DELFT UNIVERSITY OF TECHNOLOGY

MASTER THESIS APPLIED MATHEMATICS

Optimising logical measurements for the Toric QEC code

Author:

SEAN ANTHONIUS JOHANNES HENDRICUS CAMPS

To obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on the 12th of May 2026.

Student number:	4958241	
Project duration:	December 16, 2024 – May 12, 2026	
Thesis committee:	Dion Gijswijt	TU Delft
	Barbara M. Terhal,	TU Delft, supervisor
	Bas Janssens,	TU Delft

Contents

ACKNOWLEDGEMENTS

Completing this thesis would not have been possible if it was not for the support of the many people around me, and I would like to take this opportunity to express my sincere gratitude to all of you.

First and foremost, I want to thank my family, and in particular my parents, for their unwavering support, patience and compassion throughout my studies. Your faith in me, even in the moments when I doubted myself, has meant more than I can express.

I am also deeply grateful to all my friends from Krashna Musika, de Bolk, and anyone else who I have had the pleasure of meeting in all this time at this university. The music, beers, and listening ears you provided were always there to pick me up when I needed it most, and I could not have made it through without all of you.

I would also like to thank all the staff that has educated me over the last years. Specifically the board of examiners that has had to put up with my unending stream of emails about the double masters degree. I also owe a special debt of gratitude to my supervisor Barbara Terhal, whose expertise and no nonsense approach shaped this thesis in fundamental ways, and to MacKenzie Shaw, whose support and weekly meetings finally got me through most of this project. I thank you both for your patience, your insightful feedback, and your faith in my ability to see this project through.

ABSTRACT

In this thesis we numerically investigate the performance of logical measurements on the toric quantum error correcting code using the ancilla construction of Cohen et al.[1], both with and without the morphing circuit design technique. We simulate the measurement of the logical Z_1 observable on the toric code under circuit-level noise using the STIM package, exploring the effects of the number of ancilla layers and QEC rounds on the logical measurement and idle error rates.

We find that for the toric code, a single dual layer performs best and that additional layers only increase the measurement error rate without improving the idle error rate. Regarding the number of QEC rounds, we find that performing at least as many rounds as the code distance is necessary to ensure the measurement error rate scales with the code distance rather than the weight of low-order measurement errors over different QEC rounds. Beyond this threshold, additional rounds improve the measurement error rate at the cost of a higher idle error rate, presenting a tunable trade-off.

Finally, we compare the morphing and standard versions of the circuit. Despite the morphing version using half as many physical qubits, it achieves better logical measurement error rates and similar idle error rates relative to the standard circuit, leading us to conclude that the morphing version outperforms its standard counterpart.

1

INTRODUCTION

1.1. MOTIVATION

The list of problems that can be solved more efficiently with a quantum computer than with a classical computer keeps growing with examples in cryptography [2], chemistry [3], machine learning [4], to name a few. The challenge of realising a quantum computer to perform these algorithms, however, still remains one to be solved [5]. The inherent susceptibility of quantum states to noise remains one of the central issues. However through the use of quantum error correction (QEC) it is possible to take multiple noisy qubits and combine them into fewer logical qubits with less noise. It is even possible to run any quantum algorithm with arbitrary precision at the cost of increasing the total qubits and amount of timesteps thanks to the threshold theorem [6].

The threshold theorem does however come with the constraint that the starting physical qubits already need to have a noise-level below a certain threshold. This threshold depends on what QEC code is used [6]. Constructing physical qubits that meet this threshold is not trivial and the list of QEC codes that have been successfully implemented is still on the short side. A QEC code for which this threshold already has been achieved is the surface code [7].

Besides its threshold, the surface code has some other nice properties. It requires each physical qubit to participate in four parity checks, and each parity check to only check four qubits [8]. The surface code can also be embedded in a two dimensional surface. Additionally, some crucial two qubit interactions, like logical parity measurement and logical CNOT's between different logical qubits encoded across different surface codes are possible through a technique called lattice surgery [9]. However, one impractical aspect of the surface code is the fact that each encoded logical qubit requires a total number of physical qubits that scales with the square of the number of errors it can correct. The BPT bound [10] shows that this holds for all QEC codes that can be embedded in two dimensions.

Another family of QEC codes that shows great promise is the family of Bivariate Bicycle (BB) codes [11]. Its threshold is close to that of the surface code. It does however require the physical qubits to partake in a total of six parity checks and each parity check to

work with six qubits. It does however break the BPT bound and therefore achieve the same number of logical qubits with the same number of errors corrected, using fewer physical qubits. This efficiency comes at the cost of not being able to embed it in two dimensions, but instead requires two separate planes of connectivity. This property is referred to as biplanarity.

The recent paper by Shaw and Terhal [12] further explored the capabilities of the BB codes by combining it with the code design technique they call *morphing*. Morphing had previously been used to lower the connectivity requirements of the surface code [13]. With it Shaw and Terhal managed to lower the connectivity requirement of each qubit in the BB codes from six down to five whilst maintaining similar numerical performance, and consequently making the embedding task easier.

The logical qubits in different BB codes cannot directly be operated on together, but in Ref. [11] Bravyi *et al.* take the ancilla structure from Cohen *et al.* [1] to show that the logical state of the BB code can be teleported to a surface code for computational purposes, thereby showing that the BB codes could function as a memory element in a quantum computer with computation done using the surface code. More recently in Ref. [14] it has also been shown that different instances of BB codes can be connected with each other using code surgery. This result has also been used in Ref. [15] for the construction of a quantum computing architecture based on the BB codes.

Shaw and Terhal also showed in their paper in appendix F [12] how to combine both the ancilla structure from Cohen *et al.* [1] and the morphing design technique. This again allows for the logical qubits to be measured and teleported to a surface code for computation while still preserving the reduction in the connectivity requirements of the QEC code thanks to morphing. The question whether this application of the morphing technique maintains the logical performance compared to the version without morphing, is still left open.

The construction by Cohen *et al.* [1] can be applied to measure any one or two qubit Pauli observable on low-density parity check (LDPC) QEC code, making it widely applicable. By theorem 1[1] the ancilla construction has been shown to maintain the code distance when total size is scaled with the square of the distance. This is however a sufficient condition, meaning that there is a possibility that fewer layers might also suffice. Cohen *et al.* also suggest that the number of rounds of QEC needed to perform the measurement should at least equal the code distance. The exact effects of both these parameters depend on the underlying code and can be explored more extensively with simulation.

One simpler code family that is closely related to the BB codes, is the toric code. The toric code has a comparable rate to the surface code [16] and encodes an additional qubit compared to the surface code. Furthermore, the morphing technique from [13] can also be used on this code family, making it an excellent starting option for investigation into the effects of morphing with the construction from Cohen *et al.* [1].

In this thesis we will numerically investigate the performance of the toric code with the ancilla structure from Cohen *et al.* [1] both with and without morphing. With these nu-

merical results we will explore the effects of layers and rounds in the ancilla construction and compare the performance of the morphing version with the standard version.

1.2. QUANTUM ERROR CORRECTION

When you leave the ideal world of quantum information theory for the messy world that is our reality, one quickly finds out that perfect qubits, gates, and measurements are indeed just idealised abstractions. During experiments involving quantum mechanics, no matter what technology, there are plenty of things that can go wrong: Photons go missing, pure states decohere into mixed states, etc. [17]. In this section we will give a brief introduction to one way to correct these mistakes, namely quantum error correction. A basic familiarity with quantum information and computation is assumed and some elementary knowledge of (classical) error correction is helpful. For a more complete introduction we can highly recommend "Surviving as a Quantum Computer in a Classical World" by Daniel Gottesman [18], which at time of writing has not been finished but already proves to be an invaluable resource.

1.2.1. STABILISER CODES

In this subsection we will introduce the general concepts needed for QEC and build up to the full definition and capabilities of stabiliser codes. We will start defining some basic errors that a quantum computing system can undergo. Then we will explain how to perform checks on the quantum information in the system using stabilisers to find these errors. Afterwards we will combine multiple of these stabilisers to form a stabiliser code, followed directly by some examples. Then we will show how to correct the errors and introduce the notion of distance and the $[[n, k, d]]$ notation. With the elementary picture of QEC completed we will briefly talk about logical operators and logical measurements on the newly constructed code space. Lastly we will introduce the Toric code with its code space and operators.

In classical error correction, the most common errors are characterised by bit-flips. These bit-flips, as the name suggest, flip the values of 0's to 1's and vice versa on particular locations. In the quantum world these errors also occur but because the state can consist of a superposition of several values, they are instead described by a Pauli operator acting on the whole superposition.

Definition 1.2.1 (Bit-flip error)**.** A bit-flip error on position i on the space $\mathbb{C}^{2^n}$ is the Pauli string: $I_1 \otimes \cdots \otimes X_i \otimes \cdots \otimes I_n$

The phase of the state in quantum computing also holds valuable information. It would therefore be wise to also consider errors to the phase of the state. In a similar fashion to the bit-flip, there exists the phase-flip.

Definition 1.2.2 (Phase-flip error)**.** A phase-flip error on position i on the state space $\mathbb{C}^{2^n}$ is the Pauli string: $I_1 \otimes \cdots \otimes Z_i \otimes \cdots \otimes I_n$

When both of these errors happen after each other on the same position, we get the $XZ = iY$ Pauli operator, which up to global phase is equivalent to the Y Pauli operator. More generally when considering errors that consist of multiple phase-flips and bit-flips on several locations one ends up with the full group of Pauli strings, which are referred

to as Pauli errors.

Definition 1.2.3 (Pauli strings)**.** A Pauli string P on the state space $\mathbb{C}^{2^n}$ is a tensor product consisting of n elements of $\{I, X, Y, Z\}$ multiplied by a global phase $c \in \{1, i, -1, -i\}$. Its weight $w(E)$ is equal to the amount of non-identity Pauli elements in its product. The group of Pauli strings with matrix multiplication is denoted with **P**. To shorten notation we will usually write down only the non-identity elements and at which position they act e.g. $iX_1Z_3Y_4$ to mean $i \cdot X \otimes I \otimes Z \otimes Y \otimes I \otimes \cdots \otimes I$.

With some basic errors defined, we will now introduce the way we check for errors. In classical error correction bit-flips are detected by parity checks. By either adding or removing a one, the bit-flip changes the parity from even to odd, which is detected by parity checks working on that particular location. Analogously one can also detect bit-flips in quantum information by also checking the parity. This check is done by measuring an observable that is a Pauli string consisting of Z and I. For the phase-flips one can also perform parity checks by instead measuring observables that is a Pauli string consisting of X and I. More generally one can take any Pauli string as an observable to perform a parity check that is sensitive to a mix of bit-flips and/or phase flips.

Sadly, one cannot simply take a set of checks and call it a day. A bit more work is needed to combine them into a full QEC code. First of all it is preferable if the checks are measurable without affecting one another. This directly implies that all the checks should commute. Secondly, there should also be states for which all checks pass simultaneously. By treating our set of checks as the generators for a group, we get all the possible checks we could extract. The existence of a state, which passes all the checks, is equivalent to checking whether $-I$ is not in the group it generates. All together this leads to the following definition.

Definition 1.2.4 (stabiliser code)**.** A stabiliser code is defined by a commutative subgroup **S** of **P** that does not contain $-I$, together with a subspace called the code space $C(\mathbf{S}) = \{|\phi\rangle \in \mathbb{C}^{2^n} : \forall S \in \mathbf{S}, S|\phi\rangle = |\phi\rangle\}$. Independent generating sets are usually denoted by **G**. Elements of the **S** are called stabilisers. When measuring a stabiliser an output of 1 is considered a pass and -1 is considered a fail. In particular if there is an independent generating set, where each stabiliser consists of either X or Z Pauli observables, then it is called a Calderbank-Shor-Steane (CSS) code.

