codes. It is argued that choice of sequence is essential for a dynamically driven code to maintain logical information. However, in the case of subsystem codes, given that we have a large set of noncommuting measurements that must be checked to obtain a complete set of syndrome data, a suitable sequence must also be chosen, even if the sequence is found trivially. For instance, for CSS subsystem codes, one typically assumes a sequence in which we measure all of the Pauli- $X$  checks simultaneously, followed by the Pauli- $Z$  checks. Other subsystem codes require a nontrivial sequence of check measurements to read out stabilizers [98]. On the other hand, one can easily conceive of poor choices of readout sequences where stabilizers are read out inefficiently. Furthermore, at each step in the sequence, the subsystem code is projected onto a new instantaneous stabilizer group, where certain terms of the gauge group become fixed and join the stabilizer group of the code, while other terms of the gauge group are kicked out of the fixed subspace. The authors of Ref. [99] even consider changing the sequence of check measurements to improve a subsystem code to correct biased noise.

The key innovation that the examples of dynamically driven codes show us, beyond subsystem codes, is that the logical operators need not remain constant throughout a sequence of check measurements but, rather, they can be steadily deformed throughout a sequence of check measurements. As we have shown here, it is not even necessary to maintain a static stabilizer group as we undergo code transformations. Hopefully, these innovations will lead us to discover other new codes that are practical for experimental realization.

Another way in which we might consider unifying Floquet codes with other codes is at the level of syndrome-extraction circuits. Fault-tolerant syndrome-readout circuits have been proposed [100,101], where a code is teleported onto a second auxiliary system such that the teleportation operation also reveals syndrome data. One could view this teleportation operation as a code deformation, where the stabilizers of a code are teleported onto a new set of qubits with new support. One could imagine a strategy for syndrome readout where a code is periodically transferred between two subsystems. We might regard this strategy for syndrome readout as a dynamically driven code. A similar approach for syndrome readout has also been suggested using measurement-based quantum computation, where a 3D cluster state is prepared over time using a 2D qubit array [10]. We discuss this point of view from the perspective of fault-tolerant logic gates in Sec. VIID 3.

### 3. Fault-tolerant logic gates

The realization of scalable quantum computation with dynamically driven codes will require the development of fault-tolerant logic gates. Given that the dynamically condensed color codes realize the toric-code phase, we might expect to be able to perform Clifford gates with dynamically driven codes using a generalization of code deformation. These operations might include braiding punctures [9] or braiding twist defects [15,64]. The ongoing development of quantum computation with dynamically driven codes will require a more general theory to implement the microscopic details of the measurements to realize fault-tolerant logic gates. As we have already discussed in Sec. VII D 1, we may find such constructions by appealing to the physics of the parent theory of a dynamically driven code that is created by condensation. The work of Hastings and Haah [37] has already considered some lattice-surgery operations [14] with Floquet codes. If we additionally show how to initialize Floquet codes in a magic state, we can realize universal quantum computation using magic state distillation, assuming that we can perform the Clifford gates.

The development of Floquet codes may also give us new ways of realizing fault-tolerant non-Clifford gates. If one subscribes to measurement-based fault-tolerant quantum computation as an example of a Floquet-code encoding [10], then we can regard the gate constructions given in Refs. [16,17] as Floquet codes that realize non-Clifford operations on their dynamically changing code space. Indeed, in both of these cases, a 2D qubit array is driven between different codes using measurements. Additionally, a just-in-time decoder is used to reproduce the physics of a 3D code to perform non-Clifford operations in  $(2 + 1)$ D space. It will be interesting to determine to what extent we need to periodically drive these 2D systems between different stabilizer codes to realize non-Clifford gates. These examples of 2D non-Clifford gates are well understood by considering the error-correction system in a static 3D space-time. It may be an instructive exercise to develop a space-time theory for dynamically driven codes.

### 4. General noise models

For more practical implementations of fault-tolerant quantum computing, we require robust codes that demonstrate a high threshold against the noise models that real qubits experience. It has been shown that the threshold is highly sensitive to local modifications to codes undergoing noise models with biases toward particular types of error [102–106]. In addition to changes to the local bases of the physical qubits of the code, our general construction for designing dynamically condensed color codes also offers us other ways of designing dynamically driven codes. We might, for instance, choose different condensation paths through the color-code boson table. It may be that certain

paths are better suited to obtain high thresholds for different choices of biased noise models. Topological codes have also been optimized into rectangular configurations with different height and width in order to reduce overhead in the biased noise setting [103,107]. One can observe that there are no logical Pauli- $X$  operators composed of physical Pauli- $Z$  terms at any instance of the Floquet color code (see Fig. 56). This code may therefore serve as a candidate for a thin Floquet code under noise biased to introduce dephasing errors to data qubits.

A key difference between dynamically driven codes and more conventional static codes is the action of measurement errors. For stabilizer codes, a high-performance fault-tolerant syndrome-readout circuit can be found by studying an idealized situation in which measurements are error free [103,108]. Then, the decoding strategy does not differ significantly when we extend to the case where measurements are noisy, up to the dimensionality of the decoding problem. In contrast, the evaluation of detection cells for dynamically driven codes depends on a large number of measurements. Moreover, a single measurement error may affect a large number of detection cells. For instance, a measurement error in the honeycomb code will trigger four detection events. In the physically motivated limit that readout errors are very common, it may not be straightforward to determine the performance of a code tailored to correct for biased noise acting on data qubits by only considering the logical operators of the instantaneous stabilizer groups of a dynamically driven code. Given the nontrivial role of measurements in dynamically driven codes, we might expect that a high rate of measurement error may significantly compromise the performance of a tailored error-correction strategy.

Rather, given that readout is very noisy in many real architectures, it may be interesting to find dynamically driven codes that are particularly robust to measurement errors. In spite of their many similarities with respect to error correction (see Sec. VII B 3), a key difference between the honeycomb code and the Floquet color code is the number of measurements needed to evaluate a detection cell. A detection cell of the honeycomb code depends on 12 measurement outcomes, whereas a detection cell of the Floquet color code only depends on six measurements (see Fig. 48). This difference may lead to a significant contrast in their performance, particularly in the limit where measurement errors are dominant.

We will also need to consider more general types of errors in a real quantum system, such as qubit leakage errors [109] and fabrication defects [60]. Solutions to these problems typically require the use of additional auxiliary qubits and modifications to the syndrome-readout circuit. Simple and elegant modifications have been proposed [110,111] to deal with leakage for the surface code. It will be important for their practical development to find circuits that perform these same tasks for dynamically driven

codes. For the case of fabrication defects, Ref. [60] has shown a syndrome-readout protocol that demonstrates a threshold with topological codes using a defective qubit array. The protocol adds punctures to the topological code to isolate the defective components. To obtain syndrome data needed to demonstrate a threshold, the protocol also requires code deformations to the boundaries of the punctures to transform the boundaries between different types; rough and smooth. We expect that we can adapt this protocol for dynamically condensed color codes, employing the boundary theory that we have presented in Sec. VII C to transform between their rough and smooth boundaries.

### VIII. DISCUSSION AND OUTLOOK

Many architectures for practical fault-tolerant quantum computation are based on the manipulation of topological phases of matter. In this work, we have argued that anyon condensation gives us a framework to describe ways of encoding logical qubits with topological quantum error-correcting codes as well as many of the mechanisms for performing robust logical operations. We have demonstrated the application of this framework explicitly, with the color code as our key example, expressing the many fault-tolerant encodings and implementations of logic gates in the color code as applications of anyon condensation. This framework has also shown us how to generalize many of these color-code encodings and logic gates using the notion of partial condensation, where a subset of bosons of the color code are condensed on a region of the lattice to manipulate topological degrees of freedom. Finally, we have reformulated the notion of a dynamically driven Floquet code in terms of a condensate of a parent color-code theory and this new perspective has enabled us to generalize known Floquet codes as well as providing a constructive way to design the boundary stabilizers of our Floquet-code construction.

