

# Magic tricycles: Efficient magic state generation with finite block-length quantum LDPC codes

Varun Menon<sup>\*†</sup>, J. Pablo Bonilla Ataides<sup>\*</sup>, Rohan Mehta, Andi Gu, Daniel Bochen Tan,  
and Mikhail D. Lukin

Department of Physics, Harvard University, Cambridge, MA 02138, USA

November 10, 2025

## Abstract

The preparation of high-fidelity non-Clifford (magic) states is an essential subroutine for universal quantum computation, but imposes substantial space-time overhead. Magic state factories based on high rate and distance quantum low-density parity check (LDPC) codes equipped with transversal non-Clifford gates can potentially reduce these overheads significantly, by circumventing the need for multiple rounds of distillation and by producing a large number of magic states in a single code-block. As a step towards realizing efficient, fault-tolerant magic state production, we introduce a class of finite block-length quantum LDPC codes which we name tricycle codes, generalizing the well-known bicycle codes to three homological dimensions. These codes can support constant-depth physical circuits that implement logical  $CCZ$  gates between three code blocks. To construct these constant-depth  $CCZ$  circuits, we develop new analytical and numerical techniques that apply to a broad class of three-dimensional homological and balanced product codes. We further show that tricycle codes enable single-shot state-preparation and error correction, leading to a highly efficient magic-state generation protocol. Numerical simulations of specific codes confirm robust performance under circuit-level noise, demonstrating a high circuit-noise threshold of  $> 0.5\%$ . With modest post-selection, certain tricycle codes of block-lengths of only  $50 - 100$  qubits are shown to achieve logical error-rates of  $6 \times 10^{-10}$  or lower. Finally, we construct optimal depth syndrome extraction circuits for tricycle codes and present a protocol for implementing them efficiently on a reconfigurable neutral atom platform.

---

<sup>\*</sup>These authors contributed equally to this work.

<sup>†</sup>varunmenon@g.harvard.edu

# Contents

|          |                                                                                                    |           |
|----------|----------------------------------------------------------------------------------------------------|-----------|
| <b>1</b> | <b>Introduction</b>                                                                                | <b>3</b>  |
| 1.1      | Background . . . . .                                                                               | 3         |
| 1.2      | Summary of results . . . . .                                                                       | 5         |
| 1.3      | Related work . . . . .                                                                             | 6         |
| <b>2</b> | <b>Main results</b>                                                                                | <b>7</b>  |
| 2.1      | Tricycle codes . . . . .                                                                           | 7         |
| 2.2      | Transversal $CCZ$ circuits . . . . .                                                               | 9         |
| 2.3      | Single-shot distillation . . . . .                                                                 | 12        |
| 2.4      | Circuit-noise simulations . . . . .                                                                | 15        |
| 2.5      | Implementation of syndrome extraction circuits . . . . .                                           | 17        |
| <b>3</b> | <b>Discussion and Outlook</b>                                                                      | <b>19</b> |
| <b>4</b> | <b>Acknowledgments</b>                                                                             | <b>20</b> |
| <b>A</b> | <b>Code construction and properties</b>                                                            | <b>20</b> |
| <b>B</b> | <b>Single-shot guarantees</b>                                                                      | <b>26</b> |
| <b>C</b> | <b>Determining code distances</b>                                                                  | <b>29</b> |
| <b>D</b> | <b>Symmetric triple cup-product and transversal <math>CCZ</math> action</b>                        | <b>31</b> |
| <b>E</b> | <b>Numerical Leibniz rule for <math>CCZ</math> gates on <math>3D</math> balanced-product codes</b> | <b>38</b> |
| <b>F</b> | <b>Logical circuit optimization</b>                                                                | <b>41</b> |
| <b>G</b> | <b>Numerical details of noise simulations</b>                                                      | <b>42</b> |
| <b>H</b> | <b>Optimal-depth syndrome extraction circuits</b>                                                  | <b>44</b> |
| <b>I</b> | <b>Neutral atom array architecture and rearrangement procedure</b>                                 | <b>48</b> |
|          | <b>Supplementary Material</b>                                                                      | <b>60</b> |

# 1 Introduction

## 1.1 Background

Universal fault-tolerant quantum computation requires the implementation of both Clifford and non-Clifford operations on the logical qubits of a quantum error correcting code [39]. Logical Clifford gates can often be implemented *transversally* in many codes – that is, by a physical circuit with a depth that is constant in the size of the code. Transversal gate architectures are naturally fault tolerant as the spread of errors between physical qubits of the code is inherently bounded. However, a seminal result of Eastin and Knill proves that it is not possible to implement a universal gate set transversally in a quantum error correcting code [32]. Since it is desirable to implement as many gates transversally as possible, many architectures for fault-tolerant quantum computation utilize codes with a transversal Clifford gate set, while non-Clifford gates, also known as *magic gates*, are implemented by teleporting specially prepared non-Clifford resource states into the codespace. One class of protocols for the fault-tolerant preparation of such high-fidelity resource states, known as magic state distillation (MSD), relies on quantum error correcting codes with a transversal magic gate to refine a large number of noisy input magic states into a smaller number of output states with improved fidelity. To achieve the target output fidelity, typical MSD schemes such as the 15-to-1 distillation (15-1-3) protocol [17] require multiple levels of repeated distillation, as well as concatenation with an inner code of sufficiently large distance that supports transversal Clifford gates, contributing significantly to the space-time overhead of the protocol. As such, in state-of-the-art fault-tolerant architectures, the overhead associated with magic state distillation has been estimated to dominate total qubit and gate counts – for example, magic state generation dominates the latest resource estimates of factoring RSA integers with Shor’s algorithm despite significant optimizations [36, 113].

Over the last decade, remarkable progress has been made towards significantly reducing the overhead associated with magic state distillation by designing various codes with transversal non-Clifford gates and favorable code parameters [16, 54, 43, 23]. In fact, it has very recently been shown that families of quantum codes with transversal non-Clifford gates and asymptotically constant rate and relative distance exist, leading in principle to constant spatial overhead MSD protocols [73, 106]. However, these proposals come with two important caveats. First, the proposed codes in such low-overhead distillation schemes have high-weight parity checks with check weights that are typically extensive in the block length of the code. As such these codes may not be obviously fault-tolerant on their own in the presence of circuit-level noise as they require syndrome extraction circuits with depths that grow with the block-length, and thus may not support a finite threshold or exponential sub-threshold error suppression with bare-ancilla based syndrome extraction. Second, these schemes implicitly assume that Clifford operations are noiseless to execute the Clifford encoding and unencoding circuits of the codes [17] – an assumption that is justified when concatenating the distillation code with an inner code that has transversal Clifford operations such as the two-dimensional color code. Practically, the overhead of these protocols is therefore exaggerated by the requirement of a sufficiently high distance inner code so that the distillation protocol is not bottlenecked by the error-rate of Clifford operations. Moreover, the Clifford encoding and unencoding circuits can have depths scaling linearly in the number of physical qubits of the code, leading to significant temporal overhead of such protocols.

Quantum low-density parity-check (qLDPC) codes, which simultaneously admit sparse stabilizer

checks and favorable rate and distance scaling, have been proposed as a pathway toward reducing the cost of quantum error correction. In the context of quantum memory, qLDPC codes with asymptotically constant rates and relative distances have recently been shown to exist [78, 59], and several architectures for implementing qLDPC memories on superconducting qubit and neutral-atom architectures have been proposed [7, 107, 109, 15]. However, the application of general qLDPC codes to magic state generation protocols has only recently begun to be explored, in part because of the difficulty of constructing codes that host transversal non-Clifford gates. Utilizing appropriate qLDPC codes could dramatically reduce the overhead of MSD protocols for a number of reasons. Codes with high relative distances ensure effective error suppression during distillation, while high rate codes allow multiple magic states to be distilled in parallel within a single block, improving throughput. In contrast to traditional MSD schemes that rely on small codes concatenated many times, LDPC codes can potentially achieve the same target fidelity in a single round, greatly reducing space-time overhead. While this is also true for the aforementioned non-LDPC constant-overhead and low-overhead MSD protocols, crucially, qLDPC codes enable these properties while maintaining fault-tolerance with constant-depth syndrome extraction circuits. Moreover, for qLDPC codes with strongly-transversal non-Clifford gates (i.e a constant depth physical circuit of non-Clifford gates without a Clifford correction), there is no need for concatenation with an inner code with transversal Clifford gates. Constructions of practical LDPC magic state factories are thus highly desirable for fault-tolerant quantum computing architectures.

Motivated by these considerations, this work introduces a class of finite block-length, high-rate and distance quantum LDPC codes with a transversal logical  $CCZ$  circuit action for magic state distillation, which we call ‘tricycle codes’. These codes are constructed as the three-dimensional *balanced product* [19], a particular generalization of the homological product, of classical binary linear codes over Abelian group algebras [48]. This construction extends the two-block group algebra codes of [53, 62] (also known as bicycle codes [76]) into three dimensions, allowing in principle for transversal gates from the third level of the Clifford hierarchy [40]. We may thus also refer to the tricycle codes as ‘three-block Abelian group-algebra codes’. We highlight that for the task of magic state distillation, high-rate and high-distance finite block-length codes are sufficient for essentially any large-scale quantum computation. For example, codes with distances large enough to achieve a logical error rate of  $10^{-12}$  at attainable physical error rates are expected to be sufficient to produce magic states to execute large-scale quantum algorithms such as Shor’s algorithm [89] fault-tolerantly.

The theoretical formalism we utilize for constructing transversal non-Clifford gates in three-dimensional product codes comes from endowing the codes with a *cup-product*, an operation from algebraic topology and homological algebra, related to the triple-intersection of logical operators of the code, which we discuss further in Sec. 2.2 and Sec. D. At a high level, this formalism allows general product quantum CSS codes to be equipped with the necessary structure to host transversal gates ascending the Clifford hierarchy. In fact, the transversal non-Clifford gates of three-dimensional color codes can also be interpreted in terms of cup-products on the associated three-dimensional cellular complexes [11, 55]. Recent work by Breuckmann et al. introduced a general formalism to endow the cochain complexes associated with homological product and balanced product quantum codes with cup-product operations and classified the associated gates they give rise to [18]. In this work, we first present a modification of this formalism, leading to a set of conditions that high rate and distance tricycle codes with non-trivial  $CCZ$  action must satisfy (see Sec. D). The conditions resulting from this modified cup-product construction are essential to finding tricycle codes with both high rates and relative distances. We then study the transversal

non-Clifford circuits hosted by codes which satisfy these conditions in detail. More generally, we also introduce a new numerical method that is inspired by the cup-product construction for finding short-depth code-space preserving non-Clifford circuits on iterated balanced-product quantum codes, which we also apply to certain tricycle codes. We present several concrete examples of such high rate and distance tricycle codes, study their performance under realistic circuit level noise, and explore their application to practical magic state distillation protocols.

## 1.2 Summary of results

The tricycle codes we construct achieve favorable parameters at relatively small block lengths, while hosting constant-depth circuits that implement a logical  $CCZ$  action. We introduce several classes of tricycle codes with check weights  $w_x \leq 12$ ,  $w_z \leq 8$ , where  $w_x$  and  $w_z$  denote the weights of the  $X$  and  $Z$  checks of the code respectively. We use the notation  $[[N, K, D]]$  to report the block-length  $N$ , number of encoded logical qubits  $K$ , and minimum distance  $D$  of quantum codes. Some examples of codes we find have parameters  $[[48, 6, 4]]$ ,  $[[108, 21, 6]]$ ,  $[[270, 24, 8]]$ ,  $[[270, 12, 12]]$ ,  $[[480, 12, \leq 17]]$ , exhibiting highly competitive distances and rates that are compatible with a transversal non-Clifford action. A more complete list of codes along with the depths of the non-Clifford transversal circuits they host can be found in Table 1.

We show that tricycle codes host constant depth physical circuits that preserve the code-space, consisting solely of  $CCZ$  gates between three code blocks. The logical action of the  $CCZ$  circuits depends on a choice of a representative logical operator basis in each code-block, but is generically a circuit of logical  $CCZ$  gates. When such a circuit acts on the  $\overline{|+, +, +\rangle}^{\otimes K}$  logical state of three code-blocks it produces a high magic state known as a *hypergraph magic state* [114, 24]. Such a hypergraph magic state embeds  $K_{CCZ} \leq K$  many disjoint  $\overline{CCZ} \cdot |+, +, +\rangle$  magic states which can be extracted using gate-teleportation techniques. The physical  $CCZ$  circuit, factors determining their depths, and the logical action they induce for particular codes are discussed in Sec. 2.2 and Sec. D.

We further show that the tricycle codes are single-shot [9] in the  $Z$ - measurement basis. We show in Sec. 2.3 that this property enables a distillation protocol that requires only a constant number of rounds of syndrome extraction during error correction cycles instead of  $O(d)$  rounds, where  $d$  is the code distance. Moreover, we present numerical evidence that tricycle codes exhibit the stronger notion of single-shot state-preparation, allowing the initial logical state of a magic state preparation circuit to be prepared in constant depth. To the best of our knowledge, this is the first example of such a single-shot magic state distillation protocol.

We analyze the performance of select tricycle codes under a realistic circuit level noise model. By comparing the performance of codes across a range of block-lengths, we are able to ascertain a threshold of 0.5% under circuit-level noise with an MLE decoder for a certain family of tricycle codes. We also study how post-selection and full error-detection can improve the performance of these codes. To do so, we utilize a recently developed efficient cluster-based post-selection method for quantum LDPC codes [57] combined with Belief-Propagation + Localized Statistics Decoding (BP+LSD) [46]. With full error detection, we show that even the smallest tricycle code we study in detail (with parameters  $[[48, 6, 4]]$ ) can achieve logical error rates of approximately  $6 \times 10^{-10}$  at an acceptance fraction of roughly 30%. For a larger code, we estimate logical error rates  $< 10^{-12}$  with full error detection at a  $\approx 9\%$  acceptance fraction, compatible with full-scale fault tolerant quantum

computation. Note that these logical error rates are achieved in a numerical simulation of the code as a memory under multiple rounds of error correction (with post-selection). In particular, they do not account for the error introduced by the  $CCZ$  circuit when determining the fidelity of the final magic-state. However, since the  $CCZ$  circuits are relatively short (depth 8 for the codes discussed above), we anticipate that these estimates from the logical memory performance are a close proxy to the fidelity of the target magic-state – we discuss this perspective in more detail in Sec. 2.4.

We note that although the codes we present are finite-block length codes, the construction allows us to define a family of codes by choosing a single high-performing tricycle code from a given block-length. Although such a code family is unlikely to be asymptotically good, we expect that a larger block-length code with a larger distance can always be found – a claim that is supported by the distance lower bound we prove in Sec. A. Defining a threshold with respect to such a constructed code family allows one to make guarantees of sub-threshold noise suppression by selecting codes of arbitrarily large block-lengths with larger distances.

We then show how to construct compact syndrome extraction circuits of optimal depth for all tricycle codes. Finally, we present a concrete protocol for efficiently implementing the preparation and syndrome extraction of these codes on a reconfigurable neutral-atom array platform – a video illustrating the atom-moving scheme for syndrome extraction of tricycle codes is provided in the ancillary files that accompany this manuscript.

We also clarify some subtle points about the application of the cup-product gate formalism of [18]. In particular, the success of this formalism for producing transversal non-Clifford circuits on tricycle codes with high rate and distance requires a generalization of the conditions presented in section 5 of [18]. These resulting conditions will likely also be useful for other homological product and balanced product code families. We discuss these points on the theory of cup-product based gates in Sec. D. In Sec. E, we also present a complementary numerical method that generalizes the cup-product construction for finding code-space preserving  $CCZ$  circuits on three-dimensional balanced-product codes.

Our work serves as a step toward a new class of distillation protocols that leverage the structural properties of quantum LDPC codes to reduce resource overhead in universal fault-tolerant computation. Such magic-state generation schemes are naturally suited to fault-tolerant quantum computing architectures that utilize LDPC quantum memories along with universal adapters for magic state injection, such as in Ref. [109, 110] or using efficient code-switching protocols with lower-dimensional product codes [60, 101, 37].

### 1.3 Related work

Several recent works introduce binary quantum LDPC code families with asymptotically high rates and relative distances with transversal logical  $CCZ$  action. Golowich and Lin [38] constructed a family of codes based on homological products of classical Sipser-Spielman expander codes [91] with local Reed-Muller codes [80, 71] with near-constant rate and relative distance that supports transversal logical non-Clifford gates. These codes however have check-weights that are not constant but grow logarithmically in the block-length. Lin also introduced a general formalism for constructing LDPC quantum code families with transversal non-Clifford gates based on algebraic sheaves [63]. Similarly, Zhu introduced a family of topological codes with constant rate and polynomial distance with transversal  $CCZ$  gates [114, 115]. Quantum rainbow codes were introduced as generalizations

of color-codes with transversal non-Clifford gates to higher-dimensional simplicial complexes [88], achieving asymptotically constant rate and logarithmic distance. Finally, Breuckmann et al. [18] discuss how to construct constant rate and polynomial distance families with transversal gates that ascend the Clifford hierarchy using iterated homological and balanced products of chain complexes and the cup-product gate formalism they develop.

Adding to these results, this work focuses on a particular class of high-performance finite block-length codes. It is unclear precisely how large the smallest elements of the code families in the aforementioned works are, but constructions based on three-fold homological products of classical codes likely involve at-least several thousand and potentially tens of thousands of physical qubits for the smallest attainable block-lengths with good parameters, due to the cubic growth of the size of the quantum code with respect to the classical codes. The significantly smaller block-lengths of the tricycle codes presented in this work is one of the primary reasons we chose to study them. A direct comparison of the overhead and performance of the LDPC codes from the above constructions with each other and the tricycle codes we present would be an interesting direction for future work.

## 2 Main results

### 2.1 Tricycle codes

Throughout this work, we denote the binary field by  $\mathbb{F}_2 \equiv \{0, 1\}$  where arithmetic is modulo 2. The tricycle codes are quantum CSS codes [21, 94] defined by the binary block parity check matrices

$$H_X = [\mathbf{A}^T \quad \mathbf{B}^T \quad \mathbf{C}^T] \in \mathbb{F}_2^{n_G \times 3n_G} \quad H_Z = \begin{bmatrix} \mathbf{C} & \mathbf{0} & \mathbf{A} \\ \mathbf{0} & \mathbf{C} & \mathbf{B} \\ \mathbf{B} & \mathbf{A} & \mathbf{0} \end{bmatrix} \in \mathbb{F}_2^{3n_G \times 3n_G} \quad (1)$$

where  $\mathbf{A} = \sum_{i=1}^{w_a} \mathbf{A}_i \in \mathbb{F}_2^{n_G \times n_G}$ ,  $\mathbf{B} = \sum_{i=1}^{w_b} \mathbf{B}_i \in \mathbb{F}_2^{n_G \times n_G}$ ,  $\mathbf{C} = \sum_{i=1}^{w_c} \mathbf{C}_i \in \mathbb{F}_2^{n_G \times n_G}$  for constants  $w_a, w_b, w_c$ , and  $\mathbf{A}_i, \mathbf{B}_i, \mathbf{C}_i$  are permutation matrices – i.e matrices with a single 1 in every row and column. The parameter  $n_G$  is the linear size of each block and is related to the size of a finite group (hence the subscript  $G$ ) – the details of this connection are discussed in Sec. A. These parity check matrices thus define codes on  $N = 3n_G$  qubits. Modulo two arithmetic implies immediately that the CSS orthogonality condition  $H_Z H_X^T = 0$  is satisfied. Moreover, the matrices  $\mathbf{A}, \mathbf{B}, \mathbf{C}$  commute pair-wise. There are  $n_G$  many  $X$ -checks of weight  $w_a + w_b + w_c$  and  $3n_G$  many  $Z$ -checks which can be partitioned into three sets of  $n_G$  checks each with weights  $w_c + w_a$ ,  $w_c + w_b$ , and  $w_b + w_a$  respectively. By choosing  $w_a, w_b, w_c \leq w$  for a constant  $w$ , the resulting codes are  $(\leq w, \leq 3w)$ -LDPC codes in the  $X$  basis and  $(\leq 2w, \leq 2w)$ -LDPC in the  $Z$  basis, where  $(p, q)$ -LDPC means that every qubit is involved in at most  $p$  checks and each check involves at most  $q$  qubits. Examples of particular tricycle codes in Table 1 demonstrate that the checks are typically highly redundant – often leading to favorable code parameters and ease of decoding – since the total number of checks  $4n_G \gg N - K = 3n_G - K$ . This observation is encapsulated by the fact that tricycle codes possess *metachecks* in the  $Z$  basis – these are relations between the  $Z$  checks encoded in a matrix  $H_{meta}$  such that  $H_{meta} \cdot H_z = 0$ . We discuss the metacheck structure and its implications in Sec. 2.3.

To ensure practical feasibility of implementing these codes in early fault-tolerant architectures below pseudo-threshold error rates, we restrict our attention to tricycle codes with a maximum

check weight at most 12. More specifically, we consider three types of tricycle codes. The first is constructed with  $w_a = 4$ ,  $w_b = w_c = 2$ , producing codes with weight 8  $X$ -checks, and  $Z$ -checks of weight 4 and 6 – we call these codes  $4 - 2 - 2$  codes, referring to the weights  $w_a, w_b, w_c$ . The second class of codes we consider has  $w_a = w_b = 4$ ,  $w_c = 2$ , and therefore has  $X$ -checks of weight 10 and  $Z$ -checks of weight 8 and 6 – we refer to these as  $4 - 4 - 2$  codes. Finally, we consider codes with  $w_a = w_b = w_c = 4$ , for which the  $X$ -checks have weight 12 and the  $Z$ -checks have weight 8 – we call these  $4 - 4 - 4$  codes. One may similarly consider  $2 - 2 - 2$  codes with weight 6  $X$ -checks and weight 4  $Z$ -checks, but we empirically found that the rates and distances of such codes with a non-trivial transversal  $CCZ$  action are not as favorable. In particular, we could not find any such codes with  $K > 3$  encoded logical qubits and distance  $D > 7$  – such  $2 - 2 - 2$  codes with  $K = 3$  encoded logical qubits and distance  $D \leq 7$  have been studied in other recent works [51, 60].

The reason we consider  $w_a, w_b, w_c$  to be 2 or 4 is that the specific conditions for tricycle codes to host transversal  $CCZ$  circuits (see Sec. D) cannot be satisfied when any of  $w_{a,b,c} = 3$ , and  $w_{a,b,c} = 1$  leads to a trivial code. One may consider  $w_{a,b,c} > 4$  to search for codes with even more favorable parameters than the ones we present here, but we empirically found  $6 - 2 - 2$  codes to have parameters no better than  $4 - 2 - 2$  and  $4 - 4 - 2$  codes, while going to even higher check-weight will likely lead to poor thresholds.

| N   | K  | $D_X$     | $D_Z$     | Type        | CCZ depth | CCZ method |
|-----|----|-----------|-----------|-------------|-----------|------------|
| 48  | 6  | 8         | 4         | $4 - 2 - 2$ | 8         | STCP       |
| 84  | 6  | 12        | 5         | $4 - 2 - 2$ | 8         | STCP       |
| 108 | 6  | 12        | 6         | $4 - 2 - 2$ | 8         | STCP       |
| 240 | 6  | $\leq 22$ | 8         | $4 - 2 - 2$ | 8         | STCP       |
| 480 | 6  | $\leq 36$ | 10        | $4 - 2 - 2$ | 8         | STCP       |
| 108 | 12 | 11        | 4         | $4 - 4 - 2$ | 16        | STCP       |
| 180 | 12 | 15        | 6         | $4 - 4 - 2$ | 32        | STCP       |
| 108 | 15 | 12        | 6         | $4 - 4 - 4$ | 96        | NLR        |
| 270 | 24 | 15        | 8         | $4 - 4 - 4$ | 96        | NLR        |
| 324 | 12 | $\leq 32$ | $\leq 12$ | $4 - 4 - 4$ | 128       | STCP       |
| 480 | 15 | $\leq 48$ | $\leq 14$ | $4 - 4 - 4$ | 128       | STCP       |

Table 1: **Examples of tricycle codes and their parameters for various block-lengths.**  $N$  is the block-length,  $K$  the number of encoded logical qubits,  $D_X$  and  $D_Z$  are the minimum weights of logical  $X$ -type and  $Z$ -type operators respectively. The distances of codes are found exactly by using either a SAT-solver to find the shortest error [35] or a mixed integer program [56] when feasible. Otherwise, distances are estimated – sometimes, with strong statistical guarantees, in which case we report the distance as exact<sup>1</sup>– using the low-weight biased Monte-Carlo sampling method discussed in Sec. C with 100 million shots. The construction of these specific codes is described in Sec. A and Table 2. The column named ‘Type’ denotes the class of tricycle codes, labeled by the weights  $w_a - w_b - w_c$  from Eq. (1). The depths of the code-space preserving  $CCZ$  circuits with non-trivial logical action are noted for each code. The final column named ‘CCZ Method’ indicates whether the code and non-trivial  $CCZ$  circuit was constructed using the symmetric triple cup-product method (STCP) – see Sec. D – or using the numerical Leibniz rule method (NLR) – see Sec. E.

The codes in Table 1 have imbalanced distances with  $d_Z \leq d_X$ . In Sec. A, we prove that this

is the case for all tricycle codes. The imbalanced distances of these codes is a potentially useful feature for experimental platforms which are typically dominated by noise biased towards one basis. A detailed exploration of how noise bias can boost the experimental performance of tricycle codes is left for future work.

Overall, the high distances of tricycle codes combined with the transversal  $CCZ$  circuits they host (which we discuss in Sec. 2.2) make them ideal candidates for magic state factories for various target logical error rates. While the rates of these codes are lower than their two-dimensional counterpart bicycle codes (see the codes in Ref. [15] for example), they are much higher than the rates of three dimensional binary homological product codes of comparable block-length, including the  $3D$  color code.

For the remainder of this work, we primarily focus on select  $4-2-2$  type tricycle codes that are practically relevant due to their lower check-weights and depth-8  $CCZ$  circuits. The  $4-4-4$  tricycle codes generally exhibit higher rates and distances, but this comes at the price of higher check-weights and a lower circuit-noise threshold of  $\approx 0.4\%$ , as well as significantly deeper non-trivial  $CCZ$  circuits on average – additional details on the  $4-4-4$  codes are presented in Sec. D, Sec. E, and Sec. G. We empirically found that the  $4-4-2$  type codes did not offer substantial improvements in rates, distances, or thresholds compared to the  $4-2-2$  and  $4-4-4$  type codes to the extent of our code search – two examples of  $4-4-2$  codes with non-trivial  $CCZ$  are nonetheless presented in Table 1.

## 2.2 Transversal $CCZ$ circuits

Tricycle codes support constant-depth, code-space preserving circuits built from physical  $CCZ$  gates acting across three code blocks. Each block contains  $3n_G$  qubits, partitioned into three equal-size sectors labeled  $I$ ,  $II$ , and  $III$ , each with  $n_G$  qubits. We label each qubit as  $q_i^j$ , where  $i \in \{1, 2, 3\}$  denotes the code block and  $j \in \{I, II, III\}$  the sector. Within each sector, qubits are indexed arbitrarily but consistently across blocks.

The structure of a transversal  $CCZ$  circuit is encoded in a binary function

$$f_{CCZ} : Q \times Q \times Q \rightarrow \mathbb{F}_2, \quad (2)$$

where  $Q$  is the set of all physical qubits. A value  $f_{CCZ}(q_1^{j_1}, q_2^{j_2}, q_3^{j_3}) = 1$  indicates a physical  $CCZ$  gate between the specified qubits, one from each code block. Otherwise, no gate is applied. This is illustrated in Fig. 1a).  $f_{CCZ}$  is then linearly extended to be defined as a trilinear function on all computational basis states. To define a valid, code-space preserving circuit,  $f_{CCZ}$  must satisfy two additional conditions:

1.  $f_{CCZ}$  must vanish when any one or more of the arguments correspond to a binary string  $b$  defining an  $X$ -type stabilizer – i.e  $X^b = X^{b_1} \otimes X^{b_2} \dots \otimes X^{b_{3n_G}}$  is an  $X$ -type stabilizer – and the other arguments similarly correspond to non-trivial  $X$ -type logical operators. This condition ensures  $f_{CCZ}$  descends to a well-defined function on the space of logical codewords.

---

<sup>1</sup>Specifically, the method we use returns a confidence interval on the distance conditioned on certain assumptions on the distribution of minimum-weight code-words. These assumptions can in turn be hypothesis tested using the empirical distribution of low-weight code-words found from sampling. We report the distance as exact if the confidence is  $> 99.9\%$  and if the  $p$ -value associated with the hypothesis test is  $> 10\%$ . Details of the method are in Sec. C.

2.  $f_{CCZ}$  must define a constant-depth circuit of physical  $CCZ$  gates.

We expand on both conditions and their homological interpretation in Sec. D. See Proposition 5 for how  $f_{CCZ}$  is defined for tricycle codes.

In recent work, Breuckmann et al. [18] introduced a formalism for constructing appropriate trilinear functions  $f_{CCZ}$  that satisfy conditions 1 and 2 based on algebraic operations on homological product codes known as *cup-products*. We develop a modification of this formalism and derive conditions that tricycle codes must satisfy to host such transversal  $CCZ$  circuits – see Sec. D for details on the conditions. For reasons elaborated in Sec. D, we refer to this framework and the associated conditions on the codes as the *symmetric triple cup-product* formalism for transversal  $CCZ$  gates, which applies to both three-dimensional homological product and balanced product [19] quantum LDPC codes – the latter of which tricycle codes are an example of. We then show that there are tricycle codes with favorable rates and distances which satisfy the conditions of the symmetric cup-product framework. We note that tricycle codes constructed using the original cup-product conditions of [18] have recently been studied in [51, 60] – the resulting  $2-2-2$  codes always have  $K=3$  while  $4-2-2$  and other higher weight codes have  $D=2$ . We find that the new symmetric cup-product conditions we derive in this work are necessary to circumvent these limitations, and are more generally applicable to other classes of higher-dimensional balanced product quantum codes.