Because not all technologies allow for observables consisting of multiple Pauli operators, most of stabiliser are indirectly measured through the help of ancilla qubit. The general idea is to use CNOT's to entangle the state of the ancilla qubit with the outcome of the stabiliser. In figure 1.1 one can see two example circuits measuring Z_1Z_2 and X_1X_2. One can check that the ancilla's outcome indeed corresponds to the observable by calculating the results for the states consisting of $|0\rangle$ and $|1\rangle$ for Z_1Z_2 and $|+\rangle$ and $|-\rangle$ for X_1X_2. Support on more qubits can be added by performing more CNOT's with the desired qubits.

Now with initial definitions out of the way, let's have a look at a basic example. Taking $\mathbf{G} = \{Z_1Z_2, Z_2Z_3\}$ as the generators of the subgroup. We find that Z_1Z_2 forces the first and second qubit to both be 0 or 1 and Z_2Z_3 forces the second and third qubit to both be 0 or 1. Thus the code space is spanned by $C(\mathbf{S}) = \{|000\rangle, |111\rangle\}$. Those familiar with classical error correction might have already recognised this as the repetition code. Now if the

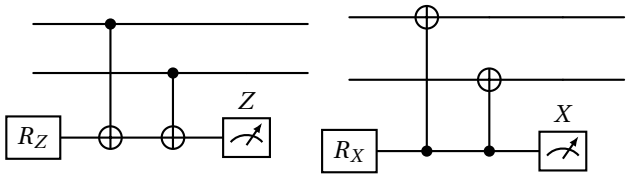


Figure 1.1: Two circuits showing the indirect measurement of stabilisers on two qubits. In both circuits the third qubit is the ancilla qubit that is used to perform the measurement. The circuit on the left measures Z_1Z_2 and the circuit on the right X_1X_2.

bit-flip X_1 happens to the state $|\phi\rangle \in C(\mathbf{S})$, then we find the following

$$Z_1Z_2 \cdot X_1 |\phi\rangle = -X_1 \cdot Z_1Z_2 |\phi\rangle = -X_1 |\phi\rangle . \tag{1.1}$$

This shows that $X_1\phi$ is an eigenvector with eigenvalue -1 of the stabiliser Z_1Z_2, and that when the stabilisers get measured, the stabiliser Z_1Z_2 will give outcome -1 which is a fail. Because the error X_1 always makes one of the stabilisers fail, it is considered detectable. It turns out that for the errors X_2 and X_3 a similar story holds. When looking at Z_1 however, we see that it commutes with the stabilisers and therefore causes none of them to fail. This consequently means that the state is not protected from phase-flips. Because in general all Pauli strings either commute or anticommute, all Pauli errors are deterministically either detected or undetected by the stabilisers.

Definition 1.2.5 (detectability)**.** A Pauli error $E \in \mathbf{P}$ is detectable by a stabiliser code $\mathbf{S}$, if $\exists S \in \mathbf{S}$ which anticommutes with E. The set of undetectable errors is given by $N(\mathbf{S}) = \{P \in \mathbf{P} : \forall S \in \mathbf{S}, PSP^\dagger \in \mathbf{S}\}$, where N stands for the normaliser of the group, and the set of detectable errors is the remaining errors usually given by $\mathbf{P} \setminus N(\mathbf{S})$.

In the previous example we saw that only bit-flips could be detected, because the stabilisers consisted only of Z Paulis which commute with phase-flips. To correct this, one can take the encoded qubits and encode them again with a repetition code but this time protecting the phase. This gives rise to the stabiliser code with $\mathbf{G} = \{Z_1Z_2, Z_2Z_3, Z_4Z_5, Z_5Z_6, Z_7Z_8, Z_8Z_9, X_1\cdots X_6, X_4\cdots X_9\}$ with the code space spanned by $\{\frac{1}{2\sqrt{2}}(|000\rangle+|111\rangle)\otimes(|000\rangle+|111\rangle)\otimes(|000\rangle+|111\rangle), \frac{1}{2\sqrt{2}}(|000\rangle-|111\rangle)\otimes(|000\rangle-|111\rangle)\otimes(|000\rangle-|111\rangle)\}$. This codes detects or includes as stabiliser all weight 1 and 2 Pauli errors and was one of the first QEC codes discovered by Shor in 1995 [19].

So far we have only talked about detecting errors, but they also need to be corrected. When the stabilisers are measured, a list of successful and failed checks gets produced and this list is referred to as the syndrome in the error correction terminology.

Definition 1.2.6 (Syndrome function)**.** The error syndrome function is a function $\sigma : \mathbf{P} \to \mathbb{Z}_2^{|\mathbf{G}|}$ such that $\forall E \in \mathbf{P}, \forall S \in \mathbf{G}$:

$$\sigma(E)_S = \begin{cases} 0 & E \text{ and } S \text{ commute} \\ 1 & E \text{ and } S \text{ anti-commute} \end{cases}$$

The syndrome of a stabiliser code has some useful properties, derived from the commutations relations of the Pauli strings.

Proposition 1.2.1. *For a stabiliser code the syndrome function σ is a group homomorphism, meaning $\forall E, F \in \mathbf{E}$:*

$$\sigma(EF) = \sigma(E) + \sigma(F) \ mod\, 2.$$

For each syndrome, the set of errors that could have caused this is syndrome is equal to a coset of the kernel of the syndrome function. In this case the kernel is $N(\mathbf{S})$. To correct the state one needs to decide which of the errors in the coset is the most likely culprit. It is sensible to assume that bit-flips and/or phase-flips are uncommon events, and therefore Pauli errors with a lower weight are more likely to have happened than higher weight Pauli errors. So for each coset of $\mathbf{S}$ the most likely culprit is the error with the least weight. This strategy is called minimum weight decoding.

Minimum weight decoding can however be computationally expensive depending on the underlying code. In those cases other decoders are available such as belief propagation [20].

Going back to the repetition code one could find the following likely errors:

$$(0,0) \rightarrow I \tag{1.2}$$
$$(1,0) \rightarrow X_1 \tag{1.3}$$
$$(0,1) \rightarrow X_3 \tag{1.4}$$
$$(1,1) \rightarrow X_2 \tag{1.5}$$

We could have just as easily taken the Y Pauli errors as culprits here, under the assumption that they are equally likely to X Pauli errors. Taking the X Pauli errors as the culprits does however mean that if Y_1 were to happen and we correct for X_1 by applying it again to correct the state, we would be left with the state

$$X_1 Y_1 \left|\phi\right\rangle = i Z_1 \left|\phi\right\rangle \equiv Z_1 \left|\phi\right\rangle. \tag{1.6}$$

So in this case the state is erroneously corrected and accidentally was changed by an operation. If we were to later measure the code in the X basis, then the measurement would give the opposite result. This is of course undesirable, so it is good to know which errors are corrected. This is sadly computationally quite intensive for larger codes. There are however some assurances for which errors are correctable, using the following property of our QEC code.

Definition 1.2.7 (Code distance)**.** The distance d of a stabiliser code is given by

$$\min_{E \in N(\mathbf{S}) \setminus \mathbf{S}} w(E).$$

For the repetition code, we see that $d = 1$ as Z_1 is undetectable and not a stabiliser. For the Shor code we find that $d = 3$ with $X_1 X_2 X_3$ as an example of an undetected error. The distance is a useful measure for seeing which class of errors are corrected, as can be seen by the next proposition.

Proposition 1.2.2. *A stabiliser code* **S** *with distance d correcting using minimum-weight decoding will correctly correct all errors E with*

$$wt(E) \leq \frac{d-1}{2}.$$

Although the distance of a quantum error correcting code does not give a complete picture of what errors are corrected, it does provide a clear lower bound on the order at which errors can be expected. With this lower bound it becomes easier to quickly compare QEC codes with one another. Together with the total number of qubits used n and the number of qubits encoded $k = \log_2(\dim(C(\mathbf{S}))) = n - |\mathbf{G}|$, the qualities of a stabiliser code are denoted as $[[n,k,d]]$. When working with a family of QEC codes $[[n,k,d]]$ is usually used to refer to a specific member of that family.

Now that the basics of finding and correcting errors are clear, the next step is to assign meaning to the code space. The choice of basis takes the form of an isomorphism between the encoded state space $\mathbb{C}^{2^k}$ and the physical state space $C(\mathbf{S})$. In this thesis we will denote encoded states with a bar, e.g. $|\overline{0}\rangle$, and the underlying physical state without a bar.

For some codes there is an obvious choice in isomorphism. The repetition code has an easy one namely:

$$\alpha|\overline{0}\rangle + \beta|\overline{1}\rangle \rightarrow \alpha|000\rangle + \beta|111\rangle.$$

In other cases like the Shor code the choice of logical basis becomes a lot less clear. One method for CSS codes, that will be used in this thesis, is to take the all-zeroes state in the physical state space and to perform a syndrome measurement on it. This does not guarantee that the stabilisers featuring Z will pass, but the state will collapse into one of the eigenspaces of those stabilisers featuring Z through the measurement. In the following rounds the state remains in that eigenspace if no noise occurred. This does mean that if the state collapsed into the -1 eigenspace of the stabiliser then it will keep failing. It is however possible to change our perspective and replace the failing stabiliser $S \in \mathbf{G}$ with $-S$. This will give rise to a new stabiliser code for which the current state passes all stabilisers and that has the same distance and $N(\mathbf{S})$. The current state can then be taken as the all-zeroes state of the encoded space. The other encoded states are left undefined. This method also serves as a way to prepare the code in the logical all-zeroes state.

With an isomorphism between the physical state space and the logical code space it also becomes possible to see how operators and observables from the physical space act on the encoded space. This is only well-behaved if the operator A maps the state space to itself, meaning it is invariant under the code space $C(\mathbf{S})$. For stabiliser codes and a Pauli operator A this is equivalent to $A \in N(\mathbf{S})$. For an observable A it is also must be able to be measured simultaneously with the stabilisers, which implies that it should commute with all the stabilisers.

Let us once again look at the repetition code and see how some operators and observables act on it. Take for example the operator $X_1X_2X_3$, on the physical state space it

maps $|000\rangle$ to $|111\rangle$ and vice versa. For the encoded space this would mean it maps $|\overline{0}\rangle$ to $|\overline{1}\rangle$, which would make it the $\overline{X}$ operator.

Or take for example the observable $Z_1Z_2Z_3$. It commutes with all stabilisers and after measurement it results in 1 for the state $|000\rangle$ and in -1 for the state $|111\rangle$, meaning it is equivalent to the logical $\overline{Z}$ observable in the code space.

It is possible to invert this relation between the isomorphism and the logical operators. One can define the isomorphism by taking one physical state to be $|\overline{0\ldots0}\rangle$, as for example found with the method for the CSS codes, and then assign a set of representative Pauli operators from the cosets of $\mathbf{S}$ in $N(\mathbf{S})$ to be the operators on the encoded space. For this assignment to be valid it also needs to respect the operator product of both groups i.e. be a group isomorphism and the physical operators assigned to logical operators with $|\overline{0\ldots0}\rangle$ as eigenvector should also have the corresponding physical state as eigenvector. Then one can apply the logical operators to the state vector representing the $|\overline{0\ldots0}\rangle$ state to find the other encoded states.

