ABSTRACT

Title of Dissertation: PRACTICAL APPLICATIONS FOR PARTIAL QUANTUM ERROR CORRECTION

Noah Berthusen
Doctor of Philosophy, 2025

Dissertation Directed by: Professor Daniel Gottesman
Department of Computer Science

Quantum computers have the theoretical potential to solve problems intractable for classical computers. However, realizing this potential requires dealing with the noise inherent in near and far-term devices. One way of doing this is to redundantly encode the quantum information in a quantum error-correcting code and manipulate the encoded states to do computation. Protecting quantum information in this way incurs additional space overhead in the form of extra qubits; this is problematic since qubits are a scarce resource, especially for near-term quantum computers. Reducing these overheads could significantly accelerate the arrival of large-scale, fault-tolerant quantum computation.

In this thesis, we address this topic of research and present techniques which aim to practically reduce the space and time overheads of implementing quantum error correction. The overarching motivation for the works presented in this thesis is the belief that it is advantageous, perhaps even essential, to measure every stabilizer generator when performing quantum error correction. To address this claim, we introduce partial quantum error correction, which we broadly

define to be using incomplete syndrome information from the code or neglecting to correct errors on some part of the system. We show that it is *not* necessary to measure every stabilizer generator in order to obtain a threshold, and we will describe several situations where we obtain better logical performance and/or reduced overheads by not doing so.

In particular, we present an error correction protocol built on a bilayer architecture that aims to reduce operational overheads when restricted to 2D local gates by measuring some generators less frequently than others. We show through numerical simulations that high-rate quantum error correcting codes implemented with this protocol achieve logical error rates comparable to the surface code while using fewer physical qubits. We then introduce adaptive syndrome extraction as a scheme to improve code performance and reduce the quantum error correction cycle time by measuring only the stabilizer generators that are likely to provide useful syndrome information. We describe and numerically evaluate a concrete example of the scheme instantiated using a concatenated code and a syndrome extraction cycle that uses quantum error detection to modify the syndrome extraction circuits in real time.

PRACTICAL APPLICATIONS FOR PARTIAL QUANTUM ERROR CORRECTION

by

Noah Berthusen

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
2025

Advisory Committee:
Professor Daniel Gottesman, Chair/Advisor
Professor Michael J. Gullans, Co-chair
Professor Andrew M. Childs
Professor Runzhou Tao
Professor Saikat Guha, Dean's Representative

Acknowledgments

Firstly, I would like to thank my advisor Daniel Gottesman for guidance and support throughout my PhD. Coming to the University of Maryland, I had no prior experience with quantum error correction, and Daniel is one of the best resources to learn from. While Daniel's work is often more theoretical, I am grateful that he supported me in specializing in a more numerical, experimental QEC focus.

I wouldn't have started in quantum computing were it not for Peter Orth. I am grateful he asked me if I wanted to learn quantum computing just as he was beginning as well. Had he not, I'd most likely be doing something less exciting and much less fulfilling. The following years and resulting research were invaluable to building my base as a researcher, and they were likely one of the main reasons why I got into a PhD program in the first place—and was successful during my time at UMD.

I am also grateful to many other professors and students from QuICS who I have had the privilege of working with. Michael Gullans has been great (un)official co-advisor and was a driving factor in getting the work of Chapter 4 started. Alexey Gorshkov was also a big help in producing Chapter 4, and his enthusiasm for the work was always appreciated. I'd also like to thank the QEC team at Quantinuum for having me as an intern during the summers. They, too, gave me great freedom in terms of research direction, and I learned a lot during my time there. Additionally, discussions with Ciarán Ryan-Anderson and Natalie Brown helped bring about the idea on which Chapter 5 is based.

I am grateful to my friends who made my time in Maryland and DC excellent. In particular, I would like to thank Yuelin Liu, Shramay Palta, and Ethan Hickman for being great housemates and filling my first year with good food. I am grateful to Matt Chan, Jon Nelson, and Joel Rajakumar for being good friends practically from the very first research day. R.I.P. Town Hall. The Los Alamos Quantum Computing summer school also brought me new friends including George Umbrarescu and Shi Jie Samuel Tan, the latter of whom was a very good research influence during the final years of my program. I'd like to extend a thanks to my old friends, Simon, Claire, Heidi, Bryce, and Austin for being great climbing partners and making my summers in Colorado great. Additionally, I'd like Dakota, Drew, Izaak, and Grant for making my time back in Iowa enjoyable and being great long-time friends.

Finally, I am grateful for my family for their love and support. Over the past three years I've probably spent more time with my cat Olive than just about anyone else, and given the number of times she has walked across my keyboard during the writing of this thesis and my other papers, she ought to be included as a coauthor. To my parents and grandparents, I am thankful for all the encouragement I have received throughout my life and for instilling in me the work ethic to accomplish this. I would especially like to thank my wonderful fiancé, Kelsey Riemenschneider, for moving across the country for me and supporting me during these exciting four years.

Table of Contents

List of Tables

List of Figures

Chapter 1: Introduction

Quantum computers have the theoretical potential to solve problems intractable for classical computers. Notably, the simulation of quantum many-body systems, as originally proposed by Feynman [1], and integer factorization [2] are believed to provide superpolynomial speed-ups over classical computing. Applying these and other methods could yield advances in material science, drug discovery, optimization, machine learning, cryptography, and more. However, the current generation of quantum computers are are small, with around $\sim 10^2 - 10^3$ qubits, and noisy, with error rates around 0.1% per operation [3–6]. At these scales, we are unable to execute the most promising applications of quantum computing, which may require closer to 10^6 qubits [7] and circuits with millions or billions of operations. In this noisy intermediate-scale quantum (NISQ) era [8], as it has been called, efforts have been instead focused on finding use for quantum computers through variational algorithms [9] and simulations of small physical systems [10, 11]. However, there have not been definitive demonstrations of quantum advantage save for contrived mathematical problems [12, 13]

Coherence times and error rates have significantly improved over the last decade; however, improvements in the hardware are unlikely to provide the error rates required to run quantum algorithms with deep circuits and large gate-counts. To achieve these long circuits and low error rates, it is likely that quantum error correction [14] (QEC) will be required. At a high level, QEC

allow us to redundantly encode quantum information in a subspace of the full 2^n-dimensional Hilbert space and occasionally check to see if errors have caused the information to leave this logical subspace. By performing operations on the degrees of freedom of this highly entangled system, we can achieve universal quantum computation on a noisy quantum computer [15, 16]. The threshold theorem [17–19] guarantees that such a procedure can work for arbitrarily long quantum computations as long as the noise rate of the system is below some threshold.

Encoding quantum information in this way incurs additional space overhead: namely, quantum error correcting codes (QECCs) encode k logical qubits into n physical qubits, where $n > k$. To minimize the total number of qubits required, we would like that the number of logical qubits scales like the number of physical qubits, $k = O(n)$, or the rate k/n is constant as $n \to \infty$. Previously, this was not possible without sacrificing the error correction capabilities of the code, its distance d. However, recent *good* quantum low-density parity-check (qLDPC) code constructions were found which have parameters scaling like $k = d = O(n)$ [20–23]. In addition to the qubits required to simply construct the code, ancilla qubits are needed to make the logical quantum computation fault tolerant. Although polylogarithmic overhead is needed in the general case, it was later shown that the use of these asymptotically good LDPC codes could reduce the required space overhead a constant [24]. Furthermore, there is a time overhead associated with performing encoded quantum computation, with the quantum error correction itself taking significant time, in addition to performing the logical gates on the QECC. Minimizing the time overhead, too, has been focus of recent research [25–28]. Practically, many of these threshold and code construction works have somewhat optimistic constraints on the capabilities of the quantum computer; reality is much more restrictive.

There are a number of potential candidates for large-scale quantum computers, including

superconducting transmon qubits [29], ion-traps [30–32], neutral-atoms [33], photonics [34], bosonic qubits [35, 36], and semiconductor spin qubits [37], among others, each with their own strengths and weaknesses. As such, for each architecture there are certain QECCs that are well-suited and others that are inefficient or even incompatible. Notable examples include the surface code [38, 39], which is ideal for the nearest neighbor connectivity of superconducting qubits, or GKP codes [40], which require the infinite-dimensional Hilbert space of bosonic qubits. On many of these platforms, there have been various experimental realizations of logical quantum memory and logical computation [41–46], including some that exhibit below threshold performance [47–52]. These demonstrations are necessary and encouraging steps towards large-scale FT quantum computing, but we are still most likely several years, or even several decades, away. Further research focused on reducing the required overheads could significantly accelerate this timeline.

In this thesis, we address this topic of research and present techniques which aim to practically reduce the space and time overheads of implementing quantum error correction. The overarching motivation for the works presented in this thesis is the belief that it is advantageous, perhaps even essential, to measure every stabilizer generator when performing quantum error correction. To address this claim, we introduce partial quantum error correction, which we broadly define to be using incomplete syndrome information from the code or neglecting to correct errors on some part of the system. We will show that is *not* necessary to measure every stabilizer generator in order to obtain a threshold, and we will describe several situations where we obtain better logical performance and/or reduced overheads by not doing so.

1.1 Relation to previous work

We briefly discuss the relation to techniques which are either named similarly or share a similar implementation to partial quantum error correction as described in this thesis. Note that there are *many* recent works that aim to reduce the time and space overheads in quantum error correction. Indeed, this is essentially the underlying motivation behind much of all current quantum error correction research.

1.1.1 Partial error correction

Partial (quantum) error correction has been coined before. In particular, Refs. [53, 54] introduced a framework that employs error-corrected "clean" qubits in conjunction with non-error-corrected "dirty" qubits, motivated by the qubit limitations imposed by near-term quantum computers. During quantum computation, clean logical qubits are interacted with dirty qubits in an attempt to increase the computational space of the (small) QECC. They present evidence to suggest that this model offers some benefits over fully noisy circuits.

The difference compared to our version of partial quantum error correction is we assume quantum computation is conducted solely on the clean logical qubits. In other words, all physical qubits used for computation (that is, apart from those used to measure, as flags, perform routing, etc.) are part of some error correcting code. It is just that by *occasionally* neglecting to measure some generators, some qubits become temporarily dirty. However, the scheduling of the generator measurements ensure that no error is ignored for too long, and that all qubits are eventually cleaned of any residual errors. This is in contrast to the partial error correction of Refs. [53, 54], in which the dirty qubits are never cleaned despite the fact that they are used for computation.

1.1.2 Floquet codes

Floquet codes [55] and dynamical codes [56, 57], as their names suggest, are QECCs generated from a sequence of low-weight measurement operators. For both Floquet codes and partial quantum error correction, the measurement schedule may repeat over some number of rounds, but we do not require the round-to-round measurements to be identical. Indeed, this is precisely what enables Floquet codes to function as a quantum memory.

The main difference between the two methods is the location of logical information: for Floquet codes, the logical qubits arise as a consequence of the particular sequence of measurements; whereas for partial quantum error correction, we focus on stabilizer codes with a static logical subspace. Note that in syndrome extraction rounds where a submaximal set of stabilizer generators is measured, the physical qubits are stabilized by fewer generators, and as such are *technically* in a different code than they were originally encoded. This new code will likely have a smaller distance, see Section 3.3, and seeing as it has fewer generators, it encodes more logical qubits. However, these additional logical qubits are not used, and the encoded information remains in the original logical subspace.

1.1.3 Single-shot QEC

Slightly less related is the concept of single-shot QEC. Usually, to be fault-tolerant to both qubit and syndrome errors, the stabilizer generators have to measured $O(d)$ times, and the entire syndrome history must be used in decoding [58]. For quantum error correcting codes with the single-shot [59, 60] property a single round of syndrome extraction suffices. In this way, single-shot codes inherently tolerate syndrome errors. Nonetheless, the syndrome measurements are

required to at least be attempted, especially for codes where the single-shot property is facilitated by soundness [60]. Codes that are single-shot can immediately be applied in a partial quantum error correction scheme in which stabilizer measurements are skipped, with some performance guarantees, see Section 3.3.

1.1.4 Approximate quantum error correction

Similar sounding, but not very related is the concept of approximate quantum error correction [61]. For approximate error correcting codes, we are not required to return exactly to the computational subspace after correction, but instead we are allowed to return to a state that is ϵ-close to a codeword. In this thesis, partial quantum error correction is still exact in the sense that whenever we do corrections, we want them to return us to the codespace.

1.2 Outline

This dissertation is structured as follows:

Chapter 2 presents the necessary background material needed for the following chapters. In particular, we provide introductions to classical and quantum error correction including the codes and decoders used in this work.

In Chapter 3, we introduce partial syndrome measurement and masking as a general technique for quantum error correction. We then motivate an application of partial syndrome measurement inspired by the so-called stacked model, in which generators acting on spatially distant qubits are measured less frequently than those which do not. Through analytical and numerical means, we investigate the performance of a simplified version of this scheme where the measured

generators are randomly selected. This chapter is based on the following publication:

- **Noah Berthusen** and Daniel Gottesman. Partial syndrome measurement for hypergraph product codes. Quantum, 8:1345, May 2024.

In Chapter 4, we study the stacked model application of Chapter 3 in a more realistic setting; namely, we present a full error correction protocol that is built on a bilayer architecture and uses bivariate bicycle codes. In doing so, we discuss code embeddings, a parallel syndrome measurement scheme using fast routing with local operations and classical communication (LOCC), and entanglement purification. This chapter is based on the following publication:

- **Noah Berthusen**, Dhruv Devulapalli, Eddie Schoute, Andrew M. Childs, Michael J. Gullans, Alexey V. Gorshkov, and Daniel Gottesman. Toward a 2D local implementation of quantum LDPC codes. PRX Quantum 6:010306, Jan 2025.

In Chapter 5, we present another application of partial quantum error correction in the form of adaptive syndrome extraction. We show that for certain codes, namely a concatenated code consisting of a error detecting code and a high-rate low-density parity-check code, syndrome extraction could be optimized by utilizing the correlations in the syndromes between concatenation layers. Using this idea, we present a framework that can 'short-circuit' syndrome extraction and skip measuring generators that are unlikely to be useful for decoding. Specifically, we look at $[[4, 2, 2]]$-concatenated hypergraph product codes and investigate the potential benefits of using these codes and the adaptive syndrome extraction in quantum memory and logical computation. This chapter is based on the following preprint:

- **Noah Berthusen**, Shi Jie Samuel Tan, Eric Huang, and Daniel Gottesman. Adaptive Syndrome Extraction. arXiv preprint arXiv:2502.14835, Feb 2025.

Finally, we conclude in Chapter 6 with a discussion and present some potential future applications of partial quantum error correction.

Other work completed during the completion of this degree but not reported in this dissertation include:

- **Noah Berthusen**, Michael J. Gullans, Yifan Hong, Maryam Mudassar, and Shi Jie Samuel Tan. Automorphism gadgets in homological product codes. To appear, 2025.

- **Noah Berthusen**, Joan Dreiling, Cameron Foltz, John P. Gaebler, Thomas M. Gatterman, Dan Gresh, Nathan Hewitt, Michael Mills, Steven A. Moses, Brian Neyenhuis, Peter Siegfried, and David Hayes. Experiments with the four-dimensional surface code on a quantum charge-coupled device quantum computer. Phys. Rev. A, 110:062413, Dec 2024.

- **Noah Berthusen**, Faisal Alam, and Yu Zhang. Multi-reference Quantum Davidson Algorithm for Quantum Dynamics. arXiv preprint arXiv:2406.08675, Jun 2024.

Chapter 2: Preliminary material

In this chapter we present an introduction to the material that will be used throughout the thesis. In particular, we provide introductions to classical and quantum error correction, including the codes, error models, and general decodering scenarios used in this work.

2.1 Classical error correction

An $[n, k, d]$ binary linear code $\mathcal{C}$ encodes k classical bits in a k-dimensional subspace of the n bit, n-dimensional space, $\mathbb{F}_2^n$. Codewords are the binary vectors $v \in \mathbb{F}_2^n$ that satisfy the equation $H \cdot v = 0$, where H is a binary matrix of size $m \times n$ called the *parity check matrix* (pcm) and arithmetic is done over $\mathbb{F}_2$. The number of logical bits k can be computed from H like $k = n - \text{rk}(H)$, where $\text{rk}(H)$ denotes the rank of H. As an example, the matrix shown below in Eq. (2.1) is the pcm for the $[7, 4, 3]$ Hamming code.

$$H = \begin{pmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{pmatrix} \tag{2.1}$$

The distance d of a binary linear code is the minimum weight of a non-zero codeword. The weight, or Hamming weight, of a vector $v \in \mathbb{F}_2^n$ is simply the number of ones in v and is denoted

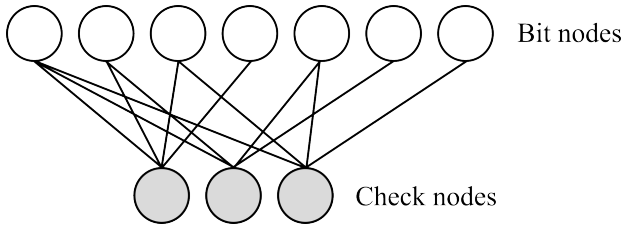

Figure 2.1: Tanner graph for the $[7, 4, 3]$ Hamming code.

by $|v|$. An $[n, k, d]$ error correcting code is able to correct all errors of weight up to $\lfloor (d-1)/2 \rfloor$, and it is able to detect all errors of weight up to $d - 1$. Note that there may be higher weight errors that are detectable and correctable, but it is not guaranteed in general.

The parity check matrix can be represented as a bipartite graph $G = (V \sqcup C, E)$ called a *Tanner graph*. For a $[n, k, d]$ binary linear code, there are two sets of nodes in the graph: the bit nodes V, with $|V| = n$, and the check nodes C, with $|C| = m$. A bit node $v \in V$ and a check node $c \in C$ are connected by an edge if v is involved in the check c. In other words, the ith bit is connected to the jth check if $H_{j,i} = 1$. In this sense, the Tanner graph can be considered the graph whose biadjacency matrix is H. Fig. 2.1 shows the Tanner graph for the $[7, 4, 3]$ Hamming code. From a Tanner graph $\mathcal{G}$ of some binary linear code $\mathcal{C}$, we can obtain the transpose code $\mathcal{C}^T$ by swapping the bit and check nodes of $\mathcal{G}$. That is, the Tanner graph remains the same, but the bit nodes are now considered check nodes and the check nodes are now considered bit nodes. Equivalently, the pcm of $\mathcal{C}^T$ is H^T. The resulting code has parameters $[m, k^T, d^T]$. Note that if H is full rank then $\mathcal{C}^T$ consists of only the zero codeword, and so $k^T = 0$ and $d^T = \infty$.

To determine whether a received message w experienced an error during transmission we compute its *syndrome*, $\sigma(w) = H \cdot w$. Since all codewords of $v \in \mathcal{C}$ are elements of the kernel of H, a non-zero outcome for $\sigma(w)$ indicates that an error has occurred. Fig. 2.2 shows two example errors on the $[7, 4, 3]$ Hamming code and the corresponding syndromes: the error depicted in

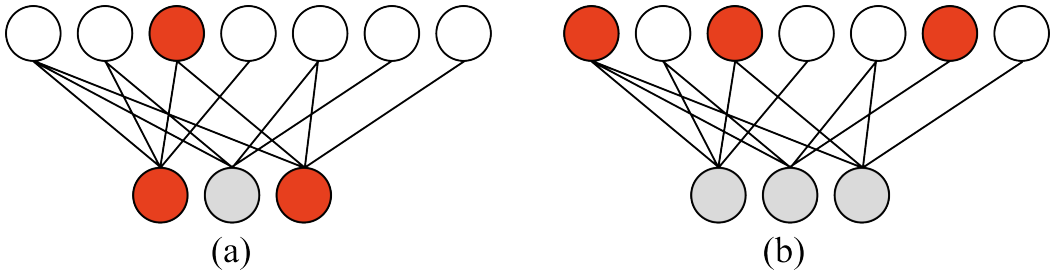

Figure 2.2: Example errors and syndromes for the $[7, 4, 3]$ Hamming code. In panel (b), a weight-three error causes a zero syndrome, indicating that the distance of the code is at most three.

panel (a) has a non-zero syndrome, and since $|w| \leq \lfloor (d-1)/2 \rfloor = 1$ we are able to uniquely identify the error. In panel (b), a weight-three error causes an all-zero syndrome, indicating that the distance of the code is at most three. For this specific code, there is no lower-weight error that causes an all-zero syndrome, and so the distance is three. Determining the smallest satisfying error, or equivalently the closest codeword to the received vector, is difficult in general [62]. This process, called *decoding*, is instead usually done using heuristic algorithms called *decoders*. We further discuss the decoding problem in Section 2.3.5. Once the decoder outputs a correction, it can be applied to the received message to (hopefully) obtain the originally sent message.

A classical code is considered a *low-density parity-check* (LDPC) code [63] if the weights of the rows and columns of its parity check matrix are bounded by a constant. We say that a code is (Δ_V, Δ_C)-LDPC if the weight of every column and row of H is bounded by $\Delta_V, \Delta_C \in O(1)$, respectively. We can equivalently say that the Tanner graph has bit node degree bounded by Δ_V and check node degree bounded by Δ_C. Due to the sparsity of their Tanner graphs, LDPC codes are amenable to efficient decoding by message-passing algorithms like belief propagation [64], allowing them to approach the Shannon limit [65]. Their performance and simplicity have caused them to find widespread use in communications including Wi-Fi, Ethernet, 5G, and more.

2.2 Quantum information

The fundamental unit of classical computation is the bit, which can be in the 0 and 1 states. The fundamental unit of computation of quantum computation, the *qubit*, can also be in discrete states called basis states:

$$|0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \quad |1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix}. \tag{2.2}$$

However, the qubit can also be in a *superposition* of $|0\rangle$ and $|1\rangle$,

$$|\psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle = \alpha \begin{pmatrix} 1 \\ 0 \end{pmatrix} + \beta \begin{pmatrix} 0 \\ 1 \end{pmatrix} = \begin{pmatrix} \alpha \\ \beta \end{pmatrix}, \tag{2.3}$$

where $\alpha, \beta \in \mathbb{C}$, and we have the condition $|\alpha|^2 + |\beta|^2 = 1$. From this it can be seen that a single qubit is just a normalized vector in $\mathbb{C}^2$. Similarly, a system of n qubits can be represented as a statevector $|\psi\rangle \in (\mathbb{C}^2)^{\otimes n} = \mathbb{C}^{2^n}$ which we call a pure state. Alternatively, we can represent the qubit state with its density matrix $\rho = |\psi\rangle\langle\psi|$. The density matrix representation allows us to also consider mixed states, which are a statistical mixture of pure states,

$$\rho = \sum_i p_i\,|\psi_i\rangle\langle\psi_i|, \tag{2.4}$$

where $\{p_i\}$ is a valid probability distribution.

Just as classical computation can be expressed using logic gates such as OR and NOT, quantum computation is accomplished by applying unitary operator U, i.e. operators satisfying $UU^\dagger = U^\dagger U = I$, where I is the identity matrix. When a unitary gate U is applied to an initial

state $|\psi\rangle$, we obtain a transformed state $|\phi\rangle = U|\psi\rangle$, which can simply be obtained by matrix multiplication. We can also express this transformation on the density matrix,

$$\sigma = |\phi\rangle\langle\phi| = U|\psi\rangle\langle\psi|U^\dagger = U\rho U^\dagger. \tag{2.5}$$

Of course, computing the action on a n qubit system will, in general, require keeping track of the statevector $|\psi\rangle \in \mathbb{C}^{2^n}$ and performing matrix multiplication with an operator $U \in \mathbb{C}^{2^n \times 2^n}$. For even a modest number of qubits this method is classically intractable, hence the potential for quantum advantage. One important set of unitary operators are the Pauli operators:

$$X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \quad Y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix} \quad Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \tag{2.6}$$

which when tensored over n qubits and phases are included form the Pauli group:

$$\mathcal{P}_n = \{I, X, Y, Z\}^{\otimes n} \times \{\pm 1, \pm i\}. \tag{2.7}$$

In quantum error correction, it often suffices to only consider errors coming from the Pauli group. This simplification is valid due to a discretization of errors [66] that occurs during quantum measurements. Indeed, arbitrary-angle rotation errors get converted into Pauli errors which can then be corrected with appropriate application of Pauli corrections.

2.2.1 Error models

During a quantum computation, the system of qubits is interacting with the external environment and may experience unintended noise and decoherence. The most general way to model noise is with a completely positive trace preserving (CPTP) map. We can represent a CPTP map with a set of Kraus operators

$$\mathcal{E}(\rho) = \sum_i E_i \rho E_i^\dagger, \tag{2.8}$$

where $\sum_i E_i E_i^\dagger = I$. Perhaps the simplest noise models are the bit- and phase-flip (dephasing) channels:

$$\mathcal{B}_p(\rho) = (1-p)\rho + pX\rho X^\dagger \qquad \mathcal{P}_p(\rho) = (1-p)\rho + pZ\rho Z^\dagger \tag{2.9}$$

Additional error models include amplitude damping, qubit loss, or the depolarizing channel, which is shown below:

$$\mathcal{D}_p(\rho) = (1-p)\rho + \frac{p}{3}X\rho X^\dagger + \frac{p}{3}Y\rho Y^\dagger + \frac{p}{3}Z\rho Z^\dagger. \tag{2.10}$$

Throughout this thesis, when we say an error has occurred the qubits, we mean that one of these CPTP maps (mainly the bit-flip and depolarizing channels) has been applied to the system.

2.3 Quantum error correction

An $[[n, k, d]]$ quantum error correcting code (QECC) $\mathcal{Q}$ encodes k logical qubits into a 2^k-dimensional subspace of the n qubit, 2^n-dimensional Hilbert space, $(\mathbb{C}^2)^{\otimes n} = \mathbb{C}^{2^n}$. A commonly used class of QECCs are *stabilizer codes* [67, 68]. A stabilizer code is defined by its *stabilizer* $\mathcal{S}$,

X	Z	Z	X	I
I	X	Z	Z	X
X	I	X	Z	Z
Z	X	I	X	Z

Table 2.1: Stabilizer generators for the $[[5, 1, 3]]$ five-qubit code.

consisting of elements of the Pauli group Eq. (2.7) whose action is the identity on the codewords $|\psi\rangle$ of $\mathcal{Q}$, e.g.,

$$\mathcal{Q} = \{S\,|\psi\rangle = |\psi\rangle\,, \forall S \in \mathcal{S}\}. \tag{2.11}$$

In other words, the quantum codewords $|\Psi\rangle \in \mathcal{Q}$ form the joint $+1$-eigenspace for the elements of $\mathcal{S}$. To have a codespace at all, we require that $-I \notin \mathcal{S}$ and that $\mathcal{S}$ forms an abelian subgroup of $\mathcal{P}_n$. $\mathcal{S}$ is generated by m independent *stabilizer generators* $\mathcal{S} = \langle S_1, ..., S_m\rangle$, and the corresponding stabilizer code encodes $n-m = k$ logical qubits. As an example, consider the $[[5, 1, 3]]$ five-qubit code [69, 70], the smallest stabilizer code that is able to correct an arbitrary single-qubit error. Its four independent stabilizers generators are shown in Table 2.1; as such, it encodes $5 - 4 = 1$ logical qubit. Denote by $N(\mathcal{S})$ the normalizer of $\mathcal{S}$, the set of Pauli operators that commute with everything in the stabilizer, $N(\mathcal{S}) = \{N \in \mathcal{P}_n \mid [N, M] = 0\ \forall M \in \mathcal{S}\}$. The distance d of $\mathcal{Q}$ is then defined to be the minimum weight of an operator in $N(\mathcal{S}) \setminus \mathcal{S}$. For the five-qubit code, it can be easily checked that the Pauli operator $P = X_1Z_3X_5$ commutes with all of the stabilizer generators. It is also straightforward to verify that P cannot be written as a product of the stabilizer generators, and so $P \in N(\mathcal{S}) \setminus \mathcal{S}$. Hence the distance of the code is at most $|P| = 3$; in this case, the distance is exactly three. In order to do computation on the encoded qubits, we identify the logical Pauli group $N(\mathcal{S})/\mathcal{S}$, which is isomorphic to the Pauli group on k qubits, $\mathcal{P}_k$. For any stabilizer code, we can find a basis of logical operators $\overline{X}_1, \overline{Z}_1, ..., \overline{X}_k, \overline{Z}_k$ which generate the Pauli group on the k logical qubits.

To determine whether the encoded quantum information has left the logical subspace, the eigenvalues of the stabilizer generators are measured. There are several ways to do this. The circuits depicted in Fig. 2.3 provide one of the most straightforward approaches, which we use throughout the thesis. Eq. (2.11) states that in the absence of errors, all generators will have a $+1$ eigenvalue; whereas a -1 eigenvalue indicates that an error has caused the encoded information to leave the codespace. Note that obtaining all $+1$ measurement results does not guarantee an error-free codespace: if $E \in N(\mathcal{S}) \setminus \mathcal{S}$, then by definition E commutes with everything in $\mathcal{S}$, yet is an unintended error. These *logical errors* are especially detrimental since they cause logical actions on one or more of the logical qubits. Let $E \in \mathcal{P}_n$ be some Pauli error on the n qubits. Since Paulis either commute or anticommute, there are only two options for each of the stabilizer generators of $\mathcal{S}$. If $[E, S_i] = 0$, then

$$S_i E \left|\psi\right\rangle = E S_i E \left|\psi\right\rangle = E \left|\psi\right\rangle , \tag{2.12}$$

i.e. $E \left|\psi\right\rangle$ is a $+1$-eigenstate of S_i. Hence, performing the circuit in Fig. 2.3 would yield a measurement result of 0. The other scenario is if $\{E, S_i\} = 0$; in this case

$$S_i E \left|\psi\right\rangle = -E S_i \left|\psi\right\rangle = -E \left|\psi\right\rangle , \tag{2.13}$$

and so $E \left|\psi\right\rangle$ is -1-eigenstate of S_i. Measuring the corresponding eigenvalue using the circuit in Fig. 2.3 would yield a result of 1. These measurement results constitute a classical syndrome $\sigma(E) \in \mathbb{F}_2^m$, which is then used as input to a decoding algorithm that outputs a correction, see Section 2.3.5. We note that the syndrome labels the 2^m cosets of $\mathcal{P}_n / N(\mathcal{S})$.

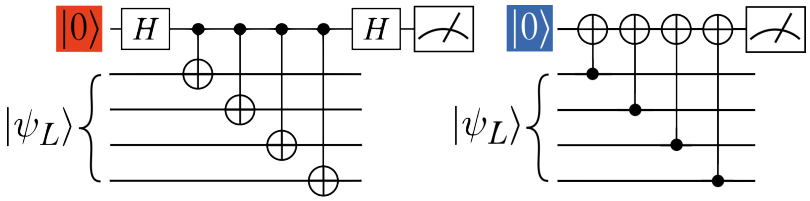

Figure 2.3: Circuits for measuring the eigenvalue of an X-type generator (red) and a Z-type generator (blue). The Z-type measurement presented here is a variation from the standard circuit which uses CZ gates.

The *binary symplectic representation* of a Pauli $P \in \mathcal{P}_n/\{\pm 1, \pm i\} = \hat{\mathcal{P}}_n$ is a bitstring consisting of two n-bit binary vectors, $(x|z) \in \mathbb{F}_2^{2n}$. The ith component of x is 0 if P acts on qubit i with I or Z and 1 if P acts on qubit i with X or Y. Similarly, the ith component of z is 0 if P acts on qubit i with I or X and 1 if P acts on qubit i with Z or Y. In other words, $\hat{\mathcal{P}}_n \cong \mathbb{F}_2^{2n}$. This transformation allows us to use techniques from classical coding theory on QECCs. In particular, we can represent the stabilizer generators as a $m \times 2n$ binary parity check matrix, H. For example, the binary symplectic representation of the stabilizer generators of the five-qubit code, Table 2.1, is then:

$$H = \left(\begin{array}{ccccc|ccccc} 1 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 \end{array}\right) \tag{2.14}$$

The *symplectic product* on two Pauli strings in the symplectic form is defined as $(x_1|z_1) \odot (x_2|z_2) = x_1 \cdot z_2 + x_2 \cdot z_1$, where multiplication and addition are performed over $\mathbb{F}_2$. There is a natural equivalence between the symplectic product and the commutativity of Pauli opera-

tors: let $P_1 = (x_1|z_1)$ and $P_2 = (x_2|z_2)$, then

$$[P_1, P_2] = 0 \iff P_1 \odot P_2 = 0. \tag{2.15}$$

And similarly when $[P_1, P_2] = 1$. It can be easily verified that the generators of the five-qubit code commute by applying the symplectic product.

CSS codes [71, 72] are a subclass of stabilizer codes where the stabilizer generators consist entirely of tensor products of X and I or Z and I. As such, these codes have a parity check matrix with the following symplectic representation:

$$H = \left(\begin{array}{c|c} H_X & 0 \\ \hline 0 & H_Z \end{array}\right). \tag{2.16}$$

Here H_X and H_Z are $m_X \times n$ and $m_Z \times n$ binary matrices, respectively, that satisfy $H_Z \cdot H_X^T = H_X \cdot H_Z^T = 0$. Note that it is not necessary for $m_X = m_Z$: an example in which $m_X \neq m_Z$ is the 3D surface code [73]. CSS codes encode $k = n - \text{rk}(H_X) - \text{rk}(H_Z)$ logical qubits. With the pcm in this form, it can be seen that error correction with CSS codes can be broken down into correcting X- and Z- type errors separately, with H_X detecting and correcting Z-type errors, and H_Z detecting and correcting X-type errors. Hence we obtain separate syndromes when decoding an error $E = (x|z)$,

$$\sigma(E) = (\sigma_Z(x), \sigma_X(z)) = (H_Z \cdot x, H_X \cdot z). \tag{2.17}$$

Decoding CSS codes is greatly simplified compared to general stabilizer codes, since $\sigma_Z(x)$ and $\sigma_X(z)$ can be treated as the syndromes for classical codes with pcms H_Z, H_X, respectively. As such, we can apply efficient classical decoders—with some modifications, see Section 2.3.5. We

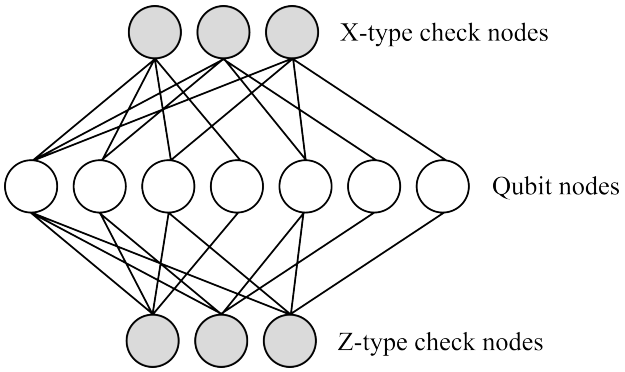

Figure 2.4: Tanner graph for the Steane code.

exclusively focus on CSS codes throughout this thesis.

We can also represent a CSS code with a Tanner graph. For a $[[n, k, d]]$ CSS code, $\mathcal{Q}$, there are now three sets of nodes in its Tanner graph $\mathcal{G}_\mathcal{Q} = (V_\mathcal{Q} \sqcup C_X \sqcup C_Z, E)$: the qubit nodes $V_\mathcal{Q}$, with $|V_\mathcal{Q}| = n$, the X-type check nodes C_X, with $|C_X| = m_X$, and the Z-type check nodes C_Z, with $|C_Z| = m_Z$. Edges are connected between $V_\mathcal{Q}$ and C_X according to H_X, and edges are connected between $V_\mathcal{Q}$ and C_Z according to H_Z. In other words, the symplectic representation of the pcm of $\mathcal{Q}$ is the biadjacency matrix of $\mathcal{G}_\mathcal{Q}$. As an example, consider the $[[7, 1, 3]]$ Steane code [74], a CSS code which is obtained by setting both H_X and H_Z to be the pcm for the $[7, 4, 3]$ Hamming code, Eq. (2.1). Fig. 2.4 shows the Tanner graph for the Steane code. Alternatively, we can represent a quantum code $\mathcal{Q}$ using its connectivity graph $\mathcal{G}'_\mathcal{Q} = (V_\mathcal{Q}, E)$. Here each vertex corresponds to a qubit in $\mathcal{Q}$, and two vertices are connected if the two qubits are in the support of the same stabilizer generator.

Quantum codes can also be (quantum) LDPC. Specifically, an $[[n, k, d]]$ stabilizer code is $(\Delta_V, \Delta_C)-$qLDPC if, for some constants Δ_V and Δ_C, each qubit is involved in at most Δ_V stabilizer generators and each generator measures at most Δ_C qubits. We can equivalently say that in the Tanner graph, each qubit node $v \in V$ has degree bounded by Δ_V and each check node

$c \in C_X \sqcup C_Z$ has degree bounded by Δ_C. For stabilizer codes, the qLDPC property is important because it means that the syndrome extraction circuits shown in Fig. 2.3 can be done in constant time. This ensures that errors cannot build up overwhelmingly during syndrome extraction which, in part, allows qLDPC codes to achieve fault-tolerant quantum computation with constant overhead [24]. Recent years have seen enormous research efforts focused on constructing qLDPC codes with better performance and better parameters [75].

2.3.1 Hypergraph product codes

Hypergraph product (HGP) codes [76] are are CSS-type codes constructed by taking the graph product of two classical linear binary codes $\mathcal{C}_1, \mathcal{C}_2$. Equivalently, they can be considered the homological product [77], or the tensor product, of the chain complex and cochain complex corresponding to the two classical codes. Let $\mathcal{C}_1$ be a binary linear code with parameters $[n_1, k_1, d_1]$, a pcm H_1, and a Tanner graph $\mathcal{G}_1$. Let $\mathcal{C}_1^T$ be the transposed code of $\mathcal{C}_1$ with parameters $[m_1, k_1^T, d_1^T]$, pcm H_1^T, and Tanner graph $\mathcal{G}_1^T$. Similarly define $\mathcal{C}_2$ and $\mathcal{C}_2^T$, with parameters $[n_2, k_2, d_2], [m_2, k_2^T, d_2^T]$, pcms H_2, H_2^T, and Tanner graphs $\mathcal{G}_2, \mathcal{G}_2^T$, respectively. The HGP code obtained by using seed codes $\mathcal{C}_1, \mathcal{C}_2$, $HGP(H_1, H_2)$, has the following pcms:

$$H_Z = \left(I_{n_1} \otimes H_2,\ H_1^T \otimes I_{m_2}\right) \qquad H_X = \left(H_1 \otimes I_{n_2},\ I_{m_1} \otimes H_2^T\right), \tag{2.18}$$

and parameters

$$[[n_1 n_2 + m_1 m_2, k_1 k_2 + k_1^T k_2^T, \min(d_1, d_2, d_1^T, d_2^T)]]. \tag{2.19}$$

When H_1 and H_2 are full-rank, then $k_1^T = k_2^T = 0$ and $d_1^T = d_2^T = \infty$, and so the parameters of the resulting HGP code, $HGP(H_1, H_2)$, reduce to

$$[[n_1 n_2 + m_1 m_2, k_1 k_2, \min(d_1, d_2)]]. \tag{2.20}$$

Furthermore, when $\mathcal{C} := \mathcal{C}_1 = \mathcal{C}_2$ is a binary linear code with parameters $[n, k, d]$ and a full-rank parity check matrix H, the parameters of the resulting HGP code, $HGP(H, H)$, further reduce to $[[n^2 + m^2, k^2, d]]$. A HGP code which is formed from two copies of a single classical code H is called a *square* HGP code. In this thesis, we exclusively investigate square HGP codes where H is full-rank. Note that if the input classical code is $(\Delta_V, \Delta_C)-$LDPC with $\Delta_V \leq \Delta_C$, then the resulting quantum code is $(2\Delta_C, \Delta_V + \Delta_C)-$qLDPC.

One notable instance of the hypergraph product is the surface code [38, 39] and toric code [18], which is formed from the product of two $[n, 1, n]$ repetition codes. The two different constructions are obtained depending on whether the pcm for the repetition code is full rank (yielding the surface code) or not (yielding the toric code). Another notable family of HGP codes are those formed from classical expander codes [78], whose parameters scale like $[n, O(n), O(n)]$. The resulting quantum codes are deemed *quantum expander codes* [79] and have parameters scaling like $[[n, O(n), O(\sqrt{n})]]$. Additionally, they are equipped with a linear time decoder [79], single-shot decoding [80], and single-shot state preparation [81]. These codes are the focus of Chapter 3 and Chapter 5.

We first give the graphical interpretation of the hypergraph product. Again consider the HGP code $\mathcal{Q}$ obtained by using seed codes $\mathcal{C}_1, \mathcal{C}_2$. The Tanner graph $\mathcal{G}_\mathcal{Q}$ of the HGP code is then obtained by performing the graph product of $\mathcal{G}_1$ and $\mathcal{G}_2$. Intuitively, this means that when the

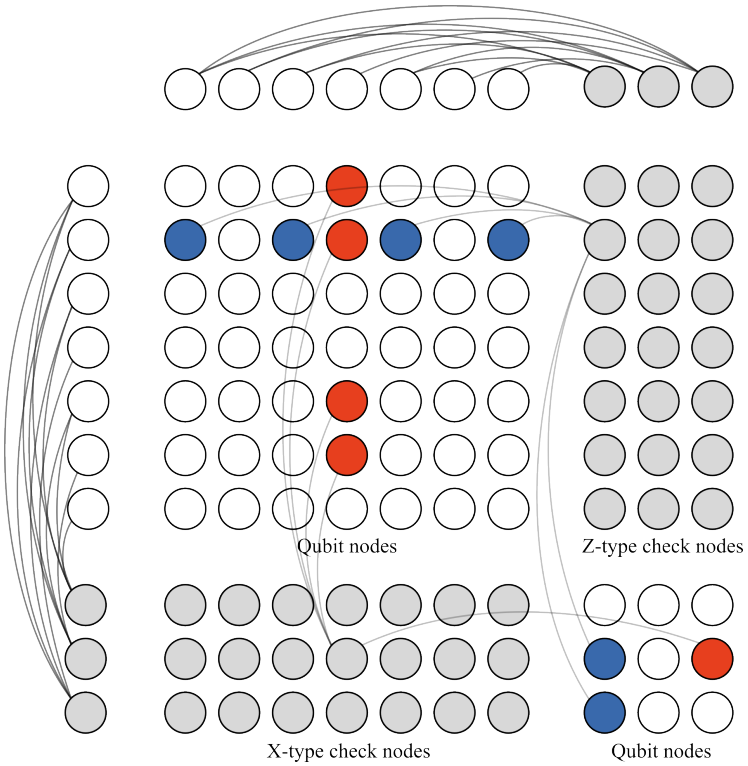

Figure 2.5: Tanner graph of a $[[58, 16, 3]]$ hypergraph product code constructed from two copies of the $[7, 4, 3]$ Hamming code. A Z-type generator (blue) and an X-type generator (red) are highlighted.

qubits and checks are laid out as shown in Fig. 2.4, each row is a copy of the nodes and edges from $\mathcal{G}_1$, and each column is a copy of the nodes and edges from $\mathcal{G}_2$. More formally, let V_1, V_2 be the bit nodes, and let C_1, C_2 be the check nodes for $\mathcal{G}_1, \mathcal{G}_2$, respectively. Then the qubits and checks of $\mathcal{Q}$ are assigned as follows: the qubits $V := V_1 \times V_2 \sqcup C_1 \times C_2$, the X-type checks $C_X := V_1 \times C_2$, and the Z-type checks $C_Z := C_1 \times V_2$. This graphical interpretation provides a convenient representation for the stabilizer generators and logical operators, see Section 2.3.1.1.

An alternative interpretation to the graphical hypergraph product can be obtained through the homological product [77]. We first give a brief review of chain complexes and $\mathbb{F}_2$ homology.

A *chain complex* is a collection of vector spaces over $\mathbb{F}_2$ together with linear maps ∂_i,

$$C = C_n \xrightarrow{\partial_n} C_{n-1} \xrightarrow{\partial_{n-1}} \cdots \xrightarrow{\partial_2} C_1 \xrightarrow{\partial_1} C_0, \tag{2.21}$$

where $\partial_{i+1}\partial_i = 0$. We refer to elements of C_i as i-chains, elements of $Z_i(C) = \ker \partial_i$ as i-cycles, and elements of $B_i(C) = \text{im } \partial_{i+1}$ as i-boundaries. The i-th homology is then defined as the vector space of i-cycles modulo i-boundaries,

$$H_i(C) = Z_i(C)/B_i(C). \tag{2.22}$$

Similarly, we also define elements of $Z^i(C) = \ker \partial_{i+1}^T$ as i-cocycles, elements of $B^i(C) = \text{im } \partial_i^T$ as i-coboundaries, and the i-th cohomology, $H^i(C) = Z^i(C)/B^i(C)$.

An $[n, k, d]$ classical error correcting code can be considered a 2-term chain complex, $C = C_1 \xrightarrow{\partial_1} C_0$, where its boundary map $\partial_1 = H$ is a linear map from $\mathbb{F}_2^n$ to the vector space of syndromes, $\mathbb{F}_2^m$. A CSS code can similarly be represented as a 3-term chain complex,

$$C = C_2 \xrightarrow{\partial_2} C_1 \xrightarrow{\partial_1} C_0, \tag{2.23}$$

where $\partial_2 = H_Z^T$ and $\partial_1 = H_X$. The condition that $\partial_{i+1}\partial_i = 0$ translates to the requirement that the X and Z checks must commute, e.g. $H_Z^T H_X = 0$, and so Eq. (2.23) defines a valid CSS code. Given an arbitrary length chain complex, one may define a CSS code by only considering two consecutive boundary operators. When identifying qubits with elements of C_i, the resulting code parameters are $n = \dim C_i$, $k = \dim H_i(C) = \dim H^i(C)$, and d is the minimum Hamming weight of a non-trivial element in $H_i(C)$ or $H^i(C)$.