With our work, the catalogue of topological features available in the color code [23] has been significantly extended to now include semipunctures, semitransparent domain walls, and generalized twist defects that appear at their end points. Importantly, the language that we use to describe these features is independent of their orientation in the  $(2 + 1)$ D space-time volume describing the computation. Because of this symmetry, we can make use of a given topological feature in many different ways to perform computational tasks. This perspective allows for these topological objects to be used for fault-tolerant state initialization and readout, to apply logical gates, and to transform between inequivalent schemes of topological encoding. As a *practical application* of this new framework, we expect that the large zoo of feasible operations available to us can be used to decrease the physical-resource overheads of fault-tolerant logical operations that plague existing constructions. Notions of lattice surgery

as discussed in Ref. [20] can be seen as first steps in this direction. The framework developed here is expected to be highly instrumental in developing schemes that are even more economical, contributing to the quest of identifying schemes of fault-tolerant quantum computing with reasonable overheads.

More broadly, we have shown that partial anyon condensation can provide us a number of novel operations within the framework of topological fault-tolerant quantum computing. These operations complement known logic operations that are completed with constant-depth unitary circuits, such as transversal gates, as well as more conventional measurement-based operations that manipulate punctures and braiding operations for non-Abelian pointlike objects such as twists. We have found the color-code model to be particularly illustrative of this idea, due to its rich and numerous symmetries, exhibited at the level of its low-energy excitations. Moving forward, it will be exciting to understand how similar mechanisms are manifest in more general topological phases. We may study, e.g., other  $(2 + 1)$ D topological quantum field theories [112], including non-Abelian theories [113], as well as higher-dimensional topological phases and fracton phases. In order to make progress along these lines, one could seek other examples of condensation processes in other phases and develop a theory of anyon condensation for general classes of phases.

Lastly, we have also shown that dynamically prepared Floquet codes can be rederived, and generalized, within the framework of anyon condensation of color-code excitations. We have been able to design key features of this novel type of code, such as the code boundaries, by appealing to the physics of the parent color-code model. Given the practicality of their realization, the development of Floquet codes may provide better ways of realizing different types of nontrivial topological phases. In future work, it will be interesting to discover to what extent our theory of Floquet codes from anyon condensation can be generalized to other types of topological phases. Further afield, it may also be that developments of the theory of dynamical codes may show us more practical ways of implementing quantum low-density parity-check codes [97].

The source code for the simulations used to produce the plot in Fig. 51 is made available in an online repository [114].

### ACKNOWLEDGMENTS

We would like to thank D. Barter and J. Bridgeman for several discussions on semitransparent domain walls that initiated this project and we thank D. Williamson for asking questions that led us to our construction for dynamically driven codes. We are also grateful for helpful discussions and comments from D. Aasen, A. Bauer, A. Cross, M. Davydova, P.J. Derks, T. Ellison, C. Gidney, O.

Higgott, M. Newman, A. Townsend-Teague, Z. Wang, and J. Wootton. B.J.B. is also grateful for the hospitality of the Center for Quantum Devices at the University of Copenhagen, where parts of this work were completed. This work has been supported by the Federal Ministry of Education and Research (Bundesministerium für Bildung und Forschung, BMBF) [Realistic quantum error mitigation (RealistiQ), Quantum computer in the solid state (QSolid), Munich Quantum Valley quantum computer demonstrators - neutral atoms based quantum computing demonstrator (MUNIQC-Atoms)], the German Research Foundation (Deutsche Forschungsgemeinschaft, DFG (CRC 183), the Quantum Flagship [Millenion, Programmable atomic large-scale quantum simulation 2.0 (PasQuans2)], and the Einstein Foundation (Einstein Research Unit on quantum devices). 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. This work is also supported by the Australian Research Council via the Centre of Excellence in Engineered Quantum Systems (EQUS) Project No. CE170100009, and by the Army Research Office (ARO) under Grant No. W911NF-21-1-0007. F.T. is supported by the Sydney Quantum Academy. B.J.B. has changed affiliation to IBM Quantum during the preparation of this paper.

## APPENDIX: DOMAIN WALLS AND BOUNDARIES IN STACKED TORIC CODE PHASES

In this appendix, we elaborate how results on the classification and construction of boundaries and domain walls in (Abelian) quantum double models apply to a phase that is equivalent to  $n$  layers of the toric code. We will see that in this case, many properties of a given (abstractly defined) boundary are easily computable. In particular, the color code is equivalent to two layers of the toric code (see Sec. II D), so  $n = 2$ .

First, let us show why it is important to study boundaries, even if one is interested in domain walls.

### 1. Folding trick

Until now, we have seen how to understand domain walls in terms of condensation and symmetry. However, in practical scenarios, one might want to have a simpler picture to understand and/or construct domain walls. To this end, we can employ the *folding trick*. It relates a domain wall to a boundary of a “folded” phase, composed of the phases on both sides of the domain wall of interest.

The folding trick allows us to think about any  $\mathcal{C}$ - $\mathcal{C}'$  domain wall as a boundary of the stacked theory  $\mathcal{C} \otimes \mathcal{C}'$  [115]. The folding trick can be easily understood pictorially by folding a plane with a domain wall along it and identifying the cut with a boundary to a vacuum (see Fig. 57). Mathematically speaking, the folding trick is used

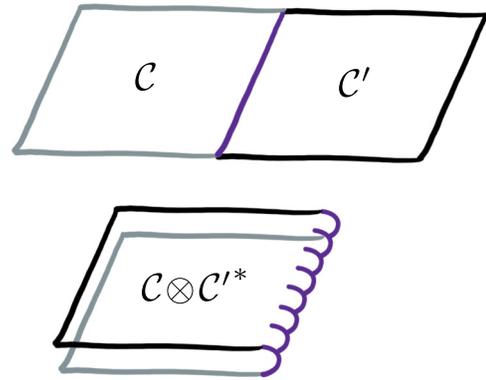


FIG. 57. A  $\mathcal{C}$  –  $\mathcal{C}'$  domain wall (top) is equivalent to a boundary of stacked phase  $\mathcal{C} \otimes \mathcal{C}'^*$  (bottom). This equivalence is often referred to as the *folding trick*. Note that  $\mathcal{C}'$  is inverted by the fold. In this work, however, since we consider qubit-stabilizer models,  $\mathcal{C}'^* = \mathcal{C}'$ .

to classify domain walls in terms of boundaries [34,63,66], which is a simpler task, since a topological boundary (of an Abelian phase) is fully described by a Lagrangian subgroup.

In order to think about domain walls as boundaries of a stacked phase, we establish what the structure of the Lagrangian subgroup (folded) tells us about the domain wall (unfolded). The fold gives a natural decomposition of the anyon fusion group  $\mathcal{C} \otimes \mathcal{C}'$ . A generator of  $\mathcal{L}$  of the form  $a \otimes a'$ , without  $a$  and  $a'$  appearing individually, means that in the unfolded picture  $a \mapsto a'$  when passing through the domain wall. For an invertible domain wall, every generator is of that form; for an opaque domain wall, all generators can be brought into the form  $a \otimes 1$  or  $1 \otimes a'$ . The semitransparent domain walls are the ones with generators of both of the above types.

The equivalence of domain walls and certain boundaries shows that, mainly for computational purposes, it suffices to consider how to construct condensable bosons for boundaries. In Sec. III C, we will show how the Lagrangian subgroup of any boundary of layers of toric codes can be directly calculated based on its classifying data.

### 2. General structure of boundaries

In Sec. III C, we have shown that domain walls—gapped interfaces of topological phases—can be understood in terms of the condensate formed by the mobile anyons. This classifies all possible domain walls up to automorphisms on both sides of the interface. In this section, we will explicitly show how this categorizes domain walls from  $n$  to  $m$  layers of the toric code. In particular, we will see how the condition that the mobile anyons have to form a condensate constrains the domain wall.

