

Ultra-High-Rate Quantum LDPC Codes

Kenta Kasai Institute of Science Tokyo

CWI QEC workshop
Amsterdam 28 October 2026

Joint work



Jong Yeon Lee
UIUC / KIAS



Koki Okada
Institute of
Science Tokyo



Nishad Maskara
MIT



Kenta Kasai
Institute of
Science Tokyo



Hengyun (Harry) Zhou
MIT

Design Principles for Ultra-High-Rate Quantum Codes: [arXiv:2609.30069](https://arxiv.org/abs/2609.30069)
Okada–Kasai, Low-Weight Canonical Logical Bases: [arXiv:2609.35601](https://arxiv.org/abs/2609.35601)

Bringing Classical LDPC Design Principles to Quantum Codes

My research seeks to recover classical LDPC design freedom under quantum commutation constraints.

Classical foundations



Sparse graphs
Degree distributions
Short-cycle control
Iterative decoding

The quantum challenge

$$H_X H_Z^T = 0 \longrightarrow$$

X/Z graphs are coupled
Asymptotically good LP families exist.
Finite-length construction and efficient decoding remain active design problems.

Our construction approach

Build commutation into design

Keep X/Z checks orthogonal.
Preserve useful design choices.

Goal: retain design freedom

Control degrees and sparsity.
Seek explicit constructions and checkable guarantees.

Toward reusable quantum LDPC design principles

Construction $\longleftrightarrow$ Distance verification $\longleftrightarrow$ Decoding $\longleftrightarrow$ Logical operations

High-rate quantum LDPC codes aim to lower qubit overhead

Within a physical-qubit budget, protect more logical information.

Physical qubits n
Data-qubit budget

Logical qubits k
Rate $R = k/n$

Minimum distance d
Protection against errors

Check weight L
Qubits per check

$[[n, k, d]]$: larger k improves data-qubit efficiency; larger d strengthens protection.

Implementation requirements: reliable, affordable check measurements

Placement and wiring

Connect the qubits required by each check. Sparsity bounds the **number** of interactions; layout determines their **length**.

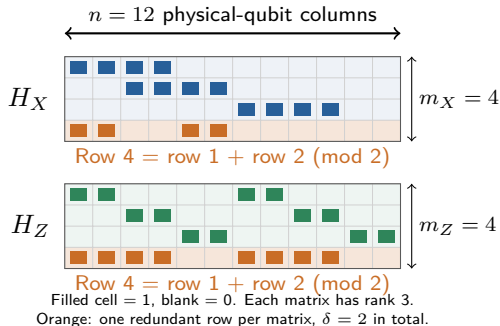
Noisy measurements

Recover despite faulty gates and readout. Include measurement ancillas, scheduling, and decoding in the resource budget.

Why LDPC? Bounded check and qubit degrees keep interactions per check and per qubit small as the code grows.

Design opportunity: high rate and useful distance, with small check weight L .

Fewer check rows guarantee room for logical qubits



n : **physical qubits**

Same columns in both matrices.

$r = m_X + m_Z$: **listed check rows**

One independent commuting check removes one binary degree of freedom.

δ : **redundant rows in total**

$\delta = (m_X - \text{rank } H_X) + (m_Z - \text{rank } H_Z)$.

k : **logical qubits left unconstrained**

$k = n - \text{rank } H_X - \text{rank } H_Z$.

Example: $n = 12$, $r = 8$, $\delta = 2$, $k = 6$.



The bar counts dimensions, not separate groups of physical qubits.

$$R = \frac{k}{n} = 1 - \frac{r}{n} + \frac{\delta}{n} \geq 1 - \frac{r}{n}. \quad \text{A positive row-count floor requires } r < n.$$

Standard BB matrices have $r = n$ and encode through row dependencies.

HGP and LP constructions can also have positive design rate.

