Reducing hardware requirements for fault-tolerant distributed quantum computers

de Bone, S.W.

DOI
10.4233/uuid:ba8ff0c4-196b-46ad-9d47-54fa02b459bc

Publication date
2024

Document Version
Final published version

Citation (APA)
de Bone, S. W. (2024). *Reducing hardware requirements for fault-tolerant distributed quantum computers*. [Dissertation (TU Delft), Delft University of Technology]. https://doi.org/10.4233/uuid:ba8ff0c4-196b-46ad-9d47-54fa02b459bc

Important note
To cite this publication, please use the final published version (if applicable).
Please check the document version above.


REDUCING HARDWARE
REQUIREMENTS FOR
fault-tolerant
DISTRIBUTED QUANTUM
COMPUTERS
SÉBASTIAN
DE BONE

Reducing hardware requirements for fault-tolerant distributed quantum computers

Proefschrift

ter verkrijging van de graad van doctor
aan de Technische Universiteit Delft,
op gezag van de Rector Magnificus prof. dr. ir. T. H. J. J. van der Hagen,
voorzitter van het College voor Promoties,
in het openbaar te verdedigen
op dinsdag 17 september 2024 om 12:30 uur

door

Sébastian Wicher DE BONE

Master of Science in Applied Physics,
Universiteit Twente, Enschede, Nederland,
geboren te Meppel, Nederland.

Dit proefschrift is goedgekeurd door de (co)promotoren.

Samenstelling promotiecommissie:

Rector Magnificus,	voorzitter
Prof. dr. S. D. C. Wehner,	Technische Universiteit Delft, promotor
Dr. E. Elkouss Coronas,	Technische Universiteit Delft, copromotor
Onafhankelijke leden:	
Prof. dr. D. E. Browne,	University College London, Verenigd Koninkrijk
Prof. dr. L. DiCarlo,	Technische Universiteit Delft
Dr. S. Santra,	Indian Institute of Technology, Mumbai, India
Dr. ir. J. S. S. M. Wong,	Technische Universiteit Delft
Prof. dr. B. M. Terhal,	Technische Universiteit Delft, reservelid

Keywords: Distributed quantum computing, multipartite entanglement generation, quantum error correction

Cover: Sébastian de Bone

Style: TU Delft House Style, with modifications by Moritz Beller

Contents

Summary ix

Samenvatting xi

List of Publications xiii

1 Perspective and overview 1
- 1.1 Introduction . . . 2
- 1.2 Thesis overview . . . 4

2 Basics of quantum computing 11
- 2.1 Introduction . . . 12
- 2.2 Quantum mechanics . . . 13
  - 2.2.1 Qubits . . . 13
  - 2.2.2 Density matrix . . . 14
  - 2.2.3 Measurement. . . . 15
  - 2.2.4 Entanglement . . . 17
- 2.3 Quantum computing . . . 17
  - 2.3.1 Operations . . . 17
  - 2.3.2 Pauli matrices . . . 18
  - 2.3.3 Other common quantum gates . . . 20
  - 2.3.4 Stabilizer operator . . . 22
  - 2.3.5 Measurement-based quantum computation. . . . 25
- 2.4 Noise models. . . . 27
  - 2.4.1 Noise channels . . . 28
  - 2.4.2 Noisy states . . . 30
  - 2.4.3 Noisy operations . . . 31
  - 2.4.4 Pauli twirling . . . 32

3 Quantum error correction 37
- 3.1 Introduction . . . 38
- 3.2 Stabilizer error-correction codes . . . 39
  - 3.2.1 Main properties . . . 39
  - 3.2.2 Code initialization . . . 40
  - 3.2.3 Error correction . . . 41
  - 3.2.4 Calderbank-Shor-Steane codes . . . 42
  - 3.2.5 Threshold . . . 44
- 3.3 $\mathbb{Z}_2$ chain complex and homology. . . . 45
  - 3.3.1 General idea . . . 45
  - 3.3.2 Example in three dimensions. . . . 46
  - 3.3.3 Cycles and homology groups. . . . 47

3.4 Two-dimensional topological codes . . . 48
3.4.1 Construction . . . 48
3.4.2 Commutation and independence requirements . . . 49
3.4.3 Error correction without measurement errors . . . 50
3.4.4 Surface code . . . 50
3.4.5 Smooth and rough lattice boundaries . . . 53
3.4.6 Distance and holes . . . 53
3.4.7 Universal quantum computing . . . 54
3.4.8 Error correction with measurement errors . . . 57
3.5 Three-dimensional cluster states . . . 59
3.5.1 Cluster state construction . . . 59
3.5.2 Foliation of two-dimensional topological codes . . . 60
3.5.3 Commutation requirements and logical subspace . . . 60
3.5.4 Error syndrome construction . . . 66
3.5.5 Error correction . . . 68
3.6 Entanglement distillation . . . 70
3.6.1 Distillation of general states . . . 71
3.6.2 Bipartite two-to-one stabilizer protocols . . . 72
3.6.3 Bipartite distillation with error-correction codes . . . 73
3.6.4 Bilocal Clifford protocols . . . 77
3.6.5 Double selection protocol . . . 81
3.6.6 Hashing protocol . . . 82
3.6.7 Multipartite two-to-one stabilizer protocols . . . 84
3.6.8 Multipartite distillation with error-correction codes . . . 90
3.6.9 GHZ creation and distillation with Bell pairs . . . 92

4 Hardware models for diamond color centers 101
4.1 Introduction . . . 102
4.2 Nitrogen-vacancy center model . . . 102
4.2.1 Electronic properties and operation . . . 103
4.2.2 Entanglement creation protocols . . . 104
4.2.3 Noise during entanglement generation . . . 105
4.2.4 Decoherence and dynamical decoupling . . . 107
4.2.5 Operation times and noise . . . 109
4.3 Bell pair model . . . 111
4.3.1 Electron-photon state . . . 111
4.3.2 Bell state measurement . . . 112
4.3.3 Single-click state . . . 113
4.3.4 Double-click state . . . 115
4.3.5 Entanglement protocol selection . . . 116

5 Creating and distilling multipartite GHZ states with Bell pairs 121
5.1 Introduction . . . 122
5.2 Bell and GHZ diagonal states . . . 123
5.3 Operations on Bell and GHZ diagonal states . . . 124
5.3.1 Fusion . . . 124

5.3.2 Non-local stabilizer measurements . . . 124
5.4 GHZ generation protocols . . . 125
5.5 Dynamic programs to optimize GHZ generation . . . 127
5.5.1 Base dynamic program . . . 127
5.5.2 Randomized version of the dynamic program . . . 129
5.5.3 Comparison between the dynamic programs . . . 133
5.6 Results . . . 133
5.6.1 Comparison with existing protocols for 4 parties . . . 133
5.6.2 Results for a large number of parties . . . 136
5.7 Conclusion . . . 137

6 Simulating GHZ protocols in the presence of memory decoherence 141
6.1 Introduction . . . 142
6.2 Binary tree protocols . . . 142
6.3 Protocol recipe construction . . . 145
6.3.1 Operation identification and ordering . . . 145
6.3.2 Operation scheduling . . . 145
6.4 Protocol recipe simulation . . . 146
6.5 Results and discussion . . . 147

7 Distributed surface code with memory decoherence 151
7.1 Introduction . . . 152
7.2 GHZ generation protocols revisited . . . 154
7.2.1 Prior GHZ protocols . . . 154
7.2.2 Dynamic program to optimize GHZ generation . . . 155
7.3 Distributed toric surface code . . . 159
7.3.1 Distributed toric code implementation . . . 159
7.3.2 Simulation process . . . 160
7.4 Diamond color center model . . . 162
7.5 Results . . . 163
7.5.1 Current state-of-the-art parameter set . . . 165
7.5.2 Near-term parameter sets . . . 165
7.5.3 GHZ cycle time sensitivity . . . 169
7.6 Discussion: feasibility of parameter sets . . . 172
7.7 Conclusion . . . 173

8 Fault-tolerant cluster states for distributed quantum computing 179
8.1 Introduction . . . 180
8.2 Methods . . . 181
8.2.1 Unit cell complex. . . . 182
8.2.2 Cell-vertex splitting . . . 183
8.2.3 Face-edge splitting . . . 183
8.2.4 Circuit-level and network noise . . . 185
8.2.5 Numerical tools and considerations . . . 187

8.3 Results . . . 188
8.3.1 Phenomenological thresholds . . . 188
8.3.2 Phenomenological thresholds with boundaries . . . 191
8.3.3 Monolithic thresholds . . . 192
8.3.4 Distributed thresholds . . . 194
8.3.5 Trade-off GHZ success and erasure probability . . . 197
8.4 Conclusion . . . 199

9 Conclusions and outlook 203
9.1 Summary of results . . . 204
9.2 Future work . . . 205
9.3 Outlook . . . 207

A Protocol recipe construction and simulation 213
A.1 Algorithm for protocol recipe construction . . . 213
A.1.1 Algorithm for operation identification and ordering . . . 213
A.1.2 Algorithm for operation scheduling . . . 214
A.2 Algorithm for protocol recipe simulation . . . 215
A.2.1 General description . . . 215
A.2.2 Step-by-step description . . . 217

B Superoperator calculation and convergence 219
B.1 Superoperator calculation . . . 219
B.2 Convergence of the average superoperator . . . 221

C Fitting procedure threshold plots 225
C.1 Regression model . . . 225
C.2 Weighted least-squares fitting procedure . . . 226
C.3 Uncertainty in fitting parameter values . . . 228

D Best-performing GHZ generation protocols 231

E Quotient boundaries in unit cell complex and crystal embeddings 237
E.1 Quotient boundaries in unit cell complex . . . 237
E.2 Crystal embeddings . . . 238

F Characterization of noisy channels 241

Acknowledgments 245

Curriculum Vitæ 249

Summary

Distributed quantum computers hold great promise in the realization of scalable and fault-tolerant quantum computers. They contain multiple nodes with small quantum devices that can generate inter-node entanglement. Next to realizing distributed architectures on a single chip, these systems can be extended to large-scale quantum networks. Connecting the nodes based on the topology of a quantum error-correction code forms an intuitive path toward fault tolerance.

Performing error-detection measurements in distributed error-correction codes requires the generation and consumption of entangled states. We focus on systems that are capable of generating remote two-qubit entanglement between pairs of connected nodes—*i.e.*, Bell pairs. Entangled states of higher weight—the so-called Greenberger-Horne-Zeilinger (GHZ) states—can be generated by fusing Bell pairs. On top of this, the quality of the generated entangled states can be increased with entanglement distillation. We implement dynamic programming to generate high-quality GHZ states by fusing and distilling Bell pairs.

The dynamic program allows us to optimize the quality of error-detection measurements for a specific distributed error-correction code: the (toric) surface code. We numerically evaluate the performance of this code with noise models based on experimental characterization of diamond color center hardware, including the typical behavior of memory decoherence in these devices. This leads to the identification of a threshold in the ratio between entanglement generation and the decoherence rates.

For a two-dimensional error-correction code like the surface code, performing error-detection over multiple time steps can be reinterpreted as measuring the qubits of a three-dimensional cluster state. This equivalence enables considering more general three-dimensional cluster states as fault-tolerant channels that transform the logical qubits of the underlying error-correction code. We use this idea to investigate distributed logical memory channels for general types of circuit-level and entanglement noise.

Our results show that efficient generation of high-quality entanglement and strategic design of error-correction channels are important aspects for developing noise-resilient distributed quantum computers.

Samenvatting

Gedistribueerde quantumcomputers bieden een veelbelovend perspectief voor de realisatie van schaalbare en fouttolerante quantumcomputers. Ze bevatten meerdere modules met kleine quantumapparaten die onderling verstrengeling kunnen genereren. Naast de realisatie van gedistribueerde architecturen op een enkele chip kunnen deze systemen ook worden toegepast op grootschalige quantumnetwerken. Hierbij biedt het verbinden van de modules volgens de topologie van een quantum-error-correctie-code een intuïtieve weg naar fouttolerantie.

De detectie van ongewenste fouten in gedistribueerde quantum-error-correctie-codes vereist het produceren en consumeren van verstrengelde toestanden. We concentreren ons op systemen die op afstand twee-qubit-verstrengeling, oftewel Bell-paren, kunnen genereren tussen twee verbonden netwerkmodules. Verstrengeling tussen meer dan twee qubits, de zogenaamde Greenberger-Horne-Zeilinger-toestanden (GHZ-toestanden), kunnen worden gegenereerd door Bell-paren te fuseren. Daarnaast kan met destillatie de kwaliteit van de geproduceerde verstrengeling worden vergroot. We gebruiken dynamisch programmeren om hoogwaardige GHZ-toestanden te produceren met fusie en destillatie van Bell-paren.

Dit dynamische programma maakt het mogelijk om de kwaliteit van error-detectie te optimaliseren voor een specifieke gedistribueerde error-correctie-code: de (torische) surface code. We evalueren het prestatievermogen van deze code numeriek met ruismodellen die gebaseerd zijn op experimentele karakterisatie van kleurcentra in diamant, inclusief de typische coherentietijden van qubits in deze systemen. Dit maakt het mogelijk een drempelwaarde te identificeren in de verhouding van de productiesnelheid van verstrengelde toestanden ten opzichte van de snelheid waarmee decoherentie optreedt.

Voor een tweedimensionale error-correctie-code, zoals de surface code, kan het uitvoeren van error-detectie in meerdere tijdsstappen ook worden geïnterpreteerd als het uitmeten van de qubits van een driedimensionale clustertoestand. Deze realisatie maakt het mogelijk om algemene driedimensionale clustertoestanden te beschouwen als fouttolerante kanalen die transformaties uitvoeren op de logische qubits van een onderliggende error-correctie-code. We gebruiken deze invalshoek om gedistribueerde logische geheugenkanalen te onderzoeken, in aanwezigheid van standaard ruis op operaties en verstrengelde toestanden.

Onze resultaten laten zien dat efficiënte productie van hoogwaardige verstrengeling en een strategisch ontwerp van error-correctie-kanalen belangrijke aspecten zijn bij de ontwikkeling van ruisbestendige gedistribueerde quantumcomputers.

List of Publications

8. S. Singh, F. Gu, S. de Bone, E. Villaseñor, D. Elkouss, and J. Borregaard, *Modular architectures and entanglement schemes for error-corrected distributed quantum computation*, arXiv: 2408.02837 [quant-ph], Aug. 2024.

7. S. de Bone, P. Möller, C. E. Bradley, T. H. Taminiau, and D. Elkouss, "Thresholds for the distributed surface code in the presence of memory decoherence," *AVS Quantum Science*, vol. 6, no. 3, p. 033801, July 2024.

   This article is included in Ch. 7 of this thesis.

6. K. Goodenough, S. de Bone, V. Addala, S. Krastanov, S. Jansen, D. Gijswijt, and D. Elkouss, "Near-term n to k distillation protocols using graph codes," *IEEE Journal on Selected Areas in Communications*, vol. 42, no. 7, pp. 1830–1849, July 2024.

5. Y. van Montfort, S. de Bone, and D. Elkouss, *Fault-tolerant structures for measurement-based quantum computation on a network*, arXiv: 2402.19323 [quant-ph], Feb. 2024.

   This manuscript is included in Ch. 8 of this thesis.

4. S. de Bone and D. Elkouss, "GHZ distillation protocols in the presence of decoherence," *ACM SIGMETRICS Performance Evaluation Review*, vol. 51, no. 2, pp. 81–83, Oct. 2023.

   This article is included in Ch. 6 of this thesis.

3. C. E. Bradley, S. W. de Bone, P. F. W. Möller, S. Baier, M. J. Degen, S. J. H. Loenen, H. P. Bartling, M. Markham, D. J. Twitchen, R. Hanson, D. Elkouss, and T. H. Taminiau, "Robust quantum-network memory based on spin qubits in isotopically engineered diamond," *npj Quantum Information*, vol. 8, no. 1, pp. 1–9, Oct. 2022

2. S. Jansen, K. Goodenough, S. de Bone, D. Gijswijt, and D. Elkouss, "Enumerating all bilocal Clifford distillation protocols through symmetry reduction," *Quantum*, vol. 6, p. 715, May 2022.

1. S. de Bone, R. Ouyang, K. Goodenough, and D. Elkouss, "Protocols for creating and distilling multipartite GHZ states with Bell pairs," *IEEE Transactions on Quantum Engineering*, vol. 1, pp. 1–10, Dec. 2020.

   This article is included in Ch. 5 of this thesis.

1

Perspective and overview

In this chapter, we set the stage: we provide a general introduction to clarify the position and significance of this thesis within the wider landscape of (theoretical) quantum computation. This allows us to contextualize the content of the thesis at the end of this chapter, where we briefly describe each chapter. Most of the elementary concepts discussed here are introduced in more detail in Chs. 2 and 3.

1

1.1 Introduction

Since their introduction throughout the twentieth century and further implementations in the twenty-first century, computers have changed our lives—both for better and for worse. All computers that we have today—whether they are PCs, laptops, smartphones, or smart devices—perform calculations by exploiting *classical* laws of physics. It is generally believed that these *classical computers* will soon be assisted by *quantum computers*: computers that use the laws of quantum mechanics to perform calculations.

For example, we have good reason to believe that quantum computers can simulate quantum mechanical systems themselves more efficiently than the computers we have today—*e.g.*, simulating atoms and molecules. This could lead to progress in many research fields, from chemistry—*e.g.*, the development of artificial fertilizer, medical drugs, or batteries—to understanding the physics of superconductivity. Moreover, quantum computing, and quantum(-enhanced) technologies in general, harbor the potential to yield advantages across other domains, such as, *e.g.*, solving algorithmic problems in optimization or database searching, as well as sensing, cryptography, secure communication, and (secure) time synchronization. It remains to be seen which of these applications prove to be most fruitful when operated on a quantum computer—this depends both on the development of quantum computers and their algorithms, as on the further development of classical computers and algorithms.

Currently, the realization of quantum computers is still in development. Examples of physical systems that are being investigated are ion traps, superconducting systems (*e.g.*, based on magnetic flux quanta or transmons), color centers in diamond (also known as defect centers), quantum dots, neutral atoms in optical lattices or flying photons (*i.e.*, light particles). Where the technology behind classical computers has converged to semiconductor-based transistors that can be operated with extremely high precision, the development of their quantum mechanical counterpart is still a dynamic and widespread research field, split between different candidate physical systems. Currently, each of these systems has its advantages and disadvantages, and it remains to be seen which system(s) can be adjusted to work in a setting with a large number of connected, high-precision calculation modules.

By themselves, many of the physical platforms in which quantum computing is being realized are not big enough for useful calculations. Therefore, at least in the years to come, one approach to building a full-fledged quantum computer is to rely on "linking" small quantum computers. For example, the sub-units may be elements mounted on a photonic chip, or form the nodes of a *quantum network* on a larger scale [1]. The goal of such an endeavor is to bundle the power of multiple quantum computers by combining their strengths and emulate a large-scale quantum computer by letting them perform calculations together. This approach is known as the *distributed* or *modular* path to quantum computing [2, 3]. The main advantage of the distributed approach is that scaling up the computer simply becomes a matter of adding more nodes or modules to the network, without the need to connect newly added modules to all parts of the network.

Next to being small, current quantum computers are also not yet precise enough to carry out useful calculations. It is expected that this remains to be the case in the years to come. In the first place, this lack of precision is caused by the operations used to perform calculations—*i.e.*, the elementary *gates*. These gates are inherently imperfect and introduce calculation errors. On top of that, there is the problem that, due to interaction with the

(classical) environment, quantum states lose their quantum properties as time progresses, and eventually fall back to states that are purely classical. We call this phenomenon (memory) *decoherence*. In current quantum computers, when compared to typical operation times, time scales in which quantum states lose their coherence are relatively small.

The substantial influence of noise illustrates the importance of *quantum error-correction codes*. These codes live by the virtue of redundancy: we typically use a large number of states to encode the information of a smaller number of *logical* states—*i.e.*, to store information collectively and non-locally over all available resources. If unwanted errors pop up during calculation or storage, we can exploit the code's redundancy to locate and correct these errors. In principle, each quantum error-correction code is designed to address a specific set of errors. For good codes, this set overlaps significantly with the errors most likely to occur. An implementation of such a code is considered *fault-tolerant* if the specific operations used to detect and correct errors (or to change the logical states) do not introduce new errors that exceed the code's correction abilities. Fault-tolerant implementations exhibit so-called *error thresholds* [4, 5], which characterize the maximum error probabilities the code can effectively handle. When an error probability remains below its threshold value, the code mainly encounters error combinations within its designed scope. However, above the threshold value, errors outside the correctable set become dominant, leading to logical failures. Overall, a *fault-tolerant quantum computer* is often described as one that operates a fault-tolerant error-correction code implementation below its error thresholds.

Typically, implementing a quantum error-correction code on a distributed quantum computer requires the creation and consumption of *entanglement* [6, 7] between the different nodes or modules. In this paradigm, a joint operation between two modules requires an entangled *Bell pair* [8], whereas a joint operation between three or more modules requires entanglement in the form of a *Greenberger-Horne-Zeilinger* (GHZ) state [9]. We require this entanglement to be of *high quality* if we want the operations of our distributed quantum computer to be of high quality as well. For distributed error correction, in particular, high-quality entanglement is important to reliably find errors. The need for high-quality entangled states can be considered a disadvantage of distributed quantum computers, as entangled states are required for both inter-node operations, and for detecting local errors with the error-correction code [10–14]. Luckily, we can make use of a concept known as *entanglement distillation* [15–17] to improve the quality of entangled states and relax the minimum requirements for elementary—*i.e.*, "pre-processed"—entanglement.

In this thesis, we consider systems capable of remotely generating entangled links between pairs of connected nodes. Several physical systems are suitable for generating this type of entanglement with coherent optical interfaces [18], such as the earlier mentioned *color centers* in diamond and *ion traps* [19, 20]. We primarily focus on color centers in diamond, which are also known as *defect centers*. Examples of these centers are *nitrogen-vacancy* centers [21–34], *silicon-vacancy* centers [35–39], and *tin-vacancy* centers [40–42].

On top of that, we mainly consider a specific error-correction code known as the *(toric) surface code* [4, 43]. A distributed implementation of this code requires the consumption of GHZ states between four network parties. We investigate **the best way to generate a high-quality GHZ state from Bell pairs**. To evaluate a specific GHZ generation protocol in practical situations, we should also take into account the **influence of memory**

1

decoherence. This then allows us to investigate the **minimal requirements to operate the distributed toric surface code fault-tolerantly** with diamond color centers. We note that this investigation is limited to diamond color centers that use Bell states as the most elementary form of entanglement.

In the final part of the thesis, we shift focus to **designing alternative error-correction channels for distributed quantum computers**. The motivation behind this research stems from the identification of a more general version of the (distributed) surface code channel by representing it as a *cluster state* [44]. Cluster states form the basis of a quantum computing paradigm called *measurement-based quantum computing* [45–47]. Identifying alternative error-correction channels can be considered as the first step towards tailoring (distributed) error-correction channels based on particular noise characteristics at play.

1.2 Thesis overview

The first part of this thesis contains three additional introductory chapters. In Ch. 2, we introduce relevant preliminaries in quantum mechanics, quantum information, and quantum computing. In Ch. 3, we describe relevant concepts of error-correction codes and discuss techniques to construct two-dimensional *topological surface codes* [4, 48, 49] and three-dimensional fault-tolerant cluster states. In Ch. 4, we discuss experimental control of nitrogen-vacancy centers and derive analytic models for Bell pairs that can be generated between two nitrogen-vacancy centers.

In the second part of the thesis, we numerically simulate states that are relevant for distributed toric surface codes. In Ch. 5, we search for protocols that create and distill GHZ states from Bell pairs. In Ch. 6, we describe how we operate these protocols in the presence of memory decoherence. In Ch. 7, we evaluate protocols from Ch. 5 in the presence of memory decoherence and use them to find noise thresholds for the distributed toric surface code with diamond color centers.

Subsequently, in Ch. 8, we construct alternative error-correction channels for distributed quantum computers. These channels arise in the form of fault-tolerant cluster states. We identify a method for constructing distributed versions of these cluster states and find noise thresholds for scenarios with non-perfect entangled states and noisy gates and measurements.

Finally, in Ch. 9, we summarize our findings, contextualize their significance, and expound on potential areas for future research. Below, we include a more detailed summary of the main chapters in this thesis:

Ch. 2 We introduce general concepts of quantum mechanics. After that, we discuss common operations in quantum computing, as well as the *stabilizer formalism* [50]. We briefly introduce the topic of measurement-based quantum computation. Finally, we focus on models for describing noise in theoretical and numerical descriptions of quantum computing.

Ch. 3 We discuss the basic elements of quantum error-correction codes. We zoom in on topological codes and their most important example: the surface code. We discuss fault-tolerant cluster states that can be used for measurement-based quantum computation. Finally, we discuss the main concepts of entanglement distillation and its relation to error-correction codes.

Ch. 4 We give an overview of the models used for describing nitrogen-vacancy centers in diamond. We discuss the experimental characterization of nitrogen-vacancy centers and introduce models and derivations for the two main protocols that can be used to generate Bell pairs between two nitrogen-vacancy centers.

Ch. 5 We investigate the creation and distillation of GHZ states out of non-perfect Bell pair states that are diagonal in the *Bell basis*. We introduce a common framework for distillation and *fusion* of diagonal Bell and GHZ states. Finally, we make use of a heuristic dynamic programming algorithm to optimize over a large class of GHZ protocols and compare the performance of these protocols against prior protocols in situations without decoherence, gate noise, and measurement noise.

Ch. 6 We provide details on how we convert the GHZ protocols from Ch. 5 into a practical set of instructions, which we call a *protocol recipe*. This allows us to simulate these protocols in the presence of gate and measurement noise, and memory decoherence.

Ch. 7 We present numerical simulations for the distributed toric surface code with color centers in diamond. In these simulations, each data qubit of the code is part of a separate color center. We use the algorithms of Chs. 5 and 6 to find well-performing GHZ creation and distillation protocol for each noise configuration considered. We investigate the effect of memory decoherence and other noise sources. Among other results, this allows us to identify a threshold in the ratio between entanglement generation and memory decoherence rates.

Ch. 8 We introduce a method for constructing distributed versions of fault-tolerant cluster states that can be used for measurement-based quantum computation. We identify error thresholds for gate, measurement, and network noise in these three-dimensional cluster states. We show that the high thresholds of non-conventional lattices against *erasure errors*—*i.e.*, qubit loss—can be used to relax the quality requirements of the entangled states. This is because they make it possible to work with a lower entanglement success rate—*i.e.*, use more steps of entanglement distillation.

References

[1] R. Van Meter and S. J. Devitt, "The path to scalable distributed quantum computing," *Computer*, vol. 49, pp. 31–42, Sept. 2016.

[2] L. K. Grover, *Quantum Telecomputation*. arXiv: quant-ph/9704012, Apr. 1997.

[3] J. I. Cirac, A. K. Ekert, S. F. Huelga, and C. Macchiavello, "Distributed quantum computation over noisy channels," *Physical Review A*, vol. 59, pp. 4249–4254, June 1999.

[4] E. Dennis, A. Kitaev, A. Landahl, and J. Preskill, "Topological quantum memory," *Journal of Mathematical Physics*, vol. 43, pp. 4452–4505, Sept. 2002.

[5] C. Wang, J. Harrington, and J. Preskill, "Confinement-Higgs transition in a disordered gauge theory and the accuracy threshold for quantum memory," *Annals of Physics*, vol. 303, pp. 31–58, Jan. 2003.

[6] A. Einstein, B. Podolsky, and N. Rosen, "Can quantum-mechanical description of physical reality be considered complete?," *Physical Review*, vol. 47, pp. 777–780, May 1935.

[7] E. Schrödinger, "Discussion of probability relations between separated systems," *Mathematical Proceedings of the Cambridge Philosophical Society*, vol. 31, pp. 555–563, Oct. 1935.

[8] J. S. Bell, "On the Einstein Podolsky Rosen paradox," *Physics Physique Fizika*, vol. 1, pp. 195–200, Nov. 1964.

[9] D. M. Greenberger, M. A. Horne, and A. Zeilinger, *Going beyond Bell's Theorem.* arXiv: 0712.0921 [quant-ph], Dec. 2007.

[10] L. Jiang, J. M. Taylor, A. S. Sørensen, and M. D. Lukin, "Distributed quantum computation based on small quantum registers," *Physical Review A*, vol. 76, p. 062323, Dec. 2007.

[11] R. Van Meter, T. D. Ladd, A. G. Fowler, and Y. Yamamoto, "Distributed quantum computation architecture using semiconductor nanophotonics," *International Journal of Quantum Information*, vol. 08, pp. 295–323, Feb. 2010.

[12] K. Fujii, T. Yamamoto, M. Koashi, and N. Imoto, *A Distributed Architecture for Scalable Quantum Computation with Realistically Noisy Devices.* arXiv: 1202.6588 [quant-ph], Feb. 2012.

[13] Y. Li and S. C. Benjamin, "High threshold distributed quantum computing with three-qubit nodes," *New Journal of Physics*, vol. 14, p. 093008, Sept. 2012.

[14] C. Monroe, R. Raussendorf, A. Ruthven, K. R. Brown, P. Maunz, L.-M. Duan, and J. Kim, "Large scale modular quantum computer architecture with atomic memory and photonic interconnects," *Physical Review A*, vol. 89, p. 022317, Feb. 2014.

[15] D. Deutsch, A. Ekert, R. Jozsa, C. Macchiavello, S. Popescu, and A. Sanpera, "Quantum privacy amplification and the security of quantum cryptography over noisy channels," *Physical Review Letters*, vol. 77, no. 13, pp. 2818–2821, 1996.

[16] C. H. Bennett, G. Brassard, S. Popescu, B. Schumacher, J. A. Smolin, and W. K. Wootters, "Purification of noisy entanglement and faithful teleportation via noisy channels," *Physical Review Letters*, vol. 76, pp. 722–725, Jan. 1996.

[17] C. H. Bennett, D. P. DiVincenzo, J. A. Smolin, and W. K. Wootters, "Mixed-state entanglement and quantum error correction," *Physical Review A - Atomic, Molecular, and Optical Physics*, vol. 54, no. 5, pp. 3824–3851, 1996.

[18] D. D. Awschalom, R. Hanson, J. Wrachtrup, and B. B. Zhou, "Quantum technologies with optically interfaced solid-state spins," *Nature Photonics*, vol. 12, pp. 516–527, Sept. 2018.

[19] D. Hucul, I. V. Inlek, G. Vittorini, C. Crocker, S. Debnath, S. M. Clark, and C. Monroe, "Modular entanglement of atomic qubits using photons and phonons," *Nature Physics*, vol. 11, pp. 37–42, Jan. 2015.

[20] R. Nigmatullin, C. J. Ballance, N. de Beaudrap, and S. C. Benjamin, "Minimally complex ion traps as modules for quantum communication and computing," *New Journal of Physics*, vol. 18, p. 103028, Oct. 2016.

[21] H. Bernien, B. Hensen, W. Pfaff, G. Koolstra, M. S. Blok, L. Robledo, T. H. Taminiau, M. Markham, D. J. Twitchen, L. Childress, and R. Hanson, "Heralded entanglement between solid-state qubits separated by three metres," *Nature*, vol. 497, pp. 86–90, May 2013.

[22] T. H. Taminiau, J. Cramer, T. Van Der Sar, V. V. Dobrovitski, and R. Hanson, "Universal control and error correction in multi-qubit spin registers in diamond," *Nature Nanotechnology*, vol. 9, no. 3, pp. 171–176, 2014.

[23] J. Cramer, N. Kalb, M. A. Rol, B. Hensen, M. S. Blok, M. Markham, D. J. Twitchen, R. Hanson, and T. H. Taminiau, "Repeated quantum error correction on a continuously encoded qubit by real-time feedback," *Nature Communications*, vol. 7, pp. 1–7, May 2016.

[24] A. Reiserer, N. Kalb, M. S. Blok, K. J. M. van Bemmelen, T. H. Taminiau, R. Hanson, D. J. Twitchen, and M. Markham, "Robust quantum-network memory using decoherence-protected subspaces of nuclear spins," *Physical Review X*, vol. 6, p. 021040, June 2016.

[25] D. Riedel, I. Söllner, B. J. Shields, S. Starosielec, P. Appel, E. Neu, P. Maletinsky, and R. J. Warburton, "Deterministic enhancement of coherent photon generation from a nitrogen-vacancy center in ultrapure diamond," *Physical Review X*, vol. 7, p. 031040, Sept. 2017.

[26] N. Kalb, A. A. Reiserer, P. C. Humphreys, J. J. Bakermans, S. J. Kamerling, N. H. Nickerson, S. C. Benjamin, D. J. Twitchen, M. Markham, and R. Hanson, "Entanglement distillation between solid-state quantum network nodes," *Science*, vol. 356, pp. 928–932, June 2017.

[27] P. C. Humphreys, N. Kalb, J. P. J. Morits, R. N. Schouten, R. F. L. Vermeulen, D. J. Twitchen, M. Markham, and R. Hanson, "Deterministic delivery of remote entanglement on a quantum network," *Nature*, vol. 558, pp. 268–273, June 2018.

[28] C. E. Bradley, J. Randall, M. H. Abobeih, R. C. Berrevoets, M. J. Degen, M. A. Bakker, M. Markham, D. J. Twitchen, and T. H. Taminiau, "A ten-qubit solid-state spin register with quantum memory up to one minute," *Physical Review X*, vol. 9, Sept. 2019.

[29] M. H. Abobeih, J. Randall, C. E. Bradley, H. P. Bartling, M. A. Bakker, M. J. Degen, M. Markham, D. J. Twitchen, and T. H. Taminiau, "Atomic-scale imaging of a 27-nuclear-spin cluster using a quantum sensor," *Nature*, vol. 576, pp. 411–415, Dec. 2019.

[30] M. Ruf, M. Weaver, S. van Dam, and R. Hanson, "Resonant excitation and Purcell enhancement of coherent nitrogen-vacancy centers coupled to a Fabry-Perot microcavity," *Physical Review Applied*, vol. 15, p. 024049, Feb. 2021.

[31] M. Pompili, S. L. N. Hermans, S. Baier, H. K. C. Beukers, P. C. Humphreys, R. N. Schouten, R. F. L. Vermeulen, M. J. Tiggelman, L. dos Santos Martins, B. Dirkse, S. Wehner, and R. Hanson, "Realization of a multinode quantum network of remote solid-state qubits," *Science*, vol. 372, pp. 259–264, Apr. 2021.

[32] M. H. Abobeih, Y. Wang, J. Randall, S. J. H. Loenen, C. E. Bradley, M. Markham, D. J. Twitchen, B. M. Terhal, and T. H. Taminiau, "Fault-tolerant operation of a logical qubit in a diamond quantum processor," *Nature*, vol. 606, pp. 884–889, June 2022.

[33] C. E. Bradley, S. W. de Bone, P. F. W. Möller, S. Baier, M. J. Degen, S. J. H. Loenen, H. P. Bartling, M. Markham, D. J. Twitchen, R. Hanson, D. Elkouss, and T. H. Taminiau, "Robust quantum-network memory based on spin qubits in isotopically engineered diamond," *npj Quantum Information*, vol. 8, pp. 1–9, Oct. 2022.

[34] L. Orphal-Kobin, K. Unterguggenberger, T. Pregnolato, N. Kemf, M. Matalla, R.-S. Unger, I. Ostermay, G. Pieplow, and T. Schröder, "Optically coherent nitrogen-vacancy defect centers in diamond nanostructures," *Physical Review X*, vol. 13, p. 011042, Mar. 2023.

[35] A. Sipahigil, R. E. Evans, D. D. Sukachev, M. J. Burek, J. Borregaard, M. K. Bhaskar, C. T. Nguyen, J. L. Pacheco, H. A. Atikian, C. Meuwly, R. M. Camacho, F. Jelezko, E. Bielejec, H. Park, M. Lončar, and M. D. Lukin, "An integrated diamond nanophotonics platform for quantum-optical networks," *Science*, vol. 354, pp. 847–850, Nov. 2016.

[36] D. D. Sukachev, A. Sipahigil, C. T. Nguyen, M. K. Bhaskar, R. E. Evans, F. Jelezko, and M. D. Lukin, "Silicon-vacancy spin qubit in diamond: A quantum memory exceeding 10 ms with single-shot state readout," *Physical Review Letters*, vol. 119, p. 223602, Nov. 2017.

[37] C. T. Nguyen, D. D. Sukachev, M. K. Bhaskar, B. MacHielse, D. S. Levonian, E. N. Knall, P. Stroganov, C. Chia, M. J. Burek, R. Riedinger, H. Park, M. Lončar, and M. D. Lukin, "An integrated nanophotonic quantum register based on silicon-vacancy spins in diamond," *Physical Review B*, vol. 100, p. 165428, Oct. 2019.

[38] C. T. Nguyen, D. D. Sukachev, M. K. Bhaskar, B. MacHielse, D. S. Levonian, E. N. Knall, P. Stroganov, R. Riedinger, H. Park, M. Lončar, and M. D. Lukin, "Quantum network nodes based on diamond qubits with an efficient nanophotonic interface," *Physical Review Letters*, vol. 123, p. 183602, Oct. 2019.

[39] C. M. Knaut, A. Suleymanzade, Y.-C. Wei, D. R. Assumpcao, P.-J. Stas, Y. Q. Huan, B. Machielse, E. N. Knall, M. Sutula, G. Baranes, N. Sinclair, C. De-Eknamkul, D. S. Levonian, M. K. Bhaskar, H. Park, M. Lončar, and M. D. Lukin, "Entanglement of nanophotonic quantum memory nodes in a telecom network," *Nature*, vol. 629, pp. 573–578, May 2024.

[40] T. Iwasaki, Y. Miyamoto, T. Taniguchi, P. Siyushev, M. H. Metsch, F. Jelezko, and M. Hatano, "Tin-vacancy quantum emitters in diamond," *Physical Review Letters*, vol. 119, p. 253601, Dec. 2017.

[41] A. E. Rugar, S. Aghaeimeibodi, D. Riedel, C. Dory, H. Lu, P. J. McQuade, Z.-X. Shen, N. A. Melosh, and J. Vučković, "Quantum photonic interface for tin-vacancy centers in diamond," *Physical Review X*, vol. 11, p. 031021, July 2021.

[42] R. Debroux, C. P. Michaels, C. M. Purser, N. Wan, M. E. Trusheim, J. Arjona Martínez, R. A. Parker, A. M. Stramma, K. C. Chen, L. de Santis, E. M. Alexeev, A. C. Ferrari, D. Englund, D. A. Gangloff, and M. Atatüre, "Quantum control of the tin-vacancy spin qubit in diamond," *Physical Review X*, vol. 11, p. 041041, Nov. 2021.

[43] S. B. Bravyi and A. Y. Kitaev, *Quantum Codes on a Lattice with Boundary.* arXiv: quant-ph/9811052, Nov. 1998.

[44] H. J. Briegel and R. Raussendorf, "Persistent entanglement in arrays of interacting particles," *Physical Review Letters*, vol. 86, pp. 910–913, Jan. 2001.

[45] R. Raussendorf and H. J. Briegel, "A one-way quantum computer," *Physical Review Letters*, vol. 86, pp. 5188–5191, May 2001.

[46] R. Raussendorf and H. Briegel, *Computational Model Underlying the One-Way Quantum Computer.* arXiv: quant-ph/0108067, Mar. 2002.

[47] R. Raussendorf, D. E. Browne, and H. J. Briegel, "Measurement-based quantum computation on cluster states," *Physical Review A*, vol. 68, p. 022312, Aug. 2003.

[48] A.Yu. Kitaev, "Fault-tolerant quantum computation by anyons," *Annals of Physics*, vol. 303, pp. 2–30, Jan. 2003.

[49] H. Bombín and M. A. Martin-Delgado, "Topological quantum distillation," *Physical Review Letters*, vol. 97, p. 180501, Oct. 2006.

[50] D. Gottesman, *Stabilizer Codes and Quantum Error Correction.* Doctoral thesis, California Institute of Technology, Pasadena, California, United States, May 1997.

2

Basics of quantum computing

In this chapter, we introduce general concepts, ideas, and formulations relevant to the field. To this end, we first introduce general aspects of quantum mechanics. After that, we discuss common operations in quantum computing, as well as the stabilizer formalism. On top of that, we briefly introduce the topic of measurement-based quantum computation that is relevant for Ch. 8. At the end of this chapter, we lay down the groundwork for modeling the main adversary of our work: noise.

2.1 Introduction

What today's computers have in common is that their calculation process obeys classical mechanics. Therefore, we typically refer to these computers as "classical computers". Classical computers perform their calculations with *bits* that can either be in a first state—*i.e.*, the 0-state—or in a second state—*i.e.*, the 1-state. All calculations can be decomposed into a number of logical gates applied to a set of bits of this form. Each of these bits is in a "fixed" state—*i.e.*, either the 0-state or the 1-state—at each stage of the calculation.

Quantum computers can be considered as more general versions of classical computers. That is because, the same principle applies to quantum computers, with the important difference that there is also a quantum mechanical *superposition* possible between the 0-state and 1-state. This gives rise to a more general version of a bit: a *qubit*, or *quantum bit*. In terms of the 0-state and 1-state of the classical bit, this means that each qubit has a probability $|\alpha|^2$ of being in the 0-state and a probability $|\beta|^2$ of being in the 1-state. Here, α and β are complex numbers. Following the laws of quantum mechanics, upon measuring a qubit, it will *collapse* to one of the two states in its superposition. Therefore, one can argue that measuring a qubit transforms it into a classical bit.

As mentioned in Ch. 1, in this thesis, we focus on so-called distributed implementations of quantum computers. These implementations consist of multiple small quantum computers that are connected via entanglement. Entanglement can be considered as one of the characteristic pillars of quantum mechanics. As introduced more formally in Sec. 2.2.4, an entangled state arises when it is no longer possible to describe the combined state of two or more parties in terms of the states of the individual parties. This effect remains when the parties are separated by a large distance. In some configurations, these parties can even be considered to be "perfectly correlated"—even to an extent beyond what is possible with classical mechanics.

At the end of this chapter, we discuss the concept of noise. Here, noise can be considered as errors appearing on (qu)bits during calculation or storage. Typically, we model noise as a set of stochastic operations that act on the state: if the influence of noise is minimal, the probability of such an erroneous operation is small, whereas, in a situation with a lot of noise, that probability is large. To limit the number of scenarios that can occur as a result of noise, one typically makes use of a limited set of operations. Limiting the number of noise scenarios is useful to make analytical or numerical calculations feasible. More often than not, the operations that describe noise are limited to or can be expressed as *Pauli matrices*, which are discussed in Sec. 2.3.2.

Modeling noise with Pauli matrices is convenient because Pauli matrix operations are common gates in quantum computing. Even more so, Pauli matrices form the basis of the stabilizer formalism, which allows one to efficiently describe a subset of quantum states—*i.e.*, the *stabilizer states*. Making use of the stabilizer formalism is another simplification that makes calculations more feasible. Calculating the state of a stabilizer state after applying a Pauli operator is as easy as considering *(anti-)commutation rules* between the Pauli operator and the *stabilizer operators* of the state, which we discuss in Sec. 2.3.2 as well.

It should be mentioned that in practical quantum computers, it is not useful to only utilize stabilizer states. That is because quantum computers with only stabilizer states offer only a polynomial advantage over classical computers when it comes to the computational complexity they can reach [1, 2]. This can be understood by realizing that we only need $2N^2$

classical bits of information to describe an N-qubit stabilizer state. To describe an N-qubit general quantum state, however, we need 2^N complex numbers (not taking into account normalization)—showing that, in general, quantum computers have the potential to solve a problem in exponentially fewer calculation steps than classical computers. Therefore, in practical quantum computers, we have to make sure we can also create quantum states that are not stabilizer states to accomplish *universal* quantum computing.

A specific class of stabilizer states is formed by the cluster states. These states can be used for the so-called measurement-based implementation of quantum computing. Here, we prepare a cluster state that can be used for any purpose. Specific quantum algorithms can then be implemented by measuring out the qubits of this cluster state. The *measurement basis* (see Sec. 2.2.3) in which the measurements are applied determines the exact logic of the calculation. In the measurement-based paradigm, even though the resource states are stabilizer states, this freedom in the measurement basis allows one to achieve universal operations.

2.2 Quantum mechanics

In this section, we introduce the basics of quantum mechanics. We discuss how qubits are defined in quantum mechanics (Sec. 2.2.1) and elaborate on density matrices (Sec. 2.2.2), measurements (Sec. 2.2.3), and entangled states (Sec. 2.2.4).

2.2.1 Qubits

A qubit serves as the basis of quantum information. In classical mechanics, states are always unambiguously defined as a single combination of bits, but in quantum mechanics, a particle or system can simultaneously be in different states. As discussed in Sec. 2.1, in its most general form a single qubit can be described as $\alpha|0\rangle + \beta|1\rangle$. Here, $\alpha, \beta \in \mathbb{C}$ are complex values that take on the role of probability amplitudes: $|\alpha|^2$ can be considered as the probability of the qubit being in the classical state $|0\rangle$ and $|\beta|^2$ as the probability of the qubit being in the classical state $|1\rangle$. To make sure the probabilities add up to one, $|\alpha|^2 + |\beta|^2 = 1$ should hold. A qubit can be in infinitely many states based on the exact values of α and β. Therefore, we typically describe a qubit mathematically as a two-dimensional vector with complex coefficients—*i.e.*, as a vector in the space $\mathbb{C}^2$. In this interpretation, $|0\rangle$ and $|1\rangle$ represent the basis states and we denote vectors with Dirac's *bra-ket* notation—*i.e.*, a quantum state vector labeled ψ is written as $|\psi\rangle$ and its Hermitian (*i.e.*, conjugate) transpose as $\langle\psi| = |\psi\rangle^\dagger = (|\psi\rangle^*)^T$. Here, $|\psi\rangle^*$ denotes the complex conjugate of $|\psi\rangle$. The condition $|\alpha|^2 + |\beta|^2 = 1$ dictates that a valid quantum state $|\psi\rangle$ is always a *normalized* vector that fulfills $\langle\psi|\psi\rangle = 1$. The normalization condition means that the state of a qubit can be visualized as a vector on the surface of a sphere. This sphere is commonly referred to as the *Bloch sphere* and is depicted in Fig. 2.1a. Interpreting the state of a qubit on the Bloch sphere is convenient because it makes operations on the qubit more intuitive to comprehend. We show examples of this in Figs. 2.1b, 2.1c, 2.2a and 2.2b.

We specifically refer to the basis with basis states $|0\rangle$ and $|1\rangle$ as the *computational basis*. The most common vectors chosen for the computational basis states are:

$$|0\rangle \equiv \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \quad |1\rangle \equiv \begin{bmatrix} 0 \\ 1 \end{bmatrix}. \tag{2.1}$$

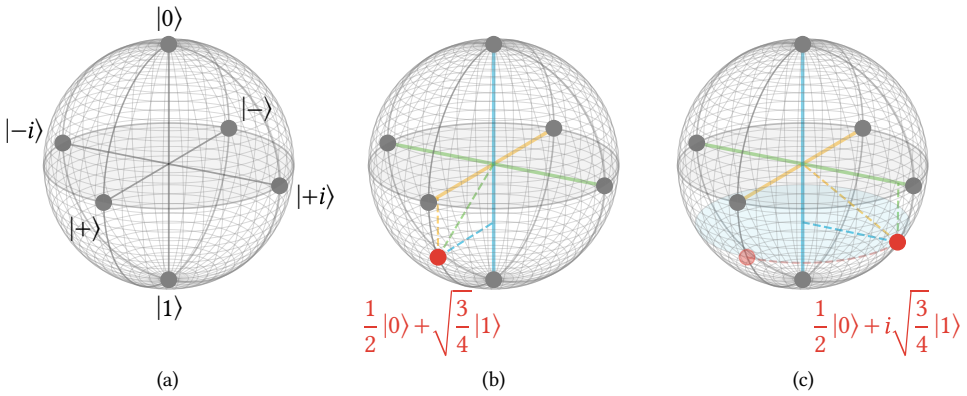


Figure 2.1: (a) The state of a qubit $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$ can be visualized as a vector (or as a point) on the surface of the Bloch sphere. On this sphere, next to the states $|0\rangle$ and $|1\rangle$, which are often defined as the eigenstates of the Pauli-Z matrix, the states $\{|+\rangle, |-\rangle\}$ and $\{|+i\rangle, |-i\rangle\}$ on opposite sides of the sphere are typically the eigenstates of the Pauli-X and Pauli-Y matrix, respectively. The Pauli matrices are introduced in Sec. 2.3.2 and can be used to calculate the Bloch sphere coordinates $(\mathrm{Tr}(X\rho), \mathrm{Tr}(Y\rho), \mathrm{Tr}(Z\rho))$ of a state ρ. (b) Example of a state on the Bloch sphere (red dot). The projection of this state on the blue, yellow, or green axis describes the measurement probabilities for measuring the qubit in the $\{|0\rangle, |1\rangle\}$, $\{|+\rangle, |-\rangle\}$, or $\{|+i\rangle, |-i\rangle\}$ basis, respectively. For example, measuring this state in the yellow $\{|+\rangle, |-\rangle\}$ basis has $\approx 93\%$ probability of resulting in $|+\rangle$ and $\approx 7\%$ probability of resulting in $|-\rangle$. Measurements are described in Sec. 2.2.3. (c) Example of a different state on the Bloch sphere (dark red dot). We see that, compared to the state in (b), the relative phase factor i in front of $|1\rangle$ leads to a $\pi/2$ rotation around the blue axis. As indicated by the projections on the three colored axes, this phase shift influences the measurement probabilities for two of the three measurement bases.

Qubits can also be described with another set of orthonormal basis states $\{|\phi_k\rangle\}_k$, as, *e.g.*, a basis defined by the states $|+\rangle \equiv (|0\rangle + |1\rangle)/\sqrt{2}$ and $|-\rangle \equiv (|0\rangle - |1\rangle)/\sqrt{2}$. We note that $\langle\psi|\phi\rangle$ represents the *inner product* between two states $|\psi\rangle$ and $|\phi\rangle$ and results in a scalar outcome, whereas the term $|\psi\rangle\langle\phi|$ describes a matrix. This principle can be used in combination with a basis to describe a general matrix $\mathcal{O}$ in terms of its matrix elements o_{ij} as $\mathcal{O} = \sum_{i,j} o_{ij} |\phi_i\rangle\langle\phi_j|$, where $|\phi_i\rangle$ and $|\phi_j\rangle$ are the basis vectors—*e.g.*, the basis vectors of the computational basis.

If we consider a system of multiple qubits, we have to increase the size of the vector space that describes their states. Each additional qubit doubles the size of the vector space. If we have multiple qubits that can be separately described as single qubits, their total quantum state is the *tensor product* of the individual vectors—denoted by the operator "$\otimes$". For example, a two-qubit state can, in the computational basis, be described as a superposition of the two-qubit basis states $|0\rangle \otimes |0\rangle \equiv |0\rangle^{\otimes 2} \equiv |00\rangle$, $|01\rangle$, $|10\rangle$ and $|11\rangle$. In general, an N-qubit state can be described by a 2^N-dimensional vector in $\mathbb{C}^{2^N}$. Often, we interpret the "label" of multi-qubit computational basis states as a binary number. This allows us to switch up notation by alternatively using their decimal representation when convenient: *e.g.*, for the two-qubit basis states, $|00\rangle \equiv |0\rangle$, $|01\rangle \equiv |1\rangle$, $|10\rangle \equiv |2\rangle$ and $|11\rangle \equiv |3\rangle$.

2.2.2 Density matrix

In a practical situation, we typically need a format that is more general than *pure state* vectors to describe quantum states. This is realized by introducing *mixed states*, or *density matrices*. Density matrices are also known as *density operators*. The density matrix format

allows us to, *e.g.*, describe statistical ensembles of quantum states. This is useful when the preparation of a quantum state is not fully known or for a thermal equilibrium at finite temperatures. Next to that, we can use density matrices to express the state of a quantum system that is part of a bigger (entangled) system, as we show below.

Formally, an N-qubit density matrix ρ is a *positive semi-definite* matrix of dimensions $2^N \times 2^N$ with its *trace* equal to 1. A positive semi-definite matrix is a matrix with non-negative eigenvalues, and the trace $\mathrm{Tr}(\rho)$ is the sum of the coefficients on the diagonal of ρ. These conditions ensure that the density matrix can always be interpreted as a statistical mixture of pure states $\rho = \sum_s p_s |\psi_s\rangle\langle\psi_s|$—*i.e.*, an ensemble with statistical ambiguity about which pure state $|\psi_s\rangle$ the system is in, fulfilling $p_s \in \mathbb{R}$, $0 < p_s \leq 1$ and $\sum_s p_s = 1$. Every positive semi-definite matrix is automatically *self-adjoint* (*i.e.*, Hermitian)—*i.e.*, the density matrix is equal to its own conjugate transpose: $\rho^\dagger = (\rho^*)^T = \rho$. A pure state $|\psi\rangle$ can be expressed as $\rho = |\psi\rangle\langle\psi|$—*i.e.*, as a density matrix of *rank* one that is described by a single vector. Given that for a pure state $\langle\psi|\psi\rangle = 1$ holds, we see that $\rho^2 = \rho$ holds for a pure state, and, therefore, also $\mathrm{Tr}(\rho^2) = 1$. For an N-qubit mixed state, however, we always have $1/2^N \leq \mathrm{Tr}(\rho^2) < 1$. The exact value for $\mathrm{Tr}(\rho^2)$ can be used as a measure for the purity of a quantum state. The mixed state with the lowest purity is the *maximally mixed state*. For N qubits, this state is created from 2^N orthonormal states $\{|\phi_k\rangle\}_k$ all occurring with the same statistical probability $1/2^N$.

The Bloch sphere representation of Fig. 2.1a can also be used to describe a mixed single-qubit state: whereas a pure state is represented by a point *on* the surface of the sphere, a mixed state can be depicted as a point *inside* the sphere. This can be understood by realizing that the Bloch sphere representation of $\rho = \sum_s p_s |\psi_s\rangle\langle\psi_s|$ can be determined with the weighted average over all vectors $\{|\psi_s\rangle\}_s$ on the sphere, using the coefficients $\{p_s\}_s$ as weights. The maximally mixed state corresponds to the point exactly in the middle of the sphere.

To understand why density operators are useful to express a state that is part of a larger quantum system, we introduce the *partial trace*. This operation "traces out" part of a state we no longer have control or access over and mathematically describes the remainder of the system in which the traced-out part is "ignored". The partial trace is related to the earlier-mentioned trace $\mathrm{Tr}(\rho) \equiv \sum_k \langle\phi_k|\rho|\phi_k\rangle$. Here, $\{|\phi_k\rangle\}_k$ is a basis for the full state ρ, meaning $\mathrm{Tr}(\rho)$ is a sum over the diagonal coefficients of ρ in the $\{|\phi_k\rangle\}_k$ basis. (Note that it can be shown that $\mathrm{Tr}(\rho)$ is independent of the exact basis chosen.) The partial trace $\mathrm{Tr}_i(\rho)$, on the other hand, performs the same operation $\sum_k \langle\phi_k^{(i)}|\rho|\phi_k^{(i)}\rangle$ over a basis $\{|\phi_k^{(i)}\rangle\}_k$ that only describes a *subset* i of the full multi-qubit system. This results in a smaller density matrix in which all information about the states of qubit(s) i is removed. As a concrete example, we consider a pure state $(|0\rangle \otimes |0\rangle + |1\rangle \otimes |1\rangle)/\sqrt{2}$. Tracing out the second qubit requires taking the trace with partial basis vectors $\{\mathbb{I} \otimes |\phi_k\rangle\}_k$, where $\mathbb{I}$ is the 2×2 identity matrix and $\{|\phi_k\rangle\}_k$ is a single-qubit basis. This converts the two-qubit pure state to the single-qubit maximally mixed state $(|0\rangle\langle 0| + |1\rangle\langle 1|)/2$.

2.2.3 Measurement

When a quantum state is measured, the state typically *collapses* to a classical state with respect to the basis on which the measurement is performed. Typically, measuring a quantum state is a probabilistic process that changes the state. For example, measuring

a qubit $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$ in the computational basis has a probability $|\alpha|^2$ of collapsing to $|0\rangle$ and a probability $|\beta|^2$ of collapsing to $|1\rangle$. Instead of measuring individual qubits, we can also *jointly* measure multiple qubits. Measuring a general state $|\psi\rangle$ in a general orthonormal basis $\{|\phi_k\rangle\}_k$ has a probability $p_k = |\langle\phi_k|\psi\rangle|^2$ of collapsing to the basis state $|\phi_k\rangle$. This is the *Born rule* of quantum mechanics. We show examples of measuring two specific single-qubit states in different bases in Figs. 2.1b and 2.1c using the Bloch sphere visualization. These figures indicate how an orthonormal basis can be depicted as an axis through the Bloch sphere. The measurement probabilities for finding the basis states of this basis are indicated by the projection of the state onto this axis.

2

When a measurement causes the state to collapse, the observer—*i.e.*, the entity or system triggering the measurement—has the possibility to learn the *post-measurement state* via a measurement outcome. This outcome is the eigenvalue of the post-measurement basis state with respect to a (*measurement*) *observable* $\mathcal{M}$. The involved basis states are the eigenvectors of the observable $\mathcal{M}$. Therefore, instead of stating that we measure an N-qubit state in a certain basis of $\mathbb{C}^{2^N}$, we can alternatively say that we perform a *projective measurement* with an observable that has these basis states as its eigenstates—*i.e.*, we project the state on one of the basis states. Technically, the observable $\mathcal{M}$ is a linear map that maps vectors in $\mathbb{C}^{2^N}$ onto each other. To guarantee that its eigenstates form a complete eigenbasis, $\mathcal{M}$ needs to be self-adjoint (*i.e.*, equal to its own conjugate transpose).

There exists an even more general representation of a quantum measurement in terms of a set of *measurement operators* $\{\mathcal{M}^{(k)}\}_k$. These operators are also called *Kraus operators* and should satisfy $\sum_k (\mathcal{M}^{(k)})^\dagger \mathcal{M}^{(k)} = \mathbb{1}$, where $\mathbb{1}$ is the identity matrix of the same dimensions as $\mathcal{M}^{(k)}$. In this general picture, the Born rule becomes more general as well: obtaining the measurement outcome m_k has a probability given by $p_k = \mathrm{Tr}(\mathcal{M}^{(k)} \rho (\mathcal{M}^{(k)})^\dagger)$. Here, ρ is the input state before the measurement. Per outcome m_k, the post-measurement state is given by

$$\rho_k = \frac{\mathcal{M}^{(k)} \rho \left(\mathcal{M}^{(k)}\right)^\dagger}{p_k} = \frac{\mathcal{M}^{(k)} \rho \left(\mathcal{M}^{(k)}\right)^\dagger}{\mathrm{Tr}\left(\mathcal{M}^{(k)} \rho \left(\mathcal{M}^{(k)}\right)^\dagger\right)}. \tag{2.2}$$

As becomes clear in Sec. 2.3.1, the measurement transformation of Eq. (2.2) is a more general version of the standard transformation of density matrices in quantum mechanics. Because ρ is assumed to be an ensemble of states $\rho = \sum_s p_s |\psi_s\rangle\langle\psi_s|$, transforming both $|\psi_s\rangle$ and $\langle\psi_s|$ requires placing the operation $\mathcal{M}^{(k)}$ on both sides of ρ. On top of that, for the operations of Sec. 2.3.1, we require $\mathrm{Tr}(\rho) = 1$ to hold both before and after the transformation. That is not the case for transformations in quantum measurements, as they allow us to probabilistically "branch off" into one of the possible operations from $\{\mathcal{M}^{(k)}\}_k$. For each of the operations $\mathcal{M}^{(k)}$, dividing the post-measurement state $\mathcal{M}^{(k)} \rho (\mathcal{M}^{(k)})^\dagger$ with its probability $p_k = \mathrm{Tr}(\mathcal{M}^{(k)} \rho (\mathcal{M}^{(k)})^\dagger)$ normalizes the state—*i.e.*, makes sure the post-measurement state fulfills $\mathrm{Tr}(\rho_k) = 1$.

A projective measurement can be interpreted as a less general version of a quantum measurement with measurement operators $\{\Pi^{(k)}\}_k$ that fulfill $(\Pi^{(k)})^\dagger \Pi^{(k')} = \delta_{k,k'} \Pi^{(k)}$, where $\delta_{k,k'}$ is the Kronecker delta. This can be translated in terms of projecting on the elements of a basis $\{|\phi_k\rangle\}_k$ via $\Pi^{(k)} = |\phi_k\rangle\langle\phi_k|$. It becomes clear that transforming ρ as $\Pi^{(k)} \rho (\Pi^{(k)})^\dagger$ projects the elements of ρ onto the basis state $|\phi_k\rangle$. Such a measurement corresponds to measuring with an observable constructed as $\Pi = \sum_k \lambda_k \Pi^{(k)}$. Here, the λ_k

values take on the role of the eigenvalues of the measurement observable Π. Using the identity $\mathrm{Tr}(\mathcal{O}_1\mathcal{O}_2\mathcal{O}_3) = \mathrm{Tr}(\mathcal{O}_3\mathcal{O}_1\mathcal{O}_2)$, we can understand that the probability of projecting with $\Pi^{(k)}$ now simplifies to $p_k = \mathrm{Tr}(\Pi^{(k)}\rho) = \langle\phi_k|\rho|\phi_k\rangle$. For $\rho = |\psi\rangle\langle\psi|$ a pure state, this further simplifies to the expression $p_k = |\langle\phi_k|\psi\rangle|^2$ above. If the eigenvalues λ_k used in the construction of Π are not unique, obtaining a non-unique eigenvalue during measurement means the state has not fully collapsed. In that case, the post-measurement state is typically in a superposition of the eigenstates that carry this degenerate eigenvalue.

2.2.4 Entanglement

Entanglement is one of the most important resources of quantum information and quantum computing. An entangled system can be defined as a state that can not be described as a tensor product of the individual states of its subsystems. That is, a two-qubit state $|\psi\rangle$ is entangled if there do not exist individual qubit states $|\phi_1\rangle$ and $|\phi_2\rangle$ that fulfill $|\psi\rangle = |\phi_1\rangle \otimes |\phi_2\rangle$. States that *can* be represented as tensor products of the individual substates are called *separable* or *product states*.

Important examples of two-qubit entangled states are the Bell pairs—also known as Bell *states* or *Einstein-Podolsky-Rosen* (EPR) pairs:

$$\begin{aligned} |\Phi^+\rangle &\equiv \frac{|00\rangle + |11\rangle}{\sqrt{2}}, \quad |\Phi^-\rangle \equiv \frac{|00\rangle - |11\rangle}{\sqrt{2}}, \\ |\Psi^+\rangle &\equiv \frac{|01\rangle + |10\rangle}{\sqrt{2}}, \quad |\Psi^-\rangle \equiv \frac{|01\rangle - |10\rangle}{\sqrt{2}}. \end{aligned} \tag{2.3}$$

For more qubits—*i.e*, for $N \geq 3$ qubits—a commonly used entangled state is the Greenberger-Horne-Zeilinger (GHZ) state, defined as $|\mathrm{GHZ}^{(N)}\rangle \equiv (|0\rangle^{\otimes N} + |1\rangle^{\otimes N})/\sqrt{2}$.

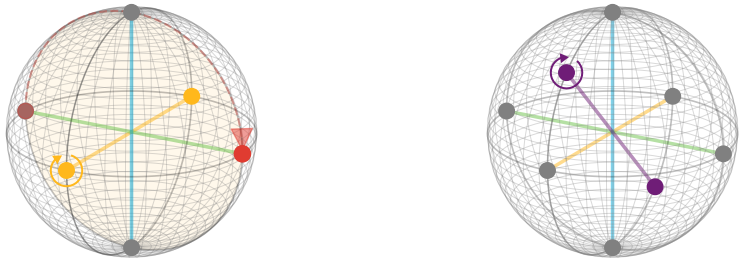
2.3 Quantum computing

In this section, we introduce the basic concepts of quantum computing. We discuss the most general version of an operation (Sec. 2.3.1), elaborate on the most common operations (Secs. 2.3.2, and 2.3.3), introduce the stabilizer formalism (Sec. 2.3.4), and go over the main ideas of measurement-based quantum computing (Sec. 2.3.5). In the field of quantum computing, operations are typically depicted in so-called *circuit notation—i.e.*, in the form of *circuit diagrams*. In these diagrams, qubits are drawn as horizontal lines moving from left to right in time. Operations on qubits are illustrated as "boxes", with the form of the box and the text in it denoting the type of operation. Examples of circuit diagrams can be found in Fig. 2.3, 2.4b, and 2.5.

2.3.1 Operations

A quantum state $|\psi\rangle$ can be transformed into another quantum state $|\phi\rangle$ with a *unitary operation* $\mathcal{U}$ via $|\phi\rangle = \mathcal{U}|\psi\rangle$. Such an operation, which is often called a gate, is a linear map that preserves the inner product between states: $\langle\psi_1|\mathcal{U}^\dagger\mathcal{U}|\psi_2\rangle = \langle\psi_1|\psi_2\rangle$ for all $|\psi_1\rangle \in \mathbb{C}^{2^N}$ and $|\psi_2\rangle \in \mathbb{C}^{2^N}$. This requires $\mathcal{U}^\dagger\mathcal{U} = \mathbb{1}$ to hold. Density matrices evolve as $\rho \mapsto \mathcal{U}\rho\mathcal{U}^\dagger$ under a unitary transformation $\mathcal{U}$.

As discussed in Sec. 2.2.3, a quantum gate can be regarded as a simpler version of a quantum measurement. Often, measurements are also classified as "operations". The most



(a) (b)

Figure 2.2: (a) Intuitive interpretation of the $R_X(\theta)$, $R_Y(\gamma)$, or $R_Z(\varphi)$ operators in Eq. (2.5) as a rotation over an angle θ, γ, or φ around the yellow x-axis, the green y-axis, or the blue z-axis in the Bloch sphere, respectively. We show explicitly how the state $|-i\rangle = (|0\rangle - i|1\rangle)/\sqrt{2}$ (dark red dot) is transformed to $|+i\rangle = (|0\rangle + i|1\rangle)/\sqrt{2}$ (bright red dot) under a $X = iR_X(\pi)$ transformation (where the phase factor i in $iR_X(\pi)$ corresponds to an irrelevant global phase factor)—*i.e.*, by rotating the state with an angle π around the yellow x-axis. (b) The Hadamard gate corresponds to a π rotation around the purple axis in the Bloch sphere.

general version of a quantum operation is a *completely positive trace-preserving* (CPTP) map—also known as a *channel*. As the name suggests, a channel should preserve $\mathrm{Tr}(\rho) = 1$ when working on a quantum state. On top of that, such a map should make sure that the quantum state stays positive semi-definite. Each channel $\mathcal{N} : \rho \mapsto \sum_k \mathcal{K}^{(k)} \rho (\mathcal{K}^{(k)})^\dagger$ can be described by a set of Kraus operators $\{\mathcal{K}^{(k)}\}_k$ fulfilling $\sum_k (\mathcal{K}^{(k)})^\dagger \mathcal{K}^{(k)} = \mathbb{1}$. A channel is an operator working on another operator—this can be either a density matrix or another channel. For that reason, a channel is sometimes called a *superoperator*.

If we apply a unitary transformation $\mathcal{U}$ right in front of a measurement with measurement operators $\{\mathcal{M}^{(k)}\}_k$, this operation can effectively be incorporated in the measurement, as long as we are not interested in the basis in which the post-measurement state is defined. This can be understood by realizing that the measurement statistics, *i.e.*, the probabilities of finding each of the measurement outcomes m_k, are the same for measuring with $\{\mathcal{M}^{(k)}\}_k$ on a state $\mathcal{U}\rho\mathcal{U}^\dagger$ as for measuring with $\{\mathcal{U}^\dagger \mathcal{M}^{(k)} \mathcal{U}\}_k$ on a state ρ. This is because the trace in the denominator of Eq. (2.2) is exactly the same in both scenarios. Therefore, the only difference between the two scenarios is that the post-measurement states will undergo a basis transformation $\mathcal{U}^\dagger \rho_k \mathcal{U}$ in the second scenario. In other words, in terms of measurement statistics, applying $\mathcal{U}$ and measuring with $\{\mathcal{M}^{(k)}\}_k$ is equivalent to measuring $\{\mathcal{U}^\dagger \mathcal{M}^{(k)} \mathcal{U}\}_k$. For a projective measurement with measurement operator Π, this simplifies to transforming the basis vectors $\{|\phi_k\rangle\}_k$ of Π as $\{\mathcal{U}^\dagger |\phi_k\rangle\}_k$.

2.3.2 Pauli matrices

The Pauli matrices (or *Pauli operators* or *Pauli gates*) play a pivotal role in all of quantum mechanics and are the most important operations in quantum computing. As unitary and Hermitian 2×2 matrices with eigenvalues $\{+1, -1\}$ they represent both measurement observables and logic gates. In the computational basis, they are defined as follows:

$$X \equiv \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}, \quad Y \equiv \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix}, \quad Z \equiv \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}. \tag{2.4}$$

Any two Pauli matrices anti-commute, meaning that $PP' = -P'P$ holds, as long as $P \neq P'$. They fulfill $X^2 = Y^2 = Z^2 = \mathbb{I}$, where $\mathbb{I}$ is the identity matrix of size 2×2. Together with $\mathbb{I}$, the Pauli matrices form the basis for any 2×2 matrix $M = \lambda_1 X + \lambda_2 Y + \lambda_3 Z + \lambda_4 \mathbb{I}$. If we allow the Pauli operators to have prefactors $\{\pm 1, \pm i\}$, they form a group $\mathcal{I}^{(1)} = \langle i\mathbb{I}, X, Z\rangle$: the one-qubit *Pauli group*. This can be understood by realizing that $Y = iXZ$ holds.

Similar properties hold for multi-qubit Pauli operators on N qubits. These are tensor products of N Pauli operators with eigenvalues $\{+1, -1\}$. These operators form the basis for any $2^N \times 2^N$ matrix. They also form a group if we allow prefactors $\{\pm 1, \pm i\}$: the N-qubit Pauli group $\mathcal{I}^{(N)}$. For an N-qubit Pauli operator $P = P_1 \otimes P_2 \otimes \cdots \otimes P_N$, we define its *weight* as the number of non-identity matrices in the tensor product of P. In the full description of the operator P, the index in the subscript typically describes the qubit number the operator works. Often, we notate operators like this without explicitly mentioning the tensor product and identity matrices—*e.g.*, P_2P_4 instead of $\mathbb{I}_1 \otimes P_2 \otimes \mathbb{I}_3 \otimes P_4$.

As a logic gate, the Pauli-X operator flips the state of a qubit in the computational basis—*i.e.*, it introduces a *bit-flip* on a qubit. The computational basis states are eigenstates of the Pauli-Z operator. This means that the Pauli-Z operator leaves the $|0\rangle$ invariant, but adds a minus sign to the $|1\rangle$ state—*i.e.*, the Pauli-Z operator accomplishes a *phase-flip* on a qubit. More generally, we can use the Pauli matrices to define arbitrary *rotations*:

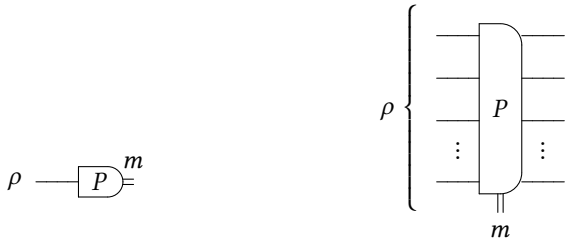
$$\begin{aligned} R_X(\theta) \equiv e^{-i\theta X/2} &= \cos\left(\frac{\theta}{2}\right)\mathbb{I} - i\sin\left(\frac{\theta}{2}\right)X = \begin{bmatrix} \cos(\theta/2) & -i\sin(\theta/2) \\ -i\sin(\theta/2) & \cos(\theta/2) \end{bmatrix}, \\ R_Y(\gamma) \equiv e^{-i\gamma Y/2} &= \cos\left(\frac{\gamma}{2}\right)\mathbb{I} - i\sin\left(\frac{\gamma}{2}\right)Y = \begin{bmatrix} \cos(\gamma/2) & -\sin(\gamma/2) \\ \sin(\gamma/2) & \cos(\gamma/2) \end{bmatrix}, \\ R_Z(\varphi) \equiv e^{-i\varphi Z/2} &= \cos\left(\frac{\varphi}{2}\right)\mathbb{I} - i\sin\left(\frac{\varphi}{2}\right)Z = \begin{bmatrix} e^{-i\varphi/2} & 0 \\ 0 & e^{i\varphi/2} \end{bmatrix}. \end{aligned} \tag{2.5}$$

As indicated in Fig. 2.2a, we can intuitively interpret these operations as rotations around the x-axis, y-axis, and z-axis of the Bloch sphere, respectively. The Pauli matrices themselves are defined as rotations over an angle π (up to a global phase factor): $X = iR_X(\pi), Y = iR_Y(\pi)$ and $Z = iR_Z(\pi)$. The rotations of Eq. (2.5) can also be used to parameterize an arbitrary unitary transformation—*e.g.*, among other possibilities, with $\mathcal{U} = e^{i\alpha} R_Z(\varphi) R_Y(\gamma) R_Z(\vartheta)$ or $\mathcal{U} = e^{i\alpha} R_Z(\varphi) R_X(\theta) R_Z(\vartheta)$. Here, α is again a global phase.

As mentioned earlier, Pauli matrices can also be used as measurement observables. In Fig. 2.3a, we depict the typical circuit diagram notation of a Pauli measurement on a single qubit. We note that measuring the Pauli-Z observable is equivalent to a measurement in the computational basis. The post-measurement states for measuring an N-qubit Pauli operator P can be calculated with the projectors

$$\Pi_\pm \equiv \frac{1}{2}\left(\mathbb{I}^{\otimes N} \pm P\right). \tag{2.6}$$

Here, Π_+ projects on the $+1$ eigenspace and Π_- projects on the -1 eigenspace of P. For a single-qubit Pauli operator P, the projectors Π_+ and Π_- correspond to $|\phi_k\rangle\langle\phi_k|$ for the eigenstates $\{|\phi_k\rangle\}_{k=1}^2$ of P with eigenvalues $+1$ and -1, respectively. As discussed in Sec. 2.2.3, a measurement observable with degenerate eigenvalues does typically not collapse the full state that is being measured. This is relevant for performing a *joint* measurement with a



(a) (b)

Figure 2.3: (a) Typical circuit diagram showing the measurement of a single-qubit Pauli observable P on a general quantum state ρ. Here, $m \in \{+1, -1\}$ depicts the possible measurement outcomes. The two lines coming out of the measurement operator indicate that the outcome m is a classical bit. Therefore, the measurement outcome is sometimes written in binary notation: $m \in \{0, 1\}$. (b) Joint Pauli measurement of a multi-qubit Pauli observable P on a multi-qubit quantum state ρ. The outcome of the measurement is again a classical bit m. However, this operation is typically a parity measurement and does not fully collapse the measured state. Therefore, qubits remain present as vertical lines even after the measurement.

multi-qubit Pauli operator, as such an operator has degenerate eigenvalues $+1$ and -1. For example, a joint operator with multiple Pauli-Z matrices only measures the *parity* of a state in the computational basis. We show the circuit diagram depiction of a joint measurement in Fig. 2.3b. In practice, carrying out a joint (*i.e.*, multi-qubit) Pauli measurement is difficult. Fortunately, there is an indirect way possible to achieve this. Fig. 2.4a shows how a joint Pauli operator $P = P_1 \otimes P_2 \otimes \cdots \otimes P_N$ can be measured with the aid of an ancillary qubit initialized in the $|+\rangle$ state. Fig. 2.4b shows how the same measurement can be achieved *non-locally* with the aid of an entangled state if the qubits are part of different network parties. These methods require *controlled*-Pauli operators, which are introduced in Sec. 2.3.3.

2.3.3 Other common quantum gates

In Sec. 2.3.2 and Fig. 2.4 it becomes clear that measuring a (multi-qubit) Pauli operator is convenient with controlled-Pauli operations. The controlled-not (CNOT, CX) operator is a two-qubit operator that only applies a Pauli-X on a target qubit if the control qubit is in the $|1\rangle$ state. The controlled-phase (CPHASE, CZ) operator works the same, but with a Pauli-Z operator. Fig. 2.5 shows how these gates are depicted in circuit notation, as well as what their matrix representations look like in the computational basis. As with Pauli operators, we sometimes use the notation $\mathrm{C}X_{i,j}$ and $\mathrm{C}Z_{i,j}$ to denote the qubits on which these two-qubit operators work. Here, by definition, the first qubit is the control qubit and the second qubit is the target qubit. The CZ and CX operators can be generalized to a controlled-Pauli operator CP and a controlled-unitary operator C$\mathcal{U}$ that, respectively, applies a general Pauli gate P or a general unitary transformation $\mathcal{U}$ if the control qubit is in the $|1\rangle$ state. A concrete example is the CY gate depicted in Fig. 2.6a.