As the general case, we consider a domain wall from  $n$  layers of the toric code,  $\text{TC}_n$ , to  $m$  layers  $\text{TC}_m$ . We show that any such domain wall is equivalent to an opaque

domain wall on a subset of layers and a fully transparent domain wall on the remaining layers up to symmetries on both sides of the interface,  $\text{Aut}(TC_n)$  and  $\text{Aut}(TC_m)$ . In the language of anyon condensation, an equivalent statement is that any condensate of  $n$  layers of toric code is in the phase of  $k$  layers of toric code, where  $k \leq n$ . We will prove this in the remainder of this section using the tools laid out in the main text.

Imagine a stabilizer model for  $n$  layers of toric code, the quantum double model [2] for  $G = \mathbb{Z}_2^n$ . By construction, it is a qubit-Pauli-stabilizer code. Any condensate can be modeled by adding the corresponding hopping terms to the stabilizer-group generators (see Sec. III D). In the toric code, any such string operator is a (multi)qubit Pauli word. Hence, the stabilizer group in the condensed phase will be a qubit Pauli group. From Ref. [24], we know that any such code is in the same phase as  $k$  layers of toric code. Counting the cosets making up the condensate given a set of condensable bosons and the conditions on them shows that  $k \leq n$ . This completes one proof of the above statement.

Given the simplicity of the stabilizer-based proof, the microscopic description of the condensate is not necessary. We have given a simple proof that demonstrates how condensates are constrained by appealing to the microscopic details of the  $G = \mathbb{Z}_2^n$  toric code at the level of the stabilizer group. We find that it is also instructive to rederive the result at the level of the low-energy particle theory of the quantum double model. Specifically, one can also work out the condensed phase explicitly by looking at the conditions defining the condensed phase when an arbitrary boson in the  $n$ -layer toric-code phase is condensed. Checking that the fusion group and the modular data of the resulting anyons coincide with  $n - 1$  layers of toric code is straightforward using  $\mathbb{Z}_2$  arithmetic. The proof for any number of condensed bosons then follows by induction.

### 3. Calculating Lagrangian subgroups

It is a well-established fact in the mathematical condensed-matter literature that the boundaries of a quantum double model of a finite group  $G$  are classified by a subgroup  $N \subset G$  and a so-called *2-cocycle class*  $[\psi] \in H^2(N, U(1))$  [62,63]. Moreover, Ref. [63] has established a formula to calculate the Lagrangian subgroup of a boundary  $(N, \psi)$  of a generic quantum double model, Abelian or non-Abelian. In this section, we explicitly show the formula in the case of  $n$  layers of the toric code and give some intuition for the quantities appearing in the expressions. We end the section with some simple examples to see how familiar boundaries of the toric and color code come out of this description.

The topological phase of  $n$  layers of the toric code is modeled by a quantum double model with gauge group  $G = \mathbb{Z}_2^n$ . For the remainder of this section, we represent its elements by binary vectors of length  $n$  and the group

multiplication by entry-wise addition modulo 2,

$$\mathbf{a} \oplus \mathbf{b} := \begin{pmatrix} a_1 + b_1 \pmod{2} \\ a_2 + b_2 \pmod{2} \\ \vdots \\ a_n + b_n \pmod{2} \end{pmatrix}. \quad (\text{A1})$$

Any subgroup is of the form  $\mathbb{Z}_2^k$  for  $k \leq n$  and is generated by  $k$  independent generators. Any such subgroup, together with a 2-cocycle on it, defines a boundary of the quantum double model. A 2-cocycle on the subgroup is a function  $\psi : N \times N \rightarrow U(1)$  with the property

$$\psi(\mathbf{b}, \mathbf{c})\psi(\mathbf{a}, \mathbf{b} \oplus \mathbf{c}) = \psi(\mathbf{a} \oplus \mathbf{b}, \mathbf{c})\psi(\mathbf{a}, \mathbf{b}) \quad (\text{A2})$$

for all  $\mathbf{a}, \mathbf{b}, \mathbf{c} \in N$ . 2-cocycles are sorted into *equivalence classes*, where two 2-cocycles  $\psi$  and  $\tilde{\psi}$  are equivalent if there exists a function  $\beta : N \rightarrow U(1)$  such that

$$\tilde{\psi}(\mathbf{a}, \mathbf{b}) = \psi(\mathbf{a}, \mathbf{b}) \frac{\beta(\mathbf{a})\beta(\mathbf{b})}{\beta(\mathbf{a} \oplus \mathbf{b})}. \quad (\text{A3})$$

The set of these equivalence classes is called the *second cohomology group* and denoted by  $H^2(N, U(1))$ . For more details on the origin of this equivalence relation and group cohomology, we refer to the appendix of Ref. [116]. In fact, the type of boundary defined by a subgroup and 2-cocycle pair only depends on the equivalence class that contains the 2-cocycle. In every 2-cocycle class, there is a special representative that satisfies

$$\psi(\mathbf{0}, \mathbf{a}) = \psi(\mathbf{a}, \mathbf{0}) = 1, \quad \forall \mathbf{a}, \quad (\text{A4})$$

where  $\mathbf{0}$  denotes the identity element in  $N$ . We call such a 2-cocycle *normalized*. Let us elaborate on the form of inequivalent normalized 2-cocycles for  $N = \mathbb{Z}_2^k$ . In fact, for  $k = 1$ , there is only a single equivalence class, represented by the trivial 2-cocycle

$$\psi_0(\mathbf{a}, \mathbf{b}) = 1 \quad \forall \mathbf{a}, \mathbf{b}. \quad (\text{A5})$$

For  $k = 2$ , however, there exists one nontrivial 2-cocycle class, which is represented by

$$\psi_1(\mathbf{a}, \mathbf{b}) = (-1)^{a_1 b_2}. \quad (\text{A6})$$

Since it is of order two, i.e.,  $(\psi_1)^2 = \psi_0 \equiv 1$ , we write  $H^2(\mathbb{Z}_2 \times \mathbb{Z}_2, U(1)) = \mathbb{Z}_2$ . Luckily, we can construct a representative of any nontrivial 2-cocycle class on  $\mathbb{Z}_2^k$  for any  $k$  from the nontrivial 2-cocycle  $\psi_1$ . Using the Künneth formula for group cohomology [116] and the fact that  $\mathbb{Z}_2$  only has trivial 2-cocycles, one can show that for  $k > 1$ , the 2-cocycles over  $\mathbb{Z}_2^k$  decompose into 2-cocycles defined on

pairs of tensor factors. Abstractly,

$$\begin{aligned} H^2(\mathbb{Z}_2^k, U(1)) &= \bigotimes_{(i,j) \text{ pairs}} H^2((\mathbb{Z}_2)_i \times (\mathbb{Z}_2)_j, U(1)) \\ &= \bigotimes_{(i,j) \text{ pairs}} \mathbb{Z}_2 = \mathbb{Z}_2^{\binom{k}{2}}, \end{aligned} \quad (\text{A7})$$

where  $(\mathbb{Z}_2)_i$  refers to the  $i$ th tensor factor of  $\mathbb{Z}_2^k$  and  $\binom{k}{2} = k(k-1)/2$  is the number of pairs in the  $k$  factors. For the 2-cocycles themselves, this means that we can write the (normalized) representative of any 2-cocycle class as a product of (normalized) 2-cocycles each of which only acts nontrivially on a pair of factors  $(\mathbb{Z}_2)_i \times (\mathbb{Z}_2)_j$  in  $\mathbb{Z}_2^k$ ,

$$\begin{aligned} \psi_{\{n_{i,j}\}}(\mathbf{a}, \mathbf{b}) &= \prod_{(i,j) \text{ pairs}} \psi_1(\mathbf{a}_{i,j}, \mathbf{b}_{i,j})^{n_{i,j}} \\ &= (-1)^{\sum_{i<j} n_{i,j} a_i b_j}, \end{aligned} \quad (\text{A8})$$

where  $\mathbf{a}_{i,j}$  denotes the restriction of  $\mathbf{a}$  on  $(\mathbb{Z}_2)_i \times (\mathbb{Z}_2)_j \subset \mathbb{Z}_2^k$  and the set of  $n_{i,j} = 0, 1$  labels the 2-cocycle classes and indicates if the 2-cocycle is nontrivial on pair  $(i, j)$ . Taken together, inequivalent boundaries of  $n$  layers of the toric code are classified by a subgroup  $\mathbb{Z}_2^k \subset \mathbb{Z}_2^n$  and a set of  $\mathbb{Z}_2$  numbers  $\{n_{i,j} \mid (i, j) \text{ pairs of factors in } \mathbb{Z}_2^k\}$ .

