Low-Overhead Quantum Fault Tolerance

Thesis by

Christopher Anand Pattison

In Partial Fulfillment of the Requirements for the
Degree of
Doctor of Philosophy

CALIFORNIA INSTITUTE OF TECHNOLOGY
Pasadena, California

2025
Defended May 16, 2025

Christopher Anand Pattison
ORCID: 0000-0003-0118-5257

To my friends
near and far

ACKNOWLEDGEMENTS

My time at Caltech and IQIM was an extremely special privilege. I am very grateful to my advisor, John Preskill, for making such an amazing environment with the freedom to explore and tackle big problems. It is only now that I realize how hard it is to find such a place and how fleeting the time is there.

The Simons Institute has been a continual source of theoretical tools and inspiration. I always come away from the many programs and workshops excited about something new.

I am thankful for my many friends, the list of which would not fit on this page, and from whom I derived the motivation to continue. I thank the many people who provided advice and inspiration early on in my graduate career including Nicolas Delfosse, Steve Flammia, Bailey Gu, Robert Huang, Alex Kubica, and Eugene Tang. I thank Harry Zhou, Dolev Bluvstein, Elie Batalle, Hannah Manetsch, Gyohei Nomura, and especially Phelan Yu for teaching me everything I know about AMO experiments.

Finally, I thank my closest collaborators—Sunny He, Anirudh Krishna, and Quynh Nguyen. Working together has been the highlight of graduate school.

ABSTRACT

Fault tolerance is an essential property of future quantum computers where a quantum computation is mapped to a new one that is resilient to operational errors. This resilience comes at an additional time and space overhead. In this thesis, we study schemes that asymptotically reduce the overhead of quantum fault tolerance in various models of computation.

In the first half, we construct a scheme for fault-tolerant quantum computation that requires nearly-logarithmic spacetime overhead assuming access to noiseless classical computation. This construction relies critically on the introduction of several new ingredients: we develop new qubit resource state distillation protocols with sub-logarithmic spacetime overhead used to perform teleported gates. We also construct a single-shot bit-flipping decoder to decode the almost-good quantum locally-testable codes of Dinur-Lin-Vidick where the quantum local testability is used crucially in order to prepare input states to the distillation. Finally, to assemble the substantial variety of gadgets, we introduce a new weight enumerator formalism that tracks the sets of jointly uncorrectable faulty spacetime locations using polynomials.

In the second half, we construct a quantum memory which has a threshold using a geometrically-local syndrome extraction circuit and almost-optimal parameters: the number of encoded qubits is nearly linear in the number of physical qubits, while the sub-threshold error suppression is nearly exponential in the number of physical qubits. Our construction surpasses known no-go results on the parameters of geometrically-local quantum codes by instead considering geometrically-local quantum circuits. The syndrome extraction circuits simulate the required long-range connectivity by performing a polynomial-depth permutation routing circuit with each qubit replaced by logarithmically-sized surface codes to retain a threshold.

PUBLISHED CONTENT AND CONTRIBUTIONS

[PKP25] Christopher A Pattison, Anirudh Krishna, and John Preskill. "Hierarchical memories: Simulating quantum LDPC codes with local gates". In: *Quantum* 9 (2025). C.A.P. conceived the project and participated in developing the key ideas and writing the manuscript., p. 1728. DOI: https://doi.org/10.22331/q-2025-05-05-1728. arXiv: 2303.04798 [quant-ph]. URL: https://arxiv.org/abs/2303.04798.

[NP24] Quynh T Nguyen and Christopher A Pattison. "Quantum fault tolerance with constant-space and logarithmic-time overheads". In: *arXiv preprint arXiv:2411.03632* (2024). Accepted at STOC'25. The author list is ordered alphabetically. All authors contributed equally. C.A.P. participated in developing the key ideas and writing the proofs and manuscript.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

Chapter 1

INTRODUCTION

The natural environment is extremely disordered. Thus, the ability to perform computation is a very unnatural task that requires us to bring a high degree of order to part of the physical world.

In addition to evidence that quantum computation is a stronger model for classical computational problems [Sho94], perhaps the most interesting usage of a quantum computer is to study physical systems as originally suggested by Feynman [Fey82]. Beyond simply simulating quantum systems, quantum computers could also allow us to directly interact quantum-mechanically with systems in nature which may unlock new experimental techniques for understanding the natural world [HKP21; Hua+22]. In this sense, a quantum computer is a novel tool to probe and understand the natural world.

In order to realize a quantum computer, it will be necessary for computations to be robust against errors in the fundamental operations. Quantum fault tolerance allows us to protect and process quantum information despite only having access to noisy quantum operations. It is the abstraction that simulates the computational model commonly used in quantum algorithms and complexity research of a noiseless quantum computer. Our main tool is a quantum error correcting code (or quantum code for short) which encodes logical information into code states which are locally indistinguishable. Error occurring on the code states are correctable with noisy quantum operations utilizing a procedure known as an error correction gadget as long as the error is not too "large."[1] The qubits of the quantum computation are encoded into the quantum code and manipulated indirectly such that errors are not spread and individual failures of circuit operations do not result in large errors.

In this thesis, we focus on low-overhead methods for quantum fault tolerance.

1.1 The difficulty of quantum fault tolerance

In classical computation, logic gates and memories are naturally robust to perturbations due to the ability to amplify classical signals. A stable memory cell based on

[1] The precise sense in which an error must be small depends on the fault-tolerance technique. One example is shown in Definition 2.3.1.

complementary metal–oxide–semiconductor (CMOS) logic can be formed simply by connecting two CMOS inverters back-to-back. Any fluctuation in the memory state (gate charge) is met by a restoring force that returns the system to one of the two fixed points.

In addition to the inherent stability of classical memories, it is also straightforward to perform fault-tolerant computation. Classical states are cloneable and distinguishable which permits the following simple strategy: Perform the computation in parallel on distinct processors. Then, periodically *copy* the computational state of each processor to the other processors. Finally, each processor *compares* the copies and replaces its own copy by the computational state held by the majority of the processors. Variants of this procedure are common in critical applications where failure would be disastrous [Avi67].[2]

Quantum fault-tolerance ends up being substantially more challenging than the classical analogue for a few reasons. With a binary alphabet $\{0, 1\}$, the only possible classical error taking a reference state to a fixed noisy is bit-flips. Quantumly, the set of operations is much larger; a noise operation may create or destroy entanglement in addition to applying bit or phase flips. The uncloneability of quantum information also makes the previously described classical fault-tolerance strategy unavailable to us

Due to entanglement, it is also not possible in general to compare individual parts of a quantum computation to a reference computation: for example, the Bell state $\Phi_+ = \frac{1}{2}(|00\rangle\langle 00| + |11\rangle\langle 11| + |00\rangle\langle 11| + |11\rangle\langle 00|)$ and the 2-qubit maximally mixed state $\rho = \frac{1}{4}(|00\rangle\langle 00| + |01\rangle\langle 01| + |10\rangle\langle 10| + |11\rangle\langle 11|)$ are distinct but indistinguishable given access to only one of the two qubits. This requires the introduction of much more burdensome and subtle definitions to prove correctness of individual computation components (gadgets).

Finally, it is a consequence of the Eastin-Knill theorem [EK09] that a quantum code with non-trivial distance cannot possess a universal gate set implemented by transversal operations. Thus, we must expend much more effort to augment the transversally-implementable gates with additional gates implemented by other means to complete a universal gate set. This in stark contrast to the classical case where only a single gate such as NAND is sufficient and implementable transversally.

[2] One should note that a simple bit-flipping description of faults in classical computer systems is not the full story. Failures in real systems may be as extreme as physical disruption of the equipment leading to indeterminate logic levels, short-circuits, electrical arcs, etc. We will not consider such failures here except to mention them in passing.

1.2 The state of quantum fault tolerance

Here, we briefly survey the current rapidly advancing state of quantum fault-tolerance in order to contextualize the results of this thesis.

Experimental advances

Despite being one of the oldest areas of quantum information, quantum fault tolerance is at a major transition point. We have only recently seen evidence [Ach+24] that the guarantees of the threshold theorems [KLZ96; Kit97; AB97] can be realized in real devices.

Such a major shift is exciting, but care must be taken to avoid concluding that the preconditions of the threshold theorems are satisfied exactly. As of writing, practical considerations dictate the sharing of single points of failure between different parts of the computation. Microwave and optical light sources, air handling, and even the physical location give rise to weak points for which failure violates the assumptions of the threshold theorems. A recent example is ionizing radiation [McE+22; Ach+24] that deposits large amounts of energy into superconducting devices. One might suspect that experiments with globally controlled gates such as neutral atom arrays [Blu+22; Blu+24] may also have such rare events although to date this has not been conclusively observed.

Fortunately, if the size of the error is under control, rare events may not be as significant of a problem as one might naively think. Surface codes [TPMP24] are naturally resilient to some limited amount of temporarily increased noise (an "error burst") as long as the noise strength and temporal duration are not too large. Other approaches such as concatenation [PKP25] may also help if the spatial extent of the error is not too large.

We are also now seeing recent advances in hardware-tailored quantum error correction. The usage of analog measurement information, known as "soft information" can aid the decoder in selecting an appropriate correction [PBSD21]. This soft information is now seeing use in quantum error correction experiments [Sun+23; Bau+24; Ali+24; Ach+24]. The rise of bosonic error correction [CMM99; GKP01; Mir+14] in experiments [Gri+20; Les+20; Siv+23; Put+25] has also recently motivated proposals for quantum error correction [Bon+21; Dar+21; Cha+22; Rui+25] that rely on highly-biased noise that can be engineered for these systems. We have also seen recent experimental proposals for superconducting and atomic platforms that attempt to convert error processes into erasures [WKPT22; Teo+23; Kub+23].

Theoretical advances

On the theory front, asymptotically-optimal quantum low-density parity check (qLDPC) codes have been constructed [PK21a] relatively recently. This concludes a long-standing open line of work to construct qLDPC codes with constant rate and linear distance. Given the resurgence of classical LDPC codes in the late 1990s [Gal62; Mac99], a research direction initiated in [MMM04] asks whether constant-rate qLDPC code families could be useful for quantum information processing tasks.

The construction of qLDPC codes with linear distance solved a major bottleneck in the practicality of constant-rate qLDPC code families that became apparent in the mid-2010s: the required block lengths of known constructions such as the quantum expander codes [TZ13; LTZ15] would be absolutely enormous [GK18; GGKL21] due to the distance scaling as the square-root of the block length. The line of work [HHO21; PK20; PK21b; BE21] leading to the construction of good qLDPC codes [PK21a] also led to new practically-relevant qLDPC codes[3] [PK21b; Bra+24; LP24].

However, it is still unclear whether one should use these modern qLDPC codes for *quantum computation.* We lack thorough understanding of the features that lead to good parameters, and often we only have heuristic decoders such as belief-propagation with ordered-statistics decoding (BP+OSD) post-processing [PK21b]. While there have been initial steps towards decoders with provable guarantees for some code families [GPT22; LZ22a; DHLV22; Gu+23], we still lack the same degree of evidence as the surface code that a qLDPC memory can be operated with extremely low logical error rates using a realistic amount of classical computation resources.

Logical gates

The construction of low-overhead logical gate gadgets also remains an active direction of research. For quantum codes possessing only one logical qubit and transversal implementations of the full Clifford group such as the color code [BM06], only one additional gate such as T is required to complete a universal gate set. The situation becomes much more complicated as the number of logical qubits per block k increases due to the "gate targeting problem." Suppose that we are interested in implementing a Clifford circuit and have access to only Clifford and Pauli gates at

[3] Such codes are not necessarily part of an asymptotic family.

the physical level. The canonical construction of transversal gates[4] applies the same gate to all qubits in the block (coordinate-wise), and there are only a constant number of distinct one and two-qubit gates. How then will we obtain enough ($\sim \log k$) generators of the Clifford group in order to implement an arbitrary Clifford circuit?

One option is to prepare resource states and teleport gates as in [Kni05; Got13; NP24]. This requires the least number of assumptions on the code but results in large practical overheads that are likely unacceptable. If we are to utilize modern qLDPC code constructions, less general approaches with lower overheads will be necessary.

The next less-general approach is a type of code switching known as lattice surgery [HFDV12] which has recently seen a long string of progress for modern code constructions [CKBB22] and further works. [5] Lattice surgery combined with generalizations of transversal gates that use constant-depth circuits such as automorphism and fold-transversal gates [Mou16; QWV23; BB24; Say+24] is potentially simple enough to yield low fault-tolerance overhead in practice.

Another promising avenue[6] to solve the gate targeting problem is when the quantum code possess a (logically) transversal CCZ that acts as $\prod_{i=1}^{k} \mathsf{CCZ}(i, i+k, i+2k)$ in some basis. Then, utilizing the fact that stabilizer codes have a weakly-transversal[7] implementation of any logical Pauli operator, one can conjugate an arbitrary pattern of logical X gates by CCZ in order to perform an arbitrary (coordinate-wise) pattern of logical CZ gates using the identity $\mathsf{CCZ}(1,2,3)\mathsf{X}_1\mathsf{CCZ}(1,2,3) = \mathsf{CZ}(2,3)$. Then, all that remains is to "connect" the different coordinates and complete a universal gate set (e.g. with Hadamard). If the code possesses symmetries such as obtained from the lifted product or generalized bicycle construction with an Abelian group [PK21b; KP13b; LP24], then automorphism gates that permute the logical qubits within a block may be available. Obtaining Hadamard is somewhat more difficult. One might attempt to obtain pairs of codes for which code switching is possible [KB15] or just simply distill the stabilizer state $|\mathsf{H}\rangle = \mathsf{H} \otimes \mathsf{I}\,|\Phi\rangle$. $|\mathsf{H}\rangle$ can be used to teleport H and its distillation is substantially easier than distilling magic states (see Section 2.4).

[4] For example, for two code blocks of length n, transversal CNOT would be $\prod_{i=1}^{n} \mathsf{CNOT}(i, i+n)$.

[5] [HJY23; ZL25; HJY23; Xu+24b; IGND24; WY24; CB24; SJY24; CHWY25; CHRY24; HCWY25]

[6] I thank Anirudh Krishna for pointing this construction out to me in 2023.

[7] A code is said to have a weakly-transversal implementation of a gate if the implementation has the form $\otimes_{i=1}^{n} V_i$ for some unitaries V_i that are supported on only one qubit each of the code block.

Long-range connectivity

Intentionally omitted from the prior discussion is the compatibility of new fault tolerance schemes with experimental connectivity constraints. Connectivity was a consideration nearly as early as the first threshold results [AB97; Got00]. Fortunately, code families [Kit97; BK98; DKLP02] naturally compatible with a 2D layout (the surface code) were constructed very early on.

Constraints 2D layouts are very natural for hardware platforms where qubits are fabricated on a chip such as superconducting qubits since the devices and couplers need to be physically placed. If one restricts themselves to a 2D layout, what constraints exist on quantum fault-tolerance schemes? The well known BPT bound [BPT10] shows that any quantum code encoding k logical qubits into n qubits defined as the ground space of a 2D-geometrically-local commuting-projector Hamiltonian[8] has distance d bounded as $kd^2 = O(n)$, asymptotically no better than the surface code. It also turns out that the BPT bound is resilient to the addition of a limited number of non-geometrically-local stabilizer generators [BK21a; BK21b]. In fact, [BK21a] implies that the Tanner graph of a good qLDPC code family must be an *expander graph.* This implies that for every subset of qubits S of less than half the block length, the number of stabilizer generators supported on both S and its complement S^c is proportional to the volume $|S|$.

One might ask whether such results preclude the implementation of a quantum memory, not necessarily defined by geometrically-local checks, with logical error rate p_L better than the surface code, $p_L = e^{-\omega(\sqrt{n/k})}$. The correct object to bound is the error-correction gadget which measures the syndrome and applies some correction after some classical computation. Consider a 2D-geometrically-local error-correction gadget of width W and depth D maintaining a quantum memory encoding k logical qubits. [DBT21] established a trade-off for the depth and number of ancillas required for error-correction gadgets that measure the syndrome of a constant-rate qLDPC code family with polynomially growing distance. In the case where the ancilla is minimal, they find that $D = \Omega(\sqrt{n})$. Separately, [BFS23] established that, for logical failure rate p_L per application of the error-correction gadget, the width of the circuit is lower bounded as $W = \Omega\left(\frac{k}{D}\sqrt{\log \frac{1}{p_L}}\right)$. For a good qLDPC code family, one would expect something like $\log \frac{1}{p_L} \sim n$, so that the bound can be interpreted as implying

[8] It is possible to do slightly better when the code is geometrically local on a hyperbolic manifold [Del13].

the depth lower bound $D = \Omega(\sqrt{n})$ for a very broad class of circuits.

We will show in Chapter 2 that these bounds do not preclude a memory maintainable by geometrically-local quantum operations with nearly optimal rate and error suppression at the cost of a time slowdown.

Hardware solutions If these more recent fault-tolerance schemes require long-range gates, perhaps we should try to add them to the experiment? There has been considerable interest in recent years to add long-range connectivity to platforms that do not natively realize it. This has been made more palatable by the reduction in qLDPC code block lengths enabled by recent constructions with better distance scaling; in 2D, a 2D embedding of an expander graph with n vertices has $\Theta(n)$ edges with length $\Theta(\sqrt{n})$. Thus, a reduction of the block length represents a substantial reduction in the amount of effort required to utilize fault-tolerance schemes based on constant-rate qLDPC code families.

In particular, suppose that one would like to implement a polynomially-sized quantum computation on W qubits that is already 2D-geometrically local (to avoid additional complexity from *logical* embedding considerations). Roughly, if the code family has exponential error suppression in the block length, then code blocks of size at most polylog W are required, so that the experimentalist only needs to implement gates of length at most $O(\text{polylog}\, W)$ — substantially smaller than the overall size of the computation. For general computations, we would also like to avoid incurring a large time-overhead from logical embedding of the circuit, but logical-level operations have substantially more flexibility.

Thus, while it is essentially asymptotically impossible[9] to embed the required expander graph into flat 3D space with unit density of qubits and connections, we might hope to accomplish the required amount of long-range connectivity in the practical regime. Initial progress suggests that this may be the case although much work needs to be done.

In superconducting qubit platforms, there have been some schemes developed to achieve long-range connectivity [McK+15; Zha+] and notably some industrial efforts [Bra+24]. Fortunately, it is often the case that a direct coupler does not have to be created: many platforms have qubits that can be physically transported

[9] If we enforce a limit on the amount of energy per unit volume, then there is an upper bound to the amount of quantum information that can be stored per unit volume. This also implies an upper limit in the amount of entanglement that can be mediated per unit volume.

without coupling to the qubit. Photonic platforms [Ale+25; Agh+25] naturally realize long-range connectivity due to the propagating nature of photons. In platforms where the qubit is encoded in states of a trapped system such as ions [KMW02; Hen+06; Pin+21], neutral atom arrays [Beu+07; Blu+22; Xu+24a], and silicon spin qubits [Mil+19], the trapping potential can often be shifted to physically transport the qubit on timescales shorter than the coherence time.

1.3 Contributions and outline

This thesis comprises two chapters. The first section of each chapter contains the main results and high-level proof ideas. Here, we briefly summarize the main challenges and context leading to the results of each chapter.

Chapter 2 In the first chapter, we will prove a threshold theorem for fault-tolerant quantum computation with asymptotically-lower overhead than previously known to be achievable. In our model of computation, no connectivity constraints are imposed, and classical computation is assumed to be perfect. For convenience, the classical computation will be assumed to run slightly faster than the quantum computation. For a quantum circuit of size V, we permit a polyloglog V depth classical circuit between each quantum timestep.[10] To separate the unitary synthesis cost, we assume that the input circuit is composed of one and two-qubit Clifford gates and CCZ gates.[11]

We consider a model of noise where noise channels are inserted into the circuit at spacetime locations L according to some distribution. Such spacetime locations are deemed to be "faulty." The distribution is said to be ϵ-locally stochastic if, for all subsets $S \subseteq L$ of spacetime locations, the probability that the set of faulty locations contains S is at most $\epsilon^{|S|}$. This is a weakening of independent noise that permits a limited amount of noise correlations. We will require that the set of faulty locations is ϵ-locally stochastic for a noise strength $\epsilon \in [0, \epsilon_*]$ that is at most some constant ϵ_*.

Given a quantum circuit C of width W and depth D and a desired simulation accuracy $\epsilon_L \in (0, 1)$, we can map C to a new circuit $\overline{C}$ such that $\overline{C}$ subject to an ϵ-locally stochastic fault distribution with $\epsilon \in [0, \epsilon_*]$ simulates C in the sense that the output distributed is ϵ_L-close in total-variation distance to that of C. Furthermore, when

[10] We note that this restriction is not fundamental and can be removed by adding rounds of a constant-depth "idling" error correction gadget that maintains the memory while any necessary classical computation finishes. This can be constructed using a single round of syndrome extraction and a constant number of decoding iterations of the bit-flipping decoder.

[11] T gates can also be used at an additive $\log V$ cost.

$\frac{WD}{\epsilon_L}$ is less than[12] a particular function $f(W)$ of the width growing faster than any quasipolynomial of W, $\overline{C}$ has depth $O(D)$ and width $O\left(W \log^{1+o(1)} \frac{WD}{\epsilon_L}\right)$.

We construct $\overline{C}$ in two main steps. Let $V = \frac{WD}{\epsilon_L}$ be the "spacetime volume" of the computation. In the first step, containing the majority of the work, we encode all qubits into the quantum-locally testable (qLTC) codes of DLV [DLV24]. We perform distillation to make prepare resource states and then apply logical gates via a teleportation gadget. This first circuit, the outer circuit, requires a noise that is less than a threshold value that is slightly vanishing as $\frac{1}{\text{polyloglog}\, V}$. In the second step, we amplify the threshold to a constant value by simulating the outer circuit with concatenated quantum Hamming codes [YK22] [13] such that the outer circuit experiences an effective error rate $\frac{1}{\text{polyloglog}\, V}$. This step retains the constant space overhead but blows up the time overhead by a factor quasipolylogloglog(V).

The origin of the two-step procedure originates from the choice of decoder for the DLV codes [DLV24]: The usage of the DLV codes required the construction of a decoder which is omitted from this thesis due to length considerations (see the full paper [NP24]). Our decoder is an small-set-flip style decoder against adversarial-noise with measurement errors that runs in parallel. This is analogous to those previously constructed for good qLDPC code families [GPT22; LZ22a; DHLV22] and in particular [Gu+23].

Unfortunately, the DLV code family possess distance that is slightly sub-linear in the blocklength, so a decoder for adversarial noise is *a priori* insufficient for stochastic noise. Fortunately, code concatenation and its computation analogue, recursive simulation, allows us to reduce the stochastic noise setting to the adversarial one. Given the features of the decoder shared with stochastic-noise decoders for quantum expander codes [FGL18b], it is reasonable to expect that a stochastic noise decoder is possible to construct at the cost of more involved analysis.

In order to manage the considerable complexity of the different gadgets, we introduce a new formalism inspired by the fault path counting in the original quantum fault-tolerance proofs [AB97; Kit97] that allows us to assemble a full circuit and prove correctness against adversarial noise (dramatically simplifying formal statements). This is managed by associating families of uncorrectable "bad" fault paths to polynomials which we refer to as *weight enumerators*. The weight enumerator polynomials correctly captures cases such as state injection where there are many

[12] This is to avoid extremely deep circuits. See the formal theorem statement Theorem 2.3.23.

[13] Used by itself, this scheme has constant space overhead and quasipolylog V time overhead.

	Space	Time
[AB99; Kit97; KLZ98; AGP05]	polylog(V)	polylog(V)
[Got13; FGL18a]	$O(1)$	poly(V)
[YK24]	$O(1)$	quasipolylog(V)
Chapter 2	$O(1)$	$\log^{1+o(1)}(V)$

Table 1.1: Comparison of quantum fault-tolerance threshold theorems ignoring differences in classical resources.

ways for the procedure to fail with different multiplicities and weights.[14] The conversion to stochastic noise then only requires the evaluation of the polynomials at the noise rate of the error model.

We conclude with a comparison to prior threshold theorems ignoring differences in the model of classical computation. For convenience, we assume that the computation size V is polynomial in the number of qubits W. Parallel work [TKY24] establishes a threshold theorem with $O(1)$ space overhead and polylog V time overhead.

Chapter 3 In the second chapter, motivated by experimental constraints, we will examine to what extent 2D-geometrically-local circuits can maintain information stored in qLDPC codes. We find that, given a constant-rate qLDPC code family with distance d_n scaling polynomially with the block length n, we can construct a family of codes and 2D-geometrically-local syndrome extraction circuits of width W, known as Hierarchical Codes, for which there is a memory threshold.

By instantiating the construction with asymptotically-good qLDPC code families, we construct family of codes and 2D-geometrically-local syndrome extraction circuits with a memory threshold for which the rate and distance are nearly optimal irrespective of connectivity constraints: below the threshold, the logical error rate per error-correction gadget falls as $p_{L,\mathrm{HC}} = e^{-\Omega(W/\mathrm{polylog}\, W)}$ per syndrome extraction cycle, and the rate goes as $\frac{1}{\mathrm{polylog}\, W}$.[15]

To compare with other 2D-geometrically local schemes, we can replace the distance d by a function of the logical error rate per error-correction gadget $\log \frac{1}{p_L}$. These parameters can be thought of as roughly equivalent for the surface code and Hierarchical Code (up to log factors). The surface code saturates the BPT bound and

[14] In surface codes, this corresponds to the phenomenon of preparing an arbitrary encoded state with constant error probability [DKLP02] where there are a few uncorrectable short error chains but many uncorrectable long error chains.

[15] A straightforward recursive application of the construction achieves a rate vanishing as slowly as $\frac{1}{e^{O(\log^* W)}}$ where $\log^*$ is the iterated logarithm.

the BFS bound [BFS23] with a constant depth circuit by encoding k_{SC} logical qubits into n_{SC} physical qubits with a logical failure rate per round of $p_{L,\mathrm{SC}}$ satisfying

$$k_{\mathrm{SC}} \left(\log \frac{1}{p_{L,\mathrm{SC}}} \right)^2 = \Theta\left(n_{\mathrm{SC}}\right) \ .$$

On the other hand, the Hierarchical Code uses a error-correction gadget with W physical qubits and depth $\sqrt{W}$ and encodes $k_{\mathrm{HC}} = \Omega\left(W/\mathrm{polylog}\, W\right)$ logical qubits with failure rate per syndrome extraction cycle $p_{L,\mathrm{HC}}$ satisfying

$$k_{\mathrm{HC}} \left(\log \frac{1}{p_{L,\mathrm{HC}}} \right)^2 = \Omega\left(\frac{W^3}{\mathrm{polylog}\, W} \right) \ .$$

Thus, by considering geometrically-local *circuits* instead of codes, one can do far better than one might otherwise expect from the BPT bound. On account of the $\sqrt{W}$ depth of the error-correction gadget, the Hierarchical Code saturates known bounds on syndrome extraction circuits [DBT21; BFS23] up to $\mathrm{polylog}(W)$ factors.

The natural next question is whether the construction possesses implications for practical applications. Preliminary estimates [PKP25] suggest that the construction may be practical in some regimes although much further study is required. The concatenated structure of the code can be used to aid decoding by outputting a so-called "soft output" [MPP24] from the inner surface code decoder. [GNBJ23] provides strong numerical evidence that, more generally, a concatenation strategy with an inner surface code can reduce the space overhead in the non-asymptotic regime.

1.4 Outlook

Asymptotically, we have shown that the overhead of quantum fault-tolerance can be quite small: The quantum fault tolerance scheme of Chapter 2 shows that the overhead of quantum fault tolerance can be brought down to be essentially logarithmic. Chapter 3 shows that the space savings of constant-rate qLDPC memories can nearly survive the addition of hardware constraints at the cost of a spacetime blowup.

However, it is unclear to what extent the asymptotic resource savings can be transferred to the practical regime. On the theory side, practical codes, decoders, and gate schemes remain open topics of research. On the hardware side, it is still unclear what a large-scale hardware device will look like: methods for implementing long range connectivity are being developed and different hardware platforms have substantially different tradeoffs and noise models.

Overall, constant-rate qLDPC code families are a promising avenue to reduce the overhead of quantum fault-tolerance for interesting quantum computations but much work remains.

Chapter 2

QUANTUM FAULT TOLERANCE WITH CONSTANT-SPACE AND ALMOST LOGARITHMIC-TIME OVERHEADS

[NP24] Quynh T Nguyen and Christopher A Pattison. "Quantum fault tolerance with constant-space and logarithmic-time overheads". In: *arXiv preprint arXiv:2411.03632* (2024). Accepted at STOC'25. The author list is ordered alphabetically. All authors contributed equally. C.A.P. participated in developing the key ideas and writing the proofs and manuscript.

The fundamental nature of fault tolerance prompts the natural question: "what is the lowest possible overhead?" In this chapter, we show that a quantum circuit with 2-qubit Clifford gates and CCZ can be simulated using nearly logarithmically more resources when permitted access to noiseless auxiliary classical computation.[1] This construction is unfortunately asymptotic in nature, but we hope that certain parts of the construction will be useful for practical applications. We also expect that pieces of our construction will find applications in quantum computational complexity theory to answer questions such as whether noisy low-depth quantum circuits are as powerful as their noiseless counterparts.

Our result required the introduction of several new ingredients compared to the prior state of the art.

1. We develop a magic state distillation protocol with $(\log \frac{1}{\varepsilon})^{\gamma}$ spacetime overhead, where $\gamma \to 0$ as $\varepsilon \to 0$. This result differs from recent works in that we use a simple and self-contained construction using Reed-Solomon codes to obtain low *spacetime* overhead (rather than just space overhead as in recent works). We show a similar result for stabilizer state distillation.

2. We prove that quantum codes based on the cubical complex construction of [DLV24] admit sequential and parallel single-shot decoders against adversarial errors of weight scaling with the code distance. In particular, our proof applies

[1] In this version, the classical computation is permitted to run slightly faster than the quantum circuit. This is not a fundamental limitation, but it does make the analysis slightly simpler.

to a recent family of almost-good qLTCs of Dinur-Lin-Vidick and the good qLDPC codes of Dinur-Hsieh-Lin-Vidick.[2]

3. Using the introduced distillation schemes, we develop fault-tolerant logical state preparation procedures with $\widetilde{O}(1)$-spacetime overhead on qLTCs. Here, the qLTC property is used to quickly test if a state is too far from the codespace before proceeding.

4. We introduce the use of multiple resource states (from a small-sized set) to obtain addressable qLDPC logic that can be easily prepared using our state preparation schemes and transversal qLDPC gates.

We obtain the main result by combining the above results with carefully chosen parameters. In doing so, we introduce a new *weight enumerator formalism* to prove fault tolerance in a composable way, which is of independent interest. To our knowledge, this gives the lowest spacetime overhead to date in the considered model of quantum fault tolerance, which, for the first time, matches that of classical fault tolerance up to sub-polylogarithmic factors.

An outline of the proof with hyperlinks is given on page 30.

2.1 Introduction

In the following section, we give the relevant background and informally summarize the main result.

Background and main result

The optimal overhead for fault-tolerant computation is a fundamental question in computer science and engineering. In the setting of classical computation, this question was essentially settled in the 1990s. [PST91; GG94] showed the existence of a Boolean function whose noiseless circuit complexity is $|C|$ but requires $\Omega(|C| \log |C|)$ noisy gates to compute. This lower bound is met by the original fault tolerance scheme in von Neumann's lectures that initiated the study of fault-tolerant computation [Von56].

In the setting of quantum computation, this question is largely open and appears to be much more challenging. On one hand, existing rigorous proofs of fault tolerance for quantum circuits are often very complicated. This complexity can be attributed to the

[2] This is deferred to the full paper [NP24]

inherent subtleties of quantum information that causes new technical and conceptual challenges [Got10], such as in conditioning on a subsystem, handling entangled ancillas, tracking propagation of quantum errors, dealing with the lack of universal transversal gate sets [EK09], etc. As such, there has only been a handful of rigorously proven end-to-end quantum fault tolerance protocols [AB99; AGP05; YK22]. On the other hand, lower bounds for the fault tolerant overhead in the quantum case are also naturally expected to be more challenging. The currently best lower bound on quantum fault tolerance overhead is from the work of [FMS22] who derived a space lower bound of $Q^{-1}n$, where Q is the quantum capacity of the noise channel[3]. To our knowledge, there is currently no spacetime lower bound that is better than linear.

Motivated by practical implementations of quantum computers, a commonly studied model of fault-tolerant quantum computation is one in which a noisy quantum circuit is executed with the help of a (small-sized) noiseless classical computation. A metric of high interest is the space overhead, i.e., the number of physical qubits used at any point in time of the quantum computation. In his seminal work [Got13], Gottesman initiated the study of constant-space-overhead quantum fault-tolerance under the assumption that a large auxiliary classical computation is free and noiseless. The works of [FGL20] further relaxed the auxilliary classical computation time to polylogarithmic. Here, we assume that an even smaller, polyloglog-time classical computation is used per quantum timestep.

Definition 2.1.1 (Quantum fault tolerance model, simplified)**.** *Let C be a Clifford+*CCZ *quantum circuit of depth D and space W. For every timestep of the quantum circuit, we run a noiseless classical circuit on the measurement history and a classical memory of depth $O(\text{polyloglog}(WD))$. The quantum operations in the following timestep are permitted to be classically controlled on individual output bits of the classical circuit.*

In this work, we place no restrictions on the connectivity of the operations. Our noise model is the commonly used locally-stochastic noise: the set of faulty gates F satisfies that, for any subset of gates S, the probability the F contains S is exponentially small $\Pr(S \subseteq F) \leq p^{|S|}$ for some p that should be thought of as the noise rate. Faulty gates are permitted to apply any CPTP noise channel after the gate.

The goal of the fault-tolerant circuit is to produce samples from a distribution that is

[3]This bound also applies for the computation model considered in this work where small-sized noiseless classical computation is assumed.

ε-close in total-variation distance (TVD) to the output distribution of the noiseless quantum circuit.

The question we study in this paper is: *what is the lowest time overhead possible while maintaining a constant space overhead in the above model?* This question is of both practical and theoretical interest. The model of quantum fault tolerance considered here has been the conventional model studied in the field, stemming from the fact that classical computation is practically perfect with current technology. However, we believe that studying the overhead of fault tolerance in this model could lead to ideas to answer the same question in the fully-quantum model. Previous to the initiation of this work, the state-of-the-art result in this line of work is due to Yamasaki and Koashi [YK24], who constructed a protocol based on concatenated quantum Hamming codes. They achieved a quasipolylog-time overhead – which is a function growing faster than $\log^{\alpha}(|C|)$ for any constant α.

The above discussion leads us to our main result. (see Theorem 2.3.23 for the full formal statement)

Theorem 2.1.2 (Main result, informal)**.** *Given any (arbitrary-connectivity) quantum circuit C on W qubits, of depth D, and composed of $|C| \leq f(W)$ gates from the Clifford+CCZ gate set, where f is a function growing faster than any quasipolynomial function, we can efficiently construct a circuit C_{FT} with the following guarantees. C_{FT} uses at most $O(W)$ physical qubits at any point in time. The quantum depth of C_{FT} is $D \times O(\log^{1+o(1)} \frac{|C|}{\varepsilon})$, and the (noiseless) auxilliary classical computation time used per quantum timestep is $O(\text{polyloglog} \frac{|C|}{\varepsilon})$. There exists a constant noise threshold p_*, such that when executed under the local stochastic noise model with noise rate $p < p_*$, C_{FT} outputs a distribution which has total-variation distance ε from the output distribution of C.*

Above, we assume the ideal circuit is composed of gates from a fixed local gate set. This assumption is simply to remove a potential polylog depth overhead in decomposing gates according to the Solovay-Kitaev theorem. Notably, the stated time overhead holds even though we are making no other assumptions about the gate connectivity and specific patterns of gate types in the ideal circuit. In fact, our fault tolerance scheme simulates any logical gate in a time overhead of at most $O(\log^{o(1)} \frac{|C|}{\varepsilon})$, and the extra logarithmic factor arises from potentially adversarially selected connectivity in the simulated circuit. The $o(1)$ exponent roughly scales as $1/\text{logloglog}(|C|/\varepsilon)$. We can also smoothly tune the overheads: the space overhead

can become $\widetilde{O}(\log \frac{|C|}{\varepsilon})$ and the time overhead can become as low as a constant, while keeping the spacetime overhead $\widetilde{O}(\log \frac{|C|}{\varepsilon})$ (see Section 2.1).

Overview of ideas and technical results

The result is based on multiple main ideas and new technical results which we overview below. To keep track of less variables, in the rest of this introduction we take the target error ε to be a constant as well as assume $|C| = \text{poly}(W)$.

Revisiting Gottesman's protocol

Our starting point is an observation that a modification of Gottesman's protocol using qLDPC codes [Got13; FGL20] suffices to obtain a time overhead of $\text{polylog}(|C|)$. Let us recall the protocol as-instantiated by Fawzi, Grospellier, and Leverrier in [FGL20, arXiv version section 2.5].

The protocol partitions the W qubits of the circuit into $\ell = \text{polylog}(W)$ registers of size $K_{\text{FGL}} = W/\ell$. Each register is then encoded in a quantum expander code [LTZ15] with parameters $[[N' = \Theta(K_{\text{FGL}}), K_{\text{FGL}}, D' = \Theta(\sqrt{K_{\text{FGL}}})]]$. The fault-tolerant circuit C' alternates between logical operation cycles and error correction (EC) cycles. The EC cycle uses an efficient decoder shown to be single-shot (robust against measurement errors), so the EC cycle can be performed in $O(1)$ quantum depth.

To perform logical gates, all gates in C are serialized so that only one gate is applied per cycle, incurring a $O(K_{\text{FGL}})$ factor in the time overhead. Each logical gate cycle fault-tolerantly simulates the corresponding logical gate using gate teleportation [GC99]. The main challenge is to prepare the necessary resource states required for teleportation. Fortunately, it suffices to use standard concatenated-code fault-tolerance schemes [KLZ96; AB99; Kit97] as in [Kni05]. The idea is that the concatenated-code protocol can serve as a general-purpose factory of ancilla states despite the additional space cost.[4] In particular, to prepare the required $O(N')$-qubit resource states to error $\varepsilon' = 1/\text{poly}(|C|)$, concatenated-code fault-tolerance yields $N' \,\text{polylog}(N'/\varepsilon') = N' \,\text{polylog}(W)$ spacetime cost. Hence, choosing $\ell = \text{polylog}(W)$ ensures that only $\Theta(W)$ qubits are used at any point in the fault-tolerant circuit. After some careful optimizations, the time overhead can be shown to be $O(W)$ with $O(1)$ space overhead.

[4] Strictly speaking, we were not able to find a proof of this important lemma (in particular, the unencoding step of the concatenated-code to obtain the physical state) in the literature. Thus, we give a proof of it in this paper. A very recent work [CFG24] also gives a proof.

We first observe that the choice of code block size and serialization described above is suboptimal. In particular, it suffices to partition W qubits into $W/\mathrm{polylog}(W)$ blocks of $K' = \mathrm{polylog}(W)$ qubits each: the logical error rate of a length-N' quantum expander code is roughly $2^{-\Theta(\sqrt{N'})}$ per EC cycle, so a blocksize of $N' = \mathrm{polylog}(W)$ suffices to maintain a low overall logical error rate of the computation.[5] Next, instead of serializing all gates in C, we only need to rearrange them such that there is at most one gate acting on a code block in each circuit layer. This task corresponds to an edge coloring problem on a degree-$O(K')$ multigraph, which can be done with $O(K') = \mathrm{polylog}(W)$ colors by a generalization of Vizing's theorem [BF91]. Hence, the serialization only incurs a $\mathrm{polylog}(W)$ time overhead.[6] Logical gates can be performed the same way as described earlier, leading to another $\mathrm{polylog}(W)$ spacetime overhead.

After further gate scheduling and serialization, one can obtain constant-space and $\mathrm{polylog}(W)$-time overheads. This optimization will be superceded by our construction, so we do not present the details. We expect this to be a large constant (say, at least 4) with the dominant contributor being the concatenated-code overhead. Hence, new ideas are required to bring this down to the optimal exponent, which we conjecture is 1 as in the classical case.

Motivated by the fundamental question of the optimal overhead of fault-tolerant computation, we reduce the exponent in the time overhead to $1 + o(1)$. In particular, we will reduce the time overhead of all logical operations to $\log^{o(1)}(W)$ while keeping the space overhead a constant. The remaining $O(\log W)$ time overhead factor originates from the serialization using edge coloring similar to described above, which we conjecture is inherent – see the discussion in Section 2.1. This leads to the claimed main result in Theorem 2.1.2.

We next overview the new ideas and technical contributions required to achieve this goal.

[5] If we instead use recent good qLDPC codes [PK22; LZ22b], we can choose a code block size of $O(\log |C|) = O(\log W)$. These codes are also known to have efficient single-shot decoders [Gu+23].

[6] Alternatively, one could use the ability to prepare arbitrary resource states to perform all the gates acting between qubits encoded in each pairs of blocks in parallel. For M blocks, there are $O(M^2)$ pairs of blocks to execute gates between, so such an optimization gives a time overhead of $O(M^2 \,\mathrm{polylog}\, W)$. This is a $O(\mathrm{polylog}\, W)$ time overhead for the original block length choice in [Got13].

Better overheads with distillation

The first improvement is to remove the polylog(W) spacetime overhead arising from resource state preparation using concatenated-code fault-tolerance. In particular, this factor is polylog(N'/ε'), where N' is the qLDPC block size and ε' is logical error rate of the simulated gate. As seen earlier, [FGL20] had the parameters $N' = W/\text{polylog}(W)$ and $\varepsilon' = 1/\text{poly}(W)$, leading to the polylog$(W)$ scaling. With our new instantiation, $N' = \text{polylog}(W)$ (our final choice will be $N' = O(\log W)$). However, the choice $\varepsilon' = 1/\text{poly}(W)$ seems unavoidable – after all, we need this logical gate error rate in order to sustain a poly(W)-sized computation. Our idea to circumvent this obstruction is to use state distillation protocols [BK05; Kni05] in combination with concatenated-code FT.

More specifically, we first use concatenated-code FT to prepare the resource state to only a sufficiently small constant error rate ε_0. This step only incurs a polyloglog(W) spacetime overhead. The second step is to use a state distillation protocol to suppress this error to the target $\varepsilon' = 1/\text{poly}(W)$. These protocols often employ a quantum code, the 'distillation code,' with a transversal gate related to the distilled resource state. In order to lead to an improved overhead, the protocol must have a favorable ratio of noisy input states to good output states, and the the protocol must not have too high of a gate depth. The second of these considerations depends on the distillation code encoding/unencoding circuit depth and classical decoder efficiency. Towards this, we prove the following two new results on distillation overheads that are of independent interest:

Theorem 2.1.3 (Stabilizer state distillation, informal)**.** *Let $|\psi\rangle$ be a $O(1)$-qubit stabilizer state. There exists a distillation protocol $\mathsf{SSD}(\varepsilon)$ that uses noiseless CNOT, Pauli operations, Pauli measurements and gives the following guarantees. There exists a constant noise threshold ε_{SSD} such that, provided with N (independent) noisy states whose infidelity with $|\psi\rangle$ is $\varepsilon_{\text{in}} < \varepsilon_{\text{SSD}}$, where $N > N(\varepsilon_{\text{SSD}})$ is a sufficiently large number, $\mathsf{SSD}(\varepsilon)$ produces $\Theta(N)$ states each of which has infidelity at most ε with $|\psi\rangle$. Furthermore, the quantum depth and classical depth of $\mathsf{SSD}(\varepsilon)$ are both* polyloglog$(1/\varepsilon)$.

Theorem 2.1.4 (Magic state distillation, informal)**.** *Let $|\mathsf{CCZ}\rangle = \mathsf{CCZ}\,|+\rangle^{\otimes 3}$. There exists a distillation protocol (using noiseless Clifford operations and measurements) $\mathsf{MSD}(\varepsilon)$ and a constant noise threshold ε_{MSD} such that the following holds. Upon receiving N (independent) noisy states whose infidelity with $|\mathsf{CCZ}\rangle$ is $\varepsilon_{\text{in}} < \varepsilon_{\text{MSD}}$, where*

$N > N(\varepsilon_{\mathrm{MSD}})$ is a sufficiently large number, $\mathsf{MSD}(\varepsilon)$ *produces* $\Theta(N/\log^{o(1)}(1/\varepsilon))$ *states each of which has infidelity at most* ε *with* $|\mathsf{CCZ}\rangle$. *Furthermore, the quantum depth and classical depth of* $\mathsf{MSD}(\varepsilon)$ *are both* $\mathrm{polyloglog}(1/\varepsilon)$. *Here, the* $o(1)$ *exponent scales as* $O(1/\mathrm{logloglog}(1/\varepsilon))$.

These results are proved in Section 2.4 and Section 2.5, respectively. The 'batch sizes' N_{SSD} and N_{MSD} are roughly $\mathrm{quasipolylog}(1/\varepsilon)$. For the first result, we are not aware of any prior results on stabilizer state distillations of this type. We suspect that this is because previous works often consider quantum codes with transversal Cliffords and hence do not run into this question. In our case, and generally for the recently developed qLDPC code families, we often have a limited set of transversal gates and thus require other fault-tolerant Clifford gate techniques such as lattice surgery [CKBB22]. Here, we opt to gate teleportation and thus need to derive the stated result. In fact, we will even use this distillation procedure to distill computational basis states.

On the other hand, the magic state distillation (MSD) question has been more intensively studied, starting from [BK05; Kni05]. Our MSD overhead result gives a yield rate of $\log^{o(1)}(1/\varepsilon)$, improving over a prior work [HH18] that achieved an exponent of ≈ 0.68.[7] The time overhead (including both quantum and classical) is only $\mathrm{polyloglog}(1/\varepsilon)$, which is crucial to obtain our main result but often disregarded in the MSD literature. The slightly super-constant MSD space overhead is not an issue, as appropriate gate scheduling can turn this into a time overhead. We note that concurrent and independent works [WHY24; Ngu24; GG24] have established that constant-space-overhead MSD is possible (see the explicit scheme in [WHY24]), however we believe the time overhead of these protocols is at least $\mathrm{polylog}(1/\varepsilon)$, arising from the unencoding circuit of the large distillation code. Hence, our protocol gives a better *spacetime* overhead (with an arguably simpler construction) which is necessary for the main result.

Applying the above distillation schemes on the logical level of a high-rate qLDPC code is not necessarily straightforward, as we need to address specific logical qubits in a large qLDPC code block with multiple logical qubits. We show how to use the above results to distill important resource states for logical operations on high-rate qLDPC codes in Section 2.4. Here, the key observation is that the noiseless operations required in the SSD scheme can be implemented transversally on the computation

[7] Stronger exponents are known for p-dimensional qudits with $p > 2$ an odd prime [KT19].

qLDPC CSS code. Once we have distilled qLDPC logical stabilizer states, we can then use them to run the MSD scheme as well. To our knowledge this is the first time this is explicitly worked out, with a rigorous fault tolerance proof.

Techniques. The SSD protocol is inspired by the concatenated-code scheme of [YK24]. We distill the stabilizer states in multiple levels using quantum Hamming codes $[[2^l - 1, 2^l - l - 1, 3]]$, which have transversal Clifford gates. As its rate quickly approaches 1, the multi-level distillation protocol gives rise to a constant-space overhead. The details of this protocol are in Section 2.4. On the other hand, the MSD protocol is based on a new family of good qudit codes supporting transversal CCZ gate, that we describe in Section 2.5. There, we give a simple and self-contained construction using (punctured) Reed-Solomon codes over binary extension fields using ideas from [KT19]. The claimed time overhead comes from efficient decoders for these codes and that the code block sizes are at most $O(\mathrm{loglog}(1/\varepsilon))$.

Let us give an intuition for why distillation improves the time overhead over simply using the concatenated-code FT scheme [AB99]: Multi-level distillation is in a sense also using concatenation, but we unencode from the code before going to the next level, whereas concatenated-code FT never unencodes. Note that we are running these protocols on top of a qLDPC code, and 'unencode' here means unencoding from the distillation code, not the computational qLDPC code, so we are still protected by the qLDPC code. Therefore, the time overhead increases additively in distillation in terms of number of levels, rather than multiplicatively as in concatenated-code FT schemes.

Addressable logic on high-rate codes

The next question is what resource states to prepare. This question arises because the gate connectivity in the simulated circuit C can be arbitrary. For example, a logical CZ gate between qubit 1 of a code block and qubit 7 of another code block needs a different resource state than CZ between qubits 5 and 32. Likewise, a Hadamard gate on qubit 1 needs a different resource state from Hadamard on logical qubit 2. We say these gates are of different types. The distillation protocols, as we show, still work to distill any of them. However, the issue is that there are too many such elementary resource state types. For example, there are $O(K'^2) = \mathrm{polylog}(W)$ possible CZ gates between two distinct code blocks. On the other hand, distillation protocols produce resource states of the same type *in batches* of size roughly

$N_{\mathrm{SSD}}, N_{\mathrm{MSD}}$ = quasipolylog$(1/\varepsilon)$, as seen above. Each layer in a circuit C may adversarially consists of gates of all types. Hence, naively, we either have to ruin the constant-space overhead to run distillation for all of these resource states in parallel (many of them will be left ununsed) or serialize the gates in C so that each layer contains gates of a single type. Either way we would incur a factor of at least polylog(W) in the overhead.

A solution to this is to design a small set of resource states that are capable of performing addressable logic. Here we do so with a set of $O(\log K') = O(\mathrm{loglog}(W))$ resource states. The idea is as follows: We will only consider resource states for logical single-qubit gates (H and S) that act on the 'first' logical qubit in a code block. For the two-qubit gates (CNOT), we only care about resource states for the gates that act on qubits 1 and 2 of the same block. Similarly, we only have resource states for the CCZ gate that acts on qubits 1, 2, 3 of the same code block. So far this set has $O(1)$ resource states. To turn them into addressable gates, we use $O(\log K') = O(\mathrm{loglog}\, W)$ special SWAP/permutation multi-qubit gates that allows us to perform low-depth arbitrary qubit permutation. Importantly, we show that the resource states for these multi-qubit states can also be distilled by running the stabilizer distillation protocol on top of the computation qLDPC code. Hence, in total we only use a set of $O(\mathrm{loglog}\, W)$ 'primitive computation states'. Using this primitive gateset, we can then perform a logical gate on any desired locations while only incurring a $O(\log(K')) = O(\mathrm{loglog}\, W)$ time overhead. The details of this are given in Section 2.3.

Quantum locally testable codes

We now move on to a subtlety that was intentionally omitted in the above discussion. We have claimed that distillation protocols can be applied on the output of low-fidelity resource states to boost the fidelity. However, these protocols are being performed on top of qLDPC codestates, and their analysis only applies when the systems are guaranteed to be in the codespace. This is not necessarily the case for noisy codestates prepared by concatenated-code FT. Let $|\psi\rangle$ be the target resource state. Informally, concatenated-code FT gives the following guarantee: with probability $\geq 1-\varepsilon_0$, a state $\psi' = \mathcal{N}(|\psi\rangle)$ is output, where $\mathcal{N}$ is a local stochastic noise channel with parameter $O(p)$. However, in the remaining case, the output state of concatenated-code FT is arbitrary. It could either be (1) a codestate with some logical error or (2) an arbitrary non-code state. We must filter out noisy states of the latter case before running the

distillation protocols as quick as possible.

One idea is to measure the qLDPC code checks and declare 'FAIL' if the syndrome corresponds to a state with particularly large amounts of error. However, in general, there are non-code states that are very far from the codespace but only violate a few checks, so that a few measurement errors could prevent a highly damaged state from being detected.[8] This challenge is exactly addressed by local testable codes and is a motivation of classical local testability, as quoted from Spielman's PhD thesis [Spi95]: "The checker can instantly request a retransmission of that block, before the decoder has wasted its time trying to decode the message." One can also define quantum locally testable codes (qLTC) [AE15; EH17], which are, by definition, qLDPC codes. Roughly speaking, a qLTC with soundness $\rho < 1$ ensures that, if m checks are violated, then the state deviates from the codespace on at most m/ρ qubits. Hence, using a constant-soundness qLTC resolves this issue without incurring any extra time overhead: We can simply declare 'FAIL' if the syndrome weight is too large which will be robust to the presence of measurement errors, see Section 2.4 for more details. In summary, the discussion up to this point *conditionally* proves our main result. Specifically,

> Suppose that there exists a qLTC family[9] with constant rate, constant relative distance, and constant soundness (called c3-qLTC). Furthermore, suppose that the qLTC family admits an efficient parallel single-shot decoder. Then quantum fault tolerance can be achieved with the overheads stated in Theorem 2.1.2.

To our knowledge this work is the first time local testability is used in a fault tolerance protocol, either classical or quantum. The only previous work that used a related notion of locality in codes is [Rom06]. There, the author constructed a classical fault tolerance protocol using local decodable codes, a notion much stronger than local testability. Such notion, however, is not possible for quantum codes [AAV13].

[8] This is a well-known issue with toric code. A common solution to this is to perform multiple rounds (roughly as many as the code distance) of syndrome measurements. This incurs a time overhead and will not suffice for our main result.

[9] There is also a technical requirement that the family is sufficiently 'dense', meaning that the code block size does not grow too fast with respect to the family index.

Single-shot decoders for cubical complex quantum codes

Unfortunately the c3-qLTC conjecture is still an open question. However, a recent breakthrough [DLV24] has nearly approached this goal, providing a family of almost-c3 qLTC. In particular, Dinur, Lin, and Vidick [DLV24] generalized the construction of good qLDPC codes from [DHLV22] to high-dimensional cubical complexes. In combination with Panteleev and Kalachev's new result [KP25] on high-dimensional product-expansion of random codes, this gives a qLTC family of constant rate, inverse-polylog relative distance, and inverse-polylog soundness. We show that a suitable instantiation of this construction suffices to obtain our main result. To this end, we establish sequential and parallel single-shot decoders for general quantum codes built on the cubical complex construction:

Theorem 2.1.5 (Single-shot decoder, informal)**.** *Let $t \geq 2$. Consider an $[[n, k, d]]$ quantum LDPC code built from a t-dimensional cubical complex using the local sheaf construction such as in [DHLV22; DLV24]. There exists a linear-time decoder $\mathcal{D}_{\text{seq}}$, and constant $A, B, C > 0$ such that the following holds. If the data errors e and syndrome measurement errors m have weights satisfying $A|e|_R + B|m| < d$ (where $|\cdot|_R$ denotes stabilizer-reduced weight), then $\mathcal{D}_{\text{seq}}$ proposes a correction such that the residual error after applying the correction is upper bounded by $C \cdot |m|$. A similar statement holds for a $O(\log n)$-time parallel decoder $\mathcal{D}_{\text{par}}$. In fact, running $\mathcal{D}_{\text{par}}$ in $O(\tau)$ time guarantees that the residual error is suppressed to $\frac{1}{2^{-\Theta(\tau)}}|e|_R + C|m|$.*

We note that the above has been heavily simplified, see Section 2.3 and Section 6 of the full version of the paper [NP24] for precise statements. This result is a bounded-distance decoder, i.e., it applies to all (adversarial) errors up to certain weight. It applies to the DHLV code [DHLV22] and also the almost-c3 qLTC in [DLV24]. In other words, the good qLDPC code family of [DHLV22] admits a single-shot decoder up to linear-weight adversarial errors, answering an open question raised by Gu et al. [Gu+23]. For the almost-c3 qLTC, this result holds for adversarial errors of weight up to $n/\text{polylog}(n)$.

Techniques. The decoder is a generalization of the small-set flip decoder in [DHLV22] to higher-dimensional cubical complexes ([DHLV22] was on a square complex, i.e., 2-dimensional cubical complex). The main technical challenge is to come up with appropriate high-dimensional generalizations of the procedures and proof techniques there. Furthermore, we show that the decoder is single-shot and parallelizable. The

sequential version goes roughly as follows: while the syndrome is still nonzero, find a small-set flip in the local view of some vertex that reduces the syndrome weight. In the case when the syndrome is noiseless, we show that if no such flip is found, then the remaining error must be either zero or very large. On the other hand, we show that the chain complex is *small-set co-boundary expanding*, which roughly says that the syndrome weight provides an upper bound on the error weight as long as the error weight is sufficiently small[10]. And since the syndrome weight only reduces throughout the algorithm by design, the error weight cannot become large. Thus when the algorithm terminates, there is no remaining error. In the noisy syndrome case, a similar proof strategy also works by noting that coboundary expansion-based proof is robust to measurement errors. Naturally, the parallel version of the decoder attempts to perform local flips in parallel. We show that, in each parallel decoding round, such flips have large overlap with the underlying error, hence reducing the syndome weight by a multiplicative constant factor per round.

The X-syndrome decoder is not the same due to the asymmetry in the code construction. Similar to [DHLV22], it is obtained via a reduction to a Z-syndrome decoding on a *related* code. This reduction starts by having each vertex locally make a 'guess' about what it thinks the error should look like within its local view. The goal is now to fix the inconsistencies between these local guesses to obtain a global guess. The second step of the reduction involves an argument following the ideas in the code distance proof of [DLV24]. In particular, for each edge, the opinions of the vertices in that edge are compared, and hence this inconsistency-fixing problem can be moved one level up in the cubical complex. This procedure is repeated until the top level of the cubical complex is reached. Then, it turns out that we can map this inconsistency-fixing problem into a Z-syndrome decoding problem on a related code. So we can invoke the Z-syndrome decoder for that code to obtain a fix, and then propagate the fix back down. Since the above inconsistency-propagation procedure can be done locally, it turns out that the X-syndrome decoder inherits the single-shotness and parallelizability from the Z-syndrome decoder. We refer to Section 6 of the full paper [NP24] for more details.

Combining everything together

Intuitively, combining what we have described so far, including the almost-c3 qLTC with single-shot decoder, we can obtain an FT procotol with overheads as

[10]The small weight condition makes this property weaker than local testability, but it is all we need for the decoders.

stated in Theorem 2.1.2 but with a sub-constant threshold of $1/\mathrm{polylog}(N') = 1/\mathrm{polyloglog}(W)$. To achieve a constant threshold, the final step is to concatenate the described protocol with the Yamasaki-Koashi protocol [YK24]. Their protocol incurs constant-space overhead and a time overhead of $\exp\big((\log\log 1/\eta)^2\big)$ to fault-tolerantly simulate a constant-sized operation to logical error rate η. Hence, setting $\eta = 1/\mathrm{polyloglog}(W)$ we only incur an extra factor of $\exp\big((\log\log\log\log W)^2\big)$ in the time overhead.

This concludes our description of the high-level ideas and new technical results leading to Theorem 2.1.2. Making the above intuitive description rigorous is extremely involved. To do so, we introduce a new framework to prove (classical or quantum) fault tolerance that we call the *weight enumerator formalism* in Section 2.2. It can be viewed as a generalization of the technique of Aharonov and Ben-Or [AB99] (dating back to [Gác86]) of counting 'bad' faulty locations. The motivation of this new framework is the need to combine many different types of fault-tolerant gadgets in non-trivial ways to construct new fault-tolerant gadgets. In contrast to traditional concatenated-code FT schemes where the same gadgets are recursively used, the large variety of currently existing fault-tolerant gadgets and ways to combine them cause this counting to become unwieldy. Here, we use a framework inspired by the weight enumerator polynomial methods from coding theory. The key observation is that we can associate polynomials to the family of bad fault sets and work with the polynomials similar to probabilities. Multiplication and addition of these polynomials corresponds nicely to operational meaning. We find this framework applicable generally and expect it will be useful in proving fault tolerance in other contexts.

Interpolating space and time overheads. We briefly describe without proofs how the tradeoff between space and time overheads can be smoothly tuned while keeping the overall spacetime overhead $\widetilde{O}(\log W)$ (the auxiliary classical time per quantum timestep is still the same as in Theorem 2.1.2). The idea is to not use all $K' = \Theta(N')$ logical qubits in a code block, but instead use only k' of them and simply ignore the other logical qubits. This reduces the main slowdown factor $O(K') = O(\log W)$ due to gate scheduling (see Section 2.1) to $O(k')$, and hence improves the time overhead to $O(k' \log^{o(1)} W)$, while increasing the space overhead to $O(K'/k')$. Choosing $k' = 1$ gives $O(\log W)$-space overhead and $O(\log^{o(1)} W)$-time overhead. To obtain $O(\log^{1+o(1)} W)$-space and $O(1)$-time overheads, we simply shift the ancilla state factories overhead completely to space by preparing ancilla states off-line, and then

using the single-shot decoder to preserve them until use. In addition, instead of concatenating with the FT protocol of [YK24] in the step of improving the threshold from $\eta = 1/\mathrm{polyloglog}(W)$ to constant, we use the constant-time overhead schemes described in [Got13, Section 8] or [Bom15], which only incurs an extra space overhead factor of $\mathrm{polylog}(1/\eta) = \mathrm{polylogloglog}(W)$.

Related works

Classical fault tolerance. The study of fault-tolerant (FT) computation was started in von Neumann's lectures [Von56], where he gave a FT scheme for classical computation with log-space and constant-time overhead. His scheme was later made rigorous and explicit by [DO77; Pip85]. A matching lower bound on the FT overhead was shown by [PST91; GG94]. These works settle the FT spacetime overhead required for classical computation.

Quantum fault tolerance. The first fault-tolerant quantum computation (FTQC) threshold theorems were shown independently by [AB99; Kit97; KLZ98], building on [Sho96]. These works established that FTQC is possible with polylog space and time overheads. Later works such as [AGP05; Got10] devised new proof methods for the concatenated-code FT scheme and improved the rigorously proved threshold value. Motivated by practical considerations, a standard model of FTQC often considers a noisy quantum computer with auxilliary polynomial-time and noiseless classical computation (in this work we only assume polyloglog-time). In this model, Gottesman [Got13] initiated the study of constant-space-overhead FTQC. [FGL20] showed that constant-space and polynomial-time overhead is possible. Very recent work [CFG24] studies FTQC with quantum output which is required to prepare resource states of [Got13] and [FGL20]. In 2022, [YK24] achieved a quasipolylog-time overhead which is the state-of-the-art before initiating our work. Concurrent to our work, we were made aware of a recent independent work initially appearing at the AQIS'24 conference [TKY24], which gives a proof of FTQC with constant-space and $\mathrm{polylog}(W)$-time overheads using substantially different techniques from ours.

Good qLDPC code decoders. Some families of good quantum LDPC codes [PK22; LZ22b] have been recently shown to admit a single-shot decoder [Gu+23]. Other provably efficient decoders for qLDPC codes include [FGL20; LZ23; DHLV22]. The recent breakthrough work [DLV24], using classical codes constructed in [KP25], construct almost-good qLTC by generalizing the code from [DHLV22].

Magic states and non-Clifford gates. The study of magic state distillation protocols are initiated in [BK05; Kni05]. Prior to this work, the state-of-the-art MSD (space) overheads are due to [HH18] (qubit case) and [KT19] (qudit case). During preparation of this work, recently posted and independent works [WHY24; Ngu24; GG24] have established that constant-space-overhead MSD by using algebraic-geometry codes. However, the time overhead was not studied in [Ngu24; GG24]. [WHY24] describe an explicit protocol, although, naively, polylog$(1/\varepsilon)$ time overhead is required due to the (un)encoding depth and syndrome measurements of the distillation code with large block size. This is insufficient to obtain our main result. See also very recent works [Zhu+23; SPW24; GL24; BDET24] on quantum LDPC codes with a non-Clifford gate.

Outlook

There are multiple immediate follow-up questions to our work, here we list a few:

- Can we drop the assumption of polyloglog-time noiseless classical computation? We believe that this requirement is not needed and can be removed with careful use of classical fault tolerance along with our constant-time parallel decoder. We leave details of this procedure and its overhead analysis to future work.

- Can we improve FTQC overhead lower bounds? [FMS22] obtain a space lower bound that is linear in the ideal circuit space. We conjecture that our result in Theorem 2.1.2 is optimal up to sub-polylog factors. We suspect that the classical functions witnessing the $\Omega(\log W)$ classical FT overhead lower bound in [GG94] can be extended to the setting of FTQC without auxiliary classical computation.

- Toward the question of an overhead lower bound, we describe here an argument for why a time overhead of $O(\log W)$ is likely required for constant-space-overhead FTQC (even with auxiliary classical computation), suggesting that Theorem 2.1.2 is likely optimal up to sub-polylog factors. This comes from the potentially adversarial gate connectivity in the ideal circuit. Consider a FTQC scheme that uses constant-rate code blocks $[[n, k = \Theta(n), d]]$ to simulate an ideal circuit C of W qubits and, for simplicity, size poly(W). We need the code distance to be $d = \Theta(\log W)$ to maintain a poly(W)-sized computation. Consider $k+1$ codeblocks $\mathcal{B}_0, \ldots, \mathcal{B}_k$. Suppose we can perform

two-qubit gates on any pairs of the $k(k+1)$ logical qubits in $O(1)$ depth. Now, an adversarial layer in the simulated circuit C may consist of two-qubit gates from each logical qubit $j \in [k]$ in $\mathcal{B}_0$ to a logical qubit in $\mathcal{B}_j$. The implementation of each gate has to touch $\geq d$ physical qubits in $\mathcal{B}_0$, thus implementing all $k = \Theta(n)$ of them needs a $\Omega(d) = \Omega(\log W)$ depth by pigeonhole principle. Choosing an initial specific grouping of logical qubits cannot avoid this issue because the connectivity in C may adversarially vary across layers. Regrouping logical qubits mid-circuit with a sorting network would also incur a $\Omega(\log W)$ time factor.

- Our construction is rather complicated. Is there a simpler construction that achieves $\widetilde{O}(\log W)$ spacetime overhead? And can we remove the sub-polylog factors?

Organization of paper

In Section 2.2 we first provide some preliminaries. We then describe our computation model and present the polynomial formalism and related notation used in Sections 2.3 and 2.4. In Section 2.3 we present the proof of our main result, with technical lemmas about distillation protocols and the single-shot decoder deferred to later sections. In Section 2.4 we construct gadgets to distill resource stabilizer and magic states using the code constructed in Section 2.5. The construction of the single-shot decoder is deferred to Section 6 of the full paper [NP24]. For convenience, we provide a diagram (below) of the high-level components of the proof with hyperlinks.

Acknowledgements

This work was done in part while the authors were visiting the Simons Institute for the Theory of Computing, supported by DOE QSA grant #FP00010905 and NSF QLCI Grant No. 2016245. We thank Anurag Anshu, Nikolas Breuckmann, Louis Golowich, Ting-Chun David Lin, Pedro Paredes, John Preskill, Shiro Tamiya, Hayata Yamasaki, and Guanyu Zhu for useful discussions. We especially thank Zhiyang (Sunny) He for insightful discussions about quantum fault tolerance proofs. QTN acknowledges support from the NSF Award No. 2238836 and support from the Harvard Quantum Initiative. CAP acknowledges funding from the Air Force Office of Scientific Research (AFOSR) FA9550-19-1-0360 and U.S. Department of Energy Office of Science, DE-SC0020290. The Institute for Quantum Information and Matter is an NSF Physics Frontiers Center.

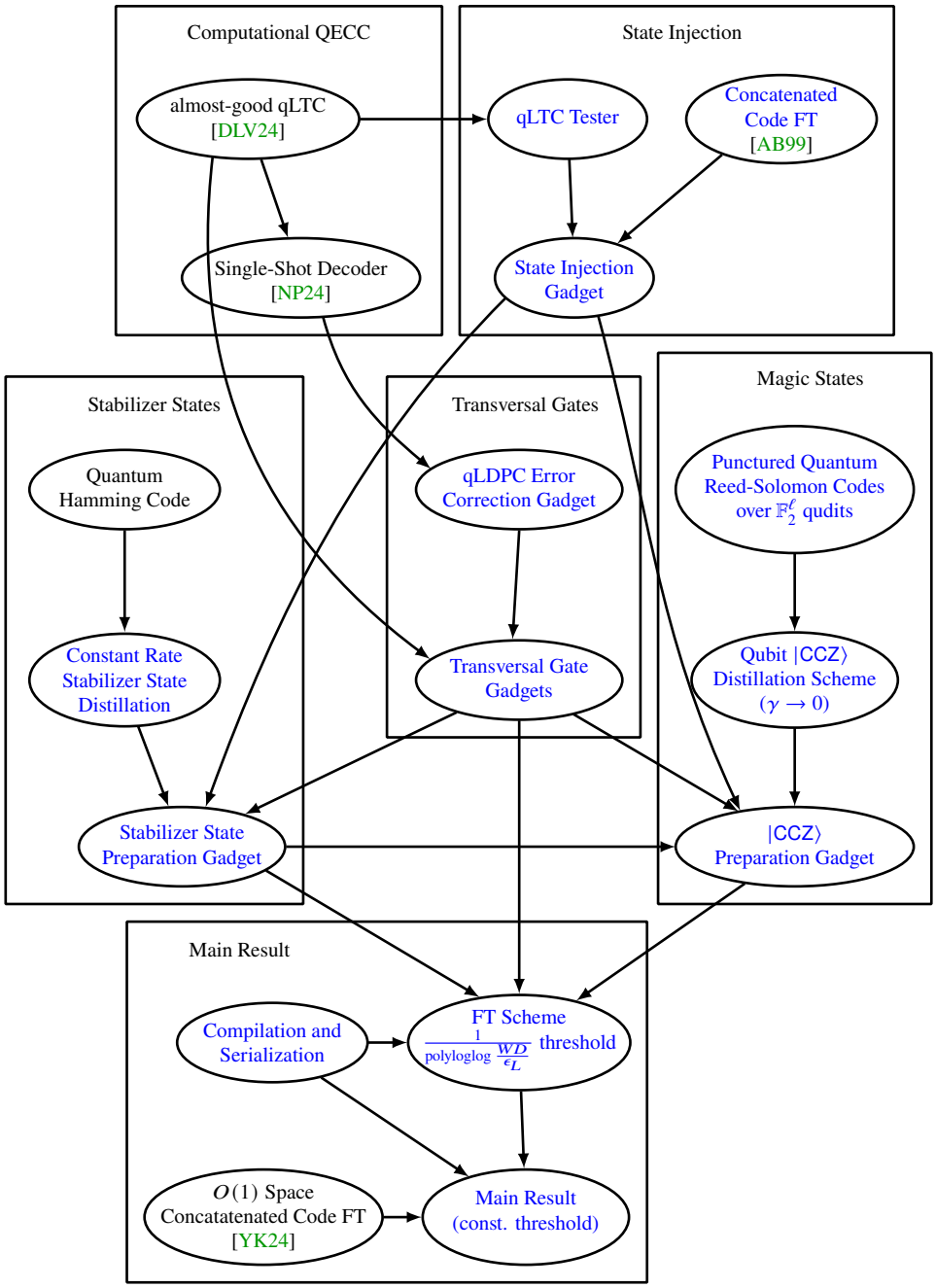


Figure 2.1: High-level organization of proof.

2.2 Model of computation and weight enumerator formalism

Aharonov and Ben-Or [AB99] prove the fault tolerance of their scheme by first working with the set of faulty operations in an adversarial sense with no randomness. However, the analysis is much more fine-grained than simply the weight of the set: The set of faulty operations is required to exclude certain "bad" sets. This allows one to prove thresholds against many types of noise including stochastic noise. In this section, we introduce a generalization of this technique inspired by the need to combine many different types of fault-tolerant gadgets in non-trivial ways to construct new fault-tolerant gadgets. The key observation is that we can associate polynomials (over $\mathbb{R}_+$) to the family of bad sets and work with the polynomials similar to probabilities while remaining in the adversarial noise paradigm. These polynomials are a special case of the weight enumerator polynomials from coding theory, so here we also refer to them as weight enumerators.