In the following discussion, as a slight abuse of terminology, we refer to the ‘depth’ of the  $CCZ$  circuit and the maximum degree of any physical qubit – i.e the number of  $CCZ$  gates a qubit is involved in – interchangeably. Indeed the maximum degree is the relevant quantity that controls the spread of errors between physical qubits of the code-blocks. The minimal depth for scheduling the gates in the circuit is upper bounded by a small constant factor of the degree (see Sec. D) [75].

The  $4-2-2$  type tricycle codes which satisfy the conditions of the symmetric triple cup-product generically host depth 8  $CCZ$  circuits. Similarly, the  $4-4-2$  codes have depth 16 or depth 32 circuits, and the  $4-4-4$  codes have depth 12, 64, or 128 depth circuits. We explain the choices made in the code-construction that lead to these circuit depths in Sec. D.

In addition to the symmetric triple cup-product framework, we introduce a more general formalism for constructing valid  $f_{CCZ}$  functions on balanced-product quantum codes [19] that is amenable to an extensive numerical search method for codes with block-lengths of a few hundred qubits – see Sec. E for details. Using this numerical method, we are able to find additional  $4-4-4$  tricycle codes with better rates and distances along with shorter-depth physical  $CCZ$  circuits than those that result from the cup-product formalism – we give two examples of such  $4-4-4$  codes in Table 1, while we leave a more thorough exploration of this method for future work.

The physical qubits of each tricycle code-block can be naturally partitioned into three sectors in accordance with block structure of the parity-check matrices in Eq. (1). The code-space preserving  $CCZ$  circuits always act across sectors of the three code-blocks, as illustrated in Fig. 1. The logical action of a transversal  $CCZ$  circuit is derived by restricting  $f_{CCZ}$  to the logical subspace, using representatives of logical  $X$  operators for each block. This produces logical  $\overline{CCZ}$  gates between triples of logical qubits whenever  $f_{CCZ}(l_1^i, l_2^j, l_3^k) = 1$ , where  $\{l_1^i\}_{i=1\dots k}$  are a representative basis set of logical  $X$  operators of the first code-block, with  $\{l_2^j\}$  and  $\{l_3^k\}$  defined similarly. The resulting state, obtained by applying the logical circuit to  $\overline{|+, +, +\rangle}^{\otimes K}$ , defines a *hypergraph magic state* [114, 24], with vertices as logical qubits and hyperedges as  $\overline{CCZ}$  gates. Recent work studies such hypergraph magic states, arguing that they can contain a large amount of magic and can be classically hard to

simulate [24].

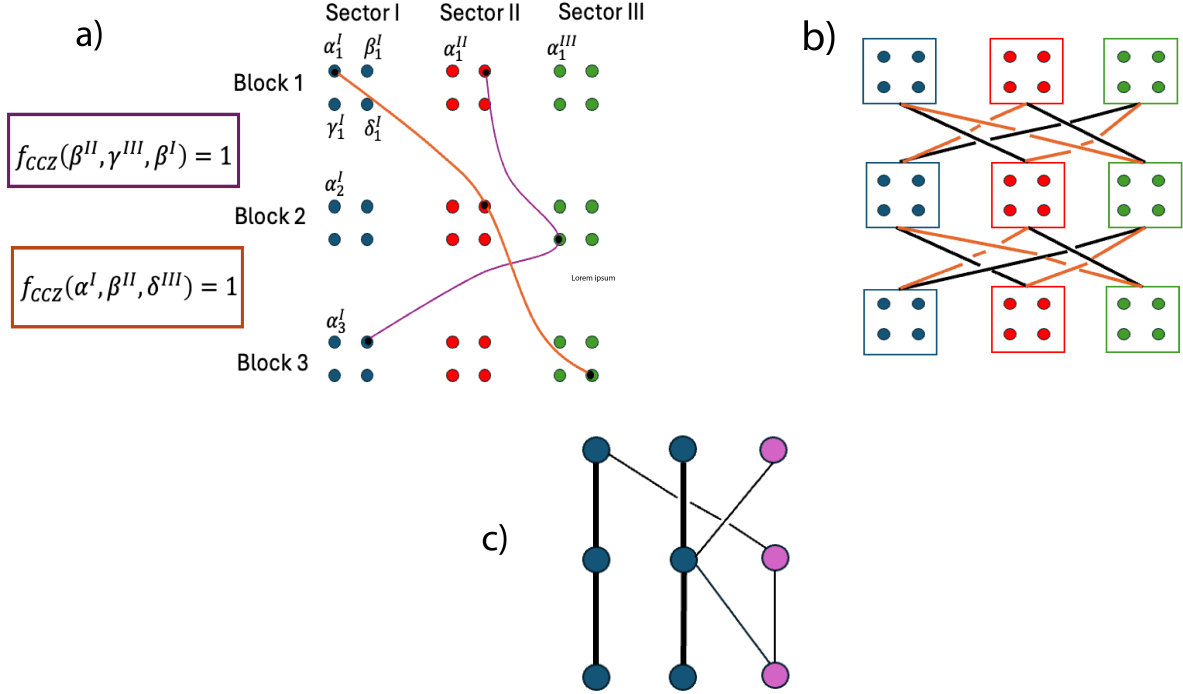


Figure 1: **Structure of transversal  $CCZ$  gates of tricycle codes.** **a)** Schematic of a transversal  $CCZ$  circuit on a 12-qubit code. Each code block is partitioned into three sectors of 4 qubits, labeled  $\alpha, \beta, \gamma, \delta$  with subscripts and superscripts indicating the code block and sector. Colored curves denote  $CCZ$  gates between triples of qubits where  $f_{CCZ}$  is non-zero. **b)** Structure of transversal  $CCZ$  circuits: all sectors participate via two disjoint sets of circuit layers denoted by orange and black edges that can individually be parallelized across qubits. Each qubit undergoes a maximum of  $l$  black and a maximum of  $m$  orange  $CCZ$  gates, leading to a maximum degree of  $l + m$ . **c)** Logical  $CCZ$  connectivity after basis optimization for a  $K = 3$  code. Circles denote logical qubits; rows correspond to separate code blocks. Thick black lines indicate usable  $\overline{CCZ}$  gates for magic state distillation ( $K_{CCZ} = 2$  shown), while thin lines involve gauge qubits (pink), initialized in  $|0\rangle$ . Blue circles represent logical qubits in disjoint triples connected only to other qubits in the triple or to gauge qubits.

While the produced hypergraph magic states are interesting and potentially computationally useful in their own right, we would also like to be able to extract smaller component magic gates from the logical circuit for quantum computation. One way to achieve this is to demand that the logical circuit consists of  $K$   $\overline{CCZ}$  gates across code-blocks, each acting on disjoint triples of logical qubits. However, a choice of logical bases that produces such a logical circuit may not always exist (and is unlikely to). We may instead try to maximize the number of extractable  $\overline{CCZ}$  gates on disjoint triples of logical qubits  $K_{CCZ} \leq K$ . Finding such a subset corresponds to computing the *subrank* of the  $f_{CCZ}$  function restricted to logical operators, a known problem in tensor rank theory [26, 52, 38, 63]. We use a mixed-integer programming method to find feasible solutions to this problem (Sec. F), and below we report the best values of  $K_{CCZ}$  found for each code in Table 1. For most codes, our solvers eventually time out within computational resource constraints without

proving optimality, suggesting that larger  $K_{CCZ}$  values than the ones we report are possible to find with tailored heuristic optimization strategies – a promising avenue for future work. The resulting logical circuits enable the extraction of  $K_{CCZ}$  usable  $\overline{CCZ}$  gates. Logical qubits not used in these gates are treated as *gauge qubits* and initialized in the  $|0\rangle$  state, which nullifies any  $\overline{CCZ}$  gates they participate in (Fig. 1d) – see Sec. D for details.

Using the methods described in Sec. F, we are able to find at-least  $K_{CCZ} \geq 2$  for all the codes in Table 1 using the associated  $CCZ$  circuits. We attempted to find larger values of  $K_{CCZ}$  for these codes, but the solver we used did not converge within a reasonable amount of time. We therefore emphasize that the  $K_{CCZ}$  values found here are lower bounds returned by exact methods that quickly become infeasible for larger  $K_{CCZ}$  – this is primarily due to the current lack of well-performing efficient heuristics for the binary tensor-subrank problem, and we anticipate that progress towards such heuristics will lead to a higher yield of individually distillable  $\overline{CCZ}$  type magic states for the codes in Table 1.

A subtle point that is understated in LDPC magic state distillation proposals is that extracting such  $K_{CCZ}$  disjoint  $\overline{CCZ}$  gates from the full logical circuit requires the ability to selectively initialize the gauge logical qubits in the  $|0\rangle$  state while other logical qubits are initialized in the  $|+\rangle$  state. For example, the proposals in Ref. [38, 63, 114, 115, 18] assume the ability to do this. However, selective state initialization is non-trivial for LDPC codes, and in general it is not clear that this is possible in constant depth. Extensions of the techniques introduced in Ref. [109] for initializing arbitrary Pauli product states in homological product codes be useful for selective initialization. Recent work has also introduced batched Pauli-product state-preparation techniques that can achieve selective state-initialization on arbitrary qLDPC codes in constant overhead, as long as many code-blocks are initialized at once in the same state [108] – we anticipate that these techniques will be very useful in a LDPC  $|\overline{CCZ}\rangle$  magic state factory setting [108]. More generally however, we would like to emphasize an alternative perspective of viewing the global hypergraph magic state of the  $K$  logical qubits as a high-magic resource state [24], that can be used to perform useful quantum computations in a target computational code with transversal Clifford operations using appropriate circuit compilation techniques. We leave a detailed exploration of this perspective along with concrete protocols for efficient gate-teleportation between tricycle codes and target Clifford gate-set computation codes for future work.

### 2.3 Single-shot distillation

We now describe how tricycle codes enable single-shot magic state generation. By initializing three code blocks fault-tolerantly in  $|\overline{+}\rangle^{\otimes K}$  using a constant number of error correction rounds and subsequently applying the transversal  $CCZ$  circuit described in the previous section, we obtain high-fidelity hypergraph magic states—all within constant circuit depth, as illustrated in Fig. 2.

Our approach to constant-depth preparation of logical  $|\overline{+}\rangle^{\otimes K}$  hinges on two essential properties: intrinsic fault tolerance in one stabilizer basis during state initialization, and the presence of metachecks that enable single-shot error correction in the complementary basis. Specifically, we initialize the physical data qubits in the product state  $|+\rangle^{\otimes n}$  and perform a single round of stabilizer measurements. As tricycle codes are CSS codes, the  $X$ -type checks are implemented as products of physical  $X$  operators acting on the input  $|+\rangle$  states. Thus, in the absence of errors, the measurement outcomes for the  $X$  checks are deterministic, which allows  $Z$  errors that occur during

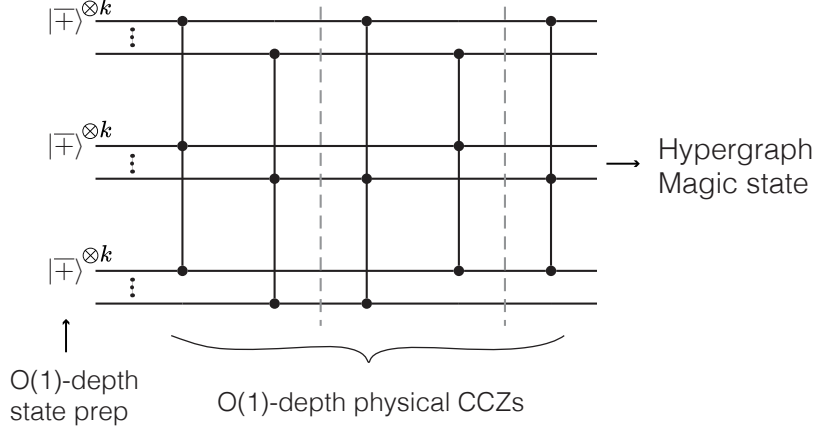


Figure 2: **Single-shot distillation with tricycle codes.** The logical  $|\bar{+}\rangle^{\otimes K}$  state of the tricycle code can be prepared fault-tolerantly in constant depth by harnessing the code’s intrinsic resilience in one basis—namely, by preparing the physical qubits such that the associated stabilizer checks are deterministic—together with single-shot error correction in the complementary, non-deterministic, basis. The logical non-Clifford operation is implemented via a constant-depth circuit composed of physical  $CCZ$  gates. The output is a logical hypergraph magic state which embeds  $K_{CCZ} \leq K$  disjoint logical  $|\overline{CCZ}\rangle$  resource states.

state preparation to be detected and corrected

However, initial  $Z$ -type stabilizer measurements produce non-deterministic  $\pm 1$  outcomes, which must be reliably fixed to  $+1$  on hardware before applying the  $CCZ$  gate [7]. In contrast to codes like the surface code, which lack single-shot state preparation, the  $Z$  parity check matrices for tricycle codes contain a large number of redundant checks. These subsequently form a robust set of metachecks, which resolve a classical code on the syndrome bits, allowing a decoder to identify and correct syndrome measurement errors. In the  $Z$  basis, these take the form

$$H_{meta} = [\mathbf{B} \quad \mathbf{A} \quad \mathbf{C}] \quad (3)$$

where the  $\mathbf{A}, \mathbf{B}, \mathbf{C}$  matrices are the same as in Eq. (1). The tolerance to measurement qubit errors is determined by the single-shot distance  $D_Z^{SS}$  [22], which is defined as the minimum weight of a faulty syndrome that passes all metachecks but is not a  $Z$  syndrome. A large  $D_Z^{SS}$  indicates considerable redundancy, facilitating the identification of sparse syndrome errors. For all codes examined (see Table 1), we found  $D_Z^{SS} = D_Z$ , or at least their upper bounds when exact distances were intractable to compute (see Sec. C). Concurrently,  $D_Z^{SS} = D_Z$  was proven for this  $H_{meta}$  analytically in [51]. It is also always possible to extend  $H_{meta}$  into a full-rank matrix [22] by forcing the decoder to repair the noisy syndrome into one that can be produced by a pattern of data qubit errors, resulting in  $D^{SS} = \infty$ .

Metachecks combine with *soundness* properties [22] to enable fault-tolerant inference of the  $Z$  stabilizer eigenvalues after only a constant number of error correction rounds under select error models. It is worth noting that this task of preparing the  $Z$  stabilizers to  $+1$  from a completely random initialization is a more challenging task than just single shot error correction [47, 9], where a decoder may assume that the  $Z$  stabilizers were fault-tolerantly fixed to  $+1$  prior to the observed

noise. For example, many hypergraph product codes enable single-shot error correction but not single-shot state preparation [47].

In Sec. B, we prove soundness properties for a certain sub-class of tricycle codes in the  $Z$  basis and review the ensuing guarantee on single-shot  $Z$  basis decodability, which applies to state preparation [109]. Moreover, we perform direct simulations of single-shot state preparation for three codes with good finite-size encoding rates and distances, showing that general tricycle codes appear to host single-shot state-preparation properties. Our results, shown in Fig. 3 demonstrate exponential suppression of the logical error with increased distance, consistent with fault-tolerant state preparation.

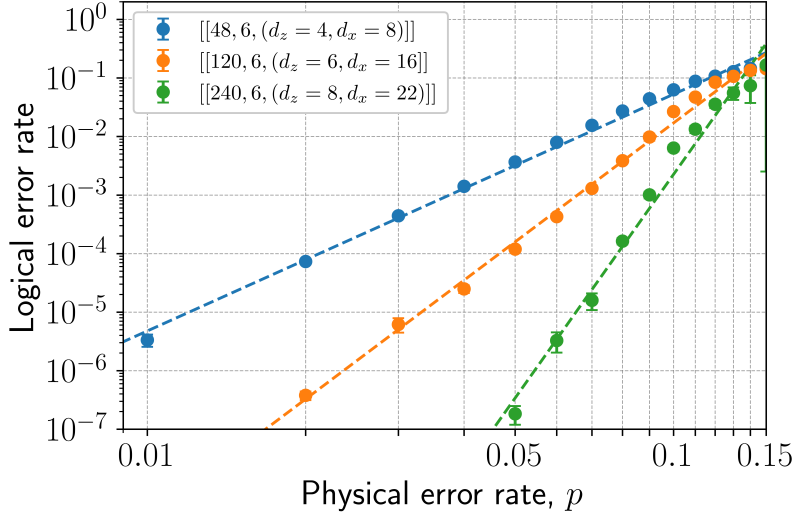


Figure 3: **Phenomenological noise simulation of single-shot state preparation in the  $Z$  basis for 4 – 2 – 2 tricycle codes.** Our method follows Ref. [47] and assumes that the initial  $Z$  syndrome is trivial. For each code, we simulate one round of syndrome measurement in which measurement errors occur with probability  $p$ , though we expect performance to improve with a larger decoding window (see Sec. 2.4). A most likely error (MLE) decoder applies a minimum weight correction to both the data and measurement qubits. Then, we simulate a noisy transversal  $Z$  basis measurement of the data qubits, decode the reconstructed syndrome with the MLE decoder, and apply the corresponding correction. A logical failure is said to occur if the residual  $X$  operator is a logical operator of the tricycle code, and the logical error rate is normalized per logical qubit. The observed phenomenological threshold is  $\gtrsim 13\%$ . Logical error rates are determined via Monte Carlo simulations; error bars indicate standard errors, computed as  $\sqrt{p_L(1 - p_L)/M}$ , where  $M$  is the number of samples.

Going back to the magic state factory, following initialization, the single-shot error correction capabilities of tricycle codes are necessary to prevent errors from spreading. Each  $CCZ$  gate will propagate a pre-existing  $X$  error into  $CZ$  errors on the other two qubits, which are collapsed into nondeterministic patterns of  $Z$  errors after syndrome measurement. To prevent such errors from building up, the  $Z$  stabilizers must continuously be repaired to  $+1$ . In codes without single-shot properties, this usually requires ‘just-in-time decoding’ [10], in which error clusters are allowed to expand modestly before being corrected, and accompanies a decrease in performance and threshold

of the code [86, 87]. The single shot  $Z$  basis of the tricycle code, in contrast, is able to immediately correct  $X$  errors (up to a small residual error), limiting repeated propagations of  $CZ$  errors. Although the  $X$  basis is not single-shot, importantly, the  $Z$  errors they detect commute with the  $CCZ$  gates, so they do not need be corrected in real time. This is opportune given that the  $CCZ$  circuit may disrupt the  $X$  stabilizer group until the circuit is completed. As the  $CCZ$  circuit is of constant depth, it suffices to measure the  $X$  stabilizers only after completion; however, incorporating intermediate correction strategies that leverage stabilizer masking [69] could be advantageous.

Together, these results demonstrate that our codes enable fault-tolerant preparation of logical  $|\overline{+}\rangle$  states in as little as a single round. As a result, magic state factories based on these codes are extremely compact: the initial state  $|\overline{+}\rangle_1 \otimes |\overline{+}\rangle_2 \otimes |\overline{+}\rangle_3$  with all  $Z$  stabilizers fault-tolerantly fixed to +1 can be prepared in constant depth, and the logical  $\overline{CCZ}$  operation can also be implemented in constant depth, as described in the previous section, yielding high-fidelity hypergraph magic states in constant depth.

## 2.4 Circuit-noise simulations

We now evaluate the performance of our codes under realistic circuit-level noise models. Our focus is on the  $4 - 2 - 2$  family of tricycle codes, characterized by modest check weights ( $w_x = 8$ ,  $w_z = 6$ ) and featuring the shortest-depth  $CCZ$  circuits from the symmetric triple cup-product construction (Sec. D). Circuit-level noise simulations of certain  $4 - 4 - 4$  codes are presented in Sec. G for comparison.

To benchmark performance, we simulate circuit-level noise affecting the entangling gates in the syndrome extraction circuits. We adopt a standard two-qubit depolarizing noise model: for a given two-qubit gate error rate  $p_{2q}$ , each entangling operation is followed, with equal probability, by one of the fifteen nontrivial two-qubit Pauli errors from  $IX, IY, \dots, ZZ$ , and otherwise experiences no error with probability  $1 - p_{2q}$ .

Tricycle codes naturally exhibit higher distance in the  $X$  basis compared to the  $Z$  basis. Here, we prioritize simulating the more challenging  $X$ -basis.  $Z$ -basis results, demonstrating robust single-shot behavior, are included in Appendix G. In the  $X$ -basis, we use the conventional  $d$ -round protocol: the syndrome extraction sequence is repeated for  $d$  cycles, and decoding is performed using the full record of accumulated syndrome information from all rounds. We utilize the depth-optimal syndrome extraction circuit structure outlined in Sec. 2.5.

For decoding, we map the syndrome extraction process to a spacetime code, where the checks correspond to parities of stabilizer measurement outcomes across consecutive time steps, and each code qubit is associated with a distinct fault location in the circuit [107, 14] (see Sec. G).

We perform the decoding for the  $d$ -rounds simulation of the  $4 - 2 - 2$  codes using a most-likely error (MLE) decoder. The resulting logical error rates, as shown in Fig. 4(b), are defined as the probability of a logical qubit failure, normalized both by the total number of error correction rounds and by the number of logical qubits in the block. We observe a gate-noise threshold of approximately 0.5%, and achieve low logical error rates at physically relevant gate errors. For example, we achieve a logical error rate of  $\sim 4 \times 10^{-5}$  at a physical error rate of 0.1% for the  $[[108, 6, (d_z = 6, d_x = 12)]]$  code.

For further performance gains, we consider post-selection, which is standard in the magic-

state distillation factory setting [16]. Specifically, we analyze the  $[[48, 6, (4, 8)]]$  code decoded using BP+LSD [46] in conjunction with the clustering post-selection metric introduced in Ref. [57]. After  $d_z = 4$  rounds of error correction, the logical error rates as a function of abort rate using the “ $\alpha = 2$  clusters llrs” post-selection metric of Ref. [57] are depicted in Fig. 4(a). Additionally, we assess full error detection whereby we discard any samples that have any stabilizers flipped. We observe that for the 48-qubit code, no stabilizer flip in roughly 32% of samples, corresponding to an acceptance fraction of 32% and yielding a very low logical error rate near  $6 \times 10^{-10}$ . For the 84-qubit code the acceptance fraction for full error detection over  $d$  rounds becomes  $\sim 8.5\%$  with a logical error rate  $< 7 \times 10^{-14}$ .

In summary, the  $4-2-2$  tricycle codes demonstrate strong performance in the  $d$ -round setting under circuit-level noise. It is worth emphasizing that in practical settings,  $d$  rounds may not always be necessary — e.g., when transversal teleportation can be employed — so these results may be viewed as lower bounds on achievable QEC performance. We also note that the numerical results presented here and in the appendix pertain to a single code undergoing rounds of error correction and therefore do not directly capture the fidelity of the logical  $CCZ$  gate. Nevertheless, since the  $CCZ$  circuit has short depth, we expect these results to serve as a reasonable proxy. A detailed simulation of the full  $CCZ$  circuit is left to future work. For additional simulation details and further numerical results, see Sec. G.

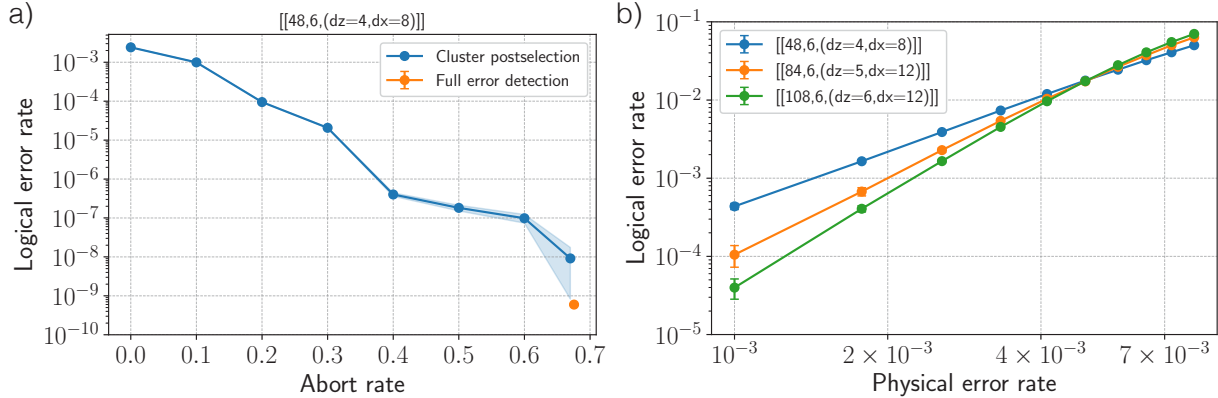


Figure 4: **Circuit-level noise simulation results for tricycle codes.** (a) Logical error rate for the  $[[48, 6, (d_z = 4, d_x = 8)]]$  code as a function of abort rate under cluster postselection (blue) and full error detection (orange). The cluster postselection data show the tradeoff between logical error rate and postselection (abort) probability using a BP+LSD decoder, while full error detection corresponds to strictly accepting only trials with no detected stabilizer flips. (b) Logical error rate versus two-qubit physical gate error rate  $p_{2q}$  for  $d$ -rounds, fault-tolerant error correction in the  $X$  basis, for tricycle codes of increasing size and distance using a most-likely error (MLE) decoder. In both panels, errors are sampled according to a standard two-qubit depolarizing circuit-level noise model and the logical error rate corresponds to the total logical error rate normalized by the number of QEC rounds and by the number of logical qubits. Logical error rates are determined via Monte Carlo simulations, with each data point corresponding to  $M$  samples; error bars indicate the standard error as  $\sqrt{p_L(1 - p_L)/M}$ .

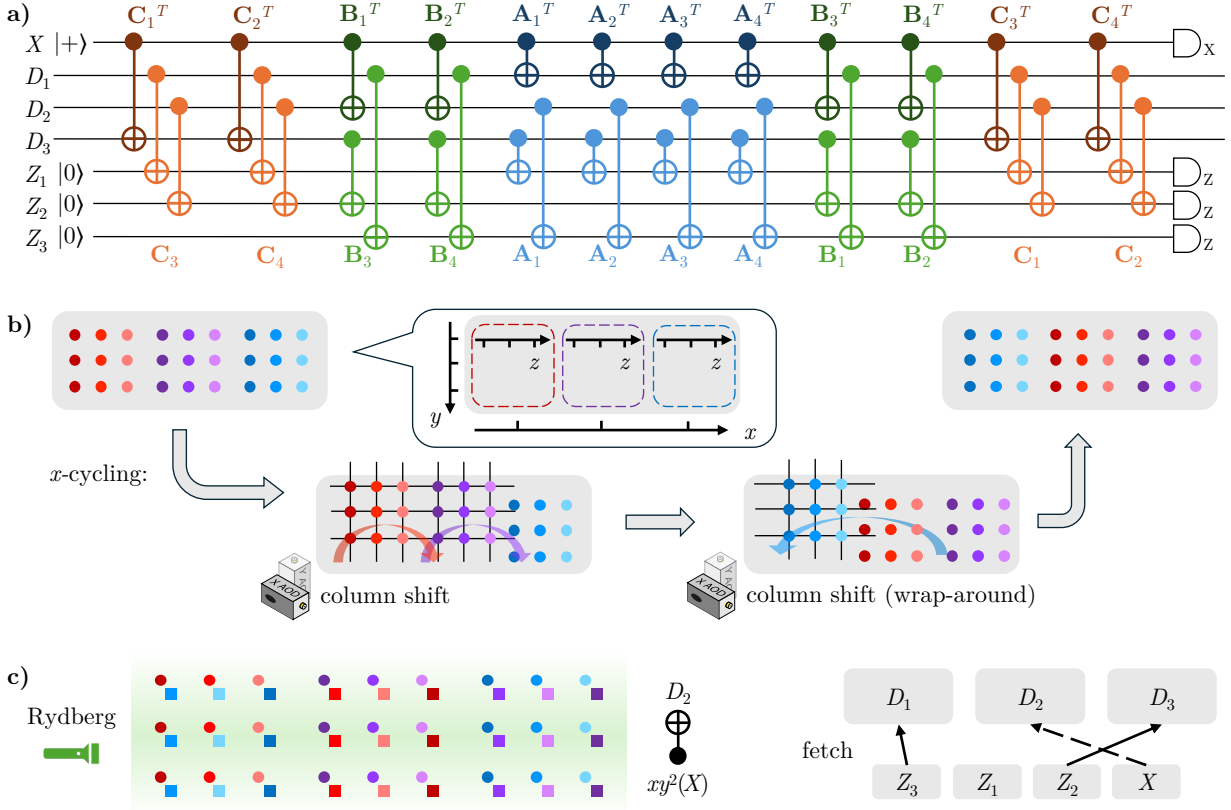


Figure 5: **Implementation of tricycle codes on neutral atom arrays.** (a) Syndrome extraction circuit. Each line denotes a sector. CNOTs are applied on pairs of qubits across two sectors, with the pairing determined by the permutation matrices. (b) Within each sector, physical qubits sharing the same  $x$  index form tiles arranged in a row; within each tile, qubits are ordered by their  $y$  and  $z$  indices. Two AOD movements can realize  $x$ -,  $y$ -, or  $z$ -cycling. (c) Parallel CNOTs can be performed by Rydberg interaction on two overlaying sectors. Data sectors reside in the entangling zone and have larger spacing to avoid intra-sector interactions. After permutation in the workspace below, check sectors are fetched to overlay with the corresponding data sectors and perform CNOTs. Then, check sectors are put back to their original positions, and perform permutation for the next layer.

