EXACTLY FOUR TWELVE-QUBIT CONSTANT-EXCITATION CSS CODES

KOREA SUPERINTELLIGENCE LABS

Abstract. A quantum code is *constant excitation* (CE) if every computational basis ket occurring in its code space has the same Hamming weight; such a code acquires only a global phase under collective coherent Z-rotations. Lai, Liou and Ouyang generalise the CSS construction to produce CE codes by adding a global shift and exhibit a $[[12,1,3]]$ example, and twelve is the least length at which a CE *CSS* code with one logical qubit exists. We classify that length. Up to permutation of the qubits — and up to the redundancy in the global shift, which is a freedom in the presentation rather than a symmetry — there are *exactly four* $[[12,1,d\geq 3]]$ CE CSS codes. We exhibit the four: their X-type codes C_1 have dimensions $5,4,4,3$ and weight enumerators $0^1 4^{15} 8^{15} 12^1$, $0^1 4^7 8^7 12^1$, $0^1 4^3 6^8 8^3 12^1$ and $0^1 4^3 8^3 12^1$, and all four have $d_X = 4$ and $d_Z = 3$, so the distance is decided by d_Z in every class. The weight enumerator of C_1 is an invariant of the equivalence, which is what makes the four provably four; the code of Lai, Liou and Ouyang lies in the first class. Two structural by-products: no $[[11,1,d\geq 3]]$ CE CSS code exists, and this now has a proof that runs no search at all, so the most expensive computation of the earlier nonexistence work is no longer needed; and every $[[13,1,d\geq 3]]$ CE CSS code is a padding of a twelve-qubit one. The classification rests on one exhaustive computation: at the single shift the structure theory leaves, 46 350 pairs (C_1,φ) survive the distance conditions, and each carries a permutation certificate identifying it with one of the four. Every theorem below is machine-checked in Lean 4; the exceptions — the passage from the quantum-mechanical definitions to the finite ones, and two clauses in the exhibition of the four codes that the classification itself does not use — are marked where they occur.

1. Introduction

Let $\theta\in\mathbb{R}$ and consider the collective rotation $\exp\bigl(-i\theta\sum_{j=1}^{n}Z_j\bigr)$: the error an n-qubit register picks up when every qubit is dephased by the same drifting clock. A superposition of computational basis kets all of Hamming weight w is an eigenvector of that operator for every θ, so it acquires only a global phase. Following [7] we call such a state a *w-CE state*, and a quantum code whose code space consists entirely of w-CE states a *w-CE code*; a CE code is immune to collective coherent noise. CE codes go back to Plenio, Vedral and Knight [9], and have been studied since as the natural carrier for dual-rail concatenation [8] and for balanced weight-2 Z-stabilizers [4].

Lai, Liou and Ouyang [7, Appendix A 1, Eq. (A1)] generalise the CSS construction so that it produces CE codes. Given linear codes $C_2\subset C_1\subseteq\mathbb{F}_2^n$ and a vector $y\in\mathbb{F}_2^n$, the logical basis states are

$$|j\rangle_L \;=\; |C_2|^{-1/2}\sum_{c\in C_2}\bigl|\,y+x^{(j)}+c\,\bigr\rangle, \tag{1}$$

where $x^{(j)}$ runs over coset representatives of C_2 in C_1; the number of logical qubits is $k=\dim C_1-\dim C_2$, and $y=0$ recovers an ordinary CSS code. The X-type stabilizers are X^g for $g\in C_2$ and the Z-type stabilizers are $(-1)^{h\cdot y}Z^h$ for $h\in C_1^{\perp}$, so the *global shift* y enters only through signs. We call a code of this shape a *CE CSS code*; it is $\mathrm{wt}(y)$-CE exactly when $\mathrm{wt}(y+v)=\mathrm{wt}(y)$ for every $v\in C_1$.

Section IV of [7] settles the small lengths for $k=1$ and leaves one gap. Their Lemma 4 rules out $[[n,1,3]]$ CE CSS codes for $n\leq 9$; their Lemma 5 rules out a $[[10,1,3]]$ one only for the dual-rail construction of their Theorem 3, and they write that this "does not rule out the existence of such a code" and that "its existence remains an open problem". The companion paper [10] closes it: there is no $[[n,1,d\geq 3]]$ CE CSS code for $n\leq 11$, any shift and any construction, so with the $[[12,1,3]]$ code printed in [7, Appendix A 6] twelve is the exact minimal length. That paper counts nodes and functionals, not codes, and its closing section names the question left over:

> "At $n=12$ we do not know how many inequivalent $[[12,1,3]]$ CE CSS codes there are"

([10], § *What is left open*, second version; the sentence continues with a second clause about $n = 8$).

This paper answers it. Two codes are equivalent when a permutation of the qubits carries the code space of one onto the code space of the other; since y and $y + c$ for $c \in C_1$ index the same set of computational basis kets, the shift is only defined modulo C_1, and that redundancy is part of the relation (Definition 2.2).

Theorem (Theorem 5.2)**.** Up to permutation of the twelve qubits and the redundancy in the global shift there are exactly four $[[12, 1, d \geq 3]]$ constant-excitation CSS codes. Explicitly, the four codes $K_\mathrm{a}, K_\mathrm{b}, K_\mathrm{c}, K_\mathrm{d}$ of Table 1 are pairwise inequivalent, and every $[[12, 1, d \geq 3]]$ CE CSS code — any global shift, any construction — is equivalent to one of them.

The four are separated by an invariant, the weight enumerator of C_1, and are already separated by its entries in weights 4 and 6, which are $(15, 0)$, $(7, 0)$, $(3, 8)$ and $(3, 0)$. All four have $d_X = 4$ and $d_Z = 3$ exactly, so all four are $[[12, 1, 3]]$ codes whose distance is decided by the Z-type condition. The published code of [7] lies in the first class (Proposition 6.1); the other three are codes that paper does not exhibit, and two of them do not even have the same $\dim C_1$.

What is not claimed. Three delimitations, each of which a reader could otherwise supply for us in the wrong direction.

The class is CE CSS*, not CE.* Distance-3 constant-excitation *stabilizer* codes that are not CSS are shorter: the $[[8, 1, 3]]$ code of Plenio, Vedral and Knight [9, Eqs. (7)–(8)] and the $[[8, 1, 3]]$ code of Chang [1, Eq. (8), v5] are both 4-CE and both non-CSS, as recomputed from their printed data in [10]: in each case the stabilizer has order 128, and its pure-X and pure-Z elements generate a subgroup of rank 4 out of 7. Nothing here counts or classifies those.