To illustrate this method, we will apply it to find an alternative encoding for the Shor code. We will start with the state $|000000000\rangle$, which already passes all the Z based stabilisers. When measuring $X_1\cdots X_6$ and $X_4\cdots X_9$ we will get a random outcome, but let's suppose both pass. From the wave function collapse we then find that the state collapse to

$$|\overline{0}\rangle = \frac{1}{2}(|000000000\rangle + |111111000\rangle + |000111111\rangle + |111000111\rangle), \tag{1.7}$$

if we then take $\overline{X} = X_1\cdots X_9$ and $\overline{Z} = Z_1\cdots Z_9$ as our representative logical operators, which respects our choice for $|\overline{0}\rangle$, we can find $|\overline{1}\rangle$ through

$$\begin{aligned}
|\overline{1}\rangle &= \overline{X}|\overline{0}\rangle && (1.8)\\
&= X_1\cdots X_9\frac{1}{2}(|000000000\rangle + |111111000\rangle + |000111111\rangle + |111000111\rangle) && (1.9)\\
&= \frac{1}{2}(|111111111\rangle + |111000000\rangle + |000111000\rangle + |000000111\rangle). && (1.10)
\end{aligned}$$

Importantly the choice of representative operators does not need to change if different outcomes were measured for $X_1\cdots X_6$ and $X_4\cdots X_9$, because $N(\mathbf{S})$ remains the same after fixing the failed X stabilisers. So in essence when using this method, the exact underlying code might change but the preparation of all-zeroes state and logical operators remains the same.

With the basics of QEC covered, let us continue with the CSS code that will be used in this thesis, namely the toric code. The toric code is defined visually by placing a square lattice on a torus and putting qubits on the edges of the lattice. On each vertex a generating X stabiliser is placed with support on the connecting edges (also called a star) and on

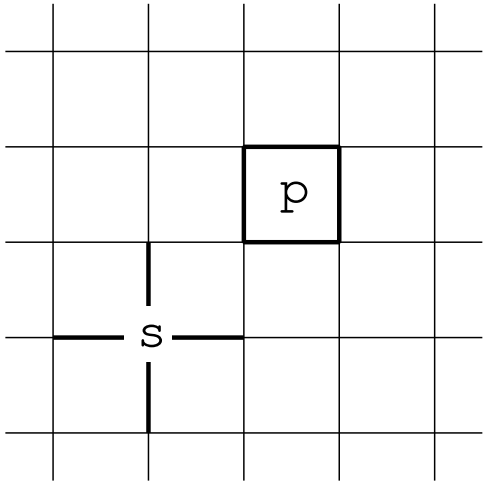


Figure 1.2: Example of a lattice that can be imposed on a torus, with a star and a plaquette stabiliser. The edges on the top and bottom are connected, as are the left and right side. Taken from [21].

each face a generating Z stabilisers is placed with support on its edges (also called a plaquette). A visualisation can be seen in Figure 1.2.

The lattice is usually taken to have the same number of divisions, L, in the horizontal and vertical direction. The total number of qubits in the code becomes then $2L^2$ and the total number of generating stabilisers is also $2L^2$. This set of generating stabilisers is however not independent as $\sum_{v\in V} X_v = I$ and $\sum_{f\in F} Z_f = I$, but by removing one X stabiliser and Z it does become independent. The QEC therefore encodes a total of $n - |\mathbf{G}| = 2L^2 - (2L^2 - 2) = 2$ logical qubits.

In Figure 1.3 one can also see another representation of the toric code with the choice of logical operators that will be used in this thesis. In general we see that all undetected errors take the form of a loop that passes around at least one of the boundaries. The smallest of such loops is exactly a straight loop of length L. Loops that do not pass around the boundary are the product of stabilisers contained in that loop. Therefore the Toric code is family with parameters $[[2L^2, 2, L]]$.

Closely related to the toric code is the rotated toric code. It is created by imposing a square lattice with a rotation of 45° on a torus. On each vertex a generating X stabiliser is placed with support on the connecting edges (also called a star) and on each face a generating Z stabilisers is placed with support on its edges (also called a plaquette). Because the lattice is not aligned with the torus each division top-left to bottom-right crosses the division top-right to bottom left twice. This leads to twice as many qubits, X stabilisers, and Z stabilisers. With the same number of divisions L in both directions this leads to the parameters $[[4L^2, 2, L]]$.

It is also possible to forgo the torus and construct QEC codes on a 2D surface it self. In [22] Bravyi and Kitaev extend their work of lattices on the torus to 2D surfaces by

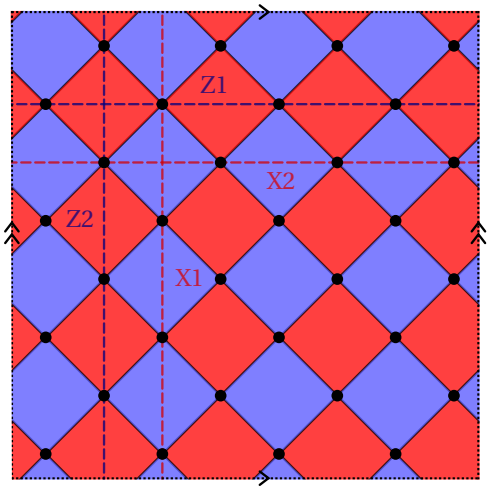


Figure 1.3: A visualisation of the toric code using dots to represent the qubits, blue shapes to represent Z stabilisers, red shapes to represent X stabilisers, and dashed lines to indicate the logical operators. The corners of the shapes indicate that the stabiliser has support on that qubit. The logical operator has support on the qubits it crosses. Top and bottom boundary are connected, as are the left and right boundary.

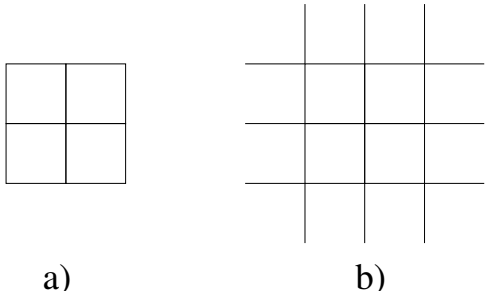


Figure 1.4: Lattices with different boundaries. a) Z boundary. b) X boundary.

introducing the notion of boundaries as can be seen in figure 1.4. Instead of the sides connecting, one can add boundaries to the toric code to find the surface code. The same can be done for the rotated toric code to find the rotated surface code. In figure 1.5 one can see both the unrotated and rotated surface code. The surface code has parameters $[[2L^2-2L+1,1,L]]$ and the rotated surface code $[[L^2,1,L]]$.

1.2.2. ERROR CHANNELS

In the last section, we only dealt with Pauli errors and while they already cover a broad range of errors, they are only a finite set in the continuous set of possible operators on the state space. The picture becomes even bleaker when we generalise errors to open quantum systems. In this subsection you will see that luckily all hope is not already lost and that correcting Pauli errors also allows for correcting Pauli Error channels. Lastly, the error model used in the simulation will be introduced.

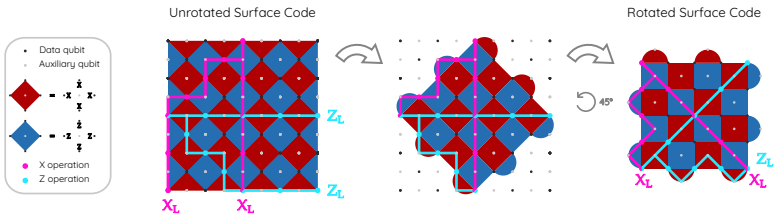


Figure 1.5: A visualisation of both the rotated and unrotated surface code using dots to represent the qubits, blue shapes to represent Z stabilisers, red shapes to represent X stabilisers, and highlighted lines to indicate the logical operators. The corners of the shapes indicate that the stabiliser has support on that qubit. The logical operator has support on the qubits it crosses.

In open quantum systems an error can be any operator on the state space. For density matrices this means the following:

Definition 1.2.8 (Error channel)**.** An error channel $\mathrm{E} : \mathscr{A} \to \mathscr{A}$ is a completely positive trace preserving map that maps density matrices to density matrices. Meaning $\forall A \in \mathscr{A}$, $\mathrm{Tr}_{\mathscr{A}}(A) = \mathrm{Tr}_{\mathscr{B}}(\mathrm{E}(A))$ and $\forall n \in \mathbb{N}$, $\mathrm{E} \otimes I_n$ is a positive map.

These error channels are as general as we can get with open quantum systems and of note is that not all of them can be recovered. Take for example the channel that maps all states to the $|0\ldots0\rangle$ state. Physically this would look like losing all energy in the system, and collapsing to the ground state. After it gets applied, all states are the same and no information from the original state can be recovered. For there to still be a chance to recover the original state we will need to focus on a more manageable set, namely the Pauli error channels.

Definition 1.2.9 (Pauli error channel)**.** An error channel E on $\mathbb{C}^{2^n}$ is a Pauli error channel if it can be written as:

$$\mathrm{E}(\rho) = \sum_{P \in \mathbf{P}} p_P P \rho P^\dagger \tag{1.11}$$

where $p_P \in [0,1]$ and $\sum_{P\in\mathbf{P}} p_P = 1$.

From theorem 2.6 in [18] we know that QEC codes with distance d can still correct Pauli error channels featuring only Paulis with weight less or equal to $\frac{d-1}{2}$. This is because the moment the ancilla qubits get measured, the resulting syndrome has to project onto one of the values. Consequently the mixed error also has to project onto set of Pauli errors with that syndrome. This then means that if the channel only features Pauli error with different syndromes, that the state becomes pure again and can be corrected by the stabiliser code.

One example of a Pauli channel, which appears a lot in the literature, is the depolarising channel. It applies no error with probability $1-p$ and the other Pauli errors with uniform probability. The one qubit version of this error channel is usually written as

$$E(\rho) = (1-p)\rho + \frac{p}{3} X\rho X^\dagger + \frac{p}{3} Y\rho Y^\dagger + \frac{p}{3} Z\rho Z^\dagger \tag{1.12}$$

In the simulation for this thesis we will be using *Circuit-level noise* as our noise model. This error model emulates the susceptibility of quantum computers to noise through

the entire execution without bias towards particular states or operations. It is also most commonly used in QEC literature for threshold computation under more stringent conditions. More precisely it applies the following error channels during execution.

- During a reset to a state, with probability p, the orthogonal state is reached (e.g. $|0\rangle$ becomes $|1\rangle$ and $|+\rangle$ becomes $|-\rangle$).
- During a measurement, with probability p, the opposite result is reported.
- After a 1-qubit gate a 1-qubit depolarizing channel is performed with parameter p.
- After a 2-qubit gate a 2-qubit depolarizing channel is performed with parameter p.

1.3. MEASURING LOGICAL OBSERVABLES WITHIN A QEC CODE

This section covers the challenge of measuring logical observables that are encoded in a QEC code. We will first illustrate the problem by showing that direct measurement of physical qubits affects stabiliser in the QEC code. Then we will show an alternative to this approach that can work within a QEC code. Finally we will introduce a larger construction that is used in this thesis from [1] and further illustrate its direct application to the toric code.

As seen in section 1.2.1, logical observables in a QEC code take the form of observables on several physical qubits. As measuring observables on multiple qubits is with some technologies not directly possible, one could instead directly measure each physical qubit separately and then use classical parity to find the result of the logical observable. This comes with the downside of collapsing the state of those particular physical qubits. This collapse results in the state leaving the code space of the QEC code. So direct separate measurements on individual physical qubits is not a good solution if quantum computation needs to continue beyond that measurement.