We begin by defining our model of computation formally. This model allows a small polyloglog-sized classically circuit between layers of the quantum circuit. This attempts to capture that classical gates are much faster than quantum gates in practice and slightly simplifies some of the arguments.

Definition 2.2.1 (Adaptive quantum circuit)**.** *An adaptive Clifford+*CCZ *quantum circuit of depth D and space W is described by a list of quantum operations for each timestep $t \in [D]$ such that each quantum operation has disjoint support. The valid quantum operations are one and two-qubit Clifford gates,* CCZ *gates, destructive Pauli basis measurements, and state initialization. Let $M = O(WD)$ be the number of measurements in the quantum circuit. To each measurement, associate a unique index $m \in [M]$. During execution, a measurement record $\mathcal{M}$ of length M bits is maintained such that $\mathcal{M}[m]$ is the result of the measurement at the end of the timestep where the measurement is executed. We also maintain a classical memory $\mathcal{N}$ of size $O(WD \operatorname{polylog}(WD))$.*

To each timestep $t \in [D]$, we associate a classical circuit C_t of depth $O(\operatorname{polyloglog}(WD))$ that has access (random access model) to the measurement record (read only) and memory (read/write). For each unitary quantum operation O *in the quantum circuit at timestep t, C_t outputs an associated control bit a_{O}. At time 0, the W qubits are initialized to $|0\rangle$, and the measurement record and memory is initialized to the all-0's string. To execute timestep $t \in [D]$, we first run C_t which reads the measurement record, reads/writes to the memory, and produces the control bits a_{O} for each unitary operation* O *at time t. Then, all unitary operations* O *for which $a_{\mathsf{O}} = 1$ and all*

measurement operations are applied to arrive at timestep $t+1$.

The output of the computation is the output of a fixed classical circuit $\mathcal{D}$ of depth $O(\text{polyloglog}(WD))$ evaluated on the measurement record.

Next, we provide some preliminary definitions that will be utilized later.

Preliminaries

We will use $\mathcal{H}_2$ to refer to the standard two-dimensional Hilbert space of a qubit. For a Hilbert space $\mathcal{H}$, we use $\mathcal{D}(\mathcal{H})$ to refer to the set of density operators supported on $\mathcal{H}$.

Definition 2.2.2 (Quantum LDPC code)**.** *A quantum CSS code $Q = CSS(H_X, H_Z)$ is said to be Δ-qLDPC if H_X and H_Z have row and column weight at most Δ.*

Our quantum codes will implicitly carry an encoding map from a dim 2^k logical Hilbert space to a dim 2^k subspace. This choice is almost arbitrary[11] with the exception of the constant quantum depth encoding circuit for the computational code (Lemma 2.4.2) and the magic state distillation code (Corollary 2.5.14). For this reason we will refer to the encoding map ϕ_C of Q as *the encoding map*.

Definition 2.2.3. *A classical code $C \subseteq \mathbb{F}_2^n$ defined by the check matrix $H \in \mathbb{F}_2^{r\times n}$ is locally testable with soundness ρ and locality Δ if H has row and column weight at most Δ and for all $x \in \mathbb{F}_2^n$,*

$$\frac{|Hx|}{r} \geq \rho \frac{d(x, C)}{n}. \tag{2.1}$$

Definition 2.2.4 (Quantum LTC [AE15; EH17])**.** *A quantum CSS code $Q = CSS(H_X, H_Z)$ is said to be (ρ, Δ)-qLTC if H_X and H_Z are both locally testable with soundness ρ and locality Δ.*

Note that a quantum code that is (ρ, Δ)-qLTC is also Δ-qLDPC.

Definition 2.2.5 (Stabilizer-reduced weight)**.** *For a stabilizer code with stabilizer set S and a Pauli error E, we say that E is stabilizer-reduced if it is minimal with respect to the application of stabilizers. That is, for every $s \in S$, the weight of $|sE| \geq |E|$. For a general Pauli error P, the stabilizer-reduced weight $|P|_R$ is the weight of a stabilizer-reduced error equivalent to P under applications of stabilizers.*

[11] As long as it takes computational basis states to computational basis states.

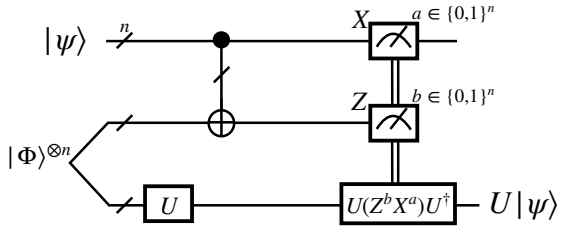


Figure 2.2: The n-qubit gate teleportation circuit. If U is in the k-th level of the Clifford hierarchy, then the correction is in level $k-1$. We will use this circuit and the stabilizer resource state $I \otimes U |\Phi\rangle^{\otimes n}$ to implement Clifford gates in our construction.

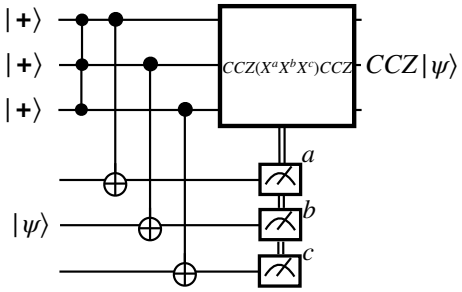


Figure 2.3: Gate teleportation for CCZ gate using the magic state $|\mathsf{CCZ}\rangle = \mathsf{CCZ}|+++\rangle$. The fix-up has the form of Pauli operations and CZ gates. The latter are in turn implemented using Figure 2.2.

Let $|\Phi\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$ denote the Bell state. We recall the standard gate teleportion scheme.

Definition 2.2.6 (Gate teleportation [GC99])**.** *For an arbitrary n-qubit unitary U, the state $|U\rangle = I \otimes U |\Phi\rangle^{\otimes n}$ (U acting on one qubit from each Bell pair) allows us to perform a teleported application of U on an input state $|\psi\rangle$: By applying the standard quantum teleportation circuit (using X and Z basis measurement and* CNOT*) between a register $|\psi\rangle$ and $|U\rangle$, and applying a U-dependent fix-up operation ($UX_iU^\dagger$ and $UZ_iU^\dagger$) conditioned on the measurement outcomes of the teleportation circuit, we are left with $U|\psi\rangle$.*

In particular, if U is in the k-th level of the Clifford hierarchy, then the fix-up operation is in the $(k-1)$-th level. We will only use CCZ, a third-level gate, and Clifford gates for which the fix-up is Pauli. For Clifford gates, the resource state is a stabilizer state and the fix-up includes Pauli operations. In this work we use this standard scheme to perform Clifford gates on the computation qLDPC code. To perform CCZ, we instead use the magic state $\mathsf{CCZ}|+++\rangle$ and the circuit in Figure 2.3.

Remark 2.2.7 (Clifford+T)**.** *The choice to use Clifford+*CCZ *instead of Clifford+T may seem unconventional, but we note that any circuit using the Clifford+T gateset may be converted to the Clifford+*CCZ *gateset with* $O(\log(WD/\varepsilon))$ additive *time cost and constant overhead, by first approximately preparing a* single T *state to* $1/\mathrm{poly}(WD/\varepsilon)$ *error and then using the (exact) catalyzed conversion* $|\mathsf{CCZ}\rangle + |T\rangle \to 2\,|T\rangle$ *to create many* $|T\rangle$ *states [BCHK20].*

Noise model

We are now ready to define what it means for a circuit to be executed in a faulty way. Note that we have not yet defined a distribution (or similar) over the ways that a circuit can be faulty. A major feature is that we can separate the proof of correctness from the distribution[12] the faults are drawn from.

Definition 2.2.8 (Fault)**.** *For an adaptive quantum circuit* $C = C_D C_{D-1} \dots C_1$ *of depth* D *and width* W*, a fault* $\mathbf{f}$ *is a sequence of* W*-qubit superoperators* $(\mathbf{f}_1, \mathbf{f}_2, \dots, \mathbf{f}_D)$*. We use* $C[\mathbf{f}]$ *to denote the execution of the circuit* C *subject to the fault* $\mathbf{f}$*. That is,*

$$C[\mathbf{f}] = \mathbf{f}_D C_D \mathbf{f}_{D-1} C_{D-1} \dots \mathbf{f}_1 C_1. \tag{2.2}$$

We say that a fault is physical *if each superoperator is completely-positive and trace-preserving (CPTP).*

For a sub-circuit C' *of* C*,* $C'[\mathbf{f}]$ *should be interpreted as the restriction of* $\mathbf{f}$ *to the locations of* C'*. Whenever such a restriction appears,* $\mathbf{f}$ *will always factorize such that the restriction is well defined.*

We remark that our notion of a fault (singular) approximately corresponds to many faults (plural) in some modern works. For convenience, we extend multiplication to faults, coordinate-wise. That is, for two faults $\mathbf{f}$ and $\mathbf{g}$, $\mathbf{fg} = (\mathbf{f}_1\mathbf{g}_1, \mathbf{f}_2\mathbf{g}_2, \dots)$.

A particularly special case of a fault will be that of a *Pauli fault*. Pauli faults are easier to work with for reasons that will be made clear shortly.

Definition 2.2.9 (Pauli superoperator)**.** *A superoperator is said to be a Pauli superoperator if it is of the form* $\rho \mapsto \alpha A \rho B$ *for some Pauli operators* A *and* B *and complex coefficient* α*. We will further say it is a diagonal Pauli superoperator if it is of the form* $\rho \mapsto \alpha A \rho A$*.*

[12] Most generally, it can be the case that the faults are not drawn from a probability distribution at all (e.g. coherent noise), but we can still upper bound the contribution from "bad" fault paths in the output distribution.

Definition 2.2.10 (Pauli faults)**.** *For a fault* $\mathbf{f} = (\mathbf{f}_1, \mathbf{f}_2, \ldots, \mathbf{f}_D)$*,* $\mathbf{f}$ *is said to be a* Pauli fault *(diagonal Pauli fault) if every superoperator* $\mathbf{f}_t$ *in the sequence is a Pauli superoperator (diagonal Pauli superoperator).*

Remark 2.2.11. *Since Pauli matrices form a complete basis for the space of matrices, it is clear that every fault can be written as a linear combination of Pauli faults. For a circuit subject to a physical fault, we can decompose it into a sum over executions of the circuit subject to Pauli faults. We will use this fact to reduce the case of general faults to that of Pauli faults. Since Pauli faults are tensor products of single-qubit operators (i.e. factorizes), this makes it convenient to prove properties of circuits in isolation.*

We now define the locations of a circuit.

Definition 2.2.12 (Location)**.** *A location of an adaptive quantum circuit is a tuple of a timestep* $t \in [D]$ *and a set of qubits* $q \subseteq [W]$ *such that a quantum operation (including identity) supported on the qubits* q *is performed at time* t*. For each* $t \in [D]$*, every qubit must be contained in exactly one location.*

We are now ready to define the analog of the support of a fault.

Definition 2.2.13 (Fault path)**.** *For an adaptive quantum circuit* C *and a fault* $\mathbf{f}$*, the fault path* $\operatorname{supp} \mathbf{f}$ *of* $\mathbf{f} = (\mathbf{f}_1, \ldots, \mathbf{f}_D)$ *is the subset of locations of* C *that* $\mathbf{f}$ *is supported on. I.e. a location* $(t \in [D], X \subseteq [W])$ *of* C *is in the fault path if* $\operatorname{supp} \mathbf{f}_t \cap X \neq \emptyset$*. Such a location is said to be* faulty*.*

Definition 2.2.14 (Local stochastic noise)**.** *For an adaptive quantum circuit* C *with the set of locations* L*, the random physical fault* $\mathbf{f}$ *is said to be distributed according to an* ϵ*-locally stochastic faults model if, for all* $S \subseteq L$*,*

$$\Pr(S \subseteq \operatorname{supp} \mathbf{f}) \leq \epsilon^{|S|}. \tag{2.3}$$

For a quantum circuit C with depth D and space W, we define $|C| = WD$. Furthermore, we say a circuit is polynomially bounded if $D = O(\operatorname{poly} W)$.

Weight enumerators

In order to combine fault tolerant gadgets using different techniques such as code concatenation and qLDPC, it becomes convenient to define a notion of a *sparse* error or fault-path that differs from [AB99]. Roughly, these are the errors that are benign

and do not affect the final outcome. We will define these sets by requiring that they exclude certain "bad" subsets. Depending on how these bad subsets are defined, this imposes structure on the error or fault path that can be used to prove correctness of circuit gadgets in the presence of such errors / fault paths.

We now define a notion analogous to a "sparse" set of [AB99].

Definition 2.2.15 (Avoiding sets)**.** *For a set Ω and a family of subsets $\mathcal{F} \subseteq P(\Omega)$, referred to as* the bad sets*, a subset $X \subseteq \Omega$ is said to be $\mathcal{F}$-avoiding if it does not contain an element of $\mathcal{F}$. That is, $X \subseteq \Omega$ is $\mathcal{F}$-avoiding if and only if*

$$\forall F \in \mathcal{F},\ F \not\subseteq X\ .$$

To this family of sets we can associate a weight enumerator polynomial. This tracks how many elements of the family of each size there are.

Definition 2.2.16 (Weight enumerators)**.** *For a finite set Ω and a family of subsets $\mathcal{F} \subseteq P(\Omega)$, it will be convenient to associate a polynomial $\mathcal{W}(\mathcal{F};\ x)$ to $\mathcal{F}$ defined as the sum*

$$\mathcal{W}(\mathcal{F};\ x) = \sum_{w=0}^{\infty} A_w x^w, \tag{2.4}$$

where $A_w = |\{F \in \mathcal{F} \mid |F| = w\}|$ is the number of elements of $\mathcal{F}$ of weight w.

Note that our definition is a special case of the weight enumerator polynomial from classical coding theory. The major motivation of this definition is that $\mathcal{W}(\mathcal{F};\ x)$ is defined such that, for a random variable E taking values in $P(\Omega)$ that is ϵ-local stochastic, the probability that E is not $\mathcal{F}$-avoiding is bounded by the evaluation $\mathcal{W}(\mathcal{F};\ x)$ at ϵ. In other words,

$$\forall S \subseteq \Omega,\ \Pr(S \subseteq E) \leq \epsilon^{|S|} \Rightarrow \Pr(S \text{ is not } \mathcal{F}\text{-avoiding}) \leq \sum_{f \in \mathcal{F}} \epsilon^{|f|} = \mathcal{W}(\mathcal{F};\ \varepsilon). \tag{2.5}$$

Having motivated the introduction of weight enumerator polynomials, a natural question is what multiplication and addition of polynomials correspond to. We begin by defining a sort of sum and product operations on the families of bad sets which produce new families of bad sets with weight enumerator given by the sum or product (Proposition 2.2.18). The addition operation is essentially the union-bound

from probability. The product operation is slightly more subtle: It is analogous to the probability of simulataneous failure of independent gadgets, but recall that the probability of failure is not entirely independent in our error model. However, we can recover a notion of independence whenever the families of bad sets do not coincide.

Definition 2.2.17 (Ring of bad sets)**.** *For a set Ω with the decomposition $\Omega_1 \cup \Omega_2 = \Omega$ and families of subsets $\mathcal{F}_1 \subseteq P(\Omega_1)$ and $\mathcal{F}_2 \subseteq P(\Omega_2)$, we define two operations between $\mathcal{F}_1$ and $\mathcal{F}_2$ to arrive at a subset $\mathcal{F} \subseteq \Omega$. Let ι_1 (ι_2) be the canonical inclusion map from Ω_1 (Ω_2) to Ω.*

The first operation is a sort of addition operation.

$$\mathcal{F}_1 \boxplus \mathcal{F}_2 := \iota_1(\mathcal{F}_1) \cup \iota_2(\mathcal{F}_2), \tag{2.6}$$

where by $\iota_1(\mathcal{F}_1)$, we mean the application of ι_1 to each element of the family $\mathcal{F}_1$.

When Ω_1 and Ω_2 are disjoint, that is $\Omega = \Omega_1 \sqcup \Omega_2$, we define a multiplication operation.

$$\mathcal{F}_1 \circledast \mathcal{F}_2 := \{\iota_1(f_1) \cup \iota_2(f_2) \mid f_1 \in \mathcal{F}_1,\ f_2 \in \mathcal{F}_2\}. \tag{2.7}$$

First note that the name addition and multiplication are deserved: Multiplication distributes over addition. Informally, a set $X \subseteq \Omega$ is $\mathcal{F}_1 \boxplus \mathcal{F}_2$-avoiding if $X|_{\Omega_1}$ is $\mathcal{F}_1$-avoiding *and* $X|_{\Omega_2}$ is $\mathcal{F}_2$-avoiding whereas X is $\mathcal{F}_1 \circledast \mathcal{F}_2$-avoiding if $X|_{\Omega_1}$ is $\mathcal{F}_1$-avoiding *or* $X|_{\Omega_2}$ is $\mathcal{F}_2$-avoiding. Later, the addition will be used for union-bounds while the second operation will be used for upper bounding the probability of a large simultaneous failure.

Conveniently, the weight enumerators are well behaved under these operations

Proposition 2.2.18. *It holds that*

$$\mathcal{W}(\mathcal{F}_1 \boxplus \mathcal{F}_2;\ x) = \mathcal{W}(\mathcal{F}_1;\ x) + \mathcal{W}(\mathcal{F}_2;\ x), \tag{2.8}$$

$$\mathcal{W}(\mathcal{F}_1 \circledast \mathcal{F}_2;\ x) = \mathcal{W}(\mathcal{F}_1;\ x)\mathcal{W}(\mathcal{F}_2;\ x). \tag{2.9}$$

Proof. A weight w element of $\mathcal{F}_1 \boxplus \mathcal{F}_2$ is a weight w element of $\mathcal{F}_1$ or of $\mathcal{F}_2$, so the coefficients of the same degree add. To prove the second property, note that any weight w element of $\mathcal{F}_1 \circledast \mathcal{F}_2$ must be the disjoint union of a weight w_1 element of $\mathcal{F}_1$ and a weight w_2 element of $\mathcal{F}_2$ where $w = w_1 + w_2$. □

The product operation will be indispensible when constructing new fault-tolerant gadgets from less resilient older ones. In several places such as recursive simulations (concatenation) and resource state distillation, we will have fault-tolerant gadgets fail in ways that are recoverable. Because of this, we will need to analyze the fault sets in a recursive way. Unfortunately, a large degree of generality is required for all potential use-cases. This motivates the following definition of a composition operation.

Definition 2.2.19 (Composition)**.** *For a proposition Q and a set X, let $\mathbb{I}_Q[X]$ denote X when Q holds and $\emptyset$ when $\neg Q$ holds.*

Fix a set Ω and $\mathcal{F} \subseteq P(\Omega)$. For ease of notation, identify $\Omega \simeq [n]$. Also, consider sets $\{\omega_i\}_{i\in\Omega}$ and families of sets $\{\mathcal{S}_i \subseteq P(\omega_i)\}_{i\in\Omega}$ that are indexed by elements of Ω. We define an operation $\mathcal{F} \bullet \{\mathcal{S}_i\}_i \subseteq P\left(\bigsqcup_{i\in\Omega} \omega_i\right)$ where

$$\mathcal{F} \bullet \{\mathcal{S}_i\}_i := \boxplus_{f\in\mathcal{F}} \left(\mathbb{I}_{n\in f}[\mathcal{S}_n] \circledast \mathbb{I}_{(n-1)\in f}[\mathcal{S}_{n-1}] \dots \mathbb{I}_{2\in f}[\mathcal{S}_2] \circledast \mathbb{I}_{1\in f}[\mathcal{S}_1]\right).$$

Note that if $\mathcal{S}_i = \mathcal{S} \subseteq P(\omega)$ for some set ω, then $\mathcal{F} \bullet \{\mathcal{S}_i\}_i \subseteq P(\Omega \times \omega)$ corresponds to all sets which for some element $f \in \mathcal{F}$ are elements of $\mathcal{S}$ on each row of $\Omega \times \omega$ where f is non-trivial. We will use the notation $\mathcal{F} \bullet \mathcal{S}$ for this special case.

Later, when defining a recursive simulation, Ω will label circuit locations to be simulated while each $\omega_{i\in\Omega}$ will label locations in the simulation of the operation associated with location i. Different gadgets may have different sets of locations which demands the additional complexity in the definition. Readers can simply consider code concatenation as a guide: The outer and inner code blocks have corresponding bad sets $\mathcal{F}_{\text{outer}}$ and $\mathcal{F}_{\text{inner}}$. Informally, the family produced by the $\bullet$ operation takes a bad set $f \in \mathcal{F}_{\text{outer}}$ of the outer code block and expands each element $e \in f$ into one of the bad sets $f' \in \mathcal{F}_{\text{in}}$ of the inner code block, so that if a subset of qubits of the concatenated code is $\mathcal{F}_{\text{outer}} \bullet \mathcal{F}_{\text{inner}}$-avoiding, then the set is $\mathcal{F}_{\text{inner}}$-avoiding on all inner code blocks except for an $\mathcal{F}_{\text{outer}}$ set (of inner code blocks). More generally, the families of bad sets of the inner set may not be uniform e.g. a gadget of a concatenated code may use different inner code gadgets (performing different operations) with different locations and bad sets.

Remark 2.2.20. *Frequently, for a weight enumerator $\mathcal{W}(\mathcal{F};\, x)$, we will only have an upper bound on $\mathcal{W}(\mathcal{F};\, x)$ on some interval $[0, c]$ for $c \in (0, 1]$. c should be thought of as a multiple (e.g. $1/2$) of a threshold value.*

Proposition 2.2.21 (Composition of weight enumerators)**.** *Using the variables as defined in Definition 2.2.19, when there exists a polynomial $p(x)$ such that on some interval $x \in I \subseteq \mathbb{R}_+$ for all $i \in \Omega$, $\mathcal{W}(\mathcal{S}_i;\ x) \leq p(x)$,*

$$\mathcal{W}(\mathcal{F} \bullet \{\mathcal{S}_i\}_i;\ x) \leq \mathcal{W}(\mathcal{F};\ p(x)) \tag{2.10}$$

for $x \in I$.

Proof. $\mathcal{F} \bullet \{\mathcal{S}_i\}_i$ is constructed as a sum over a product for each $f \in \mathcal{F}$. Using the weight enumerator sum rule, the weight enumerator is the sum of weight enumerators for each product. Each product has $|f|$ terms and the corresponding weight enumerator can be evaluated using the weight enumerator product rule. Applying the restriction $x \in I$ and the upper bound $p(x)$, each term in the sum has upper bound $p(x)^{|f|}$. For each $d \in \mathbb{N}$, the number of terms in the sum proportional to $p(x)^d$ correspond to the number of elements of $\mathcal{F}$ of weight d. □

Gadgets

It will be convenient to group together (physical) qubits into "blocks." Roughly, a block is simply a bookkeeping method to refer to collections of qubits that roughly correspond to the physical qubits of a quantum error correcting code. To reduce the notation, we intentionally leave implicit a choice of labeling of the qubits. Recall that for a stabilizer code $\mathcal{Q}$ encoding k qubits into n qubits, we also leave implicit a choice of encoding map $\phi_{\mathcal{Q}}$ from a k-qubit Hilbert space to an n-qubit Hilbert space.

Definition 2.2.22 (Code block)**.** *For a stabilizer code $\mathcal{Q}$ on n qubits and a set of qubits $[M]$, a block of $[M]$ is an n-qubit subset of $[M]$ that will be treated as forming an encoded state of $\mathcal{Q}$. A block will be further equipped with bad sets $\mathcal{F} \subseteq P([n])$. Such a block will be referred to as a $(\mathcal{Q}, \mathcal{F})$-block. For a set of qubits $A \subseteq [M]$, A is said to be* sparse *on the $(\mathcal{Q}, \mathcal{F})$-block $B \subseteq [M]$ if $A \cap B$ is $\mathcal{F}$-avoiding.*

We would like a notion to compare if the intermediate computational state is equivalent to the correct computational state in a sense that is robust to the presence of benign errors. Here, we introduce a notion of deviation inspired by [AB99] and [Kit97], but suitably generalized for our purposes.

Definition 2.2.23 (Deviation)**.** *For a set of qubits A, a family of bad sets $\mathcal{F}$, and two states ρ, σ on A. ρ is said to be $\mathcal{F}$-deviated from σ there exists an $\mathcal{F}$-avoiding set $B \subseteq A$ and a superoperator $\mathcal{E}_B$ supported only on B such that $\rho = \left(\mathcal{I}_{A \setminus B} \otimes \mathcal{E}_B\right)(\sigma)$.*

For a quantum code $\mathcal{Q}$, a state ρ is said to be $\mathcal{F}$-deviated from $\mathcal{Q}$ if there exists a (possibly mixed) code state σ in $\mathcal{Q}$ such that ρ is $\mathcal{F}$-deviated from σ.

We use the less general term "Pauli $\mathcal{F}$-deviated" ("diagonal Pauli $\mathcal{F}$-deviated") when the superoperator $\mathcal{E}_B$ is a Pauli superoperator (diagonal Pauli superoperator)

Remark 2.2.24. *Our notion of deviation most closely corresponds to the notion of a "many-to-one" code of [Kit97]. A "many-to-one" code is the set of states that can be reached from a codestate of a quantum code by application of a superoperator on at most $l \in \mathbb{N}$ qubits. I.e. the space of states that are damaged but not irrecoverably so. "many-to-one" refers to the recovery map sending damaged states to a unique codestate of a standard quantum code.*

We avoid comparing the partial traces of states as in [AB99] since 1) the reference state is required to be pure ([AB08, section 4.3.3]) in order for there to exist a superoperator taking one state to another 2) the relation should not be symmetric. This difference in definition does not substantially modify the argument of [AB99; AB08].

It will be convenient to refer to a smaller unit of an adaptive quantum circuit which we will call a *gadget*. A gadget may take a quantum input and outputs the final state of the quantum and classical registers *including the measurement history*. The requirement that the gadget also preserves the measurement history is a technical one that is needed to deal with non-Pauli errors on inputs to the gadget since the presence of only Pauli faults does not necessarily imply the output error is Pauli. At the final step of our analysis, we will discard all measurements except for the result of the logical measurements. In our case, the gadgets will often operate on a subset of the classical and quantum registers and will be invoked in parallel. This will be implicit.

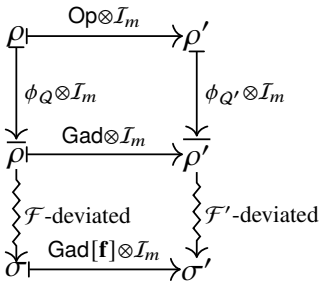
The gadgets in our construction will take as input an encoded state and output a state (on the quantum register), possibly encoded in a different code. This notion of a gadget is somewhat similar to [Kit97]. We now define a fault-tolerant gadget and give several useful propositions related to various forms of composition. We recommend that readers refer to the diagrams in parallel with the text.

Definition 2.2.25 (Fault-tolerant gadget)**.** *Fix stabilizer codes $\mathcal{Q}$ ($\mathcal{Q}'$) on n qubits (n' qubits) encoding k qubits (k' qubits) and their respective encoding maps $\phi_{\mathcal{Q}}$ ($\phi_{\mathcal{Q}'}$). A fault-tolerant gadget* Gad *operates on a $(\mathcal{Q}, \mathcal{F})$-block and returns a $(\mathcal{Q}', \mathcal{F}')$-block. To reduce notation, we will suppress the dependence on $\mathcal{Q}$, $\mathcal{F}$, and the corresponding primed analogs.*

Consider a gadget Gad *with a set of locations* $[L]$ *and a family of bad sets* $\mathcal{G} \subseteq P([L])$. *Let* $\mathbf{f}$ *be a fault. For a quantum operation* Op *mapping unencoded states from* $\mathcal{D}\left(\mathcal{H}_2^{\otimes k}\right)$ *to* $\mathcal{D}\left(\mathcal{H}_2^{\otimes k'}\right)$, Gad *is said to be a* $(\mathsf{Op}, \mathcal{G})$*-FT gadget if the following holds when the fault path* $\operatorname{supp}\mathbf{f} \subseteq [L]$ *of* $\mathbf{f}$ *is* $\mathcal{G}$*-avoiding: Without loss of generality, assume the gadget operates on the first n qubits. Then, for any* $m \in \mathbb{N}$, *and any state*[13] $\rho \in \mathcal{D}\left(\mathcal{H}_2^{\otimes(k+m)}\right)$, *if the input state* $\sigma \in \mathcal{D}\left(\mathcal{H}_2^{\otimes(n+m)}\right)$ *is* $\mathcal{F}$*-deviated from* $\phi_Q \otimes \mathcal{I}_m(\rho)$, *then the output state* $\sigma' = \mathsf{Gad}[\mathbf{f}] \otimes \mathcal{I}_m(\sigma)$ *is* $\mathcal{F}'$*-deviated from* $(\phi_{Q'} \circ \mathsf{Op}) \otimes \mathcal{I}_m(\rho)$.

A gadget is further said to be friendly *if, regardless of the input state, when* $\operatorname{supp}\mathbf{f}$ *is* $\mathcal{G}$*-avoiding, there always exists a state* $\rho' \in \mathcal{D}\left(\mathcal{H}_2^{\otimes(k+m)}\right)$ *such that the output state is* $\mathcal{F}'$*-deviated from* $(\phi_{Q'} \circ \mathsf{Op}) \otimes \mathcal{I}_m(\rho')$.

Diagramatically, the following diagram commutes: It is equivalent to apply the operation on the logical information and encode it, or to apply the gadget, and the noisy computation is always "close" to the noiseless one.



Remark 2.2.26. *The friendly property may seem somewhat unusual, but it is required in order for gadgets to be recursively simulated: In a recursive simulation, lower level gadgets may fail, leaving no guarantees on the output state. The friendly property allows the gadget following a failed gadget to return the (arbitrarily damaged) state to a well defined logical state, so that the higher simulation level can correct the resulting logical error.*

A set of FT gadgets are said to be *compatible* if the parameters of the blocks of the inputs and outputs are matching (or trivial). That is, for some $(Q, \mathcal{F})$, the inputs and outputs of all gadgets in the set are $(Q, \mathcal{F})$-blocks.

In some cases, it will be more convenient to restrict to gadgets that satisfy the fault-tolerant gadget property for Pauli noise. While we will consider general noise,

[13] The presence of the reference system is needed in order for the gadget to be inserted in a larger circuit which may have many more qubits.

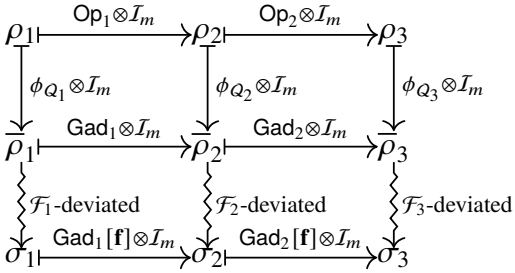
the analysis of a circuit subject to general noise will be reduced to the Pauli noise case, so that the gadgets themselves only need to be fault-tolerant to Pauli noise.

Definition 2.2.27 (Pauli fault-tolerant gadgets)**.** *A Pauli fault-tolerant gadget* Gad *is a gadget satisfying a weaker version of Definition 2.2.25 with all faults restricted to be Pauli faults and the input and output taken to be Pauli deviated.*

What follows is a formalization of properties that allow us to assembly complicated gadgets out of simple ones. The first proposition allows us to compose gadgets. The second proposition allows us to execute two gadgets side-by-side in parallel on separate inputs. The fault analysis in both of these cases will be a union-bound.

Proposition 2.2.28 (Composition of Pauli fault-tolerant gadgets)**.** *For a* $(\mathsf{Op}_1, \mathcal{G}_1)$*-Pauli FT gadget* Gad_1 *mapping a* $(Q_1, \mathcal{F}_1)$*-block to a* $(Q_2, \mathcal{F}_2)$*-block and a* $(\mathsf{Op}_2, \mathcal{G}_2)$*-Pauli FT gadget* Gad_2 *mapping a* $(Q_2, \mathcal{F}_2)$*-block to a* $(Q_3, \mathcal{F}_3)$*-block, the composition is a* $(\mathsf{Op}_2 \circ \mathsf{Op}_1, \mathcal{G}_1 \boxplus \mathcal{G}_2)$*-Pauli FT gadget that maps a* $(Q_1, \mathcal{F}_1)$*-block to a* $(Q_3, \mathcal{F}_3)$*-block.*

Proof. Let $\mathbf{f}$ be a fault with a $\mathcal{G}_1 \boxplus \mathcal{G}_2$-avoiding fault path. Let σ_1 be the input state such that there exists ρ_1 such that σ_1 is $\mathcal{F}_1$-deviated from $\phi_{Q_1}(\rho_1)$. Consider the following diagram:

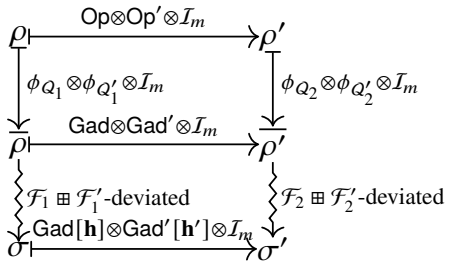


Using the assumption on the input, the only relations that need to be established is that σ_2 is $\mathcal{F}_2$-deviated from $\overline{\rho}_2$ and σ_3 is $\mathcal{F}_3$-deviated from $\overline{\rho}_3$. Using the definition of an FT gadget, the assumption that the fault path is $\mathcal{G}_1$-avoiding, and that σ_1 is $\mathcal{F}$-deviated from $\overline{\rho}_1$ establishes σ_2 is $\mathcal{F}_2$-deviated from $\overline{\rho}_2$. Applying the argument a second time establishes that σ_3 is $\mathcal{F}_3$-deviated from $\overline{\rho}_3$. □

It is also the case that two Pauli FT-gadgets acting in parallel also form a new Pauli FT-gadget. This is the reason for the m-qubit subsystem in the definition of an FT gadget (Definition 2.2.25)

Proposition 2.2.29 (Parallel Pauli FT-gadgets)**.** *For a* $(\mathsf{Op}_1, \mathcal{G}_1)$*-Pauli FT gadget* Gad *mapping a* $(\mathcal{Q}_1, \mathcal{F}_1)$*-block to a* $(\mathcal{Q}_2, \mathcal{F}_2)$*-block and a* $(\mathsf{Op}', \mathcal{G}')$*-Pauli FT gadget* Gad' *mapping a* $(\mathcal{Q}'_1, \mathcal{F}'_1)$*-block to a* $(\mathcal{Q}'_2, \mathcal{F}'_2)$*-block,* $\mathsf{Gad}\otimes\mathsf{Gad}'$ *is a* $(\mathcal{Q}\otimes\mathcal{Q}', \mathcal{G}\boxplus\mathcal{G}')$*-Pauli FT gadget that maps a* $(\mathcal{Q}_1 \otimes \mathcal{Q}'_1, \mathcal{F}_1 \boxplus \mathcal{F}_1)$*-block to a* $(\mathcal{Q}_2 \otimes \mathcal{Q}'_2, \mathcal{F}'_2 \boxplus \mathcal{F}'_2)$*-block.*

Proof. Recall that in the definition of a FT gadget that the properties continue to hold when the gadget is supported on only a subsystem. If $\mathbf{f}$ is a Pauli-fault with a $\mathcal{G} \boxplus \mathcal{G}'$-avoiding fault path then we can write it in terms of the restrictions $\mathbf{f} = \mathbf{h} \otimes \mathbf{h}'$ such that $(\mathsf{Gad} \otimes \mathsf{Gad}')[\mathbf{f}] = \mathsf{Gad}[\mathbf{h}] \otimes \mathsf{Gad}'[\mathbf{h}']$. The claim follows after utilizing the presence of the m-qubit subsystem in the definition to apply the Pauli-FT property to each gadget individually. In other words, we use the definitions show that the following diagram commutes for any $m \in \mathbb{N}$.



□

We are now ready to give an example employing many of the definitions of the last several pages.

Example 2.2.30 (Circuit correctness union bound)**.** *Consider a circuit with classical input and output that is the composition of* V *compatible FT gadgets* $C := \mathsf{Gad}_V \circ \cdots \circ \mathsf{Gad}_2 \circ \mathsf{Gad}_1$ *implementing the classical input-classical output operation* $\mathsf{Op} := \mathsf{Op}_V \circ \cdots \circ \mathsf{Op}_2 \circ \mathsf{Op}_1$ *with bad sets* $\{\mathcal{G}_i\}_{i\in[V]}$*. Let* $\mathbf{f}$ *be a random Pauli fault of* G *distributed according to* ϵ*-local stochastic noise.*

We can apply Proposition 2.2.28 inductively in time[14] *to arrive at a circuit with classical output that is correct if* $\operatorname{supp}\mathbf{f}$ *is* $\mathcal{G} := \mathcal{G}_V \boxplus \cdots \boxplus \mathcal{G}_2 \boxplus \mathcal{G}_1$*-avoiding. Suppose that for all* $i \in [V]$*,* $f(x)$ *is an upper bound for* $\mathcal{W}(G_i;\, x))$ *on some interval of* $[0, 1]$ *that contains* ϵ*. Then, the probability that* $\operatorname{supp}\mathbf{f}$ *is not* $\mathcal{G}$*-avoiding is at most*

$$\Pr(\operatorname{supp}\mathbf{f} \text{ is not } \mathcal{G}\text{-avoiding}) \leq \sum_{S\in\mathcal{G}} \Pr(S \subseteq P) \leq \mathcal{W}(\mathcal{G};\, \epsilon) = \sum_{i\in[V]} \mathcal{W}(\mathcal{G}_i;\, \epsilon)) \leq V \cdot f(\epsilon)$$

[14] If C utilized gadgets in parallel, we would first need to apply Proposition 2.2.29 inductively on each timestep.

where we have used Definition 2.2.16 to upper bound the probability that $\operatorname{supp}\mathbf{f}$ *is not* $\mathcal{G}$*-avoiding and Proposition 2.2.18 to evaluate* $\mathcal{W}(\mathcal{G}; x)$ *as a sum over* $\{\mathcal{W}(\mathcal{G}_i; x)\}_{i\in[V]}$.

For this example, much of the machinery could have been skipped and a standard union bound applied. However, later we will need to use a similar union-bound type argument in combination with a sort of independence utilizing the product operation. That is, it is rare for many gadgets with disjoint locations to fail simultaneously. The weight enumerator polynomial machinery allows us to easily analyze such a scenario which will appear in the distillation of resource states.

Decoherence of errors

In several of our gadgets, we will need to accept states that are Pauli-deviated from the codespace, but the analysis of our error correction (including state distillation) gadgets only considers states that are diagonal Pauli-deviated from the codespace (differs from a codestate by a diagonal Pauli superoperator). We show that measurement of the checks of a quantum code reduces the former case to the latter.

Lemma 2.2.31 (Decoherence of errors)**.** *For a* $[[n, k, d]]$ *stabilizer code* $\mathcal{Q}$*, let* $\mathcal{M}$ *be the channel that measures the* r *stabilizer checks of* $\mathcal{Q}$ *and outputs the measurement outcomes. For a superoperator* $\mathcal{E}_B$ *supported on a recoverable (in the sense of erasure) subset of qubits*[15] $B \subseteq [n]$ *and a codestate* ρ *of* $\mathcal{Q}$*, the application of the noise superoperator followed by measurement of the checks can be written as*

$$\mathcal{M} \circ \mathcal{E}_B(\rho) = \sum_{x\in\mathbb{F}_2^r} \alpha_x \, |x\rangle\langle x| \otimes E_x \rho E_x, \tag{2.11}$$

where each E_x *is a Pauli operator supported on* B *and the* α_x *are complex coefficients. When* $\mathcal{E}_B$ *is a physical noise channel, they satisfy* $\sum_x \alpha_x = 1$*. In other words, measurement of the checks collapses the error into a single Pauli error* as long as we remember the measurement outcome.

Proof. Let $\{S_i\}_{i\in[r]}$ be a generating set for the stabilizer group of $\mathcal{Q}$ such that measurement of S_i produces the i-th syndrome bit. We can write the projector into each syndrome eigenspace $x \in \mathbb{F}_2^r$ as $\Pi_x = \prod_{i\in[r]} \frac{1}{2}(1 + (-1)^{x[i]} S_i)$, so that

$$\mathcal{M}(\rho) = \sum_{x\in\mathbb{F}_2^r} |x\rangle\langle x| \otimes \Pi_x \rho \Pi_x.$$

[15] This can be thought of as $|B| < d$, but, in fact, many sets larger than B are also recoverable. This fact is used for robustness to stochastic noise.

The noise superoperator $\mathcal{E}_B$ can be written as a linear combination of Pauli superoperators supported only on B. Thus, for some Pauli operators $\{K_\mu\}_\mu$, $\{K'_\nu\}_\nu$ and complex coefficients $\{\beta_{\mu\nu}\}_{\mu,\nu}$, we can write

$$\mathcal{E}_B(\rho) = \sum_{\mu,\nu} \alpha_{\mu\nu} K_\mu \rho K'_\nu.$$

We will decompose this representation by separating operators by syndrome. Let σ be the syndrome map from Pauli operators to their syndromes. Recall that ρ is a code-state so that $\Pi_0 \rho \Pi_0 = \rho$. The post-measurement state is

$$\begin{aligned}
\mathcal{M} \circ \mathcal{E}_B(\sigma) &= \sum_{x \in \mathbb{F}_2^r} |x\rangle\langle x| \otimes \sum_{\mu,\nu} \alpha_{\mu\nu} \left(\Pi_x K_\mu \rho K'_\nu \Pi_x \right) \\
&= \sum_{x \in \mathbb{F}_2^r} |x\rangle\langle x| \otimes \sum_{\mu,\nu} \alpha_{\mu\nu} \left(\Pi_x K_\mu \Pi_0 \rho \Pi_0 K'_\nu \Pi_x \right) \\
&= \sum_{x \in \mathbb{F}_2^r} |x\rangle\langle x| \otimes \sum_{\substack{\mu | \sigma(K_\mu) = x \\ \nu | \sigma(K'_\nu) = x}} \alpha_{\mu\nu} \left(\Pi_x K_\mu \Pi_0 \rho \Pi_0 K'_\nu \Pi_x \right).
\end{aligned}$$

The projectors will annihilate any term in the Pauli decomposition that do not have syndrome x, so we can restrict the summation. For any two Pauli operators a, b supported on B ant the same syndrome x, their product ab has no syndrome and is also supported on B. By assumption, B is recoverable from erasure so any Pauli operator with trivial syndrome supported on B must be in the stabilizer. For any codestate of Q the action of an element of the stabilizer is trivial, so that we can write

$$\Pi_x a \Pi_0 = \Pi_x a (ab) \Pi_0 = \Pi_x b \Pi_0.$$

This implies that, for an arbitrary set of Pauli operators $\{E_x\}_x$ supported on B satisfying $\sigma(E_x) = x$, we can write

$$\begin{aligned}
\sum_{\substack{\mu | \sigma(K_\mu) = x \\ \nu | \sigma(K'_\nu) = x}} \alpha_{\mu\nu} \left(\Pi_x K_\mu \Pi_0 \rho \Pi_0 K'_\nu \Pi_x \right) &= \sum_{\substack{\mu | \sigma(K_\mu) = x \\ \nu | \sigma(K'_\nu) = x}} \alpha_{\mu\nu} \left(\Pi_x E_x \Pi_0 \rho \Pi_0 E_x \Pi_x \right) \\
&= \left(\sum_{\substack{\mu | \sigma(K_\mu) = x \\ \nu | \sigma(K'_\nu) = x}} \alpha_{\mu\nu} \right) (E_x \rho E_x).
\end{aligned}$$

The sum in parenthesis is the coefficient in the lemma statement. □

Remark 2.2.32. *It is often said that coherent errors are "decohered" by syndrome measurement into Pauli errors. Lemma 2.2.31 is the formal statement of this fact. The gadgets retain the measurement record in order to "purify" the error and more easily track it. This is also the only point where we utilize our different definition of deviation (Definition 2.2.23) from [AB99] and [AB08]. Equivalently, one could ensure that the logical state of the computation is pure and employ [AB08, Claim 1] before using Lemma 2.2.31.*

2.3 Proof of main result

In this section, we prove the main result. We will first define some notations. Then, we state several lemmas in Section 2.3, most of whose proofs will be deferred to later sections. These include the error correction gadget and the state preparation / distillation gadgets. In Section 2.3, we describe our choice of a set of primitive logical operations, for which the lemmas from Section 2.3 apply, and show how to compile the circuit using these operations. In Section 2.3, we combine the fault-tolerant gadgets and the compilation lemma to prove the main result, albeit with a slightly vanishing threshold due to the 'almost-goodness' of the qLTC family [DLV24]. Finally, in Section 2.3 we obtain the main result with a constant threshold by a concatenation step with the scheme of [YK24].

Let us start by introducing some notations. We will need to refer to sets of qubits at the logical level that are grouped together in some way. We will refer to such groups of qubits as a *register* of qubits. Later, registers will be encoded into error correcting code blocks. With respect to registers, we will refer to an operation as transversal if it acts on every qubit identically. When necessary, we will denote the target register e.g. $H(A)$ or $\mathsf{CNOT}(A, B)$ where A and B should be thought of as "unbound" variables indicated the target register unless otherwise stated. For two-qubit gates, and two registers A and B with qubits coordinates $\{1, \ldots, k\}$ and $\{k+1, \ldots, 2k\}$, a transversal two-qubit gate such as $\mathsf{CNOT}(A, B)$ acts as $\mathsf{CNOT}(1, k+1)\mathsf{CNOT}(2, k+2) \ldots \mathsf{CNOT}(k, 2k)$.

In the following definitions, for an indexed set $[N]$ separated into registers of equal size k_L, $[N] = A \sqcup B \sqcup \ldots$, and for an index $i \in [k_L]$, we use the following abuse of notation $i_A, i_B \in [N]$ to denote an index of the larger set specified by a choice of one of the registers and an index into that register. E.g. if $A = \{1, \ldots, k_L\}$ and $B = \{k_L + 1, \ldots 2k_L\}$, the notation means $i_A = i$ and $i_B = k_L + i$.

We will use qLTC code blocks of the same parameters $[[n_L, k_L, d_L]]$, referred to as

the *computational code*. Throughout, we fix an encoding map of the computational code $\mathcal{E}$ mapping from k_L qubits to n_L qubits that is implementable by a Clifford circuit and compatible with the constant quantum-depth encoding circuit from Lemma 2.4.2. Let $[W]$ be the set of qubits of the simulated circuit C. We will label this Hilbert space (with basis) by $\mathcal{H}_S$ and refer to it as the *simulated Hilbert space*. We further introduce a second Hilbert space $\mathcal{H}_L$ with basis given by m registers of size k_L (such that $mk_L \geq W$) that we refer to as the *logical Hilbert space*. We will label the qubits of $\mathcal{H}_L$ by $[m] \times [k_L]$. Much like how it is standard to decompose an N-qubit Hilbert space into $\mathcal{H}_2^{\otimes N}$, there are two relevant levels of decomposition of the logical Hilbert space $\mathcal{H}_L$: $\mathcal{H}_L = \left(\mathcal{H}_2^{\otimes k_L}\right)^{\otimes m}$ is the tensor product of single-register Hilbert spaces[16] which is further a tensor product of single-qubit Hilbert spaces. Finally, the application of $\mathcal{E}$ to each register of states in $\mathcal{H}_L$ induces a third Hilbert space $\mathcal{H}_P$, the physical Hilbert space.

Deferred lemmas

Quantum LDPC and LTC

Definition 2.3.1 (Computational code)**.** *To save notation, we will define the computational code $Q_L = CSS(H_X, H_Z)$ to be an element of a family of Δ-qLDPC codes. It will have parameters $[[n_L, k_L, d_L]]$, at most r_L rows in H_X and H_Z (the subscript 'L' refers to 'LDPC'), and will be (ρ, Δ)-LTC. For two (to be determined) integers $t_{\text{corr}},\ t_L \in [n]$ with $t_L \leq t_{\text{corr}}/3$, we will further define the family of bad sets $\mathcal{F}_{\text{corr}}$ ($\mathcal{F}_L$) associated with the computational code to be all subsets of $[n_L]$ of size t_{corr} (t_L). Thus,*

$$\mathcal{W}(\mathcal{F}_{\text{corr}};\ x) = \binom{n}{t_{\text{corr}}} x^{t_{\text{corr}}} \tag{2.12}$$

$$\mathcal{W}(\mathcal{F}_L;\ x) = \binom{n}{t_L} x^{t_L}. \tag{2.13}$$

Note that, by construction any set $X \subseteq [n_L]$ of size $|X| \geq t_L$ will contain, as a subset, at least one element of $\mathcal{F}_L$ i.e. $\mathcal{F}_L$ is the smallest set of witnesses for the property $|X| \geq t_L$. One can define similar bad sets for the setting of sublinear-distance and a stochastic noise decoder. However, their definition is more involved as it requires the use of the Δ-qLDPC property (e.g. see the proof of theorem 3 of [Got13]).

[16] This decomposition is particularly important, because the fundamental indivisible unit of space for our compiled circuits will be registers. E.g. quantum operations will act on registers instead of qubits.

With the exception of the statement of Lemma 2.3.6 and within the proofs of some gadgets, all codeblocks of the computational code will be $(\mathcal{Q}_L, \mathcal{F}_L)$ blocks. Note that if a set is $\mathcal{F}_L$-avoiding then it is also $\mathcal{F}_{\text{corr}}$-avoiding, so nearly all statements will hold with $\mathcal{F}_L$ replaced by $\mathcal{F}_{\text{corr}}$.

Later, t_{corr} and t_L will be picked in a way related to the maximum distance for which our bounded-distance decoder succeeds. If we have a stochastic-error decoder, we may also pick the sets $\mathcal{F}_{\text{corr}}$ and $\mathcal{F}_L$ accordingly: The majority of the arguments in the paper do not explicitly depend on the precise choice. Our end goal will be to construct a set of compatible gadgets where all inputs and outputs are $\mathcal{F}_L$-deviated from the codespace. Since some of our intermediate gadgets will create or spread errors, we will construct an error correction gadget that (roughly) takes a state $\mathcal{F}_{\text{corr}}$-deviated from the codespace to a state that is $\mathcal{F}_L$-deviated from the codespace.[17]

To define our gadget, we introduce the standard definition of a single-shot decoder [Bom15; Gu+23] (for adversarial Pauli errors).

Definition 2.3.2 (Single-shot decoding)**.** *Let $\mathcal{Q}$ be a quantum CSS code specified by the parity check matrices $H_X \in \mathbb{F}_2^{r_X \times N}$ and $H_Z \in \mathbb{F}_2^{r_Z \times N}$. Suppose the data state carries the Pauli errors $e_X, e_Z \in \mathbb{F}_2^N$ and the syndrome measurements incur the errors $m_X \in \mathbb{F}_2^{r_Z}$, $m_Z \in \mathbb{F}_2^{r_X}$, such that the associated noisy syndromes are $\tilde{\sigma}_X = H_Z e_X + m_X$ and $\tilde{\sigma}_Z = H_X e_Z + m_Z$. Let $\mathcal{D}$ be a decoding algorithm that receives $\tilde{\sigma}_X$, $\tilde{\sigma}_Z$ and proposes the Pauli corrections f_X, f_Z. The decoder is said to be $(\alpha(N), \beta(N), \gamma(N), \eta)$-single-shot, where $\alpha, \beta, \gamma : \mathbb{N}^+ \mapsto \mathbb{R}^+$ and $0 \leq \eta < 1$, if the following holds. For each $P \in \{X, Z\}$, provided that*

$$|e_P|_R + \alpha(N)|m_P| \leq \beta(N)N,$$

then the proposed Pauli correction f_P satisfies

$$|e_P + f_P|_R \leq \eta |e_P|_R + \gamma(N)|m_P|.$$

Here $|\cdot|_R$ denotes the stabilizer-reduced weight. When $\eta = 0$ we simply say (α, β, γ)-single-shot.

In section 6 of the full paper [NP24] we prove the following decoder result on the code construction from [DLV24].

[17] Actually, the majority of our final gadgets will only be Pauli fault-tolerant gadgets, so the states will satisfy the stronger condition of being Pauli deviated from the code space.

Theorem 2.3.3 (Computational almost-good qLTC with single-shot decoding)**.** *There exists an explicit i-indexed family of CSS quantum LDPC codes with parameters $[[N_i, K_i, D_i]]$, where $K_i = \Theta(N_i)$, $D_i = N_i/\mathrm{polylog}(N_i)$. Any instance in the family is locally testable with soundness parameter $\rho_i = 1/\mathrm{polylog}(N_i)$. Furthermore, the code family admits*

- *A $O(\log N_i)$-time parallel decoder that is $(\alpha, \beta(N_i), \gamma, 0)$-single-shot, where α, γ are constants and $\beta(N_i) = 1/\mathrm{polylog}(N_i)$.*
- *A $O(N_i)$-time sequential decoder with similar single-shot decoding statement.*
- *A $O(1)$-constant time parallel decoder that is $(\alpha, \beta(N_i), \gamma, \eta)$-single-shot, where α, γ are constants, $\beta(N_i) = 1/\mathrm{polylog}(N_i)$, and $\eta < 1$ is a constant.*

In addition, there exists a constant c such that for any $i \in \mathbb{N}^+$, it holds that $\frac{N_{i+1}}{N_i} < c$.

Above, we introduce the extra requirement on the growth rate of the code family. This is a useful condition to achieve constant-space overhead for fault-tolerant computation when the required code block size is large. We refer to code families satisfying this as 'computational code family'. In the rest of the proof we will use the logarithmic-time decoder, although we expect the constant-time decoder may be useful to further reduce the classical time.

Global variables

Here we introduce "global" variables and assumptions that will appear in the remaining sections of the proof. We will pick n_L and k_L in the proof of the main theorem. The remainder of the paper will assume that that $k_L = 2^\nu$ for some integer ν. Since our codes are almost-good, we will introduce n_L-dependence in a very controlled way: Let $\zeta : \mathbb{R}^+ \to \mathbb{R}^+$ be a monotonic invertible function $\zeta(x) = x/\mathrm{polylog}(x)$ such that for all $N \in \mathbb{N}^+$, $\zeta(N) \leq \beta(N)N$ and $\zeta(N) \leq \rho(N)N$. For a constant ϵ, we will use the notation $\widehat{\epsilon}(n_L)$ to mean $\frac{\zeta(n_L)}{n_L}\epsilon$. In all of our results, when substituted with a good qLTC, one should think of $\zeta(n_L)$ as being proportional to n_L, e.g., $n_L/100$.

We will also set

$$t_L = \left\lfloor \frac{\zeta(n_L)}{3 + (2\gamma + 1)^{-1}} \right\rfloor \leq \frac{\beta(n_L)n_L}{3 + (2\gamma + 1)^{-1}} \tag{2.14}$$

$$t_{\mathrm{corr}} = 3t_L. \tag{2.15}$$

We take n_L to be at least a large enough constant such that certain parameters within Lemma 2.3.6 (s) and Lemma 2.4.4 (s, t_{test}) and t_L are large enough (at least 1 or 2).

Resource state preparation

We also require gadgets to prepare resource states. These lemmas are proved in Section 2.4

Lemma 2.3.4 (Stabilizer resource state preparation)**.** *For a constant $b \in \mathbb{N}$, $b < 10$[18], representing the number of blocks, let $\{|\psi_i\rangle\}_{i=1}^{k_L}$ be a set of b-qubit stabilizer states such that $|\psi\rangle := |\psi_1\rangle \otimes \cdots \otimes |\psi_{k_L}\rangle$ is preparable by a constant depth unitary Clifford circuit $\mathsf{Op}_{\text{stab}}$ acting on $|0\rangle^{\otimes bk_L}$ and bk_L ancillas qubits. Then, there exists an $(\mathsf{Op}_{\text{stab}}, \mathcal{G})$-Pauli FT gadget $\mathsf{Gad}_{\text{stab}}$ preparing K copies of $|\overline{\psi}\rangle$, each encoded in b codeblocks of the computational code such that the i-th logical qubits of the b codeblocks is in the state $|\psi_i\rangle$[19] and $\mathcal{W}(\mathcal{G}; x) \leq e^{-\zeta(n_L)+O(\text{polylog}\, n_L)}$ for $x \in [0, \widehat{\epsilon_{*,\text{stab}}}(n_L))$.*

In terms of a variable $M = e^{\Theta(\log n_L \cdot \text{loglog}\, n_L)}$, we have the bound:

$$K = \Theta(M) \tag{2.16}$$

and $\mathsf{Gad}_{\text{stab}}$ has quantum depth $O(\text{polylog}\, n_L)$, width $O(Mn_L)$, and classical depth $O(\text{polylog}\, n_L)$.

The second state preparation gadget creates a magic state for performing CCZ on the first three qubits of a computational code block. Let $\mathsf{Op}_{\text{magic}}$ be the depth-2 circuit that prepares the state $\mathsf{CCZ}(1,2,3)\,|+\rangle^{\otimes k_L}$.

Lemma 2.3.5 (Magic resource state preparation)**.** *There exists an $(\mathsf{Op}_{\text{magic}}, \mathcal{G})$-Pauli FT gadget $\mathsf{Gad}_{\text{magic}}$ preparing K copies of $\left|\overline{\mathsf{CCZ}(1,2,3)}\right\rangle$ such that $\mathcal{W}(\mathcal{G}; x) \leq e^{-\zeta(n_L)+O(\text{polylog}\, n_L)}$ for $x \in [0, \widehat{\epsilon_{*,\text{magic}}}(n_L))$.*

In terms of a variable M satisfying $M = e^{O(\log n_L \cdot \text{loglog}\, n_L)}$, $M = \Omega(n_L)$, we have the bound:

$$K = \Omega(Mn_L^{-\tilde{\gamma}(n_L)}) \tag{2.17}$$

where $\tilde{\gamma}(n_L) = O_{n_L\to\infty}\left(\frac{1}{\text{loglog}\, n_L}\right) = o_{n_L\to\infty}(1)$. $\mathsf{Gad}_{\text{magic}}$ has quantum depth $O(\text{polylog}\, n_L)$, width $O(Mn_L)$, and classical depth $O(\text{polylog}\, n_L)$.

[18] Any absolute constant suffices.

[19] I.e. preparing $\psi_1 \otimes \cdots \otimes \psi_{k_L}$ "coordinate-wise."

Computational code EC gadget

Lemma 2.3.6 (Computational code EC gadget)**.** *There exists a constant* $\epsilon_{*,\mathrm{EC}} \in (0,1)$ *and gadget* $\mathsf{Gad}_{\mathrm{EC}}$ *for the computational code of depth* $2(\Delta+3)$*, width* $n_L + r_L$*, and* $A_{\mathrm{EC}} = 2(\Delta+3)(n_L+r_L)$ *locations such that:*

- $\mathsf{Gad}_{\mathrm{EC}}$ *takes a* $(Q_L, \mathcal{F}_{\mathrm{corr}})$*-block to a* $(Q_L, \mathcal{F}_L)$*-block.*
- *There is family of bad fault paths* $\mathcal{G}_{\mathrm{EC}} \subseteq P([A_{\mathrm{EC}}])$ *such that* $\mathsf{Gad}_{\mathrm{EC}}$ *is a* $(\mathsf{Id}, \mathcal{G}_{\mathrm{EC}})$*-Pauli fault-tolerant gadget (i.e. it performs identity).*
- $\mathcal{G}_{\mathrm{EC}}$ *satisfies* $\mathcal{W}(\mathcal{G}_{\mathrm{EC}};\, x) \leq e^{-\zeta(n_L)}/4$ *on* $x \in [0, \widehat{\epsilon_{*,\mathrm{EC}}}(n_L))$.

Proof. Our circuit and argument will be a standard construction (e.g. [Got13]) of a syndrome extraction circuit for a qLDPC code: Introduce at most r_L ancilla qubits, one for each row of H_X, initialized in $|+\rangle$. We can efficiently find an edge coloring $c : [r_L] \times [n_L] \to [\Delta]$ of the Tanner graph of H_X using Δ colors. In each of $i \in [\Delta]$ steps, we perform CNOT controlled on the ancilla for all pairs of data and ancilla qubits in $c^{-1}(i)$. Finally, perform H and measure the ancilla qubits in the Z basis to obtain a syndrome $\tilde{(\sigma)}_Z$ (detecting Z errors). We repeat this procedure for the Z basis with CNOT replaced by CZ and H_X by H_Z. In the absence of faults, this circuit measures the syndrome of the input state by construction.

We invoke the decoder on the measurement outcomes $(\tilde{\sigma}_X, \tilde{\sigma}_Z)$ to receive the corrections (f_X, f_Z). The final step is to apply the correction $F_c = X^{f_X} Z^{f_Z}$ (i.e. the restriction of F_c to a qubit i is $F|_i = X^{f_X[i]} Z^{f_Z[i]}$).

Here, we will use the code parameters defined in the statement of Theorem 2.3.3. Let $\ell = 2 \cdot 2^{2\Delta}$, and take $\mathcal{G}$ to be all subsets of the spacetime locations of size $s = \left\lfloor \frac{t_L}{\ell(2\gamma+1)} \right\rfloor$.[20] For n_L larger than some constant, s satisfies $s \geq 2$ (Section 2.3). We now prove that this is a Pauli fault-tolerant gadget. Recall from the definition of a Pauli fault-tolerant gadget (Definition 2.2.27), we are considering Pauli input errors and Pauli faults. To avoid cluttering notation, let us neglect the details of the m-qubit reference system in (Definition 2.2.25): Everything that we do here extends straightforwardly to a state entangled with a reference system.

First we restrict to the following case: Suppose that the input state $\tilde{\rho}$ is $\mathcal{F}_{\mathrm{corr}}$-diagonal Pauli deviated from an encoded state $\overline{\rho} = \phi_{Q_L}(\rho)$. That is, $\tilde{\rho} = E_{\mathrm{in}} \overline{\rho} E_{\mathrm{in}}$ for some

[20] Our bound will be very loose as we are only interested in asymptotics.

Pauli operator E_{in} that supported on an $\mathcal{F}_{\text{corr}}$-avoiding set. Further suppose that the Gad_{EC} is subject to a diagonal Pauli fault $\mathbf{f}$ that is $\mathcal{G}$-avoiding.

By construction of $\mathcal{G}$, $\mathbf{f}$ is supported on at most $s-1$ locations, each with at most 2 qubits.

Since Gad_{EC} is Clifford and $\mathbf{f}$ is Pauli, we can "push" the Paulis to the end and write

$$\mathsf{Gad}_{\text{EC}}[\mathbf{f}](\tilde{\rho}) = \mathsf{Gad}_{\text{EC}}[\mathbf{f}](\tilde{\rho}) = \mathbf{f}_{\text{out}}\, \mathsf{Gad}_{\text{EC}}[\mathbf{m}]\; E_{\text{in}}(\overline{\rho}) = \mathbf{f}_{\text{out}}\, E_{\text{in}}\, \mathsf{Gad}_{\text{EC}}[\mathbf{m}\sigma](\overline{\rho}), \tag{2.18}$$

where $\mathbf{m}$ is a Pauli fault stemming from $\mathbf{f}$ that is supported only on measurements and is decomposed into a part supported only on syndrome measurements detecting X errors and syndrome measurements detecting Z errors $\mathbf{m} = \mathbf{m}_X\mathbf{m}_Z$. If (σ_X, σ_Z) is the syndrome of E_{in}, then associating $\mathbf{m}_X$ with a corresponding bitstring m_X, $\tilde{\sigma}_X = \sigma_X + m_x$ and likewise for Z. σ is also a Pauli fault,[21] but it corresponds to the flipped syndrome measurements E_{in} i.e. to the tuple (σ_X, σ_Z). Let us also decompose $E_{\text{in}} = X^{e_X}Z^{e_Z}$.

Since there are at most 2Δ entangling gates, a fault in a single location can propagate to at most ℓ measurements or output qubits, so

$$\begin{aligned} |m_X| &\leq \ell(s-1) \\ |\operatorname{supp}\mathbf{f}_{\text{out}}| &\leq \ell(s-1). \end{aligned}$$

It follows that

$$\begin{aligned} |m_X| + |e_X| &\leq \ell(s-1) + t_{\text{corr}} && (2.19)\\ &= \ell\left(\left\lfloor \frac{t_L}{\ell(2\gamma+1)} \right\rfloor - 1\right) + t_{\text{corr}} && (2.20)\\ &\leq \left(2\gamma+1)^{-1} + 3\right) t_L - \ell && (2.21)\\ &\leq \beta(n_L)n_L. && (2.22) \end{aligned}$$

This is the precondition required so that the decoder (Theorem 2.3.3) succeeds. It returns a correction f_X such that the reduced weight of $f_X + e_X$ is at most $\gamma|m_X| \leq \gamma\ell(s-1)$. That is, there exists an operator $E_{\text{out},X}$ such that $E_{\text{out},X}X^{f_X+e_X}$ is in the stabilizer of the code and $|\operatorname{supp} E_{\text{out},X}| \leq \gamma\ell(s-1)$. An identical argument holds with the roles of Z and X exchanged.

[21] It is a consequence of pushing the input error past the syndrome measurement circuit. We are still measuring the syndrome, but it is unfortunate consequence of our notation that it is technically called a "fault."

Let F_c be the correction applied by the gadget, then since codestates are left invariant by measurement of the stabilizer generators, the output state can be written as

$$\mathsf{Gad}_{\mathrm{EC}}[\mathbf{f}](\tilde{\rho}) = \mathbf{f}_{\mathrm{out}}\, E_{\mathrm{in}}\, \mathsf{Gad}_{\mathrm{EC}}[\mathbf{m}\sigma](\overline{\rho}) = \mathbf{f}_{\mathrm{out}}\, E_{\mathrm{in}}\, F_c(\overline{\rho}) = \mathbf{f}_{\mathrm{out}}\, E_{\mathrm{out},X}\, E_{\mathrm{out},Z}(\overline{\rho}). \tag{2.23}$$

Finally, summing the bounds on the supports, the support of $\mathbf{f}_{\mathrm{out}} E_{\mathrm{out},X} E_{\mathrm{out},Z}$ is at most

$$\begin{aligned}\left|\mathrm{supp}\left(\mathbf{f}_{\mathrm{out}} E_{\mathrm{out},X} E_{\mathrm{out},Z}\right)\right| &\leq (2\gamma+1)\ell(s-1)\\ &\leq (2\gamma+1)\left(\frac{t_L}{2\gamma+1}-1\right)\\ &\leq t_L.\end{aligned}$$

That is, the output is $\mathcal{F}_L$-Pauli deviated from $\overline{\rho}$.

It now remains to upper bound the weight enumerator $\mathcal{W}(\mathcal{G};\, x)$. Recall that $A_{\mathrm{EC}} = \Theta(n_L)$ and $s = \Theta(t_L) = \Theta(\zeta(n_L))$, so we have the bounds

$$\mathcal{W}(\mathcal{G};\, x) = \binom{A_{\mathrm{EC}}}{s} x^s \tag{2.24}$$

$$\leq \left(\frac{Aex}{s}\right)^s \tag{2.25}$$

$$\leq \left((\mathrm{const.})\frac{n_L}{\zeta(n_L)} x\right)^{(\mathrm{const.})\zeta(n_L)}, \tag{2.26}$$

where the terms (const.) depend on Δ and γ only. Thus, there exists a constant $\epsilon_{*,EC} \in (0,1)$ such that for $x \in (0, \widehat{\epsilon_{*,EC}}(n_L))$,

$$\mathcal{W}(\mathcal{G};\, x) \leq \left[(\mathrm{const.})\frac{n_L}{\zeta(n_L)}\widehat{\epsilon_{*,EC}}(n_L)\right]^{(\mathrm{const.})\zeta(n_L)} \tag{2.27}$$

$$\leq \left[(\mathrm{const.})\epsilon_{*,EC}\right]^{(\mathrm{const.})\zeta(n_L)} \tag{2.28}$$

$$\leq e^{-\zeta(n_L)}/4. \tag{2.29}$$

We now remove the assumption on the input state and the fault: Suppose that $\tilde{\rho}$ is now only Pauli-deviated from the codestate $\overline{\rho}$ i.e. $\tilde{\rho} = \mathcal{E}_{\mathrm{in}}(\overline{\rho})$ and that $\mathbf{f}$ is a general Pauli fault. If $\mathbf{f}$ were a diagonal Pauli fault, Lemma 2.2.31 would immediately apply, so that either: 1) $\mathcal{E}_{\mathrm{in}}$ can be picked to be a diagonal Pauli superoperator, or 2) the state is in the kernel of the measurement projector. In the case that $\mathbf{f}$ is not a diagonal Pauli fault, $\sigma\mathbf{m}$ in Eq. (2.18) still must be a diagonal Pauli superoperator

due to the measurement projectors. Thus, after moving $\mathcal{E}_{\text{in}}$ across the syndrome measurement circuit, can only be non-diagonal on a portion of qubits that is in the lightcone of some non-diagonal portion of $\mathbf{f}$. In other words, the argument holds for the general case except that $\mathbf{f}_{\text{out}}$, $E_{\text{out},X}$, and $E_{\text{out},Z}$ in Eq. (2.23) are non-diagonal Pauli superoperators that resulted from the spreading of $\mathbf{f}$ as it was pushed through the circuit (and so are already accounted for). □

We will use the second property in the preparation of noisy resource states to validate that we may do further processing on them before using them in the resource state distillation process.

Lemma 2.3.7 (Computational code transversal gates)**.** *There exists a constant* $\epsilon_{*,\text{OP}} \in (0, 1)$ *such that for any*

$$\mathsf{Op} \in \{\mathsf{CNOT}(A, B), \mathsf{SWAP}(A, B), M_Z(A), M_X(A), \mathsf{PAULI}, \mathsf{Id}\},$$

there exists a gadget $\mathsf{Gad}_{\mathsf{Op}}$ *for the computational code of depth* $O(\Delta)$ *and* $A_{\mathsf{Op}} = O(A_{\text{EC}})$ *locations such that:*

- $\mathsf{Gad}_{\mathsf{Op}}$ *takes* $(\mathcal{Q}_L, \mathcal{F}_L)$*-blocks to* $(\mathcal{Q}_L, \mathcal{F}_L)$*-blocks.*
- *There is family of bad fault paths* $\mathcal{G}_{\mathsf{Op}} \subseteq P([A_{\mathsf{Op}}])$ *such that* $\mathsf{Gad}_{\mathsf{Op}}$ *is a* $(\mathsf{Op}, \mathcal{G}_{\mathsf{Op}})$*-Pauli fault-tolerant gadget.*
- $\mathcal{G}_{\mathsf{Op}}$ *satisfies* $\mathcal{W}(\mathcal{G}_{\mathsf{Op}};\, x) \leq e^{-\zeta(n_L)}$ *on* $x \in [0, \widehat{\epsilon_{*,\text{OP}}}(n_L))$.
- *At most* $\text{polylog}\, n_L$ *depth classical computation is required to decode any measurements.*

Proof. As in the proof of Lemma 2.3.6, we avoid cluttering the notation by neglecting the m-qubit auxillary system in (Definition 2.2.25). However, all steps will be compatible with its presence. We first prove the case of 2-qubit unitary gates on two encoded registers A and B: Our gadget will be to perform the gate transversally on the two blocks followed by an error correction gadget (Lemma 2.3.6) on each of the blocks. Clearly, an encoded $\mathsf{SWAP}(A, B)$ or $\mathsf{CNOT}(A, B)$ can be executed transversally (recall the computational code is CSS). The input state $\tilde{\rho}$ is Pauli deviated from $\overline{\rho} = \phi_{\mathcal{Q}_L} \otimes \phi_{\mathcal{Q}_L}(\rho)$ by an error E_{in} that is $\mathcal{F}_L$-avoiding on each of the two blocks. Define the output state $\rho' = \mathsf{Op}(\rho)$ and $\overline{\rho}' = \phi_{\mathcal{Q}_L} \otimes \phi_{\mathcal{Q}_L}(\rho')$

Let $\mathcal{G}_{\text{gate}}$ be all subsets of size t_L of the transversal gates layer and let $\mathcal{G}_{\text{EC},A}$, $\mathcal{G}_{\text{EC},B}$ be the families of bad fault paths associated with the error correction gadget defined in Lemma 2.3.6 associated with the error correction gadget acting on the A and B blocks, respectively. Our family of bad fault paths is $\mathcal{G}_{\mathsf{Op}} = \mathcal{G}_{\text{gate}} \boxplus \mathcal{G}_{\text{EC},A} \boxplus \mathcal{G}_{\text{EC},B}$. Let $\mathbf{f}$ be a Pauli fault with a $\mathcal{G}_{\mathsf{Op}}$-avoiding fault path. Let us decompose $\mathbf{f}$ into a part supported only on the layer of transversal gates, a part supported only on the A block EC gadget, and a part supported only on the B block EC gadget $\mathbf{f} = \mathbf{f}_{\text{gate}}\mathbf{f}_{\text{EC},A}\mathbf{f}_{\text{EC},B}$.