In Fig. 2.5, we see how a combination of three CX gates in series can be used to interchange the states of two qubits. This combination of gates leads to the SWAP gate. We use this gate in Fig. 2.8a to construct the main building block of measurement-based quantum computing. Furthermore, the *Hadamard* gate H in Fig. 2.5 accomplishes $HZH = X$ and $HXH = Z$. This gate also maps the computational basis states as $H|0\rangle = |+\rangle$ and

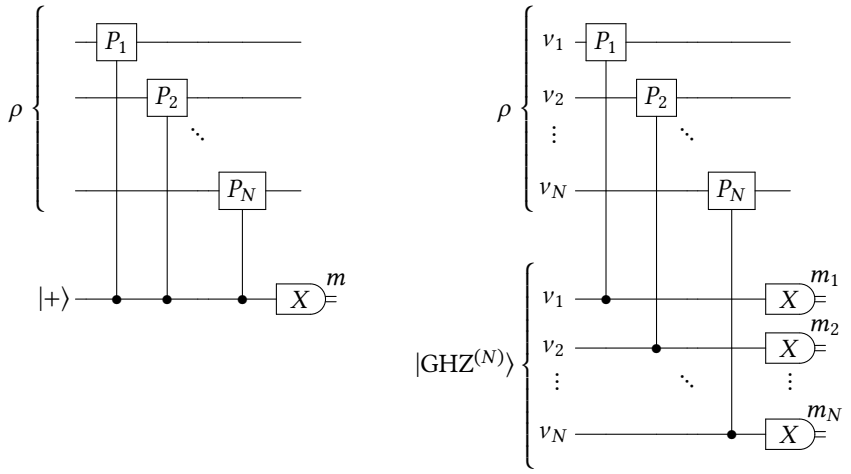


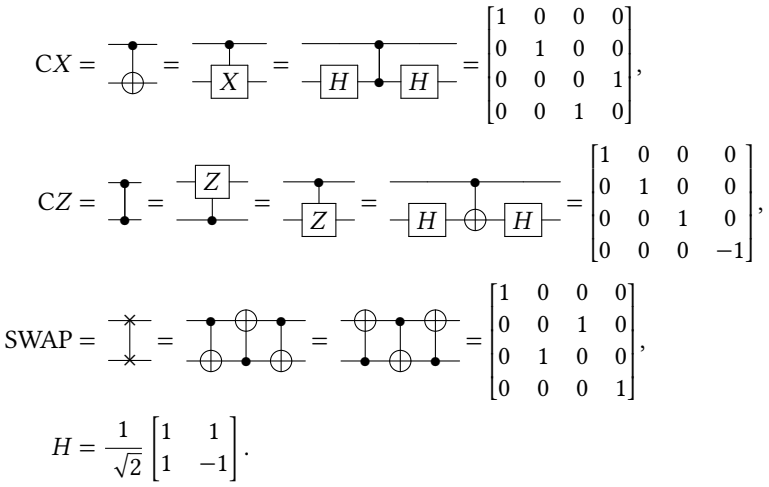
Figure 2.4: (a) Circuit diagram for measuring the joint N-qubit Pauli operator $P = P_1 \otimes P_2 \otimes \cdots \otimes P_N$ on a state ρ with the aid of an ancillary qubit initialized in the state $|+\rangle = (|0\rangle + |1\rangle)/\sqrt{2}$. The gates in this diagram are two-qubit controlled-Pauli gates—see Sec. 2.3.3 and Fig. 2.5 for more details. As introduced in Fig. 2.3a, the box at the right denotes a Pauli-X measurement with outcome $m \in \{+1, -1\}$. (b) Circuit diagram for the same measurement with the aid of an ancillary GHZ state $|\mathrm{GHZ}^{(N)}\rangle = (|0\rangle^{\otimes N} + |1\rangle^{\otimes N})/\sqrt{2}$. With this setup, it is possible to measure the Pauli operator P on qubits belonging to N parties $\nu_1, \nu_2, \ldots, \nu_N$. We call such a measurement a *non-local* measurement. Besides the entangled state, this operation requires local operations and classical communication. The parties find the full measurement outcome $m = \prod_{i=1}^{N} m_i$ by combining their individual measurement results $(m_1, m_2, \ldots, m_N) \in \{+1, -1\}^N$. For $N = 2$ parties, the ancillary state simplifies to a Bell state $|\Phi^+\rangle$.

$H|1\rangle = |-\rangle$, where the states $|+\rangle$ and $|-\rangle$ are the eigenstates of the Pauli-X matrix. Since it allows us to switch between the computational basis (*i.e.*, the Pauli-Z basis) and the equally useful Pauli-X basis, the Hadamard gate is an important gate in quantum computing. We depict the Hadamard gate as a rotation on the Bloch sphere in Fig. 2.2b.

We define a *universal gate set* as a set of quantum gates able to generate any $2^N \times 2^N$ unitary operation on an N-qubit system. A quantum computer in possession of such a gate set can perform any computational task possible with quantum computing. Several combinations of gates form a universal gate set. As we saw above, any single-qubit unitary can, up to a global phase, be generated from the Pauli-X and Pauli-Z rotations of Eq. (2.5). However, by themselves, these rotations are not universal, as they can not entangle states—*i.e.*, qubits that begin as separable states remain separable after these rotations. This gives some intuition as to why we need to add either the CZ or the CX gate to obtain a universal gate set. If we want to generate every $2^N \times 2^N$ unitary exactly up to the global phase, we also need to add other single-qubit gates to get a universal gate set.

One often-used example of a universal gate set is the set consisting of the Hadamard gate, the CX gate, and the so-called T gate. The T gate can be considered, up to a global phase, as a Pauli-Z rotation over an angle $\pi/4$:

$$T \equiv e^{i\pi/8} R_Z(\pi/4) = \begin{bmatrix} 1 & 0 \\ 0 & e^{i\pi/4} \end{bmatrix}, \qquad S \equiv T^2 = \begin{bmatrix} 1 & 0 \\ 0 & i \end{bmatrix}. \tag{2.7}$$



$$\mathrm{C}X = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}, \quad \mathrm{C}Z = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \end{bmatrix},$$

$$\mathrm{SWAP} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad H = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}.$$

Figure 2.5: Controlled-not (CX), controlled-phase (CZ), SWAP, and Hadamard (H) gates, including their matrix representation in the computational basis.

If we apply the T gate two times in succession, we get another often used quantum gate: the *phase gate* or S gate. This gate is effectively a Pauli-Z rotation over an angle $\pi/2$. The significance of the T gate becomes clear when we realize it is the "missing operation" for efficiently applying universal operations in the main error-correction schemes discussed in Ch. 3 and the rest of this thesis. The S gate transforms a Pauli-X matrix into a Pauli-Y matrix (*i.e.*, $Y = SXS^\dagger$) and transforms the eigenstates of Pauli-X into the Pauli-Y eigenstates $|+i\rangle \equiv (|0\rangle + i|1\rangle)/\sqrt{2}$ and $|-i\rangle \equiv (|0\rangle - i|1\rangle)/\sqrt{2}$. All gates that can be generated with combinations of H, CX, and S gates form the so-called *Clifford group*. This group maps the N-qubit Pauli group $\boldsymbol{\mathcal{I}}^{(N)}$ onto itself—*i.e.*, unitary operators of the N-qubit Clifford group either leave each element of $\boldsymbol{\mathcal{I}}^{(N)}$ invariant or *permute* the elements of $\boldsymbol{\mathcal{I}}^{(N)}$.

As mentioned in Sec. 2.3.2, two different single-qubit Pauli operators anti-commute. This has consequences for the commutation relations between quantum computing gates. We summarize a selection of (anti-)commutation and decomposition relations in Fig. 2.6.

2.3.4 Stabilizer operator

In this thesis, we make extensive use of the stabilizer formalism. This set of techniques allows one to efficiently describe and study error-correction codes in terms of Pauli operators, as discussed in more detail in Ch. 3. As becomes clear below and in the next chapter, the stabilizer formalism is a powerful tool to compactly describe multi-qubit states, as well as vector spaces of states. This is because, with the stabilizer formalism, it is possible to describe quantum states by a set of Pauli operators that "stabilize" the states—meaning we no longer need to describe their full state vector or density matrix.

A stabilizer operator (or *stabilizer*) of a quantum state $|\phi\rangle$ is an operator $\mathcal{O}$ that verifies $\mathcal{O}|\phi\rangle = |\phi\rangle$—*i.e.,* $|\phi\rangle$ is an eigenvector of $\mathcal{O}$ with eigenvalue $+1$ and, in consequence, leaves

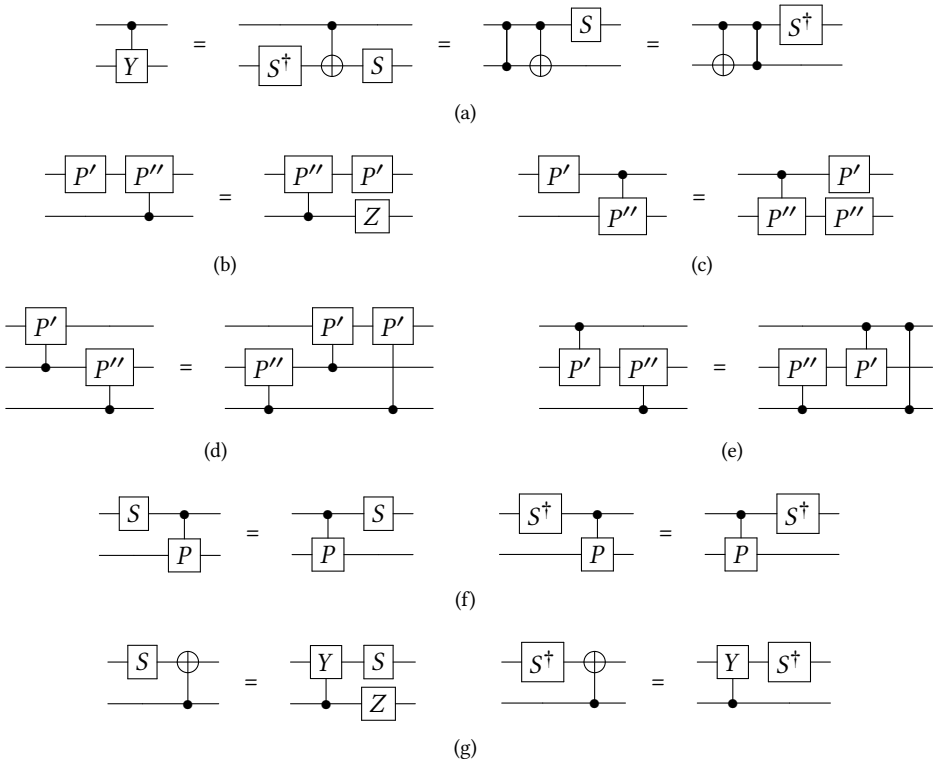


Figure 2.6: (a) Decomposition of a CY gate into a CZ gate, a CX gate and an S gate (or an $S^\dagger$ gate depending on the order of the CZ and CX gates). (b) Commutation of a gate $P' \in \{X,Y,Z\}$ through the target of a gate CP'' for $P'' \in \{X,Y,Z\}$ and $P' \neq P''$. This commutation relation also holds for P' and P'' being anti-commuting multi-qubit Pauli operators. (c) Commutation of a gate $P' \in \{X,Y\}$ through the control of a gate CP'' for $P'' \in \{X,Y,Z\}$. This commutation relation also holds for P' being a multi-qubit Pauli operator with Pauli-X or Pauli-Y on the qubit that holds the control of the CP'' gate and/or P'' being a multi-qubit Pauli operator. (d) Commutation of a CP' gate and a CP'' gate that overlap with one control and one target, for $P' \in \{X,Y,Z\}$ and $P'' \in \{X,Y\}$. This commutation relation also holds for P' being a multi-qubit Pauli operator and/or P'' being a multi-qubit Pauli operator with Pauli-X or Pauli-Y on the qubit that holds the control of the CP' gate. (e) Commutation of a CP' gate and a CP'' gate that overlap on their target qubit, for $P', P'' \in \{X,Y,Z\}$ and $P' \neq P''$. This commutation relation also holds for P' and P'' being anti-commuting multi-qubit Pauli operators. (f) Commutation of an S and an $S^\dagger$ gate through the control of a CP gate, for $P \in \{X,Y,Z\}$. (g) Commutation of an S and an $S^\dagger$ gate through the target of a CX gate.

$|\phi\rangle$ invariant. An N-qubit pure quantum state has 2^N stabilizer operators. These 2^N operators form the *stabilizer group* of the state, generated by a subset of N independent operators. All operators in this group stabilize the state. For example, the N-qubit GHZ state $|\text{GHZ}^{(N)}\rangle = (|0\rangle^{\otimes N} + |1\rangle^{\otimes N})/\sqrt{2}$ has a stabilizer group generated by the operators

$$\{X_1X_2 \ldots X_N, Z_1Z_2, Z_2Z_3, \ldots, Z_{N-1}Z_N\}. \tag{2.8}$$

A stabilizer group always includes the identity $\mathbb{1}$ on all involved qubits but never includes $-\mathbb{1}$. We call the $2^N - 1$ operators in this group that are not the identity $\mathbb{1}$ the *non-trivial* stabilizers of the state. Typically, we only consider stabilizer operators that are Pauli operators: we restrict to stabilizer states stabilized by Pauli operators. Each N-qubit

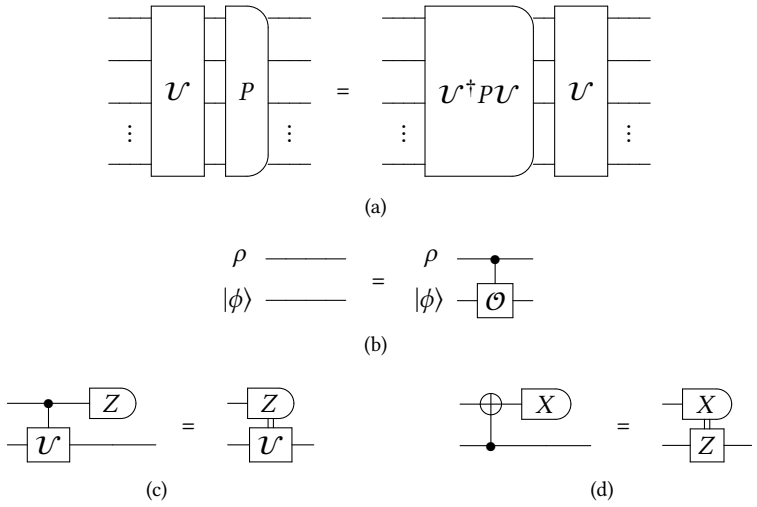


Figure 2.7: Four situations where the left circuit is equivalent to the circuit on the right. (a) A quantum gate $\mathcal{U}$ *before* a single-qubit or multi-qubit measurement of Pauli operator P can be converted to the same gate $\mathcal{U}$ applied *after* measuring the operator $\mathcal{U}^\dagger P \mathcal{U}$—in other words, the measurement operation has *commuted* with the gate. This relation is also discussed at the end of Sec. 2.3.1. (b) The controlled-$\mathcal{O}$ gate (*i.e.*, the C$\mathcal{O}$ gate) between a qubit ρ and a pure state $|\phi\rangle$ leaves both states invariant when $|\phi\rangle$ is stabilized by $\mathcal{O}$—*i.e.*, for $\mathcal{O}|\phi\rangle = |\phi\rangle$. (c) A C$\mathcal{U}_{i,j}$ gate (*i.e.*, a controlled-$\mathcal{U}$ gate) before a Pauli-Z measurement on qubit i can be converted to conditionally applying $\mathcal{U}$ on qubit j based on the measurement result—*i.e.*, $\mathcal{U}$ is applied when the measurement result is -1 but $\mathcal{U}$ is not applied when the measurement result is $+1$. (d) A C$X_{i,j}$ gate before a Pauli-X measurement on qubit j can be converted to conditionally applying Z on qubit i based on the measurement result.

stabilizer group then becomes an Abelian (*i.e.*, commutative) subgroup of the full Pauli group $\mathcal{I}^{(N)}$—*i.e.*, a group in which all operators commute with each other. The Clifford group, introduced in Sec. 2.3.3, is exactly the group of operators that map Pauli stabilizer states to other Pauli stabilizer states.

If, instead of 2^N stabilizer operators (*i.e.*, N generators), we only use 2^M stabilizers (*i.e.*, $M < N$ generators) to describe an N-qubit state, we do not describe a single state, but a subspace of $K = N - M$ states. In Ch. 3, this principle is used to define error-correction codes. In an error-correction code, the states in the subspace of the stabilizer operators become so-called logical states.

We sometimes write $\mathcal{J} = \langle g_i \rangle_{i=1}^M \equiv \langle g_1, g_2, \ldots, g_M \rangle$ to denote that a stabilizer group $\mathcal{J}$ of a state or vector space is generated by the operators $\{g_1, g_2, ..., g_M\}$. With the stabilizer formalism, it is straightforward to update the stabilizer group $\mathcal{J} = \langle g_i \rangle_{i=1}^M$ after a unitary operation or Pauli measurement on the involved qubits. A unitary operation $\mathcal{U}$ simply transforms the stabilizer group as $\mathcal{J} \mapsto \langle \mathcal{U}\, g_i\, \mathcal{U}^\dagger \rangle_{i=1}^M$. Furthermore, measuring a Pauli operator that commutes with all stabilizer operators leads to a deterministic $m = +1$ measurement outcome and does not alter the stabilizer group. On the other hand, measuring a Pauli operator P that *anti*-commutes with at least one of the stabilizer generators $g \in \{g_i\}_{i=1}^M$ has 50% probability of giving $m = +1$ and $m = -1$ measurement outcomes. In the list of post-measurement generator operators, mP replaces g, whereas all *additional* generators that anti-commute with P need to be multiplied by g.

2.3.5 Measurement-based quantum computation

A specific class of stabilizer states is the class of *graph states* [3]. These states have important applications in quantum error correction and quantum networks. A graph state is a multi-qubit stabilizer state for which one can use mathematical graphs (*i.e.*, objects with vertices and edges) to describe its stabilizer group. Specifically, one can use the graph representation to define its stabilizer generators. In this representation, qubits in the $|+\rangle$ state are placed on the vertices of the graph—*i.e.*, qubits individually stabilized by the Pauli-X operator. The edges of the graph describe inter-qubit connections realized with CZ gates between the qubits. As a result of these gates, the Pauli-X stabilizer operator on each qubit acquires additional Pauli-Z operators on its neighbors after "commuting through" the CZ gates—see Fig. 2.6b for more information. More formally, a certain graph $G = (V, E)$, with V its set of vertices and E its set of edges, corresponds to a stabilizer state $|G\rangle$ that is generated by the operators $X_i \bigotimes_{j \in \mathbb{N}_i} Z_j$, for all $i \in V$. Here, $\mathbb{N}_i = \{ j \mid (i, j) \in E \}$ is the *neighborhood* of vertex $i \in V$: the set of vertices that are connected with an edge to vertex i. Since graph states can be created by initializing all qubits in the $|+\rangle$ state and applying CZ gates between the qubits that are connected via an edge, $|G\rangle$ is defined as $|G\rangle = \prod_{(i,j)\in E} CZ_{i,j} |+\rangle^{\otimes |V|}$.

When a graph state is based on a graph described on a regular lattice, the corresponding stabilizer states are often referred to as cluster states. These cluster states typically form the resource states for measurement-based quantum computation (MBQC). This paradigm can be seen as a counterpart of the circuit-based computation that we see in, *e.g.*, Figs. 2.3, 2.4, 2.5, 2.6, and 2.7, and is used for protocols like, *e.g.*, *blind quantum computing* [4, 5]. This protocol allows a client to perform calculations on a remote quantum computer without revealing any calculation details.

In MBQC, we prepare a graph or cluster state and perform calculations on it by measuring out the qubits of this state. Typically, intermediate corrections based on previous measurement outcomes are applied to the measurement basis (bases) of the next measurement(s). This is necessary because, in MBQC, a general unitary transformation can only be achieved if we *teleport* the state to another qubit during the operation. For example, with the *one-bit teleportation* scheme of Fig. 2.8a it is possible to deterministically teleport a state $|\psi\rangle$ to another qubit, as long as we apply a Pauli-X correction in case we get a measurement outcome $m = -1$. The one-bit teleportation circuit can derived from the SWAP gate, using the equivalent quantum circuits depicted in Figs. 2.7b and 2.7c. Usually, teleportation is considered in a context where the two-qubit gate in Fig. 2.8a can not be carried out directly, *e.g.*, because the two involved qubits are not part of the same quantum system [6–8]. In that situation, we can apply the circuit of Fig. 2.8a two times in succession, where one of the two-qubit gates is replaced by a Bell pair. However, in the context of MBQC, we take a different approach and focus on situations where the two-qubit gate *can* be applied directly.

In the context of graph and cluster states, it is more convenient to work with CZ gates instead of CX gates. For the one-bit teleportation scheme, this leads to the circuit of Fig. 2.8b. This scheme works in the same way, apart from the fact that we now teleport $H|\psi\rangle$ instead of just the state itself. However, in MBQC, this is not a problem. That is because we are primarily interested in situations where we teleport a state *and* simultaneously apply a unitary $\mathcal{U}$ to the state. For a specific unitary $\mathcal{U}$, we can now simply calculate and apply $\mathcal{U}' = H\mathcal{U}$, and this effectively leads to the application of $\mathcal{U}$, as we see in Fig. 2.8c.

Even more generally, we are interested in a situation where we create a graph state

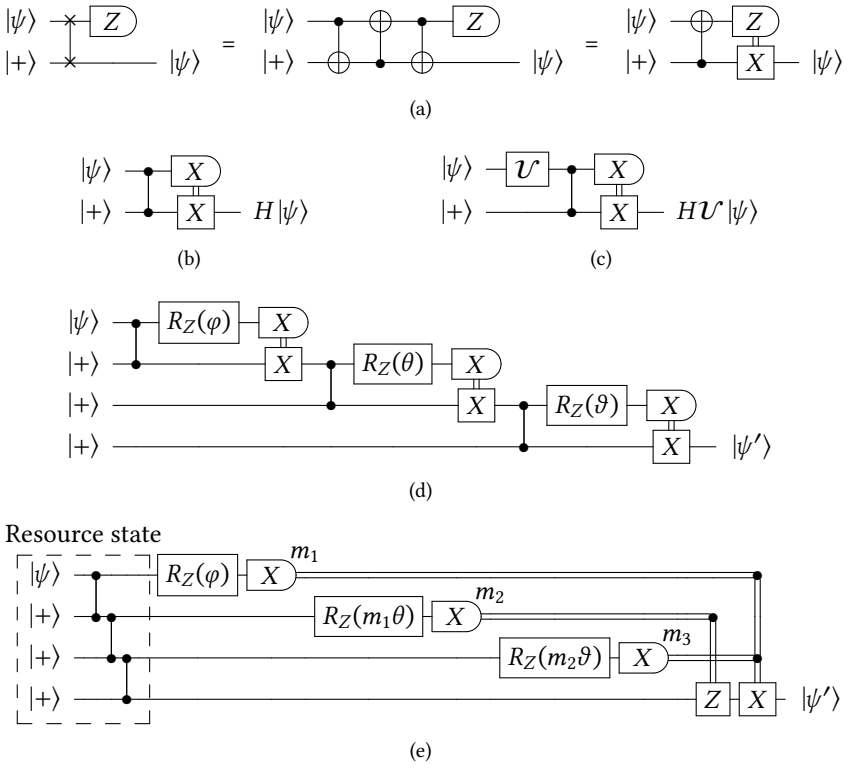


Figure 2.8: Building blocks for measurement-based quantum computation (MBQC). The measurement outcomes obey $m_1, m_2, m_3 \in \{+1, -1\}$. Double lines indicate conditional gates: if the (product of the involved) measurement outcome(s) is $+1$, this gate is not applied, but for -1, it is applied. (a) One-bit teleportation with a CX gate. We use the relation in Fig. 2.5 to express the SWAP gate as three CX gates. Using the identity in Fig. 2.7b, the first CX gate can be removed because $X|+\rangle = |+\rangle$; using the identity in Fig. 2.7c, the last CX gate can be incorporated in the Pauli-Z measurement on the first qubit. (b) One-bit teleportation with a CZ gate. (c) One-bit teleportation with an operation on the qubit before teleportation. (d) Teleportation scheme for applying a general unitary $\mathcal{U} = HR_Z(\vartheta)HR_Z(\theta)HR_Z(\varphi)$ on a state $|\psi\rangle$. The final state is $|\psi'\rangle = \mathcal{U}|\psi\rangle$. Because the Pauli-$Z$ rotations in the decomposition of the unitary commute with the CZ gates of the scheme, they can alternatively be applied right before the measurements—moreover, they can now be regarded as changing the measurement operator and we can incorporate them in the measurement operation. (e) The same scheme as (d), but now with the corrections moved to the end of the circuit, using (anti-)commutation relations of Fig. 2.6. This makes it possible to reframe the circuit in the context of MBQC. (Anti-)commutation relations change what operators have to be applied and how the corrections have to be incorporated (based on the measurement outcomes).

and apply $\mathcal{U}$ by measuring out the state. To achieve this, we decompose our unitary as $\mathcal{U} = HR_Z(\vartheta)HR_Z(\theta)HR_Z(\varphi)$, where we ignore the global phase of the operator and split up the teleportation scheme into three steps [9]. Because each of the three R_Z rotations commutes with the CZ gate, they can now be placed after the CZ gate. As explained in Sec. 2.3.1, these rotations effectively alter the measurement observable from X to $R_Z(-\theta)XR_Z(\theta)$, and similarly for the other two measurements. The associated circuit is shown in Fig. 2.8d. Finally, the conditional Pauli-X corrections can be moved to the end of the circuit, using the commutation rules $X_i \mathrm{C}Z_{i,j} = \mathrm{C}Z_{i,j} X_i Z_j$ (see Fig. 2.6b or Fig. 2.6c)

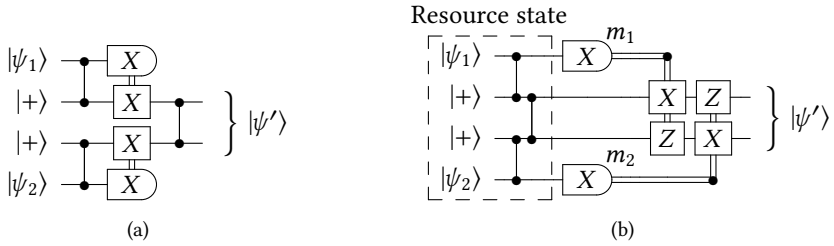


Figure 2.9: (b) Circuit that carries out a CZ gate on two qubits of a MBQC state. This circuit is based on circuit (a), that applies a CZ gate on two qubits that are teleported with the one-bit teleportation scheme of Fig. 2.8b. The output state of both circuits is $|\psi'\rangle = CZ_{1,2}\, H|\psi_1\rangle \otimes H|\psi_2\rangle$.

and $X\, R_Z(\theta) = R_Z(-\theta)\, X$ [10]. This leads to the circuit in Fig. 2.8e. We see that the chain of CZ gates on the initial states has the form of a graph state. Measuring the first three qubits in the measurement bases indicated by the rotations that achieve $\mathcal{U}$ leads to the state $\mathcal{U}|\psi\rangle$ on the final qubit of the chain, up to Pauli-Z and Pauli-X corrections based on the measurement outcomes.

Since a CZ gate fits directly in the format of the graph state, the same techniques can be used to perform a CZ gate between two teleported qubits. In Fig. 2.9, we see how we can perform a CZ gate by measuring out two qubits of a graph state. Together with the operations of Fig. 2.8, this shows that, if we have the right resource state, we can perform any quantum operation by carrying out adaptive measurements.

2.4 Noise models

Unfortunately, no quantum computer or quantum operation is perfect—this is exactly what makes the concept of the density matrix so useful. In Ch. 3, we look extensively at options to deal with unwanted errors appearing during our calculations. Here, we first introduce the building blocks for describing and modeling these errors.

We start in Sec. 2.4.1 by going over a set of general noise channels used to describe noise in quantum computing. The (inevitable) existence of noise in quantum systems is typically modeled as classical uncertainty on quantum states and channels—*i.e.*, using the density matrix formalism introduced in Sec. 2.2.2. We use this approach to describe and classify noisy quantum states in Sec. 2.4.2 and noisy quantum operations in Sec. 2.4.3.

Describing noise on quantum states and operations is another area where the Pauli operators come in. This is because it is often infeasible to track all possible transformations that can occur in a quantum state and capture them in a statistical ensemble. Instead, a typical approach is to approximate noise using only Pauli transformations. In Sec. 2.4.4, we discuss how a general state or channel can be converted to a representation based on Pauli operators. This is useful because describing noise with just Pauli errors is often easier than dealing with the full noise model, while still typically providing a good approximation. That is because, if the errors occur as arbitrary rotations of the form of Eq. (2.5) and are described by a homogeneous probability distribution, the average noise channel can be described as a channel with Pauli errors [11, 12].

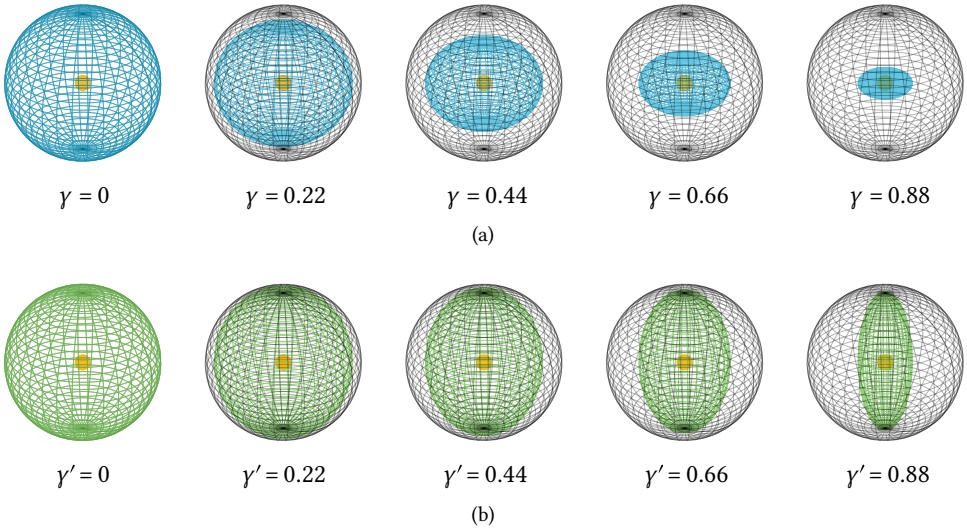


Figure 2.10: (a) Transformation of pure states $\rho = |\psi\rangle\langle\psi|$ on the Bloch sphere with the general amplitude damping channel $\mathcal{N}_{\text{GAD}}(\rho,\gamma)$ for several values of γ. The original Bloch sphere is printed with grey lines and the states after the transformation are printed in blue. The yellow point in the middle of the Bloch sphere corresponds to the maximally mixed state. We see that for $\gamma = 0$ the channel does not transform the states, whereas for $\gamma \to 1$ the channel sends all states to the maximally mixed state. (b) Transformation of pure states $\rho = |\psi\rangle\langle\psi|$ on the Bloch sphere with the phase damping channel $\mathcal{N}_{\text{PD}}(\rho,\gamma')$ for several values of γ'. The original Bloch sphere is printed with grey lines and the states after the transformation are printed in green. We see that this channel does not transform the basis states $|0\rangle$ and $|1\rangle$ themselves: it only removes phase information for a superposition of these states.

2.4.1 Noise channels

Typically, errors associated with noisy operations or states are defined as channels, using the framework introduced in Sec. 2.3.1 to define their Kraus operators. Alternatively, for some channels, it is more intuitive to describe directly how they transform the density matrix. Channels described with Pauli matrices as Kraus operators are known as *Pauli channels*. Examples of Pauli channels are the *depolarizing channels* and the *dephasing channel* (also known as the *phase damping channel*) defined below. We refer to a general Pauli channel in this thesis with the symbol $\mathcal{N}_P$. In Sec. 2.3.2, we discuss how, for qubits defined in the computational basis, Pauli-X operators introduce bit-flips and Pauli-Z operators introduce phase-flips. Since $Y = iXZ$ holds, this means that Pauli-Y operators introduce both bit-flips and phase-flips. As indicated above, most error models use the Pauli operators to introduce these flips as stochastic errors. They introduce these errors either at random or after an operation, as discussed in Sec. 2.4.3. For example, the so-called *phenomenological error model* introduces independent and identically distributed (*i.i.d.*) Pauli errors with probability p.

Apart from Pauli errors, another common error type used is the *erasure error*. This error implies the loss of a qubit—*e.g.*, a photon that is absorbed in a waveguide somewhere along the way. This loss comes with a "flag" that tells us the qubit is lost—*e.g.*, because we notice that the photon is no longer there.

Depolarizing channel. The single-qubit depolarizing channel is defined as

$$\mathcal{N}_{\text{depol1}}(\rho,\gamma) \equiv (1-\gamma)\rho + \frac{\gamma}{3}\sum_{P\in\{X,Y,Z\}} P\rho P = \left(1-\frac{4\gamma}{3}\right)\rho + \frac{4\gamma}{3}\frac{\mathbb{I}}{2}. \tag{2.9}$$

The two-qubit depolarizing channel is defined as

$$\mathcal{N}_{\text{depol2}}(\rho,\gamma) \equiv (1-\gamma)\rho + \frac{\gamma}{15}\sum_{(P,P')} (P\otimes P')\rho(P\otimes P') = \left(1-\frac{16\gamma}{15}\right)\rho + \frac{16\gamma}{15}\frac{\mathbb{I}^{\otimes 2}}{4}. \tag{2.10}$$

For the sum in Eq. (2.10), the operators (P,P') are Pauli operators taken from $\{\mathbb{I},X,Y,Z\}^2\setminus(\mathbb{I},\mathbb{I})$. Further, both the single-qubit and the two-qubit channels have $0\leq\gamma\leq 1$. For $\gamma=1$, we obtain the maximally mixed state. For the depolarizing channel, the occurrence of errors follows a uniform distribution, meaning that all non-trivial Pauli errors are equally likely to occur.

Dephasing (phase damping) channel. The dephasing channel describes what happens to a qubit when it loses its phase information as a result of a stochastic phase-flip—*i.e.*, a Pauli-Z error. It is defined as:

$$\mathcal{N}_{\text{dephase}}(\rho,\gamma) \equiv (1-\gamma)\rho + \gamma Z\rho Z. \tag{2.11}$$

In Ch. 7, we additionally make use of a variant on the dephasing channel, *i.e.*, the phase damping channel $\mathcal{N}_{\text{PD}}(\rho,\gamma')$, that is defined by the Kraus operators

$$K_{\text{PD}}^{(1)} = \begin{bmatrix} 1 & 0 \\ 0 & \sqrt{1-\gamma'} \end{bmatrix}, \qquad K_{\text{PD}}^{(2)} = \begin{bmatrix} 0 & 0 \\ 0 & \sqrt{\gamma'} \end{bmatrix}. \tag{2.12}$$

This channel is equivalent to the dephasing channel by setting $\sqrt{1-\gamma'} = 1-2\gamma$.

(General) amplitude damping channel. The *amplitude damping channel* is an example of a channel that is conveniently written in terms of its Kraus operators. It is commonly used to describe what happens to a state when it naturally relaxes (or decoheres) to the ground state of a physical system. Typically, this state is labeled as $|0\rangle$. For example, if our qubit is a photon and we use $|1\rangle$ to define the presence of the photon and $|0\rangle$ to define its absence. The amplitude damping channel can now be used to describe a channel that might lead to the loss of our photon. This standard version of the amplitude damping channel, $\mathcal{N}_{\text{AD}}(\rho,\gamma)$, is defined by the following Kraus operators:

$$K_{\text{AD}}^{(1)} = \begin{bmatrix} 1 & 0 \\ 0 & \sqrt{1-\gamma} \end{bmatrix}, \qquad K_{\text{AD}}^{(2)} = \begin{bmatrix} 0 & \sqrt{\gamma} \\ 0 & 0 \end{bmatrix}. \tag{2.13}$$

In principle, it is, however, possible to construct a *general* amplitude damping channel that relaxes to any state [13]. Specifically, another useful version of the amplitude damping channel is the one that relaxes to the maximally mixed state introduced in Sec. 2.2.2. This channel, $\mathcal{N}_{\text{GAD}}(\rho,\gamma)$, is defined by the following Kraus operators:

$$\begin{aligned} K_{\text{GAD}}^{(1)} &= \frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 0 \\ 0 & \sqrt{1-\gamma} \end{bmatrix}, & K_{\text{GAD}}^{(2)} &= \frac{1}{\sqrt{2}}\begin{bmatrix} 0 & \sqrt{\gamma} \\ 0 & 0 \end{bmatrix}, \\ K_{\text{GAD}}^{(3)} &= \frac{1}{\sqrt{2}}\begin{bmatrix} \sqrt{1-\gamma} & 0 \\ 0 & 1 \end{bmatrix}, & K_{\text{GAD}}^{(4)} &= \frac{1}{\sqrt{2}}\begin{bmatrix} 0 & 0 \\ \sqrt{\gamma} & 0 \end{bmatrix}. \end{aligned} \tag{2.14}$$

In Fig. 2.10 we show how the channels $\mathcal{N}_{\mathrm{GAD}}(\rho,\gamma)$ and $\mathcal{N}_{\mathrm{PD}}(\rho,\gamma')$ transform states on the Bloch sphere. If we combine $\mathcal{N}_{\mathrm{PD}}(\rho,\gamma)$ and $\mathcal{N}_{\mathrm{GAD}}(\rho,\gamma)$ for the same noise parameter γ, we get $\mathcal{N}_{\mathrm{depol1}}(\rho,3\gamma/4)$.

In the remainder of this thesis, if there is potential for confusion on which qubit(s) a channel works, we use the notation $\mathcal{N}_{\mathrm{name},q}$ to denote a channel defined as "name" working on qubit(s) q.

2.4.2 Noisy states

To quantify the noise present on a quantum operation, a quantum channel, or a quantum state, we typically make use of a *fidelity* measure. This measure can usually be interpreted as the probability of achieving a noiseless version of the operation, channel, or state—*i.e.*, based on the average influence of the noise source. We define the fidelity F between two quantum states ρ and ρ' as [13, 14]

$$F(\rho,\rho') \equiv \left(\mathrm{Tr}\sqrt{\sqrt{\rho}\rho'\sqrt{\rho}}\right)^2 = \left(\mathrm{Tr}\left|\sqrt{\rho}\sqrt{\rho'}\right|\right)^2. \tag{2.15}$$

If one of the two states is a pure state $\rho' = |\psi\rangle\langle\psi|$, this definition of the fidelity simplifies to $F(\rho,\rho') = \langle\psi|\rho|\psi\rangle$.

Sometimes the fidelity is referred to as the "distance" of a quantum object from its ideal version. There is also another measure to denote the distance between two quantum states ρ and ρ': the *trace distance*. In this thesis, we use the following definition for the trace distance $\mathcal{T}$ [13]:

$$\mathcal{T}(\rho,\rho') \equiv \frac{1}{2}\mathrm{Tr}\sqrt{(\rho-\rho')^\dagger(\rho-\rho')} = \frac{1}{2}\mathrm{Tr}\left|\rho-\rho'\right|. \tag{2.16}$$

The fidelity $F(\rho,\rho')$ can be used to bound the trace distance $\mathcal{T}(\rho,\rho')$ via [13]

$$1-\sqrt{F(\rho,\rho')} \leq \mathcal{T}(\rho,\rho') \leq \sqrt{1-F(\rho,\rho')}. \tag{2.17}$$

These inequalities can be useful to estimate the fidelity, as calculating the trace distance is often computationally easier.

The fidelity and trace distance can be used to, *e.g.*, describe how close a state is to one of the Bell states of Eq. (2.3). The four Bell states can be used as the basis of any two-qubit quantum state—*i.e.*, as the so-called Bell basis. Applying a depolarizing channel on, *e.g.*, the state $|\Phi^+\rangle$ gives rise to a state that is diagonal in this basis—*i.e.*, a *Bell diagonal state*:

$$\rho_{p_{00},p_{01},p_{10},p_{11}} \equiv p_{00}|\Phi^+\rangle\langle\Phi^+| + p_{01}|\Psi^+\rangle\langle\Psi^+| + p_{10}|\Phi^-\rangle\langle\Phi^-| + p_{11}|\Psi^-\rangle\langle\Psi^-|. \tag{2.18}$$

Here, the coefficients should fulfill $p_{00}+p_{01}+p_{10}+p_{11}=1$. The coefficient p_{00} can be considered as the fidelity of this state with respect to the ideal state $|\Phi^+\rangle$. Since $|\Phi^-\rangle = (\mathbb{I}\otimes Z)|\Phi^+\rangle$, $|\Psi^+\rangle = (\mathbb{I}\otimes X)|\Phi^+\rangle$, and $|\Psi^-\rangle = -i(\mathbb{I}\otimes Y)|\Phi^+\rangle$, we sometimes regard the Bell pair states $|\Phi^-\rangle$, $|\Psi^+\rangle$, and $|\Psi^-\rangle$ as noisy versions of $|\Phi^+\rangle$. This means that applying a Pauli channel—as, *e.g.*, a depolarizing or a dephasing channel—on one of the qubits of the state $|\Phi^+\rangle$ gives rise to a Bell diagonal state. If a Bell diagonal state has $p_{01}=p_{10}=p_{11}=(1-p_{00})/3$, we refer to it as an *isotropic* Bell state $\rho_{p_{00}}$. This state is also known as the *Werner state*.

		Computational basis	X_1X_2	Z_1Z_2
$\|\Phi^+\rangle$	$\|\phi_{00}\rangle$	$(\|00\rangle + \|11\rangle)/\sqrt{2}$	$+1$	$+1$
$\|\Psi^+\rangle$	$\|\phi_{01}\rangle$	$(\|01\rangle + \|10\rangle)/\sqrt{2}$	$+1$	-1
$\|\Phi^-\rangle$	$\|\phi_{10}\rangle$	$(\|00\rangle - \|11\rangle)/\sqrt{2}$	-1	$+1$
$\|\Psi^-\rangle$	$\|\phi_{11}\rangle$	$(\|01\rangle - \|10\rangle)/\sqrt{2}$	-1	-1

Table 2.1: The four Bell states from Eq. (2.3), their alternative descriptions $|\phi_{s_1s_2}\rangle$ for $s_1, s_2 \in \{0,1\}$, their representations in the computational basis, and their relations with respect to the stabilizer operators X_1X_2 and Z_1Z_2. For each state, the labels s_1 and s_2 in $|\phi_{s_1s_2}\rangle$ indicate that the state is stabilized by $(-1)^{s_1}X_1X_2$ and $(-1)^{s_2}Z_1Z_2$.

In Eq. (2.18), the labeling of the coefficients $p_{s_1s_2}$ is based on the stabilizers of the Bell states, as indicated in more detail in Table 2.1, with $s_1, s_2 \in \{0,1\}$. We additionally use the short-hand notation $|\phi_{s_1s_2}\rangle$ to describe Bell states themselves based on the signs of their stabilizers. A similar notation can be used for states based on different signs of the stabilizer generators of the GHZ state $|\text{GHZ}^{(N)}\rangle = (|0\rangle^{\otimes N} + |1\rangle^{\otimes N})/\sqrt{2}$. These states form a basis for any N-qubit state, with the basis states defined as the 2^N states $|\phi_{s_1s_2s_3\ldots s_N}\rangle$ with stabilizer generators $(-1)^{s_1}X_1X_2\ldots X_N$, $(-1)^{s_2}Z_1Z_2$, $(-1)^{s_3}Z_2Z_3$, …, $(-1)^{s_N}Z_{N-1}Z_N$. Here, $s_i \in \{0,1\}$ indicates the sign of the stabilizer generator for all $i \in \{1,2,\ldots,N\}$. In analogy with the Bell basis, we refer to this basis as the *GHZ basis*. It is also known as the *cat basis* [15]. As an example, we show the eight basis states of the 3-qubit GHZ basis in Table 2.2. For any N, the basis state $|\phi_{00\ldots0}\rangle$ is the main N-qubit GHZ state $|\text{GHZ}^{(N)}\rangle$.

In Sec. 2.4.4, we show how any state can be converted to an approximate version that is diagonal in the Bell or GHZ basis. In this thesis, *e.g.*, in Ch. 5, we use this approach to efficiently describe the states of noisy Bell and GHZ states generated for distributed quantum computing. To simplify the expressions of Bell and GHZ diagonal states, we sometimes use the capital Φ symbol to denote the density matrix corresponding to a Bell or GHZ basis state—*i.e.*, $\Phi_{s_1s_2\ldots s_N} \equiv |\phi_{s_1s_2\ldots s_N}\rangle\langle\phi_{s_1s_2\ldots s_N}|$ for a state in the N-qubit GHZ basis and $\Phi_{s_1s_2} \equiv |\phi_{s_1s_2}\rangle\langle\phi_{s_1s_2}|$ for states in the Bell basis.

2.4.3 Noisy operations

Noisy operations are usually modeled with a noise channel applied after an ideal version of the operation. For example, a noisy two-qubit gate $\mathcal{U}$ on a state ρ can be modeled with the two-qubit depolarizing channel: $\rho \mapsto \mathcal{N}_{\text{depol2}}(\mathcal{U}\rho\mathcal{U}, \gamma)$. Different noise channels can be applied based on the (characterization of the) physical implementation of the operation. Just as for noisy states, we can define the fidelity of a noisy quantum channel with its ideal version. The fidelity of two quantum channels $\mathcal{N}_1$ and $\mathcal{N}_2$ is defined as [13]

$$F(\mathcal{N}_1, \mathcal{N}_2) \equiv \min_{|\psi\rangle} F\left(\mathcal{N}_1(|\psi\rangle\langle\psi|),\ \mathcal{N}_2(|\psi\rangle\langle\psi|)\right). \tag{2.19}$$

A common type of noise in all quantum systems is decoherence. This process can be interpreted as the conversion of quantum states to states that can be described purely by the laws of classical mechanics—*i.e.*, as the loss of quantum *coherence*. A state is coherent if there exists a definite phase relation between the components in its superposition—*e.g.*,

	Computational basis	$X_1X_2X_3$	Z_1Z_2	Z_2Z_3
$\|\phi_{000}\rangle$	$(\|000\rangle+\|111\rangle)/\sqrt{2}$	$+1$	$+1$	$+1$
$\|\phi_{001}\rangle$	$(\|001\rangle+\|110\rangle)/\sqrt{2}$	$+1$	$+1$	-1
$\|\phi_{010}\rangle$	$(\|011\rangle+\|100\rangle)/\sqrt{2}$	$+1$	-1	$+1$
$\|\phi_{011}\rangle$	$(\|010\rangle+\|101\rangle)/\sqrt{2}$	$+1$	-1	-1
$\|\phi_{100}\rangle$	$(\|000\rangle-\|111\rangle)/\sqrt{2}$	-1	$+1$	$+1$
$\|\phi_{101}\rangle$	$(\|001\rangle-\|110\rangle)/\sqrt{2}$	-1	$+1$	-1
$\|\phi_{110}\rangle$	$(\|011\rangle-\|100\rangle)/\sqrt{2}$	-1	-1	$+1$
$\|\phi_{111}\rangle$	$(\|010\rangle-\|101\rangle)/\sqrt{2}$	-1	-1	-1

Table 2.2: The basis states $|\phi_{s_1s_2s_3}\rangle$ of the 3-qubit GHZ basis, their representations in the computational basis, and the stabilizers of the state. The signs s_1, s_2, and s_3 describe the relation of the basis state $|\phi_{s_1s_2s_3}\rangle$ with stabilizer generators $(-1)^{s_1}X_1X_2X_3$, $(-1)^{s_2}Z_1Z_2$, and $(-1)^{s_3}Z_2Z_3$.

the states $(|0\rangle+|1\rangle)/\sqrt{2}$ and $(|0\rangle-|1\rangle)/\sqrt{2}$ are both coherent states. If this phase relation is unknown or lost, the state has (partly) decohered. This is accompanied by the loss of quantum information to the surrounding environment. In practical situations, every state eventually fully decoheres to a mixed state—*i.e.*, a statistical ensemble—in which the uncertainties are purely classical. In theory, a system that is completely isolated from its environment does not experience decoherence. However, for quantum computing purposes, such a system is not useful, as we can not perform operations on it or read out its outcomes.

We use the channels described in Sec. 2.4.1 to model *memory* decoherence—*i.e.*, decoherence emerging over time on states that are stored in memory. For example, we can formulate the noise parameter γ of such a channel as $\gamma = 1-e^{-t/T}$, where t is the time (starting at $t=0$) and T is the coherence time of the concerning state. In this way, the state does not experience any decoherence at $t=0$ (since here $\gamma=0$) and has fully decohered at $t=\infty$ (since here $\gamma=1$). More specifically, we introduce a T_1 and T_2 coherence time associated with the general amplitude damping channel $\mathcal{N}_{\mathrm{GAD}}(\rho,\gamma_1)$ and the phase damping channel $\mathcal{N}_{\mathrm{PD}}(\rho,\gamma_2)$, respectively, with $\gamma_1 = 1-e^{-t/T_1}$ and $\gamma_2 = 1-e^{-t/T_2}$.

2.4.4 Pauli twirling

In Sec. 2.3.4, the stabilizer formalism is introduced as an effective way to describe stabilizer states: states stabilized by Pauli operators. In quantum error correction, stabilizer states are important as they simplify the analysis of error-correction codes. Since for applying a Pauli operator on a stabilizer state one only needs to consider (anti-)commutation relations of the operator with the stabilizers of the state, Pauli (error) channels (as introduced in Sec. 2.4.1) are typically relatively easy to incorporate in this analysis. Importantly, channels in which the Kraus operators can not be described as Pauli operators can be converted to Pauli channels with a technique called Pauli twirling. This, *e.g.*, allows one to approximately apply a non-Pauli channel in the stabilizer-based analysis of an error-correction code. We use Pauli twirling for exactly this purpose in Chs. 7 and 8 of this thesis. In general, twirling [16] can be applied to both states and channels.

We first consider twirling a state. We consider an N-qubit mixed state ρ and an N-qubit stabilizer group $\mathcal{J} = \langle g_i \rangle_{i=1}^{N}$ that stabilizes a pure state $|\psi\rangle$. Twirling can be applied by "averaging" ρ over all stabilizer operators of the selected state $|\psi\rangle$:

$$\mathrm{Tw}(\rho) \equiv \frac{1}{|\mathcal{J}|} \sum_{P \in \mathcal{J}} P \rho P. \tag{2.20}$$

This leads to a state that can be interpreted as the pure state $|\psi\rangle$ with a Pauli channel acting on it. To see this, one first has to use $|\psi\rangle$ to construct an N-qubit basis. This can be achieved with the so-called *destabilizer group* [17] of $|\psi\rangle$. This is a group $\overline{\mathcal{J}} = \langle d_i \rangle_{i=1}^{N}$ that is directly associated with the stabilizer group $\mathcal{J} = \langle g_i \rangle_{i=1}^{N}$ of $|\psi\rangle$. The destabilizer group $\overline{\mathcal{J}}$ has the property that one can identify generators in such a way that each generator d_i *anti-commutes* with exactly one generator g_i of $\mathcal{J}$, but *commutes* with all other generators g_j for $j \neq i$. An orthonormal set of basis states can now be identified as $\{P|\psi\rangle\}_{P \in \overline{\mathcal{J}}}$. This makes it possible to write ρ in this basis, and implement this in Eq. (2.20):

$$\begin{aligned} \rho &= \sum_{P_m, P_n \in \overline{\mathcal{J}}} \lambda_{mn} P_m |\psi\rangle\langle\psi| P_n, \\ \mathrm{Tw}(\rho) &= \frac{1}{|\mathcal{J}|} \sum_{P_s \in \mathcal{J}} \sum_{P_m, P_n \in \overline{\mathcal{J}}} \lambda_{mn} P_s P_m |\psi\rangle\langle\psi| P_n P_s = \sum_{P_m \in \overline{\mathcal{J}}} \lambda'_m P_m |\psi\rangle\langle\psi| P_m. \end{aligned} \tag{2.21}$$

Here, we have defined $\lambda'_m \equiv \lambda_{mm} = \langle\psi| P_m \rho P_m |\psi\rangle$. In Eq. (2.21), it is possible to simplify the expression for $\mathrm{Tw}(\rho)$ because, for each combination $P_m \neq P_n$, exactly half of the operators $P_s \in \mathcal{J}$ commute with P_m and anti-commute with P_n (or vice versa), while the other half (anti-)commute with both P_m and P_n—*i.e.*, all terms $P_m \neq P_n$ cancel out. We see that the twirling operation gives rise to a Pauli channel acting on $|\psi\rangle$. The operation does not alter the original contribution of $|\psi\rangle\langle\psi|$ in ρ, nor does it alter any of the other contributions that are diagonal in the $\{P|\psi\rangle\}_{P \in \overline{\mathcal{J}}}$ basis—this makes that the operation is trace-preserving. Therefore, if ρ is, *e.g.*, a noisy version of $|\psi\rangle$, twirling only changes the terms that are considered noise contributions in the expression for ρ. Moreover, twirling ρ over the stabilizer group $\mathcal{J}$ removes all off-diagonal elements of ρ in the $\{P|\psi\rangle\}_{P \in \overline{\mathcal{J}}}$ basis.

Since, as mentioned in Sec. 2.4.2, any two-qubit or multi-qubit quantum state can be defined using the Bell and GHZ basis, respectively, *Pauli twirling* can be used to convert any state into a Bell or GHZ diagonal state. This is because the basis states of the Bell and GHZ basis can be created with the degenerator groups of $|\Phi^+\rangle$ and $|\mathrm{GHZ}^{(N)}\rangle$. This realization plays an important role in entanglement distillation protocols, discussed in Sec. 3.6, as considering diagonal instead of general states simplifies their analysis.

To produce a Pauli channel approximation of a general channel defined as $\mathcal{N}(\rho) : \rho \mapsto \sum_k \boldsymbol{\mathcal{K}}^{(k)} \rho (\boldsymbol{\mathcal{K}}^{(k)})^\dagger$, one typically twirls the channel over the full N-qubit Pauli group $\boldsymbol{\mathcal{I}}^{(N)}$ via

$$\begin{aligned} \mathrm{Tw}(\mathcal{N}(\rho)) &\equiv \frac{1}{|\boldsymbol{\mathcal{I}}^{(N)}|} \sum_{P_p \in \boldsymbol{\mathcal{I}}^{(N)}} P_p \mathcal{N}(P_p \rho P_p^\dagger) P_p^\dagger = \frac{1}{|\boldsymbol{\mathcal{I}}^{(N)}|} \sum_{P_p \in \boldsymbol{\mathcal{I}}^{(N)}} \sum_k (P_p \boldsymbol{\mathcal{K}}^{(k)} P_p) \rho (P_p \boldsymbol{\mathcal{K}}^{(k)} P_p)^\dagger \\ &= \sum_{P_m \in \boldsymbol{\mathcal{I}}^{(N)}} \left(\sum_k \left| \xi_m^{(k)} \right|^2 \right) P_m \rho P_m^\dagger. \end{aligned} \tag{2.22}$$

Here, the last step makes use of the fact that, as mentioned in Sec. 2.3.2, the N-qubit Pauli group forms a basis for $2^N \times 2^N$ matrices—*i.e.*, each Kraus operator $\mathcal{K}^{(k)}$ can be decomposed in terms of Pauli operators via $\mathcal{K}^{(k)} = \sum_{P_m \in \mathcal{I}^{(N)}} \xi_m^{(k)} P_m$.

We emphasize that other implementations of twirling are possible as well. However, in this thesis, we restrict to these two types of Pauli twirling. In practice, twirling can, *e.g.*, be applied by randomly selecting an operator from the twirl group, applying that operator, and forgetting the result. However, we should rather interpret it as a theoretical trick that allows one to transform channels and states into approximate versions that are easier to apply or analyze.

References

[1] D. Gottesman, *Stabilizer Codes and Quantum Error Correction.* Doctoral thesis, California Institute of Technology, Pasadena, California, United States, May 1997.

[2] D. Gottesman, *The Heisenberg Representation of Quantum Computers.* arXiv: quant-ph/9807006, July 1998.

[3] M. Hein, W. Dür, J. Eisert, R. Raussendorf, M. V. den Nest, and H.-J. Briegel, *Entanglement in Graph States and Its Applications.* arXiv: quant-ph/0602096, Feb. 2006.

[4] A. Broadbent, J. Fitzsimons, and E. Kashefi, "Universal blind quantum computation," in *2009 50th Annual IEEE Symposium on Foundations of Computer Science*, pp. 517–526, Oct. 2009.

[5] J. van Dam, G. Avis, T. B. Propp, F. Ferreira da Silva, J. A. Slater, T. E. Northup, and S. Wehner, "Hardware requirements for trapped-ion-based verifiable blind quantum computing with a measurement-only client," *Quantum Science and Technology*, vol. 9, p. 045031, Aug. 2024.

[6] C. H. Bennett, G. Brassard, C. Crépeau, R. Jozsa, A. Peres, and W. K. Wootters, "Teleporting an unknown quantum state via dual classical and Einstein-Podolsky-Rosen channels," *Physical Review Letters*, vol. 70, pp. 1895–1899, Mar. 1993.

[7] D. Bouwmeester, J.-W. Pan, K. Mattle, M. Eibl, H. Weinfurter, and A. Zeilinger, "Experimental quantum teleportation," *Nature*, vol. 390, pp. 575–579, Dec. 1997.

[8] D. Boschi, S. Branca, F. De Martini, L. Hardy, and S. Popescu, "Experimental realization of teleporting an unknown pure quantum state via dual classical and Einstein-Podolsky-Rosen channels," *Physical Review Letters*, vol. 80, pp. 1121–1125, Feb. 1998.

[9] K. Fujii, *Quantum Computation with Topological Codes: From Qubit to Topological Fault-Tolerance*, vol. 8 of *SpringerBriefs in Mathematical Physics.* Singapore: Springer, 2015.

[10] D. Gottesman and I. L. Chuang, "Quantum teleportation is a universal computational primitive," *Nature*, vol. 402, pp. 390–393, Nov. 1999.

[11] N. Nickerson, *Practical Fault-Tolerant Quantum Computing.* Doctoral thesis, Imperial College London, London, United Kingdom, 2015.

[12] H.-B. Chen, P.-Y. Lo, C. Gneiting, J. Bae, Y.-N. Chen, and F. Nori, "Quantifying the nonclassicality of pure dephasing," *Nature Communications*, vol. 10, p. 3794, Aug. 2019.

[13] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information*. Cambridge University Press, 2000.

[14] R. Jozsa, "Fidelity for mixed quantum states," *Journal of Modern Optics*, vol. 41, pp. 2315–2323, Dec. 1994.

[15] E. N. Maneva and J. A. Smolin, "Improved two-party and multi-party purification protocols," *Contemporary Mathematics*, vol. 305, pp. 203–212, 2002.

[16] C. H. Bennett, D. P. DiVincenzo, J. A. Smolin, and W. K. Wootters, "Mixed-state entanglement and quantum error correction," *Physical Review A - Atomic, Molecular, and Optical Physics*, vol. 54, no. 5, pp. 3824–3851, 1996.

[17] S. Aaronson and D. Gottesman, "Improved simulation of stabilizer circuits," *Physical Review A*, vol. 70, p. 052328, Nov. 2004.

3

Quantum error correction

In this chapter, we introduce structures and frameworks that can be used to detect unwanted errors during quantum computation. We start by going over some of the basic elements of quantum error-correction codes. We then describe a class of topological codes that can be used to protect states against Pauli errors and discuss how codes from this class can be used to realize fault-tolerant universal quantum computing. These codes are relevant in Ch. 7 of this thesis, where we investigate a distributed implementation of a two-dimensional topological code known as the surface code. Next, we introduce codes defined on cluster states that can be used to achieve fault-tolerant versions of measurement-based quantum computation. This part is relevant to Ch. 8, where we investigate the identification and construction of distributed error-correction channels of this type. At the end of this chapter, we introduce relevant prior work on entanglement distillation, which can be used to increase the purity of noisy entangled states. Next to introducing specific protocols for entanglement distillation, we show how entanglement distillation can be regarded as an application of quantum error correction.

3.1 Introduction

In the struggle against unwanted noise sources that torment quantum computers, a lot of faith is put into quantum error-correction codes. Protecting quantum information with an error-correction code involves *encoding* the information. In general, the idea is that a large number of physical qubits hold a smaller number of encoded logical states. Therefore, such a code typically uses a large vector space—*i.e.*, a vector space associated with a large number of qubits—in which only a subset of all possible states is used for calculations. This allows us to locate and undo at least part of the noise that can occur in our system, as long as we assume none of the states used for calculation are "too closely" located near each other in the vector space and the errors only cause small deviations to these states. We convert these statements to more formal descriptions below.

A *stabilizer code* [1] is a specific type of quantum error-correction code in which the logical states—*i.e.*, the states holding the encoded information—are defined with stabilizer operators. We introduce these codes in Sec. 3.2 and describe how, in general, they can be initialized and used to correct errors. The first quantum error-correction code was introduced by Peter Shor in 1995 [2]. More codes followed soon after and continue to be developed as we speak. Given that error correction was already applied on classical computers before the first idea of a quantum computer was even brought forward, it is no surprise that these classical error-correction codes form the basis of many quantum error-correction codes. We discuss a specific class of codes inspired by classical codes in Sec. 3.2.4—this class forms a subclass of the stabilizer codes.

Using mathematical ideas and formulations in Sec. 3.3, we introduce a stabilizer code construction that exploits the topological properties of a two-dimensional lattice in Sec. 3.4. This construction leads to a promising class of stabilizer codes known as topological codes. These codes are called "topological" because they protect quantum information over a "macroscopic" distance on a *topological space*. Roughly speaking, they encode qubits as the "holes" of such a space—these holes can also be lattice boundaries. Topological codes have a large tolerance against errors that, *e.g.*, typically appear when applying a two-qubit gate. However, in their standard definition, these codes are not able to deal with errors in the measurements required to locate the errors themselves. Luckily, this can be remedied by repeating these measurements over multiple (time) layers and comparing the results from one layer to the next.

If we teleport qubits of a topological code to the next time layer we naturally end up with a code defined on a structure one dimension higher than the topological code—this dimension being the time dimension for the original topological code. In Sec. 3.5, this approach is used to reformulate topological codes as cluster states. Next to the qubits of the original code, these cluster states contain qubits that perform the error-detection measurements of the original code, as well as qubits used for teleportation of the code to other layers. As discussed below, the reconstruction of two-dimensional topological codes as three-dimensional cluster state channels allows one to identify more error-correction channels based on general three-dimensional cluster states. Since cluster states are the main resource of measurement-based quantum computation (MBQC), this directly brings topological codes to the realm of MBQC. The analysis in Secs. 3.4 and 3.5 includes brief discussions on how general operations can be performed with these error-correction structures.

3.2 Stabilizer error-correction codes

Below, we give a general introduction to the topic of stabilizer (error-correction) codes in Sec. 3.2.1. Furthermore, in Secs. 3.2.2 and 3.2.3, we discuss how one can initialize a code and use it to detect and correct errors. In Sec. 3.2.4, we focus on a subclass of stabilizer codes: the so-called *Calderbank-Shor-Steane codes* (CSS codes) [3, 4]. The main ideas in Secs. 3.2.1, 3.2.2, 3.2.3, and 3.2.4 are derived from the textbook by Nielsen and Chuang [5]. In Sec. 3.2.5, we introduce the typical threshold behavior existing in error-correction codes.

3.2.1 Main properties

In Sec. 2.3.4, we introduced the stabilizer formalism as a useful tool to describe (vector spaces of) quantum states efficiently in terms of operators that stabilize them. If it is possible to use the stabilizer formalism to describe the logical states of an error-correction code, we call the code a stabilizer code. A stabilizer code is defined by both its stabilizer generators and logical operators. The generators produce the stabilizers of the logical states, whereas the logical operators describe the relation between the logical states. The stabilizer generators, and all stabilizer operators generated by them, are N-qubit operators that leave the logical states of the error-correction code invariant. A set of generators of the stabilizer group should always be a set of *independent* operators—*i.e.*, it should be impossible to write a generator operator as a product of other generator operators. These generators, and therefore the stabilizer operators themselves, are all mutually commuting operators. All stabilizer operators should also commute with the logical operators of the code. On top of that, the encoded logical operators must mirror the (anti-)commutation relations of the single-qubit operators they portray—*e.g.*, a pair of logical X and logical Z operators should anti-commute, whereas two distinct logical operators that both have the function of a Pauli-X operator should commute.

A code with M independent stabilizer generators and N physical qubits has $K = N - M$ free qubit degrees of freedom and encodes K logical qubits. We define an $[\![N, K, d]\!]$ stabilizer code as a code with N physical qubits that encodes K logical qubits of at least a distance d apart from each other. Here, the *distance* d is defined as the minimum weight of the Pauli operator that brings us from one logical state to another—*i.e.*, the weight of the smallest (non-identity) logical operator. The distance directly corresponds to the weight of the smallest undetectable error: a code with distance d is capable of detecting errors with a maximum weight of $d-1$ [6]. If we not only want to detect but also *correct* the errors, the restriction on the separation between the logical states becomes even stricter. That is because logical states that are too close to each other could result in corrections toward the wrong codewords. Therefore, a code with distance d is capable of correcting errors with a maximum weight of $\lfloor (d-1)/2 \rfloor$.

Because the stabilizer operators leave the code invariant, in the absence of errors, measuring the stabilizer generators should always lead to $m = +1$ measurement outcomes. This can be used to detect errors in the code, as, in the presence of errors, some stabilizer generators lead to $m = -1$ measurement outcomes. The set of obtained stabilizer measurement outcomes is called the *error syndrome*. By *decoding* this error syndrome, we can find (the most likely configuration of) physical errors on the qubits of the code, and correct them. An *error syndrome decoder* is a (classical) algorithm that identifies error-correction operators from an error syndrome.

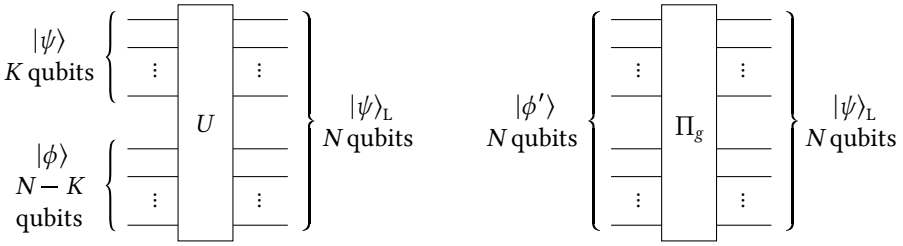


Figure 3.1: (a) Encoding a K-qubit state $|\psi\rangle$ into an N-qubit logical state $|\psi\rangle_{\mathrm{L}}$. Together with the encoding circuit U, the $(N-K)$-qubit state $|\phi\rangle$ defines the stabilizer operators of the error-correction code. (b) Realizing the encoded state $|\psi\rangle_{\mathrm{L}}$ by initializing all qubits in a certain state $|\phi'\rangle$ and measuring the code's stabilizer operators with the projector Π_g of Eq. (3.1). See Sec. 3.2.2 in the main text for more details.

3.2.2 Code initialization

As indicated in Sec. 3.2.1, an $[\![N,K,d]\!]$ stabilizer code is fully defined by the $M=N-K$ independent operators that generate the code's stabilizer operators and by the logical Z and X operators. We can alternatively define the code by an *encoding unitary* or *encoding circuit* U and the stabilizer generators $\{g_i'\}_{i=1}^{M}$ of an "ancillary" state $|\phi\rangle$ on $M=N-K$ qubits. As depicted in Fig. 3.1a, $|\phi\rangle$ and U can be used to decode a K-qubit state $|\psi\rangle$ over all N qubits. The stabilizer group of this code can be written as $\langle g_i = U g_i' U^\dagger \rangle_{i=1}^{M}$. Common choices for the ancillary state are $|\phi\rangle = |0\rangle^{\otimes M}$ and $|\phi\rangle = |+\rangle^{\otimes M}$, which lead to code stabilizers $\langle UZ_iU^\dagger\rangle_{i=1}^{M}$ and $\langle UX_iU^\dagger\rangle_{i=1}^{M}$, respectively. Typically, the logical X and Z operators of the code are identified by transforming the single-qubit Pauli-X and Pauli-Z operators on the qubits of $|\psi\rangle$ with the encoding circuit U: $\overline{X}_j = UX_jU^\dagger$ and $\overline{Z}_j = UZ_jU^\dagger$, where $j\in\{1,2,\ldots,K\}$ are the qubits of $|\psi\rangle$, and $\overline{X}_j$ and $\overline{Z}_j$ denote the logical X and Z operators, respectively. This makes that, *e.g.*, the logical basis states of the computational basis are given by $|b\rangle_{\mathrm{L}} = U|b\rangle|\phi\rangle$, for $b\in\{0,1\}^K$.

An example of an encoding circuit U is depicted in Fig. 3.2. This circuit encodes a version of the *five-qubit error-correction code* [7]: an error-correction code with $N=5$ and $K=1$ that protects a logical qubit from arbitrary single-qubit Pauli errors. The stabilizer generators and logical operators can be identified by moving operators X_i and Z_i from left to right through the circuit U (*i.e.*, commuting them through the circuit), using the (anti-)commutation relations of Figs. 2.6b and 2.6c. A suitable encoding circuit U to establish a specific stabilizer code can be identified with an algorithm described by Gottesman [1, 8]. As discussed in Sec. 2.3.4, stabilizer operators are typically associated with Pauli operators. Therefore, the encoding unitary U of a stabilizer code is typically a Clifford circuit: such a circuit transforms a stabilizer ancillary state $|\phi\rangle$ to a code with Pauli stabilizer operators.

In some situations, a logical state $|\psi\rangle_{\mathrm{L}}$ can be created by initializing the qubits in a certain state $|\phi'\rangle$ and measuring the stabilizer generators of the code—*i.e.*, by projecting on $|\phi'\rangle$ with

$$\Pi_g = \prod_{i=1}^{N-K} \frac{\mathbb{I}^{\otimes N} + g_i}{2} = \prod_{i=1}^{N-K} \frac{\mathbb{I}^{\otimes N} + U g_i' U^\dagger}{2} = \frac{1}{2^{N-K}} \sum_{P\in\langle g_i\rangle_{i=1}^{N-K}} P. \tag{3.1}$$

This procedure is depicted in Fig. 3.1b and works in situations where the initial state $|\phi'\rangle$ is an eigenstate of (a representation of) the logical operators that stabilize $|\psi\rangle$. For example, if the logical Z operators of the code are tensor products with just single-qubit Pauli-Z operators, the logical $|0\rangle_{\mathrm{L}}$ state can be initialized with $|\phi'\rangle = |0\rangle^{\otimes N}$. Similarly, $|+\rangle_{\mathrm{L}}$ can be created with $|\phi'\rangle = |+\rangle^{\otimes N}$ if the logical X operators only contain single-qubit Pauli-X operators. For any stabilizer code it is possible to define logical Z operators that are tensor products of single-qubit Pauli-Z operators and single-qubit identity operators [1]. The CSS codes of Sec. 3.2.4 have the property that, on top of that, also logical X operators consisting of just Pauli-X operators and single-qubit identities can be identified. If projecting with Π_g on the initial state $|\phi'\rangle$ does not produce the desired state $|\psi\rangle_{\mathrm{L}}$, it is always possible to additionally measure logical operators to initialize the correct logical state. For example, one can also initialize the state $|0\rangle_{\mathrm{L}}$ by, next to measuring the stabilizer generators, also measuring the logical Z operators $\{\overline{Z}_j\}_{j=1}^{K}$.

In practice, initializing $|\psi\rangle_{\mathrm{L}}$ with Π_g requires both measuring the operators g_i (and possibly also the logical operators) as well as correcting for $m_i = -1$ measurement outcomes. In the case of an $m_i = -1$ measurement outcome, one needs to apply an operator that anti-commutes with g_i but commutes with all other stabilizer operators. The stabilizers of the ancillary state $|\phi\rangle$ can be used to identify such an operator. For example, in a situation where g_i is defined as $g_i = UZ_iU^\dagger$, the operator $UX_iU^\dagger$ anti-commutes with g_i and commutes with all other stabilizer operators. Alternatively, one can choose to simply redefine the stabilizer code with $m_i g_i$ as the stabilizer generators—*i.e.*, using $-g_i$ instead of g_i in case of an $m_i = -1$ measurement outcome during initializing. This "software correction" does not influence the error-correction properties of the stabilizer code. (In case of also measuring logical operators to initialize a logical state, a -1 measurement outcome should be followed with either a correction with the corresponding anti-commuting logical operator—*e.g.*, with $\overline{X}_j$ after measuring $\overline{Z}_j$ or vice versa—or a redefinition of the logical operator.)

3.2.3 Error correction

As mentioned in Sec. 3.2.1, after initializing the error-correction code, errors can be identified and localized by measuring the stabilizer generators $\{g_i\}_{i=1}^{M}$ of the code, for $M = N - K$. Formally, a set of Pauli errors $\{\varepsilon_k\}_k$ is *correctable* if, for all k and k', the combination $\varepsilon_k^\dagger \varepsilon_{k'}$ is either a stabilizer operator or an operator completely outside the *centralizer* of the stabilizer group [5]. The centralizer is the group of operators that commute with all

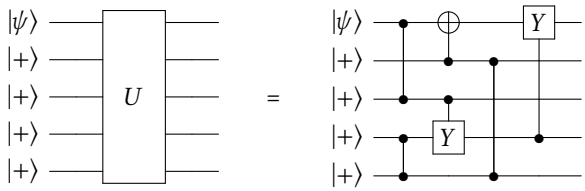


Figure 3.2: Encoding circuit U for (a version of) the $[\![5,1,3]\!]$ error-correction code. This representation of the five-qubit code uses $|\phi\rangle = |+\rangle^{\otimes 4}$ as the ancillary state. The associated stabilizer generators are $\{UX_iU^\dagger\}_{i=2}^{5} = \{X_1X_2Z_4Z_5, -X_1Z_2X_3X_4, Y_1Z_3X_4Z_5, Z_2Z_3Z_4X_5\}$. The logical Z and X operators are $\overline{Z} = UZ_1U^\dagger = Z_1Z_2Z_4$ and $\overline{X} = UX_1U^\dagger = X_1Z_3Z_4$. Other representations of the five-qubit error-correction code are possible as well, with a different encoding circuit and/or a different ancillary state $|\phi\rangle$.

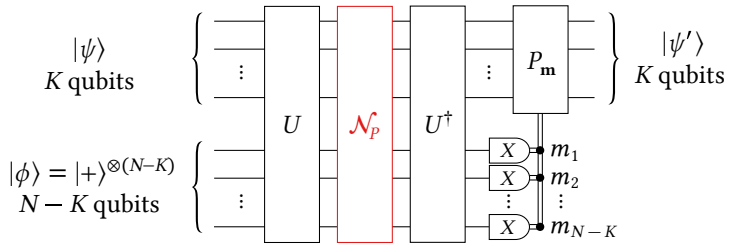


Figure 3.3: Detecting errors introduced by a noise channel $\mathcal{N}_P$ by decoding the logical information and measuring out the stabilizers of the ancillary state $|\phi\rangle$. For explanatory purposes, we use $|\phi\rangle = |+\rangle^{\otimes M}$ as the ancillary state here. In modeling error-correction codes, it is often assumed that the noise channel $\mathcal{N}_P$ only introduces Pauli errors—*e.g.*, with a depolarizing or a dephasing channel introduced in Sec. 2.4.1. An error syndrome $\mathbf{m} = \{m_1, m_2, \ldots, m_{N-K}\}$ that corresponds to an error ε in a correctable error set requires applying a correction $U^\dagger \varepsilon^\dagger U$. The operation $P_{\mathbf{m}}$ corresponds to the part of $U^\dagger \varepsilon^\dagger U$ applied on the first K qubits, as the state of the $N-K$ ancillary qubits is irrelevant at this point. If $\mathcal{N}_P$ only introduces errors of a correctable error set, one is guaranteed to end up with $|\psi'\rangle = |\psi\rangle$.

stabilizer operators. If $\varepsilon_k^\dagger \varepsilon_{k'}$ is in the centralizer, we have that $\varepsilon_k g_i \varepsilon_k^\dagger = \varepsilon_{k'} g_i \varepsilon_{k'}^\dagger$ holds for all stabilizer generators g_i. This means that it is not possible to distinguish between ε_k and $\varepsilon_{k'}$ when measuring the generators. This is, however, not a problem if $\varepsilon_k^\dagger \varepsilon_{k'}$ is a stabilizer operator itself, as in that case correcting with $\varepsilon_k^\dagger$ after an error $\varepsilon_{k'}$ implements a stabilizer operator and leaves the code invariant. The five-qubit code depicted in Fig. 3.2 can correct all single-qubit errors. This means that, for the five-qubit code, each of the errors in the correctable error set $\{X_i, Y_i, Z_i\}_{i=1}^5$ gives rise to a distinct error syndrome.

One way to measure the code's stabilizer generators is with the aid of ancillary qubits and a scheme based on Fig. 2.4a. The benefit of this approach is that an identified error ε_k can be corrected by simply applying $\varepsilon_k^\dagger$ while the information stays encoded. Alternatively, we can first decode the logical information and measure the original stabilizers of the ancillary state $|\phi\rangle$, as depicted in Fig. 3.3. Identifying a detectable error ε_k now requires a correction with an operator $U^\dagger \varepsilon_k^\dagger U$. This variant can be beneficial in situations where one wants to end up with a decoded version of the main state—*e.g.*, in entanglement distillation discussed in Sec. 3.6.

Typically, not all possible errors can be found with an error-correction code. However, a good error-correction code is usually able to find the *most probable* errors. In other words, if the errors that have the highest probability of occurring form a correctable error set of the code, the code filters these errors out with high probability. In practice, this means we might introduce *logical errors* with a small probability in situations where we misidentify an unlikely error ε_j for a more probable error ε_k from the correctable error set—*i.e.*, if we apply the wrong correction operation $\varepsilon_k^\dagger$, leading, in total, to a logical operator $\varepsilon_k^\dagger \varepsilon_j$ without us realizing. Typically, completely preventing logical errors is impossible: in practical situations, the best-performing error-correction codes are only able to identify the most likely errors and reduce the probability of errors accumulating to become a logical operator.

3.2.4 Calderbank-Shor-Steane codes

Defining a stabilizer code by an ancillary state $|\phi\rangle$ and an encoding circuit U often gives limited insight into the error-correcting properties of the code—as, *e.g.*, the distance of

the code. Therefore, it is useful to identify sets of stabilizer generators and logical operators that fulfill the (anti-)commutation requirements of a stabilizer code in different ways. Calderbank-Shor-Steane (CSS) codes combine the error-correcting properties of two classical codes C_1 and C_2 into a single quantum error-correction code. These codes form a subclass of all stabilizer codes.