Now that we have introduced the quantities that define the bulk model (finite group,  $G = \mathbb{Z}_2^n$ ) and a boundary (subgroup  $N \simeq \mathbb{Z}_2^k$  and a 2-cocycle class in  $H^2(\mathbb{Z}_2^k, U(1))$ ), we can turn our attention to how to describe the bulk anyons and their condensation at the boundary in terms of this defining data. The bulk anyons are labeled by a pair of group elements, in our case  $\mathbb{Z}_2^n \times \mathbb{Z}_2^n$ , where the first factor represents the electric part and the other one the magnetic part. For example,  $e_1 = (0, \dots | 1, \dots)$ ,  $m_1 = (1, \dots | 0, \dots)$  are, respectively, the electric and magnetic anyon on the first layer. Their composite particle, the fermion on the first layer, is represented by their sum  $f_1 = e_1 \oplus m_1 = (1, 0, \dots | 1, 0, \dots)$ . The modular data (the monodromy and topological spin) can all be computed by the so-called *character function*  $\chi_{\bullet}(\bullet) : \mathbb{Z}_2^n \times \mathbb{Z}_2^n \rightarrow \{\pm 1\}$ ,

$$\chi_{\mathbf{a}}(\mathbf{b}) = (-1)^{\sum_{i=1}^n a_i b_i}. \quad (\text{A9})$$

Note that this function couples the first and the second argument on every ‘‘layer’’ individually. This is related to the fact that in layers of toric code an anyon has nontrivial topological spin (of  $-1$ ) if it has both an electric and a magnetic component on the same layer. For the exact formulas, we refer to Ref. [117].

The way in which we calculate the Lagrangian subgroup corresponding to a boundary labeled by a subgroup  $N$  and a 2-cocycle  $\psi_{n_{i,j}}$  is to construct an indicator function  $m^{N, \{n_{i,j}\}} : \mathbb{Z}_2^n \times \mathbb{Z}_2^n \rightarrow \{0, 1\}$ , which evaluates to 1 if anyon  $(\mathbf{g}|\mathbf{h})$  can condense, i.e., is part of the Lagrangian

subgroup, and 0 if not. Applying the construction outlined in Ref. [63] to our case (where  $G = \mathbb{Z}_2^n$  and  $N \simeq \mathbb{Z}_2^k$  with the 2-cocycles described above), this function reads

$$m^{N, \{n_{i,j}\}}(\mathbf{g}|\mathbf{h}) = \delta_{\mathbf{g} \in N} \frac{1}{|N|} \sum_{\mathbf{l} \in N} \chi_{\mathbf{h}}(\mathbf{l}) \psi_{\{n_{i,j}\}}(\mathbf{l}, \mathbf{g}) \psi_{\{n_{i,j}\}}(\mathbf{g}, \mathbf{l}) \quad (\text{A10a})$$

$$\begin{aligned} &= \delta_{\mathbf{g} \in N} \frac{1}{2^k} \sum_{\mathbf{l} \in N} (-1)^{\sum_{i=1}^n l_i h_i} \\ &\quad \times (-1)^{\sum_{m=0}^k \sum_{j>m=0}^k \mathbb{1}_{\{m, n_{m,j} \mathbf{g} | m+1\}_j n_{m,j} \mathbf{g} | m}}, \end{aligned} \quad (\text{A10b})$$

where  $|_i$  denotes the projection onto the  $i$ th tensor factor of  $N$ . We see that the  $\mathbf{g}$  part (the magnetic component) is purely determined by the subgroup  $N$ . For the origin of this function, we refer to Refs. [63, 88]. By which  $\mathbf{h}$  part (electric component) it is accompanied is determined by the 2-cocycle characterizing the boundary. To understand the formula better, let us look at some examples.

### a. Example 1: Surface-code boundaries, $n = 1$

The simplest examples are the toric code boundaries, where  $G_1 = \mathbb{Z}_2$ . The four bulk anyons are labeled by  $\{(0|0), (0|1), (1|0), (1|1)\}$ , corresponding to the trivial, electric, magnetic, and fermionic charge. The group  $\mathbb{Z}_2$  has two trivial subgroups,  $N_1 = \{0\}$  and  $N_2 = G$ . Both have only trivial 2-cocycles, so the indicator function reduces to

$$\begin{aligned} m^{N_i}(\mathbf{g}|\mathbf{h}) &= \delta_{\mathbf{g} \in N_i} \frac{1}{|N_i|} \sum_{\mathbf{l} \in N_i} (-1)^{h\mathbf{l}} \\ &= \delta_{\mathbf{g} \in N_i} \prod_{\mathbf{l} \in N_i} \delta_{h\mathbf{l}=0}, \end{aligned} \quad (\text{A11})$$

where we have identified that the second part of the right-hand side is only nonzero if  $h\mathbf{l} = 0$  for all  $\mathbf{l} \in N_i$ . This yields the Lagrangian subgroups

$$\mathcal{L}_{N_1} = \{(0|0), (0|1)\} \quad \text{and} \quad (\text{A12a})$$

$$\mathcal{L}_{N_2} = \{(0|0), (1|0)\}, \quad (\text{A12b})$$

which are exactly the pure electric anyons for  $N_1$  and the pure magnetic anyons for  $N_2$  as expected for a single layer of toric code.

### b. Example 2: Color-code boundaries, $n = 2$

The color code can be unfolded to two layers of toric code (see Sec. IID), i.e., a *quantum double model* with  $G_2 = \mathbb{Z}_2 \times \mathbb{Z}_2$ . Besides the two trivial subgroups,  $N_1 = \{(0, 0)\}$  and  $N_2 = G_2$ , we find that  $G_2$  has three other subgroups,  $N_3 = \langle(0, 1)\rangle$ ,  $N_4 = \langle(1, 0)\rangle$ , and  $N_5 = \langle(1, 1)\rangle$ ,

each of which is isomorphic to  $\mathbb{Z}_2$ . As discussed above,  $N_2$  has two 2-cocycle classes, whereas the other subgroups only host a trivial one. In total, this gives six boundaries corresponding to the six Lagrangian subgroups in Eqs. (16). In the following, we will see how they can be recovered with formula in Eq. (A10).

Let us first consider the trivial 2-cocycle on any of the subgroups. This gives a similar formula to the one in the previous example,

$$\begin{aligned} m^{N_i}(g_1 g_2 | h_1 h_2) &= \delta_{(g_1 g_2) \in N_i} \frac{1}{|N_i|} \sum_{l \in N_i} (-1)^{h_1 l_1 + h_2 l_2} \\ &= \delta_{(g_1 g_2) \in N_i} \prod_{l \in N_i} \delta_{l \cdot \mathbf{h} = 0 \bmod 2}, \end{aligned} \quad (\text{A13})$$

which yields the Lagrangian subgroups

$$\mathcal{L}_{N_1} = \{(00|ab) \mid a, b = 0, 1\}, \quad (\text{A14a})$$

$$\mathcal{L}_{N_2} = \{(ab|00) \mid a, b = 0, 1\}, \quad (\text{A14b})$$

$$\mathcal{L}_{N_3} = \{(0a|b0) \mid a, b = 0, 1\}, \quad (\text{A14c})$$

$$\mathcal{L}_{N_4} = \{(a0|0b) \mid a, b = 0, 1\}, \quad (\text{A14d})$$

$$\mathcal{L}_{N_5} = \{(aa|bb) \mid a, b = 0, 1\}. \quad (\text{A14e})$$

Again, we recover the pure  $e$  (respectively,  $m$ ) condensing boundaries with the two trivial subgroups  $N_1$  and  $N_2$  (with trivial 2-cocycle). There is an additional Lagrangian subgroup for the nontrivial 2-cocycle  $\psi_1$  on  $N_2 = G$  [see Eq. (A6)]. Plugging this into the indicator function gives

$$\begin{aligned} m^{N_2, \psi_1}(g_1 g_2 | h_1 h_2) &= \delta_{(g_1 g_2) \in G} \frac{1}{4} \sum_{a_1, a_2 = 0, 1} (-1)^{a_1(h_1 + g_2) + a_2(h_2 + g_1)} \\ &= \delta_{h_1 + g_2 = 0 \bmod 2} \delta_{h_2 + g_1 = 0 \bmod 2}, \end{aligned} \quad (\text{A15a})$$

