quantum reports

MDPI

Article

A Four-Dimensional Toric Code on the D_4 Lattice: $[[16L^4, 6, 2L^2]]$ with Weight-4 X-Stabilizers

Raghu Kulkarni

SSMTheory Group, IDrive Inc., Calabasas, CA 91302, USA; raghu@idrive.com

Abstract

We study a four-dimensional toric code on the D_4 lattice, the densest four-dimensional lattice packing. Qubits sit on the $16L^4$ triangles of its Delaunay decomposition into regular 16-cells. Each X-stabilizer acts on the four triangles of a tetrahedron, the smallest possible weight, and each Z-stabilizer on the eight triangles meeting along an edge. Up to exchanging X and Z, this is the face code of the dual 24-cell honeycomb, which has been proposed before without its parameters. We prove that the code is $[[16L^4, 6, 2L^2]]$ for every even $L \geq 4$, with $d_X = 2L^2$ and $d_Z = 4L^2$ exactly. A logical operator must cast a shadow covering a two-dimensional cross-section of the torus, and a single triangle shadows an area of at most $1/2$; explicit operators built from flat planes attain the resulting bounds in both sectors. The code reaches distance d with $4d^2$ qubits against $6d^2$ for the standard hypercubic code, although rotated hypercubic codes need fewer. Hook errors from the weight-8 checks cost no more than those from the weight-4 checks. Excitations in both sectors are loops, and the slices of the lattice are face-centered cubic. Published scripts verify all results.

Keywords: quantum error correction; four-dimensional toric code; D_4 lattice; 16-cell honeycomb; code distance; self-correcting quantum memory; hook errors; lattice gauge theory

1. Introduction

The four-dimensional toric code is the standard example of a quantum memory that protects itself. In two dimensions, errors form strings whose ends are free to wander, and a long chain of errors costs no more energy than a short one. In four dimensions both kinds of error form closed loops instead, and a big loop costs more energy than a small one. Thermal noise is therefore pushed back rather than allowed to spread [1–3]. On the hypercubic lattice the code is $[[6L^4, 6, L^2]]$, with qubits on square faces and weight-6 checks on edges and cubes [1,4].

Known variants change the shape of the space or the shape of the support. Codes with qubits on the 2-faces of regular tessellations of hyperbolic four-space keep the sheet-like logical operators and gain a growing rate: Guth and Lubotzky built the first such family [5], Londe and Leverrier used the $\{4, 3, 3, 5\}$ tessellation by hypercubes [6], and Breuckmann and Londe used the self-dual $\{5, 3, 3, 5\}$ tessellation by 120-cells and decoded it single-shot [7]. Brell placed the code on fractal subsets of the hypercubic lattice that fit in three dimensions [8], and four-dimensional color codes are self-correcting as well [9,10]. Closest to this work, Jochym-O'Connor and Yoder built four-dimensional toric codes on the 24-cell tessellation with qubits on 3-cells [11], and Kubica and Vasmer showed those codes to be gauge fixings of their four-dimensional subsystem toric code, with the 16-cell honeycomb turning up as the alternated lattice of that correspondence [12]. In that correspondence, the qubits of the 16-cell honeycomb sit on its *edges*, which under duality are the 3-cells of

Check for updates

Academic Editor: Yanpeng Zhang

Received: 14 October 2026
Revised: 27 October 2026
Accepted: 30 October 2026
Published: 3 November 2026

Copyright: © 2026 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license.

the 24-cell honeycomb. The code on this lattice is then the $(1,3)$ toric code, whose logical operators are strings and volumes [11,13].

Here we take a different sector of the same geometry and put the qubits on the *triangles*, which gives the $(2,2)$ code. Both kinds of logical operator are then sheets and both kinds of excitation are loops, which is the structure that makes the four-dimensional toric code self-correcting [4]. The honeycomb carries exactly three such sectors, with qubits on edges, triangles or tetrahedra. The first and third are codes of the same type, with strings and volumes as logical operators and the roles of X and Z exchanged. The middle one is self-dual: duality takes triangles of this honeycomb to 2-cells of the 24-cell honeycomb; so, its code is the face code of either honeycomb. So, this sector is not the dual of the $(1,3)$ code and does not appear in that correspondence, and it is the only sector of the lattice in which both kinds of excitation are loops. Building a $(2,2)$ code on a four-dimensional complex is a standard construction, and in hyperbolic space it has been done on regular tessellations [5–7]. Flat four-space has exactly three regular honeycombs. The tesseractic one gives the hypercubic code. The 16-cell and 24-cell honeycombs are dual to each other, so their $(2,2)$ codes are one code with X and Z exchanged, and that code is the subject of this paper. It is the remaining regular case in flat space. The code itself has appeared before: Aasen, Haah, Hastings and Wang list the face code of the 24-cell honeycomb, with weight-8 checks on octahedra and weight-4 checks on edges, as one way to lower the stabilizer weights of four-dimensional codes [14]. That is the present code with X and Z exchanged. They give its check weights but not its distance or its other parameters. Determining them is the main result of this paper.

Recent work sets the scale for any new memory. Experiments now run surface-code memories below threshold [15], and bivariate bicycle codes, a family of quantum low-density parity-check codes, promise the same protection as the surface code with about a tenth of the qubits [16]. For four-dimensional codes, two developments matter here. First, Aasen et al. showed that choosing the periodic boundary conditions of the hypercubic code along a tilted lattice ("rotating" it) lowers its cost from $6d^2$ to between about $1.2d^2$ and $1.5d^2$ qubits, with examples such as $[[96,6,8]]$ and $[[270,6,15]]$ [14], and a fault-tolerant architecture has been designed around these codes, with weight-6 checks and syndrome circuits of depth 8 [17]. Second, the thermal behavior of the four-dimensional toric code is now understood rigorously within each logical sector [18]. On qubit count alone, the code studied here does not compete with the rotated codes. Its interest is different: it has the smallest possible X-checks, three checks of each type per qubit, and a distance that can be computed exactly for every L. Whether rotation can lower its cost too is an open question (Section 9).

We use the lattice that four-dimensional geometry singles out. D_4 is the four-dimensional checkerboard: the densest lattice packing in four dimensions, in which every point touches 24 others [19]. Its gaps are all of one kind, which is special to D_4, and filling them in gives a honeycomb of identical regular 16-cells. The resulting code has three properties we want to set out first.

(i) **The X-checks are as small as they can be.** The 3-cells of this honeycomb are tetrahedra; so, every X-check touches 4 qubits. A 3-cell has at least 4 faces; so, no check of this type can be smaller. The price is a weight-8 Z-sector.

(ii) **The distances are $d_X = 2L^2$ and $d_Z = 4L^2$.** The smaller one is twice the hypercubic value at the same L. Section 5 proves both. Section 8 works out what this costs in hardware: at a fixed target distance, the code needs one third fewer qubits than the standard hypercubic code, though rotated hypercubic codes need fewer still.

(iii) **Slicing gives the FCC lattice.** Cutting at a fixed value of the fourth coordinate leaves the face-centered cubic (FCC) lattice, and the cut passes cleanly through the honeycomb.

https://doi.org/10.3390/quantum8040116

A minimum-weight X-type logical operator of the whole four-dimensional code lies inside one such slice.

Goals and methods.

This paper has four goals: to find the exact parameters $[[n,k,d]]$ of this code for every even L, in both sectors; to explain which logical operators are smallest and why; to compare its cost with that of other four-dimensional toric codes at equal distance; and to check whether its weight-8 checks are a weak point once syndrome measurement is taken into account. The methods are of two kinds. The proofs use the chain complex of the honeycomb, a systolic ("shadow") argument for the lower bounds on the distance, and explicit operators for the matching upper bounds. Every statement is also checked by computer: exact linear algebra over $\mathbb{F}_2$ for ranks and stabilizer tests, exact rational arithmetic for the geometric constants, and satisfiability (SAT) solvers for exhaustive minimum-weight searches. The checks run at $L = 4$ and, where feasible, at $L = 6$. The scripts are listed in the Data Availability Statement and run in minutes. Structural claims hold for all even $L \geq 4$ by the arguments given. The case $L = 2$ is degenerate: the torus is too small and edges collapse; so, $L \geq 4$ throughout.