The group is qubit permutations, not ΠH*.* Koh et al. [6] enumerate CSS codes up to ΠH *equivalence* — qubit permutation together with the optional global Hadamard $H^{\otimes n}$ — and their Table VI, captioned "the number of distinct CSS codes of parameters (n, k, d) up to equivalence under qubit permutations and global Hadamard duality (ΠH equivalence) found through exhaustive code enumeration", records 14 416 codes with parameters $[[12, 1, 3]]$. That number and ours are not comparable in either direction. $H^{\otimes n}$ carries $Z^{\otimes n}$ to $X^{\otimes n}$ and does not preserve constant excitation; and their enumeration cannot see the CE condition at all, since constant excitation lives entirely in the signs, that is, in the shift y, and the words *excitation* and *dual-rail* do not occur anywhere in that paper. Their table is the nearest published count in this cell, and nothing here contradicts it.

Thirteen and fourteen qubits. Every $[[13, 1, d \geq 3]]$ CE CSS code is a padding of a twelve-qubit one (Proposition 6.2), so nothing structurally new appears at thirteen; but the *number* of classes at $n = 13$ is a different question and is left open here, and $n = 14$ is untouched. See Section 9.

On the novelty. A search recorded on 2026-08-30 and re-run on 2026-09-03 found no classification of CE codes anywhere. The arXiv metadata set for the phrases "constant excitation" and "constant-excitation" is twenty-two papers, six of them about quantum codes: [8], [1], [7] and arXiv:1809.09801, arXiv:2509.20545, arXiv:2510.21030. In the e-print sources of those six, and of [4] — which uses the phrase in its body but not in its abstract, so a metadata query does not return it and twenty-two is a floor — the strings *inequivalent*, *up to permutation* and *equivalence class* occur zero times, and the three occurrences of *classif* are about classifying noise types or stabilizer groups, never codes. The Error Correction Zoo's entry for the twelve-qubit code [3] makes no uniqueness or counting claim, and no entry of that catalogue states a count or a classification of CE codes at all, although it does record such counts where they exist: its qubit-stabilizer entry reads "There are two $[[8, 1, 3]]$ self-dual non-CSS codes". The small-code classification literature that does exist — Cross and Vandeth [2] for $n \leq 9$ under local Clifford permutation equivalence, Khesin and Li [5] for $n \leq 5$ — is not CE-aware, and local Clifford equivalence in particular does not preserve constant excitation. The nearest published number is the 14 416 of [6] discussed above.

Method. Section 2 fixes the finite object and the equivalence. Section 3 is the structure theory, and it does two things: it shows that a CE CSS code whose C_1 has full support has even length $n = 2\,\mathrm{wt}(y)$ — a published identity of Hu, Liang, Rengaswamy and Calderbank [4], re-proved here because the reduction

it powers has to be machine-checked — and that a coordinate outside $\operatorname{supp} C_1$ may be deleted. Together these force every twelve-qubit code to have full support, hence $\operatorname{wt}(y) = 6$, hence to be equivalent to one with the single sorted shift $y = 1^6 0^6$. Section 4 exhibits the four representatives. Section 5 describes the exhaustive computation at that one shift and the theorem that makes it a classification. Section 6 places the published code and settles the structure at thirteen qubits, and Section 7 records the negative controls. Section 8 says what is machine-checked and what is not, and Section 9 what is left open.

2. Constant-excitation CSS codes as finite combinatorics

Identify $\mathbb{F}_2^n$ with $\{0, 1, \ldots, 2^n - 1\}$ by binary expansion, so that coordinate i of v is bit i of v (qubit $i+1$ of the register), addition in $\mathbb{F}_2^n$ is bitwise XOR $\oplus$ and the coordinatewise product is bitwise AND $\wedge$. Write $\operatorname{wt} v$ for the Hamming weight, $\operatorname{supp} v$ for the support and $\langle u, v\rangle = \operatorname{wt}(u \wedge v) \bmod 2$ for the binary inner product.

The computational basis kets occurring anywhere in the code space (1) are exactly those indexed by $y + C_1$, because C_1 is the union of the cosets $x^{(j)} + C_2$. Hence the code is $\operatorname{wt}(y)$-CE if and only if $\operatorname{wt}(y \oplus v) = \operatorname{wt}(y)$ for every $v \in C_1$. For $k = 1$ let $\varphi\colon C_1 \to \mathbb{F}_2$ be the nonzero linear functional with $\ker \varphi = C_2$; then $C_1 \setminus C_2 = \{v \in C_1 : \varphi(v) = 1\}$, and since a linear form on C_1 vanishing on the index-two subspace $\ker \varphi$ is 0 or φ,

$$C_2^\perp \setminus C_1^\perp \;=\; \big\{\, u \in \mathbb{F}_2^n \;:\; \langle u, c\rangle = \varphi(c) \text{ for all } c \in C_1 \,\big\}.$$

The two halves of $d \geq 3$ therefore read as in the following definition, which is the object every theorem below is about. The passage just made, from (1) to Definition 2.1, is an ordinary argument about states in $(\mathbb{C}^2)^{\otimes n}$ and is not machine-checked; Section 8 lists it with the other exceptions.

Definition 2.1. Let $n \geq 0$ and $y \in \mathbb{F}_2^n$. An $[[n, 1, d \geq 3]]$ *CE CSS code with shift* y is a pair (C_1, φ) consisting of a linear subspace $C_1 \subseteq \mathbb{F}_2^n$ and a map $\varphi\colon C_1 \to \mathbb{F}_2$ additive on C_1, subject to

- (K) $\varphi(v) = 1$ for some $v \in C_1$ (so $k = \dim C_1 - \dim \ker \varphi = 1$);
- (CE) $\operatorname{wt}(y \oplus v) = \operatorname{wt}(y)$ for every $v \in C_1$ (constant excitation, with $w = \operatorname{wt} y$);
- (d_X) $\operatorname{wt}(v) \geq 3$ for every $v \in C_1$ with $\varphi(v) = 1$;
- (d_Z) for every $u \in \mathbb{F}_2^n$ with $1 \leq \operatorname{wt}(u) \leq 2$ there is $v \in C_1$ with $\langle u, v\rangle \neq \varphi(v)$.

We write $\mathcal{C} = (C_1, \varphi, y)$ and say $\mathcal{C}$ has *full support* when every one of the n coordinates is moved by some codeword, i.e. for each $i < n$ there is $v \in C_1$ with $v_i = 1$.

Condition (d_X) says $\min\{\operatorname{wt} v : v \in C_1 \setminus C_2\} \geq 3$ and (d_Z) says $\min\{\operatorname{wt} u : u \in C_2^\perp \setminus C_1^\perp\} \geq 3$, so together they say the stabilizer distance of (1) is at least 3. This is the true distance, not the quantity $\min\{d_1, d_2^\perp\}$ that [7, Appendix A 1] equates with it and that is in general only a lower bound; taking the weaker hypothesis makes every nonexistence statement below stronger, since degenerate codes are refuted too, and it makes the classification a classification of more objects. We write d_X and d_Z for the two minima themselves, so that a code with $d_X = 4$ and $d_Z = 3$ has distance exactly 3.