Evaluating the function for every anyon label, we obtain the sixth Lagrangian subgroup

$$\mathcal{L}_{N_2, \psi_1} = \{(ab|ba) \mid a, b = 0, 1\}. \quad (\text{A16})$$

Depending on the unfolding one chooses to map the color code to two toric code layers, the above Lagrangian subgroups correspond to different boundaries. For example, choosing the standard mapping Eq. (9),  $\mathcal{L}_{N_1}$  corresponds to the  $x$  boundary,  $\mathcal{L}_{N_4}$  to the red boundary,  $\mathcal{L}_{N_2, \psi_1}$  to the  $z$  boundary, etc.

### c. Example 3: Color-code domain walls, $n = 4$

Via the folding trick, domain walls in the color code are in one-to-one correspondence to boundaries of two layers of color code. Each color-code anyon is labeled by a  $\mathbb{Z}_2^{\times 4}$  element, so in the folded picture (see Sec. A 1), they are

labeled by  $\mathbb{Z}_2^{\times 4} \times \mathbb{Z}_2^{\times 4}$  elements, each factor corresponding to the anyons on either side of the domain wall (before folding). We label the magnetic fluxes on the left (right) side of the domain wall by  $m_i^{l(r)}$  and the electric charges by  $e_i^{l(r)}$ , respectively. For our purposes in  $m^{N, n_{ij}}(\bullet)$ , we represent them by binary strings

$$(\mathbf{g}|\mathbf{h}) = (g_1 g_2 g_3 g_4 | h_1 h_2 h_3 h_4) \quad (\text{A17a})$$

$$\simeq (m_1^l)^{g_1} (m_2^l)^{g_2} (m_1^r)^{g_3} (m_2^r)^{g_4} (e_1^l)^{h_1} (e_2^l)^{h_2} (e_1^r)^{h_3} (e_2^r)^{h_4}. \quad (\text{A17b})$$

Note that we have associated the left side with the first two bits in  $\mathbf{g}$  (respectively,  $\mathbf{h}$ ) and the last two with the right side of the domain wall.

There are 270 different domain walls in the color code corresponding to the different subgroups of  $\mathbb{Z}_2^4$  and potential nontrivial 2-cocycles on them. We will consider an exemplary subset thereof in this section, the three subgroups  $N_1 = \langle(1000)\rangle \simeq \mathbb{Z}_2$ ,  $N_2 = \langle(1010)\rangle \simeq \mathbb{Z}_2$ , and  $N_3 = \langle(1000), (0010)\rangle \simeq \mathbb{Z}_2 \times \mathbb{Z}_2$ , where the latter can have a nontrivial 2-cocycle.

Plugging in  $N_1$  with a trivial 2-cocycle into Eq. (A10) gives the indicator function for the corresponding Lagrangian subgroup,

$$\begin{aligned} m^{N_1}(\mathbf{g}|\mathbf{h}) &= \delta_{\mathbf{g} \in \{(0000), (1000)\}} \frac{1}{2} \sum_{l=0}^1 (-1)^{h_1 l_1} \\ &= \delta_{\mathbf{g} \in \{(0000), (1000)\}} \delta_{h_1, 0}. \end{aligned} \quad (\text{A18})$$

The associated Lagrangian subgroup is given by

$$\begin{aligned} \mathcal{L}_{N_1} &= \{(a_1 000 | 0 a_2 a_3 a_4) \mid a_i = 0, 1\} \\ &\simeq \langle m_1^l, e_2^l, e_1^r, e_2^r \rangle. \end{aligned}$$

We see that the Lagrangian subgroup is generated by anyons supported only on the left or the right side of the domain wall, so it corresponds to a fully opaque wall when unfolded (see Sec. A 1). This can be traced back to the fact that  $N_1$  is only supported on a single tensor fact of the input group  $\mathbb{Z}_2^{\times 4}$ . This generalizes to all boundaries corresponding to any subgroup that factorizes over the input tensor factors and a trivial 2-cocycle.

$N_2$  also has a single generator, (1010). This generator, however, is supported on two tensor factors, each of which was associated with a different side of the domain wall. For this subgroup, the indicator function reads

$$\begin{aligned} m^{N_2}(\mathbf{g}|\mathbf{h}) &= \delta_{\mathbf{g} \in \{(0000), (1010)\}} \frac{1}{2} \sum_{l=0}^1 (-1)^{l(h_1 + h_3)} \\ &= \delta_{\mathbf{g} \in \{(0000), (1010)\}} \delta_{h_1 + h_3 = 0 \bmod 2}, \end{aligned} \quad (\text{A19})$$

giving rise to the Lagrangian subgroup

$$\begin{aligned} \mathcal{L}_{N_2} &= \{(a_1 0 a_1 0 | a_2 a_3 a_2 a_4 \mid a_i = 0, 1)\} \\ &\simeq \langle m_1^l, m_1^r, e_1^l, e_1^r, e_2^l, e_2^r \rangle. \end{aligned} \quad (\text{A20})$$

When unfolded to a domain wall, this boundary corresponds to a semitransparent domain wall where the anyons  $m_1$  and  $e_1$  can freely pass through and the  $e_2$  particles condense individually from both sides.

For  $N_3$ , there are two associated boundaries, one for each 2-cocycle class. For the trivial 2-cocycle, the calculation goes through similarly to before. The indicator function reads

$$\begin{aligned} m^{N_3}(\mathbf{g}|\mathbf{h}) &= \delta_{\mathbf{g} \in ((1000), (0010))} \frac{1}{4} \sum_{l_1, l_2=0}^1 (-1)^{l_1 h_1 + l_2 h_3} \\ &= \delta_{\mathbf{g} \in ((1000), (0010))} \delta_{h_1=0} \delta_{h_3=0} \end{aligned} \quad (\text{A21})$$

and the associated Lagrangian subgroup is

$$\begin{aligned} \mathcal{L}_{N_3} &= \{(a_1 0 a_2 0 | 0 a_3 0 a_4 \mid a_i = 0, 1)\} \\ &\simeq \langle m_1^l, m_1^r, e_2^l, e_2^r \rangle. \end{aligned} \quad (\text{A22})$$

Again, since the Lagrangian subgroup is generated by anyons supported on a single side of the domain wall, it corresponds to an opaque one. On each side,  $m_1$  and  $e_2$  can condense individually. The fact that it is fully opaque can again be traced back to the fact that the chosen subgroup factorizes over the fold, i.e., is generated by generators solely supported on the first two or last two factors of  $\mathbb{Z}_2^{\times 4}$ . This argument no longer holds for the boundary associated with the same subgroup but a nontrivial 2-cocycle of the form

$$\psi^{N_3}(\mathbf{a}, \mathbf{b}) = (-1)^{a_1 b_3}. \quad (\text{A23})$$

Plugging this into the indicator function gives

$$\begin{aligned} m^{N_3, \psi}(\mathbf{g}|\mathbf{h}) &= \delta_{\mathbf{g} \in N_3} \frac{1}{4} \sum_{l_1, l_2=0}^1 (-1)^{l_1(h_1+g_3)+l_2(h_3+g_1)} \\ &= \delta_{\mathbf{g} \in N_3} \delta_{h_1+g_3=0 \pmod 2} \delta_{h_3+g_1=0 \pmod 2}. \end{aligned} \quad (\text{A24})$$