Instead of measuring directly and then computing the parity classically, the logical observable can also be measured indirectly. This involves getting an ancilla qubit entangled with all the relevant qubits to connect its state with the parity of the measurement and therefore the result of the logical observable. To measure $Z_1 Z_2$ for example, one could perform the following circuit shown in figure 1.6. The advantage of this approach is that the physical state does not collapse to a state outside of the code space, leaving the code space intact. The downside is that the connectivity requirements for the ancilla qubit scales with the size of the logical, which is at least d. So,in some sense a different construction is needed to effectively perform these logical observables without dramatically increasing the connectivity requirements.

In their milestone paper Cohen *et al.* [1] provided a general construction allowing for the measurement of logical observables consisting of Pauli strings through attaching an ancilla structure to the QEC code. Conceptually what this construction does, is expand the QEC code to also include the logical observable as one of the stabilisers through the addition of an ancillary surface. This way the logical measurement is performed simul-

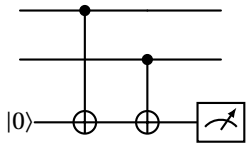


Figure 1.6: Quantum circuit for indirect measurement of $Z_1 Z_2$ using an ancilla qubit.

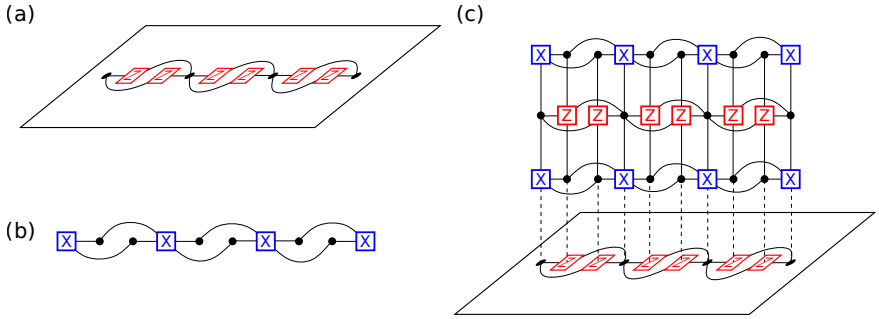


Figure 1.7: **Measurement of a logical $\overline{X}$ observable of a QEC code.** **(a)** The QEC code visualised as a square with only the relevant qubits in the support of $\overline{X}$ shown as dots and Z stabilisers with overlapping support shown in red squares. The lines connecting the dots and square represents support of the stabiliser **(b)** The dual layer that is used to measure the operator indirectly. Each qubit in the support of $\overline{X}$ is replaced with a X stabiliser and each overlapping Z stabilisers replaced with an ancilla qubit that is prepared in the $|0\rangle$ state. **(c)** The QEC code with the attached ancilla structure consisting of two dual layers. In between each dual layer, a primal layer similar to that of the support on the QEC code is added. The lines again represent support, with support also going up and down layers. the product of all the X stabilisers in the complementary layers cancels out on the qubits in the ancilla structure and becomes the logical $\overline{X}$ observable, meaning that the classical parity of the measurement of these stabilisers also gives the logical $\overline{X}$ observable. Taken from [1].

taneously while doing rounds of QEC. The additional benefit of this structure is that it maintains the code distance of the original code. Although the structure requires quite a large number of ancilla qubits, it only alters the needed connectivity of the original QEC code slightly, by adding additional qubit connection to the qubits and stabilisers that are overlapping with the logical operator. The overview of the construction can be seen in figure 1.7.

As in figure 1.7 the construction works by adding alternating primal and dual layers of qubits and stabilisers on top of the desired logical in such a way that the combined parity of all the stabilisers in the dual layers is the logical operators that is to be measured. An example of a primal layer for a logical X can be seen in part a of figure 1.7 and the dual layer in part b. In this thesis when referring to the number of layers, we mean the total number of dual layers. For a more formal description we recommend reading [1].

In this thesis we will simulate measuring the logical $\overline{Z_1}$ on the toric code. Figure 1.8 shows what the addition of the Cohen *et al.* construction for $\overline{Z_1}$ looks like on top of the toric code.

1

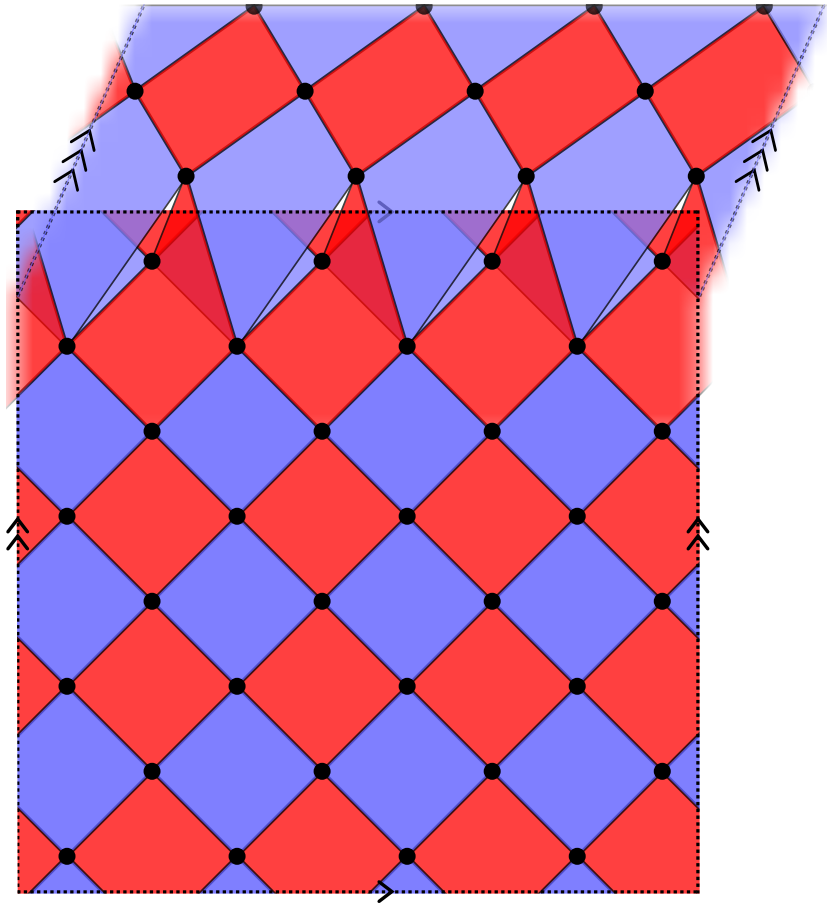


Figure 1.8: A visualisation of the toric code with ancilla system from [1] using dots to represent the qubits, blue shapes to represent Z stabilisers, red shapes to represent X stabilisers. The corners of the shapes indicate that the stabiliser has support on that qubit. The ancilla system is placed on top of $\overline{Z_1}$ and consists of two layers. The classical parity of all the Z Pauli stabilisers is $\overline{Z_1}$. The boundaries with same number of arrows are connected.

1.4. SIMULATION OF CLIFFORD QUANTUM CIRCUITS

This section discusses how the quantum circuits in this thesis will be simulated. First we will discuss the general hardness of simulating quantum computing on classical devices. Then we will present a solution to the particular case of Clifford circuits. Afterwards we will discuss how to deal with the simulation of error channels and use it to find error rates through the Monte Carlo method.

The original motivation for quantum computers was to tackle the simulation of quantum systems [23]. This was necessary because the overhead to simulate arbitrary quantum systems with a classical computer grows exponentially with the size of the quantum system. This would seem to imply that it would be hard to explore the different properties of quantum computers and algorithms through simulation before going through the effort of implementing them. Fortunately not all quantum systems are hard to simulate and the Gottesman-Knill (GK) theorem [24] provides a solution:
Theorem 1 (Gottesman-Knill theorem)**.** *Any quantum computer performing only:*

1. *Clifford group gates*
2. *Measurements of Pauli group operators*
3. *Clifford group operations conditioned on classical bits, which may be the results of earlier measurements*

can be perfectly simulated in polynomial time on a probabilistic classical computer.

The stabiliser measurement circuits, encoding circuits, and measurement circuits used in this thesis are of this type, allowing us to efficiently simulate them on classical computers.

What this does not account for however is the error channels that the circuit-level noise model requires. There is luckily however a way to work around the computational complexity of the error channels. In essence each error channel in circuit-level noise applies a set of possible Pauli gates with each certain probability factor and only during the final measurement of all the qubits does the channel project onto a mix of the Pauli gates leading to the same measurement outcome. By forcing all error channels to directly choose a Pauli to apply instead of waiting for the final measurement, the entire circuit becomes Clifford again allowing for fast simulation. It does however mean that the simulation does not directly find the exact probability of the QEC code correcting the error. Instead the average result of the probabilistic simulation coincides with the error rates of the QEC code. The Monte Carlo method enables us to estimate the error rates of the QEC code, through the repeated simulation of the entire circuit and tracking the number of successes and failures.

All these techniques for simulation have been implemented in the STIM package [25]. Using STIM we will approximate the error rates of various realisation of the toric code with the ancilla system for our analysis.

1

1.5. MORPHING CIRCUITS

In this section we will present a comprehensive summary on the morphing design procedures presented in [12], [13], [26], and [27]. We will start by first reimagining the detecting properties of stabilisers by looking at the whole measurement process from reset of the ancilla qubit to the final measurement. Then we will use this new perspective to redesign QEC codes with the morphing design technique by splitting the stabiliser measurements in several contraction rounds and performing the measurements without the need for ancilla qubits.

1.5.1. DETECTING REGIONS

In this subsection we will cover the concept of detecting regions, which identifies the places in a quantum circuit where errors would trigger the related stabiliser to fail. Then we will explain how to find these detecting regions by propagating stabilisers backwards in time through the circuit. Finally to illustrate and apply the theory, the detecting regions of the repetition and toric code are given. In this subsection we will stick again to using only Pauli errors.

In section 1.2.2 we shortly covered the circuit-level noise model, which places errors after all the gates and resets and before measurements, but did not consider how errors in the middle of the circuit affect the stabiliser measurement. To find the effects of the Pauli error on the stabilisers measurement, it is usually easier to find the equivalent circuit with a Pauli error right before the measurement. The equivalence rules for quantum circuits that are needed to propagate Pauli errors to the ancilla qubit measurement in this thesis are given in figure 1.9 and figure 1.10.

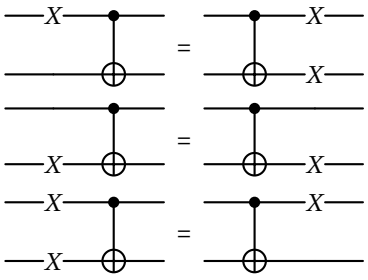


Figure 1.9: Propagation of X errors through CNOT's.

With these rules for the CNOT and treating Y Pauli operators as a product of a X and Z Pauli, all the Pauli errors accumulated during the execution of the circuit can be propagated through the circuit. With an X error just before a Z measurement flipping the result and vice versa. What you can see from these rules, is that the place and time step of the error has a significant impact on how large the error is at the stabiliser measurement. It could even be that one single qubit error could expand into an error that is no longer correctable.

