Concept example diagrams for QEC framework GUI

Drew Vandeth

IBM **Quantum**

Code Definition

IBM **Quantum**

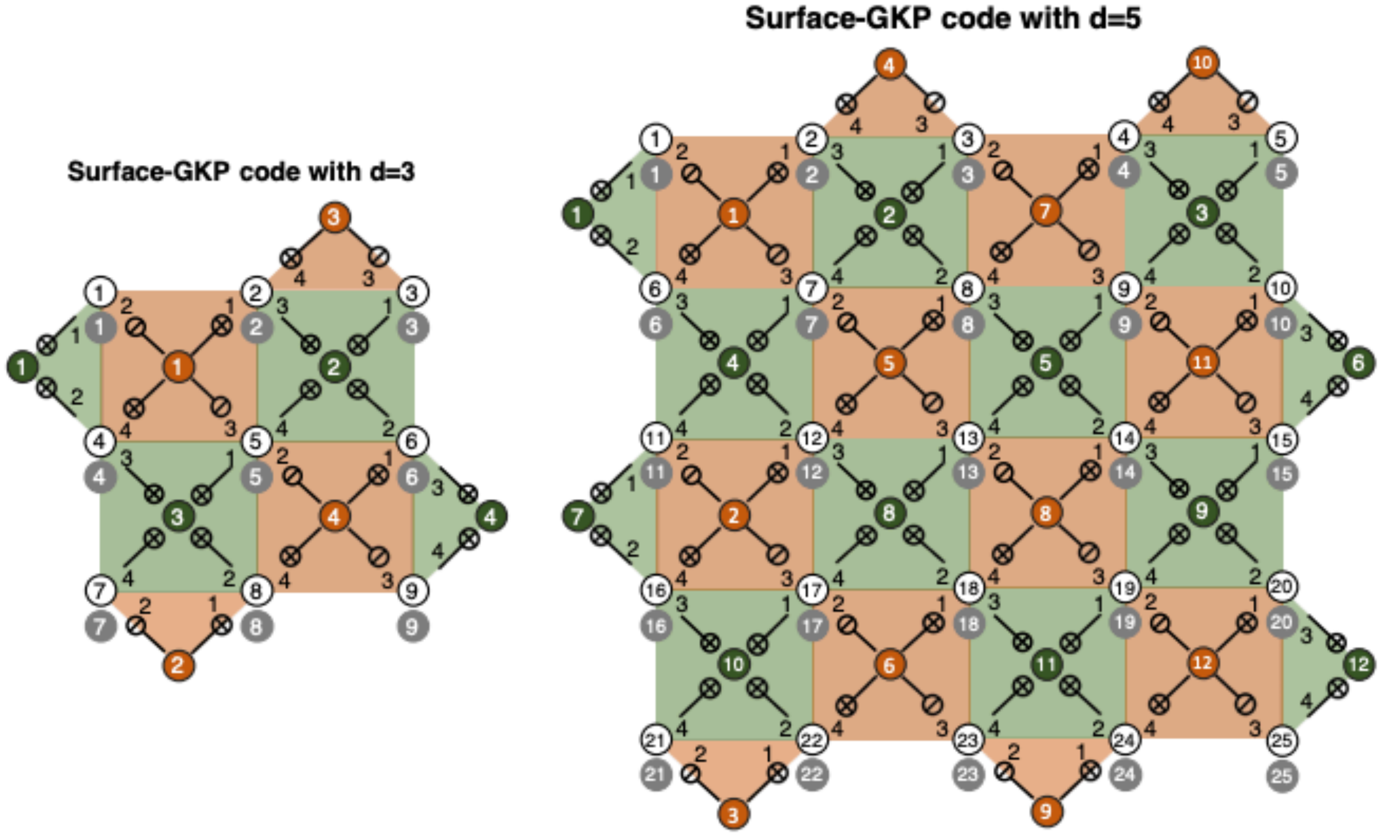


FIG. 3. The surface-GKP codes with $d = 3$ and $d = 5$. White circles represent the data GKP qubits and grey circles represent the ancilla GKP qubits that are used to measure GKP stabilizers of each data GKP qubit. Green and orange circles represent the syndrome GKP qubits that are used to measure the Z-type and X-type surface code stabilizers of the data GKP qubits, respectively. In general, there are d^2 data GKP qubits and $(d^2 - 1)/2$ Z-type and X-type syndrome GKP qubits. See also Fig. 5 for the reason behind our choice of inverse-SUM gates in the X-type stabilizer measurements.

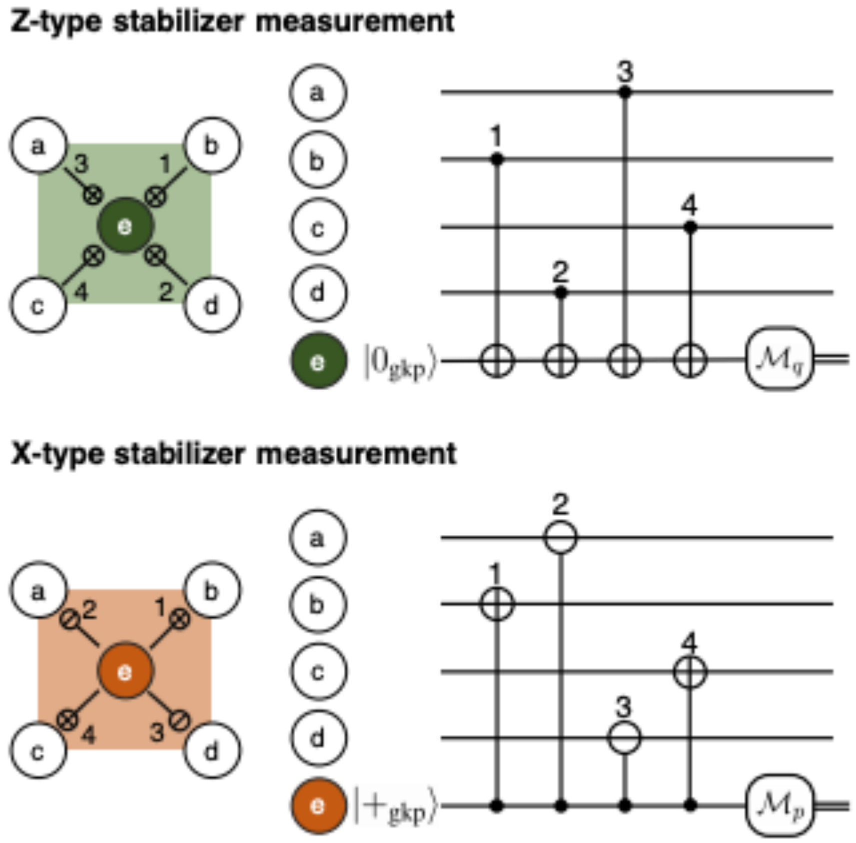


FIG. 4. Circuits for surface code stabilizer measurements.

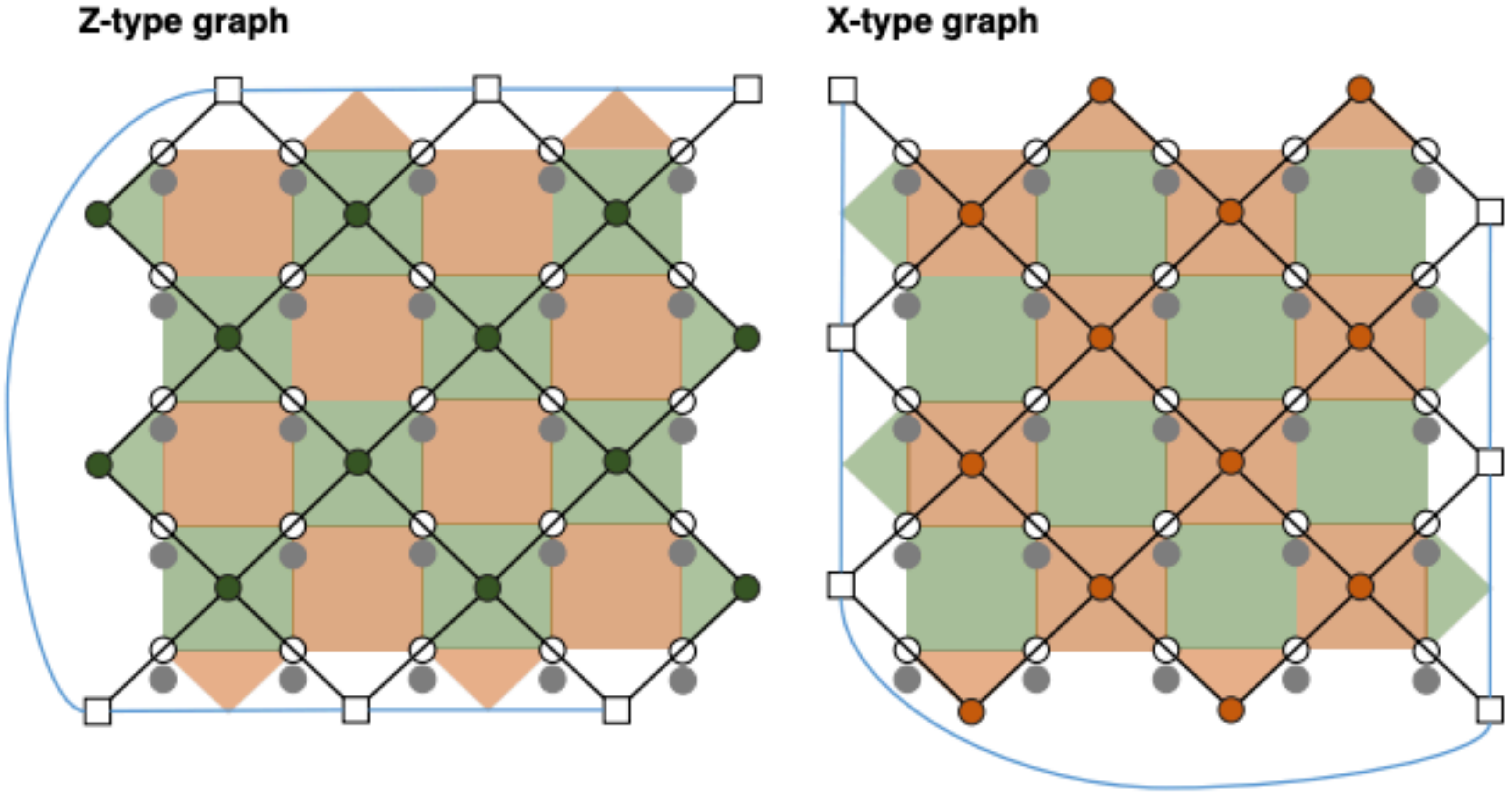


FIG. 10. Z-type and X-type 2D space graphs for the surface-GKP code with $d = 5$. These 2D graphs will be stacked up to construct Z-type and X-type 3D space-time graphs.

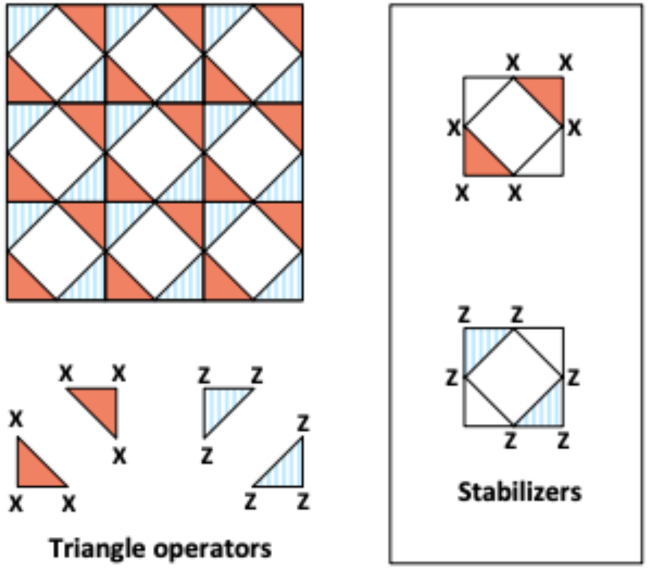


FIG. 1: Subsystem toric code. Qubits live at vertices and centers of edges of the regular square lattice. Opposite sides of the lattice are identified. *Left:* Four types of triangles and the corresponding triangle operators $G(T)$. Triangle operators that belong to different plaquettes pairwise commute. *Right:* Stabilizer operators S_p^X (top) and S_p^Z (bottom). Stabilizers are analogues of the plaquette and star operators of the standard toric code. Triangle operators commute with stabilizers. Eigenvalue of any stabilizer can be determined by measuring eigenvalues of individual triangle operators. For a lattice of size $L \times L$ the code has parameters $[[3L^2, 2, L]]$. This should be compared with the standard surface code which has parameters $[[2L^2, 2, L]]$.

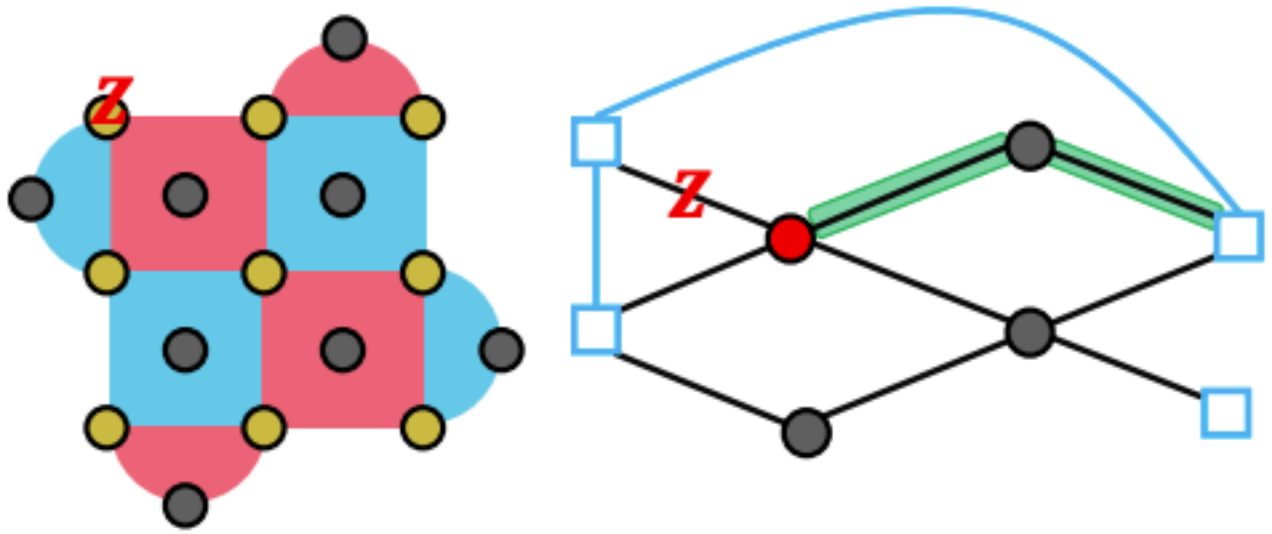


FIG. 14. On the left, we illustrate an example of a single failure resulting in a Z data qubit error (shown in red) for a $d_x = d_z = 3$ surface code. On the right, we show the decoding graph for Z-type errors, the resulting highlighted vertex and the actual path chosen during MWPM is highlighted in green. Such a path is chosen due to the lower edge weights (along the green path) computed from the conditional probabilities of the GKP error correction protocol. Such events are very rare and have a negligible effect on the total logical failure rates, especially for code distances $d \geq 5$.