Note that the nontrivial 2-cocycle couples the constraints on  $\mathbf{h}$  and  $\mathbf{g}$ . The associated Lagrangian subgroup reads

$$\begin{aligned} \mathcal{L}_{N_3, \psi} &= \{(a_1 0 a_2 0 | a_2 a_3 a_1 a_4 \mid a_i = 0, 1)\} \\ &\simeq \langle m_1^l, e_1^r, m_1^r, e_1^l, e_2^l, e_2^r \rangle. \end{aligned} \quad (\text{A25})$$

Even though the defining group factorizes over the individual tensor factors of the input group, the domain wall associated with the Lagrangian subgroup above is semitransparent. The 2-cocycle couples the constraints on  $\mathbf{g}$  and

$\mathbf{h}$  in the indicator function in a way that  $m_1^l$  only condenses when accompanied by  $e_1^r$  and vice versa. As a domain wall, this means that  $m_1$  can pass through from either side but gets transformed into  $e_1$ , i.e., the nontrivial automorphism of the upper layer toric code gets applied. On the lower layer  $e_2^l$  and  $e_2^r$  condense individually, making the domain wall semitransparent. When comparing  $\mathcal{L}_{N_3}$  with  $\mathcal{L}_{N_3, \psi}$  we see that the nontrivial 2-cocycle effectively ‘‘twists’’ the magnetic part in  $\mathcal{L}_{N_3}$  by appending electric charges on a different layer. This generalizes to any nontrivial 2-cocycle on any subgroup isomorphic to  $\mathbb{Z}_2 \times \mathbb{Z}_2$ . It appends the  $m$ -anyons on either of the two factors (of the subgroup) with electric charges on the other one.

#### d. Example 4: Trivial 2-cocycle

As seen in the first three examples, Eq. (A10) takes a particularly simple form for a boundary associated with a trivial 2-cocycle on the chosen subgroup  $N$ ,

$$m^N(\mathbf{g}|\mathbf{h}) = \delta_{\mathbf{g} \in N} \frac{1}{2^k} \sum_{\mathbf{l} \in N} (-1)^{\sum_i l_i h_i} \quad (\text{A26a})$$

$$= \delta_{\mathbf{g} \in N} \prod_{\mathbf{l} \in N} \delta_{\mathbf{h} \cdot \mathbf{l} = 0 \pmod 2}. \quad (\text{A26b})$$

We see that the constraints on  $\mathbf{g}$  and  $\mathbf{h}$  decouple. The Lagrangian subgroup consists of anyons with  $g \in N$  accompanied with a charge that has even overlap with any element in  $N$ , i.e., for which  $\sum_i l_i h_i = 0 \pmod 2 \forall \mathbf{l} \in N$ . This agrees with our understanding that in order for two anyons to braid trivially in layers of toric code, the  $e$  and the  $m$  parts have to overlap on an even number of layers. As a consistency check, we can count all  $\mathbf{g}, \mathbf{h}$  for which  $m^N(\mathbf{g}|\mathbf{k}) = 1$ . In any Abelian-anyon model, the order of a Lagrangian subgroup is the square root of the total number of anyons in the bulk. In our case, where the bulk has  $2^{2n}$  anyons, we hence expect  $2^n$  different  $(\mathbf{g}|\mathbf{k})$  that condense. The  $\mathbf{g}$  label has to be in  $N$ , giving rise to  $2^k$  different values. Additionally,  $\mathbf{h} \in \mathbb{Z}_2^n$  is constrained by  $k$  independent constraints (one for each independent generator of  $N$ ) of the form  $\mathbf{h} \cdot \mathbf{l} = 0 \pmod 2$ , each of which reduces the number of valid values by a factor of 2. Put together,  $\mathbf{h}$  can take  $2^{n-k}$  different values, independent of  $\mathbf{g}$ , and we get  $2^k 2^{n-k} = 2^n$  anyons for which  $m^N(\mathbf{g}|\mathbf{k}) = 1$ .

- 
- [1] E. Dennis, A. Kitaev, A. Landahl, and J. Preskill, Topological quantum memory, *J. Math. Phys.* **43**, 4452 (2002).
  - [2] A. Y. Kitaev, Fault-tolerant quantum computation by anyons, *Ann. Phys.* **303**, 2 (2003).
  - [3] M. Freedman, M. Larsen, and Z. Wang, A modular functor which is universal for quantum computation, *Commun. Math. Phys.* **227**, 605 (2000).