What is standard and what is new.

Several ingredients are standard, and we restate them only to keep this paper self-contained and to fix notation: the construction of a CSS code from a cell complex [1,20], the facts about D_4 and its Delaunay cells [19], the systolic argument for lower bounds on the distance [5,21], and hook errors [1]. Our results apply these tools to this honeycomb, and what is new is what comes out: the exact local constants $1/2$ and $1/4$ in the shadow argument, and the fact that both are attained, so that $d_X = 2L^2$ and $d_Z = 4L^2$ hold exactly for every L rather than up to a constant (Section 5); the description of which logical classes have flat operators (Lemma 2); the slice structure and its gauge theory reading (Section 7); and the circuit-level comparison with the hypercubic code (Section 8). A related construction on the three-dimensional close-packed lattice appears in [22].

2. The Lattice and Its Honeycomb

Take the points of $\mathbb{Z}^4$ whose coordinates add up to an even number:

$$D_4 = \{x \in \mathbb{Z}^4 : x_1 + x_2 + x_3 + x_4 \text{ is even}\},$$

and wrap them on a four-dimensional torus of even side L. Two points are neighbors when they differ by a *root*: a vector with two entries equal to ± 1 and two entries equal to 0. There are 24 roots; so, every point has 24 neighbors. That is the largest number possible in four dimensions [19,23]. The same lattice appears elsewhere in quantum error correction as the phase-space lattice of a two-mode Gottesman–Kitaev–Preskill code [24]. That use is unrelated: there, D_4 is a lattice of oscillator displacements, while, here, it supplies the cells of a qubit code.

The gaps between the points come in three families, odd-sum integer points and two families of half-integer points, but they are all the same shape. Each gap is surrounded by 8 lattice points forming a regular 16-cell (Figure 1a). Filling every gap tessellates the torus with $3L^4/2$ identical 16-cells. This is the Delaunay decomposition of D_4.

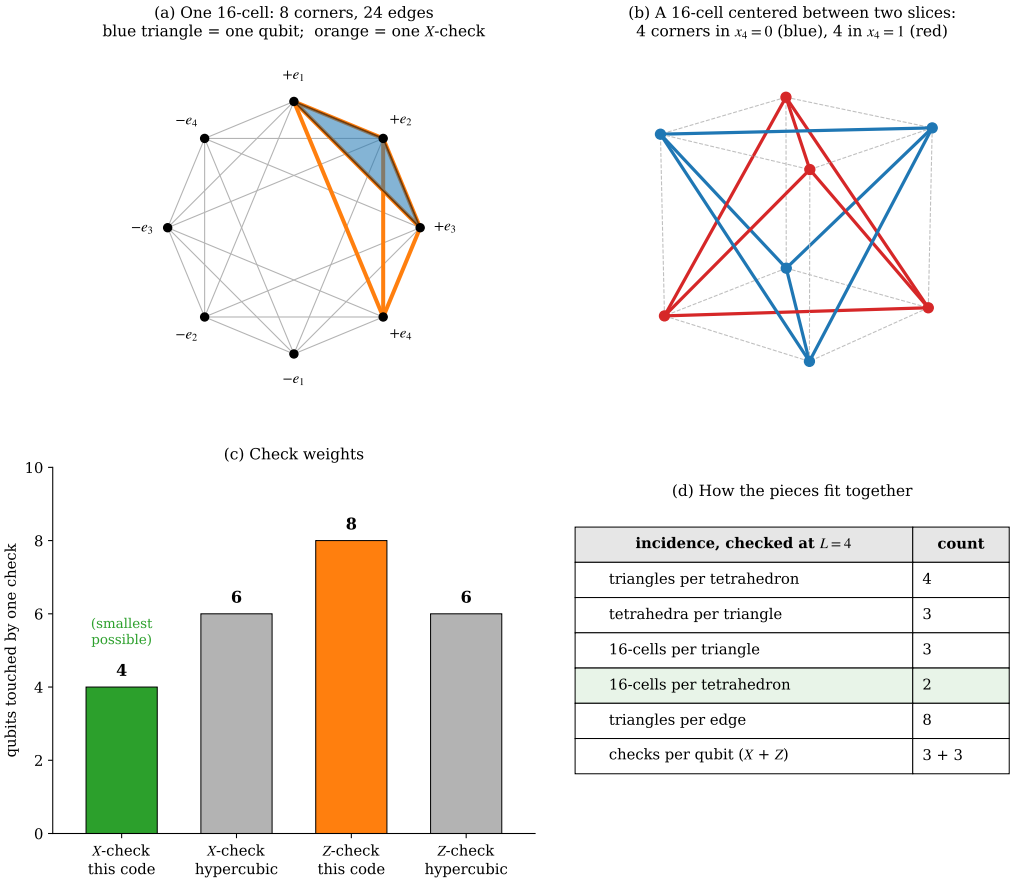
Table 1 counts the pieces of the honeycomb.

https://doi.org/10.3390/quantum8040116

Table 1. Cells of the 16-cell honeycomb on a torus of side L.

	Points	**Edges**	**Triangles**	**Tetrahedra**	**16-Cells**
count	$L^4/2$	$6L^4$	$16L^4$	$12L^4$	$3L^4/2$
$L=4$	128	1536	4096	3072	384

The alternating sum is zero, as it must be on a torus. The way the pieces fit together is checked exhaustively at $L=4$ (Figure 1d): every triangle lies in 3 tetrahedra and in 3 16-cells; every tetrahedron lies in exactly 2 16-cells; every edge lies in 8 triangles. Triangles are two steps down from the top dimension here, which is why 3 appears rather than the 2 one expects for a face of a solid. The number 2 for tetrahedra is the one that matters later. It makes each X-check an edge of the dual honeycomb, so that error signals can be read as loops there (Theorem 5).



incidence, checked at $L=4$	**count**
triangles per tetrahedron	4
tetrahedra per triangle	3
16-cells per triangle	3
16-cells per tetrahedron	2
triangles per edge	8
checks per qubit (X + Z)	3 + 3

Figure 1. The building blocks. (**a**) One 16-cell of the honeycomb, drawn flat. Its 8 corners sit at $\pm e_1, \ldots, \pm e_4$ from its center, and every pair of corners except opposite ones is joined, giving 24 edges. A qubit lives on each triangle (blue); each tetrahedron carries a weight-4 X-check (orange: the tetrahedron with corners $+e_1, \ldots, +e_4$). (**b**) A 16-cell centered between two slices; here, the one with corners at the points of $\{0,1\}^4$ with even sum: its 8 corners split 4 and 4, one group in the slice $x_4=0$ (blue) and one in the slice $x_4=1$ (red). Dashed lines are the 12 edges joining the two groups. This is the geometric content of Section 7. (**c**) Check weights against the hypercubic four-dimensional toric code: the X-sector reaches the smallest weight a check of this kind can have; the Z-sector pays weight 8. (**d**) The verified incidence counts.