Three block parameters set length, degrees, and rate

Circulant permutation matrix (CPM) blocks

Block layout shown: $J = 3, L = 8$

$$H_X = \begin{pmatrix} E_{00} & E_{01} & E_{02} & E_{03} & E_{04} & E_{05} & E_{06} & E_{07} \\ E_{10} & E_{11} & E_{12} & E_{13} & E_{14} & E_{15} & E_{16} & E_{17} \\ E_{20} & E_{21} & E_{22} & E_{23} & E_{24} & E_{25} & E_{26} & E_{27} \end{pmatrix}$$

$$H_Z = \begin{pmatrix} D_{00} & D_{01} & D_{02} & D_{03} & D_{04} & D_{05} & D_{06} & D_{07} \\ D_{10} & D_{11} & D_{12} & D_{13} & D_{14} & D_{15} & D_{16} & D_{17} \\ D_{20} & D_{21} & D_{22} & D_{23} & D_{24} & D_{25} & D_{26} & D_{27} \end{pmatrix}$$

J block rows, L block columns: JL **CPMs** in each matrix.

Each full binary matrix has JP **rows** and LP **columns**.

$$n = LP, \quad r = 2JP, \quad \boxed{R \geq 1 - 2J/L}$$

A positive design-rate floor requires $L > 2J$.

At fixed J , larger L raises the rate floor and check weight. Larger P scales length at fixed degrees.

Every block E_{il}, D_{jl} is a CPM.

$C(a)$: shift each row of I_P cyclically **left by** a positions.

$a \in \mathbb{Z}_P$, size $P \times P$.

$$C(1) = \begin{pmatrix} 0 & 0 & 0 & \mathbf{1} \\ \mathbf{1} & 0 & 0 & 0 \\ 0 & \mathbf{1} & 0 & 0 \\ 0 & 0 & \mathbf{1} & 0 \end{pmatrix} \quad (P = 4)$$

One 1 in every row and column.

column weight J , row weight L .

A bitmap example makes pairwise cancellation visible

Previous slide's block layout: $J = 3$, $L = 8$. Colors and block positions match below.

$$H_X = \begin{pmatrix} E_{00} & E_{01} & E_{02} & E_{03} & E_{04} & E_{05} & E_{06} & E_{07} \\ E_{10} & E_{11} & E_{12} & E_{13} & E_{14} & E_{15} & E_{16} & E_{17} \\ E_{20} & E_{21} & E_{22} & E_{23} & E_{24} & E_{25} & E_{26} & E_{27} \end{pmatrix}$$

$$H_Z = \begin{pmatrix} D_{00} & D_{01} & D_{02} & D_{03} & D_{04} & D_{05} & D_{06} & D_{07} \\ D_{10} & D_{11} & D_{12} & D_{13} & D_{14} & D_{15} & D_{16} & D_{17} \\ D_{20} & D_{21} & D_{22} & D_{23} & D_{24} & D_{25} & D_{26} & D_{27} \end{pmatrix}$$

Focus on the first block row of each matrix (index 0), with $P = 5$. Filled = 1. White = 0.

$$E_{00}=C(1) \ E_{01}=C(0) \ E_{02}=C(2) \ E_{03}=C(3) \ E_{04}=C(1) \ E_{05}=C(4) \ E_{06}=C(3) \ E_{07}=C(0)$$



$$D_{00}=C(4) \ D_{01}=C(0) \ D_{02}=C(2) \ D_{03}=C(1) \ D_{04}=C(0) \ D_{05}=C(1) \ D_{06}=C(2) \ D_{07}=C(2)$$



Each block column contributes $E_{0\ell} D_{0\ell}^T$.



pair {0, 3}: 0 pair {1, 2}: 0 pair {4, 6}: 0 pair {5, 7}: 0 $(H_X H_Z^T)_{00}$
 Four pairs cancel in block (0,0). Apply this rule to every block-row pair.

Circulant-permutation-matrix-based pair-partition codes (CPM-PP). Pairing requires even L .

Solve the pair equations, then select useful codes

Match the relative shifts

$E_{i\ell} = C(e_{i\ell})$, $D_{j\ell} = C(d_{j\ell})$: each label is a cyclic shift.

$$C(a)C(b)^T = C(a - b).$$

For each chosen pair $\{u, v\}$, require the same relative shift:

$$\boxed{e_{iu} - d_{ju} = e_{iv} - d_{jv} \pmod{P}}.$$

Example: one pair modulo 5

$$1 - 4 \equiv 2, \quad 3 - 1 \equiv 2 \pmod{5}.$$

Different shifts, same product $C(2)$.