## 2.5 Implementation of syndrome extraction circuits

We construct optimal-depth syndrome extraction circuits for the tricycle codes presented. Denote the data qubit sectors as  $D_1, D_2, D_3$ , the  $Z$ -check sectors as  $Z_1, Z_2, Z_3$ , and the  $X$ -check sector as  $X$ . Our construction for  $4-4-4$  codes is displayed in Fig. 5a, where each CNOT symbol represents a parallel group of physical CNOTs between a check sector and a data sector, associated with a specific permutation. For example, the first CNOT symbol corresponds to CNOTs from the  $i$ -th  $X$ -check to the  $j$ -th qubit in  $D_3$  whenever the  $(i, j)$  entry of  $\mathbf{C}_1^T$  is 1. Since  $\mathbf{C}_1^T$  is a permutation matrix, there is one and only one  $j$  for each  $i$ . The circuit achieves the optimal CNOT depth of 12. The construction consists of five stages represented by colors. For the  $4-4-2$  and  $4-2-2$  codes, the overall stage structure is preserved, although some stages contain fewer layers. The design allows

certain degrees of freedom: specifically, how to assign the matrices  $\mathbf{A}$ ,  $\mathbf{B}$ , and  $\mathbf{C}$  to the stages, as well as the order of layers within each stage. These degrees of freedom lead to 1728 possible circuits for the  $4 - 2 - 2$  codes. An exhaustive search across these 1728 circuits was performed, from which we select the circuit with the highest noise-suppression for each code in the circuit-noise simulation results of Sec. 2.4. For the general construction and a proof of its correctness, see Sec. H.

The syndrome extraction circuit can be implemented on either fixed architectures with all required couplings fabricated [104], or on reconfigurable architectures with physical qubit permutation [8, 34, 6, 84, 7, 25, 85, 70, 66, 42, 5, 81]. Here, we focus on reconfigurable neutral atom arrays.

With slight abuse of notation, we label the qubits in each sector by indices  $(x, y, z)$ , where  $1 \leq x \leq n_x$ ,  $1 \leq y \leq n_y$ ,  $1 \leq z \leq n_z$ , and  $n_G = n_x n_y n_z$ . As shown in the callout of Fig. 5b, we place qubits with the same  $x$  as tiles arranged in a row; within each tile, qubits are ordered by  $y$  and  $z$ . Qubits are held in place by traps generated by a spatial-light modulator (SLM); we assume a sufficient number of SLM traps, and thus do not depict them explicitly. Crossed 1D acousto-optic deflectors (AODs) create a 2D grid of mobile traps, enabling entire grids of qubits to be picked up and rearranged in the plane. Our qubit layout leads to efficient implementation of permutation matrices of interest. For example, Fig. 5b illustrates the process of  $x$ -cycling, corresponding to cyclically shifting the tiles: first, two tiles are transferred from the SLM to the AOD, shifted right in parallel, and deposited into ancillary SLM traps; the second step applies a ‘wrap-around’ in the opposite direction for the remaining tile. Rows and columns in the AOD grid can otherwise move freely, though their order must be preserved, so the two steps for  $x$ -cycling cannot be combined. The  $y$ -cycling and  $z$ -cycling similarly correspond to cyclically shifting rows and columns within each tile, respectively, as needed to implement polynomial terms like  $\mathbf{C}_1^T$ .

Entangling gates are performed by exciting adjacent pairs of qubits to Rydberg states. Parallel CNOTs between corresponding qubits in two sectors are realized by overlaying those sectors and applying a global Rydberg laser illumination, as shown in Fig. 5c (dots and squares denote data and check qubits, respectively. As the colors suggests, the check sector has been permuted with  $x$ -cycling by one unit and  $y$ -cycling by two units.) To prevent unintended interactions between qubits within the same sector, the pairs are separated by a sufficient distances determined by atom species and Rydberg state.

The overall layout (Fig. 5c, right) places data sectors in SLM traps within the entangling zone. For each CNOT layer, check sectors are permuted in a workspace below and fetched to the corresponding data sectors for parallel CNOTs. In the workspace, smaller grid spacing is used to reduce travel distances for permutation; during fetching, check sectors are expanded to align with data sector spacing. After entanglement, check sectors return to the workspace for the next permutation. Due to crossing movement paths, fetching and put-back of Z-check and X-check sectors must be performed separately in this example to preserve AOD column order.

Executing one syndrome cycle requires a constant number of AOD movements. Let  $t_{\text{wsp}}$  and  $t_{\text{ent}}$  denote the time for AOD to traverse a sector in the workspace and the entangling zone, respectively. In the implementation above, permutation, fetch, and put-back for all  $Z$  sectors can be done in parallel. Each CNOT layer then involves two fetch and two put-back operations, each bounded by  $3t_{\text{ent}}$  (say, from  $X$  to  $D_1$ ). Permutations for  $Z$  and  $X$  sectors may require  $x$ -,  $y$ -, and  $z$ -cycling, each up to  $2t_{\text{wsp}}$ . Thus, the per-layer duration is bounded by  $12(t_{\text{wsp}} + t_{\text{ent}})$ . Further details and possible optimization are discussed in Sec. I.

### 3 Discussion and Outlook

We have introduced a class of quantum LDPC codes—*tricycle codes*—that support constant-depth, single-shot preparation of logical magic states and show strong resilience to realistic circuit-level noise with a high threshold of  $> 0.5\%$ . We showed that certain families of tricycle codes allow compact, constant depth preparation of high-magic states known as hypergraph magic states. These hypermagic states in turn embed individually distillable  $\overline{CCZ}$ -type magic states, enabling universal quantum computation by using tricycle codes as magic state factories. The promising performance of these codes can be improved further along several directions. The asymmetry between the  $X$  and  $Z$  sectors of tricycle codes make them particularly advantageous for leveraging noise bias in entangling gate operations—a feature pervasive in leading experimental platforms [1, 7, 74, 90, 13]. Additionally, loss and leakage errors—prevalent in many hardware architectures—represent another avenue for enhancing performance; recent work has shown that appropriate handling of such errors can be leveraged to further improve logical error rates [3, 7]. While our present analysis focuses on unbiased depolarizing noise and neglects loss errors, future studies integrating tailored noise bias exploitation and explicit loss-/leakage-leveraging strategies with these codes are likely to yield further gains and represent a compelling direction for subsequent research. Moreover, improved decoding methods would be beneficial as we observe over an order-of-magnitude gap in logical error rates between standard BP+OSD and BP+LSD decoders and an exact but exponentially expensive most-likely-error (MLE) decoder. Encouragingly, recent work has proposed new decoders that enhance performance while maintaining practical inference times [72, 12, 67, 68].

Beyond their standalone utility as distillation factories, integrating tricycle codes into broader quantum architectures presents several important challenges. A key open problem is the seamless injection or teleportation of distilled magic states into computational code blocks, such as those based on high-performance bicycle codes. While lattice surgery offers a robust, code-agnostic method for logical state transfer—and has seen significant theoretical and experimental development [27, 107, 105, 49, 96, 30, 29, 4, 112, 28]—a particularly promising alternative is transversal teleportation based on natural isomorphisms between tricycle and bicycle codes. Similar to teleportation protocols between 3D and 2D color codes [95, 31], such schemes using transversal one-way CNOT gates between 3D and 2D product codes [45] may enable direct, fault-tolerant transfer of magic states. In particular, we believe this should be possible between bivariate-bicycle codes and bivariate-tricycle codes or between trivariate-bicycle codes and trivariate-tricycle codes, as they share the same product structure. Using extensions of the analytical tools in Ref. [33], we believe it should be possible to show that the logical operators of pairs of some such 2D and 3D group-algebra codes split into isomorphic sectors as in hypergraph product codes [102] – a natural setting for efficient gate teleportation with transversal one-way CNOTs. A version of this idea has recently been explored in [60]. If realized, these protocols could further reduce space-time overhead and enable highly modular architectures with efficiently integrated magic state distillation. We will explore a specific magic-state factory architecture based on high-rate tricycle codes with transversal teleportation onto a target high-rate bicycle code in a follow-up work.

Another important direction is improving the yield of disjoint  $\overline{CCZ}$  gates that can be extracted from the hypergraph magic states produced by these circuits (Sec. F). This corresponds to the problem of finding the subrank of a binary tensor [26, 52, 38, 63], which is computationally hard. While we used mixed-integer programming, new heuristic strategies could uncover larger values of  $K_{CCZ}$  for the codes we consider. In parallel, a deeper understanding of the structure of these high-

magic hypergraph states—produced by the transversal  $CCZ$  circuits described in Sec. 2.2—may allow one to compile them directly into useful quantum circuits, offering an alternative to extracting only disjoint  $\overline{CCZ}$  gates.

In summary, tricycle codes represent a significant step toward scalable, low-overhead, fault-tolerant magic state distillation with quantum LDPC codes. Ongoing work on new code constructions, transversal gate optimization, and efficient code-switching protocols will continue to strengthen the outlook for practical, high-performance quantum computing.

## 4 Acknowledgments

We thank Qian Xu, Nazli Ugur Koyluoglu, Rohith Sajith, Nishad Maskara, Shayan Majidy, Hengyun Zhou, Harry Putterman, Brenden Roberts, Jin Ming Koh, Andrei Diaconu, and Jason Cong for valuable discussions. We particularly thank Qian Xu for detailed feedback on our results and manuscript. We acknowledge financial support from IARPA and the Army Research Office, under the Entangled Logical Qubits program (Cooperative Agreement Number W911NF-23-2-0219), the DARPA MeasQuIT program (grant number HR0011-24-9-0359), the Center for Ultracold Atoms (a NSF Physics Frontiers Center, PHY-1734011), the National Science Foundation (grant numbers PHY-2012023 and CCF-2313084), the Wellcome Leap Quantum for Bio program, Harvard Quantum Initiative Postdoctoral Fellowship (DBT) and QuEra Computing. After the completion of this project, we became aware of related work studying tricycle codes, albeit with less focus on transversal magic gates [51]. Another related work that appeared around the same time also introduced a general theory for multi-cyclic Abelian group-algebra codes [61].

## Appendix

### A Code construction and properties

We describe the construction of the tricycle code parity check matrices in more detail, along with general properties of the codes and their connection to balanced products of classical group algebra codes.

Let  $G$  be a finite Abelian group, expressed as  $G = \mathbb{Z}_{m_1} \times \mathbb{Z}_{m_2} \times \cdots \times \mathbb{Z}_{m_k}$  for some finite  $k$ . Let  $n_G = |G| = \prod_{i=1}^k m_i$ . The codes we construct are defined via matrices over the ring  $R := \mathbb{F}_2[G]$ , the group algebra of  $G$  over  $\mathbb{F}_2$ . Elements of  $\mathbb{F}_2[G]$  are formal sums

$$\sum_{g \in G} a_g g, \quad \text{where } a_g \in \mathbb{F}_2,$$

with componentwise addition and multiplication induced by extending the group operation bilinearly. Since  $G$  is Abelian,  $\mathbb{F}_2[G]$  is a commutative, associative algebra over  $\mathbb{F}_2$  with identity  $1_G \in G$ .

For  $a \in \mathbb{F}_2[G]$ , we define its *weight* as  $|a| = |\{g \in G \mid a_g = 1\}|$ . Each  $a \in \mathbb{F}_2[G]$  determines a binary matrix  $\mathbf{A} \in \mathbb{F}_2^{n_G \times n_G}$  via

$$\mathbf{A}_{\alpha, \beta} = \mathbb{B}_G(a) \equiv \sum_{g \in G} a_g \delta_{\alpha, g\beta}, \quad (4)$$

where  $\alpha, \beta \in G$  and  $\delta$  is the indicator for  $\alpha = g\beta$ . The map  $\mathbb{B}_G : \mathbb{F}_2[G] \rightarrow \mathbb{F}_2^{n_G \times n_G}$  gives the regular representation of  $a$ , following [78].

Alternatively, elements of  $\mathbb{F}_2[G]$  can be identified with polynomials via the isomorphism

$$\mathbb{F}_2[G] \cong \mathbb{F}_2[x_1, \dots, x_k] / \langle x_1^{m_1} - 1, \dots, x_k^{m_k} - 1 \rangle, \quad (5)$$

where  $x_i$  corresponds to a generator of  $\mathbb{Z}_{m_i}$ . Under this identification, the weight of  $a$  is the number of monomials with nonzero coefficient in the corresponding polynomial.

To compute  $\mathbf{A}$ , let  $S_l$  denote the  $l \times l$  binary right cyclic shift matrix, and define  $\hat{S}_{m_i} = I_{m_1} \otimes \dots \otimes S_{m_i} \otimes \dots \otimes I_{m_k}$ . These matrices commute for different  $i$ . Given  $p_a(x_1, \dots, x_k)$ , the polynomial corresponding to  $a$ , the binary matrix representation is

$$\mathbf{A} = p_a(\hat{S}_{m_1}, \hat{S}_{m_2}, \dots, \hat{S}_{m_k}). \quad (6)$$

If  $|a| = w$ , then each row and column of  $\mathbf{A}$  has Hamming weight  $w$ . We can define tricycle codes formally as follows.

**Definition 1** (Tricycle codes). *A three-block Abelian group algebra code (tricycle code) is a CSS code defined by elements  $a, b, c \in \mathbb{F}_2[G]$  with weights  $w_a, w_b, w_c$ , and binary matrix representations  $\mathbf{A}, \mathbf{B}, \mathbf{C} \in \mathbb{F}_2^{n_G \times n_G}$ . The  $X$  and  $Z$  parity check matrices are defined as in Eq. (1), with component matrices formed from binarizing group algebra sums: for example, if  $a = a_1 + a_2 + a_3 + a_4$ , where  $a_i \in G$ , then  $\mathbf{A} = \sum_i \mathbb{B}(a_i)$ .*

We focus on codes with  $G$  equal to a product of three cyclic groups ( $k = 3$ ) as we found these codes to generally have better parameters than those with  $k = 1, 2$  or higher. By analogy to the nomenclature for bicycle codes [53, 15, 76], these may be referred to as trivariate-tricycle codes. The  $k = 1, 2$  cases may be referred to as generalized tricycle and bivariate-tricycle respectively. We denote the generators by monomials  $x_1 \equiv x$ ,  $x_2 \equiv y$ , and  $x_3 \equiv z$ . The polynomials and groups used in the examples of Table 1 are listed in Table 2.

| $[[\mathbf{N}, \mathbf{K}, (\mathbf{D}_X, \mathbf{D}_Z)]]$ | $(\mathbf{l}, \mathbf{m}, \mathbf{n})$ | $\mathbf{a}$                              | $\mathbf{b}$                          | $\mathbf{c}$     |
|------------------------------------------------------------|----------------------------------------|-------------------------------------------|---------------------------------------|------------------|
| $[[48, 6, (8, 4)]]$                                        | $(2, 2, 4)$                            | $y + z + xz + xyz^2$                      | $yz^2 + yz^3$                         | $y + xyz$        |
| $[[84, 6, (12, 5)]]$                                       | $(2, 2, 7)$                            | $y + z + xz + xyz^2$                      | $z^3 + xz^4$                          | $y + yz^4$       |
| $[[108, 6, (12, 6)]]$                                      | $(3, 3, 4)$                            | $x + z^2 + yz + x^2yz^3$                  | $y^2z + x^2yz^3$                      | $x^2 + x^2yz^2$  |
| $[[240, 6, (\leq 22, 8)]]$                                 | $(4, 4, 5)$                            | $xy^2z^3 + xy^3z^4 + x^2y^2z + x^2y^3z^2$ | $y^3 + x^2yz^2$                       | $xz^4 + x^3y^3z$ |
| $[[480, 6, (\leq 36, 10)]]$                                | $(4, 5, 8)$                            | $x^2z^7 + x^3y^2 + y^4z^4 + xy^3z^3$      | $z^6 + x^2y^2z^2$                     | $x^2 + x^3y^4z$  |
| $[[108, 12, (6, 4)]]$                                      | $(3, 3, 4)$                            | $z + xz^3 + xyz^2 + x^2y$                 | $y^2 + y^2z^3 + xy^2z + xy^2z^2$      | $z + xyz^3$      |
| $[[180, 12, (15, 6)]]$                                     | $(3, 4, 5)$                            | $yz^3 + y^3 + x^2yz^3 + x^2y^3z$          | $xyz^4 + xy^2z^2 + x^2yz + x^2y^2z^4$ | $z^4 + x^2z$     |

|                                   |             |                                            |                                         |                                       |
|-----------------------------------|-------------|--------------------------------------------|-----------------------------------------|---------------------------------------|
| $[[108, 15, (12, 6)]]$            | $(3, 3, 4)$ | $y + y^2z + xyz^3 + x^2y^2z^2$             | $z^2 + xy + xy^2z + x^2z^3$             | $yz^3 + y^2z + x^2 + x^2y^2z^2$       |
| $[[270, 24, (15, 8)]]$            | $(3, 5, 6)$ | $z^4 + y^3 + xy^2 + x^2yz^4$               | $y^3 + y^3z + xy^4 + x^2y^2z$           | $yz^4 + y^2z + xy^2z + xy^3z^4$       |
| $[[324, 12, (\leq 32, \leq 12)]]$ | $(3, 4, 9)$ | $y^2z + xyz + x^2z^6 + x^2y^3z^5$          | $z^4 + z^5 + xyz^7 + x^2y^3z^2$         | $yz^7 + xz^7 + xy^3 + x^2y^2$         |
| $[[480, 15, (\leq 48, \leq 14)]]$ | $(4, 5, 8)$ | $x^2z^5 + x^2yz^4 + x^3y^3z^4 + x^3y^4z^3$ | $x^2z^3 + x^2y^4z^6 + x^3z^5 + x^3yz^2$ | $yz^5 + xz^4 + x^2y^4z^4 + x^3y^2z^5$ |

Table 2: Polynomials used to construct trivariate-tricycle codes in Table 1 corresponding to elements of the group algebra of  $G = \mathbb{Z}_l \times \mathbb{Z}_m \times \mathbb{Z}_n$ . If there are fewer than three cyclic group factors, the corresponding element in the tuple of group orders  $(l, m, n)$  is left empty.

Next, we describe how tricycle codes can be viewed as three-dimensional hypergraph products (HGPs) [102] over non-binary rings, and derive a conditional distance lower bound. We will first need the following definitions.

**Definition 2** (Support and intersection subgroups). *Let  $G$  be a finite Abelian group, and  $a = \sum_{g \in G} a_g g \in \mathbb{F}_2[G]$  with  $a_g \in \{0, 1\}$ . Define the support subgroup  $G_a := \langle \{g : a_g \neq 0\} \rangle$ . For elements  $a_1, \dots, a_k \in \mathbb{F}_2[G]$ , the intersection subgroup is  $N = \bigcap_{i=1}^k G_{a_i}$ . Since  $G$  is Abelian,  $N$  is normal in each  $G_{a_i}$ .*

**Definition 3** (Connected codes). *Let  $a, b, c \in \mathbb{F}_2[G]$  with support subgroups  $G_a, G_b, G_c$ . The code defined by  $a, b, c$  is connected if  $G_a G_b G_c = G$ . Otherwise, the code decomposes into subcodes supported on cosets of the subgroup  $G_a G_b G_c \subset G$ , which has order  $|G_a||G_b||G_c|/|N|^2$ , where  $N$  is the intersection subgroup.*

This notion of connectivity extends the Abelian case of [62], which showed that connected two-block group algebra codes can be written as hypergraph products (HGP) over non-binary rings. The same argument applies here, yielding the following:

**Proposition 1** (Non-binary 3D HGP). *Let  $a, b, c \in \mathbb{F}_2[G]$  define a connected tricycle code, and let  $N = G_a \cap G_b \cap G_c$  with  $|N| = c$ . Let  $l_a = [G_a : N]$  be the index of  $G_a$  in  $N$ , and define  $l_b, l_c$  analogously. Define  $R = \mathbb{F}_2[N]$ , and consider  $R$ -valued matrices:*

$$A = A_1 \otimes I_{l_b} \otimes I_{l_c}, \quad (7)$$

$$B = I_{l_a} \otimes B_1 \otimes I_{l_c}, \quad (8)$$

$$C = I_{l_a} \otimes I_{l_b} \otimes C_1, \quad (9)$$

for appropriately chosen  $A_1 \in R^{l_a \times l_a}$ ,  $B_1 \in R^{l_b \times l_b}$ , and  $C_1 \in R^{l_c \times l_c}$ . The binary matrices  $\mathbf{A}, \mathbf{B}, \mathbf{C}$  are then obtained by applying  $\mathbb{B}_N$  element-wise:  $\mathbf{A}_{i,j} = \mathbb{B}_N(A_{i,j}) \in \mathbb{F}_2^{c \times c}$ , etc.

The parity check matrices correspond to a three-fold  $R$ -linear homological product code [111]:

$$H_X^T = \begin{bmatrix} A_1 \otimes I_{l_b} \otimes I_{l_c} \\ I_{l_a} \otimes B_1 \otimes I_{l_c} \\ I_{l_a} \otimes I_{l_b} \otimes C_1 \end{bmatrix}, \quad (10)$$

$$H_Z = \begin{bmatrix} I_{l_a} \otimes I_{l_b} \otimes C_1 & \mathbf{0} & A_1 \otimes I_{l_b} \otimes I_{l_c} \\ \mathbf{0} & I_{l_a} \otimes I_{l_b} \otimes C_1 & I_{l_a} \otimes B_1 \otimes I_{l_c} \\ I_{l_a} \otimes B_1 \otimes I_{l_c} & A_1 \otimes I_{l_b} \otimes I_{l_c} & \mathbf{0} \end{bmatrix}. \quad (11)$$

The binary versions are recovered via the binarization map  $\mathbb{B}_N$  defined in Eq. (4).

*Proof.* This follows identically from the proofs of Statements 7,8,9 in the appendix of [62]. In particular, by choosing a representative set of coset elements  $p_i \in G_a/N$ ,  $q_j \in G_b/N$ ,  $r_k \in G_c/N$  where  $i = 1 \cdots l_a$ ,  $j = 1 \cdots l_b$ ,  $k = 1 \cdots l_c$ , the  $R$ -valued matrices  $A_1$ ,  $B_1$ ,  $C_1$  are given by

$$(A_1)_{i_1, i_2} = \sum_{n \in N} a_{(p_{i_1} n p_{i_2}^{-1})} n \quad (12)$$

$$(B_1)_{j_1, j_2} = \sum_{n \in N} b_{(q_{j_1} n q_{j_2}^{-1})} n \quad (13)$$

$$(C_1)_{k_1, k_2} = \sum_{n \in N} c_{(r_{k_1} n r_{k_2}^{-1})} n \quad (14)$$

where  $a_g$ ,  $b_g$ ,  $c_g$  are the coefficients in the group algebra elements  $a = \sum_{g \in G} a_g g$  and similarly for  $b, c$ .  $\square$

This perspective of viewing connected tricycle codes as three-dimensional homological products over a ring is useful for proving the following lower bound on the code distances.

**Theorem A.1.** *Consider a connected three-block Abelian group algebra code constructed from group algebra elements  $a, b, c \in \mathbb{F}_2[G]$ , with corresponding binary matrices  $\mathbf{A}, \mathbf{B}, \mathbf{C} \in \mathbb{F}_2^{n \times n}$  where  $n = |G|$ . Further suppose that the classical binary linear codes with parity check matrices  $\mathbf{A}^T, \mathbf{B}^T, \mathbf{C}^T$  have minimum distances  $d_A^T, d_B^T, d_C^T$ . We use the convention that  $d = \infty$  if the corresponding matrix is full rank. Let  $N = G_a \cap G_b \cap G_c$  be the intersection subgroup as defined in Definition 2. The minimum distance of the quantum code is bounded below as*

$$d \geq \frac{1}{|N|} \min\{d_A^T, d_B^T, d_C^T\} \quad (15)$$

*Proof.* The proof that  $d_Z$  satisfies this inequality follows from the proof of Statement 10 in [62]. In particular, Lemma 3 of [62] also holds for three-dimensional homological products by decomposing them into a direct sum of three-dimensional homological products over cyclic group algebras (analogous to Appendix B. in [77]). The matrices  $\mathbf{A}, \mathbf{B}, \mathbf{C}$  are constructed identically, and following the proof of Lemma 3. in [62], each component code over a cyclic group algebra can be shown to have all-unit Smith Normal Forms under the assumptions of the lemma. The rest of the proof is identical, applied instead to the matrix  $H_X$  defined in Eq. (1).

Then, in Theorem A.2, we prove that  $d_Z \leq d_X$ , making Eq. (15) a lower bound for the overall code distance.  $\square$

For disconnected codes formed from  $m$  connected subcodes, the bound applies to each component and can be summed. In practice, this lower bound is often loose: the codes we construct typically exceed it by a large margin. It may be possible to improve this bound using techniques from [111] adapted to non-binary rings.

While the above bound may not be tight, it does imply the existence of a threshold for tricycle codes. This is because classical quasi-Abelian group-algebra codes with parity check matrices of the form  $\mathbf{A}^T, \mathbf{B}^T, \mathbf{C}^T$  are known to include codes that have distances that grow linearly in the block-length [62]. Combined with the above bound, this implies the existence of tricycle codes with distances that are extensive in the block-length with at-least  $\sqrt[3]{N}$  scaling, implying the existence of a threshold.

In the following theorem, we show that tricycle codes satisfy  $d_Z \leq d_X$ . This would make Theorem A.1 a lower bound on both the  $X$  and  $Z$  distances.

**Theorem A.2.** *All connected three-block Abelian group algebra codes defined by Eq. (1) satisfy  $d_Z \leq d_X$ .*

*Proof.* We will assume that none of  $A_1, B_1, C_1$  are full rank, else the tricycle code is trivial with  $k = 0$  and  $d = \infty$  by convention. Noting  $R \cong \bigoplus_{g \in N} \mathbb{F}_2$ , the map  $\mathbb{b}_N : R \rightarrow \mathbb{F}_2^{|N|}$  defined by  $\mathbb{b}_N : a = \sum_g a_g g \mapsto (a_{g_1} \cdots a_{g_{|N|}})$  is a vector space isomorphism. With action defined element-wise, it can be extended to  $\mathbb{b}_N^m : R^m \rightarrow \mathbb{F}_2^m$ ; we omit the superscript hereafter. We define the norm  $|\cdot|_R$  on  $\mathbb{R}^m$  via  $|v|_R = \sum_{i=1}^m |\{g \in G : (v_i)_g = 1\}|$  for any  $v \in R^m$ . Importantly,  $|v|_R = |\mathbb{b}_N(v)|$  in general. Let  $\mathbb{B}_N : R \rightarrow \mathbb{F}_2^{|N| \times |N|}$  be the regular representation of  $N$  introduced in Eq. (4), extended element-wise and linearly to  $R^{m \times m}$ . For any  $A \in R^{m \times m}$  and  $v \in R^m$ ,  $\mathbb{b}_N(Av) = \mathbb{B}_N(A)\mathbb{b}_N(v)$ .

Let  $x = (x_1 \ x_2 \ x_3) \in R^{l_a l_b l_c}$  denote a minimum weight logical  $X$  operator. Define

$$(\delta^{-1})^T := (A_1^T \otimes_R I_{l_b} \quad I_{l_a} \otimes_R B_1^T) \quad \delta^0 := (I_{l_a} \otimes_R B_1 \quad A_1 \otimes_R I_{l_b}). \quad (16)$$

The CSS code with  $R$ -valued  $X/Z$  parity check matrices  $(\delta^{-1})^T/\delta^0$  is a bicycle code.  $H_Z x = 0$  implies  $(x_1 \ x_2) \in \ker(\delta^0 \otimes_R I_{l_c}) = (\ker \delta^0) \otimes_R R^{l_c} \cong (\ker \delta^0)^{\oplus l_c}$ . Thus, one can express  $(x_1 \ x_2) = \bigoplus_{i=1}^{l_c} \tilde{x}_i$ ,  $\tilde{x}_i \in \ker \delta^0$ . Suppose that for at least one  $i^* \in [1, l_c]$ ,  $\tilde{x}_{i^*} \notin \text{im } \delta^{-1}$ . Then,  $\tilde{x}_{i^*}$  is a nontrivial logical  $X$  operator of this bicycle code; there exists some  $\tilde{z} \in \ker(\delta^{-1})^T \setminus \text{im } \delta^{0T}$  with  $|\mathbb{b}_N(\tilde{z})| \leq |\mathbb{b}_N(\tilde{x})|$  (Lemma 1 of Ref. [15]). We will lift  $\tilde{z}$  to an equal-weight logical  $Z$  operator of the tricycle code.

Expressing  $H_X/H_Z$  in terms of  $(\delta^{-1})^T/\delta^0$ ,

$$H_X = ((\delta^{-1})^T \otimes_R I_{l_c} \quad I_{l_a l_b} \otimes_R C_1^T) \quad H_Z = \begin{pmatrix} \delta^0 \otimes_R I_{l_c} & 0 \\ I_{2l_a l_b} \otimes_R C_1 & \delta^{-1} \otimes_R I_{l_c} \end{pmatrix}. \quad (17)$$

From the Künneth theorem for  $R$ -module cohomology (see section I of the supplementary material and [65]),

$$\frac{\ker H_X}{\text{im } H_Z^T} \cong \left( \frac{R^{l_a}}{\text{im } (\delta^{-1})^T} \otimes_R \ker C_1^T \right) \oplus \left( \frac{\ker(\delta^{-1})^T}{\text{im } \delta^{0T}} \otimes_R \frac{R^{l_c}}{\text{im } C_1^T} \right). \quad (18)$$

Since  $R^{l_c} = \text{im } C_1^T + R^{l_c}/\text{im } C_1^T$  is spanned by a basis of unit vectors, and  $\text{im } C_1^T \subsetneq R^{l_c}$  by assumption, there exists some unit vector  $\hat{e}$ ,  $|\hat{e}|_R = 1$ , such that  $z = (0 \ \tilde{z} \otimes_R \hat{e})$  is a nontrivial logical  $Z$  operator. For any  $g \in G$ ,  $r \in R$ ,  $|r|_R = |gr|_R$ . Therefore, assuming  $\tilde{x}_{i^*} \in \ker \delta^0 \setminus \text{im } \delta^{-1}$  exists,  $d_Z \leq |\mathbb{b}_N(z)| = |z|_R = |\tilde{z}|_R \leq |\tilde{x}|_R \leq |x|_R = \mathbb{b}_N(x) = d_X$ .

