

Fault-Tolerant Operations in Majorana-Based Quantum Codes: Gates, Measurements, and High-Rate Constructions

Maryam Mudassar,^{1,*} Alexander Schuckert^{1,2}, and Daniel Gottesman¹

¹*Joint Center for Quantum Information and Computer Science, University of Maryland and NIST, College Park, Maryland, USA*

²*Joint Quantum Institute, University of Maryland and NIST, College Park, Maryland, USA*

 (Received 16 September 2025; revised 9 January 2026; accepted 25 March 2026; published 24 April 2026)

Majorana-based quantum computation in nanowires and neutral atoms has gained prominence as a promising platform to encode qubits and protect them against noise. In order to run computations reliably on such devices, a fully fault-tolerant scheme is needed for state preparation, gates, and measurements. However, current fault-tolerant schemes have either been limited to specific code families or have not been developed fully. In this work, we extend the tools of qubit fault tolerance to Majorana codes. We emphasize the division between even and odd Majorana codes and how it manifests when constructing fault-tolerant gadgets for these families. We provide transversal constructions and supplement them with measurements to obtain several examples of fault-tolerant Clifford gadgets. For the case of odd codes, we give a construction for gadgets using quantum reference frames that allows one to implement operations that are forbidden due to parity superselection. We also provide a fault-tolerant measurement scheme for Majorana codes inspired by Steane error correction, enabling state preparation, measurement of logical operations, and error correction. We also point out a construction for odd Majorana codes with transversal T gates. Finally, we construct a high-rate quantum low density parity check codes (LDPC) Majorana code with logical qubits. Our work shows that all necessary elements of fault-tolerant quantum computation can be consistently implemented in fermionic hardware such as Majorana nanowires and fermionic neutral atoms.

DOI: [10.1103/bglx-ssjg](https://doi.org/10.1103/bglx-ssjg)

I. INTRODUCTION

Quantum computation is inherently susceptible to errors, and significant effort has gone into developing fault-tolerant schemes, all of which fundamentally rely on quantum error correction. However, a persistent challenge is that the physical error rates of many existing platforms remain above the fault tolerance threshold. Ongoing advances in hardware—including improvements in coherence times and gate fidelities—aim to address this. At the same time, topological quantum computing is a complementary approach that seeks to build hardware with intrinsic protection against local errors. One of the earliest examples of topological protection is the theory of Majorana zero modes (MZMs). It is believed that MZMs form at a superconductor-semiconductor interface, and any information encoded into these quasiparticles is immune to

local noise. Although progress in the discovery of MZMs has been challenging, there have been some promising efforts [1,2] and many clear proposals on their implementation [3,4]. It has been believed that noise such as local decoherence is exponentially suppressed in these systems, and topological operations such as braiding can be used to apply gates on the logical data [5]. Majorana zero modes also correspond to Ising anyons [6], and it is well known that Ising anyons are not universal. In order to do universal computing, other topologically unprotected schemes are required [7]. Moreover, existing measurement-based schemes still suffer from other noise processes [8] and require active error correction [4].

The necessity of performing active error correction on these Majorana modes opens up the field of Majorana error correction or Majorana codes. Majorana codes are stabilizer codes that encode logical qubits or logical fermionic degrees of freedom, with physical Majorana degrees of freedom. Since their introduction [9], several codes have emerged, such as small codes [10], Majorana color codes [11–13], and Majorana surface codes [14–16] to name a few. Some of these codes have been optimized to allow local low-weight measurements [15,17] or to protect against quasiparticle errors [18]. The structure of the

*Contact author: maryammu@umd.edu

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

Majorana Clifford group has also been explored recently [19–21], which allows one to perform Clifford operations on such codes, thus enabling code initialization and logical gates.

In the context of qubits based on Majorana nanowires, previous work [4] has focused on encoding qubits into small superconducting islands composed of two or three nanowires, and then concatenating these small Majorana codes with known qubit codes such as surface codes or Floquet codes. Although these designs are valuable since they can use existing techniques of fault-tolerant operations on qubit codes, they do not fully exploit the benefits of general Majorana codes. Moreover, qubit-based codes are generally only tailored to Pauli noise models and would not be able to protect against native fermionic noise. In nanowire-based systems, quasiparticle poisoning can also cause gates to fail and measurements to become faulty, so topological protection no longer suffices. Other approaches to combat quasiparticle poisoning rely on fixing the charging energy of some “islands” in Refs. [3,11], but these depend heavily on the hardware and code architecture. It is worthwhile to have a general toolkit of fault-tolerant protocols and then to optimize them later given experimental and code specifications.

Previous studies have developed fault-tolerant gate implementations for Majorana color codes that utilize code deformations and lattice surgery techniques [12,13,22], supplemented with magic state distillation [23]. However, lattice surgery techniques require d rounds of syndrome extraction, which can become expensive quickly.

Hence, it is extremely valuable to explore other means of performing fault-tolerant gates, such as transversal implementations.

There have also been recent breakthroughs in manipulating fermionic neutral atom arrays in 2D [24–26]. It has also been proposed that logical information can be encoded into an error-corrected subspace of these programmable arrays [27,28]. In the absence of number conservation, these codes have been shown to be Majorana codes, and this observation has reinvigorated interest in the field of fermionic codes.

This work extends the study of fault tolerance in Majorana codes by extending many of the standard techniques of qubit fault tolerance to Majorana codes. This can be useful on platforms that have a native fermionic architecture. Majorana codes can have logical operators that fulfill the Pauli algebra between different blocks and therefore correspond to logical qubit operators; these codes have been referred to as “even” Majorana codes [9] because they are even weight in physical Majorana operators. The other kind of codes, which have odd-physical-Majorana-weight logical operators, are called “odd” codes. Odd-weight operators fulfill fermionic anticommutation relations and are therefore called “logical fermion operators.” We will frequently refer to logical qubit/fermion operators simply

as “logical qubits/fermions.” Odd codes have the interesting property that one can encode logical qubits or logical fermions, depending on the mutual (anti) commutations. The original work on Majorana codes [9] introduced even/odd codes in Majoranas, but odd codes and their ability to encode logical fermions has only garnered attention recently [27,28] and is apparent only when considering multiple code blocks. We build on this earlier work and introduce more techniques that we hope would be valuable for these platforms. We note that odd codes can have multiple logical degrees, and one can **choose** which of these corresponds to fermions, and which are qubits. This makes Majorana codes more versatile for early fault-tolerant algorithms that require a mixture of fermion and qubit interactions. Note that in the usual sense of quantum codes, having this mixed logical degree of freedom is only featured in fermionic codes. However, logical fermions have been discussed previously as “twist defects” in qubit surface codes [29] and logical fermions and logical qubits as twist defects in Majorana surface codes [12,22]. However, performing gates on these codes requires “lattice surgery,” which can be costly. In our work, we concern ourselves primarily with transversal gates and supplement them with mode permutations and measurements, which allows us to standard Clifford and non-Clifford operations on logical qubits. Moreover, we use the notion of a quantum reference system to create analogous **fermionic** versions of these gates, such as fermionic Hadamard, fermionic CNOT, and generate fermionic magic states. Last but not the least, we give a construction for generating even Majorana LDPC codes. We became aware of another independent work [30] that shows examples for odd Majorana codes.

Sec. II gives background on Majorana codes and known mappings to qubit codes, fermionic Clifford operations and implementation of fermionic operations in terms of measurements. Section II B describes the model of fault tolerance in qubit codes, and Sec. II C describes the same for Majorana codes, which is motivated from physical error sources in both neutral atom and nanowire architectures. We do not assume a specific code family; rather we divide them into two broad classes: even Majorana codes and odd Majorana codes in Sec. III and explore their structure and limitations. In Sec. IV, we explain the idea of reference systems that we use to carry out Clifford and non-Clifford logical gates on odd Majorana codes throughout the paper. In Sec. V, we discuss transversal Clifford operations and supplement them with ancilla measurements to realize a fault-tolerant set of Clifford gates in Sec. VI. For fault-tolerant error correction, we describe a Steane-inspired measurement gadget for Majorana codes in Sec. VII. In Sec. VIII, we describe a code that has transversal T gate that can be useful in magic state distillation in a platform where one only has access to fermion-fermion Clifford operations. Finally, we discuss Majorana Calderbank-Shor-Steane (CSS) and

Majorana qLDPC codes in Secs. IX and X, respectively. We end with a discussion and possible future directions.

A. Comparison with fermion to qubit approaches

The approach in our work is to treat the logical degrees as either qubits or fermions, depending on the commutation relations, while the physical degrees are Majorana (real) fermions or physical (complex) fermions. The standard approach in digital simulation is to have physical qubits. In order to perform error correction, logical qubits are encoded in physical fermions and fermion-to-qubit mappings are commonly used to convert the fermion operators into encoded qubit operators. Therefore, the usual approach treats the fermion-to-qubit mapping independent of the error correction paradigm. There are various mappings that can be utilized and are broadly classified as linear encodings [31] (e.g., Jordan-Wigner, Bravyi-Kitaev, and ternary tree encoding) where the number of qubits are the same as the number of fermions, but local fermion terms generally correspond to nonlocal qubit terms. This nonlocality is undesirable even without error correction as the depth of the simulation algorithm grows with the weight of the simulated term.

There are some proposals for dealing with this expensive fermion-to-qubit mapping cost when doing error correction, and they mostly involve encoding logical fermions into twist defects of a surface code [29,32]. While this approach is promising, it is limited by the qubit surface code in its encoding rate and also does not allow for multiple code blocks, so it is valuable to consider alternate methods of fault-tolerant fermion simulation.

The fermion-to-qubit mapping creates an overhead. While recently, it was shown that this depth-overhead can be reduced to $O(\log^2 N)$ even without ancillas [33,34], this overhead has a large prefactor which in practice could prove prohibitive. It is important to note that this overhead is only in Clifford gates, at least on first sight, the cost of performing T gates is the same in qubit or fermionic algorithms [27].

Another important feature in considering fermionic codes is that fermionic codes are tailored to native fermionic noise. Qubit codes are designed with the idea of having logical qubits, so an independent local noise channel on physical qubits leaves a residual local error on the logical qubits. However, if the logical information consists of fermions, it is unlikely that a local qubit error would cause a similarly local residual fermion error [35].

II. BACKGROUND

A. Majorana fermions

Majorana codes are designed for architectures with physical fermionic degrees of freedom. They are defined

in terms of $2n$ Majorana operators:

$$\gamma_i = a_i + a_i^\dagger, \quad (1)$$

$$\bar{\gamma}_i = -i(a_i - a_i^\dagger), \quad (2)$$

where $a_i, a_i^\dagger$ are the fermionic creation and annihilation operators fulfilling $\{a_i, a_i^\dagger\} = 1$. Majoranas have anticommutation relations

$$\{\gamma_i, \gamma_j\} = 2\delta_{ij}, \quad \gamma_i^2 = I, \quad (3)$$

$$\{\gamma_i, \bar{\gamma}_j\} = 0. \quad (4)$$

One can define a group $\text{Maj}(2n)$ for $2n$ Majorana modes, analogous to the Pauli group P_n , where the group is generated by Majorana monomials:

$$\text{Maj}(2n) = i^r \langle \gamma_1^{c_0} \gamma_2^{c_1} \cdots \gamma_n^{c_{n-1}} \bar{\gamma}_1^{c_n} \bar{\gamma}_2^{c_{n+1}} \cdots \bar{\gamma}_n^{c_{2n-1}} \rangle, \quad (5)$$

$$\text{with } r \in \{0, 1, 2, 3\}, \quad c_i \in \{0, 1\}. \quad (6)$$

Similar to the Pauli group, the order of this group is 2^{2n+1} . In MZM-based architectures, each physical Majorana mode can be individually manipulated using gates, or Majorana bilinears can be measured. Hence logical states, which are commonly qubit states, can be encoded into these modes and controlled.

While Majorana zero modes and their manipulation was the original motivation for Majorana codes, it is interesting to note that these codes also bear value for fermionic neutral atoms. In neutral atoms, fermionic sites are realized by sites of an optical lattice or a grid of optical tweezers [25,27]. Each of these sites can either be occupied by a fermionic atom (encoding the state $|1\rangle$) or not (encoding the state $|0\rangle$). Destroying or creating a fermion is then denoted with the complex fermionic operators a_i and $a_i^\dagger$, respectively, and Majorana fermions are encoded into the real and imaginary parts of these complex fermionic operators. Similarly, electrons in dots and donors in silicon realize complex fermions and can be braided using tunneling [36] and pairing induced by the proximity effect with a superconductor. Moreover, qubit-fermion gates with hyperfine qubits in these platforms yield further flexibility [27,37].

It is worth pointing out that the normal physical setting for these systems corresponds to a particle number conservation constraint. More precisely, this means that the physical Hamiltonian H_f commutes with the total fermion number N_{tot} , which is a *symmetry*, and hence H_f can be written in a block diagonal form, and similarly, the Hilbert space can also be written in this block diagonal basis

$$\mathcal{H}_f = \bigoplus_{i=0}^N \mathcal{H}_i. \quad (7)$$

Moreover, the native unitaries are also considered to be *number-conserving* operations. However, this is not an essential constraint, and people have shown proposals where the number-conserving constraint can be lifted, either using reference systems [28] or by engineering specific **pairing** gates [27]. Doing so changes the symmetry from $S = N_{\text{tot}}$ to $S = P_{\text{tot}}$, so H_f is instead block diagonal in the total parity basis, and the Hilbert space $\mathcal{H}_f$ also becomes a Z_2 graded space. For our purpose, this means that we can use Majorana modes to represent physical degrees of freedom instead of fermions and apply parity-preserving unitaries to our system.

B. The model of fault tolerance

We first briefly define the basic model of fault tolerance for qubits [38], to set the stage for Majorana fault tolerance later. In the basic model, an ideal quantum circuit C is divided into the following location types: preparation locations, gate locations (which can have 1-qubit and 2-qubit gate locations), wait locations, and measurement locations. The ideal circuit C is realized by a physical circuit C' , in which some locations realize the correct circuit element, while others perform an incorrect operation, and are known as *faulty* locations.

In the basic model of fault tolerance, each location faces an error given by an *independent stochastic Pauli* error model. This means that given a location L in C' , there is probability p_L of a Pauli error occurring (making it a faulty location), and probability $1 - p_L$ for no error at this location.

If the location involves two qubits, the error could be a two-qubit Pauli error. The precise distribution over Pauli errors varies depending on the model being studied; depolarizing noise, in which all nonidentity Paulis are equally likely, is a common choice. Because a single fault can cause errors on two qubits, the errors being introduced are actually *correlated*. When we call this an independent error model, the meaning is that errors are independent between locations and not necessarily between qubits.

Typically, we assume that the error rate is the same for all locations of the same type, and we assume errors are independent of different locations. For instance, if the preparation location has an error rate p_P , then the probability of having faults in both of two preparation locations is p_P^2 . Generally, each location in a noisy circuit C' could have a fault, but the probability of multiple independent faults occurring is quite small, so most fault-tolerant circuits are analyzed with a small number of faults per block.

One major concern in designing fault-tolerant circuits is the possibility of error *propagation*. In particular, once an error has occurred, the operation of the circuit (even if there are no further faulty locations) can spread the effects of the error. For instance, if a qubit A has an X error in it but then a CNOT gate acts with qubit A as the control and

qubit B as the target, then after the CNOT gate, both qubits A and B now have bit flip errors.

The objective of fault-tolerant circuit design is that a noisy circuit C' simulates the ideal circuit C , while controlling error propagation. To this end, each circuit component in C is simulated by a gadget in C' that performs the correct operation on the logical qubits while limiting error propagation in acceptable ways.

The first step on the road to fault tolerance is *transversal* gadgets: each k -qubit operation is replaced by a k -block operation, where each individual physical multi-qubit gate in the gadget acts only between corresponding qubits in different blocks. Doing so limits the propagation of errors, because if a single fault occurs before the gadget, it spreads to k different code blocks with only one error per block, which can still be corrected by the code. It is impossible to design a set of universal gate gadgets that are implemented transversally, because of the Eastin-Knill theorem [39]. The Eastin-Knill theorem states that the transversal gate group G for a quantum code is discrete, which implies that it cannot be dense in $SU(2)$. To perform a universal set of gates, one can do magic state injection along with some transversal gates if the code is fixed or use code switching to switch between codes and use the transversal gadgets of multiple codes.

C. Fault tolerance in Majorana systems

For Majorana hardware, most of the fault tolerance setup is similar, except that we will use a Majorana noise model rather than a Pauli noise model. For a stochastic [40] Majorana noise model, time is discretized into finite steps τ , and at each time step, the state undergoes a Majorana monomial error

$$|\psi\rangle \rightarrow \gamma_1^{a_1} \gamma_2^{a_2} \dots \bar{\gamma}_n^{a_{2n}} |\psi\rangle. \quad (8)$$

Analogous to the Pauli case, a Majorana circuit can also be divided into locations. However, contrary to a Pauli error model where each location sees experiences a noise event with probability p , a single noise event can affect multiple Majorana modes simultaneously and cause a *correlated* error between locations, but they are usually only 2-local, so the locality assumption still holds true for a stochastic error model.

Physical fermionic errors always act on two Majorana sites. A *hopping* error between two fermionic sites, e.g., $a_i^\dagger a_j + h.c$ affects two Majorana modes on two different sites, e.g., $\gamma_i \gamma_j$, while a *phase* error $a_i^\dagger a_i$ acts on two Majorana modes at the same site $\gamma_i \bar{\gamma}_i$. For the purposes of error-correction and fault-tolerance, it is easier to treat all γ_i and $\bar{\gamma}_i$ on the same footing, even though the physical origin of noise may be different.

The first example of Majorana noise is *quasiparticle poisoning* or *quasiparticle* errors. These can occur when

the system modes interact with a fermionic bath and exchange fermions. Alternatively, since we consider multiple blocks here, a particle can also hop between two code blocks when they interact via a gate or measurement. This produces a *parity violating* error in a code block, and can be detected/corrected by making parity measurements, which may or may not always be possible depending on the code choice and measurement restrictions, but there are some techniques to combat such quasiparticle [18] errors. Another approach to control such errors is by restricting some subset of the stabilizers [11,40] to have a fixed value using high charging energy. Finally, an earlier work [40] remarks that a low quasiparticle poisoning rate changes a Majorana noise model to a qubit noise model. In the current context of multiple blocks, since we do not assume hardware constraints, quasiparticle error rates are still non-negligible and we address recovery schemes in Secs. II F and VII. Other than quasiparticle poisoning, Majorana locations also suffer from *dephasing noise*. Dephasing noise corresponds to when the same fermion mode experiences a Majorana bilinear $\gamma_i \bar{\gamma}_i$ error. A two-mode error in the same physical fermion mode is mathematically equivalent to a single Pauli error.