3. The Code

Definition 1 (D_4 face code)**.** *Put one qubit on each triangle of the 16-cell honeycomb. Then:*

- X-stabilizers*: for each tetrahedron, apply X to its 4 triangles (weight 4);*

- Z-stabilizers*: for each edge, apply Z to the 8 triangles containing that edge (weight 8).*

Every X-check has weight 4 and every Z-check has weight 8, with no exceptions anywhere on the torus (Figure 1c). Each qubit is touched by three checks of each type. The two check types commute because a tetrahedron and an edge always share an even number of triangles; in the language of chain complexes, $H_X = \partial_3^{\mathsf{T}}$ and $H_Z = \partial_2$; so, $H_X H_Z^{\mathsf{T}} = 0$ follows from $\partial\partial = 0$. We verify it numerically regardless (Data Availability Statement). This is the hypercubic construction with squares replaced by triangles and cubes by tetrahedra.

4. Numerical Verification of the Parameters at $L = 4$

Theorem 1 (Parameters at $L = 4$)**.** *At $L = 4$, the code is a valid CSS code with*

$$n = 4096, \qquad \mathrm{rk}(H_Z) = 1405, \qquad \mathrm{rk}(H_X) = 2685, \qquad k = n - \mathrm{rk}(H_Z) - \mathrm{rk}(H_X) = 6.$$

The number of logical qubits is fixed by the shape of the space rather than by the lattice. A four-dimensional torus has $\binom{4}{2} = 6$ independent two-dimensional sheets; so, $k = 6$ at every L. The six X-type logical operators are sheets that wrap the torus; the six Z-type ones are sheets in the dual honeycomb, which is the 24-cell honeycomb. Both are sheets, as in the hypercubic case [1,4].

As a further check, an exhaustive scan over all $\binom{4096}{2}$ pairs of qubits finds that the only weight-4 operators that commute with every Z-check are the 3072 tetrahedron boundaries, that is, the X-stabilizers themselves, with nothing extra, and that no weight-4 operator commutes with every X-check. Both facts are consistent with the distance computed next.

5. The Distances Are Exactly $2L^2$ and $4L^2$

We first give the argument informally. A logical operator is a sheet that wraps the torus. Look at its *shadow*: pick two of the four coordinates and squash the torus onto that two-dimensional cross-section. A wrapping sheet has to shadow the whole cross-section, with no gaps. The cross-section has area L^2, and one triangle can shadow at most an area of $1/2$; so, at least $2L^2$ triangles are needed. A close-packed plane of the lattice has exactly that many. Z-type logical operators are sheets in the dual honeycomb, where the same argument gives $4L^2$, and a flat plane crossing the lattice attains it.

This is the systolic argument that underlies distance bounds for homological codes [5,21]. What is particular to this lattice is the pair of local constants, and the fact that both are attained exactly.

The local ingredient comes first.

Lemma 1 (How big a shadow can be)**.** *Let π_{ij} be the map that keeps only the coordinates x_i and x_j. Every triangle f of the honeycomb has* $\mathrm{area}(\pi_{ij}(f)) \leq \frac{1}{2}$*, and every 2-cell f^* of the dual 24-cell honeycomb has* $\mathrm{area}(\pi_{ij}(f^*)) \leq \frac{1}{4}$*. Both values are reached.*

Proof. A triangle is $\{v, v+r, v+s\}$ where r, s and $r-s$ are all roots; so, its shadow is a triangle of area $\frac{1}{2}|r_i s_j - r_j s_i|$. A root has exactly two nonzero entries, each ± 1; so, $|r_i s_j - r_j s_i| \leq 2$. The value 2 needs r and s to both live on the coordinates i and j with opposite sign patterns, say $r = (1, 1, 0, 0)$ and $s = (1, -1, 0, 0)$. But, then, $r - s$ and $r + s$ each have a single entry ± 2, and neither is a root; so, there is no such triangle. Hence, $|r_i s_j - r_j s_i| \leq 1$ and the area is at most $\frac{1}{2}$. It is reached, for instance, by $r = (1, 1, 0, 0)$ and $s = (1, 0, 1, 0)$ shadowed onto (x_1, x_2).

A 2-cell of the dual honeycomb is the triangle joining the centers of the 3 16-cells that contain a given triangle (Section 2). Running through the 96 triangles at a lattice point and all 6 coordinate pairs gives a largest shadow of $\frac{1}{4}$. The computation is the published script `projection.py` (Data Availability Statement); it is exact arithmetic and takes a second. □

Theorem 2 (Lower bound)**.** *For even $L \geq 4$, every X-type logical operator has a weight of at least $2L^2$, and every Z-type logical operator has a weight of at least $4L^2$.*

Proof. Let z be an X-type logical operator, that is, a sheet that wraps the torus in a way no stabilizer can undo. Its wrapping class lives in $H_2(T^4; \mathbb{F}_2)$, which has a basis labeled by the six pairs of coordinates; choose a pair (i, j) appearing in the class of z, and write π for the shadow map onto (x_i, x_j).

Suppose some point p of the cross-section were missed by the shadow. Then, z would lie entirely inside the part of the torus lying over the punctured cross-section. A punctured two-dimensional torus carries no wrapping sheets at all; so, the class of z would have to be trivial in the (i, j) direction, contradicting the choice of (i, j). So, the shadow covers every point of the cross-section.

Areas can only add up; so, the following is observed:

$$\sum_{f \in \operatorname{supp} z} \text{area}\left(\pi(f)\right) \;\geq\; \text{area}\left(\text{cross-section}\right) \;=\; L^2.$$

By Lemma 1, each term is at most $\frac{1}{2}$; so, z contains at least $2L^2$ triangles.

A Z-type logical operator is the same kind of object in the dual honeycomb, where the qubits correspond one-to-one to dual 2-cells. The same argument with the constant $\frac{1}{4}$ gives at least $4L^2$. □

Theorem 3 (An X-type logical operator of exactly this size)**.** *Let $L \geq 4$ and c be even, and let*

$$\Pi_c \;=\; \{x : x_1 + x_2 + x_3 \equiv c \bmod L,\; x_4 = 0\}.$$

Then, Π_c contains exactly L^2 lattice points, and the $2L^2$ triangles they span are triangles of the honeycomb and form an X-type logical operator. Hence, $d_X \leq 2L^2$.

Proof. Because L is even, fixing $x_1 + x_2 + x_3$ modulo L also fixes it modulo 2; for c even, all L^2 solutions have an even coordinate sum and so lie in D_4. The roots that stay inside Π_c are those with $r_4 = 0$ and $r_1 + r_2 + r_3 = 0$, of which there are six; so, the points of Π_c form a triangular lattice, a close-packed plane (Figure 2a). A triangular lattice with N points on a two-dimensional torus has $3N$ edges and $2N$ triangles, and every edge belongs to exactly two of them. Every Z-check therefore sees an even number of triangles; so, the set commutes with all of them. The sheet wraps the torus: its homology class is $c_4 \neq 0$ in the notation of Lemma 2 below; so, no product of stabilizers can produce it. With $N = L^2$, its weight is $2L^2$. □

The matching Z-type operator is also flat, but it does not lie along the lattice. It is made of the triangles that a plane crosses.

Theorem 4 (A Z-type logical operator of exactly this size)**.** *Let $L \geq 4$ be even and let (a, b) be a generic point of the (x_3, x_4) torus, meaning one that lies on none of the lines $x_3 \in \mathbb{Z}$, $x_4 \in \mathbb{Z}$, $x_3 \pm x_4 \in \mathbb{Z}$. Let $Q = \{x : x_3 = a,\; x_4 = b\}$. The triangles of the honeycomb that Q passes through form a Z-type logical operator of weight exactly $4L^2$. Hence, $d_Z \leq 4L^2$.*

Proof. Q is a flat two-dimensional torus running along x_1 and x_2. It meets a cell exactly where the shadow of that cell on the (x_3, x_4) plane contains (a, b), and since (a, b) avoids the shadows of all edges, it never meets an edge or a corner.

