Research Square

Preprints are preliminary reports that have not undergone peer review.
They should not be considered conclusive, used to inform clinical practice,
or referenced by the media as validated information.

Planar low-overhead quantum memories with only nearest-neighbour gates

Jens Eisert

jenseisert@gmail.com

Freie Universität Berlin https://orcid.org/0000-0003-3033-1292

Boren Gu

Freie Universität Berlin

Tamas Noszko

The University of Edinburgh

Vincent Steffan

IQM Germany

Jens Eberhardt

Johannes Gutenberg-Universität Mainz

Joschka Roffe

The University of Edinburgh https://orcid.org/0000-0001-9202-1156

Stergios Koutsioumpas

University of Edinburgh

Physical Sciences - Article

Keywords:

Posted Date: November 3rd, 2026

DOI: https://doi.org/10.21203/rs.3.rs-11493826/v1

License: This work is licensed under a Creative Commons Attribution 4.0 International License.
Read Full License

Additional Declarations: There is **NO** Competing Interest.

Planar low-overhead quantum memories with only nearest-neighbour gates

Boren Gu,[1,*] Tamas Noszko,[2,†] Vincent Steffan,[3] Jens Niklas Eberhardt,[4]
Joschka Roffe,[2] Jens Eisert,[1,5,‡] and Stergios Koutsioumpas[2,§]

[1]*Dahlem Center for Complex Quantum Systems, Freie Universität Berlin, Berlin, 14195, Germany*
[2]*Quantum Software Lab, The University of Edinburgh, Edinburgh, EH8 9AB, United Kingdom*
[3]*IQM Quantum Computers, Munich, 80992, Germany*
[4]*Institute of Mathematics, Johannes Gutenberg-Universität Mainz, Mainz, 55128, Germany*
[5]*Helmholtz-Zentrum Berlin für Materialien und Energie, Berlin, 14109, Germany*

High-performance quantum low-density parity-check codes promise substantial reductions in the overhead of fault-tolerant quantum computation, but most constructions require long-range connectivity or qubit shuttling, both of which are difficult to realise in superconducting architectures. Here we introduce a family of quantum low-density parity-check codes that, for the first time, combines planar open-boundary layouts, finite-size advantages over surface codes, and syndrome extraction using only nearest-neighbour gates on a planar grid of qubits with either square or hexagonal connectivity. The resulting circuits achieve optimal constant-depth stabiliser measurement, independent of code size, and naturally enable periodic leakage removal by exchanging the roles of check and data qubits between syndrome-extraction rounds. We find finite-size instances such as a $[[323, 14, 15]]$ code, whose code-efficiency ratio is nearly an order of magnitude larger than that of rotated surface-code patches. At around 40 circuit qubits per logical qubit, the best directional tile-code layouts reduce the per-logical per-round logical error rate by a factor of 100 relative to rotated surface-code memories. These results show that the advantages of quantum low-density parity-check codes can survive compilation into strictly planar nearest-neighbour circuits, bringing low-overhead fault-tolerant memories closer to near-term hardware.

I. INTRODUCTION

Quantum error correction is a central requirement for turning noisy quantum devices into reliable machines that can perform useful computations beyond what is feasible classically. Indeed, fault-tolerant operations are expected to be essential for implementing algorithms beyond logarithmic depth [1–5]. The core question is therefore not whether error correction is needed, but which codes can deliver the required logical protection within the geometric and connectivity constraints of real hardware.

The surface code has long been the leading answer, especially for superconducting processors where two-qubit gates are typically restricted to nearest-neighbour interactions on a two-dimensional grid [6–11]. Its appeal lies in its locality: stabilisers are geometrically local and can be measured by short-depth nearest-neighbour circuits. This hardware compatibility, however, comes at the cost of a low encoding rate. A rotated surface-code patch with distance d encodes only one logical qubit into d^2 data qubits, leading to substantial physical-qubit overhead for large fault-tolerant computations.

Quantum low-density parity-check (qLDPC) codes offer a route beyond this overhead [12–19]. By encoding many logical qubits in a single block while retaining bounded-weight stabilisers, qLDPC codes can in principle reduce the footprint of fault-tolerant memories far below that of separate surface-code patches. Recent constructions have made this promise increasingly concrete, with improved asymptotic resource scaling and striking finite-size reductions in physical-qubit overhead [20–24]. The central obstacle is implementation: high-performance qLDPC codes typically require check-data interactions that are non-local in any planar embedding, and hence rely on qubit shuttling or reconfigurable connectivity [25–27]. In superconducting processors, one route to such connectivity is provided by flip-chip architectures, which can route signals vertically through the processor stack to couple qubits that are not nearest neighbours in the plane [17, 28–30]. This approach is in principle powerful, but it introduces additional three-dimensional integration and associated substantial fabrication complexity.

This tension has a rigorous asymptotic origin. Stabiliser codes with geometrically local checks on a two-dimensional lattice obey strong restrictions on their parameters, including bounds of the form $kd^2 = O(n)$ [31–33]. Such results rule out simultaneously high rate and large distance in the large-block-length limit under static two-dimensional locality. They do not, however, settle the finite-size question most relevant to early fault-tolerant devices [34]: can one obtain a substantial overhead advantage over the surface code while retaining a strictly planar nearest-neighbour implementation?

We answer this question affirmatively by introducing *directional tile codes*: planar qLDPC memories whose syndrome-extraction circuits are implemented on a square or hexagonal grid layout using only nearest-neighbour iSWAP gates [38–43], up to single-qubit Clifford rotations. The method leverages the iSWAP interaction as a spacetime routing primitive, extending ideas from dynamic surface codes [38, 43] and toric directional codes [35]. During syndrome extraction, each iSWAP-based interaction both couples neighbouring states and exchanges their position. Each stabiliser is therefore measured as a nearest-neighbour walk on the hardware graph, rather than through a static long-range coupling graph.

Due to this mechanism, directional tile codes are especially promising with regard to leakage noise [44, 45], a crit-

* b.gu@fu-berlin.de
† tamas.noszko@ed.ac.uk
‡ jense@zedat.fu-berlin.de
§ skoutsio@ed.ac.uk

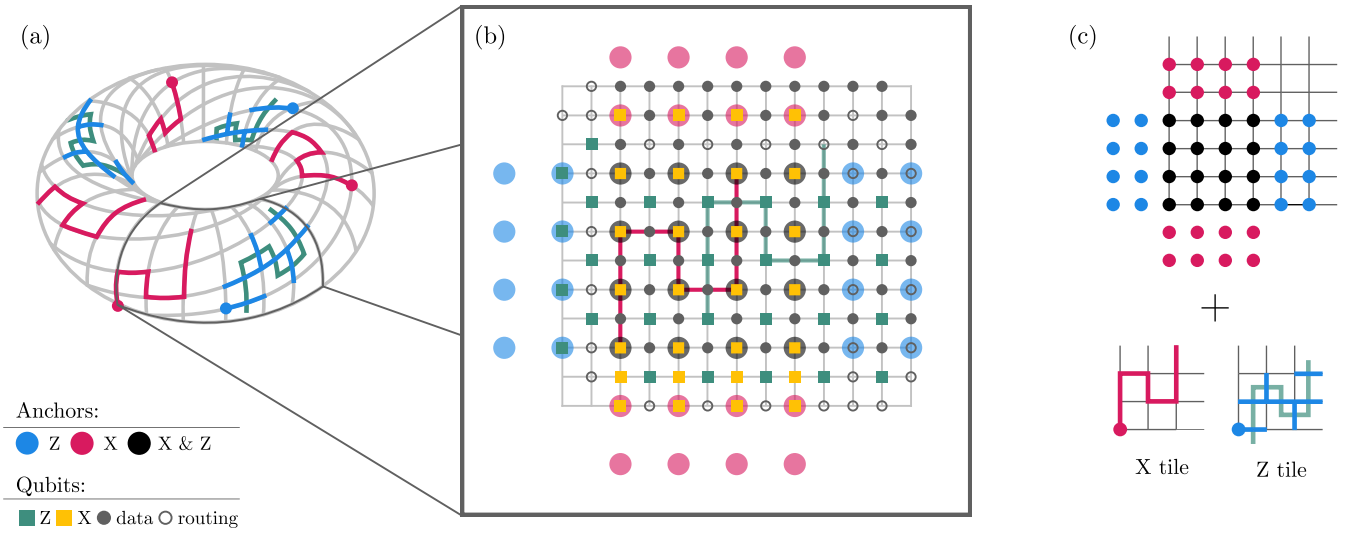


FIG. 1. **Planar directional tile codes.** A directional tile code can be viewed either as an open-boundary patch cut from a toric directional code [35], or as a planar tessellation by connected-string tiles in the spirit of tile codes [36, 37]. **(a) Toric directional code.** A toric directional code on a twisted torus, with data qubits on edges and X- and Z-type checks on vertices and plaquettes. Each stabiliser follows the directional word $\mathfrak{D} = \mathtt{N}^2\mathtt{ESEN}^2$. **(b) Planar directional tile code (this work).** The corresponding open-boundary $\mathtt{N}^2\mathtt{ESEN}^2$ directional tile code with parameters $[[60, 4, 5]]$, embedded in a hardware-like square-grid layout. Dark grey dots are data qubits, coloured squares are check qubits, and unfilled circles are routing sites for nearest-neighbour measurement walks. Highlighted X- and Z-type stabilisers have the same connected-string shape and are measured by simultaneous CXSWAP walks. The displayed layout is routing-optimised, with shifted and pruned boundary qubits; see Figure 7. Background colours mark tile anchors. **(c) Tile-code construction.** Equivalent planar tessellation by directional X- and Z-tiles whose supports form the same connected string on primal and dual lattices.

ical bottleneck for running quantum error correction protocols on superconducting hardware. On the one hand, iSWAP gates can naturally minimise leakage generation and transport [38, 46, 47]. On the other hand, our dynamic syndrome extraction circuits for directional tile codes exchange data and check qubits after each round of syndrome extraction, similarly to walking circuits for surface codes [43]. In this way, both data and check qubits are naturally reset every other round, which has been demonstrated to effectively remove the population from higher excited states [38].

The relevant benchmark is twofold. At the code level, one would like finite-size parameters that improve substantially over rotated surface-code patches [48]; a convenient measure is the *code-efficiency ratio*, kd^2/n, which equals 1 for a rotated surface-code patch with parameters $[[d^2, 1, d]]$. At the implementation level, this advantage must survive after compiling syndrome extraction into a local circuit, including all data, check, and potential routing qubits required by the hardware layout.

Directional tile codes perform strongly under both benchmarks. On the square grid, our search identifies compact instances such as the $[[323, 14, 15]]$ code, whose code-efficiency ratio is nearly an order of magnitude larger than that of a rotated surface-code patch at the same distance; see Table II. The construction further extends to a hexagonal layout with only degree-3 connectivity (Fig. 3), yielding, for example, a $[[199, 10, 7]]$ code. At the implemented-circuit level, after accounting for data, check, and routing qubits, circuit-level simulations show that the resulting nearest-neighbour layouts can outperform rotated surface-code memories at a realistic physical error rate $p = 0.001$. Across square- and hexagonal-grid connectivity, the best directional tile-code layouts both achieve up to two orders of magnitude lower per-logical per-round logical error rates than square-grid rotated surface-code memories at comparable footprints; see Figure 2.