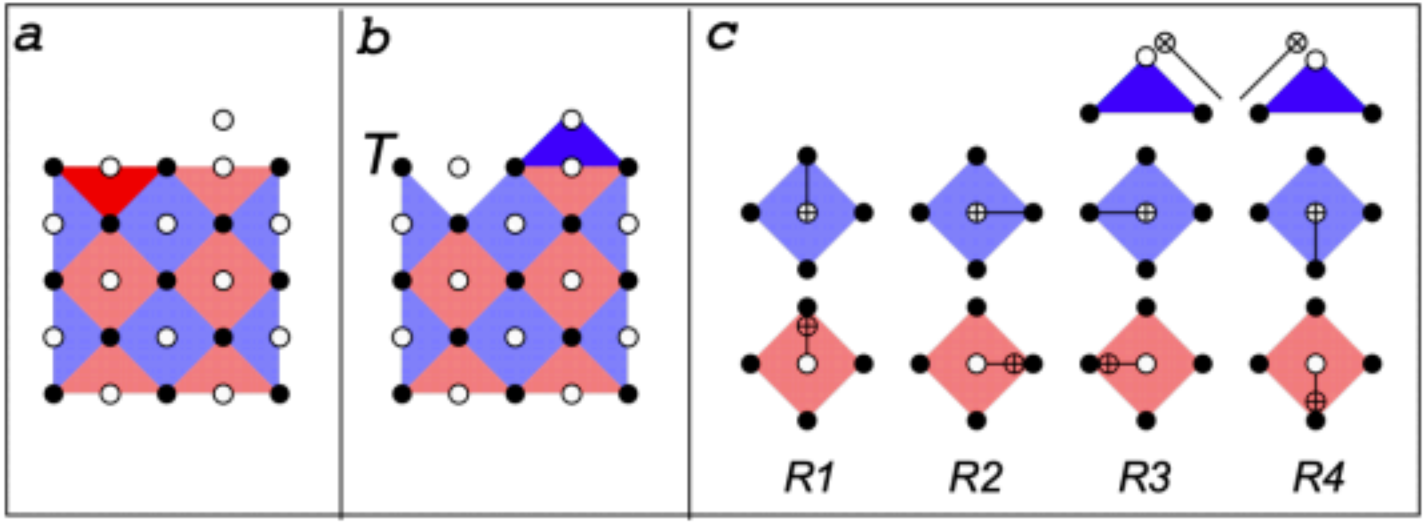


Figure 2: **Logical T-gate by code switching.** (a) Surface code $\mathcal{S}_1$ with the distance $d = 3$. Black and white circles indicate data and syndrome qubits. Red and blue faces indicate X and Z stabilizers. (b) Asymmetric surface code $\mathcal{S}_2$ exhibiting a single-qubit logical T-gate at the north-west corner of the lattice. Code switching $\mathcal{S}_1 \to \mathcal{S}_2$ is performed by turning off X stabilizer F_1 (dark red triangle) and turning on Z stabilizer G_1 (dark blue triangle). (c) The syndrome extraction cycle consists of four rounds of CNOTs R1, R2, R3, R4. To avoid clutter we only show a local schedule of CNOTs for each stabilizer. Schedules for weight-3 stabilizers are properly truncated. The schedule extends to the full lattice in a translation invariant fashion.

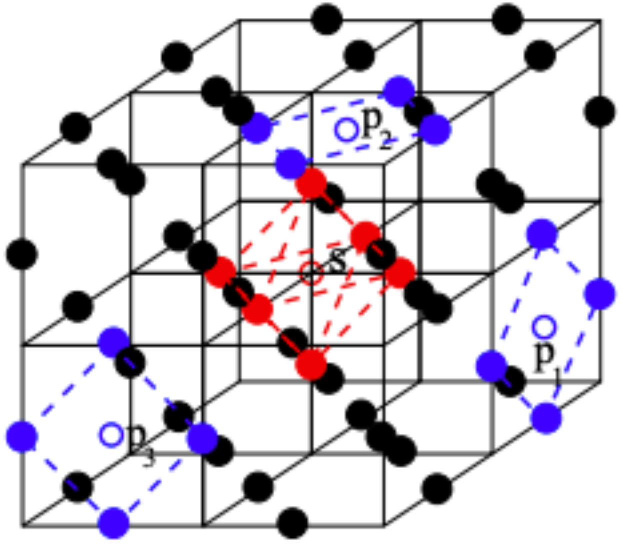


FIG. 1: (Color online) – Illustration of the Kitaev model in 3D, with explicit examples of a star operator $A_s = \prod \sigma_i^x$ at the lattice site s, and of three plaquette operators $B_p = \prod \sigma_i^z$ at the plaquette-dual sites p_1, p_2 and p_3. The σ spin index i labels respectively the 6 (red) spins around s and the 4 (blue) spins around p (connected by dashed lines).

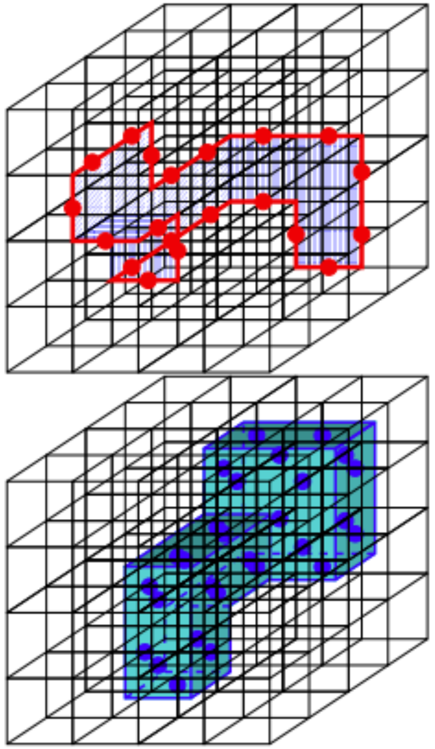
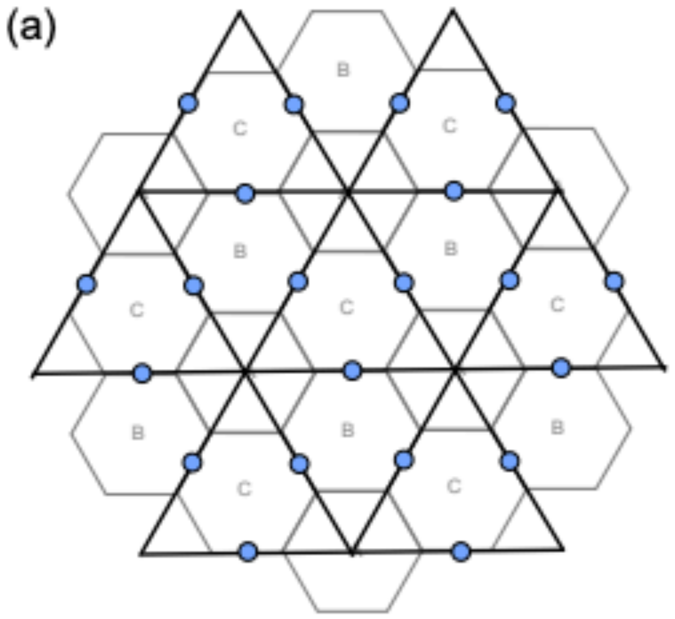


FIG. 3: (Color online) – Two examples of the underlying structures of the 3D Kitaev model: the closed σ^z *loops* along the edges of the cubic lattice, which satisfy $\prod_{\text{loop}} \sigma_i^z = 1$, and the closed σ^x *membranes* in the body-centered dual lattice, satisfying $\prod_{\text{membrane}} \sigma_i^x = 1$.



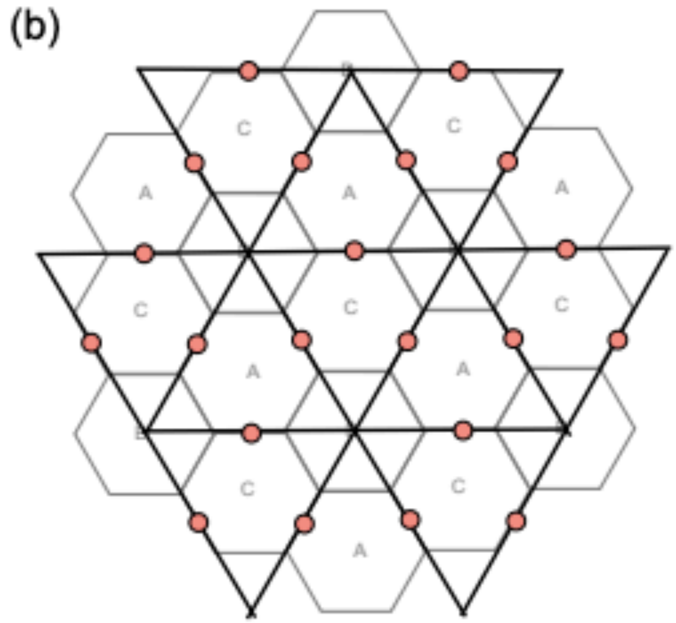


FIG. 5. (Color online) Fragments of the shrunk lattices: (a) $\mathcal{L}_A$ and (b) $\mathcal{L}_B$, obtained from $\mathcal{L}$ by shrinking A and B faces, respectively. Qubits are placed on edges, and the stabilizer generators are X-vertex and Z-face operators.

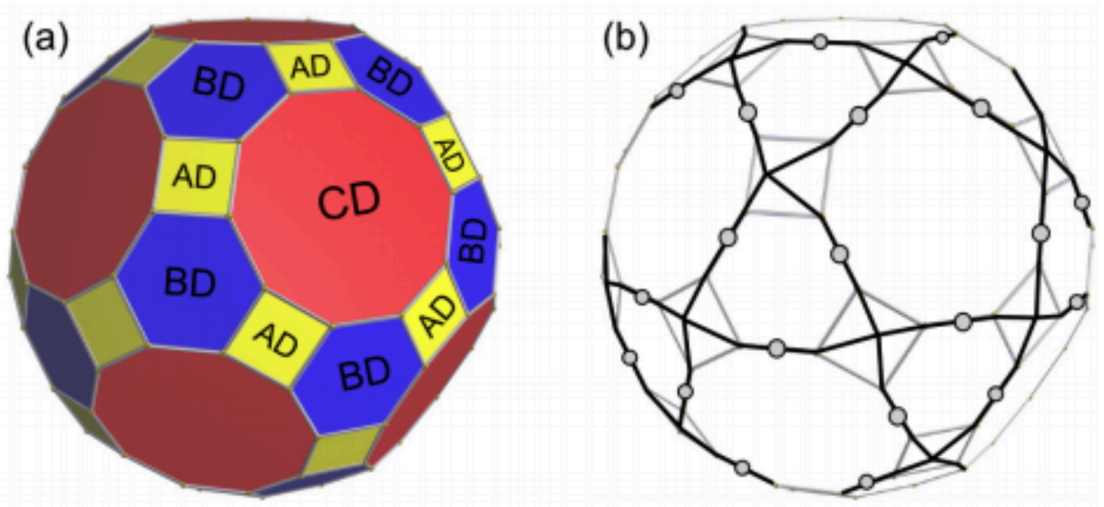


FIG. 7. (Color online) (a) The boundary ∂c of a volume c of color D in the lattice $\mathcal{L}$. Note that ∂c can be viewed as a 3-colorable and 3-valent lattice on a closed manifold (a sphere), with faces colored in AD, BD and CD. (b) A volume in the shrunk lattice $\mathcal{L}_A$ derived from c after shrinking volumes of color A. Note that qubits are placed on (a) vertices and (b) edges. The figures were created using Robert Webb's Stella software (http://www.software3d.com/Stella.php).

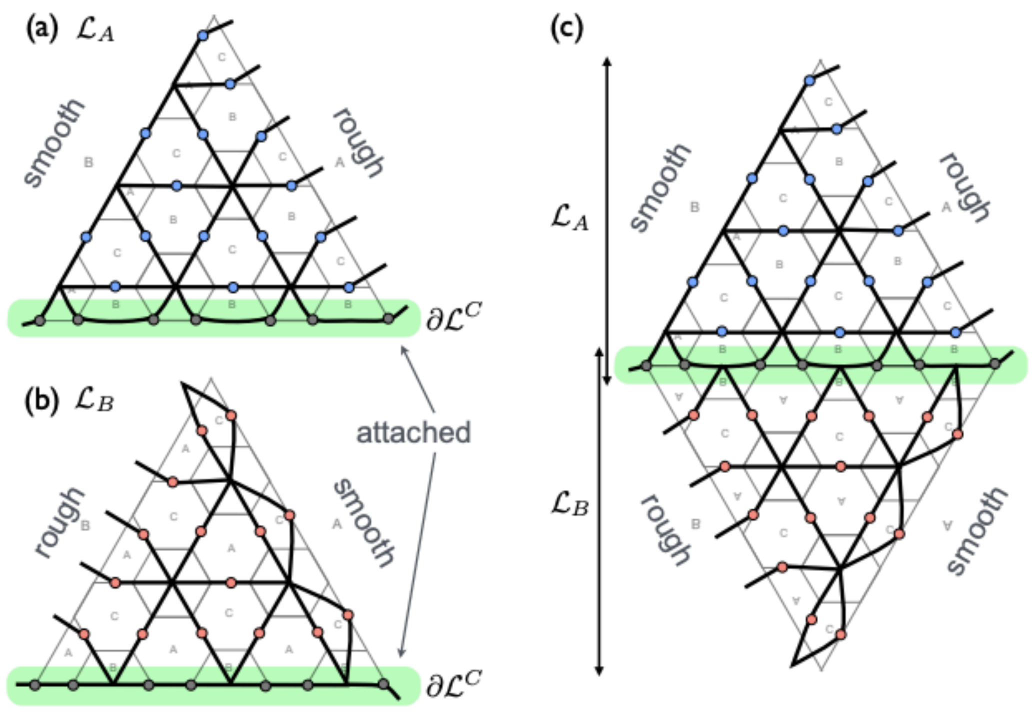
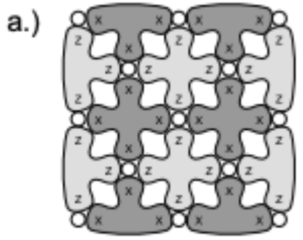


FIG. 10. (Color online) Attaching two lattices: (a) $\mathcal{L}_A$ and (b) $\mathcal{L}_B$ by identifying qubits along the boundary $\partial\mathcal{L}^C$. Note that both $\mathcal{L}_A$ and $\mathcal{L}_B$ have two qubits per edge on the boundary $\partial\mathcal{L}^C$. (c) Unfolded toric code $TC(\mathcal{L}_A\#\mathcal{L}_B)$. Blue qubits belong to the lattice $\mathcal{L}_A$, whereas red qubits belong to the (flipped) lattice $\mathcal{L}_B$.



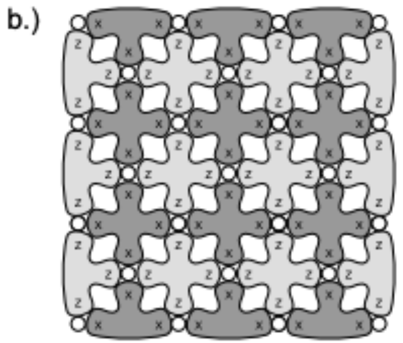


FIG. 1. a.) Distance 3 surface code. b.) Distance 4 surface code. Circles represent qubits. Bubbles represent operators (tensor product of Pauli X or Z operators) that are measured to detect errors. Note that all operators commute. Each bubble is associated with its own (syndrome) qubit, used to measure its operator (stabilizer [13]) via the circuits shown in Fig. 2. Each surface code contains a single logical qubit. A distance d surface code, properly implemented, can correct any combination of $\lfloor (d-1)/2 \rfloor$ errors.