We proceed to bound the error on the state output by the layer of gates: Since $\operatorname{supp} \mathbf{f}_{\text{gate}}$ is $\mathcal{G}_{\text{gate}}$-avoiding, the support of $\mathbf{f}_{\text{gate}}$ on each block is strictly less than t_L. The support of E_{in} is $\mathcal{F}_L$-avoiding on each of the two blocks ($\mathcal{F}_L \boxplus \mathcal{F}_L$-avoiding), so it also has weight strictly less than t_L on each of them. It follows that the state after the layer of transversal gates is then $E_2\overline{\rho}'E_2$ where E_2 has weight strictly less than $3t_L = t_{\text{corr}}$ on each of the blocks. By construction of $\mathcal{G}$, $\mathbf{f}$ is $\mathcal{G}_{\text{EC}}$-avoiding on each of the two EC gadgets, so we may apply Lemma 2.3.6 to conclude that the output state $\mathsf{Op}(\tilde{\rho})$ is $\mathcal{F}_L \boxplus \mathcal{F}_L$-Pauli deviated from $(\phi_{Q_L} \otimes \phi_{Q_L}) \circ \mathsf{Op}(\rho)$.

For $x \in [0, \epsilon_{*,\text{EC}})$, we can now bound (using Proposition 2.2.18)

$$\begin{aligned}\mathcal{W}(\mathcal{G}_{\mathsf{Op}};\, x) &= 2\mathcal{W}(\mathcal{G}_{\text{EC}};\, x) + \mathcal{W}(\mathcal{G}_{\text{gate}};\, x)\\ &\leq e^{-\zeta(n_L)}/2 + \binom{n_L}{t_L} x^{t_L}\\ &\leq e^{-\zeta(n_L)}/2 + [(\text{const.})x]^{(\text{const.})\zeta(n_L)}\,.\end{aligned}$$

Thus, there exists a constant $\epsilon_{*,\text{OP}} \in (0, \epsilon_{*,\text{EC}}]$ such that for $x \in [0, \epsilon_{*,\text{OP}}]$,

$$\mathcal{W}(\mathcal{G}_{\mathsf{Op}};\, x) \leq e^{-\zeta(n_L)}.$$

The argument for one-qubit unitaries is identical since any logical Pauli operation can be implemented by a corresponding depth-1 pattern of physical Pauli operations.

For measurements, we simply transversally measure in the appropriate basis. Constructing $\mathcal{G}_{\mathsf{Op}}$ in the same way, we can obtain a bitstring that has Hamming distance less than $2t_L/3 \leq \beta(n_L)n_L$ from the codespace. This allows us to invoke the computational code decoder and perform a (classical) decoding to obtain the measurement outcomes. □

Compilation

We will need to turn an arbitrary Clifford+CCZ circuit into one that only uses the primitive operations that we will build gadgets for. Before we give our gate set, we first need to define gates for permutation of qubits.

Register permutation

Because we will distill our resource states in bulk, we may only distill a few types. The number of types will be far fewer than the number of distinct gates that we will need to perform due to the need to address the targets. Thus, we will need a means of targeting the gates our generic resource states perform. To do this, we introduce the notion of a SWAP state. SWAP states perform teleportations of the qubits within a block. By conditionally applying a teleportation circuit consuming a SWAP state, we will obtain one bit of addressing capability. We will use a small sized generating set of size $O(\log k_L)$ to obtain full gate targeting.

Definition 2.3.8 (SWAP State)**.** *Given a set of qubits $[N]$ and a matching $M \subset [N] \times [N]$, the corresponding N-qubit* SWAP *state is a state $|\psi_M\rangle$ where each pair of qubits corresponding to an element of M is a Bell state $|\phi_+\rangle$, and each qubit that does not appear in M is a $|0\rangle$ state. I.e. letting $U = [N] \setminus (\bigcup_{m\in M} m)$, $|\psi_M\rangle$ is the state stabilized by the set of stabilizer generators*

$$\left(\bigcup_{(u,v)\in M} \{X_u X_v, Z_u Z_v\}\right) \bigcup \{Z_u\}_{u\in U}. \tag{2.30}$$

Remark 2.3.9. *Intuitively, before we have the* SWAP *states, we are only capable of coordinate-wise fault-tolerant operations. State injection allows us to prepare low-fidelity states that are entangled between coordinates. Fortunately, distillation of the* SWAP *states we will use requires only coordinate-wise gates and targeted* PAULI *operations, so we can distill these without first having a gate that generates entanglement between coordinates.*

The SWAP state is a product state of Bell states and single qubit states, so we can use this to teleport qubits using the traditional teleportation circuit.

The main operations the SWAP states will implement we call ROT:

Definition 2.3.10 (ROT)**.** *For a register of size k_L, and for any $i \in \mathbb{Z}_{k_L}$, the operation* ROT(i) *implements a cyclic shift of the register by i places. I.e. the qubit at coordinate m goes to the coordinate $m + i \bmod k_L$.*

When multiple register are present, we will subscript by the register that the operation acts on e.g. ROT$_A(i)$ *implements a shift by i on register A.*

Definition 2.3.11 (Permutation states and implementing ROT(i))**.** *Let σ be a permutation on $[k_L]$ and define two registers of k_L qubits A and B. We define the*

corresponding permutation state on AB, $|\Phi_\sigma\rangle_{AB}$ as the SWAP *state defined by the matching*

$$\{(i_A, \sigma[i]_B)\}_{i\in[k_L]}\,. \tag{2.31}$$

Given a k_L-qubit state $|\psi\rangle = \sum_{x\in\mathbb{F}_2^n} \alpha_x |x\rangle$ and a permutation state $|\Phi_\sigma\rangle_{AB}$, the application of a teleportation circuit between $|\psi\rangle$ and the A register will result in the B register being left in the state $\sum_{x\in\mathbb{F}_2^n} \alpha_x |\sigma x\rangle$. I.e. the permutation σ is applied to the state. We call this operation $\mathsf{PERM}(\sigma)$. An example of this state is illustrated in Figure 2.4. For $i \in \mathbb{Z}_{k_L}$, we use $|\mathsf{ROT}(i)\rangle$ to refer to the permutation state with σ such that consumption of the permutation state implements the operation $\mathsf{ROT}(i)$.

Proposition 2.3.12 (Classically controlled permutation)**.** *Given a permutation state $|\Phi_\sigma\rangle_{AB}$, using classically controlled Pauli operations, transversal* CNOT*, and transversal* SWAP*, we can apply classically controlled* $\mathsf{PERM}(\sigma)$ *in quantum depth $O(\log n_L)$ and using $O(1)$ depth auxiliary classical computation.*

Proof. Let C be the input register. We will conditionally apply the standard teleportation circuit transversally between registers A and C: If the control bit is 1, we perform the teleportation circuit $M_Z(C)M_X(A)\mathsf{CNOT}(A, C)$ to yield measurement outcomes $z, x \in \mathbb{F}_2^{k_L}$. For each $i \in [k_L]$, we apply the controlled Pauli operation $X_{\sigma(i)}^{z[i]} Z_{\sigma(i)}^{x[i]}$ to the $\sigma(i)$-th qubit of B. Finally, we peform $\mathsf{SWAP}(B, C)$ to swap register B with register C. If the control bit is 0, then we do nothing for the $O(1)$ timesteps required to execute the other branch of the circuit. □

Proposition 2.3.13 (Arbitrary $\mathsf{SWAP}(i_A, j_B)$)**.** *Using 4ν classically controlled* ROT *operations that are independent of i, j and one $\mathsf{SWAP}(1_A, 1_B)$ operation we can implement $\mathsf{SWAP}(i_A, j_B)$. We can further implement $\mathsf{SWAP}(i_A, j_A)$ using an additional two $\mathsf{SWAP}(1_A, 1_B)$ operations and a second register where the state is arbitrary and preserved.*

Proof. We first show how to implement $\mathsf{ROT}(s)$ for arbitrary $s \in \mathbb{Z}_{k_L}$: Recall that $k_L = 2^\nu$. Write the binary expansion of $s = \sum_{m=1}^{\nu} a_m 2^{m-1}$ where $a_m \in \{0, 1\}$. For each $m \in [\nu]$, implement $\mathsf{ROT}(2^{m-1})$ conditioned on a_m. $\mathsf{ROT}(a)\mathsf{ROT}(b) = \mathsf{ROT}(a+b)$, so the implemented operation is $\prod_{m=1}^{\nu} \mathsf{ROT}(a_m 2^{m-1}) = \mathsf{ROT}\left(\sum_{m=1}^{\nu} a_m 2^{m-1}\right) = \mathsf{ROT}(s)$.

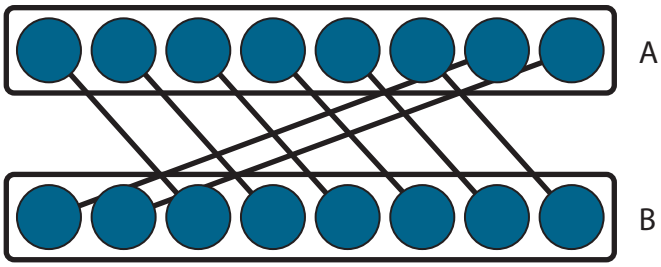


Figure 2.4: Permutation state corresponding to $k_L = 8$ with the permutation $\begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\ 3 & 4 & 5 & 6 & 7 & 8 & 1 & 2 \end{pmatrix}$ i.e. a cyclic shift by 2. We also refer to this state as $|\mathsf{ROT}(2)\rangle$. Lines indicate Bell pairs between registers A and B.

The result for $\mathsf{SWAP}(i_A, j_B)$ follows from the decomposition:[22]

$$\mathsf{SWAP}(i_A, j_B) = \mathsf{ROT}_B(j-1)\mathsf{ROT}_A(i-1)\ \mathsf{SWAP}(1_A, 1_B)\ \mathsf{ROT}_B(1-j)\mathsf{ROT}_A(1-i).$$

We rotate such that the qubits to be swapped are at the first coordinate, swap, and then apply the inverse rotation. For $\mathsf{SWAP}(i_A, j_A)$, we will reduce to a version of the previous case using a scratch register C. We rotate qubit j_A to the first position and then use the register C to hold it while we perform $\mathsf{SWAP}(1+i-j)_A, 1_C$.

$$\begin{aligned} \mathsf{SETUP} &= \mathsf{ROT}_A(j-i)\mathsf{SWAP}(1_A, 1_C)\mathsf{ROT}_A(1-j) \\ \mathsf{SETUP}^{-1} &= \mathsf{ROT}_A(j-1)\mathsf{SWAP}(1_A, 1_C)\mathsf{ROT}_A(i-j) \\ \mathsf{SWAP}(i_A, j_A) &= \mathsf{SETUP}^{-1}\mathsf{SWAP}(1_A, 1_C)\mathsf{SETUP}. \end{aligned}$$

The qubit initially at position j_A follows the sequence of positions (left to right) $(j_A, 1_A, 1_C, 1_C, 1_A, (1+i-j)_A, (1+i-j)_A, i_A)$, and the qubit initially at position i_A follows the sequence of positions $(i_A, (1+i-j)_A, (1+i-j)_A, 1_A, 1_C, 1_C, 1_A, j_A)$. Since register C is never rotated, the qubit initially at position 1_C ends at position 1_C. □

Primitive operations

We now give our primitive gate set that we will construct gadgets for.

Let PAULI denote the set of 1-qubit Pauli operators. We take H, S, and CNOT as a generating set of the Clifford group and denote this set CLIFF. When we augment CLIFF by (projective) Z-basis measurement, we will use $\mathsf{CLIFF}_\mathsf{M}$. We use $\tilde{\mathsf{M}}_\mathsf{P}$ to

[22] Recall that the composition of maps should be read from right to left.

denote standard (“destructive") measurements in the basis P. Whereas we use M_Z to denote computational basis measurements that additionally output a computational basis state initialized to the measurement outcome (“quantum non-demolition"). As a first step, we will decompose any circuit into the gate set $\mathsf{PAULI} + \mathsf{CLIFF}_\mathsf{M} + \mathsf{CCZ}$.

However, we will not implement all of these operations directly. Let A and B be arbitrary registers. Our primitive gateset can be split into three groups. The first group will be used for implementing all other gadgets via teleportation:

- $\tilde{\mathsf{M}}_Z(A)$
- $\tilde{\mathsf{M}}_X(A)$
- $\mathsf{CNOT}(A, B)$
- PAULI

The second group will be used for gate targeting (i.e. addressing logical qubits):

- $i \in [\log_2 k_L]$ $\mathsf{ROT}(2^i)$
- $\mathsf{SWAP}(1_A, 1_B)$

The final group is used only for computation:

- $H(1_A)$
- $S(1_A)$
- $\mathsf{M}_Z(1_A)$
- $\mathsf{CNOT}(1_A, 2_A)$
- $\mathsf{CCZ}(1_A, 2_A, 3_A)$

The second and third groups of gates we call the *primitive computational gates*.

Register assignment

Definition 2.3.14 (Register assignment)**.** *For a circuit C on the set of qubits $[W]$ and $m, k \in \mathbb{N}$ such that $W \leq mk$, an (m, k)-register assignment is an injective map $\phi\colon [W] \to [m] \times [k]$.*

Proposition 2.3.15 (Gate compatible register assignment)**.** *For $k \geq 12$, $m \geq 4W/k$, and one layer C of non-overlapping Clifford + CCZ gates on W qubits, there exists a (m, k)-register assignment $\phi\colon [W] \to [m] \times [k]$ such that each gate in C is supported only on the qubits $\phi^{-1}(a, [k])$ for some $a \in [m]$. Furthermore, this register assignment can be found in time $O(|C|)$.*

Proof. First note that the maximum support size of any gate in C is at most 3. This is an instance of the *bin packing problem* which requires that we assign a large number of parcels taking different sizes to bins such that no bin exceeds its capacity. Our case is that of *high-multiplicity bin packing* for which the number of distinct sizes is $O(1)$, and efficient approximate algorithms are known [FA05].

Here, we describe the first-fit bin packing algorithm [GGU72] with a single open bin: Maintain pointers $a \in [m]$ and $b \in [k]$ initialized to 1, called the open bin and load, respectively. For each gate c in the gate list C:

1. If $\alpha := |\operatorname{supp} c| > k - b + 1$, set $b = 1$ and increment a.
2. Assign the qubits $\operatorname{supp} c$ to positions $\{b, \ldots, b + \alpha\}$ of register a.
3. Set $b = b + \alpha$.

At least $\lfloor \frac{k}{3} \rfloor$ gates can be assigned to each register and there are at most W gates, so at most $\frac{W}{\frac{k}{3}-1} \leq \frac{4W}{k} \leq m$ registers will be assigned to. □

Serialization

Definition 2.3.16 (Serialized circuits)**.** *For a circuit C acting on m registers of size k, an α-serialization of C is a new circuit C' such that, for every timestep, there are at most $\frac{m}{\alpha}$ gates and no two gates act on the same register.*

Clearly, given a 1-serialization, it is straightforward to construct an α-serialization $\alpha \geq 1$ by a greedy assignment when we do not need to worry about gate conflicts.

Proposition 2.3.17 (Serialization subdivision)**.** *Given a* 1*-serialized depth-1 circuit* C *acting on* m *registers of size* k *such that no gate is supported on more than one register and* $\alpha \in [1, m]$ *we can efficiently compute an* α*-serialized circuit* C' *of* C *such that the depth of* C' *is* 2α*.*

Proof. Eagerly execute the gates in C in one of 2α timesteps such that no more than m/α gates are executed in each timestep. In each timestep, we are able to execute $\lfloor m/\alpha \rfloor$ gates. C contains at most m gates since it is 1-serialized. Thus, after 2α timesteps, there are $2\alpha\lfloor m/\alpha \rfloor \geq m$ opportunities to execute a gate.[23] □

We now construct a serialization suitable for general circuits with gates supported on multiple registers.

Lemma 2.3.18 (Serialization)**.** *Given a depth-1 circuit* C *acting on* m *registers of size* k *such that no gate is supported on more than two registers and* $\alpha \in [1, m]$*, we can efficiently compute an* α*-serialized circuit* C' *of* C *such that the depth of* C' *is* $4k\alpha$*.*

Proof. Two gates are said to be *non-conflicting* if they are supported on qubits in different registers. Our procedure will divide the gates of C into $2k$ sets of pair-wise non-conflicting gates, and each set will be executed over $(\alpha + 1)$ timesteps.

Let U be the set of gates in C supported on two registers. Then, consider the multigraph $G = ([m], E)$ where each element $(u, v) \in E$ corresponds to a gate of U with targets in registers (w_1, w_2). G has degree and multiplicity at most k, so a proper edge coloring $c \colon E \to [2k]$ using at most $2k$ colors can be efficiently computed [BF91].

We first partition $C = Q_1 \sqcup Q_2 \cdots \sqcup Q_{2k}$ where, for $i \in [2k]$, Q_i contains all gates of U corresponding to the set of edges $E_i = c^{-1}(i)$ as well as an arbitrarily selected subset of single register gates $C \setminus U$ such that all elements of Q_i are pairwise non-conflicting. There are at most k gates of C supported on each register, and each gate in $C \setminus U$ is supported on one register. Thus, a greedy assignment of elements of $C \setminus U$ to each Q_i will succeed.

For each $i \in [2k]$, we further subdivide $Q_i = \sqcup_{j \in [\alpha+1]} Q_{(i,j)}$ such that each subset has at most m/α elements i.e. $|Q_{(i,j)}| \leq m/\alpha$ using an argument identical to Proposition 2.3.17.

[23] Bounding the cases $\alpha \in [0, m/2]$ and $\alpha \in [m/2, m]$ separately.

C' has timesteps $(i, j) \in [2k] \times [\alpha + 1]$ where in timestep (i, j), we execute the gates in $Q_{(i,j)}$. The precise ordering of the timesteps is arbitrary since all gates in C had disjoint qubit support. □

Lemma 2.3.19 (Compiling)**.** *Assume $k_L \geq 12$ and $m \geq 4W/k_L$. Given a Clifford + CCZ circuit C acting on $\mathcal{H}_S$ of depth T, we can efficiently construct a new circuit C' acting on $\mathcal{H}_L$ of depth T' where*

- *All operations are primitive computational operations (Section 2.3).*
- *Only one operation is supported on each register per timestep. (1-serialized)*
- *Only one type of primitive computational operation is used per timestep.*[24]
- $\frac{T'}{T} = O(\nu k_L) = O(k_L \log k_L)$.
- *C and C' implement the same quantum channel.*

Proof. Without loss of generality, assume that C contains only operations in $\mathsf{CLIFF}_\mathsf{M}$+ CCZ with only one type of gate per layer. Otherwise, we may first replace any two-qubit gate in the Clifford group by a constant number of gates in $\mathsf{CLIFF}_\mathsf{M}$, and then replace each layer by a constant number of layers, each containing only one type of gate per layer.

For each layer C_t of C, we compute a register assignment ϕ_t using Proposition 2.3.15 such that the gates of C_t have support only on one register each. Let $S_t = S_t^{(2)} S_t^{(1)}$ be a depth-2 layer of SWAP gates acting on $\mathcal{H}_L$ that implements the permutation sending qubit $\phi_t(w)$ to $\phi_{t+1}(w)$ for each $w \in [W]$. We will seek to implement the circuit:

$$(C_D)_{\phi_D} S_{D-1} (C_{D-1})_{\phi_{D-1}} \dots S_2 (C_2)_{\phi_2} S_1 (C_1)_{\phi_1} . \tag{2.32}$$

However, we must decompose it further.

For each $v \in \{1, 2\}$, $S_t^{(v)}$ is the product of operations of the form $\mathsf{SWAP}(i_A, j_B)$. We can decompose $S_t^{(v)}$ into primitive computational operations by first constructing the 1-serialization (Lemma 2.3.18) and then further decompose each $\mathsf{SWAP}(i_A, j_B)$ operation into power-of-two ROT and $\mathsf{SWAP}(1_A, 1_B)$ (Proposition 2.3.13). Let S'_t refer to this decomposition of S_t. The depth of S'_t is $O(\nu k_L)$.

[24] We treat ROT with different parameters as distinct types.

Let $C_t^{(k_L)} C_t^{(k_L-1)} \dots C_t^{(1)}$ be the composition of single layers of operations that is a 1-serialization of $(C_t)_{\phi_t}$.[25] For each $\nu \in [k_L]$, we (in parallel) swap the support of each gate in $C_t^{(\nu)}$ (Proposition 2.3.13) to the first coordinates of the register, perform the corresponding primitive computational operation, and swap the support back. Since each gate has support at most 3, this can be done in $O(\nu)$ steps. Let C'_t refer to this sequence of operations (depth $O(\nu k_L)$).

C' is now simply:

$$C' = C'_D S'_{D-1} C'_{D-1} \dots S'_2 C'_2 S'_1 C'_1. \tag{2.33}$$

By construction C' satisfies all conditions of the lemma statement. □

Main theorem with vanishing threshold

Here we now prove the main result with a slightly sub-constant threshold due to our use of an adversarial noise single-shot decoder. It is reasonable to expect that a decoder capable of decoding "most" errors of linear weight exists (a stochastic noise single-shot decoder). However, we find that it is simpler to perform a threshold amplification step (Section 2.3) instead to amplify the threshold to constant.

Theorem 2.3.20 (Main result with vanishing threshold)**.** *There exists a function $f(x)$ growing faster than any* quasipoly(x) *and a value $\epsilon_* \in (0,1)$ such that for any $\epsilon_L \in (0,1)$ and (Clifford+*CCZ*) classical input / classical output quantum circuit C with width W and depth D satisfying*

$$\frac{WD}{\epsilon_L} \leq f(W) \tag{2.34}$$

There is a corresponding efficiently constructable classical input / classical output quantum circuit $\overline{C}$ with width $\overline{W}$ and depth $\overline{D}$ satisfying

$$\frac{\overline{W}}{W} = O_{W\to\infty}(1) \tag{2.35}$$

$$\frac{\overline{D}}{D} = O_{W\to\infty}\left(\left(\log \frac{WD}{\epsilon_L}\right)^{1+o(1)}\right) \tag{2.36}$$

and using auxiliary $O(\text{polyloglog}\,\frac{WD}{\epsilon_L})$-time classical computation per quantum timestep, such that the following guarantees hold. There is a family of bad fault paths $\mathcal{G}$ such that

[25] Since $(C_t)_{\phi_t}$ only contains gates acting on single registers, this is trivial.

- *For any $\mathcal{G}$-avoiding physical fault* $\mathbf{f}$*, the output distribution of* $\overline{C}$ *subject to* $\mathbf{f}$*,* $\overline{C}[\mathbf{f}]$*, is equal to the output distribution of* C*.*
- $\mathcal{W}(\mathcal{G};\, x) \leq \epsilon_L$ *for* $x \in \left[0, \epsilon_* \cdot c\left(\frac{WD}{\epsilon_L}\right)\right]$ *where* $c\left(\frac{WD}{\epsilon_L}\right) = \Omega_{W\to\infty}\left(\frac{1}{\text{polyloglog}\,\frac{WD}{\epsilon_L}}\right)$.

It is an immediate corollary of the above theorem that the output distribution of $\overline{C}$ subject to a random fault distributed according to a p-locally stochastic faults model is ϵ_L-close in total-variation distance (TVD) for $p \leq O\left(\frac{1}{\text{polyloglog}\,\frac{WD}{\epsilon_L}}\right)$. However, maintaining the weight enumerator allows us to work with the circuit further in the next section.

Our proof is split into roughly two parts which we organize into parts for readability that should be read as a single proof.

Proof.

Setup and compilation We use the code family from Theorem 2.3.3. If the number of logical qubits is not a power-of-two, we ignore some fraction $< 1/2$ of them, such that the remaining number of logical qubits is $k_L = 2^\nu$. This harms our rate by at most $1/2$. There exists a minimum (constant) computational code size $n_{\min}$ such that all properties required in Section 2.3 hold.

Since error analysis depends on n_L, we will do it for a general n_L and pick n_L at the end. We will indicate the circuits that appear in the main steps with a superscript (n_L) to denote that n_L has not yet been selected. For now, we simply require $n_{\min} \leq n_L$.

We use a number of registers $m = \lceil W/k_L \rceil$ (using the constant rate). We apply Lemma 2.3.19 to C to arrive at an equivalent circuit $C'^{(n_L)}$ with depth overhead $O(k_L \log k_L)$ which: 1) Only contains primitive computational operations. 2) Only applies one type of primitive computational operation per timestep. 3) At most one primitive computational operation is applied per register per timestep (is 1-serialized).

Fault-tolerant circuit construction $\overline{C}^{(n_L)}$ will have a computational code block for each register of $C'^{(n_L)}$. Let $D' = D \cdot O(k_L \log k_L)$ be the depth of $C'^{(n_L)}$ and for each timestep $t \in [D']$ of $C'^{(n_L)}$, let C'_t be the corresponding layer of gates in $C'^{(n_L)}$. The timesteps of $\overline{C}^{(n_L)}$ will be partitioned into D' *work periods* where, in each work period, we accomplish the work of a single layer of gates of $C'^{(n_L)}$.

Let K be the number of copies of the resource state prepared by a single stabilizer state preparation gadget (Lemma 2.3.4), and define $\beta = \lceil m/K \rceil$ the number of stabilizer state preparation gadgets required to m output states.

We begin the circuit by executing β stabilizers state preparation gadgets to prepare m $\left|\overline{0}\right\rangle^{\otimes k_L}$ states.[26] This takes time $O(\text{polylog}\, n_L)$.

Now, consider a given work period $t \in [D']$. By construction, only a single type of primitive computational operation O (Section 2.3) is applied in C'_t and each register is only involved in at most one primitive computational operation. I.e. there are at most m operations to be performed. In the following, the Id gadget in Lemma 2.3.7 is repeatedly applied to any of the m computational code blocks that are not otherwise involved in a gadget.

If O is a unitary Clifford operation or $\mathsf{M}_Z(1_A)$ (projective computational basis measurement), let $|\mathsf{O}\rangle$ denote the corresponding stabilizer resource state (defined in Section 2.2). This resource state is supported on at most 4 registers, so we can use the stabilizer state preparation gadget (Lemma 2.3.4) to prepare it. We run $\beta = \lceil m/K \rceil$ state preparation gadgets in parallel to produce m copies of the encoded resource state $\left|\overline{\mathsf{O}}\right\rangle$ and then consume it via state teleportation (Lemma 2.3.7) to perform O (Section 2.2). If O is $\mathsf{M}_Z(1_A)$ we additionally store the logically decoded value. This takes time $O(\text{polylog}\, n_L)$ (quantum and classical).

Otherwise, O is $\mathsf{CCZ}(1,2,3)$. We use the magic state preparation gadget (Lemma 2.3.5) to prepare $\left|\overline{\mathsf{CCZ}(1,2,3)}\right\rangle$. Let M_{magic} be the variable from the statement of Lemma 2.3.5, so that a single magic state preparation gadget produces $K_{\text{magic}} = \Omega\left(M_{\text{magic}} n_L^{-\tilde{\gamma}(n_L)}\right)$ copies of $\left|\overline{\mathsf{CCZ}(1,2,3)}\right\rangle$ and uses space $O(M_{\text{magic}} n_L)$. We execute an α-serialized version[27] of C'_t (Proposition 2.3.17) over 2α steps with $\alpha = \lceil M_{\text{magic}}/K_{\text{magic}} \rceil = O\left(n_L^{\tilde{\gamma}(n_L)}\right)$. In each step, we run $\beta_{\text{magic}} = \lceil m/M_{\text{magic}} \rceil$ magic state preparation gadgets to produce the $\frac{m}{\alpha}$ magic states required to execute a single step of the α-serialized circuit. In parallel, we also run 3 sets of β stabilizer state preparation gadgets, preparing m each of the resource states: $\left|\overline{\mathsf{CZ}(1,2)}\right\rangle$, $\left|\overline{\mathsf{CZ}(2,3)}\right\rangle$, and $\left|\overline{\mathsf{CZ}(1,3)}\right\rangle$. For each of the $\frac{m}{\alpha}$ gates in each step, we use these states and the transversal gate gadgets from Lemma 2.3.7 to execute a teleported $\mathsf{CCZ}(1,2,3)$ gate. Overall, this takes time $O(n_L^{\tilde{\gamma}(n_L)}\, \text{polylog}\, n_L)$. Again, while waiting for state

[26] It is notable that our gadget does not require any input $\left|\overline{0}\right\rangle$ states to operate on, so we can use it to initialize the computation.

[27] Recall that C_t is already 1-serialized.

preparation gadgets, the Id gadget in Lemma 2.3.7 is repeatedly applied on each of the m computation data registers.

At the end of the last layer of gates, we output the decoded results of any $\mathsf{M}(1_A)$ gadgets (corresponding to measurements of C) and discard the remainder of the measurement record and classical memory. This is done as follows. On each of the m data registers, we perform physical computational basis state measurement. On each measurement outcome bit string, we first use the parallel Z-syndrome decoder in $O(\log n_L)$ time to remove the possible bit flip errors in the bit string, then we compute the dot product of this bit string with the logical Pauli X strings to decode. The latter step can also be done in $O(\log n_L)$ classical time.

Thus, the overall depth of $\overline{C}^{(n_L)}$ is $O\left(D' n_L^{\tilde{\gamma}(n_L)} \operatorname{polylog} n_L\right) = O\left(D n_L^{1+\tilde{\gamma}(n_L)} \operatorname{polylog} n_L\right)$ with at most $O(\operatorname{polylog} n_L)$ depth classical computation to perform teleported gates.

Fault analysis and reduction to Pauli noise Having constructed the circuit $\overline{C}^{(n_L)}$, we proceed to the fault analysis. Following standard techniques ([AB99],[AGP05],[Got13]), we will reduce the analysis of an arbitrary fault to that of Pauli faults. We will define $p_* = \min(\epsilon_{*\mathrm{EC}}, \epsilon_{*,\mathrm{OP}}, \epsilon_{*\mathrm{stateprep}}, \epsilon_{*\mathrm{Mstateprep}})$ from Lemma 2.3.6, Lemma 2.3.7, Lemma 2.3.4, and Lemma 2.3.5, respectively.

We will denote Pauli faults with a superscript (P). All gadgets used are Pauli fault-tolerant gadgets that accept and output $(C_L, \mathcal{F}_L)$-blocks, so they are compatible.[28] This allows us to inductively apply the gadget composition lemmas (Proposition 2.2.28 and Proposition 2.2.29), so the overall circuit $\overline{C}^{(n_L)}$ is a $(C, \mathcal{G}^{(n_L)})$-Pauli fault-tolerant gadget where $\mathcal{G}^{(n_L)}$ is the sum ($\boxplus$) of the bad fault paths for each gadget. I.e. for a Pauli fault $\mathbf{f}^{(\mathsf{P})}$, the output distribution[29] of $\overline{C}^{(n_L)}[\mathbf{f}^{(\mathsf{P})}]$ is proportional to the output distribution of C when $\mathbf{f}^{(\mathsf{P})}$ is $\mathcal{G}^{(n_L)}$-avoiding.

Let $|\mathsf{input}\rangle$ be the classical input string (as a classical register). For an arbitrary $\mathcal{G}^{(n_L)}$-avoiding physical fault $\mathbf{f}$ (not necessarily Pauli), it can be decomposed into a sum of Pauli faults with the same fault path (operations are component-wise) and complex coefficients α.

$$\mathbf{f} = \sum_{\substack{\text{Pauli faults } \mathbf{g}^{(\mathsf{P})} \\ \operatorname{supp} \mathbf{g}^{(\mathsf{P})} \subseteq \operatorname{supp} \mathbf{f}}} \alpha_{\mathbf{g}^{(\mathsf{P})}} \mathbf{g}^{(\mathsf{P})}.$$

[28] Recall the definition of compatible FT gadgets (Definition 2.2.25).

[29] Distribution is a slight abuse of terminology since, in general, there is an overall complex amplitude that will be summed over at the end. It is at this point that we discard the measurement record of the gadgets and sum over all possible measurement outcomes (see Lemma 2.2.31).

We can now compute the output probability of $\overline{C}^{(n_L)}$ subject to an arbitrary fault. If $\overline{C}^{(n_L)}$ is subject to a Pauli fault $\mathbf{g}^{(\mathsf{P})}$ that is $\mathcal{G}^{(n_L)}$-avoiding, then the output distribution is proportional to C, i.e. $\overline{C}^{(n_L)}[\mathbf{g}^{(\mathsf{P})}] \propto C$. Using linearity, for some real constants c and $\{c_{\mathbf{g}^{(\mathsf{P})}}\}_{\mathbf{g}^{(\mathsf{P})}}$ independent of x, we can write (recall that quantum operations are superoperators)

$$\begin{aligned}
\Pr\left(\overline{C}^{(n_L)}[\mathbf{f}] \text{ outputs } x\right) &= \langle x| \left(\overline{C}^{(n_L)}[\mathbf{f}]\left(|\mathsf{input}\rangle\langle\mathsf{input}|\right)\right)|x\rangle \\
&= \sum_{\substack{\text{Pauli faults } \mathbf{g}^{(\mathsf{P})} \\ \operatorname{supp}\mathbf{g}^{(\mathsf{P})} \subseteq \operatorname{supp}\mathbf{f}}} \alpha_{\mathbf{g}^{(\mathsf{P})}} \langle x| \left(\overline{C}^{(n_L)}[\mathbf{g}^{(\mathsf{P})}]\left(|\mathsf{input}\rangle\langle\mathsf{input}|\right)\right)|x\rangle \\
&= \left(\sum_{\substack{\text{Pauli faults } \mathbf{g}^{(\mathsf{P})} \\ \operatorname{supp}\mathbf{g}^{(\mathsf{P})} \subseteq \operatorname{supp}\mathbf{f}}} \alpha_{\mathbf{g}^{(\mathsf{P})}} \cdot c_{\mathbf{g}^{(\mathsf{P})}}\right) \langle x| \left(C^{(n_L)}\left(|\mathsf{input}\rangle\langle\mathsf{input}|\right)\right)|x\rangle \\
&= c\,\langle x| \left(C^{(n_L)}\left(|\mathsf{input}\rangle\langle\mathsf{input}|\right)\right)|x\rangle\,.
\end{aligned}$$

Since $\mathbf{f}$ is a physical fault (i.e. the channels are CPTP) and C is composed of quantum operations (also CPTP channels), $P(x) = \Pr\left(\overline{C}^{(n_L)}[\mathbf{f}] \text{ outputs } x\right)$ is a normalized probability distribution over bitstrings. Therefore, it must be the case that $c = 1$. We conclude that $\overline{C}^{(n_L)}$ is a $(C, \mathcal{G}^{(n_L)})$-fault-tolerant gadget (under the restriction that the fault is a physical fault) i.e. it is fault tolerant to arbitrary physical faults with a $\mathcal{G}^{(n_L)}$-avoiding fault path.

n_L selection We now upper bound $\mathcal{G}^{(n_L)}$ and pick n_L appropriately. There are at most $A = O(WD\,\mathrm{poly}(n_L))$ gadgets, each with a weight enumerator upper bounded by $p_L(x) \leq e^{-\zeta(n_L)+O(\mathrm{polylog}\,n_L)}$ on $x \in [0, \widehat{\epsilon}_*(n_L)]$. Thus, on $x \in [0, \widehat{\epsilon}_*(n_L)]$,

$$\begin{aligned}
\mathcal{W}(\mathcal{G}^{(n_L)};\, x) &\leq A p_L(x) \\
&\leq O(WD\,\mathrm{poly}(n_L)) e^{-\zeta(n_L)+O(\mathrm{polylog}\,n_L)} \\
&\leq WD e^{-\zeta(n_L)+g(n_L)}.
\end{aligned}$$

for some $g(n_L) \leq O(\mathrm{polylog}(n_L))$ Let y be the smallest solution to

$$\zeta(y) \geq \log \frac{WD}{\epsilon_L} + g(y). \tag{2.37}$$

Since $\zeta(x) = x/\mathrm{polylog}\,x$ and $g(n_L) \leq O(\mathrm{polylog}(n_L), y = \Theta\left(\log \frac{WD}{\epsilon_L} \cdot \mathrm{polyloglog}\,\frac{WD}{\epsilon_L}\right)$. Now let n_L' to be the smallest element of the computational code N_i greater than

$\min(n_{\min}, y)$. Since our computational code satisfies $N_{i+1}/N_i = C$ for some constant $C > 1$, we have the bounds

$$n'_L = \Theta\left(\log \frac{WD}{\epsilon_L} \cdot \operatorname{polyloglog} \frac{WD}{\epsilon_L}\right). \tag{2.38}$$

We now set $\overline{C} = \overline{C}^{(n'_L)}$. So that for $x \in [0, \widehat{\epsilon}(n_L)_*]$,

$$W_{\mathcal{G}}(x) \leq \epsilon_L \ .$$

$$\widehat{\epsilon}_*(n_L) = \epsilon_* \frac{\zeta(n'_L)}{n'_L} = \epsilon_* \cdot \Omega\left(\frac{1}{\operatorname{polyloglog} \frac{WD}{\epsilon_L}}\right)$$

Using the upper bound on $\frac{WD}{\epsilon_L}$, $m \leq W$, and the space bounds of the gadgets, it follows that

$$\begin{aligned}
\frac{\overline{W}}{W} &= \frac{1}{W} O\left(\beta M + \beta_{\text{magic}} M_{\text{magic}} + m n_L\right) && (2.39)\\
&= \frac{1}{W} O\left(\left(\frac{m}{K} + 1\right) e^{\Theta(\log n_L \cdot \operatorname{loglog} n_L)} + \left(\frac{W}{n_L} + 1\right) n_L\right) && (2.40)\\
&= O\left(\frac{e^{\Theta(\log n_L \cdot \operatorname{loglog} n_L)}}{W} + \frac{n_L}{W} + 1\right) && (2.41)\\
&= O\left(\frac{\operatorname{quasipoly} n_L}{W} + 1\right) && (2.42)\\
&= O\left(\frac{\operatorname{quasipolylog} \frac{WD}{\epsilon_L}}{W} + 1\right) && (2.43)
\end{aligned}$$

We now select $f(x)$. There exists absolute constants $c \in (0, 1)$ and $c' \geq 0$ such that we can define $f(x) = \exp\left(\exp\left(c'(\log x)^c\right)\right)$ to be a quickly growing function such that when $\frac{WD}{\epsilon} \leq f(W)$, the right hand side of Eq. (2.43) is $O(1)$. We note that $f(x)$ grows faster than any $\operatorname{quasipoly}(x)$ but slower than any $\exp(\operatorname{poly}(x))$. Using $\frac{WD}{\epsilon} \leq f(W)$, we have that

$$\frac{\overline{W}}{W} = O(1) \ .$$

We move on to the time bound which is

$$\frac{\overline{D}}{D} = O\left(\left(\log \frac{WD}{\epsilon_L}\right)^{1+\tilde{\gamma}\left(\tilde{\Theta}\left(\log \frac{WD}{\epsilon_L}\right)\right)}\right)$$

where $\tilde{\Theta}(\cdot)$ suppresses doubly-logarithmic factors and $\tilde{\gamma}(n) = o(1)$ is defined in the proof of Lemma 2.3.5. □

Main theorem

Having established that we can achieve nearly-logarithmic time overhead and constant space overhead with a threshold that vanishes as $\frac{1}{\text{polyloglog}\frac{WD}{\epsilon_L}}$ (Theorem 2.3.20), the last task is to amplify our threshold to a constant. One (nearly trivial) option is to use the AB concatenated code construction again (as defined in Section 2.4) with $r \approx \log\log\log\log\frac{WD}{\epsilon_L}$ where gates and qubits in $\overline{C}$ are replaced by their concatenated code counterparts. Proposition 2.2.21 will give us an upper bound on the weight enumerator of the bad fault paths of the resulting circuit that allows us to establish a constant threshold. However, this introduces a space and time overhead of $O\left(\text{polylogloglog}\frac{WD}{\epsilon}\right)$ and we desire $O(1)$ space overhead.

The strategy will be to simulate our circuit with the concatenated codes of [YK24]. Intuitively, the simulated circuit will see a noise model that is not very different from the independent noise. A precise implementation of this program will be the subject of an upcoming work [HNP] that greatly expands the weight enumerator formalism from Section 2.2 to include "extended rectangles" (ExRecs) and to prove an extension of Proposition 2.2.28 that gives a direct proof of the following claim.

For now, we state the following implication of [YK24] within the framework of our formalism. A proof of Claim 2.3.21 without ExRecs (using definitions from Section 2.2) can be accomplished using the standard gadgets from [YK24] or [AB99] on the code family with parameters $r \in \mathbb{N}$, $[[(2^r-1)^2, (2^r-r-1)^2, 9]]$ given by concatenating each quantum Hamming code with itself once and then forgetting about the concatenated structure. This is essentially standard, so we omit these details.

We remark that [YK24] uses the gate set Clifford+T. CCZ can be exactly simulated by Clifford+T, so this distinction is not important in our use.

For circuit on a set of qubits partitioned into registers, a *register location* is the straightforward generalization of location with qubits replaced by registers. Likewise, we generalize fault paths to *register fault paths*. Occasionally, we will promote a fault path to a register fault path in the canonical way by replacing each location by the register location that it is supported in.

Claim 2.3.21 (Modified version of [YK24])**.** *There exists a constant value*[30] $\epsilon_{*,YK} \in (0,1)$*, such that: for* $r \in \mathbb{N}$ *and a classical input-classical output circuit* C *using* W *qubits and depth* D*, and a partitioning of qubits into registers of size* $k_{\text{YK}} = e^{\Theta(r^2)}$*.*

[30] Think of $\epsilon_{*,YK}$ as something like $1/2$ the threshold value computed in [YK24].

- *There is a new circuit $\overline{C}$ with width $\overline{W}$ and depth $\overline{D}$ such that $\overline{W}/W = O(1)$ and $\overline{D}/D = O(\text{poly } k_{\text{YK}})$ constructed out of gadgets of [YK24] that act only on single and pairs of registers at a time and take a single work period of size* poly k_{YK}.

- *Let Ω be the set of register locations of $\overline{C}$.*

- *This circuit is equipped with families of bad fault paths parameterized by register fault paths $\{\mathcal{G}_P\}_{P\subseteq\Omega}$ such that if a physical fault $\mathbf{f}$ is $\mathcal{G}_P$-avoiding, then the output distribution of $\overline{C}[\mathbf{f}]$ is equal to the output distribution of $C[\mathbf{g}]$ for some register physical fault $\mathbf{g}$ with a $\{P\}$-avoiding register fault path.*

- *There is a function $p_{\text{YK}}(x)$ that satisfies $p_{\text{YK}}(x) \leq e^{-O(2^r)}$ on $x \in [0, \epsilon_{*,YK}]$ such that $\mathcal{W}(\mathcal{G}_P;\ x) \leq (p_{\text{YK}}(x))^{|P|}$.*

Remark 2.3.22 (Threshold amplification using qLDPC codes)**.** *A natural question is why constant space overhead qLDPC code constructions are unsuitable for the threshold amplification step. Conveniently, because the inner codes used for the amplification step are very small, techniques that gave unacceptably high time-overhead in the outer fault-tolerance scheme can be used. Instead of using the large distillation gadgets as in Theorem 2.3.20, one could prepare resource states using a concatenated code as in [Got13] with many small sized qLDPC codes (see Section 2.1).*

The reason this does not immediately work turns out to be somewhat subtle: In order to simulate an outer fault-tolerance scheme, it must be the case that the behavior of the simulating circuit is well defined with respect to failures of the inner fault-tolerance gadgets. In order for this to be the case, the gadgets must be capable of accepting an arbitrarily damaged state. This is the behavior the friendliness property in Definition 2.2.25 captures and is the reason why we needed to construct the qLTC tester in Section 2.4. One should note that standard single-shot error-correction gadgets are not friendly. *We are aware of two ways to establish friendly error correction gadgets which we briefly outline here.*

- *The first method is to measure the syndrome of the qLDPC code a number of times equal to the distance d as in [Got13] and decode the resulting spacetime syndrome. Then, regardless of the input state, the output state is sparsely deviated from a code state. Unfortunately, in most cases, an efficient decoder for the resulting spacetime code is not known (it is unknown whether a single*

shot decoder implies a decoder for the spacetime code). A notable exception is the minimum-weight perfect matching decoder for the surface code [DKLP02]. Alternatively, one can simply use a brute-force approach since the codes are of extremely small size.

- *The second method to obtain friendly gadgets is to utilize a qLTC. Given a source of* $|0\rangle$ *states, the error correction gadget in Lemma 2.3.6 can be made friendly by applying the tester gadget Lemma 2.4.4 before each error correction gadget. In the case where the tester rejects, we swap in the known-good* $|0\rangle$ *state as in [AB99]. The preparation of logical* $|0\rangle$ *states using a qLTC is straightforward: Prepare a zero product state and measure the checks of the code. Use a bit-flipping decoder for the* syndrome error*, and then finally use Gaussian elimination to solve for a correction that satisfies the measured syndrome. The resulting state has small syndrome and hence must be close to the code space by local-testability.*

We are now ready to state and prove the main result.

Theorem 2.3.23 (Main result)**.** *There exists a function* $f(x)$ *growing faster than any* $\text{quasipoly}(x)$ *and a value* $\epsilon_* \in (0,1)$ *such that for any* $\epsilon_L \in (0,1)$ *and (Clifford+*CCZ*) classical input / classical output quantum circuit* C *with width* W *and depth* D *satisfying*

$$\frac{WD}{\epsilon_L} \leq f(W) \tag{2.44}$$

There is a corresponding efficiently constructable classical input / classical output quantum circuit $\overline{C}$ *with width* $\overline{W}$ *and depth* $\overline{D}$ *satisfying*

$$\frac{\overline{W}}{W} = O_{W\to\infty}(1) \tag{2.45}$$

$$\frac{\overline{D}}{D} = O_{W\to\infty}\left(\left(\log \frac{WD}{\epsilon_L}\right)^{1+o(1)}\right) \tag{2.46}$$

and using auxiliary $O(\text{polyloglog}\,\frac{WD}{\epsilon_L})$*-time classical computation per quantum timestep, such that the following guarantees hold. For a random physical fault* $\mathbf{f}$ *distributed according to* ϵ*-locally stochastic faults model with* $\epsilon \in [0, \epsilon_*]$*, the output distribution of* $\overline{C}$ *subject to* $\mathbf{f}$ *is* ϵ_L*-close in TVD to the output distribution of* C*.*

Construction *Proof.* First we construct a circuit $\overline{C}_{\text{qLDPC}}$ using Theorem 2.3.20. This has a family of bad fault paths $\mathcal{G}_{\text{qLDPC}}$ such that if the physical fault $\mathbf{f}$ is $\mathcal{G}_{\text{qLDPC}}$-avoiding, then the output distribution of $\overline{C}_{\text{qLDPC}}[\mathbf{f}]$ is equal to the output distribution of C, and $\mathcal{W}(\mathcal{G}_{\text{qLDPC}};\, x) \leq \epsilon_L$ for $x \in [0, \epsilon_{*,\text{LDPC}} \cdot c]$ where $c = \Omega\left(\frac{1}{\text{polyloglog } \frac{WD}{\epsilon_L}}\right)$.

We will simulate this circuit using Claim 2.3.21 with $r = \Theta\left(\log\log \frac{1}{\epsilon_{*,\text{LDPC}} c}\right)$ in a particular way (We will only pick r at the end): Recall that the blocks of Theorem 2.3.20 are of size at most $O\left(\log \frac{WD}{\epsilon_L} \cdot \text{polyloglog } \frac{WD}{\epsilon_L}\right)$ and the largest gadget is of size at most $J = O\left(\text{quasipolylog } \frac{WD}{\epsilon_L}\right)$. We will arrange the qubits of $\overline{C}_{\text{qLDPC}}$ into registers of size $k_{\text{YK}} = \exp\left[\Theta(\log\log\left(\frac{1}{\epsilon_* c}\right)^2\right]$ such that no two qubits participating in a gadget (of $\overline{C}_{\text{qLDPC}}$) are assigned to the same register.

Let W_{qLDPC} be the number of qubits in $\overline{C}_{\text{qLDPC}}$. To compute the register assignment, for each timestep t of $\overline{C}_{\text{qLDPC}}$, we compute a greedy vertex coloring on the graph $G_t = ([W_{\text{qLDPC}}], E_t)$ where two vertices $i, j \in [W_{\text{qLDPC}}]$ share a vertex if they participate in the same gadget. G has degree at most J, so there is a vertex coloring $c_t \colon [W_{\text{qLDPC}}] \to [J+1]$ with at most $J+1$ colors.

We use $J+1$ groups of (potentially many) YK registers corresponding to the $J+1$ colors. At time t, we assign each qubit $q \in [W]$ to occupy a register with color $c_t(q)$. Between timesteps t of $\overline{C}_{\text{qLDPC}}$, we swap qubits between registers in parallel to maintain the coloring constraint. This adds $\text{poly}(k_{\text{YK}})$ time overhead. Now, we apply Claim 2.3.21 to this circuit[31] to arrive at the circuit $\overline{C}_n$ which is deeper by a factor $O(\text{poly}\, k_{\text{YK}})$.

Fault analysis Let $\mathcal{G}^{(\text{reg})}_{\text{qLDPC}}$ be a family of bad register fault paths where each fault path in $\mathcal{G}_{\text{qLDPC}}$ has been replaced by the corresponding register fault path with respect to the register partitioning. By ensuring that no two qubits participating in a qLDPC gadget are assigned to the same register, it is the case that $\mathcal{W}(\mathcal{G}^{(\text{reg})}_{\text{qLDPC}};\, x) = \mathcal{W}(\mathcal{G}_{\text{qLDPC}};\, x)$. Let $\{\mathcal{G}^{(\text{YK})}_P\}_P$ be the parameterized families of bad fault paths from the statement of Claim 2.3.21. Now consider the fault set

$$\mathcal{G} = \boxplus_{P \in \mathcal{G}^{(\text{reg})}_{\text{qLDPC}}} \mathcal{G}^{(\text{YK})}_P. \tag{2.47}$$

A $\mathcal{G}$-avoiding fault $\mathbf{f}$ is $\mathcal{G}^{(\text{YK})}_P$-avoiding for every register fault path $P \in \mathcal{G}^{(\text{reg})}_{\text{qLDPC}}$. Suppose that $\mathbf{f}$ is a $\mathcal{G}$-avoiding physical fault, then (by Claim 2.3.21) there exists

[31] Actually, to a serialization of this circuit, but this is not important to the argument.

a physical fault $\mathbf{g}$ for which its register fault path is $\mathcal{G}^{(\mathrm{reg})}_{\mathrm{qLDPC}}$-avoiding such that the output distribution of $\overline{C}[\mathbf{f}]$ is the same as $\overline{C}_{\mathrm{LDPC}}[\mathbf{g}]$. Now, by construction of $\mathcal{G}^{(\mathrm{reg})}_{\mathrm{qLDPC}}$, if $\mathbf{g}$ has a register fault path that is $\mathcal{G}^{(\mathrm{reg})}_{\mathrm{qLDPC}}$-avoiding, then its fault path must be $\mathcal{G}_{\mathrm{qLDPC}}$-avoiding, so the output distribution of $\overline{C}_{\mathrm{LDPC}}[\mathbf{g}]$ is equal to the output distribution of C.

Now let $\mathbf{f}$ be a random physical fault distributed according to an ϵ-locally-stochastic faults model. Recall that this means that for any fault path S, the probability that the fault path of $\mathbf{f}$ contains S is at most $\epsilon^{|S|}$. We would like to bound the total-variation distance between the output distribution of $\mathbb{E}_{\mathbf{f}}\overline{C}[\mathbf{f}]$ and that of C. We recall that the two distributions are equal conditioned on the event $\mathbf{f}$ is $\mathcal{G}$-avoiding, so

$$\begin{aligned}
|\Pr\left(\overline{C}[\mathbf{f}] \text{ outputs } x\right) - \Pr(C \text{ outputs } x)| &\leq |\Pr_{\mathbf{f}}(\mathbf{f} \text{ is not } \mathcal{G}\text{-avoiding})| \\
&\leq \sum_{S\in\mathcal{G}} \Pr(\operatorname{supp}\mathbf{f} \subseteq S) \\
&\leq \sum_{S\in\mathcal{G}} \epsilon^{|S|} = \mathcal{W}(\mathcal{G};\ \epsilon)\ .
\end{aligned}$$

I.e. $\mathcal{W}(\mathcal{G};\ \epsilon)$ upper bounds the TVD between the two distributions.

Error rate It remains to upper bound $\mathcal{W}(\mathcal{G};\ \epsilon)$. Since $\epsilon \in [0, \epsilon_{*,\mathrm{YK}}]$, we have (using Eq. (2.47), Proposition 2.2.18, and Claim 2.3.21)

$$\begin{aligned}
\mathcal{W}(\mathcal{G};\ \epsilon) &\leq \sum_{P\in\mathcal{G}^{(\mathrm{reg})}_{\mathrm{qLDPC}}} \mathcal{W}(\mathcal{G}^{(\mathrm{YK})}_{P};\ \epsilon) \\
&\leq \sum_{P\in\mathcal{G}^{(\mathrm{reg})}_{\mathrm{qLDPC}}} (p_{\mathrm{YK}}(\epsilon))^{|P|} \\
&\leq \mathcal{W}(\mathcal{G}^{(\mathrm{reg})}_{\mathrm{qLDPC}};\ p_{\mathrm{YK}}(\epsilon))
\end{aligned}$$

We now select the minimum r such that $p_{\mathrm{YK}}(\epsilon) \leq \frac{1}{\epsilon_{*,\mathrm{LDPC}}c}$, so that

$$\mathcal{W}(\mathcal{G};\ \epsilon) \leq \mathcal{W}(\mathcal{G}^{(\mathrm{reg})}_{\mathrm{qLDPC}};\ p_{\mathrm{YK}}(\epsilon)) \leq \mathcal{W}(\mathcal{G}^{(\mathrm{reg})}_{\mathrm{qLDPC}};\ \frac{1}{\epsilon_{*,\mathrm{LDPC}}c}) \leq \epsilon_L\ .$$

$p_{\mathrm{YK}}(\epsilon) = e^{-O(2^{-r})}$, so $r = O\left(\mathrm{logloglogloglog}\,\frac{WD}{\epsilon_L}\right)$ suffices.

Circuit size Let $f'(x)$ be the function from the statement of Theorem 2.3.20. If $\frac{WD}{\epsilon_L} \leq f'(W)$ then $\frac{W_{\mathrm{qLDPC}}}{W} = O(1)$. Note that for our choice of r, $k_{\mathrm{YK}} = e^{O\left(\left(\mathrm{logloglogloglog}\,\frac{WD}{\epsilon_L}\right)^2\right)}$. $(J+1)k_{\mathrm{YK}}$ is the minimum number of registers required to

assign each qubit of a gadget of $\overline{C}_{\text{qLDPC}}$ to distinct YK registers, so the space overhead of $\overline{C}$ relative to $\overline{C}_{\text{qLDPC}}$ is at most

$$O\left(1 + \frac{Jk_{\text{YK}}}{W_{\text{qLDPC}}}\right) = O\left(1 + \frac{\text{quasipolylog}\,\frac{WD}{\epsilon_L}}{W_{\text{qLDPC}}}\right) \tag{2.48}$$

We now select $f(x)$. As in Theorem 2.3.20, there exists absolute constants $c \in (0,1)$ and $c' \geq 0$ such that we can define $f(x) = \exp(\exp(c'(\log x)^c))$ to be a quickly growing function such that 1) $f(x) \leq f'(x)$ and 2) when $\frac{WD}{\epsilon} \leq f(W)$, the right hand side of Eq. (2.48) is $O(1)$. $f(x)$ again grows faster than any quasipoly(x)

Finally, relative to $\overline{C}_{\text{qLDPC}}$, $\overline{C}$ has depth overhead

$$\text{poly}\, k_L = O\left(e^{O\left(\left(\text{logloglog}\,\frac{WD}{\epsilon_L}\right)^2\right)}\right)$$

proving the theorem with $\epsilon_* = \epsilon_{*,\text{YK}}$. □

2.4 State preparation gadgets

In this section, we develop state preparation gadgets for both logical stabilizer states (Lemma 2.3.4) and magic states (Lemma 2.3.5) of the computational qLTC. As described in the introduction section, the main idea is to (1) use existing concatenated-code fault tolerance schemes to prepare the codestates to a constant logical fidelity, and then (2) apply state distillation schemes to boost the fidelity to an arbitrary target error. We will first review the concatenated-code fault tolerance proof of Aharonov and Ben-Or [AB99] and describe how to use it to prepare qLTC codestates to constant fidelity in Section 2.4. Next, in Section 2.4 we describe how to perform state distillation protocols (whose constructions are deferred to later sections) at the logical level of the computional qLTC code to improve the logical fidelity. In Section 2.4, we present our stabilizer state distillation scheme with constant-space and polyloglog-time overheads. The magic state distillation scheme with almost-constant space and polyloglog-time overheads is deferred to Section 2.5.

Concatenated code FT scheme

We now briefly introduce the scheme of concatenated code fault-tolerance scheme of Aharonov and Ben-Or [AB99] which uses distance-9 CSS codes and Clifford+CCZ [32] We use this scheme over the similar [AGP05] to avoid introducing the analog

[32] More precisely they use Clifford+Toffoli gate set, but they use self-dual quantum codes with a transversal Hadamard and hence essentially the same scheme works for the Clifford+CCZ gate set.

of exRecs for the weight enumerators which makes the notation burdensome. We remark that we are using a modified definition of deviation from [AB99], but their argument is unmodified (e.g. [Kit97]). Nearly any concatenated code scheme suffices. The code family will be parameterized by a parameter $r \in \mathbb{N}$ corresponding to a number of concatenation levels.

Claim 2.4.1 ([AB99])**.** *There is a family of concatenated codes $Q_{\mathrm{AB},r}$ with compatible and friendly gadgets for Clifford+CCZ with the property that: There exists constants $W_{\mathrm{AB}},\ A_{\mathrm{AB}},\ B_{\mathrm{AB}},\ c_{\mathrm{AB}} > 0$ such that*

- *Each code block holds one logical qubit.*
- *The size of a block is at most B_{AB}^r.*
- *The number of locations in a gadget is at most A_{AB}^r.*
- *The maximum number of qubits used at any point in time in a gadget is at most W_{AB}^r.*
- *The gadgets are compatible and friendly as in Definition 2.2.25.*
- *For each gadget, the corresponding bad fault sets $\mathcal{G}$ have weight enumerator $\mathcal{W}(\mathcal{G};\, x) \leq (c_{\mathrm{AB}} x)^{2^r}$ on $[0, 1)$.*

we will refer to the r-th element of this family of code and gadgets as r-AB.

Noisy state preparation

We begin by describing a fault-tolerant gadget that prepares code states of qLDPC codes. The main idea is to use the concatenated-code scheme [AB99] to prepare the qLDPC codestate while being encoded in an outer concatenated code, and then unencode the outer code. This idea originates from Gottesman's seminal work [Got13], but our goal here differs from his work. While Gottesman used this gadget to prepare the codestates to a low error rate that is inverse polynomial in the computation size, we only do so to a sufficiently small constant error rate. This choice is crucial to obtain the main overhead result of this paper, as concatenated-code FT induces a polylog$(1/\varepsilon)$ overhead that prevents us from choosing ε to be too small. Additionally, we present a fault tolerance proof of the unencoding procedure, which was omitted in [Got13]. The main lemma proved in this subsection is Lemma 2.4.5.

Constant depth qLDPC encoding

We start by showing how to prepare simple qLDPC codestates (all ancilla states used in this paper satisfy the precondition below) non-fault-tolerantly in constant depth.

Lemma 2.4.2 (Constant-depth qLDPC encoding [Got13])**.** *Let $|\psi\rangle$ be a k-qubit state that can be prepared by a (Clifford+CCZ) quantum circuit of constant depth. Then the encoding of $|\psi\rangle$ into a $[[n,k]]$ CSS Δ-qLDPC code can be done by a non-fault-tolerant procedure using $O(n)$ qubits in quantum depth $O(\Delta)$ and $O(\log^2 n)$ classical depth.*

Proof. Let m_X, m_Z be the number of independent X and Z checks, so that $k = n - m_X - m_Z$. First, we observe that for any $[[n,k]]$ CSS code, up to qubit permutations, it is possible to choose the logical Pauli operators to be of the form $\overline{X}_i = X_i \otimes P_i \otimes P'_i$ and $\overline{Z}_i = Z_i \otimes Q_i \otimes Q'_i$ for each $i \in [k]$, where X_i (Z_i) acts on the i-th physical qubit, P_i and Q_i are Pauli X strings on m_Z qubits, and P'_i, Q'_i are Pauli Z strings on m_X qubits[33]. Indeed, we can perform Gaussian elimination (and possibly qubit permutations) to put the check matrices into the canonical form [Got97, Section 4], $S_X = (A_1|A_2|\,\mathbb{1}_{m_X})$ where $A_1 \in \mathbb{F}_2^{m_X \times k}$, $A_2 \in \mathbb{F}_2^{m_X \times m_Z}$, and $S_Z = (B_1|\,\mathbb{1}_{m_Z}\,|B_2)$ where $B_1 \in \mathbb{F}_2^{m_Z \times k}$, $B_2 \in \mathbb{F}_2^{m_Z \times m_X}$. Next, we imagine applying the Hadamards $H^{\otimes m_z}$ on the middle block. This sounds strange at first because it makes the code non-CSS, but we will shortly see why the Hadamards are useful. We write the new stabilizers in the symplectic representation [Got97]

$$\left(\begin{array}{c|c} A_1|A_2|\,\mathbb{1}_{m_X} & \mathbf{0} \\ \mathbf{0} & B_1|\,\mathbb{1}_{m_Z}\,|B_2 \end{array}\right) \xrightarrow{H^{\otimes m_z}} \left(\begin{array}{c|c|c|c|c|c} A_1 & \mathbf{0} & \mathbb{1}_{m_X} & \mathbf{0} & A_2 & \mathbf{0} \\ \mathbf{0} & \mathbb{1}_{m_Z} & \mathbf{0} & B_1 & \mathbf{0} & B_2 \end{array}\right). \tag{2.49}$$

In this Hadamard-transformed code, given a logical Pauli X (that contains Pauli Zs on the middle block and Pauli Xs on the first and third block) we can multiply it with stabilizers from the first m_X rows to remove the Pauli Xs from the third block. Similarly, we can multiply with the stabilizers from the last m_Z rows to remove Pauli Xs from the middle block of each logical Pauli Z. Therefore, we can choose a logical basis for this Hadamard-transformed code such that $\overline{X}'_i = X_i \otimes P''_i$ and $\overline{Z}'_i = Z_i \otimes Q''_i$ where P''_i, Q''_i are Pauli Z strings on $m_Z + m_X$ qubits. Applying $H^{\otimes m_Z}$ to return to the original code we obtain the stated logical basis.

Having chosen a convenient logical basis, we can encode any state $|\psi\rangle$ as follows. We prepare $|\psi\rangle\,|+\rangle^{m_Z}\,|0\rangle^{\otimes m_X}$. We then measure the qLDPC stabilizers. Importantly, we

[33] A similar statement can be made for general stabilizer codes.

measure the LDPC presentation of the stabilizers rather than the canonical generators from the previous paragraph, so that the measurement depth is constant $O(\Delta)$. The resulting state will be the desired logical state encoded in some coset of the code, and we can shift back to the correct code as detailed below.

For simplicity we start with the case when $|\psi\rangle$ is a product state in the Z basis, $|\psi\rangle = |a\rangle$ for $a \in \mathbb{F}_2^k$. The procedure starts with the state $\phi = |a\rangle\,|0\rangle^{\otimes(n-k)}$, which is stabilized by $(-1)^{a_i}\overline{Z}_i$ for $i \in [k]$. Next we perform the (LDPC) syndrome measurements, obtaining syndromes $\sigma_X \in \mathbb{F}_2^{m_Z^{\mathrm{LDPC}}}, \sigma_Z \in \mathbb{F}_2^{m_X^{\mathrm{LDPC}}}$. Using Gaussian elimination we solve for the Pauli corrections $e_X, e_Z \in \mathbb{F}_2^n$ such that $H_X e_Z = \sigma_Z$ and $H_Z e_X = \sigma_X$, where H_X, H_Z are matrices representing the LDPC presentation of the stabilizers. We then multiply e_X with $\overline{X}_i$ for each $i \in [k]$ such that $\overline{Z}_i$ anticommutes with e_X, yielding a Pauli string e'_X. Similarly, we multiply e_Z with $\overline{Z}_i$ for each $i \in [k]$ such that $\overline{X}_i$ anticommutes with e_Z, yielding a Pauli string e'_Z. Note that e'_Z and e'_X can be of mixed Pauli types (including both X and Z), . Applying e'_Z and e'_X produces the desired encoded state $|\overline{a}\rangle$. The same procedure can be seen to work for X basis product states as well. And thus it works for any general input state ψ by linearity.

The quantum depth consists of the physical circuit preparing ψ, the LDPC measurements, and the Pauli correction. This is a $O(\Delta)$ when ψ is a constant-depth state and uses $O(n)$ physical qubits if $m_Z^{\mathrm{LDPC}}, m_X^{\mathrm{LDPC}} = O(n)$. The classical depth is the time needed to compute the Pauli corrections, which can be done by, e.g., Gaussian elimination with gate complexity $O(n^3)$. This can be parallelized to $O(\log^2 n)$ classical depth by the algorithms in [Csa75].

□

Concatenated code unencoding

Concatenated code fault tolerance allows us to simulate the encoding circuit of our computational code in a concatenated code, but we will be left with an encoded state of both the computational code and concatenated code. Hence, we need to unencode this state from the concatenated code. Here, we establish a gadget for unencoding from concatenated codes that has controlled errors[34]. A similar claim was made in [Got13; FGL20] without proof. Recall that for each $r \in \mathbb{N}$, the set of gadgets was compatible, i.e. each encoded concatenated code block input and output

[34] A very recent work [CFG24] also gives a proof of this fact.

a $(Q_{\mathrm{AB},r}, \mathcal{F}_r)$-block for some family of bad fault sets $\mathcal{F}_r$ (the exact definition is not needed). Denote the trivial block corresponding to a single qubit on the set $\{1\}$ and the family of bad sets $\{\{1\}\}$ as a $(Q_{\mathrm{AB},r}, \mathcal{F}_0)$-block.

Lemma 2.4.3 (Concatenated code unencoding)**.** *There exists a constant $c_{\mathrm{unencode,AB}} > 0$ such that for each $r \in \mathbb{N}$ the following holds. There exists an unencoding gadget $\mathsf{Gad}_{\mathrm{unencode,AB,r}}$ that takes a single qubit state ψ encoded in a $(Q_{\mathrm{AB},r}, \mathcal{F}_r)$-block of the r-th level of AB concatenated codes, and outputs the single qubit state ψ if the input is $\mathcal{F}_r$-deviated from $\phi_{Q_{\mathrm{AB},r}}(\psi)$ and the fault set is $\mathcal{G}_{r,\mathrm{unencode}}$-avoiding where $\mathcal{W}(\mathcal{G}_{r,\mathrm{unencode}};\, x) \leq c_{\mathrm{unencode,AB}} \cdot x$ on $[0, 1/(2c_{\mathrm{AB}}))$.*

Proof. Let $\mathsf{Op}_{\mathrm{unencode}}$ be the unencoding operation for the code used in the recursive concatenation procedure. We first define $\widehat{\mathsf{Gad}}_{\mathrm{unencode,AB,r}}$ as the EC-gadget followed by the simulation of $\mathsf{Op}_{\mathrm{unencode}}$ using $(\mathfrak{r}-1)$-AB. This takes a state encoded in a $(Q_{\mathrm{AB},\mathfrak{r}}, \mathcal{F}_{\mathfrak{r}})$-block and outputs the same state encoded in a $(Q_{\mathrm{AB},\mathfrak{r}-1}, \mathcal{F}_{\mathfrak{r}-1})$-block. We will define $\mathsf{Gad}_{\mathrm{unencode,AB},\mathfrak{r}}$ recursively as

$$\mathsf{Gad}_{\mathrm{unencode,AB},\mathfrak{r}} = \mathsf{Gad}_{\mathrm{unencode,AB},\mathfrak{r}-1} \circ \widehat{\mathsf{Gad}}_{\mathrm{unencode,AB},\mathfrak{r}}$$

with $\mathsf{Gad}_{\mathrm{unencode,AB},0}$ trivial.

Let $A_{\mathrm{unencode}} = O(1)$ be the number of locations in $\mathsf{Op}_{\mathrm{unencode}}$. We use Proposition 2.2.28 and Proposition 2.2.18 to upper bound the weight enumerator polynomial for the family of bad sets at each step of the recursion. For each recursion step $\mathfrak{r} > 1$, there is a set of bad fault paths $\mathcal{H}_{\mathfrak{r},\mathrm{unencode}}$ with weight enumerator $\mathcal{W}(\mathcal{H}_{\mathfrak{r},\mathrm{unencode}};\, x) \leq A_{\mathrm{unencode}}(c_{\mathrm{AB}} \cdot x)^{2^{\mathfrak{r}-6}}$ such that if the fault path is $\mathcal{H}_{\mathfrak{r},\mathrm{unencode}}$-avoiding then the gadget takes a $(Q_{\mathrm{AB},\mathfrak{r}}, \mathcal{F}_{\mathfrak{r}})$-block $\rho_{\mathfrak{r}}$ that is $\mathcal{F}_{\mathfrak{r}}$-deviated from $\phi_{Q_{\mathrm{AB},\mathfrak{r}}}(\psi)$ to a $(CQ_{\mathrm{AB},\mathfrak{r}-1}, \mathcal{F}_{\mathfrak{r}-1})$-block $\rho_{\mathfrak{r}-1}$ that is $\mathcal{F}_{\mathfrak{r}-1}$-deviated from $\phi_{Q_{\mathrm{AB},\mathfrak{r}-1}}(\psi)$. For $\mathfrak{r} = 1$, there is no simulation and $\mathcal{W}(\mathcal{H}_{1,\mathrm{unencode}};\, x) \leq A_{\mathrm{unencode}} x$ as any error in one of the A_{encode} locations will cause an error on the output. The family of fault paths $\mathcal{H}_{\mathfrak{r},\mathrm{unencode}}$ corresponds to failure of any of the A_{unencode} level-$(\mathfrak{r}-1)$ simulation gadgets i.e. it is the sum of the bad fault paths for each of the gadgets. Claim 2.4.1 gives an upper bound on the weight enumerator polynomial for each of these gadgets.