Finally, *correlated* errors can also stem from a Majorana gate that acts on multiple modes, as a single noise event can propagate to errors on multiple modes, which may or may not change the parity of the mode. This is also analogous to circuit level noise in qubits, and transversal approaches work well to keep errors contained in a certain code block. In other schemes such as in Ref. [4], measurements are chosen to not propagate these errors.

To add measurement noise, a measurement noise vector, $\mathbf{b} = (b_1, b_2 \dots b_n)$ is added to the measurement vector $\mathbf{m} = (m_1, m_2 \dots m_n)$. To correct for measurement noise, one can use classical decoding. We demonstrate this for fault-tolerant error correction in Sec. VII.

The error model of Majorana (nanowires) can be extended to fermionic neutral atoms as well [25,27,28]: quasiparticle poisoning happens when an atom is lost, which realizes a noise operator a_i which can be expanded in terms of γ_i and $\bar{\gamma}_i$. Dephasing noise is induced by laser fluctuations of the optical lattice and optical tweezers and those errors may be correlated.

Majorana codes also suffer from similar restrictions from Eastin-Knill as qubit codes. To see this, we can use code conversion: any Majorana code can be converted to a qubit CSS code. A transversal Majorana gadget preserves the Majorana stabilizer group, so the corresponding Pauli gadget will also preserve the Pauli stabilizer and thus remain transversal on the Pauli group. Since the group of transversal gates is restricted by Eastin-Knill on qubits, and gates obtained via Majorana to qubit mapping are strictly a subset of qubit transversal gates, this implies that transversal Majorana gates are also limited by Eastin-Knill. This motivates a desire for native magic

state distillation protocols for Majoranas, which we show in Sec. VIII.

D. Majorana codes

A Majorana stabilizer code is defined by specifying the stabilizer group as the Abelian subgroup of $S_{\text{Maj}} \subseteq \text{Maj}(2n)$, s.t. $-I \notin S_{\text{Maj}}$. A Majorana stabilizer code on $2n$ modes corresponds to n physical fermions, k logical fermions, has $n - k$ stabilizer generators, and the code distance d is the minimum weight of the smallest logical operator. As a result, a Majorana code can also be written as $[[n, k, d]]_f$. The notation is worth pausing at, since it is not standard across all Majorana code papers. This is because while there are $2n$ modes, one needs to “halve” them to consider independent modes, and so the number of stabilizer generators r are $n - k$ not $2n - k$. However when considering the distance, the Hamming weight of the Majorana bit string is considered, so there is some “overcounting.” As a reason, it may appear that these codes violate some well-known bounds known for qubit codes, such as quantum Hamming bound, but this is because the parameter in Majorana codes count differently. Sometimes, an additional parameter called l_{even} is also defined since Majorana codes do not treat even and odd weight logical operators on the same footing, but in a fault tolerance context, this is not a good metric, see more discussion in Sec. V. The logical operators for such codes may have even or odd weight, depending on whether $P_{\text{tot}} \in S$ or not, see more discussion in Sec. III.

Similar to the Pauli group, the Majorana group can be converted into a symplectic vector space with the symplectic form Λ_f [41],

$$\Lambda_f = I + C, \quad (9)$$

where I is the identity matrix and C is a constant $2n \times 2n$ matrix of ones. One can then find whether Majorana operators commute by finding their symplectic inner product defined as

$$\mathbf{x} \odot \mathbf{z} = \mathbf{x} \Lambda_f \mathbf{z}^T \bmod 2, \quad (10)$$

where $\mathbf{x}$ and $\mathbf{z}$ correspond to length $2n$ Majorana binary strings. In general, the commutation relation for two Majorana binary strings $\mathbf{x}, \mathbf{z}$ is

$$\text{If } |\mathbf{x}| |\mathbf{z}| + \mathbf{x} \cdot \mathbf{z} \bmod 2 = 0, \text{ they commute,} \quad (11)$$

$$\text{If } |\mathbf{x}| |\mathbf{z}| + \mathbf{x} \cdot \mathbf{z} \bmod 2 = 1, \text{ they anticommute.} \quad (12)$$

In this paper, we make use of both odd and even Majorana strings, and the commutation rules for these are summarized as

- (1) If two Majorana strings have even weight, they commute if they overlap on an even number of sites, and anticommute otherwise.

- (2) If one Majorana string is even and the other is odd, they commute if they overlap on an even number of sites, and anticommute otherwise.
- (3) If two Majorana strings have odd weight, they commute if they overlap on an odd number of sites, and anticommute otherwise.

The *parity superselection rule* is also an important constraint to take into account when considering Majorana codes. It states that one cannot create a state, which admits a coherent superposition of an even and odd number of fermions. This means that all physical operations with respect to some *frame of reference* must preserve the parity of the system. More formally, any quantum state ρ can be decomposed as [42]

$$\rho = \rho_e \oplus \rho_o, \quad (13)$$

such that $P_{\text{tot}}\rho_e P_{\text{tot}}^\dagger = \rho_e$ and $P_{\text{tot}}\rho_o P_{\text{tot}}^\dagger = \rho_o$. Other observables and unitaries must also admit a direct sum structure with respect to the fermionic Hilbert space, which entails that they must commute with the total parity (see Appendix A for proof). In particular, this constrains the stabilizer generators: they must commute with the total fermion parity, i.e., $P_{\text{tot}} = i^n \prod_{i=1}^n \gamma_i \bar{\gamma}_i$ since they correspond to physical, measurable operators. However, it is important to recognize that the superselection rule is defined with respect to a reference, and one can use *reference systems* [43] to relax this constraint.

In general, logical operators may or may not commute with P_{tot} . If $P_{\text{tot}} \in S$, logical operators have to commute with P_{tot} and must therefore be even weight in the physical Majorana operators. Such codes are known as *even* codes. Codes that do not contain $P_{\text{tot}} \in S$ have at least one logical operator which is odd weight [9]. Such codes are called *odd* codes. As an example for an *odd* code, consider the 1D Kitaev chain on $2n$ Majoranas. One can encode one logical fermion into the Majorana zero modes as follows:

$$\Gamma_1 = \gamma_1 \quad \bar{\Gamma}_1 = \bar{\gamma}_1, \quad (14)$$

where we denoted the logical Majorana operators with Γ_1 and $\bar{\Gamma}_1$. Here the code parameters are $[[n, 1, 1]]_f$. Note that while the distance is one, the Kitaev chain can still detect and correct all local errors that occur in the bulk of the chain.

One can also combine two Kitaev chains to obtain an *even* code, also known as a Majorana tetron code [11]. Because even-weight operators commute between different code blocks, this enables encoding a logical *qubit* by $\bar{X} = i\gamma_1\gamma_2$ and $\bar{Z} = i\gamma_1\bar{\gamma}_1$ and introducing an additional parity constraint $S = \gamma_1\bar{\gamma}_1\gamma_2\bar{\gamma}_2$. We give a longer discussion of even and odd codes in Sec. III.

Majorana codes can be mapped to qubit CSS codes [9] and vice versa, and these mappings are often a useful tool

when finding Majorana codes. Moreover, classical weakly self-dual codes can also be converted to Majorana codes [44]. We review these as follows.

Lemma 1 (Vijay et al. [44]). Any classical weakly self-dual $[2n, k, d]$ code can be converted to a Majorana code $[[n, n - k, d^\perp]]$, where $d^\perp$ is the dual distance.

Short proof. For a classical code to be weakly self-dual, $C \subseteq C^\perp$, implying $GG^T = 0$, where G is the generator matrix of the classical code. For the Majorana code, the generator matrix G can be converted to the binary matrix of the stabilizer for the Majorana code. The number of physical fermions is half the number of classical bits, the rows of generator matrix become the rows of the stabilizer, hence the number of logical degrees of freedom is $n - k$. The distance of the code is the weight of smallest element in $N(S)/S$, which is the weight of the smallest element in $H \setminus G$. If the code is nondegenerate, as in the case considered in Ref. [44], then $d^\perp$ is the code distance, else $d \geq d^\perp$. ■

Lemma 2 (Bravyi et al. [9]). Any Majorana code $[[n, k, d]]_f$ can be mapped to a qubit weakly self-dual CSS code $[[2n, 2k, d]]$.

Short proof. Set $H_x = S$, $H_z = S$ for the quantum CSS code, where S is the stabilizer of the Majorana code. $H_x H_z^T = 0$ since $HH^T = 0$. The obtained qubit code has support on $2n$ qubits and has $2n - 2k$ generators. Distance remains the same since any vector in $C^\perp \setminus C$ corresponds to logical operators in both the Majorana code and the qubit CSS code. ■

Note that Lemmas 1 and 2 can be combined to obtain a weakly self-dual CSS code from a classical code, as outlined in Ref. [45]. Majorana codes can also be obtained from qubit CSS codes.

Lemma 3 (Bravyi et al. [9]). Any qubit weakly self-dual CSS code $[[n, k, d]]_f$ can be converted to a Majorana code $[[n, k, d]]_f$.

Short proof. Weakly self-dual for a CSS code implies $H_x = H_z = H$. Set $H_\gamma = H$ and $H_{\bar{\gamma}} = H$. This is a weakly self-dual Majorana CSS code. All the parameters are identical since all we have done is just map X to γ and Z to $\bar{\gamma}$. ■

Throughout the paper, we will be using these Lemmas extensively.

E. Majorana Clifford gates

Majorana string operators are defined as

$$M = i^r \gamma_1^{c_1} \dots \gamma_n^{c_n} \bar{\gamma}_1^{c_{1+n}} \dots \bar{\gamma}_n^{c_{2n}},$$

$$r = \{0, 1, 2, 3\}, c_i = \{0, 1\}. \quad (15)$$

We can therefore represent a Majorana operator up to phases as a binary string $\mathbf{c} \equiv (c_1, c_2 \dots c_{2n})$. In analogy to qubit Clifford gates, fermionic Clifford gates are defined as gates, which map one Majorana string to a single other Majorana string.

Our two elementary Clifford gates are BRAID₂

$$\text{BRAID}_2(\bar{c}) = \exp\left(-\frac{\pi}{4}\bar{c}\right), \quad \text{wt}(\bar{c}) = 2, \quad (16)$$

and BRAID₄

$$\text{BRAID}_4(\bar{v}) = \exp\left(i\frac{\pi}{4}\bar{v}\right), \quad \text{wt}(\bar{v}) = 4. \quad (17)$$

These gates act as

$$\begin{aligned} & \text{BRAID}_2(\bar{c})(\bar{m})\text{BRAID}_2(\bar{c})^\dagger \\ &= \frac{1}{2}(1 + (-1)^{\mathbf{c}\Lambda_f \mathbf{m}^T})\bar{m} \\ &+ \frac{1}{2}(-1 + (-1)^{\mathbf{c}\Lambda_f \mathbf{m}^T})\bar{c} \cdot \bar{m}, \end{aligned} \quad (18)$$

where $\Lambda_f = I + C$ and $\bar{v} \cdot \bar{m}$ denotes a product between the corresponding Majorana strings,

$$\begin{aligned} \text{BRAID}_4(\bar{v})[\bar{m}]\text{BRAID}_4(\bar{v})^\dagger &= \frac{i}{2}(1 - (-1)^{\mathbf{v}\Lambda_f \mathbf{m}^T})\bar{v} \cdot \bar{m} \\ &+ \frac{1}{2}(1 + (-1)^{\mathbf{v}\Lambda_f \mathbf{m}^T})\bar{m}. \end{aligned} \quad (19)$$

In a binary matrix form, BRAID₂($\bar{c}$) can be written as

$$\text{BRAID}_2(\bar{c})_{\alpha,\beta} = \begin{cases} 1, & \alpha = \beta, \alpha, \beta \notin \bar{c}, \\ 1, & \alpha \neq \beta, \alpha, \beta \in \bar{c}, \\ 0, & \text{otherwise,} \end{cases} \quad (20)$$

and BRAID₄($\bar{v}$) as

$$\text{BRAID}_4(\bar{v})_{\alpha,\beta} = \begin{cases} 1 & \alpha = \beta, \alpha, \beta \notin \bar{v}, \\ 1 & \alpha, \beta \in \bar{v}, \alpha \neq \beta, \\ 0 & \text{otherwise.} \end{cases} \quad (21)$$

The BRAID₄ gate can also be understood as the generator of the logical braid group. The braid group (also known

as the Artin braid group [46]) can be understood in terms of the $n - 1$ braid generators σ_i satisfying the relations

$$\sigma_i \sigma_j = \sigma_j \sigma_i \quad \text{for } |i - j| \geq 2, \quad (22)$$

$$\sigma_i \sigma_{i+1} \sigma_i = \sigma_{i+1} \sigma_i \sigma_{i+1}. \quad (23)$$

For the BRAID₂ gates, the above are known to be satisfied. To see that these relations are also satisfied by the BRAID₄ group, one can define a group of logical operators,

$$\langle \Gamma_1, \Gamma_2, \dots, \Gamma_k \rangle, \quad (24)$$

where $k = n/2$ such that

$$\begin{aligned} \text{BRAID}_4(\Gamma_1)\text{BRAID}_4^\dagger &= \Gamma_2 \\ \text{BRAID}_4(\Gamma_2)\text{BRAID}_4^\dagger &= -\Gamma_1. \end{aligned} \quad (25)$$

It can also be shown that they satisfy the following relations:

$$\sigma_i \sigma_j = \sigma_j \sigma_i \quad \text{for } |i - j| \geq 2, \quad (26)$$

$$\sigma_i \sigma_{i+1} \sigma_i = \sigma_{i+1} \sigma_i \sigma_{i+1}. \quad (27)$$

As an example, we can choose $\sigma_1 = \text{BRAID}_4(1, 2, 3, 4)$, $\sigma_2 = \text{BRAID}_4(2, 3, 4, 5)$ and $\sigma_3 = \text{BRAID}_4(5, 6, 7, 8)$, so it is obvious that two generators with indices two or more away do not overlap and hence commute. The proof for the second property is given in Appendix B. A general BRAID₄ generator can thus be constructed by conjugating a generator from the above list with BRAID₂ generators.

F. Implementation of the BRAID₂ and BRAID₄ gates

Throughout this paper, we will extensively use both BRAID₂ and BRAID₄ gates as the physical Clifford operations. In this section, we discuss how these gates may be implemented in specific hardware.

1. Majorana nanowires

In nanowires, measurement-based operations instead of unitary operations have been proposed to be the simplest to implement [4,14]. To this end, one may write both BRAID₂ and BRAID₄ gates in terms of projective measurements of 2-MZM or 4-MZM parities. A BRAID₂ gate can be expressed in terms of projectors, for example [14],

$$\Pi_{+1}^{(34)} \Pi_{+1}^{(13)} \Pi_{+1}^{(23)} \Pi_{+1}^{(34)} = \Pi_{+1}^{(34)} \otimes \text{BRAID}_2(1, 2), \quad (28)$$

where $\Pi_s^{ij} = \frac{1}{2}(\mathbb{1} + is\gamma_i\gamma_j)$.

Note that the procedure above uses two ancilla Majorana modes but disentangles them at the end. Similarly, to

implement a BRAID₄ gate, one can prepare an ancilla state in the zero state

$$i\gamma_4\gamma_5 |\bar{0}\rangle = |\bar{0}\rangle \quad (29)$$

so that one may implement a BRAID₄ gate using the following projective and corrective operations

$$2\text{BRAID}_2(2, 5)\Pi_{-1}^{(24)}\Pi_{+1}^{(0134)} = \Pi_{+1}^{(45)} \otimes \text{BRAID}_4(0, 1, 2, 3), \quad (30)$$

where $\Pi_{+1}^{(ijkl)} = \frac{1}{2}(1 + \gamma_i\gamma_j\gamma_k\gamma_l)$. In general, the BRAID₄ can be expressed as a product of projective operations and a corrective operation U_{yz} depending on the measurement outcomes,

$$\text{BRAID}_4(0123) |\psi\rangle = 2U_{yz}\Pi_y^{24}\Pi_z^{0134} |\psi\rangle. \quad (31)$$

Note that both operations reset the ancilla pair back to the original state. Therefore, unitary approaches that employ physical BRAID₂ and BRAID₄ gates are theoretically equivalent to approaches involving 2-MZM and 4-MZM mid-circuit measurements and one can be compiled exactly in terms of the other.

In Ref. [4], a measurement-based approach is proposed to perform single-qubit and two-qubit logical Clifford operations, but they depend on the ability to perform 2-MZM and 4-MZM measurements in a robust manner, along with using ancilla. To perform long-range gates, one often needs an ancilla resource state in a logical GHZ, which allows joint measurements of observables. Fault-tolerant measurements then require a lattice surgery-based approach, such as the one in highlighted in Ref. [11]. The focus of our investigation is to expand the gate set by using Clifford operations available to the codes and building transversal gadgets for these codes.

2. Analysis of BRAID₄ under measurement noise

As explained above, BRAID₄ gates in Majorana nanowires are constructed via projective measurements of parities, and the corrective operation depends on the measurement outcomes. However, if the measurement outcome that is obtained is flipped, then the correction will also be faulty. We can treat the flipped measurement outcomes on a case-by-case basis. We have the following cases:

- (1) Correct measurement outcome is $m = (y = -1, z = -1)$, error is $b = (y = 0, z = +2)$, observed vector is $m' = (y = -1, z = +1)$. The incorrect recovery U_{yz} results in gate $\text{BRAID}_4^\dagger(0123)$ being applied instead of $\text{BRAID}_4(0123)$.
- (2) Correct measurement outcome is $m = (y = +1, z = +1)$, error is $b = (y = -2, z = 0)$, observed vector

is $m' = (y = -1, z = +1)$. The incorrect recovery results in gate $\gamma_2\gamma_5 \cdot \text{BRAID}_4^\dagger(0123)$ being applied.