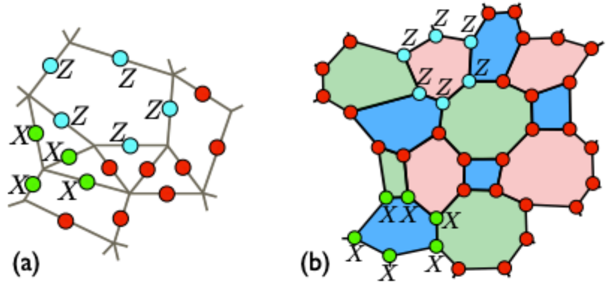


FIG. 2. (Color online) The toric code and the color code in two dimensions. (a) The toric code has qubits (red dots) placed on edges, and X-vertex (green) and Z-face (blue) stabilizer generators. (b) The color code has qubits placed on vertices, and X-face and Z-face stabilizer generators. Note that the color code can only be defined on a 3-valent and 3-colorable lattice.

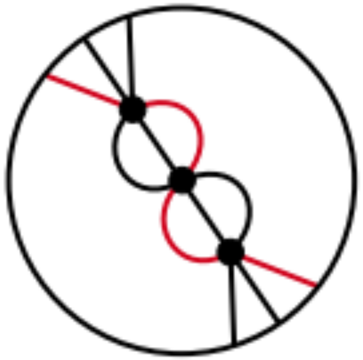


FIG. 3. The Shor code [6] as a tessellation of the real projective plane, see [9]. Antipodal points on the circle are identified. The code has three X-checks, nine physical qubits and seven Z-checks represented by vertices, edges and faces, respectively. The red line represents a logical Z-operator.

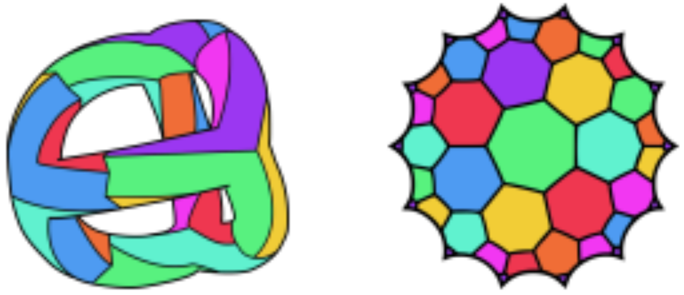
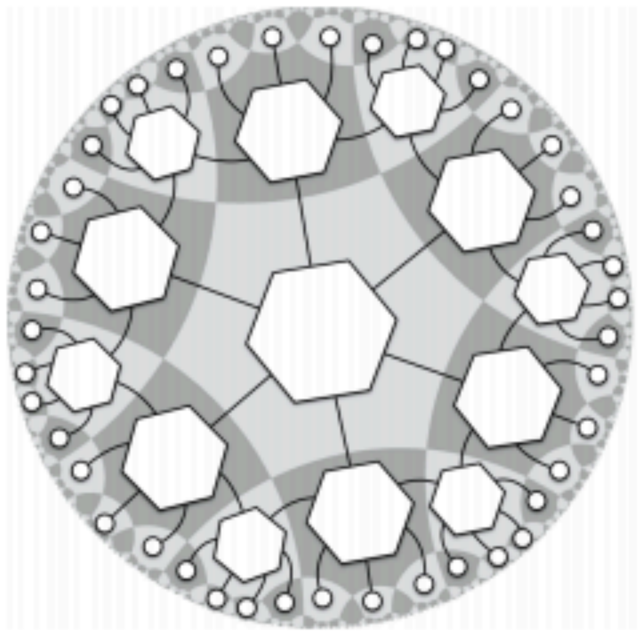


FIG. 4. A hyperbolic surface of genus 3 tessellated by heptagons. It gives rise to a code with parameters $n = 84$, $k = 6$, $d_X = 4$, $d_Z = 8$. The colors have no intrinsic meaning and are only included to guide the eye. A weight-four X-operator goes through the following four faces on the right: magenta (top), violet, green (middle), yellow (below) and back to the same magenta face (periodic boundary). A weight-eight Z-operator runs along the left-hand side of these faces.


(a) Holographic hexagon state

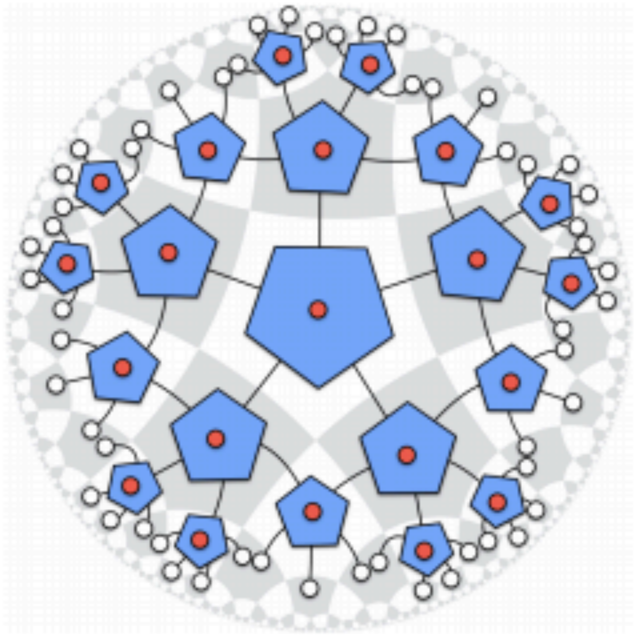

(b) Holographic pentagon code

Figure 4. White dots represent physical legs on the boundary. Red dots represent logical input legs associated to each perfect tensor.

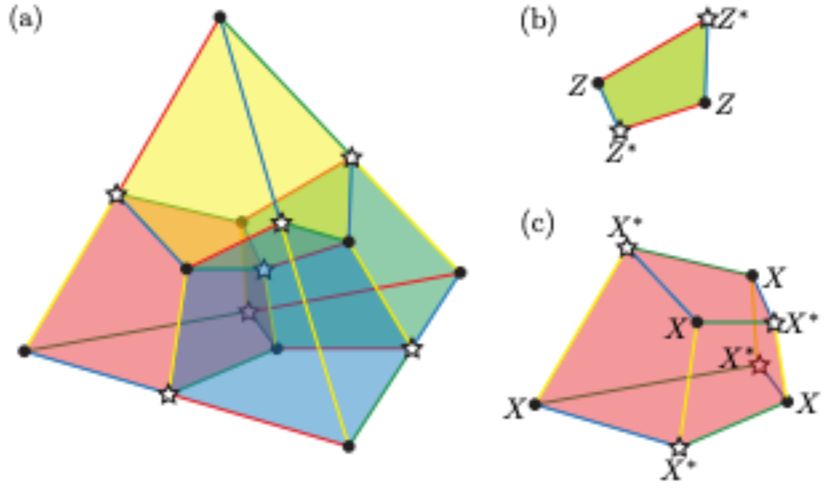


FIG. 2: (Color online) (a) The smallest instance of the qudit 3D color code with the X-type stabilizers colored red, green, blue and yellow. The 1-cells (edges) of the code are also colored. The vertices are also colored so the set $v_\bullet$ is represented by black circles and the set $v_\star$ is represented by white stars. (b) A single Z stabilizer of the tetrahedral 3D color code. The plaquette can take the color yellow if considered as a face of the green 3-cell, and vice versa. (c) A single X stabilizer of the tetrahedral 3D color code.

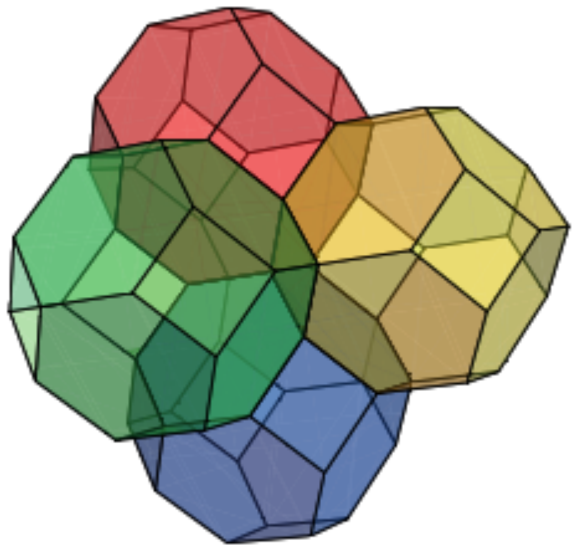


FIG. 3: (Color online) To increase the distance of the code, we tile 3D space with truncated octahedron cells (with appropriate boundary conditions). Here we show a portion of the lattice. A red cell fits into the hollow formed by the green, blue and yellow cells towards the bottom of the figure, and thus the tiling proceeds, ensuring the 4-colorability of the lattice is preserved.

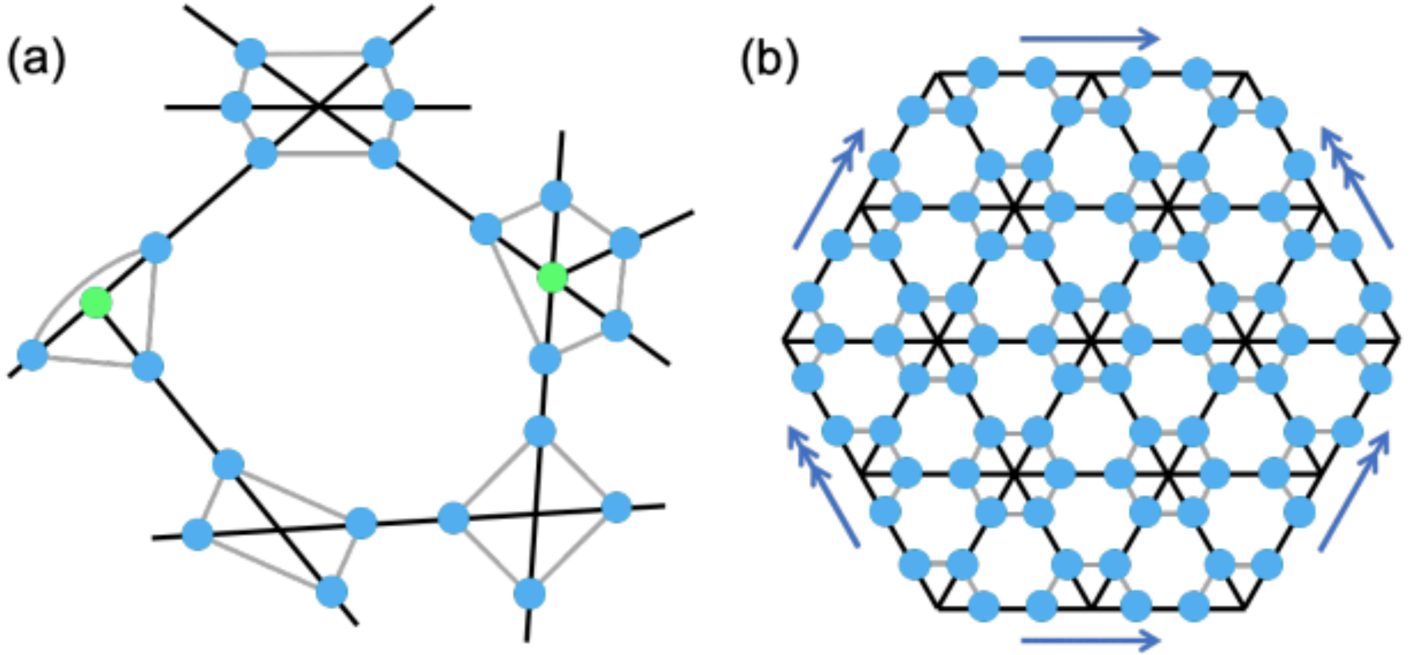


Figure 2: (a) An example section of a Majorana surface code. Majorana operators are placed on each half-edge (blue circles) and at each odd-degree vertex (green circles). The stabilizer associated to an even-degree vertex is the product of Majoranas around that vertex. The stabilizer associated to an odd-degree vertex is the product of Majoranas around that vertex and the Majorana located at that vertex. Gray lines cordon off the Majoronas involved in these stabilizers. The stabilizer associated to the pentagonal face is the product of the ten blue Majoranas on edges around that face. (b) A Majorana surface code of reference [34]. In our framework, it arises from a graph triangulating the torus. Here opposite sides of the hexagon are identified (directionally, as indicated by arrows) along with Majoranas on those edges. A total of 72 Majoranas survive this identification. All stabilizers associated to both vertices and faces are the product of six Majoranas.

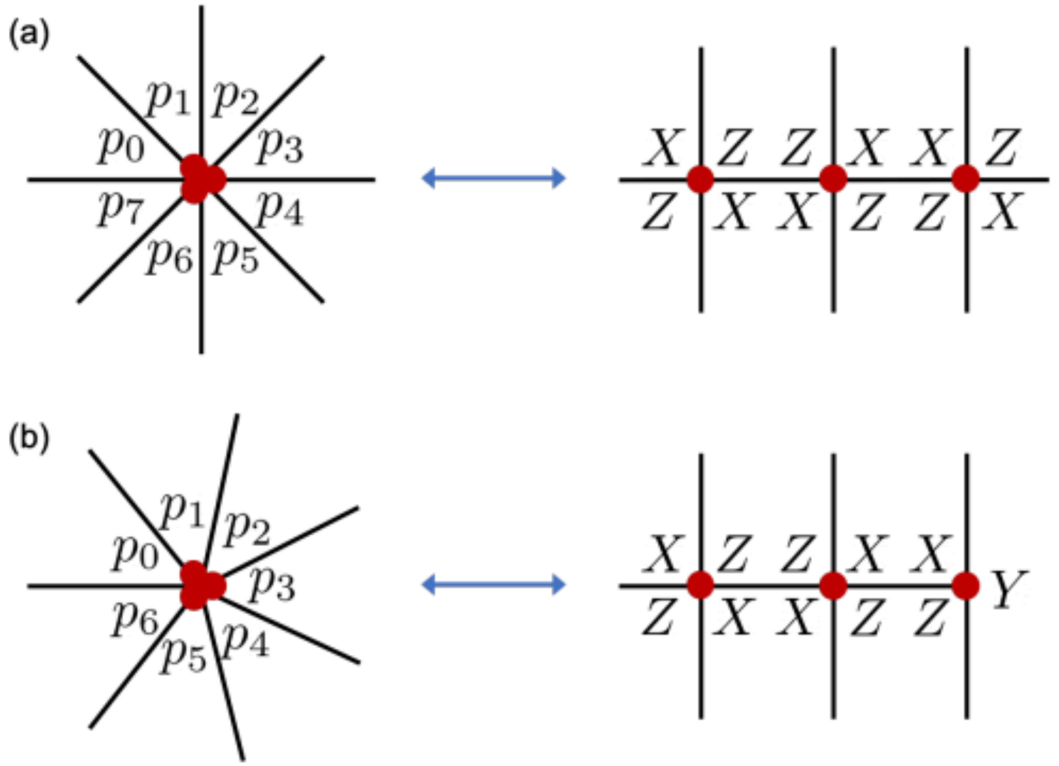


Figure 3: One way to create a CAL of length ℓ is to compose CALs of lengths three or four together. In (a), we compose three CALs with length four (right) to make a CAL with length eight (left) acting on three qubits (red circles). In (b), we use a similar composition of two length-four CALs and one length-three CAL to make a length-seven CAL. In terms of surface codes defined on graphs, this composition operation means that we can always decompose higher degree vertices into vertices with degrees just three or four. Still, we generally will include higher degree vertices in our drawings to highlight symmetries when possible.