Quantum error correcting codes, i.e. 3-term chain complexes, can be obtained by taking the homological [77], or hypergraph [76] product of two chain complexes. The difference between the two products is subtle and concerns the use of a transpose Tanner graph, D in Eq. (2.28). The construction described here corresponds to the hypergraph product. The first step of the product is to take the *double complex* of two chain complexes, $C \boxtimes D$, which is equipped with vertical boundary maps $\partial_i^v = \partial_i^C \otimes \mathbb{I}_{D_i}$ and horizontal boundary maps $\partial_i^h = \mathbb{I}_{C_i} \otimes \partial_i^D$. Here we use the notation ∂_i^A to denote the ith boundary operator of the chain complex A. See Fig. 2.6 for an example of a double complex arising from the tensor product of two 2-term chain complexes. From a double complex, we associate the *total complex* by performing the direct sum over vector spaces and boundary maps of like dimension

$$\mathrm{Tot}(C \boxtimes D)_i = \bigoplus_{i=j+k} C_j \otimes D_k = E_i \tag{2.24}$$

$$\partial_i^E = \bigoplus_{i=j+k} \partial_j^v \oplus \partial_k^h. \tag{2.25}$$

The resulting chain complex, deemed the *tensor product* of C and D, $C \otimes D$, can be used to construct a CSS code by choosing some consecutive three-term sequence. The parameters of the resulting code can be explicitly calculated or the Künneth formula can be applied,

$$H_i(C \otimes D) \cong \bigoplus_{i=j+k} H_j(C) \otimes H_k(D). \tag{2.26}$$

While the distance of a code is in general difficult to calculate, it was shown in Ref. [82] that the distances of a tensor product of an arbitrary-length chain complex with a 2-term chain complex

$$\begin{array}{ccc} C_1 \otimes D_1 & \xrightarrow{\mathbb{I}_{n_1} \otimes H_2^T} & C_1 \otimes D_0 \\ \downarrow{\scriptstyle H_1 \otimes \mathbb{I}_{m_2}} & & \downarrow{\scriptstyle H_1 \otimes \mathbb{I}_{n_2}} \\ C_0 \otimes D_1 & \xrightarrow{\mathbb{I}_{m_1} \otimes H_2^T} & C_0 \otimes D_0 \end{array}$$

Figure 2.6: A commutative diagram representing the double complex that arises from the tensor product of two 2-term chain complexes.

can be calculated exactly,

$$d_i(C \otimes D) = \min\big(d_i(C)d_0(D), d_{i-1}(C)d_1(D)\big). \tag{2.27}$$

Now, we present an explicit construction of a hypergraph product code using the homological product. We define the following two 2-term chain complexes C, D, representing the two classical codes $\mathcal{C} = [n_1, k_1, d_1], \mathcal{D} = [n_2, k_2, d_2]$, respectively, which are the basis for the product construction,

$$C = C_1 \xrightarrow{H_1} C_0 \qquad D = D_1 \xrightarrow{H_2^T} D_0. \tag{2.28}$$

Note that we are using H_2^T as the boundary map for D, so D is really the cochain complex of $\mathcal{D}$. To obtain the hypergraph product of $\mathcal{C}$ and $\mathcal{D}$, we take the tensor product of C and D, yielding the double complex shown in Fig. 2.6. Performing a direct sum over vector spaces of equal dimension according to Eq. (2.24) results in the tensor product complex

$$C_1 \otimes D_1 \xrightarrow{\partial_2} C_0 \otimes D_1 \oplus C_1 \otimes D_0 \xrightarrow{\partial_1} C_0 \otimes D_0 \tag{2.29}$$

where

$$\partial_2 = \begin{pmatrix} \mathbb{I}_{n_1} \otimes H_2^T \\ H_1 \otimes \mathbb{I}_{m_2} \end{pmatrix} \tag{2.30}$$

$$\partial_1 = \begin{pmatrix} H_1 \otimes \mathbb{I}_{n_2}, & \mathbb{I}_{m_1} \otimes H_2^T \end{pmatrix}. \tag{2.31}$$

It can be easily verified that $\partial_1 \partial_2 = 0$, and so Eq. (2.29) is a valid chain complex. Let us relabel the vector spaces and boundary maps in Eq. (2.29) to $E = E_2 \xrightarrow{\partial_2^E} E_1 \xrightarrow{\partial_1^E} E_0$. For each vector space E_i, we calculate the dimension $\dim E_i$.

$$\dim E_2 = \dim(\mathbb{F}_2^{n_1} \otimes \mathbb{F}_2^{m_2}) = n_1 m_2 \tag{2.32}$$

$$\begin{aligned} \dim E_1 &= \dim(\mathbb{F}_2^{m_1} \otimes \mathbb{F}_2^{m_2}) \\ &\quad + \dim(\mathbb{F}_2^{n_1} \otimes \mathbb{F}_2^{n_2}) = n_1 n_2 + m_1 m_2 \end{aligned} \tag{2.33}$$

$$\dim E_0 = \dim(\mathbb{F}_2^{m_1} \otimes \mathbb{F}_2^{n_2}) = m_1 n_2 \tag{2.34}$$

To consider E a CSS code, we identify the $n_1 n_2 + m_1 m_2$ physical qubits with 1-chains. Parity check matrices are then assigned to the boundary operators $\partial_2^E = H_Z^T$ and $\partial_1^E = H_X$, matching the pcms defined in Eq. (2.18). We can determine the number of logical qubits by calculating the dimension of the first homology group, $\dim H_1(E)$ with the Künneth formula, Eq. (2.26). Here, $H_1(C) \cong \mathbb{Z}_{k_1}$, and $H_0(C) \cong \mathbb{Z}_{k_1^T}$. Similarly, $H_1(D) \cong \mathbb{Z}_{k_2^T}$ and $H_0(D) \cong \mathbb{Z}_{k_2}$.

Plugging into Eq. (2.26) as expected yields,

$$\dim H_1(E) = \dim H_1(C \otimes D) \tag{2.35}$$

$$= \dim \big(H_1(C) \otimes H_0(D) \oplus H_0(C) \otimes H_1(D)\big) = k_1 k_2 + k_1^T k_2^T.$$

We can use Eq. (2.27) to calculate the distances of the resulting chain complex. The distance d_0 of the homology group $H_0(C)$ is $d_0 = 1$, unless ∂_1 has full column-rank in which case $d_0 = \infty$. For $i > 0$, we say $d_i = \infty$ if $H_i(C)$ is trivial. Otherwise, d_i is the minimum weight of a non-zero vector $x \in \ker \partial_i$, i.e., the distance of binary linear code defined by ∂_i. Hence we obtain,

$$d_1(E) = \min\big(d_1(C)\, d_0(D), d_0(C)\, d_1(D)\big) \tag{2.36}$$

$$= \min\big(d_1 \cdot 1, 1 \cdot d_2^T\big) \tag{2.37}$$

Technically, this is only d_Z, and $d = \min(d_Z, d_X)$. d_X can be obtained by also taking the minimum Hamming weight of non-zero elements in the cohomology classes, $H^i(C)$ and applying Eq. (2.36). With this, we obtain the same distance as noted in Eq. (2.19).

2.3.1.1 Canonical logical basis

The physical qubits of HGP(H, H) can be arranged into two square grids of size $n \times n$ and $m \times m$, see Fig. 2.4 and Fig. 2.7. In this representation, the stabilizer generators and logical operators have a convenient geometric structure. For each physical qubit in the code, we assign a triplet (i, j, L) or (k, ℓ, R) where $1 \leq i, j \leq n$ and $1, \leq k, \ell \leq m$. Here, i (k) denotes the

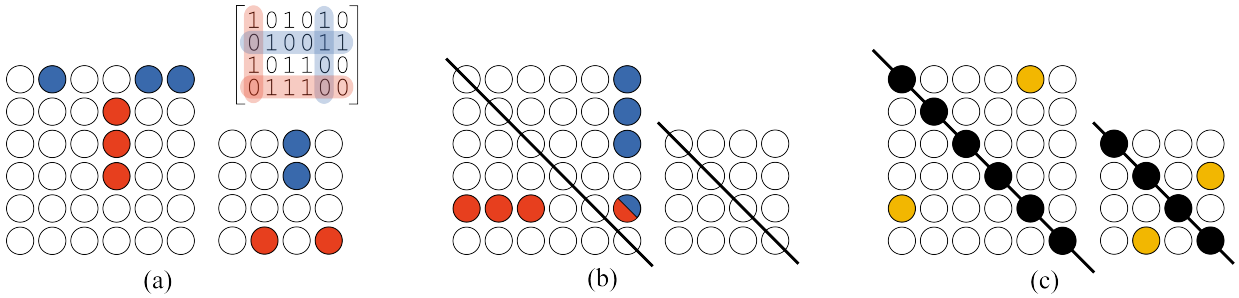

Figure 2.7: Partitioning the qubits of a $[[52, 4, 4]]$ HGP code formed from two copies of a $[6, 2, 4]$ classical code. (a) Z-type generator $S_Z(1, 3)$ (blue) and X-type generator $S_X(4, 4)$ (red) obtained by taking rows and columns of the (transpose) parity check matrix of the classical code. (b) Canonical logical operators $\overline{Z}_2$ and $\overline{X}_2$. The qubit highlighted both blue and red is the pivot qubit for that logical pair. (c) Diagonal qubits (black) and twin qubits (yellow).

row coordinate, j (ℓ) denotes the column coordinate, and L (R) specifies whether the qubit is in the left $n \times n$ sector or the right $m \times m$ sector. We denote the physical qubits on the principal diagonal of each sector, i.e. $(r, r, \bullet)$, as *diagonal* qubits. Additionally, for an $(r, c, \bullet)$ qubit, we identify the $(c, r, \bullet)$ qubit as its *mirror* qubit, with the two together considered *twin* qubits, see Fig. 2.7(c).

With this layout of the physical qubits, the stabilizer generators of the code have the property that their support is contained in a single row of one sector and a single column of the other. Specifically, X-type stabilizers have support on the kth row of the R sector and the jth column of the L sector. As such, each can be labeled by $S_X(k, j)$. Similarly, a Z-type generator labeled by $S_Z(i, \ell)$ has support on the ith row of the L sector and the ℓth column of the R sector. The stabilizer generators as expressed in this representation can by constructed by indexing specific rows and columns of the underlying classical parity check matrix H [83], see Fig. 2.7(a), or by reshaping the HGP pcms shown in Eq. (2.18). Given the H_X and H_Z matrices, the X-type stabilizer generators expressed in the representation defined above can be obtained by taking the first n^2 columns of H_X and reshaping them into a $n \times n \times M$ matrix, where M is the number of X-type

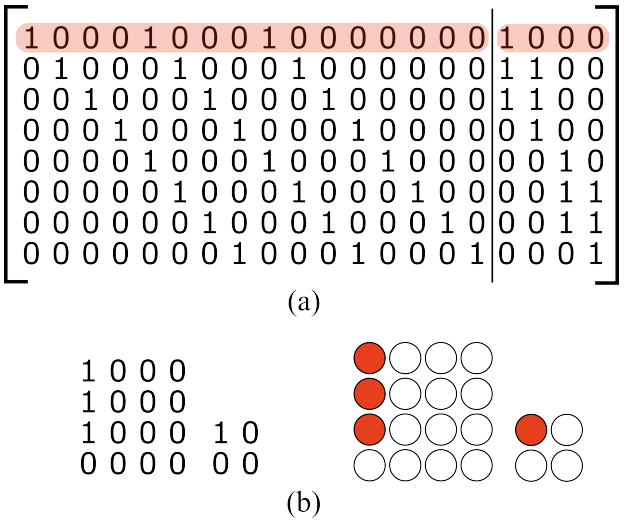

Figure 2.8: (a) The X-type parity check matrix for a $[[20, 4, 2]]$ hypergraph product code. (b) Considering the first row of the pcm, we take the first $n^2 = 16$ columns and reshape them into a 4×4 matrix. We similarly take the remaining $m^2 = 4$ columns and reshape them into a 2×2 matrix. The two matrices together constitue the first stabilizer generator in canonical form.

generators. Also take the remaining $m^2 = (n-k)^2$ columns and reshape them into a $m \times m \times M$ matrix. In this representation, the full ith stabilizer is then given by $((\bullet, \bullet, i), (\bullet, \bullet, i))$. The Z-type generators can be obtained by instead using H_Z. An example of this reshaping in shown in Fig. 2.8, where we consider H_X from a $[[20, 4, 2]]$ hypergraph product code where the underlying classical pcm comes from $\mathbb{F}_2^{2\times 4}$. To obtain the stabilizer generators in canonical form, we take the first $n^2 = 16$ qubits and reshape them into a 4×4 matrix, and similarly for the remaining $m^2 = 4$ qubits. The two matrices together constitute the first stabilizer generator in canonical form.

In this layout, we can also obtain a *canonical basis* for hypergraph product code logical operators [84]. For these codes, a canonical basis is defined to be a set of logical operators such that they have support contained in a single row or column of a single sector. Additionally, $\overline{X}_i$ and $\overline{Z}_i$ share support on exactly one qubit, whereas $\overline{X}_i$ and $\overline{Z}_j$ for $i \neq j$ do not share any support. The physical qubit on which $\overline{X}_i$ and $\overline{Z}_i$ overlap is deemed the pivot qubit for that logical qubit. Similarly to the physical qubits, the logical qubits are either diagonal logical qubits or part of a

twin logical qubit pair depending on the location of its pivot qubit. We refer to Ref. [84] where they provide a method for constructing a canonical basis for square hypergraph product codes.

2.3.2 Bivariate bicycle codes

In Chapter 4, we investigate bivariate bicycle qLDPC codes [85], which come from the wider family of generalized bicycle codes [86]. Let $\mathbb{I}_\ell$ be the $\ell \times \ell$ identity matrix and let S_ℓ be the $\ell \times \ell$ cyclic permutation matrix, which is obtained by shifting the columns of $\mathbb{I}_\ell$ one position to the right. Also let

$$x = S_\ell \otimes \mathbb{I}_m \quad \text{and} \quad y = \mathbb{I}_\ell \otimes S_m \tag{2.38}$$

for integers ℓ, m. We then define two matrices

$$A = A_1 + A_2 + A_3 \quad \text{and} \quad B = B_1 + B_2 + B_3 \tag{2.39}$$

where A_i, B_i are powers of x or y. Here we perform all arithmetic over $\mathbb{Z}_2$. Using A and B, we can construct the CSS-type BB code $BB(A, B)$ with X- and Z-parity checks that, respectively, take the form

$$H_X = [A|B] \quad \text{and} \quad H_Z = [B^T|A^T]. \tag{2.40}$$

To define a valid stabilizer code, we require that all X-type checks commute with all Z-type checks, which translates to the condition $H_X \cdot H_Z^T = AB + BA = 0$. Since $[x, y] = 0$, this condition is satisfied. As with other stabilizer codes, $k = n - \text{rk}(H_X) - \text{rk}(H_Z)$, and the distance is the minimum weight logical operator.

2.3.3 Iceberg codes

Iceberg codes [67, 72, 87], also known as quantum parity codes, are a family of CSS codes with parameters $[[n, n-2, 2]]$. The two weight-n stabilizer generators,

$$S_Z = \bigotimes_{i\in[n]} Z_i \quad \text{and} \quad S_X = \bigotimes_{i\in[n]} X_i, \tag{2.41}$$

allow the code to detect a single bit-flip and phase-flip error. Technically, any error of odd parity is detectable, but the code is unable to differentiate between the error weights. Its $n-2$ weight-2 logical operators can be defined as

$$\overline{X}_i = X_1 X_{i+1} \quad \text{and} \quad \overline{Z}_i = Z_{i+1} Z_n, \tag{2.42}$$

and it is easily verified that the commutation relations between logical operators are satisfied. The logical all-zero state $|\overline{00...0}\rangle$ for an Iceberg code is simply the n-qubit Greenberg-Horne-Zeilinger (GHZ) state:

$$|\overline{00...0}\rangle = \frac{|00...0\rangle + |11...1\rangle}{\sqrt{2}}, \tag{2.43}$$

and so to initialize the codespace we only need to prepare this state. Ref. [88] presents explicit circuits on how to do this fault-tolerantly. The same reference also explains how to do logical computation with these codes. In Appendix B.1 we present a set of logical operators that generate the Clifford group on both logical qubits.

2.3.4 Concatenated codes

Quantum code concatenation [67] is a procedure that takes two stabilizer codes and produces a larger stabilizer code. In a concatenated code, the physical qubits are first encoded in some $[[n_1, k_1, d_1]]$ stabilizer code encoding k_1 logical qubits. These k_1 logical qubits then serve as the *physical* qubits for an $[[n_2, k_2, d_2]]$ stabilizer code. This process can be repeated arbitrarily many times, with each encoding step generally increasing the error correction properties of the code. One concatenation scheme is shown below. Note that the scheme requires $k_1 \mid n_2$.

Procedure 1. (Concatenation of quantum codes, see e.g., Section 3.5 of Ref. [67]). *Consider stabilizer codes $\mathcal{Q}_1$ and $\mathcal{Q}_2$ with parameters $[[n_1, k_1, d_1]]$, $[[n_2, k_2, d_2]]$ and stabilizers $\mathcal{S}_1$, $\mathcal{S}_2$, respectively. Then a concatenated code $\mathcal{Q}$ with parameters $[[n_1 n_2/k_1, k_2, d \geq d_1 d_2/k_1]]$ can be constructed as follows:*

Divide the n_2 qubits into n_2/k_1 blocks of k_1 qubits, $B(b), b \in \{1, ..., n_2/k_1\}$. Each block of k_1 qubits is then encoded into n_1 qubits using $\mathcal{Q}_1$.

1. *For each generator $M \in \mathcal{S}_1$ include in S the Pauli M_i acting on the ith block of n_1 qubits tensored with the identity on all other blocks.*

2. *Also include in $\mathcal{S}$ every generator $M \in S_2$, where each single-qubit Pauli P_i is replaced with the corresponding logical Pauli operator according to the mapping of $B(b)$.*

As an example, let us concatenate the $[[4, 2, 2]]$ Iceberg code with itself according to Procedure 1. The resulting code will have parameters $[[8, 2, d \geq 2]]$. The generators and logical operators for the $[[4, 2, 2]]$ code are shown below:

$$S_X = X_1X_2X_3X_4 \quad S_Z = Z_1Z_2Z_3Z_4 \tag{2.44}$$

$$\overline{X}_1 = X_1X_2 \quad \overline{Z}_1 = Z_2Z_4 \tag{2.45}$$

$$\overline{X}_2 = X_1X_3 \quad \overline{Z}_2 = Z_3Z_4 \tag{2.46}$$

Intuitively, we are constructing a $[[4, 2, 2]]$ code where the physical qubits are each a logical qubit encoded in another $[[4, 2, 2]]$ code. Thus we need two code blocks to give us the required number of logical qubits. For simplicity let us denote the logical qubits of the first code block as logical qubits 1, 2, and the logical qubits of the second code block as logical qubits 3, 4. Additionally, we let logical qubits with odd indices to correspond to the specific logical representation shown in Eq. (2.45), while logical qubits with even indices correspond to Eq. (2.46). This assignment can be arbitrary, but certain assignments may provide increased distance and error correcting performance, see Section 5.2.1. Following step 1 of Procedure 1, we add the stabilizer generators for these two code blocks to the stabilizer of the concatenated code. Following step 2, we must then replace the $[[4, 2, 2]]$ stabilizers of Eq. (2.44) with the corresponding logical operators according to this assignment. Thus, for example, we substitute X_1 with $\overline{X}_1$ on the first block: X_1X_2 and we substitute X_4 with $\overline{X}_2$ on the second block: X_4X_7. $S_X = X_1X_2X_3X_4$ hence becomes $X_2X_3X_5X_6$. We do the same mapping with the two sets of logical operators, Eq. (2.45), (2.46) to obtain logical operators for the two logical qubits of the resulting concatenated code. We see that the commutation relations between $\overline{X}_i$ and $\overline{Z}_i$ are satisfied, so this set of logical operators forms a valid logical basis. There is an alternative concatenation procedure:

Procedure 2. Concatenation of quantum codes. *Consider stabilizer codes $\mathcal{Q}_1$ and $\mathcal{Q}_2$ with pa-*

	X	X	X	X	I	I	I	I
	Z	Z	Z	Z	I	I	I	I
	I	I	I	I	X	X	X	X
	I	I	I	I	Z	Z	Z	Z
	I	X	X	I	I	X	X	I
	I	Z	Z	I	I	Z	Z	I
$\overline{X}_1$	I	X	X	I	I	I	I	I
$\overline{Z}_1$	I	Z	I	Z	I	Z	I	Z
$\overline{X}_2$	X	X	I	I	X	X	I	I
$\overline{Z}_2$	I	I	I	I	I	Z	Z	I

Table 2.2: Stabilizer generators and logical operators for a $[[8, 2, 2]]$ code obtained from the concatenation of the $[[4, 2, 2]]$ code with itself according to Procedure 1.

rameters $[[n_1, k_1, d_1]]$*,* $[[n_2, k_2, d_2]]$ *and stabilizers* $\mathcal{S}_1$*,* $\mathcal{S}_2$*, respectively. Then a concatenated code* $\mathcal{Q}$ *with parameters* $[[n_1 n_2, k_1 k_2, d \geq d_1 d_2]]$ *can be constructed as follows:*

Divide the $n_1 n_2$ *qubits into* n_2 *blocks of* n_1 *qubits,* $B(b), b \in \{1, ..., n_2/k_1\}$*. Each block of* n_1 *qubits is then encoded into* k_1 *logical qubits using* $\mathcal{Q}_1$*.*

1. *For each generator* $M \in \mathcal{S}_1$ *include in* S *the Pauli* M_i *acting on the* i*th block of* n_1 *qubits tensored with the identity on all other blocks.*

2. *Using the* i*th logical qubit from each block* $B(b)$ *include in* $\mathcal{S}$ *every generator* $M \in S_2$*, where each single-qubit Pauli* P_i *is replaced with the corresponding logical Pauli operator.*

3. *Repeat the previous step* k_1 *times, for each logical qubit in* $B(b)$*.*

In Chapter 5 we use Procedure 1 to construct a concatenated code; however, we could just as easily used Procedure 2, and it may be interesting to consider reproducing Chapter 5 with this second concatenation procedure.

2.3.5 Decoding

As alluded to in the previous sections, *decoding* is process of determining an correction based on the observed syndrome. The syndrome, which is obtained using syndrome extraction circuits like those in Fig. 2.3, is given to an algorithm called a *decoder* that outputs a likely correction. Consider an error E and correction E' with $E, E' \in \mathcal{P}_n$. It might seem like a sufficient condition for decoding success, i.e. returning the quantum state to the codespace, is that $\sigma(E \oplus E') = 0$; however, this is not enough to guarantee success. It may be the case that $E \oplus E' \in N(\mathcal{S}) \setminus \mathcal{S}$, in which case the overall effect of the error and the correction is a logical error—yet $\sigma(E \oplus E') = 0$. So we instead consider the *logicals matrix* consisting of the logical representatives, $\overline{X}_1, \overline{Z}_1, ..., \overline{X}_k, \overline{Z}_k$, of the code. When expressed in the binary symplectic representation, the logical matrix for CSS codes has a similar structure to its parity check matrix, Eq. 2.16

$$L = \left(\begin{array}{c|c} L_X & 0 \\ \hline 0 & L_Z \end{array}\right) \tag{2.47}$$

Decoding is then considered a success if $L \cdot E = L \cdot E'$, otherwise we say that a logical error has occurred. We now briefly describe common noise models for simulating the logical memory performance of a QECC and the corresponding decoding techniques.

2.3.5.1 Code capacity

The simplest noise model is when only the data qubits experience noise, often at the beginning of an error correction cycle. For example, they may all experience depolarizing noise, Eq. (2.10), with probability p. Or we may consider CSS codes for simplicity, and consider bit-flip

noise and phase-flip noise, Eq. (2.9), separately. Unless there is an asymmetry between the X- and Z-type generators for a CSS code, it often suffices to investigate the performance of only one type. Hence it is often customary to consider, for example, only X-type errors and the Z-type syndrome, treating the problem like decoding a classical code. Although belief propagation performs remarkably well for classical LDPC codes, decoding CSS codes in this way yields sub-optimal performance [89–91], due to short cycles in the Tanner graph and error degeneracy. To address the failures of belief propagation, post-processing methods have been introduced [92–94]. Additionally, entirely new decoders tailored to specific quantum codes have been developed [23, 79, 83, 95, 96] (among many others), in some instances providing better performance than general-use decoders like belief propagation.

2.3.5.2 Phenomenological noise

A slightly more realistic noise model is one in which the syndromes themselves can be noisy, namely the *phenomenological* noise model. When performing syndrome extraction circuits like Fig. 2.3 on noisy quantum computers, there is the possibility of errors propagating to the ancilla qubit or the measurement of the ancilla qubit failing. In general, this results in bit-flip errors being applied to the syndrome with probability p_{synd}. As the syndrome is the main input to the decoder, incorrect syndromes can cause significant increases in the logical error rate. We say main input, as there is also the option of providing soft information to the decoder, usually in the form of error probabilities on each qubit. It has been shown that providing this extra information can improve decoding performance [97–99]. The typical method for dealing with syndrome noise is to repeat the stabilizer measurements $\Theta(d)$ times and then decode over the

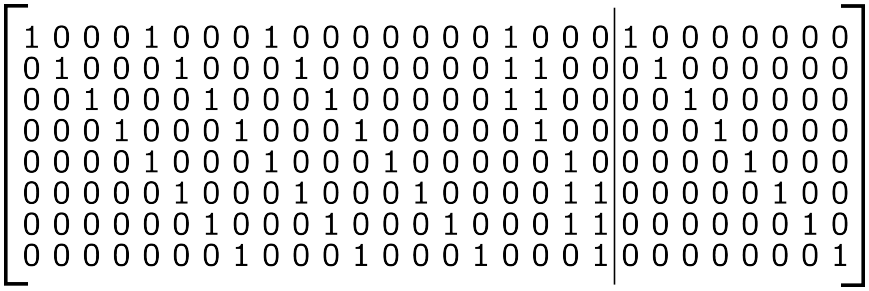

Figure 2.9: The X-type parity check matrix for a $[[20, 4, 2]]$ hypergraph product code with additional columns to account for syndrome noise.

entire syndrome history [58].

It was shown in Ref. [59] that for certain *single-shot* QECCs, only a single round of noisy stabilizer measurements is necessary to display increased error suppression with larger block-lengths. There are two sufficient conditions to be single-shot: some codes, namely those constructed from 5-term chain complexes such as the 4D surface code [58] and higher-dimensional hypergraph product codes [82, 100], have redundancies in their stabilizer generators that allow them to correct for syndrome errors before decoding according to Section 2.3.5.1. Codes with this property are said to be single-shot by soundness [60]. Other codes are single-shot by confinement [101], which very roughly states that the residual qubit and syndrome error remain bounded after decoding. To decode these codes, we follow Ref. [94] to modify the Tanner graph and parity check matrices of the CSS code: for each check node, we add a single bit node which represents the possibility of a syndrome error. Fig. 2.9 displays the pcm for a $[[20, 4, 2]]$ hypergraph product code (shown unaltered in Fig. 2.8) after including columns for potential syndrome errors. This augmented pcm can then be used in a decoding algorithm as described above. Note that when applying the correction to the system, these syndrome errors are ignored.

2.3.5.3 Circuit-level noise

In reality, errors may occur at any operation in the syndrome extraction circuit, including qubit initialization, single- and two-qubit gates, measurements, and idle locations. To model this, we instead consider the standard circuit-based depolarizing noise model [102], where for each operation in the circuit, an error is introduced with some probability p. For example, an error arising from a CNOT gate is the gate followed by one of the possible 15 non-identity two-qubit Pauli products on the control and target qubits. Although it is possible to decode circuit-level noise using the same method as for phenomological noise [103], it has been shown to be advantageous to instead use a space-time circuit-level decoder [104, 105]. The goal now is to guess the error at specific locations in the syndrome extraction circuit. Again, decoding is considered a success if the guessed errors have the same effect on the logical observables as the actual error.

The input to the space-time decoder is not the syndrome of the error, but rather the parities of the syndrome measurements between error correction rounds. In the absence of errors, the syndrome between rounds should be constant, i.e. have parity of zero. A parity of one indicates that an error occurred at some point in the previous error correction round. Following the notation of Stim [106, 107], we define the ith *detector* at time t to be the parity of the syndrome of the current and previous rounds $D_i^{(t)} = \sigma_i^{(t)} \oplus \sigma_i^{(t-1)}$, where $\sigma_i^{(t)}$ is the ith bit of the syndrome at time t. We make one change to allow for the possibility of partial quantum error correction, where we have the choice of neglecting to measure certain generators for some number of rounds, t_m. As such, detectors for these generators must compare the parities of the corresponding syndromes t_m rounds apart, $D_i^{(t)} = \sigma_i^{(t)} \oplus \sigma_i^{(t-t_m)}$. Each detector allows us to determine whether errors have

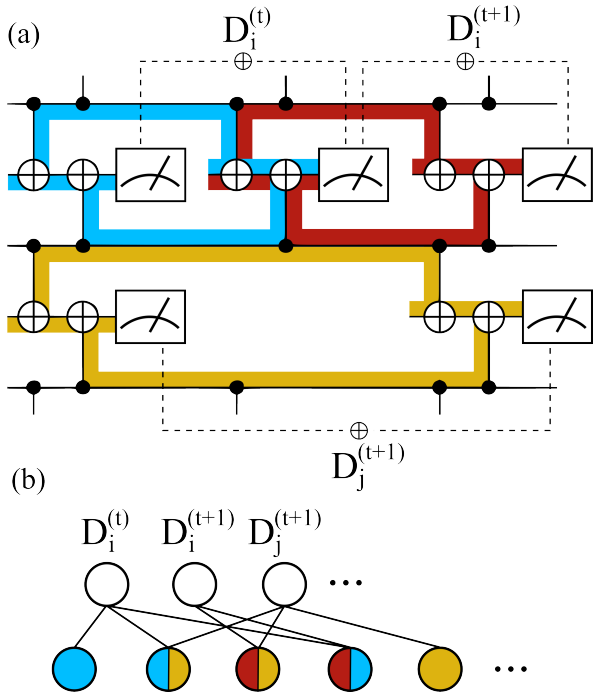

Figure 2.10: (a) Detectors for a portion of the bit-flip repetition code. The highlighted regions represent the detecting region [107] of a detector, the set of errors that would cause the detector to be triggered. The corresponding detectors are then the parities of the measurements in that region. Since syndrome j was masked for a round, the detector now represents the parities of the measurements in the region that spans three rounds. (b) The bipartite space-time decoding graph of the circuit. The check nodes of this graph are the detectors, and the bit nodes are possible errors during the execution of the circuit. A detector and error are connected by an edge if the error causes the detector to be activated. Errors on the boundary of two detecting regions cause both detectors to trigger.

occurred in a specific detecting region [107] of the circuit. Figure 2.10(a) shows a simple example of a classical repetition code circuit with its associated detectors and highlighted detecting regions.

To correct for errors in the circuit-level model, we relate the detectors with errors in the circuit by constructing a bipartite graph. Let the detectors over T rounds be the check nodes, and let every possible single- and two-qubit error over the circuit make up the bit nodes. A detector and error are connected by an edge if the error causes the detector to activate. As a practical note, many errors have the same action on the detectors and logical observables, so they

can be consolidated into a single node. Since each error in this set has the same action on the final logical observables, one can choose an arbitrary representative when checking for decoding success. Similarly, some errors will have no effect on the detectors or logical observables, and as such are not included as a node in the bipartite graph. This bipartite graph can be considered the Tanner graph of a classical code and can be decoded by any appropriate decoder to deduce the errors that have occurred. Figure 2.10(b) shows the bipartite decoding graph corresponding to the circuit of panel (a). The classes of equivalent errors from each detecting region constitute the bit nodes of the graph and are connected by edges to the appropriate detectors. For a more detailed discussion of the circuit-level noise decoding process, see Ref. [85].

Chapter 3: Masking and partial syndrome measurement

In this chapter, we introduce the notion of using a submaximal set of stabilizer generators to do quantum error correction, which we call partial syndrome measurement, and we formalize it with masking and (un)masking schedules. This procedure forms the basis for the other instantiations of partial error correction described in the following chapters. Applying this new technique, we motivate a new practical protocol for implementing nonlocal qLDPC codes on quantum hardware restricted to 2D local gates.

We first begin by introducing the concept driving the motivation for this and the next chapter: *locality*. Let G be the Tanner graph or connectivity graph of a quantum code, and consider assigning the vertices of the graph to the points in a D-dimensional grid. Such an assignment is called an *embedding*:

Definition 3. *(Graph embeddings) For a graph $G = (V, E)$, a map $\eta_\theta : V \to \mathbb{Z}^D$ is called an embedding. An embedding η_θ is a θ-embedding if for all distinct vertices $u, v \in V$,*

$$|\eta_\theta(u) - \eta_\theta(v)| \geq \theta. \tag{3.1}$$

When $\theta = O(1)$, the corresponding code is said to be *local* in D dimensions, or DD local. Equivalently, a code is considered local in $\mathbb{Z}^D$ if, when embedded in a lattice with side length

$O(\sqrt[D]{n})^D$, its generators act on qubits within a ball of constant radius. It was shown that there is an intimate relationship between locality and the parameters of a quantum code. In particular, Refs. [108, 109] showed the distance d for a local code in $\mathbb{Z}^2$ is bounded above by $O(\sqrt{n})$, and the number of logical qubits k obeys the relation $kd^2 = O(n)$. In D dimensions, these so-called Bravyi-Poulin-Terhal (BPT) bounds are generalized like $d = O(n^{1-1/D})$ and $kd^{2/(D-1)} = O(n)$.

Several quantum computing modalities are based on two-dimensional architectures with fixed qubit layouts, including superconducting qubits [110], quantum dots [111], and nitrogen-vacancy (NV) center qubits [112]. On these devices, two-qubit entangling gates are only natively available between qubits that are neighboring. We say natively because longer range entangling gates can be compiled into the available gateset; however, this incurs additional overhead that often makes these compiled gates noisier than the native two-qubit gates, see Chapter 4. As such, on these devices it is significantly easier to implement QECCs that are 2D local. Recently, a popular choice for code family with this property has been the surface code and its variations [38, 39]. While it has local, weight-four generators and a favorable $\Theta(\sqrt{n})$ distance scaling, the surface code has a rate, k/n, which tends to zero as n approaches infinity. These parameters saturate the BPT, and so they are the best we can hope to achieve for a 2D local code.

To surpass the BPT bounds, additional nonlocality is required; that is, we need $\theta = \Omega(1)$ in Definition 3. Precisely how much locality is required was first studied in Refs. [113, 114] and later refined in Ref. [115]. The latter work proved the following, optimal, statement about 2D embeddings of stabilizer codes:

Theorem 4 ([115], Theorem 1.2)**.** *There exist absolute constants $c_0, c_1 > 0$ such that the following holds. Any 2D-embedding of a $[[n, k, d]]$ stabilizer code with $kd^2 \geq c_1 \cdot n$ must have at least*

$c_0 \cdot \max(k, d)$ *interactions of length at least* $c_0 \cdot \max\left(\frac{d}{\sqrt{n}}, \left(\frac{kd^2}{n}\right)^{1/4}\right)$.

That is to say that exceeding the BPT bound is costly in terms of the amount of nonlocality that arises from any D-dimensional embedding. One such family that provides improved parameters is hypergraph product (HGP) codes [76], specifically quantum expander codes [79]. This construction has the same $\Theta(\sqrt{n})$ distance scaling, but now with a constant rate, $k = \Theta(n)$; the trade-off, however, is that the stabilizer generators of HPG codes are very nonlocal, requiring $\Omega(n)$ interactions of length $\Omega(n^{1/4})$. This nonlocality poses problems for 2D local implementation of qLDPC codes, such as HGP codes, that surpass the BPT bound.

Indeed, several recent works have provided evidence against the possibility of doing error correction on architectures restricted to 2D local gates. Delfosse *et al.* [103] investigated the problem of performing syndrome extraction circuits of HGP codes using 2D local gates and classical communication and presented numerical simulations suggesting that the resulting overhead was prohibitive. Baspin *et al.* [116] provide further evidence against 2D local implementations of qLDPC codes by deriving bounds on the amount of overhead needed to perform error correction at a given logical error rate. They show that the restriction to 2D local gates incurs polynomial overhead. However, they also note that their definition of error rate is very restrictive and that computations not satisfying this definition might not obey the overhead bound. It therefore remains possible that we could obtain more favorable performance by using alternative error correction schemes, such as partial syndrome extraction and the stacked model, which we now introduce.

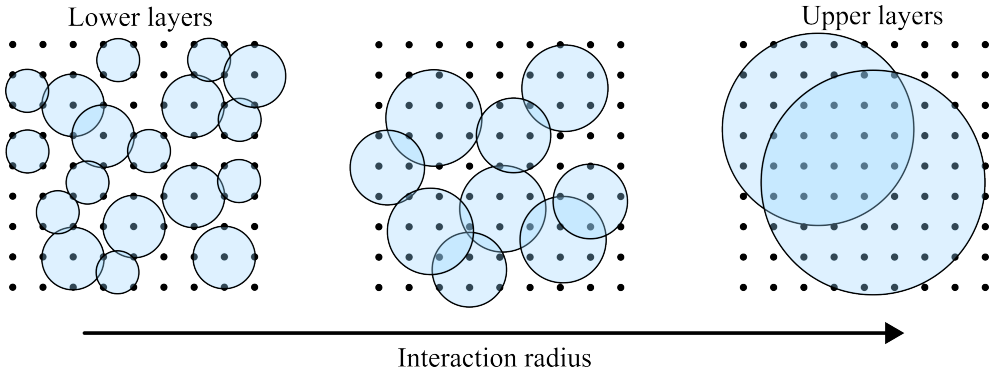

Figure 3.1: Illustration of stacked model layer interaction radius and frequency. Blue circles represent the support coverage of the stabilizer generators. For codes in the stacked model, the lower layers contain many small generators. The code will also have larger generators; however, as interaction radius increases, the frequency decreases.

3.1 The stacked model

As a concrete example of partial syndrome measurement, we show through analytic and numerical evidence that repeated quantum error correction with quantum expander codes still provides a threshold even when a constant fraction of generators are not measured. This result suggests that it may be possible to build a fault-tolerant quantum computer with nonlocal qLDPC codes on architectures restricted to 2D local gates with a procedure based on the *stacked model* [113]. After embedding a QECC in a grid of size $O(\sqrt{n}) \times O(\sqrt{n})$, the stabilizer generators are partitioned into a stack of layers based on the radius of the ball containing the qubits they act on. The bottom layer of the stack contains local generators, and as we move up the stack, the interaction radius increases while the number of generators of that size decreases. Note that the layers in the stack do not correspond to physical layers on hardware. Instead, they are a conceptual tool for partitioning the generators into sets based on their geometric size. Fig. 3.1 illustrates this: for codes in the stacked model, the relative frequency of the stabilizer generators is related to their interaction radius. Ideally, we can use codes which when embedded into $\mathbb{Z}^2$ have the

property that the number of generators decreases exponentially with increasing radius. That is, a (large) constant fraction of the generators act on qubits within a support of constant radius. The reason for wanting this is that when restricted to 2D local gates, the set of nonlocal generators takes much longer to route and measure than the local generators, see Chapter 4. The key insight is that measuring the nonlocal generators less frequently than the local ones could significantly shorten the syndrome extraction time, at the cost of potentially reduced error correction capabilities. It was shown in Ref. [113] that any code constrained to the above model has a distance that is bounded by $\widetilde{O}(n^{2/3})$ and obeys the relation $k^3d^4 = \widetilde{O}(n^5)$. Here, $\widetilde{O}(\cdot)$ is a variant of big O notation that ignores log factors, e.g. $f(n) \in \widetilde{O}(h(n))$ is equivalent to $\exists k : f(n) \in O(h(n)\log^k n)$. Quantum expander codes codes satisfy this trade-off.

However, since the publication of the paper this chapter was based on [117], another work [115] refined the bounds of Ref. [113] yielding a tighter distance bound $d \leq O(n^{2/3})$ and improved rate-distance trade-offs: $kd^4 \leq O(n^3)$, $k^3d^4 \leq O(n^3)$. Quantum expander codes $(k = \Theta(n), d = \Theta(n^{1/2}))$ no longer satisfy these bounds, and as such cannot be implemented in the stacked model. This was somewhat expected by us, given as expander graphs are notoriously hard to embed into any finite dimension [118]. Indeed, we made some effort to find specific embeddings into $\mathbb{Z}^2$ that yielded good generator size distributions; however, the resulting distributions instead often favored mid-sized generators. While this rules out quantum expander codes, it still remains the case that codes such as 2D hyperbolic codes [119], 4D hyperbolic codes [120, 121], or fiber bundle codes [122] may be implemented in the stacked model. Alternatively, there might be enough practical benefit to use codes that have the same asymptotic parameter scaling as the surface code $(k = \Theta(1), d = \Theta(\sqrt{n}))$, but encode more qubits at small blocklengths, such as semi-topological codes [92], long-range-enhanced surface codes [123], or

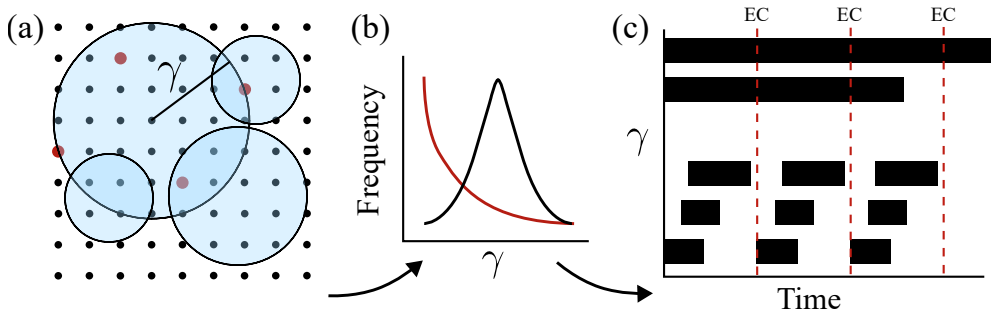

Figure 3.2: Overview of the stacked model. (a) After embedding a quantum code into $\mathbb{Z}^2$, each stabilizer generator has a parameter γ that denotes the radius of the ball containing the qubits in its support. (b) Two possible distributions of γ over the set of generators. The most advantageous distributions for this scheme are those where the relative frequency decays exponentially with increasing γ (red curve). (c) An example *schedule* for the generator measurements. The syndrome extraction circuits for the smaller generators are able to be prepared quickly, and so their syndromes are available during every round of error correction (red dashed lines). The larger generators require more time to build their syndrome extraction circuits, so this is done over a period of time that may stretch over several error correction rounds. More practically, priority is given to the smaller generators, and after completing them, the larger generators are worked on using any remaining time before an error correction round.

La-cross codes [124].

Measurement of the generators at the bottom of the stack takes constant time, since they are local. As such, these generators can be considered 'easy' in some sense, and their syndrome information can be assumed to be available during every round of error correction. As we move up the stack, the interaction radius increases. The important distinction to make is that while the generators on higher layers are nonlocal, we are still measuring them with only 2D local gates, and so extracting these syndromes takes longer than for local generators. To mitigate the extra noise arising from additional idling and the compiled long-range gates, these nonlocal generators are measured less frequently than those lower in the stack, and hence their syndrome is not always available. This scheme is depicted in Fig. 3.2. Doing this, as we will briefly argue, has the potential of reducing the overhead of performing syndrome extraction, given that the intermediate rounds of error correction are actually correcting errors.

We can roughly approximate the amount of work required to perform syndrome measurement using 2D local gates by estimating the number of SWAP gates in the extraction circuits. For a generator with interaction radius γ, the total number of SWAP gates needed to perform the syndrome measurement is proportional to γ. As a concrete example, consider a qLDPC code on $n = 100,000$ qubits which when embedded into $\mathbb{Z}^2$ results in a generator distribution where the number of generators decays exponentially with increasing γ. Drawing $O(n)$ generators from this distribution and summing the radii of the smallest 90% is $\sim 3\%$ of the total sum across all generators. Thus, we can estimate that the syndrome of these smallest 90% of generators can be obtained using only $\sim 3\%$ of the SWAP gates required to perform all of the syndrome measurements. Obtaining the remaining 10% of the syndromes requires the majority of the work, but these circuits are built up over time (see Fig. 3.2(c)), allowing for a significant portion of the full error correction capabilities to be available during each error correction round. Alternatively, the syndrome measurement circuits of the large generators can just be performed less frequently, which is the method we use in Chapter 4. Although the resulting logical error rates will be strictly larger than when using a full syndrome, the reductions in overhead may outweigh the increases in the logical error rate.

The remainder of the chapter is structured as follows. Section 3.2 introduces the idea of masking and contextualizes it with respect to the stacked model. In Section 3.3, we apply previous results about quantum expander codes to provide some analytical bounds on using masking during multi-round error correction. Section 3.4 provides empirical evidence to suggest that the analytical thresholds are better in practice. We conclude in Section 3.5 with a discussion of the remaining problems for an physical implementation of the stacked model.

3.2 Syndrome masking

The notion of *masking* was originally introduced as a way of describing fault-tolerant protocols for space-time codes [125]. We use the same idea here, although in a different context. An element of the stabilizer is considered masked if we cannot measure its eigenvalue during an error correction round. We follow the definition from [125] and define two subgroups of the stabilizer, U and T, where $U \subseteq T \subseteq S$. The *always unmasked* subgroup, U, are the stabilizers whose eigenvalues can be measured in a constant number of rounds, whereas the *temporarily unmasked* subgroup, T, are the stabilizers whose eigenvalues can be measured in a number of rounds that can scale with the size of the code, n. In general, it could be the case that $T \subsetneq S$ where the set $S \setminus T$ contains stabilizers that cannot be measured on any time scale. In this chapter, we consider the case $U \subset T = S$. The subgroups form valid stabilizer codes, and as such can be described by their parameters. Defining k for these codes has no real meaning since logical information is not being stored in the subspace; however, we can define the corresponding distances, where

$$d_U = \min |N(U) \setminus S| \quad d_T = \min |N(T) \setminus S|. \tag{3.2}$$

In other words, d_U (d_T) is the weight of the smallest Pauli operator outside of the full group that has zero syndrome when measuring only the stabilizer generators of U (T). We call d_U and d_T masked distances, whereas d is the unmasked distance. Note that $d_U \leq d_T \leq d$.

Since not every generator is measured, the resulting syndrome may have less information about the error than would otherwise be available if the full set of stabilizer generators were measured. For any number of masked generators, there is a set of *invisible* errors that have zero

syndrome on the generators of U (or T) while having a non-zero syndrome in S. In particular, the new normalizer $N(U)$ contains $N(S)$ as well as all cosets of $\mathcal{P}_n/N(S)$ labeled with undetectable error syndromes.

Furthermore, errors that were previously correctable may no longer be uniquely identifiable with the syndrome of U or T. Note that errors with a zero syndrome for U do not immediately cause logical errors, unlike errors with a zero syndrome for all of S. If an error has a non-zero syndrome for T, it will eventually be detected, once the generators of $T \setminus U$ are unmasked. The risk is that such errors will accumulate over time and become logical errors before they can be corrected.

3.2.1 Masking and the stacked model

Identifying which layers of the stack are available during an error correction round corresponds to specifying the temporarily unmasked subgroup, T_t, at each time step in the circuit, $t = 1, ..., \tau$. The always unmasked subgroup, $U \subset T_t$, is static over the execution of the circuit and so can be specified at the beginning. This set contains all local generators, as their eigenvalues can be measured in constant time and so are assumed to always be available. T_t will contain U as well as any additional layers that have completed syndrome extraction between time $t-1$ and t. Since, in general, we want to measure all generators throughout the course of the circuit, $\bigcup_t T_t = S$; however, it may not be the case that any one time step has all generators available. An equivalent interpretation is to specify $S \setminus T_t$, the set of generators whose eigenvalues are not available during time step t. For the remainder of the chapter, we consider 'applying' a mask D to be specifying this set, $S \setminus T_t$.

Algorithm 1 A simplified fault-tolerance scheme

for t = 1, ..., τ **do**

Generate an error F_t with probability p_{phys} and apply F_t to the current error:

$$E'_t := F_t \oplus E_{t-1}$$

Generate a syndrome error D_t with probability p_{synd}

Decode on the input (E_t, D_t) and correct using the decoded error $\hat{E}_t$:

$$E_t := E'_t \oplus \hat{E}_t$$

end for

Generate an error F_t with probability p_{phys} and apply to the current error:

$$E_\tau := F_\tau \oplus E_{\tau-1}$$

Decode on the input $(E_T, \varnothing)$

3.3 Analytic results

In this section, we consider previous results on quantum expander codes in the context of masking in a multi-round error correction procedure. To decode we use the small-set flip (SSF) decoder [79], a detailed description of which is given in Appendix A.1.