Choose, solve, evaluate

1. Choose pair partitions $\mathcal{M} = (M_{ij})$.
Each pair gives a linear equation for the shifts.

2. Solve the linear equations together.
Determine the shifts modulo P , then build H_X, H_Z .

3. Evaluate the completed matrices.
Rank: logical-qubit count k .
Distance: protection d .

The equations guarantee commutation. Evaluation identifies useful codes.

Pairing guarantees commutation; evaluation certifies the useful parameters.

A complete PP example: pairings and shift arrays

Column weight $J = 3$, **check weight** $L = 8$, **block size** $P = 53$. All shift arithmetic is modulo 53.

Pairing array $\mathcal{M} = (M_{ij})$: row i selects an X block row, column j a Z block row.

$$\mathcal{M} = \left(\begin{array}{c|c|c} (0,3)(1,2)(4,6)(5,7) & (0,2)(1,6)(3,7)(4,5) & (0,6)(1,5)(2,3)(4,7) \\ \hline (0,5)(1,3)(2,4)(6,7) & (0,3)(1,5)(2,7)(4,6) & (0,7)(1,4)(2,6)(3,5) \\ \hline (0,1)(2,6)(3,5)(4,7) & (0,5)(1,7)(2,4)(3,6) & (0,2)(1,6)(3,7)(4,5) \end{array} \right)$$

Each (u, v) pairs two block columns. Indices start at 0.

One simultaneous solution: scalar shift arrays $E = (e_{i\ell})$, $D = (d_{j\ell})$.

$$E = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 2 & 3 & 6 & 12 & 36 & 13 & 26 & 16 \\ 3 & 47 & 36 & 39 & 30 & 50 & 13 & 20 \end{pmatrix} \quad D = \begin{pmatrix} 4 & 48 & 48 & 4 & 25 & 15 & 25 & 15 \\ 5 & 42 & 5 & 15 & 52 & 52 & 42 & 15 \\ 6 & 40 & 39 & 39 & 20 & 40 & 6 & 20 \end{pmatrix}$$

$$H_X = (E_{i\ell}), \quad E_{i\ell} = C(e_{i\ell}), \quad H_Z = (D_{j\ell}), \quad D_{j\ell} = C(d_{j\ell}).$$

Every number becomes a 53×53 CPM. Each binary check matrix has size 159×424 .

Colored row pair $i = 1, j = 0$: the eight relative shifts are

$$(e_{1\ell} - d_{0\ell})_{\ell=0}^7 = (51, 8, 11, 8, 11, 51, 1, 1) \pmod{53}.$$

$$(H_X H_Z^T)_{10} = [C(51) + C(51)] + [C(8) + C(8)] + [C(11) + C(11)] + [C(1) + C(1)] = 0.$$

All nine row pairs satisfy their four equations, so $H_X H_Z^T = 0$.

Pairing enables randomized search for good quantum codes

Classical coding: randomness is a design resource.

Well-chosen random ensembles contain good codes.

Randomized construction and testing also help find useful finite-length codes.



Quantum CSS coding adds X–Z commutation.

Pair partitioning lets us explore random choices **within that constraint**.

1. Fix the structure

Column weight J ,
row weight L ,
prime block size P ,
pairing $\mathcal{M}$



2. Sample a solution

Solve the linear equations.
Randomly select a solution
from the solution space.
One solution fixes all shifts.



3. Select useful codes

Compute logical-qubit count k .
Verify minimum distance d .

Every sampled solution preserves
pair cancellation and sparse degrees.

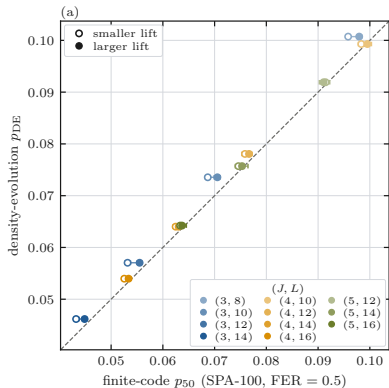
Algebra guarantees commutation; randomness explores the finite-length design space.