To construct a general CSS code, we assume that C_1 encodes K_1 classical bits, C_2 encodes K_2 bits, and both C_1 and C_2 both use a total of N bits. We also assume that C_1 can correct errors of at least weight w, and the same is true for the *dual* of C_2 (denoted by $C_2^\perp$). The dual of a code are the codewords that are orthogonal to all codewords of the code—*i.e.*, their bitwise inner product with the codewords of the code is zero (modulo 2). We further require that $C_2 \subset C_1$ holds. This means that the classical codewords contained in C_2 should be a subset of all codewords that are part of C_1. We can now construct an $[\![N, K_1 - K_2, d]\!]$ quantum stabilizer code with $d \geq 2w+1$. We define $K_1 - K_2$ logical quantum states $|x + C_2\rangle$ of our CSS code as

$$|x + C_2\rangle \equiv \frac{1}{\sqrt{|C_2|}} \sum_{y \in C_2} |x + y\rangle . \tag{3.2}$$

Here, each logical state uses a different codeword $x \in C_1$. Furthermore, $|C_2| = 2^{K_2}$ is the number of codewords in C_2. The idea is that we use C_2 to split up the codewords of C_1 into different cosets. Each coset describes one logical state of our eventual quantum code: Eq. (3.2) describes how this logical state is a superposition of all states in the coset.

We see that CSS codes are rooted in classical codes. Classical codes are always defined by either a generator matrix $\mathcal{G}$ or, alternatively, a parity matrix $\mathcal{H}$. Both matrices contain entries that are either 0 or 1. For a classical code $\mathcal{C}$ of N bits that encodes K bits (*i.e.*, that generates 2^K codewords of length N), the generator matrix $\mathcal{G}$ is of size $N \times K$. This matrix describes how bit strings $y \in \{0,1\}^K$ are encoded into codewords $x \in \mathcal{C}$ via matrix multiplication of the form $x = \mathcal{G}y$. The corresponding parity matrix $\mathcal{H}$ is of size $(N-K) \times N$. The codewords $x \in \mathcal{C}$ are defined as the *kernel* or *null space* of this matrix—*i.e.*, they fulfill $\mathcal{H}x = \mathbf{0}$, with $\mathbf{0}$ the all-zero bit string (modulo 2). The columns of $\mathcal{G}$ should be linearly independent, as do the rows of $\mathcal{H}$. For the dual $\mathcal{C}^\perp$ of a classical code $\mathcal{C}$, we have that $\mathcal{H} = \mathcal{G}^T$ holds.

For a CSS code that is constructed from classical codes C_1 and C_2, the generators of the stabilizer group can be deduced from the parity matrices of C_1 and $C_2^\perp$. That is, each of the $(N - K_1)$ rows of the parity matrix of C_1 represents a stabilizer generator of the code: this operator is a tensor product on all N qubits with an identity on qubit i if there is a 0 on position i of the row or a Pauli-Z if there is a 1 on position i. Additionally, we have a stabilizer generator for each of the K_2 rows of the parity matrix of $C_2^\perp$: a tensor product with an identity on qubit i if there is a 0 on position i or a Pauli-X operator for a 1 on position i.

Looking at Eq. (3.2), we can understand that these operators generate the stabilizer operators of the code. The Z operators generated by the rows of the parity matrix of C_1 result in an inner product of zero when multiplied with codewords $x \in C_1$, and, therefore, with all terms of the superposition terms in Eq. (3.2). This means that, for these generators, the full combination of Pauli-Z operators is sure to give a +1 phase factor for each of the terms in the superposition of Eq. (3.2).

For the operators with Pauli-X terms based on the rows of the parity matrix of $C_2^\perp$, this is more difficult to see. However, because $\mathcal{H} = \mathcal{G}^T$ holds for the dual code, we can, alternatively, view these operators as being described by the columns of the generator matrix of C_2. We now realize that X operators based on the columns of the generator matrix of C_2 generate all codewords of C_2. This means that multiplying the states of Eq. (3.2) with operators generated by these columns only permutes the states in the superposition: this effectively permutes the terms $y \in C_2$ in the sum of the expression. Therefore, these operators leave the logical states invariant as well.

A similar argumentation can be used to realize that the logical X operators for the CSS codewords are associated with columns of the generator matrix $\mathcal{G}_1$ of C_1 that can not be described as a linear combination of the columns of the generator matrix $\mathcal{G}_2$ of C_2. For a classical code, the columns of the generator matrix can be interpreted as generator operators of the classical codewords. Since $C_2 \subset C_1$, it is always possible to write each column of $\mathcal{G}_2$ as a linear combination of the columns of $\mathcal{G}_1$. As described above, these columns take on the role of the stabilizer generators of the CSS codewords. However, since $\mathcal{G}_1$ has more columns than $\mathcal{G}_2$, some columns of $\mathcal{G}_1$ become generators of the logical operators for the CSS codewords. This is why the distance of the CSS codewords against Pauli-X errors is *at least* the distance of C_1. Next to that, for the CSS codewords, logical Z operators can be directly associated with rows of the parity matrix $\mathcal{H}_2$ of C_2 that cannot be expressed as linear combinations of the rows of the parity matrix $\mathcal{H}_1$ of C_1. Therefore, since these parity matrix rows are columns in the generator matrix of $C_2^\perp$, the distance of the CSS codewords against Pauli-Z errors is given by at least the distance against bit-flip errors in the classical code $C_2^\perp$.

In summary, we see that, for CSS codes, the stabilizer generators are split into a set of operators that only contain Pauli-Z operators and a set of operators that only contain Pauli-X operators. This is exactly how we can identify CSS codes: if is possible to (re)define the stabilizer and logical operators of the code into operators containing either only Pauli-Z or only Pauli-X operators, the concerning stabilizer code is a CSS code. In Secs. 3.3 and 3.4 below, we discuss specific examples of CSS codes.

3.2.5 Threshold

As discussed, the idea of an error-correction code is to automatically correct unwanted errors that appear on the qubits in our code. In practice, the operations required to detect and correct the errors are not perfect themselves as well—*i.e.*, they, themselves, might introduce errors. This means that, intuitively, the code's capability to correct errors depends on how noisy these error-correction operations are. Generally speaking, if the operations can be carried out with high fidelity, it is beneficial to increase the code space—*i.e.*, to increase the number of qubits N of the code while keeping the number of encoded states K constant. In that case, this leads to more successful error correction and a smaller *logical error rate*, since this generally increases the distance between the logical states, which allows the code to address errors of more elaborate size. On the other hand, increasing the code space gives rise to more complexity when it comes to choosing a suitable correction. Moreover, if the operations are substantially noisy, increasing the code space typically introduces more noise than the code can correct, leading to a higher logical error rate. This shows that there is a "tipping point", or threshold, for the quality of our operations:

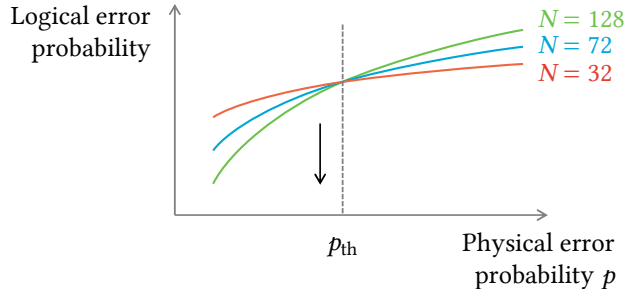


Figure 3.4: Schematic depiction of an error-correction threshold. Typically, an increase of a certain physical error probability p leads to an increase in the logical error probability. For low physical error probabilities, an error-correction code often deals better with noise when the number of data qubits N of the code is increased, whereas for high error probabilities, error-correction codes often introduce more logical errors. This behavior gives rise to a threshold value p_{th} for the concerning physical error rate: for $p < p_{\text{th}}$, increasing N decreases the probability of a logical error, whereas for $p > p_{\text{th}}$, increasing N leads to more logical errors.

keeping the physical error rate of the operations below this threshold means we can reach arbitrarily small logical error rates by selecting the corresponding size of the code space.

In a situation with multiple sources of noise—*e.g.*, gate noise, measurement noise, and memory decoherence—we typically do not have a single threshold, but rather a multi-dimensional landscape of threshold values for each noise source in play. We show a schematic representation of an error-correction threshold in Fig. 3.4. A specific example of a calculated threshold can be found in Fig. C.1 in App. C. For a specific implementation of an error-correction code, the exact value of the threshold depends on, among other aspects, the circuit chosen to carry out the operations, the type of noise introduced by the operations, and the error syndrome decoder used. As implied in Ch. 1, it is often beneficial to choose fault-tolerant implementations for measuring the code's stabilizers or performing logical operations—*e.g.*, by using operations that do not convert likely correctable errors into uncorrectable errors when propagating them through the applied circuits.

3.3 $\mathbb{Z}_2$ CHAIN COMPLEX AND HOMOLOGY

In this section, we introduce *chain complexes* and their associated *homologies* over the group $\mathbb{Z}_2$. We discuss the main idea in Sec. 3.3.1, show an example in three dimensions in Sec. 3.3.2, and introduce *cycles* in Sec. 3.3.3. The frameworks introduced here can be used to construct topological CSS codes. In these codes, cycles represent the stabilizer and logical operators. We use the basics of this section to construct two-dimensional topological codes in Sec. 3.4 and three-dimensional cluster states in Sec. 3.5. The main ideas presented in this section are derived from introductory material by Fujii [9] and Browne [10].

3.3.1 GENERAL IDEA

Homology is sometimes referred to as the study of *boundary* [10], and allows one to associate algebraic objects with topological spaces. Later in this chapter, we focus on stabilizer codes on a lattice and use the homology of the lattice to identify logical operators of these codes. Since identifying valid combinations of stabilizer and logical operators for

these codes requires dealing with Pauli operators that either commute or anti-commute, we are primarily interested in the homology of spaces with elements chosen from the group $\mathbb{Z}_2$. This group is isomorphic to the set of integers $\{0,1\}$ with addition modulo 2 as the group operation and 0 as the identity element.

A chain complex is a sequence of modules connected by boundary operators such that any two consecutive boundary maps are equal to the zero map. The boundary maps are (group) homomorphisms: group-structure preserving maps between objects of the same type—*i.e.*, between so-called *chains*. These maps tell us how to get from an element in one module to another in the next module. In our case, the modules are *vector spaces* C_i constructed over the field $\mathbb{Z}_2$, with the index i describing the *dimension* $i \in \{0,1,\ldots,D\}$ of the objects in the vector space C_i. Chains are the basic elements of homology. We call elements of a module C_i an i-chain: each i-chain is an i-dimensional object in the vector space C_i. The vector spaces C_i form a sequence $C_D \to C_{D-1} \to \cdots \to C_0$. The boundary map $\partial_i : C_i \mapsto C_{i-1}$ describes the boundary of an i-chain in the vector space one dimension lower—*i.e.*, it gives us the $(i-1)$-chain that forms the boundary of that i-chain.

The *dual complex* or *cochain complex* is an associated sequence $\overline{C}_D \to \overline{C}_{D-1} \to \cdots \to \overline{C}_0$, such that each $\overline{C}_i$ shares the same basis elements $\{(\mathbf{b}_i)_k = (\overline{\mathbf{b}}_{D-i})_k\}_k$ as C_{D-i}. Chains in the dual complex are called *dual chains* or *cochains*—with, to avoid confusion, the chains in the original complex sometimes being referred to as *primal* chains. If we use a matrix representation to describe the chain complex, we can use the relation between primal and dual vector spaces to find the relation between primal and dual boundary map operators. Given the mnth matrix element of the boundary operator $(\partial_i)_{mn}$ describing the map $(\mathbf{b}_i)_n \mapsto (\mathbf{b}_{i-1})_m$, the corresponding dual boundary describes $(\mathbf{b}_i)_n = (\overline{\mathbf{b}}_{D-i})_n \leftarrow (\mathbf{b}_{i-1})_m = (\overline{\mathbf{b}}_{D+1-i})_m$ in the opposite direction. Therefore, $(\partial_i)_{mn} = (\overline{\partial}_{D+1-i})_{nm}$ holds, or alternatively

$$\partial_i = \overline{\partial}^T_{D+1-i}. \tag{3.3}$$

3.3.2 Example in three dimensions

As a specific example, for $D = 3$, a chain complex may be defined on a three-dimensional lattice $\mathcal{L} = (Q,F,E,V)$ with cells (*i.e.*, volumes) $Q = \{\mathbf{q}_k\}_k$, faces $F = \{\mathbf{f}_k\}_k$, edges $E = \{\mathbf{e}_k\}_k$ and vertices $V = \{\mathbf{v}_k\}_k$ as basis elements of the 3-chains, 2-chains, 1-chains and 0-chains, respectively—*i.e.*, as basis elements of the vector spaces C_3, C_2, C_1 and C_0 respectively:

$$\begin{array}{ccccccc} C_3 & \xrightarrow{\partial_3} & C_2 & \xrightarrow{\partial_2} & C_1 & \xrightarrow{\partial_1} & C_0 \\ \{\mathbf{q}_k\}_k & & \{\mathbf{f}_k\}_k & & \{\mathbf{e}_k\}_k & & \{\mathbf{v}_k\}_k \end{array}$$

To construct the full i-chains, we can assign an integer 0 or 1 to each of the basis elements of C_i, with a 1 indicating this basis element is part of the chain. For example, a 1-chain, *i.e.*, a vector $\mathbf{c}_1 \in C_1$, can be expressed as a linear combination of edges $\mathbf{e}_k$:

$$\mathbf{c}_1 = \sum_k z_k \mathbf{e}_k \equiv \begin{bmatrix} z_0 & z_1 & \cdots \end{bmatrix}^T, \quad \text{where} \quad z_k \in \mathbb{Z}_2. \tag{3.4}$$

Here, we represent the vector representation relative to the basis $E = \{\mathbf{e}_k\}_k$. The coefficients $z_k \in \mathbb{Z}_2$ describe whether an edge is part of the chain or not. In some situations, primarily in Secs. 3.4.3 and 3.5.4, we conveniently denote the coefficients of $\mathbf{c}_1$ as $\mathbf{c}_1 \equiv \{z_k\}_k$ instead

of using the column vector notation of Eq. (3.4). Similar definitions hold for 0-chains $\mathbf{c}_0$, 2-chains $\mathbf{c}_2$, 3-chains $\mathbf{c}_3$, or i-chains $\mathbf{c}_i$ over vertices, faces, cells, and i-dimensional objects, respectively. A chain does not necessarily have to be a connected selection of i-dimensional basis objects. By definition of the vector space, all chains in a vector space form a group. The group operation is composed of the element-wise addition of two vectors modulo 2. The inner product $\mathbf{c}_i \cdot \mathbf{c}_i' \equiv \mathbf{c}_i^T \mathbf{c}_i'$ is computed as the dot product with addition modulo 2 applied to the pairwise multiplied coefficients. The inner product can be geometrically interpreted as a basis-independent parity measurement of the "overlap" of two i-chains.

For chain complexes defined on $\mathcal{L}$, boundary operators possess a straightforward geometric interpretation. A cell $\mathbf{q}_k$ is mapped to the faces $\{\mathbf{f}_j\}_j$ that enclose it, a face $\mathbf{f}_k$ is mapped to the edges $\{\mathbf{e}_j\}_j$ that constitute its boundary, and an edge $\mathbf{e}_k$ is mapped to the vertices $\{\mathbf{v}_j\}_j$ on its ends. The boundary map of full i-chains can be found by taking the linear combination of boundary maps on the individual basis elements that are part of the i-chain. That is because, in a modulo-2 context, boundary operators $\partial_i : C_i \mapsto C_{i-1}$ are linear maps that maintain vector addition:

$$\partial_i \left(\mathbf{c}_i + \mathbf{c}_i' \right) = \partial_i \mathbf{c}_i + \partial_i \mathbf{c}_i'. \tag{3.5}$$

As mentioned above, in the definition of a chain complex, two consecutive boundary maps result in the zero map

$$\partial_{i-1} \partial_i = 0, \tag{3.6}$$

independent of the choice of $\mathbf{c}_i$.

For the dual chain complex associated with the chain complex of this three-dimensional lattice $\mathcal{L}$, the basis elements are given by dual cells $\overline{Q} = \{\overline{\mathbf{q}}_k\}_k$, dual faces $\overline{F} = \{\overline{\mathbf{f}}_k\}_k$, dual edges $\overline{E} = \{\overline{\mathbf{e}}_k\}_k$ and dual vertices $\overline{V} = \{\overline{\mathbf{v}}_k\}_k$. Here we have that the primal vertices correspond to dual cells ($V \to \overline{Q}$), primal edges correspond to dual faces ($E \to \overline{F}$), primal faces correspond to dual edges ($F \to \overline{E}$), and primal cells correspond to dual vertices ($Q \to \overline{V}$), such that:

$$\begin{array}{ccccccc} \overline{C}_0 & \xleftarrow[\overline{\partial}_1]{} & \overline{C}_1 & \xleftarrow[\overline{\partial}_2]{} & \overline{C}_2 & \xleftarrow[\overline{\partial}_3]{} & \overline{C}_3 \\ \{\overline{\mathbf{v}}_k\}_k & & \{\overline{\mathbf{e}}_k\}_k & & \{\overline{\mathbf{f}}_k\}_k & & \{\overline{\mathbf{q}}_k\}_k \end{array}$$

We see that the associated dual boundaries $\overline{\partial}_i : \overline{C}_i \mapsto \overline{C}_{i-1}$ map objects in $\overline{C}_i$ to their boundary objects in $\overline{C}_{i-1}$. The dual boundaries on $\overline{\mathcal{L}} = (V, E, F, Q)$ exhibit comparable behavior to their primary counterparts. A dual cell (basis element of 3-cochains) $\overline{\mathbf{q}}_k = \mathbf{v}_k$ is mapped to the faces (basis elements of 2-cochains) $\{\overline{\mathbf{f}}_j\}_j$ that enclose it, which correspond to the primal edges $\{\mathbf{e}_j\}_j$ *incident* to $\mathbf{v}_k$. A dual edge (basis element of 1-cochains) $\overline{\mathbf{e}}_k = \mathbf{f}_k$ is mapped to the endpoints $\{\overline{\mathbf{v}}_j\}_j$, which are the cells (basis elements of 0-cochains) $\{\mathbf{q}_j\}_j$ adjacent to $\mathbf{f}_k$.

3.3.3 Cycles and homology groups

In Eq. (3.6) we see that, by definition, applying the boundary map twice in succession always leads to the zero map. We can call any i-chain that is the boundary of an $(i+1)$-chain, *i.e.*, $\mathbf{c}_i = \partial_{i+1} \mathbf{c}_{i+1}$, an *$i$-boundary*. Because Eq. (3.6) holds, we can now understand that any i-boundary does not have a boundary itself, *i.e.*, $\partial_i \mathbf{c}_i = \partial_i \partial_{i+1} \mathbf{c}_{i+1} = 0$. We define an *$i$-cycle* $\mathbf{c}_i$ as such a chain without a boundary: $\partial_i \mathbf{c}_i = 0$, or, by definition, $\mathbf{c}_i \in \ker \partial_i$. Any i-boundary is also automatically an i-cycle, and we can specifically refer to an i-boundary as a *trivial*

i-cycle. An example of a trivial cycle for our three-dimensional lattice $\mathcal{L}$ from Sec. 3.3.2 is a loop $\mathbf{c}_1 = \partial_2 \mathbf{f}_k$ around a specific face $\mathbf{f}_k$. We can understand that this is indeed a cycle since its boundary $\partial_1 \partial_2 \mathbf{f}_k = 0$.

Even though all i-boundaries are automatically i-cycles, the reverse is not true: it is possible to have i-cycles that are not i-boundaries. That is, there are also *non-trivial* cycles. These are cycles $\mathbf{c}_i$ that can not be expressed as the boundary of a chain $\mathbf{c}_{i+1}$ one dimension higher—*i.e.*, $\partial_i \mathbf{c}_i = 0$, but there exists no $\mathbf{c}_{i+1} \in C_{i+1}$ that satisfies $\mathbf{c}_i = \partial_{i+1} \mathbf{c}_{i+1}$.

By definition, the set of i-cycles $\ker \partial_i$ forms a group. On top of that, the set of trivial i-cycles (*i.e.*, the set of i-boundaries) is also a group: we refer to this group as $\operatorname{Im} \partial_{i+1}$. These observations provide some intuition for the fact that trivial cycles form a normal subgroup of all cycles. For each vector space C_i, the quotient group

$$\mathbb{H}_i \equiv \ker \partial_i / \operatorname{Im} \partial_{i+1} \tag{3.7}$$

can now be used to split up the group of all cycles into *equivalence classes*. Two chains $\mathbf{c}_i$ and $\mathbf{c}_i'$ are part of the same equivalence class whenever $\mathbf{c}_i = \mathbf{c}_i' + \partial_{i+1} \mathbf{c}_{i+1}$ holds. This can be understood by realizing that the full group of cycles $\ker \partial_i$ is split up into multiple cosets: within one coset, the cycles differ from each other by a trivial cycle, but to go from a cycle in one coset to a cycle in another coset, we need to add a non-trivial cycle. The groups $\mathbb{H}_i$ are called *homology groups*. Later, in Secs. 3.4.1 and 3.5.1, we define error-correction codes over a lattice by using trivial cycles as code stabilizers and non-trivial cycles as logical operators. The size of the homology group then corresponds to the number of logical qubits in the code.

Dual boundaries share a similar concept as homology, called *cohomology*. Cohomology groups are constructed following the same methodology as homology groups, wherein cycles $\ker \partial_i$ and trivial cycles $\operatorname{Im} \partial_{i+1}$ are replaced by their dualized counterparts $\ker \overline{\partial}_i$ and $\operatorname{Im} \overline{\partial}_{i+1}$, such that

$$\overline{\mathbb{H}}_i \equiv \ker \overline{\partial}_i / \operatorname{Im} \overline{\partial}_{i+1}. \tag{3.8}$$

3.4 Two-dimensional topological codes

In this section, we analyze how the concepts of Sec. 3.3 can be used to construct topological error-correction codes on a two-dimensional surface. In Secs. 3.4.1 and 3.4.2, we discuss how the stabilizer and logical operations of the codes are defined and identified. In Sec. 3.4.3, we elaborate on the error-correction process without measurement errors. For the surface code, introduced in Sec. 3.4.4 and the most important example of codes of this type, we discuss how one can realize logical qubits in Secs. 3.4.5 and 3.4.6, and how universal quantum computing can be achieved in Sec. 3.4.7. Finally, in Sec. 3.4.8 we return to the error-correction process—but this time for a situation *with* measurement errors. Considering two-dimensional topological codes with measurement errors paves the way to the cluster state error-correction channels of Sec. 3.5.

3.4.1 Construction

Using two-dimensional chain complexes $C_2 \to C_1 \to C_0$, we can construct a topological stabilizer code on a two-dimensional space $\mathcal{L}'$, by following the following steps:

1. Place qubits on all basis elements of C_1. That is, on each edge $\mathbf{e}_k$.

2. Define *primal* stabilizers as Pauli-Z strings over trivial *primal* cycles $\partial_2 \mathbf{c}_2$.

3. Define *dual* stabilizers as Pauli-X strings over trivial *dual* cycles $\overline{\partial}_2 \overline{\mathbf{c}}_2$.

4. Define primal (Z) and dual (X) logical operators as cosets of the homology group $\mathbb{H}_1$ and cohomology group $\overline{\mathbb{H}}_1$, respectively.

Since $\partial_2 \mathbf{c}_2$ may be decomposed into a sum of cycles around faces as $\partial_2(\sum_k z_k \mathbf{f}_k) = \sum_k z_k(\partial_2 \mathbf{f}_k)$, we can simply use the set of all faces $\{\mathbf{f}_k\}_k$ to generate all primal stabilizers via $g_k = Z(\partial_2 \mathbf{f}_k)$. Here, the notation $Z(\mathbf{c}_1)$ means we apply Pauli-Z operators on qubits on the edges that are part of the chain $\mathbf{c}_1$. In general, Pauli operators on the qubits of an i-chain $\mathbf{c}_i$ are denoted as $P(\mathbf{c}_i) \equiv \bigotimes_k P^{z_k}$, where $P \in \{X, Y, Z\}$. Qubits with $z_k = 0$ carry identity, and $z_k = 1$ carry P. Similarly to primal stabilizers, dual stabilizers over the dual cycles $\overline{\partial}_2(\sum_k \overline{z}_k \mathbf{v}_k)$ can be generated with all vertices $\{\mathbf{v}_k\}_k$ via $\overline{g}_k = X(\overline{\partial}_2 \mathbf{v}_k)$.

3.4.2 Commutation and independence requirements

In order to have a valid stabilizer code, all stabilizer operators must commute. For a topological code constructed as above, we can show that this is the case if $\mathbf{c}_1 \cdot \overline{\mathbf{c}}_1 = 0$ holds:

$$Z(\mathbf{c}_1) X(\overline{\mathbf{c}}_1) = (-1)^{\mathbf{c}_1 \cdot \overline{\mathbf{c}}_1} X(\overline{\mathbf{c}}_1) Z(\mathbf{c}_1). \tag{3.9}$$

Since the stabilizers have $\mathbf{c}_1 = \partial_2 \mathbf{c}_2$ and $\overline{\mathbf{c}}_1 = \overline{\partial}_2 \overline{\mathbf{c}}_2$, we have

$$\overline{\mathbf{c}}_1 \cdot \mathbf{c}_1 = \overline{\partial}_2 \overline{\mathbf{c}}_2 \cdot \partial_2 \mathbf{c}_2 = (\overline{\partial}_2 \overline{\mathbf{c}}_2)^T \partial_2 \mathbf{c}_2 = \overline{\mathbf{c}}_2^T \overline{\partial}_2^T \partial_2 \mathbf{c}_2 = \overline{\mathbf{c}}_2 \cdot \overline{\partial}_2^T \partial_2 \mathbf{c}_2 = 0, \tag{3.10}$$

where the last part holds because Eqs. (3.3) and (3.6) show us that $\partial_1 \partial_2 = \overline{\partial}_2^T \partial_2 = 0$ holds.

The required commutability of logical operators with stabilizers can be shown by realizing that a logical Z operator $\overline{Z}_j = Z(\mathbf{c}_1)$ only commutes with dual stabilizers $X(\overline{\partial}_2 \overline{\mathbf{c}}_2)$ whenever their inner product $\mathbf{c}_1 \cdot \overline{\partial}_2 \overline{\mathbf{c}}_2 = 0$. At this point, we note that the following always holds:

$$\mathbf{c}_i \cdot \overline{\partial}_{D+1-i} \overline{\mathbf{c}}_{D+1-i} = \mathbf{c}_i \cdot \partial_i^T \overline{\mathbf{c}}_{D+1-i} = \partial_i \mathbf{c}_i \cdot \overline{\mathbf{c}}_{D+1-i}. \tag{3.11}$$

Using $D = 2$ and $i = 1$ in Eq. (3.11), the requirement $\mathbf{c}_1 \cdot \overline{\partial}_2 \overline{\mathbf{c}}_2 = 0$ can be rewritten to $\partial_1 \mathbf{c}_1 \cdot \overline{\mathbf{c}}_2 = 0$. Since $\overline{\mathbf{c}}_2$ is arbitrary—*i.e.*, the coboundary of each 2-cochain $\overline{\mathbf{c}}_2$ contains a stabilizer operator—this equality only holds whenever $\partial_1 \mathbf{c}_1 = 0$ holds. This indicates that logical 1-chains, *i.e.*, the logical Z operators, must be cycles.

On top of that, logical $Z(\mathbf{c}_1)$ operators have to be independent of all primal Z stabilizers. This means that the 1-cycle $\mathbf{c}_1$ containing the logical operator cannot be a trivial cycle, and must be part of a coset of the homology group $\mathbb{H}_1$ (Eq. (3.7)) as a non-trivial cycle. The equivalence relation between two 1-chains $\mathbf{c}_1$ and $\mathbf{c}_1'$ in this coset, $\mathbf{c}_1 = \mathbf{c}_1' + \partial_2 \mathbf{c}_2$, means that the action of the corresponding logical operators $Z(\mathbf{c}_1)$ and $Z(\mathbf{c}_1')$ is the same, since there is a primal stabilizer $Z(\partial_2 \mathbf{c}_2)$ (logical identity) that relates the two.

In the same way, a logical X operator $\overline{X}_j = X(\overline{\mathbf{c}}_1)$ defined on a logical 1-cochain $\overline{\mathbf{c}}_1$ only commutes with primal Z-stabilizers whenever $\overline{\partial}_1 \overline{\mathbf{c}}_1 = 0$ holds. This cochain can also not be a trivial cycle. The dual relationship between $\mathbb{H}_1$ and $\overline{\mathbb{H}}_1$ ensures that logical operators appear in pairs $(\overline{X}_j, \overline{Z}_j)$ that mutually anti-commute [11].

3.4.3 Error correction without measurement errors

For topological codes there exists a straightforward correction procedure against error sets $\{\varepsilon_k\}_k$ composed of Pauli operators applied simultaneously across many qubits—*i.e.*, *Pauli strings*. An arbitrary string can be described as the product of two chains $\mathbf{c}_1$ and $\overline{\mathbf{c}}_1$ as $Z(\mathbf{c}_1)X(\overline{\mathbf{c}}_1)$. To collect the error syndrome, and detect errors, we collect measurement outcomes $\mathbf{m} = \{m_k\}_k$ and $\overline{\mathbf{m}} = \{\overline{m}_k\}_k$ by measuring all primal and dual stabilizer generators $g_k = Z(\partial_2 \mathbf{f}_k)$ and $\overline{g}_k = X(\overline{\partial}_2 \overline{\mathbf{f}}_k)$. For notational convenience, we define $\mathbf{m}$ and $\overline{\mathbf{m}}$ over the field $\mathbb{Z}_2$ by mapping each of their elements m_k and $\overline{m}_k$ with $\{+1, -1\} \mapsto \{0, 1\}$. For the moment, we assume all generators can be measured without measurement errors.

We define the error syndrome as the set of all primal and dual measurement outcomes $\mathbf{m}$ and $\overline{\mathbf{m}}$. Given that both the error and stabilizer generators are strings of Pauli operators, one can fully describe the measurement outcomes by deploying (anti-)commutation relations of Pauli operators. The primal error syndrome chain forms the *coboundary* of the dual errors $X(\overline{\mathbf{c}}_1)$: $\mathbf{m} = \overline{\partial}_1 \overline{\mathbf{c}}_1$. This means we can consider $\mathbf{m}$ as a 0-cochain (or, alternatively, a 2-chain). It also means the *syndrome graph* constructed with the primal stabilizers g_k is defined on the dual of the lattice. On the other hand, the syndrome constructed with the dual stabilizers $\overline{g}_k$ is defined on the primal lattice: the dual syndrome $\overline{\mathbf{m}} = \partial_1 \mathbf{c}_1$ is a 0-chain (*i.e.*, a 2-cochain) that forms the boundary of the primal errors $Z(\mathbf{c}_1)$. We see that with this construction we find Pauli-X errors by measuring primal stabilizers g_k and we find Pauli-Z errors by measuring dual stabilizers $\overline{g}_k$.

To correct the detected errors based on the obtained error syndrome $\{\mathbf{m}, \overline{\mathbf{m}}\}$, we determine recovery chains $\mathbf{r}_1$ and $\overline{\mathbf{r}}_1$. The correction can be considered successful if the sums of error and recovery chains $\mathbf{c}_1 + \mathbf{r}_1$ and $\overline{\mathbf{c}}_1 + \overline{\mathbf{r}}_1$ are trivial cycles: $\mathbf{c}_1 + \mathbf{r}_1 = \partial_2 \mathbf{c}_2$ and $\overline{\mathbf{c}}_1 + \overline{\mathbf{r}}_i = \overline{\partial}_2 \overline{\mathbf{c}}_2$. If this is the case, the corresponding net operators $Z(\partial_2 \mathbf{c}_2)$ and $X(\overline{\partial}_2 \overline{\mathbf{c}}_2)$ are stabilizer operators themselves. On the other hand, if we end up with $\mathbf{c}_1 + \mathbf{r}_1 \neq \partial_2 \mathbf{c}_2$ or $\overline{\mathbf{c}}_1 + \overline{\mathbf{r}}_1 \neq \overline{\partial}_2 \overline{\mathbf{c}}_2$, the corresponding net operator is a non-trivial cycle and corresponds to a logical operator. Occurrences of the recovery procedure that result in a non-trivial cycle are referred to as a logical error or a *logical failure*, whereas instances leading to a trivial cycle lead to *logical success*. An example of the error-correction process for the surface code (introduced in Sec. 3.4.4) can be found in Fig. 3.5.

Topological codes in the form of Sec. 3.4.1 can correct more general noise maps if we take into account linear combinations of the allowed Pauli errors. However, these codes are not able to also correct against errors in the determination of the error syndrome itself: they only work if the stabilizer generators $\{g_k\}_k$ and $\{\overline{g}_k\}_k$ can be perfectly read out, and $\mathbf{m}$ or $\overline{\mathbf{m}}$ do not flip with respect to their actual values. In Sec. 3.4.8, we consider how the scheme can be adapted to also deal with errors in measuring $\{g_k\}_k$ and $\{\overline{g}_k\}_k$.

3.4.4 Surface code

The surface code is a widely-used example of a two-dimensional topological code. For this code, we typically work with a square lattice. The primal Z string stabilizer generators are described by boundaries of all faces $\mathbf{f}_k$ of the lattice, and are often referred to as *plaquette* or *face* stabilizers. The dual X string stabilizer generators are associated with the coboundaries of all vertices $\mathbf{v}_k$ of the lattice and are often referred to as *star* or *vertex* stabilizers.

In a situation with *periodic* boundary conditions, the two-dimensional lattice can be placed on a *torus*. This version of the surface code is often called the *toric surface code*, or

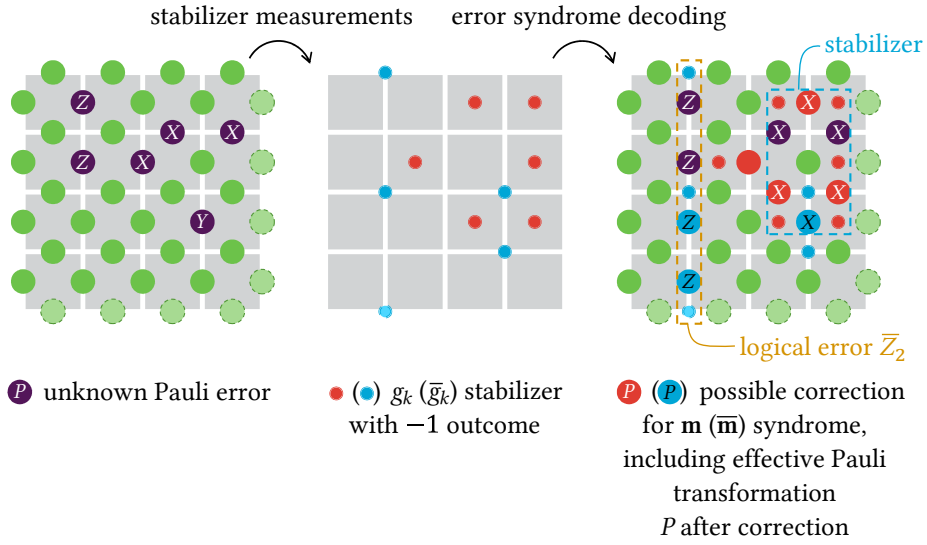


Figure 3.5: Schematic overview of the error-correction process for the toric surface code—introduced in Sec. 3.4.4 and Fig. 3.6. Unknown Pauli errors that appear on the physical data qubits can be tracked and corrected by measuring the stabilizer generators and decoding the resulting error syndrome $\{\mathbf{m}, \overline{\mathbf{m}}\}$. The unknown errors and the correction can still lead to a logical error—in this example, we unknowingly apply the logical operator $\overline{Z}_2$ of Fig. 3.6, as well as a stabilizer operator that leaves the code invariant.

simply the *toric code*. We depict the toric surface code schematically in Fig. 3.6. For the toric code, the weight of all stabilizer generators is four—*i.e.*, we have to measure stabilizer operators consisting of four non-identity Pauli operators to obtain the error syndrome. The toric code encodes two logical states. This can be understood by realizing that the rank of the (co)homology groups $\mathbb{H}_1$ and $\overline{\mathbb{H}}_1$ is equal to two for this lattice with periodic boundary conditions. For both the primal and the dual lattice, non-trivial 1-(co)cycles can be identified as 1-(co)chains going from top to bottom and from left to right in the lattice. These chains have no boundary because, due to the periodic boundary conditions, they enter the lattice at the opposite side where they disappear: they "wrap" around the holes of the torus. Alternatively, we can count the number of independent stabilizer generators to realize that the toric code should have logical qubits. This is because multiplying all plaquette or multiplying all star generators gives rise to the identity operator. Therefore, to have an independent set of generators, we need to leave out one plaquette and one star operator. Because each generator defines one-fourth of a qubit, and each qubit is part of four generators, this means that the code space has two free qubit degrees of freedom—these correspond to two encoded logical qubits.

This changes if we consider a square lattice that does *not* have periodic boundary conditions. The weight of the stabilizer generators on the border is now three or, for the generators in the corner, two. This version of the surface code, which is generally referred to as the *planar surface code*, has only one set of logical operators. This is because there is now only one non-trivial cycle on both the primal and dual lattice: a non-trivial 1-cycle terminating at the top and bottom border of the primal lattice and a non-trivial 1-cocycle

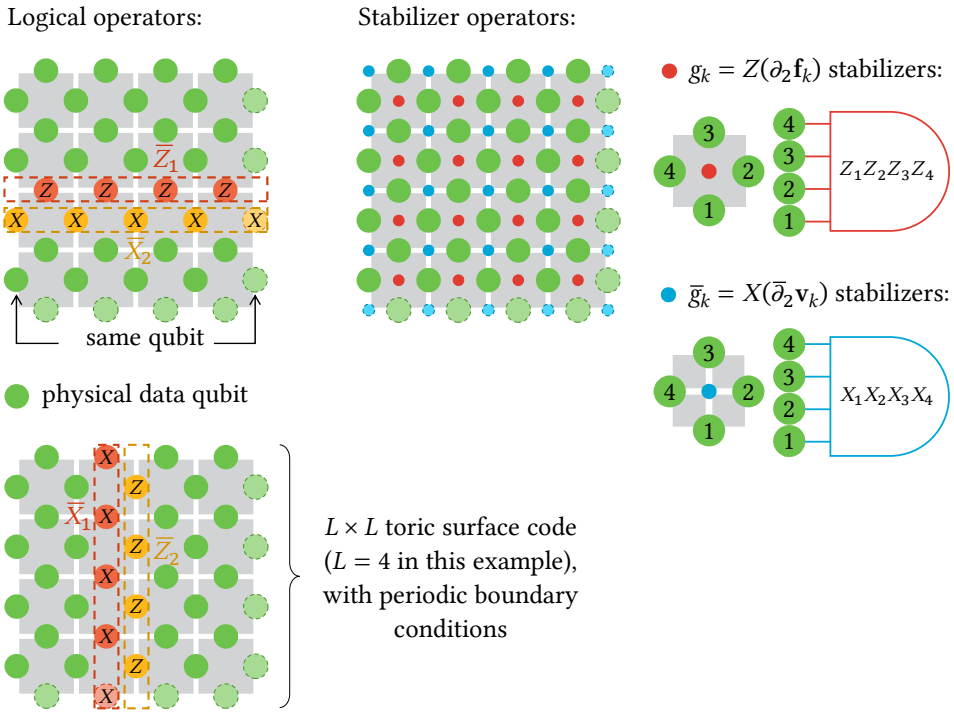


Figure 3.6: Schematic overview of the toric surface code. This code encodes two logical qubits with multi-qubit logical operators $\overline{X}_{1,2}$ and $\overline{Z}_{1,2}$: the first qubit has $\overline{X}_1$ ($\overline{Z}_1$) as its logical X (Z) operator, whereas the second qubit has $\overline{X}_2$ ($\overline{Z}_2$) as its logical X (Z) operator. The code's stabilizer generators are four-qubit operators surrounding every face ($g_k = Z(\partial_2 \mathbf{f}_k)$) and every vertex ($\overline{g}_k = X(\overline{\partial}_2 \mathbf{v}_k)$) of the lattice. A schematic depiction of the error-correction process for the toric code can be found in Fig. 3.5.

terminating at the left and right border for the dual lattice—or the other way around. We explore this version in more detail in Sec. 3.4.5.

3.4.5 Smooth and rough lattice boundaries

To better understand how non-trivial cycles terminate at the boundary of a lattice, we have to introduce the concept of *smooth* and *rough* lattice boundaries. In the following, we often explicitly refer to "lattice boundaries" to describe the border of the lattice, to avoid confusion with the boundary map and the i-boundary chains of the chain complex. The toric surface code, or any other topological code with periodic boundary conditions, contains a *closed* surface. As soon as we open up the surface—*i.e.*, as soon as we introduce *holes* in the surface—we introduce either a smooth or a rough lattice boundary.

Normally, *i.e.*, in the bulk of a lattice, an edge is adjacent to two vertices—*i.e.*, the boundary map of every single edge is a set of two vertices at each endpoint of the edge. For the edges at a smooth lattice boundary, this is still the case. The only difference here is that the vertices on the lattice boundary are connected to a smaller number of edges than in the bulk, as the edges that cross the boundary are no longer there. For a rough lattice boundary, on the other hand, this is different. Here, the vertices on the boundary stay adjacent to the same number of edges, but the edges at the boundary are only connected to one vertex: they are "dangling", as the other endpoint of these edges is beyond the boundary. We see that, when creating a lattice boundary, we have the option to either create it by removing edges (resulting in a smooth boundary) or by removing vertices (resulting in a rough boundary). If we create a smooth boundary on the primal lattices, this automatically leads to a rough boundary on the dual lattice—and vice versa.

We can now also understand better how a 1-chain can be boundaryless (*i.e.*, a cycle) if it is a connected string of edges going from one rough lattice boundary to another rough lattice boundary. This is because the edges at the endpoints of this chain have only one vertex in their boundary map. Together with the boundary maps of the other edges in the string, the full boundary map disappears. It is important to mention that this only leads to a non-trivial cycle if there is a smooth lattice boundary in between the two rough boundaries at which the chain terminates: the chain has to begin and end at *different* rough boundaries. If this is not the case, and the chain begins and ends at the *same* rough boundary, the string can be created from stabilizer operators—*i.e.*, the chain is a trivial cycle. Therefore, for the planar surface code, there are only logical states encoded if we have at least four alternating regions with smooth and rough boundaries around the boundary of the lattice.

3.4.6 Distance and holes

Above, we defined the distance of the code as the minimum weight of its logical operators. If the surface code is defined on an $L \times L$ square lattice, the minimum weight of each of its logical Z and X operators is always a string that goes straight from top to bottom, or from left to right. Therefore, for such a lattice, the distance of the surface code is L.

This changes if we punch holes in the stabilizer structure of the code that are not lattice boundaries. These holes can be created by "switching off" a (small) set of connected stabilizer generators. This leads to a hole-shaped border in the middle of the surface. Because this opens up one qubit degree of freedom, introducing such a hole realizes an additional logical qubit. The hole can either have a smooth boundary on the primal lattice

or a rough boundary on the primal lattice. In the case of a hole with a smooth boundary on the primal lattice, the introduction of the new logical state can be understood by realizing that there is now a new non-trivial 1-cycle that "wraps around" the hole and can not be constructed from the remaining primal stabilizers. On the dual lattice, there are additional non-trivial 1-cocycles that terminate at the rough boundary of the hole. In the case of a hole with a rough boundary on the primal lattice, the roles of the primal and dual lattice are reversed. The distance of such a surface with a hole is determined by the weight of the smallest non-trivial cycle: either the smallest weight of the cycles terminating at two rough boundaries or the minimum weight of the non-trivial cycle wrapped around the hole (*i.e.*, the *circumference* of the hole). It is possible to create more holes and introduce more logical qubits. In the numerical simulations presented in this thesis, we do not consider this type of logical qubit introduced by opening up holes in the lattice.

3.4.7 Universal quantum computing

As discussed in Sec. 2.3.3, a universal set of logical gates requires more operations than just the logical Z and logical X gates. How other logical gates are accomplished depends on the specific realization of logical qubits. The two main implementations for defining qubits with the surface code are either as multiple lattice patches—since each planar surface code with lattice boundaries contains one logical qubit—or as multiple (combinations of) holes in a single lattice—as discussed in Sec. 3.4.6. Below, we discuss the main approaches to realizing universal computation with these logical structures. We start with the concept of *transversality*: the most straightforward way of carrying out logical operations in error-correction codes. On top of that, we consider the injection of *magic states* in the code, which is another approach to implementing logical operations.

Transversal gates

In general, a logical gate is an operation that transforms the state of a logical qubit in the same way as the "normal" version of that gate transforms a non-encoded qubit. Typically, the most efficient way to implement a logical gate is as a so-called *transversal gate*. A logical gate on a single logical qubit is said to be transversal if its implementation consists of a tensor product of single-qubit operations—a tensor product that also preserves the code space. This is beneficial, as it guarantees that local errors stay local and do not spread out. We already saw examples of transversal logical gates as the logical Z and logical X gates of topological codes. For a logical gate on two qubits, transversality allows the use of two-qubit gates between physical qubits of both code spaces. For two logical qubits that are part of identical copies of a CSS code, a transversal CX gate can be achieved by applying individual CX gates between all physical qubits of the two codes. On top of that, for a *self-dual* CSS code, a transversal Hadamard gate can be carried out by applying a single-qubit Hadamard to all qubits of the code. A self-dual CSS code is a CSS code in which for each X-type stabilizer there is a Z-type stabilizer on the same set of qubits as the X-type stabilizer and each logical X operator is paired and anti-commutes with a logical Z operator that operates on the same qubits. Since the surface code is not self-dual and two encoded qubits do not necessarily have to be identical, other methods for carrying out transversal CX and H gates are developed—we discuss them below for the two main surface code implementations.

Logical operations for main surface code implementations

Let us first consider the implementation where logical qubits are different holes in the surface of a single lattice. For this configuration, we typically combine two holes with a single type of boundary—*i.e.*, either a rough or a smooth primal boundary—as a single qubit. This configuration is discussed, *e.g.*, in Refs. [10, 12]. Holes can be grown, shrunk, and moved around by switching stabilizer operators on and off, and adjusting the stabilizer operators on the boundary of the hole accordingly—*i.e.*, with *code deformation*. Depending on the type of boundary, the qubit's logical X and logical Z operators correspond to the circumference of one of the two holes and the multi-qubit operator that terminates by going from one hole to the other. It can be shown [10, 12] that, for one combined hole qubit with rough boundaries and another combined hole qubit with smooth boundaries, a logical CX gate can be achieved by moving one of the holes through the space in between the two holes of the other qubit: a process known as *braiding*. On top of that, we can use the one-bit teleportation scheme of Fig. 2.8a to teleport the logical state of one type of qubit to another type—*i.e.*, a qubit with rough boundaries to a qubit with smooth boundaries and vice versa—ensuring we can always perform a CX gate between any two qubits.

For logical qubits defined as planar lattices, a logical CX operation can be achieved with *lattice surgery* [13]. This comprises merging the lattices of two qubits into a single planar qubit, as well as splitting a single lattice qubit into two new planar qubits. If merging takes place between two rough (smooth) boundaries, it is equivalent to measuring the logical $\overline{X}_1\overline{X}_2$ ($\overline{Z}_1\overline{Z}_2$) operator of the two planar qubits involved, where $\overline{X}_j$ ($\overline{Z}_j$) is the logical X (logical Z) operator of the jth logical qubit. In this implementation, a logical CX operation can be achieved with a series of merging and splitting operations on three planar qubits—*i.e.*, the control qubit, the target qubit, and an ancillary planar qubit.

For both implementations, the logical Hadamard gate can be achieved in a transversal "spirit"—*i.e.*, applying H on the individual qubits of our lattice almost produces a logical Hadamard on the logical qubits. More specifically, this operation transforms all Z-type stabilizers to X-type stabilizers and vice versa—*i.e.*, Z-type stabilizers are now associated with vertices of the lattice and X-type stabilizers correspond to the faces of the lattice. It also transforms each logical Z operator into a logical X operator that terminates at different lattice boundaries compared to the logical X operator before the operation—and vice versa for each logical X operator. The standard surface code stabilizer structure can be reestablished by shifting the qubits in the lattice by half a unit cell in both the horizontal and vertical direction [10]. This operation, however, also transforms all smooth boundaries into rough boundaries and vice versa. This makes it understandable that the new logical X and logical Z operators terminate at different lattice boundaries after the transformation. If required, the original boundaries can be restored with code deformation—this also results in the logical X and logical Z operators terminating at the same lattice boundary as before the operation. Alternatively, specifically for a logical qubit represented by a (combination of) hole(s) in the lattice, a change in the type of its lattice boundary can be undone by teleporting the state onto a different logical qubit with the other boundary type [10]. If the lattice has multiple hole qubits, a logical Hadamard transformation can be achieved on an isolated qubit by switching off stabilizers around the qubit and switching them back on after the full transformation [12].

Logical operations via (magic) state injection

It is clear that, for both surface code qubit implementations, we can carry out a logical Hadamard and a logical CX gate in an efficient manner. As discussed in Sec. 2.3.3, this means that implementing a logical T gate gives rise to a universal gate set. Unfortunately, introducing the T gate is not straightforward, and can not be done transversally. This is a direct consequence of the *Eastin-Knill theorem* [14], which dictates that no quantum error-correction code can perform all gates required for a universal logical gate set transversally. Luckily, it is possible to indirectly implement a gate missing from the universal gate set by applying a transversal operation between the main code and an encoded version of a specific state [10, 15, 16]. Fig. 3.7a shows how this concept can be used to implement a $\mathcal{U}_Z$ gate [10]. This circuit is a variation on the most general one-bit teleportation scheme of Fig. 2.8a. Because $\mathcal{U}_Z$ is a Pauli-Z rotation that—when acting on the control qubit—commutes with the CX gate, we can construct this circuit from Fig. 2.8a. To realize that this works, suppose we multiply the output of Fig. 2.8a with $\mathcal{U}_Z$. Because this gate commutes with the CX gate, we can move it to the left—and instead multiply $|+\rangle$ with this gate—for the $m = +1$ scenario. For the $m = -1$ scenario, we have to adjust the correction accordingly. Converting the correction to $\mathcal{U}_Z X \mathcal{U}_Z^\dagger$ makes sure that $\mathcal{U}_Z$ is always, effectively, applied after the correction (since the $\mathcal{U}_Z$ operation before the correction cancels against the $\mathcal{U}_Z^\dagger$ part of the correction operator).

The T gate is a specific example of the more general $\mathcal{U}_Z$ rotation. To carry out the circuit of Fig. 3.7a, it is important that we can logically carry out the correction operation $\mathcal{U}_Z X \mathcal{U}_Z^\dagger$ in our error-correction code. For the T gate, this requires the implementation of logical Pauli-Y and S gates, as $TXT^\dagger = e^{-i\pi/4} YS$ holds [10]. Alternatively, the logical T gate can be performed with the circuit in Fig. 3.7b, which, among other steps, requires a joint ZZ measurement on the two logical qubits and also a possible correction with a logical S gate [17]. Unfortunately, for the main surface code implementations, one needs a similar indirect method to carry out a logical S gate—*i.e.*, by utilizing an encoded $S|+\rangle$ state. Using the approach of Fig. 3.7a, this requires correction with a logical Pauli-Y gate, as $SXS^\dagger = Y$ holds. A logical S gate can also be carried out with the circuit depicted in Fig. 3.7c. This circuit exploits the commutation relation between CX and S: similar as how the relation $\mathrm{C}X_{i,j}\, \mathbb{I}_i Z_j = Z_i Z_j\, \mathrm{C}X_{i,j}$ describes what happens when we commute a Pauli-Z gate through the target of a CX gate, the relation $\mathrm{C}X_{i,j}\, \mathbb{I}_i S_j = S_i S_j\, \mathrm{C}Z_{i,j}\, \mathrm{C}X_{i,j}$ describes commuting an S gate through the target of a CX gate.

Finally, one needs to prepare the ancillary logical states $\mathcal{U}_Z|+\rangle$ to carry out these indirect logical gates—*i.e.*, the states $T|+\rangle$ and $S|+\rangle = |+i\rangle$ for the logical gates discussed above. Since these states allow one to implement operations that do not have a transversal representation, these states are often referred to as magic states. Preparing logical versions of magic states can typically not be done efficiently and requires, *e.g.*, initializing the state on a single qubit (or a small number of qubits) and increasing the distance of the state to a full logical qubit [12]. This procedure is often referred to as *magic state injection*. A single-qubit ancillary magic state $H|\psi\rangle$ can also be injected with, *e.g.*, the one-bit teleportation scheme of Fig. 2.8b. For $\overline{Z}_j = P_1 \otimes \cdots \otimes P_N$ the logical Z operator of the logical qubit, this requires initializing the logical qubit in $|+\rangle_\mathrm{L}$, applying a controlled-(P_i) gate between $H|\psi\rangle$ and qubit i of the logical qubit for all $i \in \{1, 2, \ldots, N\}$, and measuring the ancillary state in the Pauli-X basis. Here, a -1 measurement outcome requires a correction with $\overline{X}_j$ on

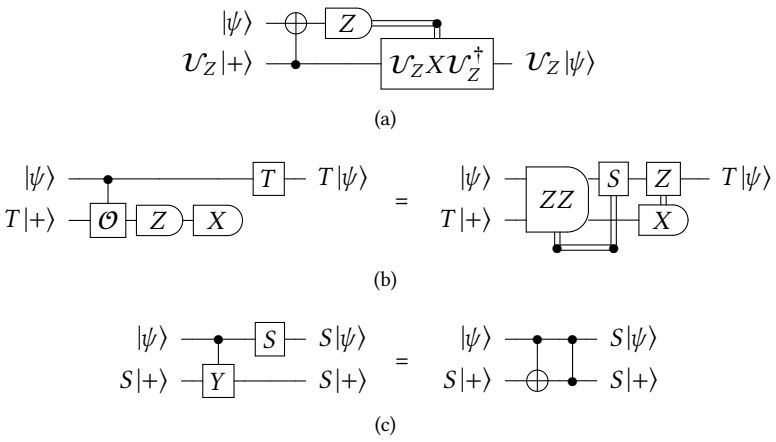


Figure 3.7: (a) Quantum circuit for implementing the (logical) gate $\mathcal{U}_Z$ on a general (logical) state $|\psi\rangle$ via an ancillary state $\mathcal{U}_Z|+\rangle$ [10]. Here, $\mathcal{U}_Z \equiv R_Z(\varphi)$ and $\mathcal{U}_Z^\dagger \equiv R_Z(-\varphi)$ denote Pauli-Z rotations. (b) Alternative circuit for implementing the (logical) gate T on a general (logical) state $|\psi\rangle$ [17]. Here, for the circuit on the left, $\mathcal{O} = e^{-i\pi/4} YS = e^{-i\pi/4} SX$, such that the state $T|+\rangle$ is the $+1$ eigenstate of $\mathcal{O}$. Therefore, the C$\mathcal{O}$ gate on the left has no effect (see Fig. 2.7b), just as the Pauli-Z and Pauli-X measurements on the bottom qubit do not influence the state of the top qubit. The C$\mathcal{O}$ gate can be converted to a CX gate, a CS gate, and $T^\dagger$ on the top qubit (which cancels against T). Using the identity in Fig. 2.7c, the CS gate in front of the Pauli-Z measurement on the bottom qubit is converted to a correction with S after the measurement. Using the identity in Fig. 2.7a, the Pauli-Z measurement on the bottom qubit is then commuted through the CX gate, resulting in the joint ZZ measurement on both qubits. Finally, using the identity in Fig. 2.7d, the CX gate before the Pauli-X measurement on the bottom qubit is converted to a correction with Z after the measurement. (c) Alternative circuit for implementing the (logical) gate S on a general (logical) state $|\psi\rangle$ [12]. The state $S|+\rangle = |+i\rangle$ is the $+1$ eigenstate of the Pauli-Y operator. This CY gate can be used to remove the S gate after converting it to a CX and a CZ gate using the identity in Fig. 2.6a. The (logical) CZ gate in the circuit on the right can be converted to a CX gate surrounded by Hadamard gates using the equality shown in Fig. 2.5.

the logical qubit, completing the injection of the magic state as $|\psi\rangle_{\mathrm{L}}$ on the logical qubit. Typically, magic state injection leads to a noisy version of the requested logical state. Using distillation, producing multiple copies of this noisy logical state allows one to improve the fidelity of one of these states by consuming the other copies [12, 16]. We discuss this in more detail in Sec. 3.6.1.

3.4.8 Error correction with measurement errors

In Sec. 3.4.3, we discuss how general two-dimensional topological codes can correct errors in a situation where the stabilizer generators $\{g_k\}_k$ and $\{\overline{g}_k\}_k$ can be measured without measurement errors. We mentioned that, in situations *with* measurement errors, their error-correction properties vanish. Here, we discuss how this is typically remedied.

Intuitively, the inability to detect errors in a situation with measurement errors can be understood by realizing that the error syndromes $\mathbf{m}$ and $\overline{\mathbf{m}}$ are 0-cochains and 0-chains, as explained in Sec. 3.4.3. If a stabilizer measurement flips because of an error, we are not able to detect this with a topological code, since such an error only shows up at that specific location. Ideally, we would like the stabilizer measurements to form the (co)boundary of a chain in a vector space one dimension higher (lower). In that way, we make sure that each

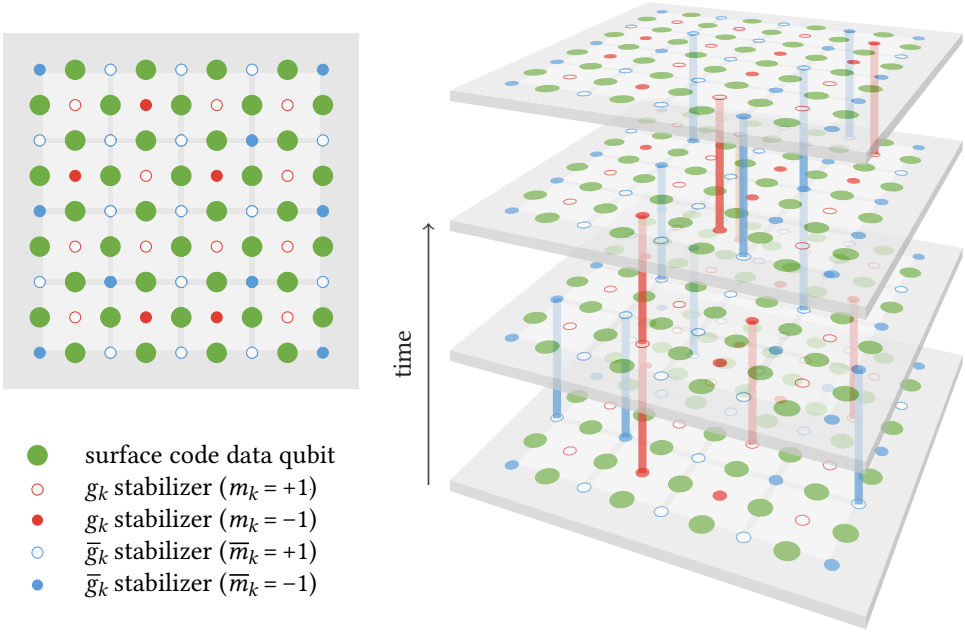


Figure 3.8: Schematic overview of the error-correction process for the surface code in a scenario with measurement errors. Primal $\{g_k\}_k$ (*i.e.*, plaquette) and dual $\{\overline{g}_k\}_k$ (*i.e.*, vertex) stabilizer generators are measured repeatedly in multiple time layers. The error syndrome now becomes a three-dimensional graph that is no longer based on individual $m_k = -1$ or $\overline{m}_k = -1$ stabilizer outcomes. Instead, stabilizer outcomes $m_k^{(t)}$ or $\overline{m}_k^{(t)}$ in a certain time layer t are only added to the syndrome graph if their sign differs from the measurement outcome of the same stabilizer in the previous time layer. In the figure, this is indicated by the vertical red and blue lines drawn between the time layers.

stabilizer borders multiple objects itself. Because, in that situation, stabilizer measurement flips show up in more than one place, errors in them also show up as strings with a boundary and can be captured. This indicates why it is necessary to resort to a different approach or code construction for achieving fault tolerance against stabilizer measurements. Alternative code constructions that offer resilience against stabilizer measurement errors are introduced in Sec. 3.5 with three-dimensional cluster states.

On top of that, a typical solution for the measurement error problem with topological codes themselves consists of consecutive measuring the stabilizer operators over multiple time layers before starting the decoding process. This gives rise to the collection $\{\mathbf{m}^{(t)}, \overline{\mathbf{m}}^{(t)}\}_t$ over multiple time layers t. The idea is that, for constructing the final error syndrome, only measurement outcomes that are *different* from the outcome of the same stabilizer operator in the previous time layer are included in the syndrome. This procedure is depicted schematically in Fig. 3.8. For two-dimensional codes, this method transforms the error syndrome into a three-dimensional syndrome graph. Including more time layers into the error syndrome generally leads to more resilience against measurement errors, but also to more computational complexity in decoding the error syndrome. In Ch. 7, we use this method to deal with noisy stabilizer measurements of the surface code.

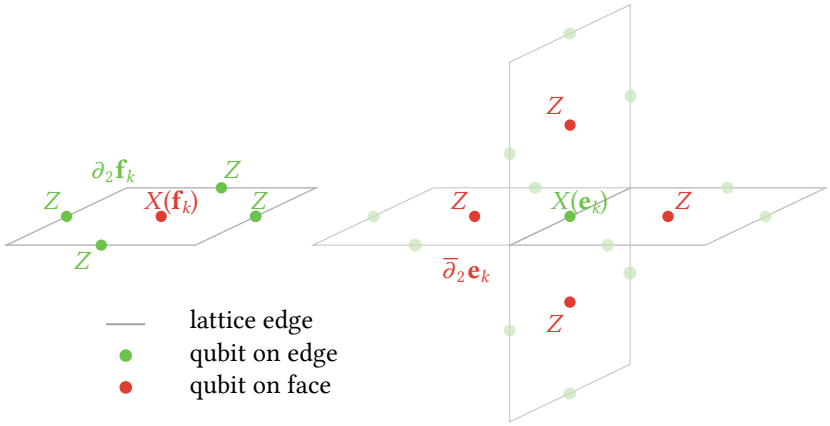


Figure 3.9: A primal and dual stabilizer generator for a three-dimensional cluster state as defined in Sec. 3.5.1. The primal generator $X(\mathbf{f}_k)Z(\partial_2 \mathbf{f}_k)$ is associated with the highlighted face on the left of the figure. The dual generator $X(\mathbf{e}_k)Z(\overline{\partial}_2 \mathbf{e}_k)$ is associated with the highlighted edge on the right of the figure. Stabilizer generators associated with the other faces and edges are not shown explicitly.

3.5 Three-dimensional cluster states

In this section, we discuss the construction and error-correction process of fault-tolerant cluster states in three dimensions. Just as in Sec. 3.4, we make use of the mathematical formalism introduced in Sec. 3.3. As outlined in Sec. 3.5.2, these cluster states can be seen as a generalization of fault-tolerant channels created over time with the two-dimensional codes of Sec. 3.4. This means that one can not only regard these cluster states as fault-tolerant resource states for MBQC but also as more general fault-tolerant channels that transport or operate on logical information. In Secs. 3.5.1 and 3.5.3, we elaborate on constructing the stabilizers of the cluster state and the general procedure for introducing a logical subspace. Additionally, we spend a few words on how the cluster state error syndrome is constructed in Sec. 3.5.4 and how error correction is performed (and can be compared to topological codes) in Sec. 3.5.5.

3.5.1 Cluster state construction

We make use of three-dimensional chain complexes $C_3 \to C_2 \to C_1 \to C_0$ to construct a cluster state on a three-dimensional space $\mathcal{L}''$, by following the following steps:

1. Place qubits on all basis elements of C_2 and $\overline{C}_2$, *i.e.*, on each face $\mathbf{f}_k$ and each edge $\mathbf{e}_k$ of the lattice.

2. For each face $\mathbf{f}_k$, *primal* stabilizer generators are defined as $g_k = X(\mathbf{f}_k)Z(\partial_2 \mathbf{f}_k)$. These operators generate stabilizer operators $X(\mathbf{c}_2)Z(\partial_2 \mathbf{c}_2)$ for each 2-chain $\mathbf{c}_2 \in C_2$.

3. For each edge $\mathbf{e}_k$, *dual* stabilizers are defined as $\overline{g}_k = X(\mathbf{e}_k)Z(\overline{\partial}_2 \mathbf{e}_k)$. These operators generate stabilizer operators $X(\overline{\mathbf{c}}_2)Z(\overline{\partial}_2 \overline{\mathbf{c}}_2)$ for each 2-cochain $\overline{\mathbf{c}}_2 \in \overline{C}_2$.

We depict an example of a primal and dual stabilizer generator associated with a single face $\mathbf{f}_k$ and edge $\mathbf{e}_k$ in Fig. 3.9. Each stabilizer generator carries a Pauli-X operator on some face

(edge) qubit and Pauli-Z operators on its direct neighbors on the (dual) boundary. This stabilizer composition is commonly associated with graph or cluster states—as introduced in Sec. 2.3.5. We note that, as discussed in more detail in Sec. 3.5.3, encoding logical information in the cluster state requires realizing a logical subspace by "switching off" some of the stabilizers.

3.5.2 Foliation of two-dimensional topological codes

In Sec. 3.4.8, we discuss how one can correct errors with topological codes in the presence of measurement noise: this is typically achieved by measuring stabilizer generators over multiple time layers. In this section, we explain how collecting syndrome measurements over multiple layers bears a close resemblance to constructing a three-dimensional cluster state in the form of Sec. 3.5.1. The analogy arises by *foliating* the two-dimensional topological in a third dimension [18, 19].

In Fig. 3.10, we depict the main idea of foliation for the (toric) surface code introduced in Sec. 3.4.4. We start with the surface code defined on the green qubits and focus on a situation in which stabilizer generators are measured over multiple layers. We split up primal stabilizers $\{g_k\}_k$ and dual stabilizers $\{\overline{g}_k\}_k$ in alternating time layers. (Note that these are the stabilizers of the topological codes as defined in Sec. 3.4.1, and not the cluster state stabilizers introduced in Sec. 3.5.1.) At this point, we introduce the circuit of Fig. 2.4a to measure these stabilizers with ancillary qubits initialized in the $|+\rangle$ state: red qubits for the primal stabilizers $\{g_k\}_k$ and blue qubits for the dual stabilizers $\{\overline{g}_k\}_k$. Since the primal stabilizer generators are Pauli-Z strings and the dual stabilizers are Pauli-X strings, we can restrict to using CZ gates, as long as we place Hadamard gates on the green data qubits between the time layers—*i.e.*, for switching between primal and dual stabilizers.

So far, the procedure completely resembles measuring stabilizers in the presence of measurement errors as discussed in Sec. 3.4.8: the green data qubits of the code are the same (physical) qubits in all layers. However, we can switch to a situation in which the states of the data qubits are teleported from one layer to the next. If we use the one-bit teleportation scheme with a CZ gate depicted in Fig. 2.8b, we get the required Hadamard gates "for free". This leads to the structure depicted in Fig. 3.11. Neglecting the states of the green data qubits in the bottom layer for the moment, all qubits in the resulting structure are initialized in the $|+\rangle$ state, and we only use CZ gates between the qubits. This means we are dealing with a cluster state. Moreover, the placement of the qubits and CZ gates on the underlying lattice is completely in agreement with the definition of Sec. 3.5.1. We elaborate on the role of the green data qubits in the bottom layer in Sec. 3.5.3.

3.5.3 Commutation requirements and logical subspace

To realize that the stabilizers in Sec. 3.5.1 commute, we can show that if two adjacent primal and dual stabilizers overlap with their Pauli-X and Pauli-Z operators, this is always on an even number of qubits. More specifically, since Eq. (3.11) shows that $\mathbf{c}_2 \cdot \overline{\partial}_2 \overline{\mathbf{c}}_2 = \partial_2 \mathbf{c}_2 \cdot \overline{\mathbf{c}}_2$, one can understand that the overlap of the X-part of a primal stabilizer with the Z-part of a dual stabilizer is always equal to the overlap of the Z-part of the primal stabilizer with the X-part of the same dual stabilizer.

The cluster states constructed with the description of Sec. 3.5.1 do not contain logical operators. This is because these cluster states are fully defined states—*i.e.*, there is no

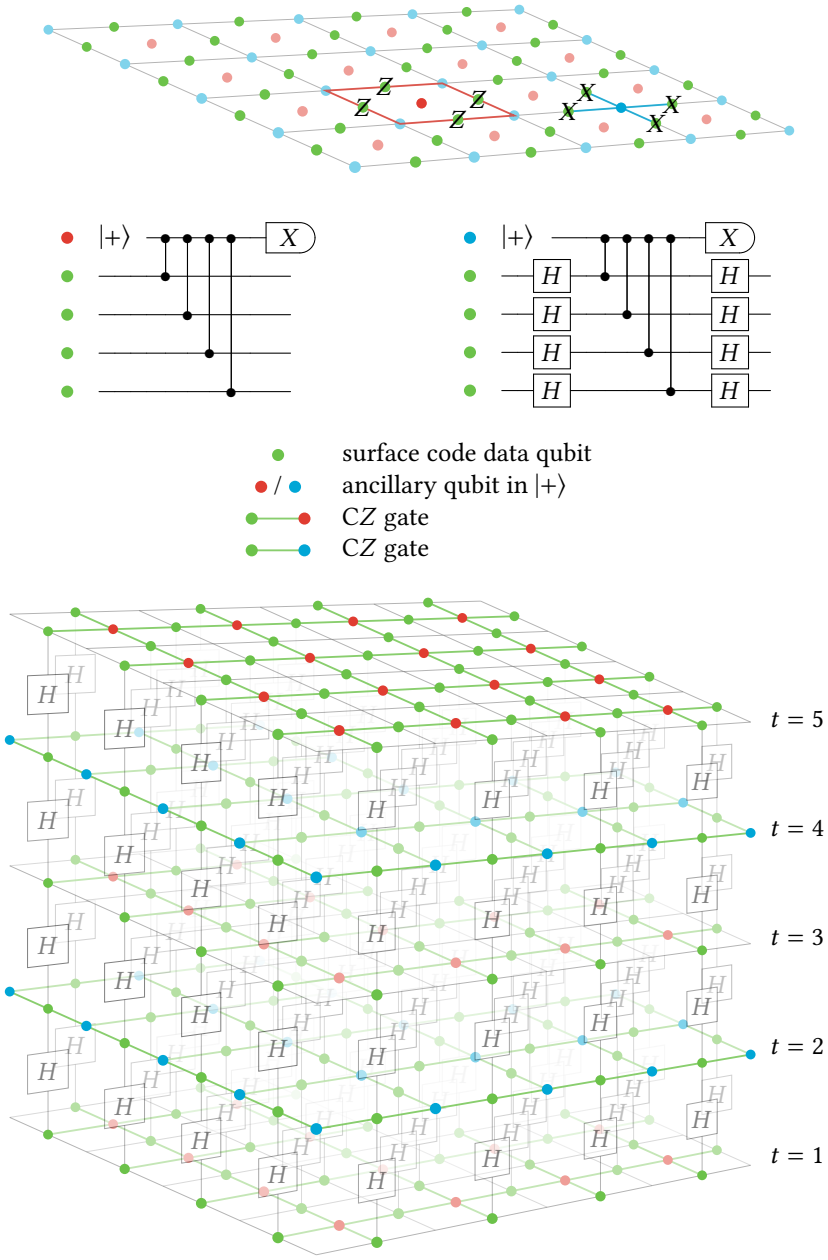


Figure 3.10: Initial steps of the foliation process of the two-dimensional surface code into a three-dimensional cluster state that is able to deal with stabilizer measurement errors. Primal $\{g_k\}_k$ (red dots) and dual $\{\overline{g}_k\}_k$ (blue dots) stabilizers are measured with ancillary qubits using the circuit of Fig. 2.4a and added to different time layers in a repetitive and alternating fashion. Hadamard gates are placed in between the time layers on the data qubits (green dots) to switch between the primal and dual stabilizers. In Fig. 3.11, the Hadamard gates are replaced with CZ gates to construct the cluster state.

3

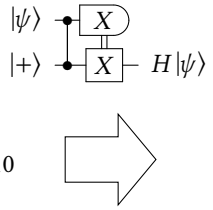


Fig. 3.10

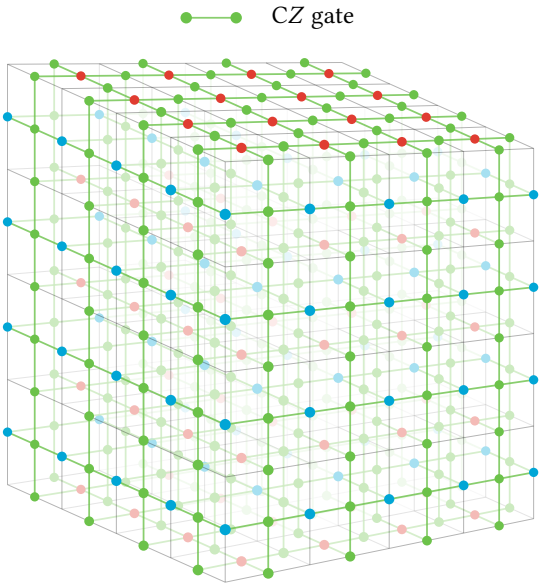


Figure 3.11: Continuation of Fig. 3.10. The one-bit teleportation scheme of Fig. 2.8b—printed here above the arrow—is used to replace the Hadamard gates between the time layers with CZ gates. The step includes adding a new qubit in the $|+\rangle$ qubit for each data qubit in every layer: in Fig. 3.10, the data qubits in each layer are the same physical qubits, whereas here, the states of the data qubits are teleported from one layer to the next. The result is a cubic cluster state. As mentioned in Sec. 3.5.2, the byproduct H gate of the teleportation scheme is used to switch between the plaquette stabilizers $\{g_k\}_k$ (red dots) and vertex stabilizers $\{\overline{g}_k\}_k$ (blue dots). The one-bit teleportation scheme contains a Pauli-X correction in case of an $m = -1$ measurement. We discuss the role of this correction for the error-correction process in Sec. 3.5.5.

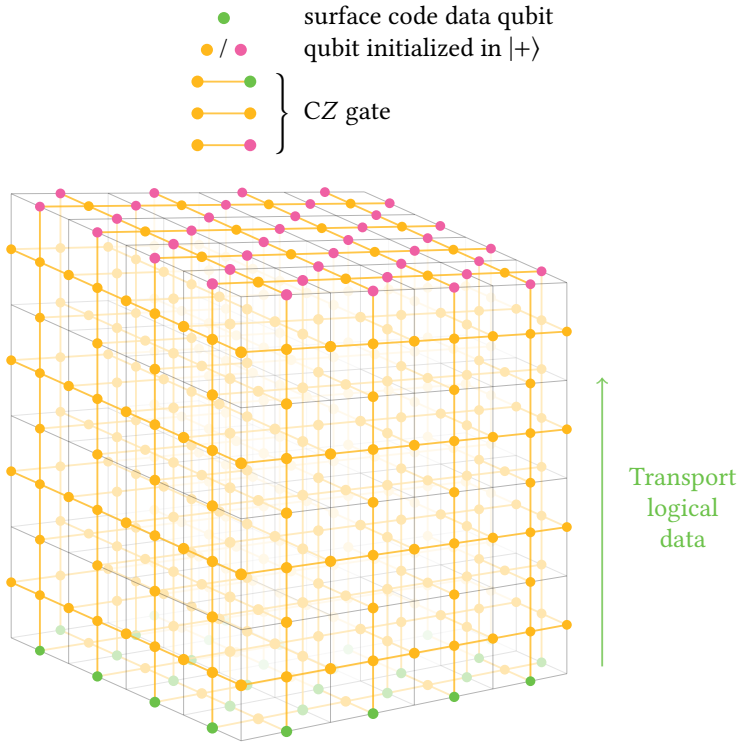


Figure 3.12: Interpretation of Fig. 3.11 as a fault-tolerant cluster state channel. With this cluster state, the logical qubits encoded on the green data qubits of the cluster state's bottom layer can be transported to the pink qubits of the top layer by measuring out all green and yellow qubits in the Pauli-X basis. The measurement outcomes are used to construct the error syndrome graph according to the procedure described in Sec. 3.5.4.

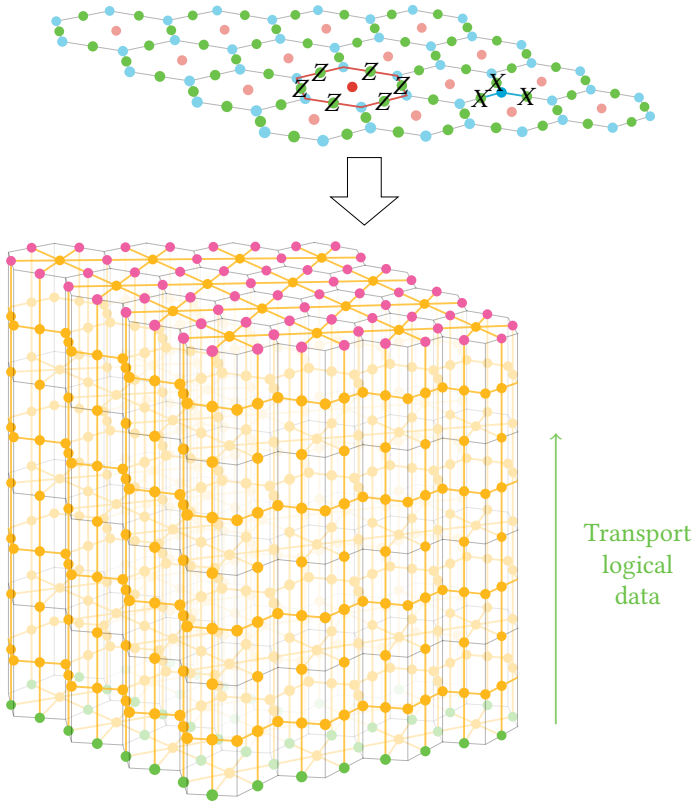


Figure 3.13: Fault-tolerant cluster state channel based on a two-dimensional topological code constructed with the hexagonal lattice. For this two-dimensional lattice, the $\{g_k\}_k$ plaquette stabilizer generators are of weight six, whereas the $\{\overline{g}_k\}_k$ vertex stabilizers are of weight three. The coloring of the different types of qubits and the CZ gates is identical to the coloring in Fig. 3.12. Measurements on the green and yellow qubits are used to construct an error syndrome graph according to the procedure of Sec. 3.5.4.

logical code space. To introduce logical operators, we need to remove some of the stabilizer operators. This is usually done by introducing lattice boundaries and switching off stabilizers on these boundaries. An intuitive example of how this works is found in the foliated surface code of Sec. 3.5.2. Here, the code's data qubits in the bottom time layer are not stabilized by cluster state stabilizers, but by the original surface code stabilizers. This provides intuition on how a general cluster state of Sec. 3.5.1 can be used as a fault-tolerant channel: by removing cluster state stabilizer generators on at least one of the boundaries of the lattice and initializing a topological code there. In Fig. 3.12, we show a more explicit example of the foliated surface code as a fault-tolerant cluster state channel.