Repeating the analysis for the other two ways to express the tricycle code as a  $R$ -homological product of a bicycle code and a classical group-algebra code, the above analysis is only incomplete in the case

$$x \in \text{im} \begin{pmatrix} T_A & 0 \\ 0 & I_{a^l b^l c^l} \\ T_C & 0 \end{pmatrix} \cap \text{im} \begin{pmatrix} T_A & 0 \\ T_B & 0 \\ 0 & I_{a^l b^l c^l} \end{pmatrix} \cap \text{im} \begin{pmatrix} 0 & I_{a^l b^l c^l} \\ T_B & 0 \\ T_C & 0 \end{pmatrix}, \quad (19)$$

where  $T_A = A_1 \otimes_R I_{l_b} \otimes_R I_{l_c}$ ,  $T_B = I_{l_a} \otimes_R B_1 \otimes_R I_{l_c}$ , and  $T_C = I_{l_a} \otimes_R I_{l_b} \otimes_R C_1$ . Such an  $x$  exists iff there exist  $u_1, u_2, u_3 \in R^{2^{l_a l_b l_c}}$  such that  $x_1 = T_A u_1 = T_A u_2$ ,  $x_2 = T_B u_2 = T_B u_3$ , and  $x_3 = T_C u_1 = T_C u_3$ , yet

$$x \notin \text{im} H_X^T = \text{im} \begin{pmatrix} T_A \\ T_B \\ T_C \end{pmatrix}. \quad (20)$$

Let  $v_A = u_1 + u_2$ ,  $v_B = u_2 + u_3$ , so  $u_1 + u_3 = v_A + v_B$ . It follows that  $v_A \in \ker T_A$ ,  $v_B \in \ker T_B$ ,  $v_A + v_B \in \ker T_C$ . Expressing

$$x = \begin{pmatrix} T_A u_1 \\ T_B u_2 \\ T_C u_3 \end{pmatrix} = \begin{pmatrix} T_A u_1 \\ T_B(u_1 + v_A) \\ T_C(u_1 + v_A + v_B) \end{pmatrix} = \begin{pmatrix} T_A u_1 \\ T_B(u_1 + v_A) \\ T_C u_1 \end{pmatrix} = H_X^T u_1 + \begin{pmatrix} 0 \\ T_B v_A \\ 0 \end{pmatrix}, \quad (21)$$

a solution for  $x$  exists iff  $(0 \ T_B v_A \ 0)$  is a nontrivial logical  $X$  operator, which requires  $v_A \notin \ker T_B$ . Note  $v_A \in \ker T_A \cap (\ker T_B + \ker T_C)$ . Now we use the following lemma.

**Lemma A.3.** *In this setting, the intersection can be distributed over subspace addition.*

$$\ker T_A \cap (\ker T_B + \ker T_C) = (\ker T_A \cap \ker T_B) + (\ker T_A \cap \ker T_C) \quad (22)$$

*Proof.* The reverse inclusion is generally true. For the forward direction, define the canonical projection  $\Pi : R^{l_a l_b l_c} \rightarrow \ker T_A$ . Any  $v \in \ker T_A \cap (\ker T_B + \ker T_C)$  satisfies  $\Pi v = v$  and can be written as  $v = t_B + t_C$ , where  $t_B \in \ker T_B$  and  $t_C \in \ker T_C$ . Since  $\Pi$  factorizes as  $\Pi = \pi \otimes_R I_{l_b} \otimes_R I_{l_c}$ , where  $\pi : R^{l_a} \rightarrow \ker A_1$ ,  $\Pi t_B \in \ker T_B$  and  $\Pi t_C \in \ker T_C$ . By definition,  $\Pi t_B, \Pi t_C \in \ker T_A$ .  $\square$

Thus,  $v_A \in (\ker T_A \cap \ker T_B) + (\ker T_A \cap \ker T_C) \subseteq \ker T_B + (\ker T_A \cap \ker T_C)$ . If we write  $v_A \in t_B + t_{AC}$ , where  $t_B \in \ker T_B$ ,  $t_{AC} \in \ker T_A \cap \ker T_C$ , then, observe that  $(0 \ T_B v_A \ 0) = H_X^T t_{AC}$  and is thus logically trivial. This contradicts Eq. (20), so Eq. (19) cannot hold.  $\square$

**Corollary A.3.1.** *Theorem A.2 extends to disconnected codes.*

*Proof.* Any disconnected tricycle code is a direct sum of connected tricycle codes. Theorem A.2 can then be applied to the component with the lowest weight logical  $X$  operator.  $\square$

Next, we explain how tricycle codes arise as the *balanced product* of classical group-algebra codes. This construction builds on the homological formulation of CSS codes and the balanced product construction from Refs. [19, 78, 77, 18], which we assume readers are familiar with in the following discussion – a review is provided in the supplementary material.

A classical group-algebra code is defined by an element  $a \in \mathbb{F}_2[G]$ , with parity check matrix  $H = \mathbb{B}_G(a)$ , where  $\mathbb{B}_G$  is the binarization map from Eq. (4). This corresponds to the 2-term cochain complex

$$\mathbb{F}_2[G] \xrightarrow{a} \mathbb{F}_2[G] \quad (23)$$

where the coboundary map is given by multiplication by  $a$  in the algebra. The bits of the classical code are associated with the 1-cochains (right) and the checks are associated with the 0-cochains (left).

Given two such complexes defined by elements  $a, b \in \mathbb{F}_2[G]$ , their balanced product results in the 3-term complex

$$R \xrightarrow{\begin{pmatrix} a \\ b \end{pmatrix}} R^2 \xrightarrow{\begin{pmatrix} b & a \end{pmatrix}} R \quad (24)$$

where  $R = \mathbb{F}_2[G]$  for notational convenience. Binarizing this complex yields the Abelian bicycle code, with  $H_X^T = [\mathbb{B}(a)^T \ \mathbb{B}(b)^T]$  and  $H_Z = [\mathbb{B}(b) \ \mathbb{B}(a)]$  [62, 53], where the 0-cochains are  $X$ -checks, 1-cochains are qubits, and 2-cochains are  $Z$ -checks (with 0,1,2 annotating the spaces in the complex from left to right). This complex arises naturally from the balanced product using the isomorphism  $R \otimes_R R \cong R$ ; see [33, 19, 78] and the supplementary material.

Tricycle codes generalize this by iterating the balanced product once again with a third classical group-algebra code defined by  $c \in \mathbb{F}_2[G]$ , giving the 4-term cochain complex:

$$\mathbb{F}_2[G] \xrightarrow{\begin{pmatrix} a \\ b \\ c \end{pmatrix}} \mathbb{F}_2[G]^3 \xrightarrow{\begin{pmatrix} c & 0 & a \\ 0 & c & b \\ b & a & 0 \end{pmatrix}} \mathbb{F}_2[G]^3 \xrightarrow{\begin{pmatrix} b & a & c \end{pmatrix}} \mathbb{F}_2[G] \quad (25)$$

This again follows from expanding the terms of the balanced product complex and applying  $R \otimes_R R \cong R$ .

The quantum CSS code associated with the tricycle construction is defined by the first three terms of Eq. (25), with consecutive spaces representing  $X$ -checks, qubits, and  $Z$ -checks respectively. Binarizing the coboundary maps yields the parity check matrices in Eq. (1), and the final map defines the metachecks in Eq. (3). This complex appears in the appendix of [33], and tricycle codes of this form are also briefly noted in [18].

This homological viewpoint is especially useful for constructing transversal logical gates. In the next section, we apply the formalism of [18] to equip the complex Eq. (25) with an algebraic structure that enables transversal  $CCZ$  circuits.

## B Single-shot guarantees

We prove that tricycle codes possess soundness properties in one basis, which in turn guarantee some fault-tolerant single-shot state preparation and error correction capabilities under adversarial noise. Our proofs heavily reference Campbell's results on the soundness of 4D homological product codes in Ref. [22].

**Definition 4.** (Soundness): Let  $t \in \mathbb{Z}_+$  and  $f : \mathbb{Z} \rightarrow \mathbb{R}$  be a function with  $f(0) = 0$ . For some ring  $R$  and norm  $|\cdot|$  on  $R^n$ , we say that a parity check matrix  $H \in R^{m \times n}$  is a  $(t, f)$ -sound map if, for any  $v \in R^n$  such that  $|Hv| < t$ , there exists some  $v' \in R^n$  such that  $Hv = Hv'$  and  $|v'| \leq f(|Hv|)$ .

To begin, we consider homological product codes formed from classical seed codes over  $\mathbb{F}_{2^k}$  for some  $k > 1$ . Let  $|\cdot|$  denote the Hamming weight: for  $v \in \mathbb{F}_{2^k}^n$ ,  $|v| = |\{i : v_i \neq 0\}|$ . A similar Hamming weight applies over  $R$ . For a classical code over  $\mathbb{F}_{2^k}$ -vector spaces, we define its distance in the usual way with respect to the Hamming weight.

**Lemma B.1.** (Extension to the first soundness lemma, [22]): *Let  $\mathcal{C} = C^0 \xrightarrow{\delta^0} C^1$  be a cochain complex over  $\mathbb{F}_{2^k}$ -vector spaces for some  $k > 1$  corresponding to a code over  $l$  dits. Denote the Hamming distance of  $\mathcal{C}$  by  $d_H(\delta^0) = d_0$  and the transpose code by  $d_0^T$ . Let  $\tilde{\mathcal{C}} = \mathcal{C} \otimes_{\mathbb{F}_{2^k}} \mathcal{C}$ , yielding the 3-term complex  $\tilde{\mathcal{C}} = \tilde{C}_{-1} \xrightarrow{\tilde{\delta}^{-1}} \tilde{C}^0 \xrightarrow{\tilde{\delta}^0} \tilde{C}^1$ , where*

$$\tilde{\delta}^{0T} = \begin{pmatrix} \delta^{0T} \otimes_{\mathbb{F}_{2^k}} I \\ I \otimes_{\mathbb{F}_{2^k}} \delta^0 \end{pmatrix}, \quad \tilde{\delta}^{-1} = \begin{pmatrix} I \otimes_{\mathbb{F}_{2^k}} \delta^{0T} \\ \delta^0 \otimes_{\mathbb{F}_{2^k}} I \end{pmatrix}. \quad (26)$$

Then,  $\tilde{\delta}^{0T}, \tilde{\delta}^{-1}$  are  $(t, f)$ -sound, where  $t = \min\{d_0, d_0^T\}$  and  $f(x) = x^2/4$ .

We provide a self-contained proof of this lemma, which mostly mirrors Appendix D in Ref. [22], for illustration. In later adaptations, we only discuss necessary modifications to results from Ref. [22].

*Proof.* We focus on  $\tilde{\delta}^{0T}$ ; the proof sketch for  $\tilde{\delta}^{-1}$  is similar. Let  $s \in \text{im } \tilde{\delta}^{0T}$  with  $|s| < t$ .  $v \in \mathbb{F}_{2^k}^n$ ,  $s = \tilde{\delta}^{0T} v$ . Letting  $s = (s_L \ s_R)$  and reshaping  $\mathbb{F}_{2^k}^l \otimes_{\mathbb{F}_{2^k}} \mathbb{F}_{2^k}^l \rightarrow \mathbb{F}_{2^k}^{l \times l}$  yields the equations  $S_L = \delta^{0T} V$  and  $S_R = V \delta^0$ . For any  $A \in \mathbb{F}_{2^k}^{l \times l}$ , let  $\text{rsp}(A) \subseteq \{1, \dots, n\}$  denote the set of indices corresponding to nonzero rows of  $A$ , and likewise with  $\text{csp}(A)$  for columns.

**Lemma B.2.** *There exists some  $w \in \mathbb{F}_{2^k}^l$  such that  $s = \tilde{\delta}^{0T} w$ ,  $|\text{csp}(S_L)| = |\text{csp}(W)|$ , and  $|\text{rsp}(S_R)| = |\text{csp}(W)|$ .*

*Proof.* At least one  $w$  satisfying the syndrome equation must exist. For any such  $w$ ,  $\text{csp}(S_L) \subseteq \text{csp}(W)$  and  $\text{rsp}(S_R) \subseteq \text{rsp}(W)$ ,  $|\text{csp}(S_L)| \leq |\text{csp}(W)|$  and  $|\text{rsp}(S_R)| \leq |\text{rsp}(W)|$ . The  $\geq$  direction is more difficult to prove. For concreteness, assume  $|\text{csp}(S_L)| < |\text{csp}(W)|$  and  $|\text{rsp}(S_R)| < |\text{rsp}(W)|$ . Then, there exist column and row vectors  $a$  and  $b^T$  in  $V$  such that  $a \in \ker \delta^{0T}$  and  $b \in \ker \delta^0$ .  $b^T$  is the  $i$ th row of  $V$ , which we denote by  $V_{i,\cdot} = b^T$ , and  $a$  is the  $j$ th column:  $V_{\cdot,j} = a$ . If  $V_{ij} \neq 0$ , then  $V' = V + V_{ij}^{-1} a b^T$  has strictly smaller support size:  $V'_{\cdot,j} = V_{\cdot,j} + V_{ij}^{-1} (a b^T)_{\cdot,j} = 0$  since  $(a b^T)_{\cdot,j} = b_j^T a = V_{ij} a$ . Identically,  $V'_{i,\cdot} = 0$ , and no new rows or columns can pick up support. Thus, we update  $V \leftarrow V'$  and repeat until the algorithm stalls, which is when there are no rows and columns  $b^T$  and  $a$  such that  $V_{ij} \neq 0$ . Up to row and column permutations, this leaves  $V$  in the form

$$V = \begin{pmatrix} D & B & 0 \\ A & C & 0 \\ 0 & 0 & 0 \end{pmatrix} \quad (27)$$

where columns in the left column block contain  $a \in \ker \delta^{0T}$ , and those in the middle column block satisfy  $\delta^{0T} a \neq 0$ . Similarly, rows  $b^T$  in the top row block satisfy  $b \in \ker \delta^0$ , whereas for those in the middle block,  $\delta^0 b \neq 0$ . Since no row-column pairs intersect,  $D = 0$ . If  $A$  is nonzero, any nonzero column  $a \in \ker \delta^{0T}$  satisfies  $|a| \geq d_0^T$ ,  $|\text{rsp}(S_R)| \geq |c| \geq d_0^T$ , which is a contradiction. Thus,  $A = 0$ , and similarly,  $B = 0$ .  $\square$

Since  $|s| = |S_L| + |S_R| \geq |\text{csp}(S_L)| + |\text{rsp}(S_R)|$ , the general identity  $(a+b)^2/4 \geq ab$  for integers  $a, b$  yields

$$|s|^2/4 \geq |\text{csp}(S_L)| |\text{rsp}(S_R)| = |\text{csp}(V)| |\text{rsp}(V)| \geq |V|. \quad (28)$$

□

Similarly, we have the following result.

**Lemma B.3.** (Extension to the second soundness lemma, [22]): *Let  $\tilde{C}^{-1} \xrightarrow{\tilde{\delta}^{-1}} \tilde{C}^0 \xrightarrow{\tilde{\delta}^0} \tilde{C}^1$  be a cochain complex over  $\mathbb{F}_{2^{k_i}}$ -vector spaces where  $\tilde{\delta}^{0T}, \tilde{\delta}^{-1}$  are  $(t, f)$ -sound with  $f(x) = x^2/4$ . Then,  $\check{C} = \tilde{C} \otimes_{\mathbb{F}_{2^k}} \tilde{C}$  is the 4-complex*

$$\check{C} = \check{C}^{-2} \xrightarrow{\check{\delta}^{-2}} \check{C}^{-1} \xrightarrow{\check{\delta}^{-1}} \check{C}^0 \xrightarrow{\check{\delta}^0} \check{C}^1 \xrightarrow{\check{\delta}^1} \check{C}^2 \quad (29)$$

where

$$\check{\delta}^{-2} = \begin{pmatrix} I \otimes_{\mathbb{F}_{2^k}} \tilde{\delta}_0^T \\ \tilde{\delta}_{-1} \otimes_{\mathbb{F}_{2^k}} I \end{pmatrix}, \quad \check{\delta}^{-1} = \begin{pmatrix} I \otimes_{\mathbb{F}_{2^k}} \tilde{\delta}_{-1}^T & 0 \\ \tilde{\delta}_{-1} \otimes_{\mathbb{F}_{2^k}} I & I \otimes_{\mathbb{F}_{2^k}} \tilde{\delta}_0^T \\ 0 & \tilde{\delta}_0 \otimes_{\mathbb{F}_{2^k}} I \end{pmatrix}, \quad (30)$$

$$\check{\delta}^{0T} = \begin{pmatrix} \tilde{\delta}_{-1}^T \otimes_{\mathbb{F}_{2^k}} I & 0 \\ I \otimes_{\mathbb{F}_{2^k}} \tilde{\delta}_{-1} & \tilde{\delta}_0^T \otimes_{\mathbb{F}_{2^k}} I \\ 0 & I \otimes_{\mathbb{F}_{2^k}} \tilde{\delta}_0 \end{pmatrix}, \quad \check{\delta}^{1T} = \begin{pmatrix} \tilde{\delta}_0^T \otimes_{\mathbb{F}_{2^k}} I \\ I \otimes_{\mathbb{F}_{2^k}} \tilde{\delta}_{-1} \end{pmatrix}. \quad (31)$$

The maps  $\check{\delta}^0$  and  $(\check{\delta}^{-1})^T$  are  $(t, g)$ -sound, with  $g(x) = x^3/4$ .

*Proof.* For  $k = 1$ , the proof of this result is Lemma 6 in [22], which in turn relies on “partial soundness” and “inheritance of soundness” results, which are Lemma 7 and Claim 3 there, respectively. For  $k > 2$ , the proofs of all three statements can be adapted with a small modification, just like Theorem B.2: during the reduction steps  $V \leftarrow V + ab^T$  is modified into  $V \leftarrow V + V_{ij}^{-1} ab^T$ . The  $\mathbb{F}_2$  case is simply the special case where  $V_{ij}^{-1} = 1$  given  $a$  and  $b^T$  intersect. Still working within the Hamming norm, the remainder of the analysis extends. □

We can invoke these to demonstrate soundness properties for some connected 4D balanced product codes.

**Theorem B.4.** *Let  $\check{C}$  denote a 4-term cochain complex Eq. (29) over  $R$ -modules, built from the classical seed code  $\mathcal{C} = C^0 \xrightarrow{\delta^0} C^1$  by repeated applications of the  $R$ -valued homological product, where  $R = \mathbb{F}_2[N]$ . Assume  $|N|$  is odd. Let  $d_0$  denote the distance of the binarized classical code  $\mathbb{b}_N(C^0) \xrightarrow{\mathbb{B}_N(\delta^0)} \mathbb{b}_N(C^1)$ . Then,  $\check{C}$  has  $(t, f)$ -soundness, where  $t = d_0/|N|$ , and  $g(x) = \frac{|N|+1}{8}x^3$*

*Proof.* Since  $|N|$  is odd,  $|N| \nmid \text{char}(\mathbb{F}_2) = 2$ , so by Maschke’s theorem,  $R$  is semisimple. From the Wedderburn-Artin theorem, it can be shown that  $R \cong \mathbb{F}_{2^{k_1}} \times \cdots \times \mathbb{F}_{2^{k_m}}$  [77]. The explicit form of the isomorphism is  $\psi : r \mapsto (r \cdot e_1, r \cdot e_2, \dots, r \cdot e_m)$  where  $\{e_1, \dots, e_m\}$  is the set of primitive orthogonal idempotents, which also satisfy a completeness relation  $\sum_i e_i = 1_R = 1_G$ . The isomorphism can be extended to matrix rings  $R^{m \times n} \cong \bigoplus_i \mathbb{F}_{2^{k_i}}^{m \times n}$ . Let  $A \in R^{n \times m}$  and  $B \in R^{m \times p}$ . Then,  $AB = \bigoplus_{i,j} A e_i B e_j = \bigoplus_i A_{(k_i)} B_{(k_i)}$ , and, identically,  $A \otimes_R B = \bigoplus_i [A_{(k_i)} \otimes_{\mathbb{F}_{2^{k_i}}} B_{(k_i)}]$ .

Therefore,  $\mathcal{C} \simeq \bigoplus_i \mathcal{C}_{(k_i)}$ , where  $\mathcal{C}_{k_i}$  is a  $\mathbb{F}_{2^{k_i}}$ -code, with the parity check matrix decomposed as  $\delta^0 \simeq \bigoplus_i \delta_{(k_i)}^0$ .

Note that for any  $v \in R^m$ ,  $|v|_R \leq |N||v|$ . Thus,  $d_H(\delta_{(k_i)}^0) \geq d_H(\delta^0) \geq d_R(\delta^0)/|N| = d_H(\mathbb{B}_N(\delta_0))/|N|$  provides a lower bound on the Hamming distance of each  $\delta_{(k_i)}^0$ . Since the tensor product over  $R$  splits component-wise, one can see the  $R$ -homological product codes also decompose component-wise, e.g.,  $\check{\mathcal{C}} = \bigoplus_i [\check{\mathcal{C}}_{(k_i)} \otimes_{\mathbb{F}_{2^{k_i}}} \check{\mathcal{C}}_{(k_i)}]$ . We apply lemmas B.1 and B.3 to each classical code component  $\mathcal{C}_{(k_i)}$ , obtaining that every  $\delta_{(k_i)}^0$  (and  $(\check{\delta}_{(k_i)}^{-1})^T$ ) are at least  $(d_H(\delta_{(k_i)}^0), g)$ -sound. Let  $s \in \text{im } \tilde{\delta}_0$  decompose as  $s = \bigoplus_i s_{(k_i)}$ . Since  $\psi$  is any isomorphism, so  $\ker \psi^{-1} = \{0\}$ ,  $|s_{(k_i)}|_H \leq |s|_H$ . If  $|s|_R \leq d_H(\mathbb{B}_N(\delta_0))/|N|$ , then it follows by soundness that for each  $i$ , there exists  $v_{(k_i)} \in \mathbb{F}_{2^{k_i}}^{6l^4}$  such that  $s_{(k_i)} = \check{\delta}_{(k_i)}^0 v_{(k_i)}$ , and  $|v_{(k_i)}| \leq |s_{(k_i)}|^3/4$ . This gives  $|v|_R \leq m|v_{(k_i)}|_H \leq m|s|_H^3/4 \leq m|s|_R^3/4 = m|\mathbb{B}_N(s)|^3/4$ . Finally, since  $m$  is the number of equivalence classes of integer pairs  $(x, y)$  such that  $y = 2^k x \bmod |N|$  for some integer  $k$  (formally known as the number of 2-cyclotomic cosets modulo  $|N|$ ), it has the upper bound  $m \leq (|N| + 1)/2$ .  $\square$

**Corollary B.4.1.** (Soundness of tricycle codes): *Consider any connected tricycle code with parameters  $[[n, k, (d_Z, d_X)]]$ , which can be expressed as the balanced product of binary classical codes  $\mathcal{C}_1, \mathcal{C}_2$ , and  $\mathcal{C}_3$ . There exists a decoder such that for any pattern of measurement errors  $u \in \mathbb{F}_2^n$  satisfying  $|u| \leq \frac{1}{2|N|} \min_i d_i$  and data errors  $v \in \mathbb{F}_2^n$  satisfying  $(|N| + 1)|u| + |v| < d_Z/2$ , the decoder takes in the syndrome  $H_Z v + u$  and outputs  $\hat{v}$  such that, modulo stabilizers,  $|(\hat{v} + v) \bmod 2|$  leaves a residual error of size at most  $(|N| + 1)|u|$ .*

*Proof.* First, note that it is easy to extend Theorem B.4 to when the classical seed codes are different, though this generally results in the bases having different soundness parameters. Then, recognizing that a tricycle code is just a 4D  $R$ -homological product code in which one of the seed codes is a degree-0 chain complex [47], one can check that the unaffected basis remains sound with the claimed  $t = \min_i d_i$  and the claimed  $g$ . As mentioned in Sec. 2.3, one can modify a minimum-weight decoder to decode the metacheck matrix  $H_{\text{meta}}$  in such a way that  $d_{ss} = \infty$ . Then, combining Theorem 1 of [22], Theorem A.2, and Theorem B.4, we obtain the corollary.  $\square$

As with Corollary A.3.1, this result can be readily extended to disconnected codes. Even though Corollary B.4.1 only applies to codes where  $|N|$  is odd, we expect, based on our numerics in Sec. 2.3 and Sec. 2.4 that codes with even  $|N|$  are sound. In fact, many of our codes in Table 2 are connected with even-order intersection subgroups.

## C Determining code distances

We use three distinct methods to either find the exact distances or determine high statistical confidence estimates of the distances of tricycle codes.

The first method is a Mixed-Integer-Program (MIP). This involves first using Gaussian row-reduction over  $\mathbb{F}_2$  to find a basis of logical  $X$  and  $Z$  operators  $L_X$  and  $L_Z$  independently for a given code. One then formulates the low-weight logical operator search as a mixed-integer-program by iterating over the basis of logical operators. For example, to find  $d_X$ , for the  $i^{\text{th}}$  logical operator  $l_i \in L_Z$ , we minimize the hamming weight of candidate bitstring solutions  $b$  under the constraints

$H_Z b = 0 \bmod(2)$  and  $b^T l_i = 1 \bmod(2)$  where the latter constraint enforces that  $b$  corresponds to an  $X$ -type logical operator that anticommutes with the  $i^{th}$  logical  $Z$  operator. The minimum weight logical operator is then found after iterating over all  $l_i$  if the solver converges within time constraints. We used the Gurobi [41] package in Python for this task.

We also sometimes use the shortest-error search method provided in the STIM package [35] combined with the PySat package [50] to directly formulate the problem of finding code distances as a Boolean constraint satisfaction (SAT) problem. For certain codes, we found that this method converged while the previous MIP method times out within time constraints, or vice-versa.

The MIP and SAT methods are not readily useful for the larger codes we consider in this work, and sometimes do not converge in one basis even for smaller codes, as in general, finding the exact distance of a linear code is known to be  $NP$ -Hard. In these cases, we estimate the minimum distance of a CSS quantum code using a Monte Carlo algorithm that performs structured sampling over the space of low-weight logical operators. The algorithm is based on the information-set algorithm of [58] for classical linear codes, with the additional step that one has to remove low-weight codewords belonging to the row-space of the opposite parity check matrix. We note that the algorithm we use is essentially identical to that of the QDistRnd package [79] that is provided in the GAP computational algebra programming language. We developed an independent implementation in the python programming language. We briefly summarize the algorithm as well as the empirical probability of success estimates derived for the QDistRnd algorithm.

We discuss how to estimate  $d_Z$ , with the analysis for  $d_X$  being identical with the parity check matrices swapped. The objective is to find a nonzero vector  $c \in \ker(H_X)$  such that  $c \notin \text{rowspan}(H_Z)$ , with minimal Hamming weight. The algorithm proceeds by first computing a basis  $G$  for the nullspace  $\ker(H_X)$  using row-reduction over  $\mathbb{F}_2$ . In each Monte Carlo trial, a random column permutation  $\pi$  is applied to the columns of  $G$ , followed by row reduction (forward elimination) to produce a low-weight generating set for  $\ker(H_X)$ . The inverse permutation  $\pi^{-1}$  is then applied to return to the original coordinate system. Rows of the resulting matrix that are linearly independent of the rows of  $H_Z$  are retained as candidate logical operators – this is once again determined using efficient row-reduction over  $\mathbb{F}_2$ . The smallest observed weight among all valid logical operators encountered over many trials is reported as the estimated distance.

To assess the reliability of the distance estimate, we track the number of rediscoveries of each distinct minimum-weight logical operator over  $N$  trials. Let  $m$  denote the number of distinct minimum-weight vectors found, and let  $n_i$  be the number of times the  $i$ -th one is observed. We define the average rediscovery count as

$$\langle n \rangle = \frac{1}{m} \sum_{i=1}^m n_i.$$

Assuming that each trial samples a minimum-weight codeword independently and with equal probability  $\lambda > 0$ , the number of times each such codeword is found follows a Poisson distribution with mean  $N\lambda$ . Under this model, the probability that *no* minimum-weight codeword is discovered in  $N$  trials is at most

$$P_{\text{fail}} \leq \exp(-\langle n \rangle),$$

where  $\langle n \rangle \approx N\lambda$ . This gives a conservative upper bound on the probability of missing the true minimum distance. In particular, when  $\langle n \rangle \gg 1$ , the failure probability becomes exponentially small.

To further evaluate sampling quality, we test whether the distribution of rediscovery counts  $\{n_i\}$  is consistent with a uniform multinomial distribution. If each minimum-weight codeword is equally likely to be sampled, the frequencies  $\{n_i\}$  should follow a multinomial distribution with equal probabilities. Deviations from this uniformity are detected using a Pearson chi-squared test. A small associated p-value indicates uneven sampling, suggesting either bias in the heuristic or incomplete exploration of the codeword space. In the main text, we only report the distance found from this method as exact if the returned confidence  $1 - P_{fail} > 0.99$  and if the p-value associated with the chi-squared test is  $> 0.1$ . To ensure reliability of the hypothesis test, we also only infer a p-value from the chi-squared test if at least 2 distinct minimum weight code-words are found, and if at least 5 occurrences of each codeword are found. In all other cases, we report the estimated distances as an upper bound, although the noise simulations in Sec. 2.4 show that noise suppression is compatible with code distances that are equal or near these estimates.

## D Symmetric triple cup-product and transversal CCZ action

Now we discuss how we derive the transversal  $CCZ$  circuits discussed in Sec. 2.2 from the cup-product logical gate formalism of [18].

First, we introduce the necessary formalism at a high level. A cup-product is an algebraic structure that can be defined on some cochain complexes, such as the one in Eq. (25). Let us label the cochain spaces of the complex in Eq. (25) as 0, 1, 2, 3-cochains from left to right, indicated with the notation  $C^0$ ,  $C^1$ ,  $C^2$ , and  $C^3$  respectively. Furthermore, we are interested in the case where the  $C^i$  are finite-dimensional vector spaces over  $\mathbb{F}_2$  with bases  $X^i$ . We define the weight  $|c|$  of a cochain  $c \in C^i$  as the number of non-zero terms in its basis expansion over  $X^i$ . For example, for the tricycle code complex in Eq. (25), the bases  $X^i$  are defined by  $G$ , and the cochain weight coincides with the weight of group-algebra elements. We will always associate qubits with the 1-cochains of a complex, with 0- and 2- cochains corresponding to spaces of  $X$  and  $Z$  checks respectively. Thus, we will label the elements of the basis of 1-cochains  $X^1$  by qubit labels  $q_1 \cdots q_N$  for the  $N$  qubits of a CSS quantum code.