Definition 2.2 (the equivalence)**.** Let $\mathcal{C} = (C_1, \varphi, y)$ and $\mathcal{D} = (C_1', \varphi', y')$ be $[[n, 1, d \geq 3]]$ CE CSS codes. They are *equivalent*, written $\mathcal{C} \sim \mathcal{D}$, when there is a permutation π of the n coordinates with

- (E1) $\pi(C_1) = C_1'$;
- (E2) $\varphi'(\pi v) = \varphi(v)$ for every $v \in C_1$;
- (E3) $\pi(y) \oplus y' \in C_1'$.

Conditions (E1) and (E2) say that π carries C_1 to C_1' and C_2 to C_2', up to which coset is named $|0\rangle_L$. Condition (E3) is not an extra symmetry: y and $y \oplus c$ for $c \in C_1$ index the same set of computational basis kets and so present the same code space, and (E3) is exactly that redundancy. With (E1)–(E3) in force, π carries the code space of $\mathcal{C}$ onto the code space of $\mathcal{D}$.

In the formal development a permutation is carried as a list of transpositions of coordinates, acted on the right; composition is concatenation of lists and inversion is reversal, so Proposition 3.4(1) below costs nothing beyond what a single transposition already knows. The verbatim formal statement of Definition 2.2, and of every theorem pinned below, is reproduced in Appendix A.

3. The structure at odd and at twelve qubits

Theorem 3.1 (the excitation is forced; known)**.** *Let $\mathcal{C} = (C_1, \varphi, y)$ be an $[[n, 1, d \geq 3]]$ CE CSS code with full support. Then $n = 2\,\mathrm{wt}(y)$. In particular a full-support CE CSS code has even length and its excitation is $n/2$.*

This is published, and we cite it as such: Theorem 3 of Hu, Liang, Rengaswamy and Calderbank [4] states, for a stabilizer code preserved by transversal $\pi/2^l$ Z-rotations at every $l \geq 2$, that the support of each X-component splits into components Γ_k on which "N_k is even and $w_H(y_k) = N_k/2$", and their Corollary 12 identifies that condition with constant excitation for CSS codes; Theorem 3.1 is the case $\Gamma = [n]$. We re-prove it rather than quote it because everything downstream — in particular the reduction to a single shift, which is what makes the classification affordable — has to be machine-checked, and because the proof is three lines.

Proof sketch. Let $D = y \oplus C_1$, the set of basis kets occurring in the code space; $|D| = |C_1| > 0$. Count $\sum_{u \in D} \mathrm{wt}(u)$ twice. Constant excitation gives $|D| \cdot \mathrm{wt}(y)$. Coordinate by coordinate: for each $i < n$ pick $c \in C_1$ with $c_i = 1$, which full support supplies; then $u \mapsto u \oplus c$ is an involution of D exchanging $\{u : u_i = 0\}$ with $\{u : u_i = 1\}$, so exactly half of D has bit i set, and summing over i gives $n|D|/2$. Divide by $|D|$. □

The full-support hypothesis is not decoration. Example 6 of [4] is a $[[5, 1, 2]]$ CE CSS code of odd length with excitation $3 \neq 5/2$; it is a padded $[[4, 1, 2]]$ code, which is what the next statement says in general.

Proposition 3.2 (shortening)**.** *Let $\mathcal{C} = (C_1, \varphi, y)$ be an $[[m+1, 1, d \geq 3]]$ CE CSS code and suppose no codeword of C_1 touches some coordinate $i \leq m$. Then deleting that coordinate produces an $[[m, 1, d \geq 3]]$ CE CSS code, with C_1 and φ unchanged and the shift truncated. Consequently, if n is odd then every $[[n, 1, d \geq 3]]$ CE CSS code arises in this way from one of length $n - 1$.*

Proof sketch. A transposition of coordinates transports the whole structure, so we may assume $i = m$. Every clause of Definition 2.1 survives truncation: wt reads only the low m bits of anything in C_1; $y \oplus v$ has the same bit m as y for $v \in C_1$, so (CE) is unchanged; (d_X) only loses a summand that is zero; and an error of weight at most 2 on m coordinates extends by a zero bit, which changes neither its weight nor any inner product with a codeword. For the consequence, apply Theorem 3.1: at odd n full support would give $n = 2\,\mathrm{wt}(y)$, so some coordinate is dead. □

Corollary 3.3. *There is no $[[11, 1, d \geq 3]]$ CE CSS code, for any global shift. The proof runs no search at eleven qubits: it shortens to length 10 and appeals to the nonexistence there* [10].

This is worth a sentence beyond its statement. The nonexistence at $n = 11$ is a theorem of [10], proved there by an exhaustive sweep over 1 504 472 pairs (C_1, φ) — some eight minutes of kernel-external evaluation, and by a wide margin the largest of that paper's searches for one logical qubit. Corollary 3.3 replaces it by the paragraph above, and the replacement is audited rather than asserted: the axiom set of the formal statement of Corollary 3.3 contains exactly one evaluation axiom, the one belonging to the $n = 10$ sweep, and the $n = 11$ sweep does not occur in it (Section 8). The same argument makes the $n = 9$ sweep of [10] redundant.

Proposition 3.4 (the equivalence, and its invariant)**.** *Fix n.*

1. *$\sim$ is an equivalence relation on $[[n, 1, d \geq 3]]$ CE CSS codes.*
2. *If $\mathcal{C} = (C_1, \varphi, y)$ is a code and $c \in C_1$, then $(C_1, \varphi, y \oplus c)$ is a code and is equivalent to it.*
3. *The weight enumerator of C_1 — the function $w \mapsto \#\{v \in C_1 : \mathrm{wt}\, v = w\}$ — is an invariant of $\sim$.*

Proof sketch. (1) Reflexivity is the identity permutation with $\pi(y) \oplus y = 0 \in C_1$; transitivity composes the permutations and uses that C_1' is closed under $\oplus$ to combine the two instances of (E3); symmetry inverts. (2) Only (CE) has to be rechecked, and $\mathrm{wt}(y \oplus c \oplus v) = \mathrm{wt}(y)$ for $v \in C_1$ because $c \oplus v \in C_1$; the equivalence is witnessed by the identity permutation. (3) A permutation of coordinates is injective and weight-preserving, so by (E1) it restricts to a weight-preserving bijection $C_1 \to C_1'$. □

Proposition 3.5 (twelve qubits is a single-shift problem)**.** *Every* $[[12, 1, d \geq 3]]$ *CE CSS code has full support; hence its shift has weight exactly* 6*; hence it is equivalent to one whose shift is* $y = 1^6 0^6$*, that is,* $y = 63$ *under the identification of Section 2.*

Proof sketch. A dead coordinate would shorten the code to an $[[11, 1, d \geq 3]]$ one by Proposition 3.2, which Corollary 3.3 forbids; so the support is full and Theorem 3.1 gives $12 = 2\,\mathrm{wt}(y)$. For the last clause, if y has a 0 before a 1 then transposing those two coordinates strictly decreases y as an integer and transports the code, so a strong induction terminates at the sorted shift $2^{\mathrm{wt}\,y} - 1 = 63$; the induction is run carrying the permutation, so it produces an equivalence and not merely a code. □

Independently of the proof, an exhaustive sweep over all thirteen sorted shifts at $n = 12$ finds accepting pairs at $\mathrm{wt}(y) = 6$ only, and none at any of the other twelve (Section 5).

