Quantum Software Mini-Project

Șerban Cercelescu

Hillary Vacation
9th of April 2024

Page intentionally left blank.

Abstract

We describe a procedure that given a quantum circuit consisting of arbitrary single qbit gates gates, arbitrary single qbit initializations and CNOTs produces a larger fault tolerant circuit using surgery on planar codes, as described in [4, 5, 10, 8, 7]. We describe the implementation of the logical gates gates and the moving of a qbit around an array of physical qbits to achieve universal fault-tolerant computation. To preserve any trace of brevity of this document, we will assume familarity with the CSS codes and refer the reader to [2] and [3] for an introduction to the topic. Despite the best efforts of the author to avoid overly technical aspects of the mathematics involved, basic familiarity with rings and modules and basic algebraic topology (chapter 2 in [9]) is likely necessary to read this paper.

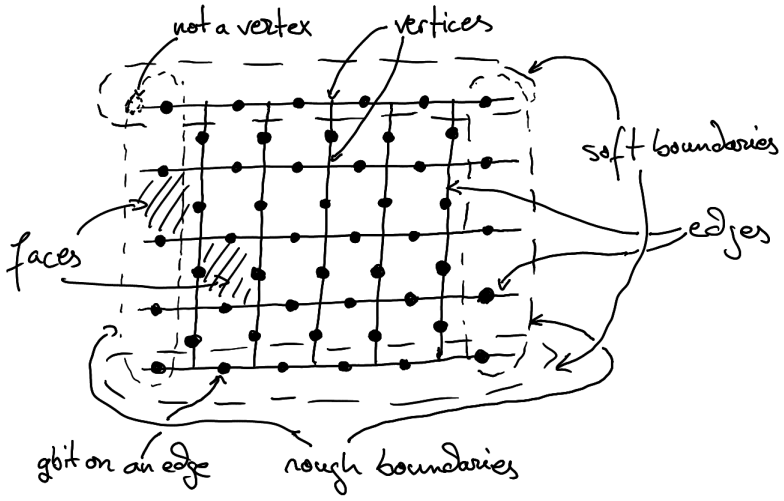
1 Quantum Error Correction

For a very long time, quantum computing was considered to be a theoretical endeavour with no realistic prospect of implementation in the real world due to the difficulty of preventing the decoherence of quantum states in any realistically constructable device, in a similar way to how analogue computing showed the promise of exponential speedups in solving certain problems, promise which succumbed to the realization that in actual implementations of analogue computers, precision errors arising from noise limited their utility. Moreover, the possibility of extending the theory of classical error correction codes to the quantum domain seemed doomed by the no-cloning theorem [5]. This, however was changed when Shor introduced his [9,1,3] code in 1995 [13]. Since then, the body of literature of quantum error correction has developed until reaching its current considerable size. One of the key lines of research in the field is that of topological quantum error correction (TQE), tracing its origin in the work of Kitaev in condensed matter physics [11] and best formalized through the correspondence between CSS codes and chain complexes as outlined in [5]. In this paper, we only discuss logical qubits encoded via planar codes as in [7] with logical gates implemented via lattice surgery as in [10, 8], however it is worth mentioning that alternative TQE codes exist, such as honeycomb cones, colour codes, hyperbolic surface codes and a plethora of higher dimensional TQE codes.

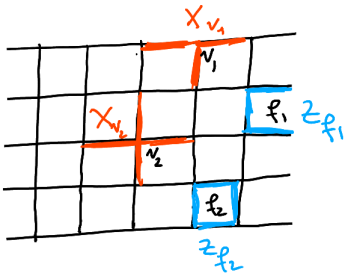
2 The Logical Qbit

In this section we discuss how to encode a logical qbit via planar codes as in [7, 4]. Consider a set of physical qbits, labeled by the edges of an $N \times M$ grid such as the 5×5 one depicted in Figure 2.1. We write V for the set of vertices of the grid, E for the set of edges of the grid and F for the set of faces of the grid.

A logical qbit is encoded by the CSS code with the following stabilizers: to each vertex v in the grid with incident edges e_1, e_2, e_3, e_4 we associate a stabilizer $\mathbf{X}_v := X_{e_1}X_{e_2}X_{e_3}X_{e_4}$, which we'll refer to by the name X-**stabilizer** or **vertex stabilizer**. For the case of a vertex v located on the rough boundary of the grid, with incident edges e_1, e_2, e_3, we define $\mathbf{X}_v := X_{e_1}X_{e_2}X_{e_3}$. To each face f in the grid bounded by the edges e_1, e_2, e_3, e_4, we associate the stabilizer $\mathbf{Z}_f := Z_{e_1}Z_{e_2}Z_{e_3}Z_{e_4}$ which we will refer to by Z-**stabilizer** or **face stabilizer**. Similarly, to a face located on the soft boundary of the grid bounded by the edges e_1, e_2, e_3, we associate the stabilizer $\mathbf{Z}_f := Z_{e_1}Z_{e_2}Z_{e_3}$. We establish the convention of always denoting stabilizers in bold.



[Figure 2.1] Most times we won't actually draw the qbits on the edges



[Figure 2.2]

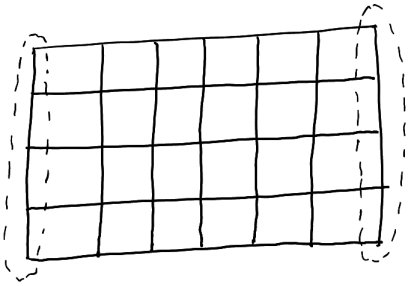
It is easy to see that indeed all the stabilizers we have defined commute, as any two face and vertex stabilizers share either two edges or none, and so $\mathcal{S} := \langle \{\mathbf{X}_v : v \in V\} \cup \{\mathbf{Z}_f : f \in F\} \rangle$ is indeed a stabilizer group.

Note that for an $N \times M$ grid, we have $2NM + N - M$ edges and thus as many physical qbits, $(M+1)(N-1)$ face stabilizers and NM vertex stabilizers, thus, by the fundamental theorem of stabilizer theory (as presented in [3]), our code encodes 1 logical qbit.

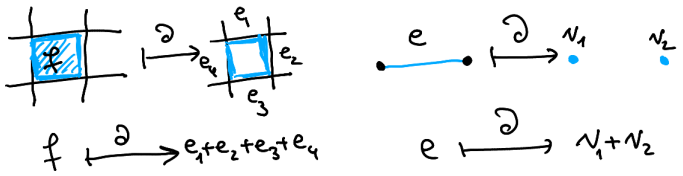
2.1 The Topology

Consider the following extension of the grid in Figure 2.1 by edges that "fill" the rough boundaries.