A cup-product is then a bilinear map

$$\cup : C^i \times C^j \rightarrow C^{i+j} \quad (32)$$

with the convention that  $C^{i+j} = 0$  if  $i+j > 3$  in the case of the tricycle complex. Not every such bilinear map defines a cup-product; the map  $\cup$  must satisfy an additional technical condition known as a Leibniz rule – see [44, 18, 114]. However, given such a cup-product, we can construct a circuit of diagonal physical gates that preserves the code-space of the CSS code associated with the cochain complex. In general, a cup-product on an  $m$ -term cochain complex will yield a physical circuit of diagonal  $C^{m-2}Z$  gates from the  $m - 1$ -th level of the Clifford hierarchy [18]. In the situation of interest, we have the 4-term cochain complex in Eq. (25), leading to a circuit of  $CCZ$  gates. As discussed in Sec. 2.2, such a circuit can be specified by a trilinear function  $f$  acting on the space of physical qubits – the 1-cochains  $C^1$  in our convention. The circuit associated with the cup-product is then defined by the function

$$f_{\cup}(q_i, q_j, q_k) = |(q_i \cup q_j) \cup q_k| \bmod 2 \in \mathbb{F}_2 \quad (33)$$

where non-zero values indicate a physical  $CCZ$  gate acting between the qubits of three distinct code-blocks labeled by those arguments of  $f_{\sqcup}$ . The  $|\cdot|$  here is the size of the support of the argument when considered as a binary vector with respect to the  $X^i$  bases. Circuits defined by such functions constructed from cup-products of finitely many 1-cochains (three, in this case) lead to sparse, finite depth circuits [18, 63]. Moreover, such functions  $f_{\sqcup}$  naturally have the property that they vanish on 1-coboundaries:  $f_{\sqcup}$  vanishes if any of its three arguments is of the form  $d^0(c)$  for  $c \in C^0$ , where  $d^0$  is the coboundary map from  $C^0$  to  $C^1$  – equivalent to the matrix  $H_X^T$ . This condition is thus equivalent to  $f_{\sqcup}$  vanishing when any one of its arguments corresponds to an  $X$ -type stabilizer while the other two arguments correspond to non-trivial  $X$ -type logical operators (see Sec. 2.2) and is called *coboundary invariance*. The coboundary invariance property is essential to ensure that the circuit defined by  $f_{\sqcup}$  descends to a well-defined logical operation [63, 18] – intuitively, this is so that the action of  $f_{CCZ}$  on logical operators does not change upon adding stabilizers to the logical operators.

The formalism of [18] constructs cup-products on classical 2-term cochain complexes which satisfy certain sufficient conditions such that homological and balanced product complexes constructed from them (see supplementary material) are also equipped with an appropriate cup-product. This yields a trilinear function of the form in Eq. (33) on the product complexes associated with quantum CSS codes. Since tricycle codes are balanced products of classical Abelian group-algebra codes, these methods readily apply, which we review here. We will now review the key elements of this formalism.

To equip a two-term cochain complex  $C^\bullet : C^0 \xrightarrow{\delta} C^1$  corresponding to a classical linear code with a cup-product, one must construct partitions of the 1-coboundaries  $B^1$  (see supplementary material), known as *pre-orientations*. For any  $a \in C^0$ , we consider partitions

$$\delta(a) = \delta_{in}(a) \sqcup \delta_{out}(a) \sqcup \delta_{free}(a) \quad (34)$$

where  $\sqcup$  denotes a disjoint union and we identify an element  $x \in C^\bullet$  with its support as binary vector. The corresponding partitions are called 'in', 'out' and 'free' preorientations of the element  $a \in C^0$  respectively. A choice of pre-orientation induces a cup-product on the classical code by the following definition.

**Definition 5** (Classical complex cup-product). *The cup-product on a 2-term cochain complex induced by the preorientations of Eq. (34) is defined as*

$$a \cup a = a \quad \forall a \in C^0 \quad (35)$$

$$a \cup x = x \quad \text{for } a \in C^0, x \in C^1 \quad \text{if } x \in \delta_{out}(a) \quad (36)$$

$$x \cup a = x \quad \text{for } a \in C^0, x \in C^1 \quad \text{if } x \in \delta_{in}(a) \quad (37)$$

$$0 \quad \text{otherwise} \quad (38)$$

Note that this cup-product is not associative in general:  $(a \cup b) \cup c \neq a \cup (b \cup c)$ .

Different choices of partitions induce different cup-products on the classical code according to the rules in Definition 5. Given three such classical group algebra codes defined by elements  $a, b, c \in \mathbb{F}_2[G]$ , each with their respective in, out, and free partitions, the three-fold balanced product tricycle code can be equipped with a natural cup-product that is inherited from the cup-products on the classical codes – see Ref. [18] for details on the inherited cup-product on quantum codes.

Additional conditions on the classical preorientations are required for the inherited cup-product on quantum codes – and hence the function  $f_{\cup}(q_i, q_j, q_k)$  as defined in Eq. (33) – to induce a code-space preserving circuit. Ref. [18] shows that a sufficient condition on the preorientations is the so called *integrated Leibniz rule*.

**Proposition 2** (Integrated Leibniz Rule). *The cup-product on classical codes from Definition 5 is said to satisfy an integrated Leibniz rule if for all  $a_1, a_2, a_3 \in C^{\bullet}$*

$$|(\delta(a_1) \cup a_2) \cup a_3| + |(a_1 \cup \delta(a_2)) \cup a_3| + |(a_1 \cup a_2) \cup \delta(a_3)| = 0 \bmod 2 \quad (39)$$

*The integrated Leibniz rule on three classical codes is a sufficient condition for the CCZ circuit defined by the inherited cup-product on the product quantum code,  $f_{\cup}(q_1, q_2, q_3) = |(q_1 \cup q_2) \cup q_3| \bmod 2$ , to be a coboundary invariant operation – i.e a code-space preserving unitary operator.*

*Proof.* See section 5 of Ref. [18]. □

From Eq. (45), further equivalent sub-conditions can be derived for the ‘in’, ‘out’, and ‘free’ partitions. These conditions are listed in proposition 5.2 of [18]. However, we note that proposition 5.2 as stated in [18] is technically incorrect unless the cup-product is associative which is not generically the case. While the cup-product is indeed associative when the ‘non-overlapping bits’ condition –  $\delta_{in}(a_1) \cap \delta_{in}(a_2) = \emptyset$  for  $a_1 \neq a_2$  (and similarly for  $\delta_{out}$ ) – holds, this is no longer true when bits of the *in* and *out* partitions are allowed to overlap, which is the case for tricycle codes when one or more of the group-algebra elements have weight  $> 2$ . The correct set of conditions can be derived by applying the cup-products in sequence as  $(a \cup b) \cup c$  in the derivation of proposition 5.2 of [18]. However, we find that the resulting conditions are too restrictive on the parameters of the code – for instance, these conditions can be found in Refs. [51, 60] in which they were used to study  $2 - 2 - 2$  codes with  $K = 3$  and  $4 - 2 - 2$  and  $4 - 4 - 4$  codes with  $D = 2$  with non-trivial CCZ action.

We instead derive a new set of conditions on the pre-orientations that follow from mixing the order of cup-product applications for different orders of arguments to define a different trilinear operation – we call this operation a *symmetric triple cup-product*, as illustrated by the following definition.

**Definition 6** (Symmetric triple cup-product). *The symmetric triple cup-product is a trilinear function on a 2-term cochain complex*

$$_ \cup _ \cup _ : C^i \times C^j \times C^k \rightarrow C^l \quad i, j, k, l \in \{0, 1\}$$

*that is non-zero only if  $i + j + k = l$  and is defined as*

$$a_1 \cup a_2 \cup a_3 \equiv (a_1 \cup a_2) \cup a_3 \quad \text{if } a_1 \in C^1, a_2 \in C^0, a_3 \in C^0 \quad (40)$$

$$(a_1 \cup a_2) \cup a_3 \quad \text{if } a_1 \in C^0, a_2 \in C^1, a_3 \in C^0 \quad (41)$$

$$a_1 \cup (a_2 \cup a_3) \quad \text{if } a_1 \in C^0, a_2 \in C^0, a_3 \in C^1 \quad (42)$$

$$(a_1 \cup a_2) \cup a_3 \quad \text{if } a_1 \in C^0, a_2 \in C^0, a_3 \in C^0 \quad (43)$$

$$0 \quad \text{otherwise} \quad (44)$$

where the cup-products in parentheses are the standard ones defined in Definition 5.

Henceforth, whenever a triple cup-product is written with no parentheses, it is to be understood as the symmetric version defined above. Note that the order of parentheses for the specific cases in Eq. (41) and Eq. (43) does not matter as for such arguments the standard cup-product is associative. The product quantum code (homological or balanced products) of three classical codes equipped with symmetric triple cup-products naturally inherits a symmetric triple cup-product operation defined on the 4-term cochain complex corresponding to the quantum code. The symmetric triple cup-product on product quantum codes is defined essentially identically as in the original cup-product construction of Ref. [18], except with the order of operations taken as in Definition 6. Similarly, one can show that the appropriate version of the integrated Leibniz rule is sufficient for the trilinear function defined by the symmetric triple cup-product on quantum codes to produce a code-space preserving *CCZ* circuit.

**Proposition 3** (Symmetric Integrated Leibniz Rule). *The symmetric triple cup-product on classical codes is said to satisfy an integrated Leibniz rule if for all  $a_1, a_2, a_3 \in C^\bullet$*

$$|\delta(a_1) \cup a_2 \cup a_3| + |a_1 \cup \delta(a_2) \cup a_3| + |a_1 \cup a_2 \cup \delta(a_3)| = 0 \bmod 2 \quad (45)$$

*The integrated Leibniz rule on the symmetric triple cup-product is a sufficient condition for the *CCZ* circuit defined by the inherited symmetric triple cup-product on the quantum code,  $f_{\cup}^{sym}(q_1, q_2, q_3) = |q_1 \cup q_2 \cup q_3| \bmod 2$ , to be a coboundary invariant operation – i.e a code-space preserving unitary operator.*

*Proof.* The proof is analogous to that of Lemma 5.1 of Ref. [18]. □

The symmetric integrated Leibniz rule can then be expressed in terms of a number of sub-conditions on the coboundary partitions.

**Proposition 4.** *A preorientation on a classical code induces a symmetric triple cup-product that satisfies the integrated Leibniz rule if the 'in', 'out', and 'free' partitions satisfy the following conditions:*

$$|\delta_{in}(a_1)| + |\delta_{out}(a_1)| = 0 \bmod(2) \quad \forall a_1 \in X^0 \quad (46)$$

$$|\delta_{in}(a_1) \cap \delta_{free}(a_2)| = 0 \bmod(2) \quad \forall a_1, a_2 \in X^0 : a_1 \neq a_2 \quad (47)$$

$$|\delta_{out}(a_1) \cap \delta_{free}(a_2)| = 0 \bmod(2) \quad \forall a_1, a_2 \in X^0 : a_1 \neq a_2 \quad (48)$$

$$|\delta_{in}(a_1) \cap \delta_{in}(a_2)| + |\delta_{out}(a_1) \cap \delta_{out}(a_2)| = 0 \bmod(2) \quad \forall a_1, a_2 \in X^0 : a_1 \neq a_2 \quad (49)$$

$$|\delta_{in}(a_1) \cap \delta_{in}(a_2) \cap \delta_{in}(a_3)| + |\delta_{out}(a_1) \cap \delta_{out}(a_2) \cap \delta_{out}(a_3)| + |\delta_{free}(a_1) \cap \delta_{in}(a_2) \cap \delta_{in}(a_3)| \quad (50)$$

$$+ |\delta_{out}(a_1) \cap \delta_{out}(a_2) \cap \delta_{free}(a_3)| + |\delta_{out}(a_1) \cap \delta_{free}(a_2) \cap \delta_{in}(a_3)| = 0 \bmod(2)$$

$$\forall a_1, a_2, a_3 \in X^0 : a_1 \neq a_2 \neq a_3$$

where  $X^0$  is the basis set for  $C^0$ .

*Proof.* By direct case-by-case computation using Definition 5, Definition 6, and the Leibniz rule. □

Note that these conditions apply to the classical codes from which any three-dimensional homological product and balanced product quantum codes are constructed. Compared to the original conditions of Ref. [19] these conditions are less restrictive and may yield product codes with  $CCZ$  gates with better parameters for other code families besides tricycle codes.

We now apply these conditions to the classical Abelian group Algebra codes which are used to construct tricycle codes. Below, we will use  $|a|$  to denote the weight of the group-algebra element  $a$  – i.e the number of group elements that appear in its expansion. We also write  $a_1 \cap a_2$  for  $a_1, a_2 \in \mathbb{F}_2[G]$  to denote the set of group elements that are common to both group-algebra elements. For a classical Abelian group-algebra code associated with the two-term cochain complex of Eq. (23) defined by an element  $a \in \mathbb{F}_2[G]$ , defining a preorientation amounts to choosing a partition of the group algebra element into smaller disjoint group-algebra elements

$$a = a_{in} + a_{out} + a_{free} \quad (51)$$

where  $a_{in}, a_{out}, a_{free} \in \mathbb{F}_2[G]$ . The preorientations are then defined as  $\delta_{in}(\alpha) = a_{in} \cdot \alpha$ ,  $\delta_{out}(\alpha) = a_{out} \cdot \alpha$  and  $\delta_{free}(\alpha) = a_{free} \cdot \alpha$  for  $\alpha \in \mathbb{F}_2[G]$  where the multiplications are within the group-algebra. Note that for balanced product codes, the formalism of Ref. [19] requires an additional condition that preorientations are preserved by group actions. The preorientations in Eq. (51) naturally satisfy this condition – for example,  $g \cdot \delta_{in}(\alpha) = g \cdot a_{in} \cdot \alpha = \delta_{in}(g \cdot \alpha)$  and similarly for the other partitions.

The conditions in Eq. (46)-Eq. (50) take a particularly simple form when the free partition is empty. In this case, the conditions on the classical preorientation of the code defined by  $a \in \mathbb{F}_2[G]$  are then (with the conditions on the  $b$  and  $c$  classical code preorientations defined identically):

$$|a_{in}| + |a_{out}| = 0 \bmod(2) \quad (52)$$

$$|a_{in} \cdot g \cap a_{in} \cdot h| + |a_{out} \cdot g \cap a_{out} \cdot h| = 0 \bmod(2) \quad \forall g, h \in G : g \neq h \quad (53)$$

$$|a_{in} \cdot f \cap a_{in} \cdot g \cap a_{in} \cdot h| + |a_{out} \cdot f \cap a_{out} \cdot g \cap a_{out} \cdot h| = 0 \bmod(2) \quad (54)$$

$$\forall f, g, h \in G : f \neq g \neq h$$

To construct high-rate and distance tricycle codes from weight-4 group-algebra elements which satisfy the above conditions, it turns out to be necessary to set the free partition to be empty (see the discussion after Theorem D.1), and thus we will work with this version of the Leibniz rule conditions. For three classical group-algebra codes with preorientations that satisfy the above conditions, the balanced product yields a symmetric triple cup-product on the quantum code's cochain complex, and consequently produces a constant depth  $CCZ$  circuit defined by the induced symmetric triple cup-product on the quantum code  $f_{CCZ}(q_1, q_2, q_3) = |q_1 \cup q_2 \cup q_3| \bmod 2$  (the precise definition of  $f_{CCZ}$  is discussed in Proposition 5 below).

Note that for weight 2 group-algebra elements, these conditions are trivially satisfied automatically by choosing  $a_{in}$  and  $a_{out}$  to be singletons. All codes in Table 1 are constructed by finding classical group-algebra elements which satisfy these conditions with a numerical search. A simple rule to generate weight 4 group-algebra elements that satisfy these conditions is to choose  $a = a_1 + a_1 \cdot s + a_2 + a_2 \cdot s$  for a fixed element  $s \in G$  such that none of the 4 terms are equal. As long as  $s^2 \neq e$ , we are able to find tricycle codes constructed from such weight-4 elements with non-trivial  $CCZ$  circuits and  $k > 3, d > 2$ , circumventing the limitations of the original conditions in [18]. We formalize this with the following theorem.

**Theorem D.1.** *There exist weight-4 elements  $a = a_1 + a_2 + a_3 + a_4 \in \mathbb{F}_2[G]$  that satisfy conditions Eq. (52) - Eq. (54) for any  $G$  with  $|G| \geq 4$ .*

*Proof.* We specify one prescription for constructing preorientations which satisfy Eq. (52) - Eq. (54). First note that  $|\alpha \cdot g \cap \beta \cdot h| = |\alpha \cap \beta \cdot hg^{-1}|$  for any  $\alpha, \beta \in \mathbb{F}_2[G]$  and  $g, h \in G$ .

Thus we can simplify the conditions to (where  $e$  denotes the group identity element below)

$$|a_{in}| + |a_{out}| = 0 \bmod(2) \tag{55}$$

$$|a_{in} \cap a_{in} \cdot w| + |a_{out} \cap a_{out} \cdot w| = 0 \bmod(2) \quad \forall w \in G : w \neq e \tag{56}$$

$$|a_{in} \cap a_{in} \cdot v \cap a_{in} \cdot w| + |a_{out} \cap a_{out} \cdot v \cap a_{out} \cdot w| = 0 \bmod(2) \tag{57}$$

$$\forall v, w \in G : v \neq w \neq e,$$

Now we choose any  $a_1, a_3 \in G$  with  $a_1 \neq a_3$  and choose another fixed element  $s \in G$  with  $s \neq e$  such that:  $a_1s \neq a_3s$  and  $a_1s \neq a_3$  and  $a_1 \neq a_3s$ . Then we define  $a_2 \equiv a_1s$  and  $a_4 \equiv a_3s$  and set the partitions  $a_{in} = a_1 + a_2$ ,  $a_{out} = a_3 + a_4$  and  $a_{free} = \emptyset$ . Eq. (55) is clearly satisfied since both partitions have weight 2. Next, notice that  $|a_{in} \cap a_{in}w| = \mathbb{1}_{w=s}$  and similarly  $|a_{out} \cap a_{out}w| = \mathbb{1}_{w=s}$ . Thus  $|a_{in} \cap a_{in}w| + |a_{out} \cap a_{out}w| = 2 \cdot \mathbb{1}_{w=s} = 0 \bmod 2$  so Eq. (56) is also satisfied. Finally, since  $a_{in}, a_{out}$  both are weight 2, it is easy to see that  $a_{in} \cap a_{in} \cdot v \cap a_{in} \cdot w = \emptyset$  and similarly for  $a_{out}$  (this is true for any weight 2 group-algebra elements and non-identity  $v \neq w$ ). Thus, Eq. (57) is also satisfied.  $\square$

The prescription in the proof of Theorem D.1 is not strictly the only way to satisfy conditions Eq. (46) - Eq. (50) for weight-4 elements. Another possibility involves working with groups  $G$  that have an *involution*  $t \in G$  such that  $t^2 = e$ . One can then construct preorientations with non-empty free partitions and one of the in and out partitions being empty – for example, by choosing  $a_{in} = a_1 + a_1t$  and  $a_{free} = a_3 + a_3t$ . However, we empirically find that tricycle codes which use such group algebra elements always have distance  $D = 2$ . More generally, we find that when the offset  $s$  used to construct the partitions of Theorem D.1 is an involution,  $D = 2$  always, although we are unable to prove it. This appears to be the same structure that causes all the  $4 - 2 - 2$  and  $4 - 4 - 4$  codes studied in Ref. [51] using the original cup-product conditions to have  $D = 2$ , as their construction can also be interpreted in terms of using an involutive group element to satisfy the associated conditions. The symmetric triple cup-product conditions we introduce instead allow us to choose offsets  $s$  that are not involutions, leading to tricycle codes with better parameters.

Next, we define the  $f_{CCZ}$  function for the quantum tricycle codes that arises from the cup-product specified by a choice of preorientations on the constituent classical group-algebra codes. The structure of the cochain complex in Eq. (25) implies that we can partition the qubits of the code into three sectors of  $|G|$  qubits each – see Sec. A. Within each sector, a qubit can be labelled by a group element. Given some order of the group elements, the notation  $g^i$  will thus be used to denote qubit  $g$  of the  $i$ -th sector ( $i = I, II, III$ ). For the following, denote  $\alpha_I \equiv a$ ,  $\alpha_{II} \equiv b$ ,  $\alpha_{III} \equiv c$  as the three group-algebra elements that specify the classical codes (this notation makes the expression below compact).

**Proposition 5.** *Consider a tricycle code defined by group-algebra elements  $\alpha_I, \alpha_{II}, \alpha_{III} \in \mathbb{F}_2[G]$  with respective ‘in’ and ‘out’ pre-orientations. The function  $f_{CCZ}$  specifying the transversal  $CCZ$  circuits in Sec. 2.2 is defined as*

$$f_{CCZ}(p^i, q^j, r^k) = |r\alpha_i^{in}\alpha_j^{in} \cap q\alpha_i^{in}\alpha_k^{out} \cap p\alpha_j^{out}\alpha_k^{out}| \cdot \delta_{i \neq j \neq k} \pmod{2} \quad (58)$$

where  $i, j, k \in \{I, II, III\}$  label the sectors of the qubits  $p, q, r$  in the three code-blocks respectively, and  $\delta_{i \neq j \neq k}$  indicates that  $f_{CCZ} = 0$  if any two or more qubits from its arguments are in the same sector. If any partition in the above expression is empty,  $f_{CCZ} = 0$  on these arguments.

*Proof (informal).* This definition follows from a straightforward but tedious computation of the cup-product on the balanced-product quantum code induced by the classical code cup-products, with the new mixed-orders we define to attain the conditions in Eq. (46)–Eq. (49).  $\square$

Note that the expression in Proposition 5 holds for any tricycle code equipped with pre-orientations that satisfy the conditions of Eq. (46)–Eq. (50), not just those constructed from weight-4 group-algebra elements. To be clear, the resulting  $CCZ$  circuit has a  $CCZ$  gate acting on qubits  $p^i, q^j, r^k$  if and only if  $f_{CCZ}(p^i, q^j, r^k) = 1$ . The  $\delta_{i \neq j \neq k}$  structure is manifest in the illustrations of the circuits in Fig. 1, where no  $CCZ$  acts between qubits in the same sector across code-blocks. One must check (numerically) whether the resulting circuits have a non-trivial logical action or not, as has been done for all the codes listed in Table 1.

From the  $f_{CCZ}$  function of Proposition 5, it is not difficult to ascertain the maximum degrees (the maximum number of  $CCZ$  gates a particular qubit is involved in) of the resulting  $CCZ$  circuits which use the construction in Theorem D.1 to construct weight-4 group-algebra elements with valid preorientations for the  $4-2-2$ ,  $4-4-2$ , and  $4-4-4$  type tricycle codes. Below, we let  $s_i$  denote the offset element in the proof of Theorem D.1 chosen for  $\alpha_i$ ,  $i \in \{I, II, III\}$ , and we list weight-4 elements before weight-2 elements – for example, for  $4-4-2$  codes we let  $\alpha_I$  and  $\alpha_{II}$  be weight-4 and  $\alpha_{III}$  be weight-2.

A straightforward analysis shows that the possible maximum circuit degrees are:

1.  $4-2-2$  codes: degree 8
2.  $4-4-2$  codes:
  - $s_1 = s_2$ : degree 16
  - If  $s_1 \neq s_2$ : degree 32
3.  $4-4-4$  codes:
  - $s_1 = s_2 = s_3$ : degree 12
  - $s_i \neq s_j = s_k$  for  $i, j, k \in \{I, II, III\}$  and  $i \neq j \neq k$ : degree 64
  - $s_1 \neq s_2 \neq s_3$ : degree 128

We found empirically that the  $4-4-4$  codes with degree 64 and degree 12 circuits from the above constructions had relatively poor rate and distance parameters, which is why they are not included Table 1.

In the main text, we abuse terminology and refer to the maximum degree of a circuit and the minimal depth of the circuit interchangeably. For the  $CCZ$  circuits considered in this work, the true

minimal circuit depth is related to the chromatic index of the corresponding tripartite 3–uniform hypergraph. A standard bound [75] relates the minimal depth and maximum degree  $\Delta$  as

$$\text{depth} \leq 3(\Delta - 1) \quad (59)$$

Another widely believed conjecture [2] states that

$$\text{depth} \leq \frac{3\Delta}{2} \quad (60)$$

for 3–uniform hypergraphs where any two hyperedges share at-most 2 vertices, as is the case for the  $CCZ$  circuits we consider in our work. For example, using an integer programming method, we found that the minimal depth of the  $CCZ$  circuit for the main example  $[[48, 6, (8, 4)]]$   $4 - 2 - 2$  code considered in the main-text (Sec. 2.4) is 10, while the maximum degree of the circuit is 8. In practice, it is the degree of the circuit that controls the spread of errors between qubits, while extra circuit layers from scheduling conflicts simply add a relatively small idling error.

## E Numerical Leibniz rule for $CCZ$ gates on $3D$ balanced-product codes

We now present a numerical method that generalizes the cup-product based gate approach to construct code-space preserving  $CCZ$  circuits on three-dimensional balanced product codes. We applied this method to two  $4 - 4 - 4$  tricycle code in Table 1 to find circuits with depth shorter than that from the symmetric triple cup-product (STCP) conditions in Eq. (46)-Eq. (50) (which typically have depth 128 for  $4 - 4 - 4$  codes). We refer to the method described in the remainder of this section as the ‘Numerical Leibniz Rule’ (NLR). This method yields code-space preserving circuits for any three-dimensional balanced product code. However, unlike the cup-product based construction, there is no guarantee on the depths of the resulting circuits. We describe a procedure that nevertheless allows some empirical control of the resulting circuit depths for certain codes, which are typically shorter than or comparable to the depths of the cup-product based  $CCZ$  circuits with appropriate choices of method hyperparameters. We present the key ideas in this section, while we intend to provide a more detailed exposition of this method in future work.

The approach involves defining an Ansatz for a code-space preserving trilinear function  $f_{CCZ}$  on  $3D$  balanced-product codes in terms of trilinear functions on the component classical codes. We then impose a generalization of the integrated Leibniz rule in Proposition 3 on each of the classical trilinear functions, which we will show is a sufficient condition for the trilinear function on the quantum code to be a coboundary invariant operation corresponding to a code-space preserving  $CCZ$  circuit. We assume familiarity with the homological formulation of balanced product codes – see the supplementary material for a brief introduction. For simplicity of terminology, we describe the method as it applies to tricycle codes, but it can be readily extended to other  $3D$  balanced product codes as well as to code-space preserving  $C^kZ$  circuits on  $k$ -dimensional balanced product codes.

We consider the cochain complex of Eq. (25) with the 1–cochains  $\tilde{C}^1$  denoting the space of

qubits. A basis for  $\tilde{C}^1$  is given by elements

$$p_1 := [p, 1, 1]_G \quad q_2 := [1, q, 1]_G \quad r_3 := [1, 1, r]_G \quad \forall p, q, r \in G \quad (61)$$

where  $\alpha_1, \alpha_2, \in G$  are 1 cochains of the classical group-algebra code complex in Eq. (23), which we denote as  $C_i^1$  ( $i = 1, 2, 3$  for each classical code), and  $1 \in G$  denotes the group identity element which is to be understood as an element of the 0-cochains of the classical complex denoted by  $C_i^0$  (see section 5.3 of Ref. [18] for more details). The subscripts 1, 2, 3 indicate which index of the vector is an element of  $C_i^1$ , while the others are elements of  $C_i^0$ .

The notation  $[\cdot, \cdot, \cdot]_G$  is to be understood as a vector in  $(\mathbb{F}_2[G])^{\oplus 3}$  subject to the equivalence relation

$$[g^{-1} \cdot \alpha, gh^{-1} \cdot \beta, h \cdot \gamma]_G \sim [\alpha, \beta, \gamma]_G \quad \forall g, h \in G \quad \alpha, \beta, \gamma \in \mathbb{F}_2[G] \quad (62)$$

This relation reflects the group-action quotient space structure of the balanced product code [19] (see supplementary material). We now construct a trilinear operation on the quantum code that shares some structural properties with the cup-product based trilinear function in Eq. (33). We start by defining appropriate functions on the classical code complexes.

**Definition 7** (Group-equivariant functions on classical group-algebra codes). *Let  $C_i$  denote the cochain complex of the  $i^{\text{th}}$  classical group algebra code ( $i = 1, 2, 3$ ) as in Eq. (23), with  $C_i^0 \equiv \mathbb{F}_2[G]$  and  $C_i^1 \equiv \mathbb{F}_2[G]$  denoting the 0 and 1 cochain spaces respectively. Define a set of functions*

$$f_i^j : C \times C \times C \rightarrow \mathbb{F}_2 \quad (63)$$

for  $i, j \in \{1, 2, 3\}$  with the properties

1.  $f_i^j(x_1, x_2, x_3)$  is non-zero only if  $x_j \in C_i^1$  and  $x_k \in C_i^0$  for  $k \neq j$ .
2. Group equivariance:  $f_i^j(x_1, x_2, x_3) = f_i^j(g \cdot x_1, g \cdot x_2, g \cdot x_3) \quad \forall g \in G$

The  $f_i^j$  functions can be viewed as a generalization of the cup-product operation on the classical codes. Now we construct a function  $f_{CCZ} : \tilde{C}^1 \times \tilde{C}^1 \times \tilde{C}^1 \rightarrow \mathbb{F}_2$  on the quantum code that is defined in terms of these functions, such that  $f_{CCZ}$  respects the group-action balancing relation in Eq. (62). To this end, we consider the following structure for  $f_{CCZ}$ .