II. PLANAR QLDPC MEMORIES FROM DIRECTIONAL WORDS

The construction is organised around *directional words*. A directional word is an ordered string of lattice steps. Geometrically, it traces a bounded-size connected string on the lattice; algebraically, this string defines the support of a stabiliser; operationally, the same ordering specifies the local measurement schedule. The paired X- and Z-type checks are obtained by placing the same string on primal and dual lattices, respectively, so that one directional word defines both the *Calderbank–Shor–Steane* (CSS) stabiliser structure and the spacetime walk used to measure it. Figure 1 illustrates this principle for the word $\mathfrak{D} = \mathtt{N}^2\mathtt{ESEN}^2$, while Figure 4 shows the nearest-neighbour physical-action layers associated with the four lattice directions.

We obtain open-boundary planar memories by using directional words as local tessellation rules. Earlier directional-code constructions realised these rules as translationally in-

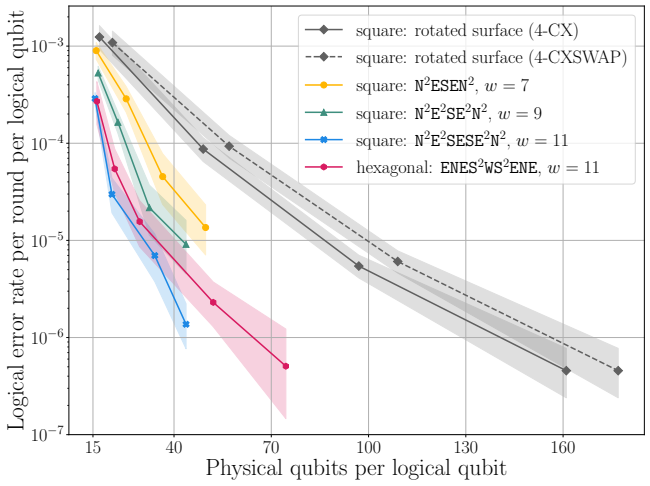


FIG. 2. **Implemented footprint advantage over rotated surface-code memories on square and hexagonal grids.** Circuit-level logical error rate averaged over X and Z bases per syndrome-extraction round per logical qubit at physical error rate $p = 0.001$, obtained from simulations of d consecutive syndrome-extraction rounds. The average is calculated as in Eq. (E1). Directional tile codes on the square grid with $w = 7, 9, 11$ and on the hexagonal grid with $w = 11$ are compared with two square-grid rotated surface-code implementations, based on CX and CXSWAP gates, respectively. For comparison at a fixed encoded memory size of 140 logical qubits, these families require 140 rotated surface-code patches; 35, 14, and 10 square-grid directional tile-code patches for $w = 7, 9, 11$, respectively; or 14 hexagonal-grid directional tile-code patches. The choice 140 is used as it is a common multiple of the logical dimensions of the codes considered. The horizontal axis gives the number of circuit qubits per logical qubit, including data, check, and routing qubits in the implemented nearest-neighbour syndrome-extraction circuits. Highlighted regions indicate hypotheses within a factor of 1000 of the maximum likelihood hypothesis, given the sampled data [49]. Representative directional tile-code instances are shown in Table II, and the full set of simulated codes is listed in Table VII.

Code family	NN	2D planar	LER gain $\approx 100\times$	Two qubit gates
Strictly local tile [50]	✓	✓	×	CP, SWAP
Bunny [51]	✓	✓	×	CP, iSWAP
Vine [52]	✓	✓	×	CP, iSWAP, SWAP
Barbell [53]	×	✓	✓	CP
Denser surface [34]	✓	✓	×	CP
Tile [36, 37]	×	✓	×	CP
Toric directional [35]	✓	×	✓	iSWAP
This work	✓	✓	✓	iSWAP

TABLE I. **Comparison of code families by nearest-neighbour (NN) implementability, planar open-boundary geometry, simulated logical-error-rate (LER) gain, and required two-qubit gates.** LER gains are relative to rotated surface-code benchmarks at $p \geq 10^{-3}$ and comparable physical overhead, following each reference's noise model and qubit-counting conventions. ✓ denotes a gain of approximately $100\times$ or more supported by directly simulated points; × denotes that this criterion is not established by the reported simulations.

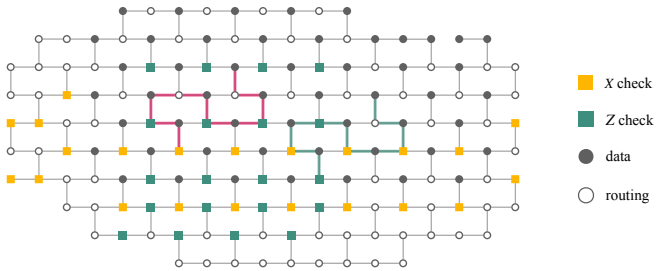


FIG. 3. **Directional tile codes on the hexagonal grid.** An $\texttt{ENES}^2\texttt{WS}^2\texttt{ENE}$ directional tile code with parameters $[[52, 10, 3]]$. The layout is rotated by 90° for presentation.

variant stabiliser patterns, with finite codes obtained by imposing periodic boundary conditions [35, 54]. Our approach is different: we apply the same local directional prescription directly to a finite planar patch, yielding hardware layouts with open boundaries rather than periodic ones. The key step is not merely to place paired connected-string X- and Z-tiles on an open patch, in the spirit of *tile codes* [36, 37, 55], but to do so while preserving the ordered walk specified by the directional word. This ensures that every bulk and boundary stabiliser remains measurable by the same nearest-neighbour iSWAP-based dynamics. In the bulk, data qubits occupy lattice edges and X- and Z-type check qubits occupy vertices and plaquettes, respectively. Near the boundary, check and data positions may be shifted along the directional words from the standard locations as part of the routing optimisation. Routing qubits are inserted only where required to keep the ordered walk nearest-neighbour on the physical square or hexagonal grid. For a directional word of weight w, one syndrome-extraction round has depth $w + 2$, including check-qubit initialisation and measurement, independent of the overall code size. Thus the same bounded object gives both bounded-weight stabilisers and optimal-depth nearest-neighbour extraction circuits.

Our construction shows that the finite-size advantages of qLDPC codes can be retained under the same square-grid nearest-neighbour constraints that underpin the hardware compatibility of surface codes, while a hexagonal-grid architecture can further relax the required connectivity. After including the routing overhead required for nearest-neighbour syndrome extraction, the implemented memories can still outperform rotated surface-code footprint curves under a uniform circuit-level noise model. This demonstrates that qLDPC overhead reductions need not rely on long-range couplers, qubit shuttling, or non-planar connectivity: the required check-data interactions can instead be generated dynamically in spacetime by native iSWAP-based walks over an open-boundary plane. Directional tile codes therefore provide a route toward low-overhead fault-tolerant memories that preserve strict two-dimensional nearest-neighbour locality while moving beyond the low en-

$\mathfrak{D}$, w	Grid	$[[n,k,d]]$	kd^2/n	LER($p = 0.001$)
$\mathtt{N^2ESEN^2}$, 7	sq.	$[[60,4,5]]$	1.67	4.54×10^{-5}
		$[[180,4,9]]$	1.80	–
$\mathtt{N^2E^2SE^2N^2}$, 9		$[[162,10,6]]$	2.22	9.10×10^{-6}
		$[[351,10,9]]$	2.31	–
$\mathtt{N^2E^2SESE^2N^2}$, 11		$[[241,14,12]]$	8.37	1.37×10^{-6}
		$[[323,14,15]]$	9.75	–
$\mathtt{ENES^2WS^2ENE}$, 11	hex.	$[[52,10,3]]$	1.73	2.71×10^{-4}
		$[[295,10,9]]$	2.75	5.07×10^{-7}

TABLE II. **Small examples of directional tile codes and their circuit-level performance.** For each directional word $\mathfrak{D}$, w denotes the stabiliser weight. "sq." and "hex." denote square- and hexagonal-grid connectivity of degree 4 and 3, respectively. All listed codes admit a nearest-neighbour syndrome-extraction circuit of depth w, or $w+2$ including check- and routing-qubit initialisation and measurement. LER denotes the logical error rate per logical qubit per syndrome-extraction round at $p = 0.001$. Monte Carlo simulations were restricted to at most 50 circuit qubits per logical qubit for square-grid codes, with additional hexagonal-grid instances simulated beyond this range; entries marked by "–" were not simulated.

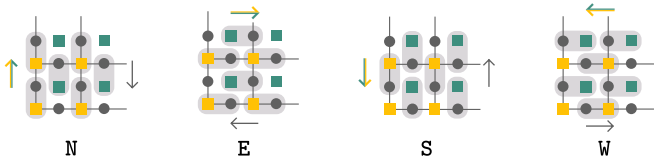


FIG. 4. **Nearest-neighbour directional layers [35].** Local physical action associated with the four elementary directions $\{\mathtt{N},\mathtt{E},\mathtt{S},\mathtt{W}\}$ in a directional word. Coloured check qubits move along the indicated direction by nearest-neighbour exchange operations, while interacting with data qubits via CX gates whenever the walk encounters a data qubit. Specifically, we apply CXSWAP(X-check, data) and CXSWAP(data, Z-check) respectively for different types of check-data pairs.

coding rate of the surface code.

III. NEAREST-NEIGHBOUR SYNDROME EXTRACTION

We now make the iSWAP-native syndrome-extraction circuit explicit. The input is a *directional word*

$$\mathfrak{D} = \vec{d}_1\vec{d}_2\cdots\vec{d}_w, \qquad \vec{d}_i \in \{\mathtt{N},\mathtt{E},\mathtt{S},\mathtt{W}\},$$

together with a square- or hexagonal-grid hardware layout consisting of data qubits D, X- and Z-type check qubits C_X, C_Z, and routing qubits R. The word $\mathfrak{D}$ specifies a sequence of nearest-neighbour directions. During syndrome extraction, each check qubit follows this sequence as a spacetime walk: whenever the walk encounters a data qubit, the circuit accumulates the corresponding stabiliser parity, and whenever it encounters a routing site, the quantum state is moved to keep the walk nearest-neighbour. Thus the ordered support of the stabiliser also guides the local schedule used to measure it.

The circuit uses only nearest-neighbour iSWAP gates and single-qubit Clifford rotations. We express each data-check interaction in terms of a CXSWAP gate, locally equivalent to an iSWAP gate [43, 56]

$$\begin{aligned}\mathrm{CXSWAP}(q_0,q_1) &= (HS^\dagger\otimes S^\dagger)\,\mathrm{iSWAP}(q_0,q_1)\,(I\otimes H),\\ &= \mathrm{SWAP}(q_0,q_1)\,\mathrm{CX}(q_0,q_1).\end{aligned}$$

The CX part accumulates the stabiliser eigenvalue on the check qubit, while the SWAP part advances the relative positions of the check, data and routing qubits. For X-type checks the check qubit acts as the control, whereas for Z-type checks the orientation is reversed, after the usual basis change; see Figure 4.