A quantum circuit is considered fault-tolerant if it prevents errors from propagating throughout the circuit; in this way, it keeps the size of the residual error manageable for the QECC. To convert a circuit into a fault-tolerant version, the qubits are first encoded in some QECC, and then each operation in the original circuit is replaced with a fault-tolerant logical *gadget* that acts on the logical qubits of the QECC. We need gadgets to initialize qubits, perform measurements, implement logical operations from $N(S)/S$, and do error correction. In general, errors may occur in any of the gadgets, so after each gadget in the circuit a round of fault-tolerant error correction is performed. To investigate how an error propagates throughout a fault-tolerant circuit more easily, we often abstract the above model and instead work with the procedure described in Al-

gorithm 1. For the purposes of analysis and simulation, we condense all gadgets, except error correction, into a single event where every qubit has a bit/phase flip error applied with probability p_{phys}. Additionally, error correction is noisy in the sense that each syndrome bit is flipped with probability p_{synd}. We later discuss how to make this scheme more realistic, but for the purposes of determining the effects of performing error correction with partial syndromes this simplified model is sufficient. In this model we consider qubit errors, E, and syndrome errors/masks, D, that follow a local stochastic noise model:

Definition 5. *(Local stochastic error model). We say that an error (E, D) is local stochastic if there are error parameters (p_{phys}, p_{synd}) such that for any F and L, $\Pr[F \subseteq E, L \subseteq D] \leq p_{phys}^{|F|} p_{synd}^{|L|}$.*

Much work has been put into the investigation of quantum expander codes and the SSF decoder in this model [80, 126, 127]. Most relevant to us is the fact that they can tolerate random qubit errors and syndrome errors of linear size, as stated in the following theorem.

Theorem 6. *(modified from Fawzi, Grospellier, Leverrier [80]). There exists a non-zero constant $p_0 > 0$ such that the following holds. Suppose that the error (E, D) each satisfy a local stochastic noise model with parameters p_{phys} and p_{synd} where $p_{phys} < p_0$ and $p_{synd} < p_0$. If we run Algorithm 4 on the input (E, D) then there exists a random variable $E_{ls} \subseteq V$ with a local stochastic distribution with parameter $p_{ls} := p_{synd}^{\Omega(1)}$ such that:*

$$\Pr\left[E_{ls} \text{ and } E \oplus \hat{E} \text{ are not equivalent}\right] \leq e^{-\Theta(\sqrt{n})} \tag{3.3}$$

In the analysis for the above theorem, Fawzi *et al.* consider an error D in the syndrome

Algorithm 2 Multi-round decoding with a fixed (random) mask

Apply a mask D with probability p_{mask}
for t = 1, ..., τ **do**
Generate an error F_t with probability p_{phys} and apply F_t to the current error:

$$E'_t := F_t \oplus E_{t-1}$$

Decode on the input (E_t, D) and correct using the decoded error $\hat{E}_t$:

$$E_t := E'_t \oplus \hat{E}_t$$

end for
Generate an error F_t with probability p_{phys} and apply to the current error:

$$E_\tau := F_\tau \oplus E_{\tau-1}$$

Decode on the input $(E_T, \varnothing)$

to be a subset of the stabilizer generators whose measurement results have been flipped. Very briefly, the argument requires that the syndrome error does not form clusters on the syndrome adjacency graph [24] for it to be tolerable. As such, p_0 must be below the percolation threshold of the syndrome adjacency graph of $\mathcal{Q}$. This value is a constant that depends only on Δ_V and Δ_C of the code. We can turn the result of a masked measurement into the above form by randomly assigning measurement outcomes to the generators included in the mask. Thus, in this context, we can say that Theorem 6 holds when a mask—turned syndrome error—D satisfies a local stochastic noise model with parameter $p_{\text{mask}} < p_0$.

The above analysis is sufficient in the case when we do a single round of masked error correction where the mask is treated like a random syndrome error. However, we will not be using random masks when implementing the stacked model in reality; instead, for a code and an embedding, we will have fixed masks corresponding to different layers. These masks are then applied on error correction rounds when the corresponding syndromes are not available, potentially several in a row. When we use the same mask over consecutive rounds, we have to be

more careful about accounting for the correlations between the syndrome errors, new qubit errors, and residual qubit errors. Accurately characterizing these correlations is likely difficult. As a first step toward this reality, we will only consider a random mask that is fixed at the beginning of the computation, see Algorithm 2. Following the notation of Algorithm 2, in each round t we have the syndrome error from the mask D, any error that was not fully corrected in the previous round E_{t-1}, and a new error F_t. When considered individually, all three error sources are local stochastic described by parameters $p_{\text{mask}}, p_{\text{res}},$ and p_{phys}, respectively. When looked at together, the new error and the syndrome error are bounded by

$$\Pr[F \subseteq E \text{ and } L \subseteq D] \leq p_{\text{phys}}^{|F|} p_{\text{mask}}^{|L|} \tag{3.4}$$

and similarly for the residual error and the new error, as per the definition of a locally stochastic error. However, we would expect to see correlations arise between the residual error and the syndrome error over the rounds, and so together they no longer obey a local stochastic noise model. Instead, they are bounded by:

$$\Pr[F \subseteq E \text{ and } L \subseteq D] \leq \min\{p_{\text{res}}^{|F|},\ p_{\text{mask}}^{|L|}\}. \tag{3.5}$$

When $\max\{p_{\text{res}}, p_{\text{mask}}\} < p_0$, the threshold from Theorem 6, we can say that the probability of clustering is at most $e^{-\Theta(\sqrt{N})}$ by plugging the error bound in Eq. (3.5) into Theorem 17 ([126]). With this, we can apply Lemma 26 ([80]) to bound the probability of the residual error obeying a local stochastic distribution, $\Pr[S \subseteq E_{\text{ls}}]$. Besides the requirement that $E \cup D$ forms clusters with low probability, we need that $\Pr[L \subseteq D] \leq p_{\text{mask}}^{|L|}$. Since we assumed that the mask was

chosen according to a local stochastic error model, this statement is satisfied for all rounds $t \leq \tau$. We are then able to apply Theorem 6 in an iterative manner, yielding the following result.

Theorem 7. *(Grospellier [127]). Let p_0 be the threshold of Theorem 6, and let p_{mask} and p_{phys} be such that:*

$$p_{mask} < \left(\frac{p_0}{2}\right)^{\Omega(1)} \quad \textit{and} \quad p_{phys} < \frac{p_0}{2}. \tag{3.6}$$

Then Algorithm 2 fails with probability at most $(\tau + 1)e^{-\Theta(\sqrt{n})}$.

If the conditions for Theorem 7 are satisfied, then we can make the failure probability for the procedure arbitrarily small by using larger codes. This result is perhaps surprising given the following two claims:

Claim 8. *Applying a random mask D with parameter p_{mask} to a qLDPC code $\mathcal{Q}$ results in a code $\mathcal{Q}' = \mathcal{Q}(S \backslash D)$ whose Tanner graph has the following degree distribution:*

$$\Pr(\deg(\mathcal{Q}'_{|v}) = i) = \binom{\deg(\mathcal{Q}_{|v})}{i} p_{mask}^{\deg(\mathcal{Q}_{|v})} (1 - p_{mask})^i \tag{3.7}$$

Here, we use the notation $\deg(Q_{|v})$ to mean the degree of node v in $\mathcal{Q}$. Since we assume $\mathcal{Q}$ to be LDPC, $\deg(v)$ is bounded by a constant Δ_V for all v, but the values may differ between vertices.

Corollary 9. *Randomly masking a constant fraction of generators results in a masked distance of $d_U = 1$ with high probability.*

Proof. Applying Claim 8 with $p_{\text{mask}} = O(1)$ gives a degree distribution where

$$\Pr(\text{Degree of qubit } v = 0) = p_{\text{mask}}^{\deg(\mathcal{Q}_{|v})} = \Omega(1) \tag{3.8}$$

for all qubits. In this case, an error on such a qubit has zero syndrome on the remaining unmasked generators, U. As this error would not be an element of the stabilizer, $d_U = 1$. □

The always unmasked subgroup U will have a bad distance d_U with high probability; however, this notion of masked distance may be slightly misleading as to the error correction capabilities of the masked code. The alternative masked distance definition Eq. (**??**) may be more accurate. Indeed, in Ref. [57] they provide an example where $d_U < d_{U_{\hat{N}}}$. As we will now show, we numerically observe decoding performance superior than a code with distance suggested by Corollary 9—even at masking percentages well above what is guaranteed by Theorem 7.

3.4 Numerical simulations

In this section, we report on the results of numerical simulations of a multi-round decoding protocol as described in Algorithm 2. Previous work has investigated the performance of HGP codes using a variety of decoders [92, 94, 103, 104, 128–131] (among many others) and gives evidence for competitive thresholds and good below-threshold performance. Here, we provide alternative evidence of exponential error suppression in both masked and unmasked cases following the methodology of Ref. [41].

We investigate a family quantum expander codes and decode them using the small-set flip decoding algorithm. As discussed in Section 2.3.1, quantum expander codes are families of hypergraph product codes where the base classical code(s) are randomly generated LDPC codes. We focus on square quantum expander codes where the Tanner graph of the single base classical code H is randomly generated and (Δ_V, Δ_C)-regular, meaning that each bit is involved in Δ_V checks, and each check is supported on Δ_C bits. Random classical codes generated in this way

are good expanders with high probability [78], and they have asymptotic parameters scaling like $[n, \Theta(n), \Theta(n)]$. The resulting HGP code, is $(\Delta_C, \Delta_V + \Delta_C)$-qLDPC (but not regular), and has parameters scaling like $[[n, O(n), O(\sqrt{n})]]$. Using the configuration model and edge swapping [94], we generate random $(5, 6)$-regular classical codes which yield a family of $(6, 11)$-qLDPC hypergraph product codes with rate $k/n \geq 1/61 \approx 0.016$. HGP codes—being CSS codes—can have bit- and phase-flip errors decoded independently. Furthermore, HGP codes constructed from a single base code have equivalent parity check matrices H_X, H_Z, and therefore, without loss of generality, we focus on the problem of decoding X-type errors. The results presented here correspond to a specific (un)masking *schedule*, which is a potential modification of Algorithm 2 and a way of specifying when and by how much to apply a mask to the syndrome. In particular, we study the following two models:

- *Simple scheduling*. Apply a mask D with a masking percentage of p_{mask} to use for all τ error correction rounds. After τ rounds, remove the mask completely and perform one error correction round with the fully unmasked syndrome.

- *Iterative scheduling*. Apply a mask D with masking percentage p_{mask}. After a multiple of 10^{t-1} rounds, for $t > 0$, remove $1 - 10^{-(t-1)}\%$ of the mask. For each of these instances, remove the same portion of the mask each time. On rounds $10^{t-1} + 1$, all generators that were temporarily unmasked in the previous round are re-masked until another 10^{t-1} rounds have passed. After τ rounds, again remove the mask completely and perform one round of error correction.

These two models are simplifications to the masking schedules that would be used when implementing the stacked model, in addition to the fact that they use random masks. Iterative

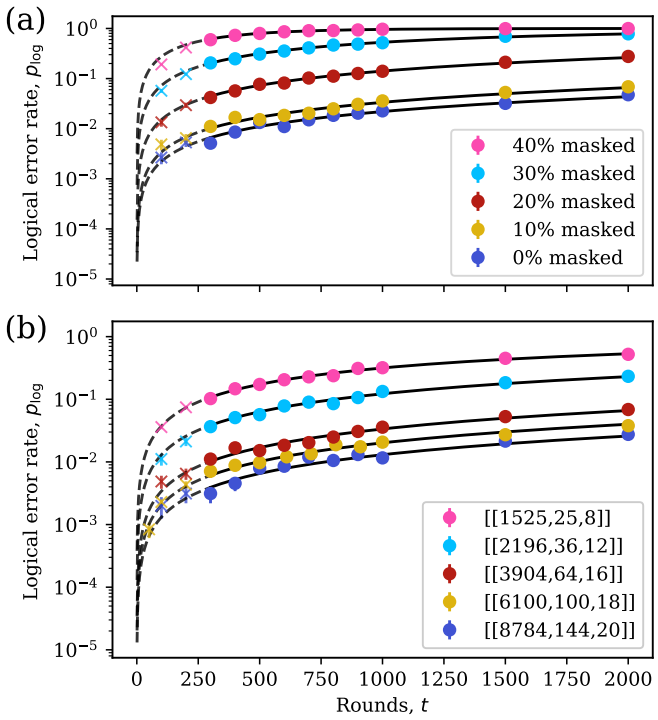

Figure 3.3: (a) Semilog plot of logical error rate as a function of the number of rounds for a $[[3904, 64, 16]]$ code and the simple unmasking schedule. (b) Logical error rate as a function of rounds across the $(6, 11)$-qLDPC code family with fixed $p_{\text{mask}} = 10\%$ and the simple unmasking schedule. Both panels include fits of Eq. (3.9), for which we only include data with $t \geq 300$.

scheduling is closer to what would be the case in reality, with the larger generators becoming available after some number of rounds. The precise timing in which this happens will depend closely on the code and the capabilities of the quantum computer, and it will likely be much different than the unmasking times studied here. Nonetheless, iterative scheduling is a step toward realism that allows us to see the benefits of unmasking more frequently.

In Fig. 3.3(a) we show the logical error rate, $p_{\log}$, as a function of the number of rounds for a $[[3904, 64, 16]]$ code and the simple unmasking schedule. Data is obtained by running Algorithm 2 for a fixed number of rounds with an error rate of $p = 0.001$ while varying the masking percentage, p_{mask}, and then recording the percentage of samples that end with a logical error. A sample is considered to end with a logical error if the final state is not equal to the initial state,

up to stabilizer elements. We extract the logical error per round, ϵ_L, by fitting the data to the exponential

$$p_{\text{log}} = 1 - (1 - \epsilon_L)^t. \tag{3.9}$$

The error bars on the fits are taken from the standard error of sampling a binomial distribution, $\sqrt{p_{\text{log}}(1 - p_{\text{log}})/N}$. In the bottom panel of Fig. 3.3, we now fix $p_{\text{mask}} = 0.1$ and show the performance of the simple unmasking schedule across the code family. The codes are labeled with their parameters as described in Section 2.3.1. While finding the distance of a code is generally hard, we are able to exhaustively search through the codewords of the base classical code to determine the distance of it, as well as the corresponding HGP code. Here, we observe even spacing between curves on the semilog plot showing exponential error suppression with code size. This behavior is more easily seen as the linear downwards trend in Fig. 3.4, which we now more precisely quantify.

We can relate a code family and values for logical error per round with an exponential error suppression factor Λ. For simple models, the equation

$$\epsilon_L = \frac{C}{\Lambda^{(d+1)/2}}, \tag{3.10}$$

where C is a fitting constant and d is the distance of the code, heuristically describes this relationship well. In Fig. 3.4(a) and (b), we show the logical error per round as a function of code distance for the simple and iterative schedules, respectively. For each masking percentage, we fit a linearized Eq. (3.10) with log ϵ_L to obtain Λ. These values are listed in Table 3.1. A value of $\Lambda > 1$ is a clear indication of operating below the threshold, as increasing the code size gives an

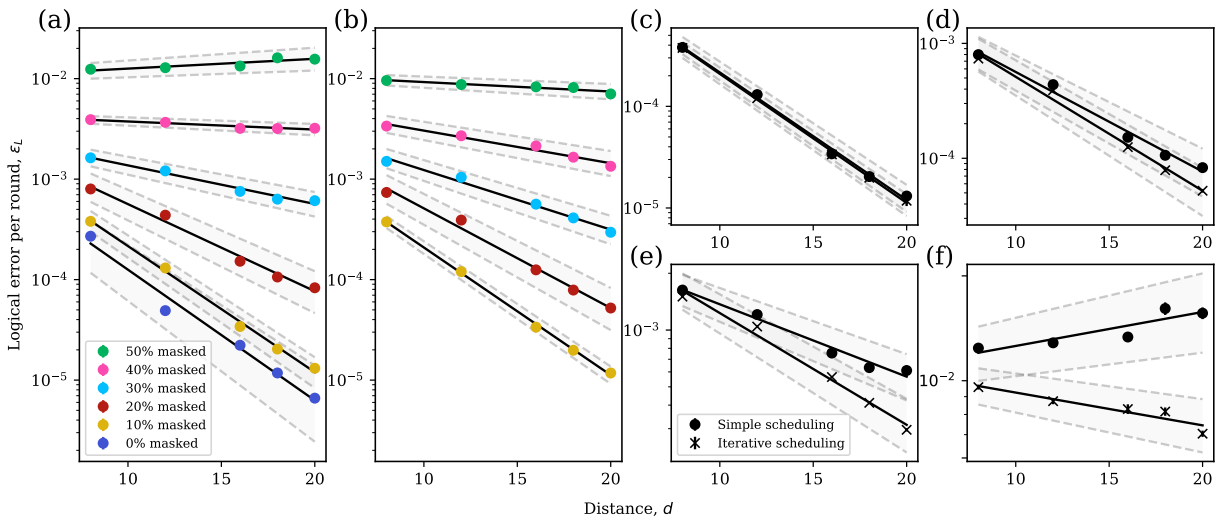

Figure 3.4: (a) Semilog plot of logical error rate per round, ϵ_L, as a function of code distance for the simple unmasking schedule and an error rate of $p = 0.001$. The fits are of a linearized Eq. (3.10) with $\log \epsilon_L$. (b) Similar results for iterative scheduling. Note that we do not include 0% masking in this case because it is equivalent to the simple schedule. Panels (c)-(f) plot the same data from panels (a)-(b) and provide easier comparisons between the simple (dot markers) and iterative (x markers) scheduling for $p_{\text{mask}} = \{10\%, 20\%, 30\%, 50\%\}$, respectively. The shaded region for all panels indicates error bars for C and Λ.

exponential decrease in the logical error rate per round. For simple scheduling, we find that for masking percentages below 50%, Λ is in this regime. Increasing p_{mask} decreases Λ, and between 40% and 50% we see a transition where $\Lambda < 1$. In this case, it is no longer advantageous to increase the code size, as it actually causes more logical errors to occur.

The results of the iterative unmasking schedule are shown in Fig. 3.4(b), where we find that it outperforms the simple schedule. For smaller masking percentages, it is not as advantageous to use a schedule with more unmasking, as there is less difference in performance between small masking percentages and completely unmasking (see Fig. 3.3(a)). However, larger masking percentages appear to benefit more from using a more frequent unmasking schedule. In fact, with iterative scheduling, it is now the case that 50% masked is back in the $\Lambda < 1$ regime, although with very little error suppression. Fig. 3.4(c)-(f), highlights this difference between schedules.

p_{mask}	Simple scheduling	Iterative scheduling
0%	1.820 ± 0.046	-
10%	1.782 ± 0.019	1.794 ± 0.010
20%	1.490 ± 0.026	1.579 ± 0.026
30%	1.193 ± 0.015	1.314 ± 0.018
40%	1.038 ± 0.007	1.161 ± 0.015
50%	0.956 ± 0.014	1.044 ± 0.009

Table 3.1: Extracted values of Λ for different masking percentages and schedules.

In both cases, we find that the results exceed the guarantees provided by Theorem 7. We find that the percolation threshold of this family of $(12, 10)-$qLDPC codes is around 2%; however, we see exponential error suppression at error rates up to $\sim 50\%$, well above this threshold.

3.4.1 2D hyperbolic surface codes

As a comparison, we benchmark the performance of a 2D hyperbolic surface code on the multi-round decoding protocol. Although codes based on tilings of closed hyperbolic surfaces have a comparatively poor asymptotic distance, $d = \Theta(\log n)$, they have a constant encoding rate. These parameters violate the Bravyi-Poulin-Terhal bounds [109], and therefore embedding these codes in 2D Euclidean space is not possible without nonlocal connections. However, they are in some sense close to being local, and so they make a good candidate for the stacked model. For the construction and threshold simulations of these codes, we point the interested reader to Ref. [119]. As the SSF decoder is not known to work for 2D hyperbolic surface codes, we instead use the minimum-weight perfect matching (MWPM) decoder [132]. While we no longer have the guarantees of Theorem 7, the MWPM decoder can be modified to work with masked stabilizer generators. To do this, we set the nodes corresponding to masked generators as boundaries in the

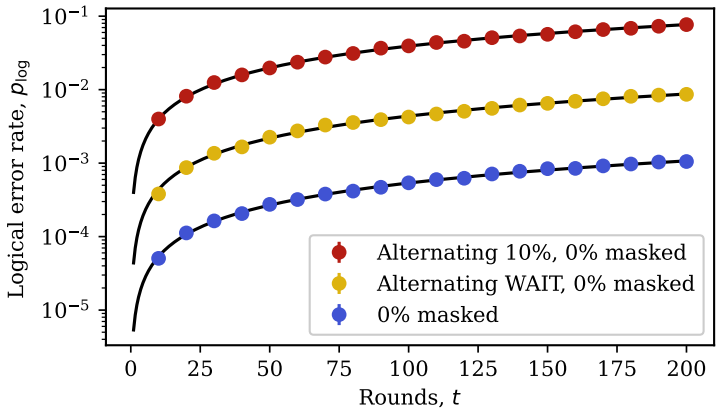

Figure 3.5: Semilog plot of logical error rate as a function of the number of rounds for a $[[360, 25, 8]]$ 2D hyperbolic surface code and an error rate of $p = 0.003$. We compare fully unmasked decoding performance (blue markers) with two iterative unmasking schedules. Yellow markers denote a schedule consisting of alternating between a round where no error correction is performed and a round where the entire syndrome is available. Red markers denote a schedule where masks of 10% and 0% are used to decode, alternating each round. Fits are of Eq. (3.9).

matching graph and set the corresponding syndrome bits to zero. Decoding normally, it is then possible to match unpaired syndrome nodes to the boundary. Note that the standard solution to decoding with syndrome noise of building a 3D matching graph with a time dimension does not work since the mask is fixed from round to round, and the repeated measurements provide no additional information.

The code we investigate comes from a family of $\{5, 4\}$-codes with an asymptotic rate of $1/10$ and has parameters $[[360, 38, 8]]$. In Fig. 3.5 we show the results of running Algorithm 1 with this code and an error rate of $p = 0.003$ for several iterative unmasking schedules. We compare completely unmasked decoding (blue markers) with a schedule that alternates between performing no error correction and 0% masked each round (yellow markers) and one where the masking percentage alternates between 10% and 0% masked each round (red markers). For these codes and decoder, we find that it is actually better to do nothing and let the errors accumulate rather than try to correct the errors with the partial syndrome. We note that we did not observe

this behavior for HGP codes, even at the higher error rate. This result is interesting as it seems to imply that the masking behavior for HPG codes is non-trivial.

One possible explanation for the difference is the single-shot [59, 60] property of quantum expander codes the SSF decoder, a property not found in hyperbolic surface codes and the MWPM decoder. Intuitively, this means that the syndrome has redundancies that make it more resilient to syndrome errors and masks. Over a multi-round decoding procedure, the single-shot property also ensures that the size of any residual error is proportional to the size of the syndrome error. Consequently, misdiagnosing an error cannot have immediate effects throughout the system, since the size of the resulting error is bounded. This is not the case with the MWPM decoder, where a well-placed syndrome error could result in a long error chain across the lattice.

3.5 Discussion

In this chapter, we introduced the concept of partial syndrome measurement and formalized it with masking and (un)masking schedules. Applying this new technique, we motivated a new practical protocol based on the stacked model for implementing nonlocal qLDPC codes on quantum hardware restricted to 2D local gates. As a first step into validating the potential of such a scheme, we investigated the ability of performing error correction while using a subset of the total syndrome information. For this simplified model we provided analytical and numerical evidence that quantum expander codes still exhibit a threshold. Additionally, we showed that this robustness is not inherent to quantum codes in general–hyperbolic surface codes perform poorly in this setting. We hypothesized, but did not prove, that a robustness to masking is related to the single-shot properties of the code and decoder.

The model we studied in this chapter was a toy model that lacked many factors which would need to be considered in a physical implementation. There are a number of questions that need to be answered to determine whether this procedure is feasible in general.

What families of codes are amenable to the stacked model? Given that quantum expander codes are no longer allowed in the stacked model, we need an alternative. Namely, we need a code family that satisfies the bounds of Ref. [115], has good masked error correcting performance, and has embeddings into $\mathbb{Z}^2$ amenable to the stacked model. Certain codes, such as generalized bicycle codes [85, 86] or certain hypergraph product codes *not* constructed from good classical expanders [123, 124] have convenient embeddings; however, the problem of optimally embedding an arbitrary Tanner graph is likely computationally intractable [133].

What do the syndrome extraction circuits look like for the stacked model? Careful thought has to be put into the syndrome extraction circuit to ensure that we do not fall into the same pitfall of accumulating too many errors while the nonlocal generators are being prepared. A naive syndrome extraction circuit consisting of SWAP gates will take $\omega(1)$ time to prepare generators of size $\omega(1)$, which is prohibitively long. Alternatively, one could use the syndrome extraction circuits of Ref. [103]; this method solves the scaling issue by utilizing ancilla qubits and gate teleportation to perform long-range CNOT gates in constant depth. Remaining technicalities include the use of entanglement distillation [70, 134] to ensure the resulting long-range CNOT gates are of high enough fidelity.

How long does it take to perform a set of masked syndrome measurements? As discussed in the previous question, performing the syndrome extraction of a single generator can be accomplished in constant time. However, when restricted to $O(n)$ ancilla qubits, the same cannot be said for a growing number of nonlocal generators. Bounds on the depth of 2D local circuits

needed to measure the *full* syndrome of a stabilizer code were developed in Ref. [103]. Extending these bounds to include specifying generator size distributions will help inform explicit unmasking schedules, which may provide better performance than the arbitrarily chosen ones studied in this work. These three questions form the basis for a practical implementation of the stacked model and are the focus of the next chapter.

While this chapter was originally motivated by implementing nonlocal QECCs on 2D local hardware, we can also consider applying the proposed methods in order to reduce overheads on quantum architectures like neutral atoms, ion traps, and semiconductor spin qubits that have the ability to implement long-range gates through qubit movement [3, 37, 43, 45, 104, 135, 136]. With this functionality these devices are able to achieve all-to-all connectivity, in turn facilitating the use of efficient, nonlocal qLDPC codes without significant additional overhead. To reduce the cost of performing error correction on these architectures we could simply reduce the number of generators that we measure: if we can achieve comparable logical error rates while measuring only $\sim 80\%$ of the generators, then we have obtained a $\sim 20\%$ shorter QEC cycle effectively for free. This specific idea also partially motivates the application of adaptive syndrome extraction which we discuss in Chapter 5.

One way to accomplish this would be to randomly choose a subset of generators to measure in each error correction cycle. The toy model studied in this chapter did essentially that, with the difference being the subset of generators measured was consistent across QEC rounds. Alternatively, we could choose a different subset for each round, while still using an unmasking schedule in which there were rounds where the full stabilizer group is measured. For quantum expander codes we can prove using a variation of the arguments in Section 3.3 that this alternative scheme has a threshold as well. More specifically, over τ error correction cycles, we can

neglect to measure some random constant fraction p_{mask} of generators each round and still expect a logical fidelity of

$$p_{\log} = \Omega\big(1 - (\tau + 1)e^{-\Theta(\sqrt{n})}\big), \tag{3.11}$$

as long as p_{mask} and the physical error rate are below some threshold. However, doing this likely comes at the cost of a reduction in robustness for some QECCs; for example, consider codes which have single-shot QEC facilitated by soundness [60]. Soundness is due to redundancies in the stabilizer generators called *metachecks* which allow for an increased tolerance syndrome errors [46, 95, 100]. Obtaining fewer syndromes would restrict the evaluation of these metachecks, removing the ability to detect and correct syndrome errors. In this case, the savings from masking may not outweigh the loss of soundness.

This does not mean that the stacked model is completely irrelevant for architectures with long-range connectivity. Since movement adds additional complications associated with qubit decoherence, heating, and loss, it is worthwhile to consider schemes that limit the amount of movement such as those developed in this and the next chapter. In the extreme case, one can consider qubits that are fixed in space and solely use local interactions to perform entangling gates. Such studies provide additional insight into the tradeoffs associated with engineering long-range connectivity through qubit motion or more complex electrical wiring.

Chapter 4: 2D local implementations of nonlocal codes

In the previous chapter we introduced locality and its relationship with the parameters of quantum error correcting codes. We noted that the surface code, despite showing promising theoretical and experimental performance on 2D local architectures [38, 39, 44, 102, 137, 138], is poorly suited to large-scale fault-tolerant quantum computation due to its vanishing rate [137, 139, 140]. For a more resource-efficient alternative, we presented qLDPC codes which surpass the parameters of the surface code [75]. As these codes can encode multiple logical qubits, the required space overhead is reduced, in some instances, to a constant [24]. The drawback, as we described, is that these high-rate qLDPC codes require many long-range connections [108, 109, 113, 114], posing difficulties for architectures such as superconducting qubits which can only perform entangling gates between 2D nearest neighbor qubits.

Several recent proposals have attempted to alleviate this overhead by taking advantage of more complex electrical wiring of the superconducting circuits [85, 130], employing code concatenation [141, 142], or using bosonic cat qubits [143]. Implementing these long-range connections is less problematic in architectures like neutral atoms, ion traps, or semiconductor spin qubits that can implement long-range gates through qubit movement [3, 37, 43, 45, 104, 135, 136]. However, as we briefly mentioned, movement adds additional complications associated with qubit decoherence, heating, and loss, and it may be worthwhile to minimize these operations.

In this chapter, we present a concrete approach to qLDPC codes that works without qubit motion or long-range couplers, inspired by the stacked model introduced in Chapter 3. As a reminder, we assume that high-rate qLDPC codes in the stacked model have the property that after embedding the code into $\mathbb{Z}^2$, the majority of the stabilizer generators are local; that is, their qubits are contained within a ball of constant radius. We claimed that most of the work required to perform the syndrome extraction circuit with 2D local gates comes from measuring the few nonlocal generators, so measuring these generators less frequently has the potential to significantly reduce the time overhead, ideally at only a minor cost to the error correction performance of the code. We showed in Chapter 3 that, for quantum expander codes, neglecting to measure a large percentage of generators could be reasonably tolerated; however, the model we considered was narrow in scope, considering only a phenomenological noise model and neglecting the problem of embedding the codes. It is therefore unclear whether such codes lend themselves well to physical implementations. Nonetheless, the results on partial error correction were optimistic and motivated the investigation of the more realistic architecture developed in this chapter.

We propose and benchmark a realistic bilayer architecture suited for near- to mid-term superconducting devices and other platforms with restricted qubit movement. We find that the recently introduced bivariate bicycle (BB) qLDPC codes [85] coming from the larger family of generalized bicycle qLDPC codes [86] are well suited for both the stacked model and the bilayer architecture. These codes have natural embeddings into $\mathbb{Z}^2$ where the generators have a repeated structure, and in some instances, a majority of the generators are geometrically small. The first property makes them amenable to a parallel syndrome measurement scheme using routing with fast local operations and classical communication (LOCC), and the second property makes them good candidates for reducing overhead using the stacked model. More generally, we develop

bounds on how quickly syndrome extraction can be performed in this manner and provide an algorithm to do so. Overall, we find that over multiple rounds of decoding, BB codes implemented in this architecture have error correction performance comparable to the standard (rotated) surface code, albeit only when the parameters in the error model lie in certain regimes.

Our work stands as an alternative architecture that may be more practical for near-term quantum computers without the ability to move qubits. As such, it is incomparable to schemes such as Ref. [104, 135] which allow for qubit movement. Several recent works have also proposed layered architectures [85, 130]; however, their motivation is in minimizing the number of crossings in the two-qubit gate connectivity. They achieve this through the use of long-range connections, the elimination of which is the main imposed constraint of our work. Refs. [141, 142] present asymptotically well performing concatenated schemes which use only local connectivity; however, the required overheads likely make them infeasible for near- and mid-term quantum computers. In particular, Ref. [142] estimates that $\sim$600 physical qubits would be needed per each logical qubit, which is an order of magnitude more than what our architecture needs to implement the $[[144, 12, 12]]$ Gross code [85], with $\sim$48 physical qubits per logical qubit. Most closely comparable to our work is Ref. [103], which aimed to implement quantum expander codes with local connectivity by using a similar teleportation-based scheme. Whereas they arrived at a negative result, the innovations in code choice, partial error correction, and syndrome extraction using entanglement purification presented here allow us to obtain more favorable performance. We also note that Ref. [25] analytically studied a scheme similar to ours in which long-range connections are facilitated by long-range entanglement. They prove that such a scheme can be fault-tolerant, albeit with polynomial overhead. In general, our approach may be easier to implement as it only requires a bilayer architecture, local connectivity, and relatively few, $O(n)$, qubits.

It, of course, also comes with challenges, which we later discuss.

The chapter is structured as follows: we first introduce the quantum computing architecture and routing assumptions we consider throughout the chapter and contextualize them with respect to the stacked model. In Section 4.4, we develop lower bounds on the routing time for our specific routing model and provide a greedy algorithm to use in implementations. Section 4.5 develops an error correction protocol built on a bilayer architecture and culminates with circuit-level simulations comparing the performance with the rotated surface code in Section 4.6. We conclude in Section 4.7 with a discussion.

4.1 Architecture

In this chapter, we consider a quantum computing architecture where qubits are located on the vertices of an $M \times M$ grid, where $M = \Theta(\sqrt{n})$. As is natural for current superconducting qubit platforms, we assume that two-qubit gates can only be performed between neighboring qubits on the grid. Any two-qubit gate which interacts qubits that are not neighboring is considered a *long-range* gate. Circuits that do not have access to long-range gates are called *2D local circuits*, and architectures that are restricted to these circuits are called *2D local architectures*. This definition generalizes to architectures based on graphs other than the grid: given a connectivity graph $G = (V, E)$ with data qubits located on the vertices, the only allowed two-qubit gates are those between qubits $u, v \in V$ that share an edge $(u, v) \in E$. Similar restrictions arise if we disallow the slow movement of atoms in neutral-atom devices, in which case the only available two-qubit gates are those performed through Rydberg-Rydberg interactions. This leads to an architecture that can perform entangling gates on qubits that are some distance R away, where R

depends on the capabilities of the device. We do not investigate this ability in this chapter, but we discuss it in Section 4.7.

Implementing general quantum circuits on real architectures requires compilation into a form that respects the connectivity constraints of the device. For the 2D local architecture we consider here, performing two-qubit operations on qubits that are not adjacent requires permuting them to be so. Doing this with swap gates requires a circuit depth proportional to the distance between the qubits. To implement stabilizer generator measurements like those shown in Fig. 2.3, this means that each data qubit must be moved to a position where it can interact with the check qubit, so one must wait for these permutations to complete before the eigenvalue can be measured. This somewhat defeats the purpose of using qLDPC codes, since a single syndrome can no longer be extracted with a constant-depth circuit. As such, it is infeasible to perform long-range stabilizer generator measurements in this way, and we instead focus on an alternative method.

4.2 Teleportation routing

Routing is the task of permuting packets of information, or tokens, on the vertices of a graph, using only interactions on edges of the graph. In quantum routing, the tokens are qubits, and the graph is specified by the architecture's connectivity constraints. Classical approaches to routing are typically built from swap gates [144–146], which can also be applied naturally to routing quantum data [147, 148]. However, more general quantum operations can enable faster routing. In particular, measurement and classical feedback enable the use of entanglement swapping to distribute entanglement and perform quantum teleportation, which can achieve speed-ups over swap-based routing for many permutations and underlying graphs [149–151], with applica-

tions including error correction [152].

We assume the *LOCC routing* model described by Devulapalli *et. al.* [149], where arbitrary single-qubit and disjoint two-qubit quantum gates can be implemented in a single time step, and we have access to fast single-qubit, mid-circuit measurements, and fast classical control of single-qubit gates. Additionally, there are a constant number of ancillary qubits for each data and check qubit, connected as attached ancillas [111, 112] or through stacked vertical layers (see Section 4.5.1). In LOCC routing, we can perform protocols such as entanglement swapping [153] and teleportation in constant depth. A specialization of LOCC routing that focuses on qubit and gate teleportation [154] is teleportation routing. During a single round of teleportation, we perform parallel entanglement swapping along multiple teleportation paths. Each vertex can be involved in at most a constant number of paths, as we allow a constant number of ancillary qubits per vertex. In this chapter, we assume only one ancilla per data qubit and use the stacked vertical layers model. This model allows direct implementation of gates between ancillas and their corresponding data qubits, as well as between ancillas whose data qubits are also directly connected (see Fig. 4.1(b)).

To perform long-range two-qubit gates, it is not necessary to actually teleport the participating qubits to adjacent locations; instead, it suffices to use the teleportation paths to implement a long-range gate with gate teleportation. The circuit shown in Fig. 4.1(a) allows us to implement arbitrarily long CNOT gates in constant quantum depth, avoiding depth overhead of swap routing and any need to reverse the operation. At the cost of utilizing ancillary qubits, this lets us extract the syndrome of a single nonlocal generator using only a constant-depth circuit.

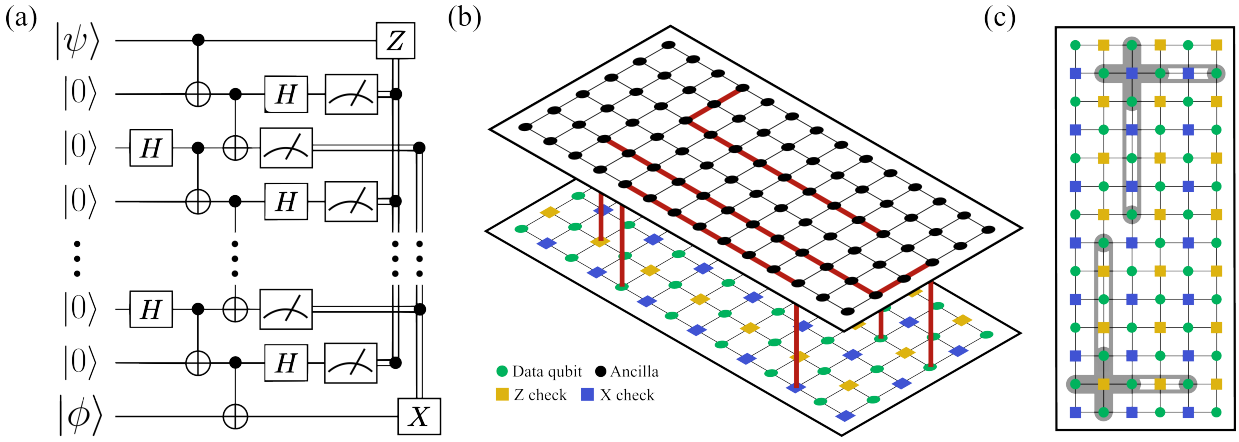

Figure 4.1: (a) Circuit to teleport a CNOT gate through a chain of n qubits using only 2D local gates. The depth of the circuit is constant regardless of the length of the chain. (b) Proposed architecture to implement nonlocal high-rate qLDPC codes using only 2D local gates. The architecture consists of two qubit layers: the bottom layer contains the data qubits and ancilla qubits allocated to perform syndrome measurements, while the top layer contains extra ancilla qubits used to perform long-range CNOT gates. Each layer has only 2D local connections, and the only connections between the layers are between qubits that are vertically adjacent. To perform a CNOT gate on two spatially distant qubits, the circuit from (a) is used along the paths of qubits highlighted in red. Multiple long-range CNOT gates may be performed in parallel, as long as the paths act on disjoint sets of qubits. (c) Example embedding for a $[[42, 12, 2]]$ BB (error detecting) code constructed with $\ell = 7, m = 3$ and by matrices $A = 1 + y^2 + y$, $B = 1 + x^5 + x$. The check structure, which is identical for all checks of both types up to mirroring, translation, and boundary conditions, is highlighted in gray.

4.3 Implementing the stacked model

An important consideration for the stacked model as described in Section 3.1 is the specific assignment of physical qubits in the architecture to data and check qubits in the code, which can be considered a type of qubit placement [155] or qubit allocation [156]. This assignment can be thought of as an *embedding* of the Tanner graph of the code in the architecture, where an embedding for a graph $G = (V, E)$ is a map $\eta : V \to \mathbb{Z}^D$. As an example, the Tanner graph for the surface code has a natural embedding into $\mathbb{Z}^2$ that allows for all of its generators to act on qubits within a constant radius; however, one could instead assign data and check

qubits to physical qubits randomly, yielding generators that still have weight four, but are no longer local. The difficulty of implementing syndrome extraction circuits is closely related to the chosen embedding. Certain codes, such as hypergraph product codes and bivariate bicycle codes discussed in Section 2.3.1, Section 2.3.2, respectively, have natural embeddings into $\mathbb{Z}^2$. We make use of this fact for bivariate bicycle codes in this chapter, and hypergraph product codes in the next chapter.

To study the impact of nonlocality on the cost of performing syndrome measurement, we must quantify the notion of generator size and size frequency. We parameterize the size of a given generator as M^γ, where $0 \leq \gamma \leq 1$ and M is the linear size of the grid. For local generators, $M^\gamma = O(1)$ implies a constant interaction radius, while the largest generators can have interaction radii $\frac{\sqrt{2}}{2}M \in \Theta(M)$ (i.e., $\gamma = 1$). For stabilizer codes, the number of independent stabilizer generators r is related to the number of physical and logical qubits in the code like $n - r = k$. For constant-rate codes, there are thus $O(n) = O(M^2)$ independent generators, which can be parameterized like $M^{2\beta}$, with $0 \leq \beta \leq 1$. With $\beta = 1$, we are considering the problem of measuring every generator, and with $\beta < 1$ we only consider some subset. We can describe the set of generators as a whole by defining a function $f(\gamma)$ to characterize the distribution of generators having size M^γ. The only constraint on $f(\gamma)$ is that it is a valid probability distribution over the domain of γ, $\int_0^1 f(\gamma)d\gamma = 1$. In practice, $f(\gamma)$ will depend on the architecture, embedding, and parameters of the code family of interest [113–115].

A rough estimate of the amount of work required to perform the syndrome extraction circuits for a given set of generators is simply to count the two-qubit gates, which in many cases is the leading contributor to the error budget. In our routing model, this value is proportional to the total length of the teleportation paths when implementing long-range CNOT gates, which can be

approximated as

$$\text{total path length} \approx M^2 \int_0^1 f(\gamma) M^\gamma d\gamma. \tag{4.1}$$

Here, the M^2 factor comes from the fact that there are $O(M^2)$ generators to measure in total, and a single generator of size γ requires a path length of M^γ. If we choose to only measure generators below a certain size γ', this corresponds to simply evaluating the integral up to γ'. We might also want to consider measuring the smallest $x\%$ of generators, in which case one can solve $x = 100 \int_0^{\gamma'} f(\gamma) d\gamma$ to find the appropriate value of γ' and then proceed in the same way.

4.4 Routing bounds

Previous work by Delfosse *et al.* [103] developed lower bounds on the depth of Clifford circuits required to measure commuting Pauli operators. In this section, we derive similar bounds taking advantage of additional information about the geometric size of the operators. These bounds do not hold in general, but are instead specific to the teleportation routing model discussed in Section 4.2. We assume there is a fixed layout of the data and check qubits that gives rise to a specific generator size distribution $f(\gamma)$. This is to avoid scenarios such as scrambled surface codes, where the difficulty of implementing the syndrome extraction circuits could be greatly reduced by permuting the qubits.

Claim 10. *Let C be a 2D local circuit measuring $M^{2\beta}$ commuting Pauli operators whose radii are greater than M^γ after embedding them in an $M \times M$ grid. Then for teleportation routing,*

$$\textit{depth}(C) = \Omega\big(M^{2\beta+\gamma-2}\big). \tag{4.2}$$

Proof. In our routing model, the maximum total length of the teleportation paths in a single time step is $O(M^2)$ since only a constant number of ancillary qubits per data qubit are allowed, and there are $\Theta(M^2)$ edges in the grid graph. The cost of measuring an operator of size $\Omega(M^\gamma)$ is dominated by implementing the long-range CNOT gate between its two furthest qubits. Although this can be done in constant depth using a dynamic circuit (Fig. 4.1(a)), it requires a teleportation path of length $\Omega(M^\gamma)$. Consequently, routing and measuring this one operator uses $\Omega(M^\gamma)$ edges of the $O(M^2)$ available edges. Measuring all $M^{2\beta}$ operators thus requires $\Omega(M^{2\beta+\gamma})$ edges. In the best case, we utilize all available edges in each circuit layer, giving a circuit depth of $\Omega(M^{2\beta+\gamma-2})$. □

In practice, it will often be the case that the edges are not optimally used, as illustrated in Fig. 4.2. We can extend this idea to the general case of an arbitrary distribution of generator sizes.

Claim 11. *Let C be a 2D local circuit measuring $M^{2\beta}$ commuting Pauli operators whose radii follow a probability distribution $f(\gamma)$ after embedding them in an $M \times M$ grid. Then for teleportation routing,*

$$\textit{depth}(C) = \Omega\big(M^{2\beta-2} \int_0^1 f(\gamma) M^\gamma d\gamma\big). \tag{4.3}$$

Proof. Just as in Claim 10, we can lower bound the circuit depth by summing the lengths of the teleportation paths required to measure the set of operators. We now have operators of different sizes, where the fraction of operators of a certain size is determined by the probability distribution $f(\gamma)$.

Thus, for a given γ, there are a number of operators proportional to $f(\gamma)M^{2\beta}$ that each require M^γ edges to measure. Since $0 \leq \gamma \leq 1$, the total teleportation path length needed to

route and measure every operator is

$$M^{2\beta} \int_0^1 f(\gamma) M^\gamma d\gamma. \tag{4.4}$$

Since we again have $O(M^2)$ edges in the grid available in each layer of the circuit, the total circuit depth is lower-bounded as in Eq. (4.3), as desired. □

4.4.1 Greedy routing

Swap routing is a straightforward approach to compiling circuits for quantum hardware with interaction constraints. Practically, this can be done using an algorithm that tries to perform the circuit using as few swap gates as possible [147, 148, 157, 158]. As mid-circuit measurement and long-range entanglement generation become more reliable [159], teleportation routing may become a more viable option to move qubits and perform long-range gates. Here, we present a simple, greedy algorithm to route an arbitrary set of operators under the routing and architecture assumptions of Sections 4.2 and 4.1, respectively. An operator consisting of a tensor product of single-qubit Paulis, such as a stabilizer generator, can only be measured once each qubit in its support has been routed. That is, a teleportation path is prepared and a long-range entangling gate is applied between the qubit and a readout ancilla qubit. Once all required gates have been applied, the operator is said to have completed routing, and the readout qubit can be measured to obtain the eigenvalue of the operator. The algorithm is described in Algorithm 3.

The circuit operations of a single iteration can be executed in parallel, so each iteration performs only a constant-depth circuit. Therefore, the total circuit depth of the routing procedure is proportional to the number of iterations. Instead of minimizing gate count, the intent of this

Algorithm 3 Greedy routing

while there are still operators to measure **do**
    Sort the operators in decreasing order according to how many of their qubits have completed routing.
    for incomplete operator $o_i = 1, 2, \ldots$ **do**
        for qubits $j = 1, 2, \ldots$ of operator o_i **do**
            Use breadth-first search to find a teleportation path for qubit j to the corresponding readout ancilla qubit.
            If no path exists, continue.
        end for
    end for
    Perform long-range entangling gates on qubits that found a teleportation path.
    Measure the readout qubit of operators that have completed routing.
end while

algorithm is to minimize the circuit depth—and saturate the bound of Claim 11—by maximizing the usage of teleportation paths. This is only possible if the partial measurements between iterations commute, such as when measuring the generators of a single type in a CSS code, in the standard surface code syndrome extraction circuit [160], or in the depth-7 BB code measurement circuit [85]. For simplicity, the syndrome extraction circuits we use route every Z-type check and then route every X-type check.