It commutes with every X-check. Take a tetrahedron. If its shadow on the (x_3, x_4) plane has zero area, the shadow lies on one of the excluded lines and Q misses the tetrahedron. Otherwise, the points of the tetrahedron's three-dimensional span that lie over (a, b) form

a line, and that line meets the tetrahedron in a segment or not at all. The two ends of the segment lie inside two of the tetrahedron's four triangles, not on an edge, and Q meets no other triangle of it. So, every X-check shares 0 or 2 qubits with the operator.

It is not a product of stabilizers. Let z be an X-type logical operator whose class has a nonzero component along the coordinate pair $(3, 4)$; such operators exist because all six classes occur. The number of triangles shared by z and the operator is the number of points where Q crosses z, and, modulo 2, it is the intersection number of Q, a torus along $(1, 2)$, with z. That number is 1. A product of Z-stabilizers shares an even number of qubits with every X-type logical operator; so, the operator is not one.

Its weight is $4L^2$. The 96 triangles at a lattice point fall into 32 classes under translation. Enumerating them (script `zsheet.py`) shows that 16 classes have a shadow of zero area on the (x_3, x_4) plane; so, Q never meets them. The other 16 all have shadows of area $\frac{1}{2}$ with integer corners, and they form 8 pairs related by the point reflection $x \mapsto -x$. A triangle of area $\frac{1}{2}$ with integer corners and its point reflection fit together into a parallelogram of area 1 with integer corners, and such a parallelogram tiles the plane under integer translations. So, among the integer translates of the two shadows in a pair, exactly one covers a generic point, once. Each integer point of the (x_3, x_4) torus is the shadow of $L^2/2$ lattice points, the ones whose $x_1 + x_2$ has the right parity. Each pair therefore contributes $L^2/2$ triangles, and the weight is $8 \cdot L^2/2 = 4L^2$ (Figure 2c). □

Corollary 1 (Parameters)**.** *For every even* $L \geq 4$*, the* D_4 *face code has* $d_X = 2L^2$ *and* $d_Z = 4L^2$*; so, the following is observed:*

$$d \;=\; \min(d_X, d_Z) \;=\; 2L^2, \qquad \textit{and the code is } [[\,16L^4,\ 6,\ 2L^2\,]].$$

The bound $d_X \geq 2L^2$ is tight for a reason one can see in the picture. Each of the $2L^2$ triangles of Π_c shadows an area of exactly $\frac{1}{2}$, the most allowed, and the shadows tile the cross-section once with no overlaps and no gaps (Figure 2b). Nothing is wasted, which is why the constant comes out exact instead of only having the right order.

Only even weights can occur.

Every triangle lies on three edges; so, adding up all the Z-checks gives the operator acting on every qubit at once. Anything commuting with all the Z-checks must overlap that operator evenly, and so has even weight. Every triangle also lies in three tetrahedra; so, the same holds in the other sector. Odd weights are therefore impossible on both sides. The first argument also holds on a slice, where each triangle still lies on three edges, which is why the X-type search below tests even weights only.

A check on the argument.

The shadow argument applies just as well to the three-dimensional face code on a slice, where it again predicts that the smallest X-type logical operator has weight $2L^2$. That code is small enough to settle by brute force. We encoded "there is a logical operator of weight at most w" as a satisfiability problem, with parity constraints from the checks, one constraint forcing the operator to be nontrivial, and a counter limiting the weight, and raised w one step at a time (script `slice3d_distance.py`, Data Availability Statement). At $L = 4$, the answer is unsatisfiable at every even weight up to 30 and satisfiable at 32. The slice code therefore has $d_X = 32 = 2L^2$ exactly, which is what the shadow argument says it should be (Figure 2d).

This search settles the X sector of the slice code only. Its full distance is smaller, because its Z-type logical operators are strings rather than sheets. The same search for Z-type operators, testing every weight from 1 upward, finds none of weight 7 or less and one of

weight $8 = 2L$. So, the slice code has $d_Z = 8$ and $d = \min(32, 8) = 8$ at $L = 4$, as expected for a three-dimensional toric code. Going to four dimensions is what makes both sectors sheet-like. We have not run the search on the four-dimensional code, where $n = 4096$ puts it out of reach; there, the distances rest on Corollary 1. As a further check on the Z side, reducing Z-type logical operators greedily against the Z-stabilizers at $L = 4$ also stops at weight $64 = 4L^2$.

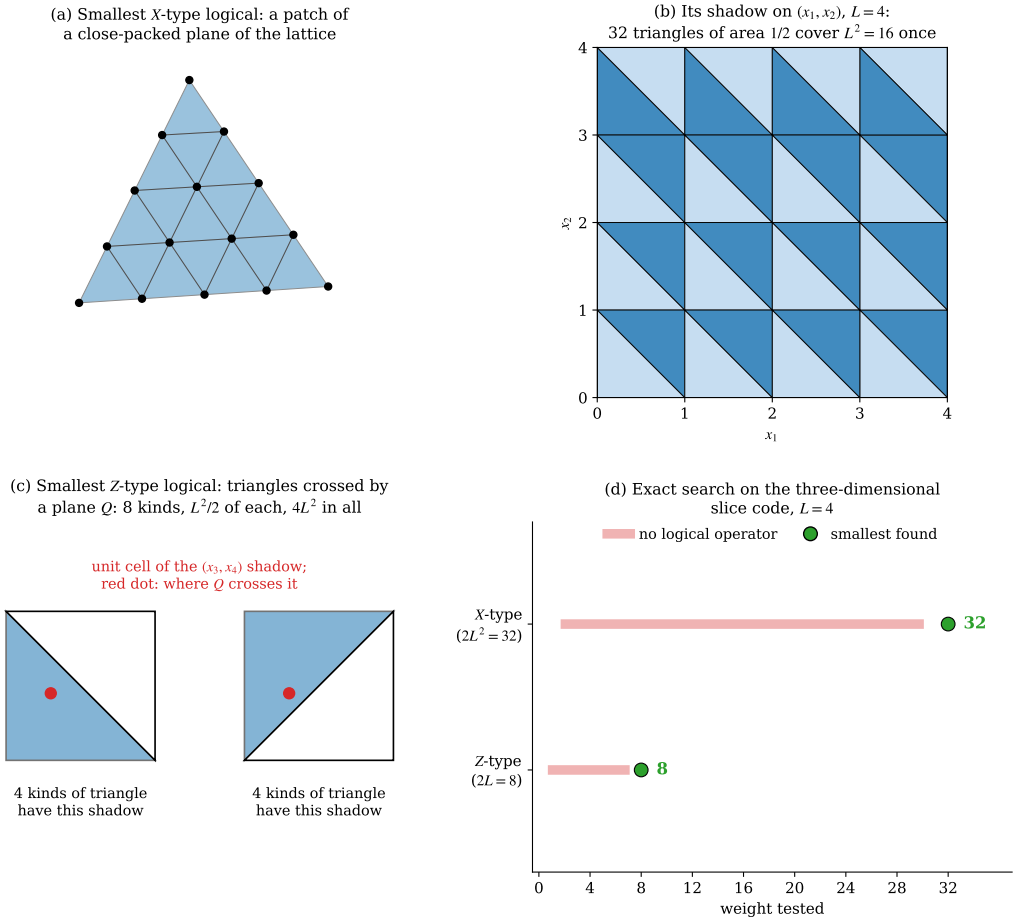


Figure 2. The distances. (**a**) A patch of a minimum-weight X-type logical operator, drawn unwrapped: a close-packed plane of the lattice. On the torus, the whole plane has L^2 points and $2L^2$ triangles (Theorem 3). (**b**) Its shadow on the (x_1, x_2) cross-section at $L = 4$: 32 triangles, each of area 1/2, tiling an area of $16 = L^2$ exactly once. Nothing is wasted, which is why the bound is attained. (**c**) A minimum-weight Z-type logical operator (Theorem 4): the triangles crossed by a plane Q along (x_1, x_2). On the (x_3, x_4) shadow, the crossing point (red) lies in one triangle of each of the two ways of cutting a unit square in half. Each of these two shadows comes from 4 kinds of triangle, and each kind occurs $L^2/2$ times, giving $4L^2$. (**d**) Exact SAT search on the three-dimensional slice code at $L = 4$ (Section 5). Red bars cover the weights at which no logical operator exists; circles mark the smallest one found. X-type: $32 = 2L^2$; odd weights are excluded by parity. Z-type: $8 = 2L$, with every weight from 1 tested.