The family of bad fault paths for the entire gadget $\mathcal{G}_{r,\mathrm{unencode}}$ is the sum of the bad fault paths in each step of the recursion. That is, for $\mathfrak{r} > 1$,

$$\mathcal{W}(\mathcal{G}_{\mathfrak{r},\mathrm{unencode}};\, x) \leq \mathcal{W}(\mathcal{G}_{\mathfrak{r}-1,\mathrm{unencode}};\, x) + \mathcal{W}(\mathcal{H}_{\mathfrak{r},\mathrm{unencode}};\, x) \tag{2.50}$$

with $\mathcal{W}(\mathcal{G}_{1,\text{unencode}};\, x) = \mathcal{W}(\mathcal{H}_{1,\text{unencode}};\, x)$. Suppose that $x \in (0, 1/(2c_{\text{AB}}))$, then evaluating the recursion gives the upper bound:

$$\mathcal{W}(\mathcal{G}_{r,\text{unencode}};\, x) \leq A_{\text{unencode}}\, x + \sum_{\mathfrak{r}=2}^{r} A_{\text{unencode}} (c_{\text{AB}} \cdot x)^{2^{\mathfrak{r}-1}} \tag{2.51}$$

$$\leq A_{\text{unencode}} x \left(1 + c_{\text{AB}} \sum_{\mathfrak{r}=2}^{\infty} (c_{\text{AB}} \cdot x)^{2^{\mathfrak{r}-1}-1}\right) \tag{2.52}$$

$$\leq A_{\text{unencode}} x \left(1 + c_{\text{AB}} \int_{1}^{\infty} (1/2)^{2^{n}-1} \mathrm{d}n\right) \tag{2.53}$$

$$\leq A_{\text{unencode}}\, (1 + c_{\text{AB}})\, x. \tag{2.54}$$

The integral can be evaluated to be less than 0.343. Thus, the result follows with $c_{\text{unencode,AB}} = A_{\text{unencode}}\, (1 + c_{\text{AB}})$. □

qLTC tester

Combining Lemma 2.4.2 and Lemma 2.4.3 allows us to prepare computational code states to a constant error rate. However, we do not have any guarantee on the prepared state when this procedure fails. Hence, we now construct a verification gadget to verify that the state prepared is not too damaged. We need this verification to be fast, and this is where the qLTC property is used.

The final ingredient will be a tester gadget that uses the qLTC property to test (in low depth) if we have prepared a state close to the code space. We use the notations and variables defined in Section 2.3.

Lemma 2.4.4 (Testing variant of computational code EC gadget)**.** *There exists a constant $\epsilon_{*,\text{tester}} \in (0, 1)$ and gadget $\mathsf{Gad}_{\text{tester}}$ for the computational code of depth $2(\Delta + 3)$, width $n_L + r_L$, and $A_{\text{EC}} = 2(\Delta + 3)(n_L + r_L)$ locations such that there is family of bad fault paths $\mathcal{G}_{\text{tester}} \subseteq P([A_{\text{EC}}])$ for which $\mathcal{W}(\mathcal{G}_{\text{tester}};\, x) \leq e^{-\zeta(n_L)}$ for $x \in [0, \widehat{\epsilon_{*,\text{tester}}}(n_L)]$. For a $\mathcal{G}_{\text{tester}}$-avoiding Pauli fault $\mathbf{f}$, and some pure state in the codespace $|\psi\rangle$, $\mathsf{Gad}_{\text{tester}}[\mathbf{f}]$ satisfies*

- $\mathsf{Gad}_{\text{tester}}[\mathbf{f}]$ *outputs to the classical memory a FAIL bit* y_{fail}.
- *If* $y_{\text{fail}} = 0$ *is output, the output is* $\mathcal{F}_{\text{corr}}$*-Pauli deviated from the code space.*
- *There exists a family* $\mathcal{F}_{\text{test}} \subseteq P([n_L])$ *such that* $\mathcal{W}(\mathcal{F}_{\text{test}};\, x) = \binom{n_L}{t_{\text{test}}} x^{t_{\text{test}}}$ *with* $t_{\text{test}} = \Theta(t_L)$.

- *If the input is $\mathcal{F}_{\text{test}}$-deviated from a codestate σ, then $y_{\text{fail}} = 0$ is always output and the output is $\mathcal{F}_{\text{corr}}$-Pauli deviated from σ.*

Proof. We implement the same circuit as in Lemma 2.3.6, without the decoding and correction steps. This will allow us to borrow the majority of the analysis. We parallel the analysis by first restricting to a state that is diagonal Pauli-deviated from some codestate σ and a diagonal Pauli fault $\mathbf{f}$.

Let $\ell = 2 \cdot 2^{2\Delta}$, and take $\mathcal{G}_{\text{tester}}$ to be all subsets of locations of a size s that will be selected later. Consider the application of $\mathsf{Gad}_{\text{tester}}[\mathbf{f}]$ for a $\mathcal{G}_{\text{tester}}$-avoiding diagonal Pauli fault $\mathbf{f}$. For now, suppose that the input state $\tilde{\rho}$ differs from the codestate σ by a diagonal Pauli superoperator E. By the same argument as in Lemma 2.3.6, the output state then differs from the codestate by a (stabilizer reduced) diagonal Pauli superoperator E_{out} which satisfies (note that we are defining s to be slightly smaller)

$$|\operatorname{supp} E_{\text{out}}| \leq |\operatorname{supp} E| + \ell(s-1). \tag{2.55}$$

Let $r_{\min} = \min(r_x, r_z) = \Omega(n_L)$ be a lower bound for the number of X and Z checks for the computational code. Using the (ρ, Δ)-qLTC property of the computational code, the Hamming weight of the noisy syndrome measured $|\tilde{\sigma}_X| + |\tilde{\sigma}_Z|$ satisfies

$$\frac{\rho r_{\min}}{n_L} |\operatorname{supp} E| - \ell(s-1) \leq |\tilde{\sigma}_X| + |\tilde{\sigma}_Z| \leq 2\Delta |\operatorname{supp} E| + \ell(s-1). \tag{2.56}$$

We reject (setting $y_{\text{fail}} = 1$) when the measured syndrome weight $|\tilde{\sigma}_X| + |\tilde{\sigma}_Z|$ is σ_{reject} or more.

$$\sigma_{\text{reject}} = \frac{\rho r_{\min}}{n_L}(t_{\text{corr}} - s(\ell-1)) - s(\ell-1).$$

When $y_{\text{fail}} = 0$, the measured syndrome satisfies $|\tilde{\sigma}_X| + |\tilde{\sigma}_Z| < \sigma_{\text{reject}}$, so the output error is at most

$$\begin{aligned} |\operatorname{supp} E_{\text{out}}| &\leq |\operatorname{supp} E| + \ell(s-1) \\ &< \left(\sigma_{\text{reject}} + \ell(s-1)\right) \frac{n_L}{\rho r_{\min}} + \ell(s-1) \\ &< t_{\text{corr}}. \end{aligned}$$

We now analyze under what conditions this test will always accept. Let $t_{\text{test}} = \left\lfloor \frac{\sigma_{\text{reject}} - \ell(s-1)}{2\Delta} \right\rfloor$ and suppose that $|\operatorname{supp} E| < t_{\text{test}}$. Then, the input state is always accepted:

$$\begin{aligned} |\tilde{\sigma}_X| + |\tilde{\sigma}_Z| &\leq 2\Delta |\operatorname{supp} E| + \ell(s-1) \\ &< 2\Delta t_{\text{test}} + \ell(s-1) \leq \sigma_{\text{reject}}. \end{aligned}$$

$\mathcal{F}_{\text{test}}$ will be all subsets of $[n_L]$ of size t_{test} which implies $\mathcal{W}(\mathcal{F}_{\text{test}};\, x) = \binom{n_L}{t_{\text{test}}} x^{t_{\text{test}}}$.

Since $\frac{\rho(n_L) r_{\min}}{n_L} = \Theta\left(\frac{\zeta(n_L)}{n_L}\right)$, for n_L larger than some constant, we can always pick a $s = \Theta(\zeta(n_L)) = \Theta(t_{\text{corr}})$ such that $s \geq 2$ and $t_{\text{test}} \geq 1$. By an identical analysis to Lemma 2.3.6 there exists an absolute constant $\epsilon_{*,\text{tester}} \in (0,1)$ such that $\mathcal{W}(\mathcal{G}_{\text{tester}};\, x) \leq e^{-\zeta(n_L)}$ for $x \in [0, \widehat{\epsilon_{*,\text{tester}}}(n_L)]$.

Finally, since $t_{\text{test}} \leq t_{\text{corr}}$, a set that is F_{test}-avoiding is also F_{corr}-avoiding. We can now release the assumption on the input state and the fault as in Lemma 2.3.6: Apply Lemma 2.2.31 to ensure that the input and output must be diagonal-Pauli superoperators except where there is a fault. If the input state $\tilde{\rho}$ is $\mathcal{F}_{\text{test}}$-Pauli deviated from a codestate σ, then it follows from the earlier argument that the output state is $\mathcal{F}_{\text{corr}}$-Pauli deviated from σ. Otherwise, if $y_{\text{fail}} = 0$, the output state is $\mathcal{F}_{\text{corr}}$-Pauli deviated from some codestate $\phi_{Q_L}(\rho')$ where ρ' is arbitrary. □

Noisy state preparation gadget

We are now ready to state and prove the main goal of this section.

Lemma 2.4.5 (Noisy state preparation gadget)**.** *For $\epsilon_{\text{noise}} \in (0,1)$ and a pure k_L-qubit state $|\psi\rangle$ that can be prepared by a Clifford+CCZ circuit $\mathsf{Op}_{\text{prep}}$ using depth D_{prep} and space W_{prep} such that $W_{\text{prep}} \geq \Theta(k_L) = \Theta(n_L)$, there exists a constant $\epsilon_{*,\text{inject}} \in (0,1)$ (independent of n_L) and a Pauli fault-tolerant gadget $\mathsf{Gad}_{\text{noisyprep}}$[35] preparing $|\overline{\psi}\rangle$ in a $(Q_L, \mathcal{F}_L)$-block with L locations such that the following properties hold:*

- *If the fault is Pauli and its fault path is $\mathcal{G}_{\text{bad}}$-avoiding, then the output state is $\mathcal{F}_L$-Pauli deviated from Q_L or FAIL is output.*
- *If the fault is Pauli and its fault path is also $\mathcal{G}_{\text{noisy}}$-avoiding, then the output state is $\mathcal{F}_L$-Pauli deviated from $\phi_{Q_L}(\psi)$.*

Where, for $x \in [0, \widehat{\epsilon_{*,\text{inject}}}(n_L)]$, $\mathcal{G}_{\text{bad}},\ \mathcal{G}_{\text{noisy}} \subseteq P([L])$ satisify

$$\mathcal{W}(\mathcal{G}_{\text{bad}};\, x) \leq 2e^{-\zeta(n_L)} \tag{2.57}$$

$$\mathcal{W}(\mathcal{G}_{\text{noisy}};\, x) \leq \epsilon_{\text{noise}} \tag{2.58}$$

[35] We omit the parameters, since this is a slightly extended notion of fault-tolerant gadget.

$\mathsf{Gad}_{\text{noisyprep}}$ *has depth* D'_{prep} *and width* W'_{prep}.

$$D'_{\text{prep}} = O\left(D_{\text{prep}} \operatorname{polylog}\left(\frac{W_{\text{prep}} D_{\text{prep}}}{\epsilon_{\text{noise}}}\right)\right), \tag{2.59}$$

$$W'_{\text{prep}} = O\left(W_{\text{prep}} \operatorname{polylog}\left(\frac{W_{\text{prep}} D_{\text{prep}}}{\epsilon_{\text{noise}}}\right)\right). \tag{2.60}$$

The family of fault paths $\mathcal{G}_{\text{bad}}$ should be thought of as those that cause the output to be so bad that *further computation is not possible* e.g. the state is too far from the codespace for the decoder to operate. We cannot let such fault paths to occur, so we will perform a testing step at the end of our gadget to increase their weight to be $\Omega(n_L)$. In order to avoid incurring a depth overhead, we use the qLTC property to perform the testing in constant quantum depth. The family of fault paths $\mathcal{G}_{\text{noisy}}$ should be thought of as those fault paths that cause the wrong logical state to be prepared. This is not immediately fatal as we will later be performing distillation of these states.

Proof of Lemma 2.4.5. Let $\mathcal{E}_{Q_L}$ be the encoding circuit of Q_L from Lemma 2.4.2 which has space at most $O(n_L)$ and constant (quantum depth). Let $\mathsf{Gad}_{\text{prepsim}}$ be the simulation of $\mathcal{E}_{Q_L} \circ \mathsf{Op}_{\text{prep}}$ by r-AB with $r = \lceil \log_2 \log_2 \frac{3\left(|\mathcal{E}_{Q_L}|+|\mathsf{Op}_{\text{prep}}|\right)}{\epsilon_{\text{noise}}} \rceil$.

Our gadget $\mathsf{Gad}_{\text{noisyprep}}$ will execute $\mathsf{Gad}_{\text{prepsim}}$, run the AB concatenated code unencoding gadgets (Lemma 2.4.3), the tester gadget (Lemma 2.4.4), and then the error correction gadget (Lemma 2.3.6). I.e.

$$\mathsf{Gad}_{\text{noisyprep}} = \mathsf{Gad}_{\text{EC}} \circ \mathsf{Gad}_{\text{tester}} \circ \left(\mathsf{Gad}_{\text{unencode,AB,r}}\right)^{\otimes n_L} \circ \mathsf{Gad}_{\text{prepsim}}$$

The bit produced by the tester gadget is the FAIL bit that this gadget outputs. The depth and width of $\mathsf{Gad}_{\text{noisyprep}}$ bounds follow from the depth/width bounds on the components. To simplify the bounds, we use the fact that $W = \Omega(n_L)$. Recall that an r-AB concatenated code gadget has width and depth at most exponential in r:

	Gad_{EC}	$\mathsf{Gad}_{\text{tester}}$	$\left(\mathsf{Gad}_{\text{unencode,AB,r}}\right)^{\otimes n_L}$	$\mathsf{Gad}_{\text{prepsim}}$
Depth	$O(1)$	$O(1)$	$O\left(\operatorname{polylog}\left(\frac{W_{\text{prep}} D_{\text{prep}}}{\epsilon_{\text{noise}}}\right)\right)$	$O\left(D_{\text{prep}} \operatorname{polylog}\left(\frac{W_{\text{prep}} D_{\text{prep}}}{\epsilon_{\text{noise}}}\right)\right)$
Width	$O(n_L)$	$O(n_L)$	$O\left(n_L \operatorname{polylog}\left(\frac{W_{\text{prep}} D_{\text{prep}}}{\epsilon_{\text{noise}}}\right)\right)$	$O\left(W_{\text{prep}} \operatorname{polylog}\left(\frac{W_{\text{prep}} D_{\text{prep}}}{\epsilon_{\text{noise}}}\right)\right)$

We now perform the fault analysis beginning with $\left(\left(\mathsf{Gad}_{\text{unencode,AB,r}}\right)^{\otimes n_L} \circ \mathsf{Gad}_{\text{prepsim}}\right)$. Let $\mathbf{f}$ be the Pauli fault path.

Using Proposition 2.2.28, we can take the sum ($\boxplus$) over the bad fault paths for each of the $(|\mathcal{E}_{Q_L}| + |\mathsf{Op}_{\text{prep}}|)$ concatenated code gadgets comprising $\mathsf{Gad}_{\text{prepsim}}$ and construct a family of bad fault paths $\mathcal{G}_1$ for $\mathsf{Gad}_{\text{prepsim}}$ with $\mathcal{W}(\mathcal{G}_1;\, x) \leq (|\mathcal{E}_{Q_L}| + |\mathsf{Op}_{\text{prep}}|)(c_{\text{AB}}x)^{2^r} \leq \epsilon_{\text{noise}}/3$ for $x \in [0, \frac{1}{3c_{\text{AB}}}]$ such that: If $\operatorname{supp}\mathbf{f}$ is $\mathcal{G}_1$-avoiding, then $\mathsf{Gad}_{\text{prepsim}}[\mathbf{f}]$ outputs a state $\tilde{\psi}_1$ that is $\mathcal{F}_r$-deviated on each concatenated code block (recall $\mathcal{F}_r$ are the bad sets for r-level AB concatenated codes) from $\phi_{Q_{\text{AB},r}} \circ \phi_{Q_L}(|\psi\rangle)$.

The bad fault paths $\mathcal{G}_1$ cause an error in preparing the initial state (encoded in the concatenated code), but this is not immediately fatal. Next, we must unencode the state and verify that it is close to the codespace.

We run the unencoding gadget on each $(Q_{\text{AB},r}, \mathcal{F}_r)$-block of $\tilde{\psi}_1$ in parallel. Let $L_{r,\text{unencode}}$ be the set of locations of a single unencoding gadget $\mathsf{Gad}_{\text{unencode,AB,r}}$. Lemma 2.4.3 guarantees that the output of a single unencoding gadget is perfect if the fault set is $\mathcal{G}_{r,\text{unencode}}$-avoiding on $L_{r,\text{unencode}}$. In order for the output to be $\mathcal{F}_{\text{test}}$-deviated from $\phi_{Q_L}(\psi)$, it must be the case that the fault path of $\mathbf{f}$ must be $\mathcal{G}_{r,\text{unencode}}$-avoiding on all but a $\mathcal{F}_{\text{test}}$-avoiding set of unencoding gadgets. That is, $\operatorname{supp}\mathbf{f}$ must be $\mathcal{G}_2 := \mathcal{F}_{\text{test}} \bullet \mathcal{G}_{r,\text{unencode}}$-avoiding on $\sqcup_{i\in[n_L]} L_{r,\text{unencode}}$.

Proposition 2.2.28 computes the weight enumerator of $\mathcal{G}_2$ in terms of $\mathcal{W}(\mathcal{G}_{r,\text{unencode}};\, x)$ and $\mathcal{W}(\mathcal{F}_{\text{test}};\, x)$. Thus for $x \in \left[0, \frac{1}{2c_{\text{unencode,AB}}}\right]$, using $t_{\text{test}} = \Theta(\zeta(n_L))$, we can then bound

$$\begin{aligned}
\mathcal{W}(\mathcal{G}_2;\, x) &= \mathcal{W}(\mathcal{F}_{\text{in}};\, \mathcal{W}(\mathcal{G}_{\text{r,unencode}};\, x)) \\
&\leq \binom{n_L}{t_{\text{test}}} \left(c_{\text{unencode,AB}} \cdot x\right)^{t_{\text{test}}} \\
&\leq \left((\text{const.})\frac{n_L}{t_{\text{test}}}x\right)^{(\text{const.})\text{test}} \\
&\leq \left((\text{const.})\frac{n_L}{\zeta(n_L)}x\right)^{(\text{const.})\zeta(n_L)}.
\end{aligned}$$

It follows that we can pick a constant $\epsilon_{*,2} \in \left(0, \frac{1}{2c_{\text{unencode,AB}}}\right]$ such that for $x \in [0, \widehat{\epsilon_{*,2}}(n_L)]$,

$$\begin{aligned}
\mathcal{W}(\mathcal{G}_2;\, x) &\leq \left((\text{const.})\frac{n_L}{\zeta(n_L)}\widehat{\epsilon_{*,2}}(n_L)\right)^{(\text{const.})\zeta(n_L)} \\
&\leq \left((\text{const.})\epsilon_{*,2}\right)^{(\text{const.})\zeta(n_L)} \\
&\leq \epsilon_{\text{noise}}/3.
\end{aligned}$$

Let $\mathcal{G}_{\text{tester}}$ and $\mathcal{G}_{\text{EC}}$ be the bad fault sets for the tester gadget $\mathsf{Gad}_{\text{tester}}$ and EC gadget Gad_{EC}, respectively, and define $\mathcal{G}_{\text{bad}} := \mathcal{G}_{\text{tester}} \boxplus \mathcal{G}_{\text{EC}}$. Assume that $\operatorname{supp} \mathbf{f}$ is $\mathcal{G}_{\text{bad}}$-avoiding.

Define $\mathcal{G}_{\text{noisy}} := \mathcal{G}_1 \boxplus \mathcal{G}_2$. At this point, the fault analysis splits into two cases, depending on whether $\left(\left(\mathsf{Gad}_{\text{unencode,AB,r}}\right)^{\otimes n_L} \circ \mathsf{Gad}_{\text{prepsim}}\right)$ is faulty.

If $\operatorname{supp} \mathbf{f}$ is $\mathcal{G}_{\text{bad}} \boxplus \mathcal{G}_{\text{noisy}}$-avoiding we are left with a post-preparation state $\tilde{\psi}_2$ that is $\mathcal{F}_{\text{test}}$-deviated from $\left|\overline{\psi}\right\rangle$. The test always passes, and the post-tester state is $\mathcal{F}_{\text{corr}}$-Pauli deviated from $\left|\overline{\psi}\right\rangle$. Thus, the EC step succeeds and the output is $\mathcal{F}_L$-Pauli deviated from $\left|\overline{\psi}\right\rangle$.

If $\operatorname{supp} \mathbf{f}$ is only $\mathcal{G}_{\text{noisy}}$-avoiding there is no guarantee on the post-preparation logical state. However, we do have the guarantee that if the tester does not output FAIL, then the post-tester state is $\mathcal{F}_{\text{corr}}$-Pauli deviated from some (possibly mixed) codestate, so the output of Gad_{EC} is $\mathcal{F}_L$-Pauli deviated from a codestate.

The desired fault-tolerance properties of the gadget follows for $x \in [0, \epsilon_{*,\text{inject}}]$ with $\epsilon_{*,\text{inject}} = \min\left(\frac{1}{3c_{\text{AB}}}, \epsilon_{*,\text{tester}}, \epsilon_{*,\text{EC}}, \epsilon_{*,2}\right)$. □

State distillation

We now proceed to define a state distillation procedure. Informally, a state distillation procedure takes M imperfect copies of a state ψ and outputs K perfect copies of ψ subject to the condition that the subset of imperfect states is sparse in a certain sense.

Definition 2.4.6 (State distillation procedure)**.** *Let $M \in \mathbb{N}$ and $\mathcal{A} \subseteq P([M])$ be a family of sets. For an n-qubit state ψ, an $(M, K, \mathcal{A})$-state distillation procedure for ψ takes an $n \cdot M$ qubit state state ρ such that there is an $\mathcal{A}$-avoiding set $A \subseteq [M]$ for which $\rho|_{[n]\times([M]\setminus A)} = \psi^{\otimes(M-|A|)}$ and outputs $\psi^{\otimes K}$.*

As in section Section 2.3, we use an overline to denote a register state encoded in a computational code block.

State distillation procedures

We will construct a state distillation scheme for stabilizer states and a particular magic state. To avoid a lengthy excursion, we will defer the proof of these lemmas to later sections.

Lemma 2.4.7 (State distillation procedure for stabilizer states)**.** *For a stabilizer state ψ on b qubits, there exists constants $\epsilon_{*,distill} \in (0,1)$, $\beta_{\text{distill}} > 0$ and a family indexed by $l \in \mathbb{N}$ of $(M_l, K_l, \mathcal{A}_l)$-state distillation procedure where $\mathcal{W}(\mathcal{A}_l;\, x) \leq (\beta_{\text{distill}} x)^{2^\ell}$ on $x \in [0, \epsilon_{*,distill}]$, $\frac{K_l}{M_l} = \Theta(1)$, and $M_l = e^{\Theta(l \log l)}$.*

Furthermore, the procedure satisfies:

- *The quantum depth is $O(l^3)$, and the width is $O(M_l)$.*
- *The classical computation has total depth $O(l^3)$.*
- *No additional input states are required beyond the M_l noisy inputs.*
- *Only the operations* CNOT*, Z/X basis measurement, and classically-controlled Pauli gates are used.*
- *With the exception of Pauli gates, the quantum gates applied do not depend on the state ψ to be distilled, only b and l.*
- *No two-qubit gates are applied between qubits with different $[b]$ coordinates.*[36]

The proof of the lemma utilizes the same constant-rate concatenated Quantum Hamming code family as used in [YK24] as a state distillation method with modifications to reduce the time overhead. Since the construction is mostly standard ideas from state distillation, we defer the proof to Section 2.4.

The second state distillation procedure will be for magic states.

Theorem 2.4.8 (State distillation procedure for magic states)**.** *There exists constants $\epsilon_{*,Mdistill} \in (0,1)$, $\beta_{\text{Mdistill}} > 0$ and a family indexed by $l \in \mathbb{N}$ of $(M_l, K_l, \mathcal{A}_l)$-state distillation procedure for $|\mathsf{CCZ}\rangle$ where $\mathcal{W}(\mathcal{A}_l;\, x) \leq (\beta_{\text{Mdistill}} x)^{(l+1)!}$ on $x \in [0, \epsilon_{*,Mdistill}]$, $\frac{K_l}{M_l} = \Omega\left(c^l\right)$, and $M_l = e^{\Theta(l \log l)}$ for some absolute constant $c > 0$.*

Furthermore, the procedure satisfies:

- *The quantum depth is $O(l^4 (\log l)^3)$, and the width is $O(M_l)$.*
- *The classical computation has total depth $O(l^4)$.*
- *$O(M_l)$ perfect $|+\rangle$ and $|0\rangle$ input states are required*

[36] In other words, the gates are applied "transversally" with regard to the qubits $[b]$.

- *$O(M_l)$ classically-controlled* CZ *gates are required*
- *Otherwise, only* CNOT*, Z/X basis measurement, and classically-controlled Pauli gates are used.*

The proof of this state distillation procedure will require the construction of new quantum codes over qudits of prime-power dimension. The codes are constructed in Section 2.5, and the proof of Theorem 2.4.8 is deferred until Section 2.5.

FT state distillation gadgets (proofs of Lemma 2.3.4 and Lemma 2.3.5)

We are now ready to prove the main lemmas for preparation of resource states. At this point, we are ready to switch to asymptotic notation. The first gadget exploits the fact that the stabilizer state distillation procedure in Lemma 2.4.7 uses the same gates regardless of the state to be distilled. Thus, we can distill resource states that differ on different coordinates of the computation code.

Lemma 2.3.4 (Stabilizer resource state preparation)**.** *For a constant $b \in \mathbb{N}$, $b < 10$*[37]*, representing the number of blocks, let $\{|\psi_i\rangle\}_{i=1}^{k_L}$ be a set of b-qubit stabilizer states such that $|\psi\rangle := |\psi_1\rangle \otimes \cdots \otimes |\psi_{k_L}\rangle$ is preparable by a constant depth unitary Clifford circuit $\mathsf{Op}_{\text{stab}}$ acting on $|0\rangle^{\otimes bk_L}$ and bk_L ancillas qubits. Then, there exists an $(\mathsf{Op}_{\text{stab}}, \mathcal{G})$-Pauli FT gadget $\mathsf{Gad}_{\text{stab}}$ preparing K copies of $|\overline{\psi}\rangle$, each encoded in b codeblocks of the computational code such that the i-th logical qubits of the b codeblocks is in the state $|\psi_i\rangle$*[38] *and $\mathcal{W}(\mathcal{G}; x) \leq e^{-\zeta(n_L)+O(\text{polylog}\, n_L)}$ for $x \in [0, \widehat{\epsilon_{*,\text{stab}}}(n_L))$.*

In terms of a variable $M = e^{\Theta(\log n_L \cdot \text{loglog}\, n_L)}$, we have the bound:

$$K = \Theta(M) \tag{2.16}$$

and $\mathsf{Gad}_{\text{stab}}$ has quantum depth $O(\text{polylog}\, n_L)$, width $O(Mn_L)$, and classical depth $O(\text{polylog}\, n_L)$.

Proof. The gadget will first consist of parallel repetitions of preparation of noisy resource states. We will then collect those resource states which pass the qLTC check for being close to the codespace and then proceed with the stabilizer state distillation procedure.

[37] Any absolute constant suffices.

[38] I.e. preparing $\psi_1 \otimes \cdots \otimes \psi_{k_L}$ "coordinate-wise."

Set $l = \log_2 \log_2 \frac{1}{e^{-\zeta(n_L)}} = O(\log n_L)$ in Lemma 2.4.7 and $\epsilon_{\text{noise}} = \min\left(\frac{1}{2\beta_{\text{distill}}}, \frac{1}{8}\right)$ in Lemma 2.4.5. Let $\epsilon_{*,\text{stab}} = \min\left(\epsilon_{*,\text{inject}}, \epsilon_{*,\text{noise}}\right)$. This choice of parameters implies that the number of states input to the stabilizer state distillation procedure (Lemma 2.4.7) $M_l = e^{\Theta(\log n_L \operatorname{loglog} n_L)}$. Let $\alpha = \min(n_L^2, M_l) = \Omega(n_L^2)$ and $\beta = \left\lceil \frac{M_l}{\alpha} \right\rceil$. By assumption, $\mathsf{Op}_{\text{stab}}$ is constant depth and has width $O(n_L)$.

The gadget is constructed as follows:

1. Consider the noisy simulation of $\mathsf{Op}_{\text{stab}}$ using the gadget $\mathsf{Gad}_{\text{noisyprep}}$ from Lemma 2.4.5. $\mathsf{Gad}_{\text{noisyprep}}$ uses width $O(n_L c(n_L))$ for some $c(n_L) = O(\text{polylog}\, n_L)$ and depth $O(\text{polylog}\, n_L)$. Execute $\mathsf{Gad}_{\text{noisyprep}}$ $\left\lceil \frac{2\alpha\beta}{c(n_L)} \right\rceil$ times in parallel $c(n_L)$ times for a total of $2\alpha\beta$ states.[39] Overall, this requires depth $O(\text{polylog}\, n_L)$ and width $O(n_L \text{ polylog}\, n_L)$, so this step has depth $O(\text{polylog}\, n_L)$ and width $O(\alpha\beta n_L) = O(M n_L)$. Each of the $2\alpha\beta$ executions of $\mathsf{Gad}_{\text{noisyprep}}$ produce a state on b code blocks. Since the code blocks are naturally indexed by $[\beta] \times [2\alpha] \times [b]$, we will refer to each group of b code blocks as a "row" and each group of $[\beta]$ rows as a "column."

2. Let $J \subseteq [\beta] \times [2\alpha]$ be the subset of state preparation gadgets for which the FAIL bit is not set (i.e. the qLTC checks are passed). In parallel, for each column $i \in [\beta]$, sort the 2α rows of computation code blocks such that the rows originally in J are the first rows in column i. This can be done in depth $O(\log \alpha)$ and width $O(\alpha\beta n_L)$ using a classically controlled transversal SWAP of the computational code blocks.

3. Define the set

   $$I = \{(i, j) \in [\beta] \times [2\alpha] \mid (i < \beta \text{ and } j \leq \alpha) \text{ or } (i = \beta \text{ and } j \leq M_l - (\beta - 1)\alpha\}$$

   of size M_l given by taking α rows from all but the last column and the remaining rows from the last column. Execute the distillation in Lemma 2.4.7 on the rows in I. Since the gates in the distillation procedure do not depend on the state and act only coordinate-wise, this can be done using the transversal gate gadgets of the computation code (Lemma 2.3.7). This has depth $O(\text{polylog}\, n_L)$, width $O(M_l n_L)$, and produces $\Omega(M_l)$ resource states.

In the following, we restrict $x \in [0, \widehat{\epsilon_{*,\text{stab}}}(n_L)]$ and upper bounds do not depend on x unless indicated.

[39] The organization of states in groups of α is required to avoid a $\log M_l$ time overhead when collecting the outputs. The preparation over $c(n_L)$ steps is required to avoid incurring space overhead.

For $(i,j) \in I$, let $\mathcal{H}_{\text{bad}}^{(i,j)}$ and $\mathcal{H}_{\text{noisy}}^{(i,j)}$ denote the families of bad fault paths $\mathcal{G}_{\text{bad}}$ and $\mathcal{G}_{\text{noisy}}$ from Lemma 2.4.5 on the (i,j)-th state preparation gadget in step-1. Let $[L]$ index the set of computational code gadgets applied in steps 2 and 3. By construction, $L = O(M_l \operatorname{polylog}(n_L))$. For each computational code gadget $l \in [L]$, let $\mathcal{H}_l$ be the corresponding set of bad fault paths.

The output is correct as long as none of the following events occur:

1. A computational code gadget or the state verification step in Lemma 2.4.5 fails:

$$\mathcal{G}_1 = \left(\boxplus_{l\in[L]}\mathcal{H}_l\right) \boxplus \left(\boxplus_{(i,j)\in I}\mathcal{H}_{\text{bad}}^{(i,j)}\right) \tag{2.61}$$

$$\mathcal{W}(\mathcal{G}_1;\, x) \le \sum_{l\in[L]} \mathcal{W}(\mathcal{H}_l;\, x) + \sum_{(i,j)\in I} \mathcal{W}(\mathcal{G}_{\text{bad}};\, x) \le (L + |I|)e^{-\zeta(n_L)} \tag{2.62}$$

$$= O\left(M_l \operatorname{polylog}(n_L) e^{-\zeta(n_L)}\right). \tag{2.63}$$

2. There exists a column with less than α rows for which FAIL was not set:[40]

$$\mathcal{G}_2 = \boxplus_{i\in[\beta]} \boxplus_{\substack{J\subseteq[2\alpha]\\|J|=\alpha+1}} \left(\circledast_{j\in J}\mathcal{H}_{\text{noisy}}^{(i,j)}\right) \tag{2.64}$$

$$\mathcal{W}(\mathcal{G}_2;\, x) \le \beta \binom{2\alpha}{\alpha+1} \left(\mathcal{W}(\mathcal{G}_{\text{noisy}};\, x)\right)^{\alpha+1} \tag{2.65}$$

$$\le \beta \cdot 2^{2\alpha} \left(\epsilon_{\text{noise}}\right)^{\alpha+1} \tag{2.66}$$

$$\le \beta \cdot \left(\frac{1}{2}\right)^{\alpha} \tag{2.67}$$

$$\le \beta e^{-\Omega(\alpha)} \le M_l e^{-\Omega(n_l^2)}. \tag{2.68}$$

3. The distillation output contains an error: $\mathcal{G}_3$,

$$\mathcal{G}_3 = \boxplus_{A\in\mathcal{A}_l} \circledast_{(i,j)\in A} \mathcal{H}_{\text{noisy}}^{(i,j)} \simeq \mathcal{A}_l \bullet \mathcal{G}_{\text{noisy}} \tag{2.69}$$

$$\mathcal{W}(\mathcal{G}_3;\, x) \le \mathcal{W}(\mathcal{A}_l;\, \mathcal{W}(\mathcal{G}_{\text{noisy}};\, x)) \tag{2.70}$$

$$\le \left(\frac{\beta_{\text{distill}}}{2\beta_{\text{distill}}}\right)^{2^l} \tag{2.71}$$

$$\le e^{-\zeta(n_L)}. \tag{2.72}$$

[40] We slightly abuse notation and consider $A_l \subseteq I$. This detail is not important.

Thus, $\mathcal{G} = \boxplus_{i=1}^{3} \mathcal{G}_i$ with

$$\mathcal{W}(\mathcal{G};\ x) \leq O(M_l \operatorname{polylog}(n_L) e^{-\zeta(n_L)}). \tag{2.73}$$

□

The construction of an analog of Lemma 2.3.4 for magic states is essentially similar with minor differences due to the need for a conditional Clifford operation and the preparation of the resource state only on the first coordinate instead of for all coordinates.

Lemma 2.3.5 (Magic resource state preparation)**.** *There exists an* $(\mathsf{Op}_{\text{magic}}, \mathcal{G})$*-Pauli FT gadget* $\mathsf{Gad}_{\text{magic}}$ *preparing* K *copies of* $\left|\overline{\mathsf{CCZ}(1,2,3)}\right\rangle$ *such that* $\mathcal{W}(\mathcal{G};\ x) \leq e^{-\zeta(n_L)+O(\operatorname{polylog} n_L)}$ *for* $x \in [0, \widehat{\epsilon_{*,\text{magic}}}(n_L))$.

In terms of a variable M *satisfying* $M = e^{O(\log n_L \cdot \operatorname{loglog} n_L)}$, $M = \Omega(n_L)$*, we have the bound:*

$$K = \Omega(M n_L^{-\tilde{\gamma}(n_L)}) \tag{2.17}$$

where $\tilde{\gamma}(n_L) = O_{n_L \to \infty}\left(\frac{1}{\operatorname{loglog} n_L}\right) = o_{n_L \to \infty}(1)$. $\mathsf{Gad}_{\text{magic}}$ *has quantum depth* $O(\operatorname{polylog} n_L)$*, width* $O(M n_L)$*, and classical depth* $O(\operatorname{polylog} n_L)$.

Proof. Let $w(x)$ be the principle branch of the Lambert W function. I.e. a solution to $w(x)e^{w(x)} = x$ for $x \geq 0$ satisfying $w(x) \geq 0$. We start by defining the function $g(x)$ which satisfies the following:

$$g(x) = \frac{\log(x)}{w(\log(x^{1/e}))}, \qquad \left(\frac{g(x)}{e}\right)^{g(x)} = x. \tag{2.74}$$

This allows us to obtain an upper bound for the inverse of the factorial:

$$\lceil g(x) \rceil ! \geq \left(\frac{g(x)}{e}\right)^{g(x)} = x. \tag{2.75}$$

We will follow the argument of Lemma 2.3.4 almost exactly. We apply the magic state distillation procedure of Theorem 2.4.8 using the transversal gate gadget (Lemma 2.3.7).

We use Theorem 2.4.8 with l such that

$$l + 1 = \lceil g\left(\zeta(n_L)\right) \rceil . \tag{2.76}$$

and $\epsilon_{\text{noise}} = \min\left(\frac{1}{e\beta_{\text{Mdistill}}}, \frac{1}{8}\right)$ in Lemma 2.4.5. As in Lemma 2.3.4, we set

$$\epsilon_{*,\text{magic}} = \min\left(\epsilon_{*,\text{inject}}, \epsilon_{*,\text{noise}}, \epsilon_{*,\text{stab}}\right) .$$

Note that $\epsilon_{*,\text{inject}}$ and $\epsilon_{*,\text{noise}}$ are not necessarily the same as in Lemma 2.4.5.

We first use $\mathsf{Gad}_{\text{stab}}$ (Lemma 2.3.4) to prepare $\Theta(M_l)$ copies of $\left|\overline{\mathsf{CZ}(1_A, 1_B)}\right\rangle$, $\left|\overline{+}\right\rangle$, and $\left|\overline{0}\right\rangle$. The $\left|\overline{\mathsf{CZ}(1_A, 1_B)}\right\rangle$ states allow us to use transversal gates (Lemma 2.3.7) to execute a teleported CZ gate on the first coordinate in the execution of the magic state distillation procedure Theorem 2.4.8. These are the necessary inputs to execute the magic state distillation procedure of Theorem 2.4.8 using transversal gates (Lemma 2.3.7) to produce K $\left|\overline{\mathsf{CCZ}(1_A, 1_B, 1_C)}\right\rangle$ states. At the end, we again use $\mathsf{Gad}_{\text{stab}}$ (Lemma 2.3.4) to prepare K copies of $\left|\overline{\mathsf{SWAP}(2_A, 1_B)}\right\rangle$ and $\left|\overline{\mathsf{SWAP}(3_A, 1_B)}\right\rangle$ which use to execute teleported $\mathsf{SWAP}(2_A, 1_B)\mathsf{SWAP}(3_A, 1_C)$ on $\left|\overline{\mathsf{CCZ}(1_A, 1_B, 1_C)}\right\rangle$, converting it to $\left|\overline{\mathsf{CCZ}(1, 2, 3)}\right\rangle$.

The family of bad fault paths associated with these invocations of the Lemma 2.3.4 gadget is added to $\mathcal{G}_1$, but it does not change the asymptotic scaling with n_L. The bound for $\mathcal{W}(\mathcal{G}_2;\ x)$ is identical to that of Lemma 2.3.4. $\mathcal{G}_3$ is significantly modified: For $x \in [0, \widehat{\epsilon_{*,\text{magic}}}(n_L))$, we have the new bound

$$\mathcal{W}(\mathcal{G}_3;\ x) \leq \left(\frac{\beta_{\text{Mdistill}}}{e \cdot \beta_{\text{Mdistill}}}\right)^{(l+1)!} \leq e^{-(l+1)!} \leq e^{-\zeta(n_L)}. \tag{2.77}$$

The depth is $O(\text{polylog}\, n_L)$, and the width is $O(n_L M_l)$. For some constant $c \in (0, 1)$, we have produced

$$K = \Omega(M_l c^l) = \Omega\left(M_l c^{g(\zeta(n_L))}\right) \tag{2.78}$$

$$= \Omega\left(M_l \exp\left[-O\left(\frac{\log n_L}{w\left(\log\left(n_L^{1/e}\right)\right)}\right)\right]\right) \tag{2.79}$$

$$= \Omega\left(M_l n_L^{-\tilde{\gamma}(n_L)}\right) \tag{2.80}$$

$\left|\overline{\mathsf{CCZ}(1_A, 1_B, 1_C)}\right\rangle$ states where $\tilde{\gamma}(n_L) = O\left(\frac{1}{w\left(\log\left(n_L^{1/e}\right)\right)}\right) = O_{n_L\to\infty}\left(\frac{1}{\log\log n_L}\right) = o_{n_L\to\infty}(1)$. □

Constant-overhead stabilizer state distillation procedure

It remains to prove Lemma 2.4.7. We first begin by defining a procedure for a single round of distillation of a stabilizer state.

We heavily use the stabilizer formalism [Got97; AG04]. For an n-qubit stabilizer state $|\psi\rangle$, we use the notation $\mathrm{Stab}(\psi)$ to refer to the subgroup of Pauli operators s such that $s|\psi\rangle = |\psi\rangle$.

For an $n \times m$ matrix A, we use the notation $A[\cdot, j]$ to refer to the j-th column, and $A[i, \cdot]$ to refer to the i-th row. For $i \in [n]$, let $e_i \in \mathbb{F}_2^n$ be the indicator vector for the i-th coordinate. In what follows, for an b qubit Pauli operator $P = P_1^{(1)} P_2^{(2)} \dots P_b^{(b)}$ (the operator $P^{(i)}$ acting on the i-th qubit) and a bit string $x \in \mathbb{F}_2^n$, we will use $P^x \in \mathcal{P}^{n \times b}$ to denote the Pauli operator supported on a set of qubits indexed by $[n] \times [b]$ given by

$$P^x \equiv \prod_{i \in [n], j \in [b]} (P^{(j)})_{i,j}^{x[i]} \in \mathcal{P}^{n \times b} \,. \tag{2.81}$$

To reduce the amount of notation, we introduce the following abuse of notation: For a matrix $H \in \mathbb{F}_2^{r \times n}$, and an b-qubit Pauli operator P, we use P^H to denote the indexed set

$$P^H \equiv \left\{ P^{H[i,\cdot]} \right\}_{i \in [r]} \subseteq \mathcal{P}^{n \times b} \,. \tag{2.82}$$

I.e. every row $i \in [r]$ of H defines an operator h. For every column $j \in [n]$ of H, the corresponding row has $h|_{\{j\} \times [b]} = P$ iff $H[i, j] = 1$.

Let Q_{distill} be a quantum CSS code with parameters $[[n, k, d]]$ with the following properties:

- There is a (check) matrix $H \in \mathbb{F}_2^{r \times n}$ such that the set $X^H \cup Z^H$ form a minimal generating set for the stabilizer of Q_{distill}.
- There exists a (generator) matrix $G \in \mathbb{F}_2^{k \times n}$ such that $GG^T = I$ and $HG^T = 0$.
- There exists an efficient algorithm $\mathcal{D}_H \colon \mathbb{F}_2^r \to \mathbb{F}_2^n$ to decode all errors of weight at most t for classical code $\ker H$. I.e. for all $x \in \mathbb{F}_2^n$ such that $x \leq t$, $\mathcal{D}_H(Hx) = x$.

Such a quantum CSS code is sometimes said to be "self-dual"[41] and possesses convenient properties such as a fully transversal implementation of the Clifford group. Quantum Hamming codes as well as color codes satisfy this property.

Let S_ψ be a minimal generating for the stabilizer $\mathrm{Stab}(\psi)$ of the b-qubit state ψ. I.e. for all $s \in S_\psi$, ψ is an eigenstate with eigenvalue $+1 \in \{\pm 1\}$. In the following, we

[41] This notion is unrelated to the notion of the dual of a classical code.

will associate measurement outcomes of Pauli operators with elements of $\mathbb{F}_2$ in the canonical way: $(+1 \mapsto 0 \in \mathbb{F}_2, -1 \mapsto 1 \in \mathbb{F}_2)$.

We first compute (by Gaussian elimination) a set of operators that each anticommute with exactly one element of S_ψ. Concretely, we have a map $\phi_\psi : S_\psi \to \mathcal{P}^b$ such that for $s \in S_\psi$, $\phi_\psi(s)s = -s\phi_\psi(s)$ and for $S_\psi \ni s' \neq s$, $\phi_\psi(s)s' = s'\phi_\psi(s)$. This is sometimes call the "destabilizer" of S_ψ.

Fix an encoding map (using only CNOT) $\mathcal{E}$ for Q_{distill} such that for $j \in [k]$, $\mathcal{E}Z_j\mathcal{E}^{-1} = Z^{G[j,\cdot]}$ and $\mathcal{E}X_j\mathcal{E}^{-1} = X^{G[j,\cdot]}$. This fixes a phase of Pauli Y, $\mathcal{E}Y_j\mathcal{E}^{-1} = (i)^{|G[j,\cdot]|-1}Y^{G[k,\cdot]} \equiv \theta_j Y_j$ where $\theta_j \in \{\pm 1\}$. We will need to "fix up" this phase in order to produce the correct state on the output. For $P \in \mathcal{P}^n$, let us notate the phase of an arbitrary Pauli under the encoding map $\mathcal{E}^{\otimes b}P_j\left(\mathcal{E}^{-1}\right)^{\otimes b} = \theta_{P,j}P$. Define the fix up operator $\Xi \in \mathcal{P}^{k\times b}$ be a Pauli operator that corrects this phase: $s \in S_\psi$, $\Xi s^{e_j} = \theta^b_{s,j}s^{e_j}\Xi$. Ξ can be computed by taking the product of destabilizer operators $\phi_\psi(s)$ on position j whenever the phase is incorrect $\left(\theta_{s,j}\right)^b = -1$.

$$\Xi = \prod_{\substack{j\in[k]\\ s\in S_\psi\\ \theta_{s,j}=-1}} \left(\phi_\psi(s)\right)^{e_j} \tag{2.83}$$

For $d, n \in \mathbb{N}$, define $\mathcal{S}^n_d = \{x \subseteq [n] \mid |x| = d\} \subseteq P([n])$ to be all subsets of $[n]$ of size d. Let $\mathcal{F}_{\text{distill}} = \mathcal{S}^n_{t+1} \bullet \mathcal{S}^b_1$ be the family of subsets of $[n] \times [b]$ that contain one element from each of $t + 1$ rows.

Proposition 2.4.9 (Single level stabilizer state distillation)**.** *In Algorithm 1, if ρ is $\mathcal{F}_{\text{distill}}$-deviated from $\psi^{\otimes n}$, then $\rho' = \psi^{\otimes k}$.*

Proof. By Lemma 2.2.31, it suffices for us to consider ρ that differs from $\psi^{\otimes n}$ by a Pauli operator E supported on an $\mathcal{F}_{\text{distill}}$-avoiding set i.e. is $\mathcal{F}_{\text{distill}}$-diagonal Pauli-deviated from $\psi^{\otimes n}$. Otherwise, it is in the kernel of the measurement channel.

First, note that $\psi^{\otimes n}$ is the simultaneous +1 eigenstate of the set of operators $\{s^{e_i}\}_{i\in[n],s\in S_\psi}$. I.e. the measurement of the set of operators s^H on $\psi^{\otimes n}$ is 0. Thus, measurement of s^H on ρ (an eigenstate) will produce a vector $\sigma \in \mathbb{F}_2^r$ such that [42] $s^H E = (-1)^\sigma E s^H$. We will indirectly measure these operators.

The measurements at Algorithm 1 all commute. When a measured operator O is not in the stabilizer S of the state, the subgroup S' of S commuting with O is retained and

[42] The phase should be interpreted "coordinate-wise."

Algorithm 1 Stabilizer state distillation algorithm

Input: $n \times b$ qubit state ρ
Output: $k \times b$ qubit state ρ_{out}

$m^{(X)} \in \mathbb{F}_2^{r \times b}$
$m^{(Z)} \in \mathbb{F}_2^{r \times b}$
$\sigma \in \mathbb{F}_2^{r}$
$U \leftarrow I$
for $j \in [b]$ **do** ▷ *Measure checks of* Q_{distill}
    $m[\cdot, j]^{(X)} \leftarrow$ Measurement of X^H on $\rho|_{[n]\times\{j\}}$
    $m[\cdot, j]^{(Z)} \leftarrow$ Measurement of Z^H on $\rho|_{[n]\times\{j\}}$
$\rho' \leftarrow$ Post-measurement state of ρ
for $s \in S_\psi$ **do** ▷ *Correct eigenvalue of* s^H
    $(a, z, x) \leftarrow z, x \in \mathbb{F}_2^b, a \in \mathbb{F}_2$ such that $s = \pm i^a Z^z X^x$ ▷ *Decompose*
    for $i \in [r]$ **do** ▷ *Compute syndrome of* s^H
        $h \leftarrow H[i, \cdot]$ ▷ *Inferred measurement of* s^h
        $\sigma_i \leftarrow a^{|h|/2} + \langle x, m[i, \cdot]^{(X)} \rangle + \langle z, m[i, \cdot]^{(Z)} \rangle$
    $c \leftarrow \mathcal{D}_H(\sigma)$ ▷ *Compute correction*
    $P \leftarrow \phi_\psi(s)$
    $U \leftarrow UP^c$ ▷ *Update correction*
▷ *Apply correction and unencode. Discarding qubits in* $([n] \setminus [k]) \times [b]$ ◁
$\rho_{\text{decoded}} \leftarrow \left(\mathcal{E}^{-1}\right)^{\otimes b} \circ U(\rho)$
$\rho_{\text{out}} \leftarrow \Xi(\rho_{\text{decoded}})$ ▷ *Fix up phases*

the new stabilizer is $\langle S', (-1)^a O \rangle$ for a random phase a (the measurement outcome). The decomposition at Algorithm 1, allows us to write $s^h = (\pm i^a)^{|h|} (Z^z)^h (X^x)^h$ at Algorithm 1. By assumption of the code properties, $H^T H = 0$, so $|h|$ is even. Furthermore, $(Z^z)^h (X^x)^h$ is proportional to the products of a subset of $\cup_{i\in[b]} (X_i)^h$ and $\cup_{i\in[b]} (Z_i)^h$. These operators all pairwise commute and were previously measured. Thus, the quantity computed at Algorithm 1 is the current eigenvalue of s^h on the post-measurement state (i.e. is the variable σ from the previous paragraph). The correction operator computed at Algorithm 1 modifies only the eigenvalues of s^H (and of s^G). By definition of the decoding map and ϕ_ψ,

$$s^H P^c E = (-1)^{Hc+\sigma} P^c E s^H = P^c E s^H \tag{2.84}$$

and for any $s' \in S_\psi$ with $s \neq s'$, $(s')^H P^c = P^c (s')^H$.

It remains to analyze $R := UE$. We will show that R commutes with $\{s^{e_i}\}_{i\in[n], s\in S_\psi}$. Since E was supported on a $\mathcal{F}_{\text{distill}}$-avoiding set, there exists a subset $I \subseteq [n]$ with $|I| \leq t$ such that $\operatorname{supp} E \subseteq I \times [b]$.

Consider an operator $s \in S_\psi$ and let all variables take their values at the end of the loop trip (starting at Algorithm 1) corresponding to s. Define the vectors $x, y \in \mathbb{F}_2^n$ such that for $i \in [n]$

$$s^{e_i} E = (-1)^{x[i]} E s^{e_i} \tag{2.85}$$

$$s^{e_i} (P^c E) = (-1)^{y[i]} (P^c E) s^{e_i} \tag{2.86}$$

Note that $Hx = \sigma$. Since E is supported on at most t columns ($|I| \leq t$), $|x| \leq t$ and so $\mathcal{D}_H(\sigma) = \mathcal{D}_H(Hx) = x = c$. Thus, $y = c + x = 0$, so R commutes with $\{s^{e_i}\}_{i \in [n], s \in S_\psi}$, the generators of $\mathrm{Stab}(\psi^{\otimes n})$.

For a Pauli operator P and a set $B \subseteq \mathcal{P}^n$, denote $\{PbP \mid b \in B\}$ by PBP. Since $\mathrm{Stab}(\rho) = E\,\mathrm{Stab}(\psi^{\otimes n})E$, the post measurement state has stabilizer given by[43]

$$\mathrm{Stab}(\rho') = \langle \{E s^G E\}_{s \in S_\psi}, \cup_{j \in [b]} (-1)^{m^{(X)}[\cdot, j]} (X_j)^H, \cup_{j \in [b]} (-1)^{m^{(Z)}[\cdot, j]} (Z_j)^H \rangle \tag{2.87}$$

Using the commutativity of R with $s^G \subseteq \mathrm{Stab}(\psi^{\otimes n})$, the post correction state has stabilizer

$$\mathrm{Stab}(U(\rho')) = \langle \{s^G\}_{s \in S_\psi}, \pm \cup_{j \in [b]} (X_j)^H, \pm \cup_{j \in [b]} (Z_j)^H \rangle \tag{2.88}$$

$(\mathcal{E}^{-1})^{\otimes b}$ unencodes the blocks. That is, for each row $i \in [k]$ of G, each $j \in [b]$, and $P \in \{X, Z\}$, $(\mathcal{E}^{-1})^{\otimes b}(P_j^{G[i,\cdot]}) = P_{(i,j)}$. Any operator of the form P_j^H is mapped to an operator supported only on the qubits $([n] \setminus [k]) \times [b]$ which are discarded. Thus $\mathrm{Stab}(\rho_{\text{decoded}}) = \langle \{\theta^b_{s,i} s^{e_i}\}_{i \in [k], s \in S_\psi} \rangle$. Finally, the application of Ξ fixes the phases so that $\mathrm{Stab}(\rho_{\text{decoded}}) = \langle \{s^{e_i}\}_{i \in [k], s \in S_\psi} = \mathrm{Stab}(\psi^{\otimes k})$. □

Proof of Lemma 2.4.7

Lemma 2.4.7 (State distillation procedure for stabilizer states)**.** *For a stabilizer state ψ on b qubits, there exists constants $\epsilon_{*,distill} \in (0,1)$, $\beta_{\text{distill}} > 0$ and a family indexed by $l \in \mathbb{N}$ of $(M_l, K_l, \mathcal{A}_l)$-state distillation procedure where $\mathcal{W}(\mathcal{A}_l;\, x) \leq (\beta_{\text{distill}} x)^{2^\ell}$ on $x \in [0, \epsilon_{*,distill}]$, $\frac{K_l}{M_l} = \Theta(1)$, and $M_l = e^{\Theta(l \log l)}$.*

Furthermore, the procedure satisfies:

- *The quantum depth is $O(l^3)$, and the width is $O(M_l)$.*
- *The classical computation has total depth $O(l^3)$.*

[43] The phase on the last two terms is not important.

- *No additional input states are required beyond the M_l noisy inputs.*
- *Only the operations* CNOT*, Z/X basis measurement, and classically-controlled Pauli gates are used.*
- *With the exception of Pauli gates, the quantum gates applied do not depend on the state ψ to be distilled, only b and l.*
- *No two-qubit gates are applied between qubits with different $[b]$ coordinates.*[44]

Proof. The quantum Hamming code satisfies the conditions required of C_{distill} in Section 2.4 [Ste96b; YK24]. H is the check matrix of a classical Hamming code and t=1. For $r \geq 3$, the parameters of the quantum Hamming code are $[[n = 2^r - 1, k = 2^r - 2r - 1, 3]]$.

We will repeatedly apply the state distillation procedure Section 2.4 with quantum Hamming codes of increasing size. In particular, we will use the sequence $(r_1, r_2, \ldots, r_l)$ where $r_\ell = \lfloor 2\log_2(4\ell) \rfloor$. Let $n_\ell = 2^{r_\ell} - 1$, $k_i = 2^{r_\ell} - 2r_\ell - 1$, $M_l := \prod_{\ell=1}^{l} n_\ell$, and $K_l := \prod_{\ell=1}^{l} k_\ell$

We will start with blocks of b qubits indexed by $[n_l] \times \cdots \times [n_1]$ initialized as $M_l := \prod_{\ell=1}^{l} n_\ell$ noisy copies of ψ. Iterating from $\ell = 1$ to $\ell = l$, for each $I \in [n_l] \times \cdots \times [n_{\ell+1}]$ and $J \in [k_{\ell-1}] \times \cdots \times [k_1]$ we apply the state distillation procedure Section 2.4 on the set of blocks $\{I\} \times [n_\ell] \times \{J\}$ to get a new set of blocks with labels $\{I\} \times [k_\ell] \times \{J\}$. Let $\mathcal{S}_d^n$ be the family of all subsets of $[n]$ of size d. At step ℓ, Proposition 2.4.9 gives that the output is $\psi^{\otimes k_\ell}$ if the input is $\mathcal{S}_2^{n_\ell} \bullet \mathcal{S}_1^b$-deviated from $\psi^{\otimes n_\ell}$. We use the fact that, by construction, no two outputs from a single application of the state distillation procedure are used together in a following state distillation procedure: If the output of a state distillation in step ℓ is not $\psi^{\otimes k_\ell}$, then the input is not $\mathcal{S}_2^{n_\ell}$-deviated from $\psi^{\otimes n_\ell}$. Since each input is from separate state distillation procedures, $t + 1$ distinct state distillation procedures at step $\ell - 1$ must have had inputs that are not $\mathcal{S}_2^{n_{\ell-1}}$-deviated from $\psi^{\otimes n_{\ell-1}}$. Inducting from $\ell = 1$, the output of the final iteration is ψ^{K_l} if the input is $\mathcal{A}_l := \mathcal{S}_2^{n_l} \bullet \cdots \bullet \mathcal{S}_2^{n_1} \bullet \mathcal{S}_1^b$-deviated from $\psi^{\otimes M_l}$.

[44] In other words, the gates are applied "transversally" with regard to the qubits $[b]$.

$\mathcal{W}(\mathcal{S}_d^n;\, x) = \binom{n}{d} x^d \le n^d x^d$, so using Proposition 2.2.21, for $x \ge 0$, we can upper bound

$$\mathcal{W}(\mathcal{A}_l;\, x) = \mathcal{W}(\mathcal{S}_2^{n_l};\, \mathcal{W}(\mathcal{S}_2^{n_{l-1}};\, \ldots \mathcal{W}(\mathcal{S}_2^{n_1};\, \mathcal{W}(\mathcal{S}_1^b;\, x)))) \tag{2.89}$$

$$\le (bx)^{2^l} \left(\prod_{i=1}^{l} n_i^{2^{(l-i)}} \right)^2 \tag{2.90}$$

$$\le (bx)^{2^l} \left(\prod_{i=1}^{l} (4i)^{2^{(l-i)}} \right)^2 \tag{2.91}$$

$$\le (64bx)^{2^l} \tag{2.92}$$

Where we have used $\sum_{i=1}^{\infty} 2^{-i} \log_2(4i) \le 3$. Then, $\beta_{\text{distill}} = 64b$ and $\epsilon_{*,distill} = 1/\beta_{\text{distill}}$.

We now turn our attention to the other parameters. First, $(\ell!)^2 4^{2\ell} = \prod_{i=1}^{\ell} 2^{\log_2(4\ell)^2} \le M_l \le \prod_{i=1}^{\ell} \frac{1}{4} 2^{\log_2(4\ell)^2} = (\ell!)^2 4^{\ell}$, so $M_l = e^{\Theta(l \log l)}$. For the rate,

$$\frac{K_l}{M_l} = \prod_{\ell=1}^{l} \frac{k_\ell}{n_\ell} \tag{2.93}$$

$$\ge \prod_{\ell=1}^{\infty} \left(1 - \frac{2r_\ell}{n_\ell} \right) \tag{2.94}$$

$$\ge \frac{1}{7} \prod_{\ell=2}^{\infty} \left(1 - \frac{8 \log_2(4\ell)^2}{(4\ell)^2} \right) \tag{2.95}$$

$$\ge \frac{1}{300} \tag{2.96}$$

where we have upper bounded the first term in the product separately and numerically evaluated $\prod_{\ell=2}^{\infty} \left(1 - \frac{8 \log_2(4\ell)^2}{(4\ell)^2} \right) \ge \frac{1}{300}$.

To prove the circuit properties, first note that using a length-n CSS code, syndrome measurement followed by unencoding can be done in depth $O(n^2)$ using only CNOT, and X/Z measurement without any ancilla qubits: For the encoding with logical operators X^G and Z^G used in the previous section, there exists a unitary encoding circuit U supported on n qubits of depth $O(n^2)$ [CG97; Got97] that uses only CNOT. For an n-qubit state and r_x, r_z such that $r_x + r_z = n - k$, $U |\psi\rangle |0\rangle^{r_z} |+\rangle^{r_x} = \left|\overline{\psi}\right\rangle$. U^{-1} maps a set of stabilizer generators of the code to $\{Z_i\}_{i=n-k}^{n-k+r_z} \cup \{X_i\}_{i=n-k+r_z}^{n}$, so executing the circuit U^{-1} followed by measurement of the last $n-k$ qubits (in either the X or Z basis) will produce a measurement outcome $a \in \mathbb{F}_2^{n-k}$ and state on the first k qubits that is equivalent to first measuring the stabilizer generators of the code and then applying U^{-1}.

Using this circuit, the overall circuit depth is $O(l \cdot n_l^2) = O(l^3)$ and uses only CNOT, X and Z basis measurements, and classically controlled Pauli gates. The main classical computation includes decoding of the quantum Hamming code, which is of similar time complexity. With the exception of the classically controlled Pauli gates, the circuit is completely determined by l and b. Since no ancillas are prepared, the width is $O(M_l)$. □

2.5 Magic state distillation with almost-constant spacetime overhead

We will distill the magic state $|\mathsf{CCZ}\rangle$. To avoid large overhead, we will require a very efficient distillation scheme. The majority of the section will be devoted to establishing the following theorem about punctured quantum Reed-Solomon codes over extension fields of $\mathbb{F}_2$.

Theorem 2.5.1. *For each $q = 2^l$ with $l \geq 3$, there exists a CSS qudit code (punctured quantum Reed-Solomon code) with parameters*

$$\left[\left[\frac{3q}{4}, \frac{q}{4}, \left\lfloor \frac{q}{3} \right\rfloor - \frac{q}{4} + 1\right]\right]_q, \tag{2.97}$$

such that $\mathsf{CCZ}_q^{\otimes 3q/4}$ acts as logical $\overline{\mathsf{CCZ}_q}^{\otimes q/4}$ on a basis of logical qudits.

The construction is similar to that of Krishna and Tillich [KT19] with the main difference being the use of extension fields. To simplify notation, we will specialize to characteristic 2 though this fact is used in a non-essential way. In Section 2.5 we will employ the qudit PQRS code to construct a distillation protocol for the qubit CCZ state with almost-constant spacetime overhead, proving Theorem 2.4.8.

Polynomials over finite fields

We begin by making some standard definitions and recalling some standard results about polynomials over finite fields. Fix some prime power $q \in \mathbb{N}$.

Notate the set of polynomials in the variable x with coefficients in $\mathbb{F}_q$ by $\mathbb{F}_q[x]$. For $k \in \mathbb{N}$, define $\mathbb{F}_q[X]_{<k} = \{P \in \mathbb{F}_q[x] \mid \deg P < k\}$ with degree of the zero polynomial defined to be $-\infty$. We use $\mathbb{F}_q^*$ to denote the set of non-zero elements of $\mathbb{F}_q$.

For a classical code C, the star product is given by coordinate-wise multiplication. I.e. for $a = (a_1, a_2, \ldots, a_n) \in C$, $b = (b_1, b_2, \ldots, b_n) \in C$,

$$a * b := (a_1 b_1, a_2 b_2, \ldots a_n b_n).$$

We use C^{*2} to denote the set $\{a * b \mid a, b \in C\}$.

For some $k \in \mathbb{N}$, fix a set of coordinates $A = \{\alpha_i\}_{i=1}^{k} \subseteq \mathbb{F}_q$ that will later be used for a systematic encoding for a Reed-Solomon code. Recall that a Reed-Solomon code is evaluations of the set $\mathbb{F}_q[X]_{<k}$ on $\mathbb{F}_q$. In order to obtain a systematic encoding, we will parameterize $\mathbb{F}_q[x]_{<k}$ in terms of the evaluations on points of A.

Definition 2.5.2. *The Lagrange interpolation polynomials for the set A are*

$$\ell_i^{(A)}(x) = \prod_{j \in [k] \setminus \{i\}} \frac{x - \alpha_j}{\alpha_i - \alpha_j}.$$

One can see that the interpolation polynomial $\ell_i^{(A)}(x)$ is zero on $A \setminus \{\alpha_i\}$ and one on α_1.

Definition 2.5.3 (Message polynomial)**.** *For a message $(m_1, m_2, \ldots m_k) \in \mathbb{F}_q^k$, the message polynomial $p_m^{(A)}(x)$ corresponding to m is*

$$p_m^{(A)}(x) = \sum_{i \in [k]} m_i \ell_i^{(A)}(x).$$

A message polynomial uniquely specifies a codeword of the code and also gives a handle on coordinate-wise multiplication of codewords. It follows from Definition 2.5.2 that $p_m^{(A)}(\alpha_i) = m_i$. Our Reed-Solomon codes are evaluations of these message polynomials, so we will find it useful to convert between codewords and the corresponding message polynomials.

Definition 2.5.4 (Evaluation map)**.** *For an evaluation set $M = \{\alpha_i\}_{i=1}^{n} \subseteq \mathbb{F}_q$, we define the evaluation map $\phi_M : \mathbb{F}_q[x] \to \mathbb{F}_q^n$*

$$P \mapsto (P(\alpha_1), P(\alpha_2), \ldots, P(\alpha_n)).$$

We will occasionally write ϕ_M^{-1} to denote the map that returns the unique lowest degree polynomial with the given evaluation set. I.e. for any $c \in \mathbb{F}_q^n$, $P = \phi_M^{-1}(c)$ satisfies $\deg P \leq n - 1$ and $\phi_M(P) = c$.

Our main tool to get a handle on products of polynomials will be Proposition 2.5.6, which states that the average over all field elements of low-degree polynomials is zero.

Proposition 2.5.5. *For any non-zero element of $a \in \mathbb{F}_q$,*

$$a^{q-1} = 1.$$

Proof. The product over all non-zero field elements is non-zero and invariant under shifts by a, so it must be the case that $a^{q-1} = 1$.

$$\prod_{x \in \mathbb{F}_q^*} x = \prod_{x \in \mathbb{F}_q^*} ax = a^{q-1} \prod_{x \in \mathbb{F}_q^*} x \in \mathbb{F}_q^*. \qquad \square$$

Proposition 2.5.6. *For a polynomial $p \in \mathbb{F}_q[x]$ with leading coefficient β*

$$\sum_{x \in \mathbb{F}_q} p(x) = \begin{cases} 0 & 0 \leq \deg p < q-1 \\ -\beta & \deg p = q-1 \end{cases}.$$

Proof. Let $\alpha \in \mathbb{F}_q$ be a primitive element for $\mathbb{F}_q$ (which always exists). Consider a monomial of degree k summed over the field $\mathbb{F}_q$. This sum is invariant under shifts by α,

$$\sum_{x \in \mathbb{F}_q} x^k = \sum_{x \in \mathbb{F}_q} (\alpha x)^k = \alpha^k \sum_{x \in \mathbb{F}_q} x^k.$$

When $k = 0$, the summand is constant and the number of terms in the sum is a multiple of the characteristic. For $k \in (0, q-1)$, $\alpha^k \neq 1$, so it must be the case that the sum is zero.

When $k = q-1$, Proposition 2.5.5 gives that there are $q-1$ non-zero terms in the sum, each equal to 1. The sum is then equal to $q-1$, the additive inverse of 1. $\square$

Extension fields and qudits

Later, in order to perform distillation, we will need to represent our qudits over the alphabet $\mathbb{F}_{p^l}$ in terms of qudits over $\mathbb{F}_p$. For the remainder of the section fix $q = p^l$ for some prime p. To do this, we will need to introduce a basis. This will not enter in any essential way until Section 2.5.

Bases

For an extension field $\mathbb{F}_{p^l}$ of degree-l over $\mathbb{F}_p$, a basis for $\mathbb{F}_{p^l}$ is a set $\{\alpha_i\}_{i \in [l]}$ such that every element $x \in \mathbb{F}_{p^l}$ has a unique decomposition $x_1\alpha_1 + x_2\alpha_2 + \cdots + x_l\alpha_l$ with $\{x_i\}_{i \in [l]} \subseteq \mathbb{F}_p$. We will also make use of the trace map $\mathrm{tr}\colon \mathbb{F}_{p^l} \to \mathbb{F}_p$ given by

$$\mathrm{tr}(x) = \sum_{i=1}^{l} x^{p^{i-1}}. \tag{2.98}$$

The trace map is $\mathbb{F}_p$-linear.

In this section, we will use two different bases which simplifies different operations.

Fact 2.5.7 (Polynomial basis [MP13])**.** *Let α be a root of a degree l polynomial irreducible over $\mathbb{F}_p$. Then $\{1, \alpha, \alpha^2, \ldots, \alpha^l\}$ forms a basis, known as a polynomial basis for $\mathbb{F}_{p^l}$ over $\mathbb{F}_p$.*

This fact arises from the standard construction of $\mathbb{F}_{p^l}$ as the quotient of $\mathbb{F}_p[X]$ by an irreducible polynomial. It will lend itself to easy multiplication of elements of $\mathbb{F}_{p^l}$ using operations in $\mathbb{F}_p$.

The second basis we will use is known as a self-dual basis.

Fact 2.5.8 (Existence of self-dual bases [SL80])**.** *Let $q = p^l$ for some prime p. For q even or q and l odd, there exists an self-dual basis $\{\sigma_i\}_{i\in[l]} \subset \mathbb{F}_q$ for $\mathbb{F}_q$ over $\mathbb{F}_p$ This basis satisfies*

$$\operatorname{tr}\left(\alpha_i \alpha_j\right) = \begin{cases} 1 & i = j \\ 0 & i \neq j \end{cases}. \tag{2.99}$$

For an element $x \in \mathbb{F}_{p^l}$ and a basis $N = \{\alpha_i\}_{i\in[l]}$ of $\mathbb{F}_{p^l}$ over $\mathbb{F}_p$, we use $x_i^{(N)} \in \mathbb{F}_p$, $i \in [l]$ to denote the expansion of x in the basis N i.e. $x = \sum_{i\in[l]} \alpha_i x_i^{(N)}$. When clear from context, we will drop the superscript N to avoid cluttering the notation.

$\mathbb{F}_q$-Qudits and CSS codes over $\mathbb{F}_q$

We will define CSS codes over qudits of dimension q. Let ω be a p-th roof of unity. For an element $a \in \mathbb{F}_p$, we will use the abuse of notation ω^a to indicate exponentiation after taking the canonical inclusion $\mathbb{F}_p \to \mathbb{Z}$.