Density evolution links degree design to decoding performance

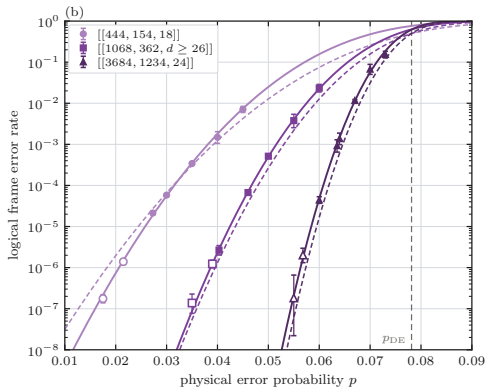
Density evolution (DE): track BP message distributions over iterations to predict a noise threshold.

Evaluation model: depolarizing **code-capacity** noise—data-qubit errors only, with perfect syndrome measurements.

Design knobs: regular column weight J , check weight L ; larger **girth** (shortest Tanner cycle) delays correlated feedback.



(a) BP (SPA): measured midpoint agrees with DE
 p_{50} gives FER = 0.5; larger lifts approach the diagonal.



(b) SPA + residual repair + normalized min-sum

Waterfall width $\sim n^{-1/2}$; dashed: DE-based finite-length fits.

Quantum DE is an ensemble heuristic; regularity alone is not optimality.

Select useful degrees, suppress short cycles, and validate finite-code decoding.

Pair-partition codes give large exact distance at short lengths

(J, L)	$[[n, k, d]]$	Lift P	Rate k/n
(3, 8)	[[472, 122, 16]]	59	0.258
(3, 8)	[[776, 198, 20]]	97	0.255
(3, 10)	[[970, 392, 16]]	97	0.404
(3, 10)	[[1630, 656, 20]]	163	0.402
(3, 12)	[[2004, 1006, 16]]	167	0.502
(3, 12)	[[2676, 1342, 18]]	223	0.501
(3, 14)	[[1414, 812, 12]]	101	0.574
(3, 14)	[[3122, 1788, 16]]	223	0.573
(4, 10)	[[220, 50, 16]]	22	0.227
(4, 10)	[[320, 70, 20]]	32	0.219
(4, 12)	[[372, 130, 16]]	31	0.349
(4, 12)	[[492, 170, 20]]	41	0.346
(4, 14)	[[518, 228, 16]]	37	0.440
(4, 14)	[[574, 252, 18]]	41	0.439
(4, 16)	[[848, 430, 18]]	53	0.507
(4, 16)	[[944, 478, 20]]	59	0.506
(5, 10)	[[170, 8, 20]]	17	0.047

Cyclic symmetry streamlines the search

$$H_X(I_L \otimes C(1)) = (I_J \otimes C(1))H_X,$$

$$H_Z(I_L \otimes C(1)) = (I_J \otimes C(1))H_Z.$$

Shift positions within every qubit block and every check block by one. This preserves the code and logical weight.

$$LP \text{ starting points} \longrightarrow L$$

One representative position per qubit block.

Certifying the exact distance

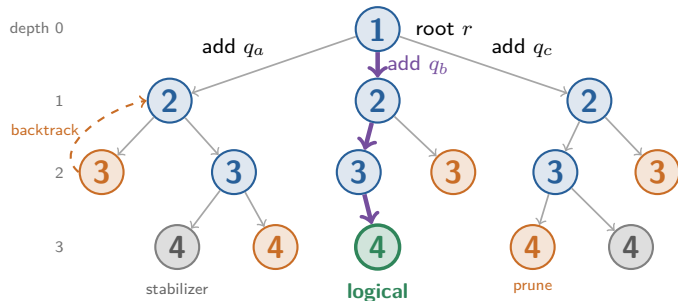
Exclude all lighter nontrivial logicals, then find a logical of weight d .

A search tree reveals low-weight logical operators

For a **Z-type** operator, select a small support S with binary indicator v :

$$H_X v^T = 0, \quad v \notin \text{Row}(H_Z), \quad \text{wt}(v) \leq W.$$

Each node is a partial support S . Its number is $|S|$.



Thick path: one possible route to a logical operator.

Abstract partial tree, not a code instance. Each edge adds one qubit.

Grow through an odd check

Choose a check with odd overlap. Try its available neighboring qubits. Update **all** checks touched.

Prune only with a valid reason

No available repair, or $|S| + b > W$.

b : a lower bound on the extra qubits needed.

W : the search weight limit.

Zero syndrome: test the row space

Inside $\text{Row}(H_Z)$: stabilizer, stop.

Outside: logical operator, return it.

Found: a distance upper bound. **Completed failure on both CSS sides:** $d > W$.