To benchmark the performance of the algorithm, we draw random examples of BB codes (see Section 2.3.2) and route the X-type generators while restricted to a single layer of ancillary qubits. For comparison, we compute the optimal routing depth according to Claim 11. Figure 4.2 shows the results of these simulations, providing evidence that the greedy routing algorithm nearly saturates Eq. (4.3). To obtain the constant factor in Fig. 4.2, we consider the smallest eight code instances and perform a fit between the asymptotic lower bound and the depth returned from the greedy routing algorithm. This constant times the theory lower bound matches closely with the routing time of small code instances; however, we begin to see the algorithm routing depth deviating as we increase the block length, indicating non-optimal performance. For code

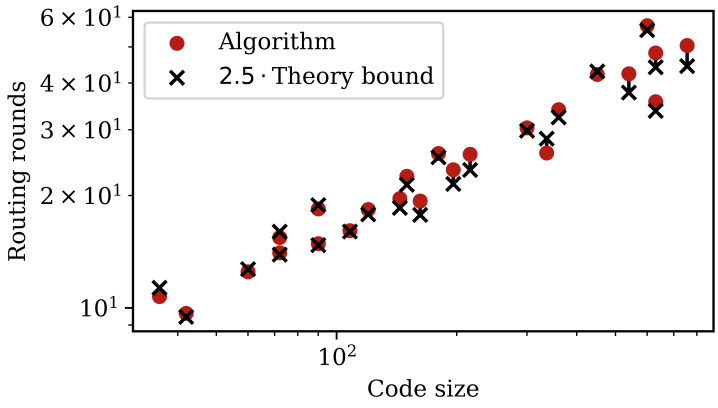

Figure 4.2: Depth from greedy routing versus $2.5\times$ the theoretical optimal depth to route the X-type generators using a single layer of ancillary qubits. Code examples are drawn randomly from the family of BB qLDPC codes (see Section 2.3.2).

sizes of practical interest, this algorithm may be a viable option to optimize teleportation routing. Certain codes, such as BB codes have additional structure that allows us to manually find routing schedules that outperform those found by the greedy algorithm.

4.5 Bilayer architecture

As detailed in Section 4.1, the main difficulty in implementing nonlocal qLDPC codes on 2D local architectures is the need to perform nonlocal two-qubit operations. To address this issue, we propose a physical implementation based on the teleportation routing model described in Section 4.2. While the proposed implementation is code agnostic, we focus on bivariate bicycle codes. We first give a brief overview of embedding BB codes into $\mathbb{Z}^2$. See Ref. [85] for a more complete discussion.

For certain choices of A_i and B_i, the resulting BB code has a *toric layout*:

Definition 12 ([85, Definition 1])**.** *A code $BB(A, B)$ has a toric layout if its Tanner graph has a spanning sub-graph isomorphic to the Cayley graph of $\mathbb{Z}_{2\mu} \times \mathbb{Z}_{2\lambda}$ for some integers μ, λ.*

This is to say that codes with a toric layout have checks that act on the four nearest-neighbor qubits, and potentially on additional nonlocal qubits. The four nearest-neighbor qubits can be measured using a standard surface code syndrome extraction circuit [160], whereas the nonlocal qubits must be measured using nonlocal interactions. In the following, the order of an element $\mathrm{ord}(M)$ of a multiplicative matrix group is the smallest positive integer such that $M^{\mathrm{ord}(M)} = \mathbb{I}$, where $\mathbb{I}$ is the identity matrix of the same dimension as M.

A BB code $BB(A, B)$ depends on choices of matrices A and B, as in Eq. (2.39), whose terms are powers of x or y, defined in Eq. (2.38). The matrices x and y depend on choices of positive integers ℓ, m, and they correspond to the dimensions of the grid in which the code $BB(A, B)$ is embedded should it satisfy Lemma 13. The μ and λ of Definition 12 are ℓ and m, respectively. In this toric layout, qubits and checks can be labeled by $\mathcal{M}$, which can be considered to be a list of integers $\mathbb{Z}_{\ell m} = \{0, 1, \ldots, \ell m - 1\}$ that represent locations on the 2D grid.

Lemma 13 ([85, Lemma 4])**.** *A code $BB(A, B)$ has a toric layout on a $2\ell \times 2m$ grid if there exist $i, j, g, h \in \{1, 2, 3\}$ such that*

1. $\langle A_i A_j^T, B_g B_h^T \rangle = \mathcal{M}$
2. $\mathrm{ord}(A_i A_j^T)\,\mathrm{ord}(B_g B_h^T) = \ell m$

Here, $\langle A_i A_j^T, B_g B_h^T \rangle$ indicates the group generated by $A_i A_j^T$ and $B_g B_h^T$. The matrices $A_i A_j^T$ and $B_g B_h^T$ then correspond to horizontal and vertical translations, respectively, on the grid. To have a toric layout, these translations must visit the ℓm X- and Z-type checks, as well as the two sets of ℓm data qubits. Practically, this can be checked by multiplying $(B_g B_h^T)^b (A_i A_j^T)^a$ for $0 \leq b < \mathrm{ord}(B_g B_h^T)$, $0 \leq a < \mathrm{ord}(A_i A_j^T)$ with a basis vector of $\mathbb{F}_2^{\ell m}$ and seeing whether the other $\ell m - 1$ basis vectors can be obtained. Satisfying this is equivalent to satisfying condition 1.

For a given choice of $A = A_1 + A_2 + A_3$ and $B = B_1 + B_2 + B_3$, there might not be assignments of i, j, g, h such that Lemma 13 is satisfied. There may also be several satisfying assignments. Notably, each satisfying assignment yields an embedding with a defined generator shape.

This repeated parity check structure of BB codes gives them embeddings that are amenable to the stacked model: given one check, other checks of the same type can be obtained with vertical and horizontal shifts on the grid, up to periodic boundary conditions. Opposite-type checks are obtained by mirroring and again performing horizontal and vertical shifts. Fig. 4.3 shows an example of an embedding for a $[[120, 8, 8]]$ code constructed with $\ell = 12, m = 5$ and by matrices $A = x^{10} + y^4 + y$, $B = 1 + x + x^2$. The check structure for the weight-6 X- and Z-type generators is indicated by the gray outline. These natural embeddings make it straightforward to search for codes where the check structure is geometrically small. While the checks are not entirely local due to the two nonlocal qubits in their support, appropriately choosing ℓ and m can make the periodic boundary conditions induce generators that are comparatively much larger. This can be done by letting $\ell \gg m$, as illustrated in Fig. 4.3. In the resulting generator distribution, the majority of the checks are geometrically small. In the context of the stacked model, the generators that are induced by the boundary conditions are those that are measured less frequently.

The architecture, as depicted in Fig. 4.1(b), consists of two layers of qubits. The bottom layer contains the data qubits and ancillary qubits to perform syndrome measurements (check qubits), laid out using an embedding that maximizes decoding performance while minimizing the number of long-range generators. The top layer contains ancilla qubits to aid in the implementation of long-range CNOT gates. In each layer, the only allowed two-qubit operations are between neighboring qubits, and operations between layers are only allowed between qubits that are vertically adjacent, i.e., at the same (x, y) location.

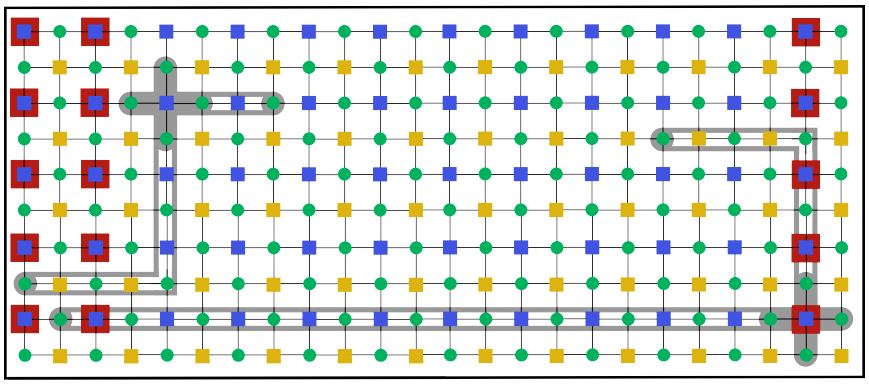

Figure 4.3: Illustration of the long- and short-range generators of a single type for the $[[120, 8, 8]]$ BB code constructed with $\ell = 12, m = 5$ and by matrices $A = x^{10} + y^4 + y, B = 1 + x + x^2$. The qubits contained in the check are highlighted in gray. The checks (blue squares) highlighted in red are the long-range checks, as they cross the long boundary. For both Z- and X-type checks, there are 12 long-range checks out of a total of 48, yielding a mask percent of 25%.

A bilayer architecture is a feasible design requirement for several types of quantum computers. As discussed in Ref. [85], it is difficult, yet not unreasonably so, to modify the current generation of superconducting hardware to support a second layer. In movement-restricted neutral-atom devices, one option is to use dual-species Rydberg arrays [161–163], where the data layer is made up of one species, and the ancilla layer the other. Alternatively, for single-species arrays, it may be practical to store multiple qubits per atom, using a combination of nuclear and electronic [164, 165] or motional qubits [166].

$[[n,k,d]]$	ℓ, m	A	B	Embedding	Mask percent	Routing steps
$[[72,8,6]]$	12,3	$x^9+y^1+y^2$	$1+x^1+x^{11}$	$\langle A_2A_3^T, B_1B_2^T\rangle$	25%	11,6
$[[90,8,6]]$	9,5	x^8+y^4+y	$y^5+x^8+x^7$	$\langle A_2A_3^T, B_2B_1^T\rangle$	22.22%	9,5
$[[120,8,8]]$	12,5	$x^{10}+y^4+y$	$1+x+x^2$	$\langle A_2A_3^T, B_1B_2^T\rangle$	25%	11,6
$[[150,8,8]]$	15,5	$x^5+y^2+y^3$	$y^2+x^7+x^6$	$\langle A_1A_2^T, B_1B_3^T\rangle$	26.66%	11,6
$[[144,12,12]]$	12,6	x^3+y+y^2	y^3+x+x^2	$\langle A_2A_1^T, B_1B_3^T\rangle$	33.33%	12,8
$[[196,12,8]]$	14,7	$x^6+y^5+y^6$	$1+x^4+x^{13}$	$\langle A_2A_3^T, B_1B_2^T\rangle$	35.71%	16,15

Table 4.1: Examples of BB qLDPC codes found through a computer search. Code distances were computed using the QDistRnd GAP package [167], with 1000 information sets and `mindist = 0` to obtain the actual distance. The 'Embedding' column reports the specific embedding into $\mathbb{Z}^2$ used for that code (see Section 2.3.2). The 'Mask percent' column denotes the percentage of generators that are "large", i.e., induced by the long boundary and masked during a portion of the error correction rounds. The 'Routing' steps column indicates the number of routing rounds required to route, purify, and measure the short-range and long-range generators, respectively. Algorithm 3 was not used to determine the circuits; instead, the repeated generator structure of the BB codes allowed us to find circuits by hand. The actual circuit depth is $11\times$ greater due to the Bell pair generation (depth 6), purification (depth 2), and implementation of the long-range CNOT gate (depth 3).

4.5.1 Syndrome extraction circuits

To implement a CNOT gate between a data qubit and a distant check qubit, we use the constant-depth circuit shown in Fig. 4.1(a). A number of ancilla qubits equal to the length of the CNOT gate are needed, and so qubits from the upper layer are utilized, as illustrated in Fig. 4.1(b). Multiple long-range CNOT gates may be performed in parallel as long as the paths act on disjoint sets of qubits. Given a set of CNOT gates to perform, an order that attempts to minimize the total depth of the circuits can be found using the greedy routing algorithm introduced in Section 4.4.1. Alternatively, we can utilize the repeated check structure of the BB codes to manually come up with highly parallelized orderings; Fig. 4.4 shows an example of how the Bell pairs needed for the long-range CNOT gates (red highlighted paths) can be implemented in parallel. The 'Routing steps' column in Table 4.1 indicates the number of routing rounds required to route, purify, and measure the short-range and long-range generators, respectively, of several BB code instances when using these hand-designed orderings. This repeated parity check structure is also useful for implementing generalized bicycle codes with reconfigurable atom arrays [135] and bosonic cat qubits [143].

Remote CNOT gates implemented in this way have an error rate proportional to the length of the chain. For short distances, the resulting error rate is not much worse than the native two-qubit CNOT error rate; however, larger chains will be prohibitively noisy. To remedy this, we can apply entanglement purification [134, 168] to the noisy Bell pairs in the ancilla layer. Figure 4.5 outlines the original purification scheme as proposed by Bennett *et al.* [134]. The protocol uses additional 'donor' Bell pairs (pink highlighted paths) to create 'source' Bell pairs (red highlighted paths) with higher fidelity. This is done by performing CNOT gates between the

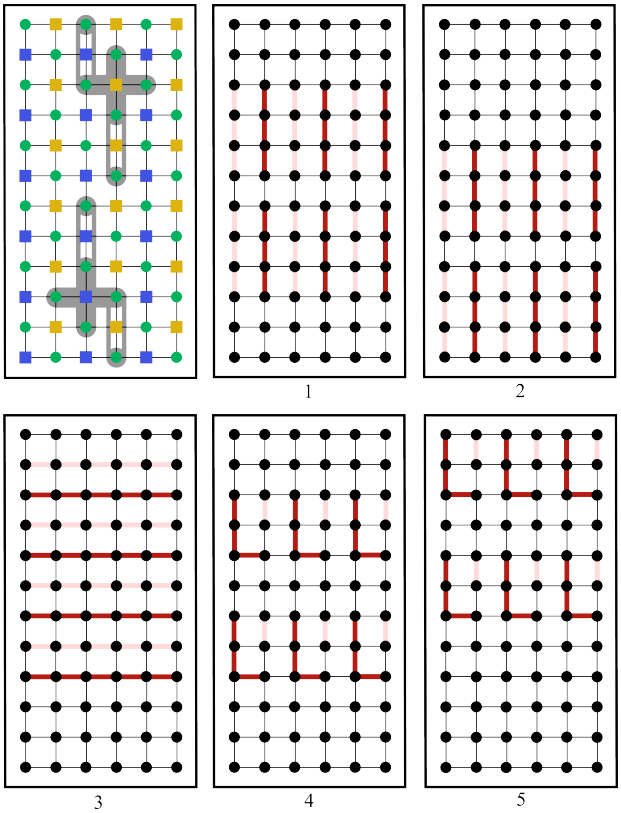

Figure 4.4: Example five-step schedule to route and purify the Bell pairs needed to measure the short-range, Z-type generators of a $[[36, 4, 4]]$ BB code constructed with $\ell = 6, m = 3$ and by matrices $A = x + y^3 + y^2, B = y^3 + x^5 + x^4$. The first panel shows the structure of the Z-type checks (yellow squares) and X-type checks (blue squares), as outlined in gray.

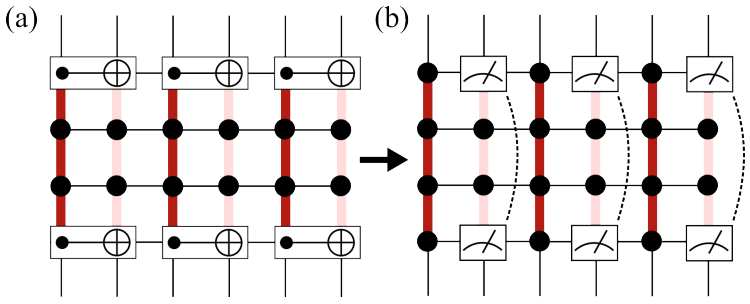

Figure 4.5: Implementing multiple long-range Bell pair purifications in parallel for a BB code. The 'source' red highlighted Bell pairs are purified using the Bennett protocol [134]. (a) CNOT gates are performed between each end of the source and the pink 'donor' Bell pairs. (b) Each end of the donor Bell pair is measured and the results compared classically. If the measurements agree, the source Bell pair is kept and used; otherwise it is discarded.

ends of the source and donor pairs, measuring the ends of the donor pairs in the computational basis, and then comparing the measurement results classically. If the results agree, the source Bell pair is kept; otherwise it is discarded. Averaging over cases where the source Bell pair is kept, it has a higher fidelity than an unpurified pair; however, in the cases where it is discarded, the corresponding long-range CNOT gate cannot be performed. We either have the option of reattempting the purification process, implementing the long-range CNOT with the flawed Bell pair, or giving up on the long-range CNOT gate (and ultimately the corresponding generator syndrome measurement) altogether. Since we already intend not to measure every generator at every error correction round, this last option is most appropriate. In the context of the bilayer architecture, both donor and source Bell pairs are routed through the ancilla layer. Practically, this means that fewer long-range CNOT gates can be implemented in parallel, since the purification process uses additional teleportation paths.

Although we now have a way to implement long-range CNOT gates, measuring every stabilizer generator in this manner incurs additional overhead. Instead, we can reduce the time overhead by applying the stacked model and partial syndrome measurement and choosing to measure the costly, large generators less frequently than the smaller ones. The frequency at which the long-range generators are measured can be tuned, with more frequent measurements potentially correcting more errors but increasing the time needed to implement error correction.

4.6 Numerical simulations

We now present the results of circuit-level error correction simulations using the class of BB quantum LDPC codes and the architecture defined in Section 4.1. Previous simulations of BB

codes showed that they greatly outperformed surface codes in terms of overhead under specific architecture assumptions [85, 135]. Here, we show that BB codes implemented with 2D local gates in the proposed bilayer architecture have comparable performance to surface codes which encode the same number of logical qubits and have roughly the same number of physical qubits.

For the following simulations, we use Stim [106] to construct the circuits and build the space-time bipartite graph used for decoding. As such, we consider a circuit-level noise model in which errors occur independently on different circuit operations. For a physical error rate p—in this chapter we consider $p = 0.1\%$—single-qubit gates have probability $p/10$ of experiencing the single-qubit depolarizing channel; two-qubit gates have probability p of experiencing the two-qubit depolarizing channel; measurement results have probability p of being flipped; qubit reset operations have probability $p/10$ of preparing the $|1\rangle$ state instead of the $|0\rangle$ state; and idle qubits experience a depolarizing channel with probability $p/50$. The assumed single-qubit, two-qubit, and measurement error rates are comparable to the performance of current ion-trap [3–5] and superconducting [6] quantum computers. However, this last condition on the idle qubit error rate is somewhat optimistic and is around an order of magnitude better than the idle error seen on production devices. We comment on this assumption in Section 4.7.

For ease of implementation, we first separately perform circuit-level simulations of the entanglement purification protocol. The simulation consists of implementing two noisy long-range Bell pairs using a circuit similar to that depicted in Fig. 4.1(a) and then performing Bennett *et al.*'s entanglement purification protocol on the two pairs. In this simplest version of the protocol, failures are not reattempted, and only a single donor Bell pair is used. Simulating the protocol many times allows us to estimate the probability that the purification protocol succeeds and, if so, the fidelity of the purified Bell pair. Fig. 4.6(a) displays the results of these simulations for

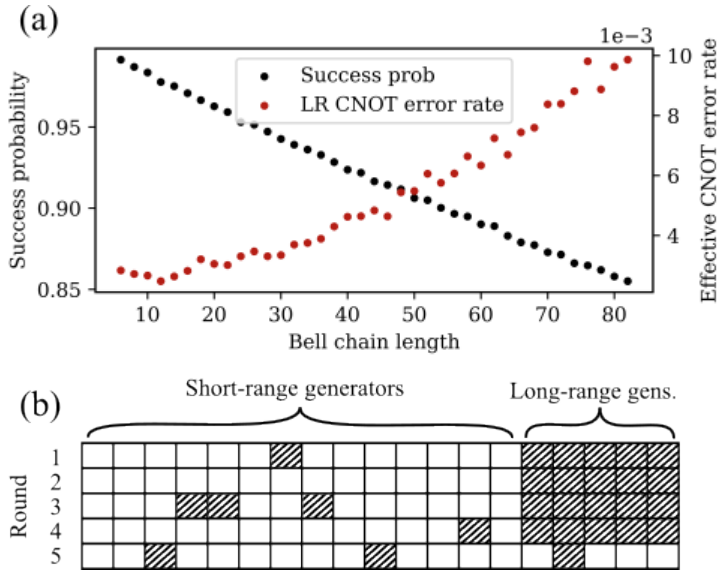

Figure 4.6: (a) Results of circuit-level simulations of the entanglement purification protocol of Ref. [134] for Bell pairs of increasing length. Two long-range Bell pairs are created using a noisy circuit similar to that of Fig. 4.1(a) and then purified with the noisy circuit depicted in Fig. 4.5. The success probability of the purification and the resulting Bell purity if successful is shown for 100 000 samples. (b) Example depiction of generator masking (indicated by a hatched fill) over several error correction rounds being affected by the entanglement purification protocol failing. In this example, the long-range generators are unmasked after five rounds.

long-range Bell pairs of different lengths under the circuit-level error model described above.

During syndrome extraction, if the entanglement purification protocol for any of the long-range CNOT gates fails, we mask the corresponding generator instead of reattempting the purifications. We can then estimate the probability that the syndrome of a generator is available, that is, all the required purifications for that generator succeed. If the purifications do succeed, then we can also estimate the error rate of the resulting long-range CNOT gate from the fidelity of the Bell pair. In the full circuit, we then implement a direct CNOT with this error rate to represent the entire procedure. Fig. 4.6(b) illustrates what this means practically: assuming the long-range generators are unmasked every five rounds, the first four rounds have these long-range generators masked (hatched fill). Additionally, due to failures of the entanglement purification protocol, some short-range generators are also masked, even though we had planned for them to always be

available. We note that these random failures are not expected to greatly impact the performance of the code, as it is unlikely that one generator will fail several rounds in a row. Thus, even if there are missed errors, they will likely be corrected when the generator does succeed in routing. In the fifth round, the long-range generators are unmasked and attempted to be measured, but only if purifications succeed can we actually obtain their syndromes. Note that with this simple purification scheme, the long-range generators are less likely to succeed, since the necessary Bell pairs are between more distant qubits and more prone to failure.

For the full error correction protocol, we begin each circuit with a single noiseless round to initialize the logical subspace. We then perform t noisy error correction rounds using the syndrome extraction circuits defined in Section 4.5.1. As the short-range generators are easier to measure, we attempt to measure them every round, whereas the costly, long-range generators are unmasked and attempted every five rounds. As described above, we additionally mask both the short- and long-range generators with probability equal to that of at least one of required purifications failing. In the cases where all purifications for a single generator succeed, we apply the two-qubit depolarizing channel after each CNOT gate with an error rate equal to that of a long-range CNOT gate performed using a Bell pair of the appropriate distance. Idling error rates are estimated using the number of steps needed to route and purify the source and donor Bell pairs for a given set of generators (see Fig. 4.4 and Table 4.1). As each step consists of Bell pair generation (depth 6), purification (depth 2), and implementation of the long-range CNOT gate (depth 3), the actual circuit depth is $11\times$ greater. To represent idling errors, a depolarizing channel is applied at the beginning of each error correction round to every qubit with probability equal to the total circuit depth times the idle error rate. Additionally, a depolarizing channel is applied to every qubit with probability $p = 0.1\%$ at the beginning of each round. Before measuring the logical

$[[n, k, d]]$	Qubits	ϵ_L
$[[128, 8, 4]]$	248	$1.4 \times 10^{-3} \pm 1.2 \times 10^{-6}$
$[[\mathbf{72}, \mathbf{8}, \mathbf{6}]]$	288	$1.6 \times 10^{-3} \pm 3.0 \times 10^{-5}$
$[[\mathbf{90}, \mathbf{8}, \mathbf{6}]]$	360	$8.9 \times 10^{-4} \pm 2.0 \times 10^{-5}$
$[[200, 8, 5]]$	392	$2.0 \times 10^{-4} \pm 6.5 \times 10^{-7}$
$[[\mathbf{120}, \mathbf{8}, \mathbf{8}]]$	480	$1.2 \times 10^{-4} \pm 2.0 \times 10^{-6}$
$[[288, 8, 6]]$	568	$9.5 \times 10^{-5} \pm 2.5 \times 10^{-7}$
$[[\mathbf{150}, \mathbf{8}, \mathbf{8}]]$	600	$5.3 \times 10^{-5} \pm 1.3 \times 10^{-6}$
$[[392, 8, 7]]$	776	$2.0 \times 10^{-5} \pm 1.5 \times 10^{-7}$
$[[\mathbf{144}, \mathbf{12}, \mathbf{12}]]$	576	$1.6 \times 10^{-4} \pm 4.6 \times 10^{-6}$
$[[300, 12, 5]]$	588	$3.0 \times 10^{-4} \pm 9.8 \times 10^{-7}$
$[[\mathbf{196}, \mathbf{12}, \mathbf{8}]]$	784	$7.9 \times 10^{-5} \pm 2.3 \times 10^{-6}$
$[[432, 12, 6]]$	852	$1.4 \times 10^{-4} \pm 3.7 \times 10^{-7}$
$[588, 12, 7]]$	1164	$2.9 \times 10^{-5} \pm 2.3 \times 10^{-7}$

Table 4.2: Code parameters, total number of qubits used, and ϵ_L as extracted from Eq. (4.6) for the simulations described in Section 4.6. Code parameters shown in bold correspond to BB code instances. Code parameters not in bold correspond to copies of the rotated surface code.

observables, we noiselessly extract the full syndrome one last time. The corresponding space-time bipartite graph is then generated, and the errors are sampled and decoded.

In this chapter, we use a decoder based on belief propagation and ordered-statistics decoding (BP-OSD) [92, 93, 169], which consists of the min-sum BP decoder followed by an order-10 combination-sweep OSD postprocessing step. Performing real time decoding using BP and higher-order OSD postprocessing may be infeasible within the fast cycle time of superconducting quantum computers; however, it was shown that good decoding performance for BB codes can be achieved while using less computationally expensive OSD parameters [170].

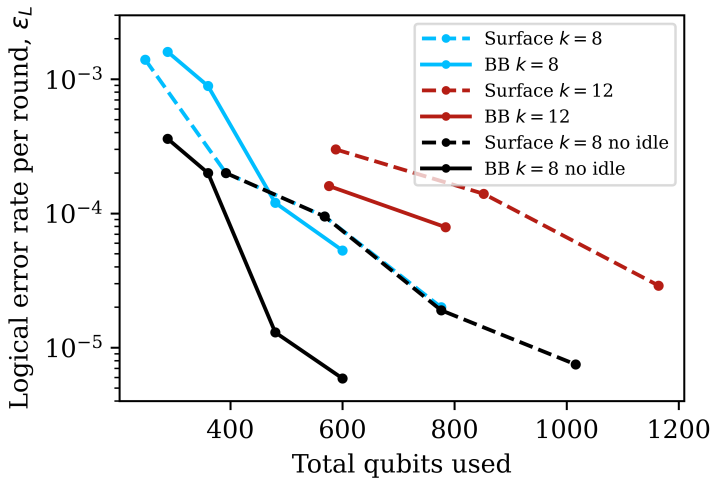

Figure 4.7: Extracted logical error rate per round, ϵ_L as a function of the total number of qubits used (data qubits plus all ancilla qubits) for several BB and surface code instances. The data is tabulated in Table 4.2. We also include simulation results in the no idle error regime, as indicated by the black lines; these results are tabulated in Table 4.3.

Table 4.2 and Fig. 4.8(a)–(b) shows the results of these simulations for several BB codes listed in Table 4.1. These codes were found by computer search and displayed, through simulations similar to those of Ref. [117], good numerical performance. For each code, every valid embedding was simulated in a simplified version of the protocol described above in order to find the embedding that yielded the best masked error correction performance. Choosing an embedding determined the percentage of generators induced by the long boundary. This percentage is listed in Table 4.1 in the Mask percent column. To our knowledge, the codes presented here are new, with the exception of the $[[144, 12, 12]]$ code, which was reported in Ref. [85]. As a comparison, we perform the same simulations with the rotated surface code which has parameters $[[d^2, 1, d]]$. To decode, we follow the same process as described in Section 2.3.5.3 but instead use the minimum-weight perfect matching decoder [132]. As the BB codes encode multiple logical qubits in a single block, multiple copies of the surface code must be used to achieve the same number of logical qubits. If $p_{SC,1}$ is the logical error rate of simulating a single rotated surface

code for t error correction rounds, then k copies of the surface code have a logical error rate

$$p_{SC,k} = 1 - (1 - p_{SC,1})^k. \tag{4.5}$$

In addition to the logical error rate, another important performance metric is the number of qubits used to achieve it. For the BB codes and the bilayer architecture, this includes the ancillary check qubits as well as the entire routing layer, which for an $[[n, k, d]]$ code uses $4n$ qubits in total. The rotated surface code uses $d^2 - 1$ additional check qubits, which brings the total number of qubits to $2d^2 - 1$ for each copy. The total number of qubits used is listed together with the code parameters in Fig. 4.8. The error bars on the data points are calculated using the standard error when sampling from a binomial distribution $\sqrt{p_{\log}(1 - p_{\log})/N}$, where N is the number of collected samples. Due to the large number of shots taken, $N \sim 10^5$, error bars in Fig. 4.8 and Fig. 4.10 are nearly invisible. Additionally, we plot a fit of

$$p_{\log} = 1 - (1 - \epsilon_L)^t \tag{4.6}$$

for both the surface and BB codes, from which we can extract the logical error rate per round, ϵ_L.

The smallest BB codes encoding $k = 8$ logical qubits are outperformed by surface codes which use fewer physical qubits. However, increasing the block length yields BB codes that surpass the performance of similarly sized surface codes. This is illustrated in Fig. 4.7, where we see the BB codes achieving a lower logical error rate per round than the surface code while utilizing fewer qubits. Increasing the number of logical qubits to $k = 12$, BB codes and the proposed architecture immediately outperform the surface codes in terms of logical error rate and

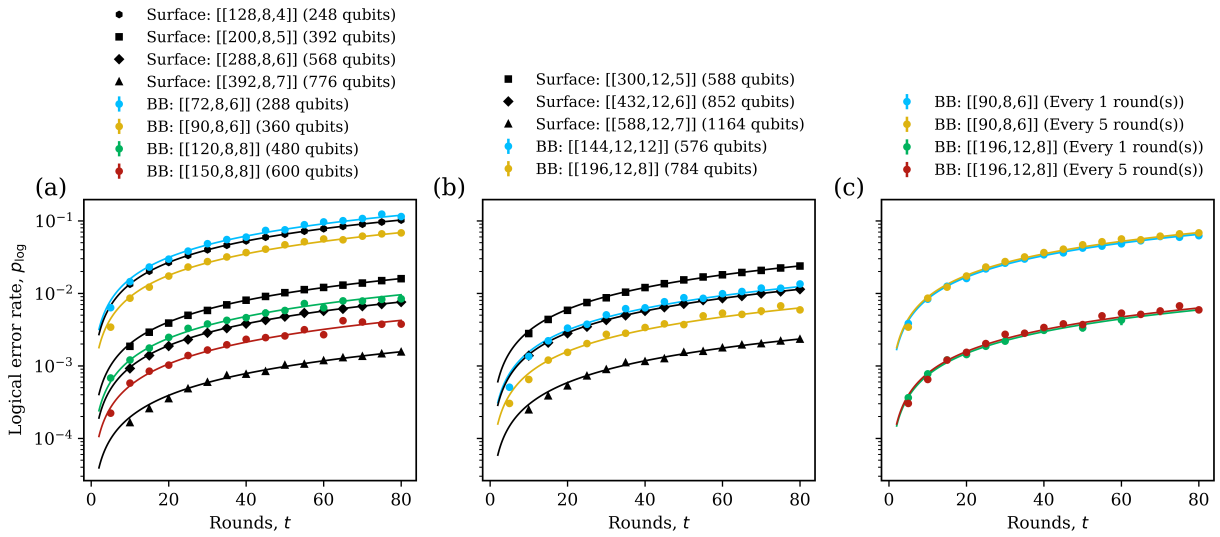

Figure 4.8: (a)–(b) Logical error rate of performing t rounds of error correction with BB codes with (a) $k = 8$ and (b) $k = 12$ on the proposed bilayer architecture. The logical error rate of the same simulations using k copies of rotated surface code is plotted as a comparison. (c) Comparison between time intervals at which the long-range generators are measured. For the BB codes in panels (a)–(b), the long-range generators were measured every five error correction rounds. A fit of Eq. (4.6) is also shown, from which we extract the logical error rate per round, ϵ_L.

space overhead. Compared to twelve patches of a $[[36, 1, 6]]$ rotated surface code using a total of 852 physical qubits and a logical error rate per round of $\epsilon_L = 1.43 \times 10^{-4}$, we find a $[[144, 12, 12]]$ BB code using 576 qubits that matches the performance, with $\epsilon_L = 1.56 \times 10^{-4}$. Additionally, we find a $[[196, 12, 8]]$ code using 784 qubits that outperforms it with $\epsilon_L = 7.89 \times 10^{-5}$. At this scale, the improvements are not so drastic, but we expect to see greater overhead benefits as the block length and number of logical qubits increase.

We now vary the interval at which the long-range generators are measured, the results of which are also shown in Fig. 4.8(c). For the $[[90, 8, 6]]$ code there are 44 (not necessarily independent) generators of a single type. Using a routing schedule that was found by hand, all 35 short-range generators of a single type can be routed, purified, and measured in 9 steps; whereas it takes 5 steps to route, purify, and measure the remaining 10 long-range generators of the same

type. Measuring the 35 short-range and 10 long-range generators of the opposite type requires an additional 9 and 5 steps, respectively. This means that measuring the long-range generators every five error correction rounds requires a circuit depth that is 28.5% shorter than if the long-range generators were measured every round ($4 \cdot 2 \cdot 9 + 2 \cdot (5 + 9) = 100$ steps versus $5 \cdot 2 \cdot (5 + 9) = 140$ steps), at a negligible increase in the logical error rate per round from $\epsilon_L = 8.41 \times 10^{-4}$ to 8.92×10^{-4}. Increasing the size of the code to $[[196, 12, 8]]$, we again see negligible differences in logical error per round performance between the two measurement schedules, from $\epsilon_L = 7.41 \times 10^{-5}$ to 7.89×10^{-5}, with the additional benefit of a 32.0% decrease in the depth of the syndrome extraction circuit when measured every five rounds. The two codes consist of 22.22% and 35.71% long-range generators, respectively, yet both remain consistent between long-range measurement intervals. As the block length increases, so does the discrepancy between the measurement times of the short- and long-range generators, increasing the circuit depth savings. Additionally, this discrepancy would disproportionally introduce more errors during the long-range measurement rounds, potentially making it more performant to measure these large generators even less frequently. However, this behavior is very dependent on the idle error rate, and changing the idle error rate may cause the two curves to deviate.

Achieving more significant reductions in circuit depth requires measuring the long-range generators much less frequently, as shown in Fig. 4.9. We display the potential circuit depth savings for two BB code instances as a function of how many error correction rounds elapse between measurements of the long-range generators. The horizontal red lines indicate the maximum potential savings, corresponding to a schedule where the long-range generators are only measured once at the end of the circuit. For example, the $[[144, 12, 12]]$ BB code requires 16 (15) steps to measure the short- (long)-range generators of a single type, which gives a maximum depth

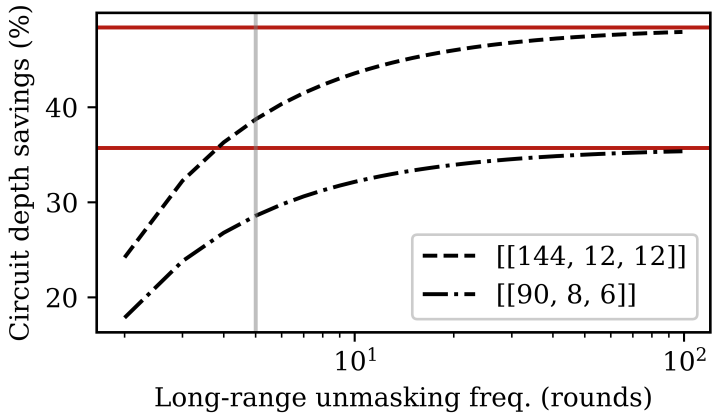

Figure 4.9: Percentage change in circuit depth, compared to a circuit that always measures every generator, as a function of number of rounds elapsed between long-range generator measurements. Horizontal solid red lines indicate the potential maximum reduction in circuit depth for the two BB code instances. The gray vertical line highlights the depth savings achieved by measuring the long-range generators every five error correction rounds, as done in Fig. 4.8(a)–(b) and Fig. 4.10.

savings of 48.4%. When measuring every five rounds, as in Fig. 4.8(a)–(b) and Fig. 4.10, we see a circuit depth savings of 38.7%, indicated by the vertical gray line. Measuring the long-range generators very infrequently will significantly degrade the error correction performance and may not be worth the reduced circuit depth. Instead, it may be more advantageous to measure the long-range generators relatively frequently, e.g., every 2–5 rounds; in that regime we still see considerable circuit depth savings (50% to 80% of the theoretical maximum), but the impact on the logical error rate is negligible.

Even with the reduced idle error rate that we consider here, idle errors are a significant error source, especially on rounds where the long-range generators are measured. In Table 4.3 and Fig. 4.10, we perform the same simulations as described above but do not apply idling errors. Due to their short syndrome extraction circuit depths, the surface codes are unaffected from the decrease in idle error. However, we now find that all BB code instances achieve better logical error rates than surface codes while using fewer physical qubits, as shown in Fig. 4.7. Indeed, the

$[[n, k, d]]$	Qubits	ϵ_L
$[[\mathbf{72}, \mathbf{8}, \mathbf{6}]]$	288	$3.6 \times 10^{-4} \pm 4.9 \times 10^{-6}$
$[[\mathbf{90}, \mathbf{8}, \mathbf{6}]]$	360	$2.0 \times 10^{-4} \pm 3.3 \times 10^{-6}$
$[[200, 8, 5]]$	392	$2.0 \times 10^{-4} \pm 6.5 \times 10^{-7}$
$[[\mathbf{120}, \mathbf{8}, \mathbf{8}]]$	480	$1.3 \times 10^{-5} \pm 8.2 \times 10^{-7}$
$[[288, 8, 6]]$	568	$9.5 \times 10^{-5} \pm 2.5 \times 10^{-7}$
$[[\mathbf{150}, \mathbf{8}, \mathbf{8}]]$	600	$5.9 \times 10^{-6} \pm 2.4 \times 10^{-7}$
$[[392, 8, 7]]$	776	$2.0 \times 10^{-5} \pm 1.5 \times 10^{-7}$
$[[512, 8, 8]]$	1016	$7.5 \times 10^{-6} \pm 4.3 \times 10^{-8}$

Table 4.3: Code parameters, total number of qubits used, and ϵ_L as extracted from Eq. (4.6) for the simulations described in Section 4.6 albeit with no idle error. Code parameters shown in bold correspond to BB code instances. Code parameters not in bold correspond to copies of the rotated surface code.

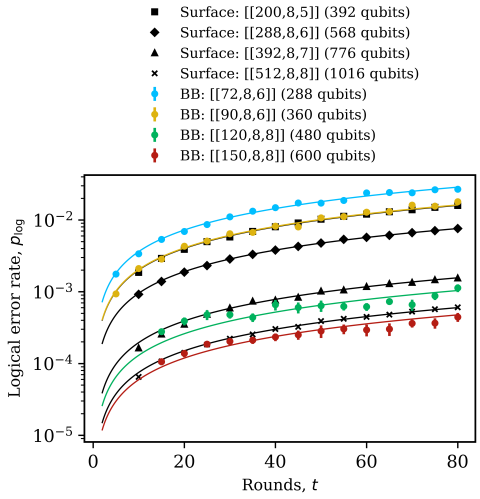

Figure 4.10: Logical error rate of performing t rounds of error correction with BB codes with $k = 8$ logical qubits on the proposed bilayer architecture. For this plot, we consider the case where the idle error rate is zero. The logical error rate of k copies of the rotated surface code, calculated using Eq. (4.5), as well as the total number of physical qubits used, is again plotted as a comparison. A fit of Eq. (4.6) is also shown, from which we extract the logical error rate per round, ϵ_L.

$[[150, 8, 8]]$ code using 600 physical qubits sees an $8.8\times$ improvement in the logical error rate per round, from $\epsilon_L = 5.31 \times 10^{-5}$ to 5.9×10^{-6}, and now outperforms eight patches of a $[[64, 1, 8]]$ rotated surface code using 1016 physical qubits with $\epsilon_L = 7.5 \times 10^{-6}$. Achieving negligible idle error rates may not be feasible, but it illustrates the regime where our protocol performs best.

4.7 Discussion

In this chapter, we have presented a bilayer architecture for implementing nonlocal qLDPC codes on quantum devices which are restricted to 2D local gates. We have shown that bivariate bicycle codes are well suited for such an architecture and described a parallelizable syndrome measurement scheme which makes use of the geometric parity check structure of the codes. Through circuit-level simulations of a multi-round decoding protocol, we found that BB codes attain comparable logical error rates to that of the rotated surface code while using fewer physical qubits. Furthermore, by applying the stacked model and masking, we achieved a decrease in the syndrome extraction time with negligible impact on the error correction performance. In that regard, we have shown an instance where partial quantum error correction and partial syndrome measurement has a significant effect on the overhead of implementing error correction.

However, there are a number of challenges that must be considered in a physical implementation of this protocol. Perhaps the most notable issue is the depth of the circuit required to perform even a single syndrome extraction. Implementing a single long-range CNOT gate requires constructing the long-range Bell pair, purifying it, and using it to implement a CNOT between a data qubit and check qubit. Although several CNOT gates can be implemented in parallel, doing this for the entire set of generators requires 10s of routing steps, translating to a

physical circuit with depth in the 100s. One consequence of the depth of the circuit is that our protocol only performs well in the regime of low idle error rate. Furthermore, per Claim 11, as the block length increases so too does the required routing time and, consequently, the physical circuit depth. This is in stark contrast to the implementation in Ref. [85], where the entire set of generators can be measured with a circuit of depth seven, albeit with the use of long-range connections. These long-range connections are a significant engineering challenge, and it is unclear whether implementing high-fidelity gates in this way is feasible.

We do find BB codes where the same parity check structure is shared between codes of increasing block length. For code families with this property, the number of steps in the syndrome extraction circuit is constant, so the noise per syndrome extraction cycle coming from idle error does not increase. However, this also means that the percentage of long-range generators and, by extension, the amount of nonlocality in the code, decreases. References [113, 114] showed that it is impossible to beat the asymptotic scaling of the surface code parameters without increasing the amount of nonlocality. Increasing the block length will yield larger k and d, but the asymptotic scaling of these codes will approach that of the surface code; however, for finite sizes we would still expect to see significant space overhead savings compared to alternative, lower-rate codes. Even for BB codes with increasing generator shape, it is feasible that the increased error correction capabilities will outpace the increase in idle error. In particular, Fig. 4.2 shows a power-law relationship between block length and routing depth. Assuming the code is operating below threshold, the exponential suppression in logical error should be sufficient to handle the increased effective idle error.

Another challenge is that the simple purification protocol presented here does not scale well, as increasing the block length would lead to low-fidelity Bell pairs and a high purifica-

tion failure rate. Although there are many entanglement purification protocols that improve the resulting Bell fidelity [171–174], using them would further increase the depth of the syndrome extraction circuits or require additional ancillary qubits. The one potential saving factor is that the vast majority of the work is done by the upper routing layer to construct and purify the Bell pairs, and the two layers interact in fewer than $1/10$ of the circuit steps. If it were possible to sufficiently isolate the data layer, akin to what is done in ion traps or reconfigurable atom arrays, it might be possible to achieve the low idling error rates that would greatly improve the performance of the protocol.

If the aforementioned issues can be solved, then scaling up should increase the advantage of qLDPC codes over the surface code. One potential solution is to improve the circuit depth of the protocol. An architectural feature that could accomplish this is the ability to perform two-qubit gates on qubits that are some distance R apart [141]. This is a natural operation on neutral-atom devices, where Rydberg-Rydberg interactions, especially dipolar ones [175], can be quite long-range. Furthermore, Rydberg-Rydberg interactions can help with syndrome extraction by naturally realizing long-range generalized multi-control multi-target CNOT gates [176]. At the same time, such long-range Rydberg-based gates may harm parallelism since only one such gate can be implemented within the Rydberg blockade radius at a time. Gates beyond nearest-neighbor could also be feasible in superconducting devices through the use of medium-range couplers or photonic interconnects [177]. When R is a constant, the asymptotic behavior will remain unchanged; however, practically this would mean that the short-range generators would be much easier to implement. With an appropriate choice of R, it would then be possible to use the depth-7 circuit of Ref. [85] to measure the short-range generators, in which case the only difficulty would be to measure the long-range generators in the proposed manner. An alternative

approach would be to add additional ancilla layers to the architecture. Although this would further increase the qubit overhead, it would allow for more parallelization during the syndrome measurement, decreasing the total circuit depth.

Chapter 5: Adaptive syndrome extraction

We now present an example of partial quantum error correction not based on the stacked model. Instead, we utilize structure in the quantum error correcting code to facilitate useful partial syndrome measurement.

Recent improvements in quantum computing hardware have allowed for demonstrations of quantum error correcting codes operating below threshold [47–51]. However, performing the syndrome extraction circuits can introduce new errors through faulty gates, idling errors, and incorrect corrections. Indeed, it is sometimes the case that bare qubits can idle longer than a logical qubit of a QECC that undergoes several rounds of syndrome extraction and error correction [178]. This issue can be further exacerbated when, for example, implementing nonlocal QECCs on hardware which only have access to local gates [103, 116], as we have seen in the previous chapters.

Quantum error correction is inherently adaptive in the sense that after extracting the syndrome and decoding, the appropriate correction is applied to return the system to the codespace. Performing this process in real time is necessary when the quantum circuit contains non-Clifford operations. Making changes to the syndrome extraction circuit based on measurement results obtained during a QEC cycle is a technique that has also seen use in a number of contexts including repeat-until-success protocols [179] and detection of hook errors [42, 180]. Similarly,

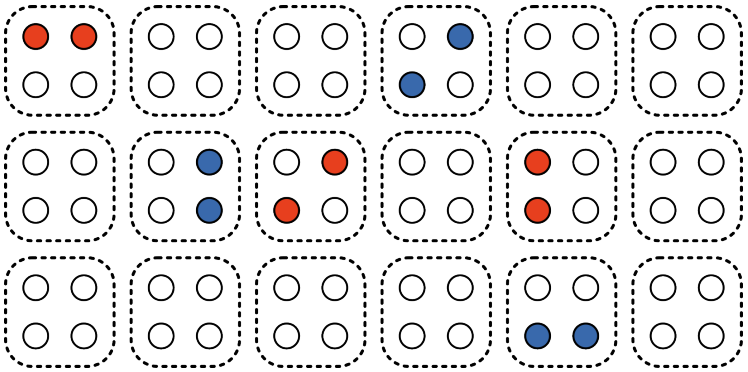

Figure 5.1: A QED+QEC concatenated code. Physical qubits are first encoded in a small error detecting code, such as the $[[4, 2, 2]]$ Iceberg code. The logical qubits of many Iceberg code blocks then act as the physical qubits of some high-rate qLDPC code.

several works perform measurement sequences based on the measurement results of previous syndrome extraction rounds in order to reduce time overhead [181–183]. And while not in real time, Ref. [184] provided a scheme for altering syndrome extraction circuits by occasionally neglecting to measure geometrically nonlocal stabilizer generators. In this work, we generalize this idea and present a framework that can 'short-circuit' syndrome extraction and skip measuring generators that are unlikely to be useful for decoding. This decision process is carried out in real time based on the results of the syndrome measurement from the current QEC cycle.