One round of syndrome extraction is given in Algorithm 1. The algorithm iterates through the directional word $\mathfrak{D} = \vec{d}_1\cdots\vec{d}_w$, applying one nearest-neighbour directional layer for each step $\vec{d}_i$. Consecutive rounds alternate between the word $\mathfrak{D}$ and the inverse word, so that the physical layout is restored without introducing long-range operations.

Algorithm 1 Nearest-neighbour syndrome extraction

Require: Directional word $\mathfrak{D} = \vec{d}_1\cdots\vec{d}_w$ and a square or hexagonal grid layout with data qubits D, check qubits C_X, C_Z, and routing qubits R
Ensure: One syndrome-extraction round using only nearest-neighbour iSWAP-based operations

Initialise C_X in $|+\rangle$ and C_Z in $|0\rangle$
Prepare routing inputs in $|0\rangle$ or $|+\rangle$ before use
for $i = 1,\ldots,w$ **do**
  for all $q \in C_X \cup C_Z$ **do**
    if $q^{\vec{d}_i} \in D$ **then** ▷ neighbour in direction $\vec{d}_i$
      if $q \in C_X$ **then**
        Apply CXSWAP$(q, q^{\vec{d}_i})$
      else
        Apply CXSWAP$(q^{\vec{d}_i}, q)$
      end if
    else if $q^{\vec{d}_i} \in R$ and $q^{\vec{d}_i}$ is prepared in $|0\rangle$ **then**
      Apply CXSWAP$(q^{\vec{d}_i}, q)$
    else if $q^{\vec{d}_i} \in R$ **then** ▷ prepared in $|+\rangle$
      Apply CXSWAP$(q, q^{\vec{d}_i})$
    end if
  end for
  for all $q \in R$ **do**
    if $q^{\vec{d}_i} \in D$ and q is prepared in $|0\rangle$ **then**
      Apply CXSWAP$(q, q^{\vec{d}_i})$
    else if $q^{\vec{d}_i} \in D$ **then** ▷ q prepared in $|+\rangle$
      Apply CXSWAP$(q^{\vec{d}_i}, q)$
    end if
  end for
  Update qubit locations after this layer
end for
Measure C_X in the X basis and C_Z in the Z basis

The routing moves in Algorithm 1 do not require bare SWAP gates, which are less natural on many superconducting platforms [40, 57, 58]. Instead, the orientation of each routing CXSWAP is chosen according to the prescribed routing preparation: a $|0\rangle$ routing input acts as the control of the CX component, whereas a $|+\rangle$ routing input acts as its target. The CX component acts trivially on the ideal input, giving

$$\mathrm{CXSWAP}(|0\rangle_r, q) = \mathrm{SWAP}(|0\rangle_r, q),$$
$$\mathrm{CXSWAP}(q, |+\rangle_r) = \mathrm{SWAP}(q, |+\rangle_r),$$

where r denotes the routing input, q is a data or check qubit, and the first argument of CXSWAP specifies the control of its CX component. As such, both parity check-data interactions and routing moves are compiled into the same nearest-neighbour iSWAP-based gate family.

Faults on routing qubits, or faults occurring during routing steps, can propagate to data or check qubits. This propagation remains bounded: in one syndrome-extraction round, each data or check qubit participates in at most w two-qubit interactions, including those involving routing qubits; see Lemma 2 in Appendix A. Routing qubits also provide additional control points in the circuit. They may be reset after each routing step, or measured so that non-trivial outcomes supply flag information for faults occurring along the routing path; see Appendix E. For a directional word of weight w, one syndrome-extraction round has optimal depth w, or depth $w+2$ including check-qubit initialisation and measurement.

This exchange-based formulation is also well aligned with recent leakage-aware approaches to superconducting quantum error correction. Time-dynamic surface-code circuits have shown that swapping data and measurement roles can help remove accumulated non-computational population, and related experiments have demonstrated surface-code cycles using iSWAP-family gates [38, 43]. Our circuits naturally support the same data-check exchange mechanism and, moreover, allow routing qubits to be reset or measured during the walk, providing built-in locations where leakage-mitigation or flagging strategies could be incorporated. We do not model leakage below, so we use this point only to highlight compatibility with existing leakage-mitigation strategies.

IV. DIRECTIONAL TILE CODES

A *tile* is a subset of edges of a $B \times B$ grid, excluding the edges on the top row and the rightmost column. We identify edges with physical data qubits, so that a tile specifies the support of an X-type or Z-type stabiliser check. To ensure that the corresponding X- and Z-type stabilisers commute, the two tiles are required to satisfy the *mutual condition*: for each horizontal, respectively vertical, edge of the X-tile with coordinate (a, b), the Z-tile contains the vertical, respectively horizontal, edge with coordinate $(B-1-a, B-1-b)$ [36].

A pair of X- and Z-tiles is called *directional* if the support of one tile forms an ordered connected string, and the other tile contains the same string on the dual lattice. Such a pair is labelled by the corresponding sequence of lattice directions, the *directional word*, $\mathfrak{D} = \vec{d}_1\vec{d}_2\cdots\vec{d}_w$. For example, the $\mathtt{N}^2\mathtt{ESEN}^2$-tiles are shown in Figure 1 (c).

The mutual condition guarantees commutation of the static stabiliser generators, but by itself it does not guarantee that the syndrome-extraction circuit in Algorithm 1 is deterministic. We therefore impose an additional parity condition, obtained as a simplified form of Theorem 1 with Layout 1 of Ref. [35]: every displacement vector of the ordered connected string with odd vertical displacement must occur an even number of times. Here, a displacement vector is the vector from one edge in the ordered support of the connected string to a later edge, with the ordering fixed by the directional word; see Figure 6.

Definition 1 (Directional tiles)**.** A pair of X- and Z-tiles is called *directional* if the tiles satisfy the mutual condition, one tile forms an ordered connected string labelled by a directional word $\mathfrak{D} = \vec{d}_1\vec{d}_2\cdots\vec{d}_w$, and the other tile contains the same string on the dual lattice. In addition, every displacement vector of the ordered connected string with odd vertical displacement must occur with even multiplicity.

The *directional tile codes* are obtained by tessellating a pair of directional tiles on an $(M+B-1)\times(N+B-1)$ rectangular grid, again with the edges on the top row and the rightmost column removed. Starting from the bottom-left vertex, we choose the vertices of an $M \times N$ subgrid as *anchors*. In the bulk, both the X- and Z-tiles are supported at each anchor. Along the boundary, X- and Z-type tiles are placed separately, as shown in Figure 1 (b) and (c), so that any X-boundary tile and Z-boundary tile either have disjoint support or intersect only on edges contained in the prescribed layout.

After tessellation, we prune unnecessary boundaries. Edges that are contained in tiles of at most one type are removed, together with any stabiliser tiles whose support becomes empty after this pruning step. The remaining edges define the data qubits, while the remaining tile anchors define the positions of the X- and Z-type check qubits. This gives a complete CSS code.

To realise the resulting directional tile code on a planar grid device using only nearest-neighbour iSWAP gates, we embed the data and check qubits into the square or hexagonal-grid hardware layout. Near the boundary, the tessellation can leave gaps between check qubits and the data qubits with which they must interact during the ordered walk. These gaps are filled with *routing qubits*, shown as uncoloured circles in Figure 1 (b) and Figure 7, so that every step of the syndrome-extraction circuit remains nearest-neighbour.

We observe that logical operators in directional tile codes have a surface-code-like boundary-to-boundary structure; see Figure 9. Consistently with this picture, the X- and Z-type distances scale linearly with the two bulk dimensions and are primarily controlled by M and N, respectively; see Table IV. For uniform-distance instances, $d = d_X = d_Z$, this gives a natural way to compare the implemented circuit size n_{circ} as a function of d; see Figure 8. For the lower-weight directional words $\mathtt{N}^2\mathtt{ESEN}^2$ and $\mathtt{N}^2\mathtt{E}^2\mathtt{SE}^2\mathtt{N}^2$, of weights 7 and 9, respectively, the resulting circuit-qubit count follows an exact quadratic dependence on d, as expected from a two-dimensional patch whose two linear dimensions

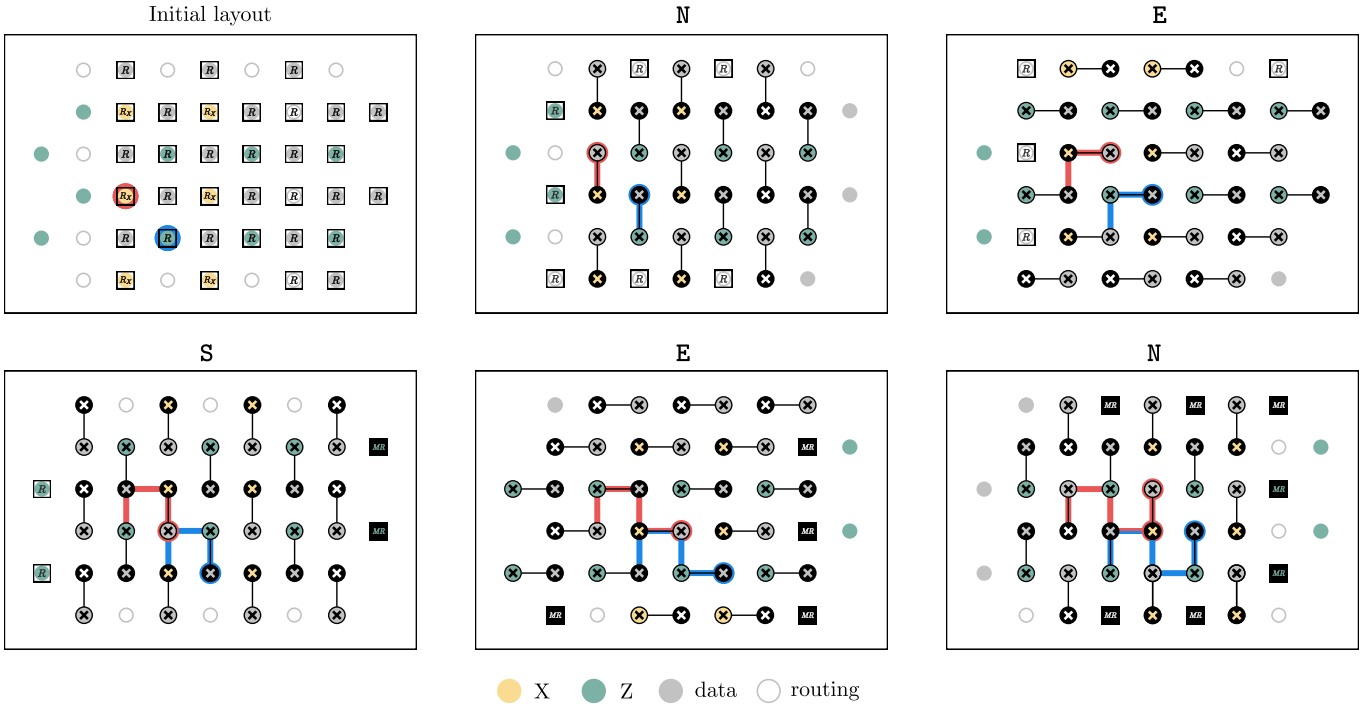


FIG. 5. **Example of one round of nearest-neighbour syndrome extraction.** Spacetime trajectory of the nearest-neighbour CXSWAP syndrome-extraction circuit for a $[[18, 2, dx = 3, dz = 2]]$ directional tile code on a square grid with directional word $\mathfrak{D} = \texttt{NESEN}$. The top-left panel shows the initial layout, while the following panels show the five directional layers N, E, S, E, and N. In each layer, check qubits are moved along the prescribed direction by parallel nearest-neighbour CXSWAP operations, as described in Algorithm 1. The red and blue paths highlight representative X- and Z-type check trajectories during the same extraction round. The circuit shown includes boundary scheduling optimisations: qubits whose first interaction occurs only in a later layer are reset only immediately before use, and check qubits whose final interaction has already occurred are measured as soon as possible. Consequently, some reset and measurement operations appear inside the directional sequence rather than only at the beginning or end of the round; these scheduling choices reduce idle time without changing the stabilisers being measured.