While these rules are nice for checking what stabilisers are triggered for one Pauli error, they do not provide us with a way to find the potential Pauli errors given a triggered

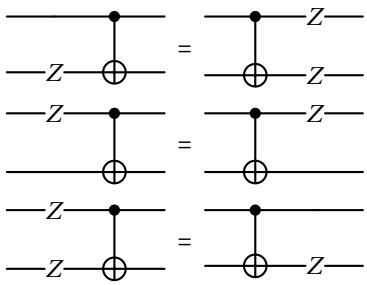


Figure 1.10: Propagation of Z errors through CNOT's.

stabiliser. We could calculate the effects of all the different Pauli errors but this would be rather tedious. Instead of propagating errors through the circuit, we can also approach the problem through the lens of moving the stabiliser measurement backwards through the circuit as discussed in [24]. This change of moving the observable instead of the state is similar to switching from the Schrödinger picture to the Heisenberg picture. If we have our state $|\phi\rangle$, operation U and observable S, then measuring S after applying U would be the same as just measuring the observable $U^\dagger SU$ because

$$\langle SU|\phi\rangle, U|\phi\rangle\rangle = \langle U^\dagger SU|\phi\rangle, |\phi\rangle\rangle. \tag{1.13}$$

When working with CNOT's the equivalence rules for observables fortunately coincides with figure 1.9 and 1.10. Keep in mind that the stabiliser is still a parity measurement here, so if the stabiliser has support on multiple locations, then it will only trigger if there are an odd number of errors (X errors for Z stabilisers and vice versa) happening at those space-time points. All the possible space-time locations, which could lead to triggers in the stabiliser measurement, together are called the detecting region.

With the basic equations covered, let us continue with an example from [13]. In figure 1.11, an example circuit for the bit-flip repetition code with distance five is given in b). The circuit is performed a total of four cycles and the detecting regions are illustrated with different colours. Important to note is that instead of checking each stabiliser by itself, the parity of the measurements of the current and last stabilisers are performed in this circuit to check if errors occurred between the two cycles. The result is a contracting and expanding pattern that repeats throughout the code.

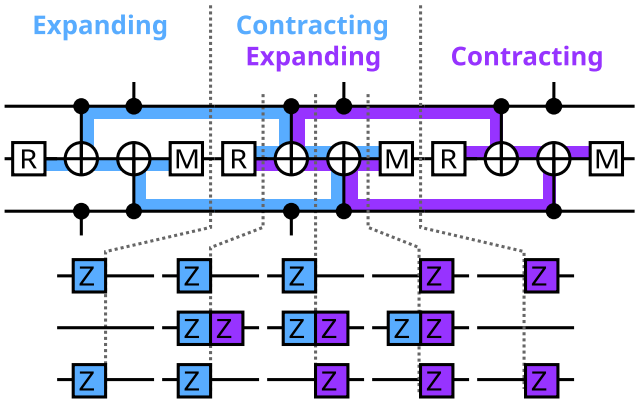
The STIM package[25] also allows for visualisation of detecting regions. STIM produces separate diagrams after each time step with the qubits on fixed positions in the plane. Each of diagrams will show the operators that were performed during that time step and the resulting detecting regions drawn as coloured shapes. The colour of the shape indicates the type of stabiliser with blue for Z[1] and red for X and the corners of the shape indicate support on that qubit. An example of this can be seen in figure 1.12.

[1] A helpful mnemonic to keep in mind is *BlueZ*.

1

Bit-flip Repetition Code

a) Detecting regions during a cycle



b) All detecting regions for a distance-5 code run for 4 cycles

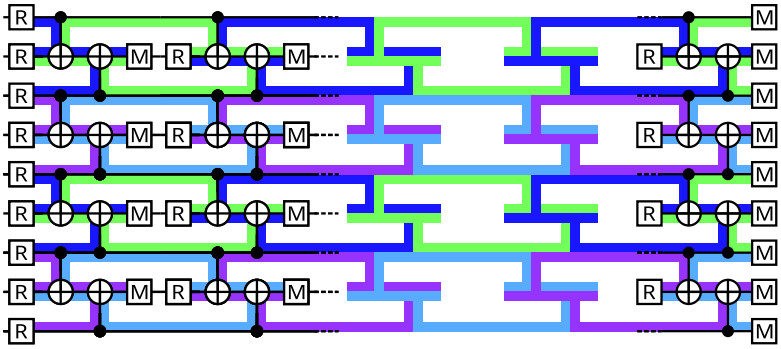


Figure 1.11: All locations of all highlighted regions shown here are of Z Pauli stabilisers. a) Two neighbouring detecting regions of the bit-flip repetition code (blue, purple), shown over three cycles. Each detecting region covers two cycles: In the first cycle, it emerges from a reset on the measure qubit and expands to cover the code stabiliser. In the second cycle, it contracts from the code stabiliser down to terminate on a measurement. During the middle cycle, where the two shown regions co-exist, time-slices of the detecting regions are shown. b) All detecting regions for a distance-5 bit-flip repetition code. The detection regions overlap such that all locations in the circuit are covered by two detecting regions, except the boundaries which are covered by one. Note the smaller detecting regions at the start of the code that emerge from resets on both measure and data qubits. On cycles three and four, the circuit elements are omitted to emphasise that the detecting region shapes alone are sufficient to understand the implementation. Note the final detecting regions terminate on measurements on both the measure and data qubits. Taken from [13].

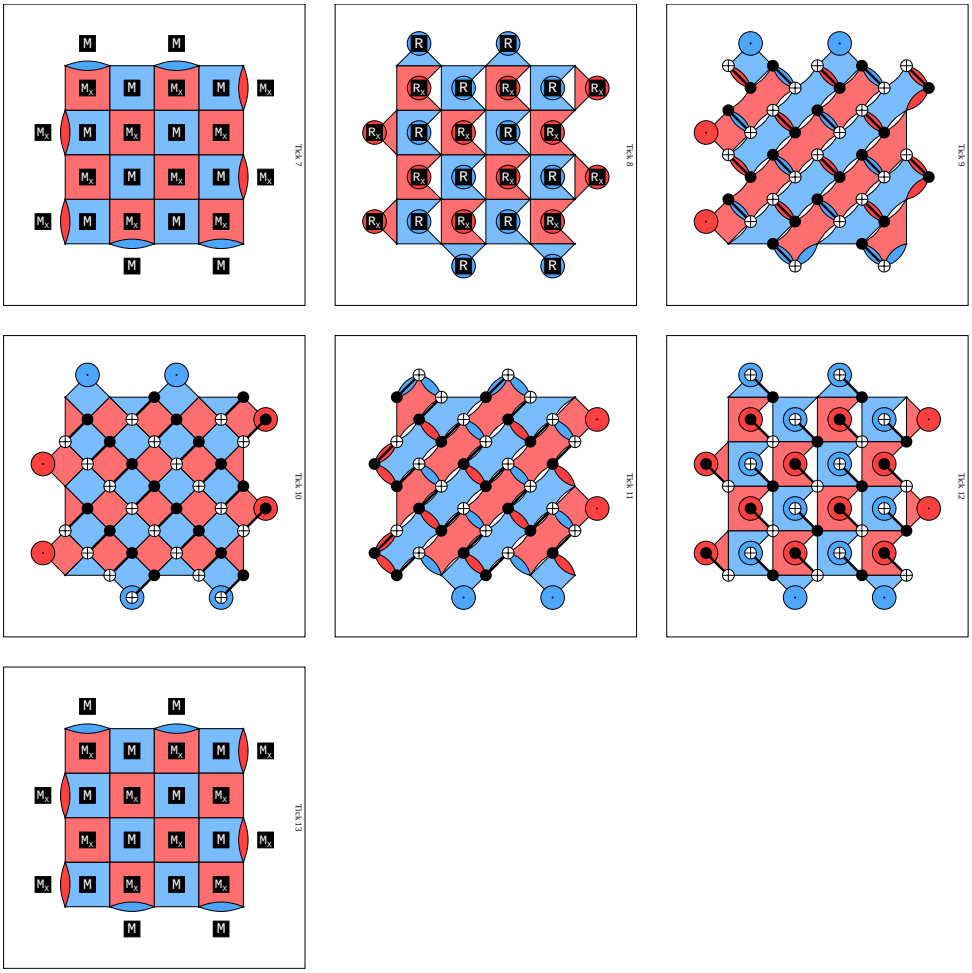


Figure 1.12: One round of error correction with the surface code visualised by STIM. From the reset new stabilisers emerge and expand to cover the qubits while at the same time the stabilisers from the previous round are contracted to the ancilla for measurement.

1.5.2. MORPHING CODES

In this section we will explain how the morphing design principle works and what the motivation behind the principle is. Then we will show a couple of examples to familiarise the reader with the structure. Afterwards we will combine the morphing design principle together with the construction from [1] to end up with the complete circuit that will be performed in this thesis is given.

When looking at the detecting regions of the stabiliser circuits, one can see that all space-time points are covered by both X and Z detecting regions. This is necessary for the circuit to be able to detect all types of errors in the execution. One could design error correcting circuits based on just detecting regions and then check that all errors up to a certain weight are covered, but doing this from scratch is complex.

Before going further into the technique of morphing, let's more closely examine the structure of the error detecting regions of the standard stabiliser code. During the execution of each QEC cycle, there is a group of expanding stabilisers and a group of contracting stabilisers. The CNOT's in the QEC cycle contract the contracting stabiliser down to the ancilla for the final measurement, whilst simultaneously expanding the expanding stabilisers to the beginning shape of the contracting stabilisers. Then the measurement removes the contracted stabilisers and the reset creates a new wave of expanding stabilisers and we have ended up at the start of the QEC cycle again.

When looking at this QEC cycle for the rotated surface code McEwen, Bacon and Gidney [13] noticed that in the middle of the QEC cycle, the surface code turned into the unrotated surface that also covered the ancilla qubits. By examining the QEC cycle with that midpoint as the home base, the QEC cycle looks like contracting half the stabilisers and then expanding the new stabiliser in such a way that the remaining stabilisers are moved into the position of the previous set of contracting stabilisers. McEwen, Bacon and Gidney realised that to end back up at the middle of the QEC cycle after contracting and measuring the stabiliser, they could also reverse the contracting circuit that was just performed. This however left the other half of the stabilisers unmeasured, so they devised a second different contraction circuit for the other half of the stabilisers. Then to perform QEC they would alternate the two contracting circuits and called it the hex-grid circuit. The hex-grid circuit achieved comparable performance to the standard circuit under their SI1000 noise model [13] and by designing the other contraction circuit to use the same CNOT connections as the first contraction circuit, they did manage to reduce the number of different two-qubit interactions needed from four per qubit down to three per qubit. The hex-grid circuit they came up with can be seen in figure 1.13 and the standard surface code version is given in figure 1.12.

In general, the morphing design principle tries to generalise this process of working from the middle of the QEC cycle. The morphing design principle splits the set of stabilisers into several subsets. Each of these subsets will individually be contracted and then expanded back to its starting shape. This middle point, from which each subset is contracted, is referred to as the mid-cycle code. The code after a particular contraction round is called the end-cycle code. The general shape of a correction circuit can be seen in figure 1.14.