Let us write F_0, E_0, V_0 for the set of all faces, edges and vertices respectively in the extended grid and let us denote by $\mathrm{F}_{\mathrm{bdr}}$, $\mathrm{E}_{\mathrm{bdr}}$, $\mathrm{V}_{\mathrm{bdr}}$ the subsets of the previous of all faces edges and vertices respectively that are completely included in the rough boundary (note that $\mathrm{F}_0 = \varnothing$). We define the **boundary maps** $\partial_2 : \mathbb{Z}_2\,\mathrm{F}_0 \to \mathbb{Z}_2\,\mathrm{E}_0$, $\partial_1 : \mathbb{Z}_2\,\mathrm{E}_0 \to \mathbb{Z}_2\,\mathrm{V}_0$ over the free modules over the sets $\mathrm{F}_0, \mathrm{E}_0, \mathrm{V}_0$ in the obvious way, as depicted in Figure 2.4:



[Figure 2.3]



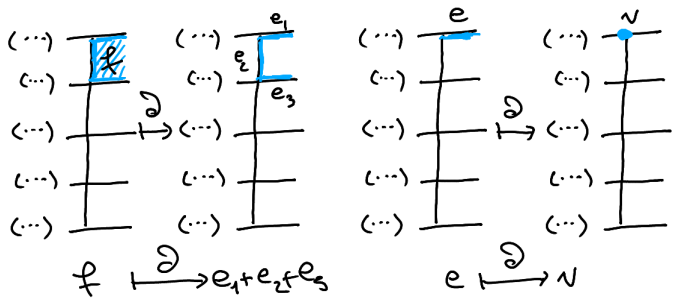
[Figure 2.4]

Note that these boundary maps restrict to the submodules corresponding to the rough boundary $\partial_2 : \mathbb{Z}_2\,\mathrm{F}_{\mathrm{bdr}} \to \mathbb{Z}_2\,\mathrm{E}_{\mathrm{bdr}}$, $\partial_1 : \mathbb{Z}_2\,\mathrm{E}_{\mathrm{bdr}} \to \mathbb{Z}_2\,\mathrm{V}_{\mathrm{bdr}}$. We define the quotient modules $C_2 := \mathbb{Z}_2\,\mathrm{F}_0 \,/\, \mathbb{Z}_2\,\mathrm{F}_{\mathrm{bdr}}$, $C_1 := \mathbb{Z}_2\,\mathrm{E}_0 \,/\, \mathbb{Z}_2\,\mathrm{E}_{\mathrm{bdr}}$, $C_0 := \mathbb{Z}_2\,\mathrm{V}_0 \,/\, \mathbb{Z}_2\,\mathrm{V}_0$ and, again, committing the mild sin of overloading the name and notation of previously defined functions, write $\partial_1 : C_2 \to C_1$, $\partial_2 : C_1 \to C_0$ for the boundary maps induced from the original ones, obtaining the following chain complex:

$$C_0 \xleftarrow{\partial_1} C_1 \xleftarrow{\partial_2} C_2$$

Note that there is a one to one correspondence between the elements in C_1 and formal $\mathbb{Z}_2$-linear combinations (or equivalently, subsets) of edges in the original grid (i.e. the grid without the rough edge extension). There is an analogous connection between C_0 and C_2 and the vertices and faces of the original grid.

The key advantage of identifying subsets of faces, edges and vertices with elements of the grid is that it follows the intuitive notion of boundary near the rough edges in a formal framework that enables us to use the theory of algebraic topology to prove and construct things. Pictorially, boundaries near the rough boundaries (left and right side of the grid behave as follows):



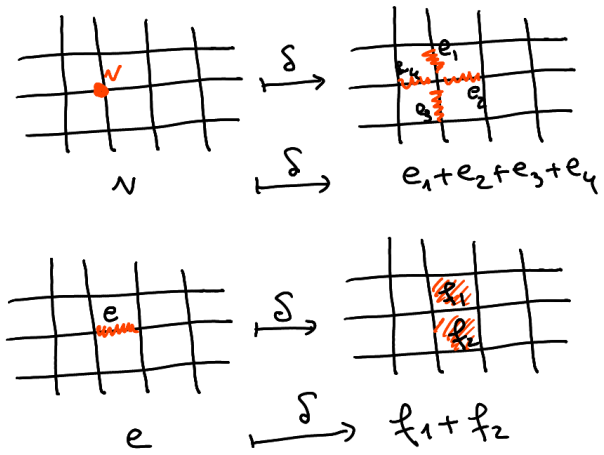
[Figure 2.5]

In line with the language of algebraic topology, for $k \in [3]$ (where $[n] = \{1, \ldots, n\}$), we will refer to members of the module C_k by **k-chains**, to boundaries of $(k+1)$-chains (i.e. to elements in the image of ∂_{k+1}) by **k-boundaries** and to k-chains whose boundary vanishes (i.e. elements in the kernel of ∂_k) by **k-cycles**.

Furthermore, we will consider the dual chain complex:

$$C^0 \xrightarrow[\delta_1]{} C^1 \xrightarrow[\delta_2]{} C^2$$

where $C^k := \mathrm{Hom}_{\mathbb{Z}_2}(C_k, \mathbb{Z}_2)$ for $k \in [3]$ and $\delta_k := \mathrm{Hom}_{\mathbb{Z}_2}(\partial_k, \mathbb{Z}_2)$ for $k \in [2]$. We call the maps δ_1, δ_2 **coboundary maps**, elements of C^k **k-cochains**, elements in the kernel of δ_k **k-cocyles** and elements in the image of δ_{k-1} **k-coboundaries**.



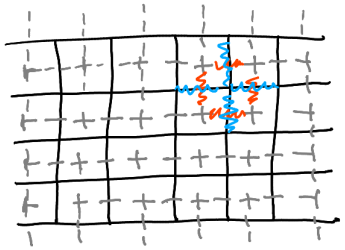
[Figure 2.6]

Noting that $C^k = \mathrm{Hom}(C_k, \mathbb{Z}_2) \simeq C_k$, we may identify k-chains with k-cochains by the isomorphism given by the "inner product on $\mathbb{Z}_2$" $-(-)^* : C_k \to C^k$ mapping an arbitrary chain $c = \sum_{e \in E} a_e e$ (for some $\{a_e \in \mathbb{Z}_2\}_{e \in E}$) to the map:

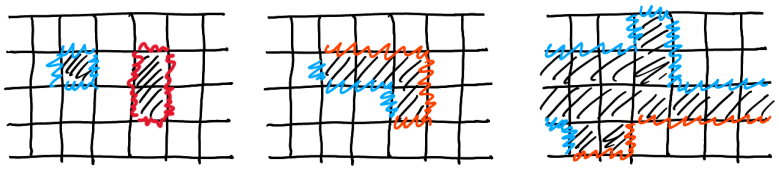
$$c^* : \sum_{e \in E} b_e e \mapsto \sum_{e \in E} a_e b_e.$$

A strong visual aid in understanding these dual chains and their coboundaries is that of the **dual lattice** (which in the case of our grid, can be thought of as the dual graph), illustrated by the dotted lines in Figure 2.7. The key idea here is that a set of edges in the dual lattice corresponds to the set of edges in the original lattice that it intersects. Moreover, one can notice how the blue cocycle in the original grid corresponds to a cycle in the dual lattice. Formally, this correspondence is given by the construction of the Lefschetz duality isomorphism in the orientable manifold with boundary $S^1 \times [0, 1]$ pictured in Figure 2.10.

We will now define the **homology** and **cohomology groups** of our chain complexes. We say that two k-chains $c_1, c_2 \in C_k$ are **homologous** if they differ by a boundary, i.e. if $c_1 - c_2 = \partial_{k+1} f$ for some $f \in C_{k+1}$. Similarly, we say that two cochains $c^1, c^2 \in C^k$ are **homologous** if they differ by a coboundary. For example, the red and blue 1-chains in the Figure 2.8 below are homologous because they differ by the boundary of the shaded region.