- [4] M. A. Levin and X.-G. Wen, String-net condensation: A physical mechanism for topological phases, *Phys. Rev. B* **71**, 045110 (2005).
- [5] H. Bombín and M. A. Martin-Delgado, Topological quantum distillation, *Phys. Rev. Lett.* **97**, 180501 (2006).
- [6] C. Nayak, S. H. Simon, A. Stern, M. Freedman, and S. Das Sarma, Non-Abelian anyons and topological quantum computation, *Rev. Mod. Phys.* **80**, 1083 (2008).
- [7] W. X. Gang, *Quantum Field Theory of Many-Body Systems: From the Origin of Sound to an Origin of Light and Electrons* (Oxford University Press, Oxford, 2007).
- [8] B. J. Brown, D. Loss, J. K. Pachos, C. N. Self, and J. R. Wootton, Quantum memories at finite temperature, *Rev. Mod. Phys.* **88**, 045005 (2016).
- [9] R. Raussendorf, J. Harrington, and K. Goyal, A fault-tolerant one-way quantum computer, *Ann. Phys.* **321**, 2242 (2006).
- [10] R. Raussendorf and J. Harrington, Fault-tolerant quantum computation with high threshold in two dimensions, *Phys. Rev. Lett.* **98**, 190504 (2007).
- [11] P. Bonderson, M. Freedman, and C. Nayak, Measurement-only topological quantum computation, *Phys. Rev. Lett.* **101**, 010501 (2008).
- [12] H. Bombín and M. A. Martin-Delgado, Quantum measurements and gates by code deformation, *J. Phys. A* **42**, 095302 (2009).
- [13] A. G. Fowler, Two-dimensional color-code quantum computation, *Phys. Rev. A* **83**, 042310 (2011).
- [14] C. Horsman, A. G. Fowler, S. Devitt, and R. Van Meter, Surface code quantum computing by lattice surgery, *New J. Phys.* **14**, 123011 (2012).
- [15] B. J. Brown, K. Laubscher, M. S. Kesselring, and J. R. Wootton, Poking holes and cutting corners to achieve Clifford gates with the surface code, *Phys. Rev. X* **7**, 021029 (2017).
- [16] H. Bombín, 2D quantum computation with 3D topological codes, *ArXiv:1810.09571* (2018).
- [17] B. J. Brown, A fault-tolerant non-Clifford gate for the surface code in two dimensions, *Sci. Adv.* **6**, eaay4929 (2020).
- [18] G. Zhu, T. Jochym-O'Connor, and A. Dua, Topological order, quantum codes, and quantum computation on fractal geometries, *PRX Quantum* **3**, 030338 (2022).
- [19] A. J. Landahl and C. Ryan-Anderson, Quantum computing by color-code lattice surgery, *ArXiv:1407.5103* (2014).
- [20] F. Thomsen, M. S. Kesselring, S. D. Bartlett, and B. J. Brown, Low-overhead quantum computing with the color code, *ArXiv:2201.07806* (2022).
- [21] B. Yoshida, Topological color code and symmetry-protected topological phases, *Phys. Rev. B* **91**, 245131 (2015).
- [22] J. C. Bridgeman, S. D. Bartlett, and A. C. Doherty, Tensor networks with a twist: Anyon-permuting domain walls and defects in projected entangled pair states, *Phys. Rev. B* **96**, 245122 (2017).
- [23] 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).
- [24] H. Bombin, G. Duclos-Cianci, and D. Poulin, Universal topological phase of two-dimensional stabilizer codes, *New J. Phys.* **14**, 073048 (2012).
- [25] A. Bhagoji and P. Sarvepalli, in *2015 IEEE International Symposium on Information Theory (ISIT)* (IEEE (Institute of Electrical and Electronics Engineers), Hong Kong, 2015), p. 1109, <https://ieeexplore.ieee.org/abstract/document/7282627>.
- [26] A. Kubica, B. Yoshida, and F. Pastawski, Unfolding the color code, *New J. Phys.* **17**, 083026 (2015).
- [27] S. Roberts and D. J. Williamson, 3-fermion topological quantum computation, *PRX Quantum* **5**, 010315 (2024).
- [28] F. J. Burnell, Anyon condensation and its applications, *Ann. Rev. Condens Matter Phys.* **9**, 307 (2018).
- [29] F. A. Bais and J. K. Slingerland, Condensate-induced transitions between topologically ordered phases, *Phys. Rev. B* **79**, 045316 (2009).
- [30] K. Duivenvoorden, M. Iqbal, J. Haegeman, F. Verstraete, and N. Schuch, Entanglement phases as holographic duals of anyon condensates, *Phys. Rev. B* **95**, 235119 (2017).
- [31] M. Iqbal, K. Duivenvoorden, and N. Schuch, Study of anyon condensation and topological phase transitions from a  $F_4$  topological phase using the projected entangled pair states approach, *Phys. Rev. B* **97**, 195124 (2018).
- [32] H. Bombín and M. A. Martin-Delgado, Family of non-Abelian Kitaev models on a lattice: Topological condensation and confinement, *Phys. Rev. B* **78**, 115421 (2008).
- [33] C. Gidney, Stability experiments: The overlooked dual of memory experiments, *Quantum* **6**, 786 (2022).
- [34] A. Kitaev and L. Kong, Models for gapped boundaries and domain walls, *Commun. Math. Phys.* **313**, 351 (2012).
- [35] L. Kong, Anyon condensation and tensor categories, *Nucl. Phys. B* **886**, 436 (2014).
- [36] M. B. Hastings and J. Haah, Dynamically generated logical qubits, *Quantum* **5**, 564 (2021).
- [37] J. Haah and M. B. Hastings, Boundaries for the honeycomb code, *Quantum* **6**, 693 (2022).
- [38] C. Gidney and M. Newman, Benchmarking the planar honeycomb code, *Quantum* **6**, 813 (2022).
- [39] M. Davydova, N. Tantivasadakarn, and S. Balasubramanian, Floquet codes without parent subsystem codes, *PRX Quantum* **4**, 020341 (2023).
- [40] H. Bombin, D. Litinski, N. Nickerson, F. Pastawski, and S. Roberts, Unifying flavors of fault tolerance with the ZX calculus, *ArXiv:2303.08829* (2023).
- [41] J. R. Wootton, Measurements of Floquet code plaquette stabilizers, *ArXiv:2210.13154* (2022).
- [42] A. Kitaev, Anyons in an exactly solved model and beyond, *Ann. Phys.* **321**, 2 (2006).
- [43] H. Bombín, Structure of 2D topological stabilizer codes, *Commun. Math. Phys.* **327**, 387 (2014).
- [44] D. Gottesman, Stabilizer codes and quantum error correction, *ArXiv:0904.2557* (2009).
- [45] D. S. Wang, A. G. Fowler, C. D. Hill, and L. C. L. Hollenberg, Graphical algorithms and threshold error rates for the 2D color code, *Quantum Inf. Comput.* **10**, 0780 (2010).
- [46] A. J. Landahl, J. T. Anderson, and P. R. Rice, Fault-tolerant quantum computing with color codes, *ArXiv:1108.5738* (2011).

- [47] P. Sarvepalli and R. Raussendorf, Efficient decoding of topological color codes, *Phys. Rev. A* **85**, 022317 (2012).
- [48] N. Delfosse and N. H. Nickerson, Almost-linear time decoding algorithm for topological codes, *ArXiv:1709.06218* (2017).
- [49] A. B. Alohious and P. K. Sarvepalli, Projecting three-dimensional color codes onto three-dimensional toric codes, *Phys. Rev. A* **98**, 012302 (2018).
- [50] Y. Li, Fault-tolerant fermionic quantum computation based on color code, *Phys. Rev. A* **98**, 012336 (2018).
- [51] D. K. Tuckett, A. S. Darmawan, C. T. Chubb, S. Bravyi, S. D. Bartlett, and S. T. Flammia, Tailoring surface codes for highly biased noise, *Phys. Rev. X* **9**, 041031 (2019).
- [52] J. X. Li, J. M. Renes, and P. O. Vontobel, Pseudocodeword-based decoding of quantum color codes, *ArXiv:2010.10845* (2020).
- [53] C. T. Chubb, General tensor network decoding of 2D Pauli codes, *ArXiv:2101.04125* (2021).
- [54] E. Sabo, A. B. Alohious, and K. R. Brown, Trellis decoding for qudit stabilizer codes and its application to qubit topological codes, *ArXiv:2106.08251* (2021).
- [55] K. Sahay and B. J. Brown, Decoder for the triangular color code by matching on a Möbius strip, *PRX Quantum* **3**, 010310 (2022).
- [56] J. R. Wootton, A family of stabilizer codes for anyons and Majorana modes, *J. Phys. A* **48**, 215302 (2015).
- [57] B. Criger and B. M. Terhal, Noise thresholds for the  $[[4, 2, 2]]$ -concatenated toric code, *Quantum Inf. Comput.* **16**, 1261 (2016).
- [58] C. Wang, J. Harrington, and J. Preskill, Confinement-Higgs transition in a disordered gauge theory and the accuracy threshold for quantum memory, *Ann. Phys.* **303**, 31 (2003).
- [59] C. Gidney, STIM: A fast stabilizer circuit simulator, *Quantum* **5**, 497 (2021).
- [60] A. Strikis, S. C. Benjamin, and B. J. Brown, Quantum computing is scalable on a planar array of qubits with fabrication defects, *Phys. Rev. Appl.* **19**, 064081 (2023).
- [61] M. Levin, Protected edge modes without symmetry, *Phys. Rev. X* **3**, 021009 (2013).
- [62] A. Davydov and D. Simmons, On Lagrangian algebras in group-theoretical braided fusion categories, *J. Algebra* **471**, 149 (2017).
- [63] S. Beigi, P. W. Shor, and D. Whalen, The quantum double model with boundary: Condensations and symmetries, *Commun. Math. Phys.* **306**, 663 (2011).
- [64] H. Bombin, Topological order with a twist: Ising anyons from an Abelian model, *Phys. Rev. Lett.* **105**, 030403 (2010).
- [65] M. Barkeshli, P. Bonderson, M. Cheng, and Z. Wang, Symmetry, defects, and gauging of topological phases, *Phys. Rev. B* **100**, 115147 (2019).
- [66] J. C. Bridgeman and D. Barter, Computing data for Levin-Wen with defects, *Quantum* **4**, 277 (2020).
- [67] T. D. Ellison, Y.-A. Chen, A. Dua, W. Shirley, N. Tantiwasadakarn, and D. J. Williamson, Pauli stabilizer models of twisted quantum doubles, *PRX Quantum* **3**, 010353 (2022).
- [68] N. Delfosse, P. Iyer, and D. Poulin, Generalized surface codes and packing of logical qubits, *ArXiv:1606.07116* (2016).
- [69] A. Benhemou, J. K. Pachos, and D. E. Browne, Non-Abelian statistics with mixed-boundary punctures on the toric code, *Phys. Rev. A* **105**, 042417 (2022).
- [70] Bryan Eastin and Emanuel Knill, Restrictions on transversal encoded quantum gate sets, *Phys. Rev. Lett.* **102**, 110502 (2009).
- [71] C. Gidney, M. Newman, A. Fowler, and M. Broughton, A fault-tolerant honeycomb memory, *Quantum* **5**, 605 (2021).
- [72] C. Vuillot, Planar Floquet Codes, *arXiv:2110.05348* (2021).
- [73] A. Paetznick, C. Knapp, N. Delfosse, B. Bauer, J. Haah, M. B. Hastings, and M. P. da Silva, Performance of planar Floquet codes with Majorana-based qubits, *PRX Quantum* **4**, 010310 (2023).
- [74] 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).
- [75] H. Bombin, Dimensional jump in quantum error correction, *New J. Phys.* **18**, 043038 (2016).
- [76] B. Eastin and E. Knill, Restrictions on transversal encoded quantum gate sets, *Phys. Rev. Lett.* **102**, 110502 (2009).
- [77] S. Bravyi and R. König, Classification of topologically protected gates for local stabilizer codes, *Phys. Rev. Lett.* **110**, 170503 (2013).
- [78] F. Pastawski and B. Yoshida, Fault-tolerant logical gates in quantum error-correcting codes, *Phys. Rev. A* **91**, 012305 (2015).
- [79] P. Webster and S. D. Bartlett, Fault-tolerant quantum gates with defects in topological stabilizer codes, *Phys. Rev. A* **102**, 022403 (2020).
- [80] P. Webster, M. Vasmer, T. R. Scruby, and S. D. Bartlett, Universal fault-tolerant quantum computing with stabilizer codes, *Phys. Rev. Res.* **4**, 013092 (2022).
- [81] S. Bravyi and A. Kitaev, Universal quantum computation with ideal Clifford gates and noisy ancillas, *Phys. Rev. A* **71**, 022316 (2005).
- [82] B. J. Brown, Conservation laws and quantum error correction: Toward a generalized matching decoder, *IEEE BITS the Information Theory Magazine* **2**, 5 (2022).
- [83] S. Dong, E. Fradkin, R. G. Leigh, and S. Nowling, Topological entanglement entropy in Chern-Simons theories and quantum Hall fluids, *J. High Energy Phys.* **2008**, 016 (2008).
- [84] B. J. Brown, S. D. Bartlett, A. C. Doherty, and S. D. Barrett, Topological entanglement entropy with a twist, *Phys. Rev. Lett.* **111**, 220402 (2013).
- [85] P. Bonderson, C. Knapp, and K. Patel, Anyonic entanglement and topological entanglement entropy, *Ann. Phys.* **385**, 399 (2017).
- [86] T. J. Yoder and I. H. Kim, The surface code with a twist, *Quantum* **1**, 2 (2017).
- [87] D. Litinski and F. von Oppen, Lattice surgery with a twist: Simplifying Clifford gates of surface codes, *Quantum* **2**, 62 (2018).
- [88] J. C. Magdalena de la Fuente, J. Eisert, and A. Bauer, Bulk-to-boundary anyon fusion from microscopic models, *J. Math. Phys.* **64**, 111904 (2023).