From Figs. 3.10, 3.11, and 3.12, it becomes clear that, for the surface code, foliation leads to a cluster state defined on a cubic lattice that can be used to transport logical qubits encoded with the surface code. It should be mentioned that one can apply foliation to any two-dimensional topological code constructed according to Sec. 3.4.1. For example, in Fig. 3.13, a fault-tolerant cluster state channel is constructed from a topological code defined

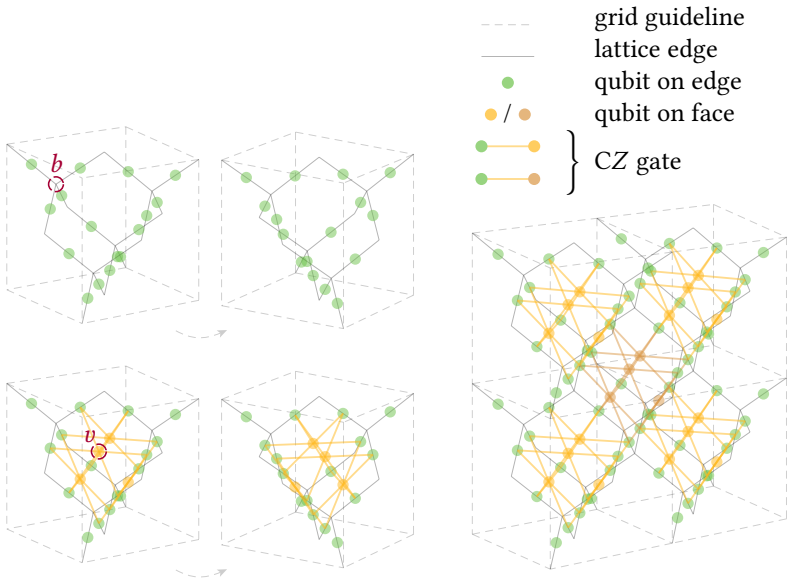


Figure 3.14: Cluster state based on the diamond lattice according to the definition of Sec. 3.5.1. Qubits (initialized in the $|+\rangle$ state) are placed on the edges (top left) and faces (bottom left) of the lattice. CZ gates are added between each face qubit and the edge qubits of the corresponding face (bottom left). On the left side of the figure, we see individual unit cells of the diamond lattice. On the right side, we see a crystal constructed from four unit cells. In the unit cells on the left, b denotes the *bond-degree* of the lattice: the number of elements in the coboundary map of a lattice vertex. Furthermore, v denotes the *valence* of the lattice: the number of elements in the boundary map of a lattice face. For the diamond lattice, $b = 4$ and $v = 6$. For a *self-dual lattice*—like the cubic lattice and the diamond lattice—b also equals the number of elements in the boundary map of each three-dimensional lattice cell, whereas v equals the number of elements in the coboundary map of each edge. A self-dual lattice is a lattice for which the dual lattice is identical to the primal lattice—*i.e.*, for which both the number of elements of the vector spaces and the boundary maps between the vector spaces are the same in the primal and dual chain complexes.

on a two-dimensional hexagonal lattice. However, foliating two-dimensional codes into three-dimensional cluster states only gives rise to a subclass of three-dimensional cluster states. For example, in Fig. 3.14, we see a cluster state constructed on a diamond lattice—a lattice that can not be constructed by foliating a two-dimensional lattice. Therefore, fault-tolerant channels based on cluster states constructed with Sec. 3.5.1 can be regarded as more general versions of topological error-correction channels in which the time dimension is replaced by a third spatial dimension. However, a better statement would be that, for these three-dimensional cluster states, the role of time and space is no longer fixed, but can be interpreted as more fluid. That is because cluster states can be initialized and measured in arbitrary directions over time. A concrete example is so-called *interleaving* in *fusion-based quantum computation* [20], which bears many resemblances to the measurement-based architectures considered here.

For a general three-dimensional cluster state, logical operators can be introduced as non-trivial (co)cycles on lattice boundaries. Multiplying these logical operators with face stabilizers of the lattice in the direction opposite to the logical operator gives rise to so-called *correlation surfaces* or *membranes*. These surfaces are versions of the logical operators that span from one lattice boundary to another and terminate at these boundaries. We exemplify the identification of correlation surfaces in Fig. 3.15. The cluster state can be interpreted as a channel that transports (or operates on) logical information encoded in a topological code. In this interpretation, the correlation surfaces describe how the code's logical operators are mapped from one lattice boundary to another on the other side of the lattice.

In this thesis, we solely focus on cluster states that serve as *memory* channels. More explicitly, we do not specifically consider logical operators, but rather think of logical operators conceptually as chains moving from one side of the lattice to the other in all three dimensions. We note that fault-tolerant cluster state channels that apply logical operators can be constructed with more elaborate correlation surfaces that connect logical representations of *different* operators on both sides of the channel. More details about this approach can be found in Fujii [9], and Bombín *et al.* [21].

3.5.4 Error syndrome construction

For the cluster states of Sec. 3.5.1, we can construct the primal (dual) error syndrome by measuring the qubits on the faces (edges) in the Pauli-X basis. For each face $\mathbf{f}_k$ (edge $\mathbf{e}_k$), this leads to a measurement outcome $m_k \in \{+1, -1\}$ ($\overline{m}_k \in \{+1, -1\}$). Error syndromes are constructed from measurement outcomes in the following way:

1. For each cell $\mathbf{q}_k$, we produce a *primal* error syndrome μ_k as the product of measurement outcomes of the qubits that lie on its boundary: $\mu_k = \prod_{\mathbf{f}_j \in \partial_3 \mathbf{q}_k} m_j$. By doing this, we classically produce syndrome outcomes μ_k for the stabilizer operators $s_k \equiv \prod_{\mathbf{f}_j \in \partial_3 \mathbf{q}_k} g_j = X(\partial_3 \mathbf{q}_k) Z(\partial_2 \partial_3 \mathbf{q}_k) = X(\partial_3 \mathbf{q}_k)$. It can be seen that the operators s_k are stabilizer operators of the code by realizing that the Pauli-Z operators of the g_j generators at the faces on the boundary of $\mathbf{q}_k$ cancel against each other.

2. For each dual cell $\overline{\mathbf{q}}_k$ (*i.e.*, vertex $\mathbf{v}_k$), we produce a *dual* error syndrome $\overline{\mu}_k = \prod_{\mathbf{e}_j \in \overline{\partial}_3 \mathbf{v}_k} \overline{m}_j$. The syndrome represents the measurement outcome of the stabilizer $\overline{s}_k \equiv \prod_{\mathbf{e}_j \in \overline{\partial}_3 \mathbf{v}_k} \overline{g}_j = X(\overline{\partial}_3 \mathbf{v}_k)$.

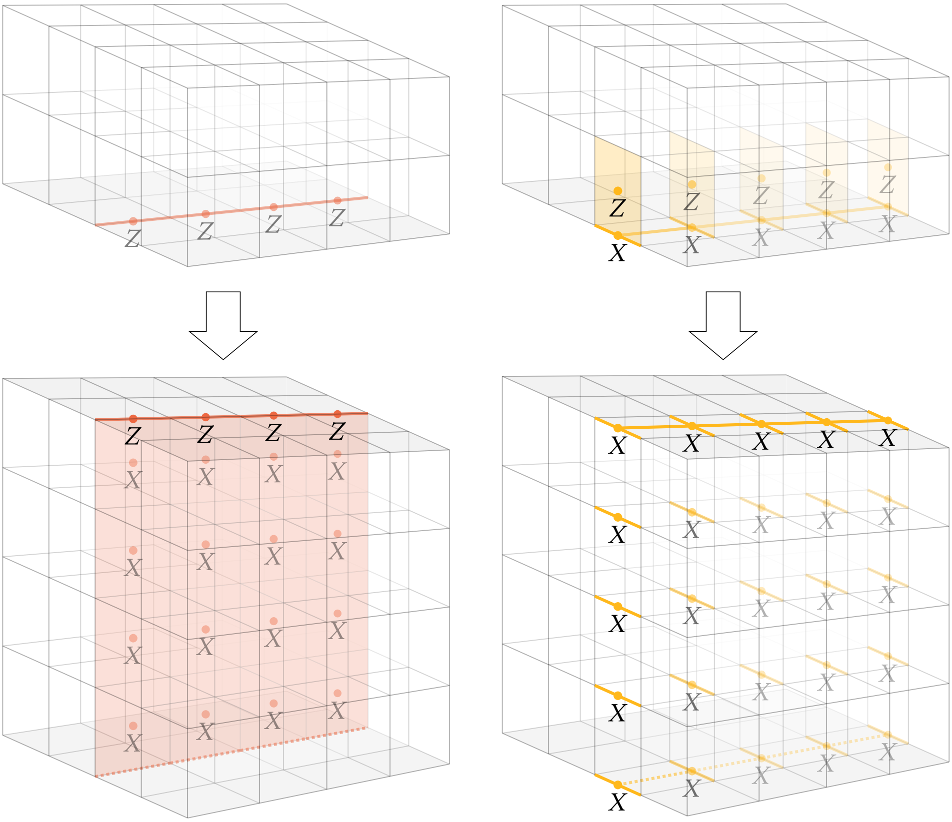


Figure 3.15: Correlation surfaces for a logical Z (right) and a logical X (left) operator of a cubic cluster state. These surfaces describe how a logical operator is transported from one lattice boundary to another—in this case, from the gray faces at the bottom layer of the cluster state to the gray faces at the top layer. (Top figures) On the gray faces of the bottom boundary, we introduce a surface code. In the top left figure, the non-trivial cycles that form logical Z operators on the surface code boundary directly become logical Z operators of the full fault-tolerant cluster state. For the surface code's logical X operators in the top right figure, we need to add Pauli-Z gates on connected faces to make sure the logical X operators commute with all remaining cluster state stabilizers. (Bottom figures) The logical operators are multiplied with stabilizers of a connected set of faces (left) and edges (right) in the bulk of the cluster state. This is done to create alternative representations of the logical operators that connect the two lattice boundaries: the correlation surfaces. Note that the set of edges constructed from the logical X operator forms a set of connected faces in the dual lattice. For convenience, periodic boundary conditions are used in the two horizontally drawn directions. As a result, many Pauli gates in the stabilizers cancel during the construction of the correlation surfaces, and two logical qubits appear on the surface codes introduced on the gray boundaries—here we depict the logical Z and logical X operators of *different* qubits. On top of that, in this figure, just as in Figs. 3.12 and 3.13, logical subspaces are created by switching off *dual* cluster state stabilizers on a *primal* lattice boundary—*i.e.*, by creating a smooth boundary on the primal lattice. However, one can alternatively create a logical subspace in a different way—*e.g.*, by switching off *primal* stabilizer generators on a *dual* lattice boundary and creating a rough boundary on the primal lattice. Alternatively, logical subspaces can be created by introducing holes in the cluster state stabilizer framework, similarly to the holes introduced in the two-dimensional surface code—see Sec. 3.4.6 and Ref. [9] for more details.

This procedure implements the collection of stabilizer measurements into higher-dimensional objects (as discussed earlier in Sec. 3.4.8). Note that in the absence of errors, all error syndromes produce outcomes $\mu_k = +1$ and $\overline{\mu}_k = +1$, even though the individual measurement outcomes m_k and $\overline{m}_k$ are likely to contain many -1 values. The full syndrome graphs are constructed as $\boldsymbol{\mu} = \{\mu_k\}_k$ and $\overline{\boldsymbol{\mu}} = \{\overline{\mu}_k\}_k$, where we define $\boldsymbol{\mu}$ and $\overline{\boldsymbol{\mu}}$ over $\mathbb{Z}_2$ by mapping their elements with $\{+1, -1\} \mapsto \{0, 1\}$.

Since all cluster state qubits are measured in the Pauli-X basis, it is sufficient to restrict to errors that appear as Pauli-Z errors prior to measurement. Because such an error is equivalent to a probabilistic Z gate before an X-basis measurement, this model also includes measurement errors. Just as for the two-dimensional code of Sec. 3.4.1, the error-correction procedure is able to deal with error sets composed of Pauli strings. Since qubits sit on edges (*i.e.*, objects in $C_1 = \overline{C}_2$) and faces (*i.e.*, objects in $C_2 = \overline{C}_1$) of the lattice, without loss of generality, we can describe an arbitrary error as $Z(\mathbf{c}_1)Z(\overline{\mathbf{c}}_1)$. The primal error syndrome $\boldsymbol{\mu} = \{\mu_k\}_k = \overline{\partial}_1 \overline{\mathbf{c}}_1$ is a graph on the dual lattice formed by the coboundary of the dual error chain $Z(\overline{\mathbf{c}}_1)$—*i.e.*, the coboundary of all Pauli-Z errors on qubits that sit on the faces of the graph. The dual syndrome graph is constructed on the primal lattice with the syndrome checks $\overline{\boldsymbol{\mu}} = \{\overline{\mu}_k\}_k$. This is the boundary $\overline{\boldsymbol{\mu}} = \partial_1 \mathbf{c}_1$ of the primal error chain $Z(\mathbf{c}_1)$—*i.e.*, the boundary of all Pauli-Z errors on the edge qubits.

3.5.5 Error correction

Given error syndrome outcomes $\boldsymbol{\mu} = \overline{\partial}_1 \overline{\mathbf{c}}_1$ and $\overline{\boldsymbol{\mu}} = \partial_1 \mathbf{c}_1$ constructed from an error $Z(\mathbf{c}_1)Z(\overline{\mathbf{c}}_1)$, error correction now consists of using an error syndrome decoder on the syndrome $\{\boldsymbol{\mu}, \overline{\boldsymbol{\mu}}\}$ to identify recovery chains $\overline{\mathbf{r}}_1$ and $\mathbf{r}_1$ such that $\overline{\partial}_1(\overline{\mathbf{r}}_1 + \overline{\mathbf{c}}_1) = 0$ and $\partial_1(\mathbf{r}_1 + \mathbf{c}_1) = 0$—*i.e.*, the sum of recovery and error chains form *cycles* in the corresponding chain complex. We identify a logical failure whenever decoding introduces a logical X and/or logical Z error across the channel. For, *e.g.*, logical subspaces as defined according to Fig. 3.15, a logical error occurs whenever the sum of a recovery and error chain forms a *non-trivial* cycle. In such a situation, a logical failure can be observed with the aid of the correlation surfaces: if the sum of a recovery and error chain anti-commutes with the correlation surface of the associated logical Z (X) operator, a logical X (Z) failure occurs. A chain of Pauli-Z operators anti-commutes with a correlation surface if it crosses the correlation surface an odd number of times.

Because, in this work, we do not explicitly consider cluster states with logical subspaces, we simply define a logical error as an odd number of crossings of $\overline{\mathbf{r}}_1 + \overline{\mathbf{c}}_1$ and/or $\mathbf{r}_1 + \mathbf{c}_1$ through a two-dimensional plane in any of the three spatial dimensions of the lattice. This approach corresponds to evaluating the fault tolerance of a cluster state as a pure quantum memory, as discussed in Sec. 3.5.3—it exclusively assesses the state's ability to protect logical information and not, *e.g.*, its ability to encode logical information or operate on it.

Just as in general for three-dimensional cluster states, the syndrome graph can be regarded as a generalization of the two-dimensional error syndrome measured over time. As explained in Sec. 3.4.8, for a two-dimensional code that experiences measurement errors, we collect measurement outcomes $\{\mathbf{m}^{(t)}, \overline{\mathbf{m}}^{(t)}\}_t$ by repeating stabilizer measurements over multiple time layers t. The error syndrome is then built up from stabilizer measurements that change outcome from one layer to the next. For a foliated two-dimensional code, this gives rise to the procedure described in Sec. 3.5.4 to construct the error syndrome

graph. The difference between the cluster state syndrome construction in Sec. 3.5.4 and the non-foliated two-dimensional error syndrome is that the measurement outcomes of qubits on connected faces in vertical directions are also included in the cluster state syndrome. This is necessary to correctly deal with the Pauli-X corrections of the one-bit teleportation scheme—see Fig. 3.11 for more details. Since, in the cluster state, qubits are teleported further before these corrections are applied, their influence shows up in the construction of the error syndrome.

As indicated with Figs. 3.12 and 3.13, in practical situations, we measure most of the cluster state qubits before decoding the syndrome graph. In situations like this, the identified Pauli-Z recovery chains $\bar{\mathbf{r}}_1$ and $\mathbf{r}_1$ can typically not be applied verbosely, since the qubits on which they are supposed to be applied are no longer part of the cluster state. To find alternative versions of $\bar{\mathbf{r}}_1$ and $\mathbf{r}_1$ that *can* be applied, one has to consider the details of the exact implementation of the cluster state as a fault-tolerant channel. For example, if the qubits are measured from bottom to top—like in Figs. 3.12 and 3.13—a version of $\bar{\mathbf{r}}_1$ ($\mathbf{r}_1$) can be found by multiplying the chain with edge (face) generators of the original cluster state in the upwards direction.

As mentioned above, evaluating whether or not a recovery chain leads to a logical error does *not* in itself require translating the recovery operator to an operator on a set of qubits that are still part of the remaining cluster state. Instead, we can simply check for anti-commutation with the correlation surfaces of the logical operators. This is because, on qubits relevant to determine (anti-)commutation—*i.e.*, qubits on elements of C_1 (C_2)—the correlation surfaces of logical X (logical Z) operators exclusively apply Pauli-X operators. Since the qubits of the cluster state are measured in the Pauli-X basis, this makes the correlation surfaces valid representations of the logical operators both before and after (part of the) cluster state qubits are measured out.

This naturally leads to a more thorough analysis of the role of Pauli-X errors in the cluster state. At first glance, it seems that Pauli-X errors are undetectable since all qubits are measured in the Pauli-X basis. This is true for errors that happen right before the measurements—*i.e.*, after the full cluster state is constructed. This situation is completely identical to Pauli-X errors on ancillary qubits used to measure the stabilizer operators of a two-dimensional topological code—*e.g.*, right before measuring the red and blue qubits in Fig. 3.10. On top of that, Pauli-X errors occurring on qubits in $|+\rangle$ before the CZ gates are applied are also undetectable. This can be understood by realizing that, even though the CZ gates cause these errors to propagate as Pauli-Z errors to neighboring edges or faces, this never leads to a flip in the syndrome outcome of a full vertex or cell—the propagated Pauli-Z gates are always bound to flip an even number of measurements in each (dual) cell. This is, of course, what should happen, since states initialized in $|+\rangle$ are stabilized by Pauli-X. In other words, Pauli-X errors that appear before the CZ gates are applied become stabilizers of the cluster state. We conclude that, in the situations considered so far, Pauli-X errors are indeed undetectable but also harmless.

On the contrary, Pauli-X errors that appear *in the middle* of constructing the cluster state are both harmful and detectable—albeit indirectly detectable. If the Pauli-X error appears on a qubit after at least one (but not all) CZ gates on that qubit are applied, the propagated Pauli-X chain *does* flip vertex or cell syndrome outcomes. Such an error does not become a stabilizer operator of the code but continues as a chain of Pauli-Z errors. This

chain is detectable by the error-correction process. Furthermore, Pauli-X errors appearing on qubits that are not measured—*i.e.*, the qubits on the final surface layer of a cluster state channel—are harmful and undetectable. This is no different from errors occurring after the last round of stabilizer measurements for two-dimensional channels. Finally, also Pauli errors appearing on the initial surface layer of the cluster state, *i.e.*, the qubits not initialized in $|+\rangle$, lead to the same vulnerable situation as in the two-dimensional case.

3.6 Entanglement distillation

In (distributed) quantum computing, we want the fidelity of (entangled) states to be high for the calculations to be performed with high precision. There is a remedy for increasing the quality of non-perfect or low-fidelity states by making use of distillation—also known as *purification*. Typically, in the context of quantum computing, distillation is defined as the use of local operations and classical communication to convert multiple quantum states into a smaller number of states with higher fidelity. In this section, we discuss the main ideas of distillation, including the first and most important distillation protocols, and share general remarks on more recent work on distillation. For most of this section, we focus on performing entanglement distillation (*i.e.*, specifically distilling entangled quantum states), as this is important in the context of distributed quantum computation. These ideas serve as the basis for the entanglement distillation operations used in Ch. 5 and onwards. However, we start in Sec. 3.6.1 by discussing the distillation of general quantum states. This is relevant in the context of magic state distillation discussed in Sec. 3.4.7.

In general and in essence, all distillation protocols use quantum error-correction codes to detect errors in quantum states. However, instead of correcting these errors, some protocols simply discard the post-measurement state if an error is detected. These protocols only use the error-*detection* properties of an error-correction code or deploy a pure *error-detecting code* altogether—*i.e.*, a code that is only able to detect errors and cannot correct them since different (equally likely) errors result in the same error syndrome. Discarding the state in case errors are detected introduces a probabilistic success factor in the distillation protocols. Below, we distinguish two types of distillation: distillation based on stabilizer measurements and distillation purely based on error correction. On top of that, we consider distillation of states shared between two parties (*i.e.*, *bipartite* distillation protocols) as well as distillation of states shared between more than two parties (*i.e.*, *multipartite* distillation protocols). Bipartite protocols with stabilizer measurements are discussed in Sec. 3.6.2 and bipartite protocols based on error-correction codes in Secs. 3.6.3 and 3.6.4. Multipartite protocols with stabilizer measurements are considered in Sec. 3.6.7 and multipartite distillation with error correction in Sec. 3.6.8. Furthermore, we include a few words on two other bipartite distillation protocols in Secs. 3.6.5 and 3.6.6, and introduce the generation and distillation of GHZ states with Bell pairs in Sec. 3.6.9. In Sec. 3.6.4, we show how distillation based on stabilizer measurements forms a subclass of distillation based on error correction, classifying both types of distillation as different sides of the same coin. In all parts of this section we extensively use ideas and concepts from Secs. 2.3.4, 3.2.2, and 3.2.3. With the exception of the analysis in Sec. 3.6.5, we assume that the distillation protocols work on non-perfect input states with perfect gates and measurements, while qubits do not (further) decohere over time. This means that the analyses only provide *estimations* for how the protocols work—*i.e.*, in reality, these protocols are less effective.

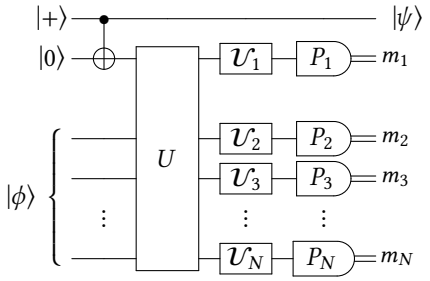


Figure 3.16: Distillation of a state $|\psi\rangle \equiv \mathcal{U}^T|+\rangle$ with an $[\![N,K,d]\!]$ stabilizer code that encodes a single logical qubit—*i.e.*, $K = 1$. The state $|\phi\rangle$ is the $(N-K)$-qubit ancillary state and U is the encoding circuit of the code—see Fig. 3.1a and Sec. 3.2.2 for more information. The protocol encodes half of the Bell pair created by the CX gate between the top two qubits. The stabilizer code should be chosen such that the operator $\mathcal{U}$ that determines $|\psi\rangle = \mathcal{U}^T|+\rangle$ has a transversal logical representation $\overline{\mathcal{U}} = \mathcal{U}_1 \otimes \mathcal{U}_2 \otimes \cdots \otimes \mathcal{U}_N$ in the code. The measurement operators P_1, P_2, ..., P_N are based on the logical X operator $\overline{X} = P_1 \otimes P_2 \otimes \cdots \otimes P_N$.

3.6.1 Distillation of general states

First, we focus on the distillation of general quantum states. Bravyi and Kitaev [16] propose distilling a state $|\psi\rangle$ by initializing the logical version $|\psi\rangle_{\mathrm{L}}$ of this state with a quantum error-correction code. This approach is only efficient as a distillation protocol for certain combinations of states, error-correction codes, and the initialization procedure. In the first place, using an encoding circuit U on $|\psi\rangle$ is ineffective, as this carries over all errors in $|\psi\rangle$ to the logical state $|\psi\rangle_{\mathrm{L}}$. More effective is initializing $|\psi\rangle_{\mathrm{L}}$ by measuring the generators of the error-correction code on an initial state $|\phi'\rangle$—*i.e.*, by projecting with Π_g in the language of Sec. 3.2.3 and Fig. 3.1b. This method can be combined with discarding the logical state if, *e.g.*, certain generator measurements indicate errors. For example, a strategy could be to only keep the final state if all measurements produce $m_i = +1$. This is an efficient strategy if observing $m_i = +1$ for all stabilizer measurements guarantees that the most probable errors are expelled from the logical state. In their paper, Bravyi and Kitaev show a specific example with the five-qubit error-correction code introduced in Sec. 3.2.3: after initializing all five qubits in an eigenstate of $e^{i\pi/4}SH$ (which has a transversal logical representation in the five-qubit code), measuring $m_i = +1$ with the generators of the five-qubit code prevents single-qubit Pauli errors from ending up in the logical state. If required, the distilled logical state $|\psi\rangle_{\mathrm{L}}$ can be converted back to $|\psi\rangle$ by applying a decoding circuit $U^\dagger$. If the initial state $|\phi'\rangle$ consists of multiple copies of the state $|\psi\rangle$ that is to be distilled, this method can be used to recursively distill a state $|\psi\rangle$ in multiple rounds. In Secs. 3.6.3 and 3.6.8 we discuss how this protocol is applied in the context of entanglement distillation.

Raussendorf *et al.* [22] and Fowler *et al.* [12] propose the protocol in Fig. 3.16 for distilling general states. The idea of this protocol is to distill a state $|\psi\rangle = \mathcal{U}^T|+\rangle$, for $\mathcal{U}$ a single-qubit unitary transformation. In this protocol, the final state $|\psi\rangle$ is initialized by encoding and measuring out one qubit of a Bell pair with an error-correction code that has $\overline{\mathcal{U}}$ as a transversal logical representation of $\mathcal{U}$. Encoding one qubit of a Bell pair with the encoding circuit U of an error-correction code leads to

$$|\Phi^+\rangle \mapsto \frac{|0\rangle\otimes|0\rangle_{\mathrm{L}} + |1\rangle\otimes|1\rangle_{\mathrm{L}}}{\sqrt{2}} = \frac{|+\rangle\otimes|+\rangle_{\mathrm{L}} + |-\rangle\otimes|-\rangle_{\mathrm{L}}}{\sqrt{2}} \equiv |\Phi^+\rangle_{\mathrm{1L}}. \tag{3.12}$$

It can be understood that applying the transversal logical operation $\overline{\mathcal{U}}$ on the logical qubit of the state in Eq. (3.12) leads to [23, 24]

$$\left(\mathbb{I}\otimes\overline{\mathcal{U}}\right)|\Phi^+\rangle_{1\mathrm{L}} = \left(\mathcal{U}^T\otimes\mathbb{1}\right)|\Phi^+\rangle_{1\mathrm{L}}. \tag{3.13}$$

A more general version of this relation comes back below in Eq. (3.16). The relation in Eq. (3.13) shows that measuring the logical X operator on the logical qubit now leads to $\mathcal{U}^T|+\rangle$ on the first qubit when $|+\rangle_{\mathrm{L}}$ is observed or $\mathcal{U}^T|-\rangle = \mathcal{U}^T Z|+\rangle$ in case of finding $|-\rangle_{\mathrm{L}}$. If instead of $\mathcal{U}^T|+\rangle$ the state $\mathcal{U}^T Z|+\rangle$ is produced, this typically requires correcting the state with $\mathcal{U}^T Z\mathcal{U}^*$. Before measuring the logical qubit, stabilizer measurements can be applied to filter out errors. Since the logical X operator is a tensor product of single-qubit Pauli operators, one can individually measure the qubits of the logical state and use this to classically construct both the state of the logical qubit in the X basis and, possibly, the parity of one or more stabilizer operators of the code. This makes it possible to discard the full state in case stabilizers with $m_i = -1$ are observed.

For $\mathcal{U}$ a diagonal matrix in the computational basis we have $\mathcal{U}^T = \mathcal{U}$ and the protocol distills a state $|\psi\rangle = \mathcal{U}|+\rangle$. In that case, the correction operator $\mathcal{U}^T Z\mathcal{U}^*$ simplifies to the Pauli-Z operation. Fowler *et al.* show two specific examples of the protocol for a diagonal matrix $\mathcal{U}$: using the 7-qubit Steane code (which has a transversal S gate) to distill $S|+\rangle$ and using the 15-qubit Reed-Muller code (with transversal T) to distill $T|+\rangle$. This makes it understandable that this protocol is mostly suitable for distilling magic states, as discussed in Sec. 3.4.7. The same applies to the protocol of Bravyi and Kitaev, although their protocol also has applications for distilling general states. Other circuit variations for magic state distillation are summarized in Kubica [25]. In the context of magic state distillation, all physical qubits of the distillation protocols become logical qubits, and the protocols produce magic states $|\psi\rangle_{\mathrm{L}}$. The use of logical qubits typically means that the indirect method discussed earlier in Sec. 3.4.7 and Fig. 3.7 is required to apply the gates $\mathcal{U}_i$ in Fig. 3.16—*i.e.*, performing these gates requires magic states themselves. This is exactly how magic state distillation with Fig. 3.16 is applied recursively: by using distilled magic states $|\psi\rangle_{\mathrm{L}}$ from previous rounds to perform the logical operations $\mathcal{U}_i$ in the current round, producing a new $|\psi\rangle_{\mathrm{L}}$ state with higher fidelity.

3.6.2 Bipartite two-to-one stabilizer protocols

We now make the transition to the distillation of entangled states. We start this topic by considering two protocols that use a Bell pair to measure a stabilizer operator on a second Bell pair with the circuit of Fig. 2.4b. For mixed states, the first distillation protocol of this type was introduced by Bennett, Brassard, Popescu, Schumacher, Smolin, and Wootters in 1996 [26]. We typically refer to this protocol as the *BBPSSW protocol.* The protocol consumes a noisy version of the $|\Phi^+\rangle$ state to distill a second noisy version of $|\Phi^+\rangle$ by using the first Bell pair to non-locally measure the Z_1Z_2 parity of the second Bell pair. The analysis of the protocol typically assumes both copies start out as isotropic states—see Sec. 2.4.2 for more information. Since $|\Phi^+\rangle$ is stabilized by the operator Z_1Z_2 (see Table 2.1), this means that the components stabilized by $-Z_1Z_2$ in the density matrix are removed if we find measurement outcome $m = +1$. This typically increases the fidelity of the post-measurement state. However, there is also a probability we end up with measurement outcome $m = -1$. In this case, we have projected on the space $-Z_1Z_2$ and removed components stabilized by

Z_1Z_2, such as our main component $|\Phi^+\rangle\langle\Phi^+|$. This essentially destroys our entangled state: it means we have to regenerate our states and retry the distillation protocol from the start. This is a common feature in most, but not all, distillation protocols: they are *probabilistic*, and obtaining the wrong measurement outcome means we have to throw the state away and start over.

The *DEJMPS protocol*—named after its inventors Deutsch, Ekert, Jozsa, Macchiavello, Popescu, and Sanpera [27]—is similar to the BBPSSW protocol. For this protocol, instead of isotropic states, one assumes the Bell pairs are the more general Bell diagonal states. By non-locally measuring other stabilizers next to Z_1Z_2 and including local rotations, the protocol is able to exploit non-uniformity in the coefficients of the Bell diagonal states—see Eq. (2.18). This can be understood by realizing that both $|\Phi^+\rangle$ and $|\Psi^+\rangle$ are stabilized by X_1X_2; both $|\Phi^+\rangle$ and $|\Phi^-\rangle$ are stabilized by Z_1Z_2; and both $|\Phi^+\rangle$ and $|\Psi^-\rangle$ are stabilized by $-Y_1Y_2$. For each of these three combinations, the other two Bell states are stabilized by minus one times these operators. This means that, along with $|\Phi^+\rangle$, performing distillation by measuring one of these operators also amplifies one other unwanted coefficient. In the DEJMPS protocol, the two-qubit operator $\{X_1X_2, Z_1Z_2, -Y_1Y_2\}$ that is measured using the scheme of Fig. 2.4b is chosen based on the smallest unwanted coefficient in the Bell diagonal state that is being distilled.

On top of that, if we use an ancillary state of the form of Eq. (2.18) to carry out the distillation measurement, the DEJMPS protocol uses single-qubit rotations to permute the coefficients p_{01}, p_{10} and p_{11} of the ancillary state. This is done to additionally only highlight the smallest unwanted coefficient in the ancillary state. Whereas the circuit of Fig. 2.4b carries out the distillation measurement perfectly for an ancillary state $|\Phi^+\rangle = (|00\rangle + |11\rangle)/\sqrt{2}$, this changes if, instead, a noisy version of this state is used. In that case, the joint Pauli operator is only measured approximately. For example, if we use $|\Phi^-\rangle$ instead of $|\Phi^+\rangle$ as the ancillary state, we measure $-P_1P_2$ instead of P_1P_2 on our main state, for $P \in \{X, Z, iY\}$. An overview of all versions of bipartite X_1X_2, Z_1Z_2, and $-Y_1Y_2$ measurements with isotropic states can be found in Table 3.1.

Table 3.1 shows that with two Bell states of the form of Eq. (2.18), no matter what Pauli operator $\{X_1X_2, Z_1Z_2, -Y_1Y_2\}$ is measured, there is always "leakage" of one of the main $|\Phi^+\rangle\langle\Phi^+|$ coefficients to the noisy part of the post-measurement state. In the DEJMPS protocol, leakage from the main state is minimized by choosing the measurement operator $\{X_1X_2, Z_1Z_2, -Y_1Y_2\}$ based on which of the $\{|\Psi^+\rangle\langle\Psi^+|, |\Phi^-\rangle\langle\Phi^-|, |\Psi^-\rangle\langle\Psi^-|\}$ coefficients of the main state is the smallest—since this coefficient is combined with the $|\Phi^+\rangle\langle\Phi^+|$ coefficient of the ancillary state for each of these operators. Leakage from the ancillary Bell state is reduced by rotating the ancillary Bell state in such a way that its smallest coefficient is the one associated with $|\Psi^+\rangle\langle\Psi^+|$—since this coefficient is always combined with the $|\Phi^+\rangle\langle\Phi^+|$ coefficient of the main state for any of the three operators. This rotation can, *e.g.*, be achieved with H_1H_2 to permute the $|\Psi^+\rangle\langle\Psi^+|$ and $|\Phi^-\rangle\langle\Phi^-|$ coefficients, with $S_1^\dagger S_2$ to permute the $|\Psi^+\rangle\langle\Psi^+|$ and $|\Psi^-\rangle\langle\Psi^-|$ coefficients, or with $R_1^\dagger R_2$ to permute the $|\Psi^-\rangle\langle\Psi^-|$ and $|\Phi^-\rangle\langle\Phi^-|$ coefficients, where $R \equiv R_X(\pi/2) = S^\dagger H S^\dagger$.

3.6.3 Bipartite distillation with error-correction codes

The protocols of Sec. 3.6.2 can be generalized to a larger class of distillation protocols using ideas and concepts from error-correction codes. For the following analysis, we assume two

	$\lvert\psi\rangle_m = \lvert\psi\rangle$	$\lvert\Phi^+\rangle_m$	$\lvert\Psi^+\rangle_m$	$\lvert\Phi^-\rangle_m$	$\lvert\Psi^-\rangle_m$
$\lvert\Phi^+\rangle_a$	X_1X_2 on $\lvert\psi\rangle$	✓ → $\lvert\Phi^+\rangle_m$	✓ → $\lvert\Psi^+\rangle_m$ (*)	✗ → $\lvert\Phi^-\rangle_m$	✗ → $\lvert\Psi^-\rangle_m$
$\lvert\Psi^+\rangle_a$	X_1X_2 on $X_2\lvert\psi\rangle$	✓ → $\lvert\Psi^+\rangle_m$ (*)	✓ → $\lvert\Phi^+\rangle_m$	✗ → $\lvert\Psi^-\rangle_m$	✗ → $\lvert\Phi^-\rangle_m$
$\lvert\Phi^-\rangle_a$	$-X_1X_2$ on $\lvert\psi\rangle$	✗ → $\lvert\Phi^+\rangle_m$	✗ → $\lvert\Psi^+\rangle_m$	✓ → $\lvert\Phi^-\rangle_m$	✓ → $\lvert\Psi^-\rangle_m$
$\lvert\Psi^-\rangle_a$	$-X_1X_2$ on $X_2\lvert\psi\rangle$	✗ → $\lvert\Psi^+\rangle_m$	✗ → $\lvert\Phi^+\rangle_m$	✓ → $\lvert\Psi^-\rangle_m$	✓ → $\lvert\Phi^-\rangle_m$

	$\lvert\psi\rangle_m = \lvert\psi\rangle$	$\lvert\Phi^+\rangle_m$	$\lvert\Psi^+\rangle_m$	$\lvert\Phi^-\rangle_m$	$\lvert\Psi^-\rangle_m$
$\lvert\Phi^+\rangle_a$	Z_1Z_2 on $\lvert\psi\rangle$	✓ → $\lvert\Phi^+\rangle_m$	✗ → $\lvert\Psi^+\rangle_m$	✓ → $\lvert\Phi^-\rangle_m$ (*)	✗ → $\lvert\Psi^-\rangle_m$
$\lvert\Psi^+\rangle_a$	Z_1Z_2 on $Z_2\lvert\psi\rangle$	✓ → $\lvert\Phi^-\rangle_m$ (*)	✗ → $\lvert\Psi^-\rangle_m$	✓ → $\lvert\Phi^+\rangle_m$	✗ → $\lvert\Psi^+\rangle_m$
$\lvert\Phi^-\rangle_a$	$-Z_1Z_2$ on $\lvert\psi\rangle$	✗ → $\lvert\Phi^+\rangle_m$	✓ → $\lvert\Psi^+\rangle_m$	✗ → $\lvert\Phi^-\rangle_m$	✓ → $\lvert\Psi^-\rangle_m$
$\lvert\Psi^-\rangle_a$	$-Z_1Z_2$ on $Z_2\lvert\psi\rangle$	✗ → $\lvert\Phi^-\rangle_m$	✓ → $\lvert\Psi^-\rangle_m$	✗ → $\lvert\Phi^+\rangle_m$	✓ → $\lvert\Psi^+\rangle_m$

	$\lvert\psi\rangle_m = \lvert\psi\rangle$	$\lvert\Phi^+\rangle_m$	$\lvert\Psi^+\rangle_m$	$\lvert\Phi^-\rangle_m$	$\lvert\Psi^-\rangle_m$
$\lvert\Phi^+\rangle_a$	$-Y_1Y_2$ on $\lvert\psi\rangle$	✓ → $\lvert\Phi^+\rangle_m$	✗ → $\lvert\Psi^+\rangle_m$	✗ → $\lvert\Phi^-\rangle_m$	✓ → $\lvert\Psi^-\rangle_m$ (*)
$\lvert\Psi^+\rangle_a$	$-Y_1Y_2$ on $iY_2\lvert\psi\rangle$	✓ → $-\lvert\Psi^-\rangle_m$ (*)	✗ → $\lvert\Phi^-\rangle_m$	✗ → $-\lvert\Psi^+\rangle_m$	✓ → $\lvert\Phi^+\rangle_m$
$\lvert\Phi^-\rangle_a$	Y_1Y_2 on $\lvert\psi\rangle$	✗ → $\lvert\Phi^+\rangle_m$	✓ → $\lvert\Psi^+\rangle_m$	✓ → $\lvert\Phi^-\rangle_m$	✗ → $\lvert\Psi^-\rangle_m$
$\lvert\Psi^-\rangle_a$	Y_1Y_2 on $iY_2\lvert\psi\rangle$	✗ → $-\lvert\Psi^-\rangle_m$	✓ → $\lvert\Phi^-\rangle_m$	✓ → $-\lvert\Psi^+\rangle_m$	✗ → $\lvert\Phi^+\rangle_m$

Table 3.1: Description of performing the circuit of Fig. 2.4b for $N = 2$ to non-locally measure X_1X_2 (top part of the table), Z_1Z_2 (middle part) and $-Y_1Y_2$ (bottom part) on a main state (labeled "m") using ancillary Bell states $\lvert\Phi^\pm\rangle_a$ and $\lvert\Psi^\pm\rangle_a$. The first column denotes the operator that is effectively measured on a general quantum state $\lvert\psi\rangle$ when using each of the ancillary states. The four columns on the right describe what happens for a distillation protocol that measures these operators on a noisy Bell state of the form of Eq. (2.18), and post-selects on a +1 measurement outcome. Here, a mark "✓" means the state is kept in case of a successful measurement outcome, whereas "✗" means the combination of states is removed. The state printed after the arrow describes how the main state is transformed by the operation. No matter what Pauli operator is measured, there is always "leakage" of the two main coefficients (*i.e.*, the coefficients associated with $\lvert\Phi^+\rangle_{m,a}$) to one of the unwanted coefficients (the coefficients that are part of the noise)—these scenarios are marked with "(*)". Performing rotations to minimize this leakage is one of the key ingredients in the DEJMPS entanglement distillation protocol.

parties A and B share N non-perfect Bell pairs between them of the form

$$\left(\mathbb{I}^{\otimes N} \otimes \mathcal{N}_P\right) |\Phi_N^+\rangle, \quad \text{where} \quad |\Phi_N^+\rangle \equiv \frac{1}{\sqrt{2^N}} \sum_{x\in\{0,1\}^N} |x\rangle_A \otimes |x\rangle_B . \tag{3.14}$$

Here, $|\Phi_N^+\rangle$ describes N *perfect* Bell pairs shared between party A and B—this state is often referred to as the *maximally entangled state*. Furthermore, $\mathcal{N}_P$ denotes a noise channel consisting of Pauli errors. Based on the discussion in Sec. 2.4.2, one can understand that a tensor product of N Bell diagonal states can be written in the form of Eq. (3.14).

We first consider a scenario in which (by assumption) party A creates N perfect Bell pairs locally before *sending* one qubit of each pair to party B—*e.g.*, with teleportation. In this scenario, the noise channel $\mathcal{N}_P$ is the result of physically sending half of the qubits from A to B. It means that A has access to the qubits before the noise channel acts and can apply an $[\![N,K,d]\!]$ stabilizer code to protect the qubits from this noise. We discuss in Sec. 3.2.2 how a logical state can be initialized by measuring the code's stabilizer generators on one of the eigenstates of (a representation of) the logical operators that stabilize the target logical state—*i.e.*, by projecting with Π_g in Eq. (3.1). K logical Bell pairs are stabilized by $\overline{X}_j \otimes \overline{X}_j$ and $\overline{Z}_j \otimes \overline{Z}_j$ for $j \in \{1,2,\ldots,K\}$, where the operators $\overline{X}_j$ and $\overline{Z}_j$ are the logical X and logical Z operators of the code, respectively. In this situation, projecting with $\Pi_g \otimes \Pi_g$ on $|\Phi_N^+\rangle$ automatically realizes K encoded Bell pairs regardless of the exact stabilizer code used. This is because, in that case, the following holds, for $P \in \{X,Z\}$ and $s_j \in \{+,-\}$:

$$\begin{aligned} s_j \left(\overline{P}_j \otimes \overline{P}_j\right)\left(\Pi_g \otimes \Pi_g\right)|\Phi_N^+\rangle &= \frac{s_j}{\sqrt{2^N}} \sum_{x\in\{0,1\}^N} \left(\Pi_g \overline{P}_j |x\rangle\right)_A \otimes \left(\Pi_g \overline{P}_j |x\rangle\right)_B \\ &= \left(\Pi_g \otimes \Pi_g\right)|\Phi_N^+\rangle . \end{aligned} \tag{3.15}$$

This can be understood by realizing that a logical operator $\overline{P}_j$ of a stabilizer code is a tensor product of single-qubit Pauli operators. First, single-qubit Pauli-X operators in $\overline{P}_j$ shuffle around the terms $x \in \{0,1\}^N$ simultaneously for both the A and B terms, which leaves the full superposition state invariant. Second, single-qubit Pauli-Z operators in $\overline{P}_j$ acting on $x \in \{0,1\}^N$ possibly introduce a minus sign, but the minus signs of the two parties always cancel against each other. Last, Pauli-Y operators have the property $Y = iXZ$, so they might introduce an overall minus sign for all $x \in \{0,1\}^N$. This minus sign is captured with $s_j = (-1)^{n_j}$, where n_j is the number of Pauli-Y operators in $\overline{P}_j$. In situations where $(\Pi_g \otimes \Pi_g)|\Phi_N^+\rangle$ is stabilized by $-\overline{P}_j \otimes \overline{P}_j$ instead of $+\overline{P}_j \otimes \overline{P}_j$, we technically initialize a logical version of $|\Phi^-\rangle$, $|\Psi^+\rangle$, or $|\Psi^-\rangle$, but this has no influence on the error-correction properties of the code. The CSS stabilizer codes of Sec. 3.2.4 always lead to $s_j = +1$ for all $\overline{P}_j$.

In Fig. 3.17, we see a general distillation circuit associated with this scenario. This protocol was introduced by Glancy *et al.* [28] and Wilde *et al.* [24]. The result of Glancy *et al.* shows that initializing logical Bell states by measuring the stabilizer generators of a stabilizer code on the state of Eq. (3.14) can itself also be considered as part of the distillation process. This is because, in this specific situation, a subset of code generators are always automatically part of the original stabilizer group of $|\Phi_N^+\rangle$. This means that the combined measurement result for this subset can be used to detect errors—*i.e.*, incorrect measurement parities during the initialization process indicates the presence of errors.

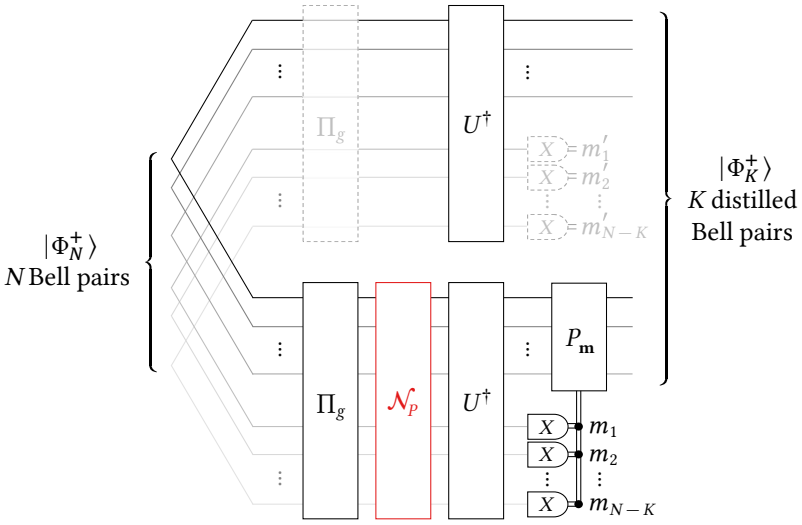


Figure 3.17: General protocol to distill K Bell pairs out of N Bell pairs with an $[\![N,K,d]\!]$ error-correction code in a scenario where a Pauli channel $\mathcal{N}_P$ corresponds to the noise associated with physically sending half of the qubits to the other party. As shown in the main text, logical Bell pairs can be initialized with the projector Π_g. On top of that, projecting with $\mathbb{I}^{\otimes N} \otimes \Pi_g$ on the qubits of one of the two parties is equivalent to projecting with $\Pi_g^T \otimes \Pi_g$ on both parties. Alternatively, instead of encoding with $\Pi_g \otimes \Pi_g$ or $\mathbb{I}^{\otimes N} \otimes \Pi_g = \Pi_g^T \otimes \Pi_g$ on $|\Phi_N^+\rangle$, encoding can be achieved by only initializing the first K pairs in the maximally entangled state $|\Phi_K^+\rangle$ and initializing the other $M = N-K$ pairs in $|+\rangle^{\otimes 2M}$, after which the encoding circuit $U \otimes U$ is applied—see Fig. 3.1 for more details. Measuring the qubits of the top party is only necessary if these qubits also acquire noise. The correction with $P_{\mathbf{m}}$ based on the measurement outcomes $\mathbf{m} = \{m_1, m_2, \ldots, m_{N-K}\}$ works the same as in Fig. 3.3. This distillation protocol is reminiscent of a protocol proposed by Glancy *et al.* [28] and Wilde *et al.* [24].

On top of that, Fig. 3.17 indicates that measuring the stabilizer generators on the qubits of *one* of the parties is sufficient to initialize the code on the qubits of *both* parties. This is possible because, for a maximally entangled state $|\Phi_N^+\rangle$ and a general operation $\mathcal{U}$ on N qubits, we have [23, 24]

$$\left(\mathcal{U}\otimes\mathbb{I}^{\otimes N}\right)|\Phi_N^+\rangle = \left(\mathbb{I}^{\otimes N}\otimes\mathcal{U}^T\right)|\Phi_N^+\rangle. \tag{3.16}$$

Since Π_g is a projector, $\Pi_g\Pi_g = \Pi_g$ holds, and therefore $(\Pi_g\otimes\mathbb{I}^{\otimes N})|\Phi_N^+\rangle = (\Pi_g\otimes\Pi_g^T)|\Phi_N^+\rangle$. The Pauli operators fulfill $X^T = X$, $Z^T = Z$, and $Y^T = -Y$, so the first party initializing a code by projecting with Π_g automatically initializes a code on the second party with the projector

$$\Pi_g^T = \prod_{i=1}^{N-K}\frac{\mathbb{I}^{\otimes N} + s_i g_i}{2}. \tag{3.17}$$

Here, $\{g_i\}_{i=1}^{N-K}$ are the stabilizer generators of the code and $s_i = (-1)^{n_i}$, where n_i represents the number of Pauli-Y operators in a specific generator g_i. We note that just as in the discussion for encoding logical Bell pairs or initializing a logical state by measuring stabilizer generators in general, possible minus signs in front of the stabilizer operators do not influence the error-correction properties of the code. On top of that, CSS codes always have $s_i = +1$ for each stabilizer generator g_i, and therefore always satisfy $\Pi_g^T = \Pi_g$.

3.6.4 Bilocal Clifford protocols

We continue with a distillation scenario in which the two parties generate N noisy Bell pairs between each other in a direct fashion. Even though we can still model these Bell pairs with the expression of Eq. (3.14), the difference with the previous scenario is that the parties can no longer encode the Bell pairs before the noise channel acts. Luckily, we can make use of Eq. (3.16) to effectively apply an error-correction code before the noise channel is applied. We depict the associated general protocol in Fig. 3.18. We interpret Fig. 3.18 as applying U on the qubits of the second party before $\mathcal{N}_P$ acts—the actual situation where the first party applies U^T after the noise channel is equivalent by Eq. (3.16). Because, in the notation of Sec. 3.2.2 and Fig. 3.1a, we decode the qubits of the second party with the second half of $|\Phi_{N-K}^+\rangle$ as the ancillary state $|\phi\rangle$, encoding with $\mathbb{I}^{\otimes N}\otimes U$ transforms the code's stabilizers as

$$\langle X_i\otimes X_i, Z_i\otimes Z_i\rangle_{i=1}^{N-K} \mapsto \langle X_i\otimes UX_iU^\dagger, Z_i\otimes UZ_iU^\dagger\rangle_{i=1}^{N-K}. \tag{3.18}$$

Here, for $P\in\{X,Z\}$, the operators $P_i\otimes P_i$ denote applying P on the ith qubits of the ancillary states of party A and B. It becomes clear that measuring the qubits of the last $M = N-K$ pairs in the Pauli-X basis after decoding with $U^\dagger$ means we only collect the error syndrome for code generators $\{X_i\otimes UX_iU^\dagger\}_{i=1}^M$—*i.e.*, we only measure half of the generators. This, however, still allows us to use the full functionality of the error-correction code, as we assume $\mathcal{N}_P$ only acts on half of the qubits of our Bell pairs: this scheme fully complies with an $[\![N,K,d]\!]$ error-correction code based around $\{X_i\}_{i=1}^M$ as the generators of its ancillary state $|\phi\rangle$ prior to encoding—*i.e.*, with the ancillary state $|\phi\rangle = |+\rangle^{\otimes M}$. If we restrict to using a *stabilizer* error-correction code in this distillation protocol, the circuit U is a Clifford circuit—as explained in Sec. 3.2.2. This is why this class of distillation protocols is sometimes referred to as (N-to-K) *bilocal Clifford protocols* [29].

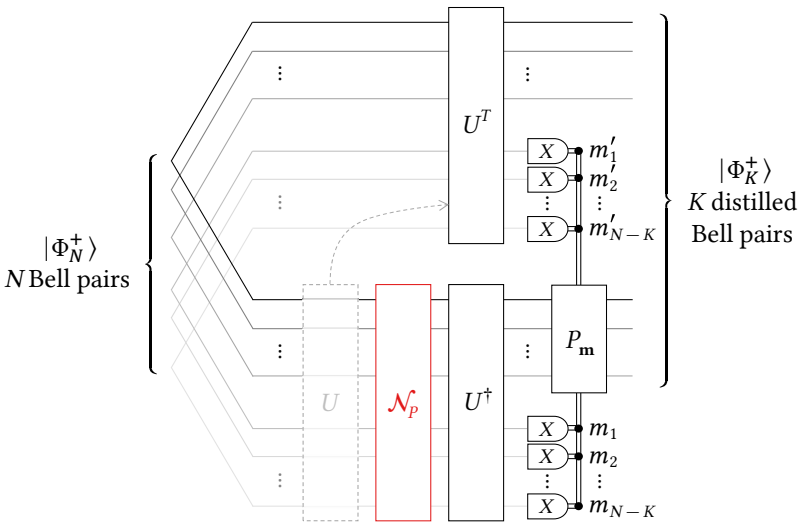


Figure 3.18: General protocol to distill K Bell pairs out of N Bell pairs with an $[\![N,K,d]\!]$ error-correction code in a scenario where a Pauli channel $\mathcal{N}_P$ corresponds to noise associated with directly generating non-perfect Bell pairs between the two involved parties. The two parties apply $U^T \otimes U^\dagger$, where U is the encoding circuit of the error-correction code, before measuring the qubits of the last $N-K$ pairs in the Pauli-X basis. The relation in Eq. (3.16) means that the operation $U^T \otimes \mathbb{I}^{\otimes N}$ is equivalent to applying $\mathbb{I}^{\otimes N} \otimes U$ before the noise channel. The correction with $P_{\mathbf{m}}$ based on the measurement outcomes $\mathbf{m} = \{m'_1 m_1, m'_2 m_2, \ldots, m'_{N-K} m_{N-K}\}$ works the same as in Fig. 3.3. We here have to combine the measurement results $m'_i \in \{+1,-1\}$ and $m_i \in \{+1,-1\}$ because the $M = N-K$ Bell pairs that form the ancillary state $|\phi\rangle$ in the notation of Fig. 3.1a are stabilized by $\langle X_i \otimes X_i \rangle_{i=1}^M$—as well as by $\langle Z_i \otimes Z_i \rangle_{i=1}^M$, but this information is lost after the measurements. This type of distillation protocol is formalized and extensively evaluated in Refs. [29–31].

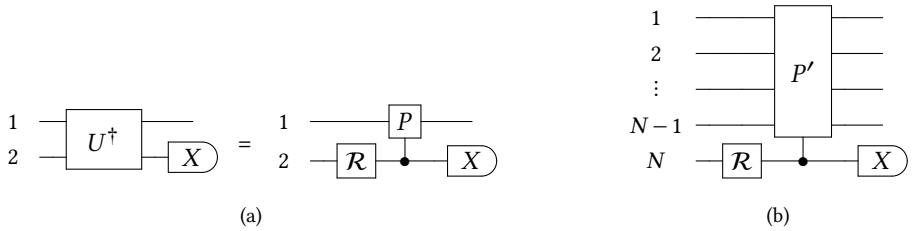


Figure 3.19: (a) Main decoding circuit associated with one step of the DEJMPS distillation protocol, for $\mathcal{R} \in \langle H, S\rangle$ and $P \in \{X, Y, Z\}$. Using this circuit as $U^T \otimes U^\dagger$ in the standard bilocal Clifford distillation protocol of Fig. 3.18 corresponds to measuring the $P^* \otimes P$ parity non-locally with the aid of the measured Bell pair, as shown earlier in Fig. 2.4b. In the context of using $U^T \otimes U^\dagger$ in Fig. 3.18, the $\mathcal{R}_2^* \otimes \mathcal{R}_2$ rotation leaves the $|\Phi^+\rangle\langle\Phi^+|$ state invariant and does not influence the noiseless contribution to the full non-local measurement. (b) Extension of the DEJMPS decoding circuit as a non-local parity measurement on multiple qubits with an $(N-1)$-qubit Pauli operator P'.

As explained in Sec. 3.2.3, typically error-correction codes are not able to find all possible errors, but good codes can detect the *most probable* errors. If we assume that the N non-perfect Bell pairs of Eq. (3.14) are Bell diagonal states of the form of Eq. (2.18), the most likely errors are single-qubit Pauli errors. This makes it understandable that the five-qubit error-correction code of Fig. 3.2 works relatively well in combination with, *e.g.*, the protocol of Fig. 3.18 as a five-to-one distillation protocol. That is because, with this error-correction code, all single-qubit Pauli errors can be identified. If the fidelity of the initial Bell diagonal states is high enough, this protocol is able to deterministically increase the (average) fidelity of the distilled Bell pair [31].

Since quantum error-correction codes are not able to identify single-qubit Pauli errors for $N < 5$, there are also no distillation protocols with a smaller N that can uniquely identify all single-qubit errors in Bell diagonal states. This is why, in distillation, error correction is often paired with probability. For example, if we find a syndrome outcome $\mathbf{m}$ that could be caused by several errors with the same probability of occurring, we could simply decide to discard the post-measurement state, generate the entangled states again, and retry the distillation protocol. In this way, we can describe the BBPSSW and DEJMPS protocols as distillation protocols with $N = 2$ and $K = 1$ of the form of Fig. 3.18. For example, one round of the DEJMPS protocol carries out $U^T \otimes U^\dagger$ with $U^\dagger = CP_{2,1}\mathcal{R}_2$, where $P \in \{X, Z, Y\}$ and $\mathcal{R} \in \langle H, S\rangle$, as depicted in Fig. 3.19a. This protocol only declares success in case the full measurement outcome is $\mathbf{m} = \{m_1' m_1 = +1\}$—*i.e.*, in case no detectable errors are identified. At each of the two parties, the DEJMPS protocol carries out a controlled-P operator with P a single-qubit Pauli operator on the qubits of one other Bell pair. Instead, the measured Bell pair can also be used to measure a multi-qubit Pauli operator P' as a non-local parity measurement on the qubits of multiple Bell pairs. This generalization is depicted in Fig. 3.19b.

On top of that, it is possible to run these protocols in a *concatenated* fashion by consuming multiple Bell pairs to perform multiple non-local parity measurements in series—examples of concatenated-DEJMPS protocols can be seen in Figs. 3.21a and 3.21b. Since every Clifford circuit $U^\dagger$ can be written as a combination of CX, H, and S gates, every N-to-K bilocal Clifford distillation protocol of the form of Fig. 3.18 can in fact be inter-

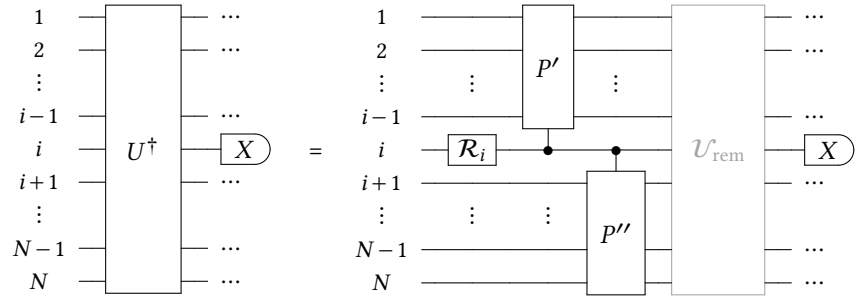


Figure 3.20: Adapting a decoding circuit $U^\dagger$ in such a way that it can be interpreted as using an ancillary state (*i.e.*, qubit i) to non-locally measure a Pauli operator on the remaining states. To realize the form on the right, we make use of the relations in Fig. 2.6a to decompose CY gates into CX and CZ gates, and the commutation relations of Figs. 2.6b-2.6g to move gates to the left or the right. We also use the relation between H gates and CZ and CX gates as presented in Fig. 2.5 to move H gates on qubit i to the right of the circuit. Using these relations, single-qubit rotations on qubit i are moved to the far left of the circuit as $\mathcal{R}_i \in \langle H_i, S_i \rangle$, so that, after this part, there are no more single-qubit rotations on qubit i. We move CP gates—for $P \in \{X, Y, Z\}$—that have their control on qubit i to the left of the circuit as well, so that we end up with a controlled-$(P' \otimes P'')$ operation here. On the right of the circuit, we are left with $\mathcal{U}_{\text{rem}}$ that contains all two-qubit gates that *do not* have their control on qubit i, plus possible single-qubit rotations on qubits $j \neq i$. The intermediate part $\mathcal{U}_{\text{rem}}$, however, is also not allowed to have gates CY with the target on qubit i—these gates have to be decomposed into a CZ gate and a CX gate using Fig. 2.6a, after which the CZ gate has to be moved to the C$P' \otimes$CP'' part of the circuit. This guarantees that the block $\mathcal{U}_{\text{rem}}$ commutes with the Pauli-X measurement on qubit i. In the context of using $U^T \otimes U^\dagger$ for a bilocal Clifford distillation protocol of Fig. 3.18, the rotation $\mathcal{R}_i^* \otimes \mathcal{R}_i$ always leaves the main Bell pair state $|\Phi^+\rangle\langle\Phi^+|$ invariant—possibly while rotating noise contributions. Measuring the qubits i at both parties in the Pauli-X basis after $U^T \otimes U^\dagger$ is therefore equivalent to measuring $P^* \otimes P$ for $P \equiv P' \otimes (\mathcal{R}_i X_i \mathcal{R}_i^\dagger) \otimes P'' = U X_i U^\dagger$.

preted as a concatenation of $N-K$ non-local parity measurements with multi-qubit Pauli operators. To analyze what operator is measured with a circuit $U^T \otimes U^\dagger$ with respect to a specific Bell pair labeled as i, we have to reconfigure $U^\dagger$ according to Fig. 3.20—*i.e.*, by moving gates that do not commute with the Pauli-X measurement on the qubits of the ith Bell pair to the beginning and by collecting all single-qubit rotations on qubits i in front of all two-qubit gates. This decomposition is necessary because gates associated with measurements on other Bell pairs can influence the operator measured with the ith Bell pair. We clarify this with an example of a concatenated-DEJMPS protocol depicted in Fig. 3.21a. Here, we see three rounds of the DJEMPS protocol, without ancillary rotations—*i.e.*, with $\mathcal{R}_4 = \mathcal{R}_3 = \mathcal{R}_2 = \mathbb{I}$. Intuitively, one would think that this circuit measures $-Y_3 \otimes Y_3$, $X_2 \otimes X_2$, and $Z_1 \otimes Z_1$ in three consecutive rounds—*i.e.*, by measuring the qubits of Bell pair 4, 3, and 2 in the Pauli-X basis, respectively, after the CP gates to the other Bell pairs. However, since the first non-local measurement on Bell pair 4 influences the second non-local measurement on Bell pair 3, the second round actually measures $X_2 Z_4 \otimes X_2 Z_4$ instead of $X_2 \otimes X_2$. This can also be understood by realizing that X_4 or Y_4 errors before the circuit acquire Y_3 operators after the gates of the first round, which can be detected with the Pauli-X measurement on the qubits of Bell pair 3. This shows that the parity measurement in a certain round of a concatenated bilocal Clifford protocol could be influenced by previous rounds.

To exemplify the requirement in Fig. 3.20 to move all single-qubit rotations on qubit i to the far left of the circuit, we slightly modify the concatenated-DEJMPS protocol of Fig. 3.21a by adding a rotation $\mathcal{R}_2 = H_2$ before the third round. As can be seen in Fig. 3.21b,

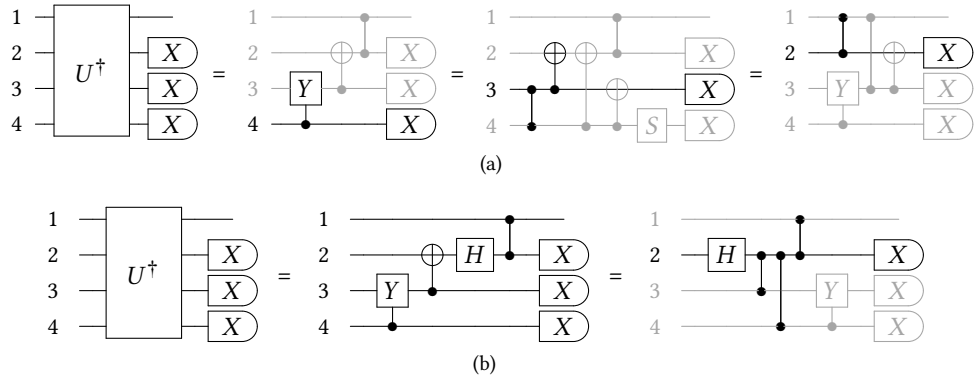


Figure 3.21: (a) Example of three concatenated rounds of the DEJMPS protocol of Fig. 3.19, without ancillary rotation ($\mathcal{R}_4 = \mathcal{R}_3 = \mathcal{R}_2 = \mathbb{I}$). As explained in Fig. 3.20 and the main text, identifying what non-local measurement is carried out with Bell pair i requires moving two-qubit gates with the control on qubit i to the front of the circuit. This is shown here explicitly for the Bell pairs on qubits 4, 3, and 2, respectively. (b) Variation on the three concatenated rounds of the DEJMPS protocol example shown in (a). The rotation $\mathcal{R}_2 = H_2$ in the third round now makes that the non-local measurement with Bell pair 2 measures $Z_1Z_3Z_4 \otimes Z_1Z_3Z_4$ instead of $Z_1 \otimes Z_1$.

this modification means the third round now measures $Z_1Z_3Z_4 \otimes Z_1Z_3Z_4$ instead of $Z_1 \otimes Z_1$, since the additional H gate makes Pauli-X and Pauli-Y errors on qubits 3 and 4 detectable with a Pauli-X measurement on qubit 2.

We emphasize that the operators measured non-locally with a circuit $U^T \otimes U^\dagger$ directly overlap with the stabilizer operators of the error-correction code associated with an encoding circuit U. For example, for the five-qubit error-correction code of Fig. 3.2, we show in Fig. 3.22 how the stabilizer operator associated with the third qubit can be identified by rewriting the decoding circuit in the form of Fig. 3.20. Whereas, as discussed in Sec. 3.2.2, it might be easier to identify stabilizer operators by commuting operators X_i through U, the process described here explains why every bilocal Clifford protocol is, in fact, a concatenation of non-local measurements. Since N Bell pairs $|\Phi_N^+\rangle$ are stabilized by $P^* \otimes P$, for P an N-qubit Pauli operator $P \in \langle X_i, Z_i \rangle_{i=1}^N$, this also directly gives an intuition why Clifford circuits are suitable to use for a distillation protocol in the form of Fig. 3.18. More details regarding the evaluation, identification, and classification of (well-performing) bilocal Clifford protocols can be found in Krastanov *et al.* [30], Zhao *et al.* [32], Jansen *et al.* [29], and Goodenough *et al.* [31].

3.6.5 Double selection protocol

The *double selection protocol* is a three-to-one distillation protocol based on two concatenated rounds of the BBPSSW or DEJMPS protocol that is often more effective than BBPSSW and DEJMPS themselves. This protocol was introduced by Fujii and Yamamoto [33]. Whereas, *e.g.*, two rounds of the DEJMPS protocol could consist of $U^\dagger = CP_{2,1}CZ_{3,2}\mathcal{R}_3$—where we have fixed the gate $CP_{3,2} = CZ_{3,2}$ between pair 3 and 2, and removed the rotation $\mathcal{R}_2 = \mathbb{I}$ on pair 2 before applying $CP_{2,1}$—the idea of double selection is to swap the gates $CZ_{3,2}$ and $CP_{2,1}$ in scenarios like this, and apply $U^\dagger = CZ_{3,2}CP_{2,1}\mathcal{R}_3$ instead. From the perspective of noiseless gates, this leads to the same error-correction code, since these

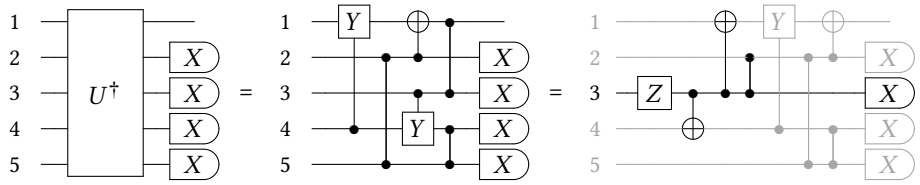


Figure 3.22: Example to show that the non-local measurements with the $U^T \otimes U^\dagger$ protocol of Fig. 3.18 are built up from the stabilizer operators of the error-correction code associated with encoding circuit U. For the decoding circuit of the five-qubit error-correction code, the non-local measurement of $X_1Z_2X_4 \otimes X_1Z_2X_4$ by measuring the Bell pair on qubit 3 in the Pauli-X basis aligns with the stabilizer $-X_1Z_2X_3X_4$ of the code, as presented in the caption of Fig. 3.2. In this comparison, the Pauli-Z rotation on qubit 3 effectively converts the X_3 measurement into a $-X_3$ measurement.

two-qubit gates commute. However, if the gates *are* noisy, exchanging $CZ_{3,2}$ and $CP_{2,1}$ has the advantage that noise originating from $CZ_{3,2}$ does no longer propagate to the first Bell pair [33, 34], which is the distilled Bell pair that is kept after measuring the second and third Bell pair. More specifically, exchanging the two-qubit gates means the probability of noise on the first Bell pair decreases at the expense of an increase in the noise probability on the third Bell pair. However, this is still beneficial, as noise on the third pair has a higher probability of being detected since this Bell pair is measured out.

The main idea of double selection can be expanded to general bilocal Clifford distillation protocols. Here, the main idea is the same: changing the order of the two-qubit gates to minimize the propagation of noise to the states that are not measured out. For four-to-one distillation protocols, specifically, this approach leads to a class of protocols that are sometimes referred to as *triple selection* [30]. Strictly speaking, changing the order of the gates means these protocols can no longer be regarded as stabilizer distillation protocols. However, they *do* still fall within the framework of a bilocal Clifford protocol of the form of Fig. 3.18. The main idea of double selection shows that the analysis in Sec. 3.6.4 is not the full story in scenarios with noisy gates. An example of the double selection distillation protocol (with $\mathcal{R}_3 = \mathbb{I}$) can be found in Block B.1 of Fig. 3.28.

3.6.6 Hashing protocol

The *hashing protocol* [23] distills a set of perfect Bell pairs in a (hypothetical) situation where the initial number of Bell pairs goes to infinity ($N \to \infty$). Using the short-hand notation $|\phi_{00}\rangle \equiv |\Phi^+\rangle$, $|\phi_{01}\rangle \equiv |\Psi^+\rangle$, $|\phi_{10}\rangle \equiv |\Phi^-\rangle$, and $|\phi_{11}\rangle \equiv |\Psi^-\rangle$ introduced in Sec. 2.4.2, we can associate each term in the density matrix of N Bell diagonal states with a classical bit string of $2N$ bits:

$$\begin{aligned} \rho &= \left(\sum_{s_1,s_2 \in \mathbb{Z}_2} p'_{s_1 s_2} \Phi_{s_1 s_2} \right) \otimes \cdots \otimes \left(\sum_{s_{2N-1},s_{2N} \in \mathbb{Z}_2} p'_{s_{2N-1} s_{2N}} \Phi_{s_{2N-1} s_{2N}} \right) \\ &= \sum_{s_1,s_2,\ldots,s_{2N} \in \mathbb{Z}_2} p_{s_1 s_2 \ldots s_{2N}} \Phi_{s_1 s_2} \cdots \Phi_{s_{2N-1} s_{2N}}. \end{aligned} \tag{3.19}$$

Here, $\Phi_{ij} \equiv |\phi_{ij}\rangle\langle\phi_{ij}|$ and $p_{s_1 s_2 \ldots s_{2N}} \equiv p'_{s_1 s_2} p'_{s_3 s_4} \cdots p'_{s_{2N-1} s_{2N}}$. Performing a distillation protocol typically gives us information on these classical bits—*i.e.*, assuming all of the 2^{2N} combinations are present in the initial density matrix of Eq. (3.19), consuming one of the N Bell

pairs in one round of distillation typically "rules out" half of the terms. For example, for just a single Bell diagonal state, measuring X_1X_2 on its two qubits tells us whether the first classical bit is 0 or 1, measuring Z_1Z_2 tells us whether the second bit is 0 or 1, and measuring $-Y_1Y_2$ tells us whether the parity of these two bits is 0 or 1. This principle is the main ingredient of the hashing protocol. Specifically, it makes use of the fact that, for infinite copies of a certain Bell diagonal state, many of the probabilities $p_{s_1s_2\ldots s_{2N}}$ associated with a certain state $\Phi_{s_1s_2}\ldots\Phi_{s_{2N-1}s_{2N}}$ vanish—meaning that the exact state $\Phi_{s_1s_2}\ldots\Phi_{s_{2N-1}s_{2N}}$ can be found without identifying all $2N$ classical bits.

To understand how this works, we first realize that, before measuring out a Bell pair, one can apply bilocal two-qubit gates between this Bell pair and other Bell pairs to "mix" the classical bits of the measured-out pair with bits of other Bell pairs. In this scenario, we consume one Bell pair to measure the parity of a general number of classical bits from the $2N$ bit string describing the N Bell diagonal states. This can be achieved with the bilocal controlled-P gates used in the DEJMPS protocol, for $P\in\{X,Z,iY\}$. For example, for two Bell diagonal states with terms $|\phi_{s_1s_2}\phi_{s_3s_4}\rangle\langle\phi_{s_1s_2}\phi_{s_3s_4}|$, measuring out the qubits of the first Bell pair in the Pauli-X basis after applying bilocal $CX_{1,3}$ and $CX_{2,4}$ gates to the other pair determines the $s_1\oplus s_3$ parity—this corresponds to the first block in Table 3.1. Furthermore, using $CZ_{1,3}$ and $CZ_{2,4}$ gates instead determines $s_1\oplus s_4$, whereas with $CiY_{1,3}$ and $CiY_{2,4}$ gates we obtain the $s_1\oplus s_3\oplus s_4$ parity—these are the second and third blocks in Table 3.1, respectively. If one instead measures the qubits of the measured-out Bell pair in the Pauli-Z basis (iY basis), s_1 is replaced by s_2 (replaced by $s_1\oplus s_2$) in the parity combinations of each of these three scenarios—alternatively, one can use the H_1H_2 and $S_1^\dagger S_2$ rotations of Sec. 3.6.2 on the measured-out Bell pair to include a different combination of its classical bits. This shows that with bilocal $\{CX,CZ,CiY\}$ gates one can simply add bits from another Bell pair to the full parity measurement.

Moreover, applying bilocal two-qubit gates to even more Bell pairs before measuring makes it possible to include any combination of bits from the $2N$ bit string into the parity measurement. This exactly describes the hashing protocol: starting with N Bell pairs, the protocol performs r rounds of parity measurements, where in each round, a random parity of the (remaining) classical bits is determined. Given that we can maximally perform N rounds, whereas (on average) each round only provides us with one bit of information, this protocol is not useful for general input states. However, it *does* become useful in a scenario where one has many copies of the same input (Bell diagonal) state. That is because, with $p'_{s_1s_2}$ the mixed state coefficients of a single copy ρ_{input}, as N becomes bigger and bigger, more and more weight of the probability distribution $p_{s_1s_2\ldots s_{2N}}$ is shifted to strings $s_1s_2\ldots s_{2N}$ that contain Np'_{00} times the bit combination 00, Np'_{01} times the combination 01, Np'_{10} times 10, and Np'_{11} times 11. On the other hand, probabilities $p_{s_1s_2\ldots s_{2N}}$ associated with other bit strings become smaller and smaller. For example, assuming $p'_{00}<1$ holds, the probability associated with $\{s_1,s_2,\ldots,s_{2N}\}=\{0,0,\ldots,0\}$ goes to zero as $N\to\infty$.