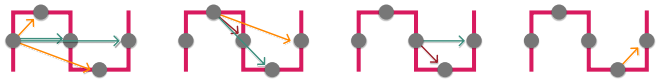


FIG. 6. **Examples of displacement vectors**. The connected string of the X-tile of the directional word $\mathfrak{D} = \texttt{NESEN}$, with data qubits highlighted by grey dots. Displacement vectors are illustrated for each starting point. Green vectors have even vertical displacement, while the different shades of yellow indicate pairs of vectors with odd vertical displacement.

both grow with distance. By contrast, for the weight-11 word $\texttt{N}^2\texttt{E}^2\texttt{SESE}^2\texttt{N}^2$, the same finite-size data are much better described by an almost linear dependence of n_{circ} on d. This finite-size behaviour remains consistent with known two-dimensional locality bounds [31–33], while revealing a favourable finite-distance regime relevant to near-term overhead comparisons.

Beyond quantum memory, the tile-code structure provides routes to logical Clifford operations. Using the canonical logical Pauli basis of tile codes [59], we analyse operations arising from derived automorphisms and geometric code symmetries. Derived automorphisms temporarily enlarge the patch along one boundary and induce logical Clifford actions that decompose into products of logical CNOT gates. For suitable palindromic directional words, reflection symmetries also define code automorphisms whose logical actions decompose into products of SWAP and CNOT gates [60]. These mechanisms offer candidates for intra-patch logical operations, complementing inter-patch lattice surgery [61–64]. Appendix D describes their logical action and additional symmetries identified through automated searches. Assessing their full fault-tolerant implementation cost under planar nearest-neighbour constraints remains an important direction for future work.

V. ROUTING OVERHEAD AND OPTIMISATION

The routing overhead introduced by our scheme is boundary dominated and scales linearly with the relevant code distances. We first prove a general boundary-scaling bound in Lemma 3: for any family of directional tile codes generated from a fixed directional word $\mathfrak{D}$, the number of routing qubits required for nearest-neighbour syndrome extraction satisfies

$$n_r = O(\sqrt{n}).$$

This follows because routing qubits are introduced only near the boundary of the planar layout, whose size grows with the linear dimension of the code. More concretely, the logical operators have a boundary-to-boundary structure, so the X- and Z-type distances are controlled by the two bulk dimensions, M and N, respectively; see Table IV. The same data show that both the standard and optimised routing overheads are linear functions of M and N. Thus, for a fixed directional word, the routing overhead is naturally expressed as

$$n_r = O(d_X + d_Z),$$

and reduces to $n_r = O(d)$ in balanced families where d_X and d_Z are comparable. This distance-linear behaviour is shown explicitly in Figure 8 for the uniform-distance slice d, where both the standard and optimised routing overheads grow approximately linearly with d, while the optimisation yields substantial constant-factor savings across all directional words. This is analogous to the routing overhead recently proved for colour-code constructions [56] and in correspondence with circuit-qubit number scaling we observed.

Since the routing overhead comes primarily from boundary routing qubits, we reduce it by removing routing sites that are free during a syndrome-extraction round. Concretely, if a routing qubit acts only as a terminal relay for a single data or check qubit q, and is not required between two data-check interactions involving q, we absorb this routing site by shifting the corresponding start or end position of q to that site; see Figure 1 (b), Figure 5, and Figure 7. We iterate this local simplification until no further terminal routing sites can be removed. The resulting circuit is accepted only after verifying that the measured stabiliser supports, detectors, and logical observables are unchanged. A detailed description of the optimisation procedure is given in Appendix C.

In multi-patch architectures, part of the routing overhead may be shared with other auxiliary regions already required for logical operations. Lattice-surgery protocols, for example, typically use padding regions between neighbouring patches as auxiliary degrees of freedom for patch merging and logical Pauli measurements [61–63]. From this perspective, boundary routing qubits need not always be viewed as a completely separate overhead: in suitable layouts, they may also serve as inter-patch padding for logical operations, or as auxiliary degrees of freedom for intra-patch logical operations derived from automorphisms [59]; see Appendix D. Since these routing qubits can be periodically reset, the same layout is also compatible with leakage-mitigation strategies used in time-dynamic and iSWAP-based circuits [38, 43].

We also optimise the timing of auxiliary qubit preparation and readout to reduce exposure to idle noise. Routing qubits are reset as late as possible before they are required to interact with a check or data qubit. Following a routing CXSWAP, the outgoing auxiliary state is measured in the corresponding basis at the earliest available time step, provided it is not required to interact in an entangling gate in the following time step. Routing inputs that are needed again are reset immediately before their next use. We apply a similar optimisation to boundary checks with shorter interaction sequences: their accumulated parities are measured as soon as the required interactions are complete, and the check inputs for the following syndrome-extraction round are reset at its start. Preparation and readout are scheduled alongside operations on disjoint qubits, without adding circuit layers. These changes shorten the intervals during which idle errors can propagate to the data or corrupt a measurement.

VI. SUMMARY AND OUTLOOK

We have introduced directional tile codes: a family of planar qLDPC memories whose stabilisers and syndrome-extraction circuits are specified by the same directional word. Each word defines a bounded-size connected string that serves both as the support of a CSS stabiliser and as the ordered nearest-neighbour walk used to measure it. By exploiting the exchange character of the iSWAP gate, these walks generate the required check-data connectivity dynamically in spacetime, rather than requiring a static long-range coupling graph. For a directional word of weight w, one syndrome-extraction round has depth $w+2$, including check-qubit preparation and measurement, independent of the code size.

This construction weakens the usual dichotomy between surface-code locality and qLDPC efficiency. The surface code is naturally compatible with planar nearest-neighbour superconducting hardware, but pays a large price in encoding rate. Conversely, many high-performance qLDPC constructions achieve much better finite-size parameters, but rely on long-range connectivity or qubit shuttling. Directional tile codes show that this trade-off is not absolute in the finite-size regime relevant to early fault-tolerant devices. We find compact open-boundary instances with high code efficiency. After compiling syndrome extraction into strictly planar nearest-neighbour circuits, this advantage persists at the implemented-layout level: in circuit-level simulations, the best directional tile-code layouts reduce the per-logical per-round logical error rate by around two orders of magnitude relative to rotated surface-code memories at a comparable footprint of about 40 circuit qubits per logical qubit on a square grid. Moreover, even when the directional tile codes are restricted to the lower-degree connectivity of a hexagonal grid, they retain nearly a two-order-of-magnitude reduction in logical error rate relative to rotated surface-code memories implemented on the square grid.

Although the implemented layouts already show substantial advantages, further gains are likely possible through more aggressive routing optimisations and a deeper co-design of directional words, planar layouts, syndrome-extraction schedules, and decoders. Several limitations and open directions remain. First, the present numerical evidence is finite-size and based on particular families of directional words; a more systematic search over words, layouts, schedules, and routing optimisations may reveal substantially better instances. Since multiple directional words are allowed at a fixed stabiliser weight, and since these words can realise codes with the same or different logical dimensions, it would also be interesting to develop code-switching techniques between directional tile codes defined by different words. Such techniques

could provide additional flexibility for logical operations, layout deformation, or architecture-level compilation. Second, our circuit-level simulations use a simplified noise model. It will be important to optimise these circuits under biased noise, coherent circuit noise, and hardware-calibrated error models. Leakage is another natural direction [38, 43]: although the resettable routing qubits provide possible locations for leakage removal and flag information, leakage has not been explicitly modelled here. Third, these memories should be integrated with logical operations. Promising routes include lattice surgery [61] between directional tile patches and intra-patch operations derived from code automorphisms [59, 60], but their full circuit-level cost in planar layouts remains to be assessed. Finally, architecture-level compilation will be needed to understand how directional tile memories perform inside larger fault-tolerant layouts, where routing regions may also serve as inter-patch workspace.

Taken together, these results suggest that qLDPC-like finite-size advantages can survive the stringent locality constraints of planar superconducting hardware when the code, native gates, and syndrome-extraction dynamics are co-designed. Rather than treating connectivity as a fixed hardware resource, directional tile codes use local exchange gates to generate the required connectivity during the error-correction cycle itself. This provides a route toward lower-overhead fault-tolerant quantum memories on strictly planar nearest-neighbour devices.

ACKNOWLEDGEMENTS

We acknowledge helpful discussions with David Byfield, Stefan Filipp, Julio Carlos Magdalena De La Fuente, György P. Gehér, Archibald Ruban and Yifan Tang.

We thank Georgia M. Nixon, Campbell K. McLauchlan, and Charles Van Rest for coordinating the submission of their independently developed related manuscript, *Vine Codes: Low-Overhead Quantum LDPC Codes on a Planar Square Grid* [52], which addresses closely related questions concerning low-overhead quantum LDPC codes on planar square-grid architectures.

FUNDING

The Berlin team has been supported by the BMFTR (QSolid, MUNIQC-Atoms, PasQuops), the DFG (CRC 183, BoLaCo, and SPP 2514), the Quantum Flagship (Millenion, PasQuanS2), the Munich Quantum Valley, Berlin Quantum, QuantERA (SDPCode), and the European Research Council (DebuQC). TN is supported by the Engineering and Physical Sciences Research Council (grant number EP/W524384/1), the University of Edinburgh and Quantinuum. JR is funded by an EPSRC Quantum Career Acceleration Fellowship (grant code: UKRI1224). JR, SK, and BG were supported by the Innovate UK project "QEC Readout Testbed" [reference number 10151107]. SK and JR were additionally supported by an EPSRC IAA Cross Institutional Project between University of Edinburgh and University of Glasgow.

AUTHOR CONTRIBUTIONS

B.G. initiated the project, developed the directional tile code construction and its theoretical framework, and proved the routing-overhead bounds. T.N. developed the code for code construction, circuit design, initial routing optimisation and prepared all figures. S.K. led the project, performed all simulations and decoding, refined the routing optimisation and generalised the layout to degree-3 hexagonal connectivity. V.S. and J.N.E. contributed ideas for leakage removal and the use of derived automorphisms. J.R. and J.E. provided scientific supervision and guided the development of the research. B.G., S.K., V.S. and J.N.E. searched for candidate codes. All authors contributed to writing and revising the manuscript.