The main observation motivating the investigation of this adaptive scheme is that in most error correction rounds, much of the syndrome information is unnecessary. For example, in a system without errors, the syndrome does nothing but confirm that the code remains in the correct subspace. In this specific case, a high-distance code and an error detecting code would function identically, although the high-distance code would require longer syndrome extraction circuits and introduce more noise. In general, generators with support on qubits with no errors provide little useful information. So, how can we cheaply determine approximate error location before committing to an expensive syndrome extraction? The approach we take in this work is to

use a concatenated code where the inner code is a small error detecting code, and the outer code is some high-rate, high-distance quantum low-density parity-check code. The adaptive scheme for this code construction is as follows: in each error correction cycle, the stabilizer generators of the underlying error detecting codes are measured first. The resulting syndrome reveals which blocks contain errors; however, since we are using error detecting codes, there is not enough information to correct the error, and so we look for more syndromes by measuring the generators of the outer level of the concatenated code. By the previous argument, we should only measure those generators which provide useful syndrome information. As such, we measure only the stabilizer generators of the outer code which have support on qubits located in quantum error detection (QED) blocks that have errors in them. Generators with no overlapping support are assumed to yield a $+1$ measurement result.

Our adaptive syndrome extraction scheme can be interpreted as a quantum weight reduction protocol in the space-time picture. Hastings first introduced quantum weight reduction as a toolkit to transform QECCs as to reduce the number of qubits that each checks acts on and the number of checks that each qubit is involved in [185, 186]. Reducing the check weights and the qubit connectivity helps to limit the propagation of errors in the QECC as well as reduce the syndrome extraction circuit depth. Prior to Hastings' work, quantum weight reduction has already been studied in the form of subsystem codes [187–190]. Since then, other works have considered weight reduction in the space-time picture of error correction [55, 125, 191–193]. In our scheme, we consider a single space-time block to contain T rounds of syndrome extraction. By choosing to measure generators of the outer code adaptively, each physical qubit of our concatenated code is now involved in fewer checks "on average". Similarly, the generators of the concatenated code act on fewer qubits "on average" in a single space-time block. While the concatenation

scheme increases the actual weight of the checks, we can ensure an overall weight reduction in the space-time picture as long as the outer code generators are measured sufficiently infrequently.

We provide a concrete example of adaptive syndrome extraction facilitated by code concatenation by considering a quantum code consisting of the $[[4, 2, 2]]$ Iceberg code [67, 72, 87] concatenated with a hypergraph product (HGP) code [76]. We show that these $[[4, 2, 2]]$-concatenated HGP codes function as better quantum memories than non-adaptive and non concatenated codes, while using fewer CNOT gates and physical qubits. For these codes, we also provide fault-tolerant logical gadgets with which we can obtain universal logical computation. In particular, we adapt the grid Pauli product measurement (GPPM) scheme of Ref. [194] and the single-shot state preparation scheme of Ref. [81] to our concatenated codes, allowing us to obtain the same space-time cost of logical computation as non-concatenated HGP codes.

The rest of the chapter is structured as follows. Section 5.1 motivates adaptive syndrome extraction and details how it may provide performance boosts as well as reductions in circuit depth. In Section 5.2 we provide an explicit concatenation structure for $[[4, 2, 2]]$-concatenated HGP codes that facilitates convenient logical gates and improved numerical performance. We then show in Section 5.3 that we can achieve universal logical computation with $[[4, 2, 2]]$-concatenated HGP codes, with the details deferred to Appendix B.2. In Section 5.4 we provide circuit-level simulations comparing the performance of the adaptive scheme with non-adaptive syndrome extraction as well as non-concatenated HGP codes. We conclude in Sections 5.5-5.6 with a discussion on potential specialized applications for the adaptive scheme, the implications of this chapter, and potential follow-up research.

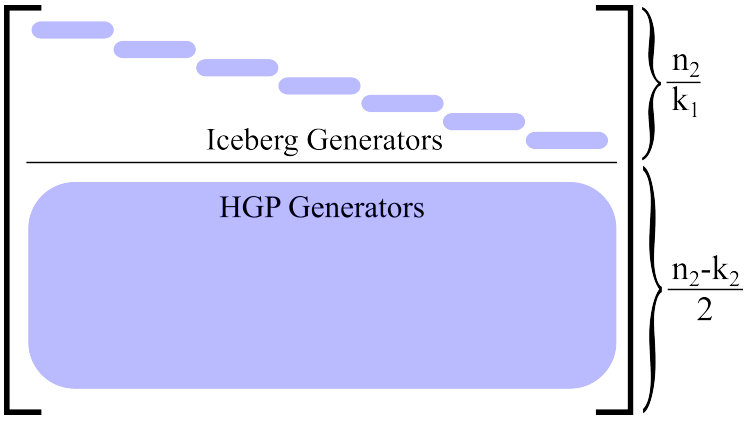

Figure 5.2: Stabilizer structure of Iceberg-concatenated HGP codes. A similar structure would arise if the HGP code was replaced with a different LDPC code.

5.1 Adaptive syndrome extraction

Using Procedure 1, we can obtain a concatenated code that is well-suited for the adaptive syndrome extraction scheme. Let $\mathcal{Q}_2$ be a $[[n_2, k_2, d_2]]$, (Δ_V, Δ_C)-qLDPC square quantum expander code, and let $\mathcal{Q}_1$ be a $[[n_1, k_1, d_1]] = [[n_1, n_1 - 2, 2]]$ Iceberg code such that $n_1 - 2 \mid n_2$. The resulting concatenated code then has parameters $[[n, k, d]] = [[n_1 n_2/(n_1 - 2), k_2, d \geq d_1 d_2/(n_1 - 2)]]$. Out of the $(n_1 n_2/(n_1 - 2)) - k_2$ generators, $2(n_2/k_1)$ come from the Iceberg code blocks, while the outer HGP code supplies its own $n_1 - k_1$ generators. For Iceberg codes, the distance lower bound provided in Procedure 1 can be improved:

Theorem 14. *Applying Procedure 1 with $\mathcal{Q}_1$ as a $[[n_1, n_1 - 2, 2]]$ Iceberg code and $\mathcal{Q}_2$ as a $[[n_2, k_2, d_2]]$ qLDPC code yields a concatenated code $\mathcal{Q}$ with parameters $[[n_1 n_2/(n_1 - 2), k_2, 2d_2 \geq d \geq d_2]]$.*

Proof. First notice that the Iceberg code logical operators, Eq. (2.42), overlap in exactly one location, i.e., on qubit 1 for X-type logicals and on qubit n for Z-type logicals. Consider a weight-w X-type Pauli operator encoded in a $[[w + 2, w, 2]]$ Iceberg code. Due to the structure

of the Iceberg logical operators, the encoded operator will have weight w if w is even; otherwise it will have weight $w+1$. The maximum distance of $2d_2$ is hence achieved when the d_2 Pauli operators coming from the HGP logical operator fall into d_2 distinct Iceberg code blocks, whereas the minimum distance of d_2 is attained if d_2 is even and all Pauli operators are in the same Iceberg block. □

By carefully choosing the assignment of qLDPC physical qubits to Iceberg logical qubits, it may be more likely to achieve the full distance of $2d_2$, see Section 5.2.1.

Briefly, the adaptive syndrome extraction scheme consists of two stages. The first stage measures the stabilizer generators of the inner Iceberg code blocks. Depending on the obtained measurement results, a potentially submaximal set of generators from the outer quantum expander code are measured. The remaining, unmeasured generators are assumed to yield a trivial syndrome. Decoding can then proceed in the normal manner. Due to the fact that the syndromes of the inner code and outer code are correlated, the total number of generators that are measured can be much less than the number of generators for the concatenated code, especially in the low error rate regime.

Let $\mathcal{S}_{IB}$ be the generators of the concatenated code coming from the Iceberg blocks, and let $\mathcal{S}_{HGP}$ be the generators of the concatenated code coming from the HGP code. We can then restrict the full syndrome to these subsets $\sigma_{\mathcal{S}_{IB}} \in \mathbb{F}_2^{2(n_2/k_1)} \subset \sigma$ or to a single generator, e.g. $\sigma^{(i)}_{\mathcal{S}_{IB}} \in \{0,1\}$. For any Pauli operator L, we denote its support $\text{supp}(L) \subset [n]$ as the set of qubits on which L acts nontrivially. For every $S \in \mathcal{S}_{IB},\ |\text{supp}(S)| = n_1$, and for every $S \in \mathcal{S}_{HGP},\ |\text{supp}(S)| \leq 2\Delta_C$.

As the simplest example, consider a single X error on the ith Iceberg block: assuming no

measurement errors, the resulting syndrome when restricted to $\mathcal{S}_{IB}$ is $\sigma_{\mathcal{S}_{IB}} = \delta_{ij}$, i.e. all zeros with a single one in the ith bit. Considering the generators in $\mathcal{S}_{HGP}$, we can say that

$$\text{supp}(\mathcal{S}_{IB}^{(i)}) \cap \text{supp}(\mathcal{S}_{HGP}^{(j)}) = \emptyset \longrightarrow \sigma_{\mathcal{S}_{HGP}}^{(j)} = 0, \tag{5.1}$$

since any generator which does not have support on the qubits of the ith Iceberg block will not have support on the error. Only those $\mathcal{S}_{HGP}^{(j)} \in \mathcal{S}_{HGP}$ such that

$$\text{supp}(\mathcal{S}_{IB}^{(i)}) \cap \text{supp}(\mathcal{S}_{HGP}^{(j)}) \neq \emptyset \tag{5.2}$$

provide additional syndrome information which helps us locate the error within the ith Iceberg block. Hence in this example, we have determined that it is not necessary to measure all $(n_1 n_2/(n_1 - 2)) - k_2$ generators; instead, it suffices to measure only the $2(n_2/k_1)$ Iceberg generators and a small set of overlapping HGP generators.

To generalize, let us first define the following function:

Definition 15. *Consider a set of stabilizer generators $\mathcal{S}$, we define $\varphi(\mathcal{S}^{(i)})$ to be the set of stabilizer generators which share support with $\mathcal{S}^{(i)}$*

$$\varphi(\mathcal{S}^{(i)}) = \{\mathcal{S}^{(j)} \mid \textit{supp}(\mathcal{S}^{(i)}) \cap \textit{supp}(\mathcal{S}^{(j)}) \neq \emptyset\} \tag{5.3}$$

When the outer HGP code is (Δ_V, Δ_C)-qLDPC, each Iceberg code block support overlaps with at most $n_1 \cdot \Delta_V$ HGP generators, i.e., $|\varphi(\mathcal{S}_{IB}^{(i)})| \leq n_1 \Delta_V$. Both n_1 and Δ_V are constants, so the number of HGP generators one needs to measure is proportional to the number of Iceberg

blocks which have detected an error. Given the syndrome of all Iceberg blocks, $\sigma_{\mathcal{S}_{IB}}$, the HGP generators which should be measured are then those which share support with Iceberg generators that have indicated an error is present in their block:

$$\{\varphi(\mathcal{S}_{IB}^{(i)}) \mid \sigma_{\mathcal{S}_{IB}}^{(i)} = 1\}. \tag{5.4}$$

This adaptive procedure makes it so that we only obtain the syndrome information necessary to diagnose the error, ultimately yielding shorter syndrome extraction circuits and less introduced noise. We note that while here we have described the outer qLDPC code as a HGP code, any other qLDPC code would function similarly.

5.2 Canonical logical basis for hypergraph product codes

As a reminder, in Section 2.3.1.1 we described a canonical representation for hypergraph product codes. The physical qubits of HGP(H, H) can be arranged into two square grids of size $n \times n$ and $m \times m$, see Fig. 5.3. The $n \times n$ grid is deemed the left sector, and the $m \times m$ grid is deemed the right sector. We denote the physical qubits on the principal diagonal of each sector, i.e. $(r, r, \bullet)$, as *diagonal* qubits. Additionally, for an $(r, c, \bullet)$ qubit, we identify the $(c, r, \bullet)$ qubit as its *mirror* qubit, with the two together considered *twin* qubits, see Fig. 2.7(c).

In this layout, we can also obtain a *canonical basis* for hypergraph product code logical operators [84]. For these codes, a canonical basis is defined to be a set of logical operators such that they have support contained in a single row or column of a single sector. Additionally, $\overline{X}_i$ and $\overline{Z}_i$ share support on exactly one qubit, whereas $\overline{X}_i$ and $\overline{Z}_j$ for $i \neq j$ do not share any support. The physical qubit on which $\overline{X}_i$ and $\overline{Z}_i$ overlap is deemed the pivot qubit for that logical qubit.

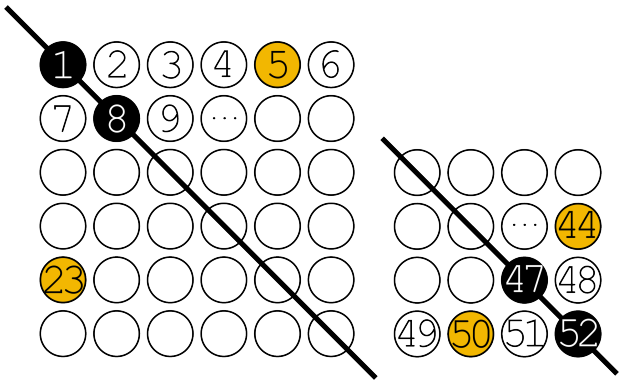

Figure 5.3: Assignment of physical HGP qubits to logical $[[4, 2, 2]]$ qubits. Physical HGP qubits are numbered starting from the top left qubit of the L sector, going row-by-row, and finishing on the bottom right qubit of the R sector. Consecutive diagonal qubits, e.g. qubits $\{1, 8\}$ and $\{47, 52\}$ are assigned to the same Iceberg code block. Twin qubits, e.g. qubits $\{5, 23\}$ and $\{44, 50\}$ are assigned to the same Iceberg block.

Similarly to the physical qubits, the logical qubits are either diagonal logical qubits or part of a twin logical qubit pair depending on the location of its pivot qubit. We refer to Ref. [84] where they provide a method for constructing a canonical basis for square hypergraph product codes.

5.2.1 Assignment of [[4,2,2]] logical qubits

We now describe a method for assigning the $[[4, 2, 2]]$ Iceberg code logical qubits to physical qubits of a hypergraph product code which takes advantage of the aforementioned geometric structure of the stabilizer generators and logical operators. Additionally, we will later show that this assignment allows the concatenated code to inherit logical gates from the base HGP code.

The left and right sectors are labeled by the $n^2 + m^2$ physical qubits as shown in Fig. 5.3, with the first n^2 qubits labeling the left sector, and the remaining $m^2 = (n - k)^2$ qubits labeling the right sector. We then assign the physical qubits to the logical qubits of the $(n^2 + m^2)/2$ $[[4, 2, 2]]$ Iceberg codes as follows: consecutive diagonal qubits, e.g. qubits $\{1, 8\}$ and $\{47, 52\}$ of Fig. 5.3, get mapped to the same Iceberg block. We also map twin qubits, e.g. $\{5, 23\}$ and

$\{44, 50\}$, to the same Iceberg code block.

As discussed in the proof of Theorem 14, we obtain suboptimal distances whenever qubits supported on the same HGP logical operator get mapped into the same Iceberg code block. While it is possible for there to exist a low weight logical operator span several rows or columns and have support on both qubits of a single Iceberg code block, this mapping reduces the chance of that happening. Indeed, we verified the resulting distances of concatenated codes constructed with this mapping using QDistRnd [167] and found that all instances achieved a distance of $2d_2$. Additionally, using a square HGP code and this mapping ensures symmetric performance between the X and Z basis, where even if $d < 2d_2$, $d_X = d_Z = d$.

5.3 Logical Computation for [[4,2,2]]-concatenated hypergraph product codes

In Ref. [194] Xu *et al.* developed grid Pauli product measurements (GPPMs) as a framework to perform fast and parallelizable logical computation for homological product codes. Here, we show that this framework can be adapted for our $[[4, 2, 2]]$-concatenated HGP codes. By combining the adapted GPPMs with a set of new logical gadgets that we obtain from the transversal gate sets and fold-transversal gate sets of the $[[4, 2, 2]]$ Iceberg code and HGP codes, respectively, we are able to show Theorem 16 and prove that we can efficiently simulate the full Clifford group on an arbitrary $[[4, 2, 2]]$-concatenated HGP code block. In particular, we demonstrate how we can utilize inter- and intra-block CNOT gates, concatenated H-SWAPs, CZ-Ss, and GPPMs to fault-tolerantly simulate a layer of Clifford gates that include H, S, and intra-block CNOT gates acting on $O(k)$ logical qubits of our $[[4, 2, 2]]$-concatenated HGP code block with low space and time overhead. We defer the definitions of the logical gadgets and the proof of Theorem 16 to

Appendix B.2.

Theorem 16 (Clifford Gates for Concatenated HGP Code)**.** *A single layer of an ideal Clifford circuit on k logical qubits with $\Theta(k)$ gates consisting of Hadamard, S, and CNOT gates can be simulated on a square HGP code concatenated with the $[[4,2,2]]$ Iceberg code blocks with either one of the following space-time costs:*

1. *$O(k)$ space and $O\left(k^{3/2}\right)$ time*
2. *$O\left(k^{3/2}\right)$ space and $O(k)$ time.*

If the square HGP code was constructed from one-generator systematic circulant (OGSC) quasi-cyclic base codes, the concatenated code can simulate the layer with either one of the following space-time costs:

1. *$O(k)$ space and $O\left(k^{5/4}\right)$ time*
2. *$O\left(k^{3/2}\right)$ space and $O\left(k^{3/4}\right)$ time.*

Our technical contributions include the following: adapting the GPPM scheme of Ref. [194], developing fault-tolerant logical gadgets, and proving the amenability of a single-shot state preparation scheme for our $[[4,2,2]]$-concatenated HGP codes. The main challenge in adapting the GPPM scheme lies in resolving the tension between the punctures introduced in the GPPM scheme and the assignment of Iceberg code logical qubits as part of our code concatenation scheme.

The fault-tolerant logical gadgets have to be formulated with the constraints of both the HGP and Iceberg codes in mind. In particular, we adapt the logical translation gadget introduced

in Ref. [194] for square HGP codes constructed from quasi-cyclic one-generator systematic circulant (OGSC) codes. We reduce the problem to the permutation of arbitrary single logical qubits between Iceberg code blocks and use measurement-based logical SWAP gates to construct the logical translation gadget for our $[[4, 2, 2]]$-concatenated HGP code. We provide explicit constructions for the other fault-tolerant logical gadgets for the case where we do not have the logical translation gadget. We direct readers to Ref. [194] for the case where the logical translation gadget is available, i.e., the square HGP codes are constructed from quasi-cyclic OGSC codes.

By bounding the soundness and confinement factors for the $[[4, 2, 2]]$-concatenated HGP code, we prove that our concatenated HGP code is still amenable to the single-shot state preparation scheme for HGP codes proposed by Hong in Ref. [81], giving us the ability to implement all of our logical gadgets in $O(1)$ logical cycles at the expense of additional space. This trade-off comes from choosing between a single-shot state preparation scheme and a regular CSS code state preparation scheme and gives rise to the two possible space-time costs for each of the two different logical gadget constructions stated in Theorem 16.

Because we can reproduce the logical gadgets in Ref. [194] for our $[[4, 2, 2]]$-concatenated HGP code, we can use the GPPM scheme to perform the "8-to-CCZ" magic state distillation protocol and implement the non-Clifford gates in parallel. Combining the magic state distillation and consumption protocol with the Clifford gate simulation, we obtain a fault-tolerant protocol for universal logical computation.

5.4 Numerical simulations

To quantify the effectiveness of the adaptive QED + QEC scheme, we perform memory experiment simulations.

5.4.1 [[4,2,2]]-concatenated decoding

Several other codes utilize concatenation where the lowest level code consists of the $[[4, 2, 2]]$ code: the C_4/C_6 code [195], the C_4/Steane code [196], many-hypercube code [88], and the $[[4, 2, 2]]$-Toric code [197]. However, in almost all of these works, error correction is done using Knill-style syndrome extraction [198]. This method is not compatible with an adaptive syndrome extraction protocol, as there is no opportunity to 'short-circuit' the syndrome extraction. We instead stick to Shor-style syndrome extraction [199], and as a consequence we are not able to use hard- and soft-decision decoding [195, 200], symbol-MAP decoding, or level-by-level minimum distance decoding [88].

In this work, we decode the $[[4, 2, 2]]$-concatenated HGP code by first attempting to address errors in the inner $[[4, 2, 2]]$ code blocks before decoding the outer HGP code. Given that the inner code is only error-detecting, we are unable to applied tailored corrections; instead, we apply the same correction whenever an Iceberg code block detects an error. As a reminder, the generators and logical operators for the $[[4, 2, 2]]$ code are shown below.

$$S_X = X_1X_2X_3X_4 \quad S_Z = Z_1Z_2Z_3Z_4 \tag{5.5}$$

$$\overline{X}_1 = X_1X_2 \quad \overline{Z}_1 = Z_2Z_4 \tag{5.6}$$

$$\overline{X}_2 = X_1X_3 \quad \overline{Z}_2 = Z_3Z_4 \tag{5.7}$$

We now go through the process for correcting Z-type errors. When decoding the inner $[[4, 2, 2]]$ codes, we apply the correction Z_4 on each code block that has detected an error. The affect of this correction is:

$$Z_1 \to Z_4 \to Z_1Z_4 = \overline{Z}_1\overline{Z}_2 \cdot S_Z \tag{5.8}$$

$$Z_2 \to Z_4 \to Z_2Z_4 = \overline{Z}_1 \tag{5.9}$$

$$Z_3 \to Z_4 \to Z_3Z_4 = \overline{Z}_2 \tag{5.10}$$

$$Z_4 \to Z_4 \to I \tag{5.11}$$

Hence we have successfully turned every single qubit physical error into zero, one, or two logical errors on the outer HGP code. Additionally, since Z_4 does not overlap with the X-type logical operators, it ensures that 1. the syndrome is consistent between decoding stages, and 2. errant corrections do not (immediately) affect the logical state of the $[[4, 2, 2]]$-concatenated HGP code. This provides a built-in robustness to measurement errors on the Iceberg code blocks. The second decoding stage treats the residual logical errors on the inner $[[4, 2, 2]]$ code blocks as physical errors on the outer HGP code. Decoding the outer code provides a *logical* correction,

which we then translate into a physical correction to apply to the system. The decoding process for X-type errors is analogous, but we instead apply X_1 as the correction for the inner Iceberg codes.

It has been shown that the decoding performance of concatenated codes can be improved by passing soft information between levels [98, 99]. In particular, observing a non-zero syndrome for an Iceberg block informs us that an error is much more likely to be located on those four physical qubits. After the application of the initial correction, Eqs. (5.8)-(5.11), this translates to an increased probability of there being a logical error on the two corresponding logical qubits. To take advantage of this extra information, we can update the channel probabilities given to the HGP decoder. Out of the four error scenarios, both $\overline{Z}_1$ and $\overline{Z}_2$ see an error in two of them; thus we can set the probability of having a logical error at 0.5 for any logical qubit of an Iceberg block detecting an error. The probabilities for all other logical qubits are set to p^2. The decoding performance could potentially be improved further by using an overlapping window [201] or circuit-level decoder [104, 202, 203]. We leave it to future work to adapt the concatenated decoding scheme presented here to these methods.

5.4.2 Memory experiments

The memory experiments consist of a noiseless initialization of the code in the joint $|\overline{0...0}\rangle$ state, r rounds of noisy syndrome extraction and correction, and a destructive measurement of the data qubits. From the destructive measurement, a final (noiseless) syndrome as well as the value of the logical qubits can be constructed. Decoding is considered a success if the logical qubits are measured in the $|\overline{0...0}\rangle$ state, otherwise we say a logical error has occurred. The logical error

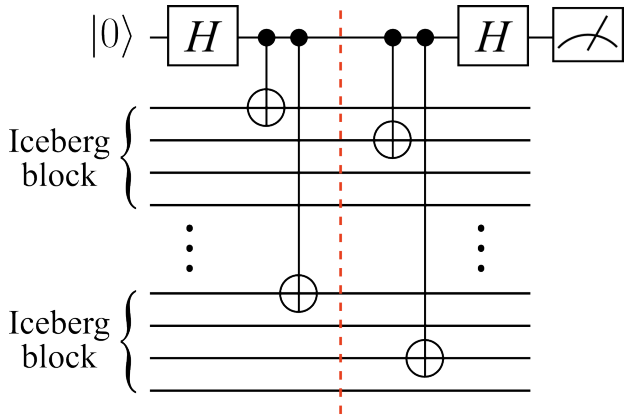

Figure 5.4: Circuit to measure an X-type concatenated HGP generator. A hook error that occurs in the middle of the circuit will propagate to a large number of detectable errors.

rate, p_L, is the fraction of samples where at least one logical qubit experiences a logical error. The standard deviation on p_L is the standard error when sampling from a binomial distribution, $\sigma_{p_L} = \sqrt{p_L(1-p_L)/N}$, where N is the number of collected shots. From p_L, we can calculate the logical error rate per round

$$\epsilon_L = 1 - (1 - p_L(r))^{1/r}, \tag{5.12}$$

where $p_L(r)$ is the observed logical error rate at round r. The standard deviation on ϵ_L, which is shown as the error bars in the plots here is then given as:

$$\sigma_{\epsilon_L} = \left(\frac{\partial \epsilon_L}{\partial p_L}\right) \sigma_{p_L} = \frac{1}{r}(1-p_L)^{1/r-1}\sigma_{p_L}. \tag{5.13}$$

Our simulations use a circuit-level noise model which is parameterized by a noise strength p and consists of the following: One-qubit Clifford gates are followed by a one-qubit depolarizing noise channel of strength $p/10$; two-qubit Clifford gates are followed by a two-qubit depolarizing noise channel of strength p; measurement results are flipped with probability p; qubit reset operations have probability p of preparing the $|1\rangle$ state instead of the $|0\rangle$ state; and, idle qubits have

a one-qubit depolarization noise channel of strength $p/10$ applied to them. This noise model, where single-qubit and idling errors are reduced, is physically relevant for ion-trap [3, 42, 48, 49] and neutral-atom [45, 50, 104] quantum computers.

To measure the stabilizer generators of the Iceberg code, we use the fault-tolerant syndrome extraction circuit from Ref. [87]. The concatenated and non-concatenated HGP generators are measured using a bare ancilla qubit and a circuit derived from an edge coloration of the Tanner graph [103], from which we can apply accurate idling error, see Ref. [103] for details. We use an implementation of the bipartite graph edge coloration algorithm by Pattison [204]. Whereas HGP codes enjoy a robustness to hook errors [58] regardless of the CNOT gate ordering during syndrome extraction [205, 206], we have to be careful with the CNOT gate ordering for the concatenated HGP generators. For certain orderings, a single error in the middle of the circuit will propagate to a large number of logical Iceberg errors. This is especially detrimental in the adaptive scheme as these logical errors will not be detected by the Iceberg blocks, and so the overlapping HGP generators will not be measured. As such, we employ the circuit shown in Fig. 5.4: By measuring the first qubit of each Iceberg logical before the second, we ensure that an error in the middle of the circuit will propagate to a large number of detectable errors.

To actually perform the circuit-level memory experiments, we use `Stim` [106]. Since the adaptive scheme requires us to make decisions in real-time based on the outcomes of stabilizer generator measurements, we have to use the slower `TableauSimulator` which allows us to access the measurement results. It is possible to record the executed circuit and generate a detector error model which could then be used to perform circuit-level decoding; however, we found that generating the detector error model and corresponding parity check matrix yielded simulation times on the order of seconds per shot. As such, we use the phenomenological decoders

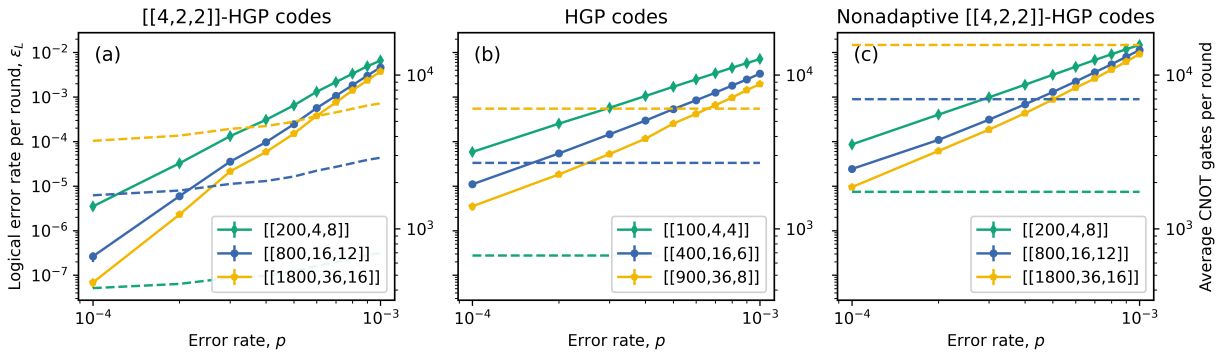

Figure 5.5: Logical error rate per round, ϵ_L as a function of the physical error rate, p. (a) $[[4, 2, 2]]$-concatenated HGP codes where the syndrome extraction circuit is adaptive. (b) Non-concatenated HPG codes using normal syndrome extraction. (c) $[[4, 2, 2]]$-concatenated HGP codes where the syndrome extraction circuit is not adaptive; that is, every generator is measured in every round. Error bars indicate the standard deviation on ϵ_L, Eq. (5.13). The dashed lines and right y-axis show the number of CNOT gates in the syndrome extraction circuit averaged over the $r = 100$ rounds.

described below. Making the change to a space-time circuit-level decoder would likely improve the performance of the scheme [104].

We initially found that the $[[4, 2, 2]]$-concatenated HGP codes implemented using the adaptive scheme were not single-shot [59], see Fig. 5.6(b). Whereas the HGP codes exhibit a logical error rate per round that stabilizes with increasing r, we see no such behavior out of the concatenated codes. On the contrary, we no longer observe a pseudothreshold when increasing r. The main cause of this behavior is most likely an accumulation of logical errors in the Iceberg code blocks; in a normal concatenated QEC scheme this is preventable as these logical errors are detected by the concatenated HGP generators in subsequent rounds. In the adaptive scheme, however, the logical errors are not detected, and the corresponding HGP generators are not measured, hence leading to an overwhelming build-up of errors. To alleviate this, we measure the entire set of generators every r' rounds to pick up residual logical errors. This *unmasking* [117] frequency is set at every 10 rounds for $p = 0.1\%$ and scales inversely with p.

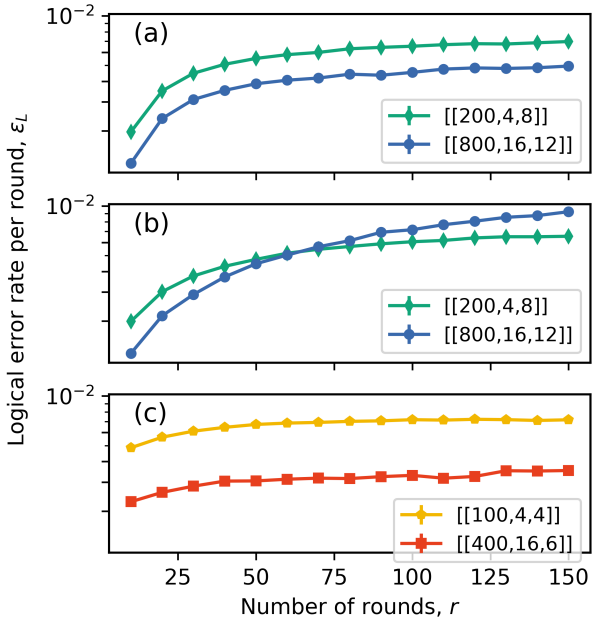

Figure 5.6: Logical error rate per round ϵ_L as a function of the number of rounds of quantum error correction. The physical error rate p is fixed at 0.1%. (a) $[[4, 2, 2]]$-concatenated quantum expander codes using the adaptive scheme where the entire set of generators is measured every 10 rounds. (b) $[[4, 2, 2]]$-concatenated HGP codes using the adaptive scheme where generators are never fully unmasked. (c) Normal quantum expander codes.

With this change we now observe single-shot performance out of the concatenated codes, with the logical error rate per round stabilizing by approximately round $r = 100$, see Fig. 5.6(a). As such, we perform $r = 100$ rounds of QEC in the following memory experiments. After each syndrome extraction, we attempt to guess a correction with only the syndrome information from that round. We use two different decoders based on the current QEC round. For intermediate rounds, $r < 100$, we use only the 'product-sum' BP [207] variant, which consists of 30 iterations of serial [169] message passing. Whether or not BP converges to a valid solution, we apply the guessed correction and continue the simulation. The final noiseless syndrome is decoded using belief propagation plus localized statistics decoding (BP-LSD) [208], which is based on belief propagation plus ordered statistics decoding (BP-OSD) [93]. We again allow only 30 iterations of product-sum BP with serial scheduling, but if a valid correction is not obtained, order-4 com-

bination sweep LSD is used. The process for decoding the $[[4, 2, 2]]$-concatenated HGP codes is identical after converting the physical errors into logical errors, following Section 5.4.1.

5.4.2.1 Quantum expander codes

We first investigate a family of quantum expander codes [79], as discussed in Section 2.3.1 and Section 5.4. Again using the configuration model and edge swapping [94], we generate random $(3, 4)$-regular classical codes which yield a family of hypergraph product codes with rate $k/n \geq 1/25$ and $[[4, 2, 2]]$-concatenated HGP codes with rate $k/n \geq 1/50$. Fig. 5.5 displays the logical error rate per round ϵ_L as a function of p for the HGP codes and the corresponding $[[4, 2, 2]]$-concatenated HGP codes. The dashed lines and right y-axis of Fig. 5.5 show the number of CNOT gates in the syndrome extraction circuit averaged over the $r = 100$ rounds.

At high error rates, the adaptive scheme is outperformed by the non-concatenated HGP codes. Additionally, we find a lower pseudothreshold when using the adaptive scheme, see Fig. 5.7. However, we see over an order of magnitude performance improvement using the adaptive scheme at low error rates. This is the regime in which we expect the $[[4, 2, 2]]$-concatenated codes and the adaptive scheme to work the best; when the errors are infrequent, the overlapping HGP generators are not measured as often, hence leading to fewer applied gates and less induced circuit noise. The non-adaptive concatenated codes perform slightly worse than their non-concatenated counterparts and significantly worse than their adaptive counterparts while using twice as many qubits and more CNOT gates.

All Iceberg-concatenated qLDPC codes have a minimum CNOT gates per round of $2n$, i.e. just measuring the two weight-n Iceberg generators. For $(3, 4)$-regular quantum expander

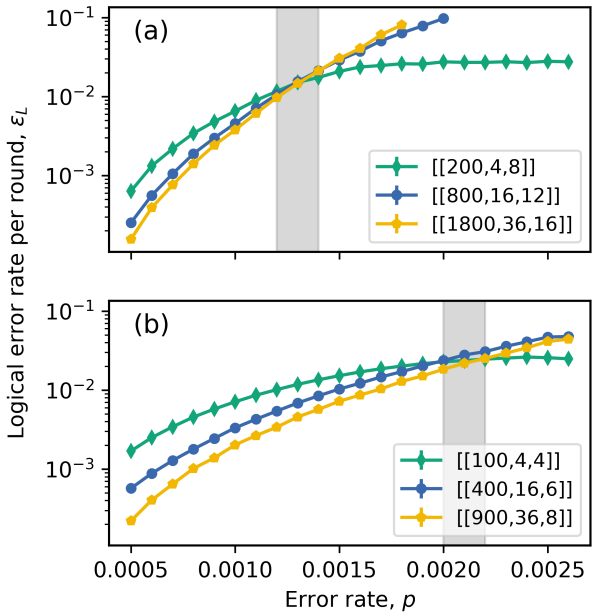

Figure 5.7: Logical error rate per round ϵ_L as a function of the physical error rate p around the pseudothreshold. (a) $[[4, 2, 2]]$-concatenated quantum expander codes using the adaptive scheme exhibit a threshold around 0.13%. (b) Non-concatenated, non-adaptive quantum expander codes display a higher pseudothreshold around 0.21%, which is comparable to that the value of 0.23% reported in Ref [103] for a similar noise model and decoder. Again, $r = 100$ rounds were used as the total simulation time.

codes, which are $(4, 7)$-qLDPC, this means we achieve a maximum $\sim 2\times$ reduction in CNOT gate count. Using qLDPC codes with higher weight generators would provide better potential savings, but a large stabilizer weight poses problems for physical implementations [186, 206]. Note that despite the reduced CNOT count, the circuit depth is often not lower for the $[[4, 2, 2]]$-concatenated codes. With all-to-all connectivity and an edge coloring of the Tanner graph, the syndrome extraction for a (Δ_V, Δ_C)-qLDPC code can be done in depth $2\Delta_C + 3$ [103]. The corresponding $[[4, 2, 2]]$-concatenated codes are $(2\Delta_V + 1, 2\Delta_C)$-qLDPC and require a circuit depth of $4\Delta_C + 8$ to measure all stabilizer generators. The additional 5 layers come from the adaptive scheme and the separate measurements of the Iceberg generators. Hence except on rounds where no Iceberg blocks detect an error, the concatenated codes require deeper syndrome extraction circuits. This is true even when a single Iceberg block detects an error, in which case the remaining blocks must sit idle until syndrome extraction has completed; however it may be feasible for an intelligent, real-time circuit compiler [209] to recognize this and perform gates on the unused blocks at the same time.

However, this circuit-depth analysis only holds on quantum computers which can implement arbitrarily many two-qubit gates in parallel: for architectures unable to achieve maximum parallelization, the CNOT count directly impacts the QEC cycle time. In particular, trapped-ion quantum computers typically have many more qubits than gating abilities, with some architectures containing a small number of zones in which gates can be performed [3, 32] and others requiring completely serial execution. An additional scenario in which CNOT count might play an outsized role is if we needed to first compile to a device without all-to-all connectivity, where the implementation of nonlocal two-qubit gates would introduce overhead and reduce parallelization. We further discuss this latter setting in Section 5.5.1.

5.4.2.2 La-cross codes

We also investigate several families of small La-cross codes [124]. La-cross codes are families of hypergraph product codes where the parity check matrix of the base classical code is circulant. That is, the parity check matrix consists of cyclic shifts of a n-dimensional vector, $(c_0, c_1, ..., c_{n-1}) \in \mathbb{F}_2^n$. The full parity check matrix $H \in \mathbb{F}_2^{n\times n}$ is then:

$$H = \begin{pmatrix} c_0 & c_1 & c_2 & \cdots & c_{n-1} \\ c_{n-1} & c_0 & c_1 & & \\ c_{n-2} & c_{n-1} & c_0 & & \vdots \\ \vdots & & & \ddots & \\ c_1 & c_2 & c_3 & \cdots & c_0 \end{pmatrix}. \tag{5.14}$$

The seed vector can be represented with a degree-n polynomial of the form:

$$h(x) = \sum_{i=0}^{n-1} c_i x^i. \tag{5.15}$$

In this work, we follow Ref. [124] and consider codes whose polynomials take the form $h(x) = 1+x+x^z$. By taking the first $n-z$ rows of H, this yields a new parity check matrix $H' \in \mathbb{F}_2^{(n-z)\times n}$. The corresponding classical code has parameters $[n, z, d]$ and the square HGP code, HGP(H', H') has parameters $[[n^2 + (n-z)^2, z^2, d]]$. Fig. 5.8 displays the logical error rate per round ϵ_L as a function of the physical error rate p for the La-cross codes and the corresponding $[[4, 2, 2]]$-concatenated La-cross codes. We again perform memory experiments for $r = 100$ rounds.

For these codes, the adaptive scheme outperforms the non-concatenated codes by nearly an

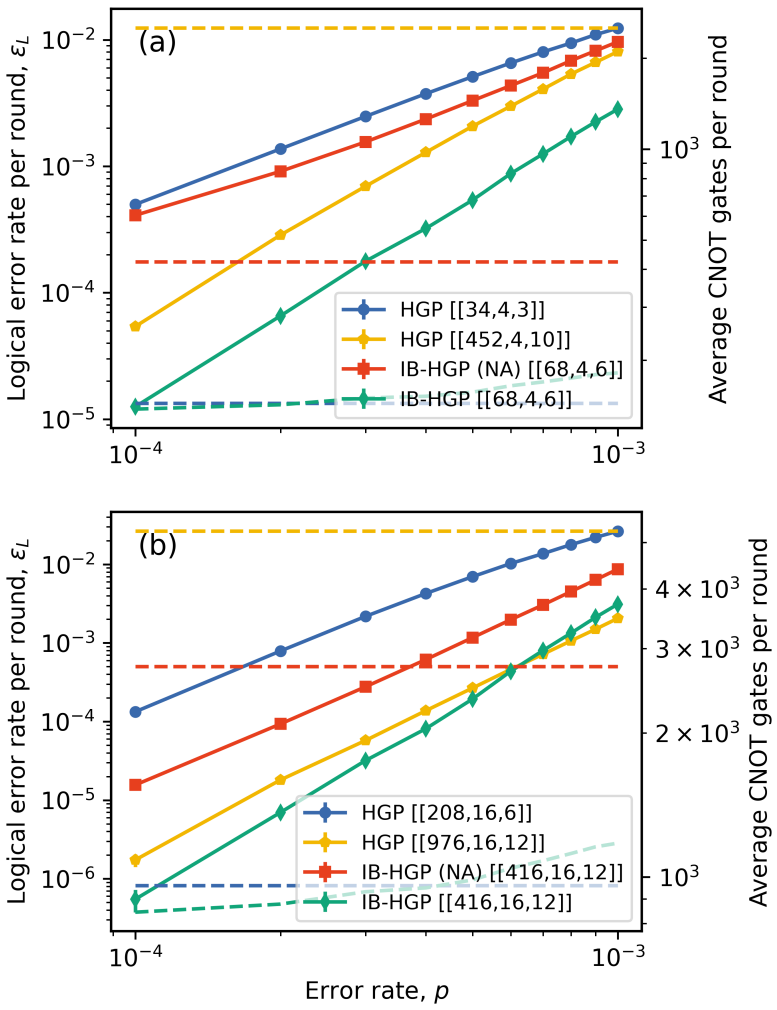

Figure 5.8: Logical error rate per round ϵ_L as a function of the physical error rate, p for a family of (a) $z = 2$ La-cross codes and the corresponding $[[4, 2, 2]]$-concatenated La-cross codes (b) $z = 4$ (concatenated) La-cross codes. The $[[4, 2, 2]]$-concatenated codes are labeled by IB-HGP. We also show a concatenated code which was measured using a non-adaptive syndrome extraction circuit, labeled as IB-HGP (NA). The dashed lines and right y-axis show the number of CNOT gates in the syndrome extraction circuit averaged over the $r = 100$ rounds.

Code	$\overline{q}$	$\overline{w}$	p	$\overline{q}_{adapt}$	$\overline{w}_{adapt}$
$[[100, 4, 4]]$	6.72	7	-	-	-
$[[400, 16, 6]]$	6.72	7	-	-	-
$[[200, 4, 8]]$	8.72	8.90	10^{-3}	3.62	5.76
-	-	-	10^{-4}	2.06	4.08
$[[208, 16, 6]]$	4.61	5.0	-	-	-
$[[400, 16, 8]]$	5.04	5.25	-	-	-
$[[416, 16, 12]]$	6.62	6.88	10^{-3}	2.83	4.86
-	-	-	10^{-4}	2.03	4.04

Table 5.1: Average check weight, $\overline{w}$, and average number of generators each qubit participates in, $\overline{q}$, for several HGP and $[[4, 2, 2]]$-concatenated HGP codes. We also show average check weight $\overline{w}_{adapt}$ and average qubit degree $\overline{q}_{adapt}$ obtained from averaging over $r = 100$ rounds of adaptive syndrome extraction at two physical error rates, p.

order of magnitude across the entire range of physical error rates. We were not able to scale up the $k = 4$ non-concatenated La-cross codes to match the performance of the $[[4, 2, 2]]$-concatenated codes. Larger, $k = 16$ non-concatenated La-cross were able to nearly match the performance of the concatenated codes and the adaptive scheme; however, they required over twice as many physical qubits and $\sim 5\times$ more CNOT gates. We again see that the non-adaptive codes perform comparatively worse, indicating that the adaptive scheme itself provides benefits to the logical error rate.

To illustrate how adaptive syndrome extraction can function as a quantum weight reduction scheme in the space-time picture, we display in Table 5.1 the average check weight $\overline{w}$ and the average number of generators each qubit participates in $\overline{q}$. When implemented using non-adaptive syndrome extraction, the $[[4, 2, 2]]$-concatenated HGP codes have higher weight checks and its qubits are involved in more checks than the corresponding non-concatenated codes. However, if we use adaptive syndrome extraction and average over many rounds, we find large reductions in

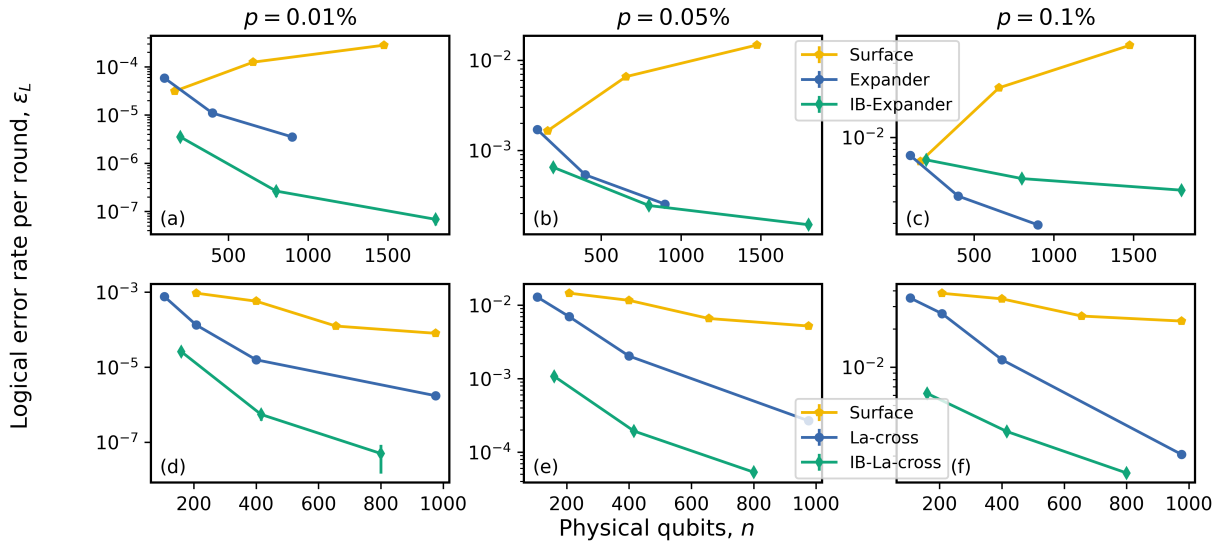

Figure 5.9: Logical error rate per round, ϵ_L as a function of the number of physical qubits. Panels (a)-(c) display quantum expander codes and $[[4, 2, 2]]$-concatenated quantum expander codes (labeled IB-Expander). As a point of reference, we also include the performance of multiple surface code patches encoding $k = 4,\ 16,\ 36$ logical qubits. Using $4 \times [[41, 1, 5]],\ 16 \times [[41, 1, 5]]$, and $36 \times [[41, 1, 5]]$ yield blocklengths similar to the IB-expander codes. Panels (d)-(f) display $z = 4$ La-cross codes and $[[4, 2, 2]]$-concatenated La-cross codes (labeled IB-La-cross). Also shown for comparison are $k = 16$ copies of the surface code, the distances of which were chosen to yield codes that approximately matched the blocklength of the IB-La-cross codes. Specifically, we used 16 patches of the $[[13, 1, 3]],\ [[25, 1, 4]],\ [[41, 1, 5]$, and $[[61, 1, 6]]$ surface codes, respectively.