For $a, b, x \in \mathbb{F}_q$, define the generalized Pauli operators (where addition is taken in $\mathbb{F}_q$)

$$X^{(q)}(a)\,|x\rangle = |x + a\rangle\,, \qquad\qquad Z^{(q)}(b)\,|x\rangle = \omega^{\operatorname{tr}(bx)}\,|x\rangle\,.$$

So that

$$\omega^{\operatorname{tr}(ab)} X^{(q)}(a) Z^{(q)}(b) = Z^{(q)}(b) X^{(q)}(a).$$

For $x, y, z \in \mathbb{F}_q$, define

$$\mathsf{CNOT}^{(q)}\,|x\rangle\,|y\rangle = |x\rangle\,|x + y\rangle\,,$$

$$\mathsf{CZ}^{(q)} |x\rangle |y\rangle = \omega^{\mathrm{tr}(xy)} |x\rangle |y\rangle , \qquad \mathsf{CCZ}^{(q)} |x\rangle |y\rangle |z\rangle = \omega^{\mathrm{tr}(xyz)} |x\rangle |y\rangle |z\rangle .$$

For linear codes $C_1, C_2 \subseteq \mathbb{F}_q$ such that $C_2^\perp \subseteq C_1$, the CSS code $\mathrm{CSS}(C_1, C_2)$ is the span of the states

$$c \in C_1 \qquad \left|c + C_2^\perp\right\rangle = \frac{1}{\sqrt{|C_2^\perp|}} \sum_{\alpha \in C_2^\perp} |c + \alpha\rangle . \tag{2.100}$$

$\mathbb{F}_q$-Qudits using $\mathbb{F}_p$-qudits

We would like to construct our qudits out of smaller qudits over the base field. We will use $|\psi\rangle_q$ to denote a qudit state with local dimension q (when different from 2). For a basis N, we will use the following representation of the qudit state

$$|x\rangle_q^{(N)} = \left|x_1^{(N)}\right\rangle \left|x_2^{(N)}\right\rangle \ldots \left|x_l^{(N)}\right\rangle . \tag{2.101}$$

When N is self-dual, we have that $\mathrm{tr}(xy) = \sum_{i \in [l]} x_i^{(N)} y_i^{(N)}$. This simplifies certain Clifford gates. In particular,

$$X^{(q)}(a) = X^{(p)}(a_1) \otimes X^{(p)}(a_2) \otimes \cdots \otimes X^{(p)}(a_m), \tag{2.102}$$

$$Z^{(q)}(a) = Z^{(p)}(a_1) \otimes Z^{(p)}(a_2) \otimes \cdots \otimes Z^{(p)}(a_m), \tag{2.103}$$

$$\mathsf{CZ}^{(q)}(a) = \mathsf{CZ}^{(p)} \otimes \mathsf{CZ}^{(p)} \otimes \cdots \otimes \mathsf{CZ}^{(p)}, \tag{2.104}$$

$$\mathsf{CNOT}^{(q)}(a) = \mathsf{CNOT}^{(p)} \otimes \mathsf{CNOT}^{(p)} \otimes \cdots \otimes \mathsf{CNOT}^{(p)}. \tag{2.105}$$

We will leave the construction of $\mathsf{CCZ}^{(q)}$ using $\mathsf{CCZ}^{(p)}$ to a later section. For the case $q = 2$, we omit the subscript/superscript.

Punctured quantum Reed-Solomon code

Fix $q = 2^l$ and parameters $k, m, s \in \mathbb{N}$ such that $q/3 \geq k \geq m \geq s > 0$ and $2k \leq q - m$. Fix an set of coordinates $A \subseteq \mathbb{F}_q$ of size k. We further will introduce a new set of coordinates $B \subseteq A$ such that $|B| = s$. Fix an evaluation set $M = \{\alpha_i\}_{i=1}^{q-s} = \mathbb{F}_q \setminus B$.

Definition 2.5.9 (Punctured quantum Reed-Solomon Code)**.** *We begin by defining two sets of polynomials*

$$\mathcal{P}_1 = \mathbb{F}_q[x]_{<k}, \tag{2.106}$$

$$\mathcal{P}_2^\perp = \{P \in \mathbb{F}_q[x]_{<m} \mid P(B) = 0\}. \tag{2.107}$$

Define $C_1 = \phi_M(\mathcal{P}_1)$ to be the evaluations of $\mathcal{P}_1$ on M and $C_2^\perp = \phi_M(\mathcal{P}_2^\perp)$ to be the evaluations of $\mathcal{P}_2^\perp$ on M. The punctured quantum Reed-Solomon code is defined to be $C = \mathrm{CSS}(C_1, C_2)$.

We will pick a basis parameterized by $m = (m_1, m_2, \ldots, m_{|A|}) \in \mathbb{F}_q^{|A|}$ *defined in the following way: For any* m*, the corresponding codeword* c_m *is* $\phi_M(p_m^{(A)})$*, the evaluation of the corresponding message polynomial with systematic encoding positions* A*. The code state* $|\overline{m}\rangle$ *is then defined to be*

$$|\overline{m}\rangle = \frac{1}{\sqrt{|C_2^\perp|}} \sum_{\alpha \in C_2^\perp} |c_m + \alpha\rangle = \left|c_m + C_2^\perp\right\rangle. \tag{2.108}$$

Before proving properties of this code, it is helpful to first compute C_2:

Lemma 2.5.10. *C_2 is the evaluations of $\mathcal{P}_2 = \mathbb{F}_q[x]_{<q-m}$ on M.*

Proof. We first write the orthogonal complement of $C_2^\perp$ as the evaluations of the set of polynomials:

$$\mathcal{P}_2' = \{P \in \mathbb{F}_q[x]_{<q} \mid \forall Q \in \mathcal{P}_2^\perp \sum_{\alpha \in M} P(\alpha)Q(\alpha) = 0\}. \tag{2.109}$$

Clearly, $C_2 = \phi_M(\mathcal{P}_2')$. We will show that this set is $\mathbb{F}_q[x]_{<q-m}$. First note that for any non-zero $Q \in \mathcal{P}_2^\perp$, it is divisible by the polynomial $R = \prod_{b \in B}(x - b)$, so there is a unique polynomial $Q' \in \mathbb{F}_q[x]$ such that $Q = Q'R$ and $\deg Q' = \deg Q - s$. R is zero on B, so we may extend the sum to write $\mathcal{P}_2'$ as

$$\mathcal{P}_2' = \{P \in \mathbb{F}_q[x]_{<q} \mid \forall Q' \in \mathbb{F}_q[x]_{<m-s} \sum_{\alpha \in \mathbb{F}_q} P(\alpha)Q'(\alpha)R(\alpha) = 0\}. \tag{2.110}$$

In the condition, if $\deg P \geq q - m$, then there exists some $Q' \in \mathbb{F}_q[x]_{<m-s}$ such that $\deg P + \deg Q' + s = q - 1$, making the sum nonzero by Proposition 2.5.6. Otherwise, the sum is zero, so

$$\mathcal{P}_2' = \mathbb{F}_q[x]_{<q-m} \equiv \mathcal{P}_2. \tag{2.111}$$

□

Proposition 2.5.11. *C is a quantum CSS code with parameters*

$$[[q - s, k - m + s, \min(q - k, m) - s + 1]]_q.$$

Proof. We have the degree bounds $k \leq m$ so $\mathcal{P}_2^\perp \subseteq \mathcal{P}_1$ implying $C_2^\perp \subseteq C_1$. Thus C is a valid quantum CSS code. By dimension counting, C encodes $k - m - s$ qubits.

For the distance, recall that the distance of $\mathrm{CSS}(C_1, C_2)$ is lower bounded by the minimum of the distances of C_1 and C_2. C_1 is the puncturing of $\mathrm{RS}(q, k)$ on s coordinates and so has distance at least $(q - k + 1) - s$. Likewise, by Lemma 2.5.10, C_2 is the puncturing of $\mathrm{RS}(q, q - m)$ and has distance at least $(m + 1) - s$. □

CCZ for punctured quantum Reed-Solomon codes

CCZ requires us to analyze the coordinate-wise product of codewords. In particular, we will encounter products of the form $\langle C_1 * C_1, C_2^\perp \rangle$ that may cause the action of $\mathsf{CCZ}^{\otimes n}$ to be non-constant on cosets of $C_2^\perp$. A sufficient condition for these products to vanish is given by the following lemma.

Lemma 2.5.12. $C_1^{*2} \subseteq C_2$.

Proof. Fix two codewords $c_1, c_2 \in C_1$, and let $p_1, p_2 \in \mathcal{P}_1$ be the corresponding polynomials such that $p_1 = \phi_M^{-1}(c_1)$ and $p_2 = \phi_M^{-1}(c_2)$. Then, $c_1 * c_2 = \phi_M(p_1 p_2)$ is the evaluation of $p_1 p_2$ on M, and $\deg(p_1 p_2) \leq \deg p_1 + \deg p_2 < 2k$. By assumption on the parameters, $2k \leq q - m$. It follows that $p_1 p_2 \in \mathcal{P}_2$, so $c_1 * c_2$ is an element of C_2 by Lemma 2.5.10. □

Before proceeding to multiplication of codewords of C_1, it is convenient to first start with multiplication of the interpolation polynomials.

Lemma 2.5.13. *For three interpolation polynomials*

$$\sum_{x \in M} \ell_a^{(A)}(x) \ell_b^{(A)}(x) \ell_c^{(A)}(x) = \begin{cases} -1 & a = b = c \in B \\ 0 & \textit{otherwise} \end{cases} . \tag{2.112}$$

Proof. By construction, the product has degree strictly less than $q - 1$, so it follows from Proposition 2.5.6 that

$$\sum_{x \in \mathbb{F}_q} \ell_a^{(A)}(x) \ell_b^{(A)}(x) \ell_c^{(A)}(x) = 0 . \tag{2.113}$$

This allows us to break up the sum into the evaluated coordinates and the removed coordinates for which we have good control over the values of the interpolation polynomials:

$$\sum_{x \in M} \ell_a^{(A)}(x) \ell_b^{(A)}(x) \ell_c^{(A)}(x) = -\sum_{x \in B} \ell_a^{(A)}(x) \ell_b^{(A)}(x) \ell_c^{(A)}(x). \tag{2.114}$$

By construction of the interpolation polynomials, if $a = b = c \in B$, the summand vanishes on all elements of B except for one element where it takes the value 1. In any other case, the product is identically zero on B. □

This straightforwardly allows us to multiply codewords.

Corollary 2.5.14. *Let $p^{(A)}_{m_1}(x)$, $p^{(A)}_{m_2}(x)$, and $p^{(A)}_{m_3}(x)$ be three message polynomials with the corresponding codewords of C_1 denoted by c_1, c_2, and c_3. Then*

$$\sum_{i\in M}(c_1)_i(c_2)_i(c_3)_i = -\sum_{i\in B}(m_1)_i(m_2)_i(m_3)_i. \tag{2.115}$$

Proof. We write

$$\sum_{i\in M}(c_1)_i(c_2)_i(c_3)_i = \sum_{x\in M} p^{(A)}_{m_1}(x)p^{(A)}_{m_2}(x)p^{(A)}_{m_3}(x), \tag{2.116}$$

and then use Definition 2.5.3 to expand the message polynomials in terms of the interpolation polynomials. The result follows after application of Lemma 2.5.13. □

Proposition 2.5.15. *For three codestates $|\overline{m}_1\rangle$, $|\overline{m}_2\rangle$, and $|\overline{m}_3\rangle$,*

$$\begin{aligned}\mathsf{CCZ}^{\otimes n}|\overline{m}_1\rangle|\overline{m}_2\rangle|\overline{m}_3\rangle &= (-1)^{\mathrm{tr}\langle m_1^{(B)}*m_2^{(B)},m_3^{(B)}\rangle}|\overline{m}_1\rangle|\overline{m}_2\rangle|\overline{m}_3\rangle\\ &\equiv \overline{\mathsf{CCZ}}^{(B)}|\overline{m}_1\rangle|\overline{m}_2\rangle|\overline{m}_3\rangle\,.\end{aligned}$$

Where, $i \in 1,2,3$, $m_i^{(B)}$ refers to the restriction of m_i to the coordinates in B, and $\overline{\mathsf{CCZ}}^{(B)}$ refers to a logical CCZ acting coordinate-wise on the logical qubits labeled by coordinates of B.

Proof. First we expand the definition of codestates (Eq. (2.108)) into a sum over elements of $C_2^\perp$. Writing c_i for the codeword of C_1 corresponding to the evaluations of $p_{m_i}(x)$, we have

$$\mathsf{CCZ}^{\otimes n}|\overline{m}_1\rangle|\overline{m}_2\rangle|\overline{m}_3\rangle \propto \sum_{\alpha,\beta,\gamma\in C_2^\perp}(-1)^{\mathrm{tr}\langle(c_1+\alpha)*(c_2+\beta),(c_3+\gamma)\rangle}|c_1+\alpha\rangle|c_2+\beta\rangle|c_3+\gamma\rangle\,. \tag{2.117}$$

After distributing star product, using linearity of the inner product, and Lemma 2.5.12, we find that the phase does not depend on α, β, or γ; it is constant on cosets of $C_2^\perp$. We can then invoke Corollary 2.5.14.

$$\begin{aligned}\mathrm{tr}\langle(c_1+\alpha)*(c_2+\beta),(c_3+\gamma)\rangle &= \mathrm{tr}\langle c_1*c_2,c_3\rangle &(2.118)\\ &= -\mathrm{tr}\langle m_1^{(B)}*m_2^{(B)},m_3^{(B)}\rangle. &(2.119)\end{aligned}$$

The result follows after using the field characteristic. □

Proof of Theorem 2.5.1

We are now ready to assembly the previous results.

Theorem 2.5.1. *For each $q = 2^l$ with $l \geq 3$, there exists a CSS qudit code (punctured quantum Reed-Solomon code) with parameters*

$$\left[\left[\frac{3q}{4}, \frac{q}{4}, \left\lfloor \frac{q}{3} \right\rfloor - \frac{q}{4} + 1\right]\right]_q, \tag{2.97}$$

such that $\mathsf{CCZ}_q^{\otimes 3q/4}$ acts as logical $\overline{\mathsf{CCZ}}_q^{\otimes q/4}$ on a basis of logical qudits.

Proof. The code is as defined in Definition 2.5.9 with

$$k = m = \lfloor q/3 \rfloor,$$
$$s = q/4.$$

These choices satisfy $q/3 \geq k \geq m \geq s > 0$ and $2k \leq q - m$ when $q = 2^l$ with $l \geq 2$. The code parameters are proven in Proposition 2.5.11 and the logical CCZ is proven in Proposition 2.5.15. □

Distillation of qubit $|\mathsf{CCZ}\rangle$

Having established Theorem 2.5.1, we will use this to construct extremely efficient magic state distillation for qubits. The magic state we would like to distill is $|\mathsf{CCZ}\rangle = \mathsf{CCZ}\,|+\rangle\,|+\rangle\,|+\rangle$. As in the previous section, let $q = 2^l$. Fix a self-dual basis $N = \{\alpha_i\}_{i\in[l]}$ for $\mathbb{F}_q$ over $\mathbb{F}_2$.

Qubits as $\mathbb{F}_q$-qudits

Our codes have only transversal $\mathsf{CCZ}^{(q)}$, so we will need a means to implement the operation using $|\mathsf{CCZ}\rangle$ and to convert the final output states to $|\mathsf{CCZ}\rangle$.

Define the states

$$|+\rangle_q = \frac{1}{\sqrt{q}} \sum_{x\in\mathbb{F}_q} |x\rangle_q\,, \qquad \left|\mathsf{CCZ}^{(q)}\right\rangle_q = \mathsf{CCZ}^{(q)}\,|+\rangle_q\,|+\rangle_q\,|+\rangle_q\,. \tag{2.120}$$

The conversion of $\left|\mathsf{CCZ}^{(q)}\right\rangle_q$ to $|\mathsf{CCZ}\rangle$ is straightforward in the appropriate basis. We first begin by constructing a circuit to change the basis of the field extension. This will give us greater flexibility when implementing operations later.

Proposition 2.5.16 (Change of basis)**.** *For two bases $N = \{\alpha_i\}_{i\in[l]}$ and $M = \{\beta_i\}_{i\in[l]}$ for $\mathbb{F}_q$ over $\mathbb{F}_2$, there is a unitary that maps*

$$|x\rangle^{(N)} \mapsto |x\rangle^{(M)} \tag{2.121}$$

for all $x \in \mathbb{F}_q$ using depth $O(l)$, $O(l^2)$ CNOT gates, and l input ancilla qubits initialized to $|0\rangle$.

Proof. N and M are bases for $\mathbb{F}_q$ over $\mathbb{F}_2$, so there exists an invertible matrix $A \in \mathbb{F}_2^{l\times l}$ such that for $x \in \mathbb{F}_q$,

$$x_j^{(M)} = \sum_{i\in[l]} A_{ij} x_i^{(N)}.$$

The initial state is $|x\rangle^{(N)} |0\rangle^{\otimes l}$. For every non-zero entry $(i, j) \in [l] \times [l]$ of A perform a CNOT controlled on the j-th qubit of the first register and targeted on the i-th qubit of the second register. This maps $|x\rangle |0\rangle^{\otimes l} \mapsto |x\rangle |y\rangle$. Then, for every non-zero entry $(i, j) \in [l] \times [l]$ of A^{-1} perform a CNOT controlled on the j-th qubit of the second register and targeted on the i-th qubit of the first register. This maps $|x\rangle |y\rangle \mapsto \left|x + A^{-1}y\right\rangle |y\rangle = |0\rangle^{\otimes l} |y\rangle$. □

Lemma 2.5.17. *Let $M = \{\alpha_i\}_{i\in[l]}$ be a basis for $\mathbb{F}_q$ over $\mathbb{F}_2$ such that $\alpha_1 = 1$. Then, there exists a circuit to convert $\left|\mathsf{CCZ}^{(q)}\right\rangle_q^{(M)}$ to $|\mathsf{CCZ}\rangle$ using $3(l-1)$ measurements and 3 classically controlled CZ and Z gates.*

Proof. Since $\{\alpha_i\}_{i\in[l]}$ form a basis, there exists a j such that $\mathrm{tr}(\alpha_j) = 1$ (if l is odd then $\alpha_1 = 1$ works). Using $\mathbb{F}_2$-linearity of the trace, for $x, y, z \in \mathbb{F}_q$, we can separate the product into

$$\mathrm{tr}(xyz) = x_1 y_1 z_j + \mathrm{tr}(\ldots), \tag{2.122}$$

where the omitted terms on the right are at most quadratic in x_1,y_1, and z_j. Then, starting with the state

$$\left|\mathsf{CCZ}^{(q)}\right\rangle_q^{(M)} = \sum_{x,y,z\in\mathbb{F}_q} (-1)^{\mathrm{tr}(xyz)} |x\rangle^{(M)} |y\rangle^{(M)} |z\rangle^{(M)}, \tag{2.123}$$

the computational basis measurement of the last $l-1$ qubits in each of the first two registers and of the qubits $[l]\backslash\{j\}$ in the third register gives the measurement outcomes $\{x_i\}_{i=2}^l$, $\{y_i\}_{i=2}^l$, and $\{z_i\}_{i\neq j}$. The post measurement state is

$$\frac{1}{2^{3/2}} \sum_{x_1,y_1,z_j\in\mathbb{F}_2} (-1)^{x_1 y_1 z_j + p(x_1,y_1,z_j)} |x_1\rangle |y_1\rangle \left|z_j\right\rangle, \tag{2.124}$$

where $p(x_1, y_1, z_j)$ is a polynomial over $\mathbb{F}_2$ that is at most quadratic in x_1,y_1, and z_j. The coefficients can be classically computed from the measurement outcomes, so the phases can be fixed by application of CZ and Z such that the final state is $|\mathsf{CCZ}\rangle$. □

We can also implement $\mathsf{CCZ}^{(q)}$ using qubit CCZ gates.

Lemma 2.5.18. $\mathsf{CCZ}^{(q)}$ *has an implementation using at most* l^3 CCZ *gates.*

Proof. For $x, y, z \in \mathbb{F}_q$ consider

$$\mathsf{CCZ}^{(q)} |x\rangle^{(N)} |y\rangle^{(N)} |z\rangle^{(N)} = (-1)^{\mathrm{tr}(xyz)} |x\rangle^{(N)} |y\rangle^{(N)} |z\rangle^{(N)} . \tag{2.125}$$

$\mathrm{tr}(xyz)$ is a degree-3 polynomial in the coefficients of the basis expansion. Let $N = \{\alpha_i\}_{i\in[l]}$. Then we may expand $\mathrm{tr}(xyz)$ as

$$\mathrm{tr}(xyz) = \sum_{i,j,k\in[l]} x_i^{(N)} y_j^{(N)} z_k^{(N)} \,\mathrm{tr}\big(\alpha_i\alpha_j\alpha_k\big). \tag{2.126}$$

For every triple (i, j, k) for which $\mathrm{tr}\big(\alpha_i\alpha_j\alpha_k\big)$ is non-zero, we perform CCZ between the i-th qubit of the x register, the j-th qubit of the y register, and the k-th qubits of the z register. □

Single-step state distillation

We begin by giving the construction for a single round of magic state distillation. Let N be a self-dual basis, and M be a polynomial basis. Fix a field size $q = 2^l$ and a quantum code C with parameters $[[n, k, d]]_q$ from Theorem 2.5.1. Let $\mathcal{E}$ be the encoding map for C.

For $d, n \in \mathbb{N}$, define $\mathcal{S}_d^n = \{x \subseteq [n] \mid |x| = d\} \subseteq P([n])$ to be all subsets of $[n]$ of size d.

Proposition 2.5.19. *If C is a t-error correcting code, then the output of Algorithm 2 is* $|\mathsf{CCZ}\rangle^{\otimes k}$ *if the input is* $W_{t+1}^n \bullet W_1^{l^3} \bullet W_1^3$ *deviated from* $|CCZ\rangle^{\otimes nl^3}$.

Proof. In the self-dual basis N, qudit CNOT can be applied using transversal qubit CNOT. The procedure first prepares the logical code state $\mathcal{E}\left(\left(|+\rangle_q^{(N)}\right)^{\otimes k}\right)^{\otimes 3}$. For each of the n positions, we consume l^3 $|CCZ\rangle$ states to perform $\mathsf{CCZ}^{(q)}$ using Lemma 2.5.18. If any one of these states is faulty, then an error is incurred on the position. However, the code is t-error correcting, so as long as at most

Algorithm 2 Magic state distillation algorithm

Input: $n \cdot l^3$ $|\mathsf{CCZ}\rangle$, $3n \cdot l$ $|+\rangle$, and $3n \cdot l$ $|0\rangle$
Data: $3n$ qubit register A
Output: k $|\mathsf{CCZ}\rangle$

1: Initialize register A with input state $A \leftarrow \left(|+\rangle_q^{(N)}\right)^{\otimes 3n} = |+\rangle^{\otimes 3nl}$
2: $\sigma_1 \leftarrow$ measurements of checks (sequentially) of the 3 blocks of register A
3: $U_{corr,1} \leftarrow$ operator $U_{corr,1}$ such that $U_{corr,1}$ corrects σ_1
4: Apply $U_{corr,1}$ to register A
5: Consume $|\mathsf{CCZ}\rangle^{\otimes n \cdot l^3}$ to apply $\left(\mathsf{CCZ}^{(q)}\right)^{\otimes n}$ to register A
6: $\sigma_2 \leftarrow$ measurements of checks of C on the 3 blocks of register A
7: $U_{corr,2} \leftarrow$ operator $U_{corr,2}$ such that $U_{corr,2}$ corrects σ_2
8: Apply $U_{corr,2}$ to register A
9: Apply $\mathcal{E}^{-1}$ to register A
10: Use $3n \cdot l$ $|0\rangle$ as space to apply change of basis $|\mathsf{CCZ}\rangle_q^{(N)} \to |\mathsf{CCZ}\rangle_q^{(M)}$ to A
11: Apply $|\mathsf{CCZ}\rangle_q^{(M)}$ to $|\mathsf{CCZ}\rangle$ conversion to A
12: Output A

t positions are faulty, the state is successfully corrected to the encoded logical $\mathcal{E}^{\otimes 3}\left(\left(\left|\mathsf{CCZ}^{(q)}\right\rangle_q^{(N)}\right)^{\otimes k}\right)$. We apply $(\mathcal{E}^{-1})^{\otimes 3}$ to get $\left(\left|\mathsf{CCZ}^{(q)}\right\rangle_q^{(N)}\right)^{\otimes k}$ We then can then use the $|0\rangle$ ancillas to convert this state to $\left(\left|\mathsf{CCZ}^{(q)}\right\rangle_q^{(M)}\right)^{\otimes k}$ (Proposition 2.5.16). This basis allows us to use Lemma 2.5.17 to obtain $|\mathsf{CCZ}\rangle^{\otimes k}$. □

Remark 2.5.20. *This is a magic state distillation method for qubits for which the magic state distillation exponent*

$$\gamma \equiv \frac{\log(n/k)}{\log d} = \frac{\log\left(3\log_2^3 q\right)}{\log(q/12)} = O\left(\frac{\log l}{l}\right) \tag{2.127}$$

can be made arbitrarily small.

Lemma 2.5.21. *Fix two functions $f, g : \mathbb{N} \mapsto \mathbb{R}$ such that and consider the family of functions defined as*

$$W_i(x) = (f(i)W_{i-1}(x))^{g(i)},$$
$$W_0(x) = x$$

if f and g satisfy the asymptotic bound

$$\frac{\log(f(i))}{\prod_{j=1}^{i-1} g(j)} = O_{i\to\infty}(i^{-2}) \tag{2.128}$$

then there exists a constant $\beta > 0$ such that

$$W_i(x) \leq (\beta x)^{\prod_{j=1}^{i} g(j)} \tag{2.129}$$

for $x \in [0, 1)$.

Proof. For $L \in \mathbb{N}$, define $h(i) = \prod_{j=i}^{L} g(j)$, so that $W_L(x)$ has the closed form

$$W_L(x) = \left(\prod_{i=1}^{L} (f(i))^{h(i)} \right) x^{h(1)}. \tag{2.130}$$

$$\tag{2.131}$$

Define $\bar{h}(i) = h(1)/h(i) = \prod_{j=1}^{i-1} g(j)$. Then, we bound

$$W_L(x) = x^{h(1)} \exp\left[h(1) \sum_{i=1}^{L} \frac{\log(f(i))}{\bar{h}(i)} \right] \tag{2.132}$$

$$\leq x^{h(1)} \exp\left[h(1) \sum_{i=1}^{\infty} \frac{\log(f(i))}{\bar{h}(i)} \right] \tag{2.133}$$

$$\leq x^{h(1)} \exp\left[h(1)(\text{const.}) \right] \tag{2.134}$$

$$= (x \cdot (\text{const.}))^{h(L)}, \tag{2.135}$$

where the last line follows for a constant independent of L due to the assumption $\frac{\log(f(i))}{\prod_{j=1}^{i-1} g(j)} = O_{i \to \infty}(i^{-2})$. □

Proof of Theorem 2.4.8

Theorem 2.4.8 (State distillation procedure for magic states)**.** *There exists constants $\epsilon_{*,Mdistill} \in (0, 1)$, $\beta_{\text{Mdistill}} > 0$ and a family indexed by $l \in \mathbb{N}$ of $(M_l, K_l, \mathcal{A}_l)$-state distillation procedure for $|\mathsf{CCZ}\rangle$ where $\mathcal{W}(\mathcal{A}_l; x) \leq (\beta_{\text{Mdistill}} x)^{(l+1)!}$ on $x \in [0, \epsilon_{*,Mdistill}]$, $\frac{K_l}{M_l} = \Omega\left(c^l\right)$, and $M_l = e^{\Theta(l \log l)}$ for some absolute constant $c > 0$.*

Furthermore, the procedure satisfies:

- *The quantum depth is $O(l^4 (\log l)^3)$, and the width is $O(M_l)$.*
- *The classical computation has total depth $O(l^4)$.*
- *$O(M_l)$ perfect $|+\rangle$ and $|0\rangle$ input states are required*

- *$O(M_l)$ classically-controlled* CZ *gates are required*
- *Otherwise, only* CNOT*, Z/X basis measurement, and classically-controlled Pauli gates are used.*

Proof. The proof proceeds nearly identically to the case of distillation of stabilizer states. For each extension field $\mathbb{F}_q$, we pick a self-dual basis over $\mathbb{F}_2$. We will pick a sequence of codes from Theorem 2.5.1 with $q_i = 64 \cdot 2^{\lceil \log_2 i \rceil}$, so that $64i \le q_i \le 128i$. The i-th code from the sequence has $n_i = 3q_i/4 \le 96i$, $k_i \ge 16i$, and is a $t_i = \lfloor \frac{d_i - 1}{2} \rfloor \ge i$-error correcting code. We parallel the argument from Lemma 2.4.7: Let $\tilde{n}_i = n_i(\log_2(q_i))^3 = O(i(\log i)^3)$.

Let $M_l := \prod_{\ell=1}^{l} \tilde{n}_\ell$, and $K_l := \prod_{\ell=1}^{l} k_\ell$.

We start with a set of $|\mathsf{CCZ}\rangle$ states indexed by $[\tilde{n}_l] \times \cdots \times [\tilde{n}_1] \simeq [M_l]$. Iterating from $\ell = 1$ to $\ell = l$, for each $I \in [\tilde{n}_l] \times \cdots \times [\tilde{n}_{\ell+1}]$ and $J \in [k_{\ell-1}] \times \cdots \times [k_1]$ we apply the state distillation procedure Algorithm 2 on the set of states $\{I\} \times [\tilde{n}_\ell] \times \{J\}$ to get a new set of states with labels $\{I\} \times [k_\ell] \times \{J\}$ (Consuming input ancilla states and applying CZ). Let $\mathcal{S}_d^n$ be the family of all subsets of $[n]$ of size d. At step ℓ, Proposition 2.5.19 gives that the output is $|\mathsf{CCZ}\rangle^{\otimes k_\ell}$ if the input is $\mathcal{S}_2^{\tilde{n}_\ell} \bullet \mathcal{S}_1^3$-deviated from $|\mathsf{CCZ}\rangle^{\otimes \tilde{n}_\ell}$. We use the fact that, by construction, no two outputs from a single application of the state distillation procedure are used together in a following state distillation procedure: If the output of a state distillation in step ℓ is not $|\mathsf{CCZ}\rangle^{\otimes k_\ell}$, then the input is not $\mathcal{S}_{t_\ell+1}^{\tilde{n}_\ell}$-deviated from $|\mathsf{CCZ}\rangle^{\otimes \tilde{n}_\ell}$. Since each input is from separate state distillation procedures, $t+1$ distinct state distillation procedures at step $\ell - 1$ must have had inputs that are not $\mathcal{S}_{t_{\ell-1}+1}^{\tilde{n}_{\ell-1}}$-deviated from $|\mathsf{CCZ}\rangle^{\otimes \tilde{n}_{\ell-1}}$.

Inducting from $\ell = 1$, the output of the final iteration is $|\mathsf{CCZ}\rangle^{\otimes K_l}$ if the input is $\mathcal{A}_l := \mathcal{S}_{t_l+1}^{\tilde{n}_l} \bullet \cdots \bullet \mathcal{S}_{t_1+1}^{\tilde{n}_1} \bullet \mathcal{S}_1^3$-deviated from $|\mathsf{CCZ}\rangle^{\otimes M_l}$.

Let $S_i(x) = \left(96 \cdot 7^3 i \log(i)^3 x\right)^{i+1}$ which satisfies $\mathcal{W}(\mathcal{S}_{t_i+1}^{\tilde{n}_i};\, x) \le (\tilde{n}_i x)^{t_i+1} \le S_i(x)$ when $x \in [0, 1/\tilde{n}_i]$. When $\mathcal{W}(\mathcal{A}_{l-1};\, x) \le 1/\tilde{n}_l$, we can bound $\mathcal{W}(\mathcal{A}_l;\, x)$ as

$$\mathcal{W}(\mathcal{A}_l;\, x) \le S_l \circ S_{l-1} \circ \cdots \circ S_2 \circ S_1(3x). \tag{2.136}$$

This recursion satisfies the preconditions of Lemma 2.5.21,[45] so there exists a constant β_{Mdistill} (independent of l) such that

$$\mathcal{W}(\mathcal{A}_l;\, x) \le (\beta_{\text{Mdistill}} x)^{\prod_{i=1}^{l}(i+1)} = (\beta_{\text{Mdistill}} x)^{(l+1)!}. \tag{2.137}$$

[45] Lemma 2.5.21 gives a superexponential error suppression at every step of the recursion, so we do not need to be worried about the polynomially small precondition $\mathcal{W}(\mathcal{A}_{l-1};\, x) \le 1/\tilde{n}_l$. It suffices to take β_{Mdistill} to be $1/2$ the constant promised by Lemma 2.5.21.

The three operations with depth $\omega(1)$ are the application of $\mathcal{E}$, $\mathcal{E}^{-1}$, and $\mathsf{CCZ}^{(q)}$. Using a self-dual basis, in step i, the encoding and unencoding unitiaries have depth $O(q_i^3) = O(i^3)$ while the $\mathsf{CCZ}^{(q)}$ has depth $O\left((\log(q_i))^3\right) = O\left((\log i)^3\right)$, so the overall depth is at most $O(l^4(\log l)^3)$. A number of ancilla qubits $O(M_l)$ is needed to sequentially measure the syndrome, so the total space required is $O(M_l)$ qubits.

Finally, $M_l \leq \prod_{i=1}^{l} \alpha i(\log i)^3 = O\left(\alpha^l l! \cdot l(\log l)^3\right)$ for some $\alpha > 0$ and $K_l \geq \prod_{i=1}^{l} 16i = 16^l l!$, so for some $c > 0$,

$$\frac{K_l}{M_l} \geq \frac{16^l l!}{\alpha^l l! \cdot l(\log l)^3} = \Omega\left(c^l\right). \tag{2.138}$$

Classical depth. We now describe a time-efficient decoder for the PQRS code, which implies to the claimed classical depth. Recall that the X checks correspond to the code $C_2^{\perp}$. According to Lemma 2.5.10, C_2 consists of the evaluations of $\mathbb{F}_q[x]_{<q-m}$ on a set M, where in the proof of Theorem 2.5.1 we set $m = \lfloor q/3 \rfloor$ and $M \subset \mathbb{F}_q$ is an evaluation set of size $|M| = 3q/4$. Hence, we can apply standard RS decoders. Here we use the Berlekamp-Welch algorithm (see Theorem 12.1.6 in [GRS22]) which decodes C_2 up to error weight $\lfloor (3q/4-(q-m)+1)/2 \rfloor = \lfloor q/12 \rfloor$ in time $O(q^3)$ (if no codeword is found the decoder outputs FAIL). Similarly, the Z checks correspond to the code $C_1^{\perp}$, where C_1 is $\mathbb{F}_q[x]_{<k}$ evaluated on M, and $k = \lfloor q/3 \rfloor$ is chosen in Theorem 2.5.1's proof. So X errors can be corrected with the Berlekamp-Welch algorithm up to error weight $\lfloor (3q/4 - k + 1)/2 \rfloor = \lfloor 5q/24 \rfloor$.[46] It follows that the total classical depth in the l-level distillation procedure is $\sum_{i=1}^{l} O(q_i^3) = \sum_{i=1}^{l} i^3 = O(l^4)$. □

[46] Alternatively, we can also use the standard twirling argument in the MSD literature [BK05], applying the Clifford $\mathsf{CCZ}^{(q)}(X^{(q)}(a_1)X^{(q)}(a_2)X^{(q)}(a_3))(\mathsf{CCZ}^{(q)})^{\dagger}$ for randomly chosen $a \in \mathbb{F}_q^3$, to restrict our decoding problem to only Z-type errors.

Chapter 3

HIERARCHICAL MEMORIES: SIMULATING QUANTUM LDPC CODES WITH LOCAL GATES

[PKP25] Christopher A Pattison, Anirudh Krishna, and John Preskill. "Hierarchical memories: Simulating quantum LDPC codes with local gates". In: *Quantum* 9 (2025). C.A.P. conceived the project and participated in developing the key ideas and writing the manuscript., p. 1728. DOI: https://doi.org/10.22331/q-2025-05-05-1728. arXiv: 2303.04798 [quant-ph]. URL: https://arxiv.org/abs/2303.04798.

3.1 Introduction

Quantum error-correcting codes encode quantum information in entangled states over many qubits. They are defined by a set of operators called stabilizer generators. Errors can accumulate in the state due to imperfect control and interactions with the environment. Stabilizer generators can be measured using syndrome-extraction circuits; the outcome of these measurements are called syndromes, classical information used to infer corrections to these errors. To minimize the probability of corrupting information beyond recovery, it is imperative to minimize the points of failure in the syndrome-extraction circuit. This can be realized by restricting the number of gates that each qubit interacts with and minimizing the total space-time volume of this circuit. The extent to which this can be done depends on the choice of error-correcting code and physical constraints.

Syndrome-extraction circuits are the workhorse of quantum memories, devices that can reliably store qubits for some fixed duration. In this paper, we are concerned with designing memories that can encode a growing number of qubits and simultaneously have a low probability of failure.[1] We focus on their design when qubits are embedded in a two-dimensional lattice and gates are subject to constraints on geometric locality.

Quantum low-density parity-check (LDPC) codes are natural candidates for constructing quantum memories. A quantum LDPC code refers to a family $\{\mathcal{Q}_t\}_t$ of $\{[\![n(t), k(t), d(t), \Delta_q, \Delta_g]\!]\}$ codes. This notation means that the t^{th} element in the family uses $n(t)$ data qubits to encode $k(t)$ logical qubits and has distance $d(t)$, i.e. it is robust to $\lfloor (d(t)-1)/2 \rfloor$ Pauli errors. We assume that for all t, $n(t) > n(t-1)$. In

[1] We leave fault-tolerant *computation* for future work.

what follows, we wish to focus on the dependence of k and d as functions of n. In this case, we implicitly parameterize the family using the size n of the code, i.e. we use the notation $\{Q_n\}_n$, and we say that we are working with a $[\![n, k(n), d(n), \Delta_q, \Delta_g]\!]$ code family.

A quantum LDPC code is one where, for all codes in the code family, every stabilizer generator only involves at most a constant number Δ_g of qubits, and each qubit is supported within at most a constant number Δ_q of stabilizer generators. Such codes can encode a number of qubits that increases with the code size; simultaneously, the probability of *any* error on the encoded level is suppressed exponentially in the distance $d(n)$.

Furthermore, the syndrome-extraction circuit C_n can be efficient as measured by two figures of merit. The *depth* of the syndrome-extraction circuit is the number of timesteps $\mathcal{T}(C_n)$ it takes to implement. The *width* of the syndrome-extraction circuit is the total number of qubits $\mathcal{W}(C_n)$ it uses (including ancilla qubits in addition to data qubits). The size or volume of the circuit is the product of the depth and the width. Building on a result by Kovalev and Pryadko [KP13a], Gottesman [Got14] constructed fault-tolerant syndrome-extraction circuits that have volume which is a constant times the volume of the noise-free syndrome-extraction circuit — there exists a threshold q such that if gates fail with fixed probability $p < q$, the probability of the circuit failing falls exponentially in the distance $d(n)$.

However, realizing this architecture in a 2-dimensional layout is challenging. It requires high-fidelity gates acting on qubits that may be far apart. Some architectures might not support such interactions.

It is known that geometric locality severely constrains quantum error-correcting codes in 2 and 3 (Euclidean) dimensions. The most famous codes that are implemented using only geometrically-local gates are surface codes [Kit03; BK98] and color codes [BM06; KB15]. Seminal results by Bravyi and Terhal [BT09], and later by Bravyi, Poulin and Terhal [BPT10] showed that these codes are optimal for quantum LDPC codes defined using geometrically-local stabilizers. Subsequently, it was shown that to implement LDPC codes where the parameters k and d are both strictly better than the surface code, we require a growing amount of long-range connectivity [BK21a; BK21b].

When restricted to using only nearest-neighbor gates in 2 dimensions, Delfosse *et al*. [DBT21] proved the following tradeoff for syndrome-extraction circuits for

constant-rate LDPC codes: [2]

$$\mathcal{T}(C_n) = \Omega\left(\frac{n}{\sqrt{\mathcal{W}(C_n)}}\right), \tag{3.1}$$

where $\mathcal{T}(C_n)$ is the depth of the syndrome-extraction circuit and $\mathcal{W}(C_n)$ is the total number of qubits, data and ancilla, used in the circuit [3]. In words, this shows that given only nearest-neighbor gates to build a syndrome-extraction circuit for constant-rate LDPC codes, we can choose to minimize either the depth or the width of C_n, but cannot do both.

This sets the stage for presenting the main questions we address in this paper: does the family of circuits saturating Equation (3.1) still have a threshold? If not, how do we modify the code and associated circuit to achieve a threshold as efficiently as possible? How do we construct the most efficient syndrome-extraction circuits given access to gates whose range is more than merely nearest neighbor? Can we improve on the bound in Equation (3.1)?

Our contributions

This paper is centered around the theme of implementing efficient quantum memories. Our main result is that our proposal, called a hierarchical code, has a threshold and that it achieves asymptotically better error suppression than the surface code. As it brings together a few different ideas, we present a short summary of each section and how to navigate the paper. Although these results build on each other, our presentation is modular — readers ought to be able to proceed to their section of choice after reading this overview and Section 3.2 where we define all the concepts required to formally state our results. (The statements of the main theorems of each section are only presented informally below.)

Section 3.3 : Permutation routing on graphs Connectivity beyond nearest-neighbor interactions is being explored in many architectures. There is evidence that some architectures can support gates of range R where R can be large [Leu+19; Per+21]. Motivated by these developments, we ask: given nearest-neighbor Clifford gates and SWAP gates of range R, can we reduce the depth of the syndrome-extraction

[2] The bound applies to classes of codes that are called *locally expanding*. The exact definition of locally-expanding codes is not relevant; the interested reader is pointed to the paper by Delfosse *et al.* [DBT21]. For our purposes, it includes some important classes of quantum LDPC codes such as hypergraph product codes [TZ14] and good quantum LDPC codes [BE21; PK21a; LZ22b; LH22].

[3] For an explanation of $O(\cdot)$, $\Theta(\cdot)$ and $\Omega(\cdot)$ notation, please refer to Appendix 3.9.

circuit for constant-rate LDPC codes? To this end, we will permute qubits to bring them within range to apply an entangling gate. This is expressed as a *permutation routing*, a task on a graph $G = (V, E)$ specified by a permutation $\alpha : V \to V$. In this task, two vertices labeled u and v connected by an edge (u, v) are allowed to exchange labels within each step. The objective is to ensure that all labels match destinations $\alpha(u)$ while minimizing the total time required. Permuting vertices in parallel is non-trivial—the paths along which one permutes different pairs can overlap and thereby require more time. Section 3.3 reviews a permutation routing algorithm due to Annexstein and Baumslag [AB90]. This algorithm yields a permutation routing on a product of two graphs given permutation routings on each of the input graphs. In Section 3.3, we build on this algorithm to permute vertices on an $L \times L$ lattice where two vertices separated by a distance R are connected by an edge using a sparse subgraph. The main technical result of this section is the following existence result.

Theorem 3.1.1 (Permutation routing)**.** *For R even, there is an efficient construction of a degree-12 graph $G = (V, E)$ whose vertex set V is identified with an $L \times L$ lattice with edges of length at most R. Given a permutation $\alpha : V \to V$, a permutation routing implementing α can be performed in depth $3L/R + O(\log^2 R)$.*

While it is itself not the main result of our paper, it will be used in service of proving Theorem 3.1.2 which demonstrates the existence of efficient syndrome-extraction circuits given SWAP gates of range up to R (but not all gates need to have length equal to R). This section is entirely technical and only discusses graph properties and permutation routings.

Section 3.4 and Section 3.5: Hierarchical codes & the bilayer architecture Given access to only nearest-neighbor gates, Delfosse *et al*. present some evidence *against* the existence of a threshold if one were to permute qubits to bring them within range to perform a CNOT (see Figure 2 of [DBT21]). In particular, in the setting where $\mathcal{W}(C_n) = \Theta(n)$ and $\mathcal{T}(C_n) = \Theta(\sqrt{n})$, it appears too many errors accumulate before we can complete executing the syndrome-extraction circuit.

We circumvent this problem using code concatenation. We concatenate a constant-rate $[\![n, k, d, \Delta_q, \Delta_g]\!]$ LDPC code $\{\mathcal{Q}_n\}$ with a $[\![d_\ell^2, 1, d_\ell]\!]$ rotated surface code $\mathcal{RS}_\ell$ to obtain the *hierarchical code* $\{\mathcal{H}_N\}$ with parameters denoted $[\![N, K, D]\!]$. This means that each qubit of the syndrome-extraction circuit for the LDPC code $\mathcal{Q}_n$, henceforth referred to as the "outer code", is itself the logical qubit of a rotated surface code $\mathcal{RS}_\ell$, which we refer to as the "inner code" or sometimes as a "tile." As a rotated surface

code can suppress errors exponentially in d_ℓ, we can suppress errors long enough to complete syndrome measurements of the outer quantum LDPC code using relatively small inner codes. The lattice length d_ℓ of the inner code only scales logarithmically in the size of the outer LDPC code, i.e. $d_\ell = \Theta(\log(n))$. Here ℓ indexes the qubits in the rotated surface code, $\ell^2 = 2d_\ell^2 - 1$. Section 3.4 is dedicated to the construction of syndrome-extraction circuits $C_N^{\mathcal{H}}$ corresponding to $\mathcal{H}_N$. The hierarchical code family $\{\mathcal{H}_N\}$ is not LDPC: The stabilizer generators for the outer code act on a number of physical qubits that scales with the size d_ℓ of the inner code. However, local operations are sufficient to implement the corresponding syndrome-extraction circuit $C_N^{\mathcal{H}}$. The main result of this section is summarized in the following theorem.

Theorem 3.1.2. *The $[\![N, K, D]\!]$ hierarchical code $\mathcal{H}_N$ is constructed by concatenating an outer code, a constant-rate $[\![n, k, d, \Delta_q, \Delta_g]\!]$ quantum LDPC code $\mathcal{Q}_n$, and an inner code, a rotated surface code $\mathcal{RS}_\ell$ where $d_\ell = \Theta(\log(n))$. Let $\rho > 0$ and $\delta \geq 1/2$, such that $k = \rho \cdot n$ and $d = \Theta(n^\delta)$. The code $\mathcal{H}_N$ has parameters*

$$K(N) = \Omega\left(\frac{N}{\log^2(N)}\right), \qquad D(N) = \Omega\left(\frac{N^\delta}{\log^{2\delta-1}[N/\log(N)]}\right).$$

There exists an explicit and efficient construction of an associated family of syndrome-extraction circuits $C_N^{\mathcal{H}}$ constructed using only local Clifford operations and SWAP *gates of range R such that*

$$\mathcal{W}(C_N^{\mathcal{H}}) = O(N), \qquad \mathcal{T}(C_N^{\mathcal{H}}) = O\left(\frac{\sqrt{N}}{R}\right).$$

Our construction works for all values of $\delta > 0$; we choose $\delta \geq 1/2$ to make theorem statements simpler. Before describing how the circuit $C_N^{\mathcal{H}}$ is constructed, we motivate why it is interesting—it has a threshold.

We work in a model where errors occur in a stochastic manner. We declare a logical failure if *any* of the K encoded qubits fail. More generally, we declare failure if any logical error occurs on the code space. The main result of Section 3.5 is the following theorem.

Theorem 3.1.3 (Informal)**.** *Consider the $[\![N, K, D]\!]$ family of hierarchical codes $\mathcal{H}_N$ and the associated family of syndrome-extraction circuits $C_N^{\mathcal{H}}$. Suppose the outer code $\mathcal{Q}_n$ has constant rate $k = \rho n$ and distance $d(n) = \Theta(n^\delta)$. If we repeat the syndrome-extraction circuit $C_N^{\mathcal{H}}$ for $d(n)$ rounds, then there exists a threshold $q \in (0, 1]$ corresponding to $C_N^{\mathcal{H}}$ such that, if each gate fails with fixed probability*

$0 < p < q$, then the probability of logical failure under minimum-weight decoding, $p_{\mathcal{H}}(N)$, obeys

$$p_{\mathcal{H}}(N) \leq \exp\left(-c_{\mathcal{H}} \cdot \frac{N^{\delta}}{\log^{2\delta}(N)}\right) ,$$

for some positive number $c_{\mathcal{H}}$ independent of N.

The theorem is only stated informally here because we have not yet defined the noise model with respect to which this result holds. We will consider a *locally decaying error model* to account for correlated errors that may occur in a circuit. This error model is defined in Section 3.2. Section 3.5 is dedicated to a proof of the existence of a threshold. We build on Gottesman's proof of the existence of a threshold for syndrome-extraction circuits (Theorem 4 of [Got14]). The central idea is the requirement that the probability of failure for a qubit *per round of syndrome extraction*, denoted p_{round}, remains a sufficiently small constant. This is reviewed in Section 3.2. Gottesman's result was based on syndrome-extraction circuits for LDPC codes that have constant depth. As Equation (3.1) highlights, this is not possible when subject to locality constraints. We study the dependence of p_{round} on the circuit depth in Section 3.5. In Section 3.5, we show that $\ell = \Theta(\log(n))$ is sufficient for $C_N^{\mathcal{H}}$ to have a threshold.

As we ask to minimize circuit width $\mathcal{W}(C_N^{\mathcal{H}})$ *and* subject the circuit to locality constraints, we pay a price — in addition to the growing depth, the number of encoded qubits $K(N)$ and distance $D(N)$ are suppressed by polylogarithmic factors in n relative to the outer code $\mathcal{Q}_n$ which has constant rate and distance $d(n) = \Theta(n^{\delta})$. Furthermore, for fixed gate error rates $p \in [0, 1]$, the sub-threshold scaling of the logical error rate $p_{\mathcal{H}}(N)$ of $\{\mathcal{H}_N\}$ is subexponential, but superpolynomial, in the distance $D(N)$; for any positive constants α, β, the logical failure probability $p_{\mathcal{H}}(N)$ vanishes faster than any polynomial function $N^{-\beta}$ but slower than any exponential function $\exp(-\alpha \cdot N)$:

$$\frac{p_{\mathcal{H}}(N)}{N^{-\beta}} \xrightarrow{N\to\infty} 0 , \qquad \frac{p_{\mathcal{H}}(N)}{\exp(-\alpha \cdot N)} \xrightarrow{N\to\infty} \infty .$$

Having motivated why we are interested in $\mathcal{H}_N$, we return to the construction of $C_N^{\mathcal{H}}$. In Section 3.4, we propose a novel bilayer architecture to implement it. We begin the section by presenting the syndrome-extraction circuit C_n for the constant-rate $[\![n, k, d, \Delta_q, \Delta_g]\!]$ LDPC code. Physical qubits are arranged in two parallel layers, each

a lattice of side length approximately $L = \Theta(\sqrt{n})$. To obtain the syndrome-extraction circuit $C_N^{\mathcal{H}}$ for the concatenated code, each of the $\mathcal{W}$ qubits in $C_n^{\mathcal{Q}}$ is replaced by a rotated surface code.

In Section 3.4, we describe how to arrange $\mathcal{W} = \mathcal{W}(C_n^{\mathcal{Q}})$ surface codes $\mathcal{RS}_\ell$ in a bilayer architecture. Each layer now has side length approximately $2L\ell$ qubits to accommodate the tiles. An instance of a single layer is shown in Figure 3.1 (a). We assume access to nearest-neighbor physical Clifford operations and SWAP gates of range R within a layer and Clifford operations between adjacent qubits in different layers. These physical qubits are aggregated into $[\![d_\ell^2, 1, d_\ell]\!]$ codes $\mathcal{RS}_\ell$. See Figure 3.1 (b). There are $2L^2$ tiles in total. Even though we are only implementing a quantum memory, we still need to understand how to perform a limited set of logical operations on tiles to implement the syndrome-extraction circuit for the outer code. The advantage of the bilayer architecture is that it allows for transversal CNOT to implement logical CNOT. We propose a new technique to perform logical SWAP operations between tiles. This yields all required logical Clifford operations between tiles to perform syndrome-extraction for the outer code.

We note that the existence of a threshold does not depend on using the bilayer architecture. For example, tiles can be arranged in a single layer and Clifford gates can be implemented via lattice surgery [Lit19; HFDV12]. For an alternative implementation in the context of measurement-based quantum computation, see [Bom+21]. Although we do not prove it here, it is possible to show that a threshold exists also in this setting using similar techniques.

The circuits $C_N^{\mathcal{H}}$ are constructed such that each lattice position remains connected to a fixed and constant-sized set of other lattice positions for any $R = \omega(1)$. Furthermore, the connectivity does not change dynamically over the course of the circuit. This way, the wiring can be decided ahead of time.

Section 3.6 : Comparisons to surface code Finally, we compare the hierarchical memory $\mathcal{H}_N$ with a simple memory that only uses rotated surface codes. At the outset, it may seem unclear whether the use of extra resources to execute the constant-rate LDPC code's syndrome-extraction circuit can be better spent simply building better surface codes which are ideally suited for 2-dimensional local interactions. We let $\{\mathcal{B}_M\}$ refer to the *basic* encoding where each logical qubit is encoded in a separate surface code; for some distance d_M, we let $\mathcal{B}_M$ be the K-fold product of the surface code, i.e. $\mathcal{B}_M = \mathcal{RS}_{\ell_M}^{\otimes K}$. The index M represents the total number of qubits in this

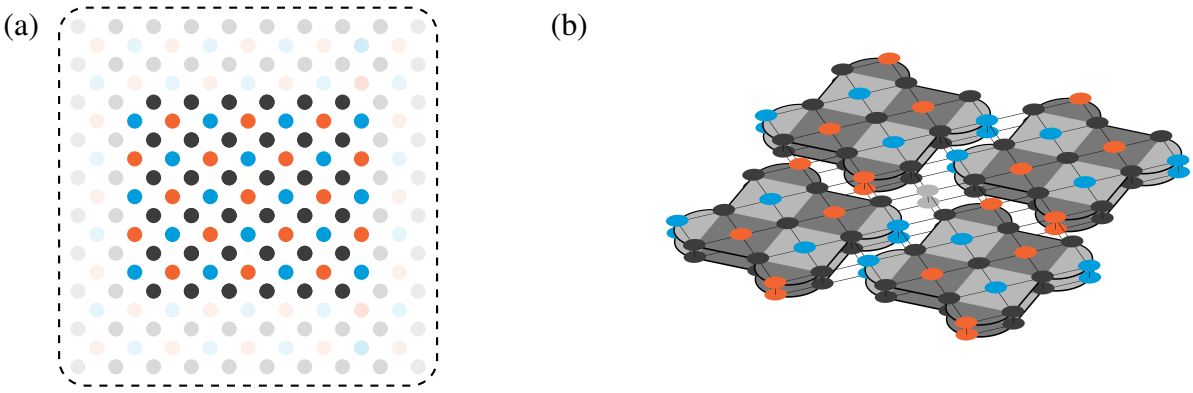


Figure 3.1: The bilayer architecture used to implement the syndrome-extraction circuit $C_N^{\mathcal{H}}$ for the hierarchical code $\mathcal{H}_N$. (a) represents a single layer of the bilayer architecture. Colored dots represent syndrome qubits and gray dots represent data qubits. Transparent dots represent inactive qubits. At any given timestep, the qubits that participate in the circuit are depicted as opaque dots and form a lattice of side length $L\ell$; its location within the larger lattice can shift relative to the second layer. This is used to facilitate logical Clifford operations. (b) represents parallel tiles of distance d_ℓ. Each tile represents an outer qubit of the hierarchical code construction. Light gray dots will be used to facilitate Clifford operations but are not used in the syndrome-extraction circuit for $\mathcal{RS}_\ell$.

encoding, i.e. $M = \Theta(Kd_M^2)$. To contrast $\mathcal{H}_N$ and $\mathcal{B}_M$, we present both an asymptotic comparison as well as numerical estimates based on some conservative assumptions.

Theorem 3.1.4 (Informal)**.** *Let $\mathcal{H}_N$ be a specific $[\![N, K, D]\!]$ hierarchical code family such that the (outer) constant-rate LDPC code $\mathcal{Q}_n$ has distance $d = \Theta(n^\delta)$. Let $\mathcal{B}_M$ be the basic encoding $\mathcal{RS}_{\ell_M}^{\otimes K}$ that encodes K qubits in separate rotated surface codes of distance d_M. Let $C_M^{\mathcal{B}}$ be the corresponding family of syndrome-extraction circuits for $\mathcal{B}_M$. Let $p_{\mathcal{B}}(M)$ denote the logical failure probability under minimum-weight decoding for $\mathcal{B}_M$ where we declare failure if* any *logical qubit fails. Suppose the gate error rate p is below the thresholds for both the basic encoding and the hierarchical code. To achieve $p_{\mathcal{B}}(M) < p_{\mathcal{H}}(N)$, we require*

$$\mathcal{W}(C_M^{\mathcal{B}}) = \Omega\left[\left(\frac{N}{\log(N)}\right)^{1+2\delta}\right], \qquad \mathcal{T}(C_M^{\mathcal{B}}) = \Omega\left[\left(\frac{N}{\log^2(N)}\right)^{\delta}\right].$$

We can compare this with parameters for $C_N^{\mathcal{H}}$ from Theorem 3.1.2. For all $\delta > 0$, the width $\mathcal{W}$ of $C_N^{\mathcal{H}}$ is less than that of $C_M^{\mathcal{B}}$. Furthermore, if the outer code has a single-shot decoder, i.e. if a constant number of applications of $C_N^{\mathcal{H}}$ are sufficient to

achieve a threshold, then the depth $\mathcal{T}$ of $C_N^{\mathcal{H}}$ is also less than that of $C_M^{\mathcal{B}}$. Efficient single-shot decoders are known to exist for constant-rate LDPC codes [LTZ15; FGL18a; FGL18b].

Having said this, it is unclear whether this advantage manifests for practically-relevant code sizes and error rates. To make such a comparison, we use numerical estimates. We choose the size $M = M(N)$ such that the syndrome-extraction circuits for the hierarchical scheme $\mathcal{H}_N$ and the basic encoding $\mathcal{B}_M$ use the same number of physical qubits. Fixing the total number of qubits in this manner, we look for a *crossover point*, the gate error rate q_0 at which the hierarchical code achieves a lower logical failure rate than the basic encoding.

We estimate the circuit-level failure rate using some assumptions about the sub-threshold scaling of the logical failure rate for LDPC codes. We assume the threshold of the surface code is 10^{-2} and the threshold for constant-rate LDPC codes under circuit-level noise is 10^{-3}. Our model takes into consideration how the logical failure rate depends on the depth of the circuit $C_N^{\mathcal{H}}$, and how hook errors could reduce the effective distance. Hook errors are harmful errors that spread from the ancilla qubits to the data qubits during syndrome extraction. These are explained in Section 3.6.

We offer evidence that against circuit-level depolarizing noise, the crossover happens at a gate error rate as high as 5×10^{-3} depending on the choice of outer code family and inner/outer code sizes. See the left-most plot in Figure 3.2. These numbers are merely a proof-of-concept and depend on the aforementioned assumptions which are discussed in Section 3.6.

We arrive at these estimates assuming all gates fail with the same probability. While such an assumption is convenient for proofs, in some architectures, it may be possible to perform SWAP operations with higher fidelity than CNOT or CZ [End+16; Bar+16; Blu+22; Hen+06; Kau+17]. For example, in ion trap and neutral atom trap architectures, SWAP gates can be performed by moving the traps. The mechanism is entirely different than that used to perform other two-qubit gates and, in principle, could have much better fidelity. These considerations are especially important to us as the main source of noise in the hierarchical scheme stems from SWAP gates. We present variations of our numerical estimates when the SWAP gates have better fidelity than the CNOT gates. The middle plot and right-most plot in Figure 3.2 represent estimates for the failure rate when the SWAP gates are $10\times$ and $100\times$ better than entangling gates respectively.

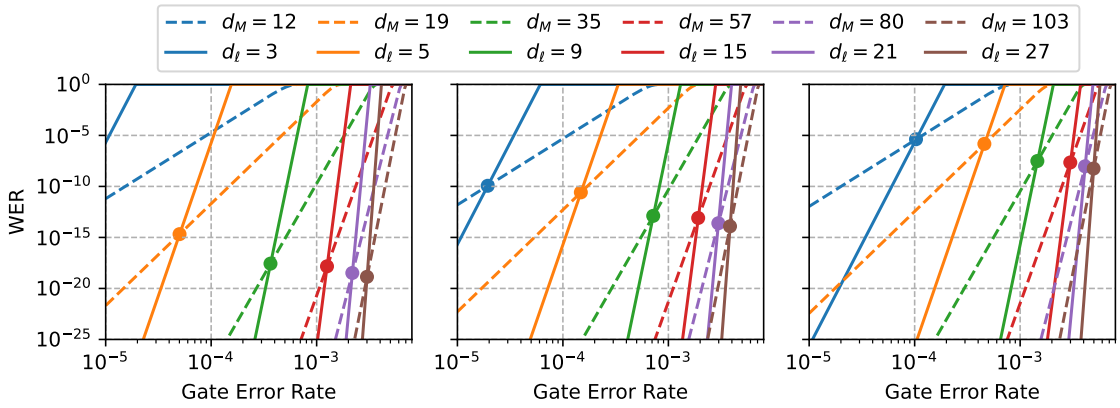


Figure 3.2: Comparing the logical failure rate for the hierarchical memory versus the logical failure rate for the basic encoding. The outer LDPC code has parameters $[\![1\ 116\ 416, 112\ 896, 119]\!]$. Each color represents an inner code of distance d_ℓ. The solid and dashed lines are estimates for the WER for the hierarchical memory and basic encoding respectively. The legend shows the size of the surface codes in each setting. For example, the solid blue line represents a hierarchical code with inner code lattice length $d_\ell = 3$. The dashed blue line represents a basic encoding that uses surface codes of lattice length 12. The three panels correspond to three different assumptions about the error rate in SWAP gates, as described in the text. In the left-most plot, SWAP gates are assumed to fail at the same rate as entangling gates. In contrast, in the middle and right plots, SWAP gates have a fidelity $10\times$ and $100\times$ better than entangling gates respectively.

As mentioned, our estimates are predicated on some assumptions. We re-examine these assumptions in Section 3.6 and propose ways to improve the failure rate for hierarchical codes. We show how we can reduce the effect of hook errors by designing noise-biased qubits. A qubit is said to have a noise bias if X and Y errors are suppressed with respect to Z errors. We can introduce a bias on Level-1 qubits using unbiased Level-0 (physical) qubits. As the inner code is a surface code, we can engineer a bias simply by making the surface code longer in one direction of our choosing. See Figure 3.3 (a). Based on our estimates, we expect this can reduce the size of the code considerably. Figure 3.3 (b) shows the crossover points for the hierarchical code and the basic encoding with the assumption that SWAP gates are $10\times$ better than entangling gates using a much smaller outer code.

Secondly, we believe that decoders for the hierarchical code can take advantage of their concatenated structure. To achieve this, we propose using message-passing decoders between the outer and inner codes. These ideas can be used in soft decoders

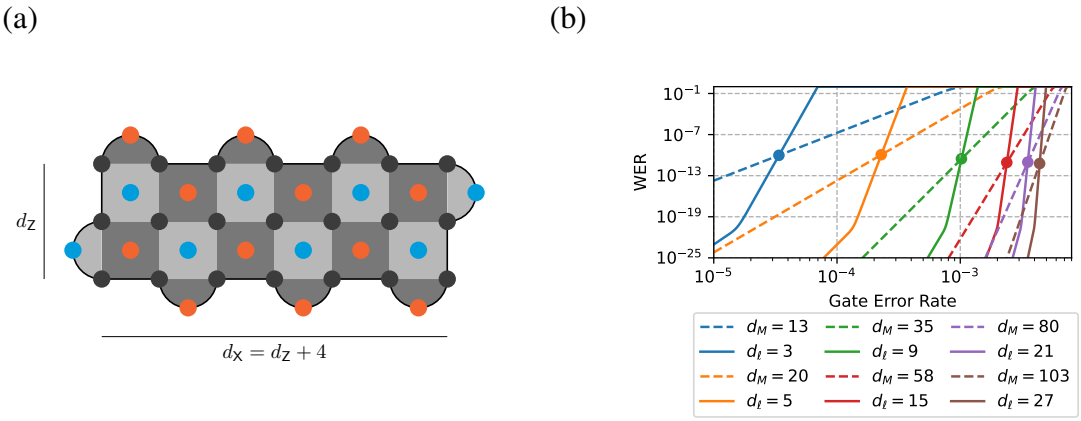


Figure 3.3: (a) Creating Level-1 qubits such that the probability of logical X failure is less than the probability of Z failure. This is accomplished by changing the aspect ratio of the tiles. (b) Estimating crossover points when SWAP gates are 10× better than entangling gates.

for the outer code to partially overcome the problems of degeneracy [PC08]. Similar ideas have proved useful in the context of concatenating GKP codes and LDPC codes [Rav+22].

Lastly, we expect the hierarchical scheme to be resistant to *burst* errors. Unlike typical errors which affect only one or two qubits at a time, burst errors can wipe out entire patches of qubits. This can happen when there exists poorly localized error mechanisms such as the absorption of cosmic rays in superconducting circuits [Vep+20; McE+22; Tho+22; Ach+22; Car+23] or blackbody radiation mediated transitions to other Rydberg states in neutral atom platforms [Fes+22; Zei+16]. Large deviations may also occur in single points of failure in the control hardware such as power supplies, local oscillators, lasers, etc [Aru+19; Kra+19; Mad+20; Eba+21; Ma+22]. Protection of surface codes from burst errors was initially studied in [Xu+22] by concatenating a small constant-sized stabilizer code with surface codes. The hierarchical scheme is robust to these errors because each inner surface code represents a qubit of the outer code which we know is resistant to some number of erasure errors.

Related work: Gottesman [Got00] demonstrated that it is possible to find a threshold using only local gates and concatenation. Svore, Divincenzo and Terhal [STD05; SDT06] studied this issue further and established a numerical lower bound on the threshold in a scheme with many layers of concatenation. Yamasaki and Koashi

[YK22] show that concatenated codes can be used to achieve constant overhead quantum computation that is also time efficient.

In contrast to these approaches, we consider a qualitatively different setting. In our hierarchical model, the concatenated code has only two layers. The outer LDPC code grows quickly to improve the error rate, while the inner code grows slowly to achieve a threshold. The number of encoded logical qubits in the code therefore increases (sublinearly) with the size of the code. Consequently, the rate of error suppression is significantly better.

Finally, Baspin *et al.* [BFS23] have recently generalized the result of Delfosse *et al.* in another direction. In contrast to the constructive approach in this paper, they approach this problem top-down — given access to *arbitrary* local operations and classical communication (not merely Clifford operations), they study syndrome-extraction circuits for LDPC codes and their ability to suppress stochastic errors. They prove the existence of a tradeoff between the parameters of the syndrome-extraction circuit and the sub-threshold error scaling (See Theorem 28 of [BFS23]). For fixed gate error rate p, suppose we use an $[\![N, K, D]\!]$ code $\mathcal{H}_N$ and desire a sub-threshold scaling of the logical failure rate $p_{\mathcal{H}}(N) = \exp(-f(N))$ for some function $f(N)$. Let C_N be the corresponding family of syndrome-extraction circuits. Assuming $f(N) = O(N)$, we express Theorem 28 of [BFS23] in our notation

$$\frac{\mathcal{W}(C_N)}{K} = \Omega\left(\frac{\sqrt{f(N)}}{\mathcal{T}(C_N)}\right) . \tag{3.2}$$

To compare with our result, suppose we only use SWAP gates of constant range, i.e. $R = O(1)$. From Theorem 3.1.2, the syndrome-extraction circuit $C_N^{\mathcal{H}}$ achieves $p_{\mathcal{H}}(N) = \exp\left(-\Theta(N^\delta/\log^{2\delta}(N))\right)$ with $\mathcal{W}(C_N^{\mathcal{H}}) = \Theta(N)$ and $\mathcal{T}(C_N^{\mathcal{H}}) = O(\sqrt{N})$.

$$\frac{\mathcal{W}(C_N^{\mathcal{H}})}{K} = O(\log(N)) , \qquad \frac{\sqrt{f(N)}}{\mathcal{T}(C_N^{\mathcal{H}})} = O\left(\frac{N^{(\delta-1)/2}}{\log^\delta(N)}\right) . \tag{3.3}$$

Comparing with Equation (3.2), we can see that the bound is satisfied for any constant $\delta > 0$. Note that such a low logical error rate is only feasible because our syndrome-extraction circuit $C_N^{\mathcal{H}}$ has polynomially growing depth.

3.2 Background & Notation

In this section, we begin by formally defining concepts needed to state our results. Section 3.2 defines syndrome-extraction circuits. We review gadgets used to construct them and how to use these gadgets to obtain a syndrome-extraction circuit given

an error correcting code. Section 3.2 reviews locally decaying distributions that describe errors on states and faults on circuits. These are general error models that can describe the types of correlated errors that we might witness in a circuit. A noise model is parameterized by a failure rate which quantifies the probability of errors. We described how error correcting codes and their associated syndrome-extraction circuits are robust to some amount of errors occurring below a *threshold* failure rate. Section 3.2 reviews syndrome-extraction circuits for concatenated codes. The hierarchical code is constructed by concatenating a constant-rate quantum LDPC code and the surface code. These are defined in Section 3.2 and Section 3.2 respectively. We review Gottesman's requirements [Got14] for the existence of a threshold. This will be an important idea in the proof of the existence of a threshold for hierarchical codes.

Basic definitions

Let $\mathcal{P} = \langle \mathsf{X}, \mathsf{Z} \rangle / \{\pm i, \pm 1\}$ denote the (projective) single-qubit Pauli group (where we ignore phases); for $n \in \mathbb{N}$, let $\mathcal{P}_n$ denote the n-fold tensor product $\mathcal{P}^{\otimes n}$. For $\mathsf{P} \in \mathcal{P}_n$, $\mathrm{supp}(\mathsf{P}) \subseteq [n]$ denotes the support of P, i.e. the set of qubits on which P acts non-trivially. The *weight* of a Pauli operator P is $|\mathrm{supp}(\mathsf{P})|$, the number of qubits in its support. For brevity, we denote this as $|\mathsf{P}|$.

For $\mathbf{a}, \mathbf{b} \in \{0,1\}^n$, let $\mathsf{X}(\mathbf{a}) = \otimes_i \mathsf{X}^{a_i}$, and $\mathsf{Z}(\mathbf{b}) = \otimes_j \mathsf{Z}^{b_j}$. Any Pauli operator $\mathsf{P} \in \mathcal{P}_n$ can be expressed uniquely as $\mathsf{P} = \mathsf{X}(\mathbf{a})\mathsf{Z}(\mathbf{b})$ for $\mathbf{a}, \mathbf{b} \in \{0,1\}^n$. We use $\mathsf{P}|_\mathsf{X}, \mathsf{P}|_\mathsf{Z} \in \{0,1\}^n$ to denote the X and Z components of P respectively, i.e. $\mathsf{P}|_\mathsf{X} := \mathbf{a}$ and $\mathsf{P}|_\mathsf{Z} := \mathbf{b}$.

Stabilizer codes: An n-qubit quantum error correcting code is the simultaneous $+1$-eigenspace of a set of commuting Pauli operators. These Pauli operators form a subgroup $\mathcal{S}$ of the Pauli group called the stabilizer group. The stabilizer group $\mathcal{S}$ is generated by elements $\mathsf{S}_1, ..., \mathsf{S}_m$. The codespace $\mathcal{Q}$ is then defined as

$$\mathcal{Q} = \{|\psi\rangle \in (\mathbb{C}^2)^{\otimes n} |\ \mathsf{S}_i |\psi\rangle = |\psi\rangle \ \ \forall i \in [m]\} .$$

The number of encoded qubits k is the base 2 logarithm of the number of linearly-independent vectors in $\mathcal{Q}$. Equivalently, given $\mathcal{S}$, it is simply $k = n - m$ (where we assume the stabilizer generators are independent).

Intuitively, the minimum distance d of the code is the minimum weight Pauli operator such that we can map one element of $\mathcal{Q}$ to a distinct element of $\mathcal{Q}$. Formally, we

write

$$d = \min_{\substack{\mathsf{P}\in\mathcal{P}_n\setminus\mathcal{S}\\ [\mathsf{P},\mathsf{S}_i]=0}} |\mathsf{P}| \ .$$

We say such a code is an $[\![n, k, d]\!]$ (stabilizer) code.