COMPETING INTERESTS

The authors declare no competing interests.

DATA AVAILABILITY

The simulation data, layouts, parity-check matrices and circuits used in this paper's decoding simulations can be found from Zenodo: https://doi.org/10.5281/zenodo.23400699.

[1] E. T. Campbell, B. M. Terhal, and C. Vuillot, Roads towards fault-tolerant universal quantum computation, Nature **549**, 172 (2017).
[2] J. Roffe, Quantum error correction: An introductory guide, Contemp. Phys. **60**, 226 (2019).
[3] J. Eisert and J. Preskill, Mind the gaps: The fraught road to quantum advantage, (2025), arXiv:2510.19928.
[4] A. A. Mele, A. Angrisani, S. Ghosh, *et al.*, Noise-induced shallow circuits and absence of barren plateaus, Nature Phys. **22**, 751 (2026).
[5] D. Stilck França and R. García-Patrón, Limitations of optimization algorithms on noisy quantum devices, Nature Phys. **17**, 1221 (2021).
[6] E. Dennis, A. Kitaev, A. Landahl, *et al.*, Topological quantum memory, J. Math. Phys. **43**, 4452 (2002).
[7] A. Y. Kitaev, Fault-tolerant quantum computation by anyons, Ann. Phys. **303**, 2 (2003).
[8] Google Quantum AI and collaborators, Quantum error correction below the surface code threshold, Nature **638**, 920 (2025).
[9] S. Krinner, N. Lacroix, A. Remm, *et al.*, Realizing repeated quantum error correction in a distance-three surface code, Na-

ture **605**, 669 (2022).
[10] Y. Zhao, Y. Ye, H.-L. Huang, *et al.*, Realization of an Error-Correcting Surface Code with Superconducting Qubits, Phys. Rev. Lett. **129**, 030501 (2022).
[11] C. Gidney, M. Newman, P. Brooks, *et al.*, Yoked surface codes, Nature Comm. **16**, 4498 (2025).
[12] N. P. Breuckmann and J. N. Eberhardt, Quantum Low-Density Parity-Check Codes, PRX Quantum **2**, 040101 (2021).
[13] P. Panteleev and G. Kalachev, Degenerate Quantum LDPC Codes With Good Finite Length Performance, Quantum **5**, 585 (2021).
[14] A. Leverrier and G. Zémor, Quantum Tanner codes, in *2022 IEEE 63rd Annual Symposium on Foundations of Computer Science (FOCS)* (2022) pp. 872–883.
[15] P. Panteleev and G. Kalachev, Quantum LDPC Codes With Almost Linear Minimum Distance, IEEE Trans. Inf. Th. **68**, 213 (2022).
[16] N. P. Breuckmann and J. N. Eberhardt, Balanced product quantum codes, IEEE Trans. Inf. Th. **67**, 6653 (2021).
[17] S. Bravyi, A. W. Cross, J. M. Gambetta, *et al.*, High-threshold and low-overhead fault-tolerant quantum memory, Nature **627**, 778 (2024).
[18] Z. Liang, B. Gu, Y.-A. Chen, *et al.*, Ultra-high-distance quantum memories from amplified qLDPC codes, (2026), arXiv:2609.37231.
[19] T.-G. Zhou, B. Gu, J. Eisert, *et al.*, Parallel algebraic surgery for qLDPC codes with constant shuttling depth, (2026), arXiv:2609.39874.
[20] P. Webster, L. Berent, O. Chandra, *et al.*, The Pinnacle architecture: Reducing the cost of breaking RSA-2048 to 100 000 physical qubits using quantum LDPC codes, (2026), arXiv:2602.11457.
[21] M. Cain, Q. Xu, R. King, *et al.*, Shor's algorithm is possible with as few as 10,000 reconfigurable atomic qubits, (2026), arXiv:2603.28627.
[22] C. Zhao, C. Duckering, A. Gu, *et al.*, Towards ultra-high-rate quantum error correction with reconfigurable atom arrays, (2026), arXiv:2604.16209.
[23] S. Khan, S. Sethi, K. Sahay, *et al.*, Architecting early fault-tolerant neutral atom systems with quantum advantage, (2026), arXiv:2604.19735.
[24] A. Wills, M. E. Beverland, L. S. Bishop, *et al.*, Concatenating algebraic codes over high-rate quantum LDPC codes, (2026), arXiv:2605.21898.
[25] S. Dasu, M. DeCross, A. Y. Guo, *et al.*, Computing with many encoded logical qubits beyond break-even, (2026), arXiv:2602.22211.
[26] Q. Xu, J. P. Bonilla Ataides, C. A. Pattison, *et al.*, Constant-overhead fault-tolerant quantum computation with reconfigurable atom arrays, Nature Phys. **20**, 1084 (2024).
[27] D. Bluvstein, S. J. Evered, A. A. Geim, *et al.*, Logical quantum processor based on reconfigurable atom arrays, Nature **626**, 58 (2024).
[28] M. Mathews, L. Pahl, D. Pahl, *et al.*, Placing and routing quantum LDPC codes in multilayer superconducting hardware, npj Quantum Inf. **12**, 114 (2026).
[29] K. Wang, Z. Lu, C. Zhang, *et al.*, Demonstration of low-overhead quantum error correction codes, Nature Phys. **22**, 308–314 (2026).
[30] M. A. Tremblay, N. Delfosse, and M. E. Beverland, Constant-overhead quantum error correction with thin planar connectivity, Phys. Rev. Lett. **129**, 050504 (2022).
[31] S. Bravyi, D. Poulin, and B. Terhal, Tradeoffs for reliable quantum information storage in 2d systems, Phys. Rev. Lett. **104**, 050503 (2010).
[32] N. Baspin and A. Krishna, Connectivity constrains quantum codes, Quantum **6**, 711 (2022).
[33] N. Delfosse, M. E. Beverland, and M. A. Tremblay, Bounds on stabilizer measurement circuits and obstructions to local implementations of quantum LDPC codes, (2021), arXiv:2109.14599.
[34] G. H. Low, W. J. Huggins, D. W. Berry, *et al.*, A denser planar surface code, (2026), arXiv:2605.30455.
[35] G. P. Gehér, D. Byfield, and A. Ruban, Directional codes: a new family of quantum LDPC codes on hexagonal- and square-grid connectivity hardware, (2025), arXiv:2507.19430.
[36] V. Steffan, S. H. Choe, N. P. Breuckmann, *et al.*, High-efficiency quantum codes on a lattice with boundary, Phys. Rev. Lett. **135**, 170601 (2025).
[37] Z. Liang, J. N. Eberhardt, and Y.-A. Chen, Planar quantum low-density parity-check codes with open boundaries, PRX Quantum **6**, 040330 (2025).
[38] A. Eickbusch, M. McEwen, V. Sivak, *et al.*, Demonstration of dynamic surface codes, Nature Phys. **21**, 1994 (2025).
[39] Y. Sung, L. Ding, J. Braumüller, *et al.*, Realization of High-Fidelity CZ and ZZ-Free iSWAP Gates with a Tunable Coupler, Phys. Rev. X **11**, 021058 (2021).
[40] Z. Chen, W. Liu, Y. Ma, *et al.*, Efficient implementation of arbitrary two-qubit gates using unified control, Nature Phys. **21**, 1489 (2025).
[41] S. Inoue, R. Li, K. Kubo, Y. Ho, *et al.*, Parametrically Driven iSWAP Gate using a capacitively shunted double-transmon coupler at the zero-flux Sweet spot, (2026), arXiv:2604.27679.
[42] T. Tiwari, S. K. Sahu, G. Ribeill, *et al.*, High-fidelity iSWAP gate with Double Transmon Coupler, (2026), arXiv:2604.27080.
[43] M. McEwen, D. Bacon, and C. Gidney, Relaxing hardware requirements for surface Code circuits using time-dynamics, Quantum **7**, 1172 (2023).
[44] A. G. Fowler, Coping with qubit leakage in topological codes, Phys. Rev. A **88**, 042308 (2013).
[45] K. C. Miao, M. McEwen, J. Atalaya, *et al.*, Overcoming leakage in quantum error correction, Nature Phys. **19**, 1780–1786 (2023).
[46] M. McEwen, D. Kafri, Z. Chen, *et al.*, Removing leakage-induced correlated errors in superconducting quantum error correction, Nat. Commun. **12**, 1761 (2021).
[47] N. Lacroix, L. Hofele, A. Remm, *et al.*, Fast Flux-Activated Leakage Reduction for Superconducting Quantum Circuits, Phys. Rev. Lett. **134**, 120601 (2025).
[48] H. Bombin and M. A. Martin-Delgado, Optimal resources for topological two-dimensional stabilizer codes: Comparative study, Phys. Rev. A **76**, 012305 (2007).
[49] C. Gidney, Stim: a fast stabilizer circuit simulator, Quantum **5**, 497 (2021).
[50] Y. Bae and C.-Y. Park, Strictly local tile-code architectures on two-dimensional planar lattices, (2026), arXiv:2607.05897.
[51] R. Zhou, X. Yuan, L. Kong, *et al.*, Bunny codes: broadening superconducting quantum error correction capability through advanced control engineering, (2026), arXiv:2606.22853.
[52] G. M. Nixon, C. K. McLauchlan, and C. C. L. van Rest, Vine codes: Low-overhead quantum LDPC codes on a planar square grid, (2026), arXiv:2606.20263.
[53] S. H. Choe, V. Steffan, F. Vigneau, *et al.*, Barbell Codes: qLDPC Codes for Superconducting Quantum Hardware, (2026), arXiv:2606.06062.
[54] M. Rowshan, Structural analysis of directional qLDPC codes, (2026), arXiv:2602.19057.

[55] Z. Liang, B. Yang, J. T. Iosue, *et al.*, Algorithmic construction of gapped boundaries for topological stabilizer codes: a route to low-overhead planar qLDPC codes, PRX Quantum 10.1103/989h-xjqv (2026), accepted for publication.

[56] S. Yoshida, C. Gidney, M. McEwen, *et al.*, Low depth color code circuits with CXSWAP gate, (2025), arXiv:2510.00370.

[57] K. X. Wei, I. Lauer, E. Pritchett, *et al.*, Native Two-Qubit Gates in Fixed-Coupling, Fixed-Frequency Transmons Beyond Cross-Resonance Interaction, PRX Quantum **5**, 020338 (2024).

[58] C. Križan, J. Biznárová, L. Chen, *et al.*, Quantum SWAP gate realized with CZ and iSWAP gates in a superconducting architecture, New J. Phys. **27**, 074507 (2025).

[59] N. P. Breuckmann, S. H. Choe, J. N. Eberhardt, *et al.*, Logical operators and derived automorphisms of tile codes, Commun. Math. Phys. **407**, 205 (2026).

[60] H. Sayginel, S. Koutsioumpas, M. Webster, *et al.*, Fault-tolerant logical clifford gates from code automorphisms, PRX Quantum **6**, 030343 (2025).

[61] Y. Yang, G. Zhang, and Y. Li, Planar fault-tolerant quantum computation with low overhead, (2025), arXiv:2506.18061.