$\overline{q}$ and $\overline{w}$. At low error rates, only the two Iceberg generators are measured, yielding $\overline{q}_{adapt}$ and $\overline{w}_{adapt}$ that approach 2 and 4, respectively. This is even better than notable topological codes like the surface code [38, 39], with $\overline{q} = \overline{w} = 4$ or the color code [210], with $\overline{q} = \overline{w} = 6$. Note, however, that the maximum stabilizer weight for the $[[4, 2, 2]]$-concatenated codes is still much larger at 14; it is just that when averaged over many syndrome extraction rounds, a majority of the generators that get measured come from the Iceberg code blocks.

To further emphasize the potential savings achievable by using the adaptive scheme, we plot in Fig. 5.9 the logical error rate per round ϵ_L versus the number of physical qubits at three different physical error rates, $p = 0.01\%,\ 0.05\%,\ 0.1\%$. The data shown here is the same as that

displayed in Fig. 5.5 and Fig. 5.8. Panels (a)-(c) display the performance of quantum expander codes. In this representation, it is clear that the adaptive scheme does not provide any benefits above $p = 0.05\%$; however, at $p = 0.01\%$ we observe a tradeoff of fewer logical qubits but lower logical error rates. As a point of reference, we also include the performance of $r = 100$ rounds of syndrome extraction for surface codes encoding $k = 4,\ 16$, and 36 logical qubits while using approximately the same number of physical qubits as the $[[4, 2, 2]]$-concatenated quantum expander codes. To obtain the logical error rate for k surface code patches, we calculate

$$p_{L,k} = 1 - (1 - p_{L,1})^k, \tag{5.16}$$

where $p_{L,1}$ is the logical error rate for a single patch. We can then compute ϵ_L using Eq. (5.12). Quantum expander codes have a constant encoding rate which means that the surface codes are not allowed to scale, hence leading to an increasing ϵ_L when encoding more logical qubits. Panels (d)-(f) display the performance of $z = 4$ La-cross codes encoding $k = 16$ logical qubits. Here we now see the $[[4, 2, 2]]$-concatenated codes and the adaptive scheme producing lower logical error rates while also using fewer physical qubits across the entire range of physical error rates. Also shown for comparison are $k = 16$ copies of the surface code, the distances of which were chosen to yield codes that approximately matched the blocklength of the $[[4, 2, 2]]$-concatenated La-cross codes. We note that La-cross codes were already shown to outperform surface codes in Ref. [124].

5.5 Applications

In this section, we propose specific use-cases where $[[4, 2, 2]]$-concatenated codes and the adaptive scheme might further outperform their non-concatenated, non-adaptive counterparts.

5.5.1 2D-local implementations of nonlocal codes

One potential application of $[[4, 2, 2]]$-concatenated codes and the adaptive scheme is in implementations of nonlocal qLDPC codes on restricted architectures. There is a close relationship between *locality* and the parameters and properties of a quantum error correcting code. After embedding the Tanner graph of a quantum code into $\mathbb{Z}^2$, a code is considered 2D-local if the syndrome extraction circuits can be implemented using only gates between neighboring qubits. Codes such as the surface code [38, 39] and color code [210] are 2D-local; as such, they are leading candidates for architectures like superconducting qubits which are currently limited to local gates only. However, with ease of implementation comes restrictions in the asymptotic rate and distance of the code, as was shown in Refs. [108, 109]. To get around these bounds, long-range interactions must be introduced [113], posing difficulties for hardware without long-range gates. Several recent works have provided methods for implementing nonlocal qLDPC codes on 2D-local architectures [25, 103, 141, 184], but it remains an open question as to the most effective methods and codes for this task.

First notice that $[[4, 2, 2]]$-concatenated codes have an embedding into $\mathbb{Z}^2$ which guarantees *some* locality regardless of the outer code choice, see Fig. 5.1. Placing an ancilla qubit in each Iceberg codeblock would allow all blocks to be measured with only 2D-local interactions. This, along with the adaptive scheme, may facilitate 2D-local implementations when the outer code is

very nonlocal. Previous work has indicated that implementing nonlocal, high-rate qLDPC codes in 2D appears to be challenging [103] without mitigating the additional cost of implementing nonlocal gates [25, 141, 184]. While the $[[4, 2, 2]]$-concatenated generators would inherit this nonlocality, the adaptive scheme ensures that the usage of these nonlocal checks would decrease at low physical error rates. Indeed, for QEC rounds in which only the Iceberg generators are measured, we obtain a syndrome extraction circuit that is completely local. This behavior might make naive 2D-local implementations of nonlocal codes feasible.

Certain codes have a Tanner graph structure which allows for convenient embeddings into $\mathbb{Z}^2$, such as the La-cross codes [124] studied here, the related generalized bicycle codes [85, 86], and long-range-enhanced surface codes [123]. One such embedding for La-cross codes is shown in Fig. 5.10, which yields an embedded generator structure consisting of some local connections and some nonlocal connections. Codes like these may be easier to implement in 2D due to their limited nonlocality. Unfortunately, if we attempt to recreate this embedding with the $[[4, 2, 2]]$-concatenated La-cross code, much of this locality is destroyed by our mapping of physical HGP qubits to logical Iceberg qubits. Instead we can use a mapping that assigns a single physical HGP qubit to each Iceberg block, with the other logical qubit acting as a gauge qubit [197]. With this one-to-one mapping between physical HGP qubits and Iceberg blocks, we can obtain essentially the same locality (although spread out by the Iceberg blocks themselves) for the concatenated generators. This is in addition to the Iceberg generators, which would continue to be local.

We note that using $[[4, 2, 2]]$-concatenated codes as a way to reduce the cost of implementing nonlocal codes was introduced by Criger and Terhal in Ref. [197], where they investigated the $[[4, 2, 2]]$-concatenated toric code. This code also has an embedding into $\mathbb{Z}^2$ in which there are both local and nonlocal checks. The authors discussed a superconducting transmon qubit ar-

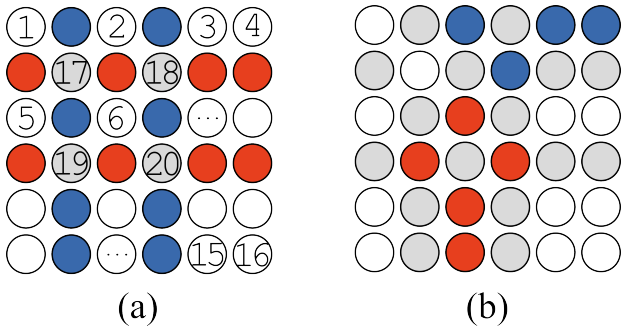

Figure 5.10: (a) Layout of the data and check qubits of a $[[20, 4, 2]]$ La-cross code. It can be interpreted as an interleaving of the L (white) and R (gray) sector qubits with qubits used for syndrome extraction readout. Red and blue circles represent X-type and Z-type check readout qubits, respectively. (b) Example X-type (red) and Z-type (blue) stabilizer generators. Remaining data and ancilla check qubits are colored white and gray, respectively. The structure of La-cross codes produce generators which have both local and nonlocal connections.

chitecture and predicted that these code constructions would perform well in settings where both short-range, high-fidelity and long-range, low-fidelity gates are available.

5.5.2 Modular QPU architectures

A related application is implementing qLDPC codes on a modular system of quantum processing units (QPUs). The current generation of quantum computers have around 10^2-10^3 physical qubits with which small experiments and calculations can be performed; however, scaling a single chip much beyond this size is unlikely due to physical and practical limitations. An alternative to this monolithic architecture is one where many smaller chips are linked together by means of, e.g., photonic interconnects [211], to form a larger, modular system. For these systems, the difficulty is now facilitating inter-module interactions and designing algorithms and QECCs which are well suited for the device.

A simple solution to this problem is to have a separate QECC for each module, and then the inter-module connections would only be used for interactions between code blocks. However, this

effectively limits the achievable distance across the entire system, and so combining modules for the same QECC might be desirable [212]. This setting fits well with the motivation of the adaptive scheme. The inter-module connections will be much slower and noisier than the intra-module connections, and as such, minimizing the usage of the inter-module connections would likely improve logical performance and speed up the computation. Concatenated codes again provide a reasonable approach: in each module suppose we put a copies of an ℓ-level concatenated code $\mathcal{Q}_\ell$. We can then form a final concatenated code from the logical qubits of the copies of $\mathcal{Q}_\ell$, hence obtaining a code that spans over all modules. The functionality of the adaptive scheme for this code would be similar to that described in Section 5.1, with the main change being that error correction is attempted with the syndrome information from the first ℓ levels. In this case, the only time when the inter-module connections would be needed during syndrome extraction is when the lower concatenation levels were unable to fully correct the error.

5.6 Discussion

In this work, we have presented adaptive syndrome extraction, a protocol which makes real-time decisions during syndrome extraction in order to minimize the cost of quantum error correction. Our protocol can be interpreted as a quantum weight reduction scheme in the space-time picture. We considered a realization of this scheme through a concatenated code construction, where the logical qubits of many $[[4, 2, 2]]$ Iceberg code blocks are encoded into a square hypergraph product code. Note, however, that the scheme is not limited to the code construction presented here; the main necessity of a code used in adaptive syndrome extraction is the ability to cheaply determine whether a region or subset of qubits has an error, in which case more expen-

sive error correction can take place. Concatenated codes provide a straightforward framework to accomplish this, but it is feasible that non-concatenated codes could be designed with this feature in mind. One such option could be the homological product of an error detecting code with a high-rate qLDPC code. Or adaptive syndrome extraction may be possible even without any additional structure; instead, error detection could be facilitated by measuring a hitting set of stabilizer generators.

Through circuit-level simulations of repeated syndrome extraction, we found that $[[4, 2, 2]]$-concatenated HGP codes implemented using adaptive syndrome extraction achieve lower logical error rates than their non-concatenated counterparts while requiring fewer CNOT gates and using fewer qubits. Although under-performing their non-concatenated counterparts at high error rates, quantum expander codes and the adaptive scheme achieved over an order of magnitude improvement at low error rates while using $\sim 2\times$ fewer CNOT gates. These improved logical error rates come at the cost of fewer logical qubits for the same number of physical qubits. La-cross codes showed a more impressive comparative improvement: when using the adaptive scheme, we found an order of magnitude improvement in the logical error rate even at current experimental physical error rates. For non-concatenated La-cross codes, matching the performance of the adaptive scheme required over twice as many physical qubits and $\sim 5\times$ more CNOT gates. Despite not reducing circuit depth per se, we described how this reduction in gate count could lead to shorter QEC cycles on certain quantum computing architectures. We also showed how the scheme provides reductions in the check weight and qubit degree when averaged over many syndrome extraction rounds.

We also provided fault-tolerant gadgets that allowed us to achieve universal logical quantum computation with $[[4, 2, 2]]$-concatenated HGP codes. Overall, we showed that our choice of

code and syndrome extraction scheme reduces the CNOT gate count for error correction without increasing the space-time cost of logical computation. In doing so, we showed how the concatenated codes can inherit logical gates from the underlying HGP codes, and we adapted the GPPM scheme of Ref. [194] and the single-shot state preparation scheme of Ref. [81] to our logical computation scheme. By developing a single-shot state preparation scheme for our $[[4, 2, 2]]$-concatenated HGP code, we showed how we can construct logical gadgets that take $O(1)$ logical cycles, giving us the opportunity to trade space for time in the logical computation protocol. Performance and overheads could potentially be further improved by using more efficient magic state distillation [213] or alternative codes [214].

Future research could investigate the potential of replacing the inner and outer codes of the $[[4, 2, 2]]$-concatenated HGP codes studied here. It may be interesting to replace the $[[4, 2, 2]]$ code with larger Iceberg codes, or to use more concatenation layers akin to the many-hypercube [88] and related codes [195, 196]. Alternatively, the outer HGP code could be replaced with alternative qLDPC codes, such as bivariate bicycle codes [85], or lifted product codes [93, 215], among others. In both of these cases, one would likely lose the logical-physical qubit mapping of Sec. 5.2.1 and the resulting logical gates; however, there might use cases in which those choices are advantageous. Finally, there is an alternative concatenation scheme to that described by Procedure 1: for input codes $\mathcal{Q}_1, \mathcal{Q}_2$ with parameters $[[n_1, k_1, d_1]]$ and $[[n_2, k_2, d_2]]$, respectively, we can obtain a concatenated code with parameters $[[n_1 n_2, k_1 k_2, d \geq d_1 d_2]]$, see e.g. Section 3.5 of Ref. [67]. The performance of such a code in the adaptive scheme would likely be comparable, but it may provide opportunities for different logical gates [197].

Chapter 6: Conclusion

Prior to the works described in this thesis, it was the general consensus it was advantageous, perhaps even essential, to measure every stabilizer generator when performing quantum error correction. In this thesis, we have introduced partial quantum error correction, which we broadly defined to be using incomplete syndrome information from the code or neglecting to correct errors on some part of the system. We have shown that is *not* necessary to measure every stabilizer generator in order to obtain a threshold, and we have described several situations where it is actually better to not do so.

In Chapter 3, we introduced partial syndrome measurement and masking as a general framework for partial quantum error correction. We then motivated an application of partial syndrome measurement inspired by the so-called stacked model, in which generators acting on spatially distant qubits are measured less frequently than those which do not. As a first step into the investigation of this scheme, we considered a simplified version where the measured generators were randomly selected. There, we gave numerical and analytical proof showing that thresholds are obtainable even when a relatively high percentage of the stabilizer generators are not measured. While this chapter was ostensibly a precursor to the work described in Chapter 4, we noted how even the simplified scheme could be immediately applied to certain quantum computing architectures in order to reduce quantum error correction overheads.

In Chapter 4, we studied the aforementioned stacked model application in a more realistic setting; namely, we presented a full error correction protocol that was built on a bilayer architecture and used bivariate bicycle codes. In doing so, we discussed code embeddings, a parallel syndrome measurement scheme using fast routing with local operations and classical communication, and entanglement purification. We found that bivariate bicycle codes implemented in this model outperformed surface code while also using fewer physical qubits. Additionally, partial syndrome measurement reduced QEC cycle time without hurting the logical error rate. The techniques we proposed, including the greedy routing algorithm and the syndrome extraction using gate teleportation and Bell pair purification, may find separate use elsewhere.

In Chapter 5, we presented another application of partial quantum error correction in the form of adaptive syndrome extraction. There, we were motivated by the fact that much of the syndrome information is unnecessary, especially in the low error rate regime. We showed that for certain codes, namely a concatenated code consisting of a error detecting code and a high-rate low-density parity-check code, syndrome extraction could be optimized by utilizing the correlations in the syndromes between concatenation layers. Specifically, we looked at $[[4, 2, 2]]$-concatenated hypergraph product codes and found that, when implemented using the adaptive syndrome extraction scheme, they functioned as better quantum memories than non-adaptive and non-concatenated codes, while also using fewer CNOT gates and physical qubits. Furthermore, we show how to achieve fault-tolerant universal logical computation with $[[4, 2, 2]]$-concatenated hypergraph product codes without increasing the space-time cost as compared to non-concatenated hypergraph product codes.

As for direct follow-up research to the works presented in this thesis, there may be alternative error correcting codes which yield better performance when implemented using the proposed

schemes. The schemes themselves likely have potential optimizations which would improve their performance. An additional potential use for partial syndrome measurement is the implementation of *high-density* parity-check codes [142]—QECCs where the check weight is not constant. These codes are typically avoided, as their high check weight make achieving fault-tolerance difficult; however, for codes where only some checks are high-weight, we could temporarily avoid measuring them, just as we avoided measuring the nonlocal checks in Chapter 4. Indeed, codes with both low- and high-weight checks (concatenated codes such as the many-hypercubes code [88] are an example of which) may provide additional benefits over qLDPC codes, and it would be interesting to study their constructions and application to this scheme.

Except for the simplified model studied in Chapter 3, this thesis was fairly focused on the estimating the real-world performance of the proposed schemes through the use of realistic, circuit-level simulations; as such, it would be an interesting task to provide analytical arguments on the success of these protocols, akin to what was done in Refs. [25, 141]. In Chapter 5, we detailed many potential variations of adaptive syndrome extraction and the presented concatenated code structure. We also discussed the idea of adaptive syndrome extraction as a form of space-time weight reduction; this concept has the potential to formalized and expanded on. Besides acting as a quantum memory, it also would be interesting to expand the concepts presented in this thesis to logical computation [216]. Ultimately, partial quantum error correction is a broad enough concept that there are sure to be many other interesting applications for a wide variety of quantum computing architectures.

Appendix A: Decoding algorithms

In this appendix, we briefly discuss the specific decoders we used throughout this thesis.

A.1 Small-set flip

The small-set flip (SSF) decoding algorithm [79] aims to imitate the classical flip decoding algorithm used to decode classical expander codes [78]. As such, it is well suited for quantum expander codes [79] where the success of the decoder is guaranteed for errors of size less than the distance, as well as random errors of linear size [126]. However, it is also a valid decoding algorithms for HGP codes constructed from seed codes that are not classical expander codes, we just do not get the analytical guarantees.

Let $\mathcal{F}$ be the powerset of the qubits in X-type generators; for example, if we have the stabilizer generator $X_1X_2X_3$, then the corresponding powerset $\mathcal{F}$ is

$$\Big\{\emptyset, \{1\}, \{2\}, \{3\}, \{1,2\}, ..., \{1,2,3\}\Big\}. \tag{A.1}$$

Also let E be the initial X-type error. A single round takes as input a guessed error $\hat{E}_i$ and the syndrome of the remaining error $\sigma_i := \sigma_Z(E \oplus \hat{E}_i)$. The decoder then goes through all 'small-sets' $f \in \mathcal{F}$ and finds the one that when flipped maximizes the decrease in syndrome

Algorithm 4 Small-set flip decoding algorithm [79]

Require: (E, D)

while $\exists F \in \mathcal{F} : |\sigma_i| - |\sigma_i \oplus \sigma_Z(F)| > 0$ **do**

$$
\begin{aligned}
F_i &= \max_{F \in \mathcal{F}} \frac{|\sigma_i| - |\sigma_i \oplus \sigma_Z(F)|}{|F|} \\
\hat{E}_{i+1} &= \hat{E}_i \oplus F_i \\
\sigma_{i+1} &= \sigma_i \oplus \sigma_Z(F_i) \\
i &= i + 1
\end{aligned}
$$

end while

return $\hat{E}_i$

weight (proportional to the weight of the correction), which is then applied to the guessed error for the next round. Fig. A.1 displays an observed syndrome and the effect of two potential qubit flips that the SSF algorithm attempts. In reality, the highlighted X-type stabilizer generator has $2^5 = 32$ potential qubit flips to try, and the pattern in panel (c) fully resolves the syndrome. All other X-type stabilizer generators are also looked at to see if they contain qubit flips that further reduce the syndrome error. As discussed in Section 2.3.5, decoding is considered a success if the guessed error $\hat{E}$ is equivalent to the actual error E, that is $E \oplus \hat{E}$ belongs to the stabilizer group.

The complete decoding procedure is listed as pseudo-code in Algorithm 4. It takes as input a tuple (E, D), where $E \subseteq V$ is an $X-$type error, and $D \subseteq C_Z$ is a potential syndrome error—that is the algorithm runs instead on the syndrome where some values have been flipped, $\sigma_Z(E) \oplus D$. We make one small change to the algorithm for the purposes of using it in the context of the stacked model introduced in Chapter 3. Specifically, we exchange using the full syndrome for one taken from some subset of the stabilizer generators $U \subseteq S$. We still search through every opposite type generator when looking for small-sets F to flip; however, the effect of flipping will only be visible on the restricted set of generators $\sigma_U(F)$. We overload the meaning of having the

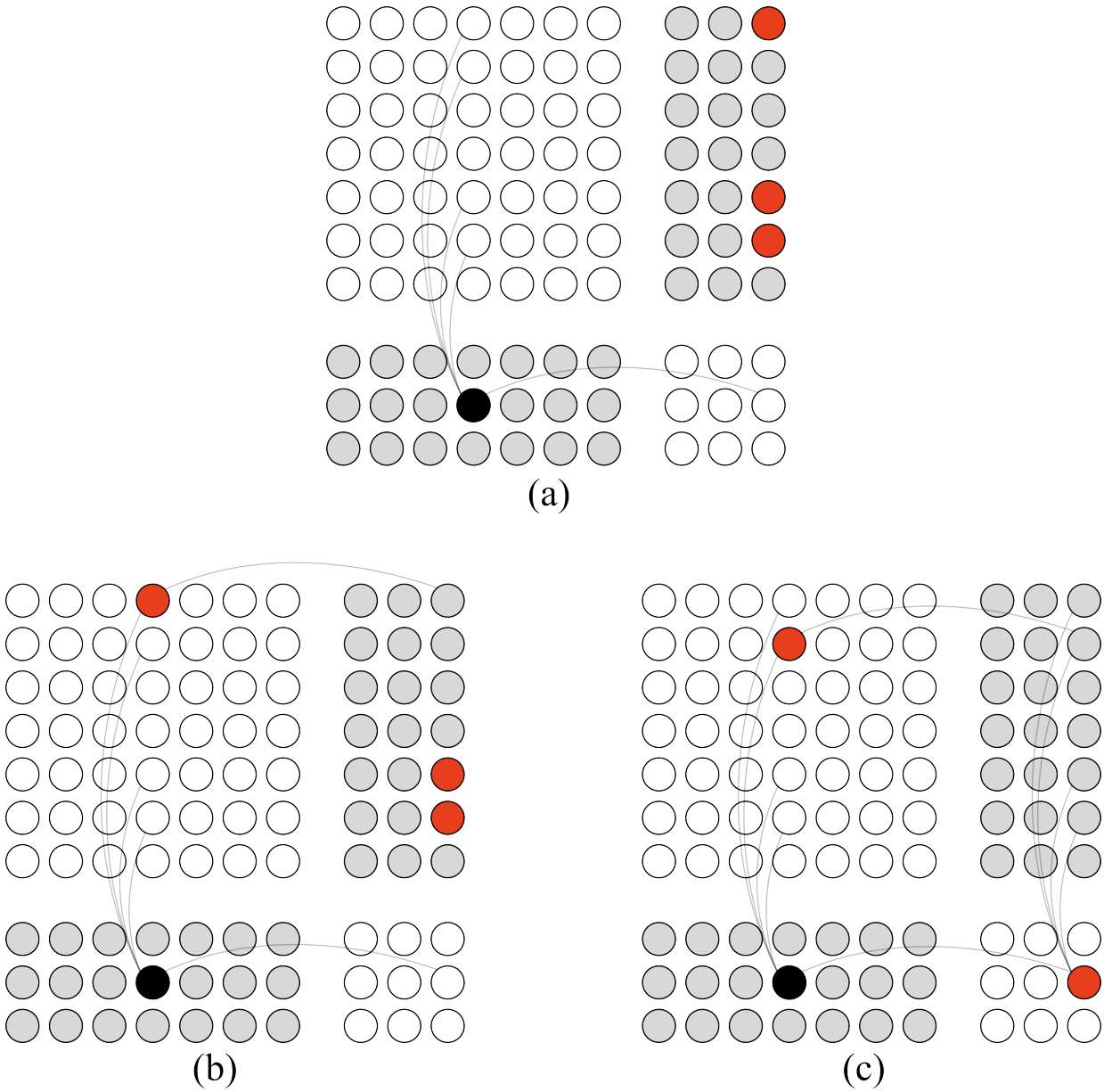

Figure A.1: Example of the search process SSF uses when searching for a correction. (a) Observed Z-type syndrome arising from an X error. An X-type stabilizer generator is highlighted in black. The SSF decoding algorithm also looks at all other X-type stabilizer generators. (b) One out of 32 qubit patterns SSF tries to flip, yielding a syndrome decrease of weight one. (c) Another qubit pattern SSF attempts, now yielding a syndrome decrease of three. SSF selects this as the most probable correction.

input (E, D) when $D \subseteq C_Z$ is interpreted as a mask, in which case the available syndrome is $\sigma_D(F)$. The chosen interpretation will be clear from context.

It was shown that the performance of SSF could be improved by using a second decoder after SSF, such as belief propagation [94], projection-along-a-line [217], or potentially ordered statistic decoding, see below.

A.2 Belief propagation

One of the most widely used decoding algorithms for both classical and quantum error correction is belief propagation (BP) [64]. BP operates through iterative message passing on the Tanner graph of the code $G = (V \sqcup C, E)$ gradually refining probability estimates until reaching a reliable decision about the most probable error consistent with the observed syndrome. It was shown that using BP, the performance of certain classical LDPC codes could approach the Shannon limit [65]. Although belief propagation performs remarkably well for classical LDPC codes, decoding CSS codes in this way yields suboptimal performance [89–91], due to short cycles in the Tanner graph and error degeneracy. To address the failures of belief propagation, post-processing methods have been introduced [92–94], such as ordered statistic decoding which we discuss in the next section. The inputs to BP are the observed syndrome $s = \sigma_Z(E)$ and prior probabilities $\{p_i\}$ of having errors on any of the (qu)bits. Note that $\Gamma(v_i)$ denotes the neighbors of a node v_i in G. The decoding process then proceeds as follows:

1. Initialize the beliefs λ_i and initial bit-to-check messages to:

$$m^0_{v_i \to c_j} = \lambda^0_i = \ln\Big(\frac{1-p}{p}\Big) \tag{A.2}$$

where p is the prior probability of a (qu)bit having an error. This information can come from the device or by track error propagation throughout a quantum circuit [106].

2. Each bit node sends a message to all connected check nodes, i.e. every $c_j \in \Gamma(v_i)$, according to

$$m^t_{v_i \to c_j} = \ln\Big(\frac{1-p}{p}\Big) + \sum_{c'_j \in \Gamma(v_i)\setminus c_j} m^{t-1}_{c'_j \to v_i} \tag{A.3}$$

3. The checks then send messages back to the bits according to

$$m^t_{c_j \to v_i} = (-1)^{s_j} 2 \tanh^{-1}\left(\prod_{v'_i \in \Gamma(c_j)\setminus v_i} \tanh\Big(\frac{m^t_{v'_i \to c_j}}{2}\Big)\right) \tag{A.4}$$

4. This process of passing messages between bits and checks is repeated either for a predetermined number of iterations or when a stopping criterion is met. Then the final belief for each bit node is computed:

$$\lambda^t_i = \lambda^0_i + \sum_{c_j \in \Gamma(v_i)} m^t_{c_j \to v_i} \tag{A.5}$$

5. Using the final belief, the value of the bits (the predicted error) is decided:

$$\hat{E}_i = \begin{cases} 0 & \lambda^t_i > 0 \\ 1 & \text{otherwise} \end{cases} \tag{A.6}$$

If, at this point, $\sigma_Z(\hat{E}) = \sigma_Z(E)$, then decoding is considered a success (notwithstanding the possibility that $\hat{E} \oplus E$ is a logical error), and no further decoding is required. If $\sigma_Z(\hat{E}) \neq \sigma_Z(E)$, then we can follow-up BP with a postprocessing decoder like OSD, see below. The beliefs λ_i provide insight into the probability that a certain bit has an error. This information can

be exploited by decoders such as OSD. There are several different variants of BP. The algorithm listed here is the product-sum algorithm; alternatively, there is also the minimum-sum, and log versions of the two.

A.3 Ordered statistic decoding

Ordered statistic decoding (OSD) was originally used for the decoding of classical binary linear codes [218], and it later introduced as a postprocessing decoding for quantum CSS codes [92, 93, 169]. It has become an extremely useful algorithm as it alleviates the problems of BP which arise from quantum degeneracy and short cycle length. Unfortunately, the runtime of OSD is quite restrictive at $O(n^3)$, and so it may not feasible when decoding codes with large blocklengths. Recently, Hillmann *et al.* developed a localized version of OSD that provides more favorable runtimes [208].

We again focus on decoding CSS codes, i.e. focusing on decoding X-type errors and Z-type errors independently. The input to OSD is the Z-type syndrome $\sigma_Z(E)$ and the reliabilities $R = \{p_1, p_2, ..., p_n\}$ of the qubits. These reliabilities are essentially the probability that a given qubit experiences error, and they can be obtained from prior knowledge about the device or from the soft-information output of a prior decoder, like BP. For a matrix $H = (h_1, h_2, ..., h_m)$, we denote $H_{\{I\}}$ as the matrix that results from taking columns from H according to $\{I\}$. We similarly define $E_{\{I\}}$ if E is a vector. With these two inputs, OSD proceeds as follows:

1. Construct $\{I\}$ as the indices from the Rank(H_Z) most likely qubits according to R.

2. Construct a new matrix $\tilde{H} = H_{\{I\}}$, where $H = H_Z$. The resulting matrix is full-rank and can be inverted.

3. Solve the equation $E_{\{I\}} = \tilde{H}^{-1}\sigma_Z(E)_{\{I\}}$.This is where the main $O(n^3)$ computational complexity from OSD comes from, since Rank(H_Z) is often proportional to the number of qubits in the code n.

4. The guessed error is then $\hat{E}_{\{I\}} = E_{\{I\}}$. The least likely, $[n]\backslash\{I\}$, qubits are assumed to not have an error, and so $\hat{E}_{[n]\backslash\{I\}} = 0$.

The algorithm described above is called OSD-0. Higher order OSD methods are for finding solutions where $\hat{E}_{[n]\backslash\{I\}} \neq 0$. Order-$w$ 'combination sweep' OSD begins by first performing OSD-0. If the syndrome of the guessed error matches the observed syndrome, further processing is not required. On the other hand, if $\sigma_Z(E) \neq \sigma_Z(\hat{E})$, then we attempt to alter $\hat{E}$. In particular, order-w combination sweep OSD searches over all weight w bitstrings for $\hat{E}_{[n]\backslash\{I\}}$. Other options for searching are possible and may provide better performance in certain situations [92, 93]. These higher-order OSD methods are even moreso computationally expensive, and so it may be advisable to only use OSD-0 or to keep w relatively small. We refer the reader to Ref. [92] for another in-depth introduction to OSD.

Appendix B: Logical computation for [[4,2,2]]-concatenated hypergraph product codes

B.1 Logical operators for the [[4,2,2]] code

B.1.0.1 Logical Clifford operators

We first discuss the different notions of gate transversality before introducing the various logical Clifford operators obtainable when we restrict ourselves to the various extents of transversality. Traditionally, transversal logical gates demand that every physical operation acts on one physical qubit in a code block. They form the foundation for fault-tolerant gates in quantum computation [219]. However, it is possible to relax this restrictive definition of transversality to what is known as *SWAP-transversal* where we allow qubit permutations as well as single-qubit Clifford gates [220]. This notion of transversality is not inherently fault-tolerant except on platforms that have the ability to physically shuttle qubits, such as ion-traps and neutral atom arrays.

In the recent work of Sayginel *et al.* [220], they derived a set of Clifford logical operators as shown in Table B.1.

Gate	Circuit	Type
$\overline{H}_1\overline{H}_2\overline{SWAP}_{1,2}$	$H_1H_2H_3H_4$	Transversal
$\overline{CZ}$	$S_1^\dagger S_2^\dagger S_3 S_4$	Transversal
$\overline{CNOT}_{1,2}$	$SWAP_{2,3}$	*SWAP*-transversal
$\overline{CNOT}_{2,1}$	$SWAP_{2,4}$	*SWAP*-transversal
$\overline{SWAP}_{1,2}$	$SWAP_{3,4}$	*SWAP*-transversal

Table B.1: *SWAP*-transversal logical Clifford gates of the $[[4, 2, 2]]$ code.

Because the logical Hadamard operator does not target individual logical qubits of the $[[4, 2, 2]]$ code, the set of *SWAP*-transversal logical Clifford gates in Table B.1 does not give us the full logical Clifford group. By including the gates shown in Table B.2, we are able to obtain a full set of logical Clifford operators; however, these gates are not transversal or *SWAP*-transversal and as such are not inherently fault-tolerant.

Gate	**Circuit**	**Type**
$\overline{S}_1$	$S_1S_3CZ_{1,3}$	General Clifford
$\overline{S}_2$	$S_1S_4CZ_{1,4}$	General Clifford
$\overline{\sqrt{X}_1}$	$\sqrt{X}_1\sqrt{X}_4C(X,X)_{1,4}$	General Clifford
$\overline{\sqrt{X}_2}$	$\sqrt{X}_1\sqrt{X}_3C(X,X)_{1,3}$	General Clifford

Table B.2: Logical Clifford gates of the $[[4, 2, 2]]$ code via embedded code technique.

B.1.0.2 State preparation

In this section, we describe how to prepare some of the common logical states for the $[[4, 2, 2]]$ code that we use for our logical computation protocol. Because the $[[4, 2, 2]]$ code is a CSS code, it is trivial to initialize it in the logical states $|\overline{00}\rangle$ and $|\overline{++}\rangle$. Thus, in this section, we

focus mainly on describing how we might be able to initialize it in the $|\overline{0+}\rangle$ state. This task is less trivial because the $[[4, 2, 2]]$ code only has global Hadamard as a transversal logical operator instead of one that targets a single logical qubit as shown in Section B.1.

In Fig. B.1, we provide a quantum circuit for transforming the logical $|\overline{00}\rangle$ state into $|\overline{0+}\rangle$ by using a joint XX measurement and a single Pauli operator for correction. Note that the joint XX measurement can be done fault-tolerantly with an additional pair of maximally-entangled ancilla qubits.

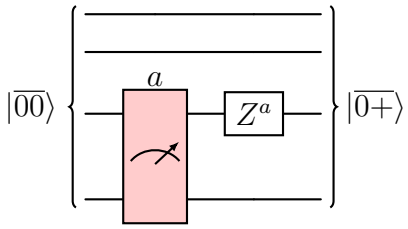

Figure B.1: Physical quantum circuit for transforming the $|\overline{00}\rangle$ state of the $[[4, 2, 2]]$ Iceberg code into $|\overline{0+}\rangle$. The four qubit lines correspond to the physical qubits of the $[[4, 2, 2]]$ code. The red measurement with measurement output a is a joint XX measurement.

It is easy to see that the quantum circuit in Fig. B.1 gives us the desired transformation. The logical $|\overline{00}\rangle$ and $|\overline{01}\rangle$ states are given by

$$|\overline{00}\rangle = \frac{|0000\rangle + |1111\rangle}{\sqrt{2}}, \quad |\overline{01}\rangle = \frac{|0011\rangle + |1100\rangle}{\sqrt{2}}. \tag{B.1}$$

Since $|\overline{0+}\rangle = \frac{1}{\sqrt{2}}\left(|\overline{00}\rangle + |\overline{01}\rangle\right)$, performing a XX measurement on the third and fourth qubits of $|\overline{00}\rangle$ and getting a measurement eigenvalue of $+1$ $(a = 0)$ gives us $|\overline{0+}\rangle$. When the eigenvalue is -1, we apply Z_3 to flip the phase to get the desired $|\overline{0+}\rangle$ state.

The $|\overline{0+}\rangle$ is particularly useful for logical qubit teleportation for our $[[4, 2, 2]]$ Iceberg code. We prepare the $|\overline{0+}\rangle$ state several times in the different logical gadgets that we construct in the

subsequent sections.

B.2 Clifford Logical Operators for [[4,2,2]]-concatenated HGP codes

In this appendix, we describe how we can obtain the full Clifford group for logical computation on a $[[4, 2, 2]]$-concatenated square HGP code. We follow the concatenation procedure described in Procedure 1 and assign logical qubits following Sec. 5.2.1. To achieve the full Clifford group for the resulting concatenated code we need the following logical gadgets:

1. $|\overline{0}\rangle$ / $|\overline{+}\rangle$ state preparation
2. Z/X basis measurements
3. Concatenated grid Pauli product measurements (cGPPMs)
4. Inter-block CNOTs
5. H-SWAP
6. CZ-S
7. Logical translation

In the subsequent sections, we discuss how each of the logical gadgets above can be constructed for our concatenated code. Because the logical translation gadget is only available for a special subclass of hypergraph product codes, we leave it until last, before we describe how we construct it. In fact, the first six logical gadgets can all be implemented without the logical translation gadget. However, we can achieve better space-time cost savings if we have a logical translation

gadget. Thus, we discuss the different costs of implementing the different logical gadgets both with and without the logical translation gadget in the subsequent sections.

B.2.0.1 State preparation

It is easy to check that our concatenated code is still a CSS code. Thus, the standard state preparation procedure would still work for our concatenated code [58]. For the sake of concreteness, we state how we can prepare the $|\overline{0}\rangle^{\otimes k}$ logical state.

1. Initialize all 2n physical qubits to be in the $|0\rangle$ state.
2. Measure all the X stabilizer generators for our concatenated code.
3. Repeat step 2 an additional $d-1$ times.
4. Decode the d sets of syndromes collectively and apply the resulting Z correction to obtain the desired $|\overline{0}\rangle^{\otimes k}$ logical state.

By exchanging X and Z as well as initializing the 2n physical qubits to be in the $|+\rangle$ state, we can prepare the $|\overline{+}\rangle^{\otimes k}$ logical state.

Alternatively, we can consider a single-shot $|\overline{0}\rangle$ / $|\overline{+}\rangle$ state preparation gadget introduced by Hong in Ref. [81]. We now proceed to show why our concatenated code is amenable to such a state preparation gadget. To do that, we need to show that the homological product of the concatenated code and some classical code has good soundness. In addition, it would be helpful for us to prove that the concatenated code has linear confinement to show that it is amenable to single-shot decoding.

To begin, we first define soundness and what it means for a quantum code to have good soundness.

Definition 17 ([60, Soundness]). *Let t be an integer and $f : \mathbb{Z} \to \mathbb{R}$ be some function called the soundness function with $f(0) = 0$. Given some set of Pauli checks M, we say it is (t, f)-sound if for all Pauli errors E with $|\sigma(E)| = x < t$, it follows that there exists an E' with $\sigma(E') = \sigma(E)$ such that wt$(E') \leq f(x)$.*

Definition 18 ([60, Good Soundness]). *Consider an infinite check family $\mathcal{M}_n$. We say the family has good soundness if each $\mathcal{M}_n$ is (t, f)-sound where:*

1. *t grows with n such that $t \geq an^b$ for some positive constants a, b. That is, $t \in \Omega\left(n^b\right)$ with $b > 0$;*

2. *and $f(x)$ is some polynomial that is monotonically increasing with x and independent of n.*

Having stated the above definitions, we now proceed to prove that the our concatenated quantum code has the following soundness:

Lemma 19 (Soundness of "Transposed" Concatenated Code). *Let the chain complex corresponding to the square HGP code be*

$$C_{-1} \xrightarrow[\delta_{-1}=H_X^\top]{} C_0 \xrightarrow[\delta_0=H_Z]{} C_1. \tag{B.2}$$

In addition, let the chain complex corresponding to our concatenated code be

$$\tilde{C}_{-1} \xrightarrow[\tilde{\delta}_{-1}=\tilde{H}_X^\top]{} \tilde{C}_0 \xrightarrow[\tilde{\delta}_0=\tilde{H}_Z]{} \tilde{C}_1 \tag{B.3}$$

Then, the maps $\tilde{\delta}_0^\top$ and $\tilde{\delta}_{-1}$ are (t, f)-sound with $f(x) = \frac{9x^2}{4} + x$ and $t = d/3$ where d is the distance of the HGP code.

Proof. From Lemma 5 in Ref. [60], we know that the HGP code that we use as our inner code has stabilizer check matrices H_X and H_Z such that $H_X^\top$ and $H_Z^\top$ are (t, f)-sound with $f(x) = x^2/4$ and $t = d$. For the subsequent part of the proof, we focus on analyzing $\tilde{H}_X^\top$ and the argument would also hold for $\tilde{H}_Z^\top$.

First of all, notice that $\tilde{H}_X^\top \in \mathbb{F}_2^{2n\times((n-k)+n/2)}$ and $H_X^\top \in \mathbb{F}_2^{n\times(n-k)}$. Denote the set of $(n-k)$ columns in $H_X^\top$ and $\tilde{H}_X^\top$ as A and the set of $n/2$ columns in $\tilde{H}_X^\top$ as B. We refer to the stabilizer generators of $\tilde{H}_X$ that correspond to the columns in A as the copied generators. The generators of $\tilde{H}_X$ that correspond to the columns in B are referred to as the new generators. Denote $\tilde{H}_X^\top|_A$ as the restriction of $\tilde{H}_X^\top$ to the columns in A.

For any error configuration $e \in \mathbb{F}_2^{(n-k)+n/2}$ that lies completely within the support of A, we have

$$\left|H_X^\top e\right| \leq \left|\tilde{H}_X^\top e\right| \leq 2\left|H_X^\top e\right| \tag{B.4}$$

from the fact that the copied generators have twice the weight of the original generators in H_X because each of the physical qubit checked by a generator in H_X is now a logical qubit that corresponds to two physical qubits in the 2n physical qubits of the concatenated code. For every $e \in \mathbb{F}_2^{n-k}$ that generates a syndrome in $\tilde{H}_X^\top|_A$ that has weight at most d, we have $\left|H_X^\top e\right| \leq \left|\tilde{H}_X^\top e\right| \leq d$. From the $(d, x^2/4)$-soundness of $H_X^\top$, there exists some $e' \in \mathbb{F}_2^{n-k}$ such that $H_X^\top e = H_X^\top e'$ and $|e'| \leq \left|H_X^\top e\right|^2/4$. Now, we claim that e' satisfies $\tilde{H}_X^\top|_A e = \tilde{H}_X^\top|_A e'$. To show that, we first note that there exists an injective homomorphism $\gamma : C_0 \to \tilde{C}_0$ since every physical qubit $i \in [n]$ is mapped to some unique pair of physical qubits (i, j) for $i, j \in [n]$ according to the

logical operator basis for the $[[4, 2, 2]]$ code. Thus,

$$\tilde{H}_X^\top|_A e = \left(\gamma \circ H_X^\top\right) e = \gamma H_X^\top e' = \tilde{H}_X^\top|_A e'. \tag{B.5}$$

Since $|e'| \leq \left|H_X^\top e\right|^2 /4 \leq \left|\tilde{H}_X^\top|_A e\right|^2 /4$, this allows us to conclude that $\tilde{H}_X^\top|_A$ is $(d, x^2/4)$-sound.

Because $\tilde{H}_X^\top|_B$ consists of $n/2$ different columns with disjoint sets of 4 consecutive ones, it is easy to see that $\tilde{H}_X^\top|_B$ is $(n/2, x/4)$-sound.

Now, we proceed to show that $\tilde{H}_X^\top$ is $\left(\frac{d}{3}, \frac{9x^2}{4} + x\right)$-sound using the arguments made regarding the soundness of $\tilde{H}_X^\top|_A$ and $\tilde{H}_X^\top|_B$. Consider an arbitrary error configuration $\tilde{e} \in \mathbb{F}_2^{(n-k)+n/2}$ such that $\left|\tilde{H}_X^\top \tilde{e}\right| = x \leq d/3$. Let $\tilde{e}|_A$ and $\tilde{e}|_B$ denote the restriction of $\tilde{e}$ to the indices in A and B respectively. We now claim that $\left|\tilde{H}_X^\top|_A \tilde{e}|_A\right| \leq 3x \leq d$. To see that, note that the qubit with index 2 in each Iceberg code block only lies in the support of the new generators since the Iceberg code's logical operator basis has $\overline{X}_1 = X_1 X_3$ and $\overline{X}_2 = X_1 X_4$. Since we have

$$\tilde{H}_X^\top \tilde{e} = \tilde{H}_X^\top|_A \tilde{e}|_A + \tilde{H}_X^\top|_B \tilde{e}|_B \pmod 2, \tag{B.6}$$

when $\left[\tilde{H}_X^\top|_A \tilde{e}|_A\right]_i = \left[\tilde{H}_X^\top|_B \tilde{e}|_B\right]_i \neq 0$ for some index i that corresponds to qubit 1, 3, or 4 in a Iceberg code block, there exists some j that corresponds to the qubit with index 2 in the same set of 4 Iceberg code qubits that contains i such that $\left[\tilde{H}_X^\top \tilde{e}\right]_j = \left[\tilde{H}_X^\top|_B \tilde{e}|_B\right]_j = 1$. Since $\left|\tilde{H}_X^\top \tilde{e}\right| = x$, we can have at most x instances of j in x different Iceberg code blocks such that $\left[\tilde{H}_X^\top \tilde{e}\right]_j = \left[\tilde{H}_X^\top|_B \tilde{e}|_B\right]_j = 1$. For each of these instances of j, we can have at most 3 i's that belong to the same 4 Iceberg code qubits that contains j such that $\left[\tilde{H}_X^\top|_A \tilde{e}|_A\right]_i = \left[\tilde{H}_X^\top|_B \tilde{e}|_B\right]_i \neq 0$. Thus, $\left|\tilde{H}_X^\top|_A \tilde{e}|_A\right| \leq 3x \leq d$. From the soundness of $\tilde{H}_X^\top|_A$ and $\tilde{H}_X^\top|_B$, we can find a $\tilde{e}'$ that has restricted

forms $\tilde{e}'|_A$ and $\tilde{e}'_B$ such that $|\tilde{e}'|_A| \leq (3x)^2/4 = 9x^2/4$, $|\tilde{e}'|_B| \leq 4x/4 = x$ where $\tilde{H}_X^\top \tilde{e} = \tilde{H}_X^\top \tilde{e}'$. Thus, we obtain the lemma statement where $\tilde{H}_X^\top$ is $\left(\frac{d}{3}, \frac{9x^2}{4} + x\right)$-sound. □

We note that the soundness factor can definitely be made tighter by performing a rigorous analysis similar to the one in the proof for Lemma 5 in Ref. [60]. However, we do not need such a tight bound for our subsequent arguments.

Now, we are ready to show that the thickened concatenated code has good soundness.

Lemma 20 (Good Soundness of Thickened Concatenated Code)**.** *Let the chain complex corresponding to our concatenated code be*

$$\tilde{C}_{-1} \xrightarrow[\tilde{\delta}_{-1}=\tilde{H}_X^\top]{} \tilde{C}_0 \xrightarrow[\tilde{\delta}_0=\tilde{H}_Z]{} \tilde{C}_1 \tag{B.7}$$

such that $\tilde{\delta}_0^\top$ and $\tilde{\delta}_{-1}$ are $\left(\frac{d}{3}, \frac{9x^2}{4} + x\right)$-sound where d is the distance of our concatenated code. Applying the homological product on the concatenated code with a classical code gives us a new length-4 chain complex

$$\breve{C}_{-2} \xrightarrow[\breve{\delta}_{-2}]{} \breve{C}_{-1} \xrightarrow[\breve{\delta}_{-1}=\breve{H}_X^\top]{} \breve{C}_0 \xrightarrow[\breve{\delta}_0=\breve{H}_Z]{} \tilde{C}_1 \tag{B.8}$$

where the map $\breve{\delta}_{-1}^\top = \breve{H}_X$ is $\left(\frac{d}{3}, \frac{9x^3}{4}\right)$-sound.