Symplectic halving maps qubit pairs to Pauli coordinates

A concrete example: $[[10, 2, 3]]_{\text{CSS}} \longrightarrow [[5, 1, 3]]$

Parent: ten physical qubits

Two groups of five; pair matching column positions.

$$H_X = (A \mid B), \quad H_Z = (B \mid A).$$

$$H_X = \left(\begin{array}{ccccc|ccccc} 1 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 \end{array} \right)$$

$$H_Z = \left(\begin{array}{ccccc|ccccc} 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \end{array} \right)$$

Halved: five physical qubits

Same bits: **X components** | **Z components**.

$$H_{\text{half}} = \left(\begin{array}{ccccc|ccccc} 1 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 \end{array} \right)$$

The four resulting Pauli checks

$$\left(\begin{array}{ccccc} X & Z & Z & X & I \\ I & X & Z & Z & X \\ X & I & X & Z & Z \\ Z & X & I & X & Z \end{array} \right)$$

The standard five-qubit perfect code.

Commutation carries over: $H_X H_Z^T = AB^T + BA^T = 0$.

Check weight does not increase: $w_{\text{parent}} = L \implies w_{\text{half}} \leq L$. Here both have weight 4.

The same binary entries describe checks on half as many qubits.

Halved codes give compact blocks with useful distance

Halving preserves rate R and adds compact choices with useful distance d .

(J, L)	$[[n, k, d]]$	Lift P	Rate k/n
(3, 8)	$[[52, 15, 7]]$	13	0.288
(3, 8)	$[[316, 81, 15]]$	79	0.256
(3, 8)	$[[388, 99, 18]]$	97	0.255
(3, 10)	$[[500, 202, 14]]$	100	0.404
(3, 12)	$[[450, 227, 10]]$	75	0.504
(3, 14)	$[[476, 274, 9]]$	68	0.576
(4, 10)	$[[90, 21, 11]]$	18	0.233
(4, 10)	$[[140, 31, 15]]$	28	0.221
(4, 10)	$[[150, 33, \leq 16]]$	30	0.220
(4, 10)	$[[200, 43, 20]]$	40	0.215
(4, 12)	$[[240, 83, \leq 17]]$	40	0.346
(4, 12)	$[[312, 107, \leq 21]]$	52	0.343
(4, 14)	$[[287, 126, \leq 16]]$	41	0.439
(4, 14)	$[[336, 147, \leq 18]]$	48	0.438
(4, 16)	$[[336, 171, \leq 15]]$	42	0.509

TABLE II. Representative halved non-CSS pair-partition codes. The full frontier is given in Table VII.

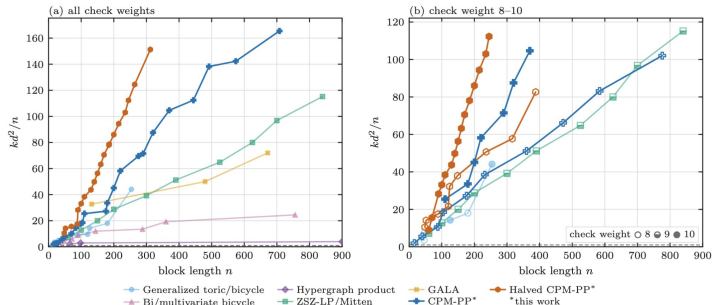


FIG. 8. Encoding efficiency across construction families. kd^2/n frontier as a function of blocklength n for different code families [6, 7, 14, 16, 23, 55–58]. (a) All check weights. (b) The same comparison restricted to check weights eight to ten, with the pair-partition templates (3, 8) and (4, 10) drawn as separate series and check weight distinguished by marker fill. The dashed line marks the rotated surface code, $kd^2/n = 1$. Each code contributes the distance its source quotes, which is an upper bound wherever the distance is not yet exact; this applies to our families and to the comparison families alike.

Entries marked $\leq$ are distance upper bounds. The comparison uses source-reported distances, including upper bounds.

Halved codes perform well with noisy measurements

Circuit-level memory simulation: physical error rate $p = 0.1\%$, 20 rounds.