4. Four codes

Table 1 gives four $[[12, 1, 3]]$ CE CSS codes, all with the sorted shift $y = 1^6 0^6$. Each is presented by a generator matrix of C_1 — rows written with coordinate 1 leftmost — and by a vector $z \in \mathbb{F}_2^{12}$ realising the functional, so that $\varphi(v) = \langle z, v\rangle$ and $C_2 = \ker\varphi = \{v \in C_1 : \langle z, v\rangle = 0\}$. Every functional on a subspace of $\mathbb{F}_2^{12}$ extends to one on $\mathbb{F}_2^{12}$, so this presentation is no loss. The integers in the last column are the same rows read as binary numbers, least significant bit first; they are the literal data of the formal development.

code	generator matrix of C_1	$\dim C_1$	z	rows as integers

K_{a}	`100001100001`
	`100010100010`
	`100100100100`
	`101000101000`
	`110000110000`
		5	`111000000000`	$2145, 1105, 585,$
				$325, 195;\ \ z = 7$

K_{b}	`000011100001`
	`100100010010`
	`101000010100`
	`110000011000`
		4	`110010000000`	$2160, 1161, 645,$
				$387;\ \ z = 19$

K_{c}	`101001110001`
	`101010110010`
	`001100100100`
	`110000011000`
		4	`011010000000`	$2277, 1237, 588,$
				$387;\ \ z = 22$

K_{d}	`000011100001`
	`001100000110`
	`110000011000`
		3	`101010000000`	$2160, 1548, 387;$
				$z = 21$

Table 1. The four classes, each with shift $y =$ `111111000000` and $\varphi = \langle z, \cdot\rangle$.

Theorem 4.1 (the four representatives)**.** *Each of* $K_{\mathrm{a}}, K_{\mathrm{b}}, K_{\mathrm{c}}, K_{\mathrm{d}}$ *satisfies Definition 2.1 at* $n = 12$ *with* $y = 1^6 0^6$*. Their* C_1 *have weight enumerators*

$$0^1 4^{15} 8^{15} 12^1, \qquad 0^1 4^7 8^7 12^1, \qquad 0^1 4^3 6^8 8^3 12^1, \qquad 0^1 4^3 8^3 12^1$$

respectively; all four have $d_X = 4$ *and* $d_Z = 3$*, so all four are* $[[12, 1, 3]]$ *codes whose distance is decided by the Z-type condition; and they are pairwise inequivalent.*

Proof sketch. Membership in Definition 2.1 and the weight enumerators are finite checks over C_1, which has at most 32 elements, and over the 4096 vectors of $\mathbb{F}_2^{12}$ for (d_Z); all are run exhaustively. That $d_X \geq 4$

rather than ≥ 3 is the same finite check with the bound raised, and a weight-4 codeword with $\varphi = 1$ in each class — the vectors $585, 645, 387, 387$ of the four C_1 — gives $d_X = 4$ exactly. That $d_Z \leq 3$, hence $= 3$, is visible in the table: the vector z itself has weight 3 and induces φ on C_1 by definition, so it is an undetectable Z-type error of weight 3. Pairwise inequivalence is Proposition 3.4(3) together with four pairs of numbers: the weight-4 and weight-6 entries of the four enumerators are $(15, 0)$, $(7, 0)$, $(3, 8)$, $(3, 0)$, and the four pairs are distinct. □

Remark 4.2. *Outside the formal development.* The generator matrix of K_{a} has a description that can be read off it directly. Put $f_j = e_j + e_{j+6}$ for $1 \leq j \leq 6$, the six *rails* pairing coordinate j with coordinate $j + 6$. The five printed rows are $f_1 + f_6$, $f_1 + f_5$, $f_1 + f_4$, $f_1 + f_3$, $f_1 + f_2$, so $C_1(K_{\mathrm{a}}) = \{\sum_{j \in S} f_j : S \subseteq \{1, \ldots, 6\},\ |S| \text{ even}\}$, whose weight enumerator is $\sum_{|S| \text{ even}} x^{2|S|} = 0^1 4^{15} 8^{15} 12^1$ as stated. Adding f_j to $y = 1^6 0^6$ moves one excitation from rail position j to rail position $j + 6$, which is why constant excitation holds; this is the dual-rail shape of the construction of [7], and Proposition 6.1 confirms that their code is in this class. We make no corresponding claim about the other three.

Remark 4.3. *Outside the formal development.* One may ask whether the group of Definition 2.2 should be enlarged by local X strings. Applying X^u to the code space of (C_1, φ, y) gives the code space of $(C_1, \varphi, y \oplus u)$, with the *same* C_1 and the same φ; the result is again a CE code exactly when $\mathrm{wt}(y \oplus u \oplus v) = \mathrm{wt}(y \oplus u)$ for all $v \in C_1$, which for $u \in C_1$ is automatic and is clause (E3). Two consequences, neither of which needs the question of which $u \notin C_1$ qualify to be settled. First, Theorem 5.2 quantifies over *every* shift, so any code obtained in this way is already in its list. Second, since C_1 is unchanged, the invariant of Proposition 3.4(3) still separates the four classes; so enlarging the group by local X strings can merge none of them, and the count stays four.

One tempting shortcut is *not* available and is worth recording, because it is easy to believe: it is not true that an X string outside C_1 always destroys constant excitation once C_1 has full support. For K_{a} the admissible strings — those u passing the displayed test $\mathrm{wt}(y \oplus u \oplus v) = \mathrm{wt}(y \oplus u)$ for all $v \in C_1$ — are exactly the 64 sums of the six rails of Remark 4.2, twice the 32 elements of C_1; for $K_{\mathrm{b}}, K_{\mathrm{c}}, K_{\mathrm{d}}$ there are 96, 64 and 216 of them against $|C_1| = 16, 16, 8$. (Only two of those four sets are groups: 96 and 216 are not powers of two.) So local X strings do move some codes to others, and the argument above, which does not use any such assertion, is what keeps the count at four.

5. The classification

By Proposition 3.5 it suffices to classify codes with the single shift $y = 1^6 0^6$. We do that by an exhaustive enumeration of the pairs (C_1, φ) carrying a certificate at every accepting pair. Three ingredients.

The space enumerated. Inclusion–exclusion for supports, $\mathrm{wt}(a \oplus b) + 2\,\mathrm{wt}(a \wedge b) = \mathrm{wt}\, a + \mathrm{wt}\, b$, turns clause (CE) into a containment: writing $V_y = \{c \in \mathbb{F}_2^n : \mathrm{wt}(c) = 2\,|\,\mathrm{supp}\, c \cap \mathrm{supp}\, y|\}$, condition (CE) says exactly $C_1 \subseteq V_y$. At $n = 12$, $y = 1^6 0^6$ the set V_y has $\sum_i \binom{6}{i}^2 = \binom{12}{6} = 924$ elements. The enumeration walks the subspaces of V_y depth-first through canonical increasing bases, exactly as in [10], whose completeness theorem — every subspace of V_y is reached at some node — is reused verbatim.