Proof. The proof for this lemma statement is effectively the same as the proof for Lemma 6 in Ref. [60] except that we use Lemma 19 from our paper instead of Lemma 5 from Ref. [60]. □

Next, we provide two important definitions regarding confinement.

Definition 21 ([101, Confinement])**.** *Let $t > 0$ be an integer and $f : \mathbb{Z} \to \mathbb{R}$ be an increasing function. For a parity-check matrix $H \in \mathbb{F}_2^{m \times n}$, we say it is (t, f)-confined if for any Pauli errors*

e with reduced weight $||e|| \leq t$, its syndrome $\sigma(e) = He$ obeys

$$f(|\sigma(e)|) \geq ||e||. \tag{B.9}$$

Definition 22 ([101, Good Linear Confinement]). *Consider an infinite check family $\mathcal{M}_n$. We say the family has good linear confinement if each $\mathcal{M}_n$ is (t, f)-confined where:*

1. *t grows with n such that $t \geq an^b$ for some positive constants a, b. That is, $t \in \Omega\left(n^b\right)$ with $b > 0$;*

2. *and $f(x)$ is some linear function that is monotonically increasing with x and independent of n.*

When a code has good linear confinement, the code is single-shot against both adversarial and stochastic noise [101]. The intuition is that the linear confinement property guarantees that the low-energy state space of the code Hamiltonian is partitioned into well-separated clusters [221].

Before we proceed to show that our concatenated code has good linear confinement, we first state the confinement property of good expander codes.

Lemma 23 ([79, Linear Confinement of Expander Codes]). *For any quantum expander code with distance d, an arbitrary error e with reduced weight $||e|| < d$ has a syndrome with weight bounded from below as $|\sigma(e)| \geq \frac{1}{3}||e||$.*

Now, we proceed to show that our concatenated code has good linear confinement.

Lemma 24 (Linear Confinement of Concatenated Code). *Let the chain complex corresponding*

to our concatenated code be

$$\tilde{C}_{-1} \xrightarrow[\tilde{\delta}_{-1}=\tilde{H}_X^\top]{} \tilde{C}_0 \xrightarrow[\tilde{\delta}_0=\tilde{H}_Z]{} \tilde{C}_1 \tag{B.10}$$

where d is the distance of our concatenated code. The stabilizer matrices $\tilde{H}_X$ and $\tilde{H}_Z$ are $\left(\frac{d}{2}, 6x\right)$-confined.

Proof. Suppose we have a Pauli error e that has reduced weight $||e|| < d/2$. To make things concrete, let e' be any weight-reduced Pauli error such that $|e'| = ||e||$ and $\sigma(e) = \sigma(e')$.

Next, let us partition the length-$2n$ sequence e' for our $[[2n, k, d]]$ concatenated code into sets of 4 qubits that belong to individual Iceberg code patches, i.e.

$$P = \{\{\mathcal{P}_{4i+1}, \mathcal{P}_{4i+2}, \mathcal{P}_{4i+3}, \mathcal{P}_{4i+4}\}\}_{i=0}^{\frac{n}{2}-1}, \tag{B.11}$$

where $\mathcal{P}$ is some Pauli operator in the set $\{I, X, Y, Z\}$. An arbitrary element of P can only have non-trivial support on 0, 1, or 2 physical qubits else we can apply the Iceberg code stabilizer generators to further reduce the weight of the error. Let $\left\{e'_{1,j}\right\}_j \subseteq P$ be the set of $\{\mathcal{P}_{4i+1}, \mathcal{P}_{4i+2}, \mathcal{P}_{4i+3}, \mathcal{P}_{4i+4}\}$ that contains exactly 1 physical qubit Pauli error. Similarly, let $\left\{e'_{2,k}\right\}_k \subseteq P$ be the set of $\{\mathcal{P}_{4i+1}, \mathcal{P}_{4i+2}, \mathcal{P}_{4i+3}, \mathcal{P}_{4i+4}\}$ that contains exactly 2 physical qubit Pauli errors. In addition, let $\sigma_{new}(e')$ and $\sigma_{copied}(e')$ correspond to the part of the syndrome that corresponds to the new and copied generators respectively such that

$$\sigma(e') = \sigma_{new}(e') \oplus \sigma_{copied}(e'). \tag{B.12}$$

Construct a length-n Pauli sequence $e'_{logical}$ from e' by converting each element in $\left\{e'_{2,k}\right\}_k$ into the corresponding logical Paulis of the Iceberg code patch. In addition, convert each element

of $\left\{e'_{1,j}\right\}_j$ into some (potentially trivial) logical Pauli operator of the Iceberg code patch depending on whether the element anti-commutes with $\overline{X}_1, \overline{X}_2, \overline{Z}_1, \overline{Z}_2$. For example, suppose $e'_{1,j} = X_1 I_2 I_3 I_4$ and $e'_{1,j'} = I_1 Z_2 I_3 I_4$, we then convert the former into $\overline{I}_1 \overline{I}_2$ since it commutes with all the logical Paulis of the Iceberg code patch and the latter into $\overline{Z}_1 \overline{I}_2$ since it only anti-commutes with $\overline{X}_1$. Lastly, let ς denote the syndrome operator of the $[[n, k, d]]$ HGP code that was used to construct our concatenated code.

Then, we have

$$
\begin{aligned}
|\sigma(e)| &= |\sigma(e')| && \text{(B.13)}\\
&= |\sigma_{new}\left(e'\right)| + |\sigma_{copied}(e')| && \text{(B.14)}\\
&= \left|\left\{e'_{1,j}\right\}_j\right| + \left|\varsigma\left(e'_{logical}\right)\right| && \text{(B.15)}\\
&\geq \left|\left\{e'_{1,j}\right\}_j\right| + \frac{1}{3}\left|\left|e'_{logical}\right|\right| && \text{(B.16)}\\
&\geq \left|\left\{e'_{1,j}\right\}_j\right| + \frac{1}{3}\left|\left\{e'_{2,k}\right\}_k\right| && \text{(B.17)}\\
&= \left|\left\{e'_{1,j}\right\}_j\right| + \frac{1}{6}\left(2\left|\left\{e'_{2,k}\right\}_k\right|\right) && \text{(B.18)}\\
&\geq \frac{1}{6}\left(\left|\left\{e'_{1,j}\right\}_j\right| + 2\left|\left\{e'_{2,k}\right\}_k\right|\right) && \text{(B.19)}\\
&= \frac{1}{6}|e'| && \text{(B.20)}\\
&= \frac{1}{6}||e||. && \text{(B.21)}
\end{aligned}
$$

We note that the third equality comes from the fact that each element in $\left\{e'_{1,j}\right\}_j$ triggers the syndrome of a single new generator and the construction of our concatenation scheme. In addition, we have $\left|e'_{logical}\right| \leq 2|e'|$ since a single physical Pauli error can be converted into at most 2 logical errors when we construct $e'_{logical}$. Thus, we can use Lemma 23 to obtain the latter term in the right

hand side of the first inequality because $|e'| < d/2$ implies that $\left|e'_{logical}\right| < d$. The second inequality comes from discarding the logical errors that were generated from $\left\{e'_{1,j}\right\}_j$. Since (B.21) holds for $||e|| = |e'| < d/2$, we have shown that the stabilizer matrices of our concatenated code have $\left(\frac{d}{2}, 6x\right)$-confinement. □

Since Lemma 20 guarantees us that the thickened concatenated code has good soundness and Lemma 24 guarantees that our concatenated code has the single-shot property, we are able to use the single-shot logical state preparation scheme in Ref. [81]. We now state the single-shot logical state preparation scheme with respect to our concatenated code for the sake of concreteness.

Suppose we have a classical repetition code with code parameters $[d, 1, d]$ where d is the distance of our $\left[\left[\tilde{n}, \tilde{k}, d\right]\right]$ concatenated HGP code. Let the $\breve{Q}$ be the $\left[\left[\breve{n}, \breve{k}, d\right]\right]$ code that emerges from thickening our concatenated HGP code in the Z-basis with the classical repetition code. In the first stage of the single-shot $|\overline{0}\rangle^{\otimes \tilde{k}}$ state preparation protocol for our concatenated HGP code, we perform the following:

1. Initialize all physical qubits in $|0\rangle^{\otimes \breve{n}}$.

2. Measure all X-type check operators to obtain an initial X-syndrome before using the X-metachecks to obtain an X-metasyndrome.

3. Decode the X-metasyndrome and repair the initial X-syndrome.

4. Decode the repaired syndrome using $\tilde{H}_X$ of the concatenated HGP code to return to $|\overline{0}\rangle^{\otimes \breve{k}}$, which is the logical all-0s state for the thickened $\left[\left[\breve{n}, \breve{k}, \breve{d}\right]\right]$ code, up to some small residual Z error.

The second stage of the protocol proceeds as follows:

1. Measure Z on all physical qubits of the thickened code except on one of the boundary concatenated HGP codes and reconstruct a bulk Z-syndrome.

2. Input the above bulk Z-measurement outcomes, Z-syndrome, and a suitable single-shot decoder for our concatenated code before iteratively decoding, inferring, and updating an X-correction vector from each layer of concatenated HGP code.

3. Apply the final X-correction vector on the boundary concatenated HGP code to obtain a single layer of concatenated HGP code that is initialized to be in the $|\overline{0}\rangle^{\otimes \tilde{k}}$ state.

By changing the basis for thickening and replacing X/Z, we can obtain the single-shot state preparation protocol for initializing the $|\overline{+}\rangle^{\otimes \tilde{k}}$ state for the concatenated HGP code.

B.2.0.2 Z/X basis measurements

To perform logical Z/X basis measurements, we simply have to perform transversal measurements on all n physical qubits of our concatenated HGP code in the Z/X basis. In other words, just as it holds for any other CSS codes, $M_Z^{\otimes n}/M_X^{\otimes n}$ gives us the logical measurement $\overline{M}_Z^{\otimes k}/\overline{M}_X^{\otimes k}$ for our concatenated HGP code.

B.2.0.3 Concatenated Grid Pauli Product Measurements (CGPPMs)

In the original grid Pauli product measurement (GPPM) scheme in Ref. [194], corresponding physical qubits in the main HGP code and the ancilla HGP code are entangled via physical CNOT gates. Our concatenated grid Pauli product measurement (CGPPM) scheme works effectively the same as these GPPMs except that we entangle the logical qubit of some Iceberg code

with the physical qubit in the ancilla HGP code. To perform such a CNOT gate, we use the circuit shown in Fig. B.2 which performs a logical CNOT that is controlled on the first logical qubit of some Iceberg code block and acts on qubit a of a HGP ancilla code block. In this paper, we refer to these CNOTs as Logical-Physical (LP) CNOTs. This circuit can be adapted to the case where we control on a different logical qubit or the case where we decide to control on the physical qubit of the HGP code instead. This particular CGPPM scheme allows us to reduce the overhead of the GPPM measurement by not having to concatenate the ancilla HGP codes with the Iceberg codes. In addition, it also allows us to sidestep the challenging task of assigning Iceberg code logical qubits to the punctured or/and augmented ancilla HGP code blocks. More details regarding how the GPPMs can be customizable to the specific Paulis of interest can be found in Algorithm 2 of Ref. [194].

In addition, our CGPPM is compatible with the selective inter-block teleportation scheme described in Algorithm 3 in Ref. [194] that allows the teleportation of any subset of the logical qubits of a square HGP code to the corresponding logical qubits of another identical square HGP code. By performing the LP CNOTs described above, we can entangle the desired subset of the logical qubits of the original concatenated HGP code block to an ancilla HGP code block before entangling it to the final concatenated HGP code block with another set of LP CNOTs. We then proceed to perform CGPPM on the ancilla HGP code block and the original concatenated HGP code block to perform the desired selective teleportation. The details can be easily derived from Algorithm 3 and Fig. 11(a) in Ref. [194].

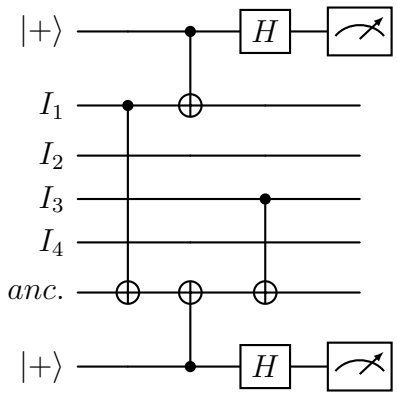

Figure B.2: By performing physical CNOTs that are controlled on the first and third qubit of the $[[4, 2, 2]]$ code and then acting on the qubit $anc.$ of the ancilla HGP code, we perform a logical CNOT that is controlled on the first logical qubit of the $[[4, 2, 2]]$ code and acting on the physical qubit a of the ancilla HGP code. The $|+\rangle$ states are included as flag qubits to catch the Z error that may appear before the two physical CNOT gates which would propagate to become $Z_1 Z_3 = \overline{Z}_1$ in the [[4,2,2]] code.

B.2.0.4 Inter-block CNOTs

For the inter-block CNOTs across all logical qubits of two identical concatenated HGP code, we perform the standard transversal physical CNOTs across all pairs of physical qubits to achieve global logical CNOTs.

Unlike the inter-block CNOTs scheme designed for square HGP codes in Ref. [194], we are unable to do a set of inter-block CNOTs on an arbitrary subset of the logical qubits in the concatenated HGP code. This is because doing that would require us to puncture or/and augment the classical codes that are used to construct the HGP code. The resulting HGP code is not amenable to our current concatenation scheme because the missing columns or rows in the resulting HGP code, in general, remove the symmetry that we use to assign the logical qubits of our $[[4, 2, 2]]$ code. While we might still be able to find a different assignment assuming that there are still an even number of physical qubits left in the resulting HGP code, we cannot do tranversal CNOTs

on a subset of logical qubits because it will, in general, require us to entangle only one logical qubit of some $[[4, 2, 2]]$ code with another logical qubit of some other $[[4, 2, 2]]$ code. It is unclear whether it is possible for us to do targeted CNOTs on the Iceberg code when the logical qubits of the Iceberg codes are in arbitrary quantum states, especially if the ancilla concatenated HGP code blocks are punctured. Thus, we construct our inter-block CNOT gadget to perform a global logical CNOT for our concatenation scheme for the HGP code with $[[4, 2, 2]]$ Iceberg code. For the rest of the paper, we work with a global inter-block CNOT gadget.

However, we note that it is possible to modify our concatenation scheme so that we utilize only one logical qubit per Iceberg code for every physical qubit of the HGP code. In other words, we have a concatenated HGP code that has a subsystem structure. In this case, the concatenated HGP code can inherit the targeted logical CNOT capability from the HGP code since its structure closely resembles that of the original square HGP code. It is also possible to formulate an alternative inter-block CNOT scheme. Consider the more restricted setting where one of the blocks is purely an ancilla block where all the logical qubits are either $|\overline{+}\rangle$ or $|\overline{0}\rangle$ states. To be explicit, suppose we have a punctured concatenated HGP ancilla block that is supposed to be the target block for a transversal logical CNOT. To construct such a punctured concatenated HGP ancilla block, we first layout the punctured HGP code in the usual geometric layout. In this picture, there will be some empty rows and columns due to the punctures. When we concatenate it with the iceberg codes, we perform the same diagonal symmetric assignment of iceberg logical qubits to the physical qubits of the punctured HGP code. For those logical qubits of the iceberg code blocks that are mapped to the punctured qubits of the HGP ancilla block, we can initialize them to be in the $|\overline{+}\rangle$ or $|\overline{0}\rangle$ state so the global transversal CNOTs do not entangle the logical qubits that are supposed to be punctured. This can allow us to do inter-block CNOTs on our concatenated HGP

codes while respecting the presence of the punctures in the base HGP ancilla blocks, allowing us to only entangle the logical qubits that are not punctured.

B.2.0.5 Intra-Block CNOTs

To implement intra-block CNOT gates between logical qubits in a single $[[4, 2, 2]]$-concatenated HGP code block, we can perform the circuit in Fig. B.3 with our CGPPM gadgets. Using similar ideas as in Ref. [194], we can perform all intra-block CNOT gates across logical qubits that are either aligned in the same column or row in $O(\sqrt{k})$ logical cycles. The intra-block CNOTs that acts on logical qubits that are not in the same column or row can be grouped into clusters where each cluster contains CNOTs that act on pairs of qubits that all reside in the same column and row. For example, if a CNOT acts on two qubits that are in row 5 and column 10 respectively and another CNOT acts on two other qubits that are also in row 5 and column 10 respectively, these two CNOTs will be in the same cluster. Each of these clusters can contain at most $\sqrt{k}$ CNOTs and each of these CNOTs will take $O(\sqrt{k})$ logical cycles to implement. This implies that each cluster takes $O(k)$ logical cycles to implement. Because these clusters can be implemented in parallel by carefully teleporting the right columns and rows to ancilla code blocks as argued in Appendix B of Ref. [194], measuring these CNOTs will take $O(k)$ logical cycles. When the logical translation gadget is available, the number of logical cycles required to perform $O(k)$ intra-block CNOTs can be reduced to $O\left(k^{3/4}\right)$ as described in Appendix B of Ref. [194].

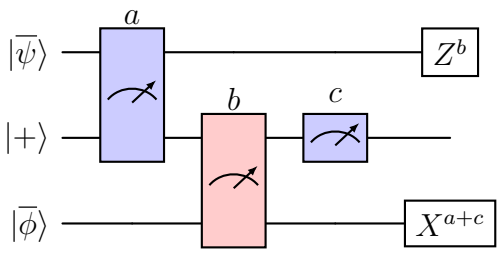

Figure B.3: Logical quantum circuit for performing logical CNOT gate between intra-block logical qubits, $|\overline{\psi}\rangle$, $|\overline{\phi}\rangle$. The gate is controlled on $|\overline{\psi}\rangle$ and targets $|\overline{\phi}\rangle$. The middle qubit line belongs to an ancilla qubit that is initialized in $|\overline{+}\rangle$. The blue measurements with measurement outputs a and c are either ZZ or single Z measurements. The red measurement with measurement output b is an XX measurement. This figure is adapted from Fig. 11(c) in Ref. [194].

B.2.0.6 H-SWAP

For a square HGP code, the logical Hadamard on all logical qubits is performed by first applying the physical Hadamard on all physical qubits before swapping the twin physical qubits that lie on opposite sides of the principal diagonal. In addition, this causes the logical qubits that lie on opposites of the principal diagonal in the logical grid to be swapped. More details about the H-SWAP gate for square HGP codes can be found in Ref. [84]. It falls under the larger fold-transversal framework of Ref. [222]. For our $[[4, 2, 2]]$-concatenated HGP code, H-SWAP is achieved by performing the following:

1. Perform physical Hadamard on all physical qubits of the concatenated code.

2. For each $[[4, 2, 2]]$ code whose logical qubits are assigned to a pair of physical qubits that are adjacent to each other on the principal diagonal of the HGP code, perform $\mathrm{SWAP}_{3,4}$.

Recall that $H_1H_2H_3H_4$ implements $\overline{H}_1\overline{H}_2\overline{\mathrm{SWAP}}_{1,2}$ on a single $[[4, 2, 2]]$ Iceberg code as shown in Table B.1. Because we have assigned the two logical qubits of every Iceberg code to the two twin qubits on opposite sides of the principal diagonal of the HGP code, step 1 would

be equivalent to Hadamards and SWAPs on the twin qubits of the HGP code. Because we do not want the SWAPs for the qubits on the principal diagonal, we perform step 2 to reverse these swaps for the Iceberg codes that are on the principal diagonal of the HGP code. Because H-SWAP also performs unwanted swaps on the twin logical qubits of the $[[4, 2, 2]]$-concatenated HGP code, we have to swap back these pairs of twin logical qubits in order to implement global logical Hadamard on the logical qubits of our concatenated code. To reverse these swaps, we can first identify pairs of diagonal lines L and L' that are equidistant from the principal diagonal such that the two diagonal lines L and L' correspond to pairs of twin logical qubits. Next, we teleport L and L' to two different ancilla $[[4, 2, 2]]$-concatenated HGP blocks that are initialized using Algorithm 4 in Ref. [194] such that the appropriate logical qubits in the ancilla blocks are set to $|\overline{+}\rangle$ and the other logical qubits are set to $|\overline{0}\rangle$. Next, for each of the two ancilla blocks, we perform a logical SWAP operation using three intra-block CNOTs with the cGPPM gadget so that the logical qubits corresponding to $L\,(L')$ are now swapped to the position of $L'\,(L)$ in the ancilla block. Lastly, teleport the two diagonal lines of logical qubits back into the original code block. Since there are $O(\sqrt{k})$ pairs of diagonal lines and the steps stated above take $O(\sqrt{k})$ logical cycles, reversing the unwanted swaps takes $O(k)$ logical cycles. While the analysis above can be tightened further, showing that implementing global logical Hadamard takes $O(k)$ logical cycles is sufficient since it is not the bottle neck for the time cost for logical computation.

To perform targeted Hadamard on a subset of logical qubits in a concatenated HGP code block, we simply have to teleport the subset of logical qubits to an ancilla $[[4, 2, 2]]$-concatenated HGP code block. We then perform global transversal Hadamard as described above on all the logical qubits in the ancilla code block before teleporting the desired subset back to the original concatenated HGP code block. An alternative method to implement targeted Hadamard on $O(k)$

logical qubits would involve iteratively teleporting columns of the logical qubits of interest to an ancilla $[[4, 2, 2]]$-concatenated HGP code block before performing logical swaps to move them to the principal diagonal of the ancilla block. Subsequently, we perform the H-SWAP logical gadget on the ancilla block. Because the logical qubits of our interest are on the principal diagonal, there is no need to reverse the unwanted logical swaps because the logical qubits on the principal diagonal are not impacted by the unwanted logical swaps. Lastly, we swap the logical qubits out of the principal diagonal before teleporting them back to the original code block. This process is repeated at most $\sqrt{k}$ times because the logical qubits of interest can have support on at most $\sqrt{k}$ columns. Since each iteration takes $O(\sqrt{k})$ logical cycles, it takes at most $O(k)$ logical cycles. Therefore, to perform targeted Hadamard on $O(k)$ logical qubits, we require $O(k)$ logical cycles. We note that performing targeted Hadamard on $O(k)$ logical qubits when given access to the logical translation gadget will take $O(\sqrt{k}\log k)$ logical cycles. Readers can find the details for the implementation in Ref. [194]. We note that the H-SWAP gadget for the $[[4, 2, 2]]$-concatenated HGP code requires less non-locality because the physical non-local SWAP gates in the regular HGP code are replaced by the logical swaps for the $[[4, 2, 2]]$ Iceberg codes that can be implemented locally. However, recall that we have to reverse the unwanted logical swaps implemented by the H-SWAP logical gadget. If we do not have the logical translation gadget, the logical swap implemented by the intra-block CNOTs may consume more non-local entangling gates.

B.2.0.7 CZ-S

Using the same fold-transversal framework discussed in Ref. [222], Quintavalle *et al.* introduced a fold-transversal CZ-S logical gate for square HGP codes [84]. The gate applies a S

gate on left diagonal logical qubits, $S^\dagger$ on right diagonal logical qubits, and CZ between logical twin qubits. The gate is performed by:

1. Apply CZ gates on pairs of twin qubits

2. Apply physical S gates to the physical qubits which lie on the principal diagonal of the L sector and $S^\dagger$ gates to the physical qubits which lie on the principal diagonal of the R sector.

From Table B.1, observe that applying $S_1^\dagger S_2^\dagger S_3 S_4$ on each Iceberg code corresponding to twin qubits of the HGP code implements the desired logical CZ gate between those twin logical qubits. The $\overline{S}$ and $\overline{S}^\dagger$ gates as listed in Table B.2 are not transversal because of the physical CZ gate and thus are not inherently fault-tolerant. By introducing two additional flag qubits, we can use the circuit in Fig. 2(c) in Ref. [223] to implement the physical CZ gate fault-tolerantly. The explicit construction of the circuit for implementing $\overline{S}_1 = S_1 S_3 CZ_{1,3}$ is shown in Fig. B.4. Note that $\overline{S}_i^\dagger$ can be obtained by simply applying $\overline{Z}_i$ after the circuit for $\overline{S}_i$. Hence we have fault-tolerant implementations of the $\overline{S}, \overline{S}^\dagger$, and $\overline{CZ}$ gates for the $[[4, 2, 2]]$ code, with which we can perform the CZ-S logical gadget on the $[[4, 2, 2]]$-concatenated HGP code.

To perform targeted S gate on any subset of logical qubits in the concatenated HGP code block, we have to adopt a slightly different procedure. Because the CZ-S gadget only allows us to generate $|i\rangle$ states on the principal diagonal, we have to teleport them to the non-diagonal logical qubits in order to be able to apply $\overline{S}$ gates on any subset of logical qubits in the concatenated HGP code block. To perform the teleportation, we can use the circuit shown in Fig. B.5. The explicit procedure for implementing logical S gates on any subset of logical qubits would involve the following:

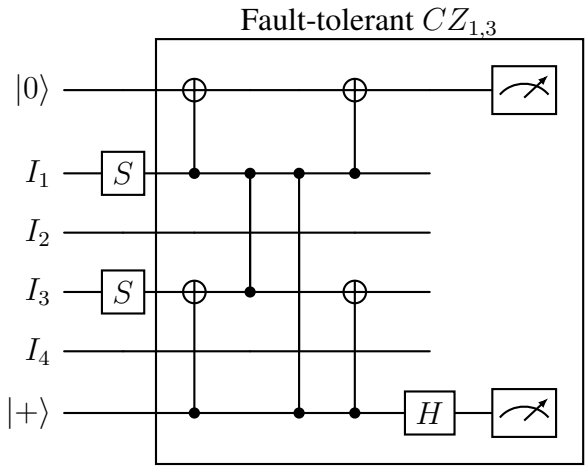

Figure B.4: Fault-tolerant circuit for implementing $\overline{S}_1 = S_1 S_3 CZ_{1,3}$ on the $[[4, 2, 2]]$ code.

1. Initialize the concatenated HGP code to be in the $|\overline{0}\rangle^{\otimes k}$ state. We can teleport the logical qubits to another concatenated HGP code block before performing the initalization if necessary.

2. Prepare $\sqrt{k}\ |\bar{i}\rangle$ states on the principal diagonal of the concatenated HGP code using Algorithm 4 in Ref. [194] and our CZ-S logical gadget without performing logical translation.

3. Teleport the original off-diagonal logical qubits back to the concatenated HGP code block.

4. Use the circuit in Fig. B.5 to teleport the $\sqrt{k}$ S gates on the principal diagonal onto the off-diagonal logical qubits that lie in the desired subset.

5. If there are excess S gates on the diagonal, we can use CGPPM to reset those logical qubits to $|\overline{+}\rangle$ so that we can teleport the original diagonal logical qubits back. If there are not enough S gates on the diagonal, we can repeat Steps 1 - 5 by carefully teleporting the logical qubits with the completed S gates away.

Suppose we are interested in applying $O(k)$ logical S gates. Steps 1, 3, 4, and 5 take $O(\sqrt{k})$ logical cycles while step 2 takes $O(\log k)$ logical cycles. We repeat steps 1 - 5 at most

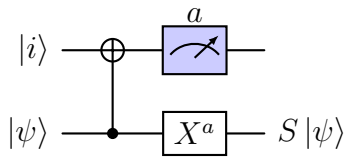

Figure B.5: Logical quantum circuit for teleporting the logical S gate to the logical qubits that do not lie on the principal diagonal. The first qubit line corresponds to some logical qubit in the $|i\rangle := S|+\rangle$ state that lies on the principal diagonal and the second qubit line corresponds to some logical qubit that is in some arbitrary $|\psi\rangle$ state and lies outside of the principal diagonal. The blue measurement with measurement output a is a single Z measurement.

$O(\sqrt{k})$ times. Thus, the entire procedure for applying $O(k)$ S gates would take $O(k)$ logical cycles. We note that our procedure is relatively slower than the one in Ref. [194] for square HGP codes because we are not able to prepare $O(k)$ $|\bar{i}\rangle$ states in parallel without the logical translation gadget. In the protocol stated in Ref. [194] where they assume that the HGP code is constructed from quasi-cyclic classical codes, they are able to implement logical S gates on $O(k)$ logical qubits in $O(\sqrt{k}\log k)$ logical cycles.

B.2.0.8 Logical translation

Some quantum error correcting codes are able to implement logical gates simply by permuting their physical qubits. One such example of an automorphism gate [85, 222] is the $\overline{SWAP}_{1,2}$ gate for the $[[4,2,2]]$ code. Using a specific subclass of quasi-cyclic base classical LDPC codes, Ref. [194] showed that the resulting HGP codes have automorphism gates which implement a translation of the logical qubits. Because it is not known as to whether there exists asymptotically good classical LDPC codes that satisfy the quasi-cyclic and one-generator-systematic circulant (OGSC) properties, we only included the construction of the logical translation gadget at the end and have discussed how we can implement the other gadgets without access to the logical translation gadget.

For our [[4, 2, 2]]-concatenated HGP code, permutation of the Iceberg logical qubits is a non-trivial task since we need to preserve the diagonal symmetry post-automorphism. To be explicit, the Iceberg code blocks are initially assigned such that the two logical qubits are mapped to adjacent diagonal qubits or twin qubits of the HGP code. Permuting the 'physical' qubits of the [[4, 2,2]]-concatenated HGP code would, in general, destroy this symmetry and prevent us from implementing logical gadgets post-permutation. Thus, we need to make sure adjacent diagonal qubits and twin qubits are part of the same Iceberg block after permutation.

To accomplish this, we use the fact that we are able to teleport a single logical qubit out of the $[[4, 2, 2]]$ code, following Ref. [219]. Suppose we are interested in teleporting away the second logical qubit of the Iceberg code that is in the $|\overline{\psi\phi}\rangle$ state. To do so, we:

1. Prepare an ancilla Iceberg code block in the $|\overline{0+}\rangle$ state following Sec. B.1.0.2.

2. Perform a transversal CNOT targeting $|\overline{\psi\phi}\rangle$ and controlled on the ancilla code block.

3. Perform a non-destructive logical Z measurement of the second logical qubit of $|\overline{\psi\phi}\rangle$.

The resulting state of the ancilla Iceberg code block is then $|\overline{0\phi}\rangle$. We can also perform an analogous procedure with $|\overline{+0}\rangle$ as the ancilla block to teleport the first logical qubit of $|\overline{\psi\phi}\rangle$. Let us now discuss how we can produce $|\overline{\psi\phi}\rangle$ from $|\overline{\psi 0}\rangle$ and $|\overline{0\phi}\rangle$ using teleportation. We can perform the following:

$$|\overline{\psi 0}\rangle \xrightarrow{H^{\otimes 4}} |\overline{+}\rangle \otimes \overline{H}\,|\overline{\psi}\rangle \xrightarrow{Tel.\,|\overline{0+}\rangle} |\overline{0}\rangle \otimes \overline{H}\,|\overline{\psi}\rangle \xrightarrow{H^{\otimes 4}} |\overline{\psi+}\rangle \tag{B.22}$$

where $Tel.\ |\overline{0+}\rangle$ teleports $\overline{H}\,|\overline{\psi}\rangle$ to an ancilla Iceberg code block. We can perform the analogous operations to $|\overline{0\phi}\rangle$ to obtain $|\overline{+\phi}\rangle$. Now, we perform a transversal logical CNOT that is controlled on $|\overline{+\phi}\rangle$ and targeting $|\overline{\psi+}\rangle$ before performing a non-destructive logical Z measurement on the

first logical qubit of $|\overline{\psi+}\rangle$. This would implement the following transformation: $|\overline{+\phi}\rangle \rightarrow |\overline{\psi\phi}\rangle$.

Now, we are ready to provide the explicit description of the logical translation gadget. For each Iceberg code block B in the concatenated HGP code, we prepare two ancilla Iceberg code blocks in $|\overline{0+}\rangle$ and $|\overline{+0}\rangle$ before teleporting the two logical qubits of B into the two different ancilla code blocks. We then assign the ancilla code blocks according to the desired automorphism gate before teleporting the new adjacent diagonal and twin qubits into the same Iceberg block. For each Iceberg code block, we use at most three additional Iceberg code blocks to perform the permutation. Because this process can be done in parallel for each Iceberg code block, the logical translation gadget requires $O(1)$ logical cycles.

B.2.0.9 Combining the Logical Gadgets

We restate the main theorem for the logical computation protocol for our $[[4, 2, 2]]$-concatenated HGP code for the readers' convenience.

Theorem 25 (Clifford Gates for Concatenated HGP Code)**.** *A single layer of an ideal Clifford circuit on k logical qubits with $\Theta(k)$ gates consisting of Hadamard, S, and CNOT gates can be simulated on a square HGP code concatenated with the $[[4, 2, 2]]$ Iceberg code blocks with either one of the following space-time costs:*

1. *$O(k)$ space and $O\left(k^{3/2}\right)$ time*
2. *$O\left(k^{3/2}\right)$ space and $O(k)$ time.*

If the square HGP code was constructed from OGSC quasi-cyclic base codes, the concatenated code can simulate the layer with either one of the following space-time costs:

1. $O(k)$ *space and* $O\left(k^{5/4}\right)$ *time*

2. $O\left(k^{3/2}\right)$ *space and* $O\left(k^{3/4}\right)$ *time.*

Proof. The proof follows directly from the construction of the logical gadgets described in the appendix. The space-time overhead for the concatenated HGP code constructed with OGSC quasi-cyclic codes can be easily shown to be true using arguments from App. B of Ref. [194]. We note that the latter space-time cost for each of the construction in the theorem statement comes from the fact that we can perform all of our logical gadgets in $O(1)$ logical cycles at the expense of an increased space cost due to the single-shot state preparation. □

Similar to Ref. [194], we emphasize that the number of logical cycles is a rather loose upper bound that comes from a constructive compilation and the actual time costs of simulating the layer of $O(k)$ Clifford gates is likely to be significantly lower. An efficient compilation of the Clifford gates would almost definitely allow us to improve the parallelism and drastically reduce the circuit depth.

B.3 Universal Computation

Using the magic state distillation and consumption protocols constructed in Section V-C-2 of Ref. [194], we are able to use the logical gadgets constructed above to perform the same "8-to-CCZ" magic state distillation protocol and implement the non-Clifford gates in parallel for k logical qubits. If we were to use a general square HGP code with good expansion, we can either perform the above with $O(k)$ space and $O\left(k^{3/2}\right)$ time or $O\left(k^{3/2}\right)$ space and $O\left(k\right)$ time. In the event where we use a square HGP constructed from OGSC quasi-cyclic base codes, our protocol allows us to perform the above with $O(k)$ space and $O\left(k \log k\right)$ time or $O\left(k^{3/2}\right)$

space and $O\left(\sqrt{k}\log k\right)$ time. Again, the less efficient space-time costs for the non-quasi-cyclic construction of the HGP codes come from the fact that we are unable to use the logical translation gadget which would reduce the number of logical cycles for the CZ-S logical gadget from $O(k)$ to $O(\sqrt{k}\log k)$.

Bibliography

[1] Richard P. Feynman. Simulating physics with computers. *International Journal of Theoretical Physics*, 21(6):467–488, 1982. URL https://doi.org/10.1007/BF02650179.

[2] Peter W. Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM Journal on Computing*, 26(5):1484–1509, 1997. URL http://dx.doi.org/10.1137/S0097539795293172.

[3] S. A. Moses, C. H. Baldwin, M. S. Allman, R. Ancona, L. Ascarrunz, C. Barnes, J. Bartolotta, B. Bjork, P. Blanchard, M. Bohn, et al. A race-track trapped-ion quantum processor. *Phys. Rev. X*, 13:041052, 2023. URL https://link.aps.org/doi/10.1103/PhysRevX.13.041052.

[4] Charlie Baldwin. Quantinuum hardware specifications, 2022. URL https://github.com/CQCL/quantinuum-hardware-specifications.

[5] IonQ Aria: Practical performance, 2024. URL https://ionq.com/resources/ionq-aria-practical-performance.

[6] IBM Quantum, 2021. URL https://quantum.ibm.com/.

[7] Craig Gidney and Martin Ekerå. How to factor 2048 bit rsa integers in 8 hours using 20 million noisy qubits. *Quantum*, 5:433, 2021. URL http://dx.doi.org/10.22331/q-2021-04-15-433.

[8] John Preskill. Quantum computing in the nisq era and beyond. *Quantum*, 2:79, 2018. URL http://dx.doi.org/10.22331/q-2018-08-06-79.

[9] M. Cerezo, Andrew Arrasmith, Ryan Babbush, Simon C. Benjamin, Suguru Endo, Keisuke Fujii, Jarrod R. McClean, Kosuke Mitarai, Xiao Yuan, Lukasz Cincio, and Patrick J. Coles. Variational quantum algorithms. *Nature Reviews Physics*, 3(9):625–644, 2021. URL http://dx.doi.org/10.1038/s42254-021-00348-9.

[10] Ehud Altman, Kenneth R. Brown, Giuseppe Carleo, Lincoln D. Carr, Eugene Demler, Cheng Chin, Brian DeMarco, Sophia E. Economou, Mark A. Eriksson, Kai-Mei C. Fu, Markus Greiner, et al. Quantum simulators: Architectures and opportunities. *PRX Quantum*, 2:017003, 2021. URL https://link.aps.org/doi/10.1103/PRXQuantum.2.017003.

[11] Andrew J. Daley, Immanuel Bloch, Christian Kokail, Stuart Flannigan, Natalie Pearson, Matthias Troyer, and Peter Zoller. Practical quantum advantage in quantum simulation. *Nature*, 607(7920):667–676, 2022. URL https://doi.org/10.1038/s41586-022-04940-6.

[12] Matthew DeCross, Reza Haghshenas, Minzhao Liu, Enrico Rinaldi, Johnnie Gray, Yuri Alexeev, Charles H. Baldwin, John P. Bartolotta, Matthew Bohn, Eli Chertkov, et al. The computational power of random quantum circuits in arbitrary geometries. *arXiv preprint arXiv:2406.02501*, 2024. URL https://doi.org/10.48550/arXiv.2406.02501.

[13] A. Morvan, B. Villalonga, X. Mi, S. Mandrà, A. Bengtsson, P. V. Klimov, Z. Chen, S. Hong, C. Erickson, I. K. Drozdov, et al. Phase transitions in random circuit sampling. *Nature*, 634(8033):328–333, 2024. URL https://doi.org/10.1038/s41586-024-07998-6.

[14] Peter W. Shor. Scheme for reducing decoherence in quantum computer memory. *Phys. Rev. A*, 52:R2493–R2496, 1995. URL https://link.aps.org/doi/10.1103/PhysRevA.52.R2493.

[15] E. Knill. Fault-tolerant postselected quantum computation: Schemes. *arXiv preprint arXiv:quant-ph/0402171*, 2004. URL https://doi.org/10.48550/arXiv.quant-ph/0402171.

[16] Sergey Bravyi and Alexei Kitaev. Universal quantum computation with ideal clifford gates and noisy ancillas. *Physical Review A*, 71(2), 2005. URL http://dx.doi.org/10.1103/PhysRevA.71.022316.

[17] D. Aharonov and M. Ben-Or. Fault-tolerant quantum computation with constant error. In *Proceedings of the 29th Annual ACM Symposium on Theory of Computing*, page 176–188. Association for Computing Machinery, 1997. URL https://doi.org/10.1137/S0097539799359385.

[18] A Yu Kitaev. Quantum computations: algorithms and error correction. *Russian Mathematical Surveys*, 52(6):1191, 1997. URL https://doi.org/10.1070/RM1997v052n06ABEH002155.

[19] Emanuel Knill, Raymond Laflamme, and Wojciech H. Zurek. Resilient quantum computation: Error models and thresholds. *Proc. R. Soc. Lond. A.*, 454(1969):365–384, 1998. URL https://doi.org/10.1098/rspa.1998.0166.

[20] Nikolas P. Breuckmann and Jens N. Eberhardt. Balanced product quantum codes. *IEEE Transactions on Information Theory*, 67(10):6653–6674, 2021. URL http://dx.doi.org/10.1109/TIT.2021.3097347.

[21] Pavel Panteleev and Gleb Kalachev. Asymptotically good quantum and locally testable classical LDPC codes. In *Proceedings of the 54th Annual ACM SIGACT Symposium on Theory of Computing*, pages 375–388, 2022. URL https://doi.org/10.1145/3519935.3520017.

[22] A. Leverrier and G. Zemor. Quantum tanner codes. In *2022 IEEE 63rd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 872–883, 2022. URL https://doi.ieeecomputersociety.org/10.1109/FOCS54457.2022.00117.

[23] Irit Dinur, Min-Hsiu Hsieh, Ting-Chun Lin, and Thomas Vidick. Good quantum LDPC codes with linear time decoders. In *Proceedings of the 55th Annual ACM Symposium on Theory of Computing*, pages 905–918, 2023. URL https://doi.org/10.1145/3564246.3585101.

[24] Daniel Gottesman. Fault-tolerant quantum computation with constant overhead. *Quantum Info. Comput.*, 14(15-16):1338–1372, 2014. URL https://doi.org/10.26421/QIC14.15-16-5.

[25] Shin Ho Choe and Robert Koenig. How to fault-tolerantly realize any quantum circuit with local operations. *arXiv preprint arXiv:2402.13863*, 2024. URL https://doi.org/10.48550/arXiv.2402.13863.

[26] Shiro Tamiya, Masato Koashi, and Hayata Yamasaki. Polylog-time- and constant-space-overhead fault-tolerant quantum computation with quantum low-density parity-check codes. *arXiv preprint arXiv:2411.03683*, 2024. URL https://doi.org/10.48550/arXiv.2411.03683.

[27] Hayata Yamasaki and Masato Koashi. Time-efficient constant-space-overhead fault-tolerant quantum computation. *Nature Physics*, 20(2):247–253, 2024. URL http://dx.doi.org/10.1038/s41567-023-02325-8.

[28] Quynh T. Nguyen and Christopher A. Pattison. Quantum fault tolerance with constant-space and logarithmic-time overheads. *arXiv preprint arXiv:2411.03632*, 2024. URL https://doi.org/10.48550/arXiv.2411.03632.

[29] Jens Koch, Terri M. Yu, Jay Gambetta, A. A. Houck, D. I. Schuster, J. Majer, Alexandre Blais, M. H. Devoret, S. M. Girvin, and R. J. Schoelkopf. Charge-insensitive qubit design derived from the cooper pair box. *Phys. Rev. A*, 76:042319, 2007. URL https://link.aps.org/doi/10.1103/PhysRevA.76.042319.

[30] D. J. Wineland, C. Monroe, W. M. Itano, D. Leibfried, B. E. King, and D. M. Meekhof. Experimental issues in coherent quantum-state manipulation of trapped atomic ions. *Journal of Research of the National Institute of Standards and Technology*, 103:259–328, 1998. URL https://tf.nist.gov/general/pdf/1275.pdf.

[31] Philipp Schindler, Daniel Nigg, Thomas Monz, Julio T Barreiro, Esteban Martinez, Shannon X Wang, Stephan Quint, Matthias F Brandl, Volckmar Nebendahl, Christian F Roos, Michael Chwalla, Markus Hennrich, and Rainer Blatt. A quantum information processor with trapped ions. *New Journal of Physics*, 15(12):123012, 2013. URL http://dx.doi.org/10.1088/1367-2630/15/12/123012.

[32] J. M. Pino, J. M. Dreiling, C. Figgatt, J. P. Gaebler, S. A. Moses, M. S. Allman, C. H. Baldwin, M. Foss-Feig, D. Hayes, K. Mayer, C. Ryan-Anderson, and B. Neyenhuis. Demonstration of the trapped-ion quantum CCD computer architecture. *Nature*, 592:209–213, 2021. URL https://doi.org/10.1038/s41586-021-03318-4.

[33] Dolev Bluvstein, Harry Levine, Giulia Semeghini, Tout T. Wang, Sepehr Ebadi, Marcin Kalinowski, Alexander Keesling, Nishad Maskara, Hannes Pichler, Markus Greiner, Vladan Vuletić, and Mikhail D. Lukin. A quantum processor based on coherent transport of entangled atom arrays. *Nature*, 604(7906):451–456, 2022. URL https://doi.org/10.1038/s41586-022-04592-6.

[34] Pieter Kok, W. J. Munro, Kae Nemoto, T. C. Ralph, Jonathan P. Dowling, and G. J. Milburn. Linear optical quantum computing with photonic qubits. *Reviews of Modern Physics*, 79(1):135–174, 2007. URL http://dx.doi.org/10.1103/RevModPhys.79.135.

[35] Victor V. Albert, Kyungjoo Noh, Kasper Duivenvoorden, Dylan J. Young, R. T. Brierley, Philip Reinhold, Christophe Vuillot, Linshu Li, Chao Shen, S. M. Girvin, Barbara M. Terhal, and Liang Jiang. Performance and structure of single-mode bosonic codes. *Phys. Rev. A*, 97:032346, 2018. URL https://link.aps.org/doi/10.1103/PhysRevA.97.032346.

[36] Atharv Joshi, Kyungjoo Noh, and Yvonne Y Gao. Quantum information processing with bosonic qubits in circuit qed. *Quantum Science and Technology*, 6(3):033001, 2021. URL https://dx.doi.org/10.1088/2058-9565/abe989.

[37] Guido Burkard, Thaddeus D. Ladd, Andrew Pan, John M. Nichol, and Jason R. Petta. Semiconductor spin qubits. *Rev. Mod. Phys.*, 95:025003, 2023. URL https://link.aps.org/doi/10.1103/RevModPhys.95.025003.

[38] S. B. Bravyi and A. Yu. Kitaev. Quantum codes on a lattice with boundary. *arXiv preprint arXiv:quant-ph/9811052*, 1998. URL https://doi.org/10.48550/arXiv.quant-ph/9811052.

[39] A.Yu. Kitaev. Fault-tolerant quantum computation by anyons. *Annals of Physics*, 303(1): 2–30, 2003. URL https://doi.org/10.1016/S0003-4916(02)00018-0.

[40] Daniel Gottesman, Alexei Kitaev, and John Preskill. Encoding a qubit in an oscillator. *Phys. Rev. A*, 64:012310, 2001. URL https://link.aps.org/doi/10.1103/PhysRevA.64.012310.

[41] Zijun Chen et al. Exponential suppression of bit or phase errors with cyclic error correction. *Nature*, 595(78677867):383–387, 2021. URL https://doi.org/10.1038/s41586-021-03588-y.

[42] C. Ryan-Anderson, J. G. Bohnet, K. Lee, D. Gresh, A. Hankin, J. P. Gaebler, D. Francois, A. Chernoguzov, D. Lucchetti, N. C. Brown, et al. Realization of real-time fault-tolerant quantum error correction. *Phys. Rev. X*, 11:041058, 2021. URL https://link.aps.org/doi/10.1103/PhysRevX.11.041058.

[43] C. Ryan-Anderson, N. C. Brown, M. S. Allman, B. Arkin, G. Asa-Attuah, C. Baldwin, J. Berg, J. G. Bohnet, S. Braxton, N. Burdick, et al. Implementing fault-tolerant entangling gates on the five-qubit code and the color code. *arXiv preprint arXiv:2208.01863*, 2022. URL https://doi.org/10.48550/arXiv.2208.01863.

[44] Youwei Zhao, Yangsen Ye, He-Liang Huang, Yiming Zhang, Dachao Wu, Huijie Guan, Qingling Zhu, Zuolin Wei, Tan He, Sirui Cao, Fusheng Chen, et al. Realization of an error-correcting surface code with superconducting qubits. *Phys. Rev. Lett.*, 129:030501, 2022. URL https://link.aps.org/doi/10.1103/PhysRevLett.129.030501.