Code $[[n, k, d]]$	Check weight	Shots	Failures	Logical failure rate per logical qubit per round
Halved $[[95, 22, 11]]$	9–10	3.73×10^8	10	6.1×10^{-11}
Halved $[[186, 65, 14]]$	11–12	5.58×10^8	0	$< 4.1 \times 10^{-12}$
Surface $[[121, 1, 11]]$	2 or 4	3.41×10^8	38	5.6×10^{-9}

Check weight counts data qubits acted on (Y counts once). Surface checks: 2 at boundaries, 4 in the bulk.

Rates are normalized as in Table III; zero observed failures give a **95% upper bound**, not zero risk.

Halved codes: hierarchical decoder

Fast message passing $\rightarrow$ relay retries

$\rightarrow$ integer-programming fallback.

Pass information between attempts; escalate unresolved cases to optimization.

Surface code: belief matching

Sum-product BP estimates fault probabilities

$\rightarrow$ reweight the matching graph

$\rightarrow$ minimum-weight perfect matching.

Compare **code-and-decoder combinations**.

Shared model: noisy two-qubit gates, resets, and measurements; explicit extraction schedules.

No idling errors; ideal preparation and final readout. Layout and full space-time overhead remain to be assessed.

Compact-code gains survive the modeled measurement noise.

Graph design helps high-rate codes decode well

CSS parent design

Regular column weight J , check weight L .

Use DE to choose degrees.

Larger girth delays correlated feedback
in each parent Tanner graph.

Surface-code scaling picture

Below threshold, larger distance d
suppresses residual logical error chains.

BP density-evolution picture

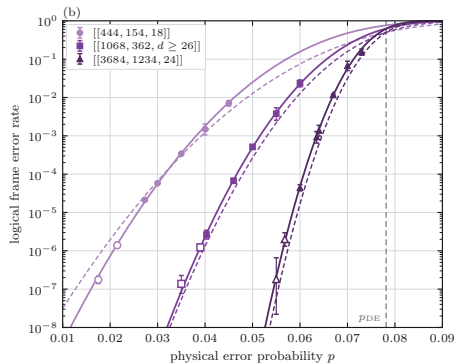
At fixed degrees, repeated message updates
improve reliability. Below the BP threshold,
message-error probability tends to zero
in the infinite-length, tree-like model.

DE is an ensemble heuristic. Regularity alone is not optimality.
Distance and trapping sets still matter.

Code-capacity evidence (slide 10)

Perfect syndrome measurements

SPA + residual repair + normalized min-sum



Waterfall width $\sim n^{-1/2}$ (empirical fit)

These PP codes give every logical qubit a light canonical pair

Koki Okada and Kenta Kasai arXiv:2609.35601v1

Low-Weight Canonical Logical Bases from Pair-Partition Codes

Rows of $L_X, L_Z \in \mathbb{F}_2^{k \times n}$: logical X / Z representatives. All products below are over $\mathbb{F}_2$.

Checks commute

$$H_X H_Z^\top = 0$$

Logicals commute with checks

$$H_Z L_X^\top = 0, \quad H_X L_Z^\top = 0$$

Canonical pairing

$$L_X L_Z^\top = I_k$$

Complete low-weight canonical bases

Code $[[n, k, d]]$	Each logical X / Z weight
$[[320, 80, 14]]$	25 / 25
$[[448, 112, 18]]$	27 / 27
$[[2048, 512, 24]]$	27 / 27

All k logical qubits have explicit partners. Maximum check weight: 10 on each side.

Canonical pairing: address each logical qubit

Specify its X / Z actions on physical qubits.

Only matching X–Z partners anticommute.

Low weight: fewer qubits per operator

Explicit small supports provide inputs to logical-measurement and logical-operation design.

Kasai's selected check matrices form an APM-PP code

Example: $J = 3$, $L = 12$. Selected block rows $i, j = 0, 1, 2$. Each F_u, G_u is a $P \times P$ affine permutation: $x \mapsto ax + b \pmod{P}$, with $\gcd(a, P) = 1$.

$$H_X = \left(\begin{array}{c|cccccc} & \textbf{A} & & & & & & \textbf{B} \\ \hline F_0 & F_1 & F_2 & F_3 & F_4 & F_5 & G_0 & G_1 & G_2 & G_3 & G_4 & G_5 \\ F_5 & F_0 & F_1 & F_2 & F_3 & F_4 & G_5 & G_0 & G_1 & G_2 & G_3 & G_4 \\ F_4 & F_5 & F_0 & F_1 & F_2 & F_3 & G_4 & G_5 & G_0 & G_1 & G_2 & G_3 \end{array} \right),$$