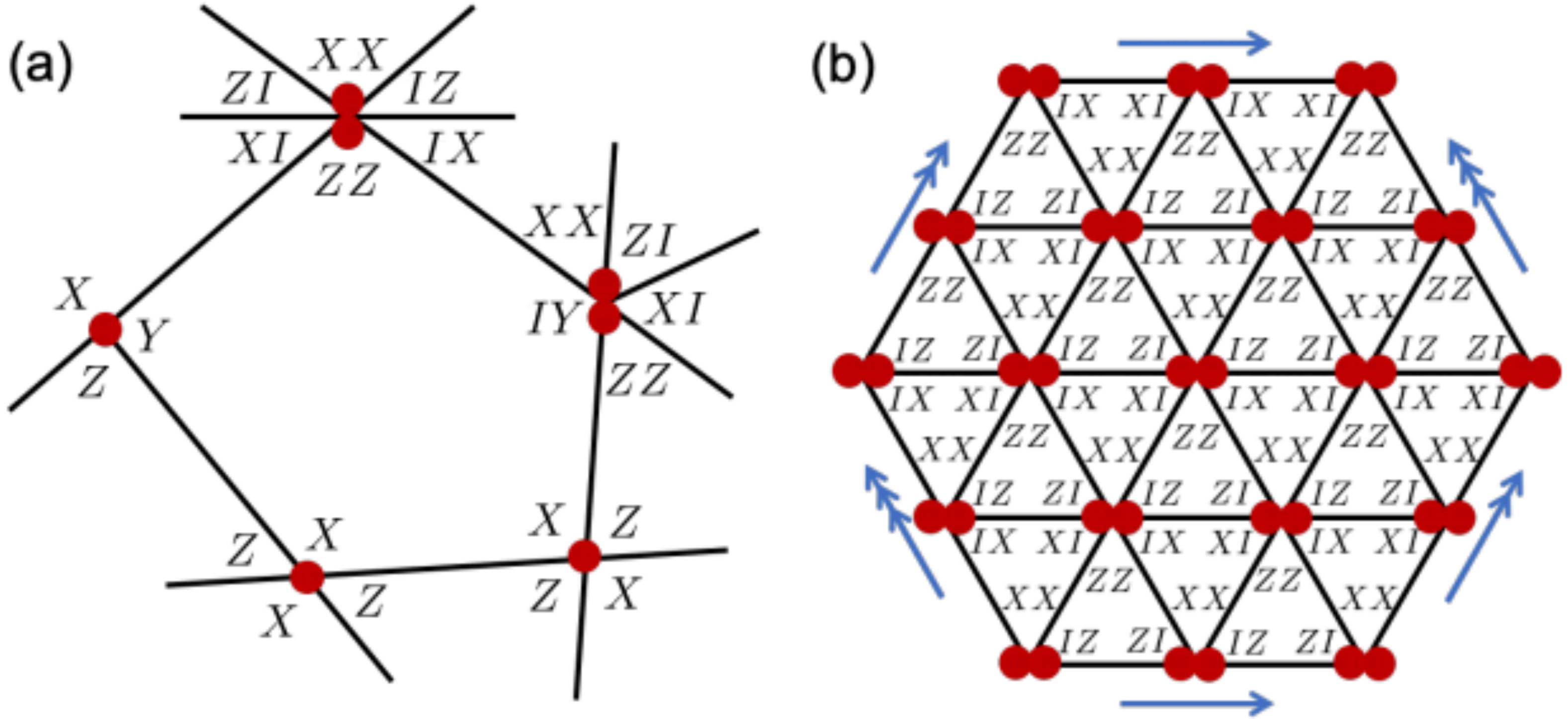


Figure 4: Qubit versions of the Majorana codes in Fig. 2. Place a single qubit on vertices with degrees three or four and a pair of qubits on vertices with degrees five or six. Around each vertex write a cyclically anticommuting list of Paulis (acting on qubits at that vertex). Each face represents a stabilizer defined as the product of all Paulis written within that face. Note that (b) depicts a toric code with $X^{\otimes 4}$ and $Z^{\otimes 4}$ stabilizers but on a lattice different from Kitaev's square lattice [1]. In this case, the code is $[\![24, 2, 4]\!]$. The number of encoded qubits is clear by Corollary 3.8 as this graph embedding is checkerboardable.

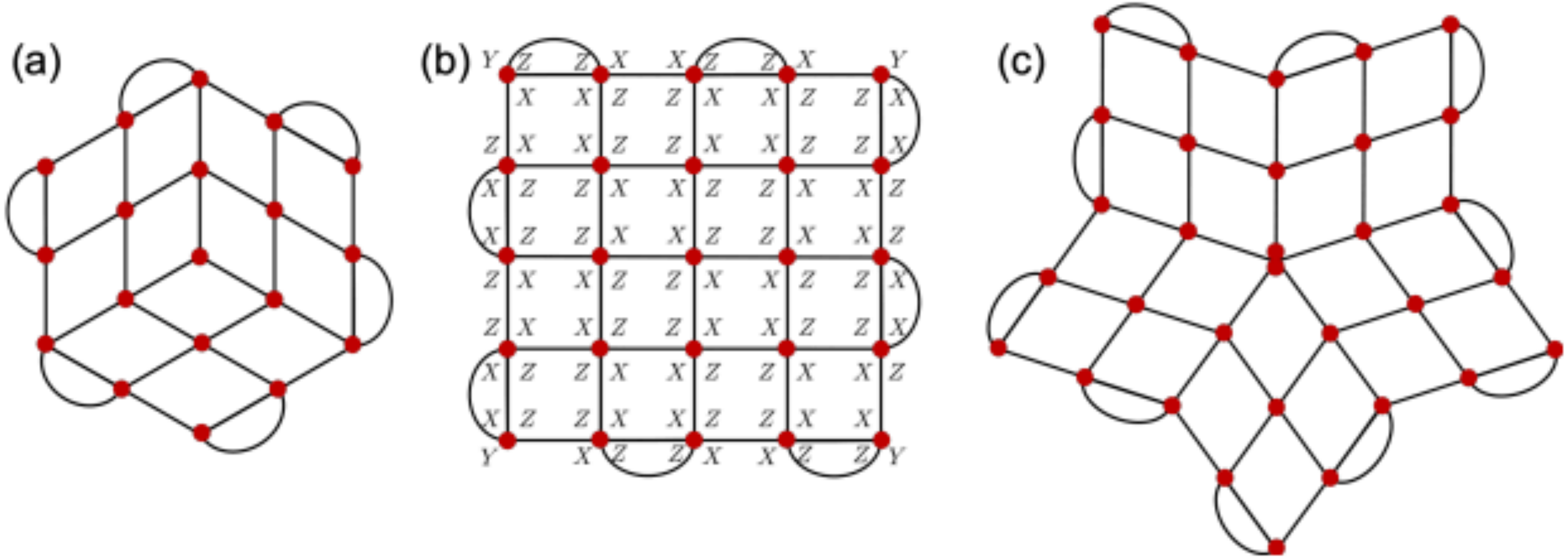


Figure 5: (a) The triangular surface code [6], (b) the rotated surface code [3] and (c) a stellated surface code [8] with even greater symmetry. Each code fits in our framework – in this case, by Definition 3.3 applied to these planar graphs. In (b), we show explicitly the assignment of CALs that gives the familiar rotated surface code. Notice that the outer face also gets assigned a stabilizer. Because of Lemma 3.2(e), the product of stabilizers on all non-outer faces is proportional to the stabilizer on the outer face.

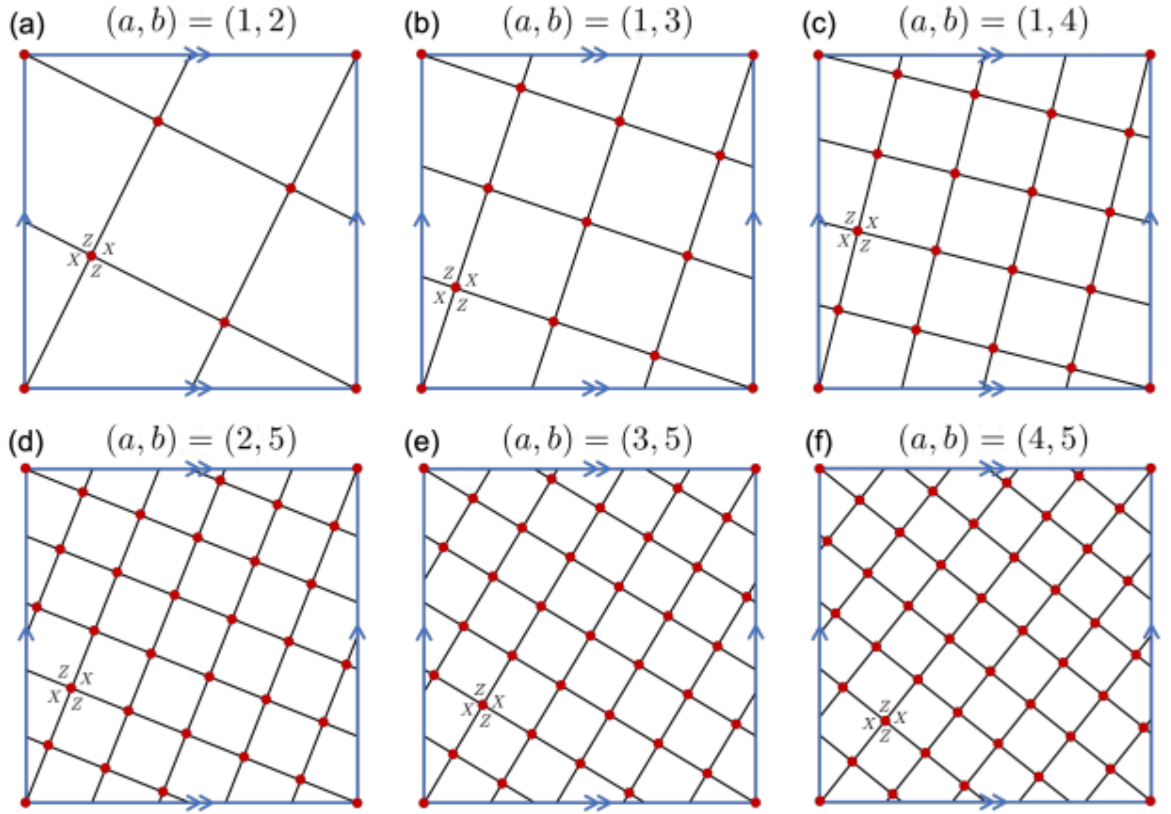


Figure 6: (a) The smallest error-correcting quantum code, the $[\![5,1,3]\!]$ code, is a surface code defined by an embedding of the 5-vertex complete graph K_5 in the torus. There are different embeddings of K_5 in the torus, but no other gives a 5-qubit code with distance more than two. Parts (b-f) show members of the cyclic toric code family with more qubits. See Theorem 3.9 and the paragraph above it for the code definition and parameters. A demonstrative CAL is written around one vertex in each graph. The codes are cyclic when this same CAL is orientated similarly around each vertex.

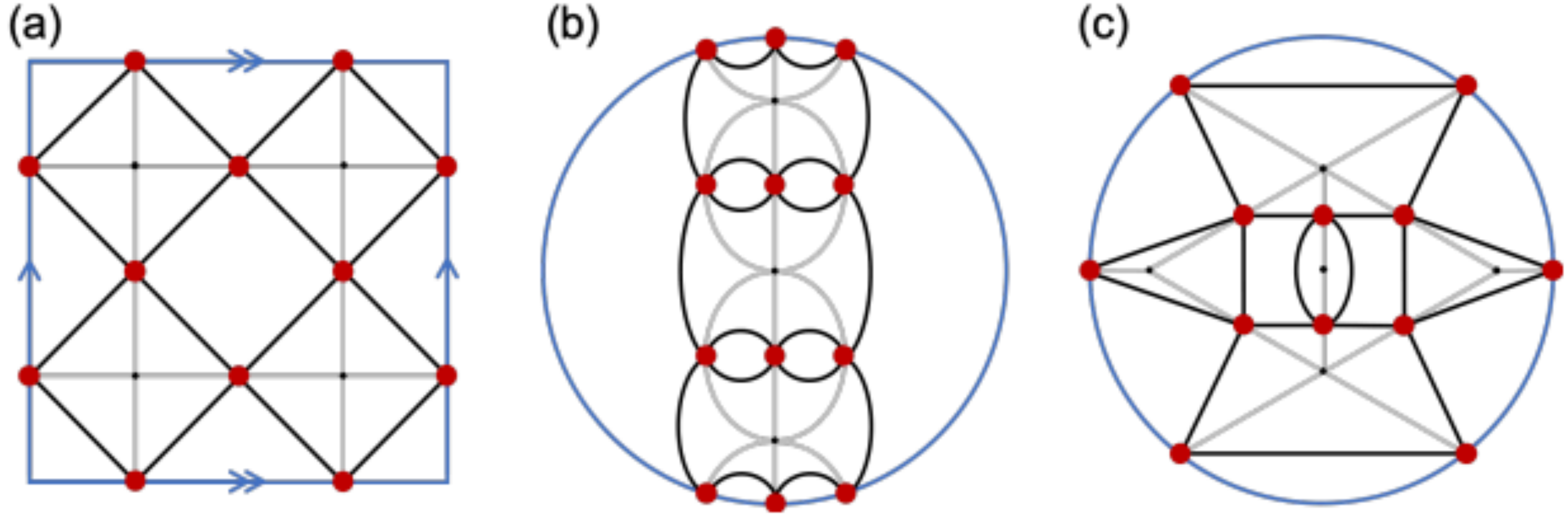
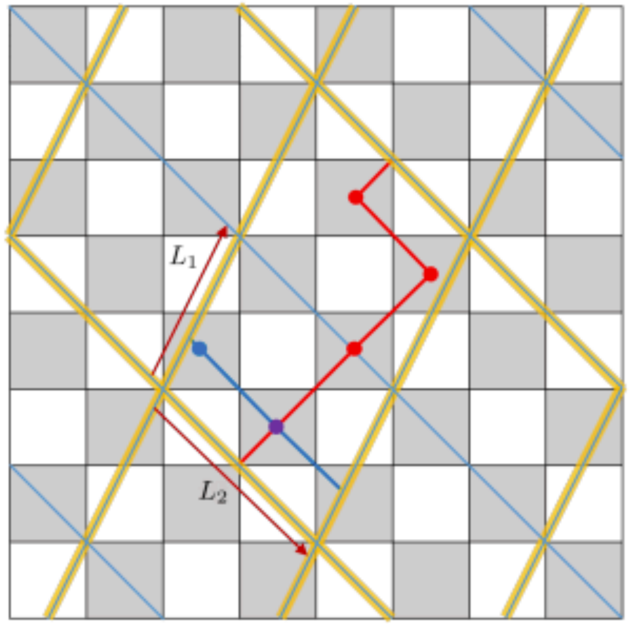
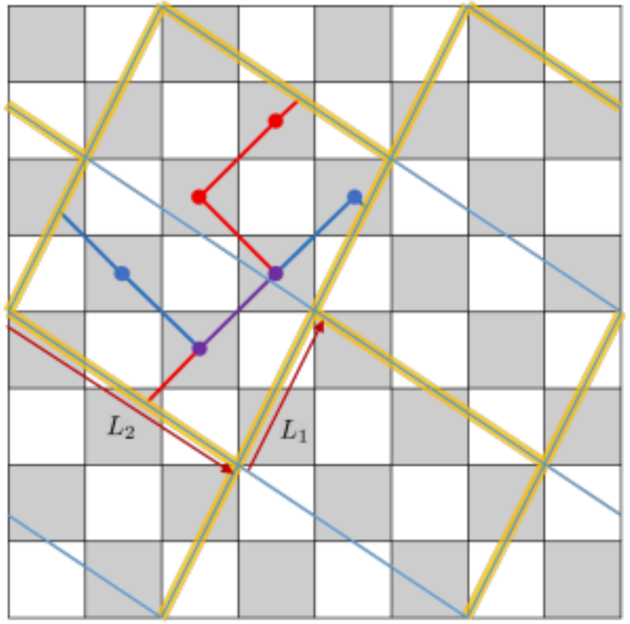


Figure 7: Graphs (gray edges with small, black vertices) on the torus (a) and projective plane, (b) and (c), are drawn with their medial graphs (black edges and large, red vertices). In (a), we get a $[\![8,2,2]\!]$ toric code. In (b), we get Shor's $[\![9,1,3]\!]$ code [41] with the gray graph coming from [39]. In (c), we show another $[\![9,1,3]\!]$ code from [39].