**Definition 8** (Product Ansatz for trilinear function on tricycle codes). *Let  $f_i^j$  with  $i, j \in \{1, 2, 3\}$  be group-equivariant trilinear functions as defined in Definition 7. Let  $p^i, q^j, r^k$  for  $i, j, k \in \{1, 2, 3\}$  be basis elements of  $\tilde{C}^1$  as in Eq. (61). Then,  $f_{CCZ}$  is a trilinear function on  $\tilde{C}^1$  defined on the basis elements as*

$$f_{CCZ}(p_i, q_j, r_k) = \begin{cases} \sum_{h_1, h_2, h_3 \in G} f_1^1(g_1 \cdot p, g_2, g_3) \cdot f_2^2(g_1^{-1} h_1^{-1}, g_2^{-1} h_2^{-1} \cdot q, g_3^{-1} h_3^{-1}) \cdot f_3^3(h_1, h_2, h_3 \cdot r) \bmod 2 & i = 1, j = 2, k = 3 \\ \sum_{h_1, h_2, h_3 \in G} f_1^1(g_1 \cdot p, g_2, g_3) \cdot f_2^3(g_1^{-1} h_1^{-1}, g_2^{-1} h_2^{-1}, g_3^{-1} h_3^{-1} \cdot r) \cdot f_3^2(h_1, h_2 \cdot q, h_3) \bmod 2 & i = 1, j = 3, k = 2 \\ \sum_{h_1, h_2, h_3 \in G} f_1^2(g_1, g_2 \cdot q, g_3) \cdot f_2^1(g_1^{-1} h_1^{-1} \cdot p, g_2^{-1} h_2^{-1}, g_3^{-1} h_3^{-1}) \cdot f_3^3(h_1, h_2, h_3 \cdot r) \bmod 2 & i = 2, j = 1, k = 3 \\ \sum_{h_1, h_2, h_3 \in G} f_1^3(g_1, g_2, g_3 \cdot r) \cdot f_2^1(g_1^{-1} h_1^{-1} \cdot p, g_2^{-1} h_2^{-1}, g_3^{-1} h_3^{-1}) \cdot f_3^2(h_1, h_2 \cdot q, h_3) \bmod 2 & i = 2, j = 3, k = 1 \\ \sum_{h_1, h_2, h_3 \in G} f_1^2(g_1, g_2 \cdot q, g_3) \cdot f_2^3(g_1^{-1} h_1^{-1}, g_2^{-1} h_2^{-1}, g_3^{-1} h_3^{-1} \cdot r) \cdot f_3^1(h_1 \cdot p, h_2, h_3) \bmod 2 & i = 3, j = 1, k = 2 \\ \sum_{h_1, h_2, h_3 \in G} f_1^3(g_1, g_2 \cdot q, g_3) \cdot f_2^2(g_1^{-1} h_1^{-1}, g_2^{-1} h_2^{-1} \cdot q, g_3^{-1} h_3^{-1}) \cdot f_3^1(h_1 \cdot p, h_2, h_3) \bmod 2 & i = 3, j = 2, k = 1 \\ 0 & \text{otherwise} \end{cases} \quad (64)$$

The above definition may appear involved, but each of the 6 nonzero cases simply defines the trilinear function on a triple of sectors of the three-blocks of the tricycle codes, producing a  $CCZ$  circuit structure as in Fig. 1, analogous to the cup-product based circuits from Sec. D. The 6-fold sum over group elements combined with group-equivariance of the component functions ensures that  $f_{CCZ}$  is well-defined on the equivalence classes of the balanced product in Eq. (62).

To ensure that the  $f_{CCZ}$  function in Eq. (64) is a coboundary invariant operation that produces a code-space preserving  $CCZ$  circuit, we impose a version of the Leibniz rule similar to that in Proposition 3 on the  $f_i^j$  functions on classical codes.

**Proposition 6** (Generalized Leibniz rule). *Suppose the  $i^{th}$  classical group-algebra code is defined by an element  $\alpha_i \in \mathbb{F}_2[G]$ , as in Eq. (23) ( $i \in \{1, 2, 3\}$ ). The functions  $f_i^j$  in Definition 7 are said to satisfy a generalized Leibniz rule if,*

$$f_i^1(\alpha_i \cdot g_1, g_2, g_3) + f_i^2(g_1, \alpha_i \cdot g_2, g_3) + f_i^3(g_1, g_2, \alpha_i \cdot g_3) = 0 \bmod 2 \quad \forall g_1, g_2, g_3 \in G \quad (65)$$

A sufficient condition for  $f_{CCZ}$  from Definition 8 to correspond to a code-space preserving circuit is for all  $f_i^j$  to satisfy the generalized Leibniz rule.

*Proof.* The proof is structurally identical to that of Lemma 5.1 of Ref. [18]. In particular, one can show that the  $f_{CCZ}$  function inherits a similarly defined generalized Leibniz rule from the component  $f_i^j$  functions due to the product structure of Eq. (64). The generalized Leibniz rule is then in turn a sufficient condition for  $f_{CCZ}$  to be well-defined on the first cohomology space of the complex Eq. (25).  $\square$

The NLR method can then be summarized as:

1. Construct a basis-set of possible group-equivariant trilinear functions  $f_i^j$  that satisfy Eq. (65) on fixed classical codes defined by  $\alpha_i \in \mathbb{F}_2[G]$  for  $i = 1, 2, 3$ .
2. Search randomly over candidates  $f_i^j$ : for each set of candidates, construct  $f_{CCZ}$  as in Eq. (64) and iterate until the maximum degree of  $f_{CCZ}$  is low.

Step 1 can be formulated efficiently. In particular, for each  $i = 1, 2, 3$ , the set of functions  $f_i^1, f_i^2, f_i^3$  are codewords of a classical binary parity check code with parity check matrix  $H_{leibniz}$  of shape  $|G|^2 \times 3|G|^2$ . Each row of  $H_{leibniz}$  corresponds to one constraint of the form Eq. (65) – there are  $|G|^2$  such constraints instead of  $|G|^3$  due to the group equivariance property of the  $f_i^j$  functions. The columns of  $H_{leibniz}$  have weight  $3|\alpha_i|$ , corresponding to the non-zero the constraint Eq. (65). Consequently, a basis of possible functions  $f_i^j$  that satisfy the generalized Leibniz rule can be found efficiently by finding a basis for the nullspace of  $H_{leibniz}$ .

Step 2. can be optimized with various heuristics. In practice, we found that the shortest depth circuits were found by generating a large set of low-weight codewords of  $H_{leibniz}$  using a variant of the algorithm described in Sec. C and randomly searching over such a set. This is the method used to find the circuits for the  $4 - 4 - 4$  type NLR codes in Table 1. We numerically verified that the resulting circuits indeed preserve the code-space, by explicitly checking the coboundary invariance condition. In general, we find that sparsity of the functions (low weight of the codewords) leads to lower-degree  $f_{CCZ}$  functions. Other strategies using simulated annealing or structured searches over a basis of codewords for  $H_{leibniz}$  could potentially lead to finding lower degree circuits for a particular code. In particular, it is currently unclear if this method can efficiently yield circuits for certain codes with depths  $\leq 20$  to be feasible for practical magic-state generation.

One caveat with this method is that it aims to construct a code-space preserving circuit for a fixed code. Consequently, for certain codes, there may be no such circuits with low degree. Therefore, a code-search must be performed in conjunction with this method to find good codes with short-depth  $CCZ$  circuits. We will study this method and its application to other codes in more detail and generality in future work.

## F Logical circuit optimization

In this section we discuss how the logical  $\overline{CCZ}$  circuits that produce the hypergraph magic states described in Sec. 2.2 can be optimized to maximize the yield of disjoint  $\overline{CCZ}$  magic-gates.

The logical connectivity is determined by the cohomology invariant  $f_{CCZ}$  function restricted to a basis of logical  $X$ -type operators for the three code-blocks. In cohomological terms, this is a basis for the first cohomology group  $H^1(C)$  of the associated cochain complex. Denote a choice of these bases as  $\{[l_i^1]\}_{i=1\dots K}, \{[l_j^2]\}_{j=1\dots K}, \{[l_k^3]\}_{k=1\dots K}$  where the  $l_i$  are representative elements of  $\ker(H_Z)$  and  $[\cdot]$  denotes an equivalence class modulo elements of  $\text{im}(H_Z^T)$ . We then construct a  $K \times K \times K$  binary 3-tensor corresponding to the restriction of  $f_{CCZ}$  to the logical operators:

$$T_{ijk}^{log} = f_{CCZ}(l_i^1, l_j^2, l_k^3) \in \mathbb{F}_2 \quad (66)$$

The tensor  $T^{log}$  encodes all the structure of the resulting logical circuit:  $T_{ijk}^{log} = 1$  implies a  $\overline{CCZ}$  gate between the  $i$ -th logical qubit of the first code block,  $j$ -th logical qubit of the second block, and  $k$ -th logical qubit of the third block. Naturally, changing the logical basis of each block will change the connectivity of the resulting circuit and the associated hypergraph magic state, corresponding to a change of basis on each leg of the  $T^{log}$  tensor. We would like to be able to extract some number of disjoint  $\overline{CCZ}$  gates from the circuit defined by  $f_{CCZ}$  by choosing an appropriate basis of logical  $X$  operators for each code block. The maximum such number of disjoint  $\overline{CCZ}$ ,  $K_{CCZ}$ , that are

extractable from the circuit is equal to the *subrank* of  $T^{log}$ . Intuitively, the subrank of a binary tensor measures the size of the largest identity-like diagonal subtensor that can be embedded within the tensor [26, 52]. More formally, the subrank of the tensor  $T^{log}$  is the largest integer  $r \geq 0$  such that there exist matrices  $M^1, M^2, M^3 \in \mathbb{F}_2^{r \times K}$  that satisfy

$$(M^1 \otimes M^2 \otimes M^3) \cdot T^{log} = \mathbb{I}^{r \times r \times r} \bmod 2 \quad (67)$$

where  $\mathbb{I}^{r \times r \times r}$  is the  $r \times r \times r$  tensor with 1s on the diagonal and 0 everywhere else. Equivalently, we can express this in components as

$$\sum_{i,j,k} T_{ijk}^{log} M_{a,i}^1 M_{b,j}^2 M_{c,k}^3 = \mathbb{1}_{a=b=c} \bmod(2) \quad a, b, c \in [1 \cdots r] \quad (68)$$

The  $M$  matrices can be viewed as change of basis matrices such that rows of  $M$  define new logical operators in a different basis that produces  $r$  disjoint  $\overline{CCZ}$  gates. In particular, the optimal choice of  $r$  logical operators for the  $I$ -th code block is given by  $\{\sum_j M_{a,j}^I (l_a^I)_j\}_{a=1 \cdots r}$ . The remaining  $K - r$  logicals are found by choosing any set of logical operators that are linearly independent from the first  $r$ . These  $K - r$  logicals are treated as gauge logical qubits, to be initialized in the  $|\bar{0}\rangle$  state – see the discussion at the end of Sec. 2.2.

Unfortunately, finding the subrank  $r$  and the associated bases  $M^1, M^2, M^3$  is a problem that is widely believed to be NP-Hard, even over the real numbers [52]. In this work, we find sub-optimal solutions using a Mixed-Integer-Programming (MIP) method – we use the Gurobi Python package for this task [41]. We first find a set of initial logical operators numerically using row-reduction over  $\mathbb{F}_2$ . For fixed  $r$  we then treat the  $3rK$  entries of  $M^1, M^2, M^3$  as binary variables for the integer program, and enforce the condition in Eq. (68) as a set of integer constraints which involve another  $2r^3$  many slack variables. The complexity of the integer program therefore grows dramatically with  $r$ . We iterate from  $r = 0$  to increasing values of  $r$  until the MIP solver converges or is unable to find a feasible solution within time constraints. This leads to the reported values of  $K_{CCZ}$  in Sec. 2.2. An important direction for future work is to find tailored optimization heuristics for the binary tensor subrank problem to extract larger values of  $K_{CCZ}$ , significantly improving the throughput of  $\overline{CCZ}$  type magic states from the hypergraph magic states produced by our circuits.

## G Numerical details of noise simulations

Here, we provide further numerical simulations of the tricycle codes to demonstrate robust single-shot error correction in the  $Z$  basis and assess fault-tolerant behavior in the  $X$  basis using repeated rounds of syndrome extraction [15].

For both types of simulation, we simulate syndrome extraction circuits for both  $X$  and  $Z$  checks of the tricycle codes. Each simulation begins by initializing the data qubits in the appropriate basis, followed by multiple cycles of syndrome extraction. At each cycle, fresh ancilla qubits are initialized, entangled with the data via two-qubit gates, and subsequently measured to extract the stabilizer values.

For both single-shot  $Z$ -basis and  $d$ -round  $X$ -basis simulations, we employ a coloration circuit for syndrome extraction together with the standard two-qubit depolarizing circuit-noise error model

also used in the main text simulations.

For the single-shot scenario, we utilize a windowed decoding protocol, wherein each window comprises a fixed number of syndrome extraction rounds followed by decoding and correction. This window is then repeated multiple times, providing a concrete probe of the code’s sustainable logical error suppression [20]. Specifically, our simulations employ windows of three rounds, each repeated fourteen times for a total of forty-two rounds—a protocol previously demonstrated to saturate sustainable performance in high-rate codes [107]. With fixed window size, only codes possessing genuine single-shot capability exhibit logical error suppression that scales with code distance; in contrast, codes lacking such features see their performance limited by the window size. Although a window size of one is theoretically sufficient, empirical studies indicate that size three yields enhanced robustness to measurement errors and generally improved logical error rates [107, 14]. For conventional  $d$ -round simulations in the  $X$  basis, the syndrome extraction is repeated for  $d$  cycles, and decoding is performed using the entire record of accumulated syndrome information.

Here, we perform decoding using a belief-propagation with ordered statistics decoding (BP+OSD) scheme as presented in Ref. [76]. To enable circuit-level decoding, we map each syndrome extraction circuit to a spacetime code [107, 14]: checks of the spacetime code correspond to parities of stabilizer measurements across consecutive rounds, and the code qubits represent distinct fault locations (e.g., a two-qubit gate error at a specific location in space and time). Formally, for each round, the value of a stabilizer at time  $t$  is  $m_t \in \{0, 1\}$ ; the detector value is defined as  $d_t := m_t + m_{t-1} \pmod{2}$ . In single-shot simulations, belief propagation is applied to each window, except for the final readout which uses BP+OSD; for  $d$ -round simulations, BP+OSD is used for decoding the full syndrome record.

All circuits were constructed using Stim [35], and BP+OSD decoding was implemented using the tools from Ref. [83, 82]. For the  $X$ -basis  $d$ -rounds simulations, we apply a BP+OSD decoder to the full spacetime code with 10000 maximum iterations of min-sum BP and min-sum scaling factor of 0. We used OSD-CS of order 10. For the  $Z$ -basis single-shot simulations, we decode each window of three cycles using a BP decoder with maximum iterations set to  $\# \text{ qubits} / 5$  and min-sum scaling factor of 0.9. To decode the final transversal measurement we use BP+OSD with maximum iterations set to  $\# \text{ qubits} / 5$ , min-sum scaling factor of 0.9 and OSD-E of order 10. Although we did not attempt extensive hyperparameter optimization, preliminary trials suggest that targeted tuning of BP+OSD parameters could produce significant gains in logical error rate.

We simulate various  $4 - 4 - 4$  tricycle code and our simulation results are summarized in Fig. 6. Logical error rate is defined as the probability that any logical qubit within the code block fails under the simulated noise model. We observe a gate-noise threshold greater than 0.4% for each of the  $X$  and  $Z$  basis results (see the discussion after Theorem A.1 in for further details). By fitting the measured logical error rates to an exponential decay model,  $p_L = \alpha(p/p_{\text{th}})^{\beta n^\gamma}$ , we estimate the achievable error rates for both  $X$  and  $Z$  bases. Table 3 presents logical error rates for two key physical error benchmarks:  $p_{2q} = 10^{-3}$ , representative of near-term hardware, and  $10^{-4}$ , representative of projected advances, together with relevant code parameters.

We observe robust single-shot performance across several tricycle code instances.  $d$ -round performance is weaker, reflecting the higher check weight and reduced distance in the  $X$  basis.

For completeness, we also simulate the single-shot performance of the  $4 - 2 - 2$  tricycle codes explored in the main text. We used the same BP+OSD decoding setup described above. The results can be found in Fig. 7, which demonstrate impressive single-shot error correction suppression for this

class of codes. In practice, however, we anticipate that the overall magic state factory performance will be limited by the  $d$ -rounds  $X$ -basis simulated in the main text, owing to its higher check weight and lower distance compared to the  $Z$  basis.

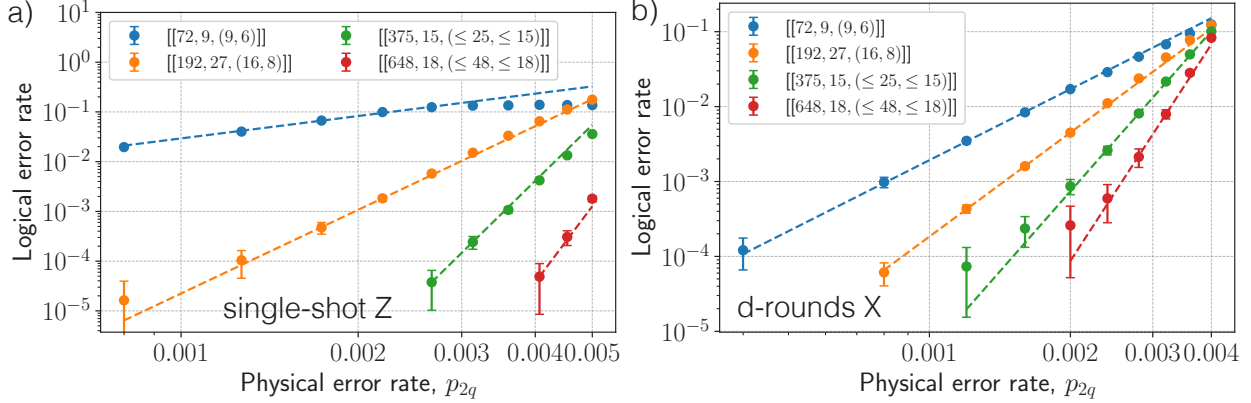


Figure 6: **Circuit-level noise simulation results for 4 – 4 – 4 tricycle codes.** (a) Logical error rate (as measured by block failure probability) versus two-qubit physical gate error rate ( $p_{2q}$ ) for single-shot error correction in the  $Z$  basis. Single-shot performance is evaluated using a windowed decoding protocol: three rounds of syndrome extraction followed by decoding and correction, with the window repeated 14 times (for 42 total rounds) to probe sustainable suppression of logical errors. (b) Logical error rate versus  $p_{2q}$  for fault-tolerant,  $d$ -round error correction in the  $X$  basis. In both panels, errors are sampled under a standard two-qubit depolarizing circuit-level noise model, and results are shown for various tricycle code. Logical error rates are determined via Monte Carlo simulations using a BP+OSD decoder, with each data point corresponding to  $M$  samples; error bars indicate standard errors, computed as  $\sqrt{p_L(1 - p_L)/M}$ .

| $[[N, K, (D_X, D_Z)]]$        | $p_L^{(X)}(0.1\%)$ | $p_L^{(Z)}(0.1\%)$  | $p_L^{(X)}(0.01\%)$ | $p_L^{(Z)}(0.01\%)$ |
|-------------------------------|--------------------|---------------------|---------------------|---------------------|
| $[[72, 9, (9, 6)]]$           | $2 \times 10^{-3}$ | $3 \times 10^{-2}$  | $1 \times 10^{-6}$  | $9 \times 10^{-4}$  |
| $[[192, 27, (16, 8)]]$        | $2 \times 10^{-4}$ | $2 \times 10^{-5}$  | $4 \times 10^{-9}$  | $6 \times 10^{-11}$ |
| $[[375, 15, (<= 25, <= 15)]]$ | $5 \times 10^{-6}$ | $4 \times 10^{-10}$ | $4 \times 10^{-13}$ | $1 \times 10^{-21}$ |
| $[[648, 18, (<= 48, <= 18)]]$ | $1 \times 10^{-7}$ | $1 \times 10^{-14}$ | $3 \times 10^{-17}$ | $2 \times 10^{-30}$ |

Table 3: **Example 4 – 4 – 4 tricycle codes and their logical error rates under circuit-level noise.** Logical error rates  $p_L^{(X)}$  and  $p_L^{(Z)}$  are shown for both  $p_{2q} = 10^{-3}$  and  $p_{2q} = 10^{-4}$ , for different  $[[N, K, D]]$  code parameters.

## H Optimal-depth syndrome extraction circuits

We provide a constructive proof of the following theorem.

**Theorem H.1.** *For tricycle codes defined by  $\mathbf{A} = \sum_{i=1}^{w_a} \mathbf{A}_i$ ,  $\mathbf{B} = \sum_{i=1}^{w_b} \mathbf{B}_i$ , and  $\mathbf{C} = \sum_{i=1}^{w_c} \mathbf{C}_i$  where  $w_a$ ,  $w_b$ , and  $w_c$  are even, there exists syndrome extraction circuits with CNOT depth  $w_a + w_b + w_c$ .*

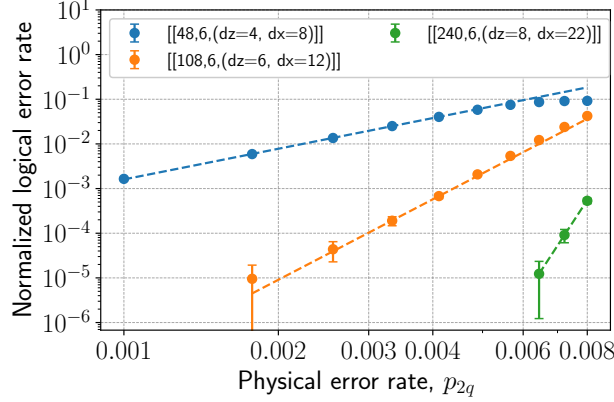


Figure 7: **Single-shot simulation results for 4 – 2 – 2 tricycle codes.** Logical error rate normalized by number of QEC rounds and number of logical qubits versus two-qubit physical gate error rate ( $p_{2q}$ ) for single-shot error correction in the Z basis. Logical error rates are determined via Monte Carlo simulations using a BP+OSD decoder, with each data point corresponding to  $M$  samples; error bars indicate standard errors, computed as  $\sqrt{p_L(1-p_L)/M}$ .

*Proof.* Let  $X \rightarrow \mathbf{A}_i^T(D_1)$  denote CNOTs from check sector  $X$  to data sector  $D_1$  according to permutation  $\mathbf{A}_i^T$ . The arrow indicates the CNOT direction: from each qubit  $j$  in  $X$  to qubit  $k$  in  $D_1$  whenever the  $(j, k)$  entry of  $\mathbf{A}_i^T$  is 1. Since  $\mathbf{A}_i^T$  are permutation matrices, the CNOTs are disjoint and can be executed in parallel.

We schedule all CNOTs in five stages, with  $w_a + w_b + w_c$  layers in total (square brackets indicate groups of CNOTs in the same layer):

- (1) for  $1 \leq i \leq w_c/2$ , apply  $[\mathbf{C}_{i+w_c/2}(D_1) \rightarrow Z_1, \mathbf{C}_{i+w_c/2}(D_2) \rightarrow Z_2, X \rightarrow \mathbf{C}_i^T(D_3)]$ ;
- (2) for  $1 \leq i \leq w_b/2$ , apply  $[\mathbf{B}_{i+w_b/2}(D_1) \rightarrow Z_3, X \rightarrow \mathbf{B}_i^T(D_2), \mathbf{B}_{i+w_b/2}(D_3) \rightarrow Z_2]$ ;
- (3) for  $1 \leq i \leq w_a$ , apply  $[X \rightarrow \mathbf{A}_i^T(D_1), \mathbf{A}_i(D_2) \rightarrow Z_3, \mathbf{A}_i(D_3) \rightarrow Z_1]$ ;
- (4) for  $1 \leq i \leq w_b/2$ , apply  $[\mathbf{B}_i(D_1) \rightarrow Z_3, X \rightarrow \mathbf{B}_{i+w_b/2}^T(D_2), \mathbf{B}_i(D_3) \rightarrow Z_2]$ ;
- (5) for  $1 \leq i \leq w_c/2$ , apply  $[\mathbf{C}_i(D_1) \rightarrow Z_1, \mathbf{C}_i(D_2) \rightarrow Z_2, X \rightarrow \mathbf{C}_{i+w_c/2}^T(D_3)]$ .

It remains to verify that this schedule measures the stabilizers while preserving logical operators. The CNOTs' effects can be tracked using a stabilizer table; since CNOTs do not mix X and Z parts, the action on each part can be analyzed separately. We focus on the X-part; the Z-part is similar.

Consider the stabilizer table extended with an arbitrary logical Pauli X, supported on 1-entries of row vectors  $u$ ,  $v$ , and  $w$ . The circuit is correct if the CNOTs yield the expected outcome:

$$\left[ \begin{array}{c|ccc|ccc} X & D_1 & D_2 & D_3 & Z_1 & Z_2 & Z_3 \\ \hline \mathbf{I} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \mathbf{A}^T & \mathbf{B}^T & \mathbf{C}^T & 0 & 0 & 0 \\ 0 & u & v & w & 0 & 0 & 0 \end{array} \right] \rightarrow \left[ \begin{array}{c|ccc|ccc} X & D_1 & D_2 & D_3 & Z_1 & Z_2 & Z_3 \\ \hline \mathbf{I} & \mathbf{A}^T & \mathbf{B}^T & \mathbf{C}^T & 0 & 0 & 0 \\ \mathbf{I} & \mathbf{A}^T & \mathbf{B}^T & \mathbf{C}^T & 0 & 0 & 0 \\ 0 & u & v & w & 0 & 0 & 0 \end{array} \right] \quad (70)$$

At stage 1, the CNOTs from  $X$  to  $D_3$  add rows from column  $X$  to those in column  $D_3$ , permuted by each  $\mathbf{C}_i^T$ . Let  $\mathbf{C}_{<}^T := \sum_{i=1}^{w_c/2} \mathbf{C}_i^T$  and  $\mathbf{C}_{>}^T := \sum_{i=1+w_c/2}^{w_c} \mathbf{C}_i^T$ ; these CNOTs correspond to right-

multiplying column  $X$  by  $\mathbf{C}_<^T$  and accumulating into  $D_3$ . The other two CNOT groups propagate from data sectors to Z-checks. As each  $\mathbf{C}_i$  is a permutation matrix,  $\mathbf{C}_i(D_1) \rightarrow Z_1$  is equivalent to  $D_1 \rightarrow \mathbf{C}_i^T(Z_1)$ ; we use this equivalence below. Thus, the CNOT action right-multiplies  $D_1$  and  $D_2$  by  $\mathbf{C}_>^T$  and accumulates the results to  $Z_1$  and  $Z_2$ , respectively. After stage 1, the table becomes

$$\left[ \begin{array}{c|ccc} \mathbf{I} & 0 & 0 & \mathbf{C}_<^T \\ 0 & \mathbf{A}^T & \mathbf{B}^T & \mathbf{C}_<^T \\ 0 & u & v & w \end{array} \middle| \begin{array}{ccc} 0 & 0 & 0 \\ \mathbf{A}^T \mathbf{C}_>^T & \mathbf{B}^T \mathbf{C}_>^T & 0 \\ u \mathbf{C}_>^T & v \mathbf{C}_>^T & 0 \end{array} \right].$$

After stage 2, the table becomes

$$\left[ \begin{array}{c|ccc} \mathbf{I} & 0 & \mathbf{B}_<^T & \mathbf{C}_<^T \\ 0 & \mathbf{A}^T & \mathbf{B}_<^T & \mathbf{C}_<^T \\ 0 & u & v & w \end{array} \middle| \begin{array}{ccc} 0 & \mathbf{A}^T \mathbf{C}_>^T & \mathbf{B}^T \mathbf{C}_>^T + \mathbf{C}_<^T \mathbf{B}_>^T = \mathbf{B}_>^T \mathbf{C}_<^T \\ u \mathbf{C}_>^T & v \mathbf{C}_>^T + w \mathbf{B}_>^T & \mathbf{B}_>^T \mathbf{C}_>^T + \mathbf{B}_>^T \mathbf{C}_<^T = \mathbf{B}_>^T \mathbf{C}_>^T + \mathbf{B}_>^T \mathbf{C}_<^T \\ & & \mathbf{A}^T \mathbf{B}_>^T + \mathbf{A}^T \mathbf{B}_<^T = 0 \\ & & u \mathbf{B}_>^T + v \mathbf{A}^T \end{array} \right],$$

where the equalities hold since all polynomial terms in  $\mathbf{A}$ ,  $\mathbf{B}$ , and  $\mathbf{C}$  commute. We will use this property throughout and consistently write products of terms in dictionary order.

After stage 3, the table becomes

$$\left[ \begin{array}{c|ccc} \mathbf{I} & \mathbf{A}^T & \mathbf{B}_<^T & \mathbf{C}_<^T \\ 0 & \mathbf{A}^T & \mathbf{B}_<^T & \mathbf{C}_<^T \\ 0 & u & v & w \end{array} \middle| \begin{array}{ccc} \mathbf{A}^T \mathbf{C}_<^T & \mathbf{B}_>^T \mathbf{C}_<^T & \mathbf{A}^T \mathbf{B}_<^T \\ \mathbf{A}^T \mathbf{C}_>^T + \mathbf{A}^T \mathbf{C}_<^T & \mathbf{B}^T \mathbf{C}_>^T + \mathbf{B}_>^T \mathbf{C}_<^T & \mathbf{A}^T \mathbf{B}_>^T + \mathbf{A}^T \mathbf{B}_<^T = 0 \\ u \mathbf{C}_>^T + w \mathbf{A}^T & v \mathbf{C}_>^T + w \mathbf{B}_>^T & u \mathbf{B}_>^T + v \mathbf{A}^T \end{array} \right].$$

After stage 4, the table becomes

$$\left[ \begin{array}{c|ccc} \mathbf{I} & \mathbf{A}^T & \mathbf{B}^T & \mathbf{C}_<^T \\ 0 & \mathbf{A}^T & \mathbf{B}^T & \mathbf{C}_<^T \\ 0 & u & v & w \end{array} \middle| \begin{array}{ccc} \mathbf{A}^T \mathbf{C}_<^T & \mathbf{B}^T \mathbf{C}_<^T & \mathbf{A}^T \mathbf{B}_<^T + \mathbf{A}^T \mathbf{B}_>^T = 0 \\ \mathbf{A}^T \mathbf{C}_>^T + \mathbf{A}^T \mathbf{C}_<^T & \mathbf{B}^T \mathbf{C}_>^T + \mathbf{B}_>^T \mathbf{C}_<^T & \mathbf{A}^T \mathbf{B}_>^T + \mathbf{A}^T \mathbf{B}_<^T = 0 \\ u \mathbf{C}_>^T + w \mathbf{A}^T & v \mathbf{C}_>^T + w \mathbf{B}_>^T & u \mathbf{B}_>^T + v \mathbf{A}^T \end{array} \right],$$

where the equalities hold since the addition is modulo 2.