- (3) Correct measurement outcome is $m = (y = +1, z = -1)$, error is $b = (y = -2, z = +2)$, observed vector is $m' = (y = -1, z = +1)$. The incorrect recovery results in gate $\gamma_2\gamma_5 \cdot \text{BRAID}_4(0123)$ being applied.

For all of the above outcomes, the phase reversal from BRAID₄ to $\text{BRAID}_4^\dagger$ corresponds to a phase error on a single mode, that when the gate acts transversally, is identified by the corresponding code block.

Another possibility that can take place during measurement is that an even weight error acts on the ancilla and data block simultaneously, resulting in an odd weight error on the data block. That is the case for example 2 in the above measurement flip as well, where an odd error acts on a data block ($\gamma_2 \cdot$ phase error) and an odd error on the ancilla. This is also an example of how a single error in a multimode gate can cause the entire gate to fail. One solution is to check if the ancilla parity has flipped, and if so, flip the parity of the code blocks as well. Another is to wait for another event, such that parity of ancilla is reset to even. While this may not result in bringing the data code block to an even parity subspace, it brings us to the error model we consider here that between multiple data blocks the error is allowed to be even or odd so long as the collective error of all data blocks is even. This kind of error is similar to a leakage error, as the state leaves the computational subspace by getting correlated with ancilla states, but is restored to global even parity state.

3. Neutral atoms

In neutral atoms, all braiding operations can be implemented unitarily.

The BRAID₂ gate in neutral atoms is implemented in two different ways, depending on whether the Majorana fermions on the same site or two different sites are braided. The former is realized by a local phase gate $\exp(-i(\pi/2)n_i)$, where $n_i = a_i^\dagger a_i = \frac{1}{2}(1 - i\tilde{\gamma}_i\gamma_i)$. The latter is realized by decomposing the braiding gate into a tunneling and pairing gate [27]

$$\begin{aligned} & \exp\left(i\frac{\pi}{4}\tilde{\gamma}_i\gamma_j\right) \\ &= \exp\left(i\frac{\pi}{4}\left(a_i^\dagger a_j + \text{h.c.}\right)\right) \exp\left(i\frac{\pi}{4}\left(a_i^\dagger a_j^\dagger + \text{h.c.}\right)\right). \end{aligned} \quad (32)$$

The tunneling gate is most easily implemented by either lowering the barrier between two sites in an optical lattice or moving two tweezers close to each other, inducing tunneling [24,25]. The pairing gate is implemented by using bosonic molecules in a Bose-Einstein condensate, which

provide a reservoir for creating and destroying fermion pairs [27].

The BRAID₄ gate in neutral atoms is implemented using the density-density interaction $\exp(-i\pi n_i n_j)$ and phase gates by decomposing

$$\exp\left(i\frac{\pi}{4}\bar{\gamma}_i\gamma_i\bar{\gamma}_k\gamma_k\right) = \exp\left(-i\pi\left(n_i - \frac{1}{2}\right)\left(n_j - \frac{1}{2}\right)\right). \quad (33)$$

Density-density interactions are implemented by using Rydberg interactions [25,47]. For a general BRAID₄ gate, we can conjugate [Eq. (33)] with BRAID₂ gates to get the desired form.

III. EVEN AND ODD MAJORANA CODES

In qubit quantum codes, logical qubits are encoded into the +1 eigenspace of the stabilizer operators. Each code block can encode multiple logical qubits, but operators in one code block usually only have support over the corresponding block and adding multiple blocks corresponds to a tensor product of each block. In Majorana codes however, logical operators can be even or odd weight, and once multiple code blocks are considered, they no longer admit a tensor product structure. The 1D Kitaev chain previously thought to encode qubits fails to do so when multiple chains are considered. We explore the consequences of even and odd codes in the following section and further give a scheme using quantum reference frames to perform logical gates on odd codes.

Definition 1. An **even** code is defined as a code where $P_{\text{tot}} \in S$.

Let us look at a few properties of these even codes that come as a consequence:

- (1) All logical operators must have **even** weight.
- (2) **Single logical per block:** If logical operators are constrained to a single block, one can only encode logical qubits here. If logicals are allowed to be nonlocal across blocks, then one can encode logical fermions.
- (3) **Multiple logicals, single block:** One can construct either a set of logical qubits such that

$$\{\bar{X}_i, \bar{Z}_i\} = 0 \quad [\bar{X}_i, \bar{X}_j] = 0 \quad (34)$$

or a set of even weight logical fermions

$$\{\Gamma_i, \bar{\Gamma}_j\} = 2\delta_{ij} \quad (35)$$

Or both, such that some degrees correspond to logical qubits, while others correspond to logical fermions. We can do this since this is just a basis

choice, so logical fermions are related to logical qubits via a logical Jordan-Wigner transformation. Note that we cannot interconvert between fermionic and qubit terms using any set of physical operations. Unitary operations preserve the commutation relations, and it can be shown that such maps, while trace-preserving, are not completely positive, so no physical process can convert between the two. See Appendix E for more details.

- (4) **Multiple logicals, multiple blocks:** Logical fermions with even weight in a block commute with logical fermions in a different block. In order to get actual logical fermions between multiple blocks, another nonlocal basis must be chosen that corresponds to a mutually anticommuting set of operators.

The most familiar setting for fault tolerance, which is the one we use here, is having multiple blocks where each logical is confined to a single block. As a result, it is most natural to consider **logical qubits** to be encoded per block in even codes.

Definition 2. An **odd** code is defined as a code where $P_{\text{tot}} \notin S$.

This entails the following properties for odd codes:

- (1) Logical operators can be both **odd** and **even** weight. There must also be at least two odd weight logical operators.
- (2) **Single logical degree per block:** If each logical is constrained to a single block, then one can choose two odd weight logical operators corresponding to a logical fermion. If logicals are allowed to be nonlocal, then one can encode logical qubits.
- (3) Odd weight logicals can also correspond to logical qubits and can be chosen so they satisfy Pauli algebra. However, between blocks, they do not overlap and hence will fail to have the correct commutation relations. We can choose new logicals that fulfil the correct Pauli algebra, but these will be nonlocal between blocks now. We will explain the issues in greater detail using the Kitaev chain later.
- (4) **Multiple logicals, single block:** Since a combination of odd and even logical operators within the same block is possible, one can choose all the odd operators to form a basis for logical fermions, and all the even weight operators to form a basis for logical Paulis. However, odd operators cannot be converted to even operators and vice versa in a closed system because of *parity superselection rule*.
- (5) **Multiple logicals, multiple blocks:** If each logical operator is chosen to be local to a block, one can ask, what is the number of logical fermions per

block? As specified, this depends on the chosen logical basis. The minimum number of logical operators with odd weight is two, if P_{tot} is chosen as one of the logical operators, and the rest except two operators corresponding to Γ_1 and $\bar{\Gamma}_1$ are even. This corresponds to one logical fermion and $k - 1$ logical qubits. To increase the number of logical fermions, one can pick a different logical basis choice in the following way: pick Γ_1 and $\bar{\Gamma}_1$ as the two odd logical operators to begin with, then choose $2l$ operators from the qubits operators to convert them into fermions via a logical Jordan-Wigner transformation, let's call these operators $q_1, q_2, \dots, q_{2l}$. Next, multiply $\bar{\Gamma}_1$ by operators $q_1 q_2 \dots q_{2l}$, and multiply each qubit operator by $\bar{\Gamma}_1$. This gives a new list of operators that are all odd weight and are all anti-commuting. As an example, consider the following code:

$$\begin{aligned} S &= \gamma_3 \gamma_4 \gamma_5 \gamma_6 \gamma_7 \gamma_8, \\ \Gamma_1 &= \gamma_1 \quad \bar{\Gamma}_1 = \gamma_2, \\ \bar{X}_2 &= i\gamma_3 \gamma_4 \quad \bar{Z}_2 = i\gamma_4 \gamma_5, \\ \bar{X}_3 &= i\gamma_6 \gamma_7 \quad \bar{Z}_3 = i\gamma_7 \gamma_8. \end{aligned}$$

With the above procedure, we can convert qubits into fermions

$$\begin{aligned} S &= \gamma_3 \gamma_4 \gamma_5 \gamma_6 \gamma_7 \gamma_8, \\ \Gamma_1 &= \gamma_1 \quad \bar{\Gamma}_1 = \gamma_2 \gamma_3 \gamma_5, \\ \Gamma_2 &= \gamma_2 \gamma_3 \gamma_4 \quad \bar{\Gamma}_2 = \gamma_2 \gamma_4 \gamma_5, \\ \bar{X}_3 &= i\gamma_6 \gamma_7 \quad \bar{Z}_3 = i\gamma_7 \gamma_8. \end{aligned}$$

An important departure now, from the even codes case, is that for the setting of logicals remaining local to a block and having multiple blocks, one can have either **logical qubits** or **logical fermions** per block, or **both**, and the number can be varied as per the prescription given above at the time of basis choice, with the caveat that at a minimum, each block contains **one** logical fermion.

A. Example: Even and odd codes

A single Kitaev chain is an instance of an odd code. Traditionally, 1D Kitaev chain has been used as a model to encode topologically protected qubits, with qubits encoded in the Majorana zero energy edge modes,

$$\bar{X}_1 = \gamma_1 \quad \bar{Z}_1 = i\gamma_1 \bar{\gamma}_1, \quad (36)$$

where $\gamma_i, \bar{\gamma}_i$ denote the edge modes. In isolation, this encodes a single logical qubit. However, if we add one more chain, it is clear that the logical operators between

multiple blocks anticommute. In order to encode logical qubits, one needs a fermion-to-qubit encoding, e.g., Jordan-Wigner transformation. Under such a transformation, the new logical operators are given as

$$\bar{X}_i = \prod_{k < i} \gamma_k \bar{\gamma}_k \gamma_i \quad \bar{Z}_i = i\gamma_i \bar{\gamma}_i, \quad (37)$$

where we assume that the chains are attached to each other in a one-dimensional geometry. There are other encodings, e.g., in Bravyi Kitaev Superfast encoding, one finds

$$\bar{X}_i = \prod_{k < i} \gamma_k \bar{\gamma}_k \gamma_i \gamma_{i+1} \quad \bar{Z}_i = \prod_{k \leq i} \gamma_k \bar{\gamma}_k. \quad (38)$$

In both cases, it is obvious that the value of each new block depends on the fermion parity of previous blocks. This is a common feature of *ancilla free* fermion-qubit encodings, since the encoding needs to preserve the overall sign. This presents issues from a fault tolerance perspective, since each block can not be thought of as a parallel error-correcting code, which makes the framework of transversal gates nontrivial for these codes, since a fault in one code block can affect the value of all subsequent logical operators. It also makes certain logical gates much harder to implement; for instance, performing a logical Hadamard on the n th qubit requires a $\mathcal{O}(n)$ depth physical circuit, which is not practical. As a result, we prefer to keep each logical operator local to a code block, and hence a Kitaev chain would encode a single logical fermion.

Two Kitaev chains can also be combined, in what is known as a Majorana tetron code,

$$\bar{X} = i\gamma_1 \gamma_2 \quad \bar{Z} = i\gamma_1 \bar{\gamma}_1. \quad (39)$$

And the total parity of the two chains is fixed to the even parity sector

$$S = \gamma_1 \bar{\gamma}_1 \gamma_2 \bar{\gamma}_2. \quad (40)$$

Here each block of four modes commutes with operators in a different block, so it can naturally only encode logical qubits.

IV. QUANTUM REFERENCE FRAMES FOR GATES ON ODD CODES

We described above that Majorana systems are subject to parity superselection rules, which further entails that all physical operations must correspond to even parity operations. In particular, for odd codes, performing an operation that switches parity of the logical operators, or performing an operation which is a coherent superposition of an even and odd operation, is forbidden by parity superselection. This will be especially relevant when creating fermionic T

states, or in the measurement of logical operators that have odd Majorana weight. It is thus vital to find a way around the parity superselection rule.

In Ref. [43], the authors describe that superselection rules are mathematically equivalent to a lack of reference frame, and hence one can lift superselection rules by defining an appropriate reference system.

As an example, consider Alice and Charlie, where Alice has a state she considers to be a superposition of opposite parity values $|\psi\rangle_A = \frac{1}{2}(\alpha|0\rangle + \beta|1\rangle)$. Alice describes her states in the fermionic occupation number basis using her own phase reference, while Charlie's reference is rotated by a certain angle ϕ . If Charlie knows this angle, then he can describe Alice's states by rotating them by the corresponding angle, i.e., $|\psi\rangle_C = e^{i\phi R} |\psi\rangle_A$, where ϕ is a Z_2 variable and R is the rotation generator. However, if Charlie does not know his angle with respect to Alice, then he will perform an average over the angles ϕ , and such an averaging will yield a completely mixed state. This leads to the loss of all coherence information, and is known as the parity superselection rule, where coherent superpositions of even and odd states are not allowed. On the other hand, Alice is allowed to prepare a state which is a coherent superposition of same P_{tot} states and Charlie will prepare the same state up to an irrelevant global phase. Such a state is termed a *parity invariant state*. Note that the superselection rule arises because of an absence of a global *reference system* relative to which both Alice and Charlie can measure their states. If both parties share a consistent reference frame, then they can each “sync” the changes the other party made without loss of coherence. For instance, one can prepare even and odd parity invariant states of the form:

$$|\Psi_e\rangle = \frac{1}{2}(|0\rangle_R |0\rangle_S + |1\rangle_R |1\rangle_S), \quad (41)$$

$$|\Psi_o\rangle = \frac{1}{2}(|0\rangle_R |1\rangle_S + |1\rangle_R |0\rangle_S). \quad (42)$$

The subscript R and S indicate that states correspond to a product Hilbert space $H_R \wedge H_S$, corresponding to the reference Hilbert space and system hilbert space, where $\wedge$ is a wedge product between states i and j

$$|i\rangle \wedge |j\rangle = -|j\rangle \wedge |i\rangle. \quad (43)$$

We have written the states above in the eigenbasis of reference parity P_R and system parity P_S . One can also choose a different basis $|i \oplus j\rangle |j\rangle$ corresponding to the observables P_{tot} and P_S , which formally corresponds to an isomorphism from $H_R \wedge H_S$ to $H_{\text{tot}} \wedge H_S$. By making this basis choice, we are describing our system relative to the global degree, and in such a setting, the reference frame is not quantum, thus it is described as “dequantization” in the literature

[43]. Thus in this dequantized picture, we have

$$|\Psi_e\rangle = \frac{1}{\sqrt{2}}(|0\rangle_{P_{\text{tot}}} (|0\rangle_S + |1\rangle_S)), \quad (44)$$

$$|\Psi_o\rangle = \frac{1}{\sqrt{2}}(|1\rangle_{P_{\text{tot}}} (|0\rangle_S + |1\rangle_S)), \quad (45)$$

which means that *relative* to the reference, one can now prepare coherent superpositions of even and odd states, and hence “violate” the parity SSR.

For multiple systems, such as Alice and Charlie in the above example, one can prepare

$$|\Psi_e\rangle = \frac{1}{2}(|0\rangle_R (|00\rangle_{AC} + |11\rangle_{AC}) + |1\rangle_R (|01\rangle_{AC} + |10\rangle_{AC})), \quad (46)$$

which after dequantization, yields

$$|\Psi_e\rangle = \frac{1}{2} |0\rangle_{P_{\text{tot}}} (|0\rangle + |1\rangle)_A (|0\rangle + |1\rangle)_C \quad (47)$$

thus proving that relative to this basis, coherent superpositions of multiple systems can be prepared. Moreover, additional states can be tensored in and be later used as ancillas while remaining uncorrelated to the reference. To prepare such a state, one can construct a circuit composed of BRAID₂ operations. More details are given in Appendix D.

We now describe how to perform gates on odd codes using the reference. These gates are different from qubit-based X and Hadamard gates as they correspond to odd operations.

- (1) **Applying odd logical Majoranas:** Here, the example is of logical “X” gate. This corresponds to a bit flip on the system, thus we call this a logical “X” or fermionic X gate. Note that such an operation is not parity preserving, hence unphysical, but we can bring the reference here and define an operation that is parity preserving globally,

$$\bar{X}_{pp} = \gamma_R \gamma_A. \quad (48)$$

For instance, if the initial state is

$$|\psi\rangle_{RAC} = |000\rangle. \quad (49)$$

Applying logical “X” gives

$$\bar{X}_{pp} |\psi\rangle = |110\rangle. \quad (50)$$

Writing in the dequantized basis and tracing over the global state gives

$$\text{Tr}_{gl}[\bar{X}_{pp} |\psi\rangle \langle \psi| \bar{X}_{pp}^\dagger] = |10\rangle \langle 10|_{AC}. \quad (51)$$

Doing this gate on multiple blocks follows quite easily: if the number of code blocks is even, then this operation can be done without a reference. If the number of blocks is odd, then parity of the reference block is flipped, as well as the global parity of all the data blocks.

- (2) **Applying logical fermionic Hadamard:** Note that the goal here is to prepare coherent superposition of even and odd parity sectors, as follows:

$$|0\rangle \rightarrow \frac{|0\rangle + |1\rangle}{\sqrt{2}} = \frac{(1 + \gamma_1)}{\sqrt{2}} |0\rangle. \quad (52)$$

The above operation can be understood as a fermionic Hadamard H_f . Again, this operation is defined with respect to a reference mode. The parity preserving gate is given as

$$\bar{H}_{pp} = \exp\left(-\frac{\pi}{4} \bar{\gamma}_A \gamma_R\right). \quad (53)$$

On the state, the action is given as

$$\bar{H}_{pp} |000\rangle_{RAC} = \frac{1}{2} (|00\rangle + |11\rangle)_{RA} |0\rangle_C. \quad (54)$$

Externalizing the reference and tracing over, we get

$$\text{Tr}_{\text{tot}}[\bar{H}_{pp} |000\rangle_{RAC} \langle 000| \bar{H}_{pp}^\dagger] = |+\rangle_A \langle +|_A |0\rangle_C \langle 0|_C. \quad (55)$$

On the operators, this gives $\bar{H}_{pp} \bar{X}_{pp} \bar{H}_{pp}^\dagger = \eta \bar{Z}_{pp}$ and $\bar{H}_{pp} \bar{Z}_{pp} \bar{H}_{pp}^\dagger = \xi \bar{X}_{pp}$ where $\eta, \xi = \pm 1$. Note that fermionic Hadamard is not a unitary operation, rather it corresponds to a rank-1 projector to the coherent state $|+\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$. An example of a unitary fermionic operation would be the fermionic phase gate: $S = \frac{1}{\sqrt{2}}(1 + i\gamma_1)$.