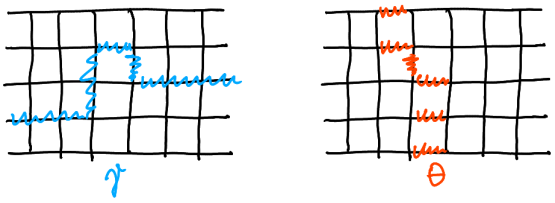
[Figure 2.7]



[Figure 2.8]

A (co)cycle that is homologous to the zero (co)chain is said to be **nullhomologous** or **contractible**. The k^{th} homology group H_k of a chain complex consists of the equivalence classes of k-cycles modulo homology (being homologous is an equivalence relation). Formally, $H_k := \text{Ker}[\partial_k]/\,\text{Im}[\partial_{k+1}]$. Analogously, the k^{th} cohomology group H^k consists of the equivalence classes of k-cycles modulo homology, or formally $H^k := \text{Ker}[\delta_{k+1}]/\,\text{Im}[\delta_k]$. It is worth stressing that elements of the (co)homology groups are not (co)cycles, but equivalence classes of (co)cycles.

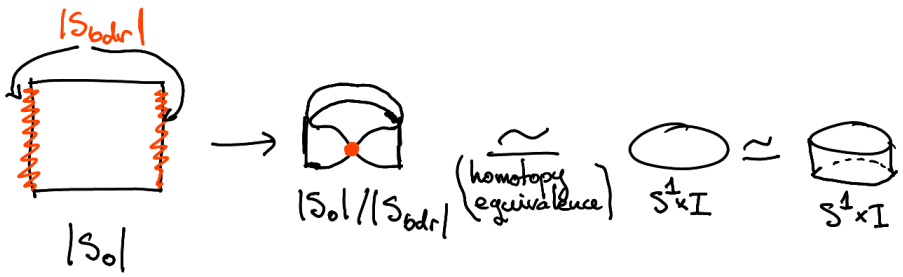
In the context of the chain complex associated to our grid, we have that $H_1 \simeq H^1 \simeq \mathbb{Z}_2$ and $H_k = H^k = 0$ for all $k \neq 1$, the proof of which will conclude this subsection. Writing $H_1 = \{[0], [\gamma]\}$ and $H^1 = \{[0], [\theta]\}$, we may depict the 1-cycle γ and 1-cocycle θ as:



[Figure 2.9]

It is important to note that a (connected) cycle (or cocycle) is non-trivial (not nullhomologous) iff it traverses the grid from one rough (or soft, respectively) boundary to the other.

We now show that $H_1 \simeq H^1 \simeq \mathbb{Z}_2$, using a purely geometric argument. We may consider the cubical complex S_0 corresponding to our grid and S_b as the subcomplex of S_0 corresponding to the rough boundary. Writing $|-|$ for the geometric realization functor, we have that $|S_0|$ is homeomorphic to $[0,1] \times [0,1]$ and S_b corresponds to its $\{0,1\} \times [0,1]$ subspace. We have that $H_1 = H_1(S, S_0; \mathbb{Z}_2) \simeq H_1(|S|, |S_0|; \mathbb{Z}_2)$ and as $|S_0|$ is a contractible good neighbourhood in $|S|$, we have that $H_1(|S|, |S_0|; \mathbb{Z}_2) \simeq H_1(|S|/|S_0|; \mathbb{Z}_2)$. Finally, $|S|/|S_0|$ visibly deformation retracts to a circle and so $H_1 \simeq H_1(|S|/|S_0|; \mathbb{Z}_2) \simeq H_1(S_1; \mathbb{Z}_2) \simeq \mathbb{Z}_2$. By Poincaré duality, it follows that $H^1 \simeq \mathbb{Z}_2$ as well.



[Figure 2.10]

For those already familiar to topological error correction, it is worth mentioning that the argument above formally justify the clear intuition that a logical $|+\rangle_L$ state can be thought of as a cycle going around a band, in the same way as for the toric code states can be thought of in terms of cycles going around it.

2.2 The Logical Qbit and Pauli Gates

At last, armed with the mathematical framework from the previous section, we may return to qbits and error correction. First, as there is a one to one correspondence between edges in our grid and physical qbits, there is also a one to one correspondence between 1-chains (elements of C_1) and elements of the "X-basis" of the system. Formally, given a chain $c = \sum_{e \in E} a_e e$, writing:

$$|c\rangle := \bigotimes_{e \in E} \begin{cases} \text{if } a_e = 0: & |+\rangle \\ \text{otherwise:} & |-\rangle \end{cases},$$

we get that $\{|c\rangle : c \in C_1\}$ is an ONB for the state space of the physical qbits in the grid. Similarly, we may label Pauli operators acting on the physical qbits by (co)chains, i.e:

$$Z_c := \otimes_{e \in E} \begin{cases} \text{if } a_e = 0: & Z \\ \text{otherwise:} & I_2 \end{cases} \quad X_{c^*} := \otimes_{e \in E} \begin{cases} \text{if } a_e = 0: & X \\ \text{otherwise:} & I_2 \end{cases}$$

We will often refer to products of X and Z gates on the physical qbits by X**-chains** and Z**-chains**. Note that we label products of Z gates by 1-chains and products of X gates by cochains. This is notationally convenient, as given cycles $c_1, c_2 \in C_1$, we have that:

$$Z_{c_1} |c_2\rangle = |c_1 + c_2\rangle \quad X_{c_1^*} |c_2\rangle = (-1)^{c_1^*(c_2)} |c_2\rangle$$

We now move towards defining a basis for the space stabilized by the group $\mathcal{S}$, which we've defined at the start of this section. First, note that given a chain c_1, $|c_1\rangle$ is stabilized by all the $\mathbf{X}_v$ stabilizers iff c_1 is a cycle. This is both easy to prove and obvious upon examining a few examples. Second, note that for any face $f \in C_2$, writing $\mathbf{Z}_f |c_1\rangle = |c_2\rangle$, we have that $c_2 = c_1 + \partial f$, so c_2 is homologous to c_1. This motivates the definition:

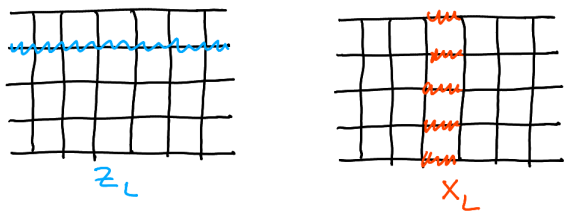
$$|+\rangle_L := \frac{1}{\sqrt{|[0]|}} \sum_{c \in [0]} |c\rangle \quad |-\rangle_L := \frac{1}{\sqrt{|[\gamma]|}} \sum_{c \in [\gamma]} |c\rangle$$

where $H_1 = \{[0], [\gamma]\}$. In other words, $|+\rangle_L$ is the uniform superposition over the states of the system corresponding to contractible cycle and $|-\rangle_L$ is the uniform superposition over the states corresponding to non-contractible cycles. One can to see that this indeed forms an ONB for the codespace, as the states have disjoint supports and are thus orthogonal, both states are stabilized by $\mathcal{S}$ and the codespace has dimension 2. Following common sense, we also define:

$$|0\rangle_L := \frac{|+\rangle_L + |-\rangle_L}{\sqrt{2}} \quad |1\rangle_L := \frac{|+\rangle_L - |-\rangle_L}{\sqrt{2}}$$

We now move towards finding the logical Pauli X_L and Z_L gates acting on the stabilized space in the obvious way: $Z_L |+\rangle_L = |-\rangle_L$, $Z_L |-\rangle_L = |+\rangle_L$, $X_L |0\rangle_L = |1\rangle_L$ and $X_L |1\rangle_L = |0\rangle_L$. To do so, we first establish some criteria for membership in the normalizer group $N(\mathcal{S})$ for products of Pauli operators over the physical qbits: a Pauli operator X_{c^*} lies in $N(\mathcal{S})$ iff c^* is a cocycle and a Pauli operator Z_c lies in $N(\mathcal{S})$ iff c is a cycle. Again, it is easy to convince yourself of this by looking at a few examples. We may now define X_L and Z_L as follows: writing $H_1 = \{[0], [\gamma]\}$, $H^1 = \{[0], [\theta]\}$ for an arbitrary choice of chain representatives of their respective (co)homology classes $\theta \in C_1$ and $\gamma \in C^1$, we set $X_L := X_\theta, Z_L := Z_\gamma$. Proving that these operators act as desired on our basis elements is a matter of straightforward computation.