[62] C. Horsman, A. G. Fowler, S. Devitt, *et al.*, Surface code quantum computing by lattice surgery, New J. Phys. **14**, 123011 (2012).

[63] B. Gu, A. Z. Liu, A. O. Quintavalle, *et al.*, QGPU: Parallel logic in quantum LDPC codes, (2026), arXiv:2603.05398.

[64] A. J. Landahl and C. Ryan-Anderson, Quantum computing by color-code lattice surgery, (2014), arXiv:1407.5103.

[65] Z. Liang, K. Liu, H. Song, *et al.*, Generalized Toric Codes on Twisted Tori for Quantum Error Correction, PRX Quantum **6**, 020357 (2025).

[66] J. N. Eberhardt and V. Steffan, Logical operators and fold-transversal gates of bivariate bicycle codes, (2024), arXiv:2407.03973.

[67] H. Putterman, K. Noh, C. T. Hann, *et al.*, Hardware-efficient quantum error correction via concatenated bosonic qubits, Nature **638**, 927 (2025).

[68] A. Hajr, B. Qing, K. Wang, *et al.*, High-Coherence Kerr-Cat Qubit in 2D Architecture, Phys. Rev. X **14**, 041049 (2024).

[69] R. Lescanne, M. Villiers, T. Peronnin, *et al.*, Exponential suppression of bit-flips in a qubit encoded in an oscillator, Nature Phys. **16**, 509 (2020).

[70] J. P. Bonilla Ataides, D. K. Tuckett, S. D. Bartlett, *et al.*, The XZZX surface code, Nature Comm. **12**, 2172 (2021).

[71] D. K. Tuckett, S. D. Bartlett, and S. T. Flammia, Ultrahigh Error Threshold for Surface Codes with Biased Noise, Phys. Rev. Lett. **120**, 050505 (2018).

[72] K. Tiurev, A. Pesah, P.-J. H. S. Derks, *et al.*, Domain wall color code, Phys. Rev. Lett. **133**, 110601 (2024).

[73] J. Roffe, L. Z. Cohen, A. O. Quintavalle, *et al.*, Bias-tailored quantum LDPC codes, Quantum **7**, 1005 (2023).

[74] Q. Xu, N. Mannucci, A. Seif, A. Kubica, *et al.*, Tailored XZZX codes for biased noise, Phys. Rev. Res. **5**, 013035 (2023).

[75] J. Das, S. Dhara, P. Medina, *et al.*, Clifford-deformed zero-rate LDPC codes with 50% biased noise thresholds, (2026), arXiv:2605.15348.

[76] A. S. Darmawan, B. J. Brown, A. L. Grimsmo, *et al.*, Practical Quantum Error Correction with the XZZX Code and Kerr-Cat Qubits, PRX Quantum **2**, 030345 (2021).

[77] L. P. Pryadko, V. A. Shabashov, and V. K. Kozin, QDistRnd: A GAP package for computing the distance of quantum error-correcting codes, J. Open Source Soft. **7**, 4120 (2022).

[78] Z. He, A. Cowtan, D. J. Williamson, *et al.*, Extractors: QLDPC architectures for efficient Pauli-based computation, (2025), arXiv:2503.10390.

[79] S. Koutsioumpas, H. Sayginel, M. Webster, *et al.*, Automorphism ensemble decoding of quantum LDPC codes, (2025), arXiv:2503.01738.

[80] O. Higgott and C. Gidney, Sparse Blossom: Correcting a Million Errors per Core Second with Minimum-Weight Matching, Quantum **9**, 1600 (2025).

[81] S. Koutsioumpas, T. Noszko, H. Sayginel, *et al.*, Colour codes reach surface code performance using vibe decoding, (2025), arXiv:2508.15743.

[82] T. Hillmann, L. Berent, A. O. Quintavalle, *et al.*, Localized statistics decoding for quantum low-density parity-check codes, Nature Comm. **16**, 8214 (2025).

[83] C. Gidney and C. Jones, New circuits and an open source decoder for the color code, (2023), arXiv:2312.08813.

Appendix A: Toric directional codes as bivariate-bicycle codes

Toric directional codes [35, 54] are obtained by tessellating a generalised torus with directional tiles. This yields a translationally invariant CSS code whose X-type stabilisers are translates of the directional word $\mathfrak{D}$, while the Z-type stabilisers are translates of the same string on the dual lattice. This translational invariance reveals their algebraic nature: toric directional codes form a distinguished subfamily of bivariate-bicycle codes [17, 65].

Algebraically, an X-type stabiliser is specified by two finite bivariate polynomials $f(x,y), g(x,y) \in \mathbb{F}_2[x,y]/(x^l + 1, y^m + 1)$, which encode its support on the horizontal and vertical edges, respectively, of an $l \times m$ grid with periodic boundary conditions. The corresponding Z-type stabiliser is supported on the dual string and is encoded, on horizontal and vertical edges respectively, by the reciprocal polynomials $\overline{g(x,y)}$ and $\overline{f(x,y)}$, where for $f(x,y) = \sum_{i,j} f_{i,j} x^i y^j$ we define $\overline{f(x,y)} = \sum_{i,j} f_{i,j} x^{-i} y^{-j} = \sum_{i,j} f_{i,j} x^{l-i} y^{m-j}$, with exponents understood modulo l and m. The CSS parity-check matrices can therefore be written as

$$H_X = \left[\mathbb{B}(f(x,y)) \mid \mathbb{B}(g(x,y))\right],$$
$$H_Z = \left[\mathbb{B}(\overline{g(x,y)}) \mid \mathbb{B}(\overline{f(x,y)})\right],$$

where $\mathbb{B}$ denotes the binary representation of the quotient ring, defined by $\mathbb{B}(x) = S_l \otimes I_m$ and $\mathbb{B}(y) = I_l \otimes S_m$, with S_l and I_l the cyclic-shift permutation matrix and the identity matrix of order l, respectively [17, 66]. A consequence of using reciprocal polynomials for the Z-type parity checks is that the corresponding Z-type stabilisers generally have the same geometric shape as the X-type stabilisers, up to a 180° rotation. Therefore, if the directional word traces an open, non-self-crossing string, the word must be palindromic for the X- and Z-type stabilisers to have identical geometry.

Lemma 2 ((w,w)-regularity)**.** Consider a toric directional code defined by a directional word $\mathfrak{D}$ of weight w. Then every stabiliser generator, corresponding to a row of H_X or H_Z, has weight w. Moreover, every data qubit participates in exactly w stabiliser checks in total. Equivalently, the Tanner graph of the code, including both X- and Z-type checks, is (w,w)-regular.

Proof. This follows directly from the bivariate-bicycle representation. Let $w_f = |\mathrm{supp}\ f|$ and $w_g = |\mathrm{supp}\ g|$, so that $w = w_f + w_g$. By construction, $\mathbb{B}(f)$ and $\mathbb{B}(\overline{f})$ are (w_f, w_f)-regular, while $\mathbb{B}(g)$ and $\mathbb{B}(\overline{g})$ are (w_g, w_g)-regular. Hence each row of H_X and H_Z has weight $w_f + w_g = w$. Similarly, for each data qubit, the total number of stabiliser checks it participates in is the sum of the corresponding column weights from H_X and H_Z, which is again $w_f + w_g = w$. □

Appendix B: Directional tile codes for biased noise

For a fixed directional word $\mathfrak{D}$, varying the aspect ratio of the rectangular base grid changes the balance between the X- and Z-type distances. Below, we list several directional tile codes with strongly unbalanced distances, making them potentially useful for biased-noise architectures [67–76].

$\mathfrak{D}$, w	M N	$[[n,k,d_X,d_Z]]$	n_r	$(kd_Xd_Z)/n$
NESEN, 5	22 3	$[[259,2,\leq 37,4]]$	58	1.14
$\mathrm{N}^2\mathrm{ESEN}^2$, 7	22 3	$[[269,4,\leq 28,5]]$	87	2.08
$\mathrm{N}^2\mathrm{E}^2\mathrm{SE}^2\mathrm{N}^2$, 9	19 4	$[[365,10,\leq 31,4]]$	196	3.40
$\mathrm{N}^2\mathrm{E}^2\mathrm{SESE}^2\mathrm{N}^2$, 11	12 13	$[[537,14,\leq 35,12]]$	216	10.94

TABLE III. **Parameters of representative directional tile codes with unbalanced distances.** Here n_r denotes the optimised number of routing qubits required for nearest-neighbour syndrome extraction. For each listed directional-word weight w, we perform a brute-force search over all allowed directional words $\mathfrak{D}$ and base-grid sizes $M \times N$ with $M + N \leq 25$. The table reports the instances with the largest unbalanced code-efficiency ratio kd_Xd_Z/n within this search space, compared with rectangular rotated surface-code patches. For distances above 20, the integer-programming distance computation does not terminate within a reasonable time; in these cases we use the probabilistic algorithm `QDistRnd` [77] to obtain distance upper bounds.

Appendix C: Routing overhead of directional tile codes

In this section, we bound the routing overhead of directional tile codes and describe the boundary optimisation used in the implemented circuits. The key point is that routing qubits are required only near the boundary of the planar layout. Consequently, for a fixed directional word, the routing overhead grows with the linear size of the patch rather than with its area.

Lemma 3 ($\sqrt{n}$-scaling of the routing overhead)**.** Consider a family of directional tile codes obtained from a fixed directional word $\mathfrak{D}$ with fixed tile size $B \times B$. Let the bulk patch have size $M \times N$, with bounded aspect ratio, and let n denote the number of data qubits. Then the number n_r of routing qubits required for nearest-neighbour syndrome extraction satisfies

$$n_r = O(\sqrt{n}).$$

Proof. Let $L_h = M + 2(B-1)$ and $L_v = N + 2(B-1)$. Since B is fixed, $L_h = M + O(1)$ and $L_v = N + O(1)$. The total number of hardware locations on vertices, edges and plaquettes is

$$\begin{aligned} n_{\mathrm{hw}} &= L_h L_v + (2L_h L_v - L_h - L_v) + (L_h - 1)(L_v - 1) \\ &= 4MN + O(M + N). \end{aligned}$$