Which logical qubits have flat operators.

Call an X-type logical operator *flat* if all its triangles lie in parallel 2-planes. The operator of Theorem 3 is flat. Write e_{ij} for the class of the coordinate torus along directions i and j; the six e_{ij} form a basis of $H_2(T^4; \mathbb{F}_2)$.

Lemma 2 (Which classes have flat operators)**.** *Every triangle of the honeycomb spans a 2-plane that lies along exactly three of the four coordinate directions, say i, j, k, and is not a coordinate plane.*

A flat X-type logical operator in such planes has class $c_l = e_{ij} + e_{ik} + e_{jk}$, where l is the missing direction. The four classes $c_1, \dots, c_4$ add up to zero and span a three-dimensional subspace of the six-dimensional space of classes. Only the four classes c_l have flat representatives. In particular, no coordinate class e_{ij} does, and the span of the c_l contains no e_{ij} at all.

Proof. A triangle is $\{v, v + r, v + s\}$ with r, s and $r - s$ roots; so, $s \neq \pm r$. If r and s had the same two nonzero coordinates, one of $r \pm s$ would have a single entry ± 2 (the case excluded in the proof of Lemma 1), and if they had none in common, $r - s$ would have four nonzero entries. Either way, $r - s$ would not be a root. So, they share exactly one coordinate: $r = \alpha e_i + \beta e_j$ and $s = \gamma e_i + \delta e_k$ with $\alpha, \beta, \gamma, \delta = \pm 1$. The plane through v spanned by r and s keeps x_l fixed and contains the lattice points $v + mr + ns$ with $m, n \in \mathbb{Z}$; on the torus, it closes up when m and n are multiples of L. Its class is then given by the 2×2 minors of r and s:

$$r_i s_j - r_j s_i = -\beta\gamma, \qquad r_i s_k - r_k s_i = \alpha\delta, \qquad r_j s_k - r_k s_j = \beta\delta,$$

all equal to ± 1, while every minor involving l is 0. Modulo 2, the class is c_l. In particular, no triangle spans a coordinate plane, whose class would have a single nonzero minor. Now, let z be a flat logical operator, with its triangles in a family of parallel planes. Each edge of z lies in exactly one plane of the family, and every triangle of z containing it lies in that same plane; so, the part of z in each plane commutes with every Z-check on its own. In one plane, the triangles form a triangular lattice on a two-dimensional torus, every edge in exactly two of them, and the only such sets that commute with all Z-checks are the empty set and the whole plane. So, z is a union of whole parallel sheets, and its class is 0 or c_l; being a logical operator, it is c_l. Each e_{ij} appears in exactly two of the c_l, those with $l \notin \{i, j\}$; so, $c_1 + c_2 + c_3 + c_4 = 0$, and any three of them are independent. The nonzero elements of their span are the four c_l, each with three terms, and the three sums $c_l + c_m$, each with four. None is a single e_{ij}. □

Geometrically, the lemma says that the coordinate planes, the obvious flat representatives of the coordinate classes, are not available. A coordinate plane $\{x_k = \text{const}, x_l = \text{const}\}$ of D_4 contains lattice points but no triangle of the honeycomb; so, it carries no operator at all, and every tilted plane that does carry triangles is pinned to the classes c_l. A representative of e_{12}, say, must bend out of every single plane. Theorem 2 still bounds its weight below by $2L^2$, but it cannot meet that bound with a flat sheet.

An exhaustive search at $L = 4$ agrees with the lemma. It finds 128 flat sheets, every one of weight 32, realizing the four classes c_l and spanning three dimensions, and every one lying inside a slice $x_l = \text{const}$, that is, inside a copy of the FCC lattice (Section 7). We have not found the minimum weights of the other classes. A seeded search at $L = 4$, one class in each of three independent directions outside the flat part, finds operators of weight 78, 116 and 126. These are upper bounds, not minima. They do not change d, which is a minimum over all classes.

6. Excitations Are Loops: What Is Proved and What Is Not

Theorem 5 (No point excitations)**.** *Neither sector of the D_4 face code has point-like excitations:*

(a) No error can light up a single X-check. The linear system $H_X y = e_t$ has no solution for one tetrahedron t (verified directly). The reason is $\partial\partial = 0$: the lit checks inside each 16-cell always come in even numbers. Every tetrahedron sits in exactly two 16-cells; so, a lit tetrahedron is an edge of the dual honeycomb, and the lit edges form closed loops there. A single dual edge is not a closed loop;

(*b*) *No error can light up a single Z-check, for a reason that needs no computation: the pattern of lit edges is always the boundary of a set of triangles, and a boundary is always a closed loop. A single edge is not a closed loop.*

What Theorem 5 establishes.

It is a statement about the stabilizer group alone, and it holds exactly for every L. In both sectors, the set of checks that an error violates is a closed loop: a loop in the dual honeycomb for X-checks, and the boundary of a set of triangles for Z-checks. No error produces an isolated point excitation. If the code is run as a passive memory with the Hamiltonian $H = -\sum_t A_t - \sum_e B_e$, one term for each check, then, the energy of an error is the total length of its loops. The shortest loops have length 3, from a single wrong qubit, which lights three checks of each type. This is the same structure that the hypercubic four-dimensional toric code has [1,4]. One detail matters for anyone building a decoder: the odd number 3 follows from the triangular geometry and is more familiar from color codes.

What it does not establish.

Self-correction is a statement about thermal noise, and loop-like excitations are necessary for it but not sufficient. Three further statements would be needed, and we prove none of them for this honeycomb. (i) *An energy barrier that grows with L.* Every sequence of single-qubit errors that carries one codeword to another should pass through a state whose loops have total length at least of order L. Growing a logical sheet one triangle at a time shows that the barrier is at most of order L; the matching lower bound needs an isoperimetric inequality on this honeycomb, like the one used for the hypercubic lattice [1,2]. (ii) *A memory time that grows exponentially with L* below some temperature. This is proved for the hypercubic code [2], and recent work describes rigorously how that code thermalizes within each logical sector [18]. (iii) *A threshold for a local decoder*, discussed in Section 9. We expect all three to carry over, because the proofs for the hypercubic code use only the local geometry of the cell complex, but this is a conjecture, not a result of this paper. Whether the code is self-correcting is therefore open. What is proved is the loop structure that self-correction requires.

7. Slicing Gives the FCC Lattice

Cut the lattice at a fixed value of the fourth coordinate. What is left is exactly the face-centered cubic lattice, and the honeycomb cuts cleanly along with it. All of the following is verified at $L = 4$:

(a) The slice contains $L^3/2$ points, $3L^3$ edges, $4L^3$ triangles and L^3 tetrahedra, which are the points, edges, triangles and tetrahedra of the three-dimensional tetrahedron-octahedron honeycomb, the complex obtained by filling the gaps of the FCC lattice; its octahedra appear as cross-sections of 16-cells, item (c);

(b) Every edge in the slice lies in exactly four triangles of the slice, which is the Z-check weight of the three-dimensional face code on that lattice;

(c) The gaps cut cleanly. A 16-cell centered in the slice meets it in an octahedron (6 of its 8 corners); a 16-cell centered between slices meets it in a tetrahedron (4 of 8, with the other 4 in the next slice, Figure 1b); a 16-cell a full step away meets it in a single corner and contributes nothing.