- (3) **Applying two-mode gates:** The reference scheme can be used to apply two-mode operations that violate parity, but since the construction is more complex, we defer the discussion to Sec. V.

A. Limitations of reference frame construction

This approach can thus be used to apply any odd weight logical operation. Note that the same reference mode is used for any odd logical operation, so an error in the reference mode will lead to the application of faulty gates. To make this mode less vulnerable, one can use code concatenation such that the reference mode is now a logical Majorana instead, and then error correction can be applied periodically to flush out the errors from the reference.

It would seem that since performing an odd logical operation requires a reference call, so in general doing n odd gates would require n calls to the reference. However, this

is not the case in specific instances. For example, if one has n fermionic Hadamard gates

$$H_f^n = \frac{1}{2^{n/2}} (1 + \gamma_1)(1 + \gamma_2) \dots (1 + \gamma_n). \quad (56)$$

This can be rewritten as follows:

$$H_f^n = \frac{1}{2^{n/2}} (1 + \gamma_1)(1 + \gamma_2 \gamma_3) \dots (1 + \gamma_{n-1} \gamma_n). \quad (57)$$

Doing so remarkably simplifies the application of the circuit, as only the first Hadamard $(1 + \gamma_1)$ would require a reference call. Note that the remaining circuit corresponds to a circuit of fermionic Clifford gates, which can likely be computed in depth $O(\log(n))$ [48]. In other words, the “resource cost” of doing one Hadamard is the same as doing it for n Hadamards, hence it acts as a “catalyst” state. A similar result is obtained using resource theoretic arguments in Ref. [49]. We use this compression argument in Sec. VIII to use a single reference state to create 15 faulty T states. It would be interesting to explore similar parallelization for other fermionic non-parity-preserving gates.

We also note that a reference system construction was also developed in Ref. [28] to perform number nonconserving operations; however our construction requires a quantum reference system, hence the size does not scale with the number of modes in the data code block.

V. TRANSVERSAL GATE GADGETS

Typically for qubit codes, transversality is the first step toward fault tolerance because of its simple construction and since it prevents the propagation of errors which is vital for a fault-tolerant gadget. The notion of transversality is often variable: sometimes it corresponds to performing the same gate on all the physical qubits in the same block, which is sometimes referred to as *strongly transversal* [50], whereas on other occasions the physical gate may vary on each qubit but still admit a tensor product decomposition, and it may not act on each qubit either. In our framework, when considering gates acting on multiple blocks, we consider the former definition which corresponds to the same gate acting on each *mode* in a code block. Another critical aspect when considering the code’s protection is also the fact that it may treat even and odd errors separately, and while it may have good protection against even parity errors, it may be vulnerable to odd errors, and this is captured by l_{even} . However, there are issues when considering this parameter when one has multiple code blocks and transversal gates. First, a physical process that results in an even parity error can affect a single code block, but it can also affect multiple code blocks, such that it acts as an odd parity error, so superselection does not actually help boost the code protection.

(This was also alluded to in Ref. [9].) Second, if one considers a transversal gate between multiple blocks, it may fail and change an error which is even on one code block to an error that is odd on both. In other words, local parity conservation is too strict of a constraint for error models and gadgets to have, so the significance of l_{even} for a block is minimal.

The Clifford operations available to us are BRAID₂ and BRAID₄. For a transversal operation, we require the stabilizer to be preserved, without increasing the weight of errors beyond what can be detected or corrected. We have BRAID₂ and BRAID₄ gates at our disposal in Majorana codes, which act on two or four copies of the code automatically preserve the code space of any Majorana code.

A. Detail of BRAID₂ and BRAID₄ transversality

Let us consider the action of BRAID₂ on two code blocks. The stabilizer is given by $S = \langle M_i \otimes I, I \otimes M_i \rangle$, where M_i are the generators on each individual block. A transversal BRAID₂, denoted as $(\text{BRAID}_2)^T$, acts on every mode i in block 1 to every mode i in block 2. For even weight operators, the transversal gate acts as a “swap” gate since the global sign disappears. This transforms the stabilizers as

$$(\text{BRAID}_2)^T (M_i \otimes I) (\text{BRAID}_2^\dagger)^T = I \otimes M_i. \quad (58)$$

On the logicals, this corresponds to

$$(\text{BRAID}_2)^T (\bar{X} \otimes I) (\text{BRAID}_2^\dagger)^T = I \otimes \bar{X}, \quad (59)$$

$$(\text{BRAID}_2)^T (\bar{Z} \otimes I) (\text{BRAID}_2^\dagger)^T = I \otimes \bar{Z}. \quad (60)$$

Hence, the logical SWAP can be carried out transversally.

Let us next look at the BRAID₄ gate. A single BRAID₄ gate acts on four Majorana modes in the following way:

$$\text{BRAID}_4(ijkl) \gamma_l \text{BRAID}_4^\dagger(ijkl) = i \gamma_j \gamma_k \gamma_l. \quad (61)$$

The transversal $(\text{BRAID}_4)^T$ acts on four blocks of the code, so the stabilizer is $S = \langle M_i \otimes I \otimes I \otimes I, I \otimes M_i \otimes I \otimes I, I \otimes I \otimes M_i \otimes I, I \otimes I \otimes I \otimes M_i \rangle$. On the logicals $(\text{BRAID}_4)^T$ corresponds to

$$(\text{BRAID}_4)^T (\bar{X} III) (\text{BRAID}_4^\dagger)^T = I \bar{X} \bar{X} \bar{X}, \quad (62)$$

$$(\text{BRAID}_4)^T (\bar{Z} III) (\text{BRAID}_4^\dagger)^T = I \bar{Z} \bar{Z} \bar{Z}. \quad (63)$$

However, certain operators commute with this $(\text{BRAID}_4)^T$ gate. For instance, note that

$$(\text{BRAID}_4)^T (\bar{X} \bar{X} II) (\text{BRAID}_4^\dagger)^T = \bar{X} \bar{X} II, \quad (64)$$

$$(\text{BRAID}_4)^T (\bar{Z} \bar{Z} II) (\text{BRAID}_4^\dagger)^T = \bar{Z} \bar{Z} II. \quad (65)$$

This implies that states prepared in a logical Bell state remain invariant under this transversal gate.

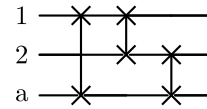


FIG. 1. Ancilla-mediated swap. Note that typically BRAID₂ gates are considered robust to local perturbations since it is *topologically protected*.

B. Mode permutation to achieve logical gates

Consider a Majorana code where $\bar{X}$ and $\bar{Z}$ are both even and have the same weight. One toy example is the Majorana tetron code, where the weight of all three logical operators is the same. Hence, a single BRAID₂ gate suffices to perform a logical Hadamard or a logical phase gate, up to Paulis. A single permutation gate, however, can also fail and can potentially cause a weight two error. To avoid this, we can apply an ancilla-mediated swap [38], described as follows: prepare an additional ancilla code block, and now instead of swapping directly between the modes 1 and 2, one can have three swap gates for each swap gate. This way, even if one swap gate fails, weight-1 errors do not become weight-2 errors on the data block, cf., Fig. 1. Note that while a single braid gate is not strictly a swap gate, three braids preserve the braid structure, i.e., we want $\gamma_1 \rightarrow \gamma_2, \gamma_2 \rightarrow -\gamma_1$, and it is easy to check that three braids below give us $\gamma_1 \rightarrow \gamma_2, \gamma_2 \rightarrow -\gamma_1, \gamma_a \rightarrow \gamma_a$.

A second example of permutation-based gates comes from exploiting the automorphism structure of quantum codes. The automorphism group of a code $\text{Aut}(C)$ is defined by the set of logical gates that arise from permutation of modes such that they preserve the stabilizer group [51,52]. For Majorana codes, one notable example of a code family with a nontrivial automorphism group is the Majorana Reed-Muller code, which are derived from applying Lemma 1 to classical Reed-Muller codes $\text{RM}(r, m)$ [44]. In order to exhibit weak self-duality, the parameters must satisfy the constraint that $m \geq 2r + 1$. As is pointed out in Ref. [44], the Majorana code obtained from an $\text{RM}(1, m)$ code $[n = 2^m, m + 1, d]$ is $[[2^m, 2^m - m - 1, d^\perp]]$.

Classical Reed-Muller codes have a rich symmetry structure, which can be extended to Quantum Reed-Muller codes as well [53]. For a $C = \text{RM}(1, m)$ code, the automorphism group $\text{Aut}(C)$ is the general affine group $GA(m, 2)$.

Consider the case of $\text{RM}(1, 4)$. The parity check matrix is given as

$$H = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{bmatrix}. \quad (66)$$

The code parameters are $[[8, 3, 4]]_f$. The logical basis vectors can be found by

$$H\Lambda_f\bar{L} = 0, \quad (67)$$

i.e., vectors that commute with the stabilizer but are not contained inside the stabilizer. This corresponds to: Note that indices 0 to n are labeled with γ while indices $n+1$ to $2n$ are labeled by $\bar{\gamma}$, and when referring to the numbers, any index position x greater than n is labeled by $\overline{x-n}$, e.g., index position 11 is 3,

$$\bar{Z}_1 = \bar{\gamma}_3\bar{\gamma}_4\bar{\gamma}_5\bar{\gamma}_6 \quad \bar{X}_1 = \gamma_5\gamma_7\bar{\gamma}_3\bar{\gamma}_4\bar{\gamma}_6\bar{\gamma}_7, \quad (68)$$

$$\bar{Z}_2 = \bar{\gamma}_5\bar{\gamma}_6\bar{\gamma}_7\bar{\gamma}_8 \quad \bar{X}_2 = \gamma_3\gamma_5\bar{\gamma}_3\bar{\gamma}_6\bar{\gamma}_7\bar{\gamma}_8, \quad (69)$$

$$\bar{Z}_3 = \gamma_2\gamma_3\gamma_5\gamma_7\bar{\gamma}_2\bar{\gamma}_3\bar{\gamma}_5\bar{\gamma}_7 \quad \bar{X}_3 = \gamma_5\gamma_6\bar{\gamma}_1\bar{\gamma}_3\bar{\gamma}_6\bar{\gamma}_7. \quad (70)$$

Note that the symmetry group corresponds to general affine transformation on the four variables [38]. This means that transformations of the form

$$\mathbf{x} \rightarrow A\mathbf{x} + b, \quad (71)$$

where $\mathbf{x} = (x_4, x_3, x_2, x_1)$ on the field $\mathbb{F}_2$ are valid symmetry transformations. One example is transformation $(x_4, x_3, x_2, x_1) \rightarrow (x_4, x_3, x_2 \oplus x_3, x_1)$. Converting the binary numbers to decimal numbers in standard lexicographic ordering, i.e., $0000 \rightarrow 1$, $0001 \rightarrow 2$, the above binary transformation corresponds to the following index permutation:

$$G = (5, 7)(6, 8)(13, 15)(14, 16). \quad (72)$$

Converting to barred notation, this gives

$$G = (5, 7)(6, 8)(\bar{5}, \bar{7})(\bar{6}, \bar{8}). \quad (73)$$

Moreover, this transforms the logical operators as follows:

$$\bar{Z}_1 \rightarrow \bar{Z}_1\bar{Z}_2 \quad \bar{X}_1 \rightarrow \bar{X}_1\bar{Z}_2, \quad (74)$$

$$\bar{Z}_2 \rightarrow \bar{Z}_2 \quad \bar{X}_2 \rightarrow \bar{X}_1\bar{X}_2\bar{Z}_1\bar{Z}_2, \quad (75)$$

$$\bar{Z}_3 \rightarrow \bar{Z}_3 \quad \bar{X}_3 \rightarrow \bar{X}_3. \quad (76)$$

This corresponds to the logical gate

$$G = \overline{CZ}(1, 2)\overline{CNOT}(2, 1). \quad (77)$$

Note that the above gate corresponds to an entangling operation, and given a single qubit logical gate on each qubit, it would correspond to the full Clifford group. In comparison, for a CSS qubit code obtained from the above Reed Muller code, it is easy to show that a similar permutation would give only CNOT gates and would require

an additional layer of Hadamard gates, which we do not require here. Note that since every logical operator has even weight, we do not need to worry about the phase resulting from BRAID₂ gates. It is also interesting to note that swap operations in this code permit both CNOT and CZ gates, as opposed to permutation gates in qubit codes which correspond only to CNOT gates in the absence of further symmetries in the code structure.

One can also have Majorana codes with transversal non-Clifford gates, but we defer the discussion of this code to Sec. VIII.

VI. FAULT-TOLERANT GATE GADGETS: SUPPLEMENTING TRANSVERSAL OPERATIONS WITH MEASUREMENTS

Transversal gates are limited by the code structure and in order to do a full set of Clifford gates, we often need to supplement with other techniques. For instance, if the logical operators have unequal weights, such as in a Majorana color code where the X and Z logical operators are different weights, then one needs operations that change the weight of the logical operators. We suggest here to use more ancilla code blocks initialized in a particular state, along with measurements, which takes us out of the setting of just transversal gates.

A. Even codes

As a toy example, we can take two tetron codes and choose logicals that have support over both tetron blocks. The standard choice of logical operators for two tetrans is

$$\bar{X}_a = \gamma_a\gamma_b \quad \bar{Z}_a = \gamma_b\gamma_c, \quad (78)$$

$$\bar{X}_b = \gamma_e\gamma_f \quad \bar{Z}_b = \gamma_f\gamma_g. \quad (79)$$

We instead pick logical operators to get a “coupled tetron” code

$$\bar{X}_a = \gamma_a\gamma_b\gamma_e\gamma_f \quad \bar{Z}_a = \gamma_a\gamma_c, \quad (80)$$

$$\bar{X}_b = \gamma_e\gamma_f \quad \bar{Z}_b = \gamma_a\gamma_c\gamma_e\gamma_g, \quad (81)$$

which now gives us two logical operators but the $\bar{X}$ and $\bar{Z}$ weights are unequal. Below, we give a transversal scheme using ancilla to perform a logical Hadamard on the first qubit. Note that the code has distance two, so it can only detect weight one errors, and the objective of the scheme is to make sure that each code block still has weight one errors while preserving the stabilizer and enacting the desired logical gate. For the sake of this example, we chose this code, but it can also be generalized for a higher distance code with perhaps different overheads on ancilla count and circuit depth.

A logical Hadamard is then implemented using the following procedure:

- (1) Take four more ancilla blocks, all encoded in the +1 eigenspace of the standard tetron code. Additionally, ancilla 2 and ancilla 3 are prepared in a logical Bell state, meaning that $\bar{X}_2\bar{X}_3$ and $\bar{Z}_2\bar{Z}_3$ are also in the stabilizer. The logical state of the first and last ancilla block can be arbitrary.
- (2) Now do a transversal BRAID_4 gate between the first three ancilla and the first data block. This transforms $\bar{X}_a$ to $\bar{X}_b\bar{X}_1\bar{X}_2\bar{X}_3$ and $\bar{Z}_a$ to $\bar{Y}_1\bar{Y}_2\bar{Y}_3$. By multiplying with the stabilizer, we get $\bar{X}_a \rightarrow \bar{X}_b\bar{X}_1$ and $\bar{Z}_a \rightarrow \bar{Y}_1$.
- (3) Do a second transversal BRAID_4 gate between the first two blocks and the first and last ancilla. This transforms $\bar{Z}_a$ to $\bar{Z}_b\bar{Z}_2\bar{Z}_3\bar{Z}_4$, and then multiplying with stabilizer gives $\bar{Z}_b\bar{Y}_4$, but leaves $\bar{X}_b\bar{X}_1$ invariant.
- (4) We perform a BRAID_2 gate here. Since this gate is **not** transversal, we need to use another ancilla (not shown) to prevent error propagation. Moreover, similar to qubit case, permutations within an error block can cause bad error propagation, so one can insert blocks of error correction after each such BRAID_4 gate.
- (5) The weight of both logicals is the desired weight, but we still want to do operations transversally that preserve the weight and the stabilizer. The third transversal BRAID_4 converts $\bar{X}_a$ to $\bar{Z}_a\bar{X}_1$ and $\bar{Z}_a$ to $\bar{X}_a\bar{Y}_4$. The ancilla qubits can be measured out in the correct basis. The whole circuit is shown in Fig. 2.
- (6) Note that this circuit only applies to this example, but similar techniques may be used to construct Hadamard for other codes with this property.

Another example is constructing transversal CNOT on even codes, akin to gadgets constructed in Ref. [54].

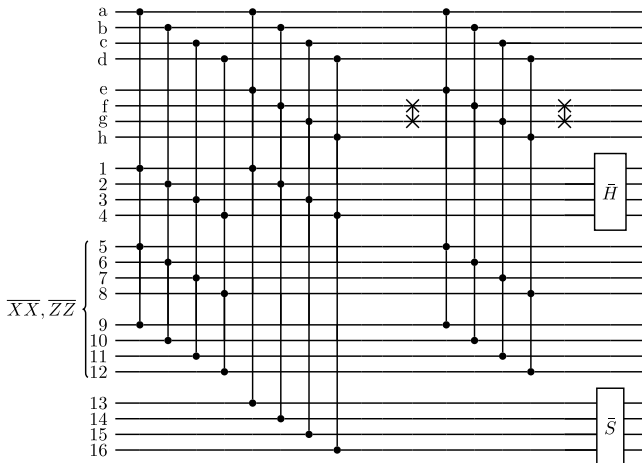


FIG. 2. Logical Hadamard for coupled tetron: $\bar{X}_a = \gamma_a\gamma_b\gamma_e\gamma_f$, $\bar{X}_b = \gamma_e\gamma_f$, $\bar{Z}_a = \gamma_a\gamma_c$, $\bar{Z}_b = \gamma_a\gamma_c\gamma_e\gamma_g$. The four-mode gate denotes BRAID_4 and the two-mode gate denotes the BRAID_2 . The ancillas are then measured out in the occupation number basis i.e., eigenbasis of $\bar{Z}_i$, $\{i = 1, 2, 3, 4\}$ for each ancilla block.