The node test. At a node whose span is a candidate C_1, the functionals that could complete it to a code are those satisfying (d_X) and (d_Z). Packing a functional on an r-dimensional span as an r-bit mask, the surviving masks are those that are nonzero, that vanish on every codeword of weight at most 2 — otherwise $d_X < 3$ — and that are not induced on the span by any error of weight 1 or 2 — otherwise $d_Z < 3$. In [10] the test was negative: a node passed when *no* mask survived, which is what a nonexistence proof needs. Here masks do survive, and the test is positive: every surviving mask must be certified.

Certificates. The object being certified is the *labelled subspace* $\mathrm{st}(C_1, \varphi) = \{\, 2v + \varphi(v) : v \in C_1 \,\}$, sorted as a list of integers, which determines the pair (C_1, φ) exactly since a label is a bit. A certificate is a permutation of the twelve coordinates, packed as at most eleven transpositions, together with a class index $c \in \{0, 1, 2, 3\}$. Writing st_c and $C_1^{(c)}$ for the labelled subspace and the X-type code of

the c-th representative, the check is the list equality $\pi\big(\mathrm{st}(C_1,\varphi)\big) = \mathrm{st}_c$ together with the membership $\pi(y) \oplus y \in C_1^{(c)}$. The certificates are data, supplied by an orbit walk written separately in C, indexed in a table of 65 536 buckets by a hash of the labelled subspace. *Nothing about them is trusted*: a wrong permutation, a wrong bucket or a wrong class index simply fails an equality and the search returns false. No canonical form under the group is ever computed, and this is what makes the computation affordable. At the fixed shift $y = 1^6 0^6$ the permutations Definition 2.2 allows out of a code are exactly the $\rho\sigma_c$ with ρ fixing y and $c \in C_1$, where σ_c exchanges the set bits of c below 6 with those at or above 6 in increasing order (there are equally many of each, since $c \in V_y$); so a canonical form for a 5-dimensional C_1 would range over $|S_6 \times S_6| \cdot |C_1| = 1.66 \times 10^7$ permutations for each of the 46 350 pairs, where reading a supplied permutation costs a few thousand operations.

Theorem 5.1 (soundness of the search)**.** *Suppose the enumeration above, run at $n = 12$ with $y = 1^6 0^6$, returns true. Then every $[[12, 1, d \geq 3]]$ CE CSS code with that shift is equivalent to one of $K_\mathrm{a}, K_\mathrm{b}, K_\mathrm{c}, K_\mathrm{d}$.*

Proof sketch. Two halves. The enumeration half — the node whose span is C_1 is visited — is the completeness argument of [10], unchanged, since the walk over subspaces of V_y is unchanged. The terminal half is new and runs the opposite way to that paper's: at the node whose span is C_1, the mask of the code's own φ is shown to survive all three filters, using (K) for nonzero-ness, (d_X) for the first and (d_Z) for the second, so the node test has produced a certificate for it. Reading that certificate gives clauses (E1)–(E3): one equality of labelled sets splits into $\pi(C_1) = C_1'$ and $\varphi'(\pi v) = \varphi(v)$ because $2v + b$ with b a bit determines v and b; and the membership test is (E3) verbatim. The only property of the certificate used is that a list of transpositions is a permutation of coordinates, along which a code transports. □

Theorem 5.2 (exactly four)**.** *Every $[[12, 1, d \geq 3]]$ constant-excitation CSS code — any global shift, any construction — is equivalent to exactly one of $K_\mathrm{a}, K_\mathrm{b}, K_\mathrm{c}, K_\mathrm{d}$. Equivalently: up to permutation of the twelve qubits and the redundancy in the global shift there are exactly four $[[12, 1, d \geq 3]]$ CE CSS codes, and all four are $[[12, 1, 3]]$ codes.*

Proof sketch. Proposition 3.5 moves an arbitrary code to the shift $y = 1^6 0^6$ along an equivalence. Theorem 5.1 then applies, *provided* the enumeration returns true at that shift, and it does: this is the one exhaustive computation the theorem rests on. Transitivity of $\sim$ (Proposition 3.4(1)) composes the two steps, and Theorem 4.1 gives the "exactly one". □

The finite search that was run, and what it measured. The certified enumeration at $n = 12$, $y = 1^6 0^6$ visits 826 985 nodes and runs the node filter over 9 905 611 functional masks — the 2^r masks at each node of rank r, the zero mask included — of which 46 350 survive and are certified; the certificate file holds one entry per surviving pair, the longest permutation needs eleven transpositions, and the 65 536-bucket table has maximum load 6. Run inside the proof assistant with the search compiled to a shared library, the whole computation is about three minutes and 790 MB of resident memory; the module containing it checks end to end in about five minutes. A reference implementation in C, written independently of the formal development, reproduces the same numbers in a few seconds, and reproduces as controls the node and pair counts published in [10] at $n = 9, 10, 11$ exactly.

Three further numbers are *measurements*, obtained from that C implementation, and nothing in this paper rests on them. A sweep over all thirteen sorted shifts at $n = 12$ visits 1 390 027 nodes, carrying 13 637 220 pairs (C_1, φ) with $\varphi \neq 0$ — the count published in [10], whose table is in pairs and not in masks — and finds 46 350 accepting pairs at $\mathrm{wt}(y) = 6$ and none at any other shift, an independent confirmation of Proposition 3.5. The four classes account for the 46 350 accepting pairs in the proportions

$$7\,200 \;+\; 16\,200 \;+\; 21\,600 \;+\; 1\,350 \;=\; 46\,350$$

for $K_\mathrm{a}, K_\mathrm{b}, K_\mathrm{c}, K_\mathrm{d}$ respectively. These orbit sizes are not machine-checked; turning them into theorems would require the uniqueness half of the canonical-basis pruning, of which [10] proves only the half that soundness needs, namely that every subspace is reached at least once.

6. Where the published code sits, and thirteen qubits

Proposition 6.1. *The* $[[12, 1, 3]]$ *CE CSS code printed in* [7, Appendix A 6] *is equivalent to* K_{a}.

Proof sketch. Its C_1 — the span of the four printed X-type generators together with the logical $\bar{X} = X_5X_6X_{11}X_{12}$ — has weight enumerator $0^1 4^{15} 8^{15} 12^1$, and in particular weight-4 entry 15 and weight-6 entry 0: a finite check over 32 codewords. By Theorem 5.2 the code is equivalent to one of the four, and by Proposition 3.4(3) the enumerator must agree with that of the class; only K_{a} has weight-4 entry 15. □

So the published code is the $\dim C_1 = 5$ class, and the other three classes are codes that [7] does not exhibit — two of them with $\dim C_1 = 4$ and 3, so not even the same X-type dimension. In stabilizer terms all four have eleven independent generators, split as $\dim C_2$ of X-type and $12 - \dim C_1$ of Z-type: $4 + 7$, $3 + 8$, $3 + 8$ and $2 + 9$.