This means that the problem is simplified to figuring out which of all possible strings with the most likely distribution of 00, 01, 10, and 11 combinations is the correct one. The set of strings with all permutations of the most likely distribution is often referred to as the *typical string set*. The size of the typical string set determines how many rounds r of the hashing protocol we have to perform, and is given by

$$2^{-N\sum_{s_1,s_2\in\mathbb{Z}_2}p'_{s_1s_2}\log_2 p'_{s_1s_2}}. \tag{3.20}$$

This indicates that strings in the typical set only contain $r = -N \sum_{s_1,s_2 \in \mathbb{Z}_2} p'_{s_1 s_2} \log_2 p'_{s_1 s_2}$ bits of information—explaining why this is the number of rounds r required to find the exact $\Phi_{s_1 s_2} \ldots \Phi_{s_{2N-1} s_{2N}}$ state. Given that, after the full protocol, one is left with $N - r$ remaining Bell pairs, the *hashing rate* [23] of a Bell diagonal state with mixed state coefficients $p'_{s_1 s_2}$ is given by $\lim_{N\to\infty}(N-r)/N = 1 + \sum_{s_1,s_2 \in \mathbb{Z}_2} p'_{s_1 s_2} \log_2 p'_{s_1 s_2}$. This rate corresponds to one minus the *von Neumann entropy* of the input state ρ_{input}.

3.6.7 Multipartite two-to-one stabilizer protocols

Even though more research has been done on bipartite distillation protocols, there is also substantial work on multipartite distillation protocols. Most of these proposals are based on probabilistic, non-local measurements of the target state's stabilizer operators. In this section, we consider a situation with M network parties that each have one qubit of an M-qubit state distributed over the parties. In this context, the most obvious—and for this thesis most relevant—state is the GHZ state $|\text{GHZ}^{(M)}\rangle$, but other states are considered as well. We are interested in distillation protocols where multiple copies of the M-qubit states are available, and a small number of them are consumed with local operations (and classical communication channels) to increase the fidelity of the remaining states. Just as for Bell pair distillation, these protocols are typically probabilistic, as they involve post-selecting on certain measurement outcomes. We start in this section by analyzing two-to-one distillation that can be used in a recurrence protocol. In the next section, we consider N-to-K distillation protocols with an $[\![N, K, d]\!]$ stabilizer code.

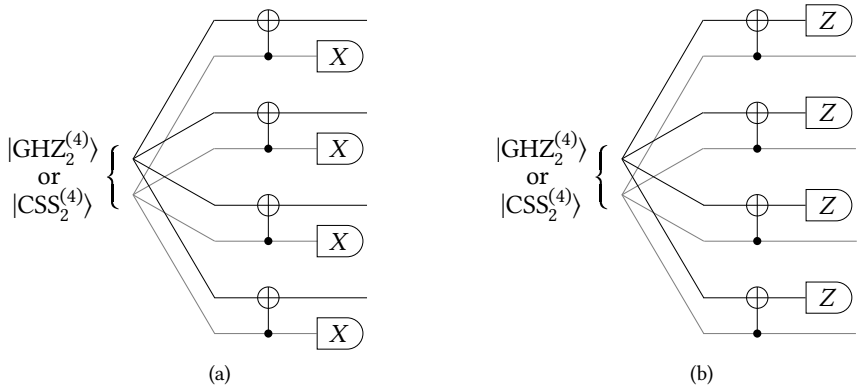


Figure 3.23: Two protocols introduced by Murao *et al.* [35]. The notation $|\text{GHZ}_2^{(4)}\rangle$ and $|\text{CSS}_2^{(4)}\rangle$ indicates two copies of a 4-qubit GHZ or CSS state shared among 4 network parties, with the parties having one qubit of each state. In both protocols, the parties carry out local CX gates between the qubits of the two states. (a) Measuring the second state—*i.e.*, the state containing the control qubits of the CX gates—reveals information about the signs of all combined X-type stabilizer operators between the two states. More details can be found in the main text. Murao *et al.* refer to this protocol as "P1". (b) Measuring the first state—*i.e.*, the state containing the target qubits of the CX gates—reveals information about the signs of the combined Z-type stabilizer operators between the two states. Murao *et al.* refer to this protocol as "P2".

Stabilizer protocols for GHZ and CSS states

In Fig. 3.23, we depict two protocols that can be considered the multipartite version of the DEJMPS protocol from Sec. 3.6.2. Here, CX gates are applied locally between the qubits of the two M-qubit states, and one of the two states is measured out. Both protocols in Fig. 3.23 are introduced by Murao *et al.* [35]. We first analyze the protocols for GHZ states, using the GHZ basis notation of Sec. 2.4.2, with basis states $|\phi_{s_1,s_2,\dots,s_N}\rangle$ as stabilizer states of the operators $(-1)^{s_1} X_1X_2\dots X_N$, $(-1)^{s_2} Z_1Z_2$, $(-1)^{s_3} Z_2Z_3$, …, $(-1)^{s_N} Z_{N-1}Z_N$. By considering how Pauli-X and Pauli-Z operators commute through CX gates, we can understand that the CX gates in the protocols of Fig. 3.23 accomplish the following for two GHZ basis states:

$$|\phi_{s_1,s_2,s_3,\dots,s_N}\rangle \otimes |\phi_{\bar{s}_1,\bar{s}_2,\bar{s}_3,\dots,\bar{s}_N}\rangle \mapsto |\phi_{s_1,s_2\oplus\bar{s}_2,s_3\oplus\bar{s}_3,\dots,s_N\oplus\bar{s}_N}\rangle \otimes |\phi_{s_1\oplus\bar{s}_1,\bar{s}_2,\bar{s}_3,\dots,\bar{s}_N}\rangle \tag{3.21}$$

We see that this transfers the information about the sign of the $X_1X_2\dots X_N$ stabilizer of the first state (*i.e.*, the target of the CX gates) to the second state (*i.e.*, the control of the CX gates). At the same time, the information about the signs of the second state's ZZ stabilizers ends up at the first state. This means that if we now measure the qubits of the second state in the Pauli-X basis (as shown in Fig. 3.23a), we no longer get information about the sign of its own $X_1X_2\dots X_N$ stabilizer but about the parity of the $X_1X_2\dots X_N$ stabilizers of the two states together. Similarly, we can measure the qubits of the first state in the Pauli-Z basis (as shown in Fig. 3.23b) to get information about the combined signs of the ZZ stabilizers of both states. We note that this is just an alternative perspective from how the non-local measurement in Fig. 2.4b works—*e.g.*, we see that the $X_1X_2\dots X_N$ measurement on the first state is exactly a non-local measurement in the style of Fig. 2.4b.

For the same reasons as with, *e.g.*, the DEJMPS protocol, this method is useful as a distillation protocol because it allows one to "filter out" relatively large coefficients in the GHZ basis density representation of a combined state. For example, using the shorthand notation $\Phi_{i_1i_2\dots i_N} \equiv |\phi_{i_1i_2\dots i_N}\rangle\langle\phi_{i_1i_2\dots i_N}|$, $\mathbf{s} \equiv (s_2,\dots,s_N)$ and $\bar{\mathbf{s}} \equiv (s_{N+2},\dots,s_{2N})$, two GHZ diagonal states involved in the protocol transform in the following way after the CX gates:

$$\sum_{s_1,s_2,\dots,s_{2N}\in\mathbb{Z}_2} p_{s_1,\mathbf{s}}\, p_{s_{N+1},\bar{\mathbf{s}}}\, \Phi_{s_1\mathbf{s}}\Phi_{s_{N+1}\bar{\mathbf{s}}} \;\mapsto\; \sum_{s_1,s_2,\dots,s_{2N}\in\mathbb{Z}_2} p_{s_1,\mathbf{s}\oplus\bar{\mathbf{s}}}\, p_{s_1\oplus s_{N+1},\bar{\mathbf{s}}}\, \Phi_{s_1\mathbf{s}}\Phi_{s_{N+1}\bar{\mathbf{s}}}. \tag{3.22}$$

Here, $\mathbf{s}\oplus\bar{\mathbf{s}}$ is element-wise addition modulo 2. If we now, *e.g.*, measure the qubits of the second state in the Pauli-X basis and post-select on states with $s_{N+1}=0$, *i.e.*, states stabilized by $(-1)^{s_{N+1}} X_1X_2\dots X_N = +X_1X_2\dots X_N$, we only leave terms with coefficients $p_{s_1,\mathbf{s}\oplus\bar{\mathbf{s}}}\, p_{s_1,\bar{\mathbf{s}}}$ in the density matrix. Since we assume the GHZ diagonal states to be noisy versions of $\Phi_{\mathbf{00}}$ with $p_{0,\mathbf{0}}$ as the highest coefficient, this means post-selecting on $s_{N+1}=0$ removes relatively high coefficients $p_{0,\mathbf{0}}\, p_{1,\bar{\mathbf{s}}}$ and $p_{1,\mathbf{s}\oplus\bar{\mathbf{s}}}\, p_{0,\mathbf{0}}$. Typically, this increases the contribution of $p_{0,\mathbf{0}}\,\Phi_{\mathbf{00}}$ in the mixture.

As indicated in Fig. 3.23, this protocol can also be applied to the more general class of CSS states [36]. A CSS state is essentially a CSS stabilizer code, as introduced in Sec. 3.2.4, that encodes zero logical qubits. In other words, it is an N-qubit state with N stabilizer generators that fulfill the CSS formalism: each generator is either purely X-type or purely Z-type. The GHZ state is a specific example of a CSS state. The two-to-one stabilizer distillation protocols for CSS states are identical to the GHZ distillation protocol—*i.e.*, they require applying CX gates between the qubits of the two states. Measuring

the qubits holding the CX control in the Pauli-X basis (Fig. 3.23a) allows the network parties to classically reconstruct the parity for each X-type stabilizer of the two CSS states. Alternatively, measuring the qubits holding the CX target in the Pauli-Z basis (Fig. 3.23b) allows the parties to find the combined sign for each Z-type stabilizer.

Stabilizer protocols for two-colorable graph states

Both GHZ states and CSS states are locally equivalent to graph states with a *chromatic number* of two [37, 38]. We introduce graph states in Sec. 2.3.5; the chromatic number (or *chromatic index*) of a graph describes how many colors are required to assign a color to each vertex without two or more adjacent vertices having the same color. The similarities between GHZ states, CSS states, and graph states led to the realization that the protocols of Fig. 3.23 can be extended to graph states. Initially, Dür, Aschauer, and Briegel [36] extended the protocols to the *two-colorable* graph states that are locally equivalent to GHZ and CSS states. Later, Kruszynska *et al.* [39] extended them to general graph states—*i.e.*, graph states with any chromatic number.

To analyze the distillation of graph states, we define another basis: the *graph state basis* [36]. As discussed in Sec. 2.3.5, each qubit i in the graph (*i.e.*, each vertex) is associated with a stabilizer generator $X_i \bigotimes_{j\in\mathbb{N}_i} Z_j$, where $\mathbb{N}_i$ is the neighborhood of the vertex i. This allows us to define basis states $|\psi_{s_1 s_2 \ldots s_N}\rangle$ that, for $i \in \{1, 2, \ldots, N\}$, are stabilized by $(-1)^{s_i} X_i \bigotimes_{j\in\mathbb{N}_i} Z_j$ for a specific N-qubit graph G. Moreover, we assume G has a chromatic number k and add each index s_i to one of the lists $\mathbf{s}_1, \mathbf{s}_2, \ldots, \mathbf{s}_k$ based on the color of the ith qubit in the graph. This defines basis states $|\psi_{\mathbf{s}_1,\mathbf{s}_2,\ldots,\mathbf{s}_k}\rangle$.

As indicated in Fig. 3.24a, GHZ states are specifically two-colorable graph states with Hadamard gates applied to one of the colors of the graph (*i.e.*, to the color associated with the leaves of the graph). The same is true for CSS states. This makes it intuitive that converting the protocols of Fig. 3.23 to a two-to-one distillation protocol on two-colorable graph states requires applying CX gates in one direction between the graph states for qubits in one color, but in the other direction for qubits of the other color. This is because, compared to the GHZ and CSS distillation protocols, the Hadamard gates flip the CX gates between one of the two colors.

The two-to-one distillation protocol for two-colorable graph states is shown schematically in Fig. 3.24b [36, 40]. The combination of CX gates depicted here transforms the basis states of two graph states as

$$|\psi_{\mathbf{s}_1,\mathbf{s}_2}\rangle \otimes |\psi_{\bar{\mathbf{s}}_1,\bar{\mathbf{s}}_2}\rangle \mapsto |\psi_{\mathbf{s}_1,\mathbf{s}_2\oplus\bar{\mathbf{s}}_2}\rangle \otimes |\psi_{\mathbf{s}_1\oplus\bar{\mathbf{s}}_1,\bar{\mathbf{s}}_2}\rangle, \tag{3.23}$$

where $\mathbf{s}_i \oplus \bar{\mathbf{s}}_i$ is again element-wise addition modulo 2. After this combination of CX gates, measuring, *e.g.*, the qubits of the second state allows one to learn the parity $\mathbf{s}_1 \oplus \bar{\mathbf{s}}_1$. Specifically, this requires measuring the qubits of the first color (*i.e.*, the color associated with qubits in $\bar{\mathbf{s}}_1$) in the Pauli-X basis, and the qubits of the second color (*i.e.*, the color associated with qubits in $\bar{\mathbf{s}}_2$) in the Pauli-Z basis. Classically communicating the measurement results then allows the N parties to reconstruct the second state's stabilizers $(-1)^{s_i} X_i \bigotimes_{j\in\mathbb{N}_i} Z_j$ associated with just the first color—*i.e.*, for $s_i \in \bar{\mathbf{s}}_1$. Consequently, just as with Eq. (3.22) above, these measurement results can be used to post-select on states that fulfill $\mathbf{s}_1 \oplus \bar{\mathbf{s}}_1 = \mathbf{0}$ and filter out contributions with relatively high coefficients in a classical mixture of diagonal states in this graph state basis.

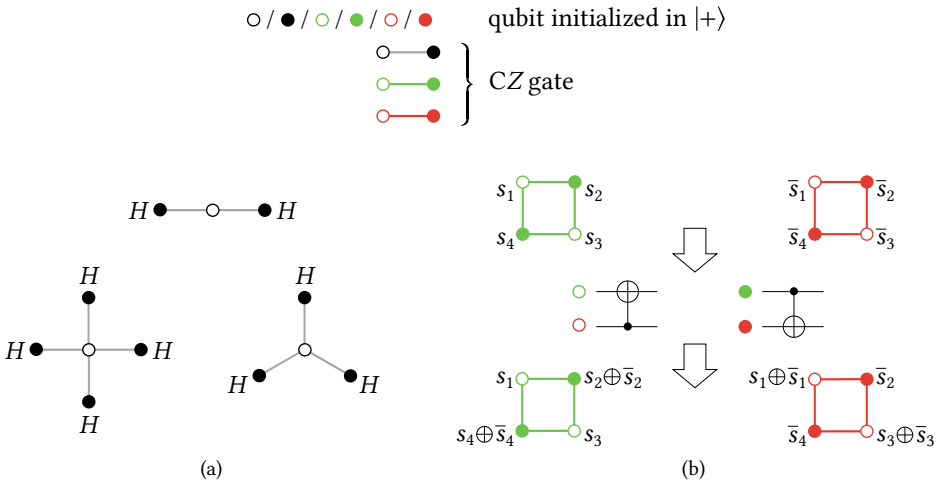


Figure 3.24: (a) A GHZ state $|\mathrm{GHZ}^{(3)}\rangle$ on the top, $|\mathrm{GHZ}^{(4)}\rangle$ on the right, and $|\mathrm{GHZ}^{(5)}\rangle$ on the left, depicted as two-colorable graph states. Any GHZ state can be represented as a star graph, with Hadamard gates applied to the leaves of the graph. (b) Schematic depiction of the two-to-one distillation protocol for two-colorable graph states $|\psi_{\mathbf{s}_1,\mathbf{s}_1}\rangle$ and $|\psi_{\bar{\mathbf{s}}_1,\bar{\mathbf{s}}_2}\rangle$ [36, 40]. After applying CX gates in different directions between the two colors, stabilizer information is transferred between the two states. In the terminology of the main text and Eq. (3.23), $\mathbf{s}_1 = (s_1, s_3)$, $\mathbf{s}_2 = (s_2, s_4)$, $\bar{\mathbf{s}}_1 = (\bar{s}_1, \bar{s}_3)$, and $\bar{\mathbf{s}}_2 = (\bar{s}_2, \bar{s}_4)$, so that the C$X$ gates between the qubits of the two colors lead to $\mathbf{s}_2 \mapsto \mathbf{s}_2 \oplus \bar{\mathbf{s}}_2$ and $\bar{\mathbf{s}}_1 \mapsto \mathbf{s}_1 \oplus \bar{\mathbf{s}}_1$. The joint stabilizers can be measured by consuming one of the two states. For the green state on the left, this requires measuring the qubits of $\mathbf{s}_1$ in the Pauli-Z basis and the qubits of $\mathbf{s}_2$ in the Pauli-X basis. For the red state on the right, this requires measuring the qubits of $\bar{\mathbf{s}}_1$ in the Pauli-X basis and the qubits of $\bar{\mathbf{s}}_2$ in the Pauli-Z basis.

Stabilizer protocols for general graph states

For GHZ states, CSS states, and two-colorable graph states, applying CX gates between two copies of the same state always results in new stabilizer states with the same general stabilizer structure—*i.e.*, it only "mixes" the signs of the states' stabilizers. Unfortunately, this no longer holds for basis states based on k-colorable graph states with $k > 2$: applying CX gates between two copies of these states changes the graphs themselves and requires a different graph state basis after the operation. This is not useful for distillation, as we want our distilled state to keep the same stabilizer structure. In general, next to transferring information about the signs of the stabilizers between the two basis states, applying a CX gate between two graph states adds or removes edges in or between these states. Specifically, as indicated in Fig. 3.25a, a $CX_{i,j}$ gate creates an edge between qubit i and all neighboring qubits of qubit j, or removes such an edge in case it was already there before the operation.

In the protocols discussed above with CX gates—*i.e.*, the protocols for GHZ states, CSS states, and two-colorable graph states—the graph transformations induced by the CX gates happen to cancel each other out. Whereas this is no longer the case for k-colorable graph states, Kruszynska *et al.* [39] realized that it is still possible to distill a k-colorable graph state $|G\rangle$ with a specific two-colorable variant of this state as the ancillary state. We include a schematic depiction of their protocol in Fig. 3.25b. The main idea is that, in a single

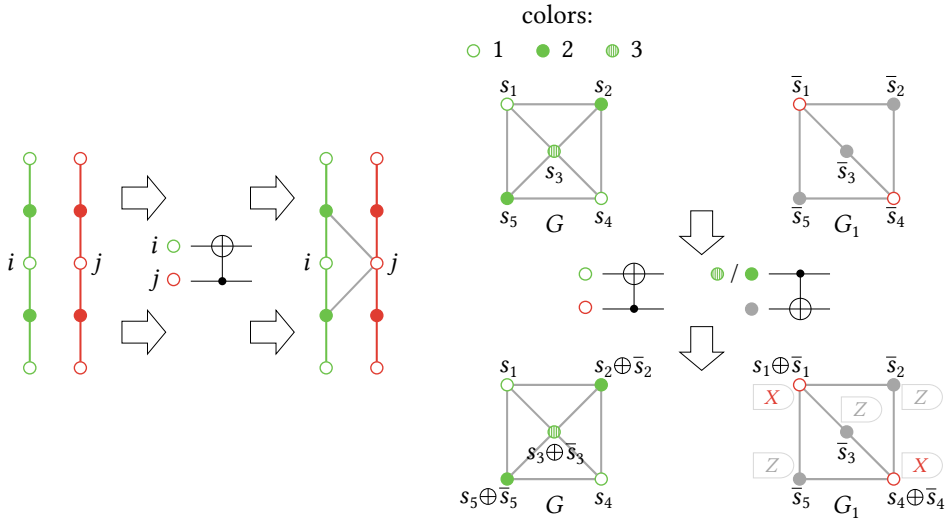


Figure 3.25: (a) Graph state transformation for applying a $CX_{i,j}$ gate between the qubits labeled i and j [39]. The CX gate adds edges between the control qubit and the neighbors of the target qubit—or removes these edges if they were present prior to the CX gate. (b) Schematic depiction of the two-to-one distillation protocol for a k-colorable graph state $|G\rangle = |\psi_{\mathbf{s}_1,\mathbf{s}_2,\mathbf{s}_3}\rangle$ with a two-colorable ancillary graph state $|G_1\rangle = |\psi_{\bar{\mathbf{s}}_1,\bar{\mathbf{s}}_2}\rangle$ [39]. This protocol measures the combined signs of the stabilizers associated with the first color of the k-colorable graph. For the graph states, a similar notation as in Fig. 3.24 is used. In the terminology of the main text and Eq. (3.23), we now have $\mathbf{s}_1 = (s_1, s_4)$, $\mathbf{s}_2 = (s_2, s_5)$, $\mathbf{s}_3 = (s_3)$, and similarly for the ancillary graph on the right. The two-colorable graph G_1 is the graph G with removed edges between colors 2 and 3—*i.e.*, between all colors that are not the main color. Using this two-colorable variant as the ancillary state ensures that the CX graph transformations of (a) cancel against each other. After the CX gates, the parties measure the ancillary state with measurement operators indicated in the figure. Classical communication of the measurement results then allows them to learn the combined signs $\mathbf{s}_1 \oplus \bar{\mathbf{s}}_1 = (s_1 \oplus \bar{s}_1, s_4 \oplus \bar{s}_4)$.

round of the protocol, the parity of the graph state stabilizers associated with one of the colors $i \in \{1, 2, \ldots, k\}$ is determined. This requires an ancillary two-colorable state $|G_i\rangle$ with all edges between colors $j_1 \in \{1, 2, \ldots, k\}$ and $j_2 \in \{1, 2, \ldots, k\}$ removed for $i \neq j_1$ and $i \neq j_2$, but with edges between the qubits of color i and qubits of the other colors in place. In Fig. 3.25b, we show a specific $k = 3$ example with the two-colorable graph G_1 corresponding to the first color of the three-colorable graph G. It can be understood that, using a two-colorable ancillary state of this type, CX gates with the main k-colorable state again cancel against each other and only mix the signs of the stabilizer operators. As indicated in Fig. 3.25b, obtaining information about the combined signs $\mathbf{s}_i \oplus \bar{\mathbf{s}}_i$ of the stabilizers associated with color i requires measuring the ancillary state's qubits of color i in the Pauli-X basis and measuring the other qubits of the ancillary state in the Pauli-Z basis.

Interestingly, the required two-colorable ancillary state $|G_i\rangle$ can be produced with the same CX gate procedure between two regular versions of the main k-colorable graph state $|G\rangle$ [39]. This is shown schematically in Fig. 3.26. In this situation, applying CX gates between two states $|G\rangle$ transforms the graphs—*i.e.*, the edge transformations of Fig. 3.25a

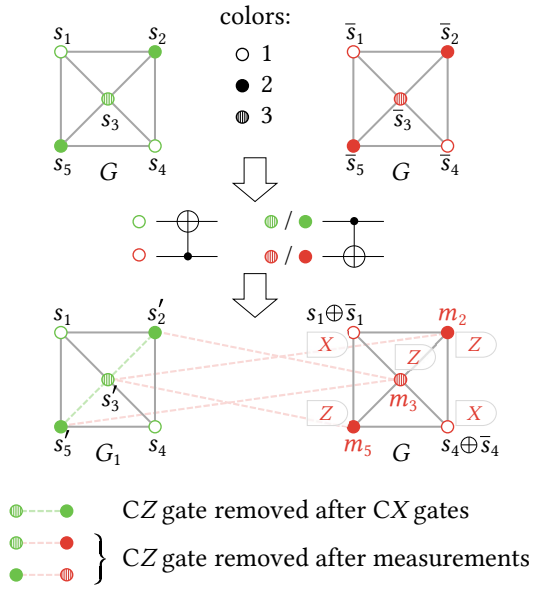


Figure 3.26: Schematic depiction of generating the two-colorable graph state $|G_i\rangle$ required for distilling a k-colorable graph state $|G\rangle$. We show the example with $i = 1$ for a specific graph state $|G\rangle$. The graph states are based on a similar notation as in Fig. 3.24b. After applying the CX gates in the indicated directions, all edges between colors j_1 and j_2 for $i \neq j_1$ and $i \neq j_2$ are removed in the graph state on the left—*i.e.*, the dashed green edges in the figure. On top of that, edges between the two graph states are created—*i.e.*, the dashed red edges. Measuring the qubits of the state on the right with the indicated measurement operators removes these edges between $|G_i\rangle$ and $|G\rangle$ and allows the parties to learn the combined signs $\mathbf{s}_i \oplus \bar{\mathbf{s}}_i$. For qubits of colors $j \neq i$, the measurement results determine the final stabilizers of the state on the left. This is because, since measuring the qubits on the right directly removes the edges between the two graphs, the signs of the stabilizers of colors $j \neq i$ in the left state stay dependent on the measurement outcomes on colors $j \neq i$ on the right. For the specific example shown here, these stabilizer operators acquire the signs $s_2' = s_2 \oplus \bar{s}_2 \oplus m_3$, $s_3' = s_3 \oplus \bar{s}_3 \oplus m_2 \oplus m_5$, and $s_5' = s_5 \oplus \bar{s}_5 \oplus m_3$ after all operations, where $m_2, m_3, m_5 \in \{0, 1\}$ are the measurement outcomes on the indicated qubits on the right.

do no longer cancel. Specifically, as indicated in Fig. 3.26, the CX gates now leave the two graph states entangled but also produce $|G_i\rangle$ in one of the states. It can be understood that measuring the other state $|G\rangle$ with Pauli-X on the qubits of color i and Pauli-Z on the other qubits removes the inter-state edges. Moreover, this also still allows the parties to learn the combined signs $\mathbf{s}_i \oplus \bar{\mathbf{s}}_i$ of the stabilizers associated with color i. Therefore, as a recurrence protocol, this protocol is most effective as a three-to-one distillation protocol. Applying this protocol iteratively allows one to measure the stabilizers associated with different colors of the main graph state.

Since the recurrence protocols discussed in this section use the same principles as the bipartite recurrence protocols, it is not surprising that they can be employed as hashing protocols similar to the bipartite hashing protocol of Sec. 3.6.6. The original protocols by Murao *et al.* in Fig. 3.23 have been extended to a multipartite hashing protocol by Maneva and Smolin [41], with further improvements and extensions to two-colorable graph states introduced by other authors [37, 38, 42, 43]. Heber and Plesch [44] show that using the protocol by Murao *et al.* to distill multipartite states is more efficient than locally producing

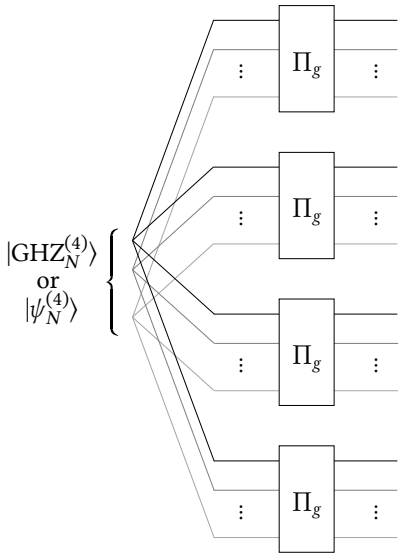


Figure 3.27: Schematic depiction of a multipartite entanglement distillation protocol considered by Glancy *et al.* [28]. Here, $|\mathrm{GHZ}_N^{(4)}\rangle$ refers to N copies of a $|\mathrm{GHZ}^{(4)}\rangle$ state—see Eq. (3.24) for more details. The operation Π_g refers to the 4 parties locally measuring the stabilizers of a stabilizer code. For the $|\mathrm{GHZ}_N^{(4)}\rangle$ state, measuring the stabilizers of an $[\![N,K,d]\!]$ CSS code initializes K logical GHZ states. Glancy *et al.* show that the protocol also works for other combinations of initial state $|\psi_N^{(4)}\rangle$ and stabilizer codes—see the main text for more details. Furthermore, here we show the protocol for N copies of a 4-qubit state, but the same protocol can be extended to N copies of an M-qubit state.

multipartite states and teleporting these states to other parties with purified Bell pairs. Riera-Sàbat *et al.* [45] consider multipartite recurrence protocols with high-dimensional entangled qudit ancillary states. A review of all early multipartite (and bipartite) distillation protocols can be found in Dür and Briegel [46].

3.6.8 Multipartite distillation with error-correction codes

Just as in the bipartite case (as discussed in Secs. 3.6.3 and 3.6.4), we can also more directly apply quantum error-correction codes to multipartite distillation. Fig. 3.27 shows a scenario with N copies of an M-qubit entangled state. The states are distributed over M parties, with each party having one qubit of each state. Each party then locally applies an $[\![N,K,d]\!]$ error-correction code on their qubits by measuring the stabilizer generators of the code—*i.e.*, by projecting with Π_g from Eq. (3.1).

The protocol of Fig. 3.27 is considered in seminal work by Glancy *et al.* [28]. We mention their work in Sec. 3.6.3 above in relation to a less general version with N Bell pairs. They analyze what combinations of states and codes work for encoding N copies of a shared M-qubit state by locally measuring the generators of an $[\![N,K,d]\!]$ stabilizer code. Specifically, they consider three types of states and codes: CSS, CSS-H, and general. We introduce CSS codes and CSS states in Sec. 3.2.4 and the first part of Sec. 3.6.7, respectively. CSS-H states (codes) are CSS states (codes) that are invariant under Hadamard transformations on all qubits—*i.e.*, the stabilizers (and logical operators) of these states (codes) do not change when

Hadamard gates are applied on all qubits, but a logical Z operator might transform to a different logical X operator as the one it anti-commutes with. (This makes CSS-H codes the more general version of the self-dual CSS codes discussed in Sec. 3.4.7.) Glancy *et al.* show that the protocol of Fig. 3.27 works for the combination of CSS-H states with general stabilizer codes. Since Bell pairs are CSS-H states, this explains why the Bell pair protocol of Sec. 3.6.3 works for all stabilizer codes. Moreover, the authors show that CSS states can be encoded by measuring the generators of CSS codes, and general states can be encoded with CSS-H codes. For these combinations of states and stabilizer codes, similarly as for the protocol in Sec. 3.6.3, a subset of code stabilizers is already present before initialization as stabilizer operators of the N shared copies of the M-qubit state. Therefore, the results of the generator measurements for encoding can be used to detect errors in the initial states.

To apply these results in a more concrete setting, we consider the protocol with N copies of an M-qubit GHZ state shared among M network parties. Since the GHZ state is a CSS state, it can be encoded with the generators of a CSS code. To show this, we use a similar notation as in Eq. (3.14) for N copies of Bell pairs, and write the full state of N copies of the M-qubit GHZ state as

$$|\mathrm{GHZ}_N^{(M)}\rangle \equiv \frac{1}{\sqrt{2^N}} \sum_{x\in\{0,1\}^N} |x\rangle_1 \otimes |x\rangle_2 \otimes \cdots \otimes |x\rangle_M = \frac{1}{\sqrt{2^N}} \sum_{x\in\{0,1\}^N} |x\rangle^{\otimes M}. \tag{3.24}$$

Just as in the bipartite scenario, we can show that the stabilizer operators of the logical GHZ states stabilize $\Pi_g^{\otimes M}|\mathrm{GHZ}_N^{(M)}\rangle$, *i.e.*, the state after each party projects their N qubits on the eigenspace of the stabilizer code. Specifically, for every logical combination of $\overline{X}_j$ and $\overline{Z}_j$ operators of the stabilizer code, the stabilizers of the logical GHZ states are generated by $\overline{X}_j \otimes \cdots \otimes \overline{X}_j = \overline{X}_j^{\otimes M}$ on the qubits of all parties and $\overline{Z}_j\overline{Z}_j$ on the qubits of two of the involved parties. These logical operators transform the encoded state in the following way:

$$\begin{aligned} \overline{X}_j^{\otimes M}\Pi_g^{\otimes M}|\mathrm{GHZ}_N^{(M)}\rangle &= \Pi_g^{\otimes M}\frac{1}{\sqrt{2^N}}\sum_{x\in\{0,1\}^N}\left(\overline{X}_j|x\rangle\right)^{\otimes M}. \\ \left(\overline{Z}_j^{\otimes 2}\otimes\mathbb{1}\right)\Pi_g^{\otimes M}|\mathrm{GHZ}_N^{(M)}\rangle &= \Pi_g^{\otimes M}\frac{1}{\sqrt{2^N}}\sum_{x\in\{0,1\}^N}\left(\overline{Z}_j|x\rangle\right)^{\otimes 2}\otimes|x\rangle^{\otimes(M-2)}. \end{aligned} \tag{3.25}$$

In the second expression, $\mathbb{1} = \mathbb{I}^{\otimes N(M-2)}$ is the identity operation on all qubits of the last $M-2$ parties. We only show one of the $\overline{Z}_j\overline{Z}_j$ logical stabilizers, but the same argument holds for the other $M-1$ logical operators of the type $\overline{Z}_j\overline{Z}_j$. Eq. (3.25) indicates why CSS codes are required for the encoded state to be a valid representation of K logical GHZ states—at least for M an odd number. The logical X (Z) operators in CSS codes are purely X-type (Z-type). For such a logical operator $\overline{Z}_j$, the term $\overline{Z}_j|x\rangle$ on the second line of Eq. (3.25) always results in $\pm|x\rangle$, with a possible minus sign canceling out against the second $\overline{Z}_j|x\rangle$ term in the expression. For a purely X-type operator $\overline{X}_j$, the term $\overline{X}_j|x\rangle$ only shuffles around the terms x in the sum over $x\in\{0,1\}^N$. Therefore, for CSS codes, both expressions in Eq. (3.25) result in $\Pi_g^{\otimes M}|\mathrm{GHZ}_N^{(M)}\rangle$. On the other hand, for a non-CSS code, $\overline{X}_j$ can (additionally) have Pauli-Z or Pauli-Y operators. In that case, for M an odd number, the terms $\overline{X}_j|x\rangle$ produce inconsistent prefactors in the full superposition sum of $|x\rangle$ over $x\in\{0,1\}^N$ after the transformation.

Similar to the Bell pair protocol discussed in Sec. 3.6.3, the multipartite protocol discussed here is mostly relevant in situations where one party generates and encodes all states, and (the most dominant source of) noise is caused by teleportation of the states to the other $M-1$ parties. The bilocal Clifford protocols of Sec. 3.6.4, which are also suitable in a scenario where the states are directly generated between the parties, do unfortunately not convert straightforwardly to the multipartite situation. In the first place, this is because it is less convenient to express (multiple copies of) a GHZ diagonal state as a Pauli channel acting on the state in Eq. (3.24)—*i.e.*, not as convenient as the format used in Eq. (3.14) for Bell pairs. That is because, producing the GHZ basis states out of the main state $|\text{GHZ}^{(M)}\rangle$ requires, *e.g.*, Pauli-Z gate on the qubit of the first party and Pauli-X gates on the qubits of the other $M-1$ parties. Next to that, the bipartite transpose relation in Eq. (3.16) does not directly hold with, *e.g.*, $|\text{GHZ}_N^{(M)}\rangle$ as the entangled state. This is also why, contrary to the bipartite situation, one party encoding its qubits with Π_g in Fig. 3.27 does not automatically initialize the same stabilizer code on the other parties.

Even though a multipartite version of the transpose relation of Eq. (3.16) is more challenging to derive than the bipartite version, Rengaswamy *et al.* [47] managed to identify it for GHZ states. For the protocol of Fig. 3.27 with $|\text{GHZ}_N^{(M)}\rangle$ for $M=3$, they show that one party initializing an $[\![N,K,d]\!]$ code results in a $[\![2N,K,d']\!]$ code on the qubits of the other two parties. Consequently, if the $[\![N,K,d]\!]$ code of the first party is a CSS code, initialization of the second party with this $[\![N,K,d]\!]$ code automatically implements the same $[\![N,K,d]\!]$ on the qubits of the remaining party. If this $[\![N,K,d]\!]$ code is not a CSS code, applying a diagonal Clifford transformation on the third party's qubits after initialization by the first two parties also implements the code on the third party.

The work of Glancy *et al.* has inspired other (multipartite) distillation protocols—*e.g.*, a protocol for stabilizer state *breeding* by Hostens *et al.* [48]. Breeding is related to hashing: it focuses on how many (pre-distilled) perfect copies of a state are required in the asymptotic limit to convert a (larger) number of noisy initial copies into (near-)perfect copies.

3.6.9 GHZ creation and distillation with Bell pairs

In this thesis, we are specifically interested in protocols that generate GHZ states by combining Bell pairs and performing distillation with either Bell pairs or the created GHZ states. Here, we introduce prior work on this protocol class and its operations. In Ch. 5 we consider creating GHZ states from Bell pairs with Block C.1 in Fig. 3.28. This operation is inspired by the approach of Block C.2 in Fig. 3.28, introduced by Nickerson [34] to generate $|\text{GHZ}^{(4)}\rangle$ by using a Bell pair $|\Phi^+\rangle$ to non-locally measure Z_1Z_3, Z_1Z_4, Z_2Z_3, or Z_2Z_4 on the state $|\Phi^+\rangle\otimes|\Phi^+\rangle$—*i.e.*, by using the circuit of Fig. 2.4b to combine three Bell pairs into a 4-qubit GHZ state. Furthermore, the use of Bell pairs as efficient resources for the distillation of multipartite entangled states, as discussed in, *e.g.*, Ch. 5 of this thesis, is also recognized in work by Krastanov, Sanchez de la Cerda, and Narang [49]. Conceptually, deploying Bell pairs in this context is related to creating [50] and distilling [51] entangled states with Bell state measurements, even though the specific operations differ.

For each GHZ generation protocol with Bell pairs, we identify two parameters. The first one is the minimum number of Bell pairs K required to create the GHZ state. This number indicates the amount of distillation taking place in the protocol, and we sometimes use it in this thesis to directly indicate the number of distillation steps used in a protocol—see

Sec. 6.2 for more details. The second parameter is the maximum number of qubits per node Q necessary to generate the GHZ state.

Nickerson *et al.* [52, 53] introduce five protocols for generating weight-4 GHZ states from Bell pairs. *Expedient* and *Stringent* are the result of a brute-force search over protocols following a concrete multi-step structure [52]. In Sec. 5.4 and Fig. 5.2, we depict simplifications of these protocols for a situation without gate noise. In Fig. 3.29, we show a representation of the full circuit diagram for the Expedient protocol, using the short-hand notation of Fig. 3.28. The protocol first generates and distills Bell pairs between neighboring nodes in one direction of the network. Subsequently, it generates and distills Bell pairs between neighbors in the other direction, after which it simultaneously combines the Bell pairs into a GHZ state and performs distillation by non-locally measuring a ZZ stabilizer operator of the GHZ state. If none of the distillation steps fail, the Expedient protocol uses $K = 22$ Bell pairs. The Stringent protocol has the same overall structure as Expedient, but applies more distillation steps, and uses a minimum of $K = 42$ Bell pairs.

The Expedient and Stringent are designed to only use $Q = 3$ qubits per node to generate the GHZ state—*i.e.*, one qubit to hold the final GHZ state, and two qubits to store intermediate states. In Ref. [53], Nickerson *et al.* introduce another protocol with $Q = 3$ that uses $K = 8$ Bell pairs to generate a weight-4 GHZ state: the *Basic* protocol. On top of that, they introduce the *Medium* and *Refined* protocols. These protocols follow the same overall structure as the Basic protocol, but use $Q = 4$ and $Q = 5$ qubits per node, respectively, and a minimum of $K = 16$ and $K = 40$ Bell pairs.

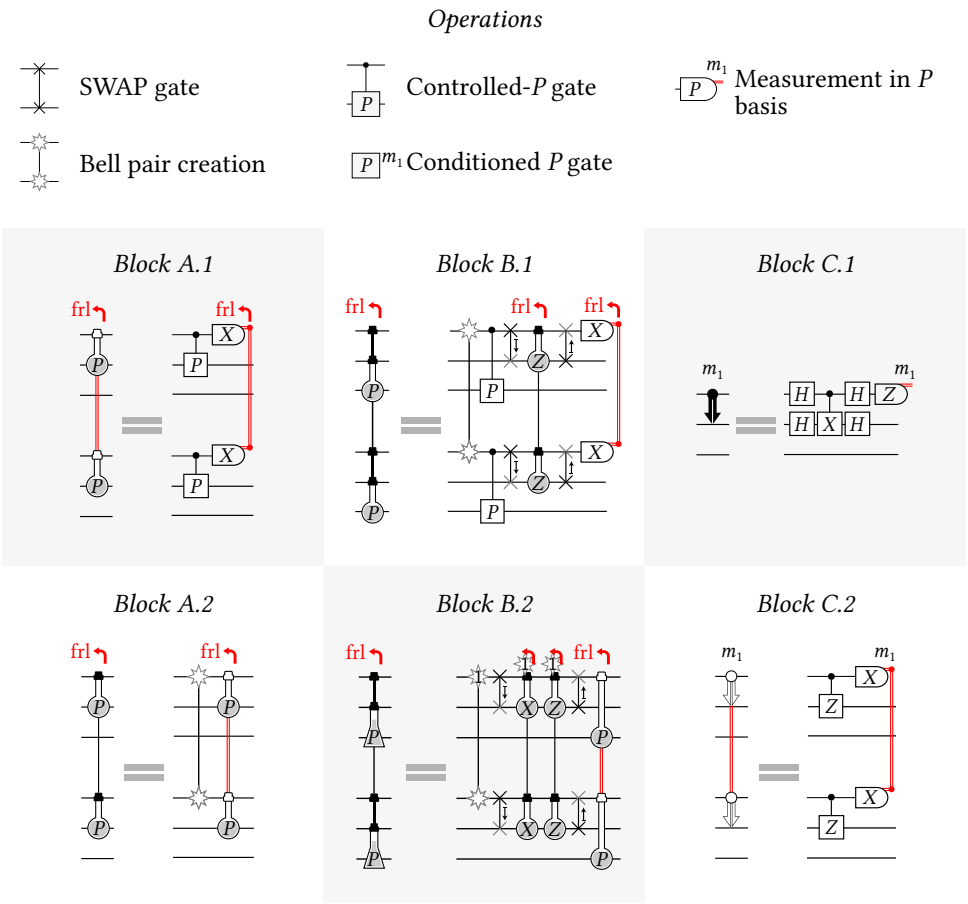


Figure 3.28: Legend for the short-hand symbols and notation used in Figs. 3.29 and 6.2. Blocks A.1 and A.2 contain versions of the DEJMPS distillation protocol of Sec. 3.6.2. Block B.1 describes versions of the double selection distillation protocol of Sec. 3.6.5. Block B.2 describes a four-to-one distillation protocol based on three concatenated steps of the DEJMPS protocol. The distillation protocols used in these blocks are of the probabilistic kind, where success is only declared in case of coinciding measurement outcomes on the two qubits of the same pair. In case of a non-coinciding measurement combination, the distillation operation fails, and the protocol returns to the indicated *failure reset level* (abbreviated as "frl"). In Blocks B.1 and B.2, the SWAP gates are added in anticipation of later chapters in this thesis—see Sec. 4.1 for more details. These SWAP gates are not fundamental elements of these protocols and can be circumvented in systems with more operational freedom compared to the systems we consider. Lastly, Blocks C.1 and C.2 describe operations that combine Bell or GHZ states to create bigger entangled states. Block C.1 is the fusion operation of Chs. 5 and 7. Block C.2 can be used to perform a non-local Pauli measurement of the form of Fig. 2.4b. The block is, *e.g.*, able to transform two Bell pairs $(|00\rangle+|11\rangle)/\sqrt{2}\otimes(|00\rangle+|11\rangle)/\sqrt{2}$ into a GHZ state $(|0000\rangle+|1111\rangle)/\sqrt{2}$ by using a third Bell pair to measure ZZ on two qubits from the different Bell pairs—*e.g.*, Z_1Z_3, Z_1Z_4, Z_2Z_3, or Z_2Z_4.

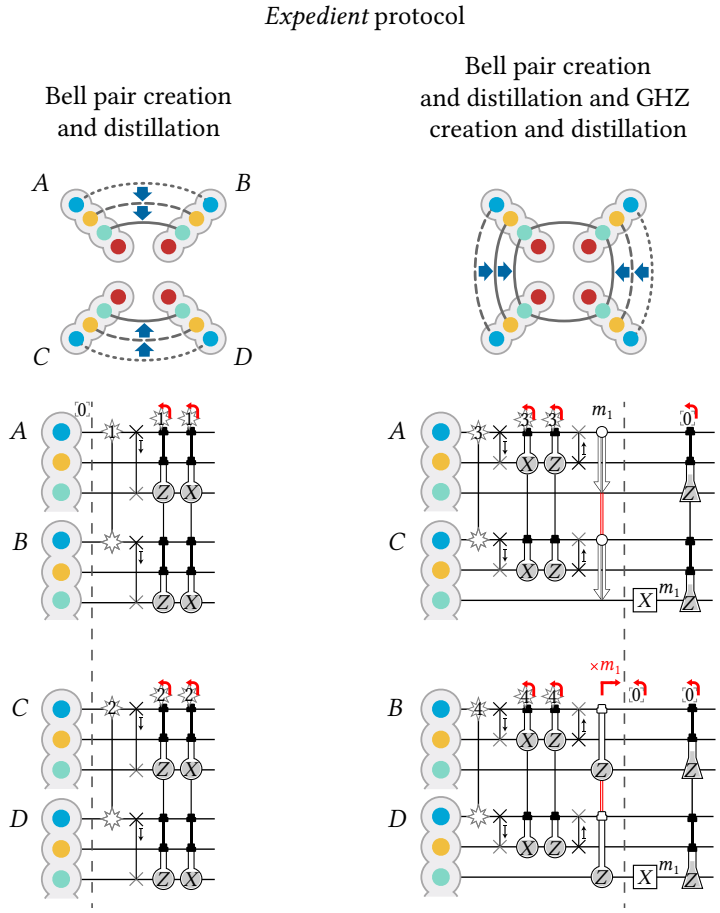


Figure 3.29: Schematic representation of the Expedient entanglement generation protocol [52]. The protocol combines Bell pairs into a weight-4 GHZ state and performs distillation along the way. More information about the notation in these diagrams can be found in Fig. 3.28. The red arrows denote failure reset levels and describe to where in the protocol we have to reset in case of a failed distillation attempt. Depending on where in the circuit the distillation steps occur, failure of a step means the entire protocol has to start from scratch, or only a specific branch of the protocol has to be carried out again. Depending on how many distillation steps fail during execution, this protocol uses a minimum of 22 Bell pairs to generate the GHZ state. In these circuits, just as in Fig. 3.28, SWAP gates are included to comply with later chapters in this thesis. These SWAP gates are not fundamental elements of the Expedient protocol.

References

[1] D. Gottesman, *Stabilizer Codes and Quantum Error Correction*. Doctoral thesis, California Institute of Technology, Pasadena, California, United States, May 1997.

[2] P. W. Shor, "Scheme for reducing decoherence in quantum computer memory," *Physical Review A*, vol. 52, pp. R2493–R2496, Oct. 1995.

[3] A. R. Calderbank and P. W. Shor, "Good quantum error-correcting codes exist," *Physical Review A*, vol. 54, pp. 1098–1105, Aug. 1996.

[4] A. Steane, "Multiple-particle interference and quantum error correction," *Proceedings of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences*, vol. 452, pp. 2551–2577, Jan. 1997.

[5] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information*. Cambridge University Press, 2000.

[6] J. Preskill, *Lecture Notes: Physics 219/Computer Science 219 Quantum Computation*. California Institute of Technology, Pasadena, United States of America, 2006.

[7] R. Laflamme, C. Miquel, J. P. Paz, and W. H. Zurek, "Perfect quantum error correcting code," *Physical Review Letters*, vol. 77, pp. 198–201, July 1996.

[8] A. Mondal and K. K. Parhi, "Quantum circuits for stabilizer error correcting codes: A tutorial," *IEEE Circuits and Systems Magazine*, vol. 24, no. 1, pp. 33–51, 2024.

[9] K. Fujii, *Quantum Computation with Topological Codes: From Qubit to Topological Fault-Tolerance*, vol. 8 of *SpringerBriefs in Mathematical Physics*. Singapore: Springer, 2015.

[10] D. Browne, *Lectures on Topological Codes and Quantum Computation*. University of Innsbruck, Innsbruck, Germany, June 2014.

[11] M. Newman, L. A. de Castro, and K. R. Brown, "Generating fault-tolerant cluster states from crystal structures," *Quantum*, vol. 4, p. 295, July 2020.

[12] A. G. Fowler, M. Mariantoni, J. M. Martinis, and A. N. Cleland, "Surface codes: Towards practical large-scale quantum computation," *Physical Review A*, vol. 86, p. 032324, Sept. 2012.

[13] D. Horsman, A. G. Fowler, S. Devitt, and R. V. Meter, "Surface code quantum computing by lattice surgery," *New Journal of Physics*, vol. 14, p. 123011, Dec. 2012.

[14] B. Eastin and E. Knill, "Restrictions on transversal encoded quantum gate sets," *Physical Review Letters*, vol. 102, p. 110502, Mar. 2009.

[15] E. Knill, *Fault-Tolerant Postselected Quantum Computation: Schemes*. arXiv: quant-ph/0402171, Feb. 2004.

[16] S. Bravyi and A. Kitaev, "Universal quantum computation with ideal Clifford gates and noisy ancillas," *Physical Review A*, vol. 71, p. 022316, Feb. 2005.

[17] A. G. Fowler and C. Gidney, *Low Overhead Quantum Computation Using Lattice Surgery*. arXiv: 1808.06709 [quant-ph], Aug. 2019.

[18] A. Bolt, G. Duclos-Cianci, D. Poulin, and T. M. Stace, "Foliated quantum error-correcting codes," *Physical Review Letters*, vol. 117, p. 070501, Aug. 2016.

[19] N. Nickerson and H. Bombín, *Measurement Based Fault Tolerance beyond Foliation*. arXiv: 1810.09621 [quant-ph], Oct. 2018.

[20] H. Bombín, I. H. Kim, D. Litinski, N. Nickerson, M. Pant, F. Pastawski, S. Roberts, and T. Rudolph, *Interleaving: Modular Architectures for Fault-Tolerant Photonic Quantum Computing*. arXiv: 2103.08612 [quant-ph], Mar. 2021.

[21] H. Bombín, C. Dawson, R. V. Mishmash, N. Nickerson, F. Pastawski, and S. Roberts, "Logical blocks for fault-tolerant topological quantum computation," *PRX Quantum*, vol. 4, p. 020303, Apr. 2023.

[22] R. Raussendorf, J. Harrington, and K. Goyal, "Topological fault-tolerance in cluster state quantum computation," *New Journal of Physics*, vol. 9, p. 199, June 2007.

[23] C. H. Bennett, D. P. DiVincenzo, J. A. Smolin, and W. K. Wootters, "Mixed-state entanglement and quantum error correction," *Physical Review A - Atomic, Molecular, and Optical Physics*, vol. 54, no. 5, pp. 3824–3851, 1996.

[24] M. M. Wilde, H. Krovi, and T. A. Brun, "Convolutional entanglement distillation," in *2010 IEEE International Symposium on Information Theory*, pp. 2657–2661, June 2010.

[25] A. M. Kubica, *The ABCs of the Color Code: A Study of Topological Quantum Codes as Toy Models for Fault-Tolerant Quantum Computation and Quantum Phases Of Matter*. Doctoral thesis, California Institute of Technology, Pasadena, California, United States, 2018.

[26] C. H. Bennett, G. Brassard, S. Popescu, B. Schumacher, J. A. Smolin, and W. K. Wootters, "Purification of noisy entanglement and faithful teleportation via noisy channels," *Physical Review Letters*, vol. 76, pp. 722–725, Jan. 1996.

[27] D. Deutsch, A. Ekert, R. Jozsa, C. Macchiavello, S. Popescu, and A. Sanpera, "Quantum privacy amplification and the security of quantum cryptography over noisy channels," *Physical Review Letters*, vol. 77, no. 13, pp. 2818–2821, 1996.

[28] S. Glancy, E. Knill, and H. M. Vasconcelos, "Entanglement purification of any stabilizer state," *Physical Review A*, vol. 74, p. 032319, Sept. 2006.

[29] S. Jansen, K. Goodenough, S. de Bone, D. Gijswijt, and D. Elkouss, "Enumerating all bilocal Clifford distillation protocols through symmetry reduction," *Quantum*, vol. 6, p. 715, May 2022.

[30] S. Krastanov, V. V. Albert, and L. Jiang, "Optimized entanglement purification," *Quantum*, vol. 3, p. 123, Feb. 2019.

[31] K. Goodenough, S. de Bone, V. Addala, S. Krastanov, S. Jansen, D. Gijswijt, and D. Elkouss, "Near-term n to k distillation protocols using graph codes," *IEEE Journal on Selected Areas in Communications*, vol. 42, pp. 1830–1849, July 2024.

[32] X. Zhao, B. Zhao, Z. Wang, Z. Song, and X. Wang, "Practical distributed quantum information processing with LOCCNet," *npj Quantum Information*, vol. 7, pp. 1–7, Nov. 2021.

[33] K. Fujii and K. Yamamoto, "Entanglement purification with double selection," *Physical Review A*, vol. 80, p. 042308, Oct. 2009.

[34] N. Nickerson, *Practical Fault-Tolerant Quantum Computing*. Doctoral thesis, Imperial College London, London, United Kingdom, 2015.

[35] M. Murao, M. B. Plenio, S. Popescu, V. Vedral, and P. L. Knight, "Multiparticle entanglement purification protocols," *Physical Review A - Atomic, Molecular, and Optical Physics*, vol. 57, no. 6, pp. R4075–R4078, 1998.

[36] W. Dür, H. Aschauer, and H. J. Briegel, "Multiparticle entanglement purification for graph states," *Physical Review Letters*, vol. 91, Sept. 2003.

[37] E. Hostens, J. Dehaene, and B. De Moor, "Hashing protocol for distilling multipartite Calderbank-Shor-Steane states," *Physical Review A*, vol. 73, p. 042316, Apr. 2006.

[38] K. Chen and H.-K. Lo, *Multi-Partite Quantum Cryptographic Protocols with Noisy GHZ States*. arXiv: quant-ph/0404133, Apr. 2008.

[39] C. Kruszynska, A. Miyake, H. J. Briegel, and W. Dür, "Entanglement purification protocols for all graph states," *Physical Review A*, vol. 74, p. 052316, Nov. 2006.

[40] K. Goyal, A. McCauley, and R. Raussendorf, "Purification of large bicolorable graph states," *Physical Review A*, vol. 74, p. 032318, Sept. 2006.

[41] E. N. Maneva and J. A. Smolin, "Improved two-party and multi-party purification protocols," *Contemporary Mathematics*, vol. 305, pp. 203–212, 2002.

[42] H. Aschauer, W. Dür, and H.-J. Briegel, "Multiparticle entanglement purification for two-colorable graph states," *Physical Review A*, vol. 71, no. 1, p. 012319, 2005.

[43] K. H. Ho and H. F. Chau, "Purifying Greenberger-Horne-Zeilinger states using degenerate quantum codes," *Physical Review A*, vol. 78, p. 042329, Oct. 2008.

[44] M. Huber and M. Plesch, "Purification of genuine multipartite entanglement," *Physical Review A*, vol. 83, p. 062321, June 2011.

[45] F. Riera-Sàbat, P. Sekatski, A. Pirker, and W. Dür, "Entanglement purification by counting and locating errors with entangling measurements," *Physical Review A*, vol. 104, no. 1, p. 012419, 2021.

[46] W. Dür and H. J. Briegel, "Entanglement purification and quantum error correction," *Reports on Progress in Physics*, vol. 70, pp. 1381–1424, Aug. 2007.

[47] N. Rengaswamy, N. Raveendran, A. Raina, and B. Vasić, "Entanglement purification with quantum LDPC codes and iterative decoding," *Quantum*, vol. 8, p. 1233, Jan. 2024.

[48] E. Hostens, J. Dehaene, and B. De Moor, "Stabilizer state breeding," *Physical Review A*, vol. 74, p. 062318, Dec. 2006.

[49] S. Krastanov, A. Sanchez de la Cerda, and P. Narang, "Heterogeneous multipartite entanglement purification for size-constrained quantum devices," *Physical Review Research*, vol. 3, no. 3, p. 033164, 2021.

[50] D. E. Browne and T. Rudolph, "Resource-efficient linear optical quantum computation," *Physical Review Letters*, vol. 95, p. 010501, June 2005.

[51] V. Kuzmin, D. Vasilyev, N. Sangouard, W. Dür, and C. Muschik, "Scalable repeater architectures for multi-party states," *npj Quantum Information*, vol. 5, no. 1, pp. 1–6, 2019.

[52] N. H. Nickerson, Y. Li, and S. C. Benjamin, "Topological quantum computing with a very noisy network and local error rates approaching one percent," *Nature Communications*, vol. 4, p. 1756, Dec. 2013.

[53] N. H. Nickerson, J. F. Fitzsimons, and S. C. Benjamin, "Freely scalable quantum technologies using cells of 5-to-50 qubits with very lossy and noisy photonic links," *Physical Review X*, vol. 4, p. 041041, Dec. 2014.

4

Hardware models for diamond color centers

In this chapter, we discuss all aspects of our model for color centers in diamond. The model specifically focuses on nitrogen-vacancy centers, but most elements apply to other color centers as well. We briefly discuss how a nitrogen-vacancy center is operated in the lab. We discuss how memory decoherence is modeled and how the model parameters are characterized. We elaborate on typical time scales for operations and how we model operation noise. We introduce the single-click and double-click entanglement generation protocols and look at their individual steps. We cover typical noise sources that complement these steps and derive mixed states and success probabilities for the Bell pairs created with these protocols. We place the Bell pair models in a scientific context by comparing them to related research. The full nitrogen-vacancy center model introduced in this chapter is used in the simulations of Chs. 6 and 7.

4.1 Introduction

In this thesis, we are interested in distributed quantum computers: multiple remote quantum computing nodes that are linked up with entangled states. The distributed approach is particularly promising for systems that host native mechanisms to create entanglement at multiple length scales—*i.e.*, both within and between nodes. This naturally leads us to systems with an optical interface. The research in this thesis is conducted with a specific optically-mediated system type in mind: color centers in diamond lattices. Such color centers arise when two neighboring carbon atoms are replaced by the combination of a replacement atom and a vacancy. The centers host long-lived electron spins that exhibit coherent optical transitions, enabling their use as a *communication qubit*. This qubit is used to create entanglement with other nodes. On top of that, it can address up to tens of proximal nuclear ^{13}C spins occurring in the host material [1–3]. These nuclear spins can be used as local processing qubits—*e.g.*, as a memory to store and manipulate quantum states [4]. Hereafter, in the context of diamond color centers, we refer to such spins as *memory qubits*. Both our model and the description below are specifically catered towards the nitrogen-vacancy (NV) color center, but other color centers—*e.g.*, the tin-vacancy (SnV) or the silicon-vacancy (SiV) center—work similarly.

In this chapter, we present typical operation and coherence times for two types of NV center devices. We label these device types as *natural abundance* and *isotopically purified*, where these names refer to the relative fraction of ^{13}C nuclear spins in the diamond lattice (which is predominantly made up of spinless ^{12}C atoms). Here, the isotopically purified devices are engineered to have a 0.01% concentration of ^{13}C spins, whereas the natural abundance devices have a 1.1% concentration. An important feature for a distributed processor is the ability of the memory qubits to preserve their state while inter-node entanglement is generated. Bradley *et al.* [5] show that isotopically purified NV center devices can store a quantum state over 10^5 optical entanglement attempt repetitions. Compared to samples with natural abundance, the ^{13}C spins have a weaker coupling to their color centers. This occurs because their spin bath is more dilute, positioning the nuclear spin at a greater average distance r away from the electron spin, while dipolar coupling decreases with $1/r^3$. This increases coherence times during optical entanglement generation by several orders of magnitude but also leads to longer time scales for carrying out intra-node logic gates. In natural abundance NV center devices, a quantum state can typically be stored over 10^3 entanglement attempt repetitions [2], while single-qubit gates on the memory qubits are typically ≈ 13 times faster and two-qubit gates are typically ≈ 50 times faster than in isotopically purified samples. Further research into isotopically engineered diamond color centers is ongoing. This will likely lead to a better understanding of the trade-off between the memory qubit concentration and the associated operation times and decoherence rates.

4.2 Nitrogen-vacancy center model

In this section, we present an overview of the nitrogen-vacancy model used in this thesis. The models and model parameters are predominantly based on experimental characterization and observation of the NV center by Pompili *et al.* [2] and Bradley *et al.* [4, 5]. Experiments were performed on NV centers at temperatures of 3.7 and 4 K.

4.2.1 Electronic properties and operation

The NV center can generally occupy one of two charge states: the *negative charge state* NV^- and the *neutral charge state* NV^0. The NV^- electron spin is formed by two unpaired electron spins, resulting in a spin triplet with the spin-z projection quantum number $m_s \in \{-1, 0, 1\}$. NV^0 occurs when one of these electrons is removed, resulting in a single-electron spin-1/2 system with $m_s \in \{-1/2, 1/2\}$. For both charge states, the ground and first excited states lie within the diamond band gap, preventing charge transfer to the diamond lattice—*i.e.*, ensuring that the electrons stay localized around the NV center. Therefore, the charge states behave optically like atoms in a vacuum and can host coherent optical transitions. Unfortunately, the NV^0 ground state has a significant spin-orbit coupling and a fast orbital relaxation process [6]. This impedes the spin qubit and makes it difficult to use it as a stable spin state.

On the other hand, the NV^- state has a stable ground-state spin qubit and hosts both a long-lived spin qubit and good optical transitions. Specifically, this state can be used to take on the role of communication qubit. By deploying either a magnetic field or lattice strain, the $m_s = -1$ level can be split from the $m_s = +1$ level. The $m_s = 0$ level is often used to define the "bright" $|\uparrow\rangle \equiv |0\rangle$ state of the electron qubit, while the $m_s = -1$ state is often used as the "dark" $|\downarrow\rangle \equiv |1\rangle$ state. This is because, upon excitation with a laser pulse, an NV center in the $|0\rangle$ state transitions to the corresponding excited state before falling back to $|0\rangle$, emitting a photon in the process. This is in contrast to exciting an NV center in $|1\rangle$, as this has a substantial probability ($> 40\%$ per cycle at 4K [7]) to decay non-radiatively into a metastable state. Here, it remains for a few hundred nanoseconds before decaying non-radiatively to either $|0\rangle$ or $|1\rangle$. This means that, upon continuous optical excitation, the NV center is automatically spin-pumped to the $|0\rangle$ state. This technique is used to initialize an NV center in the $|0\rangle$ state.

Transitions between all involved energy levels follow selection rules—*i.e.*, spin and polarization are conserved. These rules can be used to read out the electron state by evaluating the center's spin-dependent fluorescence [8]. The selection rules can also be exploited to only excite the $|0\rangle$ qubit state. This requires the use of a short—*i.e.*, 2 ns [8]—spin-selective laser pulse that is resonant with the $|0\rangle \rightarrow |e\rangle$ transition, with $|e\rangle$ an optically excited state with the same spin projection as $|0\rangle$. The typical lifetime of $|e\rangle$ is 20 ns, and upon decaying back to $|0\rangle$, a photon will be emitted. By first bringing the NV center into a superposition state $(|\uparrow\rangle + |\downarrow\rangle)/\sqrt{2}$ with microwave Rabi oscillations—typically taking tens of nanoseconds [7]—an entangled state $(|\uparrow 1\rangle + |\downarrow 0\rangle)/\sqrt{2}$ between the electron state and the state of the reflected photon can be created [8]. The presence of the photonic state now makes it possible to generate entanglement between the electronic states of two remote centers, as we discuss below.

Apart from using the communication qubit for realizing inter-node entanglement, it can also be used to manipulate the states of ^{13}C atoms in its immediate vicinity. This involves microwave or radio-frequency (RF) pulses to couple the state of the communication qubit with specific carbon atoms in its surroundings. Using this technique, we can either change the state of the memory qubit—*i.e.*, perform a single-qubit gate—or alter the states of both the communication and the corresponding memory qubit—*i.e.*, perform a two-qubit gate.

Unfortunately, stochastic ionization can convert the NV^- state to the NV^0 state. Since the electron-spin state is used to control the ^{13}C spins, ionization accordingly dephases

the ^{13}C states. As such, the coherence times of the NV center memory—as discussed in Sec. 4.2.4—are currently limited by these ionization effects. Bradley *et al.* [5] propose a method to mitigate ionization-induced memory dephasing by actively recharging the NV^0 state. They show how ionization and recharging can be performed with minimal loss in fidelity in the state of a ^{13}C spin in an isotopically engineered device with reduced memory qubit concentration. This marks an important avenue for future research.

4.2.2 Entanglement creation protocols

The generation of Bell pairs between two color centers is modeled as a probabilistic process with a success probability p_{link} and time t_{link} per attempt. In Sec. 4.3, we construct an analytic model to calculate p_{link} and the Bell pair density matrix after success, both for the *single-click* (*i.e.*, the *single-photon*) [9] entanglement protocol and for the *double-click* (*i.e.*, the *two-photon*) [10] entanglement protocol. The double-click protocol is also sometimes referred to as the *Barrett-Kok* protocol. These are the most popular entanglement protocols used with NV centers. Other diamond color centers allow for other entanglement protocols, *e.g.*, protocols based on photonic scattering and reflection, using emitters that are strongly coupled to a waveguide or a cavity [11]. This works because, for such an emitter, the center can become completely reflective or transmissive based on the electron state. We do not consider scattering-based protocols in this thesis.

Both the single-click and double-click protocols are based on bringing the electron qubit of both parties in a superposition $\sqrt{\alpha}|0\rangle + \sqrt{1-\alpha}|1\rangle$, where α is the bright-state population. Subsequently, at both parties, the bright state $|0\rangle$ is excited, creating (local) entanglement between the electron spin and a single photonic state. The photonic states of both parties are then used in two incoming arms of a 50:50 beam splitter. The two outcoming arms of the beam splitter are fed into photon detectors. If the two photonic states are indistinguishable—*i.e.*, they have the same frequency (distribution) and arrive at the same time—the beam splitter erases the *which-path* information of the photons. Observing a single photon in one of the two detectors after the beam splitter entangles the electron qubits of the two parties. Observing zero or two photons means the protocol has failed. Failure requires re-initializing the electron states at both sides, re-exciting these states, and performing a new iteration of photon detection after the beam splitter. This process is repeated until only one of the detectors clicks. For the single-click protocol, α is chosen small—typically around 10% or smaller. How small α should be chosen exactly depends on the requirements: increasing α leads to higher fidelity of the final state, but decreases the success probability, whereas decreasing α has the opposite effect. Specifically, to first order, the success probability p_{link} is given by $2\eta_{\text{ph}}\alpha$, whereas the fidelity follows $1-\alpha$. The parameter η_{ph} is introduced in Sec. 4.2.3 and denotes the probability that an emitted photon is detected in the experimental setup.

It can be difficult to control the phase of the two photons: uncertainty in the length of the paths of the two photons before arriving at the beam splitter leads to phase uncertainty and reduces the fidelity of the final entangled state. This problem is remedied with the double-click protocol. This is essentially two successful rounds of the single-click protocol applied in succession, with a bit-flip on the electron states of both parties in between the two rounds. This bit-flip means that the influence of phase uncertainty from the first round cancels against the phase uncertainty of the second round. Because of the bit-flip, the

double-click protocol no longer allows us to trade in fidelity for success probability by varying the bright space population α: both the success probability and fidelity now peak at $\alpha = 1/2$.

In the analytic model presented in Sec. 4.3, we introduce the following five noise sources: the preparation fidelity of the initial spin-photon state F_{prep}, the probability of an excitation error p_{EE}, a parameter λ based on the standard deviation of the phase uncertainty due to the path-length difference between the two arms (all modeled as dephasing channels on the involved qubits), the photon indistinguishability μ for each Bell state measurement (*i.e.*, Hong-Ou-Mandel visibility), and the total photon detection probability η_{ph} in each arm (modeled with an amplitude damping channel).

For the double-click protocol, we assume the phase uncertainty to not be relevant and set $\lambda = 1$. The parameters F_{prep}, p_{EE} and μ affect the fidelity F_{link} of the Bell pair state of the double-click protocol, whereas the parameter η_{ph} limits the success probability p_{link} of a single entanglement attempt. For the single-click protocol, the fidelity F_{link} is additionally influenced by η_{ph} and λ, and p_{link} depends on the indistinguishability μ. In Sec. 4.2.3, we discuss these noise sources in more detail.

4.2.3 Noise during entanglement generation

To achieve a more realistic representation of the entangled states generated between diamond color centers with the single-click and double-click protocols, we include several noise sources in our model. In this section, we discuss these sources and present values used for these noise parameters in prior research models in Table 4.2. In Table 4.1, we summarize typical values gathered from Table 4.2. In Sec. 4.3 below, we explicitly apply these noise sources.

State preparation errors. The first noise source included is the *preparation fidelity* F_{prep} of the electron state superposition between $|0\rangle$ and $|1\rangle$. Since a depolarizing channel introduces both amplitude and phase damping of a quantum state, it can be considered as the most "harmful" channel. Therefore, it makes sense to use the depolarizing channel to model noise that is not yet (fully) parameterized. This argument caused Coopmans *et al.* [12] to use this channel to describe electron state preparation. However, typically, NV centers mostly experience dephasing noise. Therefore, instead, in Refs. [13, 14], the dephasing channel is used to model electron state preparation. Because using the dephasing channel is more convenient, we also apply the dephasing channel to model state preparation errors.

Photon excitation errors. Next to that, we include noise caused by the excitation error probability p_{EE}. This parameter describes the probability that, in case of a successful (*i.e.*, heralded) entanglement attempt, an extra *undetectable photon* is emitted by one of the centers. This can, *e.g.*, be a photon that is emitted *during* the excitation pulse, before a second photon that is emitted after the pulse is eventually detected: a process known as *double excitation*. On top of that, at every excitation event, there is a small possibility that the non-resonant $|1\rangle$ level is excited, leading to an extra photon with a different frequency. In both situations, typically, the extra photon is lost to the environment, giving rise to a dephasing channel on the electron-photon state. In general, we can reduce the probability of double excitation by decreasing the length of the excitation pulse. This, however, leads to an increase in the probability of exciting $|1\rangle$. Prior NV entanglement models only took into account the double-excitation probability p_{DE}, as it is generally believed that the

probability of an off-resonant excitation is negligible for NV centers. This is because the polarization of the light pulse only leads to a weak driving field on transitions close to the main (bright state) transition, and other transitions are sufficiently far off-resonant. The double-excitation probability in NV centers is estimated between 4% and 7% [1, 2]. In Table 4.2, we show an overview of values used for the double-excitation probability in prior NV entanglement models.

In systems where the probability of exciting the dark state is more pronounced, it can be beneficial to design the excitation pulse to induce a π rotation on the main transition, but a full 2π rotation on the closest unwanted transition. Tiurev *et al.* [15] created a model that, based on the energy difference between the main transition and the closest unwanted transition, allows one to estimate p_{EE}—*i.e.*, both the double-excitation probability and the probability of exciting this unwanted transition. Their model shows that, typically, the larger the energy difference is between the two transitions, the smaller p_{EE} becomes.

Photon phase uncertainty. As mentioned in Sec. 4.2.2, as a result of a difference in path lengths of the two incoming arms, the single-click protocol typically suffers from a phase uncertainty between the photons of both parties. Following prior research, we model this with a third dephasing channel with channel parameter λ. This parameter is calculated from the standard deviation σ_{phase} of the phase instability via [13, 14]

$$\lambda \equiv \frac{1}{2}\left(1 + \frac{I_1(\sigma_{\text{phase}}^{-2})}{I_0(\sigma_{\text{phase}}^{-2})}\right). \tag{4.1}$$

Here, I_0 and I_1 are modified Bessel functions of the zeroth and first order. This noise source is only included in the single-click model and is neglected for the double-click model. A previously found phase instability in NV centers of 14.3° [1] corresponds to $\lambda \approx 0.984$.

Photon detection probability. Apart from suffering a phase error, photons can also get lost on the way to the detector. We make use of the parameter η_{ph} to describe the total *photon detection probability* per entanglement attempt. The parameter η_{ph} can be interpreted as the probability that a diamond color center in the state $|0\rangle$ emits a resonant photon that is detected by a photon detector after the beam splitter. More specifically, as we discuss in more detail below, it can be considered as the product of the total *collection efficiency* of the NV center with the probability that a photon is emitted in the *detection time window* and in the *zero-phonon line*. The photon detection probability only influences the success probability of the entanglement protocols.

A diamond color center in the $|0\rangle$ state does not always emit a photon with the right frequency when excited with a spin-selective laser pulse. That is because transitions between the ground and excited states can either occur resonantly—*i.e.*, in the zero-phonon line (ZPL)—or off-resonantly—*i.e.*, in the so-called *phonon side band* (PSB). In the latter scenario, we end up with both a photon and a phonon—*i.e.*, a lattice vibration. Only resonant transitions without lattice vibrations are useful for entanglement creation between two diamond color centers since phonons associated with PSB photons dephase rapidly and the photons arriving at the beam splitter should be indistinguishable. Unfortunately, for NV centers, the probability p_{ZPL} of a ZPL photon emission is approximately only 3%. This induces a bottleneck for the success probability of the entanglement scheme. Other losses hinder the success probability similarly. For example, a ZPL photon still has to be "captured" by the fiber or waveguide that brings it to the beam splitter—an effect that can be hindered

by internal reflection between diamond and air. This is included in the model with the collection efficiency probability p_{col}. On top of that, such a fiber can absorb a successfully collected photon on the way to the beam splitter. For this, we implement the *transmission probability* $p_{\text{trans}} = 10^{-l_{\text{fiber}}\gamma_{\text{trans}}/10}$. Here, l_{fiber} is the length of the fiber, and γ_{trans} is its attenuation loss in dB per unit length. Furthermore, at the detector level, there is a finite probability p_{det} that a photon falling on the detector actually leads to a click: the *detector efficiency*. Finally, after exciting the state $|0\rangle$, it takes an unknown time before a photon is emitted: assuming a Poisson distribution with a characteristic lifetime τ, the probability of emission after a time t after excitation is given by $p_{\text{emission}} = 1 - \exp(-t/\tau)$. However, because we cannot wait indefinitely, we have to select a time window t_{w} in which we accept photons in our detectors. This window usually starts slightly after the excitation pulse to make sure light from the pulse itself is not detected. We define p_{tw} as the probability that a photon is emitted during the time window used. The total photon detection probability η_{ph} can now be defined as the product of all these terms: $\eta_{\text{ph}} = p_{\text{ZPL}} p_{\text{col}} p_{\text{trans}} p_{\text{det}} p_{\text{tw}}$. Because we consider distributed quantum computers and assume small fiber lengths, we neglect the influence of p_{trans} in our model.

Photon indistinguishability. The realization that it is not exactly known when emitted photons arrive at the beam splitter indicates that it is important to include a parameter μ describing how indistinguishable photons arriving at the beam splitter are. We call this parameter the *indistinguishability* or (*Hong-Ou-Mandel*) *visibility*. As mentioned above, the information on where a detected photon came from is only perfectly deleted after the beam splitter if the photons arriving are completely indistinguishable. For completely indistinguishable photons, the Hong-Ou-Mandel effect occurs: a photon at both input arms of the beam splitter will both end up in the same output arm after the beam splitter. Distinguishable photons, on the other hand, can still end up in different arms. Aspects that can reduce indistinguishability are, *e.g.*, differences in the central frequency, spectral width, or temporal position of the photonic wave packets.

Detector dark counts. Apart from a photon detector failing to detect a photon, there is also a probability p_{DC} that a detector clicks without a (diamond color center) photon being present. We call such events *dark counts*. They can occur from detector malfunctioning as well as the detection of stray photons from the environment. If we assume the probability of a dark count is governed by the Poisson distribution, a dark count rate of f_{DC} and a time window length t_{w} lead to a probability $p_{\text{DC}} = 1 - \exp(-t_{\text{w}} f_{\text{DC}})$. We note that, in the simulations presented in this thesis, we neglect dark counts. This is because we consider regimes in which dark counts become negligible.

4.2.4 Decoherence and dynamical decoupling

In our simulation model, qubits decohere with both generalized amplitude damping (GAD) and phase damping (PD) noise, as defined in Sec. 2.4.1. The GAD channel decoheres to the maximally mixed state $(|0\rangle\langle 0| + |1\rangle\langle 1|)/2$. We assign T_1 coherence times to the GAD channel and T_2 coherence times to the PD channel, with $\gamma = 1 - e^{-t/T_1}$ and $\gamma = 1 - e^{-t/T_2}$, respectively. Decoherence of the electron qubit is governed by $T1^{\text{e}}_{\text{idle}}$ and $T2^{\text{e}}_{\text{idle}}$ coherence times. For the memory qubits, we use different coherence times $T1^{\text{n}}_{\text{link}}$ and $T2^{\text{n}}_{\text{link}}$ during optical entanglement creation (which always happens via the electron qubit), and $T1^{\text{n}}_{\text{idle}}$ and $T2^{\text{n}}_{\text{idle}}$ when the node is idling. Typical values for these coherence times with state-of-the-art

NV center setups are presented in Table 4.1. In this table, we include values for both the natural abundance samples and the isotopically purified samples.

Characterization of the decoherence model and the associated coherence times was achieved in the laboratory by monitoring the drop in probability of detecting Pauli matrix eigenstates at $t > 0$ after preparing the state at $t = 0$. For the T_1 time, the $|0\rangle$ and $|1\rangle$ states were used, and the expectation value $\langle Z\rangle$ of a measurement in the Pauli-Z basis was determined after several delay times—the relaxation time T_1 can then be determined from the observed exponential drop in the expectation value. With the $|+\rangle$ and $|-\rangle$ states and the expectation $\langle X\rangle$ for measurement in the Pauli-X basis, and with $|+i\rangle$ and $|-i\rangle$ and $\langle Y\rangle$, the T_2 time of our model could be determined. For these four states, the observed exponential decay T_{dec} corresponds to the T_2 time of our model via $1/T_2 = 1/T_1 - 2/T_{\text{dec}}$.