The bulk contains MN stabiliser anchors of each type. Thus there are $2MN$ bulk check qubits, and since the corresponding stabilisers are linearly independent,

$$n \geq 2MN = \Theta(MN).$$

Hence, data and check qubits together occupy at least $4MN$ hardware locations. Therefore

$$n_r \leq n_{\mathrm{hw}} - 4MN = O(M + N).$$

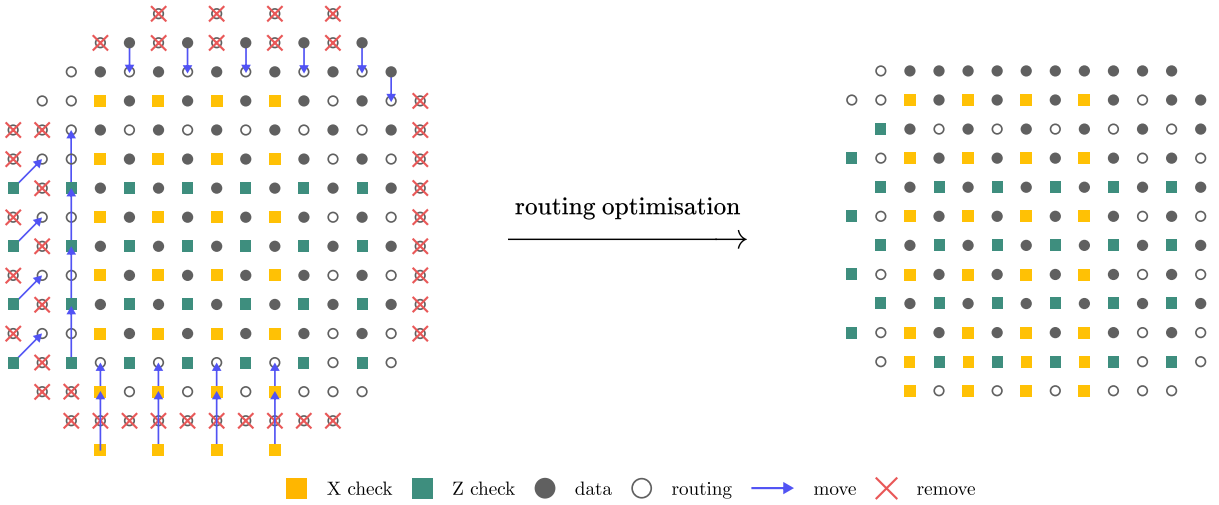


FIG. 7. **Routing optimisation.** Example of the boundary-routing optimisation used to reduce the number of auxiliary qubits in the implemented layout and circuit for the $\mathtt{N}^2\mathtt{ESEN}^2$ directional tile code with parameters $[[60, 4, 5]]$ as presented in Figure 1. The left panel shows the standard layout before optimisation: empty circles denote routing sites, red crosses mark routing sites that are removed, and blue arrows indicate shifts of data or check start positions. The right panel shows the optimised layout after pruning routing sites that are free during the syndrome-extraction walk. The optimisation preserves the measured stabiliser supports, detectors and logical observables, while reducing the total number of circuit qubits.

Under bounded aspect ratio, $M + N = O(\sqrt{MN})$, and hence

$$n_r = O(\sqrt{n}).$$

□

The boundary-scaling bound is naturally expressed in terms of distance. The logical operators of these codes have a boundary-to-boundary structure, so the X- and Z-type distances are controlled primarily by the two bulk dimensions M and N, respectively. This is reflected in the empirical fits in Table IV. The same table shows that both the standard routing overhead $\widetilde{n}_r$ and the optimised routing overhead n_r are affine functions of M and N. Thus, for a fixed directional word, the routing overhead scales linearly with the relevant boundary-to-boundary distance scale. In particular, for the balanced families used in our main overhead comparisons, this gives

$$n_r = O(d).$$

This behaviour is shown explicitly in Figure 8 for uniform-distance instances, $d = d_X = d_Z$, where both the standard and optimised routing overheads grow approximately linearly with d. The optimisation gives substantial constant-factor savings across all directional words.

We reduce the routing overhead by generating and testing simplified routing layouts. We call the primary simplification "route-window shortening". For each boundary stabiliser, we identify the first and last steps of the directional word at which it interacts with surviving data qubits. Any prefix or suffix outside this active window is not needed to measure the stabiliser support, so the check start position can be shifted and the corresponding terminal routing sites can be removed, provided that no check-start collision or data-check overlap is introduced. We then apply the same principle to terminal routing sites used only by a single data or check qubit q: if the site is not required between two data-check interactions involving q, it is absorbed into the trajectory of q by moving the appropriate start position to that site. For data qubits, this is implemented as a data-start shift onto a former routing coordinate.

After each proposed removal, we try to greedily remove more routing qubits, by iteratively deleting routing qubits near the edges of the layout and shifting the remaining qubits to fill the newly introduced hole. We then run the directional syndrome extraction normally for two rounds (forwards and backwards) and accept the move only if every stabiliser still interacts with exactly the data qubits specified by the original parity-check matrix. We call this method "trace pruning" as it relies on the trace of the directional word on the grid fixing future steps. The intuition for this is that even though in the original layout some routing qubits might appear necessary, through the displacement of early directional word steps, their neighbour routing qubits can fill the gap instead when near the middle of the schedule. Several candidate circuits are generated in this way, including variants with and without data-start shifts and with different combinations of greedy deletions. The final candidate is accepted only if the measured stabiliser supports, detectors, and logical observables are unchanged, and if the detector error model remains deterministic. Among all valid candidates, we choose the one with the smallest number of physical qubits. We show the unoptimised layout, together with the routing-site moves and deletions produced by the combined optimisation procedure, in Figure 7.

Example 4 (Routing optimisation of the $[[323, 14, 15]]$ directional tile code)**.** For the $[[323, 14, 15]]$ directional tile code generated by the weight-11 directional word $\mathtt{N}^2\mathtt{E}^2\mathtt{SESE}^2\mathtt{N}^2$, the unoptimised layout contains 346 routing sites. By combining the two optimisation procedures and repeating them with

$\mathfrak{D}$, w	$\widetilde{n_r}$	n_r	d_X	d_Z
$\mathtt{N^2ESEN^2}$, 7	$8M + 8N + 36$	$3M + 2N + 15$	$\simeq 1.10M - 0.08N + 2.25$	$\simeq 1.01N - 0.01M + 2.07$
$\mathtt{N^2E^2SE^2N^2}$, 9	$18M + 8N + 72$	$7M + 6N + 39$	$\simeq 1.51M - 0.23N + 2.54$	$\simeq 0.57N - 0.05M + 1.68$
$\mathtt{N^2E^2SESE^2N^2}$, 11	$24M + 8N + 98$	$9M + 4N + 56$	$\simeq 2.30M - 0.12N + 3.77$	$\simeq 0.87N - 0.03M + 0.74$

TABLE IV. **Routing overhead and distance scaling with bulk dimensions.** For each directional word $\mathfrak{D}$, the table gives the routing overhead before and after optimisation as functions of the bulk dimensions M and N. Here $\widetilde{n_r}$ denotes the routing overhead of the standard layout, while n_r denotes the optimised routing overhead after removing unnecessary routing qubits. The last two columns give empirical linear fits for the X- and Z-type distances, showing that d_X is primarily controlled by M and d_Z by N.

different update orders, the routing overhead can be reduced iteratively. Applying route-window shortening first reduces the number of routing sites from 346 to 235. Subsequent trace pruning of the remaining routing sites further reduces this number to 165. Thus, the combined optimisation removes 181 routing sites in total, corresponding to a reduction of approximately 52% relative to the unoptimised layout.

Appendix D: Logical gates from automorphisms and derived automorphisms

We briefly discuss possible mechanisms for implementing logical operations in the directional tile-code families considered in this work. Since these codes are constructed as tile codes, it is natural to use the structural description of logical operators developed in Refs. [36, 59]. In particular, Ref. [59] gives a canonical symplectic basis of logical Pauli operators for tile codes, with representatives supported near the lattice boundaries and generated by a finite cellular-automaton rule. We use this basis to describe the logical Pauli operators of our directional tile codes and to visualise representative logical operators on the physical qubit lattice; examples are shown in Figure 9.

A first route to logical operations is provided by the *derived-automorphism* [59]. In this approach, the tile-code patch is temporarily enlarged along one boundary, implementing a controlled deformation of the code. This deformation induces a non-trivial Clifford action on the encoded qubits. When expressed in the canonical logical basis of Ref. [59], the resulting action decomposes into products of logical CNOT gates, while remaining compatible with a low-overhead fault-tolerant implementation.

Some of the directional words studied here have an additional geometric symmetry: they are palindromic about their central step. For a directional tile code with directional word $\mathfrak{D} = \vec{d}_1\vec{d}_2\cdots\vec{d}_w$, the sequence satisfies

$$\vec{d}_i = \vec{d}_{w+1-i},$$

possibly after identifying directions under the corresponding reflection of the tile. This symmetry induces a reflection of the physical qubit layout,

$$\rho : q_i \mapsto q_{w+1-i},$$

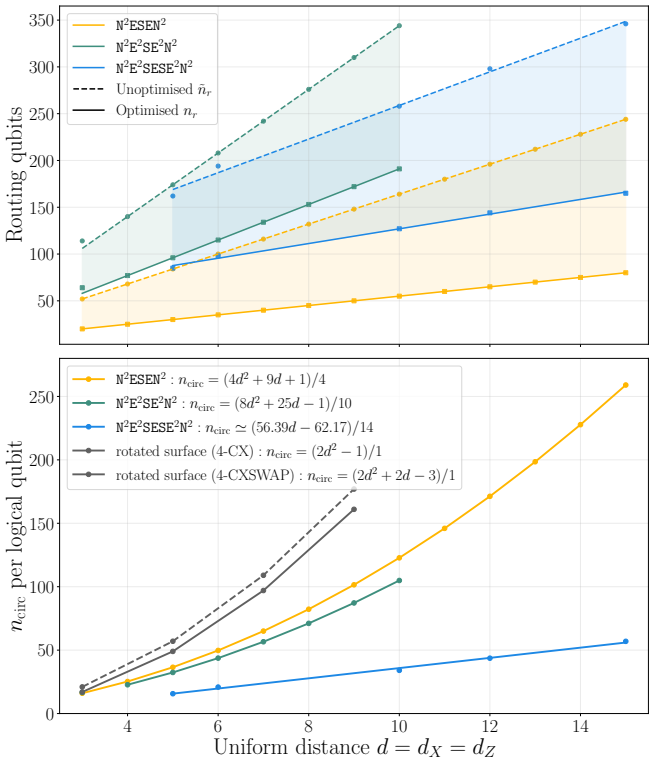


FIG. 8. **Routing and circuit-qubit overheads as a function of distance.** `Top:` Routing-qubit overhead for representative directional tile-code families with uniform distance d. Dashed curves show the standard-layout overhead $\widetilde{n_r}$, while solid curves show the optimised overhead n_r after boundary-routing simplification. Both scale approximately linearly with d, with the optimisation yielding substantial savings across all directional words. `Bottom:` Total circuit-qubit number $n_{\rm circ}$ for the corresponding optimised nearest-neighbour layouts. The $w = 7$ data follow the quadratic fit exactly over the shown range, and the $w = 9$ data apart from the $d = 3$ instance, whereas the $w = 11$ family is well described by approximately linear fits over the finite-distance regime considered.

on the tile-code lattice. Since the reflected directional word defines the same local stabiliser constraints, the corresponding physical qubit permutation preserves the stabiliser group and therefore defines a *code automorphism* [60].