Finally, after stage 5, the table becomes

$$\left[ \begin{array}{c|ccc} \mathbf{I} & \mathbf{A}^T & \mathbf{B}^T & \mathbf{C}_<^T \\ 0 & \mathbf{A}^T & \mathbf{B}^T & \mathbf{C}_<^T \\ 0 & u & v & w \end{array} \middle| \begin{array}{ccc} 0 & 0 & 0 \\ 0 & 0 & 0 \\ u \mathbf{C}_>^T + w \mathbf{A}^T = 0 & v \mathbf{C}_>^T + w \mathbf{B}_>^T = 0 & u \mathbf{B}_>^T + v \mathbf{A}^T = 0 \end{array} \right],$$

where the equalities hold since any logical Pauli X commutes with Z-checks, i.e.,  $[u \ v \ w] H_z^T = 0$ .  $\square$

The construction allows certain degrees of freedom. First, the assignment of the matrices  $\mathbf{A}$ ,  $\mathbf{B}$ , and  $\mathbf{C}$  to stages 1+5, 2+4, and 3 can be chosen arbitrarily. Second, the ordering of layers within each stage is flexible, provided that stages 1 and 5 contain complementary terms, as do stages 2 and 4. Let  $w_1$ ,  $w_2$ , and  $w_3$  denote the weights of polynomials assigned to stages 1+5, 2+4, and 3, respectively. (In the proof above,  $w_1 = w_c$ ,  $w_2 = w_b$ , and  $w_3 = w_a$ .) Then, the layers within stage 3 can be ordered  $w_3!$  ways. The order of terms in stages 1 and 5 for the X-checks can be arbitrary, resulting in  $w_1!$  possibilities. Since stages 1 and 5 must be complementary for the Z-checks, the assignment of terms to stages 1 and 5 is fixed, but permutations within each stage remain possible,

contributing an additional  $((w_1/2)!)^2$  arrangements. Similarly, for stages 2 and 4, the same counting applies with weight  $w_2$ . Thus, the total number of distinct circuit configurations is

$$w_1! \times ((w_1/2)!)^2 \times w_2! \times ((w_2/2)!)^2 \times w_3!. \quad (71)$$

For example, when  $w_1 = w_2 = 2$  and  $w_3 = 4$ , there are 96 circuits, while for  $w_1 = w_3 = 2$  and  $w_2 = 4$ , there are 384 circuits. In addition, the assignments of **A**, **B**, and **C** to the stages 1+5, 2+4, and 3 themselves can be permuted. For the 4-2-2 codes, this yields: two assignments with  $w_1 = w_2 = 2, w_3 = 4$ , two assignments with  $w_1 = w_3 = 2, w_2 = 4$ , and two assignments with  $w_2 = w_3 = 2, w_1 = 4$ . Altogether, this gives  $96 \times 2 + 384 \times 4 = 1,728$  different circuits for the 4-2-2 codes. A similar calculation for the 4-4-2 and 4-4-4 codes results in 55,296 and 1,327,104 distinct circuits, respectively.

In the counting above, same polynomial terms are scheduled together if possible, to enable more efficient atom rearrangement (more details in Sec. I). For example, in stage 3, **A**<sub>1</sub> from  $X$  to  $D_1$ , from  $D_2$  to  $Z_3$ , and from  $D_3$  to  $Z_1$  are always in the same layer, but they do not have to be. If we consider such possibilities, the number of different choices for stage 3 is no longer  $w_3!$  but  $(w_3!)^3$ . For a specific choice of the weights, the number of different circuits becomes

$$w_1! \times ((w_1/2)!)^4 \times w_2! \times ((w_2/2)!)^4 \times (w_3!)^3. \quad (72)$$

For 4-2-2 codes, this amounts to 135,168 circuits.

Theorem H.1 applies to all codes presented in the main text. For other cases, let  $W = w_a + w_b + w_c$  and  $V = |\{w \in \{w_a, w_b, w_c\} \mid w \text{ is even}\}|$ . Our scheduling method produces the following result, which is at most two CNOT layers from optimal.

**Corollary H.1.1.** *For tricycle codes with  $V = 0, 1, 2$ , and 3, there exists syndrome extraction circuits with CNOT depth  $W, W, W + 1$ , and  $W + 2$ , respectively.*

*Proof.* Theorem H.1 covers  $V = 0$ . For  $V = 1$ , assign the odd-weight polynomial as **A**, and the previous proof still applies. For  $V = 2$ , assign the odd-weight polynomials as **A** and **C**. Modifying stages 1 and 5 of in Eq. (69) as follows suffices:

$$\begin{aligned} (1) & \text{ apply } X \rightarrow \mathbf{C}_1^T(D_3); \\ & \text{for } 1 \leq i \leq \lfloor w_c/2 \rfloor, \text{ apply } [\mathbf{C}_{i+\lfloor w_c/2 \rfloor}(D_1) \rightarrow Z_1, \mathbf{C}_{i+\lfloor w_c/2 \rfloor}(D_2) \rightarrow Z_2, X \rightarrow \mathbf{C}_{1+i}^T(D_3)]; \\ (5) & \text{ for } 1 \leq i \leq \lfloor w_c/2 \rfloor, \text{ apply } [\mathbf{C}_i(D_1) \rightarrow Z_1, \mathbf{C}_i(D_2) \rightarrow Z_2, X \rightarrow \mathbf{C}_{1+i+\lfloor w_c/2 \rfloor}^T(D_3)]; \\ & \text{apply } [\mathbf{C}_{w_c}(D_1) \rightarrow Z_1, \mathbf{C}_{w_c}(D_2) \rightarrow Z_2]. \end{aligned} \quad (73)$$

Thus, stages 1 and 5 now take  $w_c + 1$  layers. For  $V = 3$ , modify stages 2 and 4 in a similar way.  $\square$

Following Ref. [15], counting both initialization and measurement of check qubits as unit depth, our construction gives the following results.

**Corollary H.1.2.** *For tricycle codes with  $V = 0, 1, 2$ , and 3, there exists  $M$ -cycle syndrome extraction circuits with depth  $(W+2)M, (W+2)M, (W+2)M + 1$ , and  $(W+3)M + 1$ , respectively.*

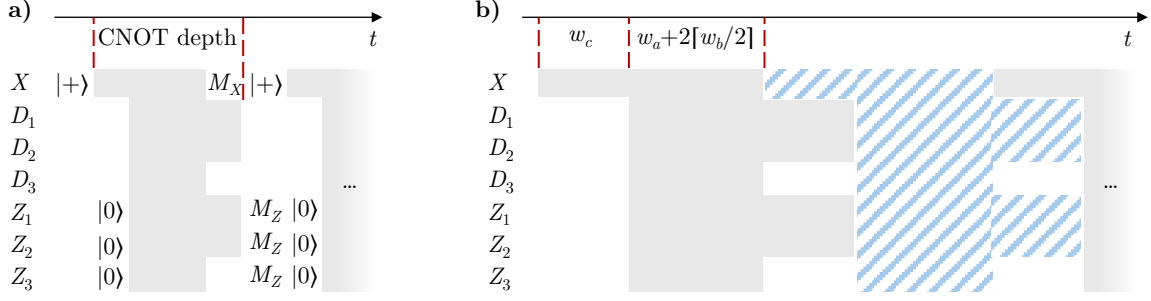


Figure 8: **Syndrome extraction circuit for multiple cycles:** (a) When  $w_c$  is odd, X-check measurements can overlap with the last CNOT layer, and Z-check initialization can overlap with the first CNOT layer. (b) Staggered scheduling with separate check qubits for even and odd cycles. Solid and hatched patterns indicate CNOTs for even and odd cycles, respectively. Measurement and initialization for one cycle can overlap with the CNOTs of the other cycle.

*Proof.* For  $V = 0$  and  $V = 1$ , initialization precedes and measurement follows the CNOT layers each cycle, adding two units per cycle. For  $V = 2$  and  $V = 3$ , as illustrated in Fig. 8a, X-check measurement can be parallelized with the final CNOT layer, and Z-check initialization with the first layer in Eq. (73), resulting in the stated depths.  $\square$

With two sets of check qubits, we use one set for odd cycles and the other set for even cycles.

**Theorem H.2.** *Given two sets of check qubits, for tricycle codes with  $V = 0, 1, 2$ , and  $3$ , there exists  $M$ -cycle syndrome extraction circuits with CNOT depth  $WM + w_c, WM + w_c, WM + w_c$ , and  $(W + 1)M + w_c$ , respectively.*

*Proof.* Adapt stages 1 and 5 in Eq. (69) as follows:

$$\begin{aligned} (1) & \text{ for } 1 \leq i \leq w_c, \text{ apply } [\mathbf{C}_i(D_1) \rightarrow Z'_1, \mathbf{C}_i(D_2) \rightarrow Z'_2, X \rightarrow \mathbf{C}_i^T(D_3)]; \\ (5) & \text{ for } 1 \leq i \leq w_c, \text{ apply } [\mathbf{C}_i(D_1) \rightarrow Z_1, \mathbf{C}_i(D_2) \rightarrow Z_2, X' \rightarrow \mathbf{C}_i^T(D_3)]; \end{aligned} \quad (74)$$

where the prime indicates check qubits for the other cycle. This yields a staggered circuit (Fig. 8b) similar to the construction for hypergraph product codes in Ref. [107]. If at least one polynomial has even weight, assign it as  $\mathbf{B}$ ; if  $V = 3$ , revise stages 2 and 4 similarly to Eq. (73).  $\square$

The above results naturally adapt to bicycle codes. For example, when  $w_a = w_b = 3$ , the circuits for bivariate bicycle codes in Ref. [15] are reproduced.

## I Neutral atom array architecture and rearrangement procedure

To concretely specify the atom rearrangement procedure, we consider the reference architecture shown in Fig. 9, primarily based on settings from recent experiments [6, 7, 25]. The neutral atom array features multiple zones for different functions: an entangling zone for CNOT operations, a storage zone for densely packed atoms, a measurement zone for imaging, and a reservoir for fresh atoms. For our purposes, we focus on the entangling and storage zones.

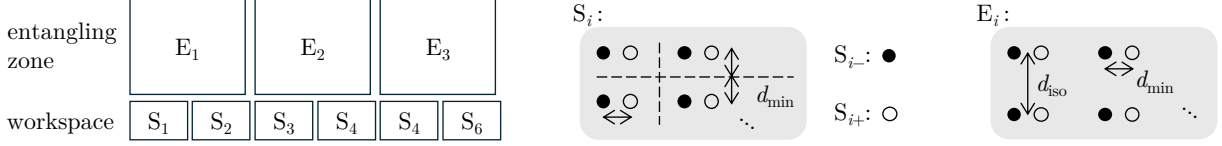


Figure 9: **Reference neutral atom array architecture.** Rydberg interactions are enabled only within the entangling zone. We slice the zones to regions  $E_i$  and  $S_i$ , each one can store a sector. Each region contains two trap arrays (dots/– and circles/+) to facilitate sector permutation or parallel CNOTs. In the workspace, traps are spaced by twice the minimal distance permitted by optical resolution,  $d_{\min}$ , allowing qubits to move between traps along dashed paths. In the entangling zone, traps are spaced so that qubit pairs involved in parallel CNOTs are separated by a distance to sufficiently isolate the Rydberg interaction,  $d_{\text{iso}}$ .

SLM traps are present throughout these zones, and atoms can be rearranged by AOD between SLM grids, subject to AOD order constraints. Parts of the storage zone adjacent to the entangling zone serve as workspaces, where sector permutations occur before each CNOT layer. We further divide the workspace into regions  $S_i$ , each supporting permutation of a sector, and each containing two SLM trap arrays:  $S_{i-}$  (dots) and  $S_{i+}$  (circles), each shaped like a sector (e.g.,  $n_y$  rows by  $n_x n_z$  columns as in Fig. 5b). These arrays are just labels; physically, all SLM traps are generated together.

For example, a  $y$ -cycling permutation of  $1 \leq m < n_y$  units for a sector in  $S_{i-}$  requires two AOD movements: shifting rows  $1, \dots, n_y - m$  in  $S_{i-}$  down to rows  $m + 1, \dots, n_y$  in  $S_{i+}$ ; and the wrap-around, shifting rows  $n_y - m + 1, \dots, n_y$  up to rows  $1, \dots, m$  in  $S_{i+}$ . After cycling, the sector is entirely in  $S_{i+}$ . Minimizing the SLM trap spacing in  $S_i$  enables faster permutation, which is limited by the optical resolving distance,  $d_{\min}$ . The vertical spacing of  $S_{i+}$  is set to  $2d_{\min}$ , allowing qubits to traverse the region along the dashes; and the horizontal spacing is  $3d_{\min}$  to additionally ensure sufficient separation between  $S_{i-}$  and  $S_{i+}$ .

Each entangling region  $E_i$  contains two arrays,  $E_{i-}$  and  $E_{i+}$ . To perform parallel CNOTs, one sector is positioned in  $E_{i-}$ , the other in  $E_{i+}$ , and a global Rydberg laser pulse is applied. We refer to the AOD movement from workspace to the entangling zone as fetching, and the movement back as put-back. To ensure that only intended pairs interact, the trap spacing in  $E_i$  is set to at least  $d_{\text{iso}}$ , a distance to sufficiently isolate Rydberg interactions. For example, with  $d_{\min} = 2 \mu\text{m}$  and  $d_{\text{iso}} = 10 \mu\text{m}$  [6], each  $E_i$  is about twice as wide as each  $S_j$ .

We now explicitly describe the atom rearrangement procedure outlined in Sec. 2.5, based on the reference architecture. For initialization, place  $D_1$  in  $E_1$ ,  $D_2$  in  $E_2$ ,  $D_3$  in  $E_3$ ,  $Z_3$  in  $S_2$ ,  $Z_1$  in  $S_3$ ,  $Z_2$  in  $S_4$ , and  $X$  in  $S_5$ . To implement the first layer, perform:

- Permute  $X$  (relabeling):  $\mathbf{C}_1 \mathbf{1}(S_5)$ ;
- Permute  $Z$  (relabeling):  $\mathbf{C}_3^T \mathbf{1}[S_2, S_3, S_4]$ ;
- Fetch  $X$ ,  $Z_1$  and  $Z_2$ :  $[S_5 \rightarrow E_3, S_3 \rightarrow E_1, S_4 \rightarrow E_2]$ ;
- CNOTs:  $[\mathbf{C}_3(D_1) \rightarrow Z_1, \mathbf{C}_3(D_2) \rightarrow Z_2, X \rightarrow \mathbf{C}_1^T(D_3)]$ ;
- Put back  $X$ ,  $Z_1$  and  $Z_2$ :  $[E_3 \rightarrow S_5, E_1 \rightarrow S_3, E_2 \rightarrow S_4]$ .

Here, ‘permute  $Z$ ’ refers to simultaneous permutation of all three  $Z$  sectors. Permutations are denoted as  $\sigma'\sigma^T(S_i)$ , where  $\sigma$  is the current permutation and  $\sigma'$  the target, within region  $S_i$ . Specifically for this first layer, since the  $X$  and  $Z$  sectors are still fresh, we can relabel their qubits instead of performing physical permutations. Fetching and put-back notation indicate initial and final regions for AOD movements, potentially involving sequential moves and grid changes. We do not specify which array within a region is used, as either suffices. CNOT notation follows previous sections, and parallelized movements are listed in square brackets. Notably, fetching and put-back for  $X$  and  $Z$  sectors can be combined for this layer.

The full implementation of the syndrome extraction circuit in Eq. (69) proceeds similarly. For convenience, the permutation notation is slightly abused: when  $i = 1$ , in expressions like  $\mathbf{C}_{i+w_c/2}^T \mathbf{C}_{i-1+w_c/2}[S_2, S_3, S_4]$ , the second matrix refers to the prior permutation on those sectors (e.g., identity for stage (1)), not literally  $\mathbf{C}_{w_c/2}$ . The accompanying video demonstrates an instance with  $n_x = n_y = n_z = 3$  and  $\mathbf{A} = yz^2 + x^2z + x^2y + x^2y^2z$ ,  $\mathbf{B} = yz^2 + xy + x^2y + x^2y^2z$ , and  $\mathbf{C} = 1 + xz^2 + x^2y^2 + x^2y^2z$ .

- (1) For  $1 \leq i \leq w_c/2$ , permute  $X$ :  $\mathbf{C}_i \mathbf{C}_{i-1}^T(S_5)$ ; permute  $Z$ :  $\mathbf{C}_{i+w_c/2}^T \mathbf{C}_{i-1+w_c/2}[S_2, S_3, S_4]$ ; fetch  $Z_1, Z_2$ , and  $X$ :  $[S_3 \rightarrow E_1, S_4 \rightarrow E_2, S_5 \rightarrow E_3]$ ; CNOTs:  $[\mathbf{C}_{i+w_c/2}(D_1) \rightarrow Z_1, \mathbf{C}_{i+w_c/2}(D_2) \rightarrow Z_2, X \rightarrow \mathbf{C}_i^T(D_3)]$ ; put back  $Z_1, Z_2$ , and  $X$ :  $[E_1 \rightarrow S_3, E_2 \rightarrow S_4, E_3 \rightarrow S_5]$ .
- (2) For  $1 \leq i \leq w_b/2$ , permute  $X$ :  $\mathbf{B}_i \mathbf{B}_{i-1}^T(S_5)$ ; permute  $Z$ :  $\mathbf{B}_{i+w_b/2}^T \mathbf{B}_{i-1+w_b/2}[S_2, S_3, S_4]$ ; fetch  $X$ :  $S_5 \rightarrow E_2$ ; fetch  $Z_3$  and  $Z_2$ :  $[S_2 \rightarrow E_1, S_4 \rightarrow E_2]$ ; CNOTs:  $[\mathbf{B}_{i+w_b/2}(D_1) \rightarrow Z_3, X \rightarrow \mathbf{B}_i^T(D_2), \mathbf{B}_{i+w_b/2}(D_3) \rightarrow Z_2]$ ; put back  $X$ :  $E_2 \rightarrow S_5$ ; put back  $Z_1$  and  $Z_2$ :  $[E_1 \rightarrow S_2, E_2 \rightarrow S_4]$ .
- (3) For  $1 \leq i \leq w_a$ , permute  $X$ :  $\mathbf{A}_i \mathbf{A}_{i-1}^T(S_5)$ ; permute  $Z$ :  $\mathbf{A}_i^T \mathbf{A}_{i-1}[S_2, S_3, S_4]$ ; fetch  $X$ :  $S_5 \rightarrow E_1$ ; fetch  $Z_3$  and  $Z_1$ :  $[S_2 \rightarrow E_2, S_3 \rightarrow E_3]$ ; CNOTs:  $[X \rightarrow \mathbf{A}_i^T(D_1), \mathbf{A}_i(D_2) \rightarrow Z_3, \mathbf{A}_i(D_3) \rightarrow Z_1]$ ; put back  $X$ :  $E_1 \rightarrow S_5$ ; put back  $Z_3$  and  $Z_1$ :  $[E_2 \rightarrow S_2, E_3 \rightarrow S_3]$ .
- (4) For  $1 \leq i \leq w_b/2$ , permute  $X$ :  $\mathbf{B}_{i+w_b/2} \mathbf{B}_{i-1+w_b/2}^T(S_5)$ ; permute  $Z$ :  $\mathbf{B}_i^T \mathbf{B}_{i-1}[S_2, S_3, S_4]$ ; fetching and put-back are the same as (2).
- (5) For  $1 \leq i \leq w_c/2$ , permute  $X$ :  $\mathbf{C}_{i+w_c/2} \mathbf{C}_{i-1+w_c/2}^T(S_5)$ ; permute  $Z$ :  $\mathbf{C}_i^T \mathbf{C}_{i-1}[S_2, S_3, S_4]$ ; fetching and put-back are the same as (1).

For  $w_a = w_b = w_c = 4$ , the above procedure results in 22 sector permutations (with relabeling used in the first layer) and 40 fetch/put-back operations. While the scheme is general and structured, inefficiencies may arise from always restoring sectors to their original locations and exclusively permuting check sectors instead of data sectors. For example, in (3), since  $\mathbf{A}_i(X) \rightarrow D_1$  is equivalent to  $X \rightarrow \mathbf{A}_i^T(D_1)$ , permuting  $D_1$  instead of  $X$  may allow both  $Z$  sectors and  $D_1$  to be permuted in parallel by  $\mathbf{A}_i^T \mathbf{A}_{i-1}$ . However, it is the change in permutation,  $\sigma'\sigma^T$ , and not the polynomial term  $\sigma$  itself, that enables such parallelization. If two sectors share  $\sigma'$  but have different  $\sigma$ , their permutations still cannot be merged, so a more careful analysis is needed to assess the potential benefits.

There is also flexibility to reorder CNOT layers or combine different CNOT groups into parallel layers, which changes  $\sigma'\sigma^T$  between layers. This sometimes results in cancellations, reducing the need for certain  $x$ -,  $y$ -, or  $z$ -cycling steps, and can be formulated as a variant of the traveling salesman problem and be solved optimally for small instances.

It is also valuable to investigate ‘torus-like layout’ strategies, where sectors are interspersed not only during entangling gates but also during permutations, provided that there are polynomial terms corresponding to nearest-neighbor interactions [15, 103]. Further optimization of sector layout and movement may be possible by mapping the problem to mathematical programming [98, 97, 100, 93] or graph theory formulations [99, 64, 92].

## References

- [1] P. Aliferis, F. Brito, D. P. DiVincenzo, J. Preskill, M. Steffen, and B. M. Terhal. Fault-tolerant computing with biased-noise superconducting qubits: a case study. *New Journal of Physics*, 11(1):013061, 2009.
- [2] N. Alon and J. H. Kim. On the degree, size, and chromatic index of a uniform hypergraph. *Journal of combinatorial theory, Series A*, 77(1):165–170, 1997.
- [3] G. Baranes, M. Cain, J. P. Bonilla Ataides, D. Bluvstein, J. Sinclair, V. Vuletic, H. Zhou, and M. D. Lukin. Leveraging atom loss errors in fault tolerant quantum algorithms. *arXiv preprint arXiv:2502.20558*, 2025.
- [4] N. Baspin, L. Berent, and L. Z. Cohen. Fast surgery for quantum ldpc codes. *arXiv preprint arXiv:2510.04521*, 2025.
- [5] M. J. Bedalov, M. Blakely, P. D. Buttler, C. Carnahan, F. T. Chong, W. C. Chung, D. C. Cole, P. Goiporia, P. Gokhale, B. Heim, G. T. Hickman, E. B. Jones, R. A. Jones, P. Khalate, J.-S. Kim, K. W. Kuper, M. T. Lichtman, S. Lee, D. Mason, N. A. Neff-Mallon, T. W. Noel, V. Omole, A. G. Radnaev, R. Rines, M. Saffman, E. Shabtai, M. H. Teo, B. Thotakura, T. Tomesh, and A. K. Tucker. Fault-tolerant operation and materials science with neutral atom logical qubits. *arXiv preprint arXiv:2412.07670*, 2024.
- [6] D. Bluvstein, S. J. Evered, A. A. Geim, S. H. Li, H. Zhou, T. Manovitz, S. Ebadi, M. Cain, M. Kalinowski, D. Hangleiter, J. P. Bonilla Ataides, N. Maskara, I. Cong, X. Gao, P. Sales Rodriguez, T. Karolyshyn, G. Semeghini, M. J. Gullans, M. Greiner, V. Vuletić, and M. D. Lukin. Logical quantum processor based on reconfigurable atom arrays. *Nature*, 626(7997):58–65, Feb. 2024.
- [7] D. Bluvstein, A. A. Geim, S. H. Li, S. J. Evered, J. Ataides, G. Baranes, A. Gu, T. Manovitz, M. Xu, M. Kalinowski, et al. Architectural mechanisms of a universal fault-tolerant quantum computer. *arXiv preprint arXiv:2506.20661*, 2025.
- [8] D. Bluvstein, H. Levine, G. Semeghini, T. T. Wang, S. Ebadi, M. Kalinowski, A. Keesling, N. Maskara, H. Pichler, M. Greiner, et al. A quantum processor based on coherent transport of entangled atom arrays. *Nature*, 604(7906):451–456, 2022.
- [9] H. Bombín. Single-shot fault-tolerant quantum error correction. *Physical Review X*, 5(3):031043, 2015.
- [10] H. Bombin. 2d quantum computation with 3d topological codes. *arXiv preprint arXiv:1810.09571*, 2018.

- [11] H. Bombin and M.-A. Martin-Delgado. Topological computation without braiding. *Physical review letters*, 98(16):160502, 2007.
- [12] J. P. Bonilla Ataides, A. Gu, S. F. Yelin, and M. D. Lukin. Neural decoders for universal quantum algorithms. *arXiv preprint arXiv:2509.11370*, 2025.
- [13] J. P. Bonilla Ataides, D. K. Tuckett, S. D. Bartlett, S. T. Flammia, and B. J. Brown. The xzzx surface code. *Nature Communications*, 12(1), Apr. 2021.
- [14] J. P. Bonilla Ataides, H. Zhou, Q. Xu, G. Baranes, B. Li, M. D. Lukin, and L. Jiang. Constant-overhead fault-tolerant bell-pair distillation using high-rate codes. *arXiv preprint arXiv:2502.09542*, 2025.
- [15] S. Bravyi, A. W. Cross, J. M. Gambetta, D. Maslov, P. Rall, and T. J. Yoder. High-threshold and low-overhead fault-tolerant quantum memory. *Nature*, 627(8005):778–782, 2024.
- [16] S. Bravyi and J. Haah. Magic-state distillation with low overhead. *Physical Review A—Atomic, Molecular, and Optical Physics*, 86(5):052329, 2012.
- [17] S. Bravyi and A. Kitaev. Universal quantum computation with ideal clifford gates and noisy ancillas. *Physical Review A—Atomic, Molecular, and Optical Physics*, 71(2):022316, 2005.
- [18] N. P. Breuckmann, M. Davydova, J. N. Eberhardt, and N. Tantivasadakarn. Cups and gates i: Cohomology invariants and logical quantum operations. *arXiv preprint arXiv:2410.16250*, 2024.
- [19] N. P. Breuckmann and J. N. Eberhardt. Balanced product quantum codes. *IEEE Transactions on Information Theory*, 67(10):6653–6674, 2021.
- [20] B. J. Brown, N. H. Nickerson, and D. E. Browne. Fault-tolerant error correction with the gauge color code. *Nature Communications*, 7(1), July 2016.
- [21] A. R. Calderbank and P. W. Shor. Good quantum error-correcting codes exist. *Physical Review A*, 54(2):1098, 1996.
- [22] E. T. Campbell. A theory of single-shot error correction for adversarial noise. *Quantum Science and Technology*, 4(2):025006, 2019.
- [23] E. T. Campbell, H. Anwar, and D. E. Browne. Magic-state distillation in all prime dimensions using quantum reed-muller codes. *Physical Review X*, 2(4):041021, 2012.
- [24] J. Chen, Y. Yan, and Y. Zhou. Magic of quantum hypergraph states. *Quantum*, 8:1351, 2024.
- [25] N.-C. Chiu, E. C. Trapp, J. Guo, M. H. Abobeih, L. M. Stewart, S. Hollerith, P. Stroganov, M. Kalinowski, A. A. Geim, S. J. Evered, S. H. Li, L. M. Peters, D. Bluvstein, T. T. Wang, M. Greiner, V. Vuletić, and M. D. Lukin. Continuous operation of a coherent 3,000-qubit system. *arXiv preprint arXiv:2506.20660*, 2025.
- [26] M. Christandl, F. Gesmundo, and J. Zuiddam. A gap in the subrank of tensors. *SIAM Journal on Applied Algebra and Geometry*, 7(4):742–767, 2023.

- [27] L. Z. Cohen, I. H. Kim, S. D. Bartlett, and B. J. Brown. Low-overhead fault-tolerant quantum computing using long-range connectivity. *Science Advances*, 8(20), 2022.
- [28] A. Cowtan, Z. He, D. J. Williamson, and T. J. Yoder. Fast and fault-tolerant logical measurements: Auxiliary hypergraphs and transversal surgery. *arXiv preprint arXiv:2510.14895*, 2025.
- [29] A. Cross, Z. He, P. Rall, and T. Yoder. Improved QLDPC Surgery: Logical Measurements and Bridging Codes. *arXiv preprint arXiv:2407.18393*, 2024.
- [30] A. Cross, Z. He, P. Rall, and T. Yoder. Linear-Size Ancilla Systems for Logical Measurements in QLDPC Codes. *arXiv preprint arXiv:2407.18393*, 2024.
- [31] L. Daguerre, R. Blume-Kohout, N. C. Brown, D. Hayes, and I. H. Kim. Experimental demonstration of high-fidelity logical magic states from code switching. *arXiv preprint arXiv:2506.14169*, 2025.
- [32] B. Eastin and E. Knill. Restrictions on transversal encoded quantum gate sets. *Physical review letters*, 102(11):110502, 2009.
- [33] J. N. Eberhardt and V. Steffan. Logical operators and fold-transversal gates of bivariate bicycle codes. *IEEE Transactions on Information Theory*, 2024.
- [34] S. J. Evered, D. Bluvstein, M. Kalinowski, S. Ebadi, T. Manovitz, H. Zhou, S. H. Li, A. A. Geim, T. T. Wang, N. Maskara, et al. High-fidelity parallel entangling gates on a neutral-atom quantum computer. *Nature*, 622(7982):268–272, 2023.
- [35] C. Gidney. Stim: a fast stabilizer circuit simulator. *Quantum*, 5:497, 2021.
- [36] C. Gidney. How to factor 2048 bit rsa integers with less than a million noisy qubits. *arXiv preprint arXiv:2505.15917*, 2025.
- [37] L. Golowich, K. Chang, and G. Zhu. Constant-overhead addressable gates via single-shot code switching. *arXiv preprint arXiv:2510.06760*, 2025.
- [38] L. Golowich and T.-C. Lin. Quantum ldpc codes with transversal non-clifford gates via products of algebraic codes. *arXiv preprint arXiv:2410.14662*, 2024.
- [39] D. Gottesman. Theory of fault-tolerant quantum computation. *Physical Review A*, 57(1):127, 1998.
- [40] D. Gottesman and I. L. Chuang. Demonstrating the viability of universal quantum computation using teleportation and single-qubit operations. *Nature*, 402(6760):390–393, 1999.
- [41] Gurobi Optimization, LLC. Gurobi Optimizer Reference Manual, 2024.
- [42] F. Gyger, M. Ammenwerth, R. Tao, H. Timme, S. Snigirev, I. Bloch, and J. Zeiher. Continuous operation of large-scale atom arrays in optical lattices. *Phys. Rev. Res.*, 6:033104, Jul 2024.
- [43] M. B. Hastings and J. Haah. Distillation with sublogarithmic overhead. *Physical review letters*, 120(5):050504, 2018.