Proposition 6.2. *Every* $[[13, 1, d \geq 3]]$ *CE CSS code is obtained by padding a twelve-qubit one with a coordinate that no codeword of* C_1 *touches; that twelve-qubit code is then equivalent to one of* $K_{\mathrm{a}}, K_{\mathrm{b}}, K_{\mathrm{c}}, K_{\mathrm{d}}$. *No enumeration at* $n = 13$ *is run, or needed, for this.*

Proof sketch. Thirteen is odd, so Proposition 3.2 applies, and Theorem 5.2 places the result. □

This is a statement about *structure* at thirteen qubits, not a count there, and the distinction is real: padded codes at $n = 13$ do exist, the dead coordinate may be placed inside or outside $\operatorname{supp} y$, giving 6-CE and 7-CE codes, and $\mathrm{wt}(y)$ cannot change within an equivalence class, so those cannot merge. How many classes result is open; see Section 9.

7. Controls

A classification proved by a certified exhaustive search can fail in ways a single true/false answer will not reveal, so the following were checked as well.

Proposition 7.1.

(1) Too few classes is refuted*: $K_{\mathrm{a}}, K_{\mathrm{b}}, K_{\mathrm{c}}, K_{\mathrm{d}}$ are pairwise inequivalent (Theorem 4.1), so "at most three" is false.* Too many is refuted*: Theorem 5.2, so "at least five" is false.*

(2) The certificate table is not a constant true. *On the labelled subspace of $C_1(K_{\mathrm{a}})$ under the functional mask* 1 *— which does not survive (d_Z) — the table reports false; under the surviving mask* 7*, which is K_{a} itself, it reports true.*

(3) The class index is load-bearing. *The identity permutation certifies the first representative's labelled subspace for class* 0 *and fails for class* 1.

(4) Equivalence is not equality of data. *Transposing the first two coordinates of K_{a} gives a code with a different C_1 that is equivalent to it; and replacing the shift $y = 63$ of K_{a} by $y \oplus 195 = 252$ gives a code with a different shift that is equivalent to it. So both the permutation clause and the shift clause of Definition 2.2 identify codes that the raw data distinguishes.*

(5) Equivalence is not everything*: again Theorem 4.1.*

8. Verification

Every numbered statement of this paper is formalised and machine-checked in Lean 4 against Mathlib, with no `sorry` and no declared axiom anywhere in the development, with three exceptions. The passage from (1) to Definition 2.1 in Section 2 is an ordinary argument about states. The two remarks labelled *outside the formal development* are marked where they occur. And two clauses of Theorem 4.1 are checked only outside the kernel: the development pins $d_X \geq 4$ and the weight-4 and weight-6 entries of the four enumerators — the entries the classification actually uses, since they are what separates the four classes — while $d_X = 4$ exactly and the remaining entries of the enumerators are the same exhaustive check run in the reference implementation of Section 5. Definition 2.1 is a structure with the displayed clauses together with the linearity clauses, and Definition 2.2 is the displayed existential with the permutation carried as a list of transpositions; Theorem 3.1, Proposition 3.2, Proposition 3.4 and the certificate-reading half of Theorem 5.1 depend only on the standard axioms `propext`,

`Classical.choice` and `Quot.sound`, so in particular every reduction that shrinks a search space, and the completeness argument for the enumeration, are free of any evaluation axiom; the full statement of Theorem 5.1 additionally inherits the four representatives' own axioms, through the step that identifies the representatives stored inside the search with the codes of Table 1. The exhaustive computations — the certified enumeration at $n = 12$, the four representatives' d_Z clause and their weight-4 and weight-6 counts, and the controls — are discharged by kernel-external evaluation, each contributing one compiler-trusted axiom to the declaration in which it occurs, and the assembled statements carry exactly the axioms one would predict and nothing else: Corollary 3.3 and Proposition 3.5 carry exactly one, that of the $n = 10$ sweep of [10] — the $n = 11$ sweep is verifiably unused — while Theorem 5.2 carries seven, namely that same one, the four representatives' own, the enumeration's, and the weight enumerators'; the full audit is Table 2. Readers who prefer not to trust the compiler may regard those specific finite evaluations as the reproducible computational content of the paper; the independent C implementation described in Section 5 agrees with all of them. Repository reference to be added at submission.

9. What is left open

Thirteen qubits. Proposition 6.2 settles the structure and not the count. Settling the count needs the padding construction run backwards, together with a check that no two paddings that ought to be different are equivalent; the codes split at least by $\mathrm{wt}(y) \in \{6, 7\}$, which is an invariant. This is cheap and was not done.

Fourteen qubits. Thirteen has no full-support code, so the next length carrying genuinely new codes is fourteen. The same machinery applies with $|V_y| = \binom{14}{7} = 3432$ against 924 here; extrapolating the measured costs puts the base enumeration alone at several CPU-hours before any certificate work, and it was priced and not run.

More than one logical qubit. For $k \geq 2$ the minimal length is fourteen [10]; the classification there is untouched, and so is the classification of CE CSS codes of distance 4, for which the least admissible length is not known.

Kernel-binding the orbit sizes. The counts 46 350 and 7 200/16 200/21 600/1 350 are measurements. Making them theorems needs the uniqueness half of the canonical-basis pruning: that each subspace is reached at exactly one node, not merely at least one. The argument is short — a canonical basis is the greedy minimal one, and two of them would differ first at some index, contradicting minimality — and was not written.