Consider a circuit on four blocks, where the first two blocks are data blocks, labeled a and b and the last two blocks are ancilla blocks, labeled 1 and 2. The circuit is given by the following unitary:

$$U = \text{BRAID}_2^T(b, 1)\text{BRAID}_2^T(a, b) \\ \times \text{BRAID}_2^T(a, 2)\text{BRAID}_4^T(a, b, 1, 2). \quad (82)$$

The circuit has the following logical action:

$$\begin{aligned} \bar{X}_a &\rightarrow \bar{X}_a\bar{X}_b\bar{X}_1 & \bar{Z}_a &\rightarrow \bar{Z}_a\bar{Z}_b\bar{Z}_1, \\ \bar{X}_b &\rightarrow \bar{X}_b\bar{X}_1\bar{X}_2 & \bar{Z}_b &\rightarrow \bar{Z}_b\bar{Z}_1\bar{Z}_2, \\ \bar{X}_1 &\rightarrow \bar{X}_a\bar{X}_1\bar{X}_2 & \bar{Z}_1 &\rightarrow \bar{Z}_a\bar{Z}_1\bar{Z}_2, \\ \bar{X}_2 &\rightarrow \bar{X}_a\bar{X}_b\bar{X}_2 & \bar{Z}_2 &\rightarrow \bar{Z}_a\bar{Z}_b\bar{Z}_2, \end{aligned}$$

while the stabilizer is preserved. Initially, the ancilla blocks are prepared in the $\bar{Z}_1$ and $\bar{Z}_2$ state. Since we fix the state of the ancilla blocks, the ancilla blocks now correspond to stabilizer states and $\bar{Z}_1$ and $\bar{Z}_2$ correspond to stabilizer elements. At the end, we measure the ancilla in $\bar{X}_1$ and $\bar{X}_2$ basis. Since the $\bar{Z}$ logical representatives anticommute with the measurement, one can multiply with stabilizer elements of the ancilla states (for $\bar{Z}_a$ multiply by $\bar{Z}_1 \cdot \bar{Z}_2$ and for $\bar{Z}_b$ multiply by $\bar{Z}_1$ where the primed operators represent transformed operators). This gives the desired transformation for a transversal CNOT.

B. Odd codes

Another approach can be to encode logical fermions into the odd operators of the code, but it may be desirable to construct a fermionic CNOT between states in the number basis. For instance, while the physical terms in a closed fermionic Hamiltonian are in a fixed parity sector, it may be desirable in a simulation algorithm to perform a parity switching operation if the system is not closed and interacts with a fermionic bath. The logical fermionic CNOT in this context would mean

$$\begin{aligned} \Gamma_a &\rightarrow \Gamma_a\Gamma_b, \\ \Gamma_b &\rightarrow \Gamma_b, \\ \Gamma_a\bar{\Gamma}_a &\rightarrow \Gamma_a\bar{\Gamma}_a, \\ \Gamma_b\bar{\Gamma}_b &\rightarrow \Gamma_a\bar{\Gamma}_a\Gamma_b\bar{\Gamma}_b, \end{aligned}$$

where $\Gamma_i, \bar{\Gamma}_i$ denote logical Majoranas. Such an operation clearly switches parity between odd and even logical operators such as $\Gamma_a \rightarrow \Gamma_a\Gamma_b$, and thus would not be allowed without a reference. The odd operators are thus defined,

using ideas from Sec. III

$$\Gamma'_a = \Gamma_a \Gamma_r, \quad (83)$$

$$\Gamma'_b = \Gamma_b \Gamma_r. \quad (84)$$

Note that Γ_a and Γ_b anticommute thus correspond to logical fermions, and similarly, Γ'_a and Γ'_b are also chosen to anticommute. In this case, we also need to fix the state of the ancilla, and here we choose it to be an eigenstate of

$$S_1 = i\Gamma_1 \Gamma_r, \quad S_2 = i\Gamma_2 \bar{\Gamma}_r. \quad (85)$$

The circuit we apply to get the desired ancilla state is

$$C = \text{BRAID}_2(\bar{2}, \bar{r}) \text{BRAID}_2(\bar{1}, r), \quad (86)$$

where $\text{BRAID}_2(\bar{2}, \bar{r})$ is a product of BRAID_2 gates between the barred Majorana operators in block two and the reference block (block r), and $\text{BRAID}_2(\bar{1}, \bar{r})$ is a product of BRAID_4 gates between the barred Majorana operators between block 1 and unbarred operators in block r .

The initial stabilizer elements for the ancilla are $i\Gamma_1 \bar{\Gamma}_1$ and $i\Gamma_2 \bar{\Gamma}_2$, which can be conjugated by the above circuit C to obtain the required state.

Next, a circuit consisting of transversal BRAID_4 and BRAID_2 can now be executed as

$$U = \text{BRAID}_2^T(a, b) \text{BRAID}_2^T(b, 2) \text{BRAID}_4^T(a, b, 1, 2), \quad (87)$$

giving

$$\Gamma'_a : \Gamma_a \Gamma_r \rightarrow \Gamma_a \Gamma_1 \Gamma_2 \Gamma_r, \quad (88)$$

$$\Gamma'_b : \Gamma_b \Gamma_r \rightarrow \Gamma_a \Gamma_b \Gamma_1 \Gamma_r, \quad (89)$$

$$\Gamma_a \bar{\Gamma}_a \rightarrow \Gamma_a \bar{\Gamma}_a \Gamma_1 \bar{\Gamma}_1 \Gamma_2 \bar{\Gamma}_2, \quad (90)$$

$$\Gamma_b \bar{\Gamma}_b \rightarrow \Gamma_a \bar{\Gamma}_a \Gamma_b \bar{\Gamma}_b \Gamma_2 \bar{\Gamma}_2, \quad (91)$$

on the data block, while the ancilla operators transform as

$$\Gamma_1 \Gamma_r \rightarrow \Gamma_a \Gamma_b \Gamma_2 \Gamma_r, \quad (92)$$

$$\Gamma_2 \bar{\Gamma}_r \rightarrow \Gamma_b \Gamma_1 \Gamma_2 \bar{\Gamma}_r. \quad (93)$$

Lastly, the ancilla blocks are measured in the occupation number basis, i.e., $i\Gamma_1 \bar{\Gamma}_1$ and $i\Gamma_2 \bar{\Gamma}_2$ basis. Note that the parity operators i.e., $\Gamma_a \bar{\Gamma}_a, \Gamma_b \bar{\Gamma}_b$ on the data blocks commute with this measurement, whereas the primed operators do not, but similar to the previous examples, they can be multiplied by the ancilla stabilizer to fix them. This gives us

$$\Gamma'_a \rightarrow \Gamma_a \Gamma_b \Gamma_r \bar{\Gamma}_r, \quad (94)$$

$$\Gamma'_b \rightarrow \Gamma_b \bar{\Gamma}_r. \quad (95)$$

We can see here that had we chosen two commuting operators, then we would have got an actual logical CNOT. But

since they are fermionic in nature, and they all mutually anticommute, instead we get a

$$\Gamma'_a \rightarrow \Gamma'_a \Gamma'_b \Gamma_r \bar{\Gamma}_r, \quad (96)$$

$$\Gamma'_b \rightarrow \Gamma'_b \Gamma_r \bar{\Gamma}_r, \quad (97)$$

which corresponds to the following logical gate:

$$\text{CNOT}(1, 2) \text{CZ}(a, r) \text{CZ}(b, r). \quad (98)$$

VII. FAULT-TOLERANT ERROR CORRECTION: STEANE ERROR CORRECTION GADGET

Along with fault-tolerant gates, we also need a way to perform stabilizer measurements in a fault-tolerant manner. In the absence of fault-tolerant measurements, an error from the ancilla that is recording the syndrome value can jump to the data and cause a logical error. For qubit codes, standard code-agnostic approaches include repeated rounds of measurement (Shor EC) or using ancilla blocks prepared and measured in a special basis (Steane or Knill style EC). We aim to extend fault-tolerant error correction to Majorana codes, in particular, we find a Steane-style EC for Majorana CSS codes. Note that contrary to qubits, Majorana operators can only be measured in pairs, thus for a code on $2n$ modes, one can only make at most n simultaneously commuting measurements.

Note that unlike qubit Steane gadgets which depend on physical CNOTs, our gadget uses the physical BRAID_4 gates and three ancilla blocks instead of one. The upshot is that unlike qubit Steane EC, our gadget can also distinguish between data and ancilla errors. See Fig. 3.

The gadget has support over four code blocks, where the first code block corresponds to the data block, while the remaining three correspond to ancilla blocks. Similar to Steane EC, we need to choose an ancilla state so that we can find error syndromes transversally, without measuring the logical information. We encode the three ancilla blocks in the same code as the data block. Moreover, we choose the following additional stabilizer conditions:

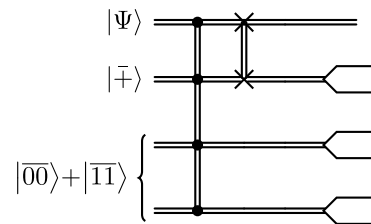


FIG. 3. Steane gadget for four blocks of $[[n, 1, d]]_f$ code., with transversal measurement in the occupation basis. This corresponds to correcting for γ_i errors. The measurement blocks correspond to measuring in the occupation number basis, followed by classical post processing. For $\tilde{\gamma}_i$, we conjugate this circuit with $\mathcal{B}$ given in the Sec. IV.

- (1) $\bar{X}_2$ is inside the stabilizer, hence the second logical block is in $|\bar{+}\rangle$.
- (2) $\bar{X}_3\bar{X}_4$ and $\bar{Z}_3\bar{Z}_4$ are also in the stabilizer, so the last two blocks are in a *logical Bell* pair $\frac{1}{2}(|\bar{00}\rangle + |\bar{11}\rangle)$.

We can correct errors of γ_i type using this circuit. Note that

- (1) Errors on the data block propagate to ancilla blocks three and four and also act on data block, at the same location.
- (2) Errors on ancilla block two propagate to errors on blocks two, three, and four.
- (3) Errors on ancilla block three propagate to errors on the data block, block two and block four.
- (4) Errors on ancilla block four propagate to errors on the data block, block two and block three.

Since measuring the syndrome on a single block does not uniquely identify the location of the fault, we need syndromes from all four blocks.

To represent the codewords of Majorana codes, we need to first select a complete set of basis states. For $2n$ Majoranas, one has n physical fermions, with one identifying the pair $\gamma, \bar{\gamma}$ with each mode. The occupation number basis $|n_1, n_2, n_3, \dots, n_n\rangle$ is the analogous computational basis in this context, and the default measurement basis. The codewords of this code are

$$|\bar{k}\rangle = \prod_i (I + g_i) N^k \overbrace{|00 \dots\rangle}^n, \quad (99)$$

where g_i are rows of matrix S . Up to phases, we can multiply them to find the following form:

$$|\bar{k}\rangle = \sum_i g_i N^k |00 \dots 0\rangle, \quad (100)$$

$$|\bar{k}\rangle = \sum_{v \in C} |\bar{k} + v\rangle. \quad (101)$$

Here, $\bar{k}$ labels the coset number. For the second code block, the states can be written as

$$|\bar{+}\rangle = \sum_{v \in C} |\bar{0} + v\rangle + \sum_{v \in C} |\bar{1} + v\rangle, \quad (102)$$

while for the third and fourth code blocks the states are

$$|\bar{++}\rangle = \sum_{v, w \in C} |\bar{0} + v, \bar{0} + w\rangle + \sum_{v, w \in C} |\bar{1} + v, \bar{1} + w\rangle. \quad (103)$$

One can measure in the occupation number basis and use the parity check matrix of the classical code C to check for

low weight γ errors. In order to correct the errors, we need to apply a recovery operation. By doing classical decoding on each block, it gives out a recovery operation that would cancel the syndrome. If the syndrome in each block is labeled s_j , $\{j = 2, 3, 4\}$, then the syndrome for the data block is given as

$$s_d = s_2 \wedge s_3 + s_2 \wedge s_4 + s_3 \wedge s_4 + s_2 \wedge s_3 \wedge s_4 \pmod{2}. \quad (104)$$

This tells us that to identify the data syndrome, we need to combine information from the remaining syndromes, since for instance, an error on the data block raises a flag on the same syndrome bits of blocks three and four, but nothing on block two, and thus the above function will evaluate the syndrome vector on the data block, which can then be used to estimate the error vector on the data and apply the corresponding correction. Moreover, since the data and ancilla blocks have error patterns, one can distinguish if the error came from a data block or ancilla block. More details about the syndrome recovery are in Appendix F.

For errors of type $\bar{\gamma}_i$, we can conjugate with the transversal transformation,

$$\mathcal{B} = \prod_{i=1}^n \text{BRAID}_2(i, i+n) = \prod_{i=1}^n \exp\left(-\frac{\pi}{4} \gamma_i \bar{\gamma}_i\right). \quad (105)$$

For γ_i errors, we prepare the second ancilla block in the logical $|\bar{+}\rangle$ block and measure transversally in the computational basis. Instead if we have $\bar{\gamma}_i$ errors, we prepare in $\mathcal{B}|\bar{+}\rangle$ and measure in $\mathcal{B}^\dagger$ [occupation basis], and since $\text{BRAID}_2 \gamma_i \text{BRAID}_2 = \bar{\gamma}_i$, this transformation is sufficient to find syndromes of $\bar{\gamma}_i$.

Recovery operation: A recovery operation as described before corresponds to an odd weight operation on the data register. To do so, we can also keep an ancilla register such that the quantum operation on the ancilla+system is an even weight operation and does not violate parity superselection. We can initialize an ancilla system with n_r modes, such that they are initially in a global even state. We can also keep an extra classical bit that stores the parity of the ancilla register. This scheme corresponds to a parity bit error detecting code and can only detect an odd number of parity errors. A recovery operation thus corresponds to acting with $\mathcal{R} \circ \mathcal{A}(\rho_{ad})$, where $\mathcal{A}$ is defined as follows:

$$\mathcal{A}(\rho_{ad}) = \sum_{j=1}^{n_r} (1 - \eta_{j+1}) \gamma_j^a \eta_{j-1}(\rho_{ad}) \sum_{j=1}^{n_r} (1 - \eta_{j+1}) \gamma_j^a \eta_{j-1}, \quad (106)$$

where $\eta_j = i\gamma_j^a \bar{\gamma}_j^a$, which basically removes a Majorana from the last mode. We can also update the parity bit when we do so. In case there is a parity flip on the ancilla, we can measure P_{tot} of the ancilla register and compare against

the stored classical bit, if those are different, then we have incurred a parity flip and one can reset this register to the prior state. Note that throughout this recovery operation, the ancilla and data registers remain unentangled: in a correct stabilizer subspace, both the ancilla and the data are even and can be written in a tensor product fashion. In the event of an odd error, and thereafter recovery, the code state is returned to its previous correct subspace that corresponds to an even code, while the ancilla is in an odd parity subspace.

A. Logical measurement in Steane

To measure logical operators of the data block, we need to prepare the ancilla block in a state such that the logical state of the data is copied to it and can be transversally measured. In qubits, to measure $\bar{Z}_d$, ancilla block is prepared in the logical $\bar{0}$ state, or an eigenstate of $\bar{Z}$ operator. A transversal CNOT then copies the state of the data block: if there is a logical bit flip on the data, then the state of the ancilla block is also flipped, and a logical $\bar{Z}$ measurement thus detects this flip. We can use a similar idea for the logical measurement here.

To measure $\bar{Z}$ on the data block, we need to copy the logical operator from the data block to the ancilla blocks. As mentioned above,

$$\bar{X}_d \rightarrow \bar{X}_d \bar{X}_3 \bar{X}_4. \quad (107)$$

By preparing ancilla blocks three and four in eigenstate of $\bar{Z}$ for each respective block, we can measure each block transversally and perform classical error correction as described above to find the closest logical state. A similar idea is present in order to perform a logical X measurement, except now the ancilla state is prepared in an eigenstate of $\bar{X}$ rather than $\bar{Z}$.

This protocol is important in two contexts, first we can combine a nonfault-tolerant encoding scheme as in Ref. [20] with this gadget, and by measuring the stabilizers of the code and the logicals, we can verify that we prepared the correct state fault tolerance and in case we detect any faults, we can discard and start again. Second, we can insert this gadget into various error locations and measure the stabilizer of the code to flush out any errors in the system.

VIII. TRANSVERSAL NON-CLIFFORD GATES

While Clifford gates and measurements are a useful set of gates since they correspond to stabilizer circuits, in order to do universal computing, one also needs non-Clifford resources. T gate is one instance, that along with fermionic Clifford operations, constitutes universal fermionic computing [55]. In most schemes performing magic state distillation, one requires a code that contains a logical T gate.

To this end, we show an example of a Majorana code that implements a transversal logical T gate.

In the qubit quantum Reed-Muller code, the codewords are associated with the cosets of $C_2^\perp$ in C_1 . Note that each element in $C_2^\perp$ satisfies the *triorthogonality* condition and, in particular, for $[[15, 1, 3]]$, each basis state in the $|\bar{0}\rangle$ has weight $0 \bmod 8$

$$|\bar{0}\rangle = \sum_{v \in C_2^\perp} |v\rangle, \quad (108)$$

where $C_2^\perp$ is generated by H_2

$$H_2 = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{bmatrix}. \quad (109)$$

As a result, applying a rotation by $R_{\pi/8}^{\otimes n}$, where $R_{\pi/8} = e^{-i(\pi/8)\sigma_z}$, gives an overall global phase of $e^{i\phi}$, where $\phi = -n\pi/8$.