- [44] A. Hatcher. *Algebraic topology*. Cambridge University Press, Cambridge, 2002.
- [45] S. Heußen and J. Hilder. Efficient fault-tolerant code switching via one-way transversal cnot gates. *arXiv preprint arXiv:2409.13465*, 2024.
- [46] T. Hillmann, L. Berent, A. O. Quintavalle, J. Eisert, R. Wille, and J. Roffe. Localized statistics decoding for quantum low-density parity-check codes. *Nature Communications*, 16(1), 2025.
- [47] Y. Hong. Single-shot preparation of hypergraph product codes via dimension jump. *arXiv preprint arXiv:2410.05171*, 2024.
- [48] W. C. Huffman, J.-L. Kim, and P. Solé. *Concise encyclopedia of coding theory*. Chapman and Hall/CRC, 2021.
- [49] B. Ide, M. G. Gowda, P. J. Nadkarni, and G. Dauphinais. Fault-tolerant logical measurements via homological measurement. *arXiv preprint arXiv:2410.02753*, 2024.
- [50] A. Ignatiev, A. Morgado, and J. Marques-Silva. PySAT: A Python toolkit for prototyping with SAT oracles. In *SAT*, pages 428–437, 2018.
- [51] A. Jacob, C. McLauchlan, and D. E. Browne. Single-shot decoding and fault-tolerant gates with trivariate tricycle codes. *arXiv preprint arXiv:2508.08191*, 2025.
- [52] S. Kopparty, G. Moshkovitz, and J. Zuiddam. Geometric rank of tensors and subrank of matrix multiplication. *arXiv preprint arXiv:2002.09472*, 2020.
- [53] A. A. Kovalev and L. P. Pryadko. Quantum kronecker sum-product low-density parity-check codes with finite rate. *Physical Review A—Atomic, Molecular, and Optical Physics*, 88(1):012311, 2013.
- [54] A. Krishna and J.-P. Tillich. Towards low overhead magic state distillation. *Physical review letters*, 123(7):070507, 2019.
- [55] A. Kubica, B. Yoshida, and F. Pastawski. Unfolding the color code. *New Journal of Physics*, 17(8):083026, 2015.
- [56] A. J. Landahl, J. T. Anderson, and P. R. Rice. Fault-tolerant quantum computing with color codes. *arXiv preprint arXiv:1108.5738*, 2011.
- [57] S.-H. Lee, L. English, and S. D. Bartlett. Efficient post-selection for general quantum ldpc codes. *arXiv preprint arXiv:2502.09542*, 2025.
- [58] J. S. Leon. A probabilistic algorithm for computing minimum weights of large error-correcting codes. *IEEE Transactions on Information Theory*, 34(5):1354–1359, 1988.
- [59] A. Leverrier and G. Zémor. Quantum tanner codes. In *2022 IEEE 63rd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 872–883. IEEE, 2022.
- [60] C. Li, J. Preskill, and Q. Xu. Transversal dimension jump for product qldpc codes. *arXiv preprint arXiv:2510.07269*, 2025.

- [61] H.-K. Lin, P. K. Lim, A. A. Kovalev, and L. P. Pryadko. Abelian multi-cycle codes for single-shot error correction. *arXiv preprint arXiv:2506.16910*, 2025.
- [62] H.-K. Lin and L. P. Pryadko. Quantum two-block group algebra codes. *Physical Review A*, 109(2):022407, 2024.
- [63] T.-C. Lin. Transversal non-clifford gates for quantum ldpc codes on sheaves. *arXiv preprint arXiv:2410.14631*, 2024.
- [64] W.-H. Lin, D. B. Tan, and J. Cong. Reuse-aware compilation for zoned quantum architectures based on neutral atoms. In *2025 IEEE International Symposium on High Performance Computer Architecture (HPCA)*, pages 127–142. IEEE, 2025.
- [65] S. Loor. From möbius strips to twisted toric codes.
- [66] S. Ma, G. Liu, P. Peng, B. Zhang, S. Jandura, J. Claes, A. P. Burgers, G. Pupillo, S. Puri, and J. D. Thompson. High-fidelity gates and mid-circuit erasure conversion in an atomic qubit. *Nature*, 622(7982):279–284, 2023.
- [67] A. S. Maan, F.-G. Herrero, A. Paler, and V. Savin. Decoding correlated errors in quantum ldpc codes. *arXiv preprint arXiv:2510.14060*, 2025.
- [68] T. Maurer, M. Bühler, M. Kröner, F. Haverkamp, T. Müller, D. Vandeth, and B. R. Johnson. Real-time decoding of the gross code memory with fpgas. *arXiv preprint arXiv:2510.21600*, 2025.
- [69] R. Mehta, J. P. Bonilla Ataides, G. Baranes, M. Cain, H. Zhou, L. Jiang, and M. D. Lukin. Dynamically generated logical gates. In preparation.
- [70] S. A. Moses, C. H. Baldwin, M. S. Allman, R. Ancona, L. Ascarrunz, C. Barnes, J. Bartolotta, B. Bjork, P. Blanchard, M. Bohn, J. G. Bohnet, N. C. Brown, N. Q. Burdick, W. C. Burton, S. L. Campbell, J. P. Campora, C. Carron, J. Chambers, J. W. Chan, Y. H. Chen, A. Chernoguzov, E. Chertkov, J. Colina, J. P. Curtis, R. Daniel, M. DeCross, D. Deen, C. Delaney, J. M. Dreiling, C. T. Ertsgaard, J. Esposito, B. Estey, M. Fabrikant, C. Figgatt, C. Foltz, M. Foss-Feig, D. Francois, J. P. Gaebler, T. M. Gatterman, C. N. Gilbreth, J. Giles, E. Glynn, A. Hall, A. M. Hankin, A. Hansen, D. Hayes, B. Higashi, I. M. Hoffman, B. Horning, J. J. Hout, R. Jacobs, J. Johansen, L. Jones, J. Karcz, T. Klein, P. Lauria, P. Lee, D. Liefer, S. T. Lu, D. Lucchetti, C. Lytle, A. Malm, M. Matheny, B. Mathewson, K. Mayer, D. B. Miller, M. Mills, B. Neyenhuis, L. Nugent, S. Olson, J. Parks, G. N. Price, Z. Price, M. Pugh, A. Ransford, A. P. Reed, C. Roman, M. Rowe, C. Ryan-Anderson, S. Sanders, J. Sedlacek, P. Shevchuk, P. Siegfried, T. Skripka, B. Spaun, R. T. Sprenkle, R. P. Stutz, M. Swallows, R. I. Tobey, A. Tran, T. Tran, E. Vogt, C. Volin, J. Walker, A. M. Zolot, and J. M. Pino. A race-track trapped-ion quantum processor. *Phys. Rev. X*, 13:041052, Dec 2023.
- [71] D. E. Muller. Application of boolean algebra to switching circuit design and to error detection. *Transactions of the IRE professional group on electronic computers*, (3):6–12, 1954.
- [72] T. Müller, T. Alexander, M. E. Beverland, M. Bühler, B. R. Johnson, T. Maurer, and D. Vandeth. Improved belief propagation is sufficient for real-time decoding of quantum memory. *arXiv preprint arXiv:2506.01779*, 2025.

- [73] Q. T. Nguyen. Good binary quantum codes with transversal ccz gate. doi: 10.48550. *arXiv preprint arXiv:2408.10140*, 2024.
- [74] D. Nigg, M. Mueller, E. A. Martinez, P. Schindler, M. Hennrich, T. Monz, M. A. Martin-Delgado, and R. Blatt. Quantum computations on a topologically encoded qubit. *Science*, 345(6194):302–305, 2014.
- [75] P. Obszarski and A. Jastrzebski. Edge-coloring of 3-uniform hypergraphs. *Discrete Applied Mathematics*, 217:48–52, 2017.
- [76] P. Panteleev and G. Kalachev. Degenerate quantum ldpc codes with good finite length performance. *Quantum*, 5:585, 2021.
- [77] P. Panteleev and G. Kalachev. Quantum ldpc codes with almost linear minimum distance. *IEEE Transactions on Information Theory*, 68(1):213–229, 2021.
- [78] P. Panteleev and G. Kalachev. Asymptotically good quantum and locally testable classical ldpc codes. In *Proceedings of the 54th annual ACM SIGACT symposium on theory of computing*, pages 375–388, 2022.
- [79] L. P. Pryadko, V. A. Shabashov, and V. K. Kozin. Qdistrnd: A gap package for computing the distance of quantum error-correcting codes. *arXiv preprint arXiv:2308.15140*, 2023.
- [80] I. S. Reed. A class of multiple-error-correcting codes and the decoding scheme. Technical report, 1953.
- [81] B. W. Reichardt, A. Paetznick, D. Aasen, I. Basov, J. M. Bello-Rivas, P. Bonderson, R. Chao, W. van Dam, M. B. Hastings, R. V. Mishmash, A. Paz, M. P. da Silva, A. Sundaram, K. M. Svore, A. Vashillo, Z. Wang, M. Zanner, W. B. Cairncross, C.-A. Chen, D. Crow, H. Kim, J. M. Kindem, J. King, M. McDonald, M. A. Norcia, A. Ryou, M. Stone, L. Wadleigh, K. Barnes, P. Battaglino, T. C. Bohdanowicz, G. Booth, A. Brown, M. O. Brown, K. Cassella, R. Coxe, J. M. Epstein, M. Feldkamp, C. Griger, E. Halperin, A. Heinz, F. Hummel, M. Jaffe, A. M. W. Jones, E. Kapit, K. Kotru, J. Lauigan, M. Li, J. Marjanovic, E. Megidish, M. Meredith, R. Morshead, J. A. Muniz, S. Narayanaswami, C. Nishiguchi, T. Paule, K. A. Pawlak, K. L. Pudenz, D. R. Pérez, J. Simon, A. Smull, D. Stack, M. Urbanek, R. J. M. van de Veerdonk, Z. Vendeiro, R. T. Weverka, T. Wilkason, T.-Y. Wu, X. Xie, E. Zalus-Geller, X. Zhang, and B. J. Bloom. Fault-tolerant quantum computation with a neutral atom processor. *arXiv preprint arXiv:2411.11822*, 2025.
- [82] J. Roffe. LDPC: Python tools for low density parity check codes, 2022.
- [83] J. Roffe, D. R. White, S. Burton, and E. Campbell. Decoding across the quantum low-density parity-check code landscape. *Phys. Rev. Res.*, 2:043423, Dec 2020.
- [84] P. Sales Rodriguez, J. M. Robinson, P. N. Jepsen, Z. He, C. Duckering, C. Zhao, K.-H. Wu, J. Campo, K. Bagnall, M. Kwon, et al. Experimental demonstration of logical magic state distillation. *Nature*, pages 1–3, 2025.
- [85] P. Scholl, M. Schuler, H. J. Williams, A. A. Eberharter, D. Barredo, K.-N. Schymik, V. Lienhard, L.-P. Henry, T. C. Lang, T. Lahaye, et al. Quantum simulation of 2d antiferromagnets with hundreds of rydberg atoms. *Nature*, 595(7866):233–238, 2021.

- [86] T. R. Scruby, D. E. Browne, P. Webster, and M. Vasmer. Numerical Implementation of Just-In-Time Decoding in Novel Lattice Slices Through the Three-Dimensional Surface Code. *Quantum*, 6:721, May 2022.
- [87] T. R. Scruby, K. Nemoto, and Z. Cai. Fault-tolerant quantum computation without distillation on a 2d device, 2025.
- [88] T. R. Scruby, A. Pesah, and M. Webster. Quantum rainbow codes. *arXiv preprint arXiv:2408.13130*, 2024.
- [89] P. W. Shor. Algorithms for quantum computation: discrete logarithms and factoring. In *Proceedings 35th annual symposium on foundations of computer science*, pages 124–134. Ieee, 1994.
- [90] M. D. Shulman, O. E. Dial, S. P. Harvey, H. Bluhm, V. Umansky, and A. Yacoby. Demonstration of entanglement of electrostatically coupled singlet-triplet qubits. *science*, 336(6078):202–205, 2012.
- [91] M. Sipser and D. A. Spielman. Expander codes. *IEEE transactions on Information Theory*, 42(6):1710–1722, 2002.
- [92] Y. Stade, W.-H. Lin, J. Cong, and R. Wille. Routing-aware placement for zoned neutral atom-based quantum computing. *arXiv preprint arXiv:2505.22715*, 2025.
- [93] Y. Stade, L. Schmid, L. Burgholzer, and R. Wille. Optimal state preparation for logical arrays on zoned neutral atom quantum computers. In *2025 Design, Automation and Test in Europe Conference (DATE)*, pages 1–7, 2025.
- [94] A. Steane. Multiple-particle interference and quantum error correction. *Proceedings of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences*, 452(1954):2551–2577, 1996.
- [95] M. Sullivan. Code conversion with the quantum golay code for a universal transversal gate set. *Physical Review A*, 109(4):042416, 2024.
- [96] E. Swaroop, T. Jochym-O’Connor, and T. J. Yoder. Universal adapters between quantum ldpc codes, 2024.
- [97] B. Tan, D. Bluvstein, M. D. Lukin, and J. Cong. Qubit mapping for reconfigurable atom arrays. In *Proceedings of the 41st IEEE/ACM International Conference on Computer-Aided Design*, pages 1–9, 2022.
- [98] D. B. Tan, D. Bluvstein, M. D. Lukin, and J. Cong. Compiling quantum circuits for dynamically field-programmable neutral atoms array processors. *Quantum*, 8:1281, 2024.
- [99] D. B. Tan, W.-H. Lin, and J. Cong. Compilation for dynamically field-programmable qubit arrays with efficient and provably near-optimal scheduling. In *Proceedings of the 30th Asia and South Pacific Design Automation Conference*, pages 921–929, 2025.

- [100] D. B. Tan, S. Ping, and J. Cong. Depth-optimal addressing of 2D qubit array with 1D controls based on exact binary matrix factorization. In *2024 Design, Automation and Test in Europe Conference and Exhibition (DATE) Proceedings*, 2024.
- [101] S. J. S. Tan, Y. Hong, T.-C. Lin, M. J. Gullans, and M.-H. Hsieh. Single-shot universality in quantum ldpc codes via code-switching. *arXiv preprint arXiv:2510.08552*, 2025.
- [102] J.-P. Tillich and G. Zémor. Quantum ldpc codes with positive rate and minimum distance proportional to the square root of the blocklength. *IEEE Transactions on Information Theory*, 60(2):1193–1202, 2013.
- [103] J. Vízslai, W. Yang, S. F. Lin, J. Liu, N. Nottingham, J. M. Baker, and F. T. Chong. Matching generalized-bicycle codes to neutral atoms for low-overhead fault-tolerance. *arXiv preprint arXiv:2311.16980*, 2024.
- [104] K. Wang, Z. Lu, C. Zhang, G. Liu, J. Chen, Y. Wang, Y. Wu, S. Xu, X. Zhu, F. Jin, Y. Gao, Z. Tan, Z. Cui, N. Wang, Y. Zou, A. Zhang, T. Li, F. Shen, J. Zhong, Z. Bao, Z. Zhu, Y. Han, Y. He, J. Shen, H. Wang, J.-N. Yang, Z. Song, J. Deng, H. Dong, Z.-Z. Sun, W. Li, Q. Ye, S. Jiang, Y. Ma, P.-X. Shen, P. Zhang, H. Li, Q. Guo, Z. Wang, C. Song, H. Wang, and D.-L. Deng. Demonstration of low-overhead quantum error correction codes. *arXiv preprint arXiv:2505.09684*, 2025.
- [105] D. J. Williamson and T. J. Yoder. Low-overhead fault-tolerant quantum computation by gauging logical operators. *arXiv preprint arXiv:2410.02213*, 2024.
- [106] A. Wills, M.-H. Hsieh, and H. Yamasaki. Constant-overhead magic state distillation. *arXiv preprint arXiv:2408.07764*, 2024.
- [107] Q. Xu, J. P. Bonilla Ataides, C. A. Pattison, N. Raveendran, D. Bluvstein, J. Wurtz, B. Vasić, M. D. Lukin, L. Jiang, and H. Zhou. Constant-overhead fault-tolerant quantum computation with reconfigurable atom arrays. *Nature Physics*, 20(7):1084–1090, 2024.
- [108] Q. Xu, H. Zhou, D. Bluvstein, M. Cain, M. Kalinowski, J. Preskill, M. D. Lukin, and N. Maskara. Batched high-rate logical operations for quantum ldpc codes, 2025.
- [109] Q. Xu, H. Zhou, G. Zheng, D. Bluvstein, J. P. Bonilla Ataides, M. D. Lukin, and L. Jiang. Fast and parallelizable logical computation with homological product codes. *Physical Review X*, 15(2):021065, 2025.
- [110] T. J. Yoder, E. Schoute, P. Rall, E. Pritchett, J. M. Gambetta, A. W. Cross, M. Carroll, and M. E. Beverland. Tour de gross: A modular quantum computer based on bivariate bicycle codes. *arXiv preprint arXiv:2506.03094*, 2025.
- [111] W. Zeng and L. P. Pryadko. Higher-dimensional quantum hypergraph-product codes with finite rates. *Physical review letters*, 122(23):230501, 2019.
- [112] G. Zheng, L. Jiang, and Q. Xu. High-rate surgery: towards constant-overhead logical operations. *arXiv preprint arXiv:2510.08523*, 2025.

- [113] H. Zhou, C. Duckering, C. Zhao, D. Bluvstein, M. Cain, A. Kubica, S.-T. Wang, and M. D. Lukin. Resource analysis of low-overhead transversal architectures for reconfigurable atom arrays. In *Proceedings of the 52nd Annual International Symposium on Computer Architecture*, pages 1432–1448, 2025.
- [114] G. Zhu. A topological theory for qldpc: non-clifford gates and magic state fountain on homological product codes with constant rate and beyond the  $n^{1/3}$  distance barrier. *arXiv preprint arXiv:2501.19375*, 2025.
- [115] G. Zhu. Transversal non-clifford gates on qldpc codes breaking the  $\sqrt{N}$  distance barrier and quantum-inspired geometry with  $\mathbb{Z}_2$  systolic freedom. *arXiv preprint arXiv:2507.15056*, 2025.

## Supplemental Material

### Review of homological formulation of CSS Codes

#### Background on cohomology

A *cochain complex*  $(C^\bullet, d^\bullet)$  is a sequence of vector spaces over a field, or more generally modules over a ring, equipped with linear maps

$$\dots \xrightarrow{d^{i-2}} C^{i-1} \xrightarrow{d^{i-1}} C^i \xrightarrow{d^i} C^{i+1} \xrightarrow{d^{i+1}} \dots$$

such that

$$d^i \circ d^{i-1} = 0 \quad \text{for all } i. \quad (75)$$

This condition ensures that the image of one map is always contained in the kernel of the next:

$$\text{im}(d^{i-1}) \subseteq \ker(d^i). \quad (76)$$

Elements of  $C^i$  are called *i-cochains*. Elements of  $\ker(d^i) \subseteq C^i$  are called *i-cocycles*, and elements of  $\text{im}(d^{i-1}) \subseteq C^i$  are called *i-coboundaries*. Since every coboundary is automatically a cocycle, we may define the *i-th cohomology group* (or space) as the space of cocycles modulo coboundaries:

$$H^i(C^\bullet) := \ker(d^i) / \text{im}(d^{i-1}). \quad (77)$$

Cohomology detects elements in  $C^i$  that are annihilated by  $d^i$  but are not exact—i.e., not the image of any element under  $d^{i-1}$ .

For context, the dual notion is that of a *chain complex*, where the maps go in the opposite direction. The homology of a chain complex is defined analogously, with

$$H_i(C_\bullet) := \ker(\partial_i) / \text{im}(\partial_{i+1}),$$

When the  $C^i$  are finite-dimensional vector spaces or free modules (modules that admit a basis), there is a natural identification  $C_i \cong (C^i)^*$ , and the differentials satisfy  $d^i = (\partial_{i+1})^T$ . In this case, there is a canonical isomorphism between homology and cohomology:

$$H^i(C^\bullet) \cong H_i(C_\bullet). \quad (78)$$

This identification is particularly useful when interpreting CSS codes and their logical operators in terms of cohomology.

#### Classical codes from cochain complexes

A classical binary linear code  $C \subset \mathbb{F}_2^n$  is a  $k$ -dimensional subspace with minimum distance  $d = \min_{c \in C} |c|$ , where  $|\cdot|$  denotes the Hamming weight. The code is specified by a parity check matrix  $H \in \mathbb{F}_2^{n \times r}$  such that  $\ker(H) = C$ .

This data can be represented as a 2-term cochain complex:

$$C^\bullet : \quad \mathbb{F}_2^r \xrightarrow{H^T} \mathbb{F}_2^n \quad (79)$$

where  $\mathbb{F}_2^n$  (in degree 1) represents the bits and  $\mathbb{F}_2^r$  (in degree 0) the checks. The map  $H^T$  is the coboundary operator, and the codewords are given by the degree-1 cohomology:  $H^1(C^\bullet) = \ker(H) = \text{coker}(H^T)$ .

A choice of basis for  $\mathbb{F}_2^r$  and  $\mathbb{F}_2^n$  corresponds to choosing generators for the checks and a coordinate system for the message space. Code parameters will be denoted by  $[n, k, d]$ .

## Quantum CSS codes as 3-term cochain complexes

A quantum CSS code is defined by a pair of classical binary codes  $C_X, C_Z \subseteq \mathbb{F}_2^n$  satisfying the inclusion  $C_Z^\perp \subseteq C_X$ , which is equivalent to  $H_X H_Z^T = 0$ . This ensures commutativity of the corresponding stabilizer operators.

This data defines a 3-term cochain complex:

$$C^\bullet : \quad \mathbb{F}_2^{r_Z} \xrightarrow{H_Z^T} \mathbb{F}_2^n \xrightarrow{H_X} \mathbb{F}_2^{r_X} \quad (80)$$

where  $\mathbb{F}_2^n$  (degree 1) represents the qubits,  $H_Z^T$  defines the  $Z$ -type stabilizers, and  $H_X$  defines the  $X$ -type stabilizers. The degree-1 cohomology  $H^1(C^\bullet) = \ker(H_X)/\text{im}(H_Z^T)$  defines the  $Z$ -logical operators, while the degree-1 homology  $H_1(C_\bullet) = \ker(H_Z)/\text{im}(H_X^T)$  defines the  $X$ -logical operators. The number of encoded qubits is

$$k = \dim H^1(C^\bullet) = \dim H_1(C_\bullet),$$

and the distance is  $d = \min(d_X, d_Z)$ , where  $d_X$  and  $d_Z$  are the minimum weights of nontrivial logical  $X$  and  $Z$  operators, respectively.

More generally, any  $n$ -term (co)chain complex with  $n \geq 3$  can define a CSS code by selecting three consecutive terms  $C^{i-1} \rightarrow C^i \rightarrow C^{i+1}$ .

## Homological products

We describe two constructions for combining classical codes to produce quantum codes: the tensor product and the balanced product. The tensor product generalizes the hypergraph product. The balanced product incorporates group symmetries to reduce size while preserving code properties.

### Tensor products of cochain complexes

Let  $(C^\bullet, \delta^C)$  and  $(D^\bullet, \delta^D)$  be cochain complexes of  $\mathbb{F}_2$ -vector spaces. Their tensor product is a new cochain complex defined by

$$(C \otimes D)^n = \bigoplus_{i+j=n} C^i \otimes D^j, \quad (81)$$

with coboundary operator

$$\delta(c \otimes d) = \delta^C(c) \otimes d + (-1)^i c \otimes \delta^D(d), \quad \text{for } c \in C^i, d \in D^j. \quad (82)$$

When working over  $\mathbb{F}_2$ , the sign  $(-1)^i$  may be omitted.

Tensoring an  $m$ -term and an  $n$ -term cochain complex yields an  $(m + n - 1)$ -term complex. This construction applies directly to the cochain complexes associated with classical codes. The hypergraph product of two classical codes is the CSS code associated with the 3-term tensor product of their respective 2-term cochain complexes. In Section 4.3, we show how the Künneth formula recovers the standard dimension and distance bounds for hypergraph product codes.

## Balanced products

The balanced product modifies the tensor product by incorporating a group symmetry. This typically reduces the code size while preserving—or sometimes improving—the relative distance and rate.

### Balanced Product of Vector Spaces

Let  $V$  and  $W$  be  $\mathbb{F}_2$ -vector spaces equipped with compatible linear  $G$ -actions. Then  $V$  and  $W$  are modules over the group algebra  $\mathbb{F}_2[G]$ . The *balanced product* is defined as

$$V \otimes_G W := V \otimes_{\mathbb{F}_2[G]} W. \quad (83)$$

This is the quotient of the standard tensor product by the subspace generated by elements of the form

$$g \cdot v \otimes w - v \otimes g \cdot w, \quad \text{for all } g \in G, v \in V, w \in W. \quad (84)$$

In terms of bases, let  $X$  and  $Y$  be  $G$ -invariant bases for  $V$  and  $W$ . Then a basis for  $V \otimes_G W$  is given by the orbits of the anti-diagonal group action:

$$X \times_G Y := X \times Y / \{(g^{-1}x, gy) : g \in G\}. \quad (85)$$

### Balanced Product of Cochain Complexes

Let  $G$  be a finite (Abelian) group, and let  $C^\bullet, D^\bullet$  be cochain complexes of  $\mathbb{F}_2$ -vector spaces equipped with a linear  $G$ -action on each term. Assume the coboundary maps  $\delta^C, \delta^D$  commute with the group action:

$$g \cdot \delta^C(x) = \delta^C(g \cdot x), \quad g \cdot \delta^D(y) = \delta^D(g \cdot y). \quad (86)$$

The balanced product complex is defined by

$$(C \otimes_G D)^n = \bigoplus_{i+j=n} C^i \otimes_G D^j, \quad (87)$$

with coboundary operator

$$\delta = \delta^C \otimes \mathbb{I} + \mathbb{I} \otimes \delta^D. \quad (88)$$

This defines a valid cochain complex. As with the tensor product, quantum CSS codes can be constructed by selecting three consecutive terms of the balanced product complex.

We will particularly be interested in the case where  $C$  and  $D$  are complexes where the cochain spaces are direct sums of  $\mathbb{F}_2[G]$  itself, as this is the scenario defining group-algebra quantum codes.

In this case, the bases for the cochain spaces are given by copies of  $G$ . For example, we may consider then consider the balanced product of two 2-term cochain complexes

$$\mathbb{F}_2[G] \xrightarrow{a} \mathbb{F}_2[G] \quad \mathbb{F}_2[G] \xrightarrow{b} \mathbb{F}_2[G]$$

where the coboundary maps are respectively defined by algebra multiplication by  $a \in \mathbb{F}_2[G]$  and  $b \in \mathbb{F}_2[G]$ . Using the notation  $R \equiv \mathbb{F}_2[G]$ , the balanced product complex is given by

$$R \otimes_R R \xrightarrow{\begin{pmatrix} a \otimes \text{id} \\ \text{id} \otimes b \end{pmatrix}} R \otimes_R R \oplus R \otimes_R R \xrightarrow{(\text{id} \otimes b \ a \otimes \text{id})} R \otimes_R R \quad (89)$$

For module tensor products of  $R$  with itself, we have a useful isomorphism  $R \otimes_R R \cong R$  given by  $r \otimes_R s \cong rs$  (see Appendix A of [33] for example). This yields the bicycle code chain complex in Eq. (24). Iterating the balanced product with a third classical complex defined by  $c \in \mathbb{F}_2[G]$  yields the complex in Eq. (25).

## Künneth Theorem

The Künneth theorem provides a formula for computing the cohomology of a tensor product of cochain complexes from the cohomologies of the constituent complexes.

Let  $C^\bullet$  and  $D^\bullet$  be cochain complexes of vector spaces over a field, such as  $\mathbb{F}_2$ . Then the Künneth theorem states:

$$H^n(C^\bullet \otimes D^\bullet) \cong \bigoplus_{i+j=n} H^i(C^\bullet) \otimes H^j(D^\bullet). \quad (90)$$

This expression holds exactly when working over a field and allows one to compute the cohomology of the product complex from the cohomologies of the factors.

More generally, suppose  $C^\bullet$  and  $D^\bullet$  are cochain complexes of modules over a finite-dimensional algebra  $A$  over a field, such as  $\mathbb{F}_2[G]$  for a finite group  $G$ . Then the Künneth formula similarly holds [65]:

$$H^n(C^\bullet \otimes_A D^\bullet) \cong \bigoplus_{i+j=n} H^i(C^\bullet) \otimes_A H^j(D^\bullet), \quad (91)$$

where  $\otimes_A$  denotes the tensor product over the algebra  $A$  – i.e the balanced product.