Appendix A. Formal statements

For reference, the statements of the formal development. Throughout, `Code n y` is Definition 2.1, `wt n v` the Hamming weight of the low n bits of `v`, `ip n u v` the binary inner product, `^^^` bitwise XOR, `spanVec bs` the subspace spanned by the list `bs`, and `pact l` the permutation given by the transposition list `l`. The statements are reproduced unchanged except that the source's Unicode is transliterated into ASCII: $\mathbb{N}$ as `N`, $\forall$ as `forall`, $\exists$ as `exists`, $\in$ as `in`, $\notin$ as `not in`, $\to$ as `->`, $\leq$ as `<=`, $\neq$ as `<>`, $\times$ as `x`, and $\wedge, \vee, \neg$ as `and`, `or`, `not`; a written-out proof term is elided as `_` and an omitted run of structure fields as `....`

```
structure Code (n y : N) where
  hy : y < 2 ^ n
  C1 : Finset N
  mem_lt : forall v in C1, v < 2 ^ n
  zero_mem : 0 in C1
  xor_mem : forall v in C1, forall w in C1, v ^^^ w in C1
  phi : N -> Bool
  phi_add : forall v in C1, forall w in C1, phi (v ^^^ w) = xor (phi v) (phi w)
  phi_ne : exists v in C1, phi v = true
  ce : forall v in C1, wt n (y ^^^ v) = wt n y
  dX : forall v in C1, phi v = true -> 3 <= wt n v
  dZ : forall u < 2 ^ n, 1 <= wt n u -> wt n u <= 2 ->
         exists v in C1, ip n u v <> phi v
```

```
def CEquiv {n y y' : N} (C : Code n y) (D : Code n y') : Prop :=
  exists l : List (N x N), (forall q in l, q.1 < n and q.2 < n) and
    C.C1.image (pact l) = D.C1 and
    (forall v in C.C1, D.phi (pact l v) = C.phi v) and
    (pact l y ^^^ y') in D.C1

theorem even_of_full_support {n y : N} (C : Code n y)
    (hfull : forall i < n, exists v in C.C1, v.testBit i = true) : n = 2 * wt n y

def Code.shorten {m y : N} (C : Code (m + 1) y)
    (h : forall v in C.C1, v.testBit m = false) : Code m (y % 2 ^ m)

theorem exists_shorter_of_odd {m y : N} (C : Code (m + 1) y) (hodd : (m + 1) % 2 = 1) :
    exists y', Nonempty (Code m y')

theorem no_code_eleven (y : N) (C : Code 11 y) : False

theorem CEquiv.trans {n y y' y'' : N} {C : Code n y} {D : Code n y'} {E : Code n y''}
    (h1 : CEquiv C D) (h2 : CEquiv D E) : CEquiv C E
theorem CEquiv.symm {n y y' : N} {C : Code n y} {D : Code n y'}
    (h : CEquiv C D) : CEquiv D C
theorem CEquiv.refl {n y : N} (C : Code n y) : CEquiv C C

def Code.rebase {n y : N} (C : Code n y) (c : N) (hc : c in C.C1) : Code n (y ^^^ c)
theorem CEquiv.rebase {n y : N} (C : Code n y) (c : N) (hc : c in C.C1) :
    CEquiv C (Code.rebase C c hc)

def wcount (n : N) {y : N} (C : Code n y) (w : N) : N :=
  (C.C1.filter (fun v => wt n v = w)).card
theorem CEquiv.wcount_eq {n y y' : N} {C : Code n y} {D : Code n y'}
    (h : CEquiv C D) (w : N) : wcount n C w = wcount n D w

theorem full_support_twelve {y : N} (C : Code 12 y) :
    forall i < 12, exists v in C.C1, v.testBit i = true
theorem wt_twelve {y : N} (C : Code 12 y) : wt 12 y = 6
theorem to_sorted {y : N} (C : Code 12 y) : exists D : Code 12 63, CEquiv C D

def C1a : Finset N := (spanVec [2145, 1105, 585, 325, 195]).toFinset
def C1b : Finset N := (spanVec [2160, 1161, 645, 387]).toFinset
def C1c : Finset N := (spanVec [2277, 1237, 588, 387]).toFinset
def C1d : Finset N := (spanVec [2160, 1548, 387]).toFinset
def Ka : Code 12 63 where C1 := C1a; phi := ip 12 7 ; ...
def Kb : Code 12 63 where C1 := C1b; phi := ip 12 19; ...
def Kc : Code 12 63 where C1 := C1c; phi := ip 12 22; ...
def Kd : Code 12 63 where C1 := C1d; phi := ip 12 21; ...

theorem K_dX_four :
    (forall v in C1a, ip 12 7 v = true -> 4 <= wt 12 v)
      and (forall v in C1b, ip 12 19 v = true -> 4 <= wt 12 v)
      and (forall v in C1c, ip 12 22 v = true -> 4 <= wt 12 v)
      and (forall v in C1d, ip 12 21 v = true -> 4 <= wt 12 v)
theorem K_dZ_three :
    (exists u < 2 ^ 12, wt 12 u = 3 and forall v in C1a, ip 12 u v = ip 12 7 v)
      and (exists u < 2 ^ 12, wt 12 u = 3 and forall v in C1b, ip 12 u v = ip 12 19 v)
      and (exists u < 2 ^ 12, wt 12 u = 3 and forall v in C1c, ip 12 u v = ip 12 22 v)
      and (exists u < 2 ^ 12, wt 12 u = 3 and forall v in C1d, ip 12 u v = ip 12 21 v)
theorem K_wcounts :
    (wcount 12 Ka 4, wcount 12 Ka 6) = (15, 0)
      and (wcount 12 Kb 4, wcount 12 Kb 6) = (7, 0)
      and (wcount 12 Kc 4, wcount 12 Kc 6) = (3, 8)
      and (wcount 12 Kd 4, wcount 12 Kd 6) = (3, 0)
```

```
theorem reps_pairwise_inequiv :
    not CEquiv Ka Kb and not CEquiv Ka Kc and not CEquiv Ka Kd
      and not CEquiv Kb Kc and not CEquiv Kb Kd and not CEquiv Kc Kd

theorem labelled_eq {A B : Finset N} {f g : N -> Bool} {pi : N -> N}
    (h : A.image (fun v => 2 * pi v + (if f v then 1 else 0))
       = B.image (fun w => 2 * w + (if g w then 1 else 0))) :
    A.image pi = B and forall v in A, g (pi v) = f v
theorem cequiv_of_cert (C : Code 12 63) {bs : List N}
    (hbs : forall b in bs, b in C.C1) (hall : forall c in C.C1, c in spanVec bs)
    (l : List (N x N)) (hb : forall q in l, q.1 < 12 and q.2 < 12)
    (D : Code 12 63) (c : N)
    (hrs : (CK.repState c).toFinset
             = D.C1.image (fun w => 2 * w + (if D.phi w then 1 else 0)))
    (hrc : (CK.repC1 c).toFinset = D.C1)
    (h1 : CK.applyPerm l (CK.stateOf bs.length (spanIdx bs) (bitsOf C.phi bs))
             = CK.repState c)
    (h2 : (CK.repC1 c).contains (pact l 63 ^^^ 63) = true) : CEquiv C D
theorem dfsC_sound (C : Code 12 63) (Vl lu : List N)
    (hVl : forall c, c < 2 ^ 12 -> inV 12 63 c = true -> c in Vl)
    (hlu : forall u in lu, u < 2 ^ 12 and 1 <= wt 12 u and wt 12 u <= 2) :
    forall (fuel : N) (bs : List N) (last : N),
      (forall b in bs, b in C.C1) ->
      (forall c in C.C1, c not in spanVec bs -> last < c) ->
      C.C1.card <= (spanVec bs).toFinset.card + fuel ->
      CK.dfsC 12 63 Vl lu fuel bs (spanIdx bs)
        (lu.map fun u => bitsOf (ip 12 u) bs) last = true ->
      CEquiv C Ka or CEquiv C Kb or CEquiv C Kc or CEquiv C Kd
theorem classify_of_searchC (h : CK.searchC 12 63 = true) (C : Code 12 63) :
    CEquiv C Ka or CEquiv C Kb or CEquiv C Kc or CEquiv C Kd

theorem searchC_twelve : CK.searchC 12 63 = true
theorem classification {y : N} (C : Code 12 y) :
    CEquiv C Ka or CEquiv C Kb or CEquiv C Kc or CEquiv C Kd
theorem exactly_four_classes :
    (forall (y : N) (C : Code 12 y),
        CEquiv C Ka or CEquiv C Kb or CEquiv C Kc or CEquiv C Kd)
      and not CEquiv Ka Kb and not CEquiv Ka Kc and not CEquiv Ka Kd
      and not CEquiv Kb Kc and not CEquiv Kb Kd and not CEquiv Kc Kd

theorem certOK_rejects :
    CK.certOK (CK.stateOf 5 (CK.spanIdx [2145, 1105, 585, 325, 195]) 1) = false
theorem certOK_accepts :
    CK.certOK (CK.stateOf 5 (CK.spanIdx [2145, 1105, 585, 325, 195]) 7) = true
theorem checkCert_class :
    CK.checkCert 4 (CK.repState 0) = true and CK.checkCert 5 (CK.repState 0) = false
theorem cequiv_nontrivial :
    CEquiv Ka (Ka.swap 0 1 _ _) and (Ka.swap 0 1 _ _).C1 <> Ka.C1
theorem cequiv_rebase_nontrivial : CEquiv Ka (Code.rebase Ka 195 _)

theorem wcount_codeTwelve :
    (wcount 12 codeTwelve 4, wcount 12 codeTwelve 6) = (15, 0)
theorem source_in_class_a : CEquiv codeTwelve Ka
theorem thirteen_is_padded (y : N) (C : Code 13 y) :
    exists (y' : N) (D : Code 12 y'),
      CEquiv D Ka or CEquiv D Kb or CEquiv D Kc or CEquiv D Kd
```