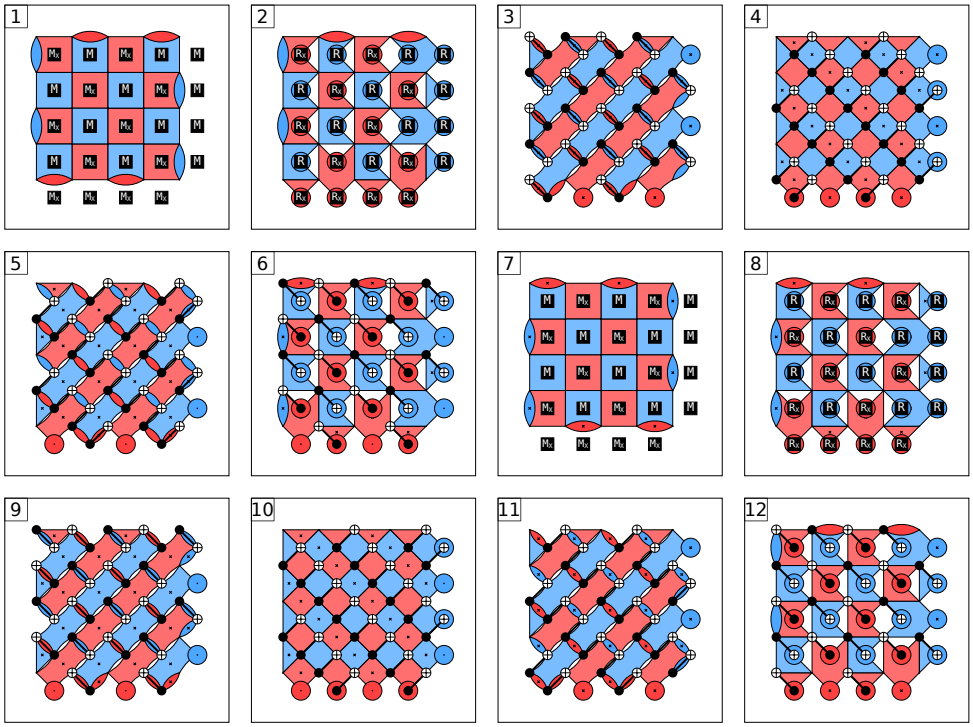


Figure 1.13: The hex-grid circuit for the surface code with both contraction circuits performed after one another. The stabilisers emerging from the reset in step 2 are marked with an x, the other set of stabilisers is unmarked. In step 3 and 4 are the stabiliser with an x are expanded back to the mid-cycle. In steps 5 to 10 the unmarked stabilisers are contracted and expanded to the mid-cycle state. In steps 11 and 12 the stabilisers with an x are contracted. Taken from [13]

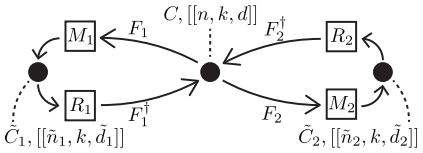


Figure 1.14: Graphical representation of the morphing design technique with two sets of stabilisers. F_1 and F_2 are the contracting circuits with the $F_1^\dagger$ and $F_2^\dagger$ being their inverse. C is the mid-cycle code and C_1, C_2 the end-cycle codes after their respective contraction circuits. Taken from [12]

.

Now that the general idea of morphing is hopefully clear, let us get more detailed with some tips on how to find these groups of contracting stabilisers and how to design circuits for them. The contraction circuits that can be made heavily depend on the underlying mid-cycle code but as a guideline try to have CNOT's work for Z and X stabilisers at the same time; try to use as many qubits as possible in each time step; and try to play around with the steps and see if it is possible to combine the initial steps of two different contraction circuits. We consider it best to have two sets of contracting stabilisers, to reduce the amount of time between stabiliser measurements. A general lower bound for the depth of a contraction circuit is $\log_2(w(S))$ of the largest stabiliser in the set.

Morphing design also does come with some downsides. The end-cycle codes that you end up with after contraction, are not guaranteed to have the same distance. This could be problematic if for example one of the cycles in your morphing circuit is significantly worse than the others and brings down the entire performance of the circuit. The distance is however lower bounded by $\frac{d}{c_i}$ where d is the distance of the mid-cycle code and c the depth of the contraction circuit i [12].

In section 1.2.1 the ancilla structure from [1] is discussed to measure the encoded logical qubits of the code by adding an additional structure to the code. In this thesis we will take the same ancilla structure and combine it with the morphing technique. To do this, we design both a morphing circuit for the toric code with and without the ancilla structure attached. Because the toric code without the ancilla system is very similar to the surface code, McEwen, Bacon and Gidney [13] also came up with the hex-grid circuit for it and put it in their supplementary document. This hex-grid circuit can be shown in figure 1.15.

The morphing circuit for the toric code with the ancilla structure can for the largest part remain the same, only the circuit for the stabilisers in the ancilla structure and the toric code stabilisers in contact with the ancilla structure needs to be changed. But the problem lies now in the fact that the toric code stabilisers in contact with the ancilla structure have support on an additional qubit from the ancilla structure increasing their weight to 5, so 3 layers of CNOT's are needed for a contraction circuit. The contraction circuits for the toric code with ancilla structure we came up with are in figure 1.16. The contraction circuit works by first detaching the ancilla from the toric code and partially contracting the stabiliser of the ancilla structure and then running the hex-grid circuit as normal on the toric code and collapsing the remaining part of the ancilla stabilisers as well.

To go between the two different morphing circuits only a small step is needed. Because the contracting subsets on the toric code line up with each other, it is possible to add the ancilla system during the reset. The reset is replaced with the reset of the circuit with the ancilla system and the other uninitialised qubits in the ancilla structure are reset to $|+\rangle$, as not to affect X stabilisers that will be in contact with them. Then the morphing circuit with the ancilla system can be performed for as long as desired. To split off the ancilla structure measure the remaining qubits in the ancilla structure in the X basis during the measurement phase and then continue the morphing circuit without the ancilla structure using the results of the additional measurements for the stabilisers that had support on the ancilla structure.

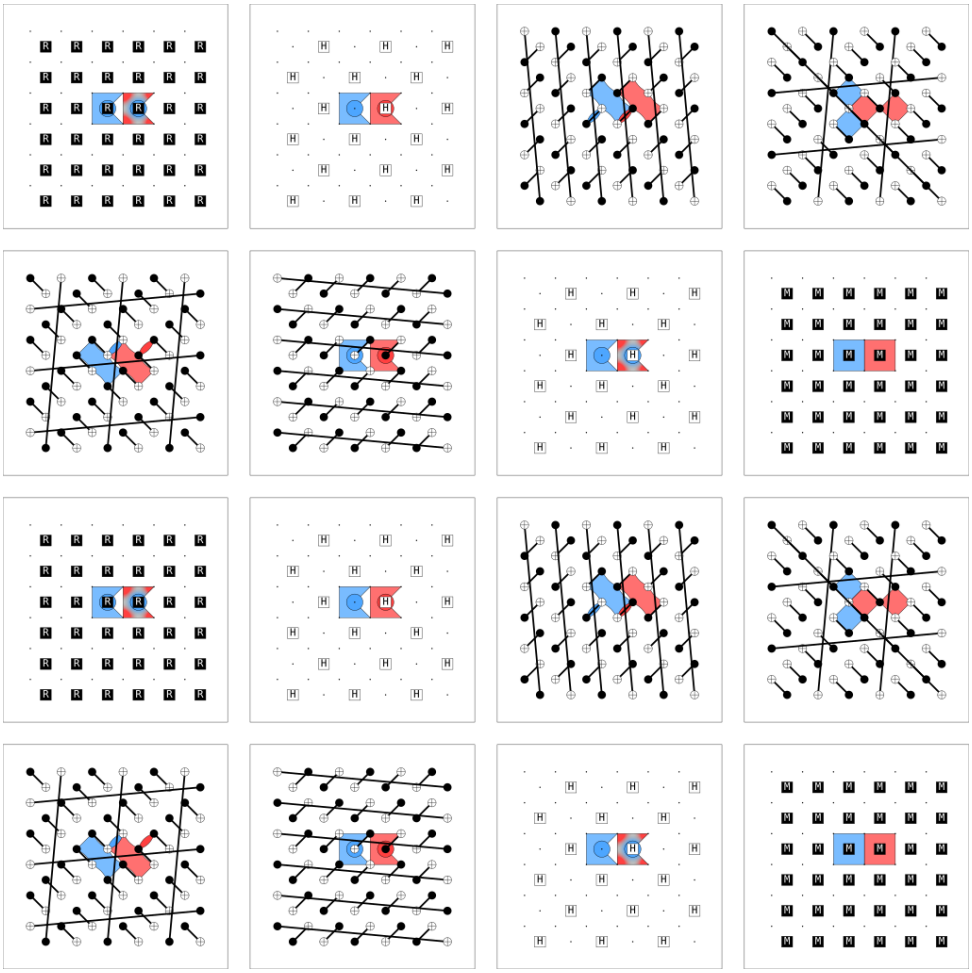


Figure 1.15: The hex-grid circuit for the toric code [[72,2,6]] with both contraction circuits performed after one another. The mid-cycle code is reached in the 4th and 12th step, which coincides with figure 1.3. The different end cycles are shown in the 8th and 16th step. Only a couple of stabilisers are shown as there is a lot of overlap from the underlying toric shape. Taken from [13]

.

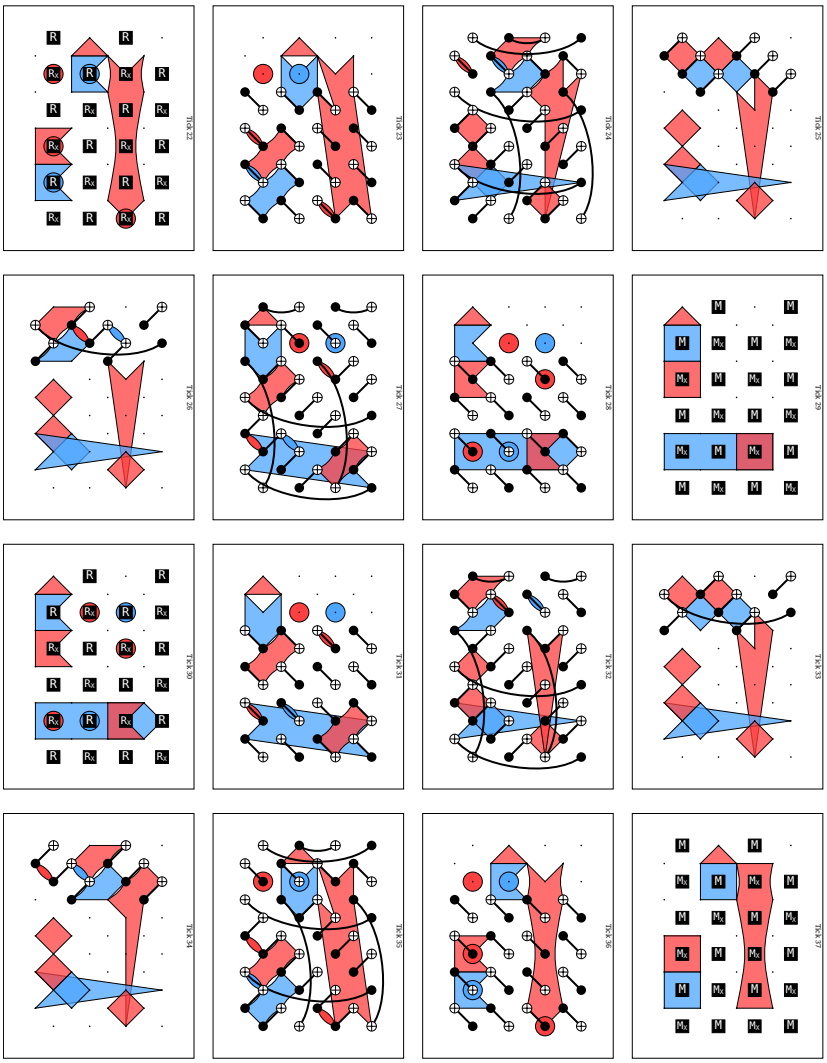


Figure 1.16: Visualisation of the morphing circuit for the toric code $[[32,2,4]]$ and 2 dual layers of ancilla structure and 1 primal layer. The upper three rows of qubits are the ancilla structure and the lower eight rows are the toric code. Not all stabilisers are shown to avoid clutter. The mid-cycle code is reached on tick 25 and 33.