(a) $L_1 = (1, 2), L_2 = (2, -2)$



(b) $L_1 = (1, 2), L_2 = (3, -2)$

Figure 16: Regular tessellations of the infinite square lattice. Inside a single cell of a tessellation, the subgraph of the infinite square lattice defines a toric code. In fact, both images show two tessellations, one outlined in blue and one outlined in yellow. The blue tessellations have periodicity vectors L_1 and L_2 and define non-checkerboardable rotated toric codes. As argued in the text, the decoding graphs of these non-checkerboardable codes are isomorphic to a connected component of the decoding graphs of the corresponding checkerboardable codes defined by the yellow tessellations. Some paths (blue and red, purple where they overlap) in the yellow codes' decoding graphs are shown, which, when mapped to a single cell of the blue tessellation, are anticommuting logical operators.

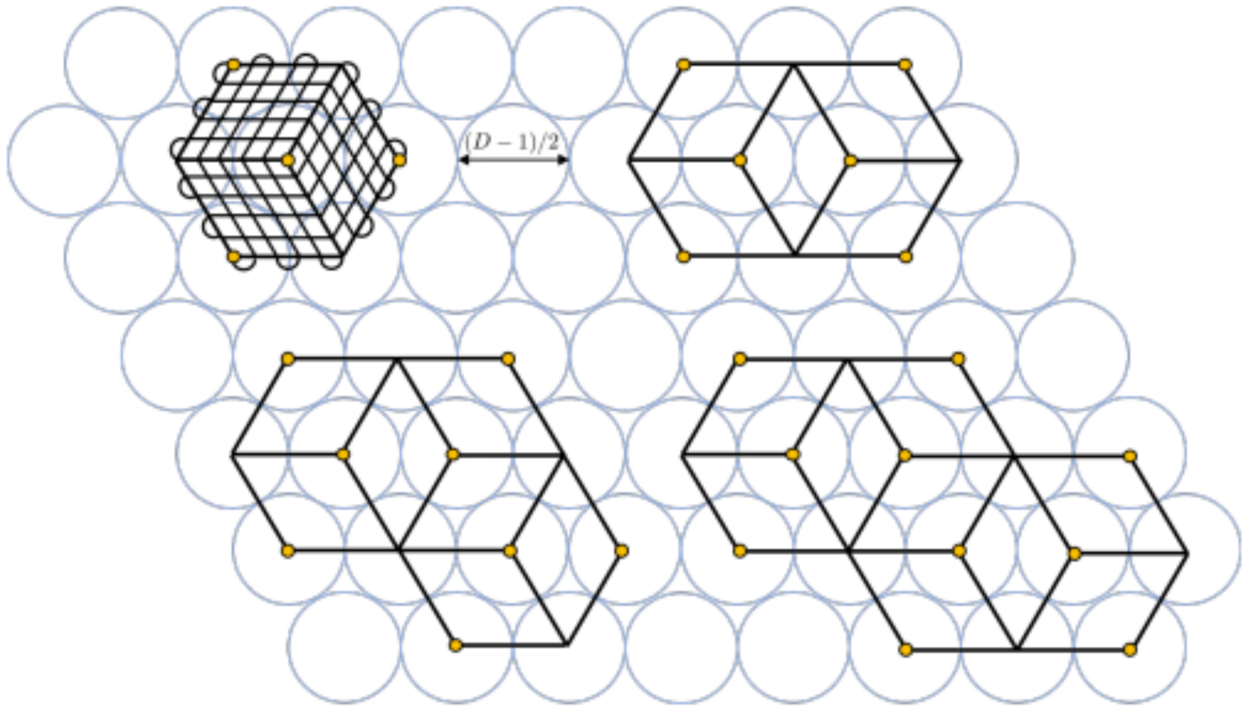


Figure 19: Circle-packing codes encoding $K = 1, 2, 3$, and 4 qubits in order of size. The packed circles in the background act as a coordinate system, and are not part of the graphs or codes. In the top left, we show the triangle code and the entire square lattices of qubits making it up. In other drawings, the lattice interiors are abstracted away. Odd-degree vertices, or twist defects, are shown as yellow circles. One can continue attaching square patches, following the pattern established in these first four examples, to grow the code down and to the right and obtain a family where $N = \frac{1}{4}(2K+1)D^2$. Therefore, $c = (2K+1)/4K$ approaches $1/2$ as K increases.

Code Analysis

IBM **Quantum**

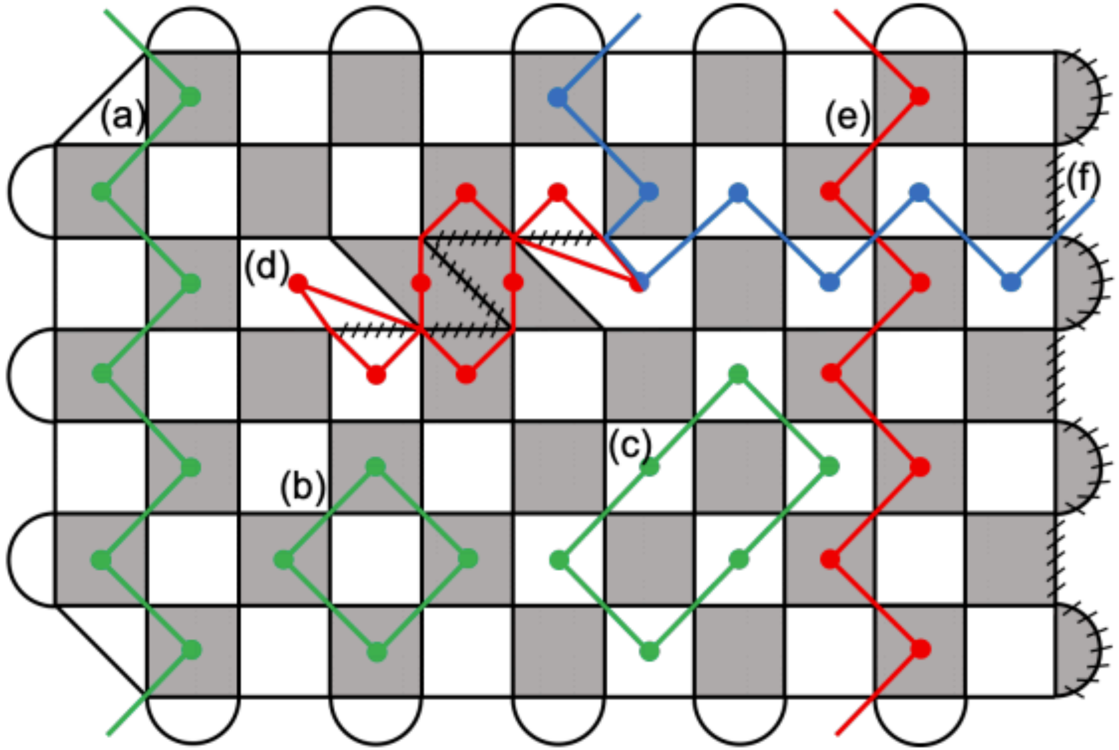


Figure 10: Example cycles in a decoding graph. The original graph has black edges, and stitched edges are part of a (particular choice of) defect. Faces are checkerboardable with this defect (the outer face should be colored black but is not shown). The entire decoding graph is not shown, but cycles labeled (a), (b), (c), (d), (e), and (f) illustrate some of it. Cycles (a), (b), (c) in green represent trivial logical operators (i.e. stabilizers), while cycles (d) and (e) in red and cycle (f) in blue represent non-trivial logical operators. We know they are non-trivial because (d) and (e) anticommute with (f). Cycles (a), (e), and (f) connect to the vertex in the outer face (not shown) via their dangling edges.

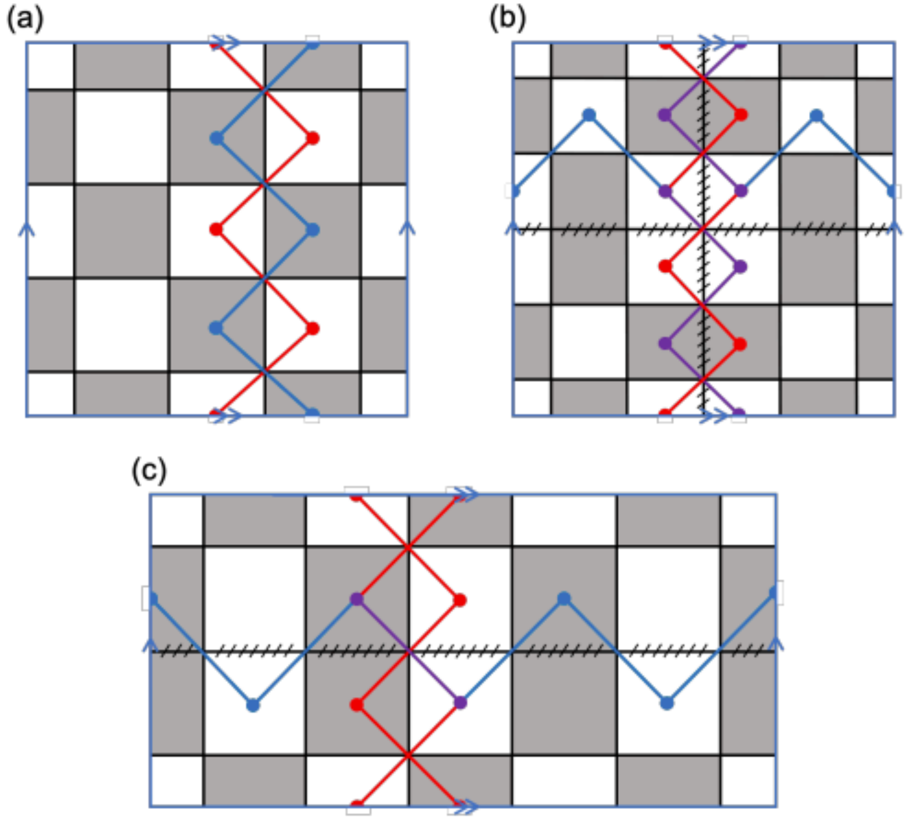


Figure 15: Three examples of square lattice toric codes. (a) An even×even code can be checkerboarded and encodes two logical qubits. We show two cycles, red and blue, in the decoding graph representing two commuting, non-trivial logical operators, e.g. $\bar{X}_1$ and $\bar{Z}_2$. (b) An odd×odd code can be checkerboarded with a defect (stitched edges) and encodes one logical qubit. We show two cycles, red and blue (with purple edges and vertices where they overlap), representing two anti-commuting logical operators, e.g. $\bar{X}$ and $\bar{Z}$. (c) An odd×even code is also checkerboardable with a defect and encodes one logical qubit.

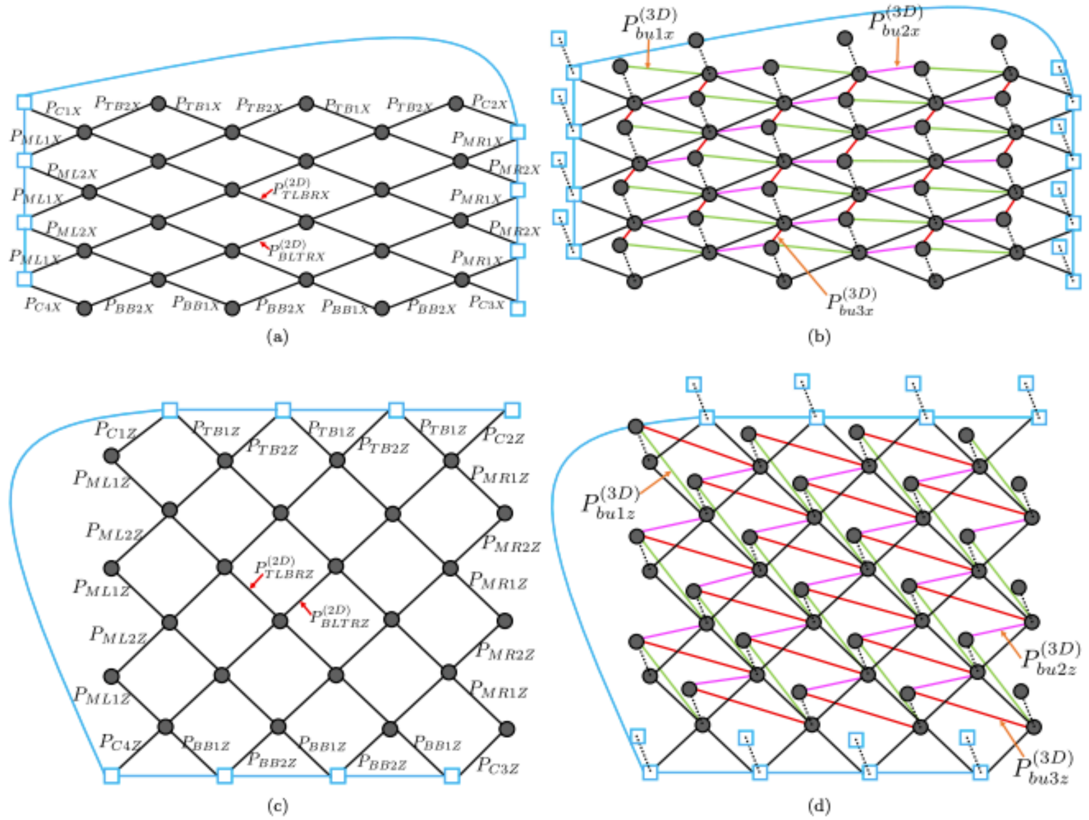


FIG. 13. Matching graphs corresponding to the surface code lattice in Fig. 5. (a) Two-dimensional matching graph with labeled boundary and bulk-type edge for X-type stabilizer measurement outcomes of the $d_x = d_z = 7$ surface code. (b) Same matching graph as in (a) but with vertical (dashed) and space-time correlated edges (with each type identified by a different color) connecting to an adjacent two-dimensional layer. (c) Two-dimensional matching graph with labeled boundary and bulk-type edge for Z-type stabilizer measurement outcomes of the $d_x = d_z = 7$ surface code. (d) Same matching graph as in (c) but with vertical and space-time correlated edges connecting to an adjacent two-dimensional layer. In all graphs, blue edges are boundary edges with zero weight, and blue squares correspond to virtual vertices connecting the boundary edges. Grey vertices correspond to measurement outcomes of the surface code stabilizers.