We now have enough information to deduce the code distance d of our planar code. We have already essentially shown that an error Z_c lies in $N(\mathcal{S})\backslash\mathcal{S}$ iff c is a non-contractible cycle and that an error X_{c^*} lies in $N(\mathcal{S})\backslash\mathcal{S}$ iff c^* is a non-contractible cocycle. As dictated by the theory of CSS codes, writing m for the smallest (as number of edges) such cycle and n for the smallest such cocycle, the code distance is $\min\{n, m\}$. In an $N \times M$ grid, we have that $n = N$ and that $m = M + 2$, as depicted in Figure 2.11. It follows that our $N \times M$ grid planar code is a $[2NM + N - M, 1, \min\{N, M + 2\}]$ code.



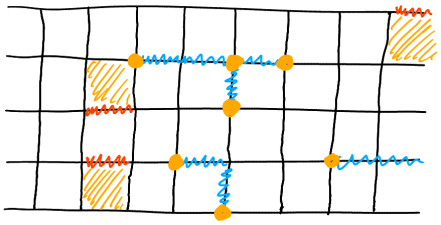
[Figure 2.11]

2.3 The Errors and their Correction

Consider a planar code that is initially in a stabilized state and then is subjected to an error operator $Z_{c_1} X_{c_2^*}$. In the Figure 2.12 (and all the ones that will follow), we shade in yellow the stabilizers that will have outcome -1 when measured, colour X operators by red and Z operators by blue.

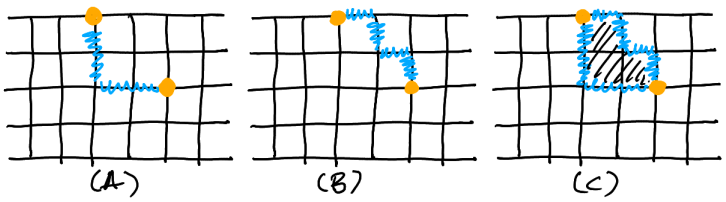
The idea is that for an error Z_c along a chain c, the stabilizers that will anticommute with Z_c are precisely those corresponding to vertices in the boundary of c. Similarly, for an error X_{c^*}, the stabilizers that will anticommute with it will correspond to the faces in the coboundary of c^*.

So syndromes of error chains correspond to their (co)boundaries. This might first seems as an issue, as for example, the following errors in Figure 2.13 A and Figure 2.13 B have the same syndrome. One would be justified to ask what would happen if the error in the first picture occurs and we apply a corrective Z-chain Z_{c^*} corresponding to the second picture. In that case,



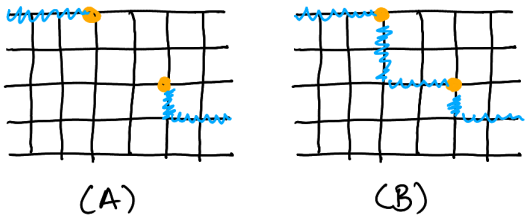
[Figure 2.12]

the overall outcome would be the one in the third picture, which, being a nullhomologous cycle, is the product of face stabilizers corresponding to the shaded faces, thus, acting trivially on the codespace. It is thus sufficient to find a chain with the given boundary that is homologous to the error that has occurred.



[Figure 2.13]

The issue would arise from applying a corrective chain as in Figure 2.14 (A), resulting in an overall operation as in Figure 2.14 (B), which is equivalent to applying a Z_L logical operator.



[Figure 2.14]

However, sticking to the mantra of stabilizer error correction, we note that the corrective operation in Figure 2.14 has higher weight than that in Figure 2.13, so in this case, the "correct" corrective Z-chain is the one we would have applied. Moreover, a hidden advantage of surface codes is that in general, errors tend to occur locally (over a topologically trivial area). Furthermore, unless happening near the (rough and soft) boundaries, errors can only occur in pairs (as an edge has two ends / is incident to two faces), so to find a minimum weight corrective chain, we usually use a minimum weight matching algorithm (a variation of Blossom's algorithm), which finds it in polynomial time. More exactly, given a syndrome where the stabilizers corresponding to the vertices $v_1, \ldots, v_n$ were measured as -1, we want to find pairs of vertices such that the sum of the lengths of the paths connecting the vertices is minimal, then apply corrective Z-chains between the vertices in each pair. The procedure for correcting X-errors is analogous, but is performed on the dual lattice.

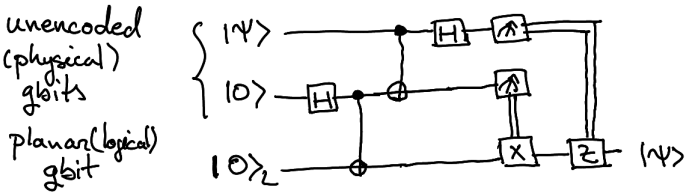
Before ending this section, it is worth mentioning that in any modern practical implementation

of a quantum error correction procedure, one wouldn't actually perform the corrective gates to ensure that the system remains in the codespace, because applying corrective gates can in turn introduce errors. Instead, one just tracks the error in software and adaps the logical gates and the interpretations of the measurements accordingly using Pauli frames, as described in [12]. We, however, chose to stick to using corrective gates for the sake of brevity and clarity of presentation. Moreover, it is worth noting that one can generally not trust the outcomes of the syndrome measurements as they can also be subject to noise. We ignore this issue as well for the same reasons.

2.4 Initializing a Planar Qbit

We first consider the problem of initializing a logical qbit in a planar code to $|0\rangle_L$, assuming that the physical qbits are in an initially arbitrary state. To this end, we first measure all the stabilizers and apply corresponding corrections to force the system into the codespace. We then measure Z_L to force the qbit to be either in state $|0\rangle_L$ or $|1\rangle_L$. If the measurement outcome is $+1$, we're done, otherwise we need to apply an X_L gate.

To initialize a qbit into an arbitrary $|\psi\rangle$ state, we use teleportation:



[Figure 2.15]

Note that we know how to implement logical Z and X gates on a planar qbit and implicitly, also how to implement CNOTs with encoded targets and classical/unencoded controls.

Finally, as we have used unencoded qbits, we cannot guarantee that the resulting encoded state is a faithful encoding of $|\psi\rangle$. To this end, we may use magic state distilation, as described in [6], which for brevity, we will not ellaborate on.