2

METHOD

To get an understanding of the probabilities of a measurement error happening through the use of the ancilla structure as defined in [1] on toric code with morphing, we will simulate it with varying code distances, amounts of QEC cycles, and numbers of layers used in the ancilla structure.

To perform these simulations we will make use of the STIM python package [25] together with sinter and ldpc [20] packages. The STIM package allows us to describe the circuits in detail with detectors and errors, after which it can efficiently simulate the noisy execution. Then, thanks to the workflow in the sinter package, we can automate the execution, decoding of the final measurement, and save the results to a csv file. The BP_OSD decoder, provided by the LDPC library, will be used to decode the final outcome.

Figure 2.1 shows the circuit we want to perform on the logical qubits in the toric code. The first logical qubit will be prepared in the $|\overline{0}\rangle$ state and will then be measured with the ancilla structure in the Z-basis under circuit-level noise. By repeatedly performing this simulation and the Monte-Carlo method we can estimate the probability of a successful measurement. We will not be running the variations with the X-basis measurements on the first qubit, since the probability of a successful measurement is identical due to the underlying symmetry between the X and Z stabilisers. The second qubit will be set to either $|\overline{0}\rangle$ or $|\overline{+}\rangle$ and will be left to collect noise during the measurement of the first qubit. After the measurement on the first qubit is performed, the second qubit is measured without noise to see if it has experienced any logical errors whilst idling. Before we can expand these logical circuits into circuits for the physical qubits, we must first set a couple of parameters relating to the underlying QEC code, the ancilla structure and the noise model. Namely:

- Code distance, d
- Number of complementary layers in the ancilla, n_a
- Number of QEC rounds with ancilla structure attached, n_r
- Error probability for the uniform circuit-level noise, p

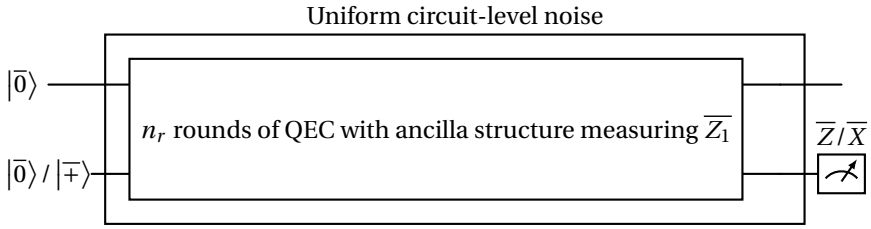


Figure 2.1: Logical level overview of the circuits simulated in this thesis

In appendix A the code that generates the corresponding circuit, can be found. An example with two rounds of QEC, together forming the full figure-8 morphing circuit, with a code of distance 4 and 2 layers can be seen in figure 1.16.

For this thesis project we decided to explore a wide range for each of the parameters to allow us the freedom to create several plots without having to wait for data. The complete CSV data set that can be imported with sinter for each experiment can be found in the attachments.

3

Results and Discussion

This chapter is split up into three sections, each answering one of the following questions:

1. What number of dual layers should be used in the ancilla system?
2. What number of rounds should be used?
3. Does the morphing version outperform the standard version?

As the data generated from the simulations is high dimensional, one figure representing all the data would be illegible. The reader is encouraged to explore the data sets or extend them themselves, if they feel there are more interesting plots to be seen that are not shown here.

3.1. What number of layers should be used in the ancilla system

The construction by Cohen *et al.* [1] comes with a degree of freedom with how many layers should be used. In their paper Cohen *et al.* recommends adding at least d dual layers to the code, where d is the distance of the original code. In figure 3.1 one can see the different measurement and idle error rate based on the number of dual layers used.

One can see that additional layers seem to have degraded the success probability of the measurement whilst not affecting the fidelity of the idle state much. This is in line with the theory from [1], in which Cohen *et al.* states that the additional layers are needed to protect the code distance from potential logical errors that start at the top layer of the ancilla system and that work their way down the ancilla to finally terminate in the underlying code. Luckily it is not possible to terminate in the toric code without having performed a full loop around the boundary, which has the same weight as in the original code, or looped back to the top of the ancilla structure, which is a product of stabilisers. Therefore the toric code does not need such protection.

Because the logical observable is the result of the parity of all the stabilisers in the dual layers, a single measurement error on one of the stabilisers already affects the result of

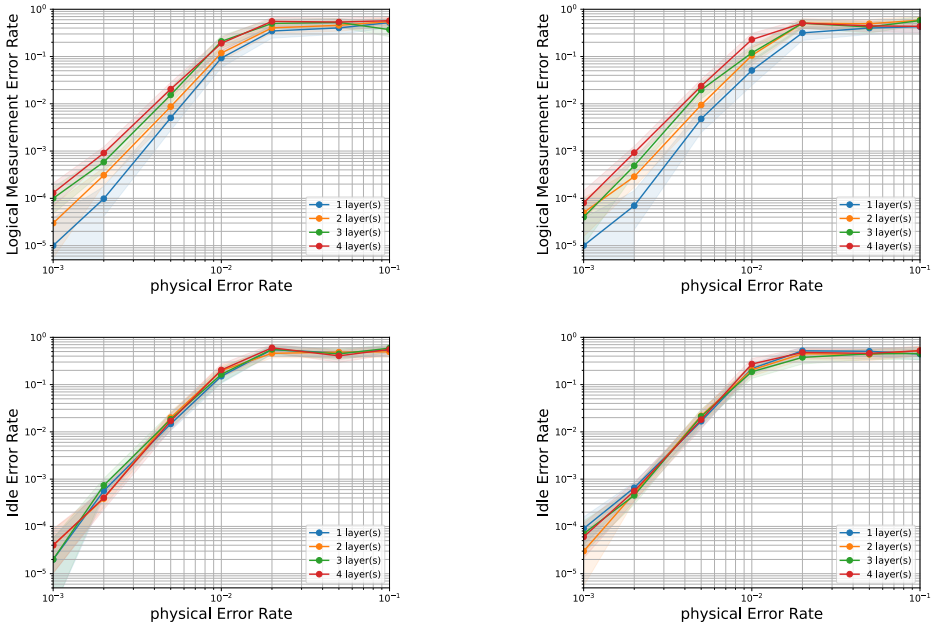


Figure 3.1: Numerical performance of the morphing version of the Cohen *et al.* construction with varying layers on a toric code with parameters $[[72,2,6]]$ after 6 rounds of QEC with respect to the uniform circuit-level depolarizing noise model and decoded using BP-OSD. The left two graphs are from the $|00\rangle$ based experiment and the right two graphs are from the $|0+\rangle$ based experiment

the logical observable. Having additional layers means that there are more stabilisers that could have potentially undergone an error affecting the final logical observable. So for the specific case of the toric code, the additional layers increase the measurement error while providing no significant improvement in idle error rate.

3.2. WHAT NUMBER OF ROUNDS OF QEC SHOULD BE USED

Although the measurement of the logical observable is performed every round, the susceptibility of the final result to one stabilisers having undergone an error, makes it necessary to perform multiple rounds of QEC with the ancilla system. The question of how many rounds is not clear cut. In figure 3.2 one can see the different measurement and idle error rates n_r based on different numbers of rounds.

One can see that there is not a clear best candidate for the number of rounds, that is best in both measurement and idle error rate. The central conflict lies in the fact that doing more rounds of QEC decreases the measurement error rate but increases the idle error rate. The measurement system benefits from the additional measurements as they give the decoder more chances to detect potential measurement errors and correct for them. On the other hand, performing additional cycles of QEC creates additional chances for errors to occur in both the toric code and the ancilla system.

Rounds	2	4	6	8
Measurement errors needed	1	2	3	4
Power	1.018	2.771	4.245	5.0711

Table 3.1: Table showing the power scaling of the logical measurement error rate based on a least square fit of a power law on the data between $5 \cdot 10^{-3}$ and 10^{-2} of the morphing version of the Cohen construction with one layer on a toric code with parameters $[[72,2,6]]$ from both the ZZ and ZX experiment. The second row shows the amount of measurement errors needed to flip the final measurement, which is the expected theoretical power scaling.

What can also be clearly seen in figure 3.2 is that doing fewer rounds than the distance of the code is very detrimental to the gradient of the graph. To illustrate this effect more concretely we fitted the power law $b \cdot x^a$ to the middle section of our measurement error rate data and the results can be seen in table 3.1. The theory of fault tolerance tells us that this effect stems from the fact that fewer QEC cycles leaves the code vulnerable to measurement errors happening on some stabiliser in the ancilla structure on multiple different cycles, leaving the code open to lower weight measurement errors. By performing more rounds of QEC, the total number of measurement errors needed to flip the result of the final measurement also increases. Because measurement errors are considered unlikely, requiring more measurement errors makes the measurement error of the total process more unlikely. The fact that these measurement errors over multiple cycles are lower in weight than the types of errors in one cycle, forces the measurement error rate to scale with their weight instead of the distance. To make sure that these errors do not dominate the distance, one can make sure that the number of measurement errors needed is just as large as the distance by performing at least as many rounds. Going beyond the distance in the number of rounds still improves the logical measurement error rate but does not seem to increase the gradient of the graph.

We would recommend performing rounds at least equal to the distance but depending on the situations one might choose to lean more towards measurement error rate or logical idle error rate. The trade-off between measurement and idle error rate can also be viewed as a Pareto front in figure 3.3.

3.3. DOES THE MORPHING VERSION OUTPERFORM THE STANDARD VERSION?

Choosing to run the morphing version over the standard version introduces more circuit complexity compared to the standard version. To determine whether the morphing version justifies this trade-off, we compare its logical measurement and idle error rates against those of the standard circuit under the same noise model and decoder.

In figure 3.4 one can see the results for one specific set of parameters. The morphing version out performs the standard version of the circuit when it comes to logical measurement and achieves similar performance when it comes to idling error rates under these parameters. For other parameters, that we have managed to explore, similar graphs were found.

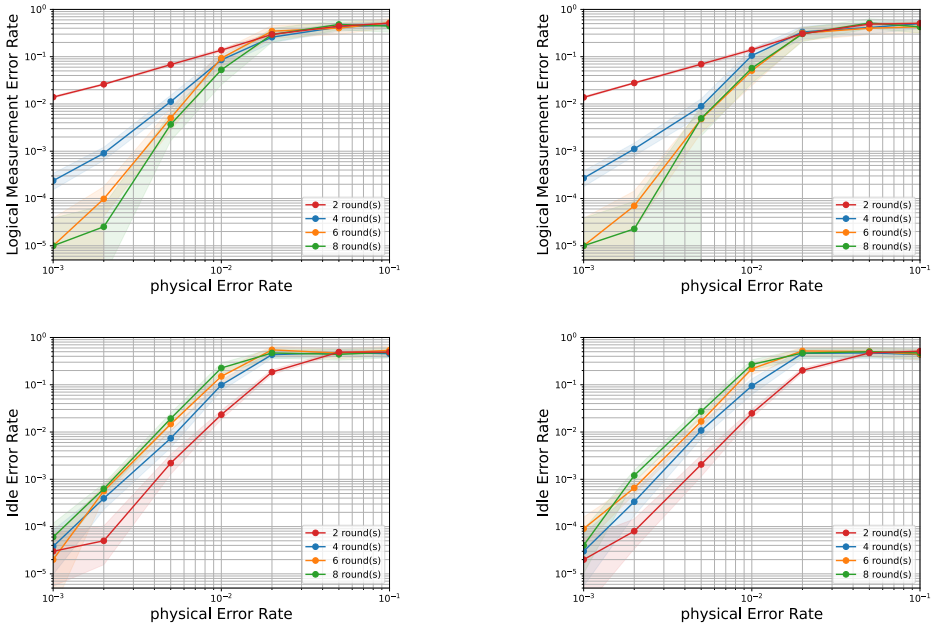


Figure 3.2: Numerical performance of the morphing version of the Cohen construction with one layer on a toric code with parameters $[[72,2,6]]$ after varying rounds of QEC with respect to the uniform circuit-level depolarizing noise model and decoded using BP-OSD. The left two graphs are from the $\overline{|00\rangle}$ based experiment and the right two graphs are from the $\overline{|0+\rangle}$ based experiment. The error bars on the logical measurement rate for 2 rounds is too small to be seen on these figures.