As mentioned in Sec. 4.2.1, the model presented here is based on the NV^- state, which corresponds to the communication qubit as a spin-1 state with spin projection quantum number $m_s \in \{-1, 0, 1\}$. Stochastic ionization can convert the NV^- state to the NV^0 state with $m_s \in \{-1/2, 1/2\}$. This effect limits the coherence times in the NV center. In our model, we do not specifically include ionization, but simply absorb its influence in the ^{13}C coherence times presented here.

Next, to mitigate decoherence from quasi-static noise processes in diamond color center experiments, *dynamical decoupling* (DD) is typically interleaved into gate sequences on both the electron and nuclear spins [4]. The main idea of DD is to limit the effect of decoherence with π-pulses that periodically flip the state to average out unwanted coupling between the system and the environment. The coherence times used in this thesis are achieved in NV center spin registers with dynamical decoupling [4]. Gate operations can only be performed in between two consecutive DD pulses—*i.e.*, at the refocusing point of the qubit spins involved in the operations. We define the center-to-center time of consecutive refocusing points as $t_{\text{DD}} = t_{\text{pulse}} + 2n_{\text{DD}}t_{\text{link}}$, where t_{pulse} is the time of a π-pulse, t_{link} is the duration of a single Bell pair generation attempt, and n_{DD} is a fixed number of Bell pair generation attempts. We assume that all memory qubits of a node are decoupled with the same decoupling sequence.

We solve the following optimization problem to determine n_{DD}—*i.e.*, to determine the waiting time between two π-pulses:

$$n_{\text{DD}}(p_{\text{link}}) = \min_{n' \in \mathbb{Z}^+} \lim_{\mathcal{A}\to\infty} \sum_{i=1}^{\mathcal{A}} \sum_{j=1}^{\mathcal{A}} p_{\text{link}}^{(i)} p_{\text{link}}^{(j)} t_{n'}^{(i,j)}. \tag{4.2}$$

Here, we perform the minimization for n' as a member of the positive integers $\mathbb{Z}^+$. The goal is to minimize n' over the average completion time of generating two Bell pairs in parallel, where we also wait for both nodes to finish their DD sequences. Here, $p_{\text{link}}^{(i)} = p_{\text{link}}(1-p_{\text{link}})^{i-1}$ and $p_{\text{link}}^{(j)} = p_{\text{link}}(1-p_{\text{link}})^{j-1}$ denote the probabilities of obtaining entanglement generation success at exactly the ith and jth attempt. Further, $t_{n'}^{(i,j)} = \lceil \max(i,j)/(2n') \rceil (2n't_{\text{link}} + t_{\text{pulse}})$ is the effective time of performing the required entanglement attempts in this specific scenario, where $\lceil \max(i,j)/(2n') \rceil$ describes how many DD sequences are required for these attempts and $2n't_{\text{link}} + t_{\text{pulse}}$ describes the time of one DD sequence. To solve Eq. (4.2) in a practical setting, it suffices to take a large number for $\mathcal{A}$ instead of letting it go to infinity. Because finding n_{DD} in this way only minimizes the waiting and refocusing time during entanglement generation in two nodes, and not

Bell pair model input	
Protocol	Single-click
F_{prep}	0.99
p_{EE}	0.04
μ	0.9
λ	0.984
η_{ph}	0.0046

Bell pair model output	
p_{link}	0.0001
F_{link}	0.8966

Operation durations		
	1.1%	**0.01%**
t_{link}	6 μs	
t_{meas}	4 μs	
$t^{\text{e}}_{X,Y}$	0.14 μs	
$t^{\text{n}}_{X,Y}$	1.0 ms	13 ms
$t^{\text{e}}_{Z,H}$	0.1 μs	
$t^{\text{n}}_{Z,H}$	0.5 ms	6.5 ms
$t_{CZ,CX,CiY}$	0.5 ms	25 ms
t_{SWAP}	1.5 ms	75 ms

Operation noise	
p_{g}	0.01
p_{m}	0.01

Decoherence		
	1.1%	**0.01%**
$T1^{\text{n}}_{\text{idle}}$	300 s	
$T1^{\text{n}}_{\text{link}}$	0.03 s	1.2 s
$T1^{\text{e}}_{\text{idle}}$	300 s	
$T2^{\text{n}}_{\text{idle}}$	10 s	
$T2^{\text{n}}_{\text{link}}$	7.5 ms	1.2 s
$T2^{\text{e}}_{\text{idle}}$	1.0 s	
t_{pulse}	1.0 ms	13 ms
n_{DD}	500	2500

Table 4.1: Typical parameters for the state-of-the-art NV center hardware, for both the natural abundance sample (1.1% ^{13}C concentration) and the isotopically purified sample (0.01% ^{13}C concentration). Time scales for the natural abundance sample are characterized by Pompili *et al.* [2]. Time scales for the isotopically purified sample are based on Bradley *et al.* [4, 5]. The Bell pair model parameters are based on the sources in Table 4.2.

the waiting time during the other operations, this process typically does not lead to the optimal n_{DD}.

4.2.5 Operation times and noise

We assume that all operations take a finite amount of time. Typical time durations in state-of-the-art NV center devices can be found in Table 4.1. Time scales for gates are based on microwave and radio-frequency pulses for single-qubit gates on the electron and ^{13}C qubits, respectively. Time scales for two-qubit gates are based on phase-controlled radiofrequency driving of the carbon spins interleaved with dynamical decoupling sequences on the electron state, following the scheme described by Bradley *et al.* [4, 16].

We consider a gate set consisting of the Pauli gates $\{X, Y, Z\}$, the Hadamard gate, the CX gate, the CZ gate, and the CiY gate. These are not the native gates of the NV center, but their true gate set can be compiled into the gate set used here without additional costs in terms of two-qubit gates [17]. We assume single-qubit gates to be noiseless, while noise in two-qubit gates is modeled with a depolarizing channel $\mathcal{N}_{\text{depol2}}(\rho, p_{\text{g}})$ as introduced in Eq. (2.10).

Measurements are restricted to measuring (single-qubit) electron qubits in the Pauli-Z basis. Measuring in the Pauli-X basis is achieved with an additional Hadamard gate. We model imperfect measurements by introducing a probability p_{m} that the measurement projects onto the opposite eigenstate of the measured operator.

	Sources							
	Kalb *et al.* [18]	Humphreys *et al.* [1]	Rozpędek *et al.* [13]	Dahlberg *et al.* [14]	Pompili *et al.* [2]	Coopmans *et al.* [12]	Hermans *et al.* [19]	Avis *et al.* [20]
Single-click	✓	✓	✓	✓	✓	✓	✓	✓
Double-click	✗	✗	✗	✗	✗	✗	✗	✓
F_{prep}	-	-	0.99	0.99	-	$0.99^{\ddagger}$	-	-
p_{DE}	-	0.04	-	0.04	0.06	0.06	0.06	0.06
σ_{phase} (rad)	-	0.25^{*}	0.25^{*}	0.25^{*}	0.26^{*}	0.35	0.37^{*}	0.21
λ	-	0.984^{*}	0.984^{*}	0.984^{*}	0.983^{*}	0.97^{*}	0.965^{*}	0.989^{*}
p_{ZPL}	0.03	-	$0.46^{\dagger}$	$0.46^{\dagger}$	-	$0.46^{\dagger}$	-	0.03
p_{col}	-	-	$0.49^{\dagger}$	0.0042	-	0.0126	-	-
p_{det}	-	-	0.8	0.8	-	0.8	-	-
p_{tw}	-	-	0.44	-	-	-	-	-
η_{ph}	0.003	0.00042(1)	0.0793^{*}	0.0037^{*}	0.00044	0.0046	0.00052	0.00051
μ	0.72(3)	0.90(2)	-	0.9	0.90	0.9	-	0.90
f_{DC} (Hz)	-	20	10	20	-	1	10	-
t_{w} (ns)	-	25	5	-	-	25	15	-
p_{DC}	$2.5 \cdot 10^{-6}$	$5 \cdot 10^{-7}$	$5 \cdot 10^{-8}$	-	$1.5 \cdot 10^{-7}$	$2.5 \cdot 10^{-8}$	$1.5 \cdot 10^{-7}$	$1.5 \cdot 10^{-7}$

Table 4.2: Parameters and parameter values used in literature to describe the Bell pair state created between two nitrogen-vacancy centers. Highlighted columns are parameters that are used in the model described in this chapter—non-highlighted rows indicate auxiliary parameters used to calculate the highlighted parameters, according to Sec. 4.2.3. Values accompanied with "$*$" are converted to the convention and channel type as presented in Sec. 4.3.1. If multiple values are listed in the source, the most optimal one is presented here. Photon detection probabilities η_{ph} are calculated without transmission loss if the required information was given in the source. Enhanced zero-phonon line probabilities p_{ZPL} and collection probabilities p_{col} accompanied with "$\dagger$" assume nitrogen-vacancy centers in (micro)cavities. The preparation fidelity accompanied with "$\ddagger$" corresponds to a depolarizing channel, whereas the model of this chapter uses a dephasing channel.

4.3 Bell pair model

In this section, we derive analytic expressions for the success probabilities and density matrices of the single-click and double-click entanglement protocols. These protocols can be used to create Bell pairs between two diamond color centers. We use the general description of the protocols as discussed in Sec. 4.2.2 and apply the noise sources introduced in Sec. 4.2.3. Our single-click model combines different elements from NV center single-click models for large-scale networks [1, 2, 12–14, 18]. More elaborate versions of the single-click and double-click models can be found in Refs. [19, 20].

4.3.1 Electron-photon state

We denote the state during different stages of the protocol with $|\psi_q^{(m)}\rangle$ or $\rho_q^{(m)}$, where q describes the qubit(s) on which the state is defined, and m labels the current stage of the protocol. Both parties $i \in \{A, B\}$ begin with the electron spin qubit "e_i" in superposition:

$$|\psi_{\mathrm{e}_i}^{(1)}\rangle = \sqrt{\alpha_i}\,|\uparrow\rangle + \sqrt{1-\alpha_i}\,|\downarrow\rangle = \sqrt{\alpha_i}\,|0\rangle + \sqrt{1-\alpha_i}\,|1\rangle\,. \tag{4.3}$$

Here, we have used the convention $|\uparrow\rangle \equiv |0\rangle$ and $|\downarrow\rangle \equiv |1\rangle$ to convert the electron spin state to the computational basis. Generating entanglement between parties A and B is possible by exciting both electron spin states: on both sides, the component $|\uparrow\rangle$ in the superposition produces a photon, whereas $|\downarrow\rangle$ stays dark. We model this by adding a photonic state $|0\rangle$ labeled with "p_i" to $|\psi_{\mathrm{e}_i}^{(1)}\rangle$ and applying $\mathcal{U}_{\mathrm{exc},i} = X_{\mathrm{e}_i}\,CX_{\mathrm{e}_i,\mathrm{p}_i}\,X_{\mathrm{e}_i}$ to this state:

$$|\psi_{\mathrm{e}_i,\mathrm{p}_i}^{(2)}\rangle = \mathcal{U}_{\mathrm{exc},i}\left(\sqrt{\alpha_i}\,|00\rangle + \sqrt{1-\alpha_i}\,|10\rangle\right) = \sqrt{\alpha_i}\,|01\rangle + \sqrt{1-\alpha_i}\,|10\rangle\,. \tag{4.4}$$

Before we apply noise to these states and perform a Bell measurement with them, we combine them into a single state. This is the combined state before the photonic excitation:

$$\begin{aligned}|\psi_{\mathrm{e}_A,\mathrm{e}_B}^{(1)}\rangle = |\psi_{\mathrm{e}_A}^{(1)}\rangle \otimes |\psi_{\mathrm{e}_B}^{(1)}\rangle = \sqrt{\alpha_A\alpha_B}\,|00\rangle + \sqrt{\alpha_A(1-\alpha_B)}\,|01\rangle \\ + \sqrt{(1-\alpha_A)\alpha_B}\,|10\rangle + \sqrt{(1-\alpha_A)(1-\alpha_B)}\,|11\rangle\,.\end{aligned} \tag{4.5}$$

This state can be written as a density matrix $\rho_{\mathrm{e}_A,\mathrm{e}_B}^{(1)} = |\psi_{\mathrm{e}_A,\mathrm{e}_B}^{(1)}\rangle\langle\psi_{\mathrm{e}_A,\mathrm{e}_B}^{(1)}|$. For convenience, in the following, we use a general 4×4 density matrix for this state—*i.e.*, $\rho_{\mathrm{e}_A,\mathrm{e}_B}^{(1)} = \sum_{(i,j)\in\mathbb{Z}_4^2} \lambda_{ij}\,|i\rangle\langle j|$ instead of the specific state in Eq. (4.5). Here, λ_{ij} are general components of this density matrix and $\mathbb{Z}_4^2 = \{0,1,2,3\}^2$. The advantage of this approach is that we can more easily apply the same protocol twice for the double-click protocol. To excite our state $\rho_{\mathrm{e}_A,\mathrm{e}_B}^{(1)}$, we add photonic registers in state $|0\rangle$, and apply $\mathcal{U}_{\mathrm{exc},A}$ and $\mathcal{U}_{\mathrm{exc},B}$ to this expanded state:

$$\rho_{\mathrm{e}_A,\mathrm{e}_B,\mathrm{p}_A,\mathrm{p}_B}^{(2)} = \left(\mathcal{U}_{\mathrm{exc},A}\otimes\mathcal{U}_{\mathrm{exc},B}\right)\left(\rho_{\mathrm{e}_A,\mathrm{e}_B}^{(1)}\otimes|00\rangle\langle00|\right)\left(\mathcal{U}_{\mathrm{exc},A}\otimes\mathcal{U}_{\mathrm{exc},B}\right)^\dagger. \tag{4.6}$$

To include non-perfect state preparation, we add dephasing channels on the electron qubits. We also add dephasing channels on the two photonic qubits to include the effect of excitation errors. During excitation, a second photon can be created, and this leads to dephasing because this photon leaks to the environment—*i.e.*, if that occurs, the environment "learns" something about the electron state(s).

$$\begin{aligned}\rho_{\mathrm{e}_A,\mathrm{e}_B,\mathrm{p}_A,\mathrm{p}_B}^{(3)} &= \mathcal{N}_{\mathrm{dephase},\mathrm{e}_B}\left(\mathcal{N}_{\mathrm{dephase},\mathrm{e}_A}\left(\rho_{\mathrm{e}_A,\mathrm{e}_B,\mathrm{p}_A,\mathrm{p}_B}^{(2)}, 1-F_{\mathrm{prep}}\right), 1-F_{\mathrm{prep}}\right), \\ \rho_{\mathrm{e}_A,\mathrm{e}_B,\mathrm{p}_A,\mathrm{p}_B}^{(4)} &= \mathcal{N}_{\mathrm{dephase},\mathrm{p}_B}\left(\mathcal{N}_{\mathrm{dephase},\mathrm{p}_A}\left(\rho_{\mathrm{e}_A,\mathrm{e}_B,\mathrm{p}_A,\mathrm{p}_B}^{(3)}, \frac{p_{\mathrm{EE}}}{2}\right), \frac{p_{\mathrm{EE}}}{2}\right).\end{aligned} \tag{4.7}$$

Here, F_{prep} is the preparation fidelity and p_{EE} is the probability of obtaining an extra photon due to an excitation error. For one of the parties, we add another dephasing channel on the photonic qubit to describe phase uncertainty as a result of pathlength difference:

$$\rho^{(5)}_{\mathrm{e}_A,\mathrm{e}_B,\mathrm{p}_A,\mathrm{p}_B} = \mathcal{N}_{\text{dephase},\mathrm{p}_A}\left(\rho^{(4)}_{\mathrm{e}_A,\mathrm{e}_B,\mathrm{p}_A,\mathrm{p}_B}, 1-\lambda\right). \tag{4.8}$$

It should be noted that, for the final expression, there is no difference between dephasing the photon or electron qubit at one side—this is because they are completely correlated.

Finally, we include an amplitude damping channel on both sides to introduce the possibility of losing a photon that was originally there—this can be either due to the photon being emitted in the PSB, the photon being absorbed on the way to the detector, or the detector failing to register the photon. With η_i the total probability of losing the photon at side i, this gives rise to

$$\rho^{(6)} \equiv \rho^{(6)}_{\mathrm{e}_A,\mathrm{e}_B,\mathrm{p}_A,\mathrm{p}_B} = \mathcal{N}_{\text{AD},\mathrm{p}_B}\left(\mathcal{N}_{\text{AD},\mathrm{p}_A}\left(\rho^{(5)}_{\mathrm{e}_A,\mathrm{e}_B,\mathrm{p}_A,\mathrm{p}_B}, \eta_A\right), \eta_B\right). \tag{4.9}$$

This is the state arriving at the beam splitter.

4.3.2 Bell state measurement

The next step of the entanglement generation protocol is to perform a Bell measurement on the photons from side A and B—*i.e.*, a measurement in the Bell basis. Because, on each side, the photon is correlated with the electron spin state, this is effectively a Bell measurement on the electron states of both sides. The measurement is typically accomplished with a 50:50 beam splitter and photon detectors behind the beam splitter. If we have photon detectors that can distinguish between one or more photons being detected simultaneously—*i.e.*, so-called *photon-number-resolving* detectors—the measurement operators of this full measurement operation are as follows [14]:

$$\begin{aligned}
\mathcal{M}^{(00)} &= \mathbb{I}^{\otimes 2} \otimes |00\rangle\langle 00|, \\
\mathcal{M}^{(10)} &= \mathbb{I}^{\otimes 2} \otimes \frac{1}{2\sqrt{2}} \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & \sqrt{1+\sqrt{\mu}}+\sqrt{1-\sqrt{\mu}} & \sqrt{1+\sqrt{\mu}}-\sqrt{1-\sqrt{\mu}} & 0 \\ 0 & \sqrt{1+\sqrt{\mu}}-\sqrt{1-\sqrt{\mu}} & \sqrt{1+\sqrt{\mu}}+\sqrt{1-\sqrt{\mu}} & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \\
\mathcal{M}^{(01)} &= \mathbb{I}^{\otimes 2} \otimes \frac{1}{2\sqrt{2}} \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & \sqrt{1-\sqrt{\mu}}+\sqrt{1+\sqrt{\mu}} & \sqrt{1-\sqrt{\mu}}-\sqrt{1+\sqrt{\mu}} & 0 \\ 0 & \sqrt{1-\sqrt{\mu}}-\sqrt{1+\sqrt{\mu}} & \sqrt{1-\sqrt{\mu}}+\sqrt{1+\sqrt{\mu}} & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \\
\mathcal{M}^{(11)} &= \mathbb{I}^{\otimes 2} \otimes \frac{\sqrt{1-\mu}}{\sqrt{2}} |11\rangle\langle 11|, \\
\mathcal{M}^{(20)} &= \mathbb{I}^{\otimes 2} \otimes \frac{\sqrt{1+\mu}}{2} |11\rangle\langle 11|, \\
\mathcal{M}^{(02)} &= \mathbb{I}^{\otimes 2} \otimes \frac{\sqrt{1+\mu}}{2} |11\rangle\langle 11|.
\end{aligned} \tag{4.10}$$

These operators describe the different scenarios of the detectors clicking. For example, getting no click corresponds to the post-measurement state $\mathcal{M}^{(00)}\rho^{(6)}(\mathcal{M}^{(00)})^\dagger/p_{00}$, where

$p_{00} = \mathrm{Tr}(\mathcal{M}^{(00)}\rho^{(6)}(\mathcal{M}^{(00)})^\dagger)$ is the probability of this scenario occurring. In these equations, μ describes how indistinguishable the two photons from both sides are. For example, if they have a different frequency, a different phase, or arrive at different times, the photons are not completely indistinguishable. Setting $\mu = 1$ corresponds to perfect indistinguishability: in that case $\mathcal{M}^{(11)}$ disappears and we end up with $\mathcal{M}^{(10)} = \mathbb{I}^{\otimes 2} \otimes |\Psi^+\rangle\langle\Psi^+|$, $\mathcal{M}^{(01)} = \mathbb{I}^{\otimes 2} \otimes |\Psi^-\rangle\langle\Psi^-|$, and $\mathcal{M}^{(20)} = \mathcal{M}^{(02)} = \mathbb{I}^{\otimes 2} \otimes (|11\rangle\langle 11|)/\sqrt{2}$. This indicates how, following the Hong-Ou-Mandel effect, two indistinguishable photons always end up in the same arm after a beam splitter. It also indicates how getting a single click in one of the two detectors corresponds to performing a Bell measurement on the photonic states—getting zero or two detector clicks leads to a state that is not entangled. In such a situation, we have to try again.

There are several modifications possible to make the Bell measurement more realistic. For example, most photon detectors are not photon-number-resolvent. This means that they, *e.g.*, can not distinguish the photonic states $|10\rangle$ and $|20\rangle$. We can adapt our measurement operators in the following way to account for this:

$$\begin{aligned}
\mathcal{M}'^{(00)} &= \mathcal{M}^{(00)}, \\
\mathcal{M}'^{(10)} &= \mathcal{M}^{(10)} + \mathcal{M}^{(20)}, \\
\mathcal{M}'^{(01)} &= \mathcal{M}^{(01)} + \mathcal{M}^{(02)}, \\
\mathcal{M}'^{(11)} &= \mathcal{M}^{(11)}.
\end{aligned} \tag{4.11}$$

On top of that, we can include the effects of detector dark counts. If p_{DC} denotes the probability of a dark count, we can modify our measurement operators according to

$$\begin{aligned}
\mathcal{M}''^{(00)} &= \mathcal{M}'^{(00)}, \\
\mathcal{M}''^{(10)} &= (1-p_{\mathrm{DC}})\mathcal{M}'^{(10)} + \sqrt{p_{\mathrm{DC}}(1-p_{\mathrm{DC}})}\mathcal{M}'^{(00)}, \\
\mathcal{M}''^{(01)} &= (1-p_{\mathrm{DC}})\mathcal{M}'^{(01)} + \sqrt{p_{\mathrm{DC}}(1-p_{\mathrm{DC}})}\mathcal{M}'^{(00)}, \\
\mathcal{M}''^{(11)} &= (1-p_{\mathrm{DC}})\mathcal{M}'^{(11)} + \sqrt{p_{\mathrm{DC}}(1-p_{\mathrm{DC}})}\mathcal{M}'^{(01)} \\
&\quad + \sqrt{p_{\mathrm{DC}}(1-p_{\mathrm{DC}})}\mathcal{M}'^{(10)} + p_{\mathrm{DC}}\mathcal{M}'^{(00)}.
\end{aligned} \tag{4.12}$$

4.3.3 Single-click state

With the measurement operators of Eq. (4.12), we can now investigate the post-measurement states for getting a single click with these measurement operators: if we observe a single click in either of the detectors (*i.e.*, with the other detector not clicking), we declare success. This comes down to measurement outcomes $k = 10$ and $k = 01$, leading to the following post-measurement states:

$$\begin{aligned}
\rho^{(7a)}_{\mathrm{e}_A,\mathrm{e}_B} &= \frac{1}{p_{10}}\mathrm{Tr}_{\mathrm{p}_A,\mathrm{p}_B}\left(\mathcal{M}''^{(10)}\rho^{(6)}_{\mathrm{e}_A,\mathrm{e}_B,\mathrm{p}_A,\mathrm{p}_B}\left(\mathcal{M}''^{(10)}\right)^\dagger\right), \\
\rho^{(7b)}_{\mathrm{e}_A,\mathrm{e}_B} &= \frac{1}{p_{01}}\mathrm{Tr}_{\mathrm{p}_A,\mathrm{p}_B}\left(\mathcal{M}''^{(01)}\rho^{(6)}_{\mathrm{e}_A,\mathrm{e}_B,\mathrm{p}_A,\mathrm{p}_B}\left(\mathcal{M}''^{(01)}\right)^\dagger\right).
\end{aligned} \tag{4.13}$$

Here, we have traced out the photonic state with a partial trace, since these states are no longer relevant after the measurement. In these expressions, the measurement probabilities

p_{10} and p_{01} are given by

$$\begin{aligned} p_{10} &= \mathrm{Tr}\left(\mathcal{M}''^{(10)}\rho^{(6)}_{\mathrm{e}_A,\mathrm{e}_B,\mathrm{p}_A,\mathrm{p}_B}\left(\mathcal{M}''^{(10)}\right)^\dagger\right), \\ p_{01} &= \mathrm{Tr}\left(\mathcal{M}''^{(01)}\rho^{(6)}_{\mathrm{e}_A,\mathrm{e}_B,\mathrm{p}_A,\mathrm{p}_B}\left(\mathcal{M}''^{(01)}\right)^\dagger\right), \end{aligned} \tag{4.14}$$

The success probability of the protocol is given by $p_{\mathrm{succ}} \equiv p_{10} + p_{01}$. Specifically, this gives rise to the following post-measurement states in case of success:

$$\begin{aligned} \rho^{(7\mathrm{a},7\mathrm{b})}_{\mathrm{e}_A,\mathrm{e}_B} &= \frac{1}{p_{\mathrm{succ}}}\begin{bmatrix} p_{\uparrow\uparrow}\lambda_{00} & & & \\ & (p_A + p_{\overline{A}})\lambda_{11} & \pm\phi\sqrt{p_A p_B}\lambda_{12} & \\ & \pm\phi\sqrt{p_A p_B}\lambda_{21} & (p_B + p_{\overline{B}})\lambda_{22} & \\ & & & p_{\boldsymbol{\chi}}\lambda_{33} \end{bmatrix}, \\ p_{\mathrm{succ}} &= p_{\uparrow\uparrow}\lambda_{00} + (p_A + p_{\overline{A}})\lambda_{11} + (p_B + p_{\overline{B}})\lambda_{22} + p_{\boldsymbol{\chi}}\lambda_{33}. \end{aligned} \tag{4.15}$$

Here, we have used the probabilities defined in Table 4.3, together with the following short-hand parameters:

$$\begin{aligned} p_{\uparrow\uparrow} &\equiv p_{AB} + p_{\overline{A}\overline{B}} + p_{A\overline{B}} + p_{\overline{A}B}, \\ \phi &\equiv \sqrt{\mu}(2F_{\mathrm{prep}} - 1)^2(2\lambda - 1)(1 - p_{\mathrm{EE}})^2. \end{aligned} \tag{4.16}$$

We can fill in the specific values λ_{ij} of the state $\rho^{(1)}_{\mathrm{e}_A,\mathrm{e}_B}$ of Eq. (4.5) to get the final result of the single-click protocol. If the set-ups at sides A and B are symmetric, and there is reason to assume that $\alpha \equiv \alpha_A = \alpha_B$ and $\eta_{\mathrm{ph}} \equiv \eta_A = \eta_B$ hold, we end up with $\lambda_{00} = \alpha^2$, $\lambda_{11} = \lambda_{12} = \lambda_{21} = \lambda_{22} = \alpha(1-\alpha)$, $\lambda_{33} = (1-\alpha)^2$, $p_A = p_B$ and $p_{\overline{A}} = p_{\overline{B}}$. The post-measurement states in a case of success then simplify to:

$$\begin{aligned} \rho^{(7\mathrm{a},7\mathrm{b})}_{\mathrm{e}_A,\mathrm{e}_B} &= \frac{1}{p_{\mathrm{succ}}}\Big(\alpha^2 p_{\uparrow\uparrow}|00\rangle\langle 00| + \alpha(1-\alpha)\left((1\pm\phi)p_A + p_{\overline{A}}\right)|\Psi^+\rangle\langle\Psi^+| \\ &\qquad + \alpha(1-\alpha)\left((1\mp\phi)p_A + p_{\overline{A}}\right)|\Psi^-\rangle\langle\Psi^-| + (1-\alpha)^2 p_{\boldsymbol{\chi}}|11\rangle\langle 11|\Big), \\ p_{\mathrm{succ}} &= (p_{\uparrow\uparrow} - 2p_A - 2p_{\overline{A}} + p_{\boldsymbol{\chi}})\alpha^2 + 2(p_A + p_{\overline{A}} - p_{\boldsymbol{\chi}})\alpha + p_{\boldsymbol{\chi}}. \end{aligned} \tag{4.17}$$

We see that this gives rise to a noisy version of the state $|\Psi^+\rangle$ for outcome $k = 10$, and to a noisy version of the state $|\Psi^-\rangle$ for outcome $k = 01$, both with fidelity given by

$$F^{(\mathrm{sc})}_{\mathrm{link}} \equiv \frac{\alpha(1-\alpha)\left((1+\phi)p_A + p_{\overline{A}}\right)}{p_{\mathrm{succ}}}. \tag{4.18}$$

In a situation in which detector dark counts can be neglected, the single-click Bell pair state, for a measurement outcome $k = 10$, simplifies to

$$\begin{aligned} \rho^{(\mathrm{sc})} &= F^{(\mathrm{sc})}_{+}|\Psi^+\rangle\langle\Psi^+| + F^{(\mathrm{sc})}_{-}|\Psi^-\rangle\langle\Psi^-| + \left(1 - F^{(\mathrm{sc})}_{+} - F^{(\mathrm{sc})}_{-}\right)|00\rangle\langle 00|, \\ F^{(\mathrm{sc})}_{\pm} &= \frac{1}{p^{(\mathrm{sc})}_{\mathrm{link}}}(1\pm\phi)\eta_{\mathrm{ph}}\alpha(1-\alpha), \\ p^{(\mathrm{sc})}_{\mathrm{link}} &= 2\eta_{\mathrm{ph}}\alpha + \eta^2_{\mathrm{ph}}\alpha^2\frac{\mu - 3}{2}. \end{aligned} \tag{4.19}$$

Here, $F_+^{(\mathrm{sc})} \equiv F_{\mathrm{link}}^{(\mathrm{sc})}$ denotes the fidelity with respect to the target Bell pair state $|\Psi^+\rangle$. Furthermore, the parameter $p_{\mathrm{link}}^{(\mathrm{sc})}$ corresponds to the success probability p_{succ} of a single attempt with this protocol.

4.3.4 Double-click state

As mentioned in Sec. 4.2.2, the double-click protocol essentially consists of two successful rounds of single-click, with bit-flips on the two electron qubits in between the rounds. To derive the post-measurement state upon two successful rounds, we can now directly use the state of Eq. (4.15) as the input coefficients λ_{ij} of the successful post-measurement state after the second iteration of the protocol. This is possible because we used a general state $\rho_{\mathrm{e}_A,\mathrm{e}_B}^{(1)} = \sum_{(i,j)\in\mathbb{Z}_4^2} \lambda_{ij} |i\rangle\langle j|$ as the input combined electron state of the single-click protocol. Because of the bit-flip in between the rounds, we have to interchange the elements $|00\rangle\langle 00| \leftrightarrow |11\rangle\langle 11|$, $|01\rangle\langle 01| \leftrightarrow |10\rangle\langle 10|$ and $|01\rangle\langle 10| \leftrightarrow |10\rangle\langle 01|$ of the state $\rho_{\mathrm{e}_A,\mathrm{e}_B}^{(7\mathrm{a},7\mathrm{b})}$ to get the λ_{ij} coefficients of the successful state in the second round. We assume that this bit-flip can be executed with perfect fidelity. This gives rise to the following double-click post-measurement state in case of a single click:

$$\rho_{\mathrm{e}_A,\mathrm{e}_B}^{(8\mathrm{a},8\mathrm{b})} = \frac{1}{p'_{\mathrm{succ}}} \begin{bmatrix} p_{\uparrow\uparrow} p_{\boldsymbol{\times}} \lambda_{33} & & & \\ & (p_A + p_{\overline{A}})(p_B + p_{\overline{B}})\lambda_{22} & \pm\phi^2 p_A p_B \lambda_{21} & \\ & \pm\phi^2 p_A p_B \lambda_{12} & (p_A + p_{\overline{A}})(p_B + p_{\overline{B}})\lambda_{11} & \\ & & & p_{\uparrow\uparrow} p_{\boldsymbol{\times}} \lambda_{00} \end{bmatrix}, \tag{4.20}$$

$$p'_{\mathrm{succ}} = p_{\uparrow\uparrow} p_{\boldsymbol{\times}} (\lambda_{00} + \lambda_{33}) + (p_A + p_{\overline{A}})(p_B + p_{\overline{B}})(\lambda_{11} + \lambda_{22}).$$

Here, p'_{succ} denotes the success probability of the entire protocol—*i.e.*, the probability of a single click in both rounds. State $\rho_{\mathrm{e}_A,\mathrm{e}_B}^{(8\mathrm{a})}$ corresponds to the post-measurement state if the same detector clicks in both rounds, whereas $\rho_{\mathrm{e}_A,\mathrm{e}_B}^{(8\mathrm{b})}$ is what we get if a different detector clicks in the two rounds. Furthermore, the λ_{ij} coefficients in Eq. (4.20) represent the coefficients of the initial $\rho_{\mathrm{e}_A,\mathrm{e}_B}^{(1)}$ state—*i.e.*, before the first round of the protocol. For the double-click protocol, there is no longer a trade-off between fidelity and success probability, and both peak for $\alpha = \alpha_A = \alpha_B = 1/2$. If we use the explicit input state of Eq. (4.5) with this bright-space population, we end up with $\lambda_{ij} = 1/4$ for all λ_{ij}. This means that the states in Eq. (4.20) can be written as

$$\begin{aligned} \rho_{\mathrm{e}_A,\mathrm{e}_B}^{(8\mathrm{a},8\mathrm{b})} = \frac{1}{4p'_{\mathrm{succ}}} \Big(& p_{\uparrow\uparrow} p_{\boldsymbol{\times}} (|00\rangle\langle 00| + |11\rangle\langle 11|) \\ & + \big((p_A + p_{\overline{A}})(p_B + p_{\overline{B}}) \pm \phi^2 p_A p_B\big) |\Psi^+\rangle\langle\Psi^+| \\ & + \big((p_A + p_{\overline{A}})(p_B + p_{\overline{B}}) \mp \phi^2 p_A p_B\big) |\Psi^-\rangle\langle\Psi^-| \Big), \\ p'_{\mathrm{succ}} = \frac{p_{\uparrow\uparrow} p_{\boldsymbol{\times}} + (p_A + p_{\overline{A}})(p_B + p_{\overline{B}})}{2}. & \end{aligned} \tag{4.21}$$

Here, we note that, unlike what we had for the single-click state of Eq. (4.17), expressing the successful post-measurement double-click state in terms of a noisy Bell pair state does not require symmetry in the noise parameters η_A and η_B. It is clear that this gives rise to a noisy version of the state $|\Psi^+\rangle$ if the same detector clicks in both rounds of the protocol,

and to a noisy version of $|\Psi^-\rangle$ if different detectors click—both with fidelity

$$F_{\text{link}}^{(\text{dc})} \equiv \frac{(p_A + p_{\overline{A}})(p_B + p_{\overline{B}}) + \phi^2 p_A p_B}{4 p'_{\text{succ}}}. \tag{4.22}$$

In a situation that allows for neglecting detector dark counts, the double-click Bell pair state simplifies to

$$\begin{aligned} \rho^{(\text{dc})} &= F_{\text{link}}^{(\text{dc})} |\Psi^+\rangle\langle\Psi^+| + \left(1 - F_{\text{link}}^{(\text{dc})}\right) |\Psi^-\rangle\langle\Psi^-|, \\ F_{\text{link}}^{(\text{dc})} &= \frac{1}{2}(1+\phi^2), \\ p_{\text{link}}^{(\text{dc})} &= \frac{\eta_{\text{ph}}^2}{2}. \end{aligned} \tag{4.23}$$

Here, we have assumed $\eta_{\text{ph}} \equiv \eta_A = \eta_B$. Furthermore, $p_{\text{link}}^{(\text{dc})}$ denotes the success probability p'_{succ}.

4.3.5 Entanglement protocol selection

For the models derived in Secs. 4.3.3 and 4.3.4, the exact values of the model parameters determine whether the single-click protocol or the double-click protocol gives better results. For a fixed set of parameter values, the double-click protocol gives rise to a fixed Bell pair state and success probability, whereas the single-click protocol has a free parameter in the bright-state population α. This allows one to trade in a higher state fidelity for a lower success probability, and vice versa. It also allows one to identify parameter value regimes in which one of the two protocols is objectively the better choice. For example, we can define α_{eq} as the value of α that leads to $p_{\text{link}}^{(\text{sc})} = p_{\text{link}}^{(\text{dc})}$ for a situation without dark counts—*i.e.*, with Eqs. (4.19) and (4.23). If, for that value α_{eq}, we have $F_{\text{link}}^{(\text{sc})} > F_{\text{link}}^{(\text{dc})}$, the single-click protocol is the best option.

This is the case for the state-of-the-art parameter set in Table 4.1, as we get $p_{\text{link}}^{(\text{dc})} \approx 1 \cdot 10^{-5}$ and $F_{\text{link}}^{(\text{dc})} \approx 0.852$ with the double-click protocol. With our single-click model, using $\alpha_{\text{eq}} \approx 0.00115$, we get a state with higher fidelity: $F_{\text{link}}^{(\text{sc})} \approx 0.905$. We note that setting α this low is typically not possible in practical situations. However, for this parameter set, also for higher values of α the single-click model produces better success probabilities and fidelities than the double-click model. For the success probability and fidelity printed in Table 4.1, we have used a higher value for α, leading to one order of magnitude higher success probability and slightly lower Bell pair fidelity.

	Probability	Ph. A	Ph. B	DC
p_A	$(1-p_{\mathrm{DC}})^2\eta_A$	✓	✗	✗
p_B	$(1-p_{\mathrm{DC}})^2\eta_B$	✗	✓	✗
$p_{\overline{A}}$	$2p_{\mathrm{DC}}(1-p_{\mathrm{DC}})(1-\eta_A)$	✓	✗	✓
$p_{\overline{B}}$	$2p_{\mathrm{DC}}(1-p_{\mathrm{DC}})(1-\eta_B)$	✗	✓	✓
p_{AB}	$\frac{1+\mu}{2}(1-p_{\mathrm{DC}})^2\eta_A\eta_B$	✓	✓	✗
$p_{\overline{A}B}$	$(1-p_{\mathrm{DC}})^2\eta_B(1-\eta_A)$	✓	✓	✗
$p_{A\overline{B}}$	$(1-p_{\mathrm{DC}})^2\eta_A(1-\eta_B)$	✓	✓	✗
$p_{\overline{AB}}$	$2(1-p_{\mathrm{DC}})p_{\mathrm{DC}}(1-\eta_A)(1-\eta_B)$	✓	✓	✓
$p_{\text{✗}}$	$2(1-p_{\mathrm{DC}})p_{\mathrm{DC}}$	✗	✗	✓

Table 4.3: Scenarios leading to only one detector clicking in the Bell measurement of Sec. 4.3.2. The columns "Ph. A" and "Ph. B" describe the presence of a photon in the two incoming modes of the beam splitter. In these columns, the symbol "✓" denotes that the photon is present and arrives at the beam splitter, whereas "✓" indicates the photon is present (*i.e.*, came out of the diamond color center) but got lost on the way to the detector. The symbol "✗" indicates the photon did not come out of the diamond color center. The column "DC" denotes whether the registered click was a dark count ("✓") or not ("✗"). The probability p_{AB} is included because we assume the detectors are not photon-number-resolvent: two indistinguishable photons at both input modes of the beam splitter always end up in the same output mode. Furthermore, we notice that five out of these nine scenarios give rise to a single click, but correspond to either zero or two photons being involved in the measurement: only p_A, p_B, $p_{\overline{A}}$ and $p_{\overline{B}}$ contribute to a successful Bell measurement.

References

[1] P. C. Humphreys, N. Kalb, J. P. J. Morits, R. N. Schouten, R. F. L. Vermeulen, D. J. Twitchen, M. Markham, and R. Hanson, "Deterministic delivery of remote entanglement on a quantum network," *Nature*, vol. 558, pp. 268–273, June 2018.

[2] M. Pompili, S. L. N. Hermans, S. Baier, H. K. C. Beukers, P. C. Humphreys, R. N. Schouten, R. F. L. Vermeulen, M. J. Tiggelman, L. dos Santos Martins, B. Dirkse, S. Wehner, and R. Hanson, "Realization of a multinode quantum network of remote solid-state qubits," *Science*, vol. 372, pp. 259–264, Apr. 2021.

[3] C. M. Knaut, A. Suleymanzade, Y.-C. Wei, D. R. Assumpcao, P.-J. Stas, Y. Q. Huan, B. Machielse, E. N. Knall, M. Sutula, G. Baranes, N. Sinclair, C. De-Eknamkul, D. S. Levonian, M. K. Bhaskar, H. Park, M. Lončar, and M. D. Lukin, "Entanglement of nanophotonic quantum memory nodes in a telecom network," *Nature*, vol. 629, pp. 573–578, May 2024.

[4] C. E. Bradley, J. Randall, M. H. Abobeih, R. C. Berrevoets, M. J. Degen, M. A. Bakker, M. Markham, D. J. Twitchen, and T. H. Taminiau, "A ten-qubit solid-state spin register with quantum memory up to one minute," *Physical Review X*, vol. 9, Sept. 2019.

[5] C. E. Bradley, S. W. de Bone, P. F. W. Möller, S. Baier, M. J. Degen, S. J. H. Loenen, H. P. Bartling, M. Markham, D. J. Twitchen, R. Hanson, D. Elkouss, and T. H. Taminiau, "Robust quantum-network memory based on spin qubits in isotopically engineered diamond," *npj Quantum Information*, vol. 8, pp. 1–9, Oct. 2022.

[6] S. Baier, C. E. Bradley, T. Middelburg, V. V. Dobrovitski, T. H. Taminiau, and R. Hanson, "Orbital and spin dynamics of single neutrally-charged nitrogen-vacancy centers in diamond," *Physical Review Letters*, vol. 125, p. 193601, Nov. 2020.

[7] C. E. Bradley, *Order from Disorder: Control of Multi-Qubit Spin Registers in Diamond*. Doctoral thesis, Delft University of Technology, Delft, The Netherlands, 2021.

[8] H. Bernien, B. Hensen, W. Pfaff, G. Koolstra, M. S. Blok, L. Robledo, T. H. Taminiau, M. Markham, D. J. Twitchen, L. Childress, and R. Hanson, "Heralded entanglement between solid-state qubits separated by three metres," *Nature*, vol. 497, pp. 86–90, May 2013.

[9] C. Cabrillo, J. I. Cirac, P. García-Fernández, and P. Zoller, "Creation of entangled states of distant atoms by interference," *Physical Review A*, vol. 59, pp. 1025–1033, Feb. 1999.

[10] S. D. Barrett and P. Kok, "Efficient high-fidelity quantum computation using matter qubits and linear optics," *Physical Review A*, vol. 71, p. 060310, June 2005.

[11] H. K. Beukers, M. Pasini, H. Choi, D. Englund, R. Hanson, and J. Borregaard, "Remote-entanglement protocols for stationary qubits with photonic interfaces," *PRX Quantum*, vol. 5, p. 010202, Mar. 2024.

[12] T. Coopmans, R. Knegjens, A. Dahlberg, D. Maier, L. Nijsten, J. de Oliveira Filho, M. Papendrecht, J. Rabbie, F. Rozpędek, M. Skrzypczyk, L. Wubben, W. de Jong, D. Podareanu, A. Torres-Knoop, D. Elkouss, and S. Wehner, "NetSquid, a NETwork Simulator for QUantum Information using Discrete events," *Communications Physics*, vol. 4, pp. 1–15, July 2021.

[13] F. Rozpędek, R. Yehia, K. Goodenough, M. Ruf, P. C. Humphreys, R. Hanson, S. Wehner, and D. Elkouss, "Near-term quantum-repeater experiments with nitrogen-vacancy centers: Overcoming the limitations of direct transmission," *Physical Review A*, vol. 99, p. 052330, May 2019.

[14] A. Dahlberg, M. Skrzypczyk, T. Coopmans, L. Wubben, F. Rozpędek, M. Pompili, A. Stolk, P. Pawełczak, R. Knegjens, J. de Oliveira Filho, R. Hanson, and S. Wehner, "A link layer protocol for quantum networks," in *Proceedings of the ACM Special Interest Group on Data Communication*, SIGCOMM '19, (New York, NY, USA), pp. 159–173, Association for Computing Machinery, Aug. 2019.

[15] K. Tiurev, P. L. Mirambell, M. B. Lauritzen, M. H. Appel, A. Tiranov, P. Lodahl, and A. S. Sørensen, "Fidelity of time-bin-entangled multiphoton states from a quantum emitter," *Physical Review A*, vol. 104, p. 052604, Nov. 2021.

[16] T. H. Taminiau, J. Cramer, T. Van Der Sar, V. V. Dobrovitski, and R. Hanson, "Universal control and error correction in multi-qubit spin registers in diamond," *Nature Nanotechnology*, vol. 9, no. 3, pp. 171–176, 2014.

[17] M. H. Abobeih, Y. Wang, J. Randall, S. J. H. Loenen, C. E. Bradley, M. Markham, D. J. Twitchen, B. M. Terhal, and T. H. Taminiau, "Fault-tolerant operation of a logical qubit in a diamond quantum processor," *Nature*, vol. 606, pp. 884–889, June 2022.

[18] N. Kalb, A. A. Reiserer, P. C. Humphreys, J. J. Bakermans, S. J. Kamerling, N. H. Nickerson, S. C. Benjamin, D. J. Twitchen, M. Markham, and R. Hanson, "Entanglement distillation between solid-state quantum network nodes," *Science*, vol. 356, pp. 928–932, June 2017.

[19] S. L. N. Hermans, M. Pompili, L. D. Santos Martins, A. R-P Montblanch, H. K. C. Beukers, S. Baier, J. Borregaard, and R. Hanson, "Entangling remote qubits using the single-photon protocol: An in-depth theoretical and experimental study," *New Journal of Physics*, vol. 25, p. 013011, Jan. 2023.

[20] G. Avis, F. Ferreira da Silva, T. Coopmans, A. Dahlberg, H. Jirovská, D. Maier, J. Rabbie, A. Torres-Knoop, and S. Wehner, "Requirements for a processing-node quantum repeater on a real-world fiber grid," *npj Quantum Information*, vol. 9, pp. 1–12, Oct. 2023.

5

Creating and distilling multipartite GHZ states with Bell pairs

The distribution of high-quality GHZ states is at the heart of many quantum communication tasks, ranging from extending the baseline of telescopes to secret sharing. GHZ states also play an important role in error-correction architectures for distributed quantum computation, where Bell pairs can be leveraged to create an entangled network of quantum computers. In this chapter, we investigate the creation and distillation of GHZ states out of non-perfect Bell pairs over quantum networks. In particular, we introduce a heuristic dynamic programming algorithm to optimize over a large class of protocols that create and distill GHZ states. All protocols considered use a common framework based on measurements of non-local stabilizer operators of the target state (i.e., the GHZ state), where each non-local measurement consumes another (non-perfect) entangled state as a resource. The new protocols outperform previous proposals for scenarios without decoherence and local gate noise. Furthermore, the algorithms can be applied to find protocols for any number of parties and any number of entangled pairs involved.

This chapter is based on S. de Bone, R. Ouyang, K. Goodenough, and D. Elkouss, "Protocols for creating and distilling multipartite GHZ states with Bell pairs," *IEEE Transactions on Quantum Engineering*, vol. 1, pp. 1–10, Dec. 2020.

5.1 Introduction

As discussed in Sec. 1.1, the promise of distributed quantum computing lies in the possibility of building a quantum computer without the difficulty of engineering a large multi-qubit device. In exchange, the feasibility of such a networked device critically relies on the availability of high-fidelity entanglement. This is because entanglement is required for the realization of multi-qubit operations between different quantum computers. In particular, entangled states are necessary for performing error-detection measurements in error-correction codes executed with distributed quantum computers.

Using error correction for fault-tolerant quantum computation relies on encoded data. To correct or track the errors in the encoded data, it is necessary to periodically perform joint measurements on different qubits. If the whole encoded state lies in a single quantum device, these joint measurements can be performed by applying the appropriate multi-qubit operations and measuring an ancilla qubit. However, in distributed implementations the joint measurements become non-local. The ingredient that enables joint measurements are GHZ states. By consuming an N-qubit GHZ state it is possible to perform a non-local measurement between N parties. The challenge of distributed quantum computation is to produce GHZ states at a fast enough rate and with high enough fidelity to enable fault-tolerant quantum computation.

Creating GHZ states is experimentally challenging. A simple protocol for creating an N-qubit GHZ state consists of *fusing* $N-1$ Bell pairs. However, the fidelity of the GHZ state degrades exponentially with N. This problem can be overcome by more complicated protocols that distill either the input Bell pairs or any of the intermediate states of a protocol. This generally improves the fidelity of the final GHZ state but comes at the price of consuming a larger number of Bell pairs.

Some of the physical platforms with a coherent optical interface for generating remote entanglement mentioned in Sec. 1.1 have already demonstrated the generation of long-lived remote entanglement [1] and even distillation [2]. However, the rate at which entanglement can be produced is slower than the gate times. In consequence, the rate at which GHZ states are produced becomes the bottleneck for the performance of distributed quantum computer implementations [3]. Moreover, while our motivation stems from distributed quantum computation, efficient GHZ generation has direct application in several other applications including secret sharing [4], anonymous transmission [5], clock synchronization [6], and extending the baseline of telescopes [7].

The goal of our research is to minimize the number of Bell pairs necessary to produce high-fidelity GHZ states. We do this by searching the protocol space for creating GHZ states out of Bell pairs. The difficulty of the problem is that given a number of parties and a number of input Bell pairs, the number of possible protocols is very large: it grows super-exponentially with these parameters. Our approach to deal with the large number of protocols is therefore to take the heuristic approximation that optimal protocols for some number of copies of a GHZ state are composed of optimal protocols for a smaller number of copies or parties. This heuristic leads to a *dynamic program*.

As became clear in Sec. 3.6, distillation is currently better understood in the bipartite case than in the multipartite case. In the bipartite case, it is even known that some protocols achieve an optimal trade-off between rate and fidelity [8]. In the context of a distributed implementation of the surface code, Nickerson *et al.* [9] optimized a family of protocols

for generating four-partite GHZ states out of noisy Bell pairs. To facilitate experimental feasibility, the GHZ distillation protocols require three qubits per node. The number of possible protocols in this family, while large, is still brute-force tractable. Subsequent work optimized a similar family of protocols in the presence of loss [3]. We introduced the protocols from Refs. [3, 9] in Sec. 3.6.9. We leave the extension of our approach to more realistic settings with loss for Chs. 6 and 7. In this chapter, in contrast with Ref. [9], we are interested in more general protocols that minimize the number of Bell pairs consumed independently of the size of the required quantum register. This different ansatz is justified by recent experimental progress with multi-qubit registers [10].

In Secs. 5.2 and 5.3, we introduce the formalism and building blocks of the GHZ generation protocols considered. In Sec. 5.4, we show that existing GHZ generation protocols are included in our search space. In Sec. 5.5, we present our dynamic program. In Sec. 5.6, we show the performance of the best GHZ creation protocols found. Finally, we draw our conclusions in Sec. 5.7.

5.2 Bell and GHZ diagonal states

Here, we introduce the notation and definitions used in the rest of this chapter, together with our model for states. We use the N-qubit GHZ basis introduced in Sec. 2.4.2, with basis states $|\phi_{s_1 s_2 s_3 \ldots s_N}\rangle$ defined by the signs of the stabilizer generators $(-1)^{s_1} X_1 X_2 \ldots X_N$, $(-1)^{s_2} Z_1 Z_2$, $(-1)^{s_3} Z_2 Z_3$, …, $(-1)^{s_N} Z_{N-1} Z_N$, for $s_i \in \{0, 1\}$ for all $i \in \{1, 2, \ldots, N\}$. As discussed in Sec. 2.4.2, we use the capital Φ symbol to denote the density matrix corresponding to a basis state—*i.e.*, $\Phi_{s_1 s_2 \ldots s_N} \equiv |\phi_{s_1 s_2 \ldots s_N}\rangle\langle\phi_{s_1 s_2 \ldots s_N}|$. For $N = 2$, the GHZ basis reduces to the Bell basis, which is also introduced in Sec. 2.4.2.

We restrict our attention to states that do not contain off-diagonal elements in the Bell basis—the so-called Bell diagonal states. This restriction does not reduce the applicability of our methods, because any bipartite qubit state can be transformed to a Bell diagonal state with the same fidelity via the Pauli twirling method of Eqs. (2.20) and (2.21) in Sec. 2.4.4, a procedure that relies on local operations and classical communication. On top of that, the operations introduced in Sec. 5.3 take Bell diagonal states to other Bell diagonal states or to states that are diagonal in the GHZ basis, allowing us to restrict our attention to just Bell and GHZ diagonal states. For N parties ν_1, ν_2, …, ν_N we write these states as:

$$\rho_{\nu_1 \nu_2 \ldots \nu_N} = \sum_{(s_1, s_2, \ldots, s_N) \in \{0,1\}^N} p_{s_1 s_2 \ldots s_N} \Phi_{s_1 s_2 \ldots s_N}. \tag{5.1}$$

Unless otherwise stated, in the remainder we use the term "state" to denote both Bell diagonal states and GHZ diagonal states, and call the $p_{s_1 s_2 \ldots s_N}$ elements in Eq. (5.1) the "coefficients" of the state. We use the shorthand $|\mathrm{GHZ}^{(N)}\rangle$ for $|\phi_{00\ldots0}\rangle$ when we need to explicitly denote the number of qubits of the state. The fidelity $F \equiv \langle \mathrm{GHZ}^{(N)}|\rho|\mathrm{GHZ}^{(N)}\rangle$ provides a measure of the closeness between a general N-qubit state ρ and the pure state $|\mathrm{GHZ}^{(N)}\rangle$. For diagonal density matrices in the Bell or the GHZ basis, the fidelity equals the value of the $p_{00\ldots0}$ coefficient. We define F_{Bell} as the fidelity of a Bell diagonal state $|\mathrm{GHZ}^{(2)}\rangle = |\Phi^+\rangle$ and F_{GHZ} or $F_{\mathrm{GHZ}}^{(N)}$ as the fidelity of an N-qubit GHZ diagonal state $|\mathrm{GHZ}^{(N)}\rangle$.

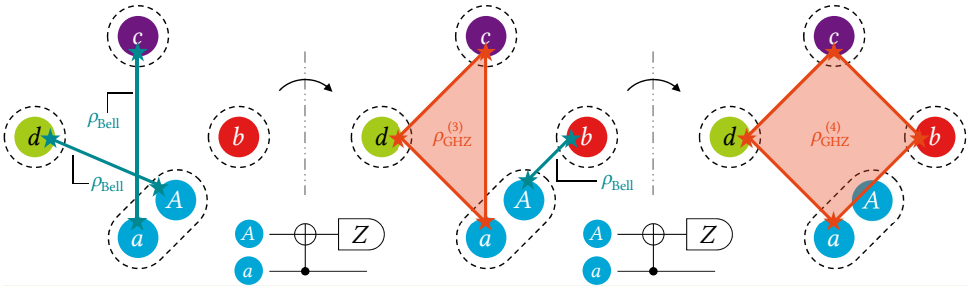


Figure 5.1: The fusion operation allows merging Bell diagonal states and GHZ diagonal states that overlap at one of the network parties. This party applies the operation depicted below the arrows, followed by local Pauli gate corrections that depend on the measurement outcome—this correction is not depicted here but can be observed in, *e.g.*, Fig. 6.2. In our implementation, ρ_{Bell} is a state of the form of Eq. (5.1) on two parties (*i.e.*, for $N = 2$ parties) and $\rho^{(N)}_{\text{GHZ}}$ is a state of the form of Eq. (5.1) on N parties. The fusion operation is included as Block C.1 in Fig. 3.28.

5

5.3 Operations on Bell and GHZ diagonal states

This section discusses two operations on Bell and GHZ diagonal states: fusion operations, that merge two states, and distillation operations, that consume one state to improve the fidelity of another state.

5.3.1 Fusion

The fusion operation merges two states. The operation takes an N_1-qubit state and an N_2-qubit state that overlap in one network node, in the sense that this node holds (at least) one qubit of each state. The fusion operation consists of a CX gate between one qubit of each state, a measurement in the Pauli-Z basis of the target qubit (see Fig. 5.1) and local Pauli gate corrections to the qubits of the other state. If the two qubits involved are qubit i of the N_1-qubit state and qubit j of the N_2-qubit, we say that we are fusing the N_1-qubit state at qubit i with the N_2-qubit state at qubit j. This results in a new $(N_1 + N_2 - 1)$-qubit state. The fusion operation is deterministic.

5.3.2 Non-local stabilizer measurements

A non-local stabilizer measurement also involves two states: a main state and an ancillary state. The ancillary state is consumed to measure a stabilizer operator on the main state. In the context of a distillation scheme, the stabilizer operator is one of the stabilizers of some target state. Then, the non-local stabilizer measurement can be understood as an error-detection scheme, as discussed in Sec. 3.6.2. An $m = +1$ outcome projects the state into the corresponding eigenspace compatible with the target state, while an $m = -1$ outcome projects the state into the corresponding eigenspace orthogonal to the target state. For this reason, the state is kept when the measurement outcome is $m = +1$ and discarded otherwise.

In Fig. 2.4b on page 21, we see a quantum circuit that measures a joint Pauli operator $P_1 P_2 \ldots P_N$ with the aid of an N-qubit state $|\text{GHZ}^{(N)}\rangle = (|0\rangle^{\otimes N} + |1\rangle^{\otimes N})/\sqrt{2}$. The qubits of the ancillary state are measured individually in the Pauli-X basis, and the network parties use classical communication to calculate the full measurement outcome. Whereas the

non-local measurement of the stabilizer in Fig. 2.4b is perfect, in practical situations the ancillary state is noisy and the operation is only carried out approximately.

The N-qubit GHZ state $|\text{GHZ}^{(N)}\rangle$ has three different types of stabilizers: ZZ stabilizers of weight 2, combinations of ZZ operators of weight M (where $M \leq N$ is always an even number), and operators of weight N consisting of combinations of $X_1 X_2 \ldots X_N$ and any number of the ZZ operators. Measuring the latter type non-locally requires another N-qubit ancillary state. The ZZ operators can be measured with a Bell pair as an ancillary state. For combinations of ZZ operators of higher weight $M \leq N$, a GHZ state of weight M is required.

5.4 GHZ GENERATION PROTOCOLS

Many GHZ creation and distillation protocols can be described by combining the operations from Sec. 5.3. In particular, fusion operations create larger multipartite states and non-local stabilizer measurements can be used to increase the fidelity of the main state. As an example, in the following, we include the Expedient and Stringent protocols from Nickerson *et al.* [9] for creating and distilling a GHZ state shared by $N = 4$ network parties. We introduced these protocols in Sec. 3.6.9. In Fig. 5.2, we include a schematic representation of both protocols. For the Expedient protocol, compared to Fig. 3.29, we now assume that all gates are noiseless, which allows us to revert the double selection protocol to two rounds of distillation in the form considered in this chapter—see also Sec. 3.6.5 for more information. For both protocols, the first part consists of non-local measurements involving all qubits on two opposite sides of the network: to distill Bell pairs between qubits a and b, and between qubits c and d in Fig. 5.2. In Fig. 5.2, this part consists of steps 1 and 2 for Expedient, and steps 1 to 8 for Stringent. The two distilled Bell pairs are stored, and the protocols continue with several rounds of non-local measurements to distill two new Bell pairs in the perpendicular direction (between qubits A and C, and between qubits B and D in Fig. 5.2). These are steps 3 and 4 for Expedient, and steps 9 to 11 for Stringent. After that, the four Bell pairs are fused into a 4-qubit GHZ state: this is step 5 for Expedient and step 12 for Stringent in Fig. 5.2. The protocols end by repeating the distillation steps for a Bell pair between qubits B and D and between A and C. Finally, these distilled Bell pairs are used to perform two ZZ non-local measurements on the 4-qubit GHZ state.

In total, the Expedient protocol consumes a minimum of $K = 22$ Bell pairs. As is clear from Fig. 5.2 and the description above, the Stringent protocol has the same main structure as the Expedient protocol but consumes a minimum of $K = 42$ Bell pairs. The additional Bell pairs are used to increase the fidelity of the states that take part in the fusion step, as well as to increase the fidelity of the Bell pairs used to perform the final ZZ non-local measurements on the 4-qubit GHZ state.

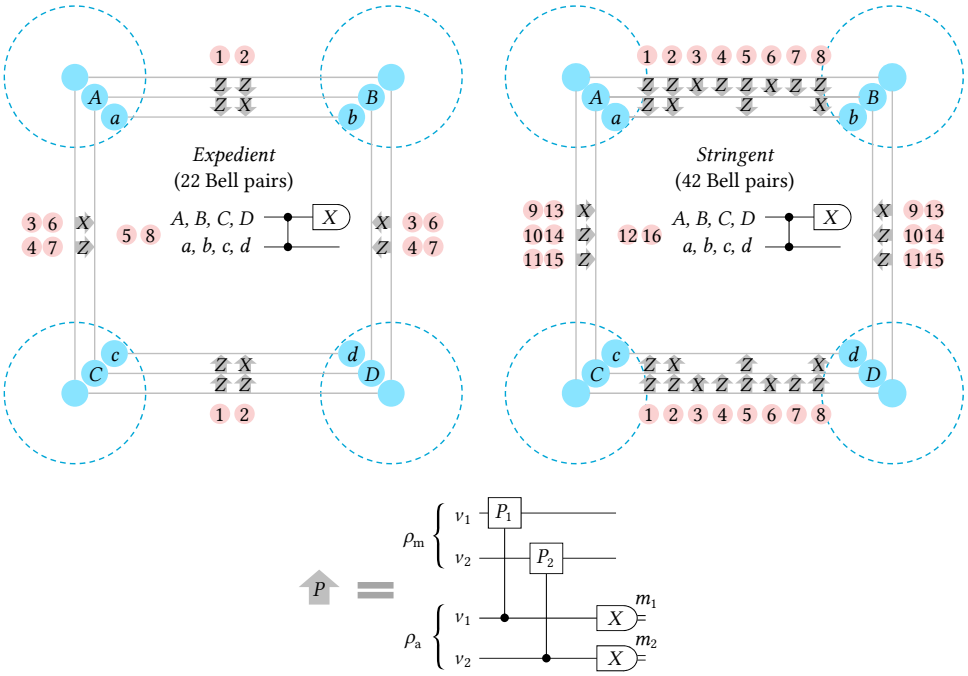


Figure 5.2: Two protocols for creating and purifying a 4-qubit GHZ diagonal state out of Bell diagonal states shared between 4 network parties [9]. For both protocols, each node (dotted circles) requires three qubits (solid blue dots). The numbers in the red dots denote the order in which individual steps are carried out. Solid gray lines depict Bell pairs shared between qubits. Gray arrows denote the consumption of a Bell pair—the Bell pairs at the beginning of the arrow need to be regenerated for the next step.

5.5 Dynamic programs to optimize GHZ generation

In this section, we discuss several algorithms to optimize GHZ generation. In particular, we consider the problem of probabilistically generating an N-qubit GHZ state with maximum fidelity F_{GHZ} starting from K isotropic Bell pairs with fidelity F_{Bell}:

$$\rho_{AB} = F_{\text{Bell}}\Phi_{00} + \sum_{(s_1,s_2)\neq(0,0)} \frac{1-F_{\text{Bell}}}{3}\Phi_{s_1 s_2}. \tag{5.2}$$

We do not restrict the number of qubits that the N nodes need to hold, and we also do not restrict their connectivity.

Ideally, we would consider all possible ways of distributing the Bell pairs over the network parties, and all possible combinations of the operations from Sec. 5.3. Unfortunately, the number of these combinations grows super-exponentially in N (see Ref. [11] for a similar argument). This makes a brute-force approach infeasible for relevant values of (N,K)—in particular for the protocols described in Sec. 5.4: $(N,K) = (4,22)$ and $(N,K) = (4,42)$. For this reason, similar to the approaches in Refs. [11, 12] for Bell pair distribution in the context of quantum repeater chains, we propose heuristic *dynamic programs* for optimizing the distribution of GHZ states.

The dynamic programs reduce the complexity of the optimization and enable finding good protocols for large values of N and K. However, the output is not necessarily optimal. In the following, we first describe a simple dynamic program for optimizing GHZ generation. Next, we build on this description to present a randomized version of the dynamic program.

5.5.1 Base dynamic program

The dynamic program takes as input the problem parameters N, K, F_{Bell} and a buffer size N_{buf}. The parameters specify the size of the final N-qubit GHZ state, the number of Bell pairs K, their fidelity F_{Bell} and the number of protocols N_{buf} to store at each intermediate step of the algorithm—*i.e.*, for each combination of the number of Bell pairs and GHZ state size. The pseudo-code of this algorithm can be found in Alg. 1.

The algorithm begins with $(n,k) = (2,1)$ and proceeds iteratively combining the solutions for smaller values of n and k until $(n,k) = (N,K)$. More precisely, for each value of (n,k) the algorithm combines the protocols found for smaller values of (n,k) in all possible ways to perform either a non-local measurement or to fuse the states, evaluates the fidelity of the resulting state for each combination, and stores the N_{buf} protocols that achieve the largest fidelity. For $(n,k) = (2,1)$, the algorithm stores a Bell pair with fidelity F_{Bell}. Subsequently, the algorithm increases k to $(n,k) = (2,2)$ and it continues increasing k, until $(n,k) = (2,K)$. At that point, n is increased by one, and k is reset to $n-1$. This process continues until N and K are reached.

We would like to stress that this is a heuristic approach and, in general, it does not lead to the optimal algorithms. This can be observed in Fig. 5.3. We plot the fidelity of the produced F_{GHZ} with respect to the fidelity F_{Bell} of the input Bell pairs. The fidelity of the produced F_{GHZ} is not always monotonically increasing. Moreover, as we increase the size of the buffer N_{buf} the program tends to find better protocols. However, lines with different values of N_{buf} cross, highlighting the suboptimality of the output.

Algorithm 1: Base dynamic program to optimize GHZ generation.

Data: Number N of qubits of final GHZ state
Number K of isotropic Bell pairs of final GHZ state
Fidelity F_{Bell} of the isotropic Bell pairs
Number N_{buf} of protocols stored in buffer per step
Result: Protocol to create an N-qubit GHZ state out of K isotropic Bell pairs

for $\{(n,k) \mid n \leqslant N, k \leqslant K, k \geqslant n-1\}$ **do**
  # Try all non-local measurement combinations.
  for *stabilizer* $\in$ *{stabilizers of* $|\text{GHZ}^{(n)}\rangle$ *}* **do**
    $n' \leftarrow$ weight of *stabilizer*
    for $k' \in [n'-1, k-n+1]$ **do**
      Measure *stabilizer* on stored $(n, k-k')$ state consuming the stored (n', k') state. Store new state if fidelity is higher than at least one of the existing (n,k) states. If more than N_{buf} states are stored at (n,k), remove the one with lowest fidelity.
  # Try all fusion combinations.
  for $n_2 \in [2, n-1]$ **do**
    $n_1 \leftarrow n - n_2 + 1$
    for $k_2 \in [n_2 - 1, k-n+1]$ **do**
      $k_1 \leftarrow k - k_2$
      for $(i,j) \in [1,n_1] \times [1,n_2]$ **do**
        Fuse stored states (n_1,k_1) at qubit i and (n_2,k_2) at qubit j. Store new state if fidelity is higher than at least one of the existing (n,k) states. If more than N_{buf} states are stored at (n,k), remove the one with lowest fidelity.
return *stored* (N,K) *state with highest fidelity*

5.5.2 Randomized version of the dynamic program

The base dynamic algorithm allows for optimizing GHZ generation for moderate sizes. Unfortunately, even if faster than brute force, it still scales exponentially with the size of the GHZ state, and, for fixed GHZ size, it scales quadratically with the size of the buffer. In this section, we discuss a randomized version of the base dynamic program. This algorithm scales to larger GHZ sizes and, in practice, finds better protocols.

The randomized algorithm takes an additional parameter compared to the base dynamic algorithm. This is the temperature T_{temp}, used to decide whether or not to keep intermediate protocols. The randomized algorithm has the same two loops over (n,k) as the base one. It has an outer loop over the GHZ state size n starting from $n=1$ to N and an inner loop over the number of Bell pairs starting from $k=n-1$ to K. The two algorithms differ in how they construct the pool of protocols for (n,k).

Similar to the base algorithm, the randomized algorithm fills a pool of N_{buf} protocols for each combination of (n,k). For $1 \leq i \leq N_{\text{buf}}$, the algorithm chooses between non-local measurement or fusion with probability one-half. It then selects uniformly at random a stabilizer to perform a non-local measurement or a fusion scheme to implement. Both the non-local measurement and the fusion scheme combine two smaller states. The parameters are singled out by the scheme choice: $(n,k-k')$ and (n',k') for the non-local measurement or $(n-n'+1,k-k')$ and (n',k') for fusion. Finally, the two states are chosen uniformly at random from the N_{buf} states stored.

The ith slot of the pool of N_{buf} states is filled with certainty with the new state if either it is the first state (*i.e.*, $i=1$) or it achieves a higher fidelity than the previous protocol. If these conditions are not met, the new state can still probabilistically be accepted with probability $e^{\Delta F_{\text{GHZ}}/T_{\text{temp}}}$, where ΔF_{GHZ} is the fidelity resulting from the new protocol minus the fidelity resulting from the previous protocol. If the value for T_{temp} is set at a high value, states with a lower fidelity than the fidelity of the previous protocol are more likely to be accepted. If the new protocol is not stored, the ith slot is filled with the $(i-1)$th state. This approach means that protocols that lead to a high F_{GHZ} end up in multiple buffer slots and are therefore more likely to be randomly selected at larger values of (n,k).

For the randomized algorithm, the value for the buffer size N_{buf} is usually set significantly larger than the value for N_{buf} in the standard algorithm. This can be understood by realizing that the standard version of the dynamic algorithm visits all allowed combinations of states at smaller (n,k), whereas the randomized version visits exactly N_{buf} combinations at each (n,k).

In the following, we investigate the effect of the configuration parameters, the buffer size N_{buf} and the temperature T_{temp}, on the performance of the algorithm. In particular, we fix $(N,K)=(4,42)$ and evaluate the fidelity of the final GHZ state as a function of the input Bell pair fidelity. We show the results in three plots in Fig. 5.4. From top to bottom, the temperatures are fixed to $T_{\text{temp}}=1\cdot 10^{-5}$, $T_{\text{temp}}=0.1$, and $T_{\text{temp}}=1$. In each plot, we show four lines corresponding to four different buffer sizes $N_{\text{buf}} \in \{1,10,50,200\}$. We see that, in general, a lower temperature gives better results. This can be understood by realizing that for low temperatures, states with higher fidelity are more likely to be stored in many slots of the buffer. This also indicates that fidelity is a good measure for determining the quality of a protocol as a building block for a larger protocol.

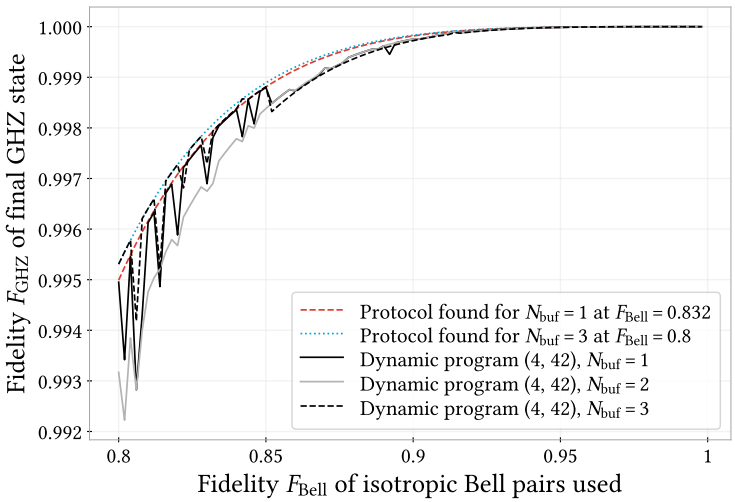


Figure 5.3: Fidelity of the final 4-qubit GHZ as a function of the input Bell pair fidelity for a fixed number of input Bell pairs $K = 42$. We plot (black solid line, gray solid line, and dashed black line) the results of the base dynamic program with a buffer size $N_{\text{buf}} \in \{1, 2, 3\}$ for each input Bell pair fidelity. We compare the results to the fidelity achieved by two fixed protocols for all input Bell pair fidelities (dashed red line and dotted blue line). The fixed protocols correspond with the protocols that the base dynamic program found for $N_{\text{buf}} = 1, F_{\text{Bell}} = 0.832$ and $N_{\text{buf}} = 3, F_{\text{Bell}} = 0.8$. They achieve approximately the convex hull of the individual protocols found by the base dynamic program with $N_{\text{buf}} = 1, N_{\text{buf}} = 3$ respectively.

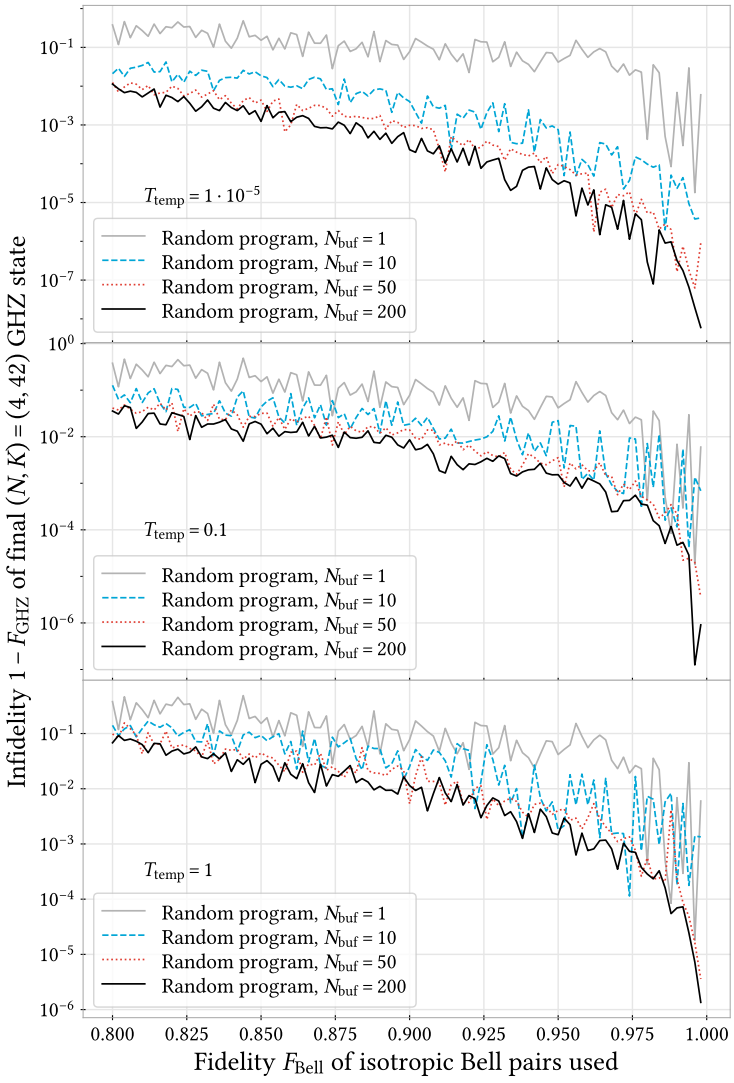


Figure 5.4: Performance of the randomized dynamic program as a function of the configuration parameters, for different fixed temperatures; from top to bottom $T_{\text{temp}} = 1 \cdot 10^{-5}$, $T_{\text{temp}} = 0.1$, and $T_{\text{temp}} = 1$.

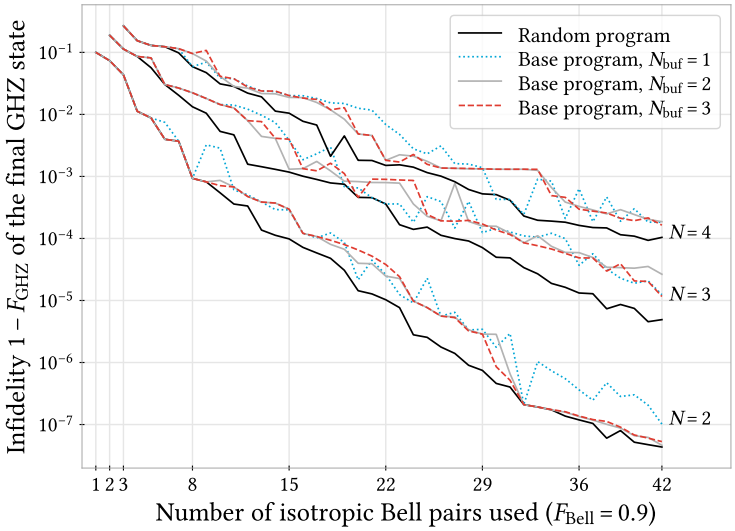


Figure 5.5: Fidelity achieved by the protocols from the base and random dynamic programs as a function of the number of Bell pairs. The fidelity of the input Bell pairs is fixed to $F_{\text{Bell}} = 0.9$. From top to bottom, each set of four lines corresponds to GHZ size $N = 4, 3, 2$. The lines labeled "base program" indicate the best protocols found by the base dynamic algorithm for different buffer sizes N_{buf}. The black solid line indicates the best protocol found by the random variant. For the base algorithm, we used buffer sizes $N_{\text{buf}} \in \{1, 2, 3\}$. For the random algorithm, we ran the algorithm 44 times, with 18 different temperatures (between $T_{\text{temp}} = 1 \cdot 10^{-5}$ and $T_{\text{temp}} = 9 \cdot 10^{-4}$) per iteration and $N_{\text{buf}} = 200$.

5.5.3 Comparison between the dynamic programs

We end the discussion of the dynamic programs by comparing the output of the base dynamic program against the randomized version. In Fig. 5.5, we compare the best results with $(N,K) = (4,42)$ of the base dynamic program against the randomized program. For the parameters chosen, the randomized dynamic program outperforms the base dynamic program.

5.6 Results

In this section, we use the dynamic algorithms to find good GHZ creation protocols. First, we investigate how the different variants of the dynamic program discussed in Sec. 5.5 compare with each other and what the optimal parameter configurations are. Then, we use the programs to investigate scenarios of interest. Given the importance of the surface code, we start with studying the distribution of 4-qubit GHZ states. Next to that, we explore how the quality of the GHZ state for the best protocols scales with the number of parties N. The source code of the dynamic algorithm can be found online [13].