[45] Dolev Bluvstein, Simon J. Evered, Alexandra A. Geim, Sophie H. Li, Hengyun Zhou, Tom Manovitz, Sepehr Ebadi, Madelyn Cain, Marcin Kalinowski, Dominik Hangleiter, et al. Logical quantum processor based on reconfigurable atom arrays. *Nature*, 626:58–65, 2024. URL http://dx.doi.org/10.1038/s41586-023-06927-3.

[46] Noah Berthusen, Joan Dreiling, Cameron Foltz, John P. Gaebler, Thomas M. Gatterman, Dan Gresh, Nathan Hewitt, Michael Mills, Steven A. Moses, Brian Neyenhuis, Peter Siegfried, and David Hayes. Experiments with the four-dimensional surface code on a quantum charge-coupled device quantum computer. *Physical Review A*, 110(6), 2024. URL https://doi.org/10.1103/PhysRevA.110.062413.

[47] Google Quantum AI and Collaborators. Quantum error correction below the surface code threshold. *Nature*, 2024. URL https://doi.org/10.1038/s41586-024-08449-y.

[48] Ben W. Reichardt, David Aasen, Rui Chao, Alex Chernoguzov, Wim van Dam, John P. Gaebler, Dan Gresh, Dominic Lucchetti, Michael Mills, Steven A. Moses, et al. Demonstration of quantum computation and error correction with a tesseract code. *arXiv preprint arXiv:2409.04628*, 2024. URL https://doi.org/10.48550/arXiv.2409.04628.

[49] M. P. da Silva, C. Ryan-Anderson, J. M. Bello-Rivas, A. Chernoguzov, J. M. Dreiling, C. Foltz, F. Frachon, J. P. Gaebler, T. M. Gatterman, L. Grans-Samuelsson, D. Hayes, et al. Demonstration of logical qubits and repeated error correction with better-than-physical error rates. *arXiv preprint arXiv:2404.02280*, 2024. URL https://doi.org/10.48550/arXiv.2404.02280.

[50] Ben W. Reichardt, Adam Paetznick, David Aasen, Ivan Basov, Juan M. Bello-Rivas, Parsa Bonderson, Rui Chao, Wim van Dam, Matthew B. Hastings, et al. Logical computation demonstrated with a neutral atom quantum processor. *arXiv preprint arXiv:2411.11822*, 2024. URL https://doi.org/10.48550/arXiv.2411.11822.

[51] Nathan Lacroix, Alexandre Bourassa, Francisco J. H. Heras, Lei M. Zhang, Johannes Bausch, Andrew W. Senior, Thomas Edlich, Noah Shutty, Volodymyr Sivak, Andreas Bengtsson, et al. Scaling and logic in the color code on a superconducting quantum processor. *arXiv preprint arXiv:2412.14256*, 2024. URL https://doi.org/10.48550/arXiv.2412.14256.

[52] Yifan Hong, Elijah Durso-Sabina, David Hayes, and Andrew Lucas. Entangling four logical qubits beyond break-even in a nonlocal code. *Phys. Rev. Lett.*, 133:180601, 2024. URL https://doi.org/10.1103/PhysRevLett.133.180601.

[53] Nikolaos Koukoulekidis, Samson Wang, Tom O'Leary, Daniel Bultrini, Lukasz Cincio, and Piotr Czarnik. A framework of partial error correction for intermediate-scale quantum computers. *arXiv preprint arXiv:2306.15531*, 2023. URL https://doi.org/10.48550/arXiv.2306.15531.

[54] Daniel Bultrini, Samson Wang, Piotr Czarnik, Max Hunter Gordon, M. Cerezo, Patrick J. Coles, and Lukasz Cincio. The battle of clean and dirty qubits in the era of partial error correction. *Quantum*, 7:1060, 2023. URL http://dx.doi.org/10.22331/q-2023-07-13-1060.

[55] Matthew B. Hastings and Jeongwan Haah. Dynamically generated logical qubits. *Quantum*, 5:564, 2021. URL http://dx.doi.org/10.22331/q-2021-10-19-564.

[56] Margarita Davydova, Nathanan Tantivasadakarn, Shankar Balasubramanian, and David Aasen. Quantum computation from dynamic automorphism codes. *Quantum*, 8:1448, 2024. URL http://dx.doi.org/10.22331/q-2024-08-27-1448.

[57] Xiaozhen Fu and Daniel Gottesman. Error correction in dynamical codes. *arXiv preprint arXiv:2403.04163*, 2024. URL https://doi.org/10.48550/arXiv.2403.04163.

[58] Eric Dennis, Alexei Kitaev, Andrew Landahl, and John Preskill. Topological quantum memory. *Journal of Mathematical Physics*, 43(9):4452–4505, 2002. URL http://dx.doi.org/10.1063/1.1499754.

[59] Héctor Bombín. Single-shot fault-tolerant quantum error correction. *Phys. Rev. X*, 5: 031043, 2015. URL https://doi.org/10.1103/PhysRevX.5.031043.

[60] Earl T Campbell. A theory of single-shot error correction for adversarial noise. *Quantum Science and Technology*, 4(2):025006, 2019. URL https://dx.doi.org/10.1088/2058-9565/aafc8f.

[61] Benjamin Schumacher and Michael D. Westmoreland. Approximate quantum error correction. *Quantum Information Processing*, 1(1/2):5–12, 2002. URL https://doi.org/10.1023/A:1019653202562.

[62] E. Berlekamp, R. McEliece, and H. van Tilborg. On the inherent intractability of certain coding problems (corresp.). *IEEE Transactions on Information Theory*, 24(3):384–386, 1978. URL https://doi.org/10.1109/TIT.1978.1055873.

[63] R. Gallager. Low-density parity-check codes. *IRE Transactions on Information Theory*, 8 (1):21–28, 1962. URL https://doi.org/10.1109/TIT.1962.1057683.

[64] Judea Pearl. Reverend bayes on inference engines: a distributed hierarchical approach. In *Proceedings of the Second AAAI Conference on Artificial Intelligence*, AAAI'82, page 133–136. AAAI Press, 1982. URL https://dl.acm.org/doi/10.5555/2876686.2876719.

[65] C. E. Shannon. A mathematical theory of communication. *The Bell System Technical Journal*, 27(3):379–423, 1948. URL https://doi.org/10.1002/j.1538-7305.1948.tb01338.x.

[66] Michael A. Nielsen and Isaac L. Chuang. *Quantum Computation and Quantum Information: 10th Anniversary Edition*. Cambridge University Press, 2010.

[67] Daniel Gottesman. Stabilizer codes and quantum error correction. *arXiv preprint arXiv:quant-ph/9705052*, 1997. URL https://doi.org/10.48550/arXiv.quant-ph/9705052.

[68] A. R. Calderbank, E. M. Rains, P. W. Shor, and N. J. A. Sloane. Quantum error correction and orthogonal geometry. *Phys. Rev. Lett.*, 78(3):405–408, 1997. URL http://dx.doi.org/10.1103/PhysRevLett.78.405.

[69] Raymond Laflamme, Cesar Miquel, Juan Pablo Paz, and Wojciech Hubert Zurek. Perfect quantum error correction code. *arXiv preprint arXiv:quant-ph/9602019*, 1996. URL https://arxiv.org/abs/quant-ph/9602019.

[70] Charles H. Bennett, David P. DiVincenzo, John A. Smolin, and William K. Wootters. Mixed-state entanglement and quantum error correction. *Phys. Rev. A*, 54:3824–3851, 1996. URL https://doi.org/10.1103/PhysRevA.54.3824.

[71] A. R. Calderbank and Peter W. Shor. Good quantum error-correcting codes exist. *Phys. Rev. A*, 54(2):1098–1105, 1996. URL http://dx.doi.org/10.1103/PhysRevA.54.1098.

[72] A. M. Steane. Simple quantum error-correcting codes. *Physical Review A*, 54(6): 4741–4751, 1996. URL http://dx.doi.org/10.1103/PhysRevA.54.4741.

[73] Michael Vasmer and Dan E. Browne. Three-dimensional surface codes: Transversal gates and fault-tolerant architectures. *Phys. Rev. A*, 100:012312, 2019. URL https://link.aps.org/doi/10.1103/PhysRevA.100.012312.

[74] Andrew Steane. Multiple particle interference and quantum error correction. *Proceedings of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences*, 452(1954):2551–2577, 1996. URL http://dx.doi.org/10.1098/rspa.1996.0136.

[75] Nikolas P. Breuckmann and Jens Niklas Eberhardt. Quantum low-density parity-check codes. *PRX Quantum*, 2:040101, 2021. URL https://link.aps.org/doi/10.1103/PRXQuantum.2.040101.

[76] Jean-Pierre Tillich and Gilles Zemor. Quantum LDPC codes with positive rate and minimum distance proportional to the square root of the blocklength. *IEEE Transactions on Information Theory*, 60(2):1193–1202, 2014. URL https://doi.org/10.1109%2Ftit.2013.2292061.

[77] Sergey Bravyi and Matthew B. Hastings. Homological product codes. In *Proceedings of the Forty-Sixth Annual ACM Symposium on Theory of Computing*, STOC '14, page 273–282, New York, NY, USA, 2014. Association for Computing Machinery. URL https://doi.org/10.1145/2591796.2591870.

[78] M. Sipser and D.A. Spielman. Expander codes. *IEEE Transactions on Information Theory*, 42(6):1710–1722, 1996. URL https://doi.org/10.1109/18.556667.

[79] Anthony Leverrier, Jean-Pierre Tillich, and Gilles Zémor. Quantum expander codes. In *2015 IEEE 56th Annual Symposium on Foundations of Computer Science*, pages 810–824, 2015. URL https://doi.org/10.1109/FOCS.2015.55.

[80] Omar Fawzi, Antoine Grospellier, and Anthony Leverrier. Constant overhead quantum fault tolerance with quantum expander codes. *Commun. ACM*, 64(1):106–114, 2020. URL https://doi.org/10.1145/3434163.

[81] Yifan Hong. Single-shot preparation of hypergraph product codes via dimension jump. *arXiv preprint arXiv:2410.05171*, 2024. URL https://doi.org/10.48550/arXiv.2410.05171.

[82] Weilei Zeng and Leonid P. Pryadko. Higher-dimensional quantum hypergraph-product codes with finite rates. *Phys. Rev. Lett.*, 122:230501, 2019. URL https://link.aps.org/doi/10.1103/PhysRevLett.122.230501.

[83] Armanda O. Quintavalle and Earl T. Campbell. Reshape: A decoder for hypergraph product codes. *IEEE Transactions on Information Theory*, 68(10), 2022. URL https://doi.org/10.1109/TIT.2022.3184108.

[84] Armanda O. Quintavalle, Paul Webster, and Michael Vasmer. Partitioning qubits in hypergraph product codes to implement logical gates. *Quantum*, 7:1153, 2023. URL http://dx.doi.org/10.22331/q-2023-10-24-1153.

[85] Sergey Bravyi, Andrew W. Cross, Jay M. Gambetta, Dmitri Maslov, Patrick Rall, and Theodore J. Yoder. High-threshold and low-overhead fault-tolerant quantum memory. *Nature*, 627:778–782, 2024. URL https://doi.org/10.1038/s41586-024-07107-7.

[86] Alexey A. Kovalev and Leonid P. Pryadko. Quantum kronecker sum-product low-density parity-check codes with finite rate. *Phys. Rev. A*, 88:012311, 2013. URL https://link.aps.org/doi/10.1103/PhysRevA.88.012311.

[87] Chris N. Self, Marcello Benedetti, and David Amaro. Protecting expressive circuits with a quantum error detection code. *Nature Physics*, 20(2):219–224, 2024. URL http://dx.doi.org/10.1038/s41567-023-02282-2.

[88] Hayato Goto. High-performance fault-tolerant quantum computing with many-hypercube codes. *Science Advances*, 10(36), 2024. URL http://dx.doi.org/10.1126/sciadv.adp6388.

[89] David Poulin and Yeojin Chung. On the iterative decoding of sparse quantum codes. *Quantum Info. Comput.*, 8(10):987–1000, 2008. URL https://dl.acm.org/doi/10.5555/2016985.2016993.

[90] Zunaira Babar, Panagiotis Botsinis, Dimitrios Alanis, Soon Xin Ng, and Lajos Hanzo. Fifteen years of quantum ldpc coding and improved decoding strategies. *IEEE Access*, 3: 2492–2519, 2015. URL https://doi.org/10.1109/ACCESS.2015.2503267.

[91] Nithin Raveendran and Bane Vasić. Trapping sets of quantum ldpc codes. *Quantum*, 5: 562, 2021. URL http://dx.doi.org/10.22331/q-2021-10-14-562.

[92] Joschka Roffe, David R. White, Simon Burton, and Earl Campbell. Decoding across the quantum low-density parity-check code landscape. *Phys. Rev. Res.*, 2:043423, 2020. URL https://link.aps.org/doi/10.1103/PhysRevResearch.2.043423.

[93] Pavel Panteleev and Gleb Kalachev. Degenerate quantum LDPC codes with good finite length performance. *Quantum*, 5:585, 2021. URL http://dx.doi.org/10.22331/q-2021-11-22-585.

[94] Antoine Grospellier, Lucien Grouès, Anirudh Krishna, and Anthony Leverrier. Combining hard and soft decoders for hypergraph product codes. *Quantum*, 5:432, 2021. URL https://doi.org/10.22331/q-2021-04-15-432.

[95] K. Duivenvoorden, N. P. Breuckmann, and B. M. Terhal. Renormalization group decoder for a four-dimensional toric code. *IEEE Transactions on Information Theory*, 65(4): 2545–2562, 2019. URL http://dx.doi.org/10.1109/TIT.2018.2879937.

[96] Oscar Higgott. Pymatching: A python package for decoding quantum codes with minimum-weight perfect matching. *ACM Transactions on Quantum Computing*, 3(3), 2022.

[97] Christopher A. Pattison, Michael E. Beverland, Marcus P. da Silva, and Nicolas Delfosse. Improved quantum error correction using soft information. *2107.13589*, 2021. URL https://doi.org/10.48550/arXiv.2107.13589.

[98] David Poulin. Optimal and efficient decoding of concatenated quantum block codes. *Physical Review A*, 74(5), 2006. URL http://dx.doi.org/10.1103/PhysRevA.74.052333.

[99] Nadine Meister, Christopher A. Pattison, and John Preskill. Efficient soft-output decoders for the surface code. *arXiv preprint arXiv:2405.07433*, 2024. URL https://doi.org/10.48550/arXiv.2405.07433.

[100] Oscar Higgott and Nikolas P. Breuckmann. Improved single-shot decoding of higher-dimensional hypergraph-product codes. *PRX Quantum*, 4(2), 2023. URL http://dx.doi.org/10.1103/PRXQuantum.4.020332.

[101] Armanda O. Quintavalle, Michael Vasmer, Joschka Roffe, and Earl T. Campbell. Single-shot error correction of three-dimensional homological product codes. *PRX Quantum*, 2: 020340, 2021. URL https://link.aps.org/doi/10.1103/PRXQuantum.2.020340.

[102] Austin G. Fowler, Ashley M. Stephens, and Peter Groszkowski. High-threshold universal quantum computation on the surface code. *Phys. Rev. A*, 80:052312, 2009. URL https://link.aps.org/doi/10.1103/PhysRevA.80.052312.

[103] Nicolas Delfosse, Michael E. Beverland, and Maxime A. Tremblay. Bounds on stabilizer measurement circuits and obstructions to local implementations of quantum LDPC codes. *arXiv preprint arXiv:2109.14599*, 2021. URL https://doi.org/10.48550/arXiv.2109.14599.

[104] Qian Xu, J. Pablo Bonilla Ataides, Christopher A. Pattison, Nithin Raveendran, Dolev Bluvstein, Jonathan Wurtz, Bane Vasic, Mikhail D. Lukin, Liang Jiang, and Hengyun Zhou. Constant-overhead fault-tolerant quantum computation with reconfigurable atom arrays. *Nat. Phys.*, 20:1084–1090, 2024. URL https://doi.org/10.1038/s41567-024-02479-z.

[105] David S. Wang, Austin G. Fowler, and Lloyd C. L. Hollenberg. Surface code quantum computing with error rates over 1%. *Physical Review A*, 83(2), 2011. URL http://dx.doi.org/10.1103/PhysRevA.83.020302.

[106] Craig Gidney. Stim: a fast stabilizer circuit simulator. *Quantum*, 5:497, 2021. URL https://doi.org/10.22331/q-2021-07-06-497.

[107] Matt McEwen, Dave Bacon, and Craig Gidney. Relaxing hardware requirements for surface code circuits using time-dynamics. *Quantum*, 7:1172, 2023. URL https://doi.org/10.22331%2Fq-2023-11-07-1172.

[108] Sergey Bravyi and Barbara Terhal. A no-go theorem for a two-dimensional self-correcting quantum memory based on stabilizer codes. *New Journal of Physics*, 11(4):043029, 2009. URL http://dx.doi.org/10.1088/1367-2630/11/4/043029.

[109] Sergey Bravyi, David Poulin, and Barbara Terhal. Tradeoffs for reliable quantum information storage in 2D systems. *Phys. Rev. Lett.*, 104:050503, 2010. URL https://doi.org/10.1103/PhysRevLett.104.050503.

[110] P. Krantz, M. Kjaergaard, F. Yan, T. P. Orlando, S. Gustavsson, and W. D. Oliver. A quantum engineer's guide to superconducting qubits. *Applied Physics Reviews*, 6(2), 2019. doi: 10.1063/1.5089550. URL http://dx.doi.org/10.1063/1.5089550.

[111] Daniel Loss and David P. DiVincenzo. Quantum computation with quantum dots. *Phys. Rev. A*, 57:120–126, 1998. URL https://link.aps.org/doi/10.1103/PhysRevA.57.120.

[112] M. V. Gurudev Dutt, L. Childress, L. Jiang, E. Togan, J. Maze, F. Jelezko, A. S. Zibrov, P. R. Hemmer, and M. D. Lukin. Quantum register based on individual electronic and nuclear spin qubits in diamond. *Science*, 316(5829):1312–1316, 2007. URL https://www.science.org/doi/abs/10.1126/science.1139831.

[113] Nouédyn Baspin and Anirudh Krishna. Quantifying nonlocality: How outperforming local quantum codes is expensive. *Phys. Rev. Lett.*, 129:050505, 2022. URL https://doi.org/10.1103/PhysRevLett.129.050505.

[114] Nouédyn Baspin and Anirudh Krishna. Connectivity constrains quantum codes. *Quantum*, 6:711, 2022. URL http://dx.doi.org/10.22331/q-2022-05-13-711.

[115] Samuel Dai and Ray Li. Locality vs quantum codes. *arXiv preprint arXiv:2409.15203*, 2024. URL https://doi.org/10.48550/arXiv.2409.15203.

[116] Nouédyn Baspin, Omar Fawzi, and Ala Shayeghi. A lower bound on the overhead of quantum error correction in low dimensions. *arXiv preprint arXiv:2302.04317*, 2023. URL https://doi.org/10.48550/arXiv.2302.04317.

[117] Noah Berthusen and Daniel Gottesman. Partial syndrome measurement for hypergraph product codes. *Quantum*, 8:1345, 2024. URL http://dx.doi.org/10.22331/q-2024-05-14-1345.

[118] Jiří Matoušek. On embedding expanders into ℓ_p spaces. *Israel Journal of Mathematics*, 102(1):189–197, 1997. URL https://doi.org/10.1007/BF02773799.

[119] Nikolas P. Breuckmann and Barbara M. Terhal. Constructions and noise threshold of hyperbolic surface codes. *IEEE Transactions on Information Theory*, 62(6):3731–3744, 2016. URL http://dx.doi.org/10.1109/TIT.2016.2555700.

[120] Matthew B. Hastings. Decoding in hyperbolic spaces: quantum LDPC codes with linear rate and efficient error correction. *Quantum Info. Comput.*, 14(13–14):1187–1202, 2014. URL https://dl.acm.org/doi/abs/10.5555/2685164.2685173.

[121] Nikolas P. Breuckmann and Vivien Londe. Single-shot decoding of linear rate LDPC quantum codes with high performance. *IEEE Trans. Inf. Theory*, 68(1):272–286, 2022. URL https://doi.org/10.1109/TIT.2021.3122352.

[122] Matthew B. Hastings, Jeongwan Haah, and Ryan O'Donnell. Fiber bundle codes: breaking the $n^{1/2}polylog(n)$ barrier for quantum LDPC codes. In *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing*, STOC '21, page 1276–1288. ACM, 2021. URL http://dx.doi.org/10.1145/3406325.3451005.

[123] Yifan Hong, Matteo Marinelli, Adam M. Kaufman, and Andrew Lucas. Long-range-enhanced surface codes. *Physical Review A*, 110(2), 2024. URL http://dx.doi.org/10.1103/PhysRevA.110.022607.

[124] Laura Pecorari, Sven Jandura, Gavin K. Brennen, and Guido Pupillo. High-rate quantum ldpc codes for long-range-connected neutral atom registers. *Nature Communications*, 16 (1), 2025. URL http://dx.doi.org/10.1038/s41467-025-56255-5.

[125] Daniel Gottesman. Opportunities and challenges in fault-tolerant quantum computation. *arXiv preprint arXiv:2210.15844*, 2022. URL https://doi.org/10.48550/arXiv.2210.15844.

[126] Omar Fawzi, Antoine Grospellier, and Anthony Leverrier. Efficient decoding of random errors for quantum expander codes. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing*, page 521–534, 2018. URL https://doi.org/10.1145/3188745.3188886.

[127] Antoine Grospellier. Constant time decoding of quantum expander codes and application to fault-tolerant quantum computation. *PhD Thesis*, 2019. Sorbonne Université.

[128] Alexey A. Kovalev, Sanjay Prabhakar, Ilya Dumer, and Leonid P. Pryadko. Numerical and analytical bounds on threshold error rates for hypergraph-product codes. *Phys. Rev. A*, 97: 062320, 2018. URL https://doi.org/10.1103/PhysRevA.97.062320.

[129] Antoine Grospellier and Anirudh Krishna. Numerical study of hypergraph product codes. *arXiv preprint arXiv:1810.03681*, 2019. URL https://doi.org/10.48550/arXiv.1810.03681.

[130] Maxime A. Tremblay, Nicolas Delfosse, and Michael E. Beverland. Constant-overhead quantum error correction with thin planar connectivity. *Physical Review Letters*, 129(5), 2022. URL http://dx.doi.org/10.1103/PhysRevLett.129.050504.

[131] Noah Berthusen, Shi Jie Samuel Tan, Eric Huang, and Daniel Gottesman. Adaptive syndrome extraction. *arXiv preprint arXiv:2502.14835*, 2025. URL https://doi.org/10.48550/arXiv.2502.14835.

[132] Oscar Higgott and Craig Gidney. Sparse blossom: correcting a million errors per core second with minimum-weight matching. *arXiv preprint arXiv:2303.15933*, 2023. URL https://doi.org/10.48550/arXiv.2303.15933.

[133] Z. Miller and J. B. Orlin. NP-completeness for minimizing maximum edge length in grid embeddings. *Journal of Algorithms*, 6(1):10–16, 1985. URL https://doi.org/10.1016/0196-6774(85)90016-1.

[134] Charles H. Bennett, Gilles Brassard, Sandu Popescu, Benjamin Schumacher, John A. Smolin, and William K. Wootters. Purification of noisy entanglement and faithful teleportation via noisy channels. *Phys. Rev. Lett.*, 76(5):722–725, 1996. URL http://dx.doi.org/10.1103/PhysRevLett.76.722.

[135] Joshua Viszlai, Willers Yang, Sophia Fuhui Lin, Junyu Liu, Natalia Nottingham, Jonathan M. Baker, and Frederic T. Chong. Matching generalized-bicycle codes to neutral atoms for low-overhead fault-tolerance. *arXiv preprint arXiv:2311.16980*, 2023. URL https://doi.org/10.48550/arXiv.2311.16980.

[136] Maxim De Smet, Yuta Matsumoto, Anne-Marije J. Zwerver, Larysa Tryputen, Sander L. de Snoo, Sergey V. Amitonov, Amir Sammak, Nodar Samkharadze, Önder Gül, Rick N. M. Wasserman, et al. High-fidelity single-spin shuttling in silicon, 2024. URL https://arxiv.org/abs/2406.07267.

[137] Austin G. Fowler, Adam C. Whiteside, and Lloyd C. L. Hollenberg. Towards practical classical processing for the surface code. *Phys. Rev. Lett.*, 108:180501, 2012. URL https://link.aps.org/doi/10.1103/PhysRevLett.108.180501.

[138] Rajeev Acharya, Igor Aleiner, Richard Allen, Trond I. Andersen, Markus Ansmann, Frank Arute, Kunal Arya, Abraham Asfaw, Juan Atalaya, Ryan Babbush, Dave Bacon, et al. Suppressing quantum errors by scaling a surface code logical qubit. *Nature*, 614:676–681, 2023. URL https://doi.org/10.1038/s41586-022-05434-1.

[139] Daniel Litinski. A game of surface codes: Large-scale quantum computing with lattice surgery. *Quantum*, 3:128, 2019. URL http://dx.doi.org/10.22331/q-2019-03-05-128.

[140] Michael E. Beverland, Prakash Murali, Matthias Troyer, Krysta M. Svore, Torsten Hoefler, Vadym Kliuchnikov, Guang Hao Low, Mathias Soeken, Aarthi Sundaram, and Alexander Vaschillo. Assessing requirements to scale to practical quantum advantage. *arXiv preprint arXiv:2211.07629*, 2022. URL https://doi.org/10.48550/arXiv.2211.07629.

[141] Christopher A. Pattison, Anirudh Krishna, and John Preskill. Hierarchical memories: Simulating quantum LDPC codes with local gates. *arXiv preprint arXiv:2303.04798*, 2023. URL https://doi.org/10.48550/arXiv.2303.04798.

[142] Craig Gidney, Michael Newman, Peter Brooks, and Cody Jones. Yoked surface codes. *arXiv preprint arXiv:2312.04522*, 2023. URL https://doi.org/10.48550/arXiv.2312.04522.

[143] Diego Ruiz, Jérémie Guillaud, Anthony Leverrier, Mazyar Mirrahimi, and Christophe Vuillot. LDPC-cat codes for low-overhead quantum computing in 2D. *arXiv preprint arXiv:2401.09541*, 2024. URL https://doi.org/10.48550/arXiv.2401.09541.

[144] Noga Alon, F. R. K. Chung, and R. L. Graham. Routing permutations on graphs via matchings. *SIAM J. Discrete Math.*, 7(3):513–530, 1994. URL https://doi.org/10.1137/S0895480192236628.

[145] Louxin Zhang. Optimal bounds for matching routing on trees. *SIAM J. Discrete Math.*, 12 (1):64–77, 1999. URL https://doi.org/10.1137/S0895480197323159.

[146] Fan Chung. *Spectral Graph Theory*. American Mathematical Society, 1996. URL https://doi.org/10.1090/cbms/092.

[147] Alexander Cowtan, Silas Dilkes, Ross Duncan, Alexandre Krajenbrink, Will Simmons, and Seyon Sivarajah. On the Qubit Routing Problem. In *TQC 2019*, volume 135 of *LIPIcs*, pages 5:1–5:32, 2019. URL https://drops-dev.dagstuhl.de/entities/document/10.4230/LIPIcs.TQC.2019.5.

[148] Andrew M. Childs, Eddie Schoute, and Cem M. Unsal. Circuit transformations for quantum architectures. In *TQC 2019*. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2019. URL https://drops.dagstuhl.de/entities/document/10.4230/LIPIcs.TQC.2019.3.

[149] Dhruv Devulapalli, Eddie Schoute, Aniruddha Bapat, Andrew M. Childs, and Alexey V. Gorshkov. Quantum routing with teleportation. *Phys. Rev. Res.*, 6:033313, 2024. URL https://link.aps.org/doi/10.1103/PhysRevResearch.6.033313.

[150] David J. Rosenbaum. Optimal quantum circuits for nearest-neighbor architectures. In *TQC 2013*, volume 22 of *LIPIcs*, pages 294–307, 2013. URL https://doi.org/10.4230/LIPIcs.TQC.2013.294.

[151] Stefan Hillmich, Alwin Zulehner, and Robert Wille. Exploiting quantum teleportation in quantum circuit mapping. In *Proceedings of the 26th Asia and South Pacific Design Automation Conference*, ASPDAC '21, page 792–797, 2021. URL https://doi.org/10.1145/3394885.3431604.

[152] Michael Beverland, Vadym Kliuchnikov, and Eddie Schoute. Surface code compilation via edge-disjoint paths. *PRX Quantum*, 3(2):020342, 2022. URL https://doi.org/10.1103/PRXQuantum.3.020342.

[153] M. Żukowski, A. Zeilinger, M. A. Horne, and A. K. Ekert. "event-ready-detectors" bell experiment via entanglement swapping. *Phys. Rev. Lett.*, 71:4287–4290, 1993. URL https://link.aps.org/doi/10.1103/PhysRevLett.71.4287.

[154] Daniel Gottesman and Isaac L. Chuang. Demonstrating the viability of universal quantum computation using teleportation and single-qubit operations. *Nature*, 402(6760):390–393, 1999. URL https://doi.org/10.1038%2F46503.

[155] Alireza Shafaei, Mehdi Saeedi, and Massoud Pedram. Qubit placement to minimize communication overhead in 2d quantum architectures. In *2014 19th Asia and South Pacific Design Automation Conference*, pages 495–500, 2014. URL https://doi.org/10.1109/ASPDAC.2014.6742940.

[156] Marcos Yukio Siraichi, Vinícius Fernandes dos Santos, Caroline Collange, and Fernando Magno Quintao Pereira. Qubit allocation. In *Proceedings of the 2018 International Symposium on Code Generation and Optimization*, pages 113–125, 2018. URL https://doi.org/10.1145/3168822.

[157] Gushu Li, Yufei Ding, and Yuan Xie. Tackling the qubit mapping problem for NISQ-era quantum devices. In *Proceedings of the Twenty-Fourth International Conference on Architectural Support for Programming Languages and Operating Systems*, pages 1001–1014, 2019. URL https://doi.org/10.1145/3297858.3304023.

[158] Steven Herbert. On the depth overhead incurred when running quantum algorithms on near-term quantum computers with limited qubit connectivity. *arXiv preprint arXiv:1805.12570*, 2020. URL https://doi.org/10.48550/arXiv.1805.12570.

[159] Elisa Bäumer, Vinay Tripathi, Derek S. Wang, Patrick Rall, Edward H. Chen, Swarnadeep Majumder, Alireza Seif, and Zlatko K. Minev. Efficient long-range entanglement using dynamic circuits. *PRX Quantum*, 5:030339, 2024. URL https://link.aps.org/doi/10.1103/PRXQuantum.5.030339.

[160] Yu Tomita and Krysta M. Svore. Low-distance surface codes under realistic quantum noise. *Phys. Rev. A*, 90:062320, 2014. URL https://link.aps.org/doi/10.1103/PhysRevA.90.062320.

[161] Yong Zeng, Peng Xu, Xiaodong He, Yangyang Liu, Min Liu, Jin Wang, D. J. Papoular, G. V. Shlyapnikov, and Mingsheng Zhan. Entangling two individual atoms of different isotopes via rydberg blockade. *Phys. Rev. Lett.*, 119:160502, 2017. URL https://link.aps.org/doi/10.1103/PhysRevLett.119.160502.

[162] Kevin Singh, Shraddha Anand, Andrew Pocklington, Jordan T. Kemp, and Hannes Bernien. Dual-element, two-dimensional atom array with continuous-mode operation. *Phys. Rev. X*, 12:011040, 2022. URL https://link.aps.org/doi/10.1103/PhysRevX.12.011040.

[163] Shraddha Anand, Conor E. Bradley, Ryan White, Vikram Ramesh, Kevin Singh, and Hannes Bernien. A dual-species rydberg array. *Nat. Phys.*, 20:1744–1750, 2024. URL https://doi.org/10.1038/s41567-024-02638-2.

[164] A. V. Gorshkov, A. M. Rey, A. J. Daley, M. M. Boyd, J. Ye, P. Zoller, and M. D. Lukin. Alkaline-earth-metal atoms as few-qubit quantum registers. *Phys. Rev. Lett.*, 102:110503, 2009. URL https://link.aps.org/doi/10.1103/PhysRevLett.102.110503.

[165] Joanna W. Lis, Aruku Senoo, William F. McGrew, Felix Rönchen, Alec Jenkins, and Adam M. Kaufman. Midcircuit operations using the omg architecture in neutral atom arrays. *Phys. Rev. X*, 13:041035, 2023. URL https://link.aps.org/doi/10.1103/PhysRevX.13.041035.

[166] Pascal Scholl, Adam L. Shaw, Ran Finkelstein, Richard Bing-Shiun Tsai, Joonhee Choi, and Manuel Endres. Erasure-cooling, control, and hyper-entanglement of motion in optical tweezers. *arXiv preprint arXiv:2311.15580*, 2023. URL https://doi.org/10.48550/arXiv.2311.15580.

[167] Leonid P. Pryadko, Vadim A. Shabashov, and Valerii K. Kozin. QDistRnd: A GAP package for computing the distance of quantum error-correcting codes. *Journal of Open Source Software*, 7(71):4120, 2022. URL https://doi.org/10.21105%2Fjoss.04120.

[168] David Deutsch, Artur Ekert, Richard Jozsa, Chiara Macchiavello, Sandu Popescu, and Anna Sanpera. Quantum privacy amplification and the security of quantum cryptography over noisy channels. *Phys. Rev. Lett.*, 77:2818–2821, 1996. URL https://link.aps.org/doi/10.1103/PhysRevLett.77.2818.

[169] Joschka Roffe. LDPC: Python tools for low density parity check codes, 2022. URL https://pypi.org/project/ldpc/.

[170] Thomas R. Scruby, Timo Hillmann, and Joschka Roffe. High-threshold, low-overhead and single-shot decodable fault-tolerant quantum memory. *arXiv preprint arXiv:2406.14445*, 2024. URL https://arxiv.org/abs/2406.14445.

[171] Liang Jiang, Jacob M. Taylor, Navin Khaneja, and Mikhail D. Lukin. Optimal approach to quantum communication using dynamic programming. *Proceedings of the National Academy of Sciences*, 104(44):17291–17296, 2007. URL https://www.pnas.org/doi/abs/10.1073/pnas.0703284104.

[172] W Dür and H J Briegel. Entanglement purification and quantum error correction. *Reports on Progress in Physics*, 70(8):1381, 2007. URL https://dx.doi.org/10.1088/0034-4885/70/8/R03.

[173] Stefan Krastanov, Victor V. Albert, and Liang Jiang. Optimized entanglement purification. *Quantum*, 3:123, 2019. URL http://dx.doi.org/10.22331/q-2019-02-18-123.

[174] Craig Gidney. Tetrationally compact entanglement purification. *arXiv preprint arXiv:2311.10971*, 2023. URL https://doi.org/10.48550/arXiv.2311.10971.

[175] M. Saffman, T. G. Walker, and K. Mølmer. Quantum information with Rydberg atoms. *Rev. Mod. Phys.*, 82:2313–2363, 2010. URL https://link.aps.org/doi/10.1103/RevModPhys.82.2313.

[176] Jeremy T. Young, Przemyslaw Bienias, Ron Belyansky, Adam M. Kaufman, and Alexey V. Gorshkov. Asymmetric blockade and multiqubit gates via dipole-dipole interactions. *Phys. Rev. Lett.*, 127:120501, 2021. URL https://link.aps.org/doi/10.1103/PhysRevLett.127.120501.

[177] N. Leung, Y. Lu, S. Chakram, R. K. Naik, N. Earnest, R. Ma, K. Jacobs, A. N. Cleland, and D. I. Schuster. Deterministic bidirectional communication and remote entanglement generation between superconducting quantum processors. *npj Quantum Inf.*, 5:18, 2019. URL https://doi.org/10.1038/s41534-019-0128-0.

[178] Xiaosi Xu, Niel de Beaudrap, Joe O'Gorman, and Simon C Benjamin. An integrity measure to benchmark quantum error correcting memories. *New Journal of Physics*, 20(2): 023009, 2018. URL http://dx.doi.org/10.1088/1367-2630/aaa372.

[179] Natalie C. Brown, John Peter Campora III, Cassandra Granade, Bettina Heim, Stefan Wernli, Ciaran Ryan-Anderson, Dominic Lucchetti, Adam Paetznick, Martin Roetteler, Krysta Svore, and Alex Chernoguzov. Advances in compilation for quantum hardware – a demonstration of magic state distillation and repeat-until-success protocols. *arXiv preprint arXiv:2310.12106*, 2023. URL https://doi.org/10.48550/arXiv.2310.12106.

[180] Ben W Reichardt. Fault-tolerant quantum error correction for steane's seven-qubit color code with few or no extra qubits. *Quantum Science and Technology*, 6(1):015007, 2020. URL http://dx.doi.org/10.1088/2058-9565/abc6f4.

[181] Christof Zalka. Threshold estimate for fault tolerant quantum computation. *arXiv preprint quant-ph/9612028*, 1996. URL https://doi.org/10.48550/arXiv.quant-ph/9612028.

[182] Nicolas Delfosse and Ben W. Reichardt. Short shor-style syndrome sequences. *arXiv preprint arXiv:2008.05051*, 2020. URL https://doi.org/10.48550/arXiv.2008.05051.

[183] Theerapat Tansuwannont, Balint Pato, and Kenneth R. Brown. Adaptive syndrome measurements for shor-style error correction. *Quantum*, 7:1075, 2023. URL http://dx.doi.org/10.22331/q-2023-08-08-1075.

[184] Noah Berthusen, Dhruv Devulapalli, Eddie Schoute, Andrew M. Childs, Michael J. Gullans, Alexey V. Gorshkov, and Daniel Gottesman. Toward a 2d local implementation of quantum low-density parity-check codes. *PRX Quantum*, 6(1), 2025. URL http://dx.doi.org/10.1103/PRXQuantum.6.010306.

[185] Matthew B Hastings. Weight reduction for quantum codes. *arXiv preprint arXiv:1611.03790*, 2016. URL https://doi.org/10.48550/arXiv.1611.03790.

[186] M. B. Hastings. On quantum weight reduction. *arXiv preprint arXiv:2102.10030*, 2023. URL https://doi.org/10.48550/arXiv.2102.10030.

[187] David Poulin. Stabilizer formalism for operator quantum error correction. *Phys. Rev. Lett.*, 95:230504, 2005. URL https://link.aps.org/doi/10.1103/PhysRevLett.95.230504.

[188] Dave Bacon. Operator quantum error-correcting subsystems for self-correcting quantum memories. *Phys. Rev. A*, 73:012340, 2006. URL https://link.aps.org/doi/10.1103/PhysRevA.73.012340.

[189] H. Bombin. Topological subsystem codes. *Phys. Rev. A*, 81:032301, 2010. URL https://link.aps.org/doi/10.1103/PhysRevA.81.032301.

[190] Nouédyn Baspin and Dominic Williamson. Wire codes. *arXiv preprint arXiv:2410.10194*, 2024. URL https://doi.org/10.48550/arXiv.2410.10194.

[191] Dave Bacon, Steven T. Flammia, Aram W. Harrow, and Jonathan Shi. Sparse quantum codes from quantum circuits. *IEEE Transactions on Information Theory*, 63(4):2464–2479, 2017. URL https://doi.org/10.1109/TIT.2017.2663199.

[192] Nicolas Delfosse and Adam Paetznick. Spacetime codes of clifford circuits. *arXiv preprint arXiv:2304.05943*, 2023. URL https://doi.org/10.48550/arXiv.2304.05943.

[193] Julio C Magdalena de la Fuente. Dynamical weight reduction of pauli measurements. *arXiv preprint arXiv:2410.12527*, 2024. URL https://doi.org/10.48550/arXiv.2410.12527.

[194] Qian Xu, Hengyun Zhou, Guo Zheng, Dolev Bluvstein, J Ataides, Mikhail D Lukin, and Liang Jiang. Fast and parallelizable logical computation with homological product codes. *arXiv preprint arXiv:2407.18490*, 2024. URL https://doi.org/10.48550/arXiv.2407.18490.

[195] E. Knill. Quantum computing with realistically noisy devices. *Nature*, 434:39–44, 2005. URL https://doi.org/10.1038/nature03350.

[196] Satoshi Yoshida, Shiro Tamiya, and Hayata Yamasaki. Concatenate codes, save qubits. *arXiv preprint arXiv:2402.09606*, 2024. URL https://doi.org/10.48550/arXiv.2402.09606.

[197] Ben Criger and Barbara Terhal. Noise thresholds for the [4,2,2]-concatenated toric code. *Quantum Info. Comput.*, 16(15–16):1261–1281, 2016. URL http://dx.doi.org/10.26421/QIC16.15-16.

[198] E. Knill. Scalable quantum computing in the presence of large detected-error rates. *Phys. Rev. A*, 71:042322, 2005. URL https://link.aps.org/doi/10.1103/PhysRevA.71.042322.

[199] P.W. Shor. Fault-tolerant quantum computation. In *Proceedings of 37th Conference on Foundations of Computer Science*, pages 56–65, 1996.

[200] H. Goto and H Uchikawa. Fault-tolerant quantum computation with a soft-decision decoder for error correction and detection by teleportation. *Sci. Rep.*, 3, 2013. URL https://doi.org/10.1038/srep02044.

[201] Shilin Huang and Shruti Puri. Increasing memory lifetime of quantum low-density parity check codes with sliding-window noisy syndrome decoding. *Phys. Rev. A*, 110:012453, 2024. URL https://link.aps.org/doi/10.1103/PhysRevA.110.012453.

[202] David S. Wang, Austin G. Fowler, and Lloyd C. L. Hollenberg. Surface code quantum computing with error rates over 1%. *Phys. Rev. A*, 83:020302, 2011. URL https://link.aps.org/doi/10.1103/PhysRevA.83.020302.

[203] Anqi Gong, Sebastian Cammerer, and Joseph M. Renes. Toward low-latency iterative decoding of QLDPC codes under circuit-level noise. *arXiv preprint arXiv:2403.18901*, 2024. URL https://doi.org/10.48550/arXiv.2403.18901.

[204] Christopher Pattison. QEC utilities for practical realizations of general qLDPC codes, 2024. URL https://github.com/qldpc/exp_ldpc.

[205] Argyris Giannisis Manes and Jahan Claes. Distance-preserving stabilizer measurements in hypergraph product codes. *arXiv preprint arXiv:2308.15520*, 2023. URL https://doi.org/10.48550/arXiv.2308.15520.

[206] Shi Jie Samuel Tan and Lev Stambler. Effective distance of higher dimensional hgps and weight-reduced quantum LDPC codes. *arXiv preprint arXiv:2409.02193*, 2024. URL https://doi.org/10.48550/arXiv.2409.02193.

[207] David J. C. MacKay and Radford M. Neal. Near shannon limit performance of low density parity check codes. *Electronics Letters*, 33, 1997. URL https://doi.org/10.1049/el:19970362.

[208] Timo Hillmann, Lucas Berent, Armanda O. Quintavalle, Jens Eisert, Robert Wille, and Joschka Roffe. Localized statistics decoding: A parallel decoding algorithm for quantum low-density parity-check codes. *arXiv preprint arXiv:2406.18655*, 2024. URL https://doi.org/10.48550/arXiv.2406.18655.

[209] Kun Fang, Munan Zhang, Ruqi Shi, and Yinan Li. Dynamic quantum circuit compilation. *arXiv preprint arXiv:2310.11021*, 2023. URL https://doi.org/10.48550/arXiv.2310.11021.

[210] H. Bombin and M. A. Martin-Delgado. Topological quantum distillation. *Physical Review Letters*, 97(18), 2006. URL http://dx.doi.org/10.1103/PhysRevLett.97.180501.

[211] C. Monroe, R. Raussendorf, A. Ruthven, K. R. Brown, P. Maunz, L.-M. Duan, and J. Kim. Large-scale modular quantum-computer architecture with atomic memory and photonic interconnects. *Phys. Rev. A*, 89:022317, 2014. URL https://link.aps.org/doi/10.1103/PhysRevA.89.022317.

[212] Armands Strikis and Lucas Berent. Quantum low-density parity-check codes for modular architectures. *PRX Quantum*, 4(2), 2023. URL http://dx.doi.org/10.1103/PRXQuantum.4.020321.

[213] Guanyu Zhu. A topological theory for qLDPC: non-clifford gates and magic state fountain on homological product codes with constant rate and beyond the $n^{1/3}$ distance barrier. *arXiv preprint arXiv:2501.19375*, 2025. URL https://doi.org/10.48550/arXiv.2501.19375.

[214] Alexander J. Malcolm, Andrew N. Glaudell, Patricio Fuentes, Daryus Chandra, Alexis Schotte, Colby DeLisle, Rafael Haenel, Amir Ebrahimi, Joschka Roffe, Armanda O. Quintavalle, Stefanie J. Beale, Nicholas R. Lee-Hone, and Stephanie Simmons. Computing efficiently in QLDPC codes. *arXiv preprint arXiv:2502.07150*, 2025. URL https://doi.org/10.48550/arXiv.2502.07150.

[215] Pavel Panteleev and Gleb Kalachev. Quantum LDPC codes with almost linear minimum distance. *IEEE Transactions on Information Theory*, 68(1):213–229, 2022. URL http://dx.doi.org/10.1109/TIT.2021.3119384.

[216] Hengyun Zhou, Chen Zhao, Madelyn Cain, Dolev Bluvstein, Casey Duckering, Hong-Ye Hu, Sheng-Tao Wang, Aleksander Kubica, and Mikhail D. Lukin. Algorithmic fault tolerance for fast quantum computing. *arXiv preprint arXiv:2406.17653*, 2024. URL https://doi.org/10.48550/arXiv.2406.17653.

[217] Lev Stambler, Anirudh Krishna, and Michael E. Beverland. Addressing stopping failures for small set flip decoding of hypergraph product codes. *arXiv preprint arXiv:2311.00877*, 2023. URL https://doi.org/10.48550/arXiv.2311.00877.

[218] M.P.C. Fossorier and Shu Lin. Soft-decision decoding of linear block codes based on ordered statistics. *IEEE Transactions on Information Theory*, 41(5):1379–1396, 1995. URL https://doi.org/10.1109/18.412683.

[219] Daniel Gottesman. Theory of fault-tolerant quantum computation. *Phys. Rev. A*, 57:127–137, 1998. URL https://link.aps.org/doi/10.1103/PhysRevA.57.127.

[220] Hasan Sayginel, Stergios Koutsioumpas, Mark Webster, Abhishek Rajput, and Dan E Browne. Fault-tolerant logical clifford gates from code automorphisms. *arXiv preprint arXiv:2409.18175*, 2024. URL https://doi.org/10.48550/arXiv.2409.18175.

[221] Anurag Anshu, Nikolas P. Breuckmann, and Chinmay Nirkhe. NLTS hamiltonians from good quantum codes. In *Proceedings of the 55th Annual ACM Symposium on Theory of Computing*, STOC 2023, page 1090–1096, New York, NY, USA, 2023. Association for Computing Machinery. URL https://doi.org/10.1145/3564246.3585114.

[222] Nikolas P. Breuckmann and Simon Burton. Fold-transversal clifford gates for quantum codes. *Quantum*, 8:1372, 2024. URL http://dx.doi.org/10.22331/q-2024-06-13-1372.

[223] Rui Chao and Ben W Reichardt. Fault-tolerant quantum computation with few qubits. *npj Quantum Information*, 4(1):42, 2018. URL https://doi.org/10.1038/s41534-018-0085-z.