The slices carry the code's best X-type logical operators. By Theorem 3, one minimum-weight X-type logical operator of the whole four-dimensional code is a close-packed plane of a slice, and by Lemma 2, each sheet of a flat X-type logical operator lies in a slice $x_l = \text{const}$ for some coordinate l. Everything said here about x_4 holds for any coordinate, by symmetry.

An existing decoder for the slice geometry.

The FCC lattice of a slice carries a known three-dimensional toric code with qubits on its edges [12,25], and it is dual to the rhombic dodecahedral lattice. On that lattice, Vasmer, Browne and Kubica proved that a local decoder based on the sweep rule has a nonzero error threshold, and they measured how well it works when the checks themselves are noisy [26]. The sweep rule works on any locally Euclidean lattice, and the sweep decoder built from it covers toric codes in three or more dimensions [27]; so, it is the first thing to try here.

Two warnings. The slice inherits the geometry but not the code. A tetrahedron that straddles a slice leaves only one of its triangles inside; so, cutting the checks down to a slice gives weight-1 operators rather than the three-dimensional code. We have also not run the sweep decoder on the 16-cell honeycomb or measured its threshold there. The slice says where to start looking.

The 16-cell honeycomb is also the alternated lattice through which Kubica and Vasmer relate the four-dimensional subsystem toric code to the 24-cell toric codes of Jochym-O'Connor and Yoder [11,12], and the three-dimensional subsystem toric code of that same correspondence is the standard single-shot code [12,28]. The lattice is already present in the single-shot and self-correction literature, and its triangle sector has been listed [14] but not analyzed. Writing down a dimension-jump protocol [29] along a slice remains open.

A Lattice Gauge Theory Reading

The stabilizer Hamiltonian of a toric code is a $\mathbb{Z}_2$ lattice gauge theory at a special point of its couplings [1,30,31]. Both the slice code and the full code have such a reading, and the second one gives a concrete prediction.

The slice: The face code of the tetrahedron-octahedron honeycomb is the ordinary three-dimensional toric code on the dual rhombic dodecahedral lattice. Its qubits (triangles) are the links of that lattice, its weight-4 Z-checks (edges) are the rhombic plaquettes, and its X-checks (tetrahedra and octahedra) are the Gauss-law constraints at the vertices of degree 4 and 8. Its ground state is therefore the deconfined phase of $\mathbb{Z}_2$ gauge theory on the rhombic dodecahedral lattice, written in the Hamiltonian form of Kogut and Susskind [31,32], and its string-like Z-type logical operators of weight $2L$ (Section 5) are Wilson loops.

The full code: In four dimensions, the D_4 face code is a $\mathbb{Z}_2$ gauge theory whose basic variables live on plaquettes (the triangles) rather than links. Both its electric and its magnetic excitations are loops (Theorem 5). This reading has a direct use. Dennis et al. showed that the best possible error threshold of a toric code under independent noise is a phase transition of a statistical model with random couplings [1]. For the hypercubic four-dimensional code, that model is the four-dimensional random-plaquette $\mathbb{Z}_2$ gauge model. It is self-dual, and Takeda and Nishimori used this to predict the threshold $p_c \approx 0.110$, the root of $H(p) = 1/2$, where H is the binary entropy [33]. For the D_4 face code, the same mapping gives the random-plaquette model on the 24-cell honeycomb for one type of error and on the 16-cell honeycomb for the other. These lattices are not self-dual but dual to each other. That is the case covered by the duality conjecture of Takeda, Sasamoto and Nishimori [34], which predicts that the two optimal thresholds satisfy:

$$H(p_X) + H(p_Z) = 1.$$

The conjecture is known to be accurate, though not always exact, in the cases where it has been tested. Checking this relation numerically for the D_4 face code would test both the code and the conjecture, and it is a prediction that comes from lattice gauge theory rather than from coding theory.

8. Comparison with Other Four-Dimensional Toric Codes

The natural comparison is at equal distance, not at equal side length. Side length is not comparable between two different lattices; distance is what a device designer is buying. The standard hypercubic code on an L^4 torus needs $L^2 = d$; hence, $n = 6d^2$ qubits. The D_4 face code needs $2L^2 = d$; hence, $n = 16L^4 = 4d^2$. Table 2 also lists the rotated hypercubic codes of Aasen et al. [14,17].

Table 2. The D_4 face code compared with the hypercubic four-dimensional toric code and its rotated versions.

	Hypercubic [1]	**Rotated Hypercubic [14]**	**D_4 Face Code**
cells	cubic	cubic	16-cells
qubit cell	squares	squares	triangles
parameters	$[[6L^4, 6, L^2]]$	e.g. $[[96, 6, 8]]$, $[[270, 6, 15]]$	$[[16L^4, 6, 2L^2]]$
qubits at equal d	$6d^2$	about $1.2d^2$ to $1.5d^2$	$4d^2$
X-check weight	6	6	**4**
Z-check weight	6	6	8
checks per qubit	4 + 4	4 + 4	**3 + 3**
distance known for	all L	listed examples	all L

Comparing instead at equal side length would suggest the D_4 code spends 8/3 times as many qubits. That comparison is misleading, because at the same L, the D_4 code also has twice the distance. At equal distance, it uses two thirds as many qubits as the standard hypercubic code, with smaller X-checks and fewer checks per qubit in both sectors. What it pays for this is a weight-8 Z-sector. The largest check in the code grows from 6 to 8. On hardware where cost is set by the largest check rather than by the qubit count, that is the number to weigh.

The rotated codes do better on qubit count, by a factor of about three. The two ideas are independent. Rotation changes how the torus is closed up, while the D_4 face code changes the cells. The D_4 face code on an L^4 torus is the analogue of the unrotated hypercubic code, and closing up the 16-cell honeycomb along a tilted lattice is an open problem (Section 9). Whether the two gains combine is not known.

The Weight-8 Checks and Hook Errors

A bigger check takes a longer circuit to measure, and a longer circuit gives a single fault more chances to spread onto data qubits. This is the natural worry about a weight-8 sector; so, we checked it.

Measure a weight-w Z-check by preparing an ancilla in $|0\rangle$, applying one CNOT from each data qubit, and measuring the ancilla. A Z fault on the ancilla after k of the CNOTs travels back through the remaining $w - k$ and lands as a Z error on that suffix of the data qubits. One fault can therefore produce a data error of weight up to $w - 1$: up to 7 here. These are the hook errors of Ref. [1].

What matters is not the raw weight but the smallest weight the error can be reduced to using the stabilizers, since two errors differing by a stabilizer are the same error. Adding the whole check turns a suffix of length s into the complementary prefix of length $w - s$; so, the long suffixes are cheap and the middle one is the worst. Running every suffix through an exact minimum-weight coset search (script `hooks.py`; the code is translation invariant; so, one check of each type settles it) gives Table 3.

https://doi.org/10.3390/quantum8040116

Table 3. Hook errors: the smallest weight to which each suffix of a check circuit reduces modulo the stabilizers ($L = 4$).

Suffix Length	**1**	**2**	**3**	**4**	**5**	**6**	**7**
weight-8 Z-check, reduced to	1	2	3	**4**	3	2	1
weight-4 X-check, reduced to	1	**2**	1				

So one fault behaves like a weight-2 error in the X sector and a weight-4 error in the Z sector, and the circuit distances are at least:

$$\frac{d_X}{2} = \frac{2L^2}{2} = L^2, \qquad \frac{d_Z}{4} = \frac{4L^2}{4} = L^2.$$

The two sectors come out equal. The Z sector has twice the distance and twice the hook weight, and the factors cancel; so, the weight-8 checks are no worse in this respect than the weight-4 ones. In this count, both circuit distances come out at $L^2 = d/2$; for the hypercubic code, the same count gives $d/3$, as shown next.