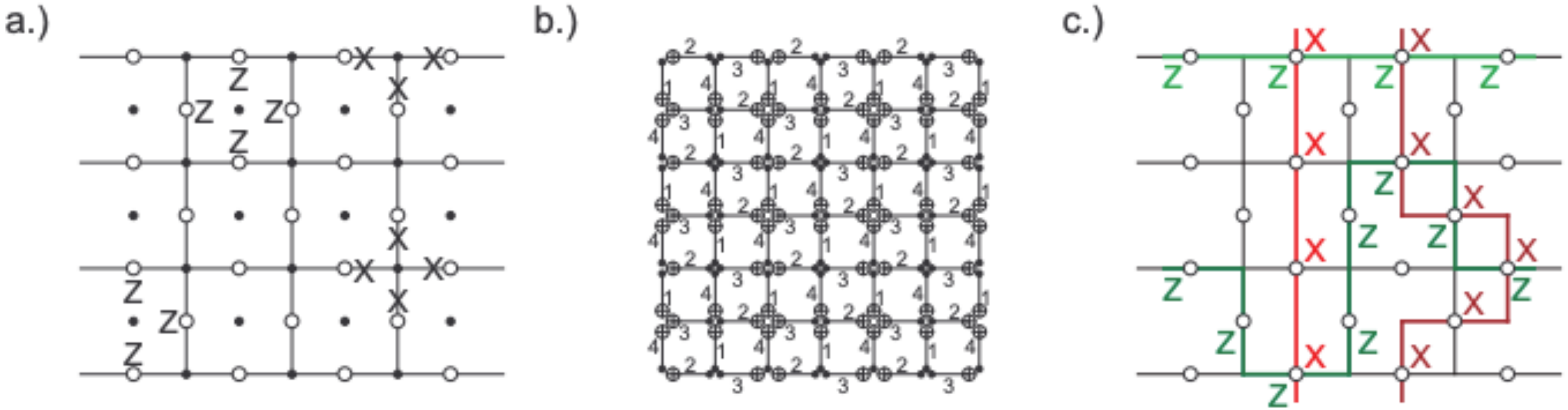


FIG. 2: a.) 2-D lattice of data (open circles) and syndrome (dots) qubits along with examples of the data qubit stabilizers. b.) Sequence of CNOTs permitting simultaneous measurement of all stabilizers. The numbers indicate the relative gate order. c.) Examples of logical operators.

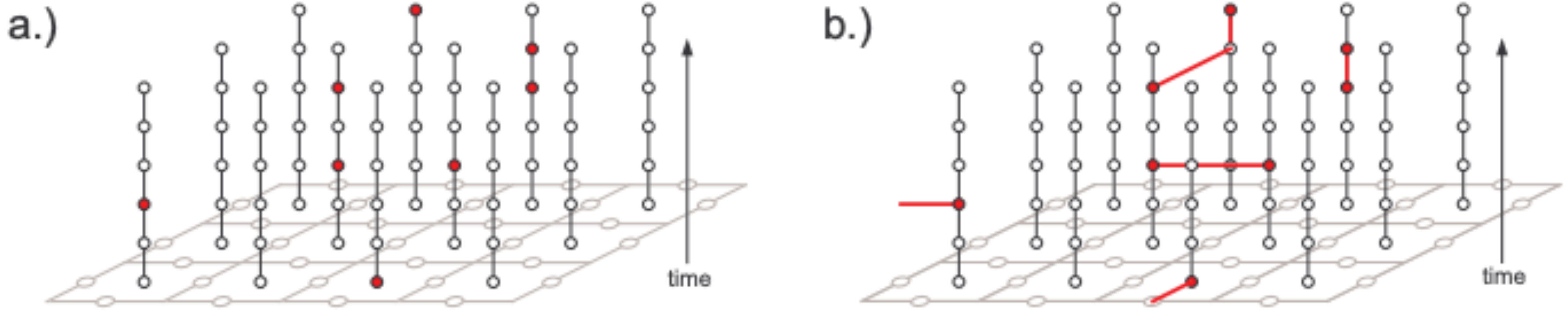


FIG. 3: a.) Locations in space and time, indicated by red dots, where and when the reported syndrome is different from that in the previous time step. Note that this is not a three-dimensional physical structure, just a three-dimensional classical data structure. b.) Optimal matching highly likely to lead to a significant reduction of the number of errors if bit-flips are applied to the spacelike edges.

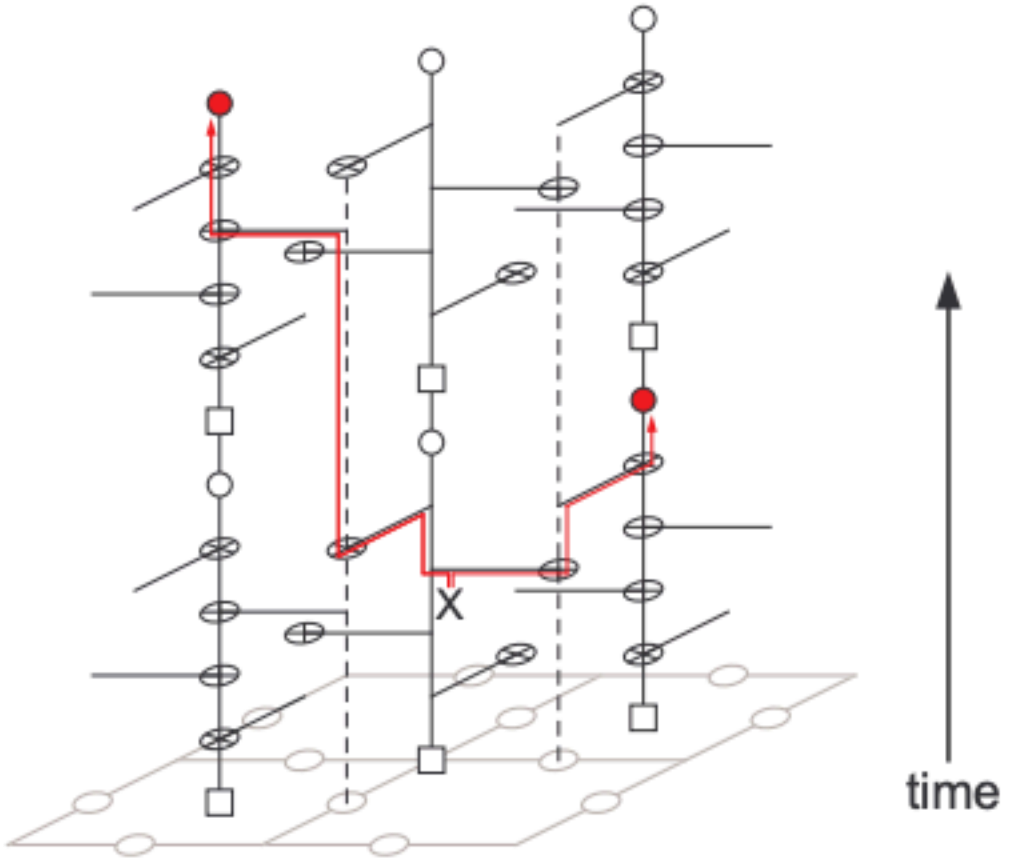
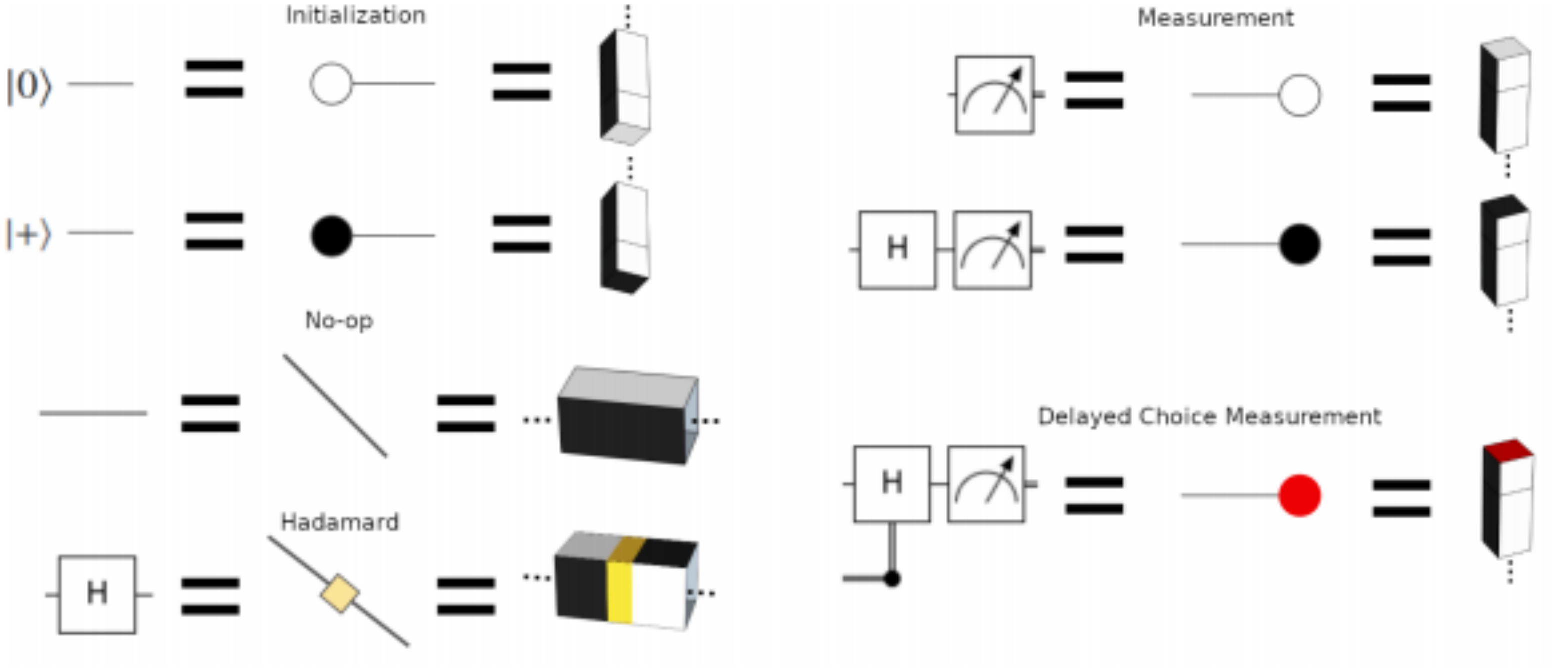


FIG. 5: Errors on syndrome qubits can propagate to two locations separated by two units of space and one unit of time.

Code Analysis


Initialization
$|0\rangle$
$|+\rangle$
No-op
Hadamard
H
Measurement
H
Delayed Choice Measurement
H

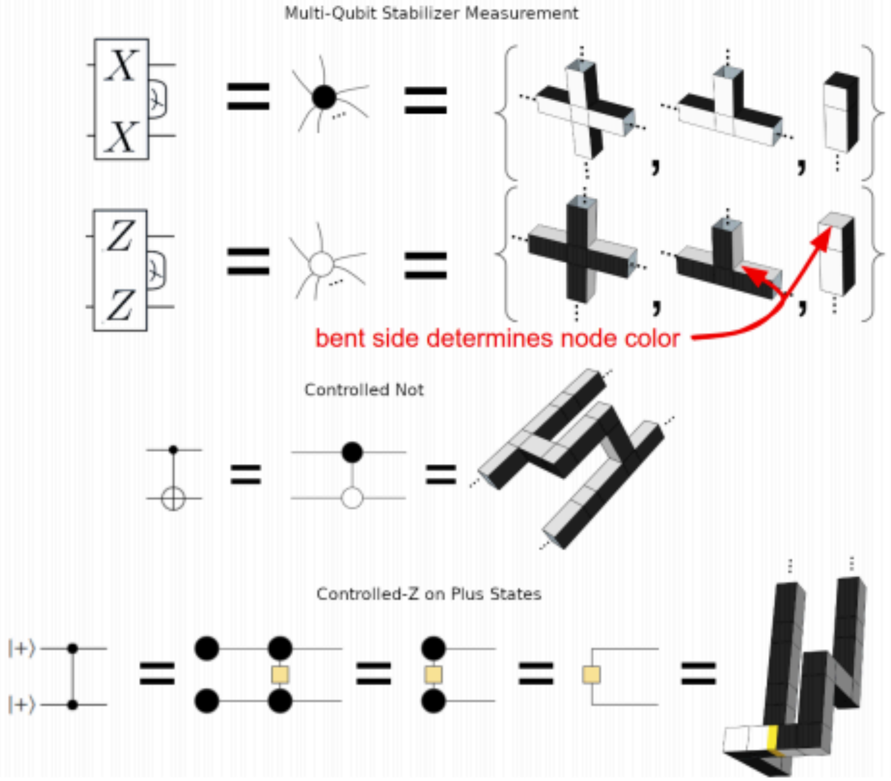

Multi-Qubit Stabilizer Measurement
X
X
Z
Z
bent side determines node color
Controlled Not
Controlled-Z on Plus States
|+⟩
|+⟩

Figure 1: Equivalent concepts expressed in quantum circuit diagrams, ZX calculus graphs, and 3d topological diagrams. In circuit diagrams, time goes from left to right. In ZX calculus graphs, there is no preferred time direction. In 3d topological diagrams, times goes from bottom to top. We never show Pauli operations in lattice surgery diagrams or in ZX calculus graphs, because they are performed by the classical control system instead of by operating on qubits. Our usage of the ZX calculus is somewhat non-standard in that we consider ZX graphs to be equivalent if they are equal modulo Pauli operations, we use a non-standard node coloring that matches the coloring of our topological diagrams, and we introduce a "delayed choice node" to represent adaptive effects coming from the classical control system. We exaggerate the spacing of our 3d topological diagrams, as in [11], so that it is possible to see how the components are interconnected.

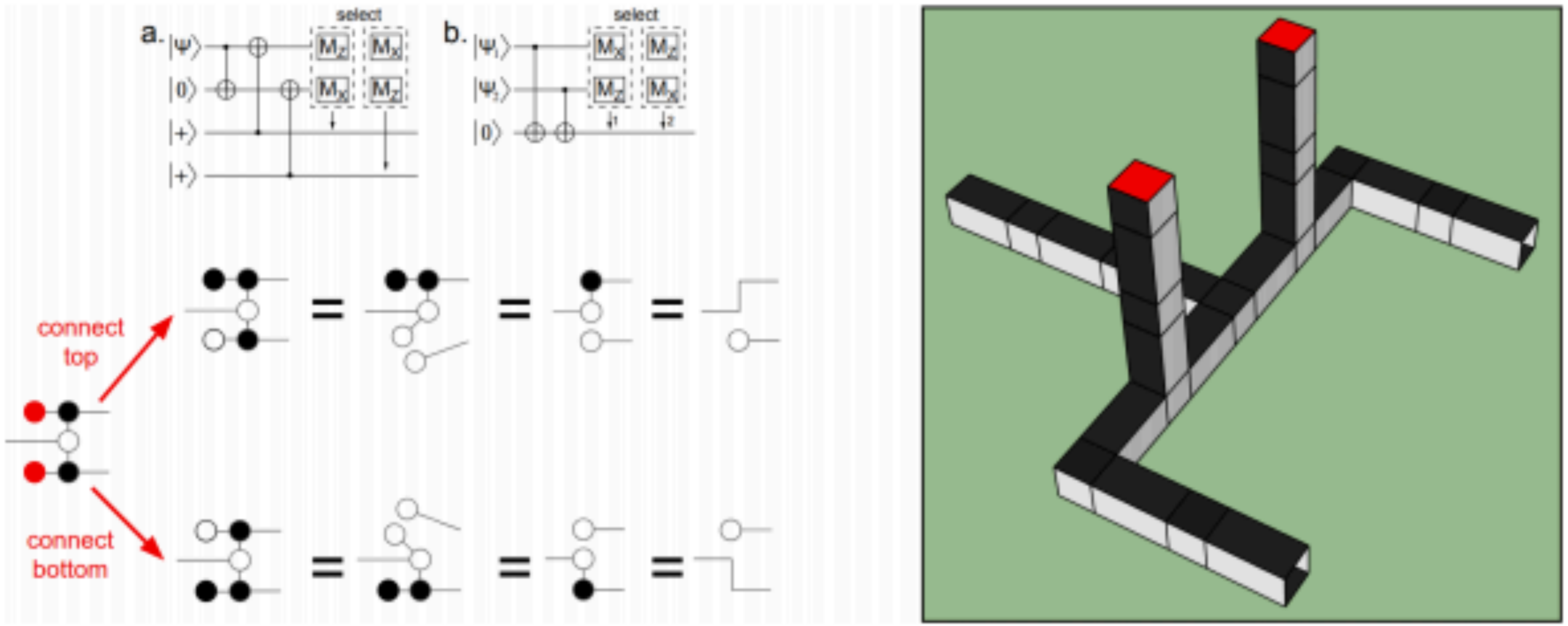


Figure 2: Delayed choice multiplex/demultiplex construction from [8]. Top left is a circuit diagram directly from [8]. The bottom left shows the process as a ZX calculus graph which, unlike the circuit diagram, is identical for multiplexing and demultiplexing. Known rewrite rules are used to show equivalence with the claimed "choose which route is connected" functionality. On the right side is a 3d topological diagram of a lattice surgery implementation of the construction. The vertical poles coming out of the branches of the fork are the routing qubits used to control which of the two branches connects to the rear trunk. They can be extended arbitrarily. The red squares atop the routing qubit columns are placeholders for an eventual X or Z basis measurement, to be determined by classical control software.