One cancelling pair: $i = j = 0$

A: the column-2 contribution

$$F_1(G_5^T)^T = F_1G_5$$

B: the column-12 contribution

$$G_5(F_1^T)^T = G_5F_1$$

Equal by design; cancel in $\mathbb{F}_2$:

$$F_1G_5 + G_5F_1 = 0$$

$$H_Z = \left(\begin{array}{c|cccccc} & \textbf{A} & & & & & & \textbf{B} \\ \hline G_0^T & G_5^T & G_4^T & G_3^T & G_2^T & G_1^T & F_0^T & F_5^T & F_4^T & F_3^T & F_2^T & F_1^T \\ G_1^T & G_0^T & G_5^T & G_4^T & G_3^T & G_2^T & F_1^T & F_0^T & F_5^T & F_4^T & F_3^T & F_2^T \\ G_2^T & G_1^T & G_0^T & G_5^T & G_4^T & G_3^T & F_2^T & F_1^T & F_0^T & F_5^T & F_4^T & F_3^T \end{array} \right).$$

A: block column 2. B: block column 12.

Pairing rule: $F_u G_{j-i-u} = G_{j-i-u} F_u$ for $i, j = 0, 1, 2$, $u \in \mathbb{Z}_6$.

Indices are modulo 6. Equal pairs cancel over $\mathbb{F}_2$, giving $H_X H_Z^T = 0$.

APM-PP: affine-permutation-matrix pair-partition codes.

Kasai: structured affine blocks. CPM-PP: every multiplier $a = 1$.

K. Kasai, Quantum 10, 2205 (2026): [[9216, 4612, $d \leq 48$]], a distance upper bound. APM-PP relation: Okada-Kasai, arXiv:2607.14091v4, Sec. VII.

Only selected check rows are shown. The full Kasai construction additionally controls nonorthogonality of complementary parent rows. *I didn't name these codes after myself.*

Kasai-based and GALA codes connect algebra to atom arrays

Crossed acousto-optic deflectors (AODs) support coordinated row and column motion in reconfigurable neutral-atom arrays.

Zhao et al.: Kasai-based codes

Additional affine-permutation conditions simplify atom rearrangement between syndrome-extraction gate layers.

Layouts, movement-time estimates, and circuit-noise simulations.

Yang et al.: GALA

Group-Action Lifts with Active orthogonality

Group structure supplies explicit AOD schedules and logical operations.

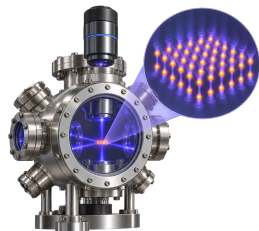
Includes selected hardware-compatible Kasai constructions.

Evidence: code design and simulations, not experimental demonstrations. Both studies exclude idling errors from their circuit-noise simulations.

Both studies connect algebraic structure to hardware constraints.

Chen Zhao, Casey Duckering, Andi Gu, Nishad Maskara, Hengyun Zhou, arXiv:2604.16209v2, Secs. 2–3, App. E.

Willers Yang, Casey Duckering, Arpit Dua, *Designer Codes from GALA*, arXiv:2608.07431v2, Secs. S2.2 (Prop. 5), S5.1, S6.



Neutral-atom platform

Conceptual illustration

Pair-partition codes combine high rate with certified distance

Goal: lower the qubit cost of fault-tolerant quantum computing

High data-qubit rate: choosing fewer check rows than qubits guarantees capacity.

Compatible sparse checks: pair equal overlap products to enforce commutation.

Certified code distance: exclude lighter logicals and verify a matching witness.

$$[[320, 70, 20]] \quad [[944, 478, 20]]$$

Halving: compact non-CSS blocks, including $[[200, 43, 20]]$.

Circuit-level tests: promising results for selected halved codes.

Beyond distance: complete low-weight canonical logical bases.

Open questions for quantum error correction

Study decoding and logical operations for these concrete code families.

Evaluate total overhead under realistic circuit and connectivity constraints.

I would welcome your ideas and opportunities to collaborate.

Thank you. Questions are welcome.