5.6.1 Comparison with existing protocols for 4 parties

First, we investigate protocols for $N = 4$. In Fig. 5.6 we compare the best protocols that our dynamic algorithms find for parameters $(N,K) = (4,14)$, $(N,K) = (4,22)$ and $(N,K) = (4,42)$ with the Expedient and Stringent protocols from Nickerson *et al.* [9]. The figure shows the infidelity (one minus the fidelity) of the output GHZ state as a function of the fidelity of the input Bell pairs. We see that under the conditions considered here, the new protocols create higher-quality GHZ states with the same or even with a smaller number of Bell pairs. On the other hand, while Expedient and Stringent require $Q = 3$ qubits per node, these protocols typically require more qubits per node. For example, the $(N,K) = (4,14)$ protocol found with our dynamic program requires that two of the nodes have four qubits, as can be seen in Fig. 5.7. The best $(N,K) = (4,22)$ protocol found by the random dynamic program can be performed if all four nodes have four qubits (see Fig. 5.7), and the best $(N,K) = (4,42)$ found can be achieved with $Q = 5$ (see Fig. 5.7).

Let us now investigate to which degree the new protocols achieve these higher fidelities consuming a smaller amount of resources. We note that K represents the *minimum* number of Bell pairs needed to generate a GHZ state. These protocols are probabilistic, and the success probability depends on the fidelity of the states, going from one if the states are perfect to zero if the measured state is a minus-one eigenstate of one of the measurement operators. If the protocol fails at some step, the step needs to be run again from the beginning. This means that the average number of Bell pairs might be different from K.

A related figure of merit that might be more relevant in practice is the average number of entanglement generation steps. For this, we assume that network nodes can generate a Bell pair deterministically with one other node over some unit time step. Hence different pairs of nodes can generate entanglement in parallel over some unit time step. For instance, the two left nodes in Fig. 5.7 can generate entanglement in parallel to the right nodes. If the duration of the time step is qualitatively larger than the gate time, the number of entanglement generation time steps represents to first-degree approximation of the duration of the protocol.

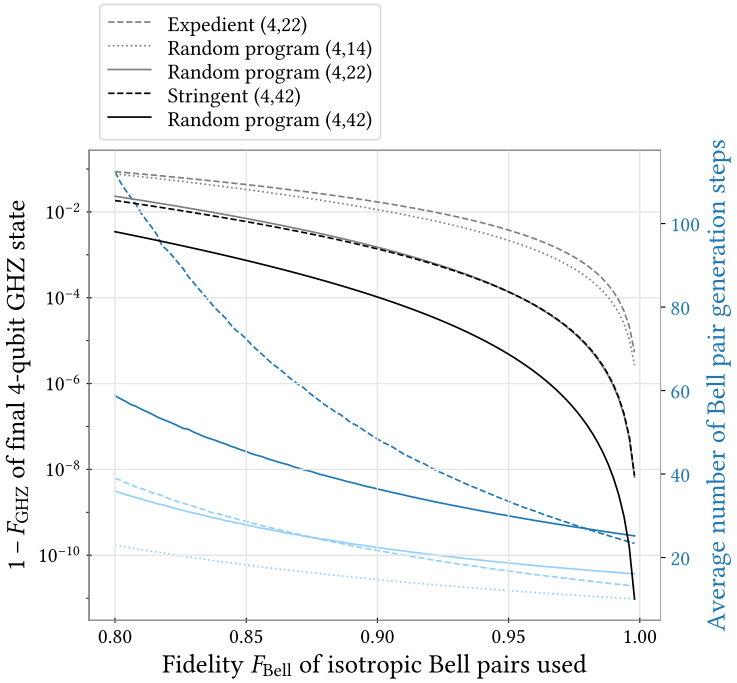


Figure 5.6: Comparison between the Expedient and Stringent protocols [9] (see Fig. 5.2), and the best algorithms found with the dynamic programs for $(N,K) = (4,14)$, $(N,K) = (4,22)$ and $(N,K) = (4,42)$. The protocols are found with the randomized version of the dynamic program, using the settings and parameters discussed in Fig. 5.5. The blue lines show the average number of Bell pair generation steps for each of the protocols (y-axis on the right). To calculate this metric we take the creation of one Bell pair as one time step, and neglect the duration of all other elements of the protocols. The averages are calculated by executing the protocols 10^5 times for each fidelity.

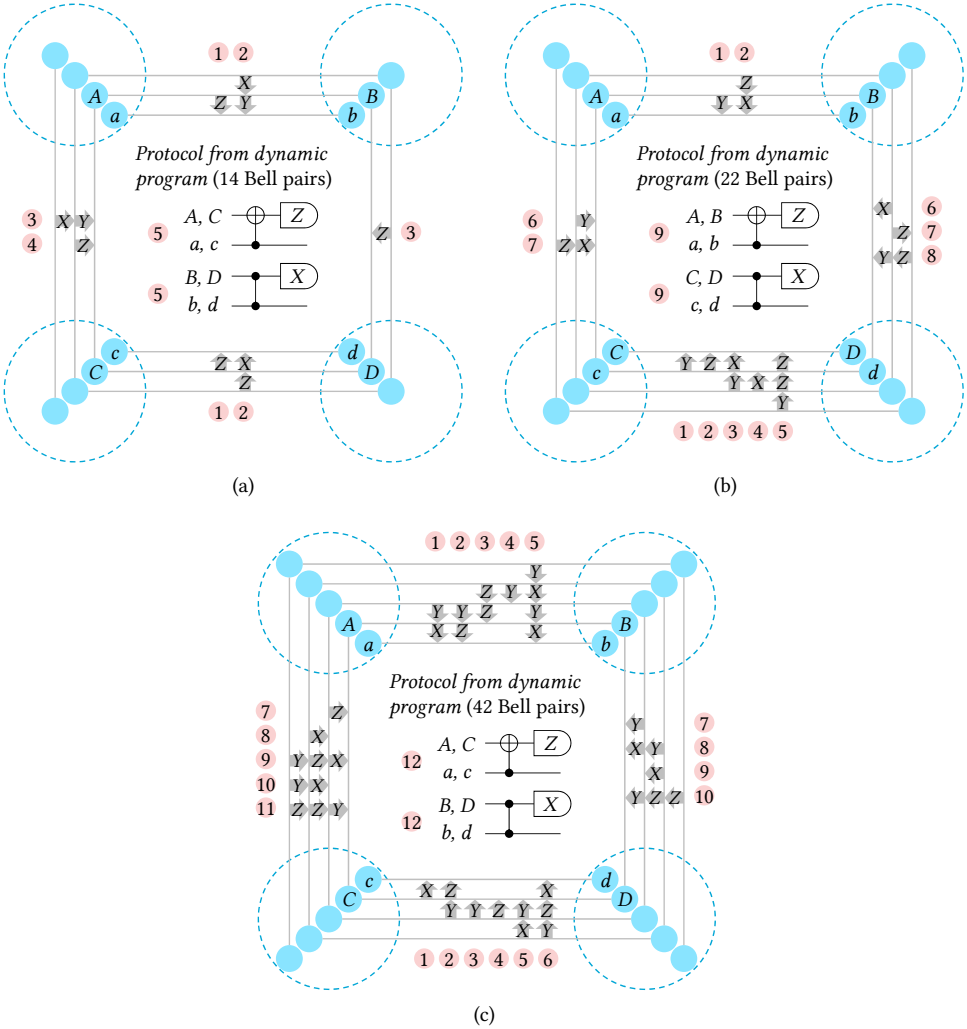


Figure 5.7: Protocols for creating and purifying a 4-qubit GHZ diagonal state out of 14, 22, and 42 Bell diagonal states shared between 4 network parties, found with the randomized version of the dynamic program presented in this chapter. See Fig. 5.2 for more information about the notation.

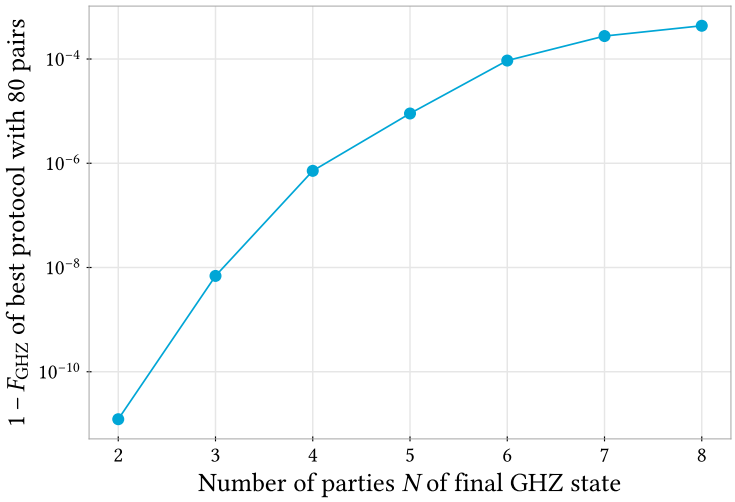


Figure 5.8: Infidelity ($1 - F_{\mathrm{GHZ}}$) of the best protocols found for an N-qubit GHZ as a function of N and input Bell pair fidelity $F_{\mathrm{Bell}} = 0.9$ and $K = 80$. The protocols are found with the randomized version of the dynamic program, using the settings and parameters discussed in Fig. 5.5.

We find that K serves as a good indicator of the average number of entanglement generation steps. We see in Fig. 5.6 that lower values of K correspond with a lower average number of generation steps. Moreover, minimizing K leads to a reduction in the average number of generation steps. Interestingly, the average number of generation steps of Stringent and Expedient cross with some of the new protocols. In particular, Stringent—a $(4, 42)$ protocol—crosses with the new $(4, 42)$ protocol, and Expedient—a $(4, 22)$ protocol—crosses with the new $(4, 22)$ protocol. The reason for this is the higher symmetry in the structure of Expedient and Stringent (see Fig. 5.2). In particular, they use the same number of Bell pairs at opposite sites of the network, whereas the new protocols contain small asymmetries in this respect. For very high input fidelities the success probabilities of all distillation steps are close to one, leading to a lower average number of entanglement generation steps.

5.6.2 Results for a large number of parties

In this section, we investigate the trade-offs between the number of parties and the number of Bell pairs for $N > 4$.

First, for a fixed number of Bell pairs, we investigate how the GHZ fidelity drops as we increase the number of parties. In particular, we fix $K = 80$ and vary N from $N = 2$ to $N = 8$. In Fig. 5.8 we show the fidelity F_{GHZ} of the final N-qubit GHZ state as a function of N for a fixed Bell pair fidelity $F_{\mathrm{Bell}} = 0.9$. We see that even for this high number of input pairs, the output fidelity drops sharply with the number of parties.

Second, we invert the question and investigate how many Bell pairs are necessary to achieve a fixed target GHZ fidelity $F_{\mathrm{GHZ}} = 0.999565$ for a different target number of parties.

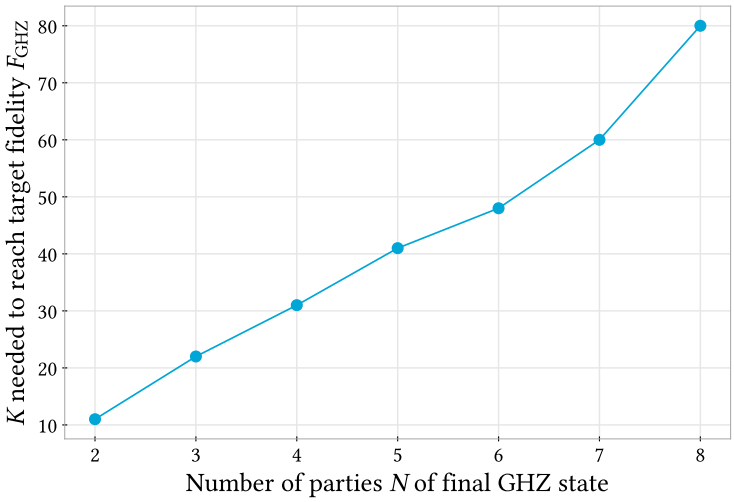


Figure 5.9: Number K of isotropic Bell pairs with $F_{\mathrm{Bell}} = 0.9$ needed to make an N-qubit GHZ state with $F_{\mathrm{GHZ}} \geq 0.999565$ as a function of N. This is the best F_{GHZ} found for an $(8, 80)$ GHZ state with the random dynamic program in all our attempts. The protocols are found with the randomized version of the dynamic program, using the settings and parameters discussed in Fig. 5.5.

This is the best F_{GHZ} found for an $(8, 80)$ GHZ state with the random dynamic program in all attempts. In Fig. 5.9, we show the number of Bell pairs K with fidelity $F_{\mathrm{Bell}} = 0.9$ needed to create an N-qubit GHZ state with fidelity $F_{\mathrm{GHZ}} \geq 0.999565$. We observe that, for the available data, the number of pairs scales roughly linearly with the number of parties.

5.7 Conclusion

In this chapter, we search for protocols that generate high-fidelity GHZ states out of non-perfect Bell pairs. The goal is to minimize the number of Bell pairs for creating a high-quality GHZ state. We do this by using a dynamic program to search the protocol space, allowing two types of operations: fusion and distillation.

We find protocols that, in situations without gate and memory noise, distill GHZ states with higher fidelity compared to previously known protocols. Compared to previous research [9], the protocols found require roughly half the number of pairs to achieve a similar fidelity. Out of the different algorithm variants that we implement, the randomized version finds the best protocols in most regimes. For $N = 2$ to $N = 8$ parties involved, we investigate how the fidelity of the final GHZ state decreases by increasing the number of parties with a fixed number of pairs ($K = 80$) and calculate how many Bell pairs are needed for the distribution of an N-qubit GHZ state of a fixed fidelity. Our programs can be used to find protocols for an arbitrary number of parties and entangled states involved.

GHZ states are required for implementing error-correction codes in distributed quantum computing. The distributed implementation of codes beyond the surface code has not been thoroughly explored. However, the case for the surface code in distributed implementations

is weaker [14], as richer connectivities than direct neighbors can be achieved with relative ease. Our tools open the door to implementations of alternative quantum error-correction codes that require high-quality GHZ states of weights different than four.

Whereas these results are promising, they do not quantify the precise effect on the noise threshold for distributed implementations of the surface code. In Ch. 6, we evaluate how a subset of the new protocols behaves in more realistic scenarios by converting them to a set of instructions. In Ch. 7, we present a follow-up of the search for GHZ protocols including noise and loss. For the situation considered here, increasing the number of Bell pairs to perform additional distillation steps always increases the fidelity of the final states. This is no longer the case for scenarios that include memory decoherence or gate noise. Hence, as becomes clear in Ch. 6, for noisy settings, the quality of the memory determines whether including more distillation steps leads to an increase in the fidelity of the final state.

Acknowledgments

The authors would like to thank Tim Coopmans, Stacey Jeffery, Tim Taminiau, and Filip Rozpedek for helpful discussions and feedback. This work was supported by the Netherlands Organization for Scientific Research (NWO/OCW), as part of the Quantum Software Consortium Program under Project 024.003.037/3368. This article was presented in part at the "Quantum Resource Estimation Workshop" as part of the International Symposium on Computer Architecture, Valencia, Spain, May 2020.

References

[1] P. C. Humphreys, N. Kalb, J. P. J. Morits, R. N. Schouten, R. F. L. Vermeulen, D. J. Twitchen, M. Markham, and R. Hanson, "Deterministic delivery of remote entanglement on a quantum network," *Nature*, vol. 558, pp. 268–273, June 2018.

[2] N. Kalb, A. A. Reiserer, P. C. Humphreys, J. J. Bakermans, S. J. Kamerling, N. H. Nickerson, S. C. Benjamin, D. J. Twitchen, M. Markham, and R. Hanson, "Entanglement distillation between solid-state quantum network nodes," *Science*, vol. 356, pp. 928–932, June 2017.

[3] N. H. Nickerson, J. F. Fitzsimons, and S. C. Benjamin, "Freely scalable quantum technologies using cells of 5-to-50 qubits with very lossy and noisy photonic links," *Physical Review X*, vol. 4, p. 041041, Dec. 2014.

[4] M. Hillery, V. Bužek, and A. Berthiaume, "Quantum secret sharing," *Physical Review A*, vol. 59, pp. 1829–1834, Mar. 1999.

[5] M. Christandl and S. Wehner, "Quantum anonymous transmissions," in *Advances in Cryptology - ASIACRYPT 2005* (B. Roy, ed.), Lecture Notes in Computer Science, (Berlin, Heidelberg), pp. 217–235, Springer, 2005.

[6] P. Kómár, E. M. Kessler, M. Bishof, L. Jiang, A. S. Sørensen, J. Ye, and M. D. Lukin, "A quantum network of clocks," *Nature Physics*, vol. 10, pp. 582–587, Aug. 2014.

[7] E. T. Khabiboulline, J. Borregaard, K. De Greve, and M. D. Lukin, "Optical interferometry with quantum networks," *Physical Review Letters*, vol. 123, p. 070504, Aug. 2019.

[8] F. Rozpędek, T. Schiet, L. P. Thinh, D. Elkouss, A. C. Doherty, and S. Wehner, "Optimizing practical entanglement distillation," *Physical Review A*, vol. 97, June 2018.

[9] N. H. Nickerson, Y. Li, and S. C. Benjamin, "Topological quantum computing with a very noisy network and local error rates approaching one percent," *Nature Communications*, vol. 4, p. 1756, Dec. 2013.

[10] C. E. Bradley, J. Randall, M. H. Abobeih, R. C. Berrevoets, M. J. Degen, M. A. Bakker, M. Markham, D. J. Twitchen, and T. H. Taminiau, "A ten-qubit solid-state spin register with quantum memory up to one minute," *Physical Review X*, vol. 9, Sept. 2019.

[11] K. Goodenough, D. Elkouss, and S. Wehner, "Optimizing repeater schemes for the quantum internet," *Physical Review A*, vol. 103, p. 032610, Mar. 2021.

[12] L. Jiang, J. Taylor, N. Khaneja, and M. Lukin, "Optimal approach to quantum communication using dynamic programming," *Proceedings of the National Academy of Sciences*, vol. 104, pp. 17291–17296, Oct. 2007.

[13] S. de Bone, R. Ouyang, K. Goodenough, and D. Elkouss, "Data underlying the publication: Protocols for creating and distilling multipartite GHZ states with Bell pairs." 4TU.ResearchData, https://doi.org/10.4121/12936761, Oct. 2021.

[14] E. T. Campbell, B. M. Terhal, and C. Vuillot, "Roads towards fault-tolerant universal quantum computation," *Nature*, vol. 549, pp. 172–179, Sept. 2017.

6

Simulating GHZ protocols in the presence of memory decoherence

In this chapter, we introduce a novel heuristic approach designed to optimize the performance of GHZ creation and distillation protocols under memory decoherence. Introducing limiting coherence times as a prominent noise factor makes it imperative to integrate realistic operation times into simulations and incorporate strategies for operation scheduling. Our methodology converts GHZ protocols from Ch. 5 into a practical set of instructions—demonstrating, through simulations, the production of GHZ states of higher quality than with previously known protocols. This advancement contributes to the field of distributed quantum computing by addressing the need for high-quality entanglement required for operations between different quantum computers.

This chapter is based on S. de Bone and D. Elkouss, "GHZ distillation protocols in the presence of decoherence," *ACM SIGMETRICS Performance Evaluation Review*, vol. 51, no. 2, pp. 81-83, Oct. 2023.

6.1 Introduction

As discussed in Sec. 5.1, combining a distributed quantum computer with an error-correction code requires entangled states to perform the syndrome measurements that detect errors. For the standard distributed surface code, with each data qubit in a different quantum computer, we require $|\mathrm{GHZ}^{(4)}\rangle = (|0\rangle^{\otimes 4} + |1\rangle^{\otimes 4})/\sqrt{2}$ states. In such a system, one GHZ state is generated and consumed per parity and measurement cycle. In Secs. 3.6.9 and 5.4, we discuss protocols introduced by Nickerson *et al.* [1, 2] to generate GHZ states in this context. In the absence of memory decoherence, these protocols produce high-quality GHZ states. However, their large number of distillation steps makes them less useful in practical devices with limited coherence times [3]. Therefore, there is a need for GHZ protocols that perform better in realistic scenarios.

In Sec. 5.3, we introduce techniques to construct abstract GHZ protocols from fusion and distillation operations. These protocols can be carried out in multiple ways. In particular, the ordering of the operations can have a strong effect on their performance in the presence of decoherence. Here, we present a method for converting GHZ distillation protocols into a set of ordered instructions. We couple this method with GHZ protocols found with the dynamic program for GHZ creation in Sec. 5.5 and show that, in the presence of memory decoherence, the resulting GHZ states have higher fidelities than previously known protocols.

6.2 Binary tree protocols

We consider GHZ creation protocols over N parties in a network: the *network nodes* $\{v^{(i)}\}_{i=1}^{N}$. The protocols are constructed with the methods of Alg. 1 in Sec. 5.5 and can be represented as a directed *binary tree*. In Fig. 6.1a, we show a graphical depiction of the binary tree corresponding to the *Modicum* protocol. Each node in a binary tree is either created from zero or two direct *children* of the graph. At the top of the binary tree, we find the operation that creates the final GHZ state. At each non-leave node, there is either a fusion or a distillation operation and at the leaves of the tree, we find the *elementary links* $\{e_i\}_{i=1}^{K}$. These are the Bell pairs created between two of the network nodes. The elementary links do not have children. The fusion operations $\{f_i\}_{i=1}^{\mathcal{F}}$ and distillation operations $\{d_i\}_{i=1}^{\mathcal{D}}$ in the tree each have two children. We consider distillation as a probabilistic operation that only succeeds if the measurement outcome is $m = +1$. In case of an outcome $m = -1$, we reapply all operations in sub-tree of the operation $d \in \{d_i\}_{i=1}^{\mathcal{D}}$—*i.e.*, d itself and all operations below it. More information about the specific operations applied for fusion and distillation can be found in Sec. 5.3.

Each binary tree satisfies $\mathcal{D} + \mathcal{F} = K - 1$, where $\mathcal{D}$, $\mathcal{F}$ and K are the number of distillation operations, fusion operations, and elementary links, respectively. Since creating a weight-N GHZ state requires a minimum of $N-2$ fusion operations working on $N-1$ elementary links, all additional fusion operations create entangled states used for distillation purposes. Therefore, we use K (the number of elementary links in a binary tree) as a proxy for the amount of distillation that takes place in a GHZ protocol.

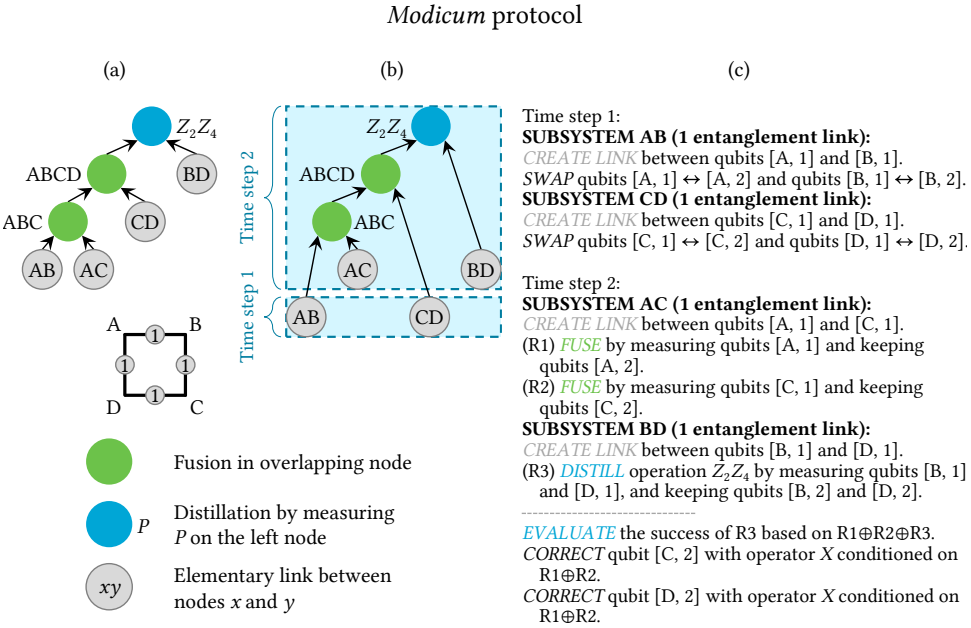


Figure 6.1: (a) Binary tree with $K = 4$ found with the dynamic program of Sec. 5.5. (b) Identified time steps for the operations in this binary tree. (c) Identified protocol recipe for this binary tree. A schematic circuit representation of this protocol can be found in Fig. 6.2.

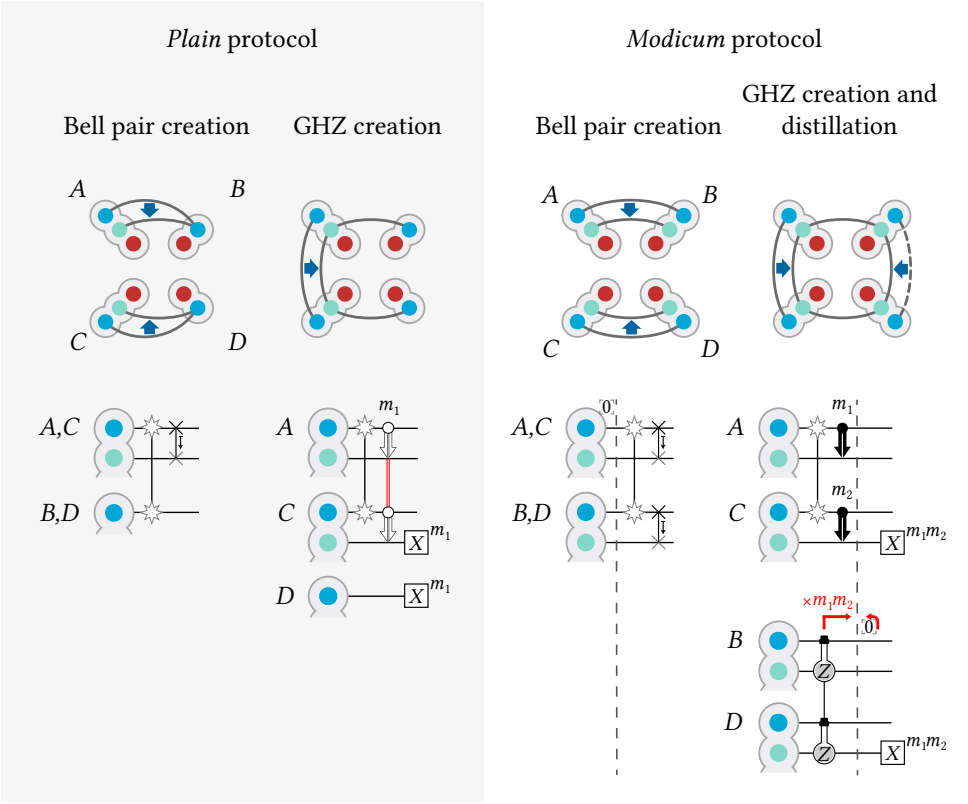


Figure 6.2: Schematic representation of the Plain and Modicum protocols that create weight-4 GHZ states. More information about the notation in these diagrams can be found in Fig. 3.28. The Plain protocol consumes 3 Bell pairs to create the GHZ state and performs no distillation. On the other hand, Modicum performs one round of distillation after the GHZ state is created and uses a minimum of 4 Bell pairs (depending on how many distillation attempts fail). The red arrow in the Modicum protocol denotes a failure reset level: it describes to where in the protocol we have to reset in case the distillation attempt fails. As mentioned earlier in Figs. 3.28 and 3.29, SWAP gates are included to satisfy the hardware restrictions of Sec. 4.1. In other situations, using these SWAP gates can possibly be avoided.

6.3 Protocol recipe construction

In the presence of memory decoherence, the order in which the operations are applied can have a strong impact on the performance. Therefore, we convert the binary tree of a GHZ creation protocol to a protocol recipe. This is a set of instructions that describes what specific operations have to be applied on what qubits of the network nodes. In this section, we include a condensed description of the algorithm we use to create a protocol recipe. In Secs. A.1.1 and A.1.2 of App. A, we discuss this algorithm in more detail. In Fig. 6.1, we depict the translation of a protocol given by a binary tree (Fig. 6.1a) into a set of time steps (Fig. 6.1b) and finally into a protocol recipe (Fig. 6.1c).

6.3.1 Operation identification and ordering

The first step of the protocol recipe construction consists of ordering the generation of the elementary links of the binary tree. For this, we follow a recursive approach. We start at the top of the tree. At each step, we select the sub-tree with the largest size, choosing the left one in case of ties. When we reach an elementary link at one of the leaves of the tree, we add this link to an ordered list with elementary links, together with all other non-overlapping elementary links that can be carried out simultaneously. For each elementary link e that we add to the list, we check if the parent operation p of e can be applied. A parent can be applied if both of its children are contained in the list of operations. If p can be applied, we add p to the list after e. Then we check if the parent of p can be applied; if it can be added, we add it after p, *etc.*

At this stage of constructing the protocol recipe, we make use of several physically-inspired assumptions based on the systems considered in Ch. 4. For example, we assume a single communication qubit $c^{(j)}$ per node $v^{(j)}$ that can be used to perform operations and a fixed number of memory qubits $\{m_q^{(j)}\}_q$ that can be used to store states. We also assume that a node can only be involved in one entanglement operation at a time, and entanglement can only be created between two communication qubits. For this reason, SWAP gates are necessary to free up the communication qubits. For example, for a typical 2-to-1 Bell pair distillation step in the style of the DEJMPS distillation protocol [4] discussed in Sec. 3.6.2, we first create a Bell pair between two communication qubits $c^{(1)}$ and $c^{(2)}$, we swap them to the memory qubits $m_1^{(1)}$ and $m_1^{(2)}$, create a new Bell pair between communication qubits $c^{(1)}$ and $c^{(2)}$ and then carry out the distillation circuit. If distillation succeeds, this leads to a distilled Bell pair on memory qubits $m_1^{(1)}$ and $m_1^{(2)}$. If we want to further use this distilled Bell pair as an ancillary state in a later operation, we typically have to swap it back to the communication qubits. Specifically, this means that for each operation included in the list mentioned in the previous paragraph, we include a set of instructions that are physically possible with the target hardware.

More details about the ordering and identification of all required operations can be found in Sec. A.1.1 of App. A.

6.3.2 Operation scheduling

Once all operations are listed, we schedule them. The schedule of operations consists of a series of *time steps*, with each time step divided into different *time blocks*. The time blocks are created such that they can be executed in parallel in different, non-overlapping parts of the network. We schedule the operations by looping over the list of operations, and, for

each operation o, we check if o can be placed in an existing time block. This is possible if the network nodes in which o operates overlap with the network nodes in which the other operations of the time block operate. If it is not possible to add o to an existing time block, we create a new one. At the end of each time step, we add all required fusion corrections, as well as distillation operations that need to be evaluated at the end of the time step. The result is a structure that we refer to as the protocol recipe. An example of a protocol recipe can be found in Fig. 6.1c. A detailed description of the algorithm used for scheduling operations can be found in Sec. A.1.2 of App. A.

6.4 Protocol recipe simulation

In this section, we sketch the logic for the efficient simulation of a protocol recipe in the form of Sec. 6.3. More details about the implementation can be found in Sec. A.2 of App. A.

The GHZ creation protocols considered are probabilistic. To avoid situations in which they do not finish within the coherence time, we impose a *GHZ cycle time* after which the GHZ protocol is aborted. To be able to deal with memory decoherence and the cycle time, we assign an internal time variable to each network node. At the end of each time step, we synchronize the time in all nodes. This corresponds to the protocol only moving to the next time step when all operations in the current time step are finished. On top of that, if the GHZ state is successfully created before the GHZ cycle time, we add memory decoherence until the GHZ cycle time is reached. We use the GHZ cycle time in Ch. 7 but not in the results of Sec. 6.5.

During normal execution of a protocol recipe, we say that we are in *execution mode*. If, in execution mode, a distillation operation is unsuccessful, the target state of the distillation operation has to be recreated from scratch. In that case, we have to track back to an earlier stage in the protocol: the *failure reset level*. If the failure reset level is part of the same time block as where the distillation failure occurred, we can simply try to recreate the state without notifying the nodes outside the time block. If this is not the case, and we have to move back to an earlier time step, all network nodes are notified, and the protocol resets to the earlier time step. This process also involves removing all states that "sit in the way" for recreating the failed state—*i.e.*, that use memory qubits required for regenerating this state. On the other hand, all states on memory qubits that are not needed are kept, so that, when we return to the point where the distillation failure occurred, these states can be used in the remainder of the protocol. This also includes distillation operations that are only partially carried out.

If it is necessary to move back to an earlier time step after a distillation failure, we first store the time t_{fail} at which the full measurement outcome was known and enter *reconstruction mode*. In this mode, we (re)apply all operations in this time step until t_{fail} is reached—*i.e.*, if they were already simulated before the distillation operation failed, we make sure all probabilistic operations get the same outcome as in the original execution. This is done to make sure operations in all time blocks of this time step are carried out until t_{fail}. We then re-initialize execution mode and move back to the failure reset level associated with the failed distillation operation(s) to reconstruct the failed state(s).

As soon as we have successfully reconstructed these state(s), we proceed with the rest of the protocol. This means that, in execution mode, we always have to make use of a data structure that indicates which operations need to be carried out, and which operations

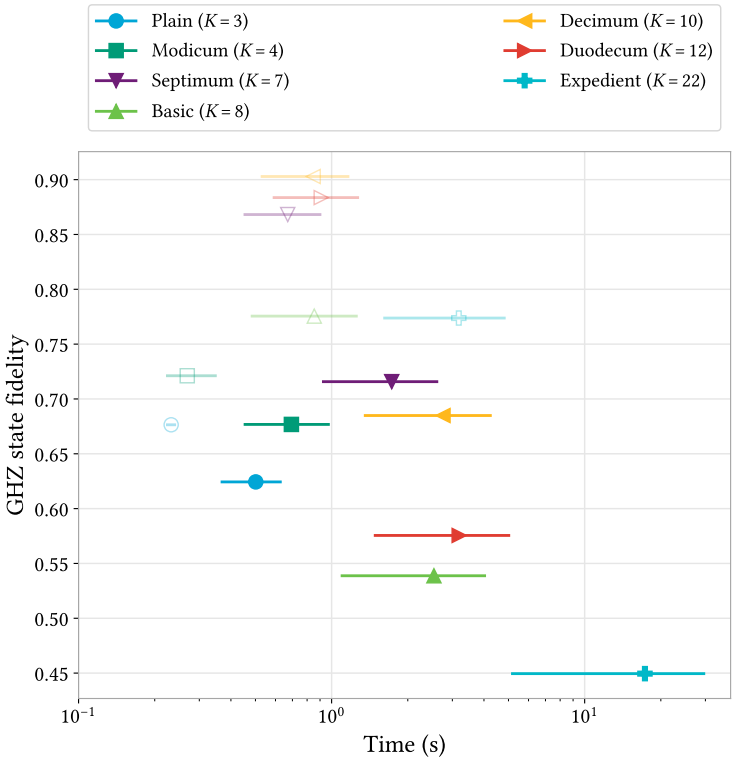


Figure 6.3: Average completion time and average GHZ state fidelity for a selection of GHZ creation protocols with $N = 4$, using the circuit simulator from Refs. [3, 5], the hardware models from Ch. 4, and the parameter values introduced in Table 4.1 for the isotopically purified NV center sample. Full markers correspond to a success probability of $p_{\text{link}} = 10^{-4}$ per Bell pair entanglement attempt, whereas open (and faded) markers correspond to a boosted success probability of $p_{\text{link}} \approx 10^{-2}$ per attempt. For the $p_{\text{link}} \approx 10^{-2}$ data, we use the same Bell pair entanglement model and parameter values, but with $\eta_{\text{ph}} = 0.4472$ and $n_{\text{DD}} = 195$. (We refer to Sec. 4.2.4 for more details about the parameter n_{DD}.) Horizontal error bars show the distribution of the completion time (68.2% confidence interval). Each data point is based on a total of 10^5 Monte Carlo simulations.

need to be skipped. This is necessary to correctly deal with the recreation of states that suffered from failed distillation earlier in the execution process. For example, after moving back to a failure reset level, only operations needed for recreating the state of the failed distillation step have to be reapplied.

6.5 Results and discussion

In this section, we simulate three protocol recipes in the form of Sec. 6.3 by executing them according to the algorithm described in Sec. 6.4. We execute these protocols using an open-source simulator [3, 5]. Since we are interested in protocols for measuring the parities of a distributed surface code, we simulate GHZ generation protocols for $N = 4$. In Fig. 6.3,

we show the average GHZ fidelity and completion time of a selection of protocols for the hardware models and parameters of the isotopically purified NV center sample introduced in Ch. 4. We show the results for the state-of-the-art Bell pair parameters of Table 4.1. These parameters lead to a success probability per entanglement attempts of $p_{\text{link}} = 10^{-4}$ and a Bell pair fidelity $F_{\text{link}} \approx 0.90$ with the single-click entanglement protocol of Eq. (4.19). Next to that, we also show results for a boosted success probability of $p_{\text{link}} \approx 10^{-2}$, achieved by increasing the photon detection probability to $\eta_{\text{ph}} = 0.4472$.

In Fig. 6.3, we include a total of seven GHZ generation protocols. These include the earlier introduced Modicum, Expedient [1], and Basic [2] protocols, as well as the *Plain* protocol. The Plain protocol contains no distillation steps: it fuses three Bell pairs into a four-qubit GHZ state. Circuit diagrams for the Plain and Modicum protocols can be found in Fig. 6.2. More information about the Expedient and Basic protocols can be found in Secs. 3.6.9 and 5.4. The remaining three protocols in Fig. 6.3 are found with the dynamic protocol for the GHZ protocol search in Ch. 7: the Septimum protocol is introduced in Fig. 7.3, and the Decimum and Duodecum protocols are introduced in App. D.

We observe that the new protocols from the dynamic program coupled with the recipe construction heuristic outperform prior protocols in the parameter regimes considered. In Ch. 7, we investigate whether the increased GHZ fidelities that we report in Fig. 6.3 are sufficient to operate the distributed surface code below its noise thresholds. We also optimize over protocols created with the dynamic program of Sec. 5.5 by making use of the algorithms described in Secs. 6.3 and 6.4 and the open-source circuit simulator [3, 5]. Even though this route is not further explored in this thesis, we repeat that both the dynamic program from Sec. 5.5 as the methods described in this chapter can be applied to a general number of network nodes N. This opens the door for simulating distributed error-correction codes beyond the square surface code.

Acknowledgements

We gratefully acknowledge support from the joint research program “Modular quantum computers” by Fujitsu Limited and Delft University of Technology, co-funded by the Netherlands Enterprise Agency under project number PPS2007. We thank SURF (www.surf.nl) for the support in using the National Supercomputer Snellius.

References

[1] N. H. Nickerson, Y. Li, and S. C. Benjamin, “Topological quantum computing with a very noisy network and local error rates approaching one percent,” *Nature Communications*, vol. 4, p. 1756, Dec. 2013.

[2] N. H. Nickerson, J. F. Fitzsimons, and S. C. Benjamin, “Freely scalable quantum technologies using cells of 5-to-50 qubits with very lossy and noisy photonic links,” *Physical Review X*, vol. 4, p. 041041, Dec. 2014.

[3] C. E. Bradley, S. W. de Bone, P. F. W. Möller, S. Baier, M. J. Degen, S. J. H. Loenen, H. P. Bartling, M. Markham, D. J. Twitchen, R. Hanson, D. Elkouss, and T. H. Taminiau, “Robust quantum-network memory based on spin qubits in isotopically engineered diamond,” *npj Quantum Information*, vol. 8, pp. 1–9, Oct. 2022.

[4] D. Deutsch, A. Ekert, R. Jozsa, C. Macchiavello, S. Popescu, and A. Sanpera, "Quantum privacy amplification and the security of quantum cryptography over noisy channels," *Physical Review Letters*, vol. 77, no. 13, pp. 2818–2821, 1996.

[5] P. Möller, S. de Bone, and D. Elkouss, "Data/software underlying the publication: Robust quantum-network memory based on spin qubits in isotopically engineered diamond." 4TU.ResearchData, https://doi.org/10.4121/16887658, July 2022.

7

Distributed surface code with memory decoherence

In the search for scalable, fault-tolerant quantum computing, distributed quantum computers based on color centers in the diamond lattice are promising candidates. These systems can be realized with photonic channels connecting closely-collocated color centers. We present numerical simulations of a memory channel using the distributed toric surface code, where each data qubit of the code is part of a separate color center, and the error-detection performance depends on the quality of four-qubit GHZ states generated between the centers. In this chapter, we quantitatively investigate the effect of decoherence and evaluate the advantage of GHZ creation protocols tailored to the level of decoherence. We do this by applying our framework with models developed from experimental characterization of nitrogen-vacancy centers, as discussed in Ch. 4. For diamond color centers, coherence times during entanglement generation are orders of magnitude lower than coherence times of idling qubits and represent a limiting factor. Our model predicts error probability thresholds for gate and measurement reduced by at least a factor of four compared to prior models which neglected decoherence. We also find a threshold of $4 \cdot 10^2$ in the ratio between the entanglement generation and the decoherence rates, setting a benchmark for experimental progress.

This chapter is based on S. de Bone, P. Möller, C.E. Bradley, T.H. Taminiau, and D. Elkouss, "Thresholds for the distributed surface code in the presence of memory decoherence," *AVS Quantum Science*, vol. 6, no. 3, p. 033801, July 2024.

7.1 Introduction

As discussed in Ch. 3, fault tolerance is naturally achieved in distributed quantum computers by establishing their connectivity according to the architecture of a topological error-correction code. In this chapter, we investigate such a system with the diamond color centers from Ch. 4 as the nodes of the distributed quantum computer. We combine these centers into a fault-tolerant quantum computer by employing memory qubits of the available nodes as data qubits of the toric surface code. We emphasize that the obtained insights are more general and that our simulation tools allow for implementing error models based on general hardware implementations.

In Ch. 3 we explain how error-correction codes hold a smaller number of logical states with many physical qubits, and unwanted errors can be identified and corrected by measuring the stabilizer operators of the code. For such a system, fault tolerance goes hand-in-hand with the existence of thresholds for local sources of error: if one manages to keep the error sources at play below their threshold values, one can make the logical error rate arbitrarily small by increasing the code space of the error-correction code.

The toric code has a depolarizing phenomenological error probability threshold of approximately 10% to 11% [1]. This error model assumes that all qubits of the code are part of the same quantum system, stabilizer measurements can be carried out perfectly, and the qubits experience depolarizing noise in between the stabilizer measurement rounds. A more precise analysis with a *circuit-level* error model yields error probability thresholds between 0.9% and 0.95% [2]. In this model, the stabilizer measurement circuit is simulated with noisy gates and measurements and it is implicitly assumed that the connectivity of the system allows direct two-qubit gates between adjacent qubits of the code topology. Therefore, this error model corresponds to a *monolithic* architecture.

If one wants to implement the toric code in a network setting, where every data qubit of the code is part of a separate network node, the stabilizer operators can be measured with the aid of four-qubit GHZ states. These GHZ states can be created by fusing three or more Bell pairs created between the involved nodes. Nickerson *et al.* [2, 3] analyzed the distributed toric code in this setting. They included protocols with a relatively large number of entanglement distillation steps that create high-quality GHZ states from imperfect Bell pairs. They found thresholds for the local qubit operations between 0.6% and 0.82% [2]—*i.e.*, slightly below the monolithic thresholds. In their threshold calculations, Nickerson *et al.* do not explicitly consider circuit operation times and do not include qubit memory decoherence during entanglement creation—*i.e.*, the notion that the quality of the code's data qubits decays over time. However, in current physical systems of interest, decoherence during entanglement creation typically constitutes a large source of error. For state-of-the-art NV centers, coherence times during optical Bell pair generation are one to two orders of magnitude lower [4–6] than estimated by Nickerson *et al.* The influence of this decoherence is further increased by the reality that success probabilities per optical Bell pair generation attempt currently fall significantly short of unity [7, 8].

Therefore, next to the errors in operations and in entangled states considered in Refs. [2, 3], decoherence of quantum states over time emerges as the third primary source of noise for accurate assessment of distributed quantum computing systems. The influence of memory qubit decoherence during entanglement creation can be captured with the *link efficiency* η^*_{link} [9]. This parameter quantifies the average number of entangled pairs that

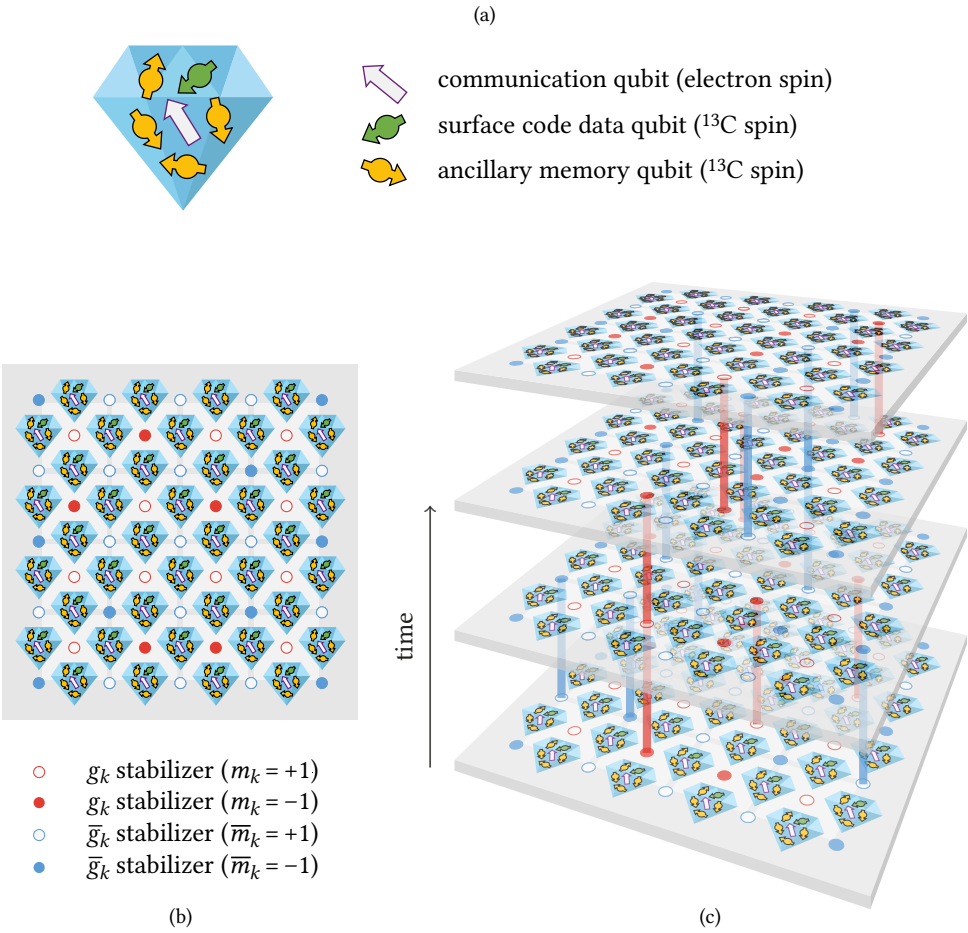


Figure 7.1: (a) Schematic impression of a diamond color center—also known as a diamond defect center. The communication qubit is used to generate Bell pairs and to perform gates on the available qubits in the diamond color center. Out of all carbon spin memory qubits available, one is selected as the data qubit of the code. The other available memory qubits are used to store intermediate entangled states during the GHZ creation process. (b) Schematic impression of how a network of diamond color centers can be used to realize a surface code on a 4×4 square lattice. Each center holds one data qubit of the error-correction code on one of its memory qubits. GHZ states are generated to measure the stabilizer operators of the code, resulting in an error syndrome of $+1$ and -1 stabilizer measurement outcomes. (c) Stabilizer operators are measured consecutively in different time layers. A flip in stabilizer measurement outcome from one layer to the next is registered. The three-dimensional error syndrome that is created in this way is fed to an error syndrome decoder to locate errors.

can be generated within the coherence times.

To investigate the influence of the coherence times, we develop a time-tracking simulator and implement realistic operation durations. Additionally, considering the pivotal role of the operation order in this new scenario, we formulate a strategy for scheduling operations. We find that, with realistic operation and coherence times, the thresholds with the GHZ generation protocols of Refs. [2, 3] disappear. In this chapter, we investigate the quantitative impact of memory decoherence and optimize over GHZ generation protocols with less distillation that can overcome this. For a range of different coherence times during entanglement generation, we find two-qubit gate error and measurement error probability thresholds for diamond color centers up to 0.24%. We find that fault tolerance is reachable with $\eta^*_{\text{link}} \approx 4 \cdot 10^2$. This improves on the prior results of $\eta^*_{\text{link}} = 2 \cdot 10^5$ for the idealized time scale estimates of Nickerson *et al.* [2]. However, this link efficiency is still above the state-of-the-art hardware reaching up to $\eta^*_{\text{link}} \approx 10$ [9].

In the remainder of this chapter, Sec. 7.2 describes GHZ creation and distillation protocols necessary for the distributed surface code. Consequently, in Sec. 7.3, we present the full cycle of stabilizer measurements of the surface code. In Sec. 7.4, we describe error models that allow us to investigate a specific hardware implementation in the distributed surface code setting: diamond color centers. Finally, in Sec. 7.5, we investigate the parameter regimes necessary for fault tolerance with these error models.

7.2 GHZ generation protocols revisited

As mentioned in Sec. 7.1, the stabilizer operators of a distributed quantum error-correction code can be measured by consuming GHZ states. In the following, we discuss protocols that create GHZ states by combining Bell pairs. We summarize prior work in Sec. 7.2.1. In Sec. 7.2.2, we discuss our method for generating GHZ protocols. This method is an extension on the dynamic program introduced in Sec. 5.5.

7.2.1 Prior GHZ protocols

In Secs. 3.6.7, 3.6.8 and 3.6.9, we discussed prior work on the generation and distillation of GHZ states. Here, we focus on protocols that combine Bell pairs into a four-qubit GHZ state and discuss seven of them, following earlier discussions in Secs. 3.6.9, 5.4 and 6.5. Below, K denotes the minimum number of Bell pairs required to generate the GHZ state and Q indicates the maximum number of qubits necessary per network node.

First, we consider the Plain protocol ($K = 3$, $Q = 2$) and the Modicum ($K = 4$, $Q = 2$) protocol introduced in Ch. 6. These protocols create a GHZ state with no distillation or only a single distillation step, respectively. The Plain protocol is the simplest protocol for creating a GHZ state from Bell pairs; it fuses three Bell pairs into a four-qubit GHZ state without any distillation. The Modicum protocol uses a fourth Bell pair to perform one round of distillation on the GHZ state. Circuit diagrams for the Plain and Modicum protocols can be found in Fig. 6.2. On top of that, we consider five GHZ protocols found by Nickerson *et al.* in the context of distributed implementations of the toric code: Expedient ($K = 22$, $Q = 3$) and Stringent ($K = 42$, $Q = 3$) from Ref. [2], and Basic ($K = 8$, $Q = 3$), Medium ($K = 16$, $Q = 4$) and Refine ($K = 40$, $Q = 5$) from Ref. [3]. More information about these protocols can be found in Secs. 3.6.9 and 5.4.

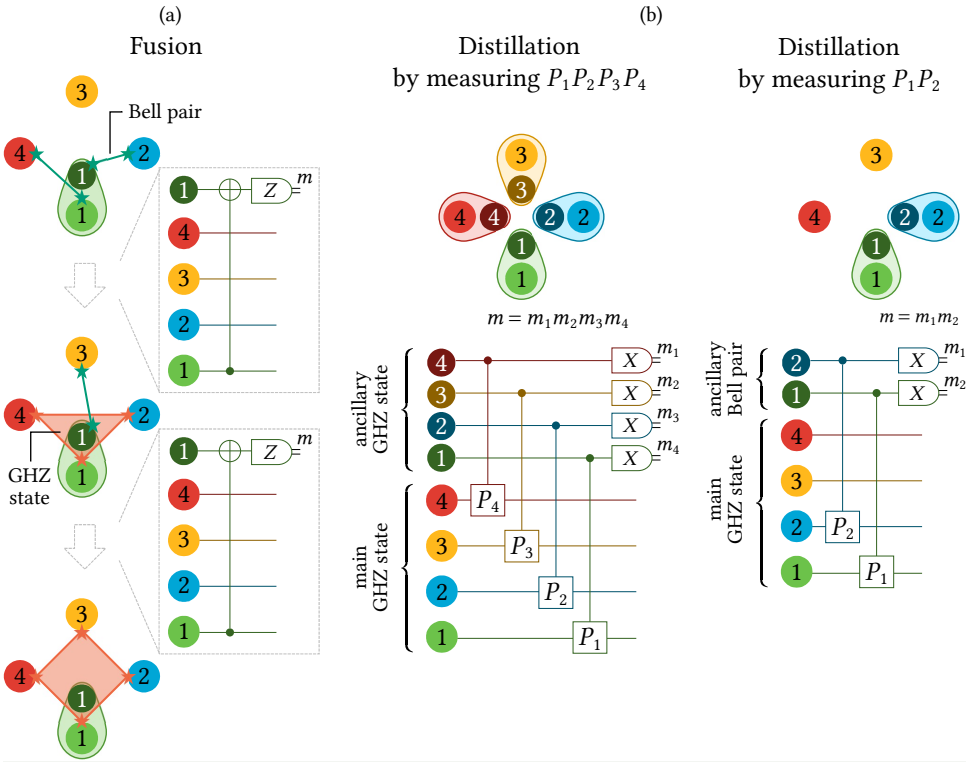


Figure 7.2: (a) Example of the fusion operation applied in the dynamic program of Alg. 2. Fusion can be applied on any two states $|\mathrm{GHZ}^{(N_1)}\rangle$ and $|\mathrm{GHZ}^{(N_2)}\rangle$ that overlap in one or more network nodes. (b) Examples of the distillation operation of Alg. 2. Distillation consists on using one state of the form $|\mathrm{GHZ}^{(N_1)}\rangle$ to measure a stabilizer of $|\mathrm{GHZ}^{(N_2)}\rangle$—this could, *e.g.*, be the operator $X_1X_2\ldots X_{N_2}$, or Z_1Z_2.

7

7.2.2 Dynamic program to optimize GHZ generation

In this section, we present a method for optimizing GHZ creation with realistic noise models. We focus on creating GHZ states of the form $|\mathrm{GHZ}^{(N)}\rangle = (|0\rangle^{\otimes N} + |1\rangle^{\otimes N})/\sqrt{2}$, where $N \in \{2,3,4\}$ represents the number of parties. For convenience, we use the notation $|\mathrm{GHZ}^{(2)}\rangle$ to describe the state of a Bell pair.

We use a dynamic program based on the dynamic program of Sec. 5.5 to optimize over the space of GHZ protocols. This program generates GHZ protocols by fusing Bell pairs to create GHZ states and distilling Bell or GHZ states by consuming other ancillary Bell pairs or GHZ states. Fig. 7.2 depicts the two building blocks—these elements are introduced in Sec. 5.3. Distillation involves the use of an ancillary state to non-locally measure a stabilizer of the main Bell or GHZ state. In this process, local control-Pauli gates between ancillary and main state qubits are followed by individual measurements of the ancillary state qubits in the Pauli-X basis. Obtaining an even number of -1 measurement outcomes marks a successful distillation attempt. If distillation fails, the post-measurement state is discarded and (part of) the protocol has to be carried out again. Fusion is executed to create GHZ states out of Bell pairs and to create a larger GHZ state. A state of the form $|\mathrm{GHZ}^{(N_1)}\rangle$ can

be fused with a state $|\mathrm{GHZ}^{(N_2)}\rangle$ by applying a CX gate between one qubit of both states and measuring out the target qubit in the Pauli-Z basis. Obtaining a $+1$ measurement outcome results in the state $|\mathrm{GHZ}^{(N_1+N_2-1)}\rangle$. A -1 measurement outcome leads to the same state after local Pauli-X corrections.

In Alg. 2, we present a schematic, pseudo-code version of the dynamic program we used to generate and evaluate GHZ protocols in this chapter. This algorithm is an expanded version of Alg. 1 in Sec. 5.5. In this algorithm, each protocol is created with either a fusion or a distillation operation that combines two smaller GHZ protocols encountered earlier in the search. The protocols created in this fashion can be depicted with a directed binary tree graph. An example graph is given in Fig. 7.3a. For the distillation steps in the binary tree diagrams, we consume the state on the right to distill the state on the left.

Each binary tree corresponds to multiple inequivalent protocols depending on the time ordering of the steps. As introduced in Ch. 6, we define a protocol recipe as a set of instructions for implementing the protocol. The recipe includes the ordering of operations and state generation. An example of a protocol recipe can be seen in Fig. 7.3c. This step was not required in previous research on distributed surface codes, as the noise models used in previous research did not include memory decoherence. Without a notion of time, the execution order of the tree's branches is irrelevant.

As can be seen in Fig. 7.3, the conversion to a protocol recipe contains SWAP gates. These gates are required to match the connectivity constraints of our example hardware model—see Secs. 4.1 and 7.4 for more details. The SWAP gates should therefore not be considered as fundamental elements of these protocols and can be circumvented or neutralized in hardware systems with more operational freedom. We implement SWAP gates as three CX gates.

Whereas we did not optimize over the conversion from binary tree to protocol recipe, we considered two heuristics to limit the influence of decoherence and SWAP gates. To limit decoherence, we prioritize creating larger branches of the tree. Here, a branch is defined as an element of the binary tree (*i.e.*, an operation in the GHZ protocol) including all elements that (in)directly point towards it. The size of the branch is the number of elements it contains. Because, generally speaking, creating small branches is faster than creating large branches, this heuristic aims to minimize waiting times for completed intermediate branches of the GHZ protocol. The SWAP gate count can be limited by making sure a Bell pair that is consumed in a distillation step is the last state to be generated. This prevents the protocol from having to swap this state back and forth between the memory. For this reason, if two branches have equal size, we prioritize creating the left branch over the right one. We discuss a concrete example of this approach in Sec. 6.3.1.

In constructing the protocol recipe, we first use these heuristics to determine the order in which the elementary Bell pairs are generated—*i.e.*, the leaves of the binary tree. By following this order, we then check for each Bell pair if other Bell pairs in non-overlapping network nodes can be generated simultaneously. Here, we prioritize based on proximity in the binary tree. We include instructions for distillation, fusion, and SWAP operations at the earliest possible point of execution. This approach gives rise to a unique conversion from binary tree to protocol recipe. More detailed descriptions of the protocol recipe construction and execution procedure can be found in Ch. 6 and App. A.

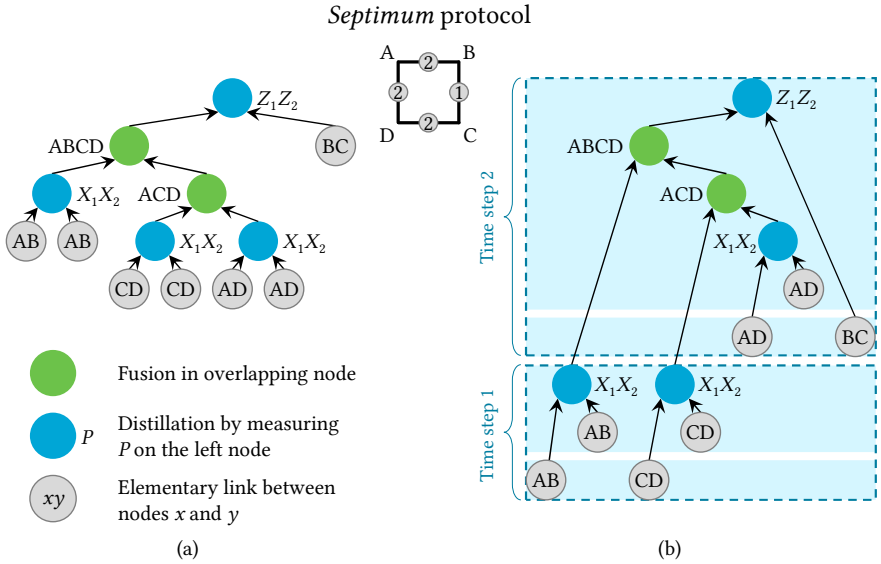
While this dynamic program explores a large class of protocols, not all of the seven

Algorithm 2: Base dynamic program for GHZ protocols search.

Data: Number N of qubits of final GHZ state
Minimum number K of Bell pairs used for final GHZ state
Set V with model parameters used for protocol evaluation
Number N_{buf} of protocols stored in buffer per step
Number $N^{(\text{so})}$ of Monte Carlo shots used per protocol

Result: Protocols to create N-qubit GHZ states by using a minimum of K Bell pairs.

for $\{(n,k)\,|\,2 \leqslant n \leqslant N, n-N+K \geqslant k \geqslant n-1\}$ **do**
  # Try all non-local measurement combinations.
  for *stabilizer* $\in$ {*stabilizers of* $|\text{GHZ}^{(n)}\rangle$ } **do**
    $n' \leftarrow$ weight of *stabilizer*
    for $k' \in [n'-1, k-n+1]$ **do**
      for $(p_1, p_2) \in [1, N_{\text{buf}}] \times [1, N_{\text{buf}}]$ **do**
        $\mathcal{P}_1 \leftarrow$ protocol p_1 in buffer at $(n, k-k')$
        $\mathcal{P}_2 \leftarrow$ protocol p_2 in buffer at (n', k')
        Construct binary tree protocol $\mathcal{P}_{\text{new}}$ that measures *stabilizer* on $\mathcal{P}_1$ by consuming $\mathcal{P}_2$
        Construct protocol recipe $\mathcal{R}_{\text{new}}$ and evaluate quality over $N^{(\text{so})}$ iterations times using V
        Store protocol if average performance is better than worst protocol in buffer
  # Try all fusion combinations.
  for $n_2 \in [2, n-1]$ **do**
    $n_1 \leftarrow n - n_2 + 1$
    for $k_2 \in [n_2 - 1, k-n+1]$ **do**
      $k_1 \leftarrow k - k_2$
      for $(p_1, p_2) \in [1, N_{\text{buf}}] \times [1, N_{\text{buf}}]$ **do**
        $\mathcal{P}_1 \leftarrow$ protocol p_1 in buffer at (n_1, k_1)
        $\mathcal{P}_2 \leftarrow$ protocol p_2 in buffer at (n_2, k_2)
        for $(i, j) \in [1, n_1] \times [1, n_2]$ **do**
          Construct binary tree protocol $\mathcal{P}_{\text{new}}$ by fusing $\mathcal{P}_1$ at qubit i and $\mathcal{P}_2$ at qubit j
          Construct protocol recipe $\mathcal{R}_{\text{new}}$ and evaluate quality over $N^{(\text{so})}$ iterations using V
          Store protocol if average performance is better than worst protocol in buffer



Time step 1:
SUBSYSTEM AB (2 entanglement links):
CREATE LINK between qubits [A, 1] and [B, 1].
SWAP qubits [A, 1] ↔ [A, 2] and qubits [B, 1] ↔ [B, 2].
CREATE LINK between qubits [A, 1] and [B, 1].
DISTILL operation X_1X_2 by measuring qubits [A, 1] and [B, 1], and keeping qubits [A, 2] and [B, 2].
SUBSYSTEM CD (2 entanglement links):
CREATE LINK between qubits [C, 1] and [D, 1].
SWAP qubits [C, 1] ↔ [C, 2] and qubits [D, 1] ↔ [D, 2].
CREATE LINK between qubits [C, 1] and [D, 1].
DISTILL operation X_1X_2 by measuring qubits [C, 1] and [D, 1], and keeping qubits [C, 2] and [D, 2].

(a)

Time step 2:
SUBSYSTEM AD (2 entanglement links):
CREATE LINK between qubits [D, 1] and [A, 1].
SWAP qubits [D, 1] ↔ [D, 3] and qubits [A, 1] ↔ [A, 3].
CREATE LINK between qubits [D, 1] and [A, 1].
DISTILL operation X_1X_2 by measuring qubits [D, 1] and [A, 1], and keeping qubits [D, 3] and [A, 3].
SWAP qubits [A, 1] ↔ [A, 3] and qubits [D, 1] ↔ [D, 3].
(R1) *FUSE* by measuring qubits [A, 1] and keeping qubits [A, 2].
(R2) *FUSE* by measuring qubits [D, 1] and keeping qubits [D, 2].
SUBSYSTEM BC (1 entanglement link):
CREATE LINK between qubits [B, 1] and [C, 1].
(R3) *DISTILL* operation Z_1Z_2 by measuring qubits [B, 1] and [C, 1], and keeping qubits [B, 2] and [C, 2].

EVALUATE the success of R3 based on R1⊕R2⊕R3.
SUBSYSTEM CD:
CORRECT qubit [C, 2] with operator X conditioned on R1⊕R2.
CORRECT qubit [D, 2] with operator X conditioned on R1⊕R2.

(c)

Figure 7.3: (a) Binary tree with $K = 7$ found with the dynamic program of Alg. 2: the *Septimum* protocol. In this directed graph, the top vertex represents the final state. Each vertex describes how its corresponding state is created from a fusion or distillation operation involving its two children. At the leaves of the graph, we find the elementary links: the Bell pairs. (b) We split the binary tree into multiple time steps that describe the order in which the protocol is carried out. The subtree involving the links between nodes C and D is identified as the part that we want to carry out first, since it is the left part of the largest branch of the binary tree. The subtree involving links between A and B is added to time step 1, as it can be carried out in parallel. (See Sec. 7.2.2 for more information.) (c) The timed binary tree is converted to an explicit set of operations: a protocol recipe. Here, we add necessary SWAP gates, conditional corrections for fusion operations, and evaluations of distillation operations. We also add instructions for distillation failure (not printed here). During the execution of this protocol, the system waits until all branches of a time step are completed before continuing to the next time step. We identify that this protocol recipe uses a maximum of $Q = 3$ qubits per network node.

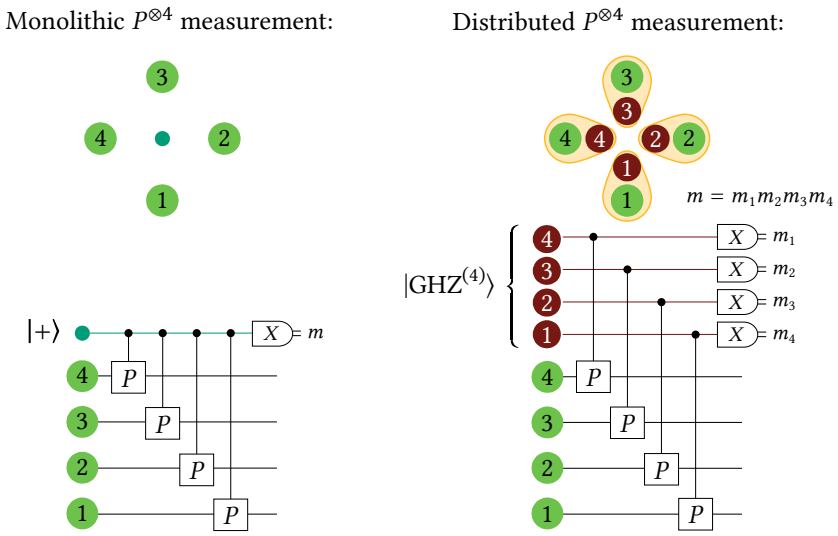


Figure 7.4: Measurement of a weight-4 stabilizer operator $P^{\otimes 4}$ of the toric surface code, both for a monolithic and a distributed implementation. Here, $P \in \{X, Z\}$ denotes both types of the stabilizer generators of the toric surface code—*i.e.*, the operators g_k and $\overline{g}_k$ in Fig. 3.6. In the monolithic version, the generators can be measured with an ancillary qubit initialized in the $|+\rangle$ state. In the distributed version, each stabilizer measurement requires the generation of an ancillary GHZ state between the nodes involved in the stabilizer measurement.

protocols that we introduced in the previous section can be generated. This is because, to suppress calculation time, the program limits distillation steps to operations that use an ancillary entangled state to non-locally measure a stabilizer operator of the main state, in a sequential manner. The protocols Refined, Expedient, and Stringent, however, make use of the double selection distillation block that does not directly fit into this stabilizer distillation framework, as elaborated in Sec. 3.6.5.

7.3 Distributed toric surface code

In this section, we discuss the steps of a distributed toric code and our approach for its simulation. We first address details regarding our distributed surface code implementation in Sec. 7.3.1. After that, we focus on simulation details and settings in Sec. 7.3.2.

7.3.1 Distributed toric code implementation

In the toric surface code, data qubits are placed on the edges of an $L \times L$ lattice with periodic boundary conditions. It encodes two logical qubit states. The stabilizers of the code come in two forms: the product of Pauli-X operators on the (four) qubits surrounding every vertex of the lattice, and the product of Pauli-Z operators on the four qubits surrounding every face (or plaquette) of the code. We introduce the surface code more formally in Sec. 3.4.4 and show a schematic representation of the toric surface code and the error-correction process in Figs. 3.5 and 3.6.

For the distributed implementation, we consider a network topology with node con-

nectivity matching the connectivity of the toric code lattice. We present a schematic impression in Fig. 7.1. Each data qubit of the toric code is placed in a separate network node—*i.e.*, in a separate diamond color center. The nodes have access to local memory qubits to create and store entangled links between them. Entangled links can be used to create four-qubit GHZ states, as described in Sec. 7.2, which are then consumed to measure the stabilizers of the code. Fig. 7.4 shows a depiction of the procedure. The outcomes of the stabilizer measurements, *i.e.*, the error syndrome, are fed to a decoder to estimate the underlying error pattern. In this chapter, we consider an implementation by Hu [10, 11] of the *Union-Find* [12] error decoder.

We point out that we simulate the toric surface code as a logical quantum *memory*, and do not consider its resilience against, *e.g.*, logical operations or operations required for initializing logical information. This means we restrict the study to the code's ability to protect general logical states. We opted for the toric surface code over the planar surface code (*i.e.*, the surface code with lattice boundaries) because, on top of the weight-4 stabilizer operators in the bulk, the planar code has lower-weight stabilizers at its boundaries. For distributed implementations, measuring these lower-weight stabilizers requires additional entanglement protocols. This makes simulating the planar code more complicated. Studies reveal that the introduction of boundaries typically has a limited effect on the code's threshold values, yet it is likely to result in a slower suppression of the logical error rates below these thresholds [13].

7.3.2 Simulation process

We split the simulation of the distributed toric code into two levels: simulation of the toric code's stabilizer measurements with the aid of GHZ states and simulation of L rounds of stabilizer measurements of the code itself.

The first level characterizes the stabilizer measurement. To this aim, we use Monte Carlo simulation to construct the (average) *Choi state* associated with using the protocol's GHZ state to measure the plaquette or star stabilizer operator on half of the maximally entangled state. Exploiting channel-state duality, the Choi state from each iteration is converted to a superoperator describing the stabilizer measurement channel. A formal introduction on channel characterization with a maximally entangled state can be found in Ref. [14]. The superoperator construction is described in detail in App. B and schematically illustrated in Fig. 7.5a.

The second level is a toric code simulator that takes as noise model the average superoperator obtained in the first level. Following previous research [2], we consider a stabilizer measurement cycle consisting of two rounds of plaquette stabilizer measurements and two rounds of star stabilizer measurements. This is because the constraint that each network node only has one single communication qubit in our example hardware model makes it impossible to simultaneously generate entanglement for overlapping stabilizers—see Secs. 4.1 and 7.4 for more details. By splitting the full cycle up into four rounds, each color center becomes part of exactly one stabilizer measurement per round. This process is schematically depicted in Fig. 7.5b. We note that a different hardware model could require, or benefit from, a different scheduling of the stabilizer measurements as the one used here.

Due to entanglement distillation steps in the GHZ creation protocol, GHZ generation is probabilistic. To fix the duration of the rounds we impose a GHZ cycle time t_{GHZ}: if GHZ

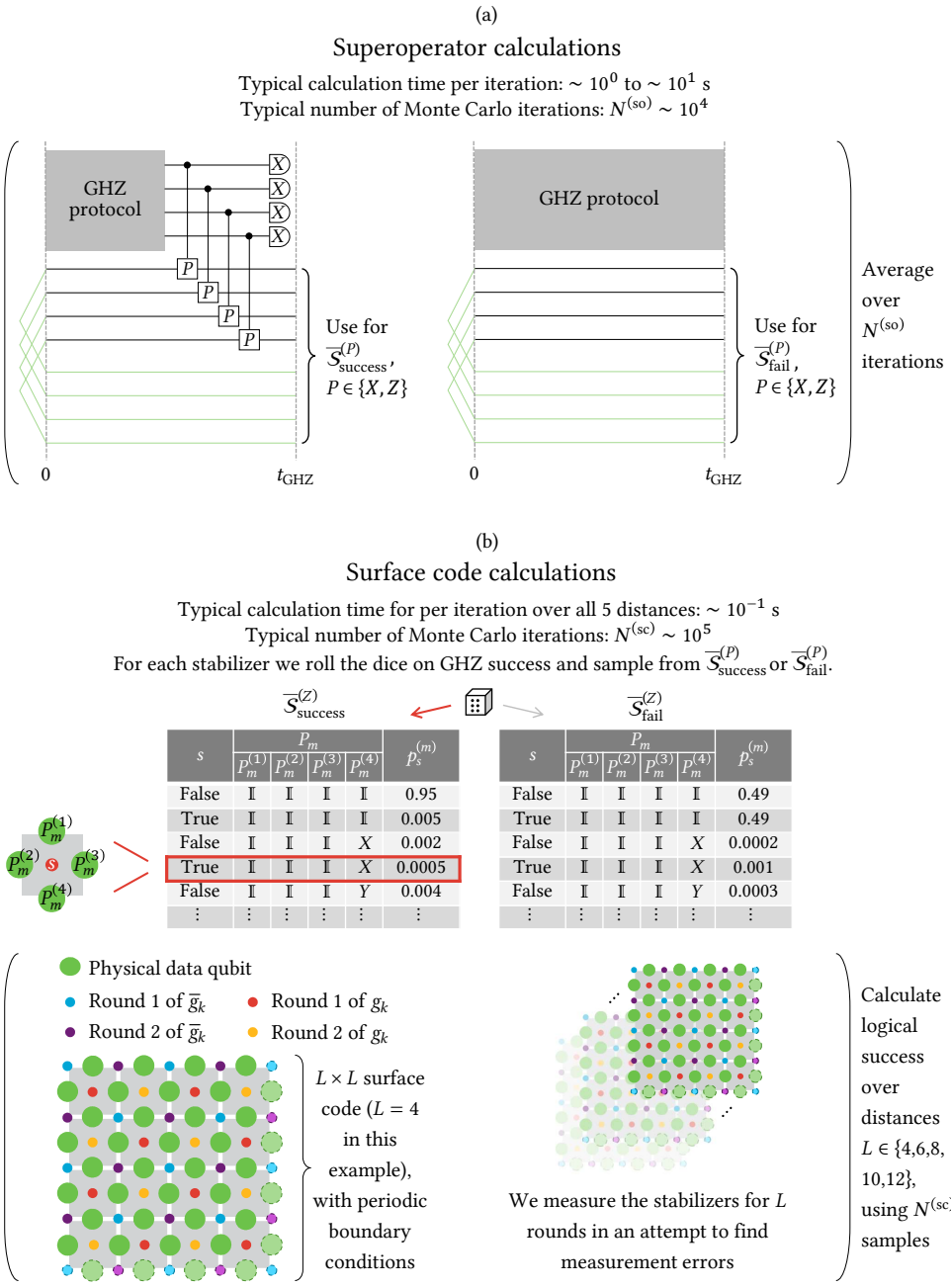


Figure 7.5: Calculation process for error probability threshold simulations of the distributed surface code. This process is called for each specific protocol recipe and parameter set combination. The full calculation consists of two levels of Monte Carlo simulations: (a) the calculation of the superoperators $\overline{\mathcal{S}}^{(P)}_{success}$ and $\overline{\mathcal{S}}^{(P)}_{fail}$, for $P \in \{X, Z\}$, and (b) the surface code simulations using these superoperators.

generation is not successful within this time, it aborts. In that case, the corresponding stabilizer operator can not be measured. This information could be given to the decoder in the form of an erasure symbol. However, to leverage existing decoders, we opt to duplicate the last measured value of the stabilizer. This choice is suboptimal and better thresholds could be expected for decoders that can handle erasures and noisy measurements.

The GHZ cycle time is a free parameter that we explore in Sec. 7.5. Based on these results, we derived a heuristic-driven approach for selecting a suitable GHZ cycle time for a specific protocol at a specific set of error probabilities. As one expects, protocols with more distillation steps K take (on average) longer to finish. This means that, compared to a protocol with a smaller K, they require a longer t_{GHZ} to reach the same GHZ completion probability p_{GHZ}. However, because decoherence plays a larger role at a higher t_{GHZ}, we typically see that the threshold values obtained with protocols with higher K peak at a lower GHZ completion probability. This becomes clear in Figs. 7.9 and 7.10, where we plot how the threshold values change when using different GHZ cycle times for four protocols with varying K. To find t_{GHZ}, we first identify an adequate GHZ completion probability as $p_{\mathrm{GHZ}}^{(\mathrm{aim})} = (100.2 - K/10)\%$ using the protocol-specific parameter K. On top of that, we use prior knowledge for a rough estimate $p_{\mathrm{th}}^{(\mathrm{est})}$ of the value of the threshold. We then determine the distribution of the protocol's duration at the error probability $p_{\mathrm{th}}^{(\mathrm{est})}$, by running it without a GHZ cycle time. Finally, using this distribution, we determine t_{GHZ} by selecting a time at which *at least* a fraction $p_{\mathrm{GHZ}}^{(\mathrm{aim})}$ of the iterations finishes.