This automorphism can in principle be realised as a permu-

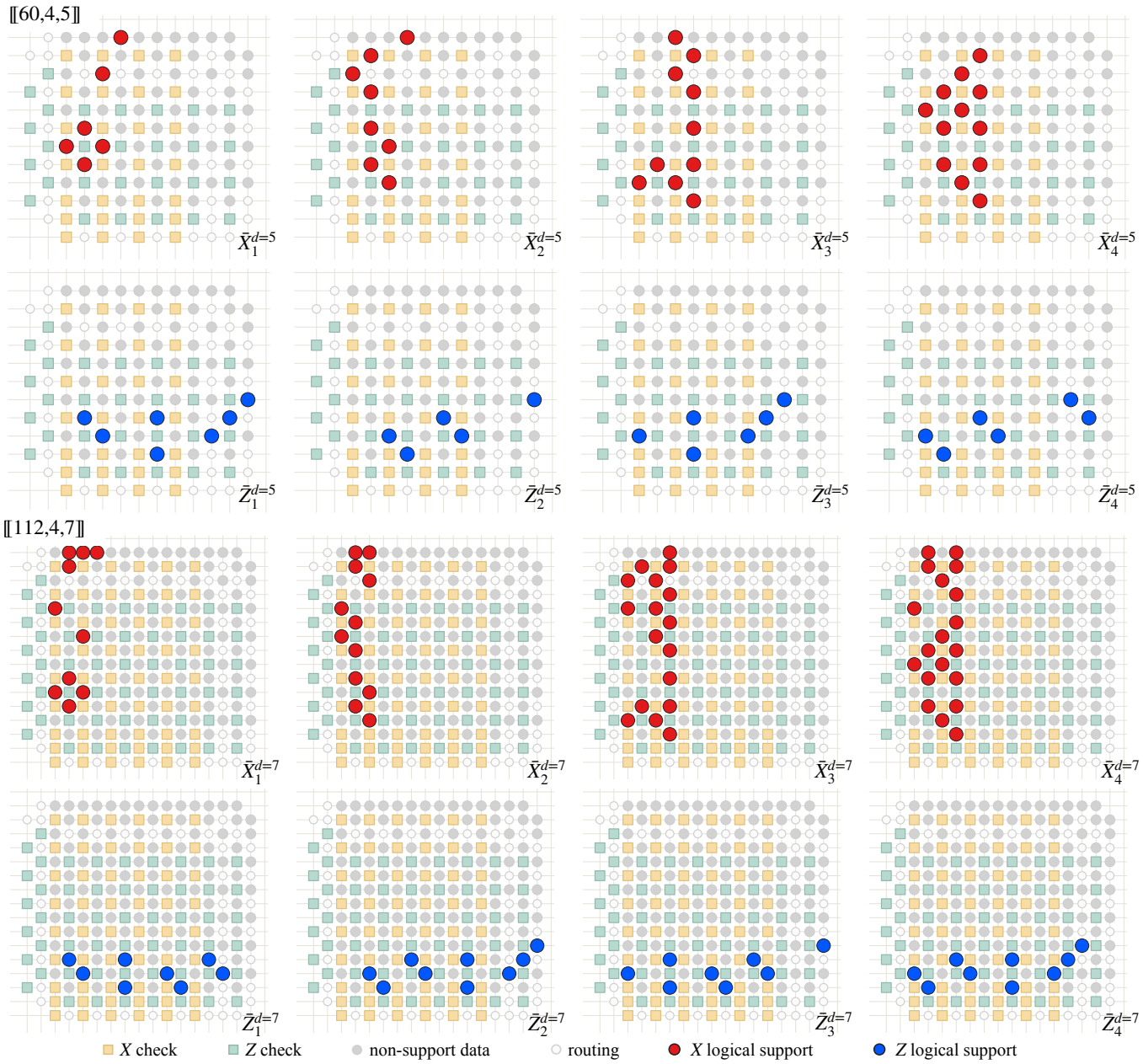


FIG. 9. **Canonical logical Pauli bases for the $\mathtt{N}^2\mathtt{ESEN}^2$ directional tile codes with stabiliser weight $w = 7$.** We show two representative codes from this directional-word family, with parameters $[[60, 4, 5]]$ and $[[112, 4, 7]]$. Supports of X-type logical operators are shown in red, while supports of Z-type logical operators are shown in blue. The displayed operators form canonical logical Pauli bases constructed using the method of Ref. [59], such that $\bar{X}_i$ and $\bar{Z}_j$ anti-commute if and only if $i = j$, and otherwise commute.

tation of physical qubits, or, equivalently, by relabelling physical qubits. Although the reflected qubits generally lie on opposite sides of the directional word, so the permutation is not geometrically nearest-neighbour in the original planar layout, we do not need to realise it as an actual SWAP circuit. Instead, each qubit is re-indexed by its reflected position, and the stabiliser generators, decoder information, and Pauli frame are updated accordingly. Using the canonical logical Pauli basis, we find that this reflection automorphism acts non-trivially on the encoded qubits. The induced logical Clifford operation decomposes into a product of logical SWAP and CNOT gates.

We complement these analytic observations with an automated search for further code automorphisms. Following the framework of Ref. [60], one can represent a stabiliser code as a binary linear code, compute generators of its automorphism group, impose the allowed Clifford constraints, and determine the induced logical action of each physical symmetry. Applying this approach to our code families reveals multiple automorphism generators and large symmetry groups, suggesting that the palindromic reflection is only one instance of a

$[[n,k,d]]$	$[[24,4,3]]$	$[[40,4,4]]$	$[[60,4,5]]$	$[[84,4,6]]$	$[[112,4,7]]$	$[[144,4,8]]$	$[[180,4,9]]$
$\lvert\mathrm{Aut}\rvert$	2^9	2^{13}	2^{17}	2^{21}	2^{24}	2^{23}	2^{33}

TABLE V. **Automorphism-group orders for selected $\mathtt{N^2ESEN^2}$ directional tile codes.** For each code instance $[[n,k,d]]$, the table gives the order of the physical-qubit permutation + single qubit Clifford gate group preserving the stabiliser code.

richer algebraic structure. These additional symmetries may provide useful ingredients for fault-tolerant Clifford computation [78] or symmetry-assisted decoding [79]. Examples of automorphism-group orders for the $\mathtt{N^2ESEN^2}$ directional tile code family with stabiliser weight $w = 7$ are listed in Table V.

Appendix E: Circuit-level memory simulations with only nearest-neighbour operations

Operation	Error channel	Probability
Two-qubit Cliffords	two-qubit depolarising	p
One-qubit Clifford	one-qubit depolarising	p
Reset in X (Z) basis	phase (bit) flip	p
Measure in X (Z) basis	phase (bit) flip	p
Idling qubits	one-qubit depolarising	p

TABLE VI. **The uniform noise model.** Operations, their associated error channels and probabilities for the uniform noise model.

For our numerical evaluations, we perform circuit-level memory simulations with the number of syndrome-extraction rounds set equal to the code distance d for each code instance. The circuits are compiled using CXSWAP gates and simulated under the uniform depolarising noise model specified in Table VI. In our main results, shown in Figure 2, we fix the physical noise parameter to $p = 0.001$.

All simulations are executed using `Stim` [49]. For each circuit, we sample up to 10 million shots, terminating early once 30 logical errors have been observed, except for the largest weight 11 hex code, which spans 745 qubits across 9 rounds and is simulated for up to 10 errors instead on each basis. For the surface code patches, we use correlated pymatching [80]. Decoding for the directional tile codes is performed with `VibeLSD` [81], which performs a parallel ensemble of serial min-sum schedules. Here we use an ensemble of size of 16 for 5 iterations, for the unconverged shots we extend the ensemble to 32 and subsequently to 64 and 128 paths progressively. For shots that still do not converge on any path, we fall back to `LSD` [82] order 0. Although the detector error models for directional tile-code circuits are denser than those of standard surface-code circuits, due to the longer directional walks and the presence of routing-qubit detectors, we find that this generic `BP`-based decoder already gives strong circuit-level performance. We have not attempted to optimise the decoder for these circuits, and further improvements may be possible with decoding methods tailored to the denser spacetime structure of directional tile codes.

From the simulated shots, we derive estimate $\widehat{p}_X = E_X/N_X$ and $\widehat{p}_Z = E_Z/N_Z$, where E_X, N_X and E_Z, N_Z are the logical errors and total simulated shots of the X and Z basis experiments respectively. We calculate the probability that at least one of the $2k$ total observables across both bases fails across the entire patch. We hence consider $\widehat{u} = 1-(1-\widehat{p}_X)(1-\widehat{p}_Z)$, and form $N_* = \min(N_X, N_Z)$, $E_* = \mathrm{round}(N_*\widehat{u})$, and $\widetilde{u} = E_*/N_*$, following the fusion rule [83]. The final estimate is

$$\widehat{s}_{XZ} = 1 - \left\{[1 + (2(1-\widetilde{u})^{1/(2k)} - 1)^{1/r}]/2\right\}^{2k}, \qquad \text{(E1)}$$

equivalent to Sinter's conversion with `pieces=r` and `values=2k` [49].

Code family	No. patches	$[[n,k,d]]$	$p_L(p=0.001)$
$\mathtt{N^2ESEN^2}$, 7 (square)	35	$[[24,4,3]]$	9.00×10^{-4}
		$[[40,4,4]]$	2.87×10^{-4}
		$[[60,4,5]]$	4.54×10^{-5}
		$[[84,4,6]]$	1.36×10^{-5}
$\mathtt{N^2E^2SE^2N^2}$, 9 (square)	14	$[[52,10,3]]$	5.30×10^{-4}
		$[[76,10,4]]$	1.65×10^{-4}
		$[[115,10,5]]$	2.17×10^{-5}
		$[[162,10,6]]$	9.10×10^{-6}
$\mathtt{N^2E^2SESE^2N^2}$, 11 (square)	10	$[[74,14,5]]$	2.89×10^{-4}
		$[[104,14,6]]$	2.98×10^{-5}
		$[[182,14,10]]$	6.98×10^{-6}
		$[[241,14,12]]$	1.37×10^{-6}
$\mathtt{ENES^2WS^2ENE}$, 11 (hexagon)	14	$[[52,10,3]]$	2.71×10^{-4}
		$[[74,10,4]]$	5.45×10^{-5}
		$[[104,10,5]]$	1.56×10^{-5}
		$[[199,10,7]]$	2.30×10^{-6}
		$[[295,10,9]]$	5.07×10^{-7}
Rotated surface codes (4-CX, square)	140	$[[9,1,3]]$	1.24×10^{-3}
		$[[25,1,5]]$	8.72×10^{-5}
		$[[49,1,7]]$	5.44×10^{-6}
		$[[81,1,9]]$	4.56×10^{-7}
Rotated surface codes (4-CXSWAP, square)	140	$[[9,1,3]]$	1.09×10^{-3}
		$[[25,1,5]]$	9.33×10^{-5}
		$[[49,1,7]]$	6.07×10^{-6}
		$[[81,1,9]]$	4.56×10^{-7}

TABLE VII. **All codes included in the circuit-level simulations of Figure 2.** Several patches of each code are used to build a quantum memory of 140 logical qubits, which is the least common multiple of the logical dimensions of the codes listed, in order to make a fair comparison. The grid connectivity is indicated below each directional word, and both rotated surface-code implementations use the square grid. Here $p_L(p=0.001)$ denotes the circuit-level logical error rate averaged over X and Z bases per-logical per-round at physical error rate $p = 0.001$.

Supplementary Files

This is a list of supplementary files associated with this preprint. Click to download.

- NESENanimation.mp4