The code is said to be a CSS code if every generator can be chosen such that it is a tensor product of only X or Z Pauli operators [CS96; Ste96a]. We can define the X- and Z-distances d_X and d_Z of a CSS code as

$$d_\mathsf{X} = \min_{\substack{\mathsf{P}\in\{\mathsf{I},\mathsf{Z}\}^{\otimes n}\setminus\mathcal{S}\\ [\mathsf{P},\mathsf{S}_i]=0}} |\mathsf{P}| \qquad d_\mathsf{Z} = \min_{\substack{\mathsf{P}\in\{\mathsf{I},\mathsf{X}\}^{\otimes n}\setminus\mathcal{S}\\ [\mathsf{P},\mathsf{S}_i]=0}} |\mathsf{P}| \ .$$

Let $1 \le b \le m_\mathsf{X}$ and $1 \le c \le m_\mathsf{Z}$ index the X-type and Z-type stabilizer generators $\{\mathsf{S}_b^\mathsf{X}\}$ and $\{\mathsf{S}_c^\mathsf{Z}\}$.

Syndrome-extraction circuits & measurement gadgets: A syndrome-extraction circuit C for a code $\mathcal{Q}$ can be composed of the following elements that are allowed to be classically controlled.

Definition 3.2.1 (Clifford operations)**.** *Consider a set of qubits arranged in a lattice in* 2 *dimensions. We define the set* $\mathcal{K}$ *of* elementary Clifford operations *as follows:*

1. *Initialization of new qubits in state* $|0\rangle$ *or* $|+\rangle$*,*
2. *Single-qubit Pauli gates,*
3. *Two-qubit Clifford gates* CNOT *between nearest-neighbor qubits,*
4. *Single-qubit Pauli* X *and* Z *measurements,*
5. *Physical* SWAP *operation with range up to* R*.*

At any given timestep, a qubit in C can be involved in at most one of these operations. In addition, we assume instantaneous classical communication and access to classical computation for processing measurement data.

To obtain the syndrome, we use gadgets to measure Pauli operators which are described as follows. Consider a CSS code $\mathcal{Q}$ with m_X X-type stabilizer generators $\mathcal{S}^\mathsf{X} = \{\mathsf{S}_b^\mathsf{X}\}_{b=1}^{m_\mathsf{X}}$ and m_Z Z-type stabilizer generators $\mathcal{S}^\mathsf{Z} = \{\mathsf{S}_b^\mathsf{Z}\}_{c=1}^{m_\mathsf{Z}}$. The entire set of stabilizer generators is $\mathcal{S} = \mathcal{S}^\mathsf{X} \cup \mathcal{S}^\mathsf{Z}$.

1. **For** $1 \leq b \leq m_{\mathsf{X}}$, measure a product of X operators:
   a) Initialize the b^{th} ancilla qubit in $|+\rangle_b$.
   b) Perform a CNOT gate controlled on the b^{th} ancilla qubit and targeted on each qubit in the support of $\mathsf{S}_b^{\mathsf{X}}$.
   c) Perform a measurement of the b^{th} ancilla in the X basis.
2. **For** $1 \leq c \leq m_{\mathsf{Z}}$**,** measure a product of Z operators:
   a) Initialize the c^{th} ancilla qubit in $|0\rangle_c$.
   b) Perform a CNOT gate targeted on the c^{th} ancilla qubit and controlled on each data qubit in the support of $\mathsf{S}_c^{\mathsf{Z}}$.
   c) Perform a measurement of the c^{th} ancilla in the Z basis.

Figure 3.4 illustrates gadgets for measuring an X operator of weight 5 and a Z operator of weight 4. Given a circuit C, we use two figures-of-merit to quantify its size:

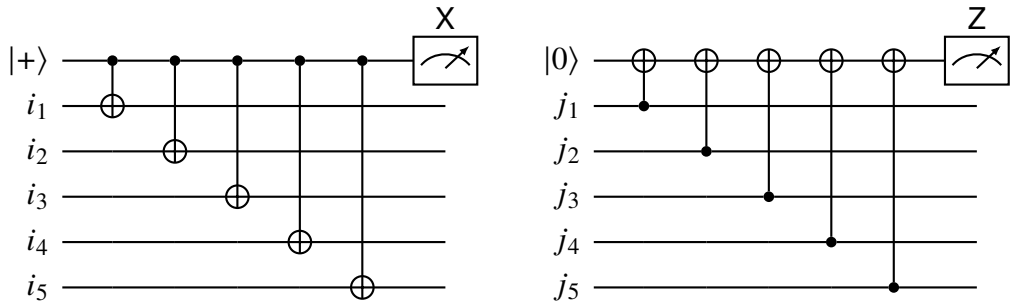


Figure 3.4: Performing the syndrome extraction corresponding to the operator $\mathsf{X}_{i_1}\mathsf{X}_{i_2}\mathsf{X}_{i_3}\mathsf{X}_{i_4}\mathsf{X}_{i_5}$ on the left and $\mathsf{Z}_{j_1}\mathsf{Z}_{j_2}\mathsf{Z}_{j_3}\mathsf{Z}_{j_4}\mathsf{Z}_{j_5}$ on the right. The measurements are performed on some qubits $\{i_1, ..., i_5, j_1, ..., j_5\} \subseteq [n]$.

1. $\mathcal{W}(C)$: the width of the circuit, i.e. the total number of physical qubits, data and ancilla, used in the circuit.
2. $\mathcal{T}(C)$: the depth of the circuit, i.e. the number of timesteps required to measure all syndromes.

We assume operations in $\mathcal{K}$ can be performed in parallel (subject to the constraint that each qubit is only involved in one operation at a time). We shall present one way of using parallel operations to build efficient syndrome-extraction circuits for quantum LDPC codes in Section 3.3.

Noise & imperfect syndrome-extraction circuits

In practice, C is imperfect. In general, errors on multiple locations with complicated correlations can arise at the end of a syndrome-extraction circuit. Under the action of a two-qubit gate for instance, single-qubit errors which occur with probability p can transform into two-qubit correlated errors which occur with probability p. Two-qubit gates themselves can fail and introduce errors on both qubits where there were none before. As yet another example, small clusters of qubits that are near each other can also fail together, for example, due to crosstalk, stray magnetic fields, etc. These errors are outside the scope of an i.i.d. errors model and hence, we consider a generalization.

Definition 3.2.2. *Let $n \in \mathbb{N}$ and* $\mathrm{Pow}(n) = \{E : E \subseteq [n]\}$. *Consider a probability distribution* $\widehat{\Pr} : \mathrm{Pow}(n) \to [0,1]$ *and for $E \subseteq [n]$, let* $\Pr(E)$ *be the total probability*

$$\Pr(E) = \sum_{E' \supseteq E} \widehat{\Pr}(E') \ .$$

We say the distribution Pr *is locally decaying with rate $p \in [0,1]$ if for all $E \subseteq [n]$,*

$$\Pr(E) \leq p^{|E|} \ .$$

We first consider general errors on an n-qubit state. We assume every set of qubits has some probability of being corrupted by an arbitrary Pauli error. Consider a Pauli operator $\mathsf{E}' \in \mathcal{P}_n$ such that $\mathsf{E}' = \mathsf{X}(\mathbf{x}')\mathsf{Z}(\mathbf{z}')$. Let $\widehat{\mathcal{E}}(\mathbf{x}', \mathbf{z}')$ be the probability of the error E'. By definition, $\widehat{\mathcal{E}}$ is itself a map from $\mathrm{Pow}(n) \times \mathrm{Pow}(n)$ to $[0,1]$. Let $\mathcal{X}(\mathbf{x}) : \mathrm{Pow}(n) \to \mathbb{R}$ and $\mathcal{Z}(\mathbf{z}) : \mathrm{Pow}(n) \to \mathbb{R}$ denote

$$\mathcal{X}(\mathbf{x}) = \sum_{\mathbf{x}' \supseteq \mathbf{x}} \sum_{\mathbf{z}'} \widehat{\mathcal{E}}(\mathbf{x}', \mathbf{z}') \ , \qquad \mathcal{Z}(\mathbf{z}) = \sum_{\mathbf{x}'} \sum_{\mathbf{z}' \supseteq \mathbf{z}} \widehat{\mathcal{E}}(\mathbf{x}', \mathbf{z}') \ . \tag{3.4}$$

In other words, $\mathcal{X}$ and $\mathcal{Z}$ denote the probability that a random error E distributed according to $\widehat{\mathcal{E}}$ has X and Z components that contain $\mathbf{x}$ and $\mathbf{z}$ respectively. For brevity, we have used $\mathbf{x}' \supseteq \mathbf{x}$ and $\mathbf{z}' \supseteq \mathbf{z}$ to mean that the supports of $\mathbf{x}$, $\mathbf{z}$ are contained in $\mathbf{x}'$, $\mathbf{z}'$ respectively. Treating $\mathcal{X}$ and $\mathcal{Z}$ separately in this way does not prevent correlations between X and Z errors.

Definition 3.2.3 (Locally decaying errors on qubits)**.** *Given an n-qubit state with Pauli errors distributed according to $\widehat{\mathcal{E}}$. We say that errors are described by a* locally decaying errors model *to mean that $\mathcal{X}$ and $\mathcal{Z}$ are both locally decaying distributions with failure rate p.*

We want to extend this idea to describe errors caused by faulty circuits. A *location* in a circuit C refers to a one- or two-qubit gate (including identity), single-qubit preparation or single-qubit measurement operation at some timestep $1 \leq t \leq \mathcal{T}(C)$. A fault location is a location which performs a random Pauli operation following the desired Clifford operation. We assume that a fault location introduces a Pauli operator on the qubits in its support chosen according to some distribution $\widehat{\mathcal{F}}$. Given a set F of fault locations in C, the support of F is the set $\text{supp}(F) \subseteq [\mathcal{W}(C)]$ of qubits that are in some location in F.

For a set F' of locations, let $\widehat{\mathcal{F}}(F')$ denote the probability of the set of locations F' being faulty. For a set F of fault locations, the total probability $\mathcal{F}(F)$ is

$$\mathcal{F}(F) = \sum_{F' \supseteq F} \widehat{\mathcal{F}}(F') \, . \tag{3.5}$$

Definition 3.2.4 (Locally decaying faults on circuits)**.** *Let C be a depth* 1 *circuit with faults distributed according to $\widehat{\mathcal{F}}$. We say that the faults are described by a* locally decaying faults model *if $\mathcal{F}$ is a locally decaying distribution with failure rate p_{phys}—for all sets of locations F,*

$$\mathcal{F}(F) \leq p_{\text{phys}}^{|F|} \, .$$

Note that the probability of failure falls with the number of locations $|F|$ and not the number of qubits $|\,\text{supp}(F)|$.

In practice, different locations may have different failure rates. To prove the existence of a threshold, we assume that p_{phys} is the maximum failure probability across all gates. We return to this assumption in Section 3.6, where we discuss how the logical failure rate behaves if gates have different failure rates.

Definition 3.2.4 pertains to circuits of depth 1—we assume faults in successive timesteps are independent. In a more general model for faults, we could include arbitrary fault patterns for a circuit of growing depth so long as the probability of a particular fault path falls exponentially with the size of the fault path.

As a state undergoes circuit operations, errors can spread and accumulate. Consider a CNOT gate acting on two qubits. Figure 3.5 illustrates how a generating set of 2-qubit Pauli operators $\{\mathsf{XI}, \mathsf{IZ}, \mathsf{IX}, \mathsf{ZI}\}$ on these two qubits evolve under ideal CNOT. The error doubles in size in the worst-case scenario. As shorthand, we say that Pauli operators 'flow' within circuits to refer to this spreading. X operators flow down a CNOT and Z operators flow up.

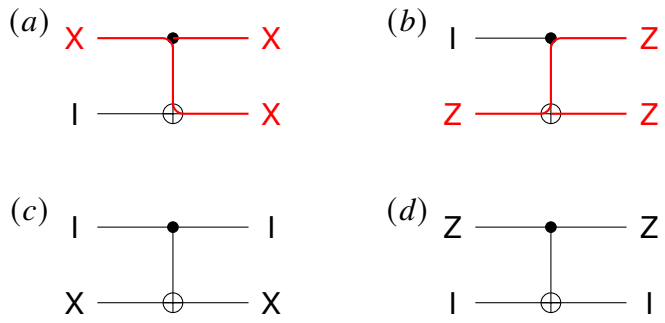


Figure 3.5: Evolution of Pauli errors under the action of CNOT. The first qubit is the control qubit and the second qubit is the target. The operators $\mathsf{X}\otimes\mathsf{I}$ and $\mathsf{I}\otimes\mathsf{Z}$ double in size. The red paths show how X 'flows down' a CNOT gate and Z 'flows up' a CNOT gate.

The structure of syndrome-extraction circuits is special. For $\mathsf{P}\in\{\mathsf{X},\mathsf{Z}\}$, a controlled-P gate within the syndrome-extraction circuit uses ancilla qubits as control qubits and data qubits as target qubits (See Figure 3.4). This means that errors only flow in limited ways—for example, X errors always flow from ancilla qubits to data qubits, and Z errors flow from data qubits to ancilla qubits when CNOT gates are applied.

Implementing the imperfect circuit C, we obtain an imperfect syndrome. To overcome this problem, we repeat the syndrome-measurement circuit for r *rounds*. Let $\boldsymbol{\sigma}=(\boldsymbol{\sigma}_{\mathsf{X}}^{(1)},\boldsymbol{\sigma}_{\mathsf{Z}}^{(1)},\ldots,\boldsymbol{\sigma}_{\mathsf{X}}^{(r)},\boldsymbol{\sigma}_{\mathsf{Z}}^{(r)})$ be the r faulty syndromes.

Failure rate per round:

Consider a corrupted code state $\mathsf{E}\,|\psi\rangle$ where ψ is a code state and $\mathsf{E}=\mathsf{X}(\mathbf{e}_x)\mathsf{Z}(\mathbf{e}_z)$ is some Pauli operator. If the syndrome-extraction circuit C has no faults, the joint state of the data and ancilla qubits after one round of syndrome extraction is described by

$$\mathsf{E}\,|\psi\rangle\otimes\mathsf{Z}(\boldsymbol{\sigma})\,|+\rangle^{\otimes m}\ , \tag{3.6}$$

where $\boldsymbol{\sigma}$ represent the ideal syndromes for X- and Z-type stabilizer generators.

However, because of faults in the circuit, the state after the circuit is

$$(\mathsf{D}\otimes\mathsf{A})(\mathsf{E}\,|\psi\rangle\otimes\mathsf{Z}(\boldsymbol{\sigma})\,|+\rangle^{\otimes m})\ , \tag{3.7}$$

where D and A represent errors on the data and ancilla qubits respectively caused by faults in C that then spread.

Let $\widehat{\mathcal{E}}'(\mathsf{D}\otimes\mathsf{A})$ denote the probability of errors *per round* on the qubits. Let $\mathcal{X}'$, $\mathcal{Z}'$ denote the induced distributions for errors on data and ancilla qubits of X and Z type respectively.

Definition 3.2.5 (Probability of errors per round)**.** *We say that the probability of errors per round is locally decaying with failure rate* $p_{\text{round}} \in [0, 1]$ *such that* $\mathcal{X}'$, $\mathcal{Z}'$ *are locally decaying distributions with failure rates* p_{round} *respectively.*

Definition 3.2.5 thus considers one round of syndrome extraction not as individual operations, but in aggregate; it then associates a failure probability p_{round} with the entire round associated with the probability of witnessing X and Z errors. Thus, p_{round} can be a function of the code size N, as well as other details of the implementation such as the specific syndrome-extraction circuit used.

A priori, $\widehat{\mathcal{E}}'$ can depend on r and the input error E. However, as entangling gates restrict the direction of error propagation, errors do not propagate from one data qubit to another or from one ancilla qubit to another. In Section 3.5, we use this to show that p_{round} does not depend on how many prior rounds of the syndrome-extraction circuit have already been applied. We show that p_{round} is a function of p_{phys} of the form $a \cdot p_{\text{phys}}^b$, where a is a function of the depth $\mathcal{T}(C)$ and b is a function of the degrees Δ_q and Δ_g.

Recovering the state: After performing r rounds of syndrome extraction, a *decoding algorithm* $\mathtt{dec} : (\mathbb{F}_2^m)^{\times r} \to \mathcal{P}_n$ maps the observed syndrome $\boldsymbol{\sigma}$ to a deduced error.

The applied correction may not completely correct all errors due to faults in the syndrome extraction circuit. We declare success if, after applying the correction, the final state is 'not too far' from the desired output of the ideal circuit C. To this end, we consider the ideal recovery map $\mathcal{R}$ [AGP05]—a fictitious quantum channel that is not subject to geometric constraints or noise. We gauge the accuracy of the circuit $\widetilde{C}$ using the logical failure probability p_Q, which is the probability that the residual error is correctable by the ideal recovery map. To be precise, p_Q is the probability that *any* logical qubit fails in one round of error correction. The probability p_Q also referred to as the *Word Error Rate* (WER).

Ideal recovery map & Thresholds: To understand whether a scheme is scalable, we are interested in properties of a *family* of codes $\{Q_n\}$ to process an ever increasing number n of qubits. Consider a code family $\{Q_n\}$ and suppose errors are described by a locally decaying distribution $\mathcal{E}$ with failure rate p_{in}. Let $\{C_n\}$ be the corresponding set of syndrome-extraction circuits to $\{Q_n\}$, where faults are described by $\mathcal{F}$, a

locally decaying distribution with failure rate $p_{\text{phys}} \in [0, 1]$. We can compute p_{round} as a function of p_{phys} as shown in Section 3.5.

For our purposes, we say that the family has a *threshold* with respect to the noise model and decoding algorithm if there exists a pair $q_{\text{in}}, q_{\text{round}} \in (0, 1]$ such that if

$$p_{\text{in}} < q_{\text{in}} , \qquad p_{\text{round}} < q_{\text{round}} , \tag{3.8}$$

the probability of logical failure $p_Q \to 0$ as the size of the code $n \to \infty$. The logical probability of failure is defined with respect to family of ideal recovery maps. It depends on p_{in} and p_{phys} and the thresholds.

Whether a threshold exists with respect to a given noise model, the exact value of the threshold, as well as how quickly the logical failure probability decreases as a function of n (e.g. polynomially or exponentially), depend not only on the choice of quantum error-correcting code Q_n, but also the implementation of the syndrome-extraction circuit C_n and the decoding algorithm. In our construction, the code family is a concatenated code where the syndrome-extraction circuit is subject to constraints on geometric locality.

While the state after error correction is 'close enough' to the codespace, undoing the deduced error may not correct all errors. The remaining errors on the state are described by $\mathcal{E}_{\text{res}}$ that is a locally decaying distribution with failure rate p_{res}. We can perform another round of error correction and thereby keep the state alive for arbitrary duration if $p_{\text{res}} < p_{\text{in}}$. For this reason, we will specify the residual failure rate after error correction in addition to the logical failure probability p_Q.

Concatenated codes

A concatenated code is a quantum code obtained via the composition of two codes, an inner code Q_0 and an outer code Q. We consider the simple case of a $[\![n_0, 1, d_0]\!]$ code Q_0 that only encodes 1 qubit and a suitable $[\![n, k, d]\!]$ code Q.

Code parameters: The concatenated code, denoted $\mathcal{H}$ with parameters $[\![N, K, D]\!]$, is constructed by replacing each qubit of the code Q by a copy of Q_0, resulting in n copies of the inner code Q_0. The benefit of this construction is that the distance D of the code $\mathcal{H}$ is amplified with respect to the constituent codes. To be precise,

$$N = n \cdot n_0 , \qquad K = k , \qquad D = d \cdot d_0 .$$

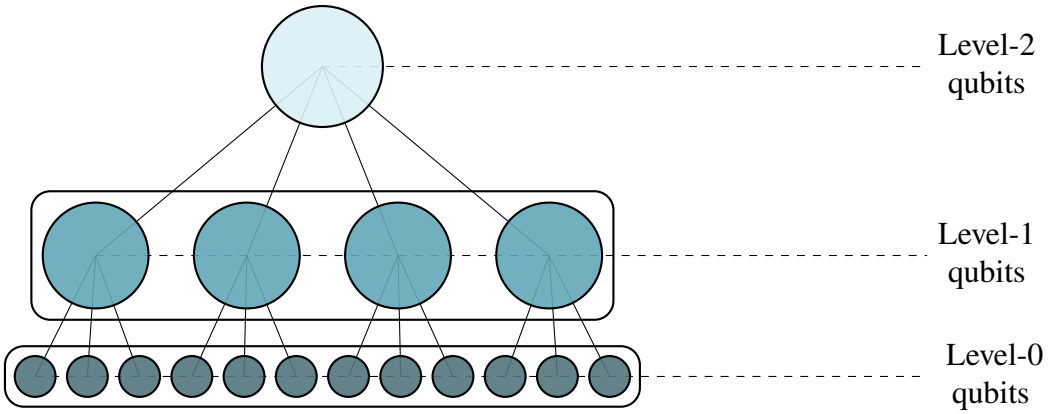


Figure 3.6: Visualizing a concatenated code $\mathcal{H}$.

The physical qubits are referred to as Level-0 qubits, the logical qubits of $\mathcal{Q}_0$ which form the block $\mathcal{Q}$ are referred to as Level-1 qubits, and the logical qubits of $\mathcal{H}$ are referred to as Level-2 qubits. See the schematic in Figure 3.6.

When errors on qubits are distributed in an i.i.d. manner, the advantage of concatenation becomes apparent when we "coarse grain" details of the concatenated code. Consider a simple setting where qubits are subject to independent X and Z errors. Suppose we use the code $\mathcal{Q}$ without concatenation. By assumption, the probability of failure of each of the physical qubits is p. However, after concatenation, the probability of failure of the Level-1 qubits is suppressed—it fails with probability proportional to $p^{d_0/2}$. This is because at least $d_0/2$ errors are required to cause a logical error for $\mathcal{Q}_0$. The inner code thus adds an extra layer of protection and consequently, the logical failure rate for the outer code is that much lower. As we shall see, we have to be more careful when making this sort of argument in the context of circuits.

Syndrome-extraction circuit: Let $C^{\mathcal{Q}_0}$ and $C^{\mathcal{Q}}$ denote the syndrome-extraction circuits for $\mathcal{Q}_0$ and $\mathcal{Q}$ respectively such that both can be implemented in 2 dimensions using $\mathcal{K}$, the set of local Clifford operations and R-local SWAP gates. To implement a CNOT between distant qubits, we may need to permute qubits using SWAP gates to bring them within range of a two-qubit gate. We discuss to how to design such a permutation in Section 3.3.

A syndrome-extraction circuit $C^{\mathcal{H}}$ for the concatenated code $\mathcal{H}$ can be expressed in terms of the syndrome-extraction circuits $C^{\mathcal{Q}_0}$ and $C^{\mathcal{Q}}$. Each data and ancilla qubit

in the syndrome-extraction circuit for $C^{\mathcal{Q}}$ is now replaced with a copy of $\mathcal{Q}_0$. Each gate in $C^{\mathcal{Q}}$ is replaced by the corresponding logical Clifford gate between Level-1 qubits. Thus, even for constructing a quantum memory, we need to understand how to perform a restricted set of inner code logical operations in a fault-tolerant manner. We perform error correction either after the logical gate or in an interleaved manner. We discuss this in the context of our explicit architecture in Section 3.4.

The ideal recovery map $\mathcal{R}_{\mathcal{H}}$ for $\mathcal{H}$ is obtained by first decoding the n copies of the inner code $\mathcal{Q}_0$ using $\mathcal{R}_0$ and then decoding the outer code using $\mathcal{R}_{\mathcal{Q}}$. Here $\mathcal{R}_{\mathcal{Q}_0}$ and $\mathcal{R}_{\mathcal{Q}}$ refer to the ideal recovery maps for $\mathcal{Q}_0$ and $\mathcal{Q}$ respectively. Thus $\mathcal{R}_{\mathcal{H}} = \mathcal{R}_{\mathcal{Q}} \circ (\mathcal{R}_{\mathcal{Q}_0})^{\otimes n}$.

We generalize the notion of location in the context of circuits. A Level-1 location refers to a Level-1 gate, including the error correction rounds. The location is faulty if it implements the incorrect logical operation on the Level-1 qubits in its support. In the context of $C^{\mathcal{Q}}$, a single Level-1 location in the circuit could refer to a SWAP gate or an entangling gate or a preparation or measurement of a logical qubit of $\mathcal{Q}_0$.

When "coarse graining" circuits for concatenated codes, more care is needed than the i.i.d. errors setting. We illustrate using the following examples.

Problem # 1: Level-1 failure rates are not additive

Consider a n_0-qubit code state of the inner code $\rho_{\text{in}}^{(0)}$ with Level-0 errors E_{in}. The error E_{in} is not catastrophic—the ideal decoder $\mathcal{R}_0$ can correct it. The state is therefore correctable.

Consider the syndrome-extraction circuit $C^{\mathcal{Q}_0}$ with Level-0 faulty locations F. Suppose there is some error supported on $\text{supp}(F)$ but that this error is not a logical error. We may then be tempted to extend the notion of correctability to include circuits and declare the circuit $C^{\mathcal{Q}_0}$ correctable. However, this is misleading as a correctable circuit acting on a correctable input state need not produce a correctable output state.

Let $\rho_{\text{out}}^{(0)}$ denote the output state and $\mathsf{E}_{\text{out}}^{(0)}$ denote the errors on this state. Suppose the faulty locations F result in an error E_F. The product $\mathsf{E}_{\text{in}} \cdot \mathsf{E}_F$ might not be correctable. In addition, the errors E_{in} and E_F can spread in unpredictable ways within the circuit. We therefore cannot calculate the Level-1 output failure probability by merely knowing the input state and the faults individually resulted in correctable errors. We need additional structure.

Problem # 2: Level-0 failure rate is not always sustainable

Secondly, the thresholds are decoder dependent. By definition, the ideal decoder $\mathcal{R}_0$ has no faults; if the errors on the state $\rho_{\text{out}}^{(0)}$ are correctable, then $\mathcal{R}_{\mathcal{Q}_0}$ is successful. On the other hand, $C^{\mathcal{Q}_0}$ can contain faults and may be unable to deal with as many errors as the ideal decoder $\mathcal{R}_{\mathcal{Q}_0}$. This can result in instances where the output state $\rho_{\text{out}}^{(0)}$ will be correctable by $\mathcal{R}_0$; by our criteria for success, the output state is decodable. However, the number of residual errors may be above the threshold for error correction. In other words, as error correction is itself faulty, these faults can combine with existing errors to cause a logical failure.

In our construction, we address these problems in Section 3.5. We shall show that for sufficiently low failure rates, we can indeed ignore dealing with the syndrome-extraction circuit for the outer code assuming a failure rate that depends on the inner code. This statement relies on the structure of LDPC codes and surface codes. We now proceed to review these codes.

Constant-rate LDPC codes

An $[\![n, k, d]\!]$ code family $\{\mathcal{Q}_n\}$ is said to be a low-density parity-check code if

1. each stabilizer generator S_i, $i \in [m]$, only acts non-trivially on at most a constant number Δ_g of qubits for all elements in $\{\mathcal{Q}_n\}$.
2. each qubit only participates in at most a constant number Δ_q of stabilizer generators for all elements in $\{\mathcal{Q}_n\}$.

To include the degree of stabilizer generators and qubits, we shall say that a code family $\{\mathcal{Q}_n\}$ is an $[\![n, k, d, \Delta_q, \Delta_g]\!]$ LDPC family.

We will choose the outer code to be a code with constant rate, i.e. $k = \Theta(n)$. Constructing a constant-rate LDPC code is a non-trivial task because there is a conflict between the constraints on stabilizer generators. On one hand, all stabilizer generators need to commute with each other to form a well-defined stabilizer code; on the other hand, the stabilizer generators need to have weight at most Δ_g. Despite these difficulties, there exists constant-rate quantum LDPC codes, i.e. $k(n) = \Theta(n)$, with distance $d(n) = \Theta(n^\delta)$ for $0 < \delta \leq 1$.

LDPC codes have a threshold [KP13a; Got14] *if* operations in $\mathcal{K}$ are not subject to any locality constraints. In this setting, we can construct syndrome-extraction circuits where each qubit is involved in a constant number of two-qubit gates.

Consider a family of $[\![n, k, d, \Delta_q, \Delta_g]\!]$ quantum LDPC codes $\{\mathcal{Q}_n\}$ where $k = \rho \cdot n$ for some constant $\rho > 0$ and distance $d = \Theta(n^\delta)$ for some $\delta > 0$. Suppose qubits are subject to the following errors:

1. the input state is subject to locally decaying errors with failure rate per qubit p_{in}.
2. the syndrome-extraction circuit is subject to locally decaying faults with failure rate per gate p_{phys}.

We restate a result from Gottesman [Got14] (Theorem 4) which guarantees the existence of a threshold for arbitrary LDPC codes. In this construction, we require $r = d(n)$ rounds of syndrome extraction. After syndrome extraction, the (imperfect) syndromes are processed by a minimum-weight decoder `dec`. We do not describe the decoder in detail here and merely note that it exists. For generic LDPC codes, the minimum-weight decoder is not necessarily efficient.

There exist $q_{\text{in}}, q_{\text{round}}$ in the interval $(0, 1]$ such that when

$$p_{\text{in}} \leq q_{\text{in}} , \qquad p_{\text{round}} \leq q_{\text{round}} , \tag{3.9}$$

the following is true.

The minimum-weight decoder `dec` yields a correction such that:

1. the final state is recoverable by an ideal recovery operator $\mathcal{R}_Q$ with probability at least to $1 - p_Q(n)$ where $p_Q(n) := \exp[-\Theta(d(n))]$. To be precise, $p_Q(n)$ is the probability of failure *per round of syndrome extraction*.
2. the physical qubits have residual errors that are described by a locally decaying error model with failure rate at most p_{round}.

The first condition guarantees that the probability of logical failure falls exponentially with the distance of the code. It is worth noting that we declare a logical failure if *any* logical qubit fails. This is *qualitatively* different from codes that only encode a constant number of qubits.

The second condition on the residual error is not what is in the theorem statement of Theorem 4 of [Got14]; however, the proof implies it. For sufficiently low values of p_{phys}, it guarantees that we can continue to perform error correction for

arbitrarily many rounds (conditioned on no logical errors). In other words, we require $p_{\text{round}} < p_{\text{in}}$.

We highlight that this result applies to arbitrary LDPC codes, i.e. it is independent of the rate of the code. In particular, it applies to the surface code.

We note that the threshold is stated in terms of p_{round}, and not directly in terms of p_{phys}. This is for two reasons: (1) this is how Theorem 4 of [Got14] is itself stated, and (2) in our construction, the dependence of p_{round} on p_{phys} can change depending on the depth of the syndrome-extraction circuit. Stating the thresholds in this manner will allow us to derive the functional dependence between p_{round} and the depth of the syndrome-extraction circuit. In Gottesman's construction [Got14], the syndrome-extraction circuit is constant depth and therefore p_{round} is also a constant. In contrast, our construction is more complicated because of constraints on geometric locality.

It is known that codes defined by geometrically-local stabilizer generators in 2 dimensions cannot achieve both constant rate and growing distance [BT09; BPT10]. To achieve a constant rate and distance $d = \Theta(n^{\delta})$ with fixed degrees Δ_q and Δ_g, the amount of non-locality scales with the parameters k and d [BK21a; BK21b]. In other words, there exist $\Theta(n)$ stabilizer generators such that qubits in their support cannot be close to each other in the 2-dimensional lattice. In the context of syndrome-extraction circuits, the result by Delfosse *et al.* [DBT21] states that the depth of the syndrome-extraction circuit will grow when we only have geometrically-local gates and a limited number of ancilla qubits. (Recall Equation (3.1).)

In Section 3.5, we show that p_{round} grows if the syndrome-extraction circuit C is constrained by geometric locality. In other words, it is *not* constant and we need an approach different than Gottesman's to prove the existence of a threshold. In our alternative approach using the hierarchical code, the growth of the inner code suppresses Level-1 logical errors sufficiently to ensure that the Level-2 logical failure rate drops rapidly as the outer LDPC code scales up.

Finally, we discuss the choice of quantum LDPC code. While the result above applies to generic quantum LDPC codes, more is known about specific constructions. Quantum expander codes are one family of constant-rate quantum LDPC codes for which $d = \Theta(\sqrt{n})$ [TZ14; LTZ15]. It has been rigorously proven that these codes can be equipped with an *efficient* decoder called `small-set-flip` [FGL18a; FGL18b]. Furthermore, it was shown that the decoder is *single shot* meaning that it

only requires a constant number of rounds of syndrome measurements for the decoder to function even when the syndrome is noisy. Similar to Gottesman's requirements for the existence of a threshold, all that is needed in Fawzi *et al*. [FGL18a] is for p_{round} to remain constant. However, unlike Gottesman's construction, it was shown that these codes have an efficient decoding algorithm that only require a constant number of rounds of syndrome measurement. Thus, if we wish to implement a quantum expander code, we can use the same machinery presented in this paper to justify an efficient single-shot decoder for the outer code.

We refer to LDPC codes with distance scaling as $d = \Theta(n)$ as good codes. For nearly 2 decades, it was unclear whether good codes even existed. Following a series of breakthroughs, [PK20; EKZ20; KT21; HHO21; BE21], this impasse was famously crossed first by Panteleev & Kalachev [PK21a] and later by Leverrier & Zémor [LZ22b]. Furthermore, these codes have the single-shot property (albeit with an inefficient decoder) as guaranteed by Quintavalle *et al*. [QVRC21].

In this paper, we do not place any constraints on the outer LDPC code other than it have constant rate $\rho > 0$ and distance $d = \Theta(n^\delta)$ for $1/2 \leq \delta \leq 1$. Our construction works for all δ, but we choose $\delta \geq 1/2$ to simplify some theorem statements.

Surface codes

We consider the rotated surface code [BK98; HFDV12], arguably the simplest code that can be laid out on a 2-dimensional lattice. The surface code is an LDPC code, albeit with vanishing asymptotic rate.

An example is shown in Figure 3.7. The code is implemented on a *rotated* lattice, i.e. the points of the lattice correspond to the vertices of squares that run in 45 degree angles relative to the x and y axes. The points of the lattice are labeled (a, b) where $a, b \in \mathbb{Z}/2$. Each black dot represents a data qubit; these are located on integer points, i.e. on points (a, b) where $(a, b) \in \mathbb{Z}$. Each colored dot represents a syndrome qubit; these are located on half-integer points, i.e. on points $(a + 1/2, b + 1/2)$ where $(a, b) \in \mathbb{Z}$. Corresponding to each blue face, we define an X-type stabilizer generator that jointly measures $\mathsf{X}^{\otimes 4}$ on adjacent data qubits. Similarly, corresponding to each red face, we define a Z-type stabilizer generator that jointly measures $\mathsf{Z}^{\otimes 4}$ on adjacent qubits. The semi-circles represent stabilizers that only act on two qubits in their support, i.e. they measure $\mathsf{X}^{\otimes 2}$ or $\mathsf{Z}^{\otimes 2}$ jointly.

The rotated surface code $\mathcal{RS}_\ell$ encodes exactly one qubit and has distance d_ℓ. It uses d_ℓ^2 data qubits and $d_\ell^2 - 1$ syndrome qubits. The total number of qubits is thus

$\ell^2 = 2d_\ell^2 - 1$. We use ℓ to parameterize the code family. We also refer to each code as a tile.

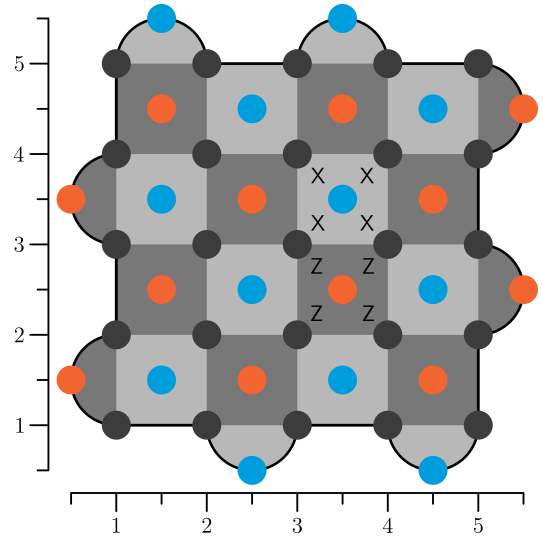


Figure 3.7: A surface code of distance $d_\ell = 5$. Each dark gray dot represents a data qubit. Light faces correspond to X checks and dark faces correspond to Z checks. They are measured using the qubit represented as a blue or orange dot respectively in the center of each face. Note that data qubits reside at integer points $\mathbb{Z}^2$ and ancilla qubits reside at the points of this lattice shifted by $(1/2, 1/2)$.

Using operations $\mathcal{K}$, the syndrome-extraction circuit for the surface code has depth 6.

Thresholds for error correction: To motivate our noise model, we consider a simple setting where $n = 1$, i.e. we have a single tile $\mathcal{RS}_\ell$. Suppose we are given physical qubits, each qubit in some fixed computational-basis state, and use the syndrome-measurement circuit to *project* this state onto a (fixed) code state of the surface code. If the physical qubits are subject to locally decaying errors at failure rate $p_{\text{phys}}^{(0)}$, we can derive the Level-1 probability of failure $p_{\mathcal{RS}}^{(1)}(\ell)$ for the surface code. The superscripts denote the noise on Level-1 and Level-0 qubits respectively.

Contrast this with the scenario where we obtain the surface code from another party. Upon receipt, we are only informed that the tile has already failed with failure rate $p_{\text{in}}^{(1)}$; we do not have additional information, such as syndrome histories from prior rounds of error correction. If the code has not *already* failed, then we are guaranteed that the physical failure rate is $p_{\text{phys}}^{(0)}$. Failure after error correction can thus result in two ways: either the tile fails prior to us receiving the state with probability $p_{\text{in}}^{(1)}$ *or* conditioned on it being correct, it fails because of error correction with probability

$p^{(1)}_{\mathcal{RS}}(\ell) = \exp(-c_{\mathrm{EC}} \cdot \ell)$ for some positive number c_{EC} that does not depend on ℓ. The Level-1 failure rate after error correction is thus $p^{(1)}_{\mathrm{in}} + p^{(1)}_{\mathcal{RS}}(\ell)$.

When performing *repeated* rounds of error correction, we require the Level-1 failure rate $p^{(1)}_{\mathrm{in}}$ to bound the probability that the code has already failed in prior rounds.

Surface codes will form the inner code in our concatenated construction. Consider the syndrome-extraction circuit C for the constant-rate LDPC code $\mathcal{Q}_n$. Suppose $\mathcal{W} = \mathcal{W}(C)$ is the width of the circuit. We require an arrangement of $\mathcal{RS}_\ell^{\otimes \mathcal{W}}$ in two dimensions. In Section 3.4, we introduce a *bilayer* architecture for arranging tiles in two parallel layers. We return to the explicit description of this layout in Section 3.4.

Consider an input state of the code $\mathcal{RS}_\ell^{\otimes \mathcal{W}}$. The errors are distributed in the following manner:

1. Level-1 errors are described by $\mathcal{E}^{(1)}$, a locally decaying distribution with failure rate $p^{(1)}_{\mathrm{in}}$.
2. Level-0 errors are described by $\mathcal{E}^{(0)}$, a locally decaying distribution with failure rate $p^{(0)}_{\mathrm{in}}$.

Faults in the syndrome-extraction circuit are described by $\mathcal{F}^{(0)}$, a Level-0 locally decaying distribution with failure rate $p^{(0)}_{\mathrm{phys}}$.

The code $\mathcal{RS}_\ell^{\otimes \mathcal{W}}$ is itself an LDPC code (with vanishing rate asymptotically), and therefore, we can apply Theorem 4 from [Got14]. We note that although the original theorem is itself is not stated in this way, the proof implies the following.

There exist thresholds $q^{(0)}_{\mathrm{in}}$, $q^{(0)}_{\mathrm{round}}$ on Level-0 failure rates such that, below threshold, the probability of logical failure after error correction is described by a locally decaying Level-1 error $p^{(1)}_{\mathrm{in}} + p^{(1)}_{\mathcal{RS}}(\ell)$ where $p^{(1)}_{\mathcal{RS}}(\ell) = \exp(-c_{\mathrm{EC}} \cdot \ell)$ for some positive number c_{EC} that is independent of ℓ.

In addition, the state after error correction is described by locally decaying errors with failure rate proportional to $p^{(0)}_{\mathrm{round}}$. This guarantees that if we are sufficiently below threshold, then the number of residual errors is low enough such that we can apply another round of error correction.

Unlike the case for general LDPC codes, surface codes possess a minimum weight decoder that runs in $\mathrm{poly}(n)$ time by mapping the decoding problem to a minimum-weight perfect matching problem.

Logical Clifford operations: As highlighted in the subsection on concatenated codes, we need to implement logical Clifford operations for the surface code to be able to use it within a concatenated construction. Extending our notation from Definition 3.2.1, we let $\mathcal{K}_0$ denote the physical geometrically-local Clifford gates and R-local SWAP operation on the physical qubits. Let C_0 be the syndrome-extraction circuit for $\mathcal{RS}_\ell$ constructed using $\mathcal{K}_0$.

Let $\mathcal{K}_1$ denote the corresponding logical operations on the surface code. Single-tile operations in $\mathcal{K}_1$ — state preparation in a fixed stabilizer state, (destructive) measurement of logical Pauli operators and applying Pauli corrections — can be performed using operations in $\mathcal{K}_0$ regardless of how two-tile gates are implemented. The only Clifford operations we require are two-tile operations: CNOT and R-local SWAP. These are discussed in Section 3.4.

3.3 Permutation routings on sparse graphs in two dimensions

In this section, we prove Theorem 3.1.1, restated here for convenience.

Theorem. *For R even, there is an efficient construction of a degree-12 graph $G = (V, E)$ whose vertex set V is identified with an $L \times L$ lattice with edges of length at most R. Any permutation $\alpha : V \to V$ can be performed in depth $3L/R + O(\log^2 R)$.*

We use this result in the next section to construct syndrome-extraction circuits for quantum LDPC codes. We shall study permutation routings on graphs and focus on $\mathrm{NN}_2(L, R)$, the $L \times L$ lattice in 2 dimensions where two vertices share an edge if they are separated by a distance of at most R. Based on the idea of a permutation routing on product graphs, we demonstrate that we can implement an arbitrary permutation in depth $O(L/R)$. For the special case of the 2D lattice, we can make heavy use of sorting networks to find implementations of target permutations.

Using sorting networks to implement long-range connectivity is itself not a new idea [Bea+13]. For instance, it was used in Delfosse *et al.* [DBT21] to construct syndrome-extraction circuits for quantum expander codes to match the bound in Equation (3.1). The results in this section generalize this idea to arbitrary syndrome-extraction circuits with constant spatial overhead. To the best of our knowledge, this is the first work to construct sparse syndrome-extraction circuits when R can scale as a function of L.

Permutation Routing on product graphs

A *permutation routing* is sometimes explained in terms of a pebble-exchange game, where pebbles are placed on the vertices of an (undirected) connected graph $G = (V, E)$. The pebble on vertex $u \in V$ has an address $\alpha(u)$. The addresses of all the pebbles together specify a permutation α on the vertices of G. We are allowed to swap any two pebbles along an edge of G. Formally, every vertex has a label and for every edge $(u, v) =: e \in E$, we are equipped with an edge permutation $\pi(e)$ that exchanges the labels of u and v. Edge permutations can be performed in parallel as long as every pebble is involved in at most one edge permutation in one timestep. We say β is a simple permutation on G if it is the product of edge permutations $\{\pi(e)\}_e$ that commute. The objective of the pebble-exchange game is to find a minimum sequence of simple permutations so that the pebble that began at u is located on the vertex $\alpha(u)$ afterwards. In other words, we wish to find the smallest sequence of simple permutations $\beta_1, ..., \beta_{\mathcal{T}(\alpha)}$ such that $\alpha = \beta_{\mathcal{T}(\alpha)} \circ \cdots \circ \beta_1$. Here $\mathcal{T}(\alpha)$ denotes the minimum number of simple permutations required to perform α. We represent permutations using the one-line notation [Wik22] where $\alpha = \begin{pmatrix} \alpha_1 & \alpha_2 & \cdots & \alpha_n \end{pmatrix}$ means 1 is mapped to α_1, 2 is mapped to α_2 and so on. Given any permutation α, the permutation α^{-1} can be computed efficiently by applying the permutation to a list of consecutive integers $[n]$.

The R-nearest-neighbor graph: The R-nearest-neighbor graph in 1 dimension of length L is denoted $\mathrm{NN}_1(L, R) = (V, E)$ where

$$V = \{1, ..., L\}, \qquad E = \{(u, v) : |u - v|_2 \leq R\}\,, \tag{3.10}$$

where $|\cdot|_2$ represents the standard 2-norm. This is the graph for which the vertices are simply the positive integers up to L and two vertices are connected by an edge if their difference is less than R. In particular, consider the graph $\mathrm{NN}_1(L, 1)$ which corresponds to the path graph.

Fact: We can perform an arbitrary permutation α of pebbles placed on the vertices of the path graph $\mathrm{NN}_1(L, 1)$ in depth $L - 1$ [Knu97]. The explicit permutation routing algorithm `Path-Routing` is presented in Algorithm 3.

To illustrate, we consider a permutation α on the path graph on 8 vertices in Figure 3.8. Here, $\alpha = \begin{pmatrix} 6 & 7 & 2 & 5 & 3 & 4 & 8 & 1 \end{pmatrix}$.

We can generalize this concept and define the R-nearest-neighbor graph in 2 dimensions which we denote by $\mathrm{NN}_2(L, R)$. It has vertices $\{\mathbf{u} : \mathbf{u} = (u_x, u_y) \in$

Algorithm 3 `Path-Routing`(α)

Input: Permutation α.
Output: simple permutations $\beta_1, ..., \beta_{L-1}$ such that $\alpha = \beta_{L-1} \circ \cdots \circ \beta_1$.

`labels` $\leftarrow \{\alpha^{-1}(1), ..., \alpha^{-1}(L)\}$
$t = 1$.
while $t \leq L - 1$ **do**
  $\beta_t \leftarrow \{1, \ldots, L\}$
  for $i \in \{1, ..., \lfloor L/2 \rfloor\}$ **do**
    $a \leftarrow 2i - 1$ if t is even else $2i$
    $b \leftarrow 2i$ if t is even else $2i + 1$
    if `label`(a) < `label`(b) **then** ▷ *Swap*
      $\beta_t(a) \leftarrow b$
      $\beta_t(b) \leftarrow a$
      Exchange `label`(a) and `label`(b).
  $t \leftarrow t + 1$.
return $\beta_1, ..., \beta_{L-1}$.

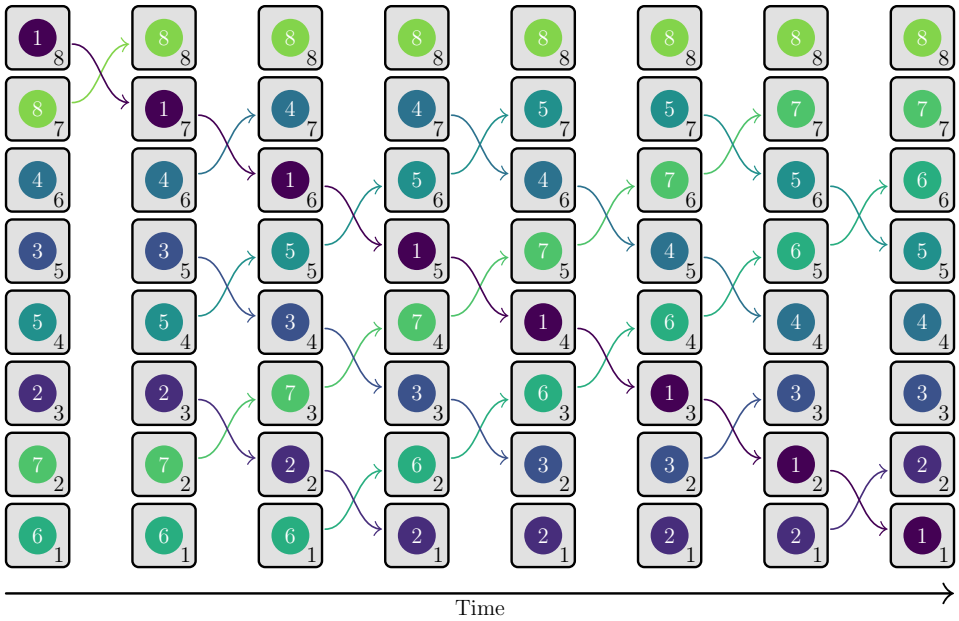


Figure 3.8: Example: implementing the permutation $\alpha = \begin{pmatrix} 6 & 7 & 2 & 5 & 3 & 4 & 8 & 1 \end{pmatrix}$ with nearest-neighbor swaps using Algorithm 3.

$[L] \times [L]\}$; two vertices $\mathbf{u}, \mathbf{u}'$ share an edge if $|\mathbf{u} - \mathbf{u}'|_2 \leq R$. Our objective is to build up to a routing algorithm on this graph. Before considering this general case, we study the case where $R = 1$. The idea used there will be used again for general R in the next subsection.

Routing on graph products: The main idea we present in this subsection are techniques due to Annexstein and Baumslag [AB90] to route on Cartesian products of graphs. They showed that we can derive routing algorithms for the Cartesian product $G_1 \times G_2$ using routing algorithms for graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$.

The general routing algorithm `Product-Routing` presented in Algorithm 4 below applies to any two graphs G_1 and G_2 for which routing routines are known. Each $v_1 \in V_1$, defines a "row" $\mathcal{R}_{v_1} = \{v_1\} \times V_2$, and each $u_2 \in V_2$ defines a "column" $\mathcal{C}_{u_2} = V_1 \times \{u_2\}$ [4]. We call a permutation α of the vertices of $G_1 \times G_2$ a *row permutation* if the permutation respects a decomposition into rows i.e. for all $v_1 \in V_1$, $\alpha : \mathcal{R}_{v_1} \to \mathcal{R}_{v_1}$. Likewise, for a *column permutation*, we have that for all $u_2 \in V_2$, $\alpha : \mathcal{C}_{u_2} \to \mathcal{C}_{u_2}$. A row or column permutation can be implemented using routing routines for G_2 or G_1 by applying the routine to each copy in the Cartesian product.

Lemma 3.3.1 (Annexstein & Baumslag [AB90])**.** *For any routing α_{12} on $G_1 \times G_2$, there exist column permutations α_1, α_1' and row permutations α_2 on G_2 such that: $\alpha_{12} = \alpha_1 \circ \alpha_2 \circ \alpha_1'$. These permutations can all be computed in polynomial time. If the permutations α_1 and α_1' require depth at most $\mathcal{T}_1$ and α_2 requires depth at most $\mathcal{T}_2$. Then α_{12} requires depth at most $2\mathcal{T}_1 + \mathcal{T}_2$.*

We provide some intuition for this lemma. At first glance, one might expect that a row permutation α_1 followed by a column permutation α_2 ought to suffice. However, this will not always work—if two pebbles in a row share the same destination column, then no row permutation will be able to send both the pebbles to the correct column.

To avoid collisions, we start the procedure with an additional step. We first send pebbles to rows in which no other pebbles shares the same destination column, so that the routing procedure performs a column permutation, a row permutation, and finally a column permutation. This problem will be rephrased as an edge-coloring problem where each color corresponds to the intermediate row that qubits will be routed through.

We first construct a bipartite multigraph B over the vertices $(V_2 \sqcup V_2')$ with left and right vertex sets both copies of V_2 [5]. To each pebble, we associate an edge between the initial column on the left and the destination column on the right. B is bipartite

[4] This notation is inspired by thinking about the vertices arranged as a matrix of size $V_1 \times V_2$.

[5] A multigraph is a generalization of a graph where two vertices are allowed to share multiple edges

and has degree at most $|V_1|$, so there exists an efficiently computable edge coloring with $|V_1|$ colors [Sch+03] i.e. a decomposition into $|\mathcal{R}|$ disjoint matchings.

To each color, $\tau \in [V_1]$, we will assign an arbitrary row. For each pebble (edge), we will first pre-route it to the assigned row (color) before completing a final routing along the rows then columns. In a valid coloring, no two edges (pebbles) of the same color (intermediate row) are incident to the same vertex (column). In the first step, this means that, for every column, each pebble has a unique intermediate destination row. Further, in the row permutation step, for every row, each pebble has a unique destination column. Finally, in the last column permutation step, each pebble is in its destination column, so, for each column, each pebble has a unique destination row.

We assume we are given blackbox access to an efficient edge coloring algorithm for bipartite graphs [Sch+03] and call it via a subroutine `Edge-Coloring` in Algorithm 4.

Algorithm 4 `Product-Routing`(α)

Input: Permutation $\alpha : V_1 \times V_2 \to V_1 \times V_2$
Output: Row permutations α_1, α_1' and a column permutation α_2 such that $\alpha = \alpha_1 \circ \alpha_2 \circ \alpha_1'$.

Initialize bipartite graph $B \leftarrow (V_2 \sqcup V_2', \varnothing)$ with no edges.
for Every $(v_1, u_2) \in V_1 \times V_2$ **do**
    Draw an edge between $u_2 \in V_2$ and $u_2' \in V_2'$ if $\alpha(v_1, u_2) = (v_1', u_2')$.
$\tau \leftarrow$ `Edge-Coloring`(B)
for $(v_1, u_2) \in E$ **do**
    $\alpha_1'(v_1, u_2) \leftarrow (\tau(e), u_2)$
    $\alpha_2(\tau(e), u_2) \leftarrow (\tau(e), u_2')$
    $\alpha_1(\tau(e), u_2') \leftarrow (v_1', u_2')$

To illustrate Algorithm 4, we describe how to obtain the permutation routing on the nearest-neighbor graph in 2-dimensions $\mathrm{NN}_2(L, 1)$. Noting that $\mathrm{NN}_2(L, 1) \cong \mathrm{NN}_1(L, 1) \times \mathrm{NN}_1(L, 1)$, Lemma 3.3.1 implies that an arbitrary permutation on $\mathrm{NN}_2(L, 1)$ can be implemented using a product of permutations on the components $\mathrm{NN}_1(L, 1)$. Recalling that any permutation on the path graph $\mathrm{NN}_1(L, 1)$ can be done in depth $L - 1$ implies the following corollary.

Corollary 3.3.2. *Any permutation α on $\mathrm{NN}_2(L, 1)$ can be performed in depth $3L - 3$.*

Proof. $\mathrm{NN}_2(L, 1) \cong \mathrm{NN}_1(L, 1) \times \mathrm{NN}_1(L, 1)$, and we can route on $\mathrm{NN}_1(L, 1)$ using an even-odd sorting network (Algorithm 3). Using Algorithm 4, we have that the total number of steps is $3(L - 1)$. ∎

In a different context, the above claim was also made Thompson & Kung [TK77].

Figure 3.9 shows an example of a permutation of such a lattice using Algorithm 4.

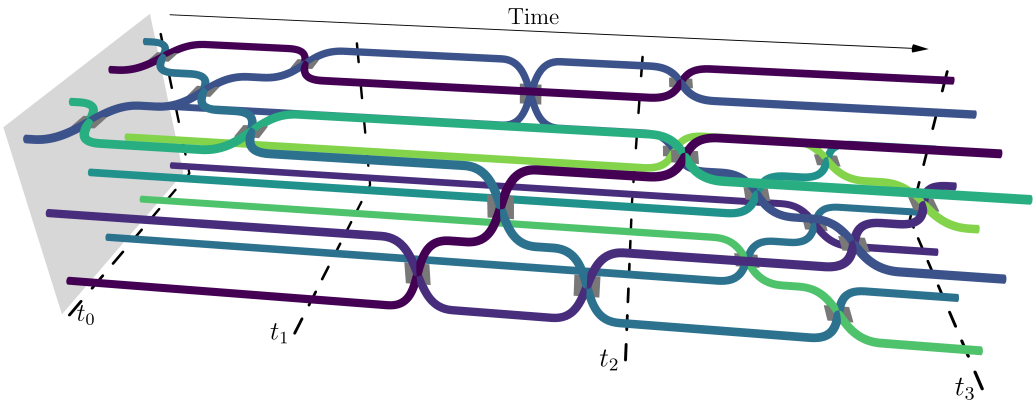


Figure 3.9: Visualizing the routing algorithm via the space-time path of individual qubits. For each swap location, a gray rectangle indicated the plane of the swap is drawn for visualization purposes. Note that in the intervals (t_0, t_1) and (t_2, t_3) there are only row swaps, and in the interval (t_1, t_2) there are only column swaps. Each of the swaps within a single row or column we obtain by a 1D even-odd sorting network (Figure 3.8).

Permutation routing given long-range gates

In this subsection, we show how to route on $\mathrm{NN}_2(L, R)$. We do this by finding graph approximations – subgraphs of our original graph that we can route on nearly as well. We will approximate $\mathrm{NN}_2(L, R)$ using a two-step approach—first we show that the complete graph times a 2-dimensional nearest-neighbor graph approximates $\mathrm{NN}_2(L, R)$ well; we then show that a sparse graph approximates the complete graph well. Together, this will result in a circuit with sparse connectivity that exploits long-range connectivity of range R.

The complete graph & sparse approximations: The complete graph K_m is a graph on m vertices with edges between every pair of vertices. Any permutation α on K_m can trivially be accomplished in depth 2. Using a complete graph will simplify some of the analysis in this section. However, K_m is a dense graph; in turn, the corresponding syndrome-extraction circuit we construct from it will require that qubits are involved in a super constant number of two-qubit gates. To avoid this problem, we replace K_m by a sparse graph in exchange for a modest increasing in

the depth of permutations. We state some facts about sparse approximations to the complete graph K_m.

Definition 3.3.3 (Spectral Expander)**.** *Let G be a d-regular graph on m vertices where all eigenvalues of the adjacency matrix except for the largest $\{\lambda_i\}_{i=2}^{m}$ satisfy $|\lambda_i| \leq \lambda < d$. G is said to be an* (m, d, λ)-spectral expander.

Fact 3.3.4 ([Fri08])**.** *For even $d \geq 4$ and any $\epsilon > 0$, there is an efficient randomized*[6] *algorithm that returns a random d-regular graph such that it is an (m, d, λ)-spectral expander for $\lambda = 2\sqrt{d-1} + \epsilon$.*

This fact establishes that a random regular graph is a good spectral expander with high probability. For $d = 4$, such a graph can be defined on any even number of vertices, so this family is extremely flexible. For convenience, we will set $d = 4$ and m even. We will call a random 4-regular graph picked in this way on m vertices $\mathcal{E}_m$.

The next fact concerns routing on spectral expanders in an efficiently computable manner.

Fact 3.3.5 ([ACG93])**.** *Let G be an (m, d, λ)-spectral expander. Then, any permutation $\sigma : [m] \to [m]$ can be performed in depth $O\left(\frac{d^2}{(d-\lambda)^2} \log^2(m)\right)$.*

Take together, we can replace K_m by a random 4-regular subgraph $\mathcal{E}_m$ on which we can route in depth $O(\log^2(m))$. For our purposes, we assume that the routing algorithm for these sparse graphs can be accessed in a black-box manner.

Now we are prepared to move on to implementing permutations on $\mathrm{NN}_2(L, R)$. The depth we will find is nearly optimal, even when $R > 1$. For convenience, let us assume that L/R is an integer. First, note that at distances shorter than R, $\mathrm{NN}_2(L, R)$ locally "looks" like a complete graph on R-vertices. We can leverage this to find a spanning subgraph[7] of $\mathrm{NN}_2(L, R)$ that is a product of graphs that we know how to route on: K_R and $\mathrm{NN}_2(L/R, 1)$.

Lemma 3.3.6. *If R divides L, $\mathrm{NN}_1(L/R, 1) \times K_R$ is a spanning subgraph of $\mathrm{NN}_1(L, R)$.*

[6] We note that the result in [Fri08] only shows that a regular random graph will be an expander with high probability. However, spectral expansion is efficiently checkable, so this process may be repeated until success.

[7] A subgraph H of a graph G is said to be a spanning subgraph if all vertices of G are contained in H.

Proof. Using the coordinates $[L/R] \times [R]$ for $\mathrm{NN}_1(L/R, 1) \times K_R$ and $[L]$ for $\mathrm{NN}_1(L, R)$, we can map the vertices of $\mathrm{NN}_1(L/R, 1) \times K_R$ to those of $\mathrm{NN}_1(L, R)$ using the bijection $\eta\colon [L/R] \times [R] \to [L]$

$$(a, b) \xrightarrow{\eta} (a-1)R + b\ .$$

Away from the boundary, the neighbors of an arbitrary vertex (a, b) of $\mathrm{NN}_1(L/R, 1) \times K_R$ are $(a \pm 1, b)$ and $(a, [R] \setminus \{b\})$. A vertex (a, b) at a boundary is adjacent to the vertices $(a, [R] \setminus \{b\})$ and one of $(a-1, b)$ or $(a+1, b)$; whichever is in the graph. All elements of these sets are at most a distance R from (a, b) under η, so it is a valid edge in $\mathrm{NN}_1(L, R)$. ∎

Clearly, $\mathrm{NN}_1(L, R) \times \mathrm{NN}_1(L, R)$ is a spanning subgraph of $\mathrm{NN}_2(L, R)$ given by retaining only those edges connecting vertices within a single row or column.

Corollary 3.3.7. *Any permutation on* $\mathrm{NN}_2(L, R)$ *can be performed in depth* $3L/R+9$.

Proof. Denote $\mathrm{NN}_1(L/R, 1) \times K_R$ by H. By Lemma 3.3.6, H is a spanning subgraph of $\mathrm{NN}_1(L, R)$, and $\mathrm{NN}_1(L, R) \times \mathrm{NN}_1(L, R)$ is a spanning subgraph of $\mathrm{NN}_2(L, R)$. It follows that $H \times H$ is a spanning subgraph of $\mathrm{NN}_2(L, R)$, so any simple permutation for $H \times H$ is a simple permutation for $\mathrm{NN}_2(L, R)$.

Permutations on K_R and $G_2 = \mathrm{NN}_1(L/R, 1)$ can be implemented in depth 2 and $L/R-1$, respectively. Setting $G_1 = K_R$ and $G_2 = \mathrm{NN}_1(L/R, 1)$ in Lemma 3.3.1, any permutation on H can therefore be implemented in depth $(L/R-1)+2\cdot 2 = L/R+3$. Invoking Lemma 3.3.1 again with $G_1 = G_2 = H$, we can implement any permutation on $H \times H$ and hence, $\mathrm{NN}_2(L, R)$ in depth $3L/R + 9$. ∎

Note even though there are many edges that we do not use, the lowest depth routing can be no better than the graph diameter of $\mathrm{NN}_2(L, R)$ which is roughly $\sqrt{2}L/R$, so this is nearly optimal. Furthermore, owing to the translation invariance of $\mathrm{NN}_2(L/R, 1)$, the embedding of $K_R \times K_R \times \mathrm{NN}_2(L/R, 1)$ is also translation invariant—far from the boundaries, the graph remains the same locally under translation of R units in the vertical and horizontal directions.

Now, $\mathrm{NN}_2(L, R)$ is not sparse: the degree of each vertex grows as R^2. For practical purposes, it would be convenient if only a sparse subgraph of $\mathrm{NN}_2(L, R)$ were used in the routing routine. In Corollary 3.3.7, we used only the edges contained in a subgraph $\mathrm{NN}_1(L/R, 1) \times K_R \times \mathrm{NN}_1(L/R, 1) \times K_R$. $\mathrm{NN}_1(L/R, 1)$ is sparse, but K_R

is not. However, we can replace K_R by a sparse expander graph so that the subgraph we use is sparse. Fact 3.3.5 supplies such a family of graphs and a depth $O(\log^2 R)$ routing subroutine.

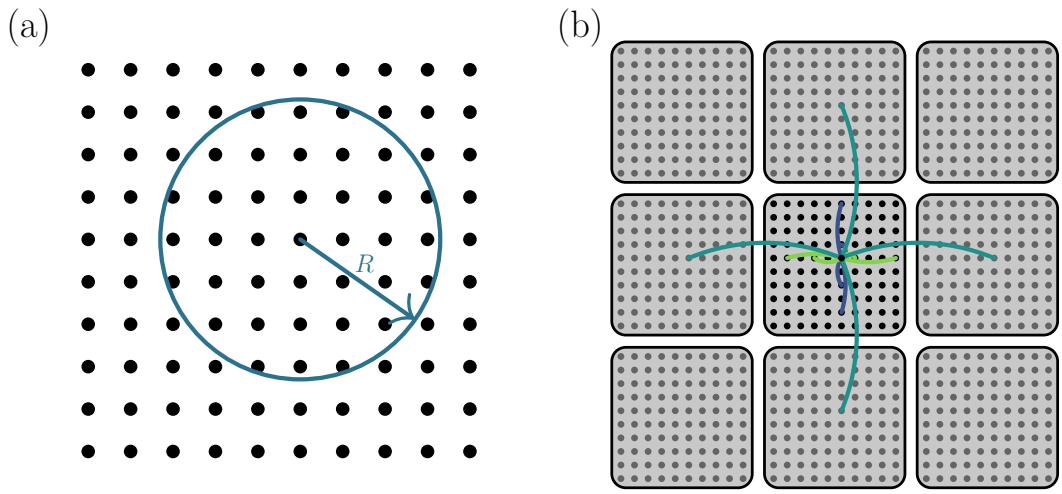


Figure 3.10: (a) A square lattice $\mathrm{NN}_2(L, R)$ with a qubit on each lattice point. The blue circle of radius R denotes the interaction radius for a qubit in the lattice. Such a circle exists around each qubit; we only draw one for clarity. (b) Approximating the lattice using the sparse product graph $\mathcal{E}_R \times \mathcal{E}_R \times \mathrm{NN}_2(L/R, 1)$. Different colored edges come from different factors in the product.

We now bring these ideas together formally in the following corollary.

Corollary 3.3.8. *For R even, there is an efficiently constructable degree-12 spanning subgraph of 2-dimensional R-nearest-neighbor lattice $\mathrm{NN}_2(L, R)$ on which any permutation can be performed in depth $3L/R + O(\log^2 R)$.*

Proof. We will use replace the use of the fully connected graph in Corollary 3.3.7 with a sparse expander. Consider a 4-regular random graph $\mathcal{E}_R$ generated according to fact 3.3.4. By fact 3.3.5, we can route on $\mathcal{E}_R$ in depth $O(\log^2 R)$.

Now consider the graph $H = \mathcal{E}_R \times \mathrm{NN}_1(L/R, 1)$. $\mathcal{E}_R$ is a spanning subgraph of K_R, so it follows by lemma 3.3.6 that H is a spanning subgraph of $\mathrm{NN}_1(L/R, 1)$. Further, using Lemma 3.3.1, we can implement any permutation on H in depth $L/R + O(\log^2 R)$, so we can also implement any permutation on $H \times H$ in depth $3L/R + O(\log^2 R)$.

By an identical argument to Corollary 3.3.7, we have that $H \times H$ is a spanning subgraph of $\mathrm{NN}_2(L, R)$. Further, $H \times H$ has vertex degree 12[8] since the max vertex degree of the product of graphs is the sum of the max vertex degrees of the factors. ■

This subgraph is illustrated in figure 3.10 (b). Later, we will use the contents of the corollary to construct syndrome-extraction circuits with two-qubit gates of range at most R and where each qubit only needs to interact with a constant number of other qubits.

3.4 Bilayer implementation of hierarchical codes

In this section, we will prove Theorem 3.1.2. Note that R can be taken to be any number (such as 1) and the required connectivity is always sparse. We state Theorem 3.1.2 here in two parts and present the proof for each part in turn.

Theorem 3.4.1 (Theorem 3.1.2, Part 1)**.** *The $[\![N, K, D]\!]$ hierarchical code $\mathcal{H}_N$ is constructed by concatenating an outer code, a constant-rate $[\![n, k, d, \Delta_q, \Delta_g]\!]$ quantum LDPC code $\mathcal{Q}_n$ and an inner code, a rotated surface code $\mathcal{RS}_\ell$ where $d_\ell = \Theta(\log(n))$. Let $\rho > 0$ and $\delta \geq 1/2$, such that $k = \rho \cdot n$ and $d = \Theta(n^\delta)$. The code $\mathcal{H}_N$ has parameters*

$$K(N) = \Theta\left(\frac{N}{\log(N)^2}\right), \qquad D(N) = \Omega\left(N^\delta / \log^{2\delta-1}\left[\frac{N}{\log(N)}\right]\right).$$

Proof. We first define the hierarchical code family $\{\mathcal{H}_N\}$ and corresponding syndrome-extraction circuits $\{C_N^{\mathcal{H}}\}$. The element of this family indexed by $N = N(n) = n \cdot d_\ell^2$ is created by concatenating an outer LDPC code $\mathcal{Q}_n$ with an inner surface code $\mathcal{RS}_\ell$, where $\ell = \Theta(\log(n))$. Recall $\ell = 2d_\ell^2 - 1$ is the total number of qubits used to construct the rotated surface code $\mathcal{RS}_\ell$. The justification for this choice of ℓ will follow in the next section.

To express n in terms of N, the following bounds will be useful:

$$n = O\left(\frac{N}{\log(N)}\right), \qquad n = \Omega\left(\frac{N}{\log^2(N)}\right). \tag{3.11}$$

It follows from the definition of a concatenated code that the number of encoded qubits is $K = k(n)$, the code distance is $D = d(n) \cdot d_\ell$ (See Section 3.2). As $\delta \geq 1/2$,

[8] Degree-8 can be achieved by replacing the $\mathcal{E}_R \times \mathcal{E}_R$ factor in the decomposition with a single expander graph after some modification of parameters.

$1 - 2\delta \leq 0$. Using Equation (3.11), we can write

$$K = \Omega\left(\frac{N}{\log^2(N)}\right) , \qquad D(N) = \Omega\left(N^{\delta}\ \log^{1-2\delta}\left[\frac{N}{\log(N)}\right]\right) . \tag{3.12}$$

This completes the proof. ∎

The second portion of Theorem 3.1.2, stated below, guarantees the existence of a syndrome-extraction circuit for the hierarchical code constructed in Theorem 3.4.1.

Theorem 3.4.2 (Theorem 3.1.2, Part 2)**.** *There exists an explicit and efficient construction of an associated family of syndrome-extraction circuits $C_N^{\mathcal{H}}$ using only local Clifford operations and* SWAP *gates of range R such that*

$$\mathcal{W}(C_N^{\mathcal{H}}) = \Theta(N) , \qquad \mathcal{T}(C_N^{\mathcal{H}}) = O\left(\frac{\sqrt{N}}{R}\right) .$$

The rest of this section is dedicated to the proof of Theorem 3.4.2. We construct the syndrome-extraction circuit $C_N^{\mathcal{H}}$ for the concatenated code with the stated parameters. This circuit is constructed in a bilayer architecture and is described in detail below. A bilayer construction of the syndrome-extraction circuit $C_n^{\mathcal{Q}}$ is described in Section 3.4. To obtain $C_N^{\mathcal{H}}$, each outer qubit in the syndrome-extraction circuit $C_n^{\mathcal{Q}}$ is replaced by a copy of the inner code $\mathcal{RS}_\ell$ as described in Section 3.2.

If $C_n^{\mathcal{Q}}$ requires $\mathcal{W} = \mathcal{W}(C_n^{\mathcal{Q}})$ qubits, then we need a layout for $\mathcal{W}$ surface codes in 2 dimensions. In Section 3.4, we propose an implementation of $\mathcal{RS}_\ell^{\otimes \mathcal{W}}$ using a bilayer 2-dimensional architecture. The advantage of this architecture is that entangling gates between codes can be performed in a transversal manner which reduces the number of extra ancilla qubits. In Section 3.4, we describe a novel implementation of SWAP gates for this architecture. This completes the set of logical Clifford operations $\mathcal{K}_1$. We will bring these elements together to construct the syndrome-extraction circuit $C_N^{\mathcal{H}}$ in Section 3.4.