Analogously, consider the T gate for Majoranas,

$$R_{\pi/8} = e^{-\frac{\pi}{8}\gamma_i\bar{\gamma}_i} = e^{-i\frac{\pi}{8}(-i\gamma_i\bar{\gamma}_i)}. \quad (110)$$

We discussed before that in general, for Majorana CSS codes, $C_2^\perp$ is not necessarily contained in C_1 . However, if one starts with a qubit CSS code such that $C_2^\perp \subseteq C_1$ and chooses the parity check matrices $H_\gamma = H_x$ and $H_{\gamma\bar{\gamma}} = H_z$, then the corresponding Majorana code would have $C_2^\perp \subseteq C_1$ and would also be self-orthogonal. In other words, H_γ generates $C_2^\perp$. This allows us to write codewords, for instance, for 30 Majoranas, $[[15, 1, 3]]_f$ code,

$$|\bar{0}\rangle = \sum_{v \in C_2^\perp} |v\rangle, \quad (111)$$

$$|\bar{1}\rangle = \sum_{v \in C_2^\perp} |1 + v\rangle, \quad (112)$$

where $|1\rangle \in C_1 \setminus C_2^\perp$ and corresponds to all ones vector. More precisely, C_1 corresponds to punctured $\mathcal{R}(1, 4)$ (last index is deleted) and $C_2^\perp$ corresponds to the even subcode, with the all ones vector removed.

From the same argument, the $|\bar{0}\rangle$ has weight $0 \bmod 8$, and by applying fermionic $(R_{\pi/8})^n$ gives $\phi = -n\pi/8$. The $|\bar{1}\rangle$ state has weight $7 \bmod 8$. Applying the $(R_{\pi/8})^n$ yields

$$\prod_i^n R_{\pi/8} |\bar{1}\rangle = e^{-i(n-7)\pi/8} e^{i7\pi/8} |\bar{1}\rangle \quad (113)$$

$$= e^{-in\pi/8} e^{i7\pi/4} |\bar{1}\rangle \quad (114)$$

$$= e^{-in\pi/8} e^{-i\pi/4} |\bar{1}\rangle. \quad (115)$$

When applied to a superposition of $|\bar{0}\rangle$ and $|\bar{1}\rangle$ states, this results in

$$\prod_i^n R_{\pi/8}(|\bar{0}\rangle + |\bar{1}\rangle) = e^{i\pi/8} |\bar{0}\rangle + e^{-i\pi/8} |\bar{1}\rangle, \quad (116)$$

which just shows that transversal $(R_{\pi/8})^n$ is indeed a logical gate of this code that corresponds to a logical $R_{-\pi/8}$ gate. Note that the logical operators of this code are odd, which implies that the above state can only be described with respect to a reference.

Generically, triorthogonal codes are used to define codes with a transversal T gate [56]. In the construction of such codes, the odd weight rows of a triorthogonal matrix are used to define logical qubits, and the odd weight is necessary for the correct anticommutation relations $\{\bar{X}_i, \bar{Z}_i\} = 0$. Here, we are promoting the binary matrix to correspond to Majorana CSS codes via Lemma 3, so we have a similar constraint to have odd weight binary strings for the logical operators, since $\bar{X}$ and $\bar{Z}$ are chosen to have support on γ part and $\bar{\gamma}$, respectively.

This code is also useful to implement universal fault-tolerant gates from using code-switching [57,58], since both $[[7, 1, 3]]$ and $[[15, 1, 3]]$ correspond to Majorana Steane code $[[7, 1, 3]]_f$ and Majorana Reed-Muller code $[[15, 1, 3]]_f$, where Steane code implements Clifford gates and Reed-Muller implements T gate. Switching can be implemented by measuring stabilizers of the respective code and applying gauge operators if a correction is required. Since gauge operators correspond to faces of the code and are of even weight, they can be measured without the need of a reference. While we do not show this, it is likely that higher distance 3D color codes can also be converted to Majorana Reed-Muller codes, since such codes are triorthogonal and automatically satisfy the self-orthogonality property required for Majorana codes.

A. Magic state distillation using this code

The Eastin-Knill theorem prevents a qubit quantum code to have transversal universal set of gates, and a similar restriction holds for Majorana codes, hence it would be important to do magic state distillation for Majorana codes as well. In Ref. [27], the authors propose to do magic state distillation on qubits and then use fermion-qubit gates to perform logical T_f gates on their logical fermions. We propose to instead perform magic state distillation natively on fermionic modes, thus removing the need to have fermion-qubit coupling gates.

A single magic state on a fermion mode is described as

$$|T_f\rangle = |0\rangle + e^{-i\pi/4} |1\rangle = R_{\pi/8} |+\rangle, \quad (117)$$

where the $|0\rangle, |1\rangle$ represent the fermion occupation basis. Note that a coherent superposition of $|0\rangle$ and $|1\rangle$ can

be accomplished using a reference state via a fermion Hadamard gate,

$$H |00\rangle_{rd} \rightarrow |00\rangle_{rd} + |11\rangle_{rd} \xrightarrow{\text{dequantized}} |0\rangle_r \frac{1}{\sqrt{2}} (|0\rangle_d + |1\rangle_d) \quad (118)$$

One can prepare 15 noisy T states by implementing simultaneous Hadamard gates as described in Sec. IV and applying $\pi/8$ rotations to each state

$$R_{\pi/8}^{\otimes 15} H^{\otimes 15} |0\rangle^{\otimes 15}. \quad (119)$$

Note that doing so still requires only one reference state call. To make this process robust, each noisy T state and the reference state can be encoded into an odd fermion code Q , thus corresponding to logical T states $|\bar{T}_f\rangle$.

Here is where the transversal T code will be useful: one measures the projectors of the $[[15, 1, 3]]$ on the 15 logical T states. We can use the fault-tolerant measurement gadget of the odd code Q ,

$$\Pi_{15} \bar{R}_{\pi/8}^{\otimes 15} |\bar{+}\rangle^{\otimes 15}. \quad (120)$$

The projectors commute with $R_{\pi/8}^{\otimes 15}$ since it is transversal and project $|\bar{+}\rangle^{\otimes 15}$ to $|\bar{+}\rangle$. If there are any errors on the code, they can be detected and the code discarded. Else, we can decode the inner $[[15, 1, 3]]_f$ to return an encoded $|T_f\rangle$ with improved fidelity.

The approach described above is for when encoding logical fermions. If the goal is to encode logical qubits in Majorana fermions, then it would be simpler to use qubit $[[15, 1, 3]]$ code and concatenate with an outer Majorana tetron code. While this would require 60 Majoranas, it will no longer need any reference state operations since the code state is even.

IX. MAJORANA CSS CODES

In Ref. [9], a qubit weakly self-dual CSS code is converted to a Majorana CSS code i.e., one code specifies the support of γ_i , while the other specifies the support of $\bar{\gamma}_i$. In such a construction, since qubit codes have to satisfy the constraint that $H_x H_z^T = 0 \pmod{2}$, codes for both γ and $\bar{\gamma}$ are the same. However, for Majorana codes this is actually not required in general, so we extend this construction here. Consider two classical codes C_1, C_2 , and their generator matrices correspond to G_1 and G_2 , respectively. The Majorana CSS construction is given as

$$S = \begin{bmatrix} G_1 & 0 \\ 0 & G_2 \end{bmatrix}. \quad (121)$$

The rows of the stabilizer matrix have to commute, but since the symplectic form is different for the Majoranas,

the constraint on the classical codes is also different. Consider a row $\mathbf{x}$ drawn from G_1 and $\mathbf{z}$ drawn from G_2 . For the rows to commute, we have the relation

$$[\mathbf{x} \mid 0] \Lambda_f \begin{bmatrix} 0 \\ \mathbf{z}^T \end{bmatrix} = 0. \quad (122)$$

The evaluation of the above gives

$$\sum_{i=1}^N x_i \sum_{j=1}^N z_j = 0. \quad (123)$$

This implies that either $\mathbf{x}$ or $\mathbf{z}$ has even weight, or both do. Since stabilizer measurements correspond to physical operations which are even weight, the typical choice is to have stabilizer generators with even weight, so we can further impose that both rows have even weight. Elements of stabilizers from the same classical code also should commute; therefore, we require that two vectors $\mathbf{x}_1$ and $\mathbf{x}_2$ from the same classical code fulfill

$$\mathbf{x}_1 \Lambda_f \mathbf{x}_2^T = 0, \quad (124)$$

which gives

$$\sum_{i=1}^N x_{1i} x_{2i} = 0. \quad (125)$$

By requiring that each row is even weight, Eq. (125) can be simplified to obtain

$$\sum_{i=1}^N x_{1i} x_{2i} = 0, \quad (126)$$

which is

$$\mathbf{x}_1 \cdot \mathbf{x}_2 = 0. \quad (127)$$

This implies that both codes C_1 and C_2 should be weakly self-dual, i.e., $C_1 \subseteq C_1^\perp$, $C_2 \subseteq C_2^\perp$. Therefore, one can use any two classical $[n, k_1, d_1]$ and $[n, k_2, d_2]$ self-orthogonal codes to construct a Majorana CSS code, $[[n, n - (k_1 + k_2), \min(d_1, d_2)]]_f$. In general, for degenerate Majorana codes, $d \geq \min(d_1, d_2)$. It is possible to have degenerate Majorana codes, e.g., high-distance Majorana color codes, where each stabilizer has weight six, but distance can be varied macroscopically and can be scaled as l . For such a case, product of two low weight detectable errors E and E' , which correspond to short strings of Majoranas, can give a plaquette operator, and thus lie inside S .

The Majorana LDPC code obtained below is constructed through Lemma 3, however, in general, one can expect to have an LDPC code construction that can be a Majorana

CSS code but not a qubit CSS code. Another important observation in this construction is that if $P_{\text{tot}} \in S$, then one cannot form logical fermions since in a CSS code, logical operators can be either of type γ or $\bar{\gamma}$ but not both, and in order for all of them to mutually anticommute, they must have operators of type γ and $\bar{\gamma}$.

X. MAJORANA QUANTUM LDPC CODE

For a Majorana code, a high-rate encoding would be beneficial when encoding multiple logical qubits or fermions while having a high distance. To obtain a Majorana LDPC code, one can always start with a qubit LDPC code and apply a “doubling” transformation, by concatenating each qubit with a tetron code. However, any such transformation will inevitably result in even Majorana codes. It is therefore interesting to find other ways to construct Majorana LDPC codes that cannot be obtained from qubit LDPC codes. Note that we have the following constraint on the stabilizer of Majorana codes

$$SS^T = 0. \quad (128)$$

This also precludes quantum LDPC codes generated through well-known product constructions, such as the hypergraph product code, which are not weakly self-dual. However, as described in the previous section, one can choose a classical self-orthogonal code and that would correspond to a Majorana LDPC code. Note that while there are families of random classical LDPC codes which have large distance with high probability, the odds of finding a random LDPC code that is also self-orthogonal are low. There is some existing literature on self-orthogonal classical LDPC codes that is applicable here [59,60].

One such construction is given in Ref. [61], which gives a classical weakly self-dual code, and can be converted to a quantum weakly self-dual code via [45]. One can then use Lemma 3 to convert to a Majorana code. We now summarize the construction and the key results.

The construction involves the use of **Cayley graphs**. Let there be a group G and a subset of that group S . A Cayley graph $\mathcal{C}(G, S)$ is defined as a graph whose vertex set is G , and there exists an edge between two vertices g, g' iff there exists $s \in S$ such that $g = sg'$.

If one chooses S such that S generates G , and S has the following properties:

- (1) The number of generators in S is even.
- (2) For all $g \in G$, there are an even number of distinct expressions of g s.t. $g = st^{-1}$, where $s, t \in S$.

Then the adjacency matrix $\mathcal{R}$ of the Cayley graph has the following properties:

- (1) Each row has even weight, since each element in G is connected to an even number of vertices.

- (2) Each row is orthogonal to any other row. Consider two elements $g_1, g_2 \in G$. In the adjacency matrix, they have a 1 in common iff $tg_1 = sg_2$, i.e., they are connected to the same element. This is equivalent to $g_1g_2^{-1} = st^{-1}$. Since $g_1g_2^{-1} \in G$, and there are an even number of expressions for any group element, the overlap is even.

The adjacency matrix $\mathcal{R}$ now has the property of self-orthogonality. This generates a classical code C such that

$$C \subseteq C^\perp. \quad (129)$$

To generate a Majorana code, we can either

- (1) Promote the generator matrix of this code to the stabilizer matrix of the Majorana code. By construction, this will have the following parameters: if the classical code C has parameters $(2n, k, d)$, then the Majorana code will have $[[n, n - k, d^\perp]]$, where $d^\perp$ is the dual distance given by the weight of smallest element in $C^\perp$.
- (2) Let $H_1 = \mathcal{R}$ and $H_2 = \mathcal{R}$. Now, the parameters, given a classical code $[n, k, d]$ (where $\mathcal{R}$ is generator matrix for the classical code), are $[[n, n - 2k, D^*]]$, where $D^* \geq d^\perp$.

We will use the second method in this paper to generate a Majorana LDPC code, and our main contribution here, is to show that the Majorana codes generated through this will have even weight.

In general, k and d of the classical code are given as

$$k = \dim(C) = \text{rank}(\mathcal{R}), \quad (130)$$

$$d = \min(\text{wt}(C \setminus C^\perp)) = \min(\text{wt}(\text{Im}(\mathcal{R}) \setminus \ker(\mathcal{R}))), \quad (131)$$

while for the Majorana code, the parameters are

$$K = \dim(C) = N - 2\text{rank}(\mathcal{R}) \quad (132)$$

$$D^* = \min(\text{wt}(C^\perp \setminus C)) = \min(\text{wt}(\ker(\mathcal{R}) \setminus \text{Im}(\mathcal{R}))). \quad (133)$$

Note that $d^\perp$ is too loose of a lower bound for distance of the code since the rows of generator matrix have low weight by design; therefore, the distance is computed directly. Finding these is nontrivial in general, but [61] computes these for the case of the repetition code. The construction is as follows:

- (1) Consider the parity check matrix H_n for a repetition code $[n + 1, 1, n + 1]$, with size $n \times (n + 1)$. The columns of this matrix correspond to S , while the group $G = \mathbb{F}_2^n$, i.e., all bitstrings of length n . This

gives a vertex set of size 2^n . Note that the size of S must be even, so a nontrivial LDPC code only exists for repetition codes with odd n .

- (2) Find the adjacency matrix corresponding to the above, represented as $\mathcal{R}(S_n, \mathbb{F}_2^n)$.

For odd n and $n \geq 3$, the rank is given as

$$\text{rank}(\mathcal{R}) = N - \ker(\mathcal{R}), \quad (134)$$

$$= 2^n - \left(2^{n-1} + 2^{\frac{n-1}{2}}\right). \quad (135)$$

Finally, the Majorana code parameters are

$$K = N - 2(\text{rank}(\mathcal{R})) = 2^{\frac{n+1}{2}}, \quad (136)$$

$$D^* = \min(\text{wt}(\ker(\mathcal{R}) \setminus \text{Im}(\mathcal{R}))) = 2^{\frac{n-1}{2}}. \quad (137)$$

Thus we have found a $[N, \sqrt{2N}, \sqrt{N/2}]$ code, which implies that it is high rate and high distance, in fact, the distance is comparable to a hypergraph product (HGP) construction [62], but k is $O(\sqrt{N})$ as opposed to $O(N)$ for HGP codes. Moreover, check weights also grow as $O(\log(N))$, which is a weaker notion of LDPC while for HGP codes, the checks are normally assumed to be constant weight.

One important question is whether this code encodes logical qubits or logical Majoranas. In order to check that, we can see whether $P_{\text{tot}} \in S$. If that is the case, then all logical operators can only be **even** weight. Note that if the block length of the classical code is odd, it is trivial to show that $P_{\text{tot}} \notin S$, but if block length is even, as in this case, it is not obvious whether $P_{\text{tot}} \in S$ or not. We show that $P_{\text{tot}} \in S$ in the following theorem.

Theorem 1. A Majorana code generated from the construction in Ref. [61] using the repetition code as the underlying classical code has only even weight logical operators.

Proof. In order to test whether a code has even or odd logical operators, we need to test whether the total parity is contained inside the stabilizer or not. We can look at two different cases, when n is odd and when n is even.

Once again, we have a Cayley graph $C(G, S)$, and the stabilizer matrix of our desired LDPC code corresponds to its adjacency matrix. For the repetition code, the parity check matrix has the form

$$H_n = [e_1, e_2, \dots, e_n, e_1 \oplus e_2 \oplus \dots \oplus e_n], \quad (138)$$

where checks correspond to the rows of H_n . e_i are the column vectors of length n and correspond to the canonical basis vectors of $\mathbb{F}_2^n$. The columns of H_n form the subset S while $G = \mathbb{F}_2^n$. Note that for $n + 1$, H_{n+1} is the same

except with an additional e_{n+1} vector added to above. The adjacency matrix is defined $A_{n+1} \equiv \mathcal{R}(\mathbb{F}_2^n, S_n)$ and in this case, it has the following nice recursive form because of the structure of the parity checks of repetition code

$$A_{n+1} = \begin{bmatrix} A_n + J_n & I_n + J_n \\ I_n + J_n & A_n + J_n \end{bmatrix}, \quad (139)$$

where J_n is the binary antidiagonal matrix,

$$[J_n]_{(i, 2^{n-1}-i)} = 1 \quad \text{for } i = \{0, 1, \dots, 2^{n-1}\}, \quad (140)$$

and I_n is the identity matrix, each of dimension 2^{n-1} . To see why one gets the above form, note the following:

Each element $\mathbf{x}$ in $\mathbb{F}_2^n$ can be mapped to an integer $\mathbf{p}$ in range $\{0, \dots, 2^n - 1\}$, corresponding to its lexicographic ordering where $\mathbf{x}$ is the binary representation of $\mathbf{p}$. For instance,

$\mathbf{x}$	$\mathbf{p}$
$\overbrace{00 \dots 0}^n$	0
00 ... 1	1
$\vdots$	$\vdots$
11 ... 1	$2^n - 1$

$\mathbf{p}$ indexes the position in the adjacency matrix A_{n+1} .

Note that the top left block of A_{n+1} corresponds to a binary antidiagonal matrix. This is because $\exists x \in \mathbb{F}_2^n$ such that we can map to another element $y \in \mathbb{F}_2^n$ by multiplying with $s \in S_n$ where s is the all ones vector. This basically swaps 0 and 1 in x . What this means is that:

$$[A_{n+1}]_{(p, 2^n-1-p)} = 1. \quad (141)$$

Hence we have J_n in the top right half and bottom left half of A_{n+1} . Consider an element $x \in \mathbb{F}_2^n$ whose n^{th} position from the right is 0. This corresponds to a $\mathbf{p}$ in range $\{0, \dots, 2^{n-1} - 1\}$. Adding an element e_n maps p to $p + 2^n$, which gives I in top right block, and symmetrically in the bottom left block.