Table 2 is the axiom audit. Every declaration depends on `propext`, `Classical.choice` and `Quot.sound`; the middle column lists what else. Here `ten` is the evaluation axiom of the $n = 10$ sweep of [10]; `a`, `b`, `c`, `d` are those of the four representatives' d_Z clauses; `sea` is that of the certified enumeration at $n = 12$; `wc` that of the weight-4 and weight-6 entries of their enumerators; `dZ3` that

of $d_Z \le 3$; and src, src2 those of the published code's own d_Z clause and of its weight-4 and weight-6 counts.

declaration	extra axioms	paper
`even_of_full_support`	—	Thm 3.1
`Code.shorten`, `exists_shorter_of_odd`	—	Prop 3.2
`no_code_eleven`	ten	Cor 3.3
`CEquiv.trans`, `.symm`, `.wcount_eq`	—	Prop 3.4
`Code.rebase` (`propext`, `Quot.sound` only)	—	Prop 3.4(2)
`full_support_twelve`, `wt_twelve`, `to_sorted`	ten	Prop 3.5
`Ka`, `Kb`, `Kc`, `Kd`	a / b / c / d	Table 1
`K_dX_four`	—	Thm 4.1
`K_dZ_three`	dZ3	Thm 4.1
`K_wcounts`, `reps_pairwise_inequiv`	wc, a–d	Thm 4.1
`labelled_eq`, `cequiv_of_cert`	—	Thm 5.1
`dfsC_sound`, `classify_of_searchC`	a–d	Thm 5.1
`searchC_twelve`	sea	Thm 5.2
`classification`	ten, a–d, sea	Thm 5.2
`exactly_four_classes`	ten, wc, a–d, sea	Thm 5.2
`certOK_rejects`, `certOK_accepts`	one each	Prop 7.1(2)
`checkCert_class` (`propext` only)	—	Prop 7.1(3)
`cequiv_nontrivial`, `cequiv_rebase_nontrivial`	a	Prop 7.1(4)
`wcount_codeTwelve`	src, src2	Prop 6.1
`source_in_class_a`	src, src2, ten, wc, a–d, sea	Prop 6.1
`thirteen_is_padded`	ten, a–d, sea	Prop 6.2

TABLE 2. Axiom audit, as reported by the proof assistant.

REFERENCES

[1] E.-J. Chang, *Quantum dual extended Hamming code immune to collective coherent errors*, Phys. Rev. A **112** (2025), no. 5, 052410; arXiv:2503.05249v5.

[2] A. Cross and D. Vandeth, *Small binary stabilizer subsystem codes*, arXiv:2501.17447v1. (Abstract, verbatim: "We establish a database consisting of a representative of every binary quantum stabilizer code under local Clifford permutation equivalence for $n \le 9$.")

[3] *The Error Correction Zoo* (V. V. Albert and P. Faist, eds.), entry `css_12_1_3`, `errorcorrectionzoo.org`, consulted 2026-09-03 (entry created 2026-05-26, last edited 2026-08-17).

[4] J. Hu, Q. Liang, N. Rengaswamy and R. Calderbank, *Mitigating coherent noise by balancing weight-2 Z-stabilizers*, IEEE Trans. Inform. Theory **68** (2022), no. 3, 1795–1808; arXiv:2011.00197v5. (Theorem 3, Corollary 12 and Example 6 are numbered as in v5, which carries the journal's formatting.)

[5] A. B. Khesin and A. M. Li, *Equivalence classes of quantum error-correcting codes*, arXiv:2406.12083v1. (Its tables of equivalence classes of $[n, 1]$ and $[n, 2]$ codes run to $n = 5$.)

[6] J. M. Koh, A. Gong, A. C. Diaconu, D. B. Tan, A. A. Geim, M. J. Gullans, N. Y. Yao, M. D. Lukin and S. Majidy, *Entangling logical qubits without physical operations*, arXiv:2601.20927v1. (Contains an exhaustive enumeration of CSS codes up to $n = 14$ under permutation and global-Hadamard equivalence; Table VI.)

[7] C.-Y. Lai, P.-H. Liou and Y. Ouyang, *Fault-tolerant quantum error correction for constant-excitation stabilizer codes under coherent noise*, arXiv:2507.10395v1 (14 July 2025), quant-ph. (Consulted 2026-09-03: still a single version, no journal reference.)

[8] Y. Ouyang, *Avoiding coherent errors with rotated concatenated stabilizer codes*, npj Quantum Information **7** (2021), article 87; arXiv:2010.00538.

[9] M. B. Plenio, V. Vedral and P. L. Knight, *Quantum error correction in the presence of spontaneous emission*, Phys. Rev. A **55** (1997), no. 1, 67–71; arXiv:quant-ph/9603022.

[10] Korea Superintelligence Labs, *Exact minimal lengths for constant-excitation quantum codes: eight qubits for stabilizer codes, twelve and fourteen for CSS*, companion manuscript, 2026, second version. Neither that manuscript nor the present one is posted at the time of writing; their archival identifiers will be supplied together at submission, and the sentence quoted in Section 1 is from the second version.