Before doing so, we take a brief detour in Section 3.4 to design Level-1 qubits with noise bias. We will return to this construction in Section 3.6 to deal with hook errors.

Syndrome-extraction circuit $C_n^{\mathcal{Q}}$ for the outer code

In this section, we design a family of syndrome-extraction circuits $\{C_n^{\mathcal{Q}}\}$ for a constant-rate $[\![n, k, d, \Delta_q, \Delta_g]\!]$ LDPC code $\{\mathcal{Q}_n\}$. We assume $k = \rho \cdot n$ for $\rho > 0$ and that $m = n - k$ is the number of stabilizer generators. In Section 3.2, we described

measurement gadgets to measure each stabilizer generator. We now describe how these gadgets can be implemented in parallel subject to constraints on geometric locality. We first state the existence of an *ideal* circuit $(C_n^Q)^{\text{ideal}}$ which is not constrained by geometric locality. While we include the proof of this construction for the sake of completeness, we note that the idea and the proof itself have been used before—for example, see [DBT21]. For this reason, the proof is relegated to Appendix 3.10.

Define constants Δ and m_0 such that

$$\Delta := \max(\Delta_q, \Delta_g) \,, \qquad m_0 := \max(m_{\mathsf{X}}, m_{\mathsf{Z}}) \,.$$

The circuit $(C_n^Q)^{\text{ideal}}$ is divided into two phases, where in each phase we measure either X or Z syndromes. Each phase requires at most $(\Delta + 2)$ stages. It satisfies

$$\mathcal{W} := \mathcal{W}[(C_n^Q)^{\text{ideal}}] = n + m_0 \,, \qquad s := \mathcal{T}[(C_n^Q)^{\text{ideal}}] = 2(\Delta + 2) \,. \tag{3.13}$$

The first phase proceeds as follows:

1. All m_{X} ancilla qubits are prepared in the state $|+\rangle$.
2. In each intermediate step $1 < t \leq \Delta + 1$, there is a subset P_t of all $\mathcal{W}$ qubits such that P_t is a disjoint union of m_{X} pairs of qubits, where each pair contains one ancilla and one data qubit respectively. These pairs correspond to control and target qubits, respectively, for CNOT.
3. All m_{X} ancilla qubits are measured in the X basis.

The second phase is structurally similar with minor modifications:

1. All m_{Z} ancilla qubits are prepared in the state $|0\rangle$.
2. In each intermediate step $1 < t \leq \Delta + 1$, there is a subset P_t of all $\mathcal{W}$ qubits such that P_t is a disjoint union of m_{Z} pairs of qubits, where each pair contains one ancilla and one data qubit respectively. These pairs correspond to target and control qubits, respectively, for CNOT.
3. All m_{Z} ancilla qubits are measured in the Z basis.

We now use this circuit to construct the syndrome-extraction circuit C_n^Q that is constrained by geometric locality. It will have the same space footprint $\mathcal{W}$, but its depth will be different.

Setup: Qubits are arranged in two parallel layers where each layer is a grid of dimensions $L \times L$. We assume we have access to Clifford operations $\mathcal{K}$ where, in addition to nearest-neighbor gates between two qubits in the same layer, we can perform nearest-neighbor gates between two qubits that are adjacent but in different layers. We also assume that SWAP gates of range $R > 1$ are restricted to a single layer.

Initialization: To accommodate all $\mathcal{W}$ qubits required for $(C_n^Q)^{\text{ideal}}$, it is sufficient to choose the smallest integer L that satisfies $2L^2 \geq \mathcal{W}$. Initially, data qubits and syndrome qubits are distributed arbitrarily. While further optimization is likely possible, this will not affect the asymptotics, and certainly results in an upper bound on the circuit volume.

Partition C_n^Q into stages: The circuit C_n^Q will be partitioned into *s stages*, where in each stage, we prepare and measure ancilla qubits or simulate long-range entangling gates between pairs of qubits specified by P_t. To simulate a long-range entangling gate, we use a series of SWAP gates which bring each pair specified by P_t close together, followed by the desired entangling gate when they are sufficiently close. The preparation and measurement stages are straightforward. We now describe how to perform the long-range entangling gate.

Simulating long-range: In each simulation stage, qubits are arranged such that each pair of P_t are adjacent but in different layers. In the first step of each stage, we apply a permutation to the ancilla qubit in each pair of P_t to ensure that both qubits in the pair are not in the same layer. Qubits that are not in P_t remain stationary. For simplicity, we only permute qubits in the top layer and keep the bottom layer stationary. This specifies a permutation α_t on the top layer. As this layer is an $L \times L$ lattice, we can use Algorithm 4 to design a circuit so that SWAP operations can be performed in parallel. Using Theorem 3.1.1, we can construct a sparse spanning subgraph of $\mathrm{NN}_2(L, R)$ with vertex degree 12 such that any permutation α_t can be accomplished in depth at most $3L/R + O(\log^2(R))$. This is followed by nearest-neighbor entangling gates as specified by P_t. The simulation stages have depth

$$3\frac{L}{R} + O(\log^2(R)) + 1 \,. \tag{3.14}$$

Accounting for preparation and measurement steps in each phase, the circuit C_n^Q has parameters

$$\mathcal{W}(C_n^Q) = \mathcal{W} = n + m_0 \,, \qquad \mathcal{T}(C_n^Q) \leq 2\Delta \left(3\frac{L}{R} + O(\log^2(R)) + 1\right) + 4 \,. \quad (3.15)$$

We note that if $R = o(L)$, then the depth is $O(\sqrt{n}/R)$ (as $L = O(\sqrt{n})$. We shall assume that this is the case for the rest of the paper. We include the bound in Equation (3.15) with constants (ex. the 3 preceding L/R) and the dependence on the degree Δ to highlight that, for $R = 1$, this is not merely an asymptotic result and can actually be executed in practice.

The bounds in Equation (3.15) represent an achievability result—any quantum LDPC code can be simulated in depth $O(\sqrt{n}/R)$ as stated in the theorem above. However, it is not asymptotically tight for all code families (for example, consider the surface code). We expect future versions of this bound to depend on k and d, and how they scale as functions of n.

Circuit connectivity: In addition to providing a bound on the depth of permutations, Theorem 3.1.1 guarantees that each lattice position interacts with at most 12 other locations all within a range R. This implies that the connectivity of the circuit C_n^Q we have constructed can be 'static'—once qubits have been connected by wires of length at most R, we do not change it afterwards.

Implementation of the inner code

We have shown that C_n^Q is constructed using two parallel layers of qubits, where each layer is a lattice of dimensions $L \times L$. Here L is the smallest integer such that $2L^2 \geq \mathcal{W}$, where $\mathcal{W}$ is the number of qubits used by C_n^Q. To construct the syndrome-extraction circuit $C_N^{\mathcal{H}}$, we use two parallel *rotated* lattices. Each qubit in C_n^Q is replaced by a rotated surface code $\mathcal{RS}_\ell$ where $\ell = \Theta(\log(n))$. As each tile uses $\ell^2 = 2d_\ell^2 - 1$ physical qubits, the circuit $C_N^{\mathcal{H}}$ requires at least $2L^2 \cdot \ell^2$ physical qubits. We also use additional physical qubits which we refer to as *buffer* qubits to facilitate logical Clifford operations between tiles. See Figure 3.11.

Buffer qubits are either placed along the periphery of each lattice or between tiles:

1. First, we include a thin band of "buffer" qubits along the perimeter of each layer for reasons that we will explain shortly. The band has thickness $(\ell + 1)/2$ and therefore this adds at most $2L(\ell + 1)^2$ ancilla qubits per layer. These are denoted as transparent dots in Figure 3.11 (a).

2. Second, for each surface code, we have d_ℓ^2 data qubits, $d_\ell^2 - 1$ ancilla qubits, and 1 extra buffer qubit (light gray) for later convenience. These are denoted using dark gray, orange/blue and light gray respectively in Figure 3.11 (b).

In total, accounting for both qubits used in tiles as well as buffer qubits, we use at most $2(\ell+1)^2(L+1)^2$ physical qubits. These are arranged in two parallel (rotated) lattices of side length $(L+1)\cdot(\ell+1)$[9].

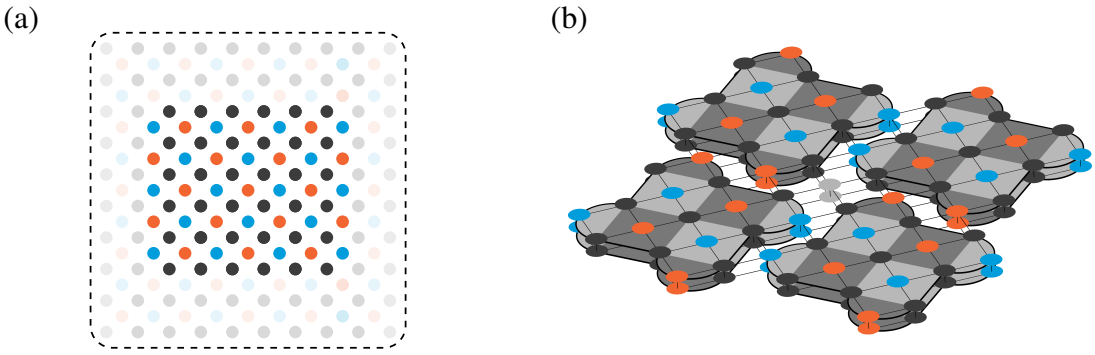


Figure 3.11: (a) A top-down view of one layer of the physical layout. At any given time, only some of the qubits are active; these are denoted using dots with solid color. In contrast, there are qubits that are inactive that can be used for performing logical operations; these are denoted using dots that are transparent. (b) A small $2\times2\times2$ unit cell of the physical layout containing 8 distance-3 rotated surface code tiles. Physical qubits are drawn as dark gray dots. X- and Z-type stabilizer generators within a tile are indicated by a light or dark gray region with the colored dot used as an ancilla. Thin lines indicate gate connectivity: Each qubit has 5 neighbors, 4 in-plane and 1 out-of-plane. The light gray qubit in the center is unused when the tiles are idle. Note that this layout does not contain additional ancilla qubits between tiles for lattice surgery: all operations will be performed transversally.

Physical gates can act either on two neighboring qubits in the same layer, or on adjacent qubits in different layers. Using only $\mathcal{K}_0$ operations, we construct the necessary primitives to implement the syndrome-extraction circuits $C_N^{\mathcal{H}}$.

As described in Section 3.4, the circuit $C_n^{\mathcal{Q}}$ is divided into s stages. Implementing the syndrome-extraction circuit for the outer code requires logical Clifford operations $\mathcal{K}_1$.

At the outset, both data and ancilla tiles are arranged arbitrarily. Syndromes are measured in two phases—first we measure the X-type syndromes and then the Z-type

[9]Note that there are $\sqrt{2}L\ell$ qubits to a side due to the rotated lattice.

syndromes. The first and last stages of each phase correspond to single-tile logical state preparation and single-tile logical measurements. Single-tile logical operations of state preparation and measurements can be done using only nearest-neighbor gates. If Level-0 qubits can be prepared in $|0\rangle$ or $|+\rangle$, then we can prepare Level-1 $\left|\overline{0}\right\rangle$ and $\left|\overline{+}\right\rangle$ simply by performing the syndrome-extraction circuit which projects the state into the code space. Similarly, we can perform destructive measurements of logical Pauli operators $\overline{\mathsf{X}}$ and $\overline{\mathsf{Z}}$ using single-qubit measurements of X and Z.

For each stage where we simulate long-range entangling gates, there exists a partition of outer qubits P_t; Here, P_t is the set of $m_0 = \max(m_{\mathsf{X}}, m_{\mathsf{Z}})$ pairs, where each pair has one outer ancilla qubit and one outer data qubit respectively. Depending on whether we are measuring X or Z syndromes, we perform the logical CNOT gate using the outer data qubit as target or the outer ancilla qubit as target. Data and syndrome tiles that are involved in entangling gates are always arranged such that the tiles are adjacent but in different layers. As the rotated surface code is a CSS code, we can perform CNOT gates between surface code blocks corresponding to data and ancilla qubits using transversal operations. For all these operations — single-tile preparation, single-tile measurement, and transversal CNOT — we perform surface code error correction after the operation.

To complete the description of the Level-1 syndrome extraction circuit, it only remains to explain how the logical SWAP operation is implemented. We propose a novel way to perform this gate when $R < \ell$; this is the focus of Section 3.4. We show that an arbitrary permutation requires $O(L \cdot d_\ell)$ steps. Error correction for SWAP gates is performed in an *interleaved* manner—we perform a single round of error correction after each step as described below.

We conclude Section 3.4 by discussing connectivity requirements. As mentioned in the introduction, we construct syndrome-extraction circuits such that once two lattice positions have been connected, this does not need to change dynamically over the course of the circuit. Furthermore, each lattice position only ever interacts with a constant-sized set of other lattice locations. For both error correction and logical operations, lattice positions (that store a physical qubit) will be involved in CNOT gates. It would be preferable if the pairs of lattice positions that need to interact did not change dynamically over the course of the circuit, and instead could be chosen ahead of time.

If the circuit C_n^Q is implemented such that each lattice position requires only connectivity to a constant sized set of other lattice positions, then the entire syndrome-

extraction circuit $C_N^{\mathcal{H}}$ for the hierarchical code $\mathcal{H}_N$ will only use sparse connectivity of two-qubit physical gates. The proof of this claim is straightforward for single-tile logical state preparation and measurement — these are accomplished using local physical operations. Secondly, by construction, logical entangling gates are implemented transversally and therefore the connectivity does not change. Finally, we will show in Section 3.4 that for a given L and R, the connectivity required to implement an arbitrary permutation of tiles will be chosen ahead of time and will not change dynamically.

SWAP Gate

As discussed above, we can perform logical CNOT transversally. To complete $\mathcal{K}_1$, the final ingredient we need are SWAP gates. We first focus on the special case $R = 1$, and then generalize the construction to arbitrary R. To implement the permutation returned by Algorithm 4, it suffices to perform SWAP gates only along one orientation of the lattice at a time, either vertically or horizontally. This restriction is utilized to create a resource efficient SWAP operation that requires no additional ancilla qubits. The key insight is that movement of individual tiles may be accomplished by moving *all* the tiles within a single layer. By performing the transversal SWAP after movement and then moving back, we can accomplish a SWAP operation between two surface codes that are not directly on top of each other.

Nearest-neighbor logical SWAP gates

High-level overview: We first provide a high-level overview of the SWAP operation and refer to Figure 3.12. Consider a two parallel rows of tiles in the bilayer architecture as shown in Figure 3.12 (a). The tiles in the top row are labeled a_1,..., a_4 and the tiles in the bottom row are labeled b_1,...,b_4. The tiles labeled $\varnothing$ are buffer qubits along the periphery. For ease of visualization, the picture depicts a single-tile width of buffer qubits along the top layer. In practice, we use two half-tile width of buffer qubits in both layers. In this example, we swap tiles a_2 and a_3; however, this process can be generalized to swap tiles in parallel. This is accomplished in 5 steps:

1. The logical SWAP operation begins by exchanging alternate tiles between two rows. In the bilayer architecture, this exchange is performed in a checkerboard pattern, i.e. we swap alternate tiles along both rows and columns. This can be accomplished using what we call the *staggered* SWAP *primitive*.

2. We can then slide the entire top layer one tile width to the left. In the bilayer architecture, this will be accomplished using what we call the *walking primitive* that we describe below. The top layer will shift a half-tile width in one direction while the bottom layer will move a half-tile width in the other. This is the reason we use buffer qubits along the periphery—to accommodate tiles after the walk step.

3. Pairs of tiles that we wish to exchange are now adjacent in different layers. For each pair that we wish to swap, we perform a swap operation using nearest-neighbor SWAP gates between adjacent layers.

4. The last two steps are the inverse of the first two steps—we apply the walk primitive and then perform a swap operation between layers on alternate tiles.

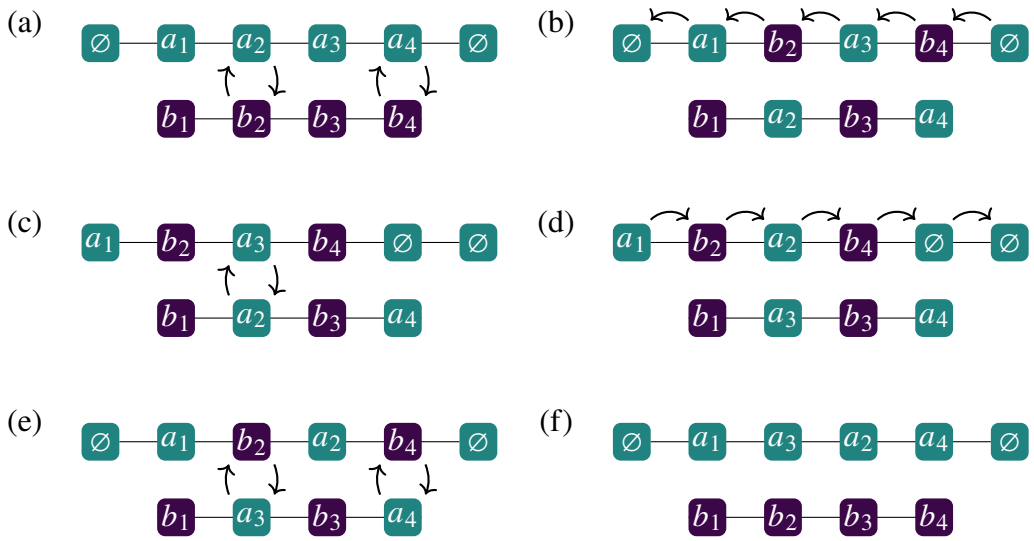


Figure 3.12: Consider two parallel rows of tiles $a_1, ..., a_4$ and $b_1, ..., b_4$ in different layers, one on top of another. Tiles are depicted as squares. The tiles with the label ∅ represent a tile width of buffer qubits on the periphery of the top layer. This example demonstrates how to swap two tiles a_2 and a_3. To begin, we swap alternate tiles in each row as shown in (a). We then use the walk operation to move tiles one unit as shown in (b). For every pair of tiles we wish to exchange, we perform a inter-layer SWAP as shown in (c). In this example, we only wish to swap tiles a_2 and a_3 so the other tiles remain stationary. We then undo the transformation by reversing the walk in (d) and undoing the alternate exchange in (e). The final panel (f) is the desired state.

Walking primitive: By placing a $1/2$-tile wide strip of buffer qubits on the periphery of the lattice, we can "walk" the entire memory by swapping the physical-level

ancilla qubits and surface code data qubits (Figure 3.14).[10] Using this walking primitive, we can move an entire layer an entire tile width in depth $2d_\ell$ using only SWAP-gates. This is a global operation. We will use this primitive in two ways: First, we implement a transversal SWAP between two tiles that are in different layers in the staggered-SWAP primitive (explained below). Second, we will use this repeatedly to move an entire layer half-a-tile width in some direction.

Staggered SWAP primitive: When possible, we would like to avoid applying gates directly between data qubits of surface code blocks as this would introduce (small) extra correlations in the logical failure probability of tiles. Instead of a direct transversal swap between data blocks, the vertical SWAP can instead be performed between data qubits in one layer and syndrome qubits in the other layer. This is accomplished via the staggered SWAP primitive. See Figure 3.13.

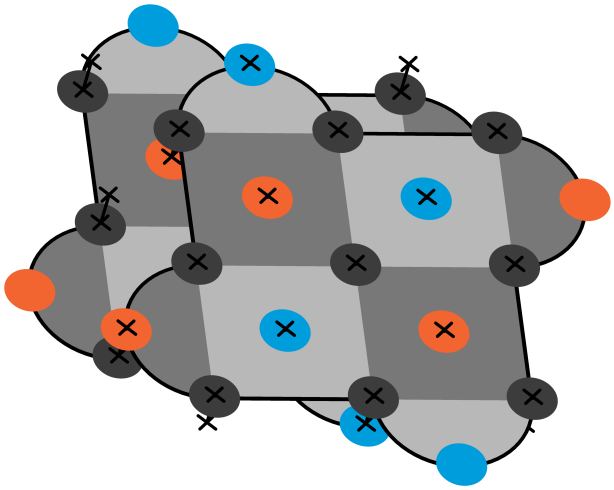


Figure 3.13: Performing a staggered SWAP operation between layers. Physical qubits are arranged such that data (ancilla) qubits in the top layer are adjacent to ancilla (data) qubits in the bottom layer. Each qubit in the top layer is swapped with the qubit immediately below it. This allows us to exchange tiles between layers without two data qubits directly interacting with each other.

By default, the qubits in the two layers are positioned such that a data qubit in the top layer is above a data qubit in the bottom layer. This facilitates performing logical CNOT gates via transversal physical CNOT gates. We can perform a stagger operation using the walking primitive. This positions data qubits in the top layer above ancilla qubits in the bottom layer. We can then apply a transversal SWAP between layers, and undo the stagger operation if need be. Over the course of the

[10] Not to be confused with the extra buffer qubit per tile.

Level-1 logical SWAP, however, we only need to undo the stagger operation at the very end. Throughout the logical SWAP, the two layers remain staggered. If syndrome qubits are reset before use in a syndrome-extraction round, the surface code blocks have undergone a somewhat complicated idle operation with no correlated errors generated between the two surface code blocks.

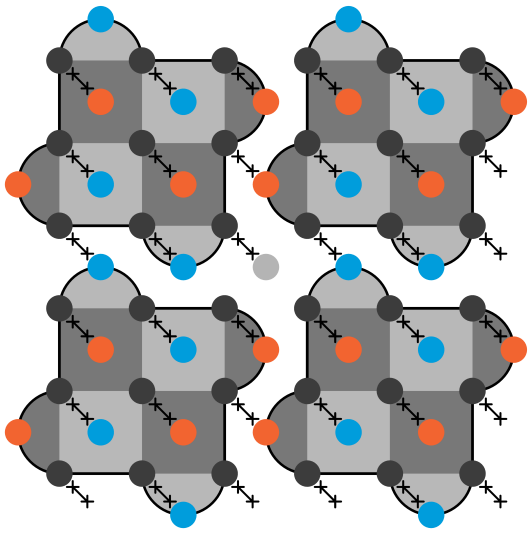


Figure 3.14: Walking primitive used in the SWAP implementation with qubits outside of the 2×2 unit cell not drawn. Swaps of Level-0 qubits are drawn as black lines with crosses. Data qubits become ancilla qubits and ancilla qubits become data qubits, so that syndrome extraction is possible at every step. A full 1-unit step to the right of the data qubits can be accomplished following this $1/2$ unit step by swapping each (initially) syndrome qubit with the data qubit up and to the right. Later, we will increase the speed at which the top and the bottom layers are shifted relative to each other by shifting the top layer in one direction and the bottom layer in the other.

Level-1 logical SWAP: We are ready to describe the logical SWAP operation.

We begin by exchanging every other tile. This procedure is illustrated in Figure 3.15; this can be compared to Figure 3.12. Steps 1 and 3 (half step shifts) are performed globally with step 2A (vertical swap) or 2B (half step shifts) performed whether or not a logical swap or logical identity is scheduled for a given tile. The step 2B is necessary for the logical identity gate, because step 2A would otherwise lead to data qubits of adjacent tiles directly next to each other instead of separated by an ancilla qubit. In this way, syndrome extraction can optionally be performed after every layer of SWAP gates.

To summarize, our SWAP-gate is implemented as follows where we account for the depth of each operation:

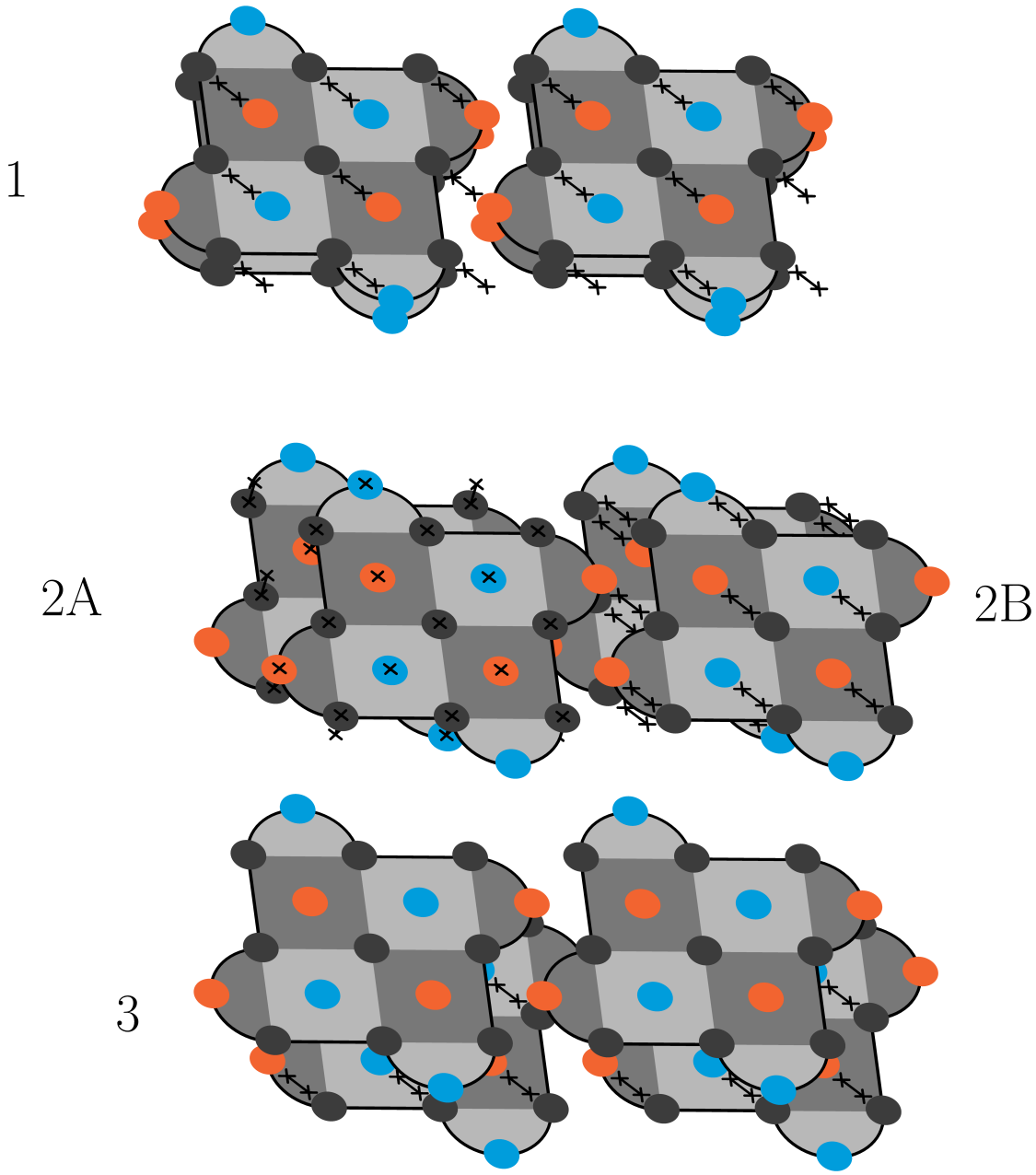


Figure 3.15: 3-step transversal swap implementation between two stacked syndrome tiles that avoids directly swapping data qubits. The pair of tiles on the right undergoes an identity operation while the pair of tiles on the left are swapped. In step 1, the top layer is shifted by a half-unit using SWAP gates on the top layer. In step 2, the pairs of tiles are either swapped using vertical SWAP gates (2A) or shifted using horizontal SWAP gates to keep alignment (2B). Finally, in step 3, the lower layer is shifted back by a half-unit using SWAP gates in the bottom layer. This operation has the property that syndrome extraction can be performed in all three timesteps. The perspective is inclined slightly to show both layers.

1. Use the staggered SWAP on every other tile in a checkerboard pattern to put them in different layers (depth-3).

2. Use the walking primitive to translate the top and bottom layers d_ℓ lattice sites in opposite directions so that originally adjacent tiles are now stacked (depth-d_ℓ).

3. Optionally perform a staggered SWAP (depth-3).

4. Translate the top and bottom layers d_ℓ lattice sites back (depth-d_ℓ)

5. Bring the tiles back to the same layer by undoing a staggered SWAP (depth-3).

At every step, we have the necessary ancilla qubits to perform surface code syndrome extraction. At first, we might think to perform d_ℓ rounds of error correction after each of the 5 steps above. However, because we are working with transversal SWAP operations, we only perform a *single* round of error correction after each step. Ideal SWAP gates do not spread errors, and therefore, the SWAP operation can be seen as a syndrome-extraction circuit on each tile with a higher failure rate. While performing just a single round of error correction may reduce the threshold, we expect this change to be minimal. Furthermore, it reduces the depth of the logical SWAP operation, so our tile SWAP-gate takes $2d_\ell + 9$ steps of physical SWAP-gates.

Recall from Section 3.4 that the syndrome-extraction circuit C_n^Q is split into $s = 2\Delta+4$ stages. Besides the preparation and measurement stages, we simulate a long-range CNOT between pairs of data and ancilla qubits in each stage. The stage begins by picking an element of each pair and ensuring that they are in different layers. We can then use the logical SWAP described here to permute tiles.

Note that the SWAP operations can be performed in parallel between tiles on the same layer; the SWAP operations exchange tiles in the same row or column as required by the routing algorithm presented in Algorithm 4. We can use Lemma 3.3.1 to show that any permutation of tiles on an $L \times L$ lattice can be accomplished in $3L - 3$ steps. This allows any desired permutation on the $L \times L \times 2$ lattice of tiles to be accomplished in depth $(2d_\ell + 9)(3L - 3)$.

In fact, we can optimize this further to avoid repeating redundant operations. For all but the first and last swap operations: 1) Steps 1 and 5 can be omitted. 2) Within step 3, we may omit the walking step that offsets the upper and lower layers by a half lattice site, so the staggered SWAP becomes a simple transversal swap. Using these

optimizations, any permutation on the $L \times L \times 2$ lattice of tiles can be accomplished in depth t_{route} where

$$t_{\text{route}} := (2d_\ell + 1)(3L - 3) + 8 \,. \tag{3.16}$$

Logical permutation routings

We restrict our attention to range R SWAP gates in a single layer. Interlayer operations are strictly nearest-neighbor gates, and can be accomplished using the primitives discussed in Section 3.4. In the following lemma, we show that an arbitrary permutation routing of tiles can be accomplished in depth $O(L\ell/R)$.

Lemma 3.4.3. *Consider access to physical* SWAP *operations with range* $R = o(L \cdot \ell)$. *We can implement any arbitrary permutation of Level-1 qubits in the bilayer architecture in depth*

$$O\left(L\ell/R\right).$$

Proof. We proceed in two cases, $R \geq \ell$ and $R < \ell$.

Case 1: $R \geq \ell$

Level-0 SWAP gates of range R can be used to implement Level-1 transversal gates of range $R_1 = \lfloor R/\ell \rfloor$. We can route on the Level-1 lattice $\mathrm{NN}_2(L, R_1)$ using Corollary 3.3.8 with Level-1 tiles swapped transversally. This guarantees that the depth of any permutation routing of tiles is $O(L/R_1)$. Each transversal SWAP is followed by a single round of syndrome extraction of the rotated surface code; this requires constant depth and does not affect the depth of permutation routing.

Case 2: $R < \ell$

The range R Level-0 SWAP gate can be used to speed up the walking primitive presented in Section 3.4. Parallelized Level-1 nearest-neighbor SWAP gates implemented in this way take time $\Theta(\ell/R)$. Combined with the Corollary 3.3.2, we have that routing takes time $O(L\ell/R)$. ∎

Biased-noise qubits

In Section 3.6, we will be interested in suppressing certain kinds of Pauli errors. When a qubit experiences X or Z errors with an asymmetric rate, it is said to be noise-biased. In this section, we will explain how to introduce such a noise bias on Level-1 qubits by modifying the bilayer architecture.

Let $\eta \geq 1$ be the desired noise bias of the Level-1 qubits, and suppose Z errors occur with a probability p and are η-times more likely to occur than X or Y errors. We can introduce a noise bias $\eta > 1$ on Level-1 qubits by elongating the surface code into rectangular regions where the minimum-weight X logical operator is longer than the minimum weight Z logical operator. See Figure 3.16. Suppose we have a rectangular surface code patch of dimensions d_X by d_Z such that the minimum weight X logical operator has weight d_X and the minimum weight Z logical operator has weight d_Z. Considering the minimum weight logical operators, we expect a failure rate $\propto p^{\lceil d_X/2 \rceil}$ in the X basis and $\propto p^{\lceil d_Z/2 \rceil}$ in the Z basis. By taking $d_X > d_Z$, we can introduce a noise bias.

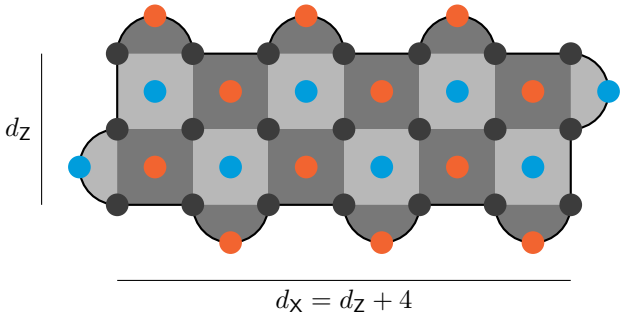


Figure 3.16: Creating a biased Level-1 qubit by using a rectangular surface code. In this picture $d_X = 7$ and $d_Z = 3$. The bias is therefore $\eta = O(p^{-2})$.

For simplicity, we assume $d_X = d_Z + \lceil \log(\eta)/\log(p) \rceil$ to guarantee a bias of *at least* η. We also assume that all Level-1 qubits, data and ancilla both, have been biased. We update the bilayer architecture to use two $L_X \times L_Z$ grids of rotated surface code tiles with X and Z distances d_X and d_Z respectively. Consider a constant-rate LDPC code Q_n with rate ρ. To accommodate $\mathcal{W}$ outer qubits, we let L be defined by $2L^2 = \mathcal{W}$. We then define L_Z and L_X to be the smallest integers satisfying $L_Z \geq L \cdot \sqrt{d_X/d_Z}$ and $L_X \geq L \cdot \sqrt{d_Z/d_X}$.

It is not sufficient that the qubits themselves are noise biased. In addition, we also require that all gates preserve this bias. We consider each Clifford operation in turn:

1. **Preparation & Measurement:** Within the syndrome-extraction circuit for the outer code where all measurement ancilla qubits are prepared in the $|\overline{+}\rangle$ state (Section 3.2), X errors on the ancilla qubits are suppressed by a factor of η.

2. **Entangling gates:** In the bilayer architecture, entangling gates are performed transversally. Transversal gates are naturally bias preserving.

3. SWAP **gates:** The way we perform SWAP operations does not change as all tiles have the same dimensions. The buffer region on the periphery of the lattice must be slightly increased to accommodate the elongated surface code tiles during the walking operation. SWAP operations themselves do not spread errors and are also bias preserving. [11]

Together, this completes the requirements for implementing logical Clifford operations $\mathcal{K}_1$.

Syndrome-extraction circuits for hierarchical codes

The syndrome-extraction circuit $C_N^{\mathcal{H}}$ for $\mathcal{H}_N$ is the circuit $C_n^{\mathcal{Q}}$ where we replace each outer qubit by a tile $\mathcal{RS}_\ell$. In the circuit $C_N^{\mathcal{H}}$, each gate of $C_n^{\mathcal{Q}}$ from the set $\mathcal{K}$ is replaced by the corresponding element in $\mathcal{K}_1$ followed by surface code error correction on each outer qubit. Recall that in Section 3.4, we discussed how to perform preparation, measurement and logical entangling gates. In Section 3.4, we discussed how to perform SWAP gates.

Theorem 3.4.4. *Each element $\mathcal{H}_N$ has an associated 2-dimensional syndrome-extraction circuit $C_N^{\mathcal{H}}$ with the following properties:*

$$\mathcal{W}(C_N^{\mathcal{H}}) = \Theta(N) \,, \qquad \mathcal{T}(C_N^{\mathcal{H}}) = O\left(\frac{\sqrt{N}}{R}\right) \,.$$

Further, each lattice position in $C_N^{\mathcal{H}}$ only interacts with a fixed set of other lattice positions whose size is independent of N.

Proof. Consider the family of $[\![n, k, d, \Delta_q, \Delta_g]\!]$ quantum LDPC codes $\{\mathcal{Q}_n\}$ of constant rate $\rho > 0$.

From Section 3.4, $\mathcal{W}(C_n^{\mathcal{Q}}) = \Theta(n)$. To construct $C_N^{\mathcal{H}}$, each qubit in $C_n^{\mathcal{Q}}$ is replaced by a surface code $\mathcal{RS}_\ell$. It follows that

$$\mathcal{W}(C_N^{\mathcal{H}}) = \Theta(\ell)^2 \cdot \Theta(n) = \Theta(N) \,. \tag{3.17}$$

Secondly, each $\mathcal{K}_1$ operation in $C_N^{\mathcal{H}}$ requires depth $\Theta(\ell)$ for error correction. This is because entangling gates are implemented transversally followed by d_ℓ rounds of error correction and, per Lemma 3.4.3, the Level-1 logical SWAP operation requires depth $O(L \cdot \ell/R) = O(\sqrt{N}/R)$.

[11] Recall that for two single qubit operators A and B, $\mathsf{SWAP}(A \otimes B) = (B \otimes A)\mathsf{SWAP}$.

This implies that

$$\mathcal{T}(C_N^{\mathcal{H}}) = O\left(\frac{\sqrt{N}}{R}\right) . \tag{3.18}$$

By assumption, logical two-tile operations in $\mathcal{K}_1$ can be implemented such that the set of lattice positions that interact with each other remain fixed. Furthermore, the construction of Section 3.4 guarantees that each of the outer positions in $C_N^{\mathcal{H}}$ only interact with a fixed and constant-sized set of other outer positions. Therefore, all of the physical positions of $C_N^{\mathcal{H}}$ need only interact with a fixed and constant-sized set of positions in $C_N^{\mathcal{H}}$.

This completes the proof. ■

3.5 Overhead, threshold and asymptotics

In this section, we prove that the hierarchical code has a threshold if we use the syndrome-extraction circuits $C_N^{\mathcal{H}}$ presented in Section 3.4. We present a formal version of Theorem 3.1.3.

We recall the bound by Delfosse *et al*. in Equation (3.1)

$$\mathcal{T}(C_n^{\mathcal{Q}}) = \Omega\left(\frac{n}{\sqrt{\mathcal{W}(C_n^{\mathcal{Q}})}}\right) . \tag{3.1}$$

According to this bound, the physical circuit $C_n^{\mathcal{Q}}$ cannot have constant depth and space footprints simultaneously. This blowup in the volume of the circuit introduces additional failure modes. Consequently, saturating these bounds by no means ensures the existence of a threshold. This then seems to have defeated the purpose of simulating a non-local circuit using local operations.

In this section, we present a geometrically-local construction of a circuit $C_N^{\mathcal{H}}$ that encodes a growing number of encoded qubits *and* guarantees that a threshold exists. We use code concatenation to define a family $\{\mathcal{H}_N\}$ that we call *hierarchical* codes. The N^{th} element of this family is obtained by concatenating an LDPC code $\mathcal{Q}_n$ and a rotated surface code $\mathcal{RS}_\ell$ of size $\Theta(\ell^2)$. Here $N = \Theta(n \cdot \ell^2)$.

In Section 3.5, we study the failure rate per round p_{round} and establish its dependence on p_{phys} for a syndrome-extraction circuit for any $[\![n, k, d, \Delta_q, \Delta_g]\!]$ quantum LDPC code. The circuits are themselves faulty and are described by a locally decaying faults model. We show in Section 3.5 that logical gates in the bilayer architecture guarantee

that Level-1 logical errors in surface code blocks are suppressed exponentially following logical Clifford operations. This allows us to deal with the Level-1 syndrome-extraction circuit for the outer code directly without having to keep track of Level-0 failure probabilities. Permuting tiles can introduce Level-1 correlated errors among the tiles. In Section 3.5, we show that there exists a choice of ℓ such that the $p^{(1)}_{\text{round}}$ is an arbitrarily small constant. We can then invoke Gottesman's threshold theorem which we discussed in Section 3.2 to prove the existence of a threshold.

We conclude this overview by highlighting some features of this construction.

1. We show that $\ell = \Theta(\log(n))$ is sufficient to achieve a threshold for the outer code. If the code $\mathcal{Q}_n$ has constant rate, then the code $\mathcal{H}_N$ has rate $O(\log(n)^{-2}) \to 0$ as $n \to \infty$.

2. Although outer and inner codes are both LDPC codes, $\mathcal{H}_N$ is itself no longer an LDPC code—it uses stabilizer measurements that have weight $\log(n)$ as the side length of the inner surface codes is $\ell = \Theta(\log(n))$. However, the logical operators of the surface code can be measured (destructively) using only single-qubit measurements.

3. There is a cost to locality—the sub-threshold scaling of the logical failure rate is qualitatively different from the typical exponential error suppression as a function of the distance. Instead, we see a strictly sub-exponential, but still superpolynomial, suppression of the logical failure rate with the distance.

Evolution of errors in syndrome-extraction circuit $C^{\mathcal{Q}}_n$

Consider an $[\![n, k, d, \Delta_q, \Delta_g]\!]$ LDPC code family $\{\mathcal{Q}_n\}$ with constant rate, i.e. $k = \rho \cdot n$ for some constant $\rho \in (0, 1)$ and distance $d = \Theta(n^\delta)$ for some constant $\delta \in (0, 1]$. In Section 3.2, we discussed how Gottesman's proof for the existence of a threshold for LDPC codes depends on the number of errors *per round* of syndrome extraction on both data and ancilla qubits. In this section, we shall study how the probability of errors per round depends on the circuit $C^{\mathcal{Q}}_n$.

Recall the circuit $C^{\mathcal{Q}}_n$ described in Section 3.4. Qubits are arranged on two parallel lattices where each lattice has dimensions $L \times L$. Here, L is the smallest natural number that satisfies $2L^2 \geq \mathcal{W}$. In total the circuit has $s = 2\Delta + 4$ stages which can be broken down as follows. The syndrome-measurement circuit $C^{\mathcal{Q}}_n$ is divided into two phases, one for each of X- and Z-type syndrome measurements. Each phase is further

divided into $\Delta + 2$ stages, where $\Delta = \max(\Delta_q, \Delta_g)$. In addition to one stage each to prepare and measure ancillas, there are Δ stages where we simulate long-range entangling gates. In each such stage, we permute qubits in the lattice using SWAP gates. Given access to SWAP operations of range R, the permutation has bounded depth $\mathcal{T}_{\text{perm}} = O(L/R)$. This is then followed by nearest-neighbor entangling gates.

Our first result is technical and allows one to compose locally decaying distributions.

Lemma 3.5.1. *Let* $\Pr_1$ *and* $\Pr_2$ *be two independent and locally decaying distributions on* $[n]$ *with rates* p_1 *and* p_2*. Consider the distribution* $\widehat{\Pr} : \mathrm{Pow}(n) \to \mathbb{R}$ *defined as*

$$\widehat{\Pr}(E) = \sum_{\substack{E_1, E_2 \subseteq [n] \\ E \subseteq E_1 \cup E_2}} \widehat{\Pr}_2(E_1) \cdot \widehat{\Pr}_1(E_2) \,. \tag{3.19}$$

Then Pr *is a locally decaying distribution with rate* $p = p_1 + p_2$*, i.e. for all* $E \subseteq [n]$*,*

$$\Pr(E) \leq (p_1 + p_2)^{|E|} \,. \tag{3.20}$$

Proof. For $E \subseteq [n]$, we can write

$$\begin{aligned}
\Pr(E) &= \sum_{\substack{E_1, E_2 \subseteq [n] \\ E \subseteq E_1 \cup E_2}} \widehat{\Pr}_1(E_1) \widehat{\Pr}_2(E_2) && (3.21)\\
&\leq \sum_{\substack{E_1, E_2 \subseteq E \\ E = E_1 \sqcup E_2}} \Pr_1(E_1) \Pr_2(E_2) && (3.22)\\
&\leq \sum_{w=0}^{|E|} \binom{|E|}{w} p_1^w p_2^{|E|-w} && (3.23)\\
&= (p_1 + p_2)^{|E|} && (3.24)
\end{aligned}$$

The result follows. ∎

Lemma 3.5.2. *Let* $\mathbf{e} \in \mathbb{F}_2^n$ *be a random binary vector such that* $E = \mathrm{supp}(\mathbf{e})$ *is distributed according to a locally decaying distribution with rate* p*. Let* $\mathbf{M} \in \mathbb{F}_2^{m \times n}$ *with row and column weight at most* Δ *and* $\mathbf{f} = \mathbf{M} \cdot \mathbf{e}$ *be a random variable induced from* $\mathbf{e}$*. Then* $F = \mathrm{supp}(\mathbf{f})$ *is distributed according to a* $2^{\Delta} p^{1/\Delta}$ *locally decaying distribution.*

Proof. For a set of bits $A \subseteq [n]$, denote its image under $\mathbf{M}$ by $\mathbf{M}(A)$ defined as the union of columns $\mathbf{M}_i$ of $\mathbf{M}$ in A. I.e. $\mathbf{M}(A) := \bigcup_{i \in A} \mathrm{supp}\, \mathbf{M}_i$. The probability that

an error set $F \subseteq [m]$ on the output occurs

$$\Pr(F) = \sum_{\substack{E \subseteq [n] \\ F \subseteq \mathbf{M}(E)}} \widehat{\Pr}(E) \,, \tag{3.25}$$

i.e. in order for F to have occurred, there must be a set of errors on the input such that F is in the image. We can rewrite this sum in terms of the *largest subset* of the powerset $\mathcal{I} \subseteq \mathrm{Pow}(n)$ such that for any single set $E \in \mathcal{I}$:

1. We have that $F \subseteq \mathbf{M}(E)$.
2. For all non-empty subsets $G \subset E$, $F \nsubseteq \mathbf{M}(E \setminus G)$.

Each element of $\mathcal{I}$ is minimal in the sense that it is a subset of no other element of $\mathcal{I}$ (Assumption 2) while still having F in its image (Assumption 1). The second condition will allow us to replace the $\widehat{\Pr}(\cdot)$ with $\Pr(\cdot)$ in the sum without loosening the upper bound. Additionally, the column weight of $\mathbf{M}$ is at most Δ, so the size of each element $E \in \mathcal{I}$ is at least $|F|/\Delta$. Using the locally decaying distribution assumption yields

$$\begin{aligned} \Pr(F) &\leq \sum_{E \in \mathcal{I}} \Pr(E) && (3.26) \\ &\leq |\mathcal{I}| \cdot p^{|F|/\Delta} \,. && (3.27) \end{aligned}$$

It now remains to count the number of elements of $\mathcal{I}$: Let $J \subseteq [m]$ be the preimage of F in the sense that for every element e in J, the intersection of $\mathbf{M}(\{e\})$ with F is not empty. The row weight of $\mathbf{M}$ is at most Δ, so J is no larger than $|F|\Delta$. Every set E in $\mathcal{I}$ must satisfy $E \subseteq J$ or else there would be some element $a \in E$ such that $F \subseteq \mathbf{M}(E \setminus \{a\})$ (contradicting Assumption 2 on $\mathcal{I}$). Finally, there are $2^{|J|}$ subsets of J, so $|\mathcal{I}| \leq 2^{|J|} \leq 2^{|F|\Delta}$. Continuing with the bound, we have that

$$\begin{aligned} \Pr(F) &\leq |\mathcal{I}| \cdot p^{|F|/\Delta} && (3.28) \\ &\leq 2^{|F|\Delta} \cdot p^{|F|/\Delta} && (3.29) \\ &= \left(2^{\Delta} \cdot p^{1/\Delta}\right)^{|F|} \,. && (3.30) \end{aligned}$$

The result follows. ∎

The syndrome-extraction circuit C_n^Q has a special structure—errors do not spread from one data qubit to another or from one ancilla qubit to another. We show that

this implies that D and A are distributed according to a locally decaying distribution. Before doing so, we review the *symplectic representation* formalism which we use in the following proofs.

The symplectic representation: For $\mathcal{W} \in \mathbb{N}$, consider any Pauli operator $\mathsf{P} \in \mathcal{P}_{\mathcal{W}}$ and suppose it is expressed as $\mathsf{P} = \mathsf{X}(\mathbf{p}_x)\mathsf{Z}(\mathbf{p}_z)$ for $\mathbf{p}_x, \mathbf{p}_z \in \mathbb{F}_2^{\mathcal{W}\times 1}$. Clifford unitary operators U map Pauli operators to Pauli operators under conjugation, i.e. $U\mathsf{P}U^\dagger$ is a Pauli operator. Equivalently, this can be represented as a linear map on $\mathbf{p}_x$ and $\mathbf{p}_z$. Corresponding to U, there exists a matrix $\mathbf{M} \in \mathbb{F}_2^{2\mathcal{W}\times 2\mathcal{W}}$[12] such that the action of U on P can equivalently can be expressed as

$$\begin{pmatrix} \mathbf{p}_x \\ \mathbf{p}_z \end{pmatrix} \to \mathbf{M} \cdot \begin{pmatrix} \mathbf{p}_x \\ \mathbf{p}_z \end{pmatrix} \pmod 2 \ . \tag{3.31}$$

All arithmetic on symplectic vectors is performed modulo 2; we drop the 'mod 2' suffix in the equations that follow.

Recall that the $\mathcal{W} = \mathcal{W}(C_n^Q)$ qubits in C_n^Q are partitioned into data qubits and ancilla qubits respectively. Controlled-P gates for $\mathsf{P} \in \mathcal{P}$ only use the ancilla qubits as control and data qubits as target. Let $\mathbf{d}_x, \mathbf{d}_z \in \mathbb{F}_2^{n\times 1}$ and $\mathbf{a}_x, \mathbf{a}_z \in \mathbb{F}_2^{m_0\times 1}$ represent the Pauli operators D and A on data and ancilla qubits respectively. For the purposes of understanding how errors accumulate over one round of syndrome measurements, we are not interested in the physical locations of the qubits. As far as their action on D and A are concerned, we treat SWAP gates as (noisy) idle gates [13].

In any given timestep of C_n^Q where we apply entangling gates, all qubits interact with the same type of gate (CNOT or CZ) or remain idle. The corresponding symplectic matrices have a very special form. We can write the joint evolution of D and A under the Clifford transformation acting on the X- and Z-components separately:

1. If qubits are only involved in CNOT operations that use ancilla qubits as control qubits and data qubits as target qubits, then there exists a matrix $\mathbf{M} \in \mathbb{F}_2^{m_\mathsf{X}\times n}$ such that

$$\begin{pmatrix} \mathbf{d}_x \\ \mathbf{a}_x \end{pmatrix} \to \begin{pmatrix} (\mathbf{M})^T \cdot \mathbf{a}_x + \mathbf{d}_x \\ \mathbf{a}_x \end{pmatrix} , \qquad \begin{pmatrix} \mathbf{d}_z \\ \mathbf{a}_z \end{pmatrix} \to \begin{pmatrix} \mathbf{d}_z \\ \mathbf{a}_z + \mathbf{M} \cdot \mathbf{d}_z \end{pmatrix} . \tag{3.32}$$

[12] The matrix $\mathbf{M}$ has additional structure—it is symplectic [Got97], but this is not relevant for this proof.

[13] Colloquially, SWAP gates change the locations of qubits in physical space, not in 'math' space. For instance, suppose each qubit has a label $1, ..., n$ and we choose to represent the vector $\mathbf{p}_x$ as $(\mathbf{p}_x(1), ..., \mathbf{p}_x(n))$, where $\mathbf{p}_x(i)$ represents the Pauli operator on the i^{th} qubit. Then moving qubits around in physical space using SWAP gates does not affect the i^{th} component $\mathbf{p}_x(i)$. For this reason, we ignore the action of SWAP on $\mathbf{p}_x$ and $\mathbf{p}_z$.

For every pair of qubits indexed by $i \in [m_0]$ and $j \in [n]$ that are the control and target of a CNOT, the (i, j) entry of $\mathbf{M}$ is 1. The other entries are 0. In this setting, we note that $\mathbf{a}_x$ and $\mathbf{d}_z$ remain invariant.

2. If qubits are only involved in CZ operations that use ancilla qubits as control qubits and data qubits as target qubits, then there exists a matrix $\mathbf{N} \in \mathbb{F}_2^{m_Z \times n}$ such that

$$\begin{pmatrix} \mathbf{d}_x \\ \mathbf{a}_x \end{pmatrix} \to \begin{pmatrix} \mathbf{d}_x \\ \mathbf{a}_x \end{pmatrix}, \qquad \begin{pmatrix} \mathbf{d}_z \\ \mathbf{a}_z \end{pmatrix} \to \begin{pmatrix} \mathbf{d}_z + \mathbf{N}^T \cdot \mathbf{a}_x \\ \mathbf{a}_z + \mathbf{N} \cdot \mathbf{d}_x \end{pmatrix}. \tag{3.33}$$

For every pair of qubits indexed by $i \in [m_Z]$ and $j \in [n]$ that are the control and target of a CZ, the (i, j) entry of $\mathbf{N}$ is 1. The other entries are 0. In this setting, we note that $\mathbf{d}_x$ and $\mathbf{a}_x$ remain invariant.

In the symplectic representation, we can see that the structure of a syndrome-extraction circuit is special because in each phase where we measure either X or Z syndromes, there is always an invariant subspace (for example, $\mathbf{d}_x$, $\mathbf{a}_z$ when measuring X-type syndromes).

The induced error model: In the symplectic representation, a faulty Clifford operation can be expressed as an affine map—there exists random variables $\mathbf{b}_x$, $\mathbf{b}_z \in \mathbb{F}_2^{W \times 1}$ such that the noisy operation can be expressed as

$$\begin{pmatrix} \mathbf{q}_x \\ \mathbf{q}_z \end{pmatrix} = \mathbf{M} \cdot \begin{pmatrix} \mathbf{p}_x \\ \mathbf{p}_z \end{pmatrix} + \begin{pmatrix} \mathbf{b}_x \\ \mathbf{b}_z \end{pmatrix}. \tag{3.34}$$

The errors $\mathbf{b}_x$, $\mathbf{b}_z$ are caused by faults. The faults are themselves are distributed according to the locally decaying distribution $\mathcal{F}$ with failure rate p_{phys}. Let $\mathcal{X}$, $\mathcal{Z}$ be the induced distributions over $\mathbf{b}_x$ and $\mathbf{b}_z$. For example, $\mathcal{X}(\mathbf{b}_x)$ represents the sum of the probabilities over all events where the error is $(\mathbf{b}'_x, \mathbf{b}'_z)$ such that $\text{supp}(\mathbf{b}_x) \subseteq \text{supp}(\mathbf{b}'_x)$. In other words, it represents the total probability that the error has a non-trivial X component on $\text{supp}(\mathbf{b}_x)$.

When the circuit C is composed of elements from $\mathcal{K}$, we can say more about the induced distributions $\mathcal{X}$ and $\mathcal{Z}$.

Lemma 3.5.3. *Consider a Clifford circuit of depth* 1 *composed of elements from* $\mathcal{K}$. *The induced total probabilities* $\mathcal{X}$, $\mathcal{Z}$ *are locally decaying distributions with failure rate* $\sqrt{p_{\text{phys}}}$.

Proof. We shall prove this statement for the distribution $\mathcal{X}$; the proof for the distribution $\mathcal{Z}$ is identical. Fix an arbitrary vector $\mathbf{b}_x \in \{0,1\}^{\mathcal{W}}$.

Suppose a fault F results in some error $\mathbf{b}'_x$ such that $\mathrm{supp}(\mathbf{b}'_x) \supseteq \mathrm{supp}(\mathbf{b}_x)$. This implies that F must obey $\mathrm{supp}(F) \supseteq \mathrm{supp}(\mathbf{b}_x)$. Let F be the smallest set of fault locations such that $\mathrm{supp}(F) \supseteq \mathrm{supp}(\mathbf{b}_x)$. Because the circuit C has depth 1 and is composed entirely of only 1- and 2-qubit gates, this implies that $|F| \leq |\mathbf{b}_x| \leq 2|F|$.

By definition, the total probability of the fault F is $\mathcal{F}(F)$ a locally decaying distribution with failure rate p_{phys}.

$$\mathcal{X}(\mathbf{b}_x) \leq \mathcal{F}(F) \leq (p_{\mathrm{phys}})^{|F|} \tag{3.35}$$

$$\leq (\sqrt{p_{\mathrm{phys}}})^{|\mathbf{b}_x|} \,. \tag{3.36}$$

The result follows. ∎

We are now ready to study p_{round} and its dependence on C_n^Q. To set the stage, we first consider ideal syndrome extraction in the absence of circuit faults. We focus our attention on the extraction of X-type syndromes and note that the analysis for the Z-type syndromes is identical.

Consider a corrupted code state $\mathsf{E}\,|\psi\rangle$ where ψ is a code state and $\mathsf{E} = \mathsf{X}(\mathbf{e}_x)\mathsf{Z}(\mathbf{e}_z)$ is some Pauli operator. If the syndrome-extraction circuit C_n^Q has no faults, the joint state of the data and ancilla qubits after the circuit is described by[14]

$$\mathsf{E}\,|\psi\rangle \otimes \mathsf{Z}(\boldsymbol{\sigma}_{\mathsf{X}})\,|+\rangle^{\otimes m_{\mathsf{X}}} \,, \tag{3.37}$$

where $\boldsymbol{\sigma}_{\mathsf{X}}$ represent the ideal syndromes for X-type stabilizer generators.

In this setting, we can use Equation (3.32) to update X- and Z-components of Pauli operators under the action of CNOT. Initially, the X and Z components of the state $\mathsf{E}\,|\psi\rangle \otimes |+\rangle^{\otimes m_{\mathsf{X}}}$ can be expressed as $(\mathbf{e}_x|\mathbf{0})$, $(\mathbf{e}_z|\mathbf{0})$, where $\mathbf{0}$ is the all zeros vector of length m_{X}. The vector $\mathbf{0}$ means that we assume that the input to the circuit is the state $\mathsf{E}\,|\psi\rangle \otimes |+\rangle^{\otimes m_{\mathsf{X}}}$; preparation faults on the ancilla occur in the first timestep. For $1 < t < \Delta + 2$, we apply CNOT gates specified by a matrix $\mathbf{M}^{(t)} \in \mathbb{F}_2^{m_{\mathsf{X}} \times n}$. If we do not apply an entangling gate (i.e. when we SWAP qubits), then $\mathbf{M}^{(t)}$ is a matrix of zeros. Otherwise, the (i,j) entry of $\mathbf{M}^{(t)}$ is 1 if and only if the i^{th} syndrome qubit and the j^{th} data qubit are involved in a CNOT gate in the t^{th} timestep. In the absence of circuit faults, the X components of the error $(\mathbf{e}_x|\mathbf{0})$ are left unaffected during the

[14] The ancillas are disentangled from the data block by measurement.

phase where we measure X-type syndromes. On the other hand, the Z-components transform as

$$\begin{pmatrix} \mathbf{e}_z \\ \mathbf{0} \end{pmatrix} \mapsto \begin{pmatrix} \mathbf{e}_z \\ \sum_t \mathbf{M}^{(t)} \cdot \mathbf{e}_z \end{pmatrix} . \tag{3.38}$$

The vector $\sum_t \mathbf{M}^{(t)} \cdot \mathbf{e}_z$ is the X-type syndrome $\boldsymbol{\sigma}_{\mathsf{X}}$. In other words, $\sum_t \mathbf{M}^{(t)} =: \mathbf{H}_{\mathsf{X}}$ is the symplectic representation of the X-type stabilizer generators. Note that $\mathbf{H}_{\mathsf{X}}$ is a sparse matrix with at most Δ_q ones per row and Δ_g ones per column.

Next, we move on to the setting where circuit components are faulty. The final state of the data and ancilla qubits is different from Equation (3.37) because of circuit faults. We express it as

$$(\mathsf{D} \otimes \mathsf{A}) \left(\mathsf{E} \left| \psi \right\rangle \otimes \mathsf{Z}(\boldsymbol{\sigma}_{\mathsf{X}}) \left| + \right\rangle^{\otimes m_{\mathsf{X}}} \right) , \tag{3.39}$$

where, D and A represent errors due to faults in the circuit C_n^Q on the data qubits and ancilla qubits respectively. The symplectic representation of the final Pauli operator on the data and ancilla qubits is

$$\begin{pmatrix} \mathbf{e}_x + \mathbf{d}_x \\ \mathbf{a}_x \end{pmatrix} , \qquad \begin{pmatrix} \mathbf{e}_z + \mathbf{d}_z \\ \boldsymbol{\sigma}_{\mathsf{X}} + \mathbf{a}_z \end{pmatrix} . \tag{3.40}$$

The probability of errors per round, p_{round}, is the maximum failure rate for the distributions describing $\mathbf{d}_x, \mathbf{d}_z, \mathbf{a}_x$, and $\mathbf{a}_z$.

Theorem 3.5.4. *The induced distributions $\mathcal{X}$ and $\mathcal{Z}$ that govern the errors* $\mathsf{D} \otimes \mathsf{A}$ *are locally decaying distributions with failure rate p_{round}, where*

$$p_{\text{round}} \leq 2^{\Delta+1} \cdot \mathcal{T}(C_n^Q) \cdot (p_{\text{phys}})^{1/(2\Delta+2)} ,$$

where $\Delta = \max(\Delta_q, \Delta_g)$ is the number of stages in the circuit C_n^Q.

Proof. Recall that the circuit C_n^Q proceeds in two phases, with the first phase used to measure X-type syndromes and the second phase used to measure Z-type syndromes. For brevity, we allow $\mathcal{T}_{\mathsf{X}}$ and $\mathcal{T}_{\mathsf{Z}}$ to be the depth of the circuit C_n^Q corresponding to each phase; this means $\mathcal{T}(C_n^Q) = \mathcal{T}_{\mathsf{X}} + \mathcal{T}_{\mathsf{Z}}$. Here, we focus on the first phase of C_n^Q which is used to measure X-type syndromes and study the evolution of Z-type errors; the proof of the remaining three cases is identical and for this reason we omit them.

Let $\mathbf{b}_x^{(t)}, \mathbf{b}_z^{(t)} \in \{0,1\}^n$ and $\mathbf{c}_x^{(t)}, \mathbf{c}_z^{(t)} \in \{0,1\}^{m_{\mathsf{X}}}$ be the errors on data and ancilla qubits induced by faults caused at time t. In turn, these errors can spread to other

qubits and interact with errors at later times. Using Equation (3.32) repeatedly, we can write the final error $\mathbf{e}_z + \mathbf{d}_z, \boldsymbol{\sigma}_{\mathsf{X}} + \mathbf{a}_z$ in terms of the errors at each step as follows:

$$\begin{pmatrix} \mathbf{e}_z + \mathbf{d}_z \\ \boldsymbol{\sigma}_{\mathsf{X}} + \mathbf{a}_z \end{pmatrix} = \begin{pmatrix} \mathbf{e}_z + \sum_t \mathbf{b}_z^{(t)} \\ \sum_t \mathbf{M}^{(t)} \cdot \mathbf{e}_z + \sum_t \mathbf{M}^{(t)} \cdot \sum_{t'<t} \mathbf{b}_z^{(t')} + \sum_t \mathbf{c}_z^{(t)} \end{pmatrix} . \tag{3.41}$$

As we are only measuring X-type syndromes, all sums are over timesteps t in the first phase of the circuit. For timesteps t where we do not apply a CNOT, all entries of $\mathbf{M}^{(t)}$ are 0.

We can simplify Equation (3.41) by eliminating $\mathbf{e}_z$ and $\boldsymbol{\sigma}_{\mathsf{X}} = \sum_t \mathbf{M}^{(t)} \cdot \mathbf{e}_z$:

$$\begin{pmatrix} \mathbf{d}_z \\ \mathbf{a}_z \end{pmatrix} = \begin{pmatrix} \sum_t \mathbf{b}_z^{(t)} \\ \sum_t \mathbf{M}^{(t)} \cdot \sum_{t'<t} \mathbf{b}_z^{(t')} + \sum_t \mathbf{c}_z^{(t)} \end{pmatrix} . \tag{3.42}$$

While it is a straightforward consequence of the linear evolution under symplectic transformations, being able to write $\mathbf{d}_z$ and $\mathbf{a}_z$ without $\mathbf{e}_x$ and $\mathbf{e}_z$ means that the Z-components of the errors $\mathbf{d}_z$ and $\mathbf{a}_z$ do not depend on the input error E.

Furthermore, the special structure of the syndrome-extraction circuit is reflected here — $\mathbf{d}_z$ is simply the sum of the errors $\mathbf{b}_z^{(t)}$ caused by faulty gates at each step. In other words, Z errors on data qubits are not affected by Z errors on ancilla qubits.

We simplify this further using two observations. First, we will find it useful to reorder the sums within this equation as follows:

$$\sum_t \mathbf{M}^{(t)} \cdot \sum_{t'<t} \mathbf{b}_z^{(t')} = \sum_t \sum_{t'} \mathbb{1}[t' < t]\ \mathbf{M}^{(t)} \cdot \mathbf{b}_z^{(t')} = \sum_{t'} \left(\sum_{t>t'} \mathbf{M}^{(t)} \right) \cdot \mathbf{b}_z^{(t')} . \tag{3.43}$$

where $\mathbb{1}[t' < t]$ is the indicator function, i.e. it is 1 when $t' < t$ and 0 otherwise. The terms on the right-hand sides of these equations have a natural interpretation — for example, the error $\mathbf{b}_z^{(t')}$ that occurs on data qubits at time t' can propagate to ancilla qubits at times $t > t'$.

Second, it is difficult to directly deal with sums of random vectors modulo 2 that appear in Equation (3.42). Instead, we re-write Equation (3.42) in terms of the *support* of the vectors. To this end, we note that

$$\operatorname{supp}\left(\sum_{t'>t} \mathbf{M}^{(t')} \right) \subseteq \operatorname{supp}\left(\sum_t \mathbf{M}^{(t)} \right) = \operatorname{supp}(\mathbf{H}_{\mathsf{X}}) . \tag{3.44}$$

Together, these observations mean we can rewrite Equation (3.42) as

$$\operatorname{supp}\begin{pmatrix}\mathbf{d}_z\\ \mathbf{a}_z\end{pmatrix} \subseteq \operatorname{supp}\begin{pmatrix}\sum_t \mathbf{b}_z^{(t)}\\ \mathbf{H}_\mathsf{X}\cdot\sum_t \mathbf{b}_z^{(t)} + \sum_t \mathbf{c}_z^{(t)}\end{pmatrix} \tag{3.45}$$

$$\subseteq \bigcup_t \operatorname{supp}\begin{pmatrix}\mathbf{b}_z^{(t)}\\ \mathbf{H}_\mathsf{X}\cdot\mathbf{b}_z^{(t)} + \mathbf{c}_z^{(t)}\end{pmatrix} \tag{3.46}$$

$$\subseteq \bigcup_t \operatorname{supp}\left[\begin{pmatrix}\mathbf{I}_n & \mathbf{0}\\ \mathbf{H}_\mathsf{X} & \mathbf{I}_{m_\mathsf{X}}\end{pmatrix}\begin{pmatrix}\mathbf{b}_z^{(t)}\\ \mathbf{c}_z^{(t)}\end{pmatrix}\right] . \tag{3.47}$$

where $\mathbf{I}_n$ and $\mathbf{I}_{m_\mathsf{X}}$ are identity matrices of dimensions n and m_X respectively. We pause to explain the two simplifications in words. Substituting $\sum_{t'>t}\mathbf{M}^{(t')}$ with $\mathbf{H}_\mathsf{X}$ corresponds to a worst-case setting—an error on an ancilla qubit can propagate to *all* data qubits in its support *regardless* of when the error on the ancilla qubit occurs. Second, by dealing with the union of the supports of the vectors instead of the vectors themselves, we upper bound the maximum size of the final error. Evaluating the probability of this event allows us to upper bound the probability of a final error $\mathsf{D}\otimes\mathsf{A}$.

We can bound the probabilities of the terms in Equation (3.47). The errors at time t,

$$\begin{pmatrix}\mathbf{b}_z^{(t)}\\ \mathbf{c}_z^{(t)}\end{pmatrix} ,$$

are independent of errors occurring at $t' \neq t$ — induced errors at different timesteps are independent because faults occurring at different timesteps are independent. As shown in Lemma 3.5.3, the induced distributions over errors at each timestep are locally decaying distributions with failure rate $\sqrt{p_{\text{phys}}}$.

Next, Lemma 3.5.2 describes how the distribution is transformed when errors undergo linear transformations. Consider the terms in Equation (3.47):

$$\begin{pmatrix}\mathbf{I}_n & \mathbf{0}\\ \mathbf{H}_\mathsf{X} & \mathbf{I}_{m_\mathsf{X}}\end{pmatrix}\begin{pmatrix}\mathbf{b}_z^{(t)}\\ \mathbf{c}_z^{(t)}\end{pmatrix} , \tag{3.48}$$

$\mathbf{H}_\mathsf{X}$ has row and column weight at most Δ, so the block matrix that appears in Equation (3.48) has column weight at most $\Delta+1$. By Lemma 3.5.2, each term in

the union is distributed according to a locally decaying distribution with failure rate $2^{\Delta+1}(p_{\text{phys}})^{1/2(\Delta+1)}$.

Finally, Lemma 3.5.1 allows us to bound the failure rate of the compositions of independent locally decaying distributions. This, in turn, is an upper bound on the rate of the locally decaying distribution $\mathcal{Z}$ over $\mathbf{d}_z, \mathbf{a}_z$. The union extends over the depth $\mathcal{T}_{\mathsf{X}}$ of the circuit required to measure X-type syndromes terms. Applying Lemma 3.5.1 repeatedly, we find $\mathcal{Z}$ is a locally decaying distribution with failure rate

$$2^{\Delta+1} \cdot \mathcal{T}_{\mathsf{X}} \cdot (p_{\text{phys}})^{1/2(\Delta+1)} \,. \tag{3.49}$$

By an identical argument, the X errors are distributed according to a $2^{\Delta+1} \cdot \mathcal{T}_{\mathsf{X}} \cdot p_{\text{phys}}^{1/2(\Delta+1)}$ locally decaying distribution. In turn, this means that the induced distributions $\mathcal{X}$ and $\mathcal{Z}$ are locally decaying distributions with failure rate $2^{\Delta+1} \cdot \mathcal{T}_{\mathsf{X}} \cdot (p_{\text{phys}})^{1/2(\Delta+1)}$.

Repeating the same analysis for the Z-type syndrome measurements, we find that the $\mathcal{X}$ and $\mathcal{Z}$ distributions describing induced errors are locally decaying distributions with failure rate

$$2^{\Delta+1} \cdot \mathcal{T}_{\mathsf{Z}} \cdot (p_{\text{phys}})^{1/2(\Delta+1)} \,. \tag{3.50}$$

We can use Lemma 3.5.1 again to bound the failure rate per for the entire circuit C_n^Q. As $\mathcal{T}(C_n^Q) = \mathcal{T}_{\mathsf{X}} + \mathcal{T}_{\mathsf{Z}}$, we arrive at the result that $\mathcal{X}$ and $\mathcal{Z}$ are locally decaying distributions with failure rate p_{round} where

$$p_{\text{round}} = 2^{\Delta+1} \cdot \mathcal{T}(C_n^Q) \cdot (p_{\text{phys}})^{1/2(\Delta+1)} \,. \tag{3.51}$$

■

When qubits are arranged on an $L \times L$ lattice, the circuit depth $\mathcal{T}(C_n^Q)$ is $O(\sqrt{n}/R)$. If gates are constrained by geometric locality, i.e. $R = \omega(L)$, then the depth of the circuit C_n^Q grows with the code size n. However, for the existence of a threshold, we require p_{round} to be some fixed constant. We therefore only achieve a threshold if the physical failure probability vanishes as the size of the code increases:

$$p_{\text{phys}} = O\left[\left(\frac{1}{\mathcal{T}(C_n^Q)}\right)^{2(\Delta+1)}\right] \,. \tag{3.52}$$

However, if we were to use a concatenated construction, where the outer code is the constant-rate LDPC code $\mathcal{Q}_n$ and the inner code is a surface code $\mathcal{RS}_\ell$, then we can choose p_{phys} to decrease *exponentially* with the size of the inner code. We study this in the next section.