Lastly, we need to remove the contribution of $e_1 + e_2 + \dots + e_n$ from A_n so we add J_n to the top left matrix, which as described above, corresponds to adding an all one vector.

For each column in A_{n+1} , the number of ones is given by

$$(\# \text{ of ones per col } A_{n+1}) = (\# \text{ of ones per col } A_n) + 1, \quad (142)$$

since for each column, the identity matrix contributes a single one, while the ones in J_n matrices cancel each other. In

addition, A_4 has four ones in each column, so in general, A_{n+1} has $n + 1$ ones in each column.

For odd $n + 1$, showing that P_{tot} is inside the group is trivial since each column has an odd number of ones, so adding all the rows mod 2 in general will give P_{tot} . Odd $n + 1$ also has a trivial logical dimension.

For even $n + 1$, let's look at the top half of the matrix

$$[A_n + J_n \quad I_n + J_n]. \quad (143)$$

The size of the above matrix is $2^{n-1} \times 2^n$, hence each submatrix is of size 2^{n-1} . Represent the left submatrix by L and the right submatrix by R .

Note now that the sum of the top half of R corresponds to the all ones vector of size 2^{n-1} . Thus the sum of the first 2^{n-2} rows of $[0 \mid R]$ gives the vector

$$[00 \dots 0 \mid 1 \dots 1] \quad (144)$$

of length 2^n , and the ones part is 2^{n-1} . Let us now look at the top half of the left submatrix L ,

$$L = [A_n + J_n]. \quad (145)$$

The sum of the all rows of top half (or the first 2^{n-2} rows) of J_n gives

$$[00 \dots 0, 11 \dots 1] \quad (146)$$

of length 2^{n-1} , where the ones part is of length 2^{n-2} . The comma separation indicates that we are looking at only the left submatrix. Let us look at the contribution from the top half of A_n . Note that top half of A_n is

$$[A_{n-1} + J_{n-1} \mid I_{n-1} + J_{n-1}]. \quad (147)$$

The right part of the above matrix is all zeros. Since A_n corresponds to the odd $n + 1$ case, from the previous argument we can conclude the sum of bottom left half of A_n gives 0, the sum of left columns gives

$$[11 \dots 1, 00 \dots 0] \quad (148)$$

of length 2^{n-1} , where the ones vector is of size 2^{n-2} . Combining (148) and (146) gives all ones in the right submatrix. Combining this with (145) gives the total parity P_{tot} inside the stabilizer and concludes the proof. ■

To look at an example for the sake of illustration, let us look at the $n + 1 = 4$ case. The adjacency matrix is

given as

$$A_4 = \begin{bmatrix} 0 & 1 & 1 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 & 1 & 1 & 0 \end{bmatrix}. \quad (149)$$

The parameters for this code are $N = 8, K = 2^2 = 4, D^* = 2$. Also each row has an even overlap with any other row, which gives it the self-orthogonal property. Also note that the rank of this matrix is two. We can use the CSS construction for Majoranas to obtain a Majorana CSS code with the following operators:

$$S_1 = \gamma_2 \gamma_3 \gamma_5 \gamma_8, \quad (150)$$

$$S_2 = \gamma_1 \gamma_4 \gamma_6 \gamma_7, \quad (151)$$

$$S_3 = \bar{\gamma}_2 \bar{\gamma}_3 \bar{\gamma}_5 \bar{\gamma}_8, \quad (152)$$

$$S_4 = \bar{\gamma}_1 \bar{\gamma}_4 \bar{\gamma}_6 \bar{\gamma}_7. \quad (153)$$

There are four logical operators, which can be found by computing the kernel of the stabilizer matrix,

$$\bar{X}_1 = \gamma_2 \gamma_3 \quad \bar{Z}_1 = \gamma_2 \gamma_5, \quad (154)$$

$$\bar{X}_2 = \gamma_1 \gamma_4 \quad \bar{Z}_2 = \gamma_1 \gamma_6, \quad (155)$$

$$\bar{X}_3 = \bar{\gamma}_2 \bar{\gamma}_3 \quad \bar{Z}_3 = \bar{\gamma}_2 \bar{\gamma}_5, \quad (156)$$

$$\bar{X}_4 = \bar{\gamma}_1 \bar{\gamma}_4 \quad \bar{Z}_4 = \bar{\gamma}_1 \bar{\gamma}_6. \quad (157)$$

This gives an $[[8, 4, 2]]_f$ code, with eight physical fermions. The above construction gives logical qubits, but one can also find logical fermions by choosing an anticommuting basis, although such logicals may not be of a CSS type. We note that there has been some subsequent work on Majorana LDPC codes that encode logical fermions [30]. The authors have suggested that, similar to our approach, one may use the literature on self-orthogonal classical codes and choose the odd examples from such codes, to construct an odd Majorana code. While they do demonstrate numerical evidence of such codes, it still seems unclear how to construct a family of odd Majorana codes and will be an interesting question to answer in the future.

XI. DISCUSSION AND CONCLUSION

In this paper, we have explored new directions of fault tolerance in fermionic codes. We have demonstrated that many of the standard techniques in qubit fault tolerance can be extended well to fermionic codes, and this gives us a toolkit of Clifford and non-Clifford transversal as well as measurement-based gadgets. In particular, a transversal

T gate for the $[[15, 1, 3]]_f$ allows native fermionic magic state distillation. We have emphasized the key differences between even and odd codes; even codes most naturally encode logical qubits, whereas odd codes can encode both logical qubits and logical fermions. This is an interesting direction to consider in algorithms where a combination of both data types, fermion and qubits, will be relevant. One such example is quantum phase estimation, where logical fermions are used for implementing the simulation algorithm and logical qubits as ancilla qubits; another can be simulation of lattice gauge theories [63]. One drawback of odd codes is that some of the physical operations are disallowed due to parity superselection rules. However, we demonstrate that a phase reference can be used to lift the superselection rule on the data modes. This allows new kinds of operations on fermionic modes, which we term as fermionic Hadamards and fermionic CNOTs. Again, we anticipate their usefulness in fermionic algorithm subroutines, e.g., variational quantum eigensolver (VQE), or to prepare a fermionic T state, where preparing superposition states is often critical. Another question which is a topic of intense research in qubit error correction is construction of LDPC codes; we also explore this topic and give the requirements for a Majorana LDPC code and give a family of even Majorana codes.

Our framework opens up many new research directions. We have introduced proof-of-concept techniques for carrying out logical gates transversally and accompanied with low-weight measurements. Many of these techniques were introduced for qubits early on, but then went through significant optimizations depending on hardware constraints, noise model, and capabilities. It would be very interesting to take our gadgets and adapt them to specific Majorana codes and architectures and evaluate numerical thresholds. Another tool that we have utilized is using quantum phase reference to lift super-selection rules; we hope this inspires more use of reference frame constructions for lifting symmetry constraints in other codes. We have also used BRAID₄ gates widely and while we think it is possible to have a unitary implementation, we have not been able to develop one that corresponds to topologically protected operation, so this is also an open direction. Moreover, we have used classical codes to develop even Majorana LDPC codes, and while there has been some recent progress [30] in finding odd Majorana codes, there are still some open directions in finding families of odd codes. We leave these for future work in this field.

ACKNOWLEDGMENTS

M.M. acknowledges useful discussions with Yifan Hong, Riley Chien, and Chris Feichsin. Both M.M. and D.G. are supported under RQS QLCI grant OMA-2120757. A.S. acknowledges support by the U.S. Department of Energy, Office of Science, National Quantum

Information Science Research Centers, and Quantum Systems Accelerator (QSA).

DATA AVAILABILITY

No data were created or analyzed in this study.

APPENDIX A: PARITY SUPERSELECTION AS A CONSTRAINT ON OPERATORS

We explained in Sec. IV why parity superselection obstructs certain operations. Here we give a concrete form for why unitaries (in the absence of reference frames) can only be physically realizable if they commute with the total parity.

Consider the density operator, which in general can take the form

$$\rho = \rho_e \oplus \rho_o. \quad (\text{A1})$$

Assume that the starting state is ρ_e , suppose we perform a parity switching unitary U such that we have ρ_o . We can represent a general unitary by the following form:

$$U = \begin{bmatrix} U_{ee} & U_{eo} \\ U_{oe} & U_{oo} \end{bmatrix}. \quad (\text{A2})$$

Conjugating $\rho = \rho_e$, with U , it is clear that one gets ρ_o corresponding to iff U_{eo} is nonzero. However, parity superselection rules only allows U to have a block diagonal structure, so we arrive at a contradiction. This means that we cannot implement parity switching operations unitarily.

APPENDIX B: DETAILS ON BRAID₄ IMPLEMENTATION

A Braiding transformation on a system of Ising anyons can be written in terms of F and R matrices acting on the Ising fusion space, and in particular,

$$\text{BRAID}_2 = FR^2F. \quad (\text{B1})$$

However, for the BRAID₄ case, it is likely that this corresponds to a fusion scenario, since F and R only correspond to basis transformations and do not change the number of outcomes. The simplest example of fusion corresponds to measurement, or projection operators, which are depicted as four-mode projectors. In the aforementioned scheme, we showed that a BRAID₄ can be written in terms of four-mode projectors, while using an ancilla mode. One can ask if it can be simplified to just two-mode projectors. We show that this is not possible, as follows.

Consider the starting state of the stabilizer group to be

$$S = \langle M_1, M_2, \dots \rangle. \quad (\text{B2})$$

For the sake of simplicity, these can be taken to be weight-two operators. The goal of a BRAID₄ is to increase the

weight of some stabilizer upon conjugation such that for any choice of generators, there is at least one element with weight four. We can prove that two-mode measurements are not sufficient to do this. Assume that one measures an observable N , which anticommutes with two generators (note that for the above choice of generators, it can anticommute with at most two). Then the updated stabilizer is

$$S = \langle M_1 M_2, N, \dots \rangle. \quad (\text{B3})$$

At this stage, multiplying $M_1 M_2 N$ gives an operator of weight two

$$|M_1 M_2 N| = |M_1 M_2| + |N| - 2|M_1 M_2 \cap N|, \quad (\text{B4})$$

$$= |M_1| + |M_2| + |N| - 4 = |M_1| + |M_2| - 2. \quad (\text{B5})$$

So this choice has still not changed the distribution of generators to the correct form. We can pick another generator, e.g., N' that anticommutes with N , but not with $M_1 M_2$, so as to remove N from the stabilizer as the next measurement; however, this will also overlap on an even number of sites with $M_1 M_2$, so the distribution does not change.

To see that BRAID₄ operators generate the logical BRAID₄ group, we can show that the generators satisfy the Yang-Baxter identity,

$$\sigma_i \sigma_{i+1} \sigma_i = \sigma_{i+1} \sigma_i \sigma_{i+1}. \quad (\text{B6})$$

The above identity is satisfied for the following choice of generators:

$$\sigma_i = \text{BRAID}_4(1, 2, 3, 4), \quad (\text{B7})$$

$$\sigma_{i+1} = \text{BRAID}_4(2, 3, 4, 5). \quad (\text{B8})$$

$$(\text{B9})$$

This is easy to check.

$$\begin{aligned} \text{BRAID}_4(1, 2, 3, 4) &= \frac{1}{\sqrt{2}}(1 + i\gamma_1 \gamma_2 \gamma_3 \gamma_4) \\ &= \frac{1}{\sqrt{2}}(1 + \Gamma_1 \Gamma_2), \end{aligned} \quad (\text{B10})$$

$$\begin{aligned} \text{BRAID}_4(2, 3, 4, 5) &= \frac{1}{\sqrt{2}}(1 + i\gamma_2 \gamma_3 \gamma_4 \gamma_5) \\ &= \frac{1}{\sqrt{2}}(1 + \Gamma_2 \Gamma_3), \end{aligned} \quad (\text{B11})$$

where $\Gamma_1 = \gamma_1$, $\Gamma_2 = i\gamma_2 \gamma_3 \gamma_4$, $\Gamma_3 = \gamma_5$

$$\sigma_i \sigma_{i+1} \sigma_i = \frac{1}{2\sqrt{2}}(1 + \Gamma_1 \Gamma_2)(1 + \Gamma_2 \Gamma_3)(1 + \Gamma_1 \Gamma_2). \quad (\text{B12})$$

Define $\Gamma_1 \Gamma_2 \equiv a_1$, $\Gamma_2 \Gamma_3 \equiv a_2$

$$(R \otimes I)(I \otimes R)(R \otimes I) = (1 + a_1)(1 + a_2)(1 + a_1) \quad (\text{B13})$$

$$(1 + a_1 + a_2 + a_1 a_2 + a_1 + a_1^2 + a_2 a_1 + a_1 a_2 a_1). \quad (\text{B14})$$

It can be verified that

$$a_1^2 = a_2^2 = -1, \{a_1, a_2\} = 0, \quad (\text{B15})$$

so the expression above gives

$$(R \otimes I)(I \otimes R)(R \otimes I) = 2a_1 + 2a_2, \quad (\text{B16})$$

which is symmetric under exchange of a_1 and a_2 , showing that the Yang-Baxter identity is satisfied. For the braid group, we have a set of n linearly independent Majorana operators and conjugation by a braid operator corresponds to a swap between n Majorana basis vectors. Similarly, BRAID_4 performs swaps between $2k$ logical Majorana operators Γ_i , which are defined as follows:

$$\Gamma_{2i+1} = \gamma_{4i+1} \quad \text{for } i = \{0, 1, \dots, k-1\}, \quad (\text{B17})$$

$$\Gamma_{2i+2} = i\gamma_{4i+2}\gamma_{4i+3}\gamma_{4i+4}. \quad (\text{B18})$$

BRAID_4 thus acts on logical Majorana operators as

$$\text{BRAID}_4^1(\Gamma_1)(\text{BRAID}_4^1)^\dagger = \Gamma_2, \quad (\text{B19})$$

$$\text{BRAID}_4^1(\Gamma_2)(\text{BRAID}_4^1)^\dagger = -\Gamma_1. \quad (\text{B20})$$

And correspondingly

$$\Gamma_1 \xleftrightarrow{\text{BRAID}_4^1} \Gamma_2 \xleftrightarrow{\text{BRAID}_4^2} \Gamma_3 \xleftrightarrow{\text{BRAID}_4^3} \dots \Gamma_{k-1} \xleftrightarrow{\text{BRAID}_4^k} \Gamma_k \quad (\text{B21})$$

modulo phases, where BRAID_4 gates are defined

$$\text{BRAID}_4^{2i} = \text{BRAID}_4(4i-2, 4i-1, 4i, 4i+1), \quad (\text{B22})$$

$$\text{BRAID}_4^{2i-1} = \text{BRAID}_4(4i-3, 4i-2, 4i-1, 4i). \quad (\text{B23})$$

APPENDIX C: DECODING EVEN AND ODD CODES AND LOGICAL JORDAN-WIGNER TRANSFORMATION

It is argued in Ref. [20] that when $P_{\text{tot}} \in S$, one needs an ancilla to find a decoding map. We justify this further for even codes and show that they can only be decoded partially.

Consider an even code, and one requires a decoding map $\mathcal{D}$ such that

$$\mathcal{D} : S_{\text{enc}} \rightarrow i\gamma_i \bar{\gamma}_i \quad i = \{0, \dots, n-k\}. \quad (\text{C1})$$

However, if such a map exists, then $P_{\text{tot}} \notin S$, and one will have at least one logical operator which is odd weight. But since a decoding map can only be constructed from even weight Cliffords, it is not possible to change the parity of logical operators. This means that either the logical operators have support outside of data qubits or that one can only partially decode such a code such that the stabilizer group after the partial decoding still consist of P_{tot} .

Consider now an odd code, which has been decoded completely and the stabilizer elements only have support over the first $2r$ modes. Since the logical operators must commute with the stabilizer, they can correspond to even or trivial support on the stabilizer modes. Suppose I choose a basis of logical operators (here I number the $2r+1$ as 1 and so on) such that the first mode is an odd operator and thus corresponds to a logical fermion, while the rest are even weight but correspond to logical fermions. We show how a **logical** Jordan-Wigner transformation on the remaining modes gives logical qubits.

$$\Gamma_1 = \gamma_1 \bar{\gamma}_1 \quad \bar{\Gamma}_1 = \gamma_1 \quad (\text{C2})$$

$$\Gamma_2 = \gamma_2 \gamma_3 \quad \bar{\Gamma}_2 = \gamma_2 \bar{\gamma}_3 \quad (\text{C3})$$

$$\vdots \quad (\text{C4})$$

$$\Gamma_k = \gamma_2 \gamma_k \quad \bar{\Gamma}_k = \gamma_2 \gamma_{k+1} \quad (\text{C5})$$

then I can perform a logical Jordan-Wigner transformation to obtain qubit operators as follows:

$$\Gamma_1 = \gamma_1 \bar{\gamma}_1 \quad \bar{\Gamma}_1 = \gamma_1 \quad (\text{C6})$$

$$\Gamma_2 = \gamma_2 \gamma_3 \quad \bar{\Gamma}_2 = \gamma_2 \bar{\gamma}_3 \quad (\text{C7})$$

$$\vdots \quad (\text{C8})$$

$$\Gamma_k = \gamma_2 \prod_{i=3} \gamma_i \bar{\gamma}_i \gamma_{k+1} \quad \bar{\Gamma}_k = \gamma_2 \prod_{i=3} \gamma_i \bar{\gamma}_i \bar{\gamma}_{k+1}. \quad (\text{C9})$$

We can convert Majoranas into qubits, using physical Jordan-Wigner transformation

$$\Gamma_2 = Y_2 X_3 \quad \bar{\Gamma}_2 = Y_2 Y_3 \quad (\text{C10})$$

$$\vdots \quad (\text{C11})$$

$$\Gamma_k = Y_2 X_{k+1} \quad \bar{\Gamma}_k = Y_2 X_{k+1}, \quad (\text{C12})$$

which represents the decoded qubits in the even parity sector.