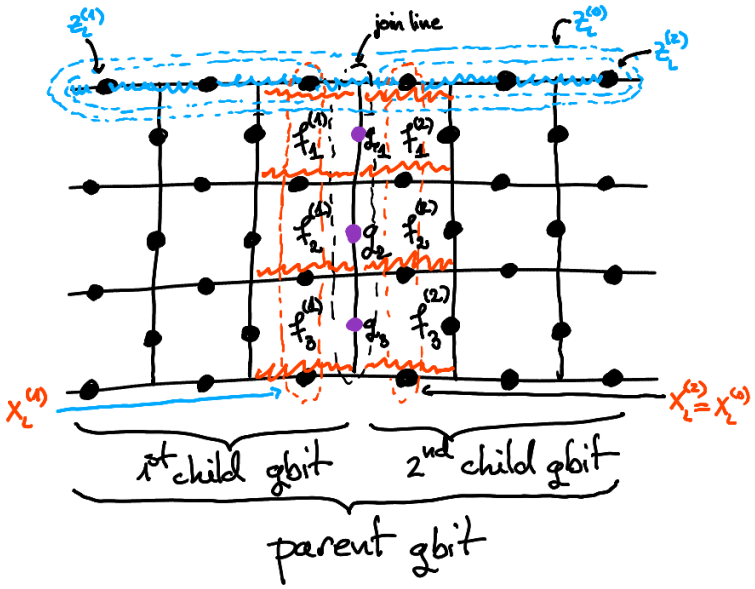
3 Lattice Surgery

God forgive me for my choice of stabilizer indexing.

In the previous section, we've covered how to encode individual qbits, how to initialize them and how to implement logical Pauli gates. In this section we discuss interactions between multiple qbits via lattice surgery. Lattice surgery consists of four operations: **rough splits**, **rough merges**, **soft splits** and **soft merges**. We will first discuss rough splits and merges, their soft counterparts being implemented analogously, and then discuss how to use these operations to implement arbitrary logical unitaries over a single logical qbit and how to implement a logical CNOT.

3.1 Rough Splits

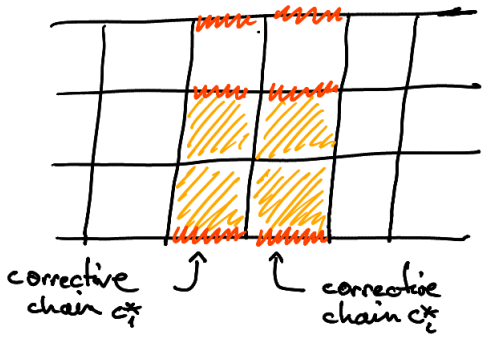
The rough split is an operation whose input is one logical qbit and whose output consists of two horizontally neighbouring qbits. We will describe the operation in the context of a concrete example:



[Figure 3.1]

The first step in the rough split is the measurement in the Z-basis of the purple qbits q_1, q_2, q_3. We call the original grid the parent qbit and denote the logical Pauli operators on the parent qbit by $X_L^{(0)}$ and $Z_L^{(0)}$, which we choose to be the ones depicted in Figure 3.1. After the measurement, we obtain two surfaces (on the left and right of the join line) corresponding to the two child qbits, with corresponding logical Pauli operators $X_L^{(0)}, Z_L^{(0)}, X_L^{(1)}, Z_L^{(1)}$, again, as depicted; moreover, we disregard (stop measuring and considering) the X stabilizers along the split. Let $\mathbf{Z}^{(0)}_{f_1^{(1)}}, \mathbf{Z}^{(0)}_{f_2^{(1)}}, \mathbf{Z}^{(0)}_{f_3^{(1)}}, \mathbf{Z}^{(0)}_{f_1^{(2)}}, \mathbf{Z}^{(0)}_{f_2^{(2)}}, \mathbf{Z}^{(0)}_{f_3^{(2)}}$ denote the (4 qbit) face stabilizers of the parent qbit, let $\mathbf{Z}^{(1)}_{f_1^{(1)}}, \mathbf{Z}^{(1)}_{f_2^{(1)}}, \mathbf{Z}^{(1)}_{f_3^{(1)}}$ denote the (3 qbit) face stabilizer of the first child qbit and $\mathbf{Z}^{(2)}_{f_1^{(2)}}, \mathbf{Z}^{(2)}_{f_2^{(2)}}, \mathbf{Z}^{(2)}_{f_3^{(2)}}$ denote the (3 qbit) face stabilizer of the second child qbit. Note that these face stabilizers are the only stabilizers of the children qbits which are affected by the measurement and thus the only ones that may result in a negative syndrome. Moreover, note that for every i, $\mathbf{Z}^{(0)}_{f_i^{(1)}} \mathbf{Z}^{(0)}_{f_i^{(2)}} = \mathbf{Z}^{(1)}_{f_i^{(1)}} \mathbf{Z}^{(2)}_{f_i^{(2)}}$, so given that the measurement commutes with both $\mathbf{Z}^{(0)}_{f_i^{(1)}} \mathbf{Z}^{(0)}_{f_i^{(2)}}$ and $\mathbf{Z}^{(1)}_{f_i^{(1)}} \mathbf{Z}^{(2)}_{f_i^{(2)}}$, assuming that the parent qbit was originally in the codespace, the measurement outcome of $\mathbf{Z}^{(1)}_{f_i^{(1)}} \mathbf{Z}^{(2)}_{f_i^{(2)}}$ must be $+1$, from which we may deduce that the measurement outcomes of $\mathbf{Z}^{(1)}_{f_i^{(1)}}$ and $\mathbf{Z}^{(2)}_{f_i^{(2)}}$ must be the same, for which we write $\mathbf{Z}^{(1)}_{f_i^{(1)}} \equiv \mathbf{Z}^{(2)}_{f_i^{(2)}}$. Perhaps, the motto for the previous observation should be, Z-syndromes always come in neighbouring pairs. Therefore, the error correcting X cochains we apply need be symmetrical (see for example Figure 3.2).

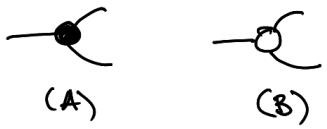
Finally, note that $X_L^{(0)} \equiv X_L^{(1)} \equiv X_L^{(2)}$ before applying the corrections and that measuring $Z_L^{(0)}$ before the split measurements and corrections can only have the same outcome as measuring $Z_L^{(1)} Z_L^{(2)}$ after the split, as the total correction operator applied $X_{c_1^*} X_{c_2^*}$ commutes with $Z_L^{(0)} =$



[Figure 3.2]

$Z_L^{(1)} Z_L^{(2)}$. The only mapping from the logical state of the parent qbit to the joint logical state of the children qbits which obeys these relations is the isometry $|+\rangle \mapsto |++\rangle$, $|-\rangle \mapsto |--\rangle$ - i.e. spider (A) in Figure 3.3.

The implementation of the soft split is analogous, but with the measurements being performed along an horizontal direction in the X-basis. The soft split corresponds to spider (B) in Figure 3.3.