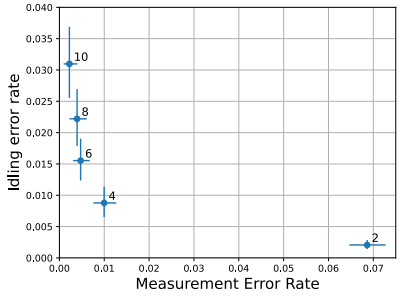


Figure 3.3: Pareto front showing the trade-off between logical measurement error rate and idle measurement error rate for the morphing version of the Cohen construction with one layer on a toric code with parameters $[[72,2,6]]$ with circuit-level noise with $p = 0.005$ for different numbers of rounds.

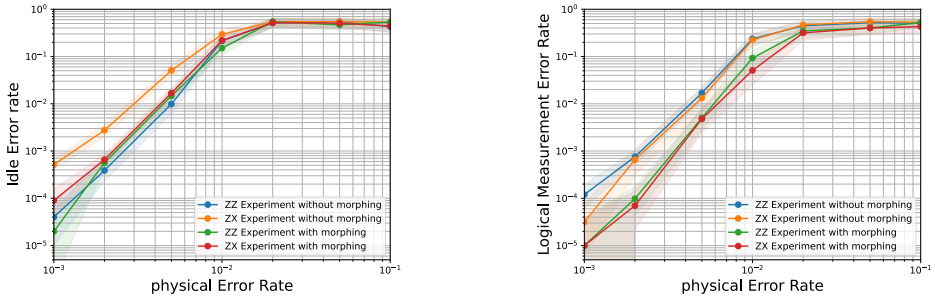


Figure 3.4: Numerical performance comparison between a morphing version of the toric code [[72,2,6]] and the standard toric code [[72,2,6]] with both one layer of the Cohen construction and 6 rounds of QEC with respect to the uniform circuit-level depolarizing noise model and decoded using BP-OSD.

This comparison is however not a clean apples to apples comparison. Firstly, the standard version of the toric code uses twice as many qubits in comparison to the morphing version. This doubling stems from the fact that the standard version needs ancilla qubits for stabiliser measurement but the morphing version instead collapse the stabilisers down onto already existing qubits to measure the stabilisers, foregoing the need for ancilla qubits entirely.

A more direct comparison could be made if the morphing version of the toric code would be compared to the standard circuit on one of end-cycle codes. This way the extra ancilla qubits are accounted for in the half of the qubits that were measured to reach the end-cycle code. With this comparison the starting point and end point of the entire measurement would also be the exact same. In the case of the toric code that would be the rotated toric code.

These differences, however, work in favour of the standard toric code, meaning that despite the disadvantages the morphing version managed to out perform its standard counterpart. We therefore conclude that the morphing version outperforms the standard circuit.

BIBLIOGRAPHY

[1] Lawrence Z. Cohen et al. “Low-overhead fault-tolerant quantum computing using long-range connectivity”. In: *Science Advances* 8.20 (2022), eabn1717. DOI: 10.1126/sciadv.abn1717. eprint: https://www.science.org/doi/pdf/10.1126/sciadv.abn1717. URL: https://www.science.org/doi/abs/10.1126/sciadv.abn1717.

[2] Lov K. Grover. *A fast quantum mechanical algorithm for database search*. 1996. arXiv: quant-ph/9605043 [quant-ph]. URL: https://arxiv.org/abs/quant-ph/9605043.

[3] Daniel A. Lidar and Haobin Wang. “Calculating the thermal rate constant with exponential speedup on a quantum computer”. In: *Phys. Rev. E* 59 (2 Feb. 1999), pp. 2429–2438. DOI: 10.1103/PhysRevE.59.2429. URL: https://link.aps.org/doi/10.1103/PhysRevE.59.2429.

[4] Jacob Biamonte et al. “Quantum machine learning”. In: *Nature* 549.7671 (Sept. 2017), pp. 195–202. ISSN: 1476-4687. DOI: 10.1038/nature23474. URL: http://dx.doi.org/10.1038/nature23474.

[5] Sukhpal Singh Gill et al. “Quantum computing: A taxonomy, systematic review and future directions”. In: *Software: Practice and Experience* 52.1 (2022), pp. 66–114.

[6] Dorit Aharonov and Michael Ben-Or. *Fault-Tolerant Quantum Computation With Constant Error Rate*. 1999. arXiv: quant-ph/9906129 [quant-ph]. URL: https://arxiv.org/abs/quant-ph/9906129.

[7] Rajeev Acharya et al. “Quantum error correction below the surface code threshold”. In: *Nature* 638.8052 (Dec. 2024), pp. 920–926. ISSN: 1476-4687. DOI: 10.1038/s41586-024-08449-y. URL: http://dx.doi.org/10.1038/s41586-024-08449-y.

[8] S. B. Bravyi and A. Yu. Kitaev. *Quantum codes on a lattice with boundary*. 1998. arXiv: quant-ph/9811052 [quant-ph]. URL: https://arxiv.org/abs/quant-ph/9811052.

[9] Dominic Horsman et al. “Surface code quantum computing by lattice surgery”. In: *New Journal of Physics* 14.12 (Dec. 2012), p. 123011. ISSN: 1367-2630. DOI: 10.1088/1367-2630/14/12/123011. URL: http://dx.doi.org/10.1088/1367-2630/14/12/123011.

[10] Sergey Bravyi, David Poulin, and Barbara Terhal. “Tradeoffs for Reliable Quantum Information Storage in 2D Systems”. In: *Physical Review Letters* 104.5 (Feb. 2010). ISSN: 1079-7114. DOI: 10.1103/physrevlett.104.050503. URL: http://dx.doi.org/10.1103/PhysRevLett.104.050503.

[11] Sergey Bravyi et al. “High-threshold and low-overhead fault-tolerant quantum memory”. In: *Nature* 627.8005 (Mar. 2024), pp. 778–782. ISSN: 1476-4687. DOI: 10.1038

/s41586-024-07107-7. URL: http://dx.doi.org/10.1038/s41586-024-07107-7.

[12] Mackenzie H. Shaw and Barbara M. Terhal. "Lowering Connectivity Requirements for Bivariate Bicycle Codes Using Morphing Circuits". In: *Physical Review Letters* 134.9 (Mar. 2025). ISSN: 1079-7114. DOI: 10.1103/physrevlett.134.090602. URL: http://dx.doi.org/10.1103/PhysRevLett.134.090602.

[13] Matt McEwen, Dave Bacon, and Craig Gidney. "Relaxing Hardware Requirements for Surface Code Circuits using Time-dynamics". In: *Quantum* 7 (Nov. 2023), p. 1172. ISSN: 2521-327X. DOI: 10.22331/q-2023-11-07-1172. URL: http://dx.doi.org/10.22331/q-2023-11-07-1172.

[14] Andrew W. Cross et al. *Improved QLDPC Surgery: Logical Measurements and Bridging Codes*. 2025. arXiv: 2407.18393 [quant-ph]. URL: https://arxiv.org/abs/2407.18393.

[15] Theodore J. Yoder et al. *Tour de gross: A modular quantum computer based on bivariate bicycle codes*. 2025. arXiv: 2506.03094 [quant-ph]. URL: https://arxiv.org/abs/2506.03094.

[16] D. S. Wang et al. *Threshold error rates for the toric and surface codes*. 2009. arXiv: 0905.0531 [quant-ph]. URL: https://arxiv.org/abs/0905.0531.

[17] Thaddeus D Ladd et al. "Quantum computers". In: *nature* 464.7285 (2010), pp. 45–53.

[18] Daniel Gottesman. "Surviving as a Quantum Computer in a Classical World". Unpublished manuscript, PDF. 2024. URL: https://www.cs.umd.edu/~dgottesm/QECCbook-2024.pdf.

[19] Peter W. Shor. "Scheme for reducing decoherence in quantum computer memory". In: *Phys. Rev. A* 52 (4 Oct. 1995), R2493–R2496. DOI: 10.1103/PhysRevA.52.R2493. URL: https://link.aps.org/doi/10.1103/PhysRevA.52.R2493.

[20] Joschka Roffe et al. "Decoding across the quantum low-density parity-check code landscape". In: *Physical Review Research* 2.4 (Dec. 2020). ISSN: 2643-1564. DOI: 10.1103/physrevresearch.2.043423. URL: http://dx.doi.org/10.1103/PhysRevResearch.2.043423.

[21] A.Yu. Kitaev. "Fault-tolerant quantum computation by anyons". In: *Annals of Physics* 303.1 (Jan. 2003), pp. 2–30. ISSN: 0003-4916. DOI: 10.1016/s0003-4916(02)00018-0. URL: http://dx.doi.org/10.1016/S0003-4916(02)00018-0.

[22] S. B. Bravyi and A. Yu. Kitaev. *Quantum codes on a lattice with boundary*. 1998. arXiv: quant-ph/9811052 [quant-ph]. URL: https://arxiv.org/abs/quant-ph/9811052.

[23] Richard P. Feynman. "Simulating physics with computers". In: *International Journal of Theoretical Physics* 21.6 (June 1982), pp. 467–488. ISSN: 1572-9575. DOI: 10.1007/BF02650179. URL: https://doi.org/10.1007/BF02650179.

[24] Daniel Gottesman. *The Heisenberg Representation of Quantum Computers*. 1998. arXiv: quant-ph/9807006 [quant-ph]. URL: https://arxiv.org/abs/quant-ph/9807006.

[25] Craig Gidney. "Stim: a fast stabilizer circuit simulator". In: *Quantum* 5 (July 2021), p. 497. ISSN: 2521-327X. DOI: 10.22331/q-2021-07-06-497. URL: https://doi.org/10.22331/q-2021-07-06-497.

3

[26] Craig Gidney and Cody Jones. *New circuits and an open source decoder for the color code*. 2023. arXiv: 2312.08813 [quant-ph]. URL: https://arxiv.org/abs/2312.08813.

[27] Dripto M. Debroy et al. "LUCI in the Surface Code with Dropouts". In: *Quantum* 9 (Dec. 2025), p. 1936. ISSN: 2521-327X. DOI: 10.22331/q-2025-12-11-1936. URL: http://dx.doi.org/10.22331/q-2025-12-11-1936.

A

CODE AND DATASET

The Jupyter notebook used in the making of this thesis together with the datasets it generated can be found at https://github.com/SAJHCamps/Thesis-project-AM.