Finally, we comment that the factor $2^{\Delta+1}$ that appears in Theorem 3.5.4 can very likely be reduced. However, this particular version of the theorem is sufficient for our purposes, namely to prove the existence of a threshold for the hierarchical scheme. For readers interested in applying the hierarchical scheme to the real world, we shall estimate the logical failure rate of the hierarchical scheme numerically in Section 3.6.

Coarse graining concatenated circuits

In the next two sections, we will analyze the concatenated code by applying Gottesman's theorem described in Section 3.2 to both the inner code and the outer code. In this section, we apply it to the inner code; for $\mathcal{W} = \mathcal{W}(C_n^Q)$, the inner code $\mathcal{RS}_\ell^{\otimes \mathcal{W}}$ is itself an LDPC code. In Section 3.5, we will apply Gottesman's theorem to the outer code.

In Section 3.2, we described how we cannot ignore the details of the Level-0 syndrome-extraction circuit in a concatenated code. In this section, we show that if logical gates on surface codes are performed as described in Section 3.4, then they are fault tolerant. We show the existence of a threshold $q^{(0)}_{\text{phys}}$ such that if the failure rate per round is below $q^{(0)}_{\text{phys}}$, then we can directly study Level-1 operations and ignore Level-0 operations.

Consider an input state $\rho_{\text{in}} \in (\mathcal{RS}_\ell)^{\otimes \mathcal{W}}$ in the bilayer architecture. Let Level-0 faults on the syndrome-extraction circuit be distributed according to a locally decaying distribution with failure rate $p^{(0)}_{\text{phys}}$.

The failure rate per round on the data qubits and the syndrome qubits is the same because data and syndrome qubits both interact with 4 other qubits. Let $q^{(0)}_{\text{in}}$, $q^{(0)}_{\text{round}}$ be the thresholds for surface code error correction as defined in Section 3.2. Suppose we are below threshold. Then after error correction, tiles that have not failed are described by a locally decaying Level-0 error model with failure rate $p^{(0)}_{\text{round}}$. Theorem 3.5.4 guarantees that the failure rate per round grows with the depth of the syndrome-extraction circuit; it also relies on the degree of the qubits and stabilizer generators. If we measure X and Z syndromes separately, the depth of the syndrome-extraction circuit is at most 12. The degree of the qubits and stabilizers is 4. Using Theorem 3.5.4, we can bound the failure rate per round of surface code syndrome extraction:[15]

$$p^{(0)}_{\text{round}} < 384 \left(p^{(0)}_{\text{phys}} \right)^{1/10} . \tag{3.53}$$

[15]The constant $2^{\Delta+1} = 32$ and $32 \times 12 = 384$.

This bound can be much better—for example X and Z syndromes can be measured in parallel which, in turn, can reduce the depth of the circuit; we can also likely reduce the constant 384 in front of p_{round}. However, we continue to use the bound in Equation (3.53) for simplicity.

We can use Theorem 3.5.4 to show that the logical operations for the bilayer architecture are fault tolerant. We argue that both the Level-0 and Level-1 failure rates after the operation are constant.

Theorem 3.5.5. *Let C be the circuit on a state $\rho_{\text{in}} \in \mathcal{RS}_\ell^{\otimes \mathcal{W}}$ such that each tile is involved in at most one logical gate in $\mathcal{K}_1$. Tiles that have not suffered a logical error are described by a locally decaying error with Level-0 input failure rate $p^{(0)}_{\text{round}}$. There exists a threshold $q^{(0)}_{\text{phys}}$ such that, if $p^{(0)}_{\text{phys}} \leq q^{(0)}_{\text{phys}}$, then*

1. *the circuit C is described by a Level-1 locally decaying faults model with Level-1 failure rate $p^{(1)}_{\text{phys}} := \exp(-c_{\text{EC}} \cdot \ell)$.*
2. *the output is described by a Level-0 locally decaying errors model with failure rate less than $p^{(0)}_{\text{round}}$.*

Proof. Let $\rho_{\text{in}} \in \mathcal{RS}_\ell^{\otimes \mathcal{W}}$ be a noisy code state with Level-0 errors described by a locally decaying distribution with failure rate $p^{(0)}_{\text{round}}$.

For sufficiently low logical failure rate, we can use Gottesman's result presented in Section 3.2 to bound the failure rate for error correction and to show that after error correction, the Level-0 errors are locally decaying distributions with failure rate $p^{(0)}_{\text{round}}$.

State preparation: Suppose we wished to prepare the state $\left|\overline{0}\right\rangle^{\otimes m}$ for the code $\mathcal{RS}_\ell^{\otimes m}$. Each Level-0 qubit is prepared in $|0\rangle$ and we then perform the syndrome-extraction circuit for the surface code on all m copies. The Level-0 errors are described by a locally decaying error model with failure rate $p^{(0)}_{\text{in}} = p^{(0)}_{\text{phys}}$. The faults in the syndrome-extraction circuit C are also described by a locally decaying faults model with failure rate $p^{(0)}_{\text{phys}}$. Error correction is successful if

$$p^{(0)}_{\text{phys}} < q^{(0)}_{\text{in}}\,, \qquad p^{(0)}_{\text{round}} < q^{(0)}_{\text{round}}\,. \tag{3.54}$$

Entangling gates: Entangling gates between data and ancilla blocks are performed in a transversal manner. Errors due to faults in the transversal gate are distributed

according to a locally decaying distribution with failure rate $p^{(0)}_{\text{phys}}$. Lemma 3.5.1 shows that the input to error correction is a state with Level-0 errors described by a locally decaying distribution with failure rate $p^{(0)}_{\text{round}} + p^{(0)}_{\text{phys}}$.

Error correction is successful if

$$p^{(0)}_{\text{round}} + p^{(0)}_{\text{phys}} < q^{(0)}_{\text{in}} , \qquad p^{(0)}_{\text{round}} < q^{(0)}_{\text{round}} . \tag{3.55}$$

SWAP gates: Assume that the Level-0 failure rate is $p^{(0)}_{\text{round}}$. The logical SWAP operation is decomposed entirely in terms of physical SWAP operations. As these are non-entangling operations, the error distribution is a locally decaying distribution with failure rate $\sqrt{p^{(0)}_{\text{phys}}}$. We can use Lemma 3.5.1 to find the effective failure rate per round. This is equal to the sum of the failure rate per round of syndrome extraction and the failure rate of the SWAP gate itself. Note that because the SWAP gate has larger depth, we perform more than d_ℓ rounds of syndrome extraction.

Therefore, the failure rate per round on both data and ancilla qubits is $p^{(0)}_{\text{round}} + \sqrt{p^{(0)}_{\text{phys}}}$. Error correction is successful if

$$p^{(0)}_{\text{round}} < q^{(0)}_{\text{in}} , \qquad p^{(0)}_{\text{round}} + \sqrt{p^{(0)}_{\text{phys}}} < q^{(0)}_{\text{round}} . \tag{3.56}$$

Logical measurement of Pauli operators:

We will wish to measure logical operators on tiles that represent Level-1 ancilla qubits. Consider a state with Level-0 errors distributed according to a locally decaying distribution with failure rate $p^{(0)}_{\text{round}}$. We first study the logical measurement of a single tile.

To destructively measure the logical X (Z) operator on a single tile, we can measure each of the physical qubits in the X (Z) basis. This is permitted by our available operations in $\mathcal{K}_0$. Faults on measurements are distributed according to a locally decaying distribution with rate $p^{(0)}_{\text{phys}}$. The resulting distribution on the output bits is a locally decaying distribution with rate $p^{(0)}_{\text{round}} + p^{(0)}_{\text{phys}}$. We can use each of the individual Level-0 qubit outputs to infer the values of each of the X-type (Z-type) stabilizer generators and correct Z (X) errors. This fails with probability $\exp(-c_{EC} \cdot \ell)$.

We can now study all tiles that undergo measurement. As measurements on each tile are performed separately, this induces a Level-1 measurement error with probability $\exp(-c_{EC} \cdot \ell)$.

In the mean time, tiles that represent data qubits remain idle for 1 timestep. As we assume idle errors are distributed according to a locally decaying distribution with failure rate $p^{(0)}_{\text{phys}}$, the Level-0 error rates on these tiles are $p^{(0)}_{\text{round}} + p^{(0)}_{\text{phys}}$. Error correction is successful if

$$p^{(0)}_{\text{round}} + p^{(0)}_{\text{phys}} < q^{(0)}_{\text{in}} \ . \tag{3.57}$$

Combining requirements for all operations: We can use Equation (3.53) to state $p^{(0)}_{\text{round}} < 384(p^{(0)}_{\text{phys}})^{1/10}$ and note that both $p^{(0)}_{\text{phys}}$ and $\sqrt{p^{(0)}_{\text{phys}}}$ are less than $(p^{(0)}_{\text{phys}})^{1/10}$. Therefore, we can define the threshold $q^{(0)}_{\text{phys}}$ using the bounds in Equations (3.54), Equation (3.55), Equation (3.56) and Equation (3.57):

$$q^{(0)}_{\text{phys}} = \min\left[\left(\frac{q^{(0)}_{\text{in}}}{385}\right)^{10}, \left(\frac{q^{(0)}_{\text{round}}}{385}\right)^{10}\right] \ . \tag{3.58}$$

Below threshold, we can invoke Gottesman's result to guarantee error suppression; we obtain a logical failure rate $p^{(1)}_{\text{phys}} = \exp(-c_{\text{EC}} \cdot \ell)$.

■

The syndrome-extraction circuit $C^{\mathcal{H}}_N$ has a threshold

In this section, we prove that the hierarchical code $\mathcal{H}_N$ has a threshold if we measure syndromes using the circuit $C^{\mathcal{H}}_N$. We review the construction first and the corresponding assumptions on failure rates. Thus far, we have simply stated the relationship between ℓ and n, i.e. that $\ell = \Theta(\log(n))$, without justification. We show in Lemma 3.5.6 that letting the inner code have size $\ell = \Theta(\log(n))$ is indeed sufficient to achieve arbitrarily small, but constant, Level-1 failure rate per round $p^{(1)}_{\text{round}}$. We bring these elements together in Theorem 3.5.7 to show that the hierarchical construction has a threshold.

Recall that the hierarchical code $\mathcal{H}_N$ is constructed by concatenating an outer $[\![n, k, d, \Delta_q, \Delta_g]\!]$ constant-rate LDPC code $\{\mathcal{Q}_n\}$ and inner $[\![d^2_\ell, 1, d_\ell]\!]$ code $\mathcal{RS}_\ell$. The family $\mathcal{Q}_n$ has parameters $k = \rho \cdot n$ for $\rho > 0$ and distance $d = \Theta(n^\delta)$ for $\delta > 0$.

Suppose we are given an input state of the concatenated code $\mathcal{H}_N$ subject to the following error model:

1. Errors on the state are distributed according to locally decaying distributions:

   a) on Level-0 with failure rate $p^{(0)}_{\text{in}}$, and

b) on Level-1 with failure rate $p_{\text{in}}^{(1)}$.

2. Level-0 faults in the circuit are distributed according to a locally decaying distribution with failure rate $p_{\text{phys}}^{(0)}$.

Let $p_{\text{round}}^{(0)}$ denote the failure rate per round of syndrome extraction for the rotated surface code. As the depth of the syndrome-extraction circuit for the rotated surface code is constant, for fixed values of $p_{\text{phys}}^{(0)}$, $p_{\text{round}}^{(0)}$ is also a constant (See Equation (3.53)). We assume $p_{\text{in}}^{(0)} \leq p_{\text{round}}^{(0)}$ because it will make the following statements easier.

Qubits are laid out on a bilayer architecture as described in Section 3.4. Physical qubits are aggregated to form $\mathcal{W}(C_n^{\mathcal{Q}})$ rotated surface codes $\mathcal{RS}_\ell$; these form $2L^2$ tiles where L is the smallest integer satisfying $2L^2 \geq \mathcal{W}(C_n^{\mathcal{Q}})$.

The product code $\mathcal{RS}_\ell^{\otimes \mathcal{W}}$ is itself an LDPC code. The tiles will be used to simulate long-range entangling gates required to perform the syndrome-extraction circuit $C_n^{\mathcal{Q}}$ for the outer code. Single-tile preparation and measurement, and two-tile entangling gates are described in Section 3.4; Level-1 SWAP gates and permutations of tiles were described in Section 3.4. Recall that $q_{\text{phys}}^{(0)} \in (0, 1]$ was defined in Section 3.5. Per Theorem 3.5.5, if the input state has Level-0 errors described by a locally decaying distribution with failure rate $p_{\text{round}}^{(0)}$ and $p_{\text{phys}}^{(0)} < q_{\text{phys}}^{(0)}$, Level-1 circuit faults are distributed according to a locally decaying distribution with failure rate $p_{\text{phys}}^{(1)}$ and Level-0 residual errors are described by a locally decaying distribution with failure rate $p_{\text{round}}^{(0)}$ on tiles that have not failed. This result allows us to coarse grain the Level-0 circuit and study Level-1 errors and faults directly.

For the outer code to have a threshold, we require that the $[\![n, k, d, \Delta_q, \Delta_g]\!]$ LDPC code family $\{\mathcal{Q}_n\}$ has a syndrome-extraction circuit such that $p_{\text{round}}^{(1)}$ remains a sufficiently small constant as discussed in Section 3.2. In the following lemma, we show that $\ell = \Theta(\log(n))$ is sufficient to achieve this.

Lemma 3.5.6. *Suppose Level-1 faults on the syndrome-extraction circuit $C_N^{\mathcal{H}}$ are distributed according to a locally decaying distribution with failure rate $p_{\text{phys}}^{(1)}$. Then, for arbitarily small constant $\epsilon > 0$, $p_{\text{round}}^{(1)} < \epsilon$ can be achieved using $\ell = \Theta(\log(n))$.*

Proof. From Theorem 3.5.4, the failure rate per round scales as

$$p_{\text{round}}^{(1)} = 2^{\Delta+1} \cdot \mathcal{T}(C_n^{\mathcal{Q}}) \cdot \left(p_{\text{phys}}^{(1)}\right)^{1/2(\Delta+1)}, \tag{3.59}$$

where $\Delta = \max(\Delta_q, \Delta_g)$ is some constant for a fixed family $\mathcal{Q}_n$. We want $p^{(1)}_{\text{round}}$ to be an arbitrarily small constant $\epsilon > 0$. Per Theorem 3.5.5, Level-1 faults are distributed according to a locally decaying distribution with failure rate which implies that $p^{(1)}_{\text{phys}} = \exp(-c_{\text{EC}} \cdot \ell)$. From Section 3.4, $\mathcal{T}(C^{\mathcal{Q}}_n) = O(L/R) = O(\sqrt{n}/R)$. Therefore, the upper bound on $p^{(1)}_{\text{phys}}$ can be satisfied by choosing $\ell = \Theta(\log(n))$. ■

In particular, there exists a threshold $q^{(1)}_{\text{round}}$ for the outer code $\mathcal{Q}_n$ for the syndrome-extraction circuit $C^{\mathcal{Q}}_n$. This can be achieved for some ℓ such that $\ell = \Theta(\log(n))$.

Theorem 3.5.7. *There exists a choice of ℓ such that $\ell = \Theta(\log(n))$ and thresholds $q^{(0)}_{\text{in}}$, $q^{(0)}_{\text{phys}}$ and $q^{(1)}_{\text{in}}$ such that if*

$$\max\left(p^{(0)}_{\text{in}}, p^{(0)}_{\text{round}}\right) < q^{(0)}_{\text{in}}\,, \qquad p^{(0)}_{\text{phys}} < q^{(0)}_{\text{phys}}\,, \qquad p^{(1)}_{\text{in}} < q^{(1)}_{\text{in}}\,,$$

then the following is true. With probability at least $1 - p_{\mathcal{H}}(N)$, the state after error correction is correctable by an ideal decoder where, for some positive number $c_{\mathcal{H}}$ that is independent of N,

$$p_{\mathcal{H}}(N) < \exp\left(-c_{\mathcal{H}} \cdot \frac{N^{\delta}}{\log^{2\delta}(N)}\right)\,.$$

Furthermore, the residual errors are distributed according to a locally decaying distribution with failure rates $p^{(1)}_{\text{round}}$ and $p^{(0)}_{\text{round}}$ on Level-1 and Level-0 respectively.

Proof. Suppose $p^{(0)}_{\text{in}} < q^{(0)}_{\text{in}}$ and $p^{(0)}_{\text{phys}} < q^{(0)}_{\text{phys}}$. By definition, this is sufficient to perform Level-1 logical gates and surface code error correction as per Theorem 3.5.5.

Next, the LDPC code has thresholds $q^{(1)}_{\text{in}}$ and $q^{(1)}_{\text{round}}$ (See Section 3.2). The input state has Level-1 errors described by a locally decaying distribution with failure rate $p^{(1)}_{\text{in}}$. For the syndrome-extraction circuit on the outer LDPC code to be successful, we require $p^{(1)}_{\text{in}} < q^{(1)}_{\text{in}}$.

Finally, we require that the Level-1 failure rate per round is below the corresponding threshold

$$p^{(1)}_{\text{round}} < q^{(1)}_{\text{round}}\,. \tag{3.60}$$

From Lemma 3.5.6, this can be achieved using $\ell = \Theta(\log(n))$.

By definition, syndrome-extraction is successful if the ideal decoder $\mathcal{R}_{\mathcal{H}}$ is able to recover the final state. The code $\mathcal{H}_N$ fails if the outer LDPC code $\mathcal{Q}_n$ fails, i.e. the probability of failure is $p_{\mathcal{H}}(N) = p_{\mathcal{Q}}(n) = \exp(-c_{\mathcal{Q}} \cdot d(n)) = \exp\big(-c_{\mathcal{Q}} \cdot \Theta(n^{\delta})\big)$.

Using Equation (3.11), we can express the probability of failure $p_{\mathcal{H}}(N)$ in terms of N:

$$p_{\mathcal{H}}(N) < \exp\left(-c_{\mathcal{H}} \cdot \frac{N^{\delta}}{\log^{2\delta}(N)}\right) . \tag{3.61}$$

for some positive number $c_{\mathcal{H}}$ that is independent of N.

Residual errors are distributed according to locally decaying distribution:

1. on Level-1 with failure rate $p^{(1)}_{\text{round}}$; this is guaranteed by Gottesman's result applied to the outer code.
2. on Level-0 errors failure rate $p^{(0)}_{\text{round}}$; This is guaranteed by Theorem 3.5.5.

The result follows. ■

We reiterate that $p_{\mathcal{H}}(N)$ is an upper bound on the failure rate for the Level-2 error probability distribution.

This analysis depends crucially on the failure rate of the SWAP gates; $p^{(1)}_{\text{round}}$, and therefore the size of the inner code, scales with the depth of the circuit because of noisy SWAP operations. In proving Theorem 3.5.7, we were agnostic to the failure modes in the circuit and assumed that all Level-1 two-qubit gates fail with probability $p^{(1)}_{\text{phys}}$. However, if the fidelity of physical SWAP gates can be improved over the fidelity of entangling gates, this can reduce the overhead for the hierarchical scheme significantly. We provide evidence for this in Section 3.6 where we estimate the logical failure rate for the hierarchical scheme. In certain architectures such as trapped neutral atoms, SWAP gates can be performed by physically moving the trap [Blu+22]. In this case, the failure rate for the SWAP gates may have no direct connection to the failure rate for CNOT operations.

3.6 Comparisons with the basic encoding

We have shown that the hierarchical code $\{\mathcal{H}_N\}$ has a syndrome-extraction circuit that can be constructed using gates restricted by geometric locality such that it has a threshold. Below threshold, the WER is suppressed superpolynomially, but subexponentially in N. It is natural to ask whether the resources spent in performing SWAP gates can be better spent simply building a more robust surface code. In this section, we consider the basic encoding $\mathcal{B}_M$ which encodes K logical qubits in surface codes $\mathcal{RS}_{\ell_M}$. We compare the hierarchical scheme and the basic enconding in different ways.

We show in Section 3.6 that for a target WER, the syndrome-extraction circuit for the hierarchical memory is more efficient than the syndrome-extraction circuit for the basic encoding. This is measured by the depth and width of the corresponding circuits. We will state and prove a formal version of Theorem 3.1.4. Depending on the value of the threshold for the outer LDPC codes however, it is not immediately obvious that this scaling manifests for practical block lengths. In the rest of this section, we present numerical estimates for the WER $p_{\mathcal{H}}(N)$ of the hierarchical memory and contrast it with the WER $p_{\mathcal{B}}(M)$ for the basic encoding. We do this by demanding a fixed total number of qubits for both schemes and compare $p_{\mathcal{H}}(N)$ and $p_{\mathcal{B}}(M)$. We demonstrate that there is a *crossover point*, i.e. a value of the physical error rate where, for fixed total number of qubits, the hierarchical memory outperforms the basic encoding, i.e. $p_{\mathcal{H}}(N) < p_{\mathcal{B}}(M)$. In our estimates, this happens at gate error rates roughly between 10^{-3} and 10^{-4}. While these are preliminary estimates, they are promising nonetheless as they are in the realm of possibility.

In Section 3.6, we briefly discuss the codes we use as outer and inner codes. To estimate the crossover point, we make some assumptions about the noise model, gates, and decoder. Owing to these assumptions, our estimates should only be interpreted as a proof-of-principle that the overhead of the hierarchical scheme pays off in a reasonable parameter regime. In Section 3.6, we present the results of our simulations. All together we believe these assumptions, especially those related to the decoder, code, and noise model, are conservative. We return to these assumptions in Section 3.6 and for each assumption, we outline how one might expect it to change (1) in the future, and (2) in a more realistic setting. In general, we expect that with careful engineering (e.g. high-rate linear-distance codes, architecture-specific considerations, improved decoding algorithms) and more realistic noise modeling (e.g. including significant error correlations), the cross-over to when hierarchical memories outperform surface codes will occur at smaller numbers of logical qubits, higher physical error rates, and higher target WERs than in our estimates.

Asymptotic comparison with surface code

We have proved the existence of a threshold when we simulate an LDPC code using local gates. However, the existence of a threshold alone might not warrant switching over to a different scheme when there already exists an excellent local scheme — the surface code. We recall that we are only constructing a quantum memory, and not a scheme for universal, fault-tolerant quantum computation. In this section, we ask how the surface code would perform if we used the same total number of qubits used

in the concatenated scheme above to plainly encode all logical qubits. We find that there is a space-time tradeoff to implementing a hierarchical scheme.

The hierarchical scheme $\{\mathcal{H}_N\}$ with corresponding fault tolerant syndrome-extraction circuits $\{C_N^{\mathcal{H}}\}$ achieves the following costs:

$$\mathcal{W}(C_N^{\mathcal{H}}) = \Theta(N) \qquad \mathcal{T}(C_N^{\mathcal{H}}) = O\left(\frac{\sqrt{N}}{R}\right) . \tag{3.62}$$

This family encodes $k(n) = \rho \cdot n$ qubits. Note that the depth is for a *single* round of syndrome extraction. We will return to this point shortly.

Consider the *basic encoding* defined by the family $\{\mathcal{B}_M\}$ where $M = \Theta(k \cdot \ell_M^2)$, and $\mathcal{B}_M = \bigotimes_{i=1}^{k} \mathcal{RS}_{\ell_M}$ is a k-fold product of rotated surface codes $\mathcal{RS}_{\ell_M}$. Each code $\mathcal{RS}_{\ell_M}$ has distance $d_M = \Theta(\ell_M)$. Let the corresponding circuits be denoted $\{C_M^{\mathcal{B}}\}$.

To compare with $\mathcal{H}_N$, we probe the parameters of $C_M^{\mathcal{B}}$ required to guarantee the same logical error suppression. Let $p_{\mathcal{B}}(M)$ denote the failure rate for the Level-1 logical probability of failure for $\mathcal{B}_M$—we declare failure if any of the k logical qubits fail.

We assume that the Level-0 physical failure rates are sufficiently below threshold to perform surface code error correction, i.e. $p_{\text{phys}}^{(0)} < q_{\text{phys}}^{(0)}$. Furthermore, we also assume that the code state $\mathcal{B}_M$ is prepared such that there are no input errors, i.e. $p_{\text{in}}^{(1)} = p_{\text{in}}^{(0)} = 0$. This allows us to isolate the rate of error suppression because of error correction.

Lemma 3.6.1. *Let $\{\mathcal{B}_M\}$ be the basic encoding such that $p_{\mathcal{B}}(M) < \exp\left(-c_{\mathcal{H}} \cdot N^{\delta}/\log(N)^{2\delta}\right)$ where $c_{\mathcal{H}}$ is a positive constant. Then*

$$\mathcal{W}(C_M^{\mathcal{B}}) = \Omega\left[\left(\frac{N}{\log(N)}\right)^{1+2\delta}\right] , \qquad \mathcal{T}(C_M^{\mathcal{B}}) = \Omega\left[\left(\frac{N}{\log^2(N)}\right)^{\delta}\right] .$$

Proof. By assumption, $p_{\text{in}}^{(1)} = p_{\text{in}}^{(0)} = 0$ and $p_{\text{phys}}^{(0)} < q_{\text{phys}}^{(0)}$ and therefore we can perform error correction. The Level-1 logical failure probability for the code $\mathcal{RS}_{\ell_M}^{\otimes k}$ is described by a locally-decaying error model with failure rate $p_{\mathcal{RS}}(\ell_M)$ (See Section 3.2), where

$$p_{\mathcal{RS}}(\ell_M) = \exp(-c_{\text{EC}} \cdot \ell_M) . \tag{3.63}$$

We declare failure if any of the k tiles of $\mathcal{B}_M$ fails, which implies that

$$\begin{array}{ccccc} p_{\mathcal{RS}}(\ell) & \leq & p_{\mathcal{B}}(M) & \leq & 1 - (1 - p_{\mathcal{RS}}(\ell))^k \\ \exp(-c_{\text{EC}} \cdot \ell_M) & \leq & p_{\mathcal{B}}(M) & \leq & n \cdot \exp(-c_{\text{EC}} \cdot \ell_M) \end{array} . \tag{3.64}$$

To guarantee that the error rate $p_{\mathcal{B}}(M)$ is lower than $p_{\mathcal{H}}(N)$, we at least require that

$$\exp(-c_{\mathrm{EC}} \cdot \ell_M) \leq \exp\left(-c_{\mathcal{H}} \cdot \frac{N^{\delta}}{\log(N)^{2\delta}}\right) . \tag{3.65}$$

This implies that $\ell_M = \Omega(N^{\delta}/\log(N)^{2\delta})$.

We can now compute the space and depth requirements for $C_M^{\mathcal{B}}$. The space cost $\mathcal{W}(C_M^{\mathcal{B}})$ is $\Theta(k \cdot \ell_M^2)$. The hierarchical memory $\mathcal{H}_N$ uses a constant-rate $[\![n, k, d, \Delta_q, \Delta_g]\!]$ quantum LDPC code $\mathcal{Q}_n$ where $k(n) = \rho \cdot n$ and $d(n) = \Theta(n^{\delta})$. This implies that

$$\mathcal{W}(C_M^{\mathcal{B}}) = \Omega\left[\left(\frac{N}{\log(N)}\right)^{1+2\delta}\right] . \tag{3.66}$$

Furthermore, each tile requires ℓ_M rounds of error correction; syndrome-extraction circuits on separate tiles can be run in parallel. Therefore

$$\mathcal{T}(C_M^{\mathcal{B}}) = \Theta(\ell_M) = \Omega(N^{\delta}/\log^{2\delta}(N)) . \tag{3.67}$$

This completes the proof. ■

Comparing with Equation (3.62), the basic encoding requires a larger space overhead for all $\delta > 0$:

$$\frac{\mathcal{W}(C_M^{\mathcal{B}})}{\mathcal{W}(C_N^{\mathcal{H}})} = \Omega\left[\frac{N^{2\delta}}{\log^{1+2\delta}(N)}\right] . \tag{3.68}$$

As stated, however, the time overhead is worse. Although the depth of the syndrome-extraction circuit $C_N^{\mathcal{H}}$ is $O(\sqrt{N}/R)$, we will need to perform $d(n)$ rounds of syndrome extraction to be fault tolerant. However, this is not a fundamental requirement; it is due to the nature of Gottesman's proposal in [Got14] which uses an inefficient minimum-weight decoder. There exist constant-rate LDPC codes that possess efficient, single-shot decoding algorithms, i.e. syndrome extraction only needs to be performed a constant number of times for the decoding algorithm to work [LTZ15; FGL18a; FGL18b]; furthermore the algorithm requires $O(N)$ time. For such codes, we can compare the depth of the syndrome-extraction circuit $\mathcal{T}(C_N^{\mathcal{H}})$ and that of the basic encoding $C_M^{\mathcal{B}}$. In addition to the width blowup, the basic encoding also requires a larger time overhead when $\delta > 1/2$.

$$\frac{\mathcal{T}(C_M^{\mathcal{B}})}{\mathcal{T}(C_N^{\mathcal{H}})} = \Omega\left[\frac{N^{\delta-1/2}}{R \cdot \log^{2\delta}(N)}\right] . \tag{3.69}$$

Using LDPC codes with single-shot decoding algorithms, the hierarchical memory is a more efficient way to achieve a low logical error rate in terms of both circuit depth and width.

Having said this, it is not clear if this advantage manifests for practical block lengths.

For small codes and high error rates, it may well be that it is still optimal to use the basic encoding. We expect to see a *crossover point*—a value of physical error rate where the hierarchical scheme $\{\mathcal{H}_N\}$ has a lower logical failure rate than the basic encoding $\{\mathcal{B}_M\}$ with the same overhead. Where exactly this crossover happens will depend on a number of parameters that are specific to the implementation, including the choice of the outer code, its threshold and our choice of decoders. In the rest of this section, we attempt to estimate where this happens.

Setup for numerical estimates

Outer code: We choose a quantum expander code as our outer code [TZ14; LTZ15]. We do not utilize any of the structure of the code, so any LDPC code with constant rate and polynomially scaling distance will suffice. For these reasons, we only briefly discuss the code construction. We pick a classical code by sampling the check matrix from the ensemble of $m \times n$ matrices with 5 ones in each column and 8 ones in each row.

In particular, we pick a classical code with length 896 and 336 encoded bits. Work by Litsyn and Shevelev [LS02] computes the asymptotic weight distribution of codewords — with high probability, this code has distance 119. The resulting quantum code has parameters

$$N = 1\,116\,416, \quad K = 112\,896, \quad D = 119, \quad \Delta_q = 16, \quad \Delta_g = 13\,. \tag{3.70}$$

We choose this code as it has a high rate which is necessary to reduce the amount of overhead in the scheme. The large block length we consider here is a consequence of using code families with sub-linear distance scaling. However, the full trade-off for rate, distance, check weight, etc for linear-distance codes has not yet been explored. We note that even at a relative distance d/n of 10^{-3}, a linear-distance code of such large block length would achieve a distance of roughly 10^3.

While the hierarchical memory construction has good asymptotic performance guarantees, if the overhead is too high then the hierarchical memory wins only at an extremely low WER.[16]

[16] 1 year is $\approx 10^{16}$ nanoseconds, 1 Hubble time is $\approx 10^{10}$ years or $\approx 10^{26}$ nanoseconds. For any

Inner code: As in the earlier sections, we consider the square rotated surface codes $\mathcal{RS}_\ell$ for the inner code. In our estimates, we allow for $d_\ell = 3, 5, 9, 15, 21, 27$.

We make some assumptions about errors on both the logical and physical levels. We present these assumptions together below and discuss justifications for some assumptions in what follows.

Level-0 noise model: We assume circuit-level Pauli noise on each physical qubit. We treat SWAP gates and other Clifford operations separately.

1. Each t-qubit gate (except SWAP gates) at the physical level fails with a probability p and leaves behind one of the $4^t - 1$ non-trivial t-qubit Pauli operators picked uniformly at random. We assume that qubit reset completely removes all traces of the original state. However, it may reset to the wrong computational basis state with probability p.

2. The failure probability of the physical SWAP-gate is $r_{\mathsf{SWAP}} \cdot p$, where $r_{\mathsf{SWAP}} = 1, 10^{-1}, 10^{-2}$. In this setting, the surface code syndrome-extraction circuit is performed every $1/r_{\mathsf{SWAP}}$ SWAP-gates, so that at the physical level the circuit-level noise model remains relatively unchanged for different values of r_{SWAP}. This assumption will be discussed in detail in Section 3.6.

Level-1 noise model: We assume that the surface code fails at a probability $p^{(1)}_{\mathcal{RS}}(d_\ell)$, and that the effective noise witnessed by the outer code because of all the SWAP gates is $p^{(1)}$.

1. The logical error rate $p^{(1)}_{\text{phys}}(d_\ell)$ of each $\ell \times \ell$ rotated surface code tile is [WFH11; FMMC12]

$$p^{(1)}_{\text{phys}}(d_\ell) \approx 0.1 \left(\frac{p}{10^{-2}}\right)^{\lceil d_\ell/2 \rceil} \tag{3.71}$$

   per d_ℓ physical level timesteps where one physical level timestep is one round of syndrome extraction plus one (optional) transversal gate which totals roughly 6 gates. This assumption is discussed in Section 3.6.

2. The effective error rate per long-range CNOT gate is $p^{(1)} = 1 - (1 - p^{(1)}_{\text{phys}}(d_\ell))^{t_{\text{route}}+1}$. It is analogous to the two-qubit gate error rate in the model with long-range gates. t_{route} is the time required for permutation routing presented in Equation (3.16).

practical purpose a WER of 10^{-25} per gate time should suffice.

Level-2 noise model: Finally, we assume that the logical failure rate for the LDPC code, $p_Q(n)$, is consistent with a minimum-weight decoder.

1. For our LDPC code, we assume that the WER under circuit-level Pauli noise using long-range gates is
$$p_Q = \left(\frac{p^{(1)}}{10^{-3}}\right)^{10} \tag{3.72}$$
per cycle of syndrome extraction. The threshold of the code is assumed to be about 10^{-3} under circuit noise. The exponent is 10 rather than half the distance which is ~ 55 because of hook errors. This assumption is discussed in Section 3.6.

If desired, readers can skip ahead to the numerical estimates in Section 3.6 and return to the justification of the noise model later.

Decoder performance for the inner code

Consider Equation (3.71) for the scaling of the logical failure rate for a surface code of distance d_ℓ. We assumed a surface code threshold of 10^{-2}.

This equation neglects:

1. finite-size effects present at very small code distances.
2. the slight reduction in threshold from inserting a layer of gates failing with rate p between syndrome extraction cycles[17]. Recall that this is necessary to implement a logical SWAP operation in the bilayer architecture as discussed in Section 3.4.
3. the distinction between rotated and standard surface codes. Owing to this, the expression for the logical error rate is an order of magnitude estimate. We expect our conclusions should be somewhat insensitive to the precise form of the logical error rate and also apply to more general locally decaying error models. For calculational convenience, we assume that the failure rate q after T syndrome extraction rounds is given by $1 - q = \left(1 - p^{(1)}_{\text{phys}}(d_\ell)\right)^{T/d}$.

[17] In general, the precise value of the circuit-level threshold already requires some assumptions about what gates are native in the device: The optimal syndrome extraction circuit with our physical layer layout requires 5 to 8 gates depending on these assumptions, so the insertion of an additional gate is relatively unimportant.

Physical SWAP fidelity

Recall that simulating a long-range CNOT via SWAP gates results in a CNOT failure rate of $p^{(1)}$. We assume that the effective failure rate witnessed by the outer code is

$$p^{(1)} = 1 - (1 - p^{(1)}_{\text{phys}}(d_\ell))^{\frac{r_{\text{SWAP}} t_{\text{route}}}{d_\ell} + 1} . \tag{3.73}$$

The parameter t_{route} is the time required to perform a permutation routing in the bilayer architecture as specified in Equation (3.16). For convenience, we restate it here

$$t_{\text{route}} = (2d_\ell + 1)(3L - 3) + 8 . \tag{3.16}$$

The parameter r_{SWAP} bounds the (in)fidelity of the SWAP operation in terms of the CNOT gate (in)fidelity as we now explain.

In the previous sections, we assumed that all gates failed at the same rate. As noted in Section 3.5, the main source of noise in the hierarchical model stems from the SWAP gates. This worst-case model was convenient for a proof of the existence of a threshold. Furthermore, in many devices the SWAP gate is implemented using the same mechanism as the two-qubit entangling gates and so the noise rates are comparable. However, this is not the only way to implement SWAP gates.

In platforms where the qubits can be physically moved, we can effectively "rewire" the connectivity of the device at runtime. Physically swapping qubits does not require the qubit degree of freedom to be coupled to, and so one might expect that it is an easier task to perform with higher fidelity or speed. Such techniques have been demonstrated in some experimental platforms: Rearrangable tweezers in Rydberg platforms [End+16; Bar+16; Blu+22] and ion shuttling in trapped ion platforms [Hen+06; Kau+17]. In this setting, it is possible that the SWAP gate has much higher fidelity than CNOT gates.

Accordingly, in our model, we assign a constant r_{SWAP} which specifies the ratio of SWAP-gate and idle noise to CNOT-gate noise. With a less noisy r_{SWAP}, we perform $1/r_{\text{SWAP}}$ level-0 SWAP operations per round of surface code syndrome extraction such that the physical error rate in the surface code syndrome extraction circuit remains constant with respect to r_{SWAP}. Utilizing this optimization, an entire permutation takes $r_{\text{SWAP}} t_{\text{route}}$ rounds of syndrome extraction. Equation (3.73) is in terms of the surface code cycle (d_ℓ rounds of syndrome extraction), and the total number of surface code cycles is $\frac{r_{\text{SWAP}} t_{\text{route}}}{d_\ell}$ for a permutation and 1 for an entangling gate. We have omitted floor and ceiling functions in this discussion for simplicity.

For example, in a neutral atom system [Blu+22], an array of qubits with a coherence time of seconds was rearranged with an average rearrangement speed of several microseconds per lattice site moved. If the dominant source of errors in rearrangement is due to idle errors, then we should assign an infidelity to the SWAP gate of roughly 10^{-5} whereas the two-qubit gate possessed an infidelity of about 10^{-2} i.e. $r_{\mathsf{SWAP}} \approx 10^{-3} - 10^{-2}$. Routing does not require generating entanglement, so the qubit can remain encoded in well-isolated degrees of freedom. Owing to this, we consider three scenarios: where r_{SWAP} is 10^{0}, 10^{-1}, or 10^{-2}.

We note that it is a simplification to consider the rearrangement primitive in each platform (tweezers, ion shuttling, etc.) as simply SWAP-gates: Frequently there are effects like accumulated motional heating, recooling, acceleration speed limits, etc, but we expect the basic routing ideas and qualitative conclusions remain the same even in the more complicated setting.

Hook errors

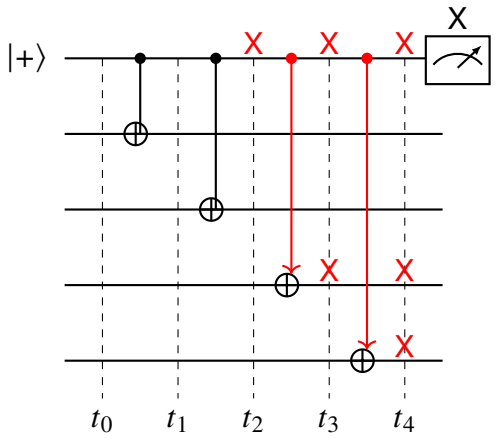


Figure 3.17: Hook errors: errors flowing onto data qubits. In this case, an X error appears in the midst of a syndrome-extraction circuit. This then propagates to the data qubits.

Current decoder technology for LDPC codes is relatively immature, so we assume a WER scaling consistent with a minimum-weight decoder. At physical failure rate p, we assume that the logical failure rate of an $[\![n, k, d, \Delta_q, \Delta_g]\!]$ LDPC code $\mathcal{Q}_n$ below threshold is dominated by a term proportional to p^t where t is the smallest number of fault locations that is uncorrectable. If our intuition is informed by an i.i.d. error model on qubits, we may expect $t \approx d/2$. However, this is not true in the context

of syndrome-extraction circuits as corrupted syndrome qubits can spread errors to many data qubits.

These errors, called hook errors [DKLP02], are harmful errors that can dominate the lower error rate performance of the quantum code. By a rough estimate, they can reduce the distance of a $[\![n, k, d, \Delta_q, \Delta_g]\!]$ LDPC code by a factor of $\Delta_g/2$. To explain how they work, consider the example measurement circuit for an X-type stabilizer generator as shown in Figure 3.4. An X error on the ancilla can propagate to a much larger data error.

In theory, the hook error is $O(1)$-sized and the syndrome extraction circuit is fault-tolerant. However, addressing these errors can significantly reduce the size of the LDPC code required to achieve a target logical failure rate.

Suppose ancilla qubits fail with probability p. A measurement circuit for a weight w operator can create hook errors with weight ranging from 1 up to w. If the circuit is measuring the checks of a code, the weight of the hook error can be reduced by using the measured stabilizer generator giving a maximum reduced weight of $\lfloor w/2 \rfloor$. An error is uncorrectable if it has weight at least $\lceil d/2 \rceil$. If we assume that each ancilla failure results in an error of weight $\lceil \Delta_g/2 \rceil$, then we only need t failures to cause an uncorrectable error, where t satisfies

$$t \cdot \lfloor \Delta_g/2 \rfloor \geq \lceil d/2 \rceil \, . \tag{3.74}$$

Then the probability of logical failure is p^t where $t \approx d/\Delta_g$.

This assumption is conservative — hook errors depend on the choice of syndrome-extraction circuit and may be minimized by particular choices of gate scheduling. For example, in the rotated surface code, there is a two-qubit gate schedule for the syndrome-extraction circuit such that the hook error has intersection-1 with a logical operator [TS14; CB18]. Using such a schedule, the below-threshold scaling is $\propto p^{\lceil d/2 \rceil}$ as one would expect from a depolarizing noise model. For general LDPC codes, the existence of measurement schedules that reduce the effects of hook errors is not yet clear.

While there exist many methods for suppressing hook errors such as Shor, Steane or Knill error correction [NC02], nearly all require more ancilla qubits. This presents a trade-off where, for a given number of qubits, either a larger block length code with correspondingly better parameters or a more resource-intensive syndrome-extraction circuit could be used. In the setting of a constant-rate LDPC code, larger distances

come with more logical qubits, so the lowest overhead solution is to use the naive syndrome-extraction circuit with as large a code as possible[18]. Later, in Section 3.6, we will propose a method to mitigate the effects of hook errors outside of the asymptotic regime.

Decoder performance for the outer code

Hook errors discussed in the previous section need to be considered in the context of the concatenated scheme — the probability that an ancilla qubit fails is $p^{(1)}$.

Following the discussion in Section 3.6 on hook errors, $\lceil \lceil d/2 \rceil / \lfloor \Delta_g/2 \rfloor \rceil = 10$ for the code selected in Section 3.6. Assuming a threshold of about 10^{-3} under circuit noise, the WER under circuit-level Pauli noise using long-range gates goes as

$$\left(\frac{p^{(1)}}{10^{-3}} \right)^{10} . \tag{3.75}$$

per cycle of syndrome extraction.

For context, a slightly better threshold of about 3×10^{-3} has been observed for $(3, 4)$ hypergraph product codes using efficient decoders [TDB21] under circuit-level noise for syndrome-extraction circuits with long-range gates.

In practice, more information is available to the decoder owing to the concatenated structure: A decoder using this extra information about individual qubit reliability is likely to have a better threshold. We return to this subject in Section 3.6.

Results

Using the duration of the hierarchical code syndrome extraction cycle as the unit of time that defines the WER, the results of the estimates are shown in Figure 3.18 for $r_{\mathsf{SWAP}} = 10^0$, 10^{-1}, or 10^{-2} and several sizes of inner rotated surface code. We can see the better scaling with gate error rate that the hierarchical memory achieves. While the LDPC code distance is fairly large, the "effective" distance has been reduced immensely by the weight-6 hook errors (potentially arising from the measurement of weight-13 check operators); because the outer code has distance $d = 119$, under our pessimistic assumptions just 10 fault locations are sufficient to cause an uncorrectable error. We expect future LDPC codes with better distance and better understanding of hook errors in syndrome extraction circuit gate scheduling will improve the WER scaling.

[18] For very resource-constrained settings, it may still be worthwhile to use more sophisticated syndrome extraction circuits for a better effective relative distance.

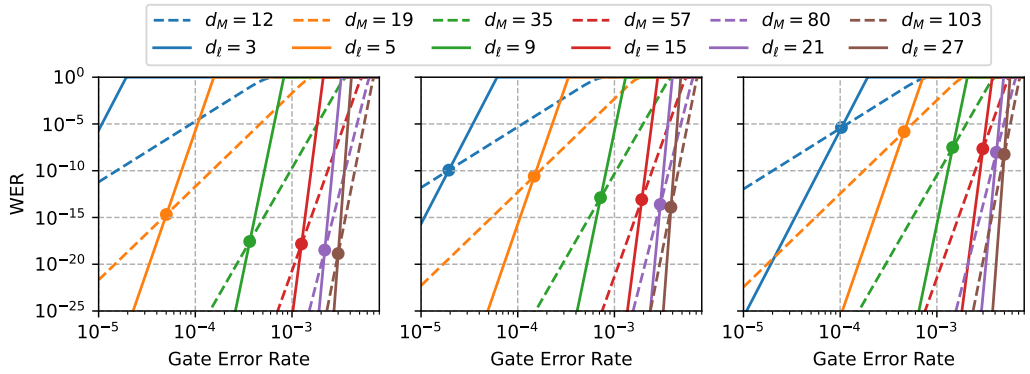


Figure 3.18: Comparison of a hierarchical memory (solid line) using a $(5, 8)$ quantum expander code with parameters $[\![1\,116\,416, 112\,896, 119]\!]$ and inner code distance d_ℓ, and a surface code (dashed line) with distance d_M. Lines of the same color use roughly the same number of physical qubits including all necessary ancilla qubits. All memories store 112 896 logical qubits. The 3 plots correspond to different values of r_{SWAP} equal to 10^0, 10^{-1}, and 10^{-2} (left to right) under the decoder performance assumptions made in Section 3.6. The surface code distance is rounded up, so it always uses slightly more qubits. The WER is with respect to the hierarchical memory syndrome extraction cycle.

Under a standard circuit-level noise model, below a gate error rate of around 10^{-3}, and for a target WER of 10^{-20} to 10^{-10}, the hierarchical scheme may realize significant resource savings. Especially so if SWAP gates have much lower gate error rates than CNOT gates. We plot such a comparison in Figure 3.19 with a gate error rate of 3×10^{-3} (99.7% gate fidelity) and $r_{\mathsf{SWAP}} = 0.1$ With further engineering and more careful modeling, we believe the overhead of the hierarchical scheme can be reduced much more, so that the crossover point occurs at a practically relevant gate error rate and target WER. In the next section we will outline two ideas that will improve the performance of the hierarchical scheme: The decoder for the outer code is given far more information about the level-1 qubit reliabilities than in a circuit level noise model, and in the presence of noise bias, the syndrome extraction circuit can be tailored to reduce the effects of hook errors.

Another reason we expect this estimate is conservative is that we have assumed that the noise model is independent circuit noise which creates only 2-body correlated errors in the underling surface codes. In the setting of large, long-lived quantum memories, we expect it will become necessary to address noise sources that affect large patches of the system. Sources of such noise could include cosmic rays

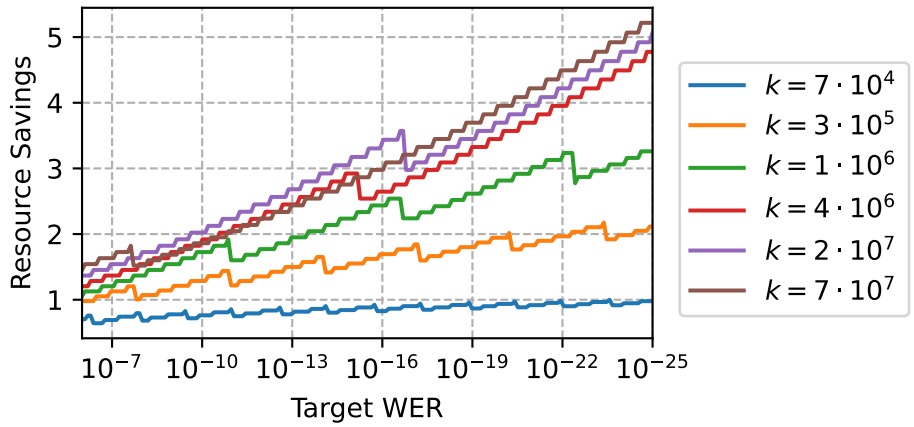


Figure 3.19: Estimated resource savings over surface codes for a hierarchical memory for $r_{\text{SWAP}} = 10^{-1}$ and a gate error rate of $3 \cdot 10^{-3}$ under the performance assumption of Section 3.6. The resource here refers to the total space footprint of the circuits; the y-axis represents the ratio $\mathcal{W}(C_M^{\mathcal{B}})/\mathcal{W}(C_N^{\mathcal{H}})$. We plot the resource savings for the $(4, 8)$ family of quantum expander codes with input code block lengths $512 \cdot 2^m$ for $m \in \{0, 1, 2, 3, 4, 5\}$. The number of logical qubits is indicated in the legend. Discontinuities in the plot are due to discretization of the surface code distance. Rare noise sources that create high weight errors may provide further resources savings over surface codes.

(superconducting qubits), large deviations in global control devices such as lasers (AMO systems), lightning strikes, power supply ripples, etc. For large memories, different parts of the memory may rely on systems operating independently (ex. lasers, fridges, power supplies, etc) which would make such "global" noise large on the scale of any reasonable surface code patch, but small on the scale of the full hierarchical memory. Concatenation of surface codes with constant length outer codes [Xu+22] has previously been considered in order to address such issues. It may be practical[19] to protect against such noise sources with a hierarchical scheme without additional overhead.

Future Performance Improvements

Having concluded a rough estimate of what the performance of the hierarchical memory might look like, we outline some ideas that could further improve the performance of the hierarchical memory relative to surface codes. In this section, we re-examine the WER for LDPC codes using biased-noise qubits and message-passing

[19] Physics is local, so a very large surface code is likely sufficient, but it may be impractically large.

decoders.

Noise-Bias Tailored Syndrome Extraction

As discussed in Section 3.6, hook errors can be very damaging for general LDPC codes. In this section, we estimate the failure rate for hierarchical codes by making further assumptions on the dependence of logical failure given η-biased qubits. In particular, Equation (3.76) presented below is an ansatz for the logical failure probability p_Q of the outer code. However, we expect that this estimate can be considerably improved in the future by investigating in more detail how p_Q and depends on the bias η.

X errors on the ancilla qubit will propagate to an X or Z error on the data while Z errors on the ancilla qubit will simply flip the measurement outcome without propagating to a higher weight data error. If X errors can be suppressed on the ancilla qubits, then hook errors become much less likely. In many platforms, such noise is common or can be engineered into the experiment [Gri+20; Les+20; Con+22]. Noise bias has been exploited in the past by tailoring the quantum error correction scheme [AP08; WBP15; Pur+20; Bon+21; Rof+22] to the noise.

In Section 3.4, we introduced a technique to modify the bilayer architecture such that Level-1 qubits are noise biased. We can use this noise bias to suppress errors on the ancilla (X) that propagate to higher weight data errors. We modify the assumptions of Subsection 3.6 and Equation (3.75) in a way that attempts to capture this behavior. Further study will be needed to make more precise estimates of logical error rates in this modified architecture.

The modified bilayer architecture uses elongated Level-1 qubits. If we choose the X distance to be larger than the Z distance according to $d_X = d_Z + \lceil 2\log(\eta)/\log(1/p)\rceil$, then the logical X error rate of the inner code is suppressed relative to the logical Z error rate by the bias factor $1/\eta$. If the accuracy threshold of the outer code is still 10^{-3} as we assumed for the case without noise bias, then for the modified architecture our estimate for the Level-2 WER becomes

$$p_Q = \left(\frac{p^{(1)}}{10^{-3}}\right)^{\lceil d/2\rceil} + \left(\frac{p^{(1)}/\eta}{10^{-3}}\right)^{\lceil\lceil d/2\rceil/\lfloor\Delta_g/2\rfloor\rceil} . \tag{3.76}$$

The first term is the contribution from Level-1 logical Z errors; these do not propagate from ancilla to data, so that $\lceil d/2\rceil$ Level-1 errors are needed to cause a logical error at level 2. The second term arises from Level-1 logical X errors. These can propagate from ancilla to data, but they occur at a rate suppressed by the bias factor $1/\eta$.

Since surface codes are CSS codes, the X and Z noise can be corrected independently, so the X and Z logical failure rates can be examined independently up to small correlations introduced by Y-errors. Ignoring these correlations and assuming that Equation (3.71) still holds with d replaced by d_X or d_Z,

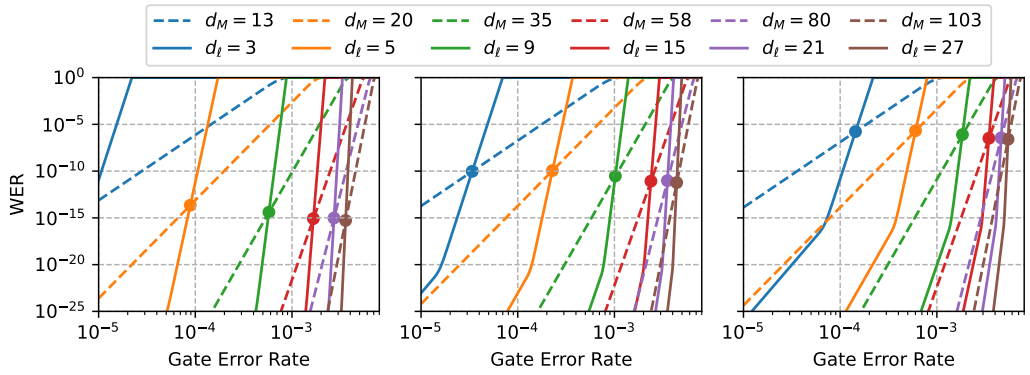


Figure 3.20: Comparison of a hierarchical memory (solid line) using a $(4, 8)$ quantum expander code with parameters $[\![327\,680, 65\,536, 32]\!]$ and inner code distance $d_Z = d_\ell$, and a surface code (dashed line) with distance ℓ_M. Lines of the same color use roughly the same number of physical qubits including all necessary ancilla qubits. The noise bias permits a smaller block length, so all memories store 65 536 logical qubits. The 3 plots correspond to different values of r_{SWAP} equal to 10^0, 10^{-1}, and 10^{-2} (left to right) under the modified decoder performance assumptions made in subsection 3.6. The surface codes underlying the hierarchical memory are rectangular with $d_X = 2d_\ell + 1$ The WER is for one round of the hierarchical memory's syndrome-extraction cycle.

We plot a similar comparison to Figure 3.18 with $d_X = 2d_Z + 1$, so that $\lceil d_X/2 \rceil = 2\lceil d_Z/2 \rceil$[20] (Figure 3.20). Using the greater resilience to hook errors, we also pick a smaller code with higher rate with parameters $[\![327\,680, 65\,536, 32]\!]$. Notice the effect of the bias is to increase the rate at which the WER falls with the level-0 gate error rate. The increased slope only persists until the two terms in Equation (3.76) become equal. One can see that using the bias, the hook errors are greatly suppressed leading to a better logical failure rate scaling in practically relevant regimes and a crossover point at a larger physical gate error rate.

[20]This choice is somewhat arbitrary. For a given WER target and gate error rate, the optimal aspect ratio is likely to be such that the target is at the "kink" of the WER in Figure 3.20.

Decoders that use the concatenated structure

Our asymptotic analysis used the underlying surface codes in a black-box manner—when decoding the outer LDPC code $\{Q_n\}$, the tiles had either failed or succeeded. In contrast to this "hard information", much more information is available to the decoder for the outer code in the hierarchical setting. We may have access to "soft information", i.e. information about how reliable individual surface code patches are, which can then be passed to the outer code decoder. It is known that maximum-likelihood decoding on each level of a concatenated code, together with message passing between levels is an optimal decoding algorithm [Pou06].

Choice of outer code decoder: Soft information can be used in the quantum setting using Belief Propagation (BP), a class of iterative algorithms. Broadly, in each iteration, BP makes a series of graph-local decisions—qubits that are in the support of a stabilizer generator exchange information and update their beliefs about whether they have been corrupted. As there are only a constant number of qubits in the support of each stabilizer generator, the decision requires a constant-sized computation. Although it is very successful in the classical setting, BP faces difficulties when applied to quantum codes. In the classical setting, BP converges to an distribution over bits that corresponds to the most likely error. In the quantum setting, it was pointed out early on [PC08] that degeneracy is a major issue for BP—there are many errors that are equivalent as they differ only by a stabilizer generator. However, BP is unable to tell the difference and gets stuck in a local minimum. One simple way to get around this issue would be if more information were available about the qubits. If each qubit were known to fail with a different probability – even if that difference is small — it can help BP avoid local minima.

Since then, many ideas have been developed to use soft information in the quantum setting that overcome the shortcomings of BP [PK21b; RWBC20; QVRC21; GGKL21; KL22; LP19; DMS22]. We now discuss ways to obtain soft information from the surface code.

Choice of inner code decoder: The tensor network decoders [BSV14; Chu21; TBF18; Bon+21; Tuc+19] are one class of surface code decoders that yield such soft information. The decoder outputs the probability of different (coset) logical failures for each tile. Unfortunately, it is unclear how to implement these algorithms in the fault-tolerant setting where syndrome information is unreliable. This setting requires growing bond dimension which makes implementing the decoder quite challenging.

More recently, BP decoders have been implemented for surface codes [RWBC20; OR22; Ach+22]. It is conceivable that such an algorithm could serve as a soft decoder for the surface codes as well.

A natural question is whether standard decoders such as Min-Weight Perfect Matching (MWPM) [DKLP02] or the Union-Find Decoder (UFD) [DN17] could be modified to yield soft information. For simplicity, consider bit flip noise at a rate of p. We define the *decoding graph* given by associating a vertex with each measured stabilizer generator. We add a special *boundary vertex* to which we associate the total parity of all measured stabilizer generators. Including the boundary vertex, each single qubit error is detected in exactly two places. For each error, an edge is added between the vertices where it is detected. To each edge, assign the weight $-\log\left(\frac{p}{1-p}\right)$ which is the log-likelihood of an error. The most likely error given the syndrome is then a subset of edges with minimal weight that produces the syndrome and can be computed efficiently by mapping onto the minimum-weight perfect matching problem.

On average, the expected weight of an error (and correction) will be linear in the block length. This is asymptotically larger than the distance of a surface code, so the most important feature of the correction is its *shape*. The Union-Find Decoder operates in two steps: First, it identifies clusters such that a valid correction is contained within the support of the clusters. Then, it treats the identified clusters as an erasure and runs an erasure correction decoder which produces a valid correction contained within the erasure.

One such way to obtain soft information from this process is to compute the log-likelihood of the minimum weight error that would lead to a logical fault when combined with the erasure. This can be computed efficiently by setting the edge weights within the erasure to 0 and computing the minimal weight path between inequivalent boundaries. Call this quantity ϕ. We note that when no errors are detected, $\phi = -d\log\left(\frac{p}{1-p}\right)$, and when the cluster spans the system, $\phi = 0$. In the first case, it is extremely unlikely ($\propto p^d$) for a logical fault to have occurred while in the latter case, there is a 50% probability for a logical fault to have occurred. When passed to an outer-level decoder, ϕ or a monotonic function of ϕ may yield sufficient information to improve the logical failure rate dramatically.

3.7 Conclusions

We have constructed a quantum memory with a threshold using geometrically local gates to simulate long-range connectivity. We did so by constructing a code family

$\{\mathcal{H}_N\}$ that we refer to as the hierarchical code. N indexes the size of the code; the N^{th} element $\mathcal{H}_N$ of this code is obtained by concatenating a constant-rate quantum LDPC code $\mathcal{Q}_n$ (the n-qubit outer code) and the surface code $\mathcal{RS}_\ell$ (the inner code). The outer code has a number of encoded qubits $k(n) = \rho \cdot n$ and distance $d(n) = \Theta(n^\delta)$ for positive constants ρ, δ. Our construction builds on Gottesman's proof of the existence of a threshold using quantum LDPC codes (Theorem 4 from [Got14]). The central idea in Gottesman's construction is that if the failure rate *per round* of syndrome extraction, denoted p_{round}, is a sufficiently small constant, then logical errors can be suppressed exponentially in the distance of the code. We showed that the requisite constant error rate per round can be achieved using geometrically-local gates if the inner code has suitable properties.

Although $\mathcal{H}_N$ is no longer an LDPC code, local operations suffice for extracting the error syndrome. In Section 3.4, we presented an explicit family of syndrome-extraction circuits $\{C_N^{\mathcal{H}}\}$ for $\mathcal{H}_N$. This circuit has width $\mathcal{W}(C_N^{\mathcal{H}}) = \Theta(N)$ and depth $\mathcal{T}(C_N^{\mathcal{H}}) = O(\sqrt{N}/R)$, where R denotes the range of physical SWAP gates. To describe this circuit for the hierarchical code, we first presented a construction of the syndrome-extraction circuit C_n for the outer LDPC code $\mathcal{Q}_n$ in Section 3.4. This circuit is based on a bilayer architecture — physical qubits are laid out in two layers in 2 dimensions. In our concatenated construction, the outer qubits of $\mathcal{Q}_n$ are replaced by rotated surface codes referred to as tiles. In Section 3.4, we demonstrated how to perform Level-1 logical Clifford operations on tiles using physical nearest-neighbor gates, including a novel technique for performing nearest-neighbor logical SWAP gates. We also discussed how to perform logical SWAP operations on tiles with range R_1 using physical SWAP operations with range R_0.

In Section 3.5, we showed that for fixed values of the physical failure rate p_{phys}, the error rate per round of syndrome-extraction, p_{round}, is a polynomial function of the depth $\mathcal{T}(C_N^{\mathcal{H}})$. Using an inner surface code with linear size ℓ, which can suppress errors exponentially in ℓ, we can guarantee that the Level-1 error rate per round is a constant by choosing $\ell = \Theta(\log(n))$. The resulting concatenated code $\mathcal{H}_N$ encodes a number of encoded qubits $K = \Omega(N/\log(N)^2)$. Furthermore, if the distance of the LDPC code $\mathcal{Q}_n$ is $d(n) = \Theta(n^\delta)$, then $\mathcal{H}_N$ can suppress errors superpolynomially; the Word Error Rate (WER) satisfies $p_{\mathcal{H}}(N) < \exp\left(-\Theta[N^\delta/\log^{2\delta}(N)]\right)$. Given access to physical SWAP operations of range R, the syndrome-extraction circuit $C_N^{\mathcal{H}}$ has depth $O(\sqrt{N}/R)$.

Using this architecture we made numerical estimates of the WER $p_{\mathcal{H}}(N)$ in Sec-

tion 3.6. We contrasted this with the WER $p_{\mathcal{B}}(M)$ of the *basic encoding* $\mathcal{B}_M$, where all logical qubits are encoded using only the surface code. We first made comparisons in the asymptotic regime, showing in Section 3.6 that if the outer constant-rate LDPC code has an efficient single-shot decoder, then a target logical error rate can be achieved more efficiently using the hierarchical encoding rather than the basic encoding.

We then proceeded with numerical estimates probing whether this advantage holds for practical code sizes and noise parameters. For this purpose, we compared the WERs of the basic encoding and hierarchical encoding when both schemes use the same total number of physical qubits. We found that the physical error rate has a *crossover point*; when the physical error rate is below this value, the hierarchical code outperforms the basic encoding. To perform these estimates, we made assumptions about the noise model and about the WER for surface codes and LDPC codes, and we assessed the impact of these assumptions on our conclusions. We also discussed some ways to reduce the WER of hierarchical codes by modifying the syndrome-extraction circuit, improving the fidelity of SWAP operations, and using more sophisticated decoding algorithms.

1. We made the conservative assumption that propagation of error from Level-1 ancilla qubits to Level-1 data qubits reduces the effective distance of the outer code by a factor of Δ_g, the degree of the outer-code stabilizer generators. This error propagation can be mitigated if the noise in Level-1 qubits is highly biased, with X errors occurring much less frequently than Z errors. Even if the noise afflicting the physical qubits is unbiased, this Level-1 noise bias can be enforced by using an asymmetric surface code as the inner code of the hierarchical scheme.

2. The failure rate of the outer code grows in proportion to the depth of the permutation routing, and hence is sensitive to the error rate of Level-1 SWAP operations. By improving the error rate of physical SWAP gates we can improve the performance of the hierarchical code significantly.

3. We assumed that the decoding algorithm for the outer code makes no use of the syndrome information from the inner code blocks. We expect that a much better decoding scheme for the hierarchical code can be achieved by exploiting such information from the inner code when decoding the outer code.

Finally, we also highlighted that a hierarchical architecture might deal effectively with "burst" errors that damage a large cluster of physical qubits simultaneously. A severe burst error could corrupt several of the inner-code tiles, but the resulting Level-1 erasure errors can be adequately addressed by the decoder for the outer code.

3.8 Acknowledgements

AK is supported by the Bloch Postdoctoral Fellowship from Stanford University. AK acknowledges funding from NSF award CCF-1844628. CAP acknowledges funding from the Air Force Office of Scientific Research (AFOSR), FA9550-19-1-0360. JP acknowledges funding from the U.S. Department of Energy Office of Science, Office of Advanced Scientific Computing Research, (DE-NA0003525, DE-SC0020290), the U.S. Department of Energy QuantISED program (DE-SC0018407), the U.S. Department of Energy Quantum Systems Accelerator, the Air Force Office of Scientific Research (FA9550-19-1-0360), and the National Science Foundation (PHY-1733907). The Institute for Quantum Information and Matter is an NSF Physics Frontiers Center. We thank Nicolas Delfosse, Mary Wootters, Anthony Leverrier, Nouédyn Baspin, Bailey Gu, Alex Kubica, David Schuster, Manuel Endres, Michael Vasmer, and Pavel Panteleev for helpful conversations.

3.9 Appendix

1. Set notation: for natural numbers $n \in \mathbb{N}$, $[n] = \{1, ..., n\}$.

2. Sums over sets: For a set S and a subset $A \subseteq S$, the sum $\sum_{B \supseteq A} f(B)$ is taken over all subsets $B \subseteq S$ such that $A \subseteq B$.

3. Asymptotics: for functions $f, g : \mathbb{N} \to \mathbb{R}$, we say

   a) $f(n) = O(g(n))$ if there exists an $n_0 \in \mathbb{N}$ and a positive number c independent of n such that for all $n > n_0$, $f(n) \leq g(n)$.

   b) $f(n) = \Omega(g(n))$ if $g(n) = O(f(g))$.

   c) $f(n) = \Theta(g(n))$ if there exists an $n_0 \in \mathbb{N}$ and positive numbers a, b independent of n such that $a \cdot g(n) \leq f(n) \leq b \cdot g(n)$.

   We may use $O_p(\cdot)$, $\Omega_p(\cdot)$ and $\Theta_p(\cdot)$ to indicate that the numbers a, b and c may depend on some parameter p pertinent to the problem at hand.

4. (Circuit) Step: A single timestep in which each qubit may participate in only one gate.

5. (Circuit) Stage: This refers to the time interval in the circuit $C_n^{\mathcal{Q}}$ required to simulate one entangling gate. One stage has at most $\mathcal{T}_{\text{perm}}$ steps.

6. (Measurement) Round: A complete measurement of all the stabilizer generators of the code producing one outcome for each stabilizer generator.

7. $\mathcal{K}$ is the set of Clifford operations we use to construct syndrome-extraction circuits in 2 dimensions. It includes the following elements.

   a) Initialization of new qubits in state $|0\rangle$ or $|+\rangle$,

   b) Single-qubit Pauli gates,

   c) Two-qubit Clifford gates CNOT and CZ between nearest-neighbor qubits,

   d) Single-qubit Pauli X and Z measurements,

   e) Physical SWAP operation with range R.

8. In the context of concatenated codes, $\mathcal{K}$ carries subscripts $\mathcal{K}_0$, $\mathcal{K}_1$ to refer to Level-0 (physical) and Level-1 (logical) Clifford operations.

3.10 Constructing the ideal syndrome-extraction circuit $(C_n^{\mathcal{Q}})^{\text{ideal}}$

In this section, we return to the claim in Section 3.4. We prove that the syndrome-extraction circuit $(C_n^{\mathcal{Q}})^{\text{ideal}}$ for a $[\![n, k, d, \Delta_q, \Delta_g]\!]$ code can be constructed such that its depth is at most $s := 2\Delta + 4$, where $\Delta = 2\max(\Delta_q, \Delta_g)$.

Proof. By definition, each qubit participates in at most Δ_q stabilizer generators and each stabilizer generator contains at most Δ_g qubits in its support. We use the *Tanner graph* $\mathcal{T}(\mathcal{Q}_n) = (V \cup C^{\mathsf{X}} \cup C^{\mathsf{Z}}, E)$, a tripartite graph corresponding to the code $\mathcal{Q}_n$ where:

1. There is a vertex $v \in V$ for each qubit in the code. $|V| = n$.

2. There is a vertex $u_i^{\mathsf{X}} \in$ for each X-type generator $\mathsf{S}_i^{\mathsf{X}}$. $|C^{\mathsf{X}}| = m_{\mathsf{X}}$.

3. There is a vertex w_j^{Z} for each Z-type generator $\mathsf{S}_j^{\mathsf{Z}}$. $|C^{\mathsf{Z}}| = m_{\mathsf{Z}}$.

Consider the bipartite Tanner graph $\mathcal{T}^{\mathsf{X}} = (V \cup C^{\mathsf{X}}, E)$ that corresponds to the X-type generators of the code $\mathcal{Q}$.

In each step, each qubit can be involved in at most one gate. This can be phrased as a graph coloring problem: we color the edges of $\mathcal{T}^{\mathsf{X}}$ such that no two edges incident

to a vertex have the same color. Since $\mathcal{T}^{\mathsf{X}}$ is bipartite, such an edge coloring can be computed efficiently using $\max(\Delta_q, \Delta_g)$ colors [Sch+03].

To measure the X-type syndromes, the first phase of the circuit $(C_n^Q)^{\text{ideal}}$ is partitioned into $\max(\Delta_q, \Delta_g)$ steps. In the t^{th} step, we perform the two-qubit gates corresponding to the edge color t.

Once completed, the same process is repeated for the Z-type syndromes. Following a similar line of reasoning, this requires $\Delta = \max(\Delta_q, \Delta_g)$ applications of two-qubit gates.

The circuit thus has two phases: first the X-type syndromes are measured followed by the Z-type syndromes [21] which completes a measurement of all stabilizer generators.

The total number of entangling stages is therefore 2Δ, where $\Delta = \max(\Delta_q, \Delta_g)$. Accounting for one stage for preparing and measuring ancilla qubits in each phase, we have a total of $s = 2\Delta + 4$ stages to measure syndromes. ■

[21] This is unlike the surface code where both types of syndromes are measured at once.