- [89] D. Litinski, A game of surface codes: Large-scale quantum computing with lattice surgery, *Quantum* **3**, 12 (2019).
- [90] D. Litinski, M. S. Kesselring, J. Eisert, and F. von Oppen, Combining topological hardware and topological software: Color-code quantum computing with topological superconductor networks, *Phys. Rev. X* **7**, 031048 (2017).
- [91] D. Poulin, Stabilizer formalism for operator quantum error correction, *Phys. Rev. Lett.* **95**, 230504 (2005).
- [92] B. J. Brown and D. J. Williamson, Parallelized quantum error correction with fracton topological codes, *Phys. Rev. Res.* **2**, 013303 (2020).
- [93] O. Higgott, PyMatching: A PYTHON package for decoding quantum codes with minimum-weight perfect matching, *ACM Transactions on Quantum Computing* **3**, 1 (2022).
- [94] S. B. Bravyi and A. Y. Kitaev, Quantum codes on a lattice with boundary, *ArXiv:quant-ph/9811052* (1998).
- [95] D. Aasen, D. Bulmash, A. Prem, K. Slagle, and D. J. Williamson, Topological defect networks for fractons of all types, *Phys. Rev. Res.* **2**, 043165 (2020).
- [96] H. Bombin, Gauge color codes: Optimal transversal gates and gauge fixing in topological stabilizer codes, *New J. Phys.* **17**, 083002 (2015).
- [97] N. P. Breuckmann and J. N. Eberhardt, Quantum low-density parity-check codes, *PRX Quantum* **2**, 040101 (2021).
- [98] H. Bombin, Topological subsystem codes, *Phys. Rev. A* **81**, 032301 (2010).
- [99] O. Higgott and N. P. Breuckmann, Subsystem codes with high thresholds by gauge fixing and reduced qubit overhead, *Phys. Rev. X* **11**, 031039 (2021).
- [100] A. M. Steane, Active stabilization, quantum computation, and quantum state synthesis, *Phys. Rev. Lett.* **78**, 2252 (1997).
- [101] E. Knill, Quantum computing with realistically noisy devices, *Nature* **434**, 39 (2005).
- [102] D. K. Tuckett, S. D. Bartlett, and S. T. Flammia, Ultrahigh error threshold for surface codes with biased noise, *Phys. Rev. Lett.* **120**, 050505 (2018).
- [103] J. P. Bonilla Ataides, D. K. Tuckett, S. D. Bartlett, S. T. Flammia, and B. J. Brown, The XZZX surface code, *Nat. Commun.* **12**, 2172 (2021).
- [104] A. Dua, A. Kubica, L. Jiang, S. T. Flammia, and M. J. Gullans, Clifford-deformed surface codes, *ArXiv:2201.07802* (2022).
- [105] B. Srivastava, A. Frisk Kockum, and M. Granath, The  $XYZ^2$  hexagonal stabilizer code, *Quantum* **6**, 698 (2022).
- [106] J. F. San Miguel, D. J. Williamson, and B. J. Brown, A cellular automaton decoder for a noise-bias tailored color code, *Quantum* **7**, 940 (2023).
- [107] C. Chamberland, K. Noh, P. Arrangoiz-Arriola, E. T. Campbell, C. T. Hann, J. Iverson, H. Putterman, T. C. Bohdanowicz, S. T. Flammia, A. Keller, G. Refael, J. Preskill, Liang Jiang, A. H. Safavi-Naeini, O. Painter, and F. G. S. L. Brandão, Building a fault-tolerant quantum computer using concatenated cat codes, *PRX Quantum* **3**, 010329 (2022).
- [108] D. K. Tuckett, S. D. Bartlett, S. T. Flammia, and B. J. Brown, Fault-tolerant thresholds for the surface code in excess of 5% under biased noise, *Phys. Rev. Lett.* **124**, 130501 (2020).
- [109] P. Aliferis and B. M. Terhal, Coping with qubit leakage in topological codes, *Quantum Inf. Comput.* **7**, 139 (2007).
- [110] M. Suchara, A. W. Cross, and J. M. Gambetta, Leakage suppression in the toric code, *Quantum Inf. Comput.* **15**, 997 (2015).
- [111] J. Ghosh and A. G. Fowler, Leakage-resilient approach to fault-tolerant quantum computing with superconducting elements, *Phys. Rev. A* **91**, 020302 (2015).
- [112] I. Cong, M. Cheng, and Z. Wang, Universal quantum computation with gapped boundaries, *Phys. Rev. Lett.* **119**, 170504 (2017).
- [113] K. Laubscher, D. Loss, and J. R. Wootton, Universal quantum computation in the surface code using non-Abelian islands, *Phys. Rev. A* **100**, 012338 (2019).
- [114] GitHub repository, <https://github.com/BenjaminJamesBrown/FloquetColorCode>.
- [115] Technically, one of the phases needs to be inverted. But for stabilizer codes, as considered here, the inversion does not change the phase.
- [116] X. Chen, Z.-C. Gu, Z.-X. Liu, and X.-G. Wen, Symmetry protected topological orders and the group cohomology of their symmetry group, *Phys. Rev. B* **87**, 155114 (2013).
- [117] A. Coste, T. Gannon, and P. Ruelle, Finite group modular data, *Nucl. Phys. B* **581**, 679 (2000).