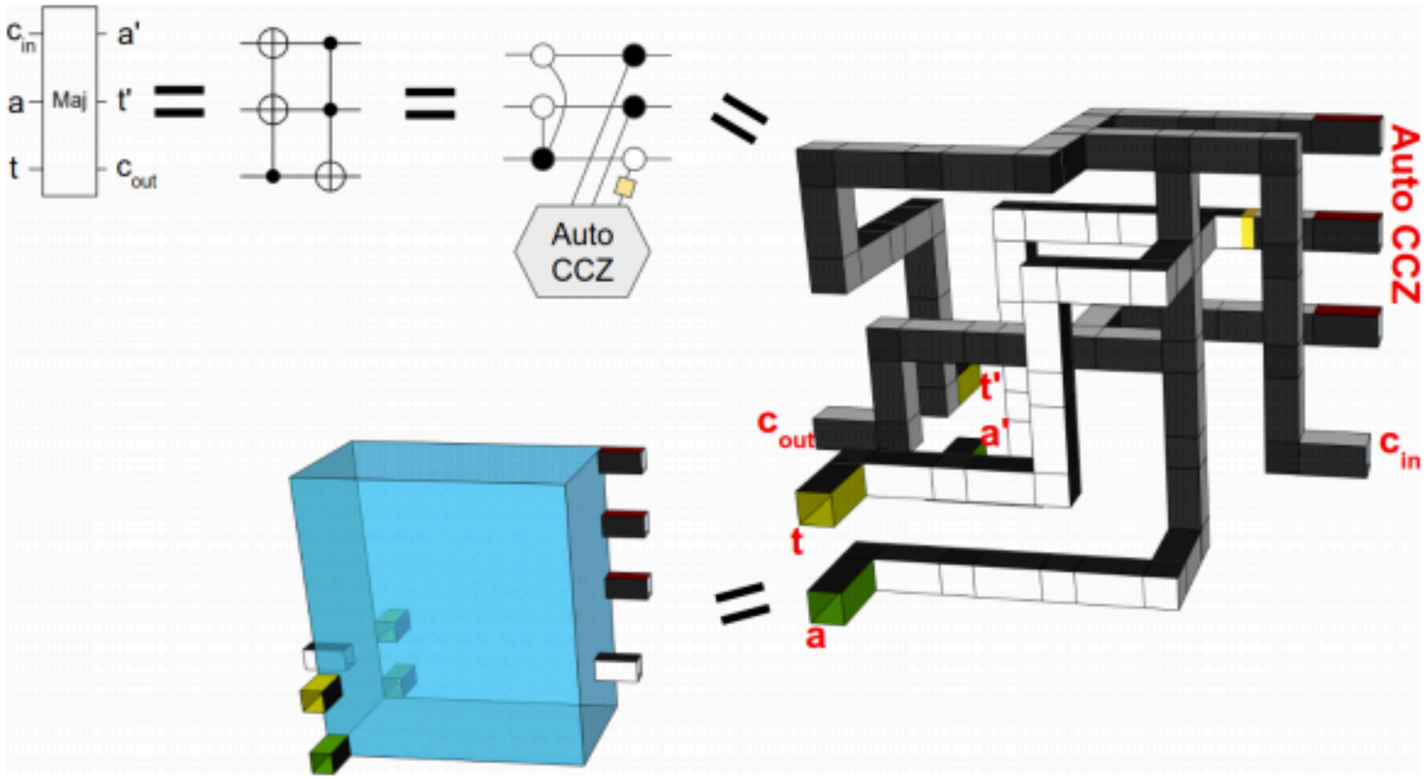


Figure 13: Lattice surgery implementation of the MAJ operation. Top-left is the MAJ circuit from Cuccaro's paper [4]. Top-center is a ZX calculus graph with function equivalent to the circuit. Center-right is a lattice surgery implementation of the ZX graph (stored in ancillary file "maj.skp" which can be opened online using Sketchup). Bottom-center is a simplified representation of the 3d model that shows only the ports and the 3x3x5 bounding box.

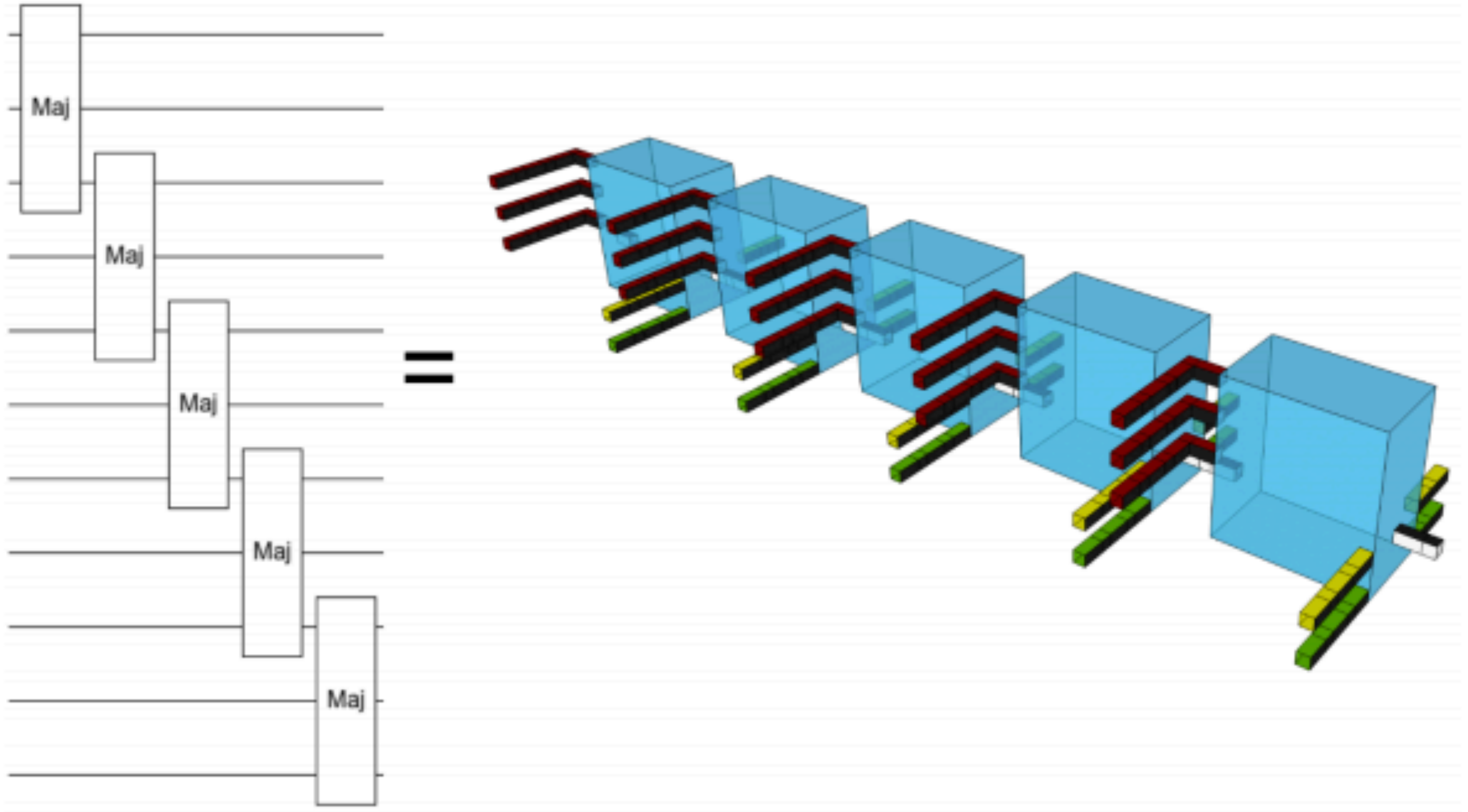


Figure 14: Lattice surgery implementation of a series of MAJ operations. Each light blue box is an instance of Figure 13. To enable parallelization, the carry propagation is done left to right, through space, instead of forward through time. Dark and light coloring indicates boundary type, but otherwise coloring is for labelling. Red bars are injecting three qubits from an AutoCCZ state, green bars are passing qubits from the target register through the block (front to back), and yellow bars are passing qubits from the offset register through the block (front to back).

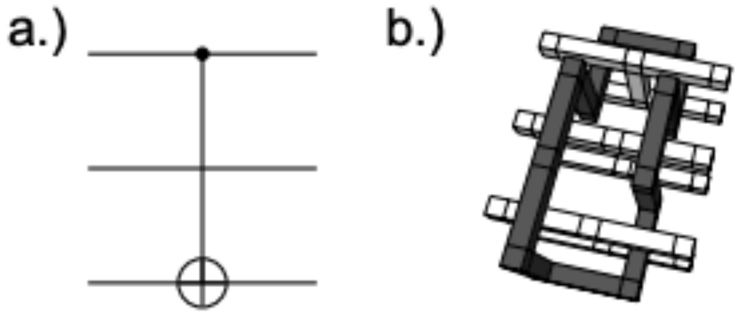


FIG. 4: a.) CNOT quantum circuit example. b.) Equivalent surface code CNOT [12, 13, 19]. Time runs from left to right. The scale of the figure is set by the code distance d. Small cubes are $d/4$ a side. Longer blocks have length d. Each unit of d in the temporal direction represents a round of error detection. Each unit of d in the two spatial directions represents two qubits. The structures are called defects, and represent space-time regions in which error detection has been turned off.

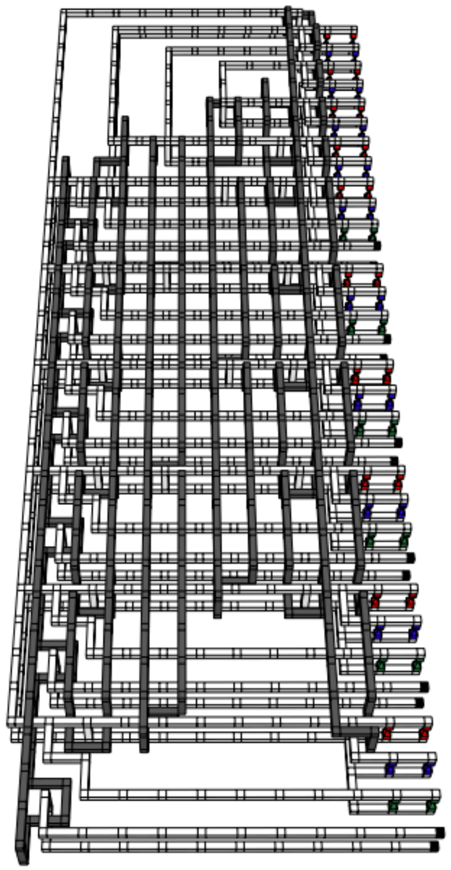


FIG. 6: Depth 12 compressed surface code implementation of Fig. 3. A larger version of this figure can be found in Appendix A, along with step-by-step images explaining how it was obtained.

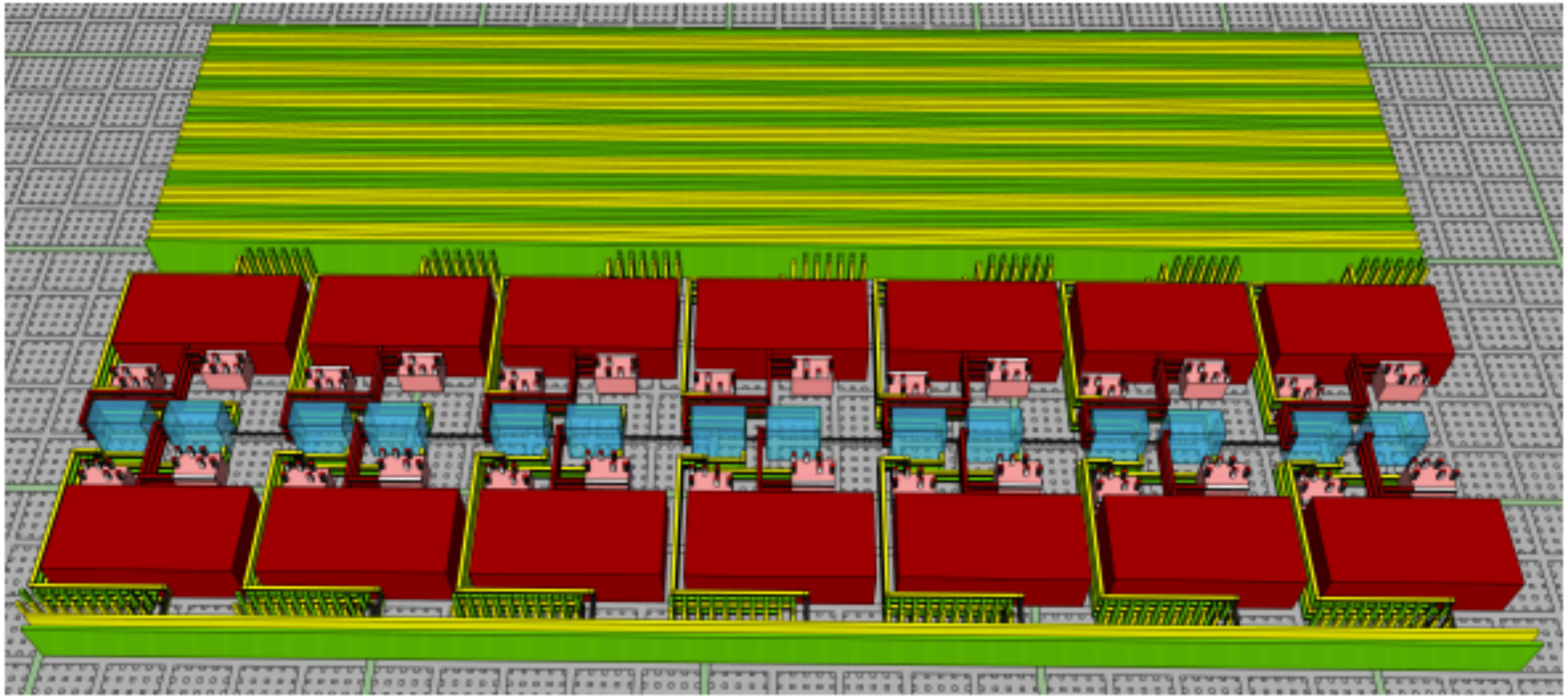


Figure 16: Surface code activity during an addition, assuming a level 2 code distance of 27 and a level 1 code distance of 17. This 3d model is stored in ancillary file "adder-layout.skp", which can be opened online using Sketchup. Yellow and green rows are qubits from the target and input registers of the addition. Dark blue rows are qubits from an idle register. Red boxes are CCZ magic state factories, auto-corrected by the pink boxes (see Figure 3). Light blue boxes are the MAJ operation of Cuccaro's adder (see Figure 13), arranged into a spacelike sweep to keep ahead of the control software (see Section 3). The full adder is formed by repeating this pattern, with the operating area gradually sweeping up and then down through the green/yellow data (see Figure 18).