To model GHZ generation failures, at the first simulation level, we construct two separate average superoperators per stabilizer type: a *successful* superoperator $\overline{\mathcal{S}}_{\mathrm{success}}$ for iterations where the GHZ state is created within t_{GHZ}, and an *unsuccessful* superoperator $\overline{\mathcal{S}}_{\mathrm{fail}}$ for iterations where the GHZ could not be created. Both superoperators incorporate the influence of decoherence up to the cycle time on the code's data qubits.

7.4 Diamond color center model

In this chapter, we make extensive use of the models and parameters introduced in Ch. 4. This means that we model decoherence and dynamical decoupling according to Sec. 4.2.4. We model gates and measurements according to Sec. 4.2.5 and we model Bell pair generation according to the descriptions and derivations in Secs. 4.2.2, 4.2.3, and 4.3. We assign an operation time to all operations and list the operation times in Table 7.1. We neglect the influence of classical communication times, as we consider distances between network nodes to be relatively small, but include synchronization of network nodes when classical communication is required.

We assume that each diamond color center only possesses a single communication qubit. Within each node, we assume that measurements are only possible on its communication qubit, and local (*i.e.*, intra-node) two-qubit gates always require the communication qubit to be the control qubit. These requirements mean we have to use SWAP gates to measure the state of a memory qubit or to use a memory qubit as the control of a two-qubit gate, as can be seen in Fig. 7.3. Last, we assume that a Bell pair between two centers can only be generated between the communication qubits of the two centers. We note that, in general, one could design (combinations of) diamond color center nodes with multiple communication qubits. Whereas this limits the number of SWAP gates required for distillation, it gives rise to extra requirements on the memory robustness of the communication qubits.

On top of that, we assume that the diamond color centers used in this chapter contain a natural abundance of carbon memory qubits (1.1%). This means that we trade off lower coherence times during entanglement generation for faster gates. This choice was made because it is believed that in future systems entanglement success rates are required to be several orders of magnitude higher than current state-of-the-art. In those regimes, fewer entanglement attempts are required and the influence of decoherence during entanglement generation becomes smaller. It is believed that one then benefits more from having the faster operations that samples with natural concentrations of ^{13}C atoms offer. This decision was also made because Bradley *et al.* [9] found an idling memory coherence time of 10 seconds in isotopically purified samples, which is too low for running the GHZ creation protocols described in this thesis with these samples. This coherence time—comparable to those achieved in natural abundance devices [15]—was limited predominantly by other impurities in the diamond, but the expected linear scaling of the coherence time with the isotopic concentration remains to be demonstrated in future work. In regimes with high entanglement success rates, the time duration of the GHZ creation protocols is almost solely described by the duration of the two-qubit CZ, CX, CiY, and SWAP gates, which are ≈ 50 slower for the isotopically purified samples. Thus, the operation time of any protocol also takes ≈ 50 time longer with isotopically purified samples. Equivalent performance for isotopically purified samples would require that $T2^{\mathrm{n}}_{\mathrm{idle}}$ also increases by a factor of 50—*i.e.*, from 10 seconds to 500 seconds.

Next to the state-of-the-art parameter set for the natural abundance NV center of Ch. 4, we also introduce a "near-future" diamond color center parameter set. The Bell pair parameter values of this set are based around $F_{\mathrm{prep}} = 0.999$, $p_{\mathrm{EE}} = 0.01$, $\mu = 0.95$, $\lambda = 1$ and $\eta_{\mathrm{ph}} = 0.4472$. Sec. 4.3.5 shows that the single-click protocol is the best option for the state-of-the-art parameter set. For the near-future set, the double-click model becomes favorable with $p^{(\mathrm{dc})}_{\mathrm{link}} \approx 0.1$ and $F^{(\mathrm{dc})}_{\mathrm{link}} \approx 0.953$. Setting α to get the same success probability with single-click now only leads to $F^{(\mathrm{sc})}_{\mathrm{link}} \approx 0.873$. Technically, it is possible to reach slightly higher fidelities for this parameter set with single-click, but this gives rise to success probabilities that are unusable in combination with the coherence times considered in this chapter. We list the parameter values used in this chapter in Table 7.1.

The link efficiency η^{*}_{link}, introduced in Sec. 7.1, is defined in terms of the model parameters p_{link}, t_{link}, $T1^{\mathrm{n}}_{\mathrm{link}}$ and $T2^{\mathrm{n}}_{\mathrm{link}}$ as

$$\eta^{*}_{\mathrm{link}} \equiv \frac{2p_{\mathrm{link}}}{t_{\mathrm{link}}((T1^{\mathrm{n}}_{\mathrm{link}})^{-1} + (T2^{\mathrm{n}}_{\mathrm{link}})^{-1})}. \tag{7.1}$$

7.5 Results

In this section, we investigate the sensitivity of the distributed toric surface code performance with respect to several physical parameters of the diamond color center hardware model. In particular, we investigate the influence of two-qubit gate and measurement noise, the entanglement success probability, the coherence times during entanglement generation, and the quality of the generated Bell pairs on the noise thresholds. The threshold values p_{th} are determined with fits of the logical success rates versus the lattice size (L) and local two-qubit gate and measurement error rate ($p_{\mathrm{g}} = p_{\mathrm{m}}$). The details of the fitting procedure can be found in App. C.

	State-of-the-art [9, 15]	**Fig. 7.6**	**Fig. 7.7**	**Fig. 7.8**
		Bell pair model input		
Protocol	Single-click [16]	Double-click [17]		
F_{prep}	0.99 [7, 18]	0.999		
p_{EE}	0.04 [8]	$p_{\text{EE}}(f_\phi)$	0.01	
μ	0.9 [8, 19]	0.95		
λ	0.984 [8, 20]	1		
η_{ph}	0.0046 [21]	0.4472		$\eta_{\text{ph}}(f_\eta)$
		Bell pair model output		
p_{link}	0.0001	0.1		$p_{\text{link}}(f_\eta)$
F_{link}	0.8966	$F_{\text{link}}(f_\phi)$	0.9526	
		Operation durations		
t_{link}	$6 \cdot 10^{-6}$ s			
t_{meas}	$4 \cdot 10^{-6}$ s			
$t^{\text{e}}_{X,Y}$	$0.14 \cdot 10^{-6}$ s			
$t^{\text{n}}_{X,Y}$	$1.0 \cdot 10^{-3}$ s			
$t^{\text{e}}_{Z,H}$	$0.1 \cdot 10^{-6}$ s			
$t^{\text{n}}_{Z,H}$	$0.5 \cdot 10^{-3}$ s			
$t_{CZ,CX,CiY}$	$0.5 \cdot 10^{-3}$ s			
t_{SWAP}	$1.5 \cdot 10^{-3}$ s			
		Decoherence		
$T1^{\text{n}}_{\text{idle}}$	300 s			
$T1^{\text{n}}_{\text{link}}$	0.03 s [19]	0.3 s	$0.03 f_{\text{dec}}$ s	0.3 s
$T1^{\text{e}}_{\text{idle}}$	300 s			
$T2^{\text{n}}_{\text{idle}}$	10 s			
$T2^{\text{n}}_{\text{link}}$	0.0075 s [19]	0.075 s	$0.0075 f_{\text{dec}}$ s	0.075 s
$T2^{\text{e}}_{\text{idle}}$	1.0 s			
t_{pulse}	$1.0 \cdot 10^{-3}$ s			
n_{DD}	500	18		Eq. (4.2)
		Link efficiency—see Eq. (7.1)		
η^*_{link}	$2 \cdot 10^{-1}$	$2 \cdot 10^3$	$200 f_{\text{dec}}$	$\eta^*_{\text{link}}(f_\eta)$
		Operation noise		
p_{g}	0.01	Threshold		
p_{m}	0.01			

Table 7.1: Simulation parameters used. The parameters are introduced in Ch. 4 and Sec. 7.4. More details on the values used for scaling parameters f_{dec}, f_ϕ and f_η, as well as the relations used for $p_{\text{EE}}(f_\phi)$, $F_{\text{link}}(f_\phi)$, $\eta_{\text{ph}}(f_\eta)$, $p_{\text{link}}(f_\eta)$ and $\eta^*_{\text{link}}(f_\eta)$, can be found in the captions of the respective figures.

7.5.1 Current state-of-the-art parameter set

Let us first consider a parameter set inspired by state-of-the-art NV center hardware (the first column of Table 7.1). The operation times in this set are based on typical time scales in nitrogen-vacancy centers with a natural ^{13}C concentration [9, 15, 19]—see Ch. 4 for more details. The Bell pair parameter values are a collection of the best parameters in current NV center literature—see the first column of Table 7.1 for relevant citations. In the following, we explicitly refer to this parameter set as the "state-of-the-art" parameter set. As discussed in more detail in Sec. 4.3.5, for this set the single-click entanglement protocol outperforms the double-click protocol.

We did not find a noise threshold for the state-of-the-art parameter set—neither with the existing GHZ protocols of Sec. 7.2.1 nor when optimization over GHZ protocols discussed in Sec. 7.2.2. We identify two main limitations. The first one is the link efficiency: in this regime, the average entanglement generation times are longer than coherence times during entanglement generation—*i.e.*, $\eta^*_{\text{link}} < 1$. On top of that, the Bell pair fidelity is relatively low. A low Bell pair fidelity requires complex distillation protocols to achieve high-quality GHZ states. This, in turn, magnifies the impact of decoherence.

7.5.2 Near-term parameter sets

As expected, further experimental progress and improved fidelities are required for fault-tolerant quantum computation. In the remainder of this section, we characterize two key parameters that drive the code performance in this regime. These findings can be used to guide future hardware development. Specifically, we investigate the effect of improving the Bell pair fidelity and the link efficiency.

Sensitivity to Bell pair fidelity

First, we investigate the influence of the Bell pair fidelity by using a near-future setting parameter set—see the second column in Table 7.1. Compared to the state-of-the-art parameter set of Sec. 7.5.1, in this set coherence times during entanglement creation and the photon detection probability are one and two orders of magnitude higher, respectively. The double-click entanglement protocol now gives rise to the best combination of entanglement success probability and Bell pair fidelity, as explained in more detail in Sec. 7.4. This means that these near-future parameters allow for an increase in the link efficiency by four orders of magnitude compared to the state-of-the-art parameter set of Sec. 7.5.1—see Eqs. (4.19), (4.23), and (7.1).

In our Bell pair model, several parameters contribute to the infidelity of the Bell pair states similarly—*i.e.*, through the parameter ϕ of Eq. (4.16) that captures all dephasing noise of the model. To investigate the sensitivity of the performance with respect to the Bell pair fidelity, we vary the influence of dephasing by scaling the probability of double-excitation probability and off-resonant excitation errors. These are considered one of the leading error sources in present experiments [22]. We show the results in Fig. 7.6. In this figure, the bottom of the horizontal axis indicates the Bell pair fidelity; the top indicates the corresponding excitation error probability.

We find $p_{\text{g}} = p_{\text{m}}$ thresholds between $p_{\text{th}} = 0.0066(25)\%$ for $F_{\text{link}} \approx 0.78$ and $p_{\text{th}} = 0.193(4)\%$ for $F_{\text{link}} \approx 0.96$. Interestingly, the minimum fidelity for which we find a threshold, $F_{\text{link}} \approx 0.78$, is lower than the state-of-the-art Bell pair fidelity demonstrated with both the

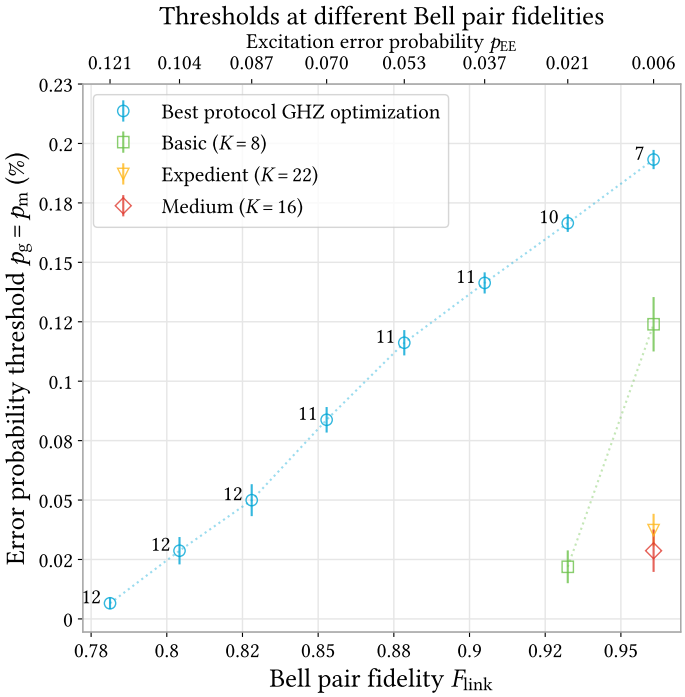


Figure 7.6: Toric surface code error probability thresholds found for $p_g = p_m$, at various Bell pair fidelities F_{link} (see Table 7.1 for the parameter values). For all points on the horizontal axis of this plot, we have set $\mu = 0.95$ and $F_{prep} = 0.999$, and varied the excitation error probability p_{EE}. This leads to different values for the parameter ϕ that describes the fidelity of the Bell pairs (see Sec. 4.3). For $f_\phi \in \{0, 1, 2, 3, 4, 5, 6, 7, 8\}$, we use $p_{EE}(f_\phi) = 1 - (\phi(f_\phi)/(\sqrt{\mu}(2F_{prep} - 1)^2))^{1/2}$, with $\phi(f_\phi) = 0.72 + 0.03 f_\phi$. In case of similar performance for the best protocols found in the GHZ optimization, we show the protocol with the lowest K value. This value is printed above the blue markers.

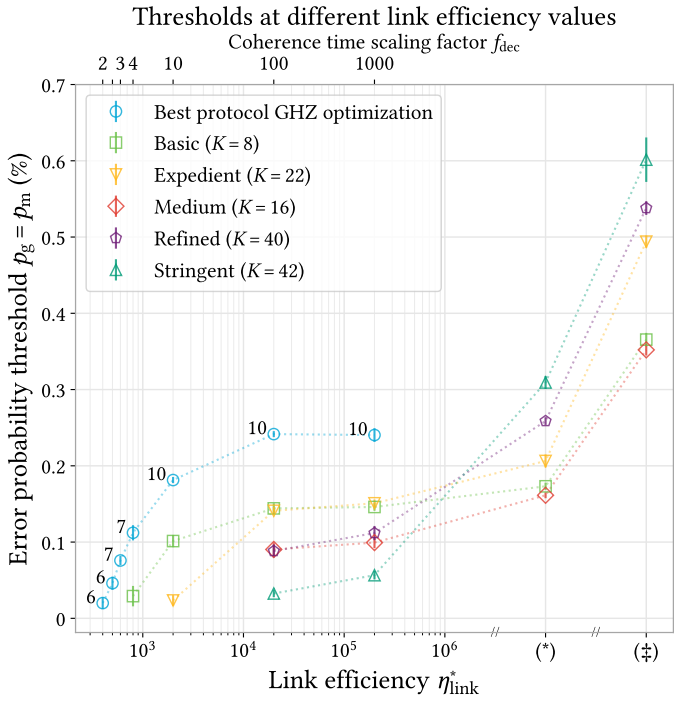


Figure 7.7: Toric surface code error probability thresholds found for $p_g = p_m$, at various values of the coherence times during entanglement generation $T1^n_{\text{link}}(f_{\text{dec}}) = 0.03 f_{\text{dec}}$ seconds and $T2^n_{\text{link}}(f_{\text{dec}}) = 0.0075 f_{\text{dec}}$ seconds. The f_{dec} factors considered are printed on the top horizontal axis of the plot. The other simulation parameters are in the third column of Table 7.1. The corresponding link efficiency is $\eta^*_{\text{link}}(f_{\text{dec}}) = 2 f_{\text{dec}} \cdot 10^2$. In case of similar performance for the best protocols found in the GHZ optimization, we show the protocol with the lowest K value. This value is printed above the blue markers. Point (*) shows calculations for a scenario without decoherence. Point (‡) shows calculations for a scenario without decoherence and with noiseless SWAP gates.

single-click and double-click protocols [7, 22]. This is possible because the link efficiency allows performing several distillation steps. We find different optimal protocols as a function of the Bell pair fidelity. In particular, we find that the optimal protocols require more distillation steps as we reduce the Bell pair fidelity, ranging from $K = 12$ for $F_{\text{link}} \approx 0.78$ to $K = 7$ for $F_{\text{link}} \approx 0.96$. We find lower thresholds as we decrease the Bell pair fidelity since the more complex distillation protocols amplify the effect of decoherence and require more gates. Furthermore, since existing GHZ creation protocols either have a small number ($K \leq 8$) or many ($K \geq 16$) distillation steps, we can understand why the new protocols with $K \in \{10, 11, 12\}$ outperform them in this regime.

Sensitivity to the link efficiency

Second, we investigate the influence of the link efficiency for near-future parameter values. In particular, we make use of a Bell pair fidelity $F_{\text{link}} \approx 0.95$—close to the highest value in the previous subsection—and we investigate two options for varying the link efficiency.

First, we vary the link efficiency by varying the coherence times during entanglement

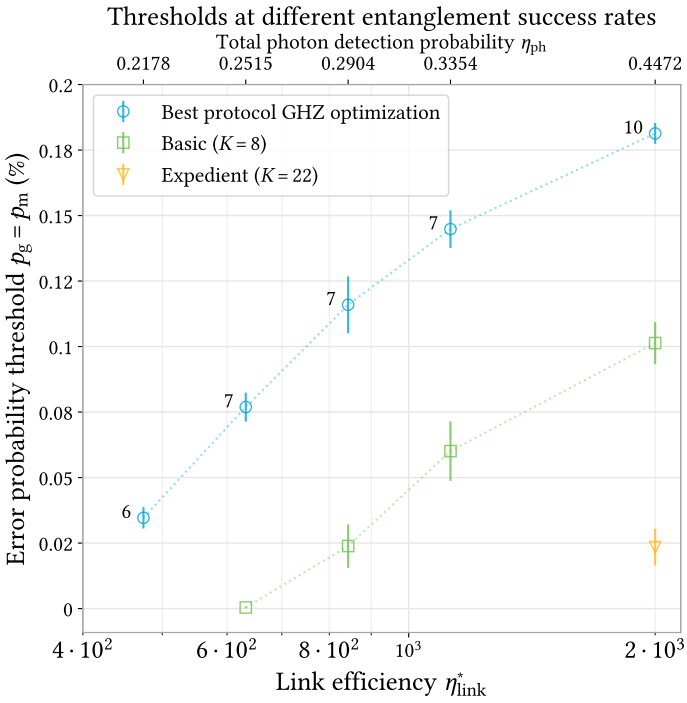


Figure 7.8: Toric surface code error probability thresholds found for $p_{\mathrm{g}} = p_{\mathrm{m}}$, at various values of the total photon detection probability η_{ph}. This parameter takes on values $\eta_{\mathrm{ph}}(f_\eta) = \sqrt{2} \cdot 10^{0.0625 f_\eta - 0.8125}$, for $f_\eta \in \{0, 1, 2, 3, 5\}$. With the double-click protocol, this gives rise to $p_{\mathrm{link}}(f_\eta) = 10^{0.125 f_\eta - 1.625}$ and $\eta^*_{\mathrm{link}}(f_\eta) = 0.002 \cdot 10^{0.125 f_\eta + 5.375}$. The other simulation parameters are in the fourth column of Table 7.1. In case of similar performance for the best protocols found in the GHZ optimization, we show the protocol with the lowest K value. This value is printed above the blue markers.

generation. For this investigation, which we report in Fig. 7.7, we use the parameter set of the third column of Table 7.1. In this set, we use a high photon detection probability $\eta_{\mathrm{ph}} = 0.4472$, leading to an optical entanglement success probability of $p_{\mathrm{link}} = 0.1$. The $p_{\mathrm{g}} = p_{\mathrm{m}}$ threshold values vary between $p_{\mathrm{th}} = 0.020(8)\%$ with coherence times corresponding to $\eta^*_{\mathrm{link}} = 4 \cdot 10^2$ and $p_{\mathrm{th}} = 0.240(9)\%$ for coherence times corresponding to $\eta^*_{\mathrm{link}} = 2 \cdot 10^5$. For coherence times corresponding to $\eta^*_{\mathrm{link}} = 3 \cdot 10^2$ and lower, we did not find thresholds. At the other end of the spectrum, we evaluate the thresholds in an idealized modular scenario: in particular, in the absence of decoherence and with perfect SWAP gates (the last two points on the horizontal axis of Fig. 7.7). The last point corresponds to a scenario similar to the one analyzed in Ref. [2]. We report a similar threshold value. For the Stringent protocol, the difference of $p_{\mathrm{th}} = 0.775\%$ reported in Ref. [2] and $p_{\mathrm{th}} = 0.601(29)\%$ found here can be attributed to the choice of the error-syndrome decoder and a reduced number of syndrome measurement cycles.

We now verify that, in this regime, similar thresholds can instead be found by varying the link efficiency via the entanglement generation rate. Specifically, we vary the entanglement success probability by adjusting the total photon detection probability η_{ph}. For this

investigation, which we report in Fig. 7.8, we use the parameter set in the fourth column of Table 7.1. This set contains coherence times during entanglement generation that are ten times higher than the state-of-the-art coherence times of Sec. 7.5.1 [19]. We find $p_g = p_m$ thresholds between $p_{th} = 0.035(4)\%$ for a photon detection probability corresponding to $\eta^*_{link} \approx 4.7 \cdot 10^2$ and $p_{th} = 0.181(4)\%$ for a photon detection probability corresponding to $\eta^*_{link} = 2 \cdot 10^3$. At photon detection probability corresponding to $\eta^*_{link} \approx 3.6 \cdot 10^2$ and lower, we are not able to find threshold values.

The second investigation gives rise to a similar required link efficiency ($\eta^*_{link} \approx 4.7 \cdot 10^2$) as the first investigation ($\eta_{link} \approx 4 \cdot 10^2$). The small difference can be attributed to the slightly larger influence of the idling coherence time $T2^n_{idle}$ in a scenario with a smaller entanglement rate. This shows that the link efficiency captures the key trade-off between cycle duration and decoherence rate, even when experimental overhead such as dynamical decoupling is accounted for. Furthermore, we find that the parameter set used determines which GHZ protocol works the best. However, for a large range of parameters close to the state-of-the-art set, one protocol with $K = 7$ performs the best. We call this protocol *Septimum* and detail it in Fig. 7.3. In particular, this protocol is (one of) the best-performing protocol(s) at $F_{link} \approx 0.96$ in Fig. 7.6, in the range $5 \cdot 10^2 \lessapprox \eta^*_{link} \lessapprox 2 \cdot 10^3$ in Fig. 7.7, and in the range $6.3 \cdot 10^2 \lessapprox \eta^*_{link} \lessapprox 1.1 \cdot 10^3$ in Fig. 7.8. We identify four additional well-performing protocols found with our dynamic program in App. D.

7.5.3 GHZ cycle time sensitivity

In the following, we investigate the sensitivity of threshold values to the GHZ cycle time and the associated GHZ completion probability. We present the results in Fig. 7.9. In this figure, we see a clear dependence of the optimal GHZ completion probability on protocol complexity. In particular, protocols that take longer to finish (*i.e.*, protocols with more distillation steps) peak at lower GHZ completion probabilities than those that finish faster, due to their increased susceptibility to decoherence. We see that for a protocol with relatively small K, GHZ cycle times that correspond to GHZ completion probabilities between 99.2 and 99.8% give rise to the highest threshold values in the parameter regimes considered here, whereas protocols with a large K peak at GHZ completion probabilities between approximately 92.5 and 98.5%.

We notice that, for some GHZ protocols, noise thresholds are found at relatively low GHZ completion probabilities of 90% and lower. This behavior can be directly attributed to the decoder heralding failures in the GHZ generation: as mentioned in Sec. 7.3.2, we utilize the stabilizer outcome from the previous time layer if a GHZ protocol does not finish within the GHZ cycle time, as opposed to naively performing the stabilizer measurement with the state produced by an unfinished GHZ protocol.

These results show that thresholds can strongly vary on the GHZ cycle time. For computational reasons, except for the results in this subsection, we do not optimize over the GHZ cycle time. Instead, we use a heuristic method to select this time based on the K value of each protocol. We describe this method in Sec. 7.3.2.

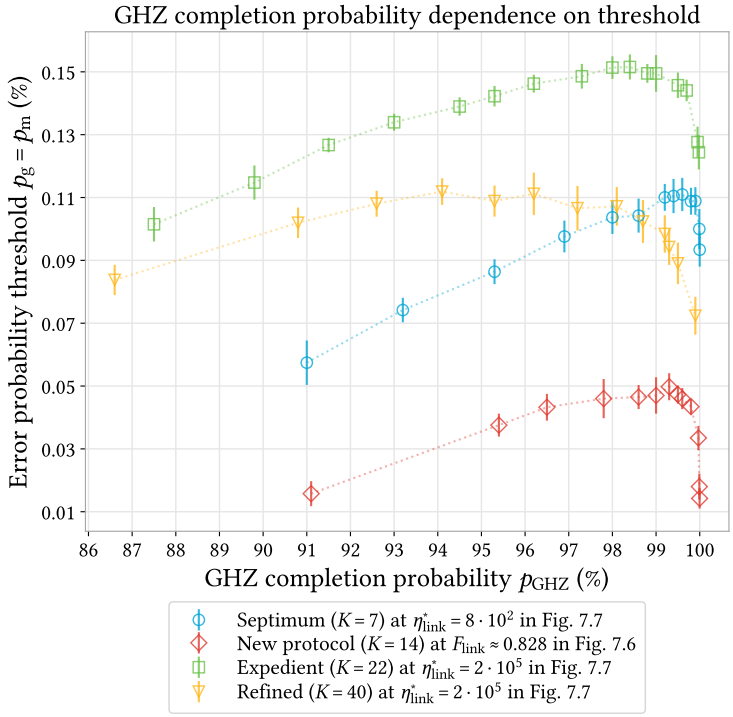


Figure 7.9: Dependence of toric surface code error probability thresholds for $p_g = p_m$ on GHZ completion probability p_{GHZ}. The dependence is plotted for four protocols with a varying number of distillation steps K. Each data point is calculated with a different GHZ cycle time t_{GHZ}. The GHZ completion probability is the probability for a protocol to finish within t_{GHZ}. In Fig. 7.10, we plot the threshold values against the specific t_{GHZ} times used to achieve these results.

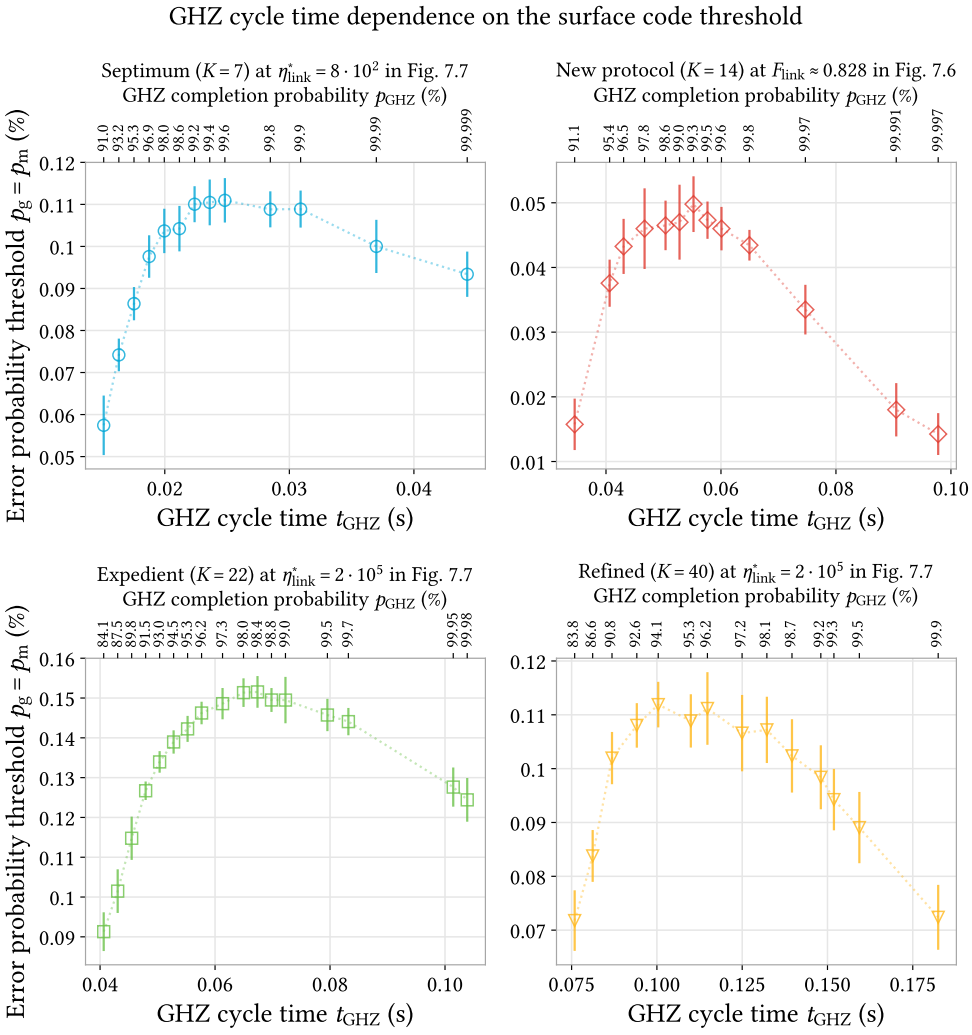


Figure 7.10: Dependence of surface code error probability thresholds for $p_{\text{g}} = p_{\text{m}}$ on GHZ cycle time t_{GHZ}. Each t_{GHZ} gives rise to probability p_{GHZ} that a protocol has to finish within t_{GHZ}: for each t_{GHZ}, the associated probabilities are printed on the top x-axis of each plot. In Fig. 7.9, we directly plot the threshold values against p_{GHZ} for the same four protocols.

7.6 Discussion: feasibility of parameter sets

The previous section shows that the state-of-the-art parameter set is above the threshold. We identified two apparent drivers for this behavior: the Bell pair fidelity and the link efficiency. The sensitivity investigation shows that with a high link efficiency, the requirements on the Bell pair fidelity are modest, while even with a high Bell pair fidelity a high link efficiency is still necessary.

Let us first discuss the experimental feasibility of the minimum link efficiency $\eta^*_{\text{link}} \gtrapprox 4 \cdot 10^2$ found in Fig. 7.8. First of all, the link efficiency can be increased by either increasing the coherence times of the data and memory qubits, or by increasing the entanglement success probability—or with a combination of both. In the previous section, we find thresholds with a high success probability ($p_{\text{link}} = 0.1$) and a modest increase in the coherence times. However, we also find that with high coherence times during entanglement generation (ten times higher than the state-of-the-art [19]) and Bell pair fidelities of $F_{\text{link}} \approx 0.95$, the total photon detection probability needs to fulfill $\eta_{\text{ph}} \gtrapprox 0.2$ (Fig. 7.8). This is a factor fifty above the state-of-the-art parameter value.

The total photon detection probability is the product of multiple probabilities (see Sec. 4.2.3). Present network experiments utilizing NV centers are particularly limited by two of these: the probability of emitting in the zero-phonon-line (ZPL, $\approx 3\%$ [23]) and the total collection efficiency ($\approx 10 - 13\%$ [7, 19]). Both values are expected to increase by Purcell enhancement of the NV center emission, for example with the use of optical microcavities [24, 25] or nanophotonic devices [26]. However, even with such devices, the feasibility remains unclear. For microcavities, predicted ZPL emission and collection probabilities are of the order 10 to 46% [24, 25] and 49% [18], respectively. Moreover, the successful integration of Purcell-enhanced and optically-coherent NV centers in nanophotonic devices remains an open research challenge due to the detrimental effects of surface charges [27].

This realization has led to an increased interest in other color centers in the diamond lattice, as, *e.g.*, SiV and SnV color centers [28]. These centers have higher intrinsic emission probabilities into the ZPL—for SnV centers this is reportedly in the area of 60% [29], whereas SiV centers approximately emit 70 to 80% into the ZPL [30]. Additionally, the inversion symmetry of SnV and SiV centers makes them less susceptible to proximal charges, facilitating integration into nanophotonic devices. Nanophotonic structures offer advantages over microcavities, such as stronger cooperativities enabled by the small mode volumes [31], and reduced sensitivity to vibrations of the cryostat hosting the emitter [24]. A disadvantage of SnV and SiV centers over NV centers is the fact that they need to be operated at lower temperatures [32] or under high strain [33, 34] to achieve similar coherence times.

Additionally, these alternative trajectories provide opportunities for "direct" GHZ generation schemes, where a GHZ state is created without Bell pair fusion [35, 36]. Contrary to the photon-emission-based Bell pair generation with NV centers, these direct GHZ state generation schemes could be based on the transmission or reflection of photons. Since, for nodes with a single communication qubit, SWAP gates are unavoidable when performing fusion, getting rid of SWAP gates during GHZ state generation could relax the requirements for, *e.g.*, the link efficiency and the photon detection probability.

7.7 Conclusion

In this chapter, we investigate the influence of decoherence and other noise sources on a fault-tolerant distributed quantum memory channel with the toric code. For this, we developed an open-source package that optimizes GHZ distillation for distributed stabilizer measurements and quantifies the impact of realistic noise sources [37]. This simulator can be considered an extension of the simulator used in Ch. 6. The GHZ protocols found with the package of Ref. [37] are compatible with a second open-source package that calculates logical error rates of the (distributed) surface code [11].

We focus our attention on a specific set of noise models inspired by diamond color centers. We first observe that state-of-the-art nitrogen-vacancy center hardware does not yet satisfy the thresholds. A parameter-sensitivity analysis shows that the main driver of the performance is the link efficiency, giving a benchmark for future experimental efforts. The photon detection probability of state-of-the-art hardware appears to represent the main challenge for operating the surface code below threshold. Sufficient photon detection probabilities could be achieved with the help of Purcell enhancement of NV center emission, or using other color centers such as silicon-vacancy centers or tin-vacancy centers. Other color centers also pave the way for schemes that directly generate GHZ states between the communication qubits of more than two nodes—*i.e.*, without fusing Bell pairs [35, 36].

With our detailed noise models, we find threshold values up to 0.24%. This is three to four times lower than prior thresholds found with less-detailed models. Similarly, the optimal distillation protocols have a small number of distillation steps compared to prior work. For a large parameter regime of parameters, a protocol consuming a minimum of seven Bell pairs is optimal. Its experimental demonstration would be an important step for showing the feasibility of this approach for scalable quantum computation.

We perform a thorough optimization of GHZ distillation protocols. However, further improvements in other elements of the distributed architecture could partially bridge the gap with the performance of monolithic architectures. For instance, the surface code decoder could model unfinished GHZ distribution rounds as erasure noise. The conversion of binary trees to protocol recipes can be optimized and Bell pair distribution could be scheduled dynamically. On top of that, since our software allows for the implementation of general hardware models, further research could focus on analyzing and understanding a broad range of physical systems in the distributed context. In addition to exploring alternative hardware systems, it would be intriguing to implement a more in-depth model of the system's micro-architecture.

Acknowledgments

The authors would like to thank Hans Beukers, Johannes Borregaard, Kenneth Goodenough, Fenglei Gu, Ronald Hanson, Sophie Hermans, Stacey Jeffery, Yves van Montfort, Jaco Morits, Siddhant Singh, and Erwin van Zwet for their helpful discussions and feedback. We gratefully acknowledge support from the joint research program "Modular quantum computers" by Fujitsu Limited and Delft University of Technology, co-funded by the Netherlands Enterprise Agency under project number PPS2007. This work was supported by the Netherlands Organization for Scientific Research (NWO/OCW), as part of the Quantum Software Consortium Program under Project 024.003.037/3368. This work was

supported by the Netherlands Organisation for Scientific Research (NWO/OCW) through a Vidi grant. This project has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No. 852410). D.E. was partially supported by the JST Moonshot R&D program under Grant JPMJMS226C. We thank SURF (www.surf.nl) for the support in using the National Supercomputer Snellius.

Author contributions

Sébastian de Bone, David Elkouss and Paul Möller designed and conceptualized the study. Conor Bradley and Tim Taminiau designed, built, and conducted experiments to characterize color centers. Conor Bradley and Tim Taminiau curated the hardware parameter sets. Sébastian de Bone built the software to generate and evaluate GHZ protocols. Sébastian de Bone and Paul Möller built the software to carry out distributed surface code simulations. Sébastian de Bone carried out and analyzed the numerical simulations. The manuscript is written by Sébastian de Bone with input and revision from all authors. David Elkouss supervised the project.

Sébastian de Bone: Conceptualization (equal); data curation (lead); formal analysis (lead); methodology (equal); software (equal); visualization (lead); writing - original draft (lead); writing - review and editing (lead). **Conor Bradley**: Resource (lead); writing - review and editing (supporting). **David Elkouss**: Conceptualization (equal); formal analysis (supporting); supervision (lead); writing - original draft (supporting); writing - review and editing (supporting). **Paul Möller**: Conceptualization (equal); methodology (equal); software (equal). **Tim Taminiau**: Resource (supporting); supervision (supporting); writing - review and editing (supporting).

References

[1] D. Browne, *Lectures on Topological Codes and Quantum Computation.* University of Innsbruck, Innsbruck, Germany, June 2014.

[2] N. H. Nickerson, Y. Li, and S. C. Benjamin, "Topological quantum computing with a very noisy network and local error rates approaching one percent," *Nature Communications*, vol. 4, p. 1756, Dec. 2013.

[3] N. H. Nickerson, J. F. Fitzsimons, and S. C. Benjamin, "Freely scalable quantum technologies using cells of 5-to-50 qubits with very lossy and noisy photonic links," *Physical Review X*, vol. 4, p. 041041, Dec. 2014.

[4] A. Reiserer, N. Kalb, M. S. Blok, K. J. M. van Bemmelen, T. H. Taminiau, R. Hanson, D. J. Twitchen, and M. Markham, "Robust quantum-network memory using decoherence-protected subspaces of nuclear spins," *Physical Review X*, vol. 6, p. 021040, June 2016.

[5] N. Kalb, A. A. Reiserer, P. C. Humphreys, J. J. Bakermans, S. J. Kamerling, N. H. Nickerson, S. C. Benjamin, D. J. Twitchen, M. Markham, and R. Hanson, "Entanglement distillation between solid-state quantum network nodes," *Science*, vol. 356, pp. 928–932, June 2017.

[6] N. Kalb, P. C. Humphreys, J. J. Slim, and R. Hanson, "Dephasing mechanisms of diamond-based nuclear-spin memories for quantum networks," *Physical Review A*, vol. 97, p. 062330, June 2018.

[7] B. Hensen, H. Bernien, A. E. Dréau, A. Reiserer, N. Kalb, M. S. Blok, J. Ruitenberg, R. F. L. Vermeulen, R. N. Schouten, C. Abellán, W. Amaya, V. Pruneri, M. W. Mitchell, M. Markham, D. J. Twitchen, D. Elkouss, S. Wehner, T. H. Taminiau, and R. Hanson, "Loophole-free Bell inequality violation using electron spins separated by 1.3 kilometres," *Nature*, vol. 526, pp. 682–686, Oct. 2015.

[8] P. C. Humphreys, N. Kalb, J. P. J. Morits, R. N. Schouten, R. F. L. Vermeulen, D. J. Twitchen, M. Markham, and R. Hanson, "Deterministic delivery of remote entanglement on a quantum network," *Nature*, vol. 558, pp. 268–273, June 2018.

[9] C. E. Bradley, S. W. de Bone, P. F. W. Möller, S. Baier, M. J. Degen, S. J. H. Loenen, H. P. Bartling, M. Markham, D. J. Twitchen, R. Hanson, D. Elkouss, and T. H. Taminiau, "Robust quantum-network memory based on spin qubits in isotopically engineered diamond," *npj Quantum Information*, vol. 8, pp. 1–9, Oct. 2022.

[10] S. Hu, "Quasilinear time decoding algorithm for topological codes with high error threshold," Master's thesis, Delft University of Technology, Delft, The Netherlands, June 2020.

[11] S. Hu, "OOP surface code simulations." `https://github.com/watermarkhu/qsurface/tree/4e31ae0`.

[12] N. Delfosse and N. H. Nickerson, "Almost-linear time decoding algorithm for topological codes," *Quantum*, vol. 5, p. 595, Dec. 2021.

[13] H. Bombín, C. Dawson, R. V. Mishmash, N. Nickerson, F. Pastawski, and S. Roberts, "Logical blocks for fault-tolerant topological quantum computation," *PRX Quantum*, vol. 4, p. 020303, Apr. 2023.

[14] M. M. Wilde, *Quantum Information Theory*. Cambridge University Press, 2013.

[15] C. E. Bradley, J. Randall, M. H. Abobeih, R. C. Berrevoets, M. J. Degen, M. A. Bakker, M. Markham, D. J. Twitchen, and T. H. Taminiau, "A ten-qubit solid-state spin register with quantum memory up to one minute," *Physical Review X*, vol. 9, Sept. 2019.

[16] C. Cabrillo, J. I. Cirac, P. García-Fernández, and P. Zoller, "Creation of entangled states of distant atoms by interference," *Physical Review A*, vol. 59, pp. 1025–1033, Feb. 1999.

[17] S. D. Barrett and P. Kok, "Efficient high-fidelity quantum computation using matter qubits and linear optics," *Physical Review A*, vol. 71, p. 060310, June 2005.

[18] F. Rozpędek, R. Yehia, K. Goodenough, M. Ruf, P. C. Humphreys, R. Hanson, S. Wehner, and D. Elkouss, "Near-term quantum-repeater experiments with nitrogen-vacancy centers: Overcoming the limitations of direct transmission," *Physical Review A*, vol. 99, p. 052330, May 2019.

[19] M. Pompili, S. L. N. Hermans, S. Baier, H. K. C. Beukers, P. C. Humphreys, R. N. Schouten, R. F. L. Vermeulen, M. J. Tiggelman, L. dos Santos Martins, B. Dirkse, S. Wehner, and R. Hanson, "Realization of a multinode quantum network of remote solid-state qubits," *Science*, vol. 372, pp. 259–264, Apr. 2021.

[20] A. Dahlberg, M. Skrzypczyk, T. Coopmans, L. Wubben, F. Rozpędek, M. Pompili, A. Stolk, P. Pawełczak, R. Knegjens, J. de Oliveira Filho, R. Hanson, and S. Wehner, "A link layer protocol for quantum networks," in *Proceedings of the ACM Special Interest Group on Data Communication*, SIGCOMM '19, (New York, NY, USA), pp. 159–173, Association for Computing Machinery, Aug. 2019.

[21] T. Coopmans, R. Knegjens, A. Dahlberg, D. Maier, L. Nijsten, J. de Oliveira Filho, M. Papendrecht, J. Rabbie, F. Rozpędek, M. Skrzypczyk, L. Wubben, W. de Jong, D. Podareanu, A. Torres-Knoop, D. Elkouss, and S. Wehner, "NetSquid, a NETwork Simulator for QUantum Information using Discrete events," *Communications Physics*, vol. 4, pp. 1–15, July 2021.

[22] S. L. N. Hermans, M. Pompili, H. K. C. Beukers, S. Baier, J. Borregaard, and R. Hanson, "Qubit teleportation between non-neighbouring nodes in a quantum network," *Nature*, vol. 605, pp. 663–668, May 2022.

[23] H. Bernien, B. Hensen, W. Pfaff, G. Koolstra, M. S. Blok, L. Robledo, T. H. Taminiau, M. Markham, D. J. Twitchen, L. Childress, and R. Hanson, "Heralded entanglement between solid-state qubits separated by three metres," *Nature*, vol. 497, pp. 86–90, May 2013.

[24] M. Ruf, M. Weaver, S. van Dam, and R. Hanson, "Resonant excitation and Purcell enhancement of coherent nitrogen-vacancy centers coupled to a Fabry-Perot microcavity," *Physical Review Applied*, vol. 15, p. 024049, Feb. 2021.

[25] D. Riedel, I. Söllner, B. J. Shields, S. Starosielec, P. Appel, E. Neu, P. Maletinsky, and R. J. Warburton, "Deterministic enhancement of coherent photon generation from a nitrogen-vacancy center in ultrapure diamond," *Physical Review X*, vol. 7, p. 031040, Sept. 2017.

[26] L. Li, T. Schröder, E. H. Chen, M. Walsh, I. Bayn, J. Goldstein, O. Gaathon, M. E. Trusheim, M. Lu, J. Mower, M. Cotlet, M. L. Markham, D. J. Twitchen, and D. Englund, "Coherent spin control of a nanocavity-enhanced qubit in diamond," *Nature Communications*, vol. 6, p. 6173, Jan. 2015.

[27] L. Orphal-Kobin, K. Unterguggenberger, T. Pregnolato, N. Kemf, M. Matalla, R.-S. Unger, I. Ostermay, G. Pieplow, and T. Schröder, "Optically coherent nitrogen-vacancy defect centers in diamond nanostructures," *Physical Review X*, vol. 13, p. 011042, Mar. 2023.

[28] C. M. Knaut, A. Suleymanzade, Y.-C. Wei, D. R. Assumpcao, P.-J. Stas, Y. Q. Huan, B. Machielse, E. N. Knall, M. Sutula, G. Baranes, N. Sinclair, C. De-Eknamkul, D. S. Levonian, M. K. Bhaskar, H. Park, M. Lončar, and M. D. Lukin, "Entanglement of

nanophotonic quantum memory nodes in a telecom network," *Nature*, vol. 629, pp. 573–578, May 2024.

[29] J. Görlitz, D. Herrmann, G. Thiering, P. Fuchs, M. Gandil, T. Iwasaki, T. Taniguchi, M. Kieschnick, J. Meijer, M. Hatano, A. Gali, and C. Becher, "Spectroscopic investigations of negatively charged tin-vacancy centres in diamond," *New Journal of Physics*, vol. 22, p. 013048, Jan. 2020.

[30] G. Moody, V. J. Sorger, D. J. Blumenthal, P. W. Juodawlkis, W. Loh, C. Sorace-Agaskar, A. E. Jones, K. C. Balram, J. C. F. Matthews, A. Laing, M. Davanco, L. Chang, J. E. Bowers, N. Quack, C. Galland, I. Aharonovich, M. A. Wolff, C. Schuck, N. Sinclair, M. Lončar, T. Komljenovic, D. Weld, S. Mookherjea, S. Buckley, M. Radulaski, S. Reitzenstein, B. Pingault, B. Machielse, D. Mukhopadhyay, A. Akimov, A. Zheltikov, G. S. Agarwal, K. Srinivasan, J. Lu, H. X. Tang, W. Jiang, T. P. McKenna, A. H. Safavi-Naeini, S. Steinhauer, A. W. Elshaari, V. Zwiller, P. S. Davids, N. Martinez, M. Gehl, J. Chiaverini, K. K. Mehta, J. Romero, N. B. Lingaraju, A. M. Weiner, D. Peace, R. Cernansky, M. Lobino, E. Diamanti, L. T. Vidarte, and R. M. Camacho, "2022 Roadmap on integrated quantum photonics," *Journal of Physics: Photonics*, vol. 4, p. 012501, Jan. 2022.

[31] A. Sipahigil, R. E. Evans, D. D. Sukachev, M. J. Burek, J. Borregaard, M. K. Bhaskar, C. T. Nguyen, J. L. Pacheco, H. A. Atikian, C. Meuwly, R. M. Camacho, F. Jelezko, E. Bielejec, H. Park, M. Lončar, and M. D. Lukin, "An integrated diamond nanophotonics platform for quantum-optical networks," *Science*, vol. 354, pp. 847–850, Nov. 2016.

[32] D. D. Sukachev, A. Sipahigil, C. T. Nguyen, M. K. Bhaskar, R. E. Evans, F. Jelezko, and M. D. Lukin, "Silicon-vacancy spin qubit in diamond: A quantum memory exceeding 10 ms with single-shot state readout," *Physical Review Letters*, vol. 119, p. 223602, Nov. 2017.

[33] Y.-I. Sohn, S. Meesala, B. Pingault, H. A. Atikian, J. Holzgrafe, M. Gündoğan, C. Stavrakas, M. J. Stanley, A. Sipahigil, J. Choi, M. Zhang, J. L. Pacheco, J. Abraham, E. Bielejec, M. D. Lukin, M. Atatüre, and M. Lončar, "Controlling the coherence of a diamond spin qubit through its strain environment," *Nature Communications*, vol. 9, p. 2012, May 2018.

[34] P.-J. Stas, Y. Q. Huan, B. Machielse, E. N. Knall, A. Suleymanzade, B. Pingault, M. Sutula, S. W. Ding, C. M. Knaut, D. R. Assumpcao, Y.-C. Wei, M. K. Bhaskar, R. Riedinger, D. D. Sukachev, H. Park, M. Lončar, D. S. Levonian, and M. D. Lukin, "Robust multi-qubit quantum network node with integrated error detection," *Science*, vol. 378, pp. 557–560, Nov. 2022.

[35] H. K. Beukers, M. Pasini, H. Choi, D. Englund, R. Hanson, and J. Borregaard, "Remote-entanglement protocols for stationary qubits with photonic interfaces," *PRX Quantum*, vol. 5, p. 010202, Mar. 2024.

[36] S. Singh, F. Gu, S. de Bone, E. Villaseñor, D. Elkouss, and J. Borregaard, *Modular Architectures and Entanglement Schemes for Error-Corrected Distributed Quantum Computation.* arXiv: 2408.02837 [quant-ph], Aug. 2024.

[37] S. de Bone, P. Möller, and D. Elkouss, "Data/software underlying the publication: Thresholds for the distributed surface code in the presence of memory decoherence." 4TU.ResearchData, https://doi.org/10.4121/708d4311-49b1-4ec2-b3cb-292d267df6be, Jan. 2024.

8

Fault-tolerant cluster states for distributed quantum computing

In this chapter, we introduce a method to construct fault-tolerant MBQC architectures and numerically estimate their performance over various types of networks. This makes it possible to construct fault-tolerant cluster states in the context of distributed quantum computation. We gauge error thresholds of the architectures with an efficient stabilizer simulator to investigate the resilience against both circuit-level and network noise. We show that an architecture based on the diamond lattice outperforms the conventional cubic lattice for both monolithic (i.e., non-distributed) and distributed implementations. Moreover, the high erasure thresholds of non-cubic lattices can be exploited further in a distributed context, as their performance may be boosted through entanglement distillation by trading in entanglement success rates against erasure errors during the error-decoding process. These results highlight the significance of lattice geometry in the design of fault-tolerant measurement-based quantum computing on a network, emphasizing the potential for constructing robust and scalable distributed quantum computers.

This chapter is based on the manuscript Y. van Montfort, S. de Bone, and D. Elkouss, *Fault-tolerant structures for measurement-based quantum computation on a network*, arXiv: 2402.19323 [quant-ph], Feb. 2024.

8.1 Introduction

Large-scale quantum computation with low error probabilities requires handling noise in a correct and efficient manner—one would like to fault-tolerantly transmit, store, and process quantum information with quantum computing hardware. As discussed in Ch. 3, quantum error-correction encompasses methods to achieve fault tolerance from faulty hardware. Topological error-correction codes, including surface codes, are a promising avenue to achieve this goal, as these codes have high error thresholds against local errors, and only require nearest-neighbor interactions between qubits in a two-dimensional layout.

As discussed in Sec. 3.4.8 and Ch. 7, surface codes achieve fault tolerance by repeatedly combining measurement outcomes of consecutive rounds of stabilizer measurements. Geometrically, the addition of a time dimension to a two-dimensional syndrome graph creates a three-dimensional structure, that may be interpreted as a noisy quantum channel that propagates logical information in the direction of time. For example, the conventional surface code may be constructed from qubits on a square lattice, but its corresponding decoding graphs are three-dimensional cubic lattices. As discussed in Sec. 3.5.2, foliation is a method to transform any surface code to a three-dimensional cluster state that forms the resource for fault-tolerant MBQC, with the same geometry as the corresponding logical quantum channel of the surface code [1, 2]. Although foliated codes can be interpreted as having replaced time with another spatial dimension, the resulting cluster states can be initiated and consumed in different directions as time progresses, lifting the rigid duality of time and space in surface codes—we elaborate on this idea in Sec. 3.5.3.

However, foliation does not exhaust all possible fault-tolerant cluster states—*i.e.*, there are *non-foliated* three-dimensional cluster states that cannot be constructed from a two-dimensional surface code. We study these types of lattices because they may produce higher fault-tolerant error thresholds compared to conventional surface codes, at least when assuming simple combinations of independent and identically distributed (i.i.d.) single-qubit and measurement errors [3]. In practice though, non-foliated lattices do not necessarily produce higher error thresholds, because faulty operations during cluster state preparation and measurement typically introduce highly-coupled and non-identically distributed errors. Both the complexity of the error decoder (through the syndrome graph) and the complexity of the error model (through the quantum circuit) eventually determine the error threshold.

In general, the *valence* v of the qubits in the cluster state provides a good indicator for the complexity of the error model, with lower valencies corresponding to smaller circuits and fewer coupled errors. Similarly, the *bond-degree* b of the vertices in the error syndrome graph proxies the complexity of the error decoder, with lower bond-degrees yielding syndromes that encode more information about the locations of errors surrounding the syndrome. Here, the valence v and bond-degree b are defined as the valency of vertices and edges in the syndrome graph, respectively, and are introduced in Fig. 3.14 for the diamond lattice. Unfortunately, these quantities suffer a trade-off in three-dimensional space [3].

Nickerson and Bombín [3] confirm that the threshold against erasure noise is primarily determined by the bond-degree b and the resilience against erasure noise is a good indicator of the performance under phenomenological noise. Because these results do not take into account the influence of the qubit valence v, and because state-of-the-art hardware is currently not capable of realizing large cluster states that can achieve sufficiently low logical

error probabilities, some research has focused its attention on modular implementations of fault-tolerant MBQC states [4–6]. The entire cluster state is prepared from resource states that are generated by small, separate devices, and entangling operations between devices create the entanglement to encode the entire cluster state. Several physical systems are suitable for modular MBQC architectures via optical interfaces [7].

In this chapter, we explore fault-tolerant cluster states under more realistic noise models compared to previous investigations [3]. This brings their realization in MBQC closer to reality. We numerically estimate fault-tolerant thresholds of previously proposed cluster states [3] for both monolithic (*i.e.*, non-distributed) and distributed implementations. We use an efficient stabilizer simulator and the Union-Find decoder [8] to evaluate the performance of our architectures. In our distributed noise models, we consider circuit-level noise during state preparation, measurement noise of single-qubit measurements, and *network noise* between nodes introduced by qubits prepared in the GHZ basis. On top of that, we investigate the first step towards including entanglement distillation for one of the distributed cluster states considered and find that including distillation is particularly effective in the context of measurement-based fault tolerance. This is due to the high erasure-type error thresholds of non-foliated lattices that may shield against probabilistic entanglement generation resulting from distillation.

In this chapter, we make extensive use of chain complex concepts discussed in Sec. 3.3. We also use Sec. 3.5, where, following the ideas brought forward by Fujii [9], these concepts are combined to construct topologically protected cluster states with the aid of the $\mathbb{Z}_2$ chain complex. In Sec. 3.5.4, the error-correction process for three-dimensional cluster states is discussed. These introductions are used in Sec. 8.2 below to construct distributed cluster states. We also use Sec. 8.2 to introduce our numerical methods. In Sec. 8.3 we present the results of the numerical simulations, and in Sec. 8.4 we conclude.

8.2 Methods

Here, we are interested in crystalline cluster states built up from cellulations of flat three-dimensional space. There are various methods to find such structures. Two approaches that have previously been used are the splitting procedures on a known (foliated) structure, such as the cubic cluster state [3], and an algebraic approach based on combinatorial tiling theory [10]. We briefly discuss the former method below and show some of the lattices found with this method in Fig. 8.1. Although we have not used the latter method to construct new cluster states, we emphasize that the zoo of fault-tolerant cluster states merits further investigation under the noise models considered here.

Below, we first discuss an extension of the concepts of the chain complex as discussed in Sec. 3.3. By adding indices that describe translational symmetry, we can use the chain complex to describe unit cells of a lattice that generate a full lattice. We discuss this method in Sec. 8.2.1. In Sec. 8.2.2, we describe the *cell-vertex splitting* operation of Nickerson and Bombín [3] in the context of this unit cell complex—this allows us to transform unit cells that describe three-dimensional lattices. In Sec. 8.2.3, we define a *face-edge splitting* operation that allows us to replace cluster state qubits with entangled states or Bell measurements. In Sec. 8.2.4, we introduce the noise models used for circuit-level and network noise and describe our method for generating entanglement in a distributed cluster state. In Sec. 8.2.5, we discuss how we use the stabilizer formalism to model and transfer Pauli errors in

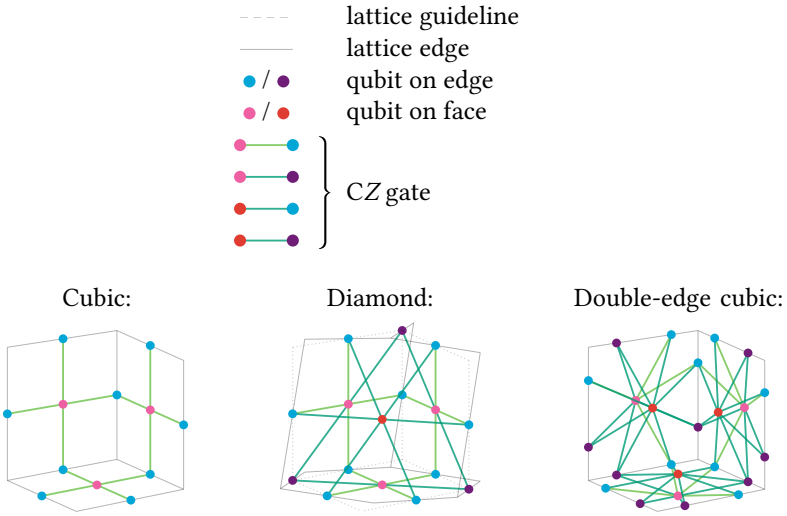


Figure 8.1: Cluster states obtained through splitting. The cubic cluster state can be created by foliating the standard toric surface code. The diamond cluster state is obtained through two splits of the primal and dual vertex in the cubic unit cell. The complex is regular and self-dual, with each face connected to six edges. The double-edge cubic cluster state is obtained through multiple simple splits of primal and dual vertices. Each face is "double"-sided, supporting two different qubit that are connected to eight surrounding edges.

the circuits that we use to construct and measure cluster states, and we elaborate on the numerical aspects of our model and simulations.

8.2.1 Unit cell complex

The *unit cell complex* is the set of basis elements (atoms) together with their boundary relations (bonds) in the crystal that forms a block with translation symmetry along the sides of the unit cell. We use the *Miller index* notation with square brackets $[abc]$ or $[\mathbf{r}]$ for a translation $\mathbf{r} \equiv a_x\mathbf{x} + a_y\mathbf{y} + a_z\mathbf{z}$ along the *lattice vectors* $\mathbf{x}$, $\mathbf{y}$ and $\mathbf{z}$ that form the sides of the unit cell. Negative indices are denoted in the usual way as $[\bar{a}_x\bar{a}_y\bar{a}_z]$ or $[\bar{\mathbf{r}}]$ for a translation $\bar{\mathbf{r}} \equiv -a_x\mathbf{x} - a_y\mathbf{y} - a_z\mathbf{z}$. The construction of the unit cell complex follows from the choice of lattice vectors. The subset of basis elements $\{(\mathbf{b}_i)_k\}_k$ in C_i that are equivalent under translations $\mathbf{r}$ is mapped to a single *quotient* element $\mathbf{q}_i$, which serves as a basis vector for a new vector space Q_i over $\mathbb{Z}_2$. Similarly, the boundary relation between elements $(\mathbf{b}_i)_n$ and $(\mathbf{b}_{i-1})_m$ is mapped to a relation to their quotient elements $(\mathbf{q}_i)_n$ and $(\mathbf{q}_{i-1})_m$ in the form of a *quotient* boundary map $\partial_i^{[\mathbf{r}]} : Q_i \mapsto Q_{i-1}$. Because some of the boundary relations are present between elements *across* two unit cells (such as a face with boundary edges from adjacent unit cells), quotient boundaries have a Miller index $[\mathbf{r}]$ that represents the translation $\mathbf{r}$ required to jump to its neighboring unit cell. Intracellular boundaries are encoded by $\partial_i^{[0]}$, whilst intercellular boundaries are encoded by $\partial_i^{[a_x a_y a_z]}$ for a non-zero translation $\mathbf{r} \equiv a_x\mathbf{x} + a_y\mathbf{y} + a_z\mathbf{z}$. A more detailed description of the quotient boundaries in the unit cell complex can be found in Sec. E.1 of App. E.

Given a unit cell complex, an *embedding* is a map that takes each Q_i and the quotient maps $\partial_i^{[\mathbf{r}]}$ to a crystalline chain complex $C_3 \to C_2 \to C_1 \to C_0$, with lattice dimensions given

by the embedding. For a periodic lattice of $N_{\text{cells}} \equiv N_{\mathbf{x}} \times N_{\mathbf{y}} \times N_{\mathbf{z}}$ unit cells along the $\mathbf{x}$, $\mathbf{y}$, and $\mathbf{z}$ lattice directions, respectively, vector spaces of chains take the form

$$C_i = Q_i^{\oplus N_{\text{cells}}} = Q_i \otimes L'. \tag{8.1}$$

Here, $L' \equiv \mathbb{Z}_2^{\oplus N_{\text{cells}}}$ is an N_{cells}-dimensional vector space over $\mathbb{Z}_2$. The N_{cells} basis vectors of L' represent the lattice points, such that linear combinations in L' may be associated with the subset of lattice points that have non-zero coefficients. Intuitively, the embedding is realized by repeating the unit cell elements Q_i over each lattice point given by a displacement vector $\mathbf{r}$. We formalize this process in Sec. E.2 of App. E.

8.2.2 Cell-vertex splitting

The cell-vertex splitting procedure of Nickerson and Bombín [3] can be phrased in terms of the unit cell complex introduced in Sec. 8.2.1. A visual example of a split in two dimensions can be seen in Sec. E.2 of App. E: splitting each face of the square lattice in Fig. E.1a diagonally results in the triangular lattice shown in Fig. E.1b. The cell-vertex splitting operation splits a vertex (*i.e.*, a dual cell) of the lattice. This creates a new lattice with a lower bond-degree b. Here, we review the general case of an n-split in three dimensions, with a simple split following from the case $n = 1$. Importantly, the splitting number n alone is not sufficient to uniquely characterize a split: one should also specify the new boundary relations between split vertices and edges. We denote the split vertex with $\mathbf{v}_0$. The recipe for an n-split is then as follows:

1. Let $E' = \{(\mathbf{v}_j, \mathbf{v}_0) \mid \mathbf{v}_j \in \mathbb{N}_0\}$ be the set of incident edges on $\mathbf{v}_0$ with neighborhood $\mathbb{N}_0$. Choose n disjoint subsets $E'_i \in E'$ ($i = 1, 2, \ldots, n$) that will each connect to a new vertex.

2. Create n new vertices $\mathbf{v}_i$, and connect each $\mathbf{v}_i$ to the incident edges E'_i. The $\mathbf{v}_0$ vertex connects to the remaining edges $E'_0 = E' \bigcup_i E'_i$, which might be the empty set.

3. Create n new edges $\mathbf{e}_i = (\mathbf{v}_i, \mathbf{v}_0)$. The corresponding boundary relations are encoded in $\partial_1^{[0]}$ of the unit cell complex with Miller index $[0]$.

4. Fix the remaining boundary maps $\partial_2^{[\mathbf{r}]}$. That is, for each $\mathbf{v}_i$ and $\forall \mathbf{r}$, calculate the dual boundary $\mathbf{c}_2 \equiv \sum_{\mathbf{p}} \bar{\partial}_2^{[\mathbf{p}]} \bar{\partial}_3^{[\mathbf{r}-\mathbf{p}]} \mathbf{v}_i$. By the zero map conditions (Eq. (E.2) in Sec. E.1 of App. E), the right-hand side should be zero. That is, we connect faces $\mathbf{f}_j \in \mathbf{c}_2$ to the newly created edge $\mathbf{e}_i$ with Miller index $\bar{\mathbf{r}}$.

8.2.3 Face-edge splitting

The cell-vertex splitting procedure described in Sec. 8.2.2 changes both the number of syndromes and the connectivity between syndromes in the syndrome graph. We can define an additional splitting operation on faces (dual edges) of such a complex, keeping the fault-tolerant properties of the corresponding quantum channel intact, but changing the underlying structure of the cluster state. We discuss this operation in this section. This operation splits the edges and faces of a lattice and allows one to distribute a fault-tolerant cluster state over multiple parties. Such a split essentially separates the information that is obtained by measuring a single qubit of the cluster state into a multi-qubit measurement

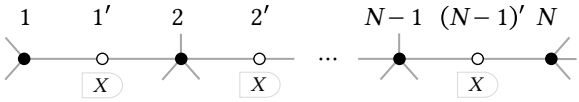


Figure 8.2: A subgraph of $2N-1$ qubits. Qubits marked primed indices are measured in the Pauli-X basis. Unmeasured qubits are connected arbitrarily to the rest of the graph. If one assumes that every measurement outcome $m=+1$, the unmeasured qubits are initialized in an N-partite Bell/GHZ state.

on two or more new qubits. The face-edge split decreases the individual connectivity of the qubits introduced by the split: they all have a lower number of connections than the original qubit.

Usually, a cluster state as described in Sec. 3.5 is constructed with the aid of CZ gates on qubits initialized in the $|+\rangle$ state. Alternatively, one may replace CZ gates with other entangling operations that lead to the same stabilizer states. Consider the subgraph of a graph state as in Fig. 8.2. Qubits at odd positions are marked with integers $i \in \{1,2,\ldots,N\}$ and their right neighbors at even positions with a primed index $i' \neq N'$. In this graph, odd qubits have an arbitrary number of neighbors, whereas even qubits only neighbor the two odd qubits on either side. After measuring the even qubits i' in the Pauli-X basis, the post-measurement $m_{i'}=+1$ graph state stabilizers are

$$\langle X_{1'}, X_{2'}, \ldots, X_{N-1'}, \prod_{i=1}^{N}\Big(X_i \prod_{j\in\mathbb{N}(i)} Z_j\Big), Z_1Z_2, Z_2Z_3, \ldots, Z_{N-1}Z_N\rangle. \tag{8.2}$$

Here, $\mathbb{N}(i)=\{j \mid (i,j)\in E\}$ are the qubits connected to the odd qubit i *outside* the subgraph depicted in Fig. 8.2. The disentangled measured qubits play no further role in the graph state stabilizers. In practice, one may replace these "virtual" qubits with measurement outcomes $m=+1$ and initialize the unmeasured qubits in the state stabilized by Eq. (8.2)—the resulting state is the same. In constructing graph states, CZ gates transform an X-type stabilizer $\prod_{i=1}^{N} X_i \mapsto \prod_{i=1}^{N}(X_i \prod_{j\in\mathbb{N}(i)} Z_j)$, whilst leaving the Z-type stabilizers untouched. This means that we can alternatively initialize the odd qubits i as an N-qubit $|\mathrm{GHZ}^{(N)}\rangle$ state stabilized by $\langle\prod_{i=1}^{N} X_i, Z_1Z_2, Z_2Z_3, \ldots, Z_{N-1}Z_N\rangle$, before applying the C$Z$ gates to qubits outside the subgraph. If $N=2$, one can initialize with the bipartite Bell pair state $\langle X_1X_2, Z_1Z_2\rangle$.

The above procedure shows how the subgraph in Fig. 8.2 need not be initialized through CZ gates, so long as there is a protocol that can create $|\mathrm{GHZ}^{(N)}\rangle$. Our new splitting procedure on faces produces subgraphs like Fig. 8.2 in a systematic way. Importantly, the closed-cell stabilizers of the cluster state stay intact on the split geometry.

Like a cell-vertex split, a face-edge split subdivides an existing face into two or more parts, adding new edges to separate the newly created faces. We give an example of this procedure on a square in Fig. 8.3. The newly created edges each support an additional qubit, always laying adjacent to two faces. Therefore, the subgraph supported by split edges and faces is a chain in the form of Fig. 8.2. We may replace the cluster state supported by the N connected faces with an N-partite GHZ state. Because the removed qubits correspond to "virtual" $m=+1$ measurement outcomes, their even parity plays no further role in the evaluation of error syndromes.

We can extend this procedure to both the primal and dual complex, producing GHZ states on both faces and edges. An example based on the cubic cluster state is given

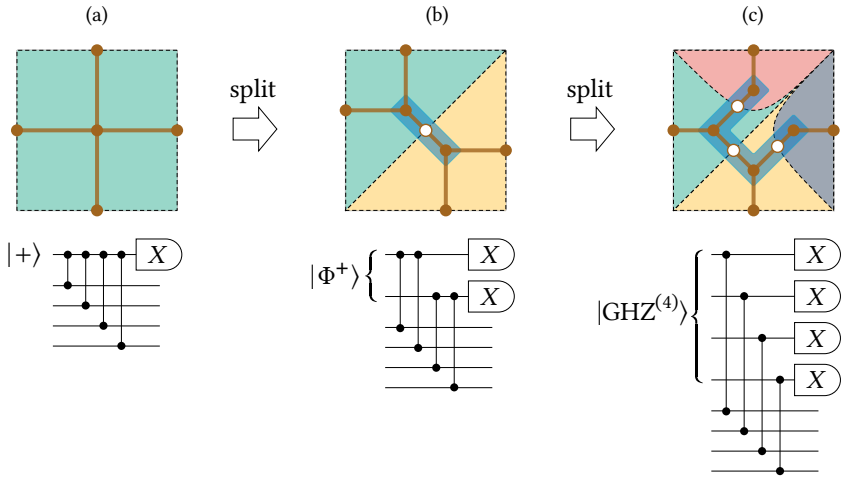


Figure 8.3: Face splitting on a square. (a) Without a split, the square represents a monolithic cluster state without distributed entanglement. The corresponding circuit of $|+\rangle$ state initialization, CZ gates, and Pauli-X basis measurement is also drawn. (b) A hybrid approach. The distributed face qubit is split over two parties represented by two faces, forming a bipartite GHZ or Bell state. The parity check is performed through two bilocal CZ gates of each face to its neighboring qubits and two local measurements. (c) A fully distributed approach. A face is split into four parties that share a GHZ state. Each party performs a single CZ gate to its neighbor and performs a measurement.

in Fig. 8.4. Starting from a monolithic architecture, a full 4-partite split of primal faces produces an architecture with nodes containing five qubits on a single edge and each of the adjacent split faces. Nodes are entangled with one another by GHZ states on every face. We may perform the same procedure for dual faces (primal edges), further reducing the number of qubits in each node to two.

We do not consider this in the rest of this chapter but note that, instead of initializing the qubits introduced by a face-edge split as an entangled state and measuring them individually, one can alternatively initialize these qubits regularly in $|+\rangle$ and measure them out with a joint *(Type-II) fusion measurement* [11–13]. This provides one with a method to transform (fault-tolerant) cluster states into so-called *fusion networks* that form the basis of fusion-based quantum computing [6].

8.2.4 Circuit-level and network noise

In this section, we describe the noise models used for monolithic and distributed threshold calculations. For the monolithic simulations, the entire cluster state is built from $|+\rangle$ state preparation, followed by CZ gates between every connected face-edge pair in the cluster state, concluded with a Pauli-X basis measurement of every qubit. In this model, we do not consider the effects of memory decoherence. For circuit-level noise, the following noise sources are included:

1. Noisy state preparation as a classical mixture $(1-p_\mathrm{p})|+\rangle\langle+| + p_\mathrm{p}|-\rangle\langle-|$. That is, applying a phase-flip error on $|+\rangle$ with probability p_p.

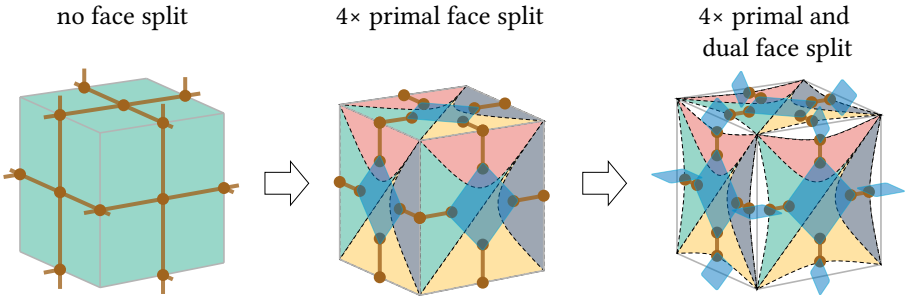


Figure 8.4: Face-edge splitting on a cubic cell. Starting from the left, a monolithic cluster state is transformed into a distributed cluster state by a four-partite split for each of its faces (see also Fig. 8.3). Performing the same procedure for dual faces (primal edges) produces a cluster state that is fully distributed, where each network node contains two cluster state qubits.

2. Two-qubit depolarizing noise following every CZ gate, as described by Eq. (2.10). We use p_g as the parameter describing two-qubit gate noise.

3. Classical bit-flips following every Pauli-X basis measurement: a measurement outcome m is flipped to $-m$ with probability p_m. The corresponding noisy measurement channel takes on the form $\widetilde{\mathcal{P}}^{\pm}(\rho) = (1-p_m)\Pi_{\pm}\rho\Pi_{\pm} + p_m\Pi_{\mp}\rho\Pi_{\mp}$, where $\Pi_{\pm}(\rho) \equiv |\pm\rangle\langle\pm|$ are the desired projectors onto the Pauli-X basis.

For the entangled states used in the distributed simulations, we assume that parties have shared access to Bell pairs in the isotropic form

$$\rho_{p_n} = (1-p_n)\,|\Phi^+\rangle\langle\Phi^+| + \frac{p_n}{3}|\Phi^-\rangle\langle\Phi^-| + \frac{p_n}{3}|\Psi^+\rangle\langle\Psi^+| + \frac{p_n}{3}|\Psi^-\rangle\langle\Psi^-|. \tag{8.3}$$

We refer to p_n as the parameter that describes "network noise". In this chapter, we do not take into account specific physical systems. We justify using isotropic states by noting that a depolarizing channel (*i.e.*, the most general noise channel) converts a perfect Bell pair state to an isotropic state. Additionally, isotropic states can be considered a general proxy for non-perfect Bell pairs, because every Bell pair state can be twirled into an isotropic state using local operations and classical communication.

To generate GHZ states from these Bell pairs, we use the straightforward method based on local parity measurements [14]. Fundamentally, a GHZ state can be created from Bell pairs by a local projective measurement of the ZZ parity between two halves of two pairs shared by multiple parties. The circuits to create a 3-partite GHZ from two Bell pairs and a 4-partite GHZ state from three Bell pairs are drawn schematically in Fig. 8.5.

We note that the circuits in Fig. 8.5 do not include distillation. Better quality GHZ states can typically be generated if the Bell pairs used to carry out the projective measurement are pre-distilled, or if GHZ states are distilled after creation. This, however, introduces a probabilistic factor to the GHZ creation protocols, which leads to higher numerical complexity in simulating the circuits.

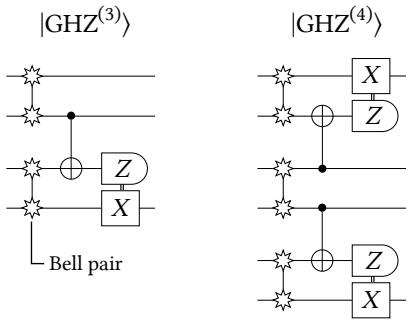


Figure 8.5: Fusion circuits for weight-3 and weight-4 GHZ states. In each of the circuits, Bell pairs are fused through the application of local CX gates and a subsequent Z basis measurement of the target bits. The conditional bit-flips ensure that GHZ states have the desired form $|0\ldots0\rangle + |1\ldots1\rangle$.

8.2.5 Numerical tools and considerations

To decode error syndrome graphs, we have implemented a version of the Union-Find decoder. This decoder is particularly attractive given its almost-linear time complexity and ease of implementation for both Pauli and erasure errors. Despite being a sub-optimal decoder, the phenomenological threshold for the cubic cluster state (2.6%) is close to that of the Minimum-Weight Perfect-Matching decoder (2.9%) [3], which is in turn not far from the optimal threshold (3.3%) [15]. Furthermore, since the Union-Find decoder is maximum-likelihood over the erasure channel (due to its built-in *peeling* decoder [16]), we expect this performance gap to shrink further over noisy channels that are a combination of Pauli and erasure errors. We have specifically implemented the "weighted-growth" version of the Union-Find decoder, as introduced in the original Union-Find manuscript [8]. Our implementation can be found in the repository of Ref. [17].

Error models per unit cell are constructed with a circuit simulator. Unit cells are defined according to the description in Sec. 8.2.1, together with the splitting methods of Secs. 8.2.2 and 8.2.3. The simulator is implemented as a classical efficient stabilizer simulator. Per operation, the simulator applies a full error channel as a series of Pauli operators, ending with a measurement in the N-qubit Pauli basis. The choice for Pauli noise is justified in these simulations because the input states are (convex combinations of) stabilizer states, the operation noise is typically Pauli noise, and measurement noise is described by classical bit-flips. In App. F, we describe the details of constructing an error channel. Pauli twirling a state at the end of a series of non-Clifford operations is equivalent to Pauli twirling the individual operations before applying them. This allows us to also use the simulator in situations where one is interested in the Pauli-twirled version of the full error channel—in that case, it suffices to twirl the individual channel components before applying them.

The calculated error models are used to sample errors in Monte Carlo style on qubits of the full crystalline cluster state. These cluster states are constructed from the associated unit cell complex according to the crystal embedding procedure described in Sec. E.2 of App. E. Per Monte Carlo sample, we decode the syndrome graph and assign a logical failure whenever there is a logical error for a single pair of logical X and Z operators of the channel—see Sec. 3.5.5 for more details. Error thresholds are determined with fits of