The same count for the hypercubic code.

Running the same search on the hypercubic code at $L = 4$ (also in `hooks.py`), the weight-6 checks of both types reduce their suffixes to weights $1, 2, 3, 2, 1$. One fault there acts like a weight-3 error, and the counting bound is $L^2/3 = d/3$ in each sector. For the D_4 face code, it is $L^2 = d/2$ in each sector. So, by this measure, the weight-8 checks do not put the D_4 code behind the hypercubic one; at equal distance, it comes out ahead, $d/2$ against $d/3$.

Correlated errors from the weight-8 circuits.

A hook is itself a correlated error. One fault on the ancilla of a weight-8 check spreads to as many as 7 data qubits at once, against 5 for a weight-6 check. Its effect on decoding is set by its reduced weight, which is what Table 3 lists: at worst, 4 here against 3 for the hypercubic checks, on a distance that is twice as large. Three further effects of the longer circuit are not captured by that table.

(i) *Several faults on one ancilla:* Z faults on the same ancilla after k_1 and after k_2 CNOTs leave errors on the data qubits between those two positions, which is always a contiguous segment of the CNOT order. Every such segment of a weight-8 check reduces to weight at most 4 (`hooks.py`); so, two faults never do more than one worst-case fault.

(ii) *Faults that also flip the measured outcome:* These are syndrome errors, and the four-dimensional structure catches them. Because $\partial\partial = 0$, the lit Z-checks at every vertex and the lit X-checks in every 16-cell must come in even numbers; so, a wrong outcome shows up as a broken loop. This is the redundancy on which single-shot decoding of four-dimensional codes relies [7,28].

(iii) *Time and gate count:* A weight-8 check needs at least eight rounds of CNOTs, which allows for more idle errors. This is not a cost the hypercubic code avoids: there each data qubit sits in eight checks; so, measuring all checks at once also needs at least 8 rounds, and measuring the two types one after the other takes $6 + 6 = 12$ rounds there and $4 + 8 = 12$ here. In gate count, the D_4 code is lighter. Each qubit takes part in six CNOTs per syndrome-extraction cycle against eight, and at equal distance the total is $24d^2$ CNOTs per cycle against $48d^2$.

Two limits remain. The circuit distances above are lower bounds from counting, not measured values, and the order of the CNOTs was not optimized. A good order, as for the surface code [35], or flag qubits [36,37], can only lower the effective hook weight. And none of this is a threshold, which needs a decoder and a full simulation with circuit-level noise.

9. Open Problems

Rotation: Closing up the 16-cell honeycomb along a tilted sublattice of D_4, as Aasen et al. do for the hypercubic lattice [14], may lower the qubit count below $4d^2$. The shadow argument of Section 5 should adapt, since it only uses the area of a cross-section.

The classes with no flat operator: Their smallest logical operators weigh at least $2L^2$, and the search at $L = 4$ finds operators of weight 78, 116 and 126 in three independent directions. Identifying them would complete the picture of the code's logical structure.

Thermodynamics: Loop-only excitations are the structure that self-correction needs, but the energy barrier, the memory time and the behavior within each logical sector (Section 6) deserve the treatment the hypercubic code has received [1,2,18].

The optimal thresholds: Computing p_X and p_Z from the random-plaquette models of Section 7 would test the relation $H(p_X) + H(p_Z) = 1$.

Decoding: A decoder matched to the triangular geometry, where a single error lights 3 checks of each type, is the practical next step. The hypercubic four-dimensional toric code already has several: local decoders based on Toom's rule [38], a renormalization-group decoder [39], neural network decoders [40], and the sweep decoder [27]. The first three are written for the hypercubic lattice, while the sweep rule is defined on any locally Euclidean lattice; none has been run on this honeycomb. There is also a three-dimensional starting point. By Section 7, the slices carry the rhombic dodecahedral geometry, and a sweep decoder with a proven threshold already exists there [26]. By the counting bound of Section 8, the weight-8 checks cost no more at circuit level than the weight-4 ones; so, they are not the obstacle. What is still missing is a measured threshold. Adapting the sweep decoder and running it under a depolarizing gate model would turn the qubit counts of Section 8 into claims about performance.

Single-shot decoding through a slice: Given the code's place in the subsystem toric code correspondence [11,12], a dimension-jump protocol [29] in which the four-dimensional memory hands logical information to a three-dimensional code along a slice looks natural. Making it explicit is the most interesting problem this construction poses.

10. Conclusions

Filling the gaps of the D_4 lattice gives a honeycomb of regular 16-cells, and putting qubits on its triangles gives a four-dimensional toric code with parameters

$$[[\, 16L^4,\ 6,\ 2L^2 \,]]$$

for every even $L \geq 4$, with $d_X = 2L^2$ and $d_Z = 4L^2$ exactly: uniform weight-4 X-checks, which is the smallest possible, weight-8 Z-checks, three checks of each type per qubit, and no point-like excitations in either sector. The code had been proposed, in its dual form on the 24-cell honeycomb [14]; its parameters are what this paper adds. The distance follows from the observation that a logical operator must shadow a whole cross-section of the torus, while a single triangle can shadow only an area of $1/2$; a close-packed plane of the lattice meets that bound exactly, tiling the cross-section once with nothing wasted, and a flat plane crossing the lattice does the same in the other sector. Because the distance is twice the hypercubic value at equal side length, the code reaches a given distance with $4d^2$ physical qubits where the standard hypercubic code needs $6d^2$; rotated hypercubic codes need fewer [14], and rotating this code is the natural next step. Everything needed to reproduce these statements is in the published scripts and runs in minutes on a laptop.

Funding: This research received no external funding.

Data Availability Statement: The original data presented in the study are openly available in GitHub at https://github.com/raghu91302/ssmtheory/blob/main/d4_toric_code_scripts.zip (accessed on 29 October 2026), as a single archive of the scripts that verify the results. Apart from `projection.py`, which is a local computation, and the helper `satdist.py`, each script rebuilds the lattice from scratch at $L = 4$ (and $L = 6$ where stated) and prints its results. `verify_all.py` checks every structural number in this paper, including the cell counts, the incidence structure, CSS validity, the parameters of Theorem 1, both shadow constants, the close-packed plane, the flat sheets, the slice property and Theorem 5(a), printing a pass or fail for each; a few minutes. `projection.py` computes the two shadow constants of Lemma 1 in exact arithmetic and needs only the standard library; seconds. `zsheet.py` builds the Z-type operator of Theorem 4 at $L = 4$ and $L = 6$, checks its weight, that it commutes with every X-check and that it is nontrivial, and checks the counting step of the proof; standard library only, seconds. `supplementary.py` reproduces the weight-4 scan of Section 4, the greedy check on d_Z, and the search over classes with no flat operator; about ten minutes. `satdist.py` holds the $\mathbb{F}_2$ helpers and the SAT encoding used by several of the others. `slice3d_distance.py` is the brute-force search on the slice code in both sectors, and `hooks.py` reproduces the hook-error table of Section 8, the segment check, and the same table for the hypercubic code; a few minutes each. `lemmas.py` checks the flat sheets and the classes they span, `planes.py` checks the close-packed planes at $L = 4$ and $L = 6$, and `make_figs.py` regenerates both figures. Some scripts need `numpy`, and `make_figs.py` also needs `matplotlib`; only `slice3d_distance.py` and `hooks.py` need `python-sat`.

Conflicts of Interest: Raghu Kulkarni is an employee of SSMTheory Group, IDrive Inc. The views expressed in this article are those of the author and do not necessarily represent the views of IDrive Inc.