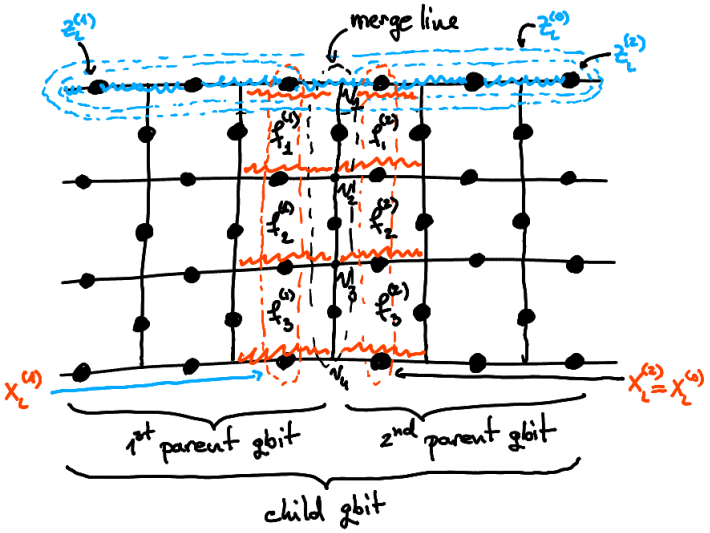
[Figure 3.3]

3.2 The Rough Merge

The rough join is an operation whose input consists of two neighbouring logical qbits and whose output consists of a single qbit and bit of classical information. Similarly to the split operation, the input qbits are named "parents" and the output qbit is named the "child". Consider two neighbouring planar qbits as in Figure 3.4:

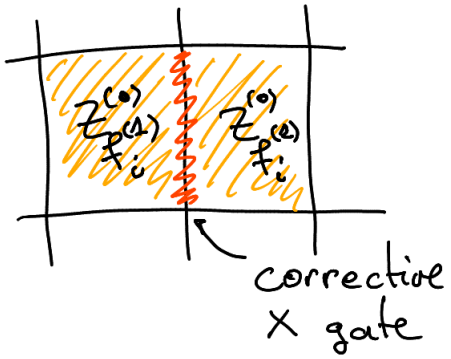
The first step in the merge process is initializing the purple qbits in the $|+\rangle$ state, then measuring the newly introduced X-stabilizers $\mathbf{X}_{v_1}, \dots \mathbf{X}_{v_4}$ obtaining outcomes $x_1, \dots, x_4 \in \{-1, +1\}$. We will write b for the product $b = x_1 x_2 x_3 x_4$ and note that performing the aforementioned stabilizer measurements is equivalent to measuring $X_L^{(1)} X_L^{(2)}$ (which would have outcome b if measured after).

Similarly to the split operation, we label the (3 qbit) face stabilizers of the parent qbits by $\mathbf{Z}^{(1)}_{f_1^{(1)}}, \mathbf{Z}^{(1)}_{f_2^{(1)}}, \mathbf{Z}^{(1)}_{f_3^{(1)}}$ and $\mathbf{Z}^{(2)}_{f_1^{(2)}}, \mathbf{Z}^{(2)}_{f_2^{(2)}}, \mathbf{Z}^{(2)}_{f_3^{(2)}}$ and label the (4 qbit) face stabilizers of the child qbit by $\mathbf{Z}^{(0)}_{f_1^{(2)}}, \mathbf{Z}^{(0)}_{f_2^{(2)}}, \mathbf{Z}^{(0)}_{f_3^{(2)}}, \mathbf{Z}^{(0)}_{f_1^{(2)}}, \mathbf{Z}^{(0)}_{f_2^{(2)}}, \mathbf{Z}^{(0)}_{f_3^{(2)}}$. Note that for each i, the product stabilizer $\mathbf{Z}^{(0)}_{f_i^{(1)}} \mathbf{Z}^{(0)}_{f_i^{(2)}} =$



[Figure 3.4]

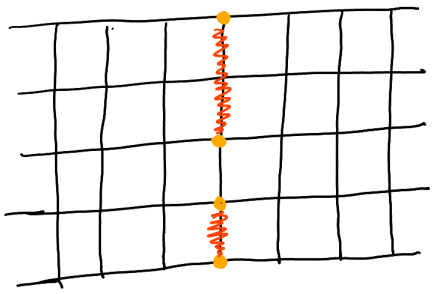
$\mathbf{Z}^{(1)}_{f_i^{(1)}}\mathbf{Z}^{(2)}_{f_i^{(2)}}$ commutes with the X-stabilizer measurements we've performed, and by an analogous argument to the split argument the syndromes of the face stabilizers along the merge line will come in pairs and we will remedy any such error detected by the $\mathbf{Z}^{(1)}_{f_i^{(1)}}$ and $\mathbf{Z}^{(2)}_{f_i^{(2)}}$ operators by applying an X gate to the edge in between the two faces. Note that the resulting corrective X-chain we apply will commute with all logical operators of both parent and children qbits $Z_L^{(i)}, X_L^{(i)}$ (for $i \in \{0,1,2\}$).



[Figure 3.5]

The merge operation now splits into two cases. If $b = +1$, we just connect pairs of X-stabilizers that measured -1 by corrective Z chains until fixing all of them. If we choose these Z chains so that they are contained in the merge line, we obtain a correction chain that again commutes with all logical operators $Z_L^{(i)}, X_L^{(i)}$ (for $i \in \{0,1,2\}$).

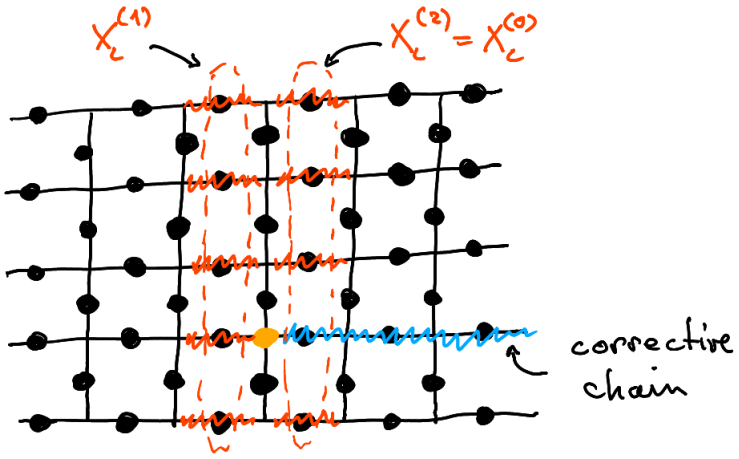
The overall effect of the operation in this case is the following: performing the measurement with outcome $b = 1$ (the measurement being equivalent to measuring $X_L^{(1)}X_L^{(2)}$) forces the joint system of the two parent qbits to be in a state in the subspace spanned by $\{|{+}{+}\rangle, |{-}{-}\rangle\}$. As $Z_L^{(1)}Z_L^{(2)} = Z_L^{(0)}$ and $X_L^{(0)} = X_L^{(1)} (\equiv X_L^{(2)})$, we note that the only mapping from the space of



[Figure 3.6]

possible states of the two qbits to those of the parent qbit is the unitary given by $|{+}{+}\rangle \mapsto |{+}\rangle$, $|{-}{-}\rangle \mapsto |{-}\rangle$ - i.e. spider (A) in Figure 3.8.

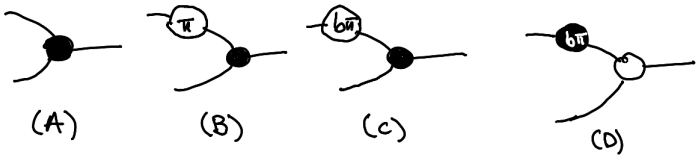
If $b = -1$, we proceed in the same way as before, by pairing up X-stabilizers measured as -1 until this time (because of the odd parity of the number of such stabilizers), we end up with a single X-stabilizer measuring -1, which we may only correct by connecting it to one of the rough boundaries via a corrective X-chain. Suppose that we connect it to the right boundary as indicated in Figure 3.7.