Code Operations

IBM **Quantum**

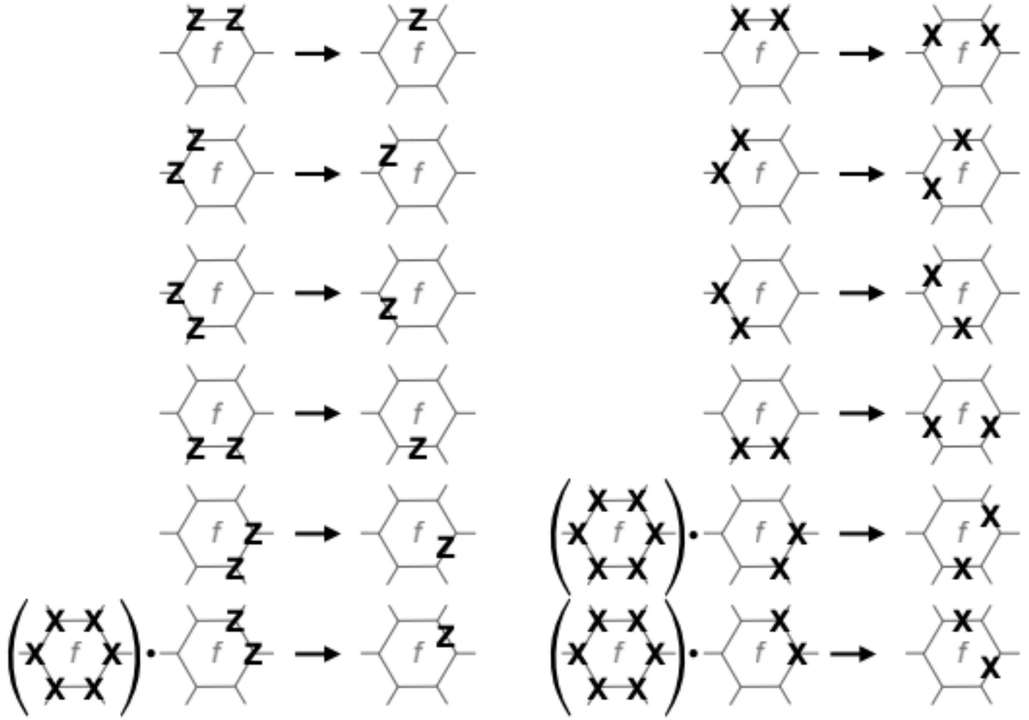
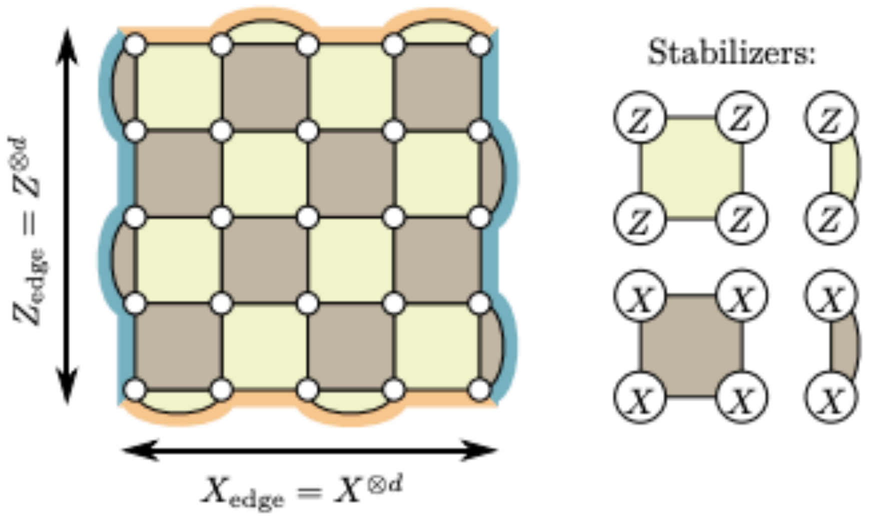


FIG. 3. Transformation of the operators of the color code $CC(\mathcal{L})$ supported on qubits of the face f colored in C under the disentangling unitary transformation U_f.



$Z_{\mathrm{edge}} = Z^{\otimes d}$
$X_{\mathrm{edge}} = X^{\otimes d}$
Stabilizers:
Z
X

Figure 1: An example of a surface code qubit with code distance $d = 5$. Physical qubits are located on the vertices, and the faces define the two- and four-qubit Z type (bright) and X type (dark) stabilizer operators. X strings along the X edge (orange) are logical X_L operators, whereas Z strings along Z edges (blue) are Z_L operators.

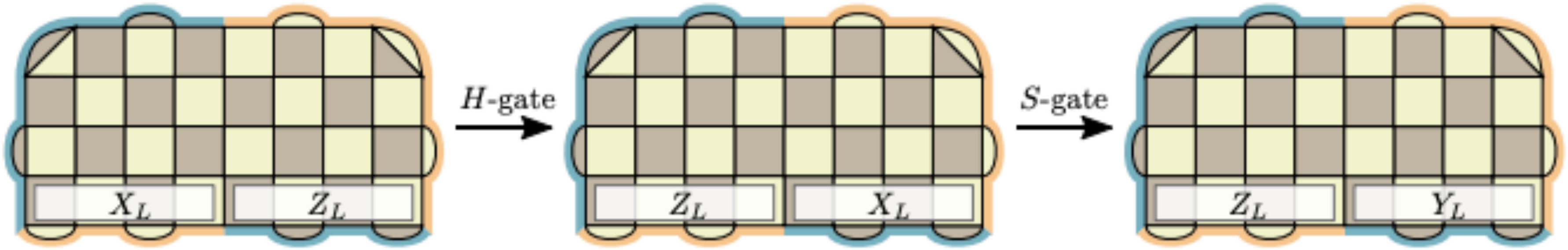


Figure 2: An example of edge tracking with a wide surface code qubit. Starting from the default encoding $X_{\text{edge}} = X_L$ and $Z_{\text{edge}} = Z_L$, an H gate changes it to $X_{\text{edge}} = Z_L$ and $Z_{\text{edge}} = X_L$, and a subsequent S gate modifies it to $X_{\text{edge}} = Z_L$ and $Z_{\text{edge}} = Y_L$.

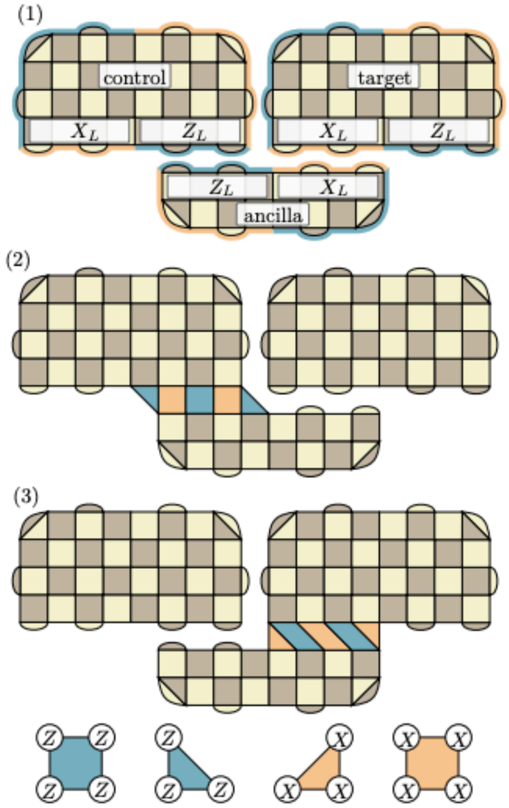


Figure 4: CNOT by lattice surgery corresponding to the gate circuit in Fig. 3. (1) All qubits are in the default encoding $X_{\text{edge}} = X_L$ and $Z_{\text{edge}} = Z_L$, and the ancilla is initialized in the $|+\rangle$ state. (2) To measure the $Z_L \otimes Z_L$ parity between control and target, the two-qubit boundary stabilizers are merged (orange), and new Z type stabilizers (blue) are introduced, whose product is precisely the parity. (3) Similarly, the $X_L \otimes X_L$ parity between ancilla and target is measured by the product of new X type stabilizers (orange).

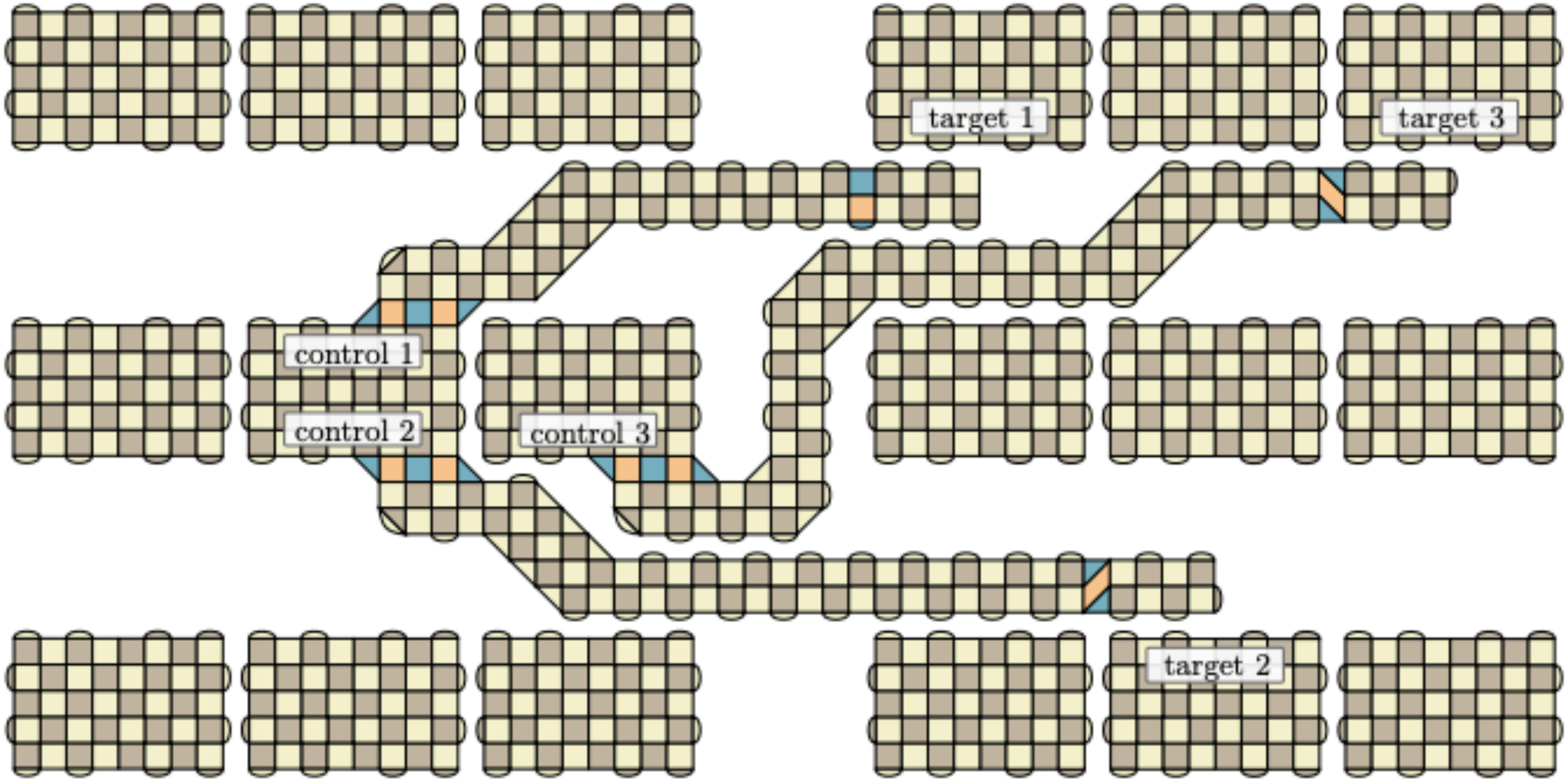


Figure 8: Example of a two-dimensional arrangement of surface code qubits, where qubits are grouped in blocks of six. The long ancilla qubits can be used for three simultaneous long-range CNOT gates.

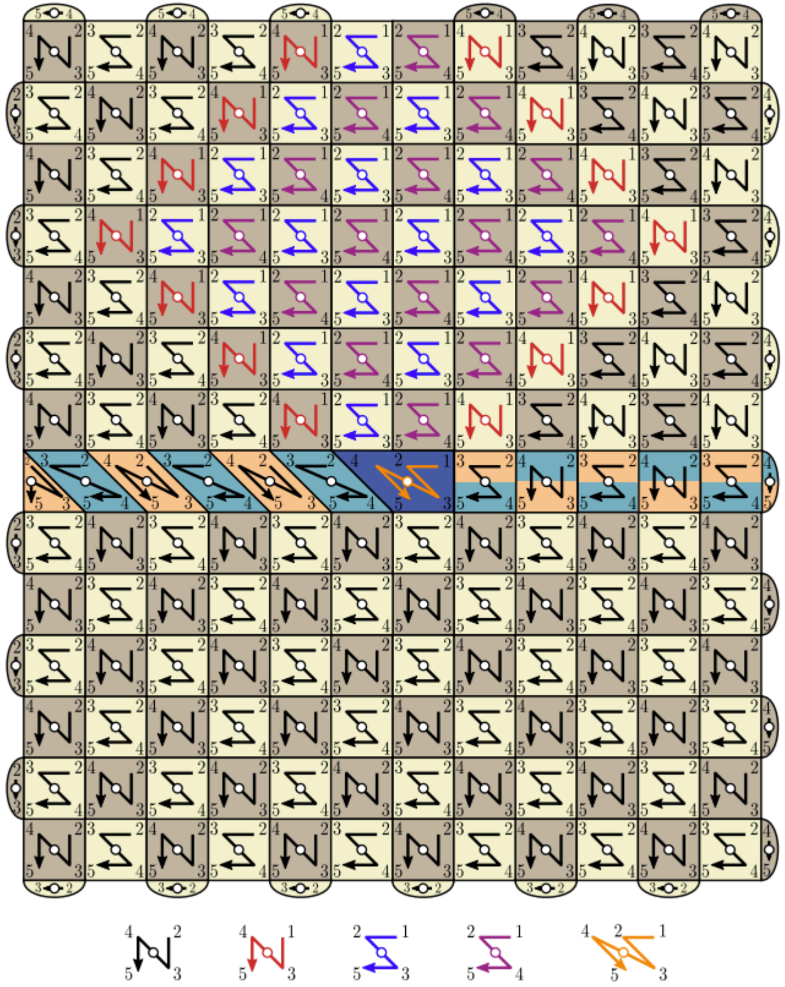


Figure 16: Possible ordering of two-qubit gates that fulfills the three conditions shown in Fig. 15. This stabilizer configuration is the most general, as it involves bulk stabilizers, a dislocation line, and a twist defect. It corresponds to a twist-based lattice surgery between a double-sided qubit and a standard rectangular qubit.