References

1. Dennis, E.; Kitaev, A.; Landahl, A.; Preskill, J. Topological quantum memory. *J. Math. Phys.* **2002**, *43*, 4452. [CrossRef]
2. Alicki, R.; Horodecki, M.; Horodecki, P.; Horodecki, R. On thermal stability of topological qubit in Kitaev's 4D model. *Open Syst. Inf. Dyn.* **2010**, *17*, 1. [CrossRef]
3. Brown, B.J.; Loss, D.; Pachos, J.K.; Self, C.N.; Wootton, J.R. Quantum memories at finite temperature. *Rev. Mod. Phys.* **2016**, *88*, 045005. [CrossRef]
4. Error Correction Zoo. (2, 2) Loop Toric Code. Available online: https://errorcorrectionzoo.org/c/4d_surface (accessed on 29 October 2026).
5. Guth, L.; Lubotzky, A. Quantum error correcting codes and 4-dimensional arithmetic hyperbolic manifolds. *J. Math. Phys.* **2014**, *55*, 082202. [CrossRef]
6. Londe, V.; Leverrier, A. Golden codes: Quantum LDPC codes built from regular tessellations of hyperbolic 4-manifolds. *arXiv* **2017**, arXiv:1712.08578.
7. Breuckmann, N.P.; Londe, V. Single-shot decoding of linear rate LDPC quantum codes with high performance. *IEEE Trans. Inf. Theory* **2022**, *68*, 272. [CrossRef]
8. Brell, C.G. A proposal for self-correcting stabilizer quantum memories in 3 dimensions (or slightly less). *New J. Phys.* **2016**, *18*, 013050. [CrossRef]
9. Bombín, H.; Chhajlany, R.W.; Horodecki, M.; Martin-Delgado, M.A. Self-correcting quantum computers. *New J. Phys.* **2013**, *15*, 055023. [CrossRef]
10. Albert, V.V.; Faist, P. Handbook of Error-Correcting Codes. *arXiv* **2026**, arXiv:2606.11484.
11. Jochym-O'Connor, T.; Yoder, T.J. Four-dimensional toric code with non-Clifford transversal gates. *Phys. Rev. Res.* **2021**, *3*, 013118. [CrossRef]
12. Kubica, A.; Vasmer, M. Single-shot quantum error correction with the three-dimensional subsystem toric code. *Nat. Commun.* **2022**, *13*, 6272. [CrossRef] [PubMed]
13. Error Correction Zoo. D_4 Hyper-Diamond Lattice. Available online: https://errorcorrectionzoo.org/c/dfour (accessed on 29 October 2026).
14. Aasen, D.; Haah, J.; Hastings, M.; Wang, Z. Geometrically enhanced topological quantum codes. *arXiv* **2025**, arXiv:2505.10403.
15. Google Quantum AI and collaborators (Acharya, R. *et al.*) Quantum error correction below the surface code threshold. *Nature* **2025**, *638*, 920. [CrossRef] [PubMed]
16. Bravyi, S.; Cross, A.W.; Gambetta, J.M.; Maslov, D.; Rall, P.; Yoder, T.J. High-threshold and low-overhead fault-tolerant quantum memory. *Nature* **2024**, *627*, 778. [CrossRef] [PubMed]

17. Aasen, D.; Hastings, M.; Kliuchnikov, V.; Bello-Rivas, J.M.; Paetznick, A.; Chao, R.; Reichardt, B.W.; Zanner, M.; da Silva, M.P.; Wang, Z.; et al. A topologically fault-tolerant quantum computer with four dimensional geometric codes. *arXiv* **2025**, arXiv:2506.15130.
18. Bergamaschi, T.; Gheissari, R.; Liu, Y. Rapid mixing for Gibbs states within a logical sector: A dynamical view of self-correcting quantum memories. *arXiv* **2025**, arXiv:2507.10976.
19. Conway, J.H.; Sloane, N.J.A. *Sphere Packings, Lattices and Groups*, 3rd ed.; Springer: New York, NY, USA, 1999. [CrossRef]
20. Bombín, H.; Martin-Delgado, M.A. Homological error correction: Classical and quantum codes. *J. Math. Phys.* **2007**, *48*, 052105. [CrossRef]
21. Freedman, M.H.; Meyer, D.A.; Luo, F. $\mathbb{Z}_2$-systolic freedom and quantum codes. In *Mathematics of Quantum Computation*; Chapman and Hall/CRC: Boca Raton, FL, USA, 2002.
22. Kulkarni, R. A 67%-rate CSS code on the FCC lattice: $[[192, 130, 3]]$ from weight-12 stabilizers. *arXiv* **2026**, arXiv:2603.20294.
23. Musin, O.R. The kissing number in four dimensions. *Ann. Math.* **2008**, *168*, 1. [CrossRef]
24. Royer, B.; Singh, S.; Girvin, S.M. Encoding qubits in multimode grid states. *PRX Quantum* **2022**, *3*, 010335. [CrossRef]
25. Vasmer, M.; Browne, D.E. Three-dimensional surface codes: Transversal gates and fault-tolerant architectures. *Phys. Rev. A* **2019**, *100*, 012312. [CrossRef]
26. Vasmer, M.; Browne, D.E.; Kubica, A. Cellular automaton decoders for topological quantum codes with noisy measurements and beyond. *Sci. Rep.* **2021**, *11*, 2027. [CrossRef] [PubMed]
27. Kubica, A.; Preskill, J. Cellular-automaton decoders with provable thresholds for topological codes. *Phys. Rev. Lett.* **2019**, *123*, 020501. [CrossRef] [PubMed]
28. Bombín, H. Single-shot fault-tolerant quantum error correction. *Phys. Rev. X* **2015**, *5*, 031043. [CrossRef]
29. Bombín, H. Dimensional jump in quantum error correction. *New J. Phys.* **2016**, *18*, 043038. [CrossRef]
30. Wegner, F.J. Duality in generalized Ising models and phase transitions without local order parameters. *J. Math. Phys.* **1971**, *12*, 2259. [CrossRef]
31. Kogut, J.B. An introduction to lattice gauge theory and spin systems. *Rev. Mod. Phys.* **1979**, *51*, 659. [CrossRef]
32. Kogut, J.; Susskind, L. Hamiltonian formulation of Wilson's lattice gauge theories. *Phys. Rev. D* **1975**, *11*, 395. [CrossRef]
33. Takeda, K.; Nishimori, H. Self-dual random-plaquette gauge model and the quantum toric code. *Nucl. Phys. B* **2004**, *686*, 377. [CrossRef]
34. Takeda, K.; Sasamoto, T.; Nishimori, H. Exact location of the multicritical point for finite-dimensional spin glasses: A conjecture. *J. Phys. A* **2005**, *38*, 3751. [CrossRef]
35. Tomita, Y.; Svore, K.M. Low-distance surface codes under realistic quantum noise. *Phys. Rev. A* **2014**, *90*, 062320. [CrossRef]
36. Chao, R.; Reichardt, B.W. Quantum error correction with only two extra qubits. *Phys. Rev. Lett.* **2018**, *121*, 050502. [CrossRef] [PubMed]
37. Chamberl, C.; Beverl, M.E. Flag fault-tolerant error correction with arbitrary distance codes. *Quantum* **2018**, *2*, 53. [CrossRef]
38. Breuckmann, N.P.; Duivenvoorden, K.; Michels, D.; Terhal, B.M. Local decoders for the 2D and 4D toric code. *Quantum Inf. Comput.* **2017**, *17*, 181. [CrossRef]
39. Duivenvoorden, K.; Breuckmann, N.P.; Terhal, B.M. Renormalization group decoder for a four-dimensional toric code. *IEEE Trans. Inf. Theory* **2019**, *65*, 2545. [CrossRef]
40. Breuckmann, N.P.; Ni, X. Scalable neural network decoders for higher dimensional quantum codes. *Quantum* **2018**, *2*, 68. [CrossRef]

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.