[Figure 3.7]

Note that the corrective chain anticommutes with the logical operators $X_L^{(0)}, X_L^{(2)}$ and commutes with all others. By an argument analogous to the one in the $b = +1$ case, we have that the initial state of the parent qbits is forced to lie in the space spanned by $|{+}{-}\rangle$ and $|{-}{+}\rangle$ and the mapping from the space of possible states of the parent qbits to that of the children qbit is given by the unitary mapping $|{+}{-}\rangle \mapsto |{+}\rangle$ and $|{-}{+}\rangle \mapsto |{-}\rangle$ - i.e. spider (B) in figure 3.8. Putting it all together, the merge operation corresponds to the (branched) spider (C) in Figure 3.8, where b is the outcome of the measurements.

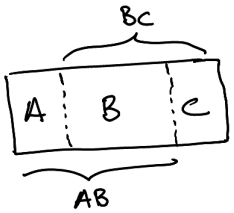
Finally, the soft merge operation corresponds to spider (D) in Figure 3.8 and is implemented analogously, acting on two vertically neighbouring qbits and by measuring newly introduced Z (face) stabilizers between them.



[Figure 3.8]

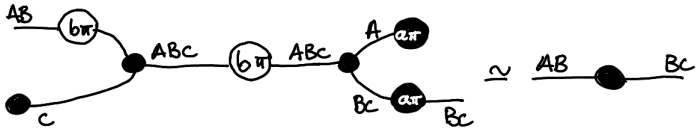
3.3 Moving Logical Qbits around the Grid

In this section, we discuss moving around the grid which is our quantum computer. Consider three neighbouring rectangular regions in the grid as in Figure 3.9, where each region corresponds to a logical qbit.



[Figure 3.9]

Suppose that a logical qbit inhabits the region AB and that we want to move it slightly to the right into the region BC. To this end, we initialize the logical qbit in the region C to $|0\rangle_L$; we apply a merge operation measuring outcome b and then a logical Z operation to the resulting qbit in the region ABC; we then split along the edge between A and BC and finally, we measure the qbit in region A in the Z basis obtaining outcome a, to then apply a corrective gate to BC if $a = 1$. This is best explained diagrammatically:



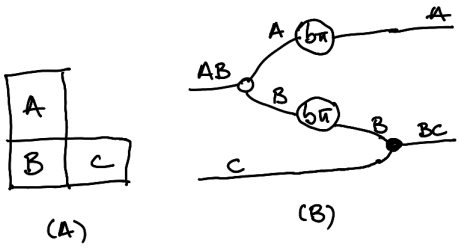
[Figure 3.10]

We may analogously move logical qbits vertically via soft merge and split operations. Similarly, we may "shrink" or "enlarge" a logical qbits (by performing half of the move operation operation).

3.4 Logical CNOTs via lattice surgery

Consider two logical qbits arranged as in Figure 3.11 (A) below, one living in the region AB and one in region C. To implement a CNOT gate we perform the following operations: first, we soft-split the AB logical qbit into qbits corresponding to the regions A and B. We then perform

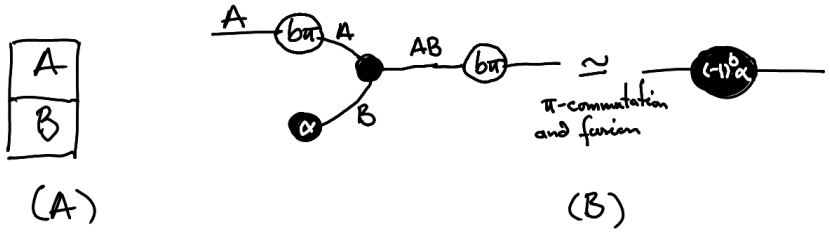
a rough merge between the qbits in regions B and C with measurement outcome b; if $b = 1$ we apply a corrective Z gate on the qbit in the region A. Again, it's probably more productive to to look at the diagram in Figure 3.12 (B) in order to understand the procedure. One can check that the outcome of this operation is a pair of qbits in regions A and BC whose joint state equals that resulting from applying a CNOT to the original qbits from the regions AB and C.



[Figure 3.11]

3.5 Logical Single Qbit Unitaries

To implement an arbitrary unitary gate over a logical qbit, it suffices to show how to implement arbitrary phase gates thanks to the Euler decomposition of matrices in $SU(2)$. Consider the following procedure, involving two qbits in the neighbouring regions A and B as depicted in Figure 3.12 (A).



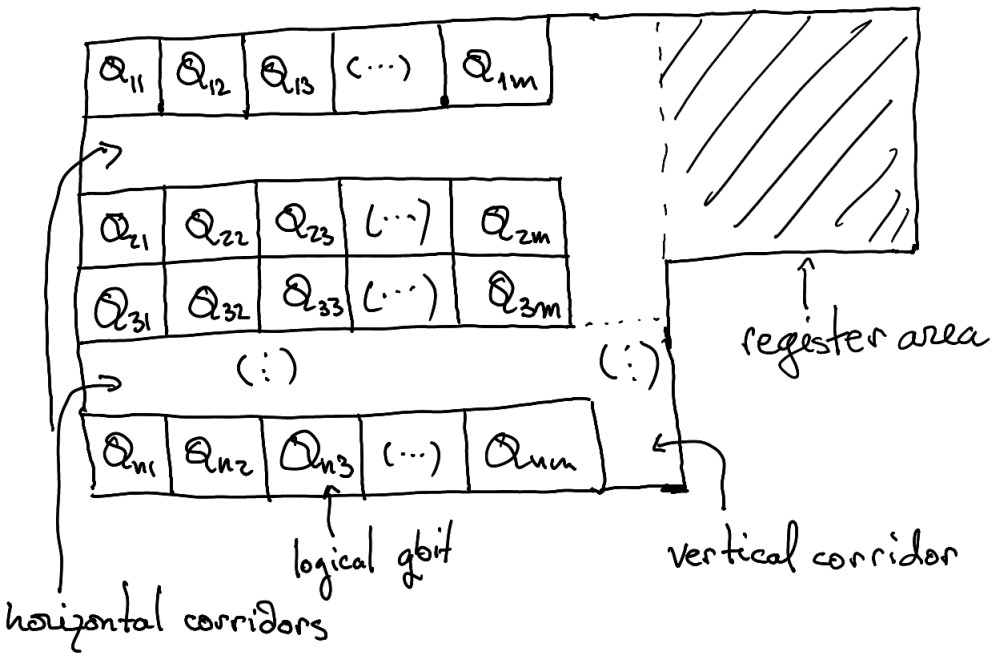
[Figure 3.12]

We initialze the qbit in B in the state $1/\sqrt{2}(|+\rangle + e^{i\alpha}|-\rangle)$, measuring the outcome b. If we measure $b = 1$, we apply a corrective logical Z in the resulting qit and by the diagram equality in Figure 3.12 (B), we have that the logical effect of our operation is the application of a $X[(-1)^b\alpha]$ phase gate to our original qbit. If we measure $b = 0$ we're done, otherwise, we repeat applying the procedure until the overall applied phase becomes α, which statistically should happen after at most three steps in most cases, by a straightforward probabilistic argument. $Z[\alpha]$ phase gates are implemented analogously (just flip the colours in the diagram).

4 Putting it all Together

We are now ready to describe a procedure that transforms a quantum circuit into a sequence of lattice surgery operations that implements its logic in a fault tolerant manner. First, we arrange

the logical qbits in a sequence of rows and corridors between them on the grid of physical qbits which is our quantum computer, as in Figure 4.1.