APPENDIX D: ENCODING CIRCUIT WITH REFERENCE MODES

In Sec. IV we emphasized that for logical operators with odd weight, we need a reference state to violate parity superselection. This requires the preparation of a global parity invariant state. In other words, we have a parity constraint on the system and reference. A general parity invariant state on n parties can be written as

$$|\psi\rangle_{\text{inv}} = \left(\frac{1 + P_{\text{tot}}}{\sqrt{2^{n-1}}} \right) \sum_{\{i_1, i_2, \dots, i_n=0,1\}} |i_1, i_2, \dots, i_n\rangle, \quad (\text{D1})$$

where

$$|i_1, i_2, \dots, i_n\rangle = \gamma_1^{i_1} \gamma_2^{i_2} \dots \gamma_n^{i_n} |\text{vac}\rangle \quad (\text{D2})$$

where one of the parties, say the first one, is the reference, and the rest are the system.

For two modes, labeled R and A , respectively, denoting reference and Alice, the parity invariant state is

$$|\psi\rangle_{\text{inv}} = \frac{1}{\sqrt{2}} (|00\rangle_{RA} + |11\rangle_{RA}). \quad (\text{D3})$$

To prepare the general state in D1, one can use the circuit

$$\begin{aligned} C|00\dots 0\rangle &= \prod_{i=0}^{n-2} \text{BRAID}_2(1, n-i)^\dagger |00\dots 0\rangle \\ &= \text{BRAID}_2(1, n)^\dagger \text{BRAID}_2(1, n-1)^\dagger \\ &\quad \dots \text{BRAID}_2(1, 2)^\dagger |00\dots 0\rangle. \end{aligned} \quad (\text{D4})$$

For $n = 2$, where first mode now refers to reference and second to Alice This can be prepared using BRAID_2 gates, as follows:

$$|\psi\rangle_{\text{inv}} = \text{BRAID}_2(R, A)^\dagger (|00\rangle) \quad (\text{D5})$$

$$= \frac{|00\rangle_{RA} + |11\rangle_{RC}}{\sqrt{2}}. \quad (\text{D6})$$

For $n = 3$, as is given in Sec. IV,

$$|\psi\rangle_{\text{inv}} = \frac{1}{2} (|000\rangle_{RAC} + |011\rangle_{RAC} + |101\rangle_{RAC} + |110\rangle_{RAC}) \quad (\text{D7})$$

can be found using

$$|\psi\rangle_{\text{inv}} = \text{BRAID}_2^\dagger(R, C) \text{BRAID}_2^\dagger(R, A) |000\rangle. \quad (\text{D8})$$

APPENDIX E: JORDAN-WIGNER AS A LINEAR MAP

Quantum channels correspond to linear maps that are CPTP. One question that someone may have would be whether a logical transformation between logical operators that behave as qubits and logical operators that behave as a fermions is a physical map at all? We show that it is not a physical map since it is not CP.

An (inverse) Jordan Wigner transformation maps operators with qubit commutation relations to operators with fermionic anticommutation relations. In particular,

$$\Lambda(X_i) = \prod_{j < i} Z_j X_i, \quad (\text{E1})$$

$$\Lambda(Y_i) = \prod_{j < i} Z_j Y_i, \quad (\text{E2})$$

where Λ corresponds to the inverse Jordan Wigner map. It is clear that the input Pauli strings commute for different i , while output strings mutually anticommute, so a unitary transformation cannot actually do this. Next, to check if this map is CPTP, we can use the Pauli transfer matrix and Choi Jamiolkowski formalism. The Pauli transfer matrix form is

$$(R_\Lambda)_{ij} = \frac{1}{d} \sum \text{Tr}[P_i \Lambda(P_j)]. \quad (\text{E3})$$

For the sake of simplicity, we can evaluate for a two-qubit inverse Jordan Wigner transform. The matrix has the following expression:

$$(R_\Lambda)_{i,i} = 1 \quad i \notin \{4, 5, 7, 8, 10, 11, 13, 14\}, \quad (\text{E4})$$

$$(R_\Lambda)_{i,i} = 0 \quad i \in \{4, 5, 7, 8, 10, 11, 13, 14\}, \quad (\text{E5})$$

$$(R_\Lambda)_{4,7} = (R_\Lambda)_{7,4} = (R_\Lambda)_{5,8} = (R_\Lambda)_{8,5} = 1, \quad (\text{E6})$$

$$(R_\Lambda)_{10,13} = (R_\Lambda)_{13,10} = -(R_\Lambda)_{14,11} = (R_\Lambda)_{11,14} = -i. \quad (\text{E7})$$

One can then compute Choi matrix corresponding to this

$$\rho_{ij} = \frac{1}{d^2} \sum_{i,j} (R_\Lambda)_{ij} P_j^T \otimes P_i. \quad (\text{E8})$$

This can be evaluated to find that the eigenvalues of this matrix are not all positive, hence it is not CP, but it is still TP since the first row of R corresponds to $(100\dots 000)$, which is the necessary condition.

APPENDIX F: DETAILS OF THE SYNDROME VECTOR IN STEANE ERROR CORRECTION

As shown in Sec. VII, we have four blocks instead of two blocks as in standard Steane correction. This also

TABLE I. Input table.

s_d	s_2	s_3	s_4
1	0	0	0
0	1	0	0
0	0	1	0
0	0	0	1

TABLE II. Output table.

s_d	s_2	s_3	s_4
1	0	1	1
0	1	1	1
1	1	0	1
1	1	1	1

gives us extra information of the error source. For the sake of simplicity, we explain for γ errors and a similar idea follows for $\bar{\gamma}$ errors.

Suppose I start in the initial states as specified in Sec. IV, and before we implement the circuit, we were to measure the syndromes of each of the blocks in the occupation basis. For instance, if I had an error γ_i for $i = \{1, \dots, n\}$ on the data block, I will get a syndrome bit 1 for the data block on some j bit $j = \{1, 2, \dots, n - k\}$. In practice, I do not actually measure these syndromes, but for the sake of this explanation, we deem them as “input syndrome vectors.” In Table I, 1 denotes that one of the syndrome bit is flipped, while 0 denotes that no syndrome bit is flipped. We assume that a syndrome bit flip here means that there was an error in that block.

Table II shows the syndrome outputs on each block. Note that we do not actually measure s_d since the data block holds the information, rather the goal is to use information from s_2, s_3, s_4 to find the syndrome on the data block.

From Table II, it is now obvious that if all three ancilla syndrome vectors are flipped, then there is no error on data, but if two of them are flipped, then there is an error on data block. This is captured by the Boolean function given in Eq. (104). Moreover, since all four output strings are different even if we ignore the data syndrome, this means that we can distinguish between all four sources of syndromes, which is different from the standard qubit approach.

-
- [1] M. Aghaee, *et al.*, Interferometric single-shot parity measurement in InAs–Al hybrid devices, *Nature* **638**, 651 (2025).
- [2] T. Liu, C. Y. Wan, H. Yang, Y. Zhao, B. Xie, W. Zheng, Z. Yi, D. Guan, S. Wang, H. Zheng, C. Liu, L. Fu, J. Liu, Y. Li, and J. Jia, Signatures of hybridization of multiple Majorana zero modes in a vortex, *Nature* **633**, 71 (2024).

- [3] J. Alicea, New directions in the pursuit of Majorana fermions in solid state systems, *Rep. Prog. Phys.* **75**, 076501 (2012).
- [4] D. Aasen *et al.*, Roadmap to fault tolerant quantum computation using topological qubit arrays, [arXiv:2502.12252 \[quant-ph\]](#).
- [5] A. Y. Kitaev, Unpaired Majorana fermions in quantum wires, *Phys. Usp.* **44**, 131 (2001).
- [6] J. K. Pachos, *Introduction to Topological Quantum Computation* (Cambridge University Press, Cambridge, 2012).
- [7] P. Bonderson, M. Freedman, and C. Nayak, Measurement-only topological quantum computation, *Phys. Rev. Lett.* **101**, 010501 (2008).
- [8] A. Alase, M. C. Goffage, M. C. Cassidy, and S. N. Coppersmith, Decoherence of Majorana qubits by 1/f noise, [arXiv:2506.22394 \[cond-mat.mes-hall\]](#).
- [9] S. Bravyi, B. Leenhuis, and B. M. Terhal, Majorana fermion codes, *New J. Phys.* **12**, 083039 (2010).
- [10] M. B. Hastings, Small Majorana fermion codes, [arXiv:1703.00612 \[quant-ph\]](#).
- [11] D. Litinski and F. von Oppen, Quantum computing with Majorana fermion codes, *Phys. Rev. B* **97**, 205404 (2018).
- [12] C. McLauchlan and B. Béri, A new twist on the Majorana surface code: Bosonic and fermionic defects for fault-tolerant quantum computation, *Quantum* **8**, 1400 (2024).
- [13] Y. Li, Fault-tolerant fermionic quantum computation based on color code, *Phys. Rev. A* **98**, 012336 (2018).
- [14] A. Tran, A. Bocharov, B. Bauer, and P. Bonderson, Optimizing Clifford gate generation for measurement-only topological quantum computation with Majorana zero modes, *SciPost Phys.* **8**, 091 (2020).
- [15] 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).
- [16] Y. Li, Noise threshold and resource cost of fault-tolerant quantum computing with Majorana fermions in hybrid systems, *Phys. Rev. Lett.* **117**, 120403 (2016).
- [17] R. Chao, M. E. Beverland, N. Delfosse, and J. Haah, Optimization of the surface code design for Majorana-based qubits, *Quantum* **4**, 352 (2020).
- [18] S. Kundu and B. Reichardt, Majorana subsystem qubit codes that also correct odd-weight errors, *New J. Phys.* **26**, 073029 (2024).
- [19] V. Bettaque and B. Swingle, The structure of the Majorana Clifford group, [arXiv:2407.11319 \[quant-ph\]](#).
- [20] M. Mudassar, R. W. Chien, and D. Gottesman, Encoding Majorana codes, *Phys. Rev. A* **110**, 032430 (2024).
- [21] C. K. McLauchlan and B. Béri, Fermion-parity-based computation and its Majorana-zero-mode implementation, *Phys. Rev. Lett.* **128**, 180504 (2022).
- [22] D. Litinski and F. von Oppen, Braiding by Majorana tracking and long-range CNOT gates with color codes, *Phys. Rev. B* **96**, 205413 (2017).
- [23] T. E. O’Brien, P. Rožek, and A. R. Akhmerov, Majorana-based fermionic quantum computation, *Phys. Rev. Lett.* **120**, 220504 (2018).
- [24] Z. Z. Yan, B. M. Spar, M. L. Prichard, S. Chi, H.-T. Wei, E. Ibarra-García-Padilla, K. R. Hazzard, and W. S.

- Bakr, Two-dimensional programmable tweezer arrays of fermions, *Phys. Rev. Lett.* **129**, 123201 (2022).
- [25] D. González-Cuadra, D. Bluvstein, M. Kalinowski, R. Kaubuegger, N. Maskara, P. Naldesi, T. V. Zache, A. M. Kaufman, M. D. Lukin, H. Pichler, B. Vermersch, J. Ye, and P. Zoller, Fermionic quantum processing with programmable neutral atom arrays, *Proc. Natl. Acad. Sci.* **120**, e2304294120 (2023).
- [26] P. Bojović, T. Hilker, S. Wang, J. Obermeyer, M. Barendregt, D. Tell, T. Chalopin, P. M. Preiss, I. Bloch, and T. Franz, High-fidelity collisional quantum gates with fermionic atoms, *arXiv:2506.14711* [cond-mat.quant-gas].
- [27] A. Schuckert, E. Crane, A. V. Gorshkov, M. Hafezi, and M. J. Gullans, Fermion-qubit fault-tolerant quantum computing, *arXiv:2411.08955* [quant-ph].
- [28] R. Ott, D. González-Cuadra, T. V. Zache, P. Zoller, A. M. Kaufman, and H. Pichler, Error-corrected fermionic quantum processors with neutral atoms, *arXiv:2412.16081* [quant-ph].
- [29] A. J. Landahl and B. C. A. Morrison, Logical fermions for fault-tolerant quantum simulation, *arXiv:2110.10280* [quant-ph].
- [30] C.-Y. Xu, Z.-C. Liu, and Y. Xu, Fermion-to-fermion low-density parity-check codes, *arXiv:2508.15323* [quant-ph].
- [31] M. Chiew, B. Harrison, and S. Strelchuk, Ternary tree transformations are equivalent to linear encodings of the fock basis, *arXiv:2412.07578* [quant-ph].
- [32] F. Šimković IV, M. Leib, and F. R. F. Pereira, Low-weight high-distance error-correcting fermionic encodings, *Phys. Rev. Res.* **6**, 043123 (2024).
- [33] N. Maskara, M. Kalinowski, D. Gonzalez-Cuadra, and M. D. Lukin, Fast simulation of fermions with reconfigurable qubits, *arXiv:2509.08898*.
- [34] N. Constantinides, J. Yu, D. Devulapalli, A. Fahimniya, L. Schaeffer, A. M. Childs, M. J. Gullans, A. Schuckert, and A. V. Gorshkov, Low-depth fermion routing without ancillas, *arXiv:2510.05099* [quant-ph].
- [35] D. Zhang and T. Cubitt, Quantum error transmutation, *arXiv:2310.10278* [quant-ph].
- [36] X. Wang, E. Khatami, F. Fei, J. Wyrick, P. Namboodiri, R. Kashid, A. F. Rigosi, G. Bryant, and R. Silver, Experimental realization of an extended Fermi-Hubbard model using a 2D lattice of dopant-based quantum dots, *Nat. Commun.* **13**, 1 (2022).
- [37] A. Rad, A. Schuckert, E. Crane, G. Nambiar, F. Fei, J. Wyrick, R. M. Silver, M. Hafezi, Z. Davoudi, and M. J. Gullans, Analog quantum simulator of a quantum field theory with fermion-spin systems in silicon, *arXiv:2407.03419*.
- [38] D. Gottesman, *Surviving as a Quantum Computer in a Classical World* (University of Maryland, 2024), draft textbook for CMSC858G: Quantum Error Correction, <https://www.cs.umd.edu/~dgottesm/QECCbook-2024.pdf>.
- [39] B. Eastin and E. Knill, Restrictions on transversal encoded quantum gate sets, *Phys. Rev. Lett.* **102**, 110502 (2009).
- [40] C. Knapp, M. Beverland, D. I. Pikulin, and T. Karzig, Modeling noise and error correction for Majorana-based quantum computing, *Quantum* **2**, 88 (2018).
- [41] R. W. Chien and J. Klassen, Optimizing fermionic encodings for both hamiltonian and hardware, *arXiv:2210.05652* [quant-ph].
- [42] N. T. Vidal, M. L. Bera, A. Riera, M. Lewenstein, and M. N. Bera, Quantum operations in an information theory for fermions, *Phys. Rev. A* **104**, 032411 (2021).
- [43] S. D. Bartlett, T. Rudolph, and R. W. Spekkens, Reference frames, superselection rules, and quantum information, *Rev. Mod. Phys.* **79**, 555 (2007).
- [44] S. Vijay and L. Fu, Quantum error correction for complex and Majorana fermion qubits, *arXiv:1703.00459* [cond-mat.mes-hall].
- [45] A. M. Steane, Error correcting codes in quantum theory, *Phys. Rev. Lett.* **77**, 793 (1996).
- [46] L. H. Kauffman, Majorana fermions and representations of the braid group, *Int. J. Mod. Phys. A* **33**, 18300235 (2018).
- [47] D. Jaksch, J. I. Cirac, P. Zoller, S. L. Rolston, R. Côté, and M. D. Lukin, Fast quantum gates for neutral atoms, *Phys. Rev. Lett.* **85**, 2208 (2000).
- [48] C. Moore and M. Nilsson, Parallel quantum computation and quantum codes, *arXiv:quant-ph/9808027* [quant-ph].
- [49] G. Gour and R. W. Spekkens, The resource theory of quantum reference frames: Manipulations and monotones, *New J. Phys.* **10**, 033023 (2008).
- [50] E. Kubischta and I. Teixeira, Family of quantum codes with exotic transversal gates, *Phys. Rev. Lett.* **131**, 240601 (2023).
- [51] A. R. Calderbank, E. M. Rains, P. W. Shor, and N. J. A. Sloane, Quantum error correction via codes over $\text{gf}(4)$, *arXiv:quant-ph/9608006* [quant-ph].
- [52] H. Sayginel, S. Koutsoumpas, M. Webster, A. Rajput, and D. E. Browne, Fault-tolerant logical Clifford gates from code automorphisms, *arXiv:2409.18175* [quant-ph].
- [53] N. Berthussen, M. J. Gullans, Y. Hong, M. Mudassar, and S. J. S. Tan, Automorphism gadgets in homological product codes, *arXiv:2508.04794* [quant-ph].
- [54] D. Gottesman, Theory of fault-tolerant quantum computation, *Phys. Rev. A* **57**, 127 (1998).
- [55] S. B. Bravyi and A. Y. Kitaev, Fermionic quantum computation, *Ann. Phys.* **298**, 210 (2002).
- [56] S. Bravyi and J. Haah, Magic-state distillation with low overhead, *Phys. Rev. A* **86**, 052329 (2012).
- [57] J. T. Anderson, G. Duclos-Cianci, and D. Poulin, Fault-tolerant conversion between the steane and reed-muller quantum codes, *Phys. Rev. Lett.* **113**, 080501 (2014).
- [58] F. Butt, S. Heußen, M. Rispler, and M. Müller, Fault-tolerant code-switching protocols for near-term quantum processors, *PRX Quantum* **5**, 020345 (2024).
- [59] J. Farinholt, Quantum LDPC codes constructed from point-line subsets of the finite projective plane, *arXiv:1207.0732* [quant-ph].
- [60] D. J. C. MacKay, G. Mitchison, and P. L. McFadden, Sparse-graph codes for quantum error correction, *IEEE Trans. Inf. Theory* **50**, 2315 (2004).
- [61] A. Couvreur, N. Delfosse, and G. Zémor, A construction of quantum LDPC codes from Cayley graphs, *IEEE Trans. Inf. Theory* **59**, 6087 (2013).
- [62] J.-P. Tillich and G. Zémor, Quantum LDPC codes with positive rate and minimum distance proportional to the square root of the blocklength, *IEEE Trans. Inf. Theory* **60**, 1193 (2014).
- [63] L. Spagnoli, A. Roggero, and N. Wiebe, Fault-tolerant simulation of lattice gauge theories with gauge covariant codes, *Quantum* **10**, 1968 (2026).