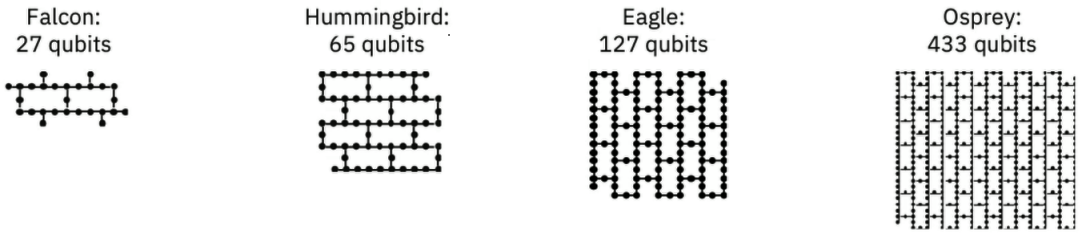
[Figure 4.1]

For each gate U in the input circuit, in topological order acting on one or two qbits, we move the involved qbits in the register area via movements (as described in section 3.3.) via the corridors. In the register area, we then perform the corresponding logical operations (as described in sections 3.4 and 3.5) and then return the qbits to their original positions.

We end this paper by mentioning the following things:

- An implicit lie present in most introductory courses in quantum information is that of full connectivity. In the circuit model, we may apply a CNOT gate, for example, with no restriction on what the control qbit and what the target qbit may be. This, unfortunately, is generally not the case and cannot be the case in any envisionable version of scalable quantum computing (superconducting qbit arrays, networked ion traps etc.) without moving the qbits next to each other by very time consuming and error prone swap operations. For example, in the hardware produced by IBM, only the the qbits seen as neighbouring in Figure 4.2 may interact, and only via controlled Z gates. It is an advantage of the surface code that all operations on the physical qbits occur locally, not necessitating the otherwise extremely expensive operation of moving a physical qbit far around the grid to perform an interaction.
- As mentioned before, errors in a quantum error correction code tend to occur locally and thus in a topologically trivial area of the qbit. In this regard, the code distance of our surface code hides its great advantage, namely that in practice it is more error resilient than other codes with larger distance, while also allowing for fault tolerant universal computation.

- Quantum error correction is a field of quantum computation where one can easily see the expressive limitations of the circuit model. For example, the merge and split operations in lattice surgery cannot be easily expressed in the circuit model due to their non-unitarity, and while in that case, the ZX-calculus somewhat comes to the rescue, it still does not have the expressive power to convey the dynamic nature of errors occurring over a time interval or the dynamic and adaptive operations of syndrome measurement and error remediation. These are all issues of the software that controls the quantum hardware and thus, in the author's opinion, fall beyond any shadow of a doubt in the field of quantum software. To remedy this apparent hole in the literature, I would say that formal languages from the theory of classical parallel computing will soon need adapting to the quantum realm.



[Figure 4.2] Credits to IBM Research

5 Exam Stuff

This section has the sole purpose of addressing the questions posed in the exam paper.

(a) The first question is mostly addressed in the abstract and the first section. I do slightly fear that the topic of my answer does slightly run against the spirit of this course, which is about programming on NISQ devices, given that large scale fault tolerant computations protocols as described in section 4 are almost by definition what will end the NISQ era. To that end, I would say that the procedure I've described, applied for only a few logical qbits (few enough to fit in hardware forseseable in the next few years if we are to trust in the feasibility of the IBM roadmap) may be applied for fault-tolerant quantum cryptography, where one need not use that many qbits at a time.

(b) This question is addressed in the entirety of the paper.

(c) This question is addressed in the abstract.

(d) The procedure always works, in the sense that it will always produce a series of logical operations equivalent to the ones described by the input circuit.

(e) The performance of the routine in terms of error correction is touched upon when mentioning the code distance for a single encoded qbit in section 2.2 and in section 4. Due to the high number of qbits involved it is impossible to run the procedure on the currently publicly available hardware (though an adaptation of the procedure to the hexagonal grid structure of the IBM Condor, which is not publicly available for use, might be possible) or to simulate it. I have to admit that the omission of a threshold computation for the code in this paper is indeed borderline criminal, as it would have served as the most convincing argument for its efficiency; unfortunately, I found it utterly impossible familiarise myself with both the error correction schemes and learn enough statistical mechanics to understand how to apply the

ising model to a qbit grid in the short time I've had available to submit my exam sheet. Google, however, gave convincing theoretical and experimental arguments that topological quantum error correction works well in [1].

We now make the case for originality, as required:

- Section 2 mostly follows the lines of [4] and [7]. We do however, provide a significantly more rigorous exposition of the mathematical formalism. For example, we do justify the use of relative homology groups, find bases for the codespace, spell out the formal connection between the Lefschetz duality (which somehow, I think is new to the literature) and the dual lattice and provide a geometrical interpretation of states in a planar code. On the non-formal front, we do also explicitly describe initializing a logical qbit in an arbitrary state.
- In section 3, we mostly follow [8, 10], to which we add the following the operation of moving a qbit (note for fairness: I heard that one can move a logical qbit around the grid in a keynote on youtube that I can no longer find, but I did figure out how to implement such a move by myself), we describe how to introduce arbitrary logical single qbit unitaries rather than just T gates and we explicitly describe what the syndrome correction gates are.
- The protocol presented in Section 4 is pretty much fully original to the best of my knowledge.

I guess that because in my enthusiastic naivety I thought that it'd be a good idea to merge teaching myself quantum error correction and doing this exam, I've learnt the very hard way why people do entire doctoral degrees on this single topic.

References

[1] Suppressing quantum errors by scaling a surface code logical qubit. *Nature*, 614(7949):676–681, 2023.

[2] A Kay C Macchiavello A Ekert, T Hosgood. Introduction to quantum information science. Unpublished book available at https://www.qbit.guide/.

[3] John van de Wetering Aleks Kissinger. Picturing quantum software.

[4] Héctor Bombín. An introduction to topological quantum codes. *arXiv preprint arXiv:1311.0277*, 2013.

[5] Hector Bombin and Miguel A Martin-Delgado. Homological error correction: Classical and quantum codes. *Journal of mathematical physics*, 48(5), 2007.

[6] Sergey Bravyi and Alexei Kitaev. Universal quantum computation with ideal clifford gates and noisy ancillas. *Physical Review A*, 71(2):022316, 2005.

[7] Sergey B Bravyi and A Yu Kitaev. Quantum codes on a lattice with boundary. *arXiv preprint quant-ph/9811052*, 1998.

[8] Niel de Beaudrap and Dominic Horsman. The zx calculus is a language for surface code lattice surgery. *Quantum*, 4:218, 2020.

[9] Allen Hatcher. *Algebraic topology*. Cambridge University Press, Cambridge, 2002.

[10] Dominic Horsman, Austin G Fowler, Simon Devitt, and Rodney Van Meter. Surface code quantum computing by lattice surgery. *New Journal of Physics*, 14(12):123011, 2012.

[11] A Yu Kitaev. Quantum error correction with imperfect gates. In *Quantum communication, computing, and measurement*, pages 181–188. Springer, 1997.

[12] E. Knill. Quantum computing with realistically noisy devices. *Nature*, 434(7029):39–44, March 2005. URL: http://dx.doi.org/10.1038/nature03350, doi:10.1038/nature03350.

[13] Peter W Shor. Scheme for reducing decoherence in quantum computer memory. *Physical review A*, 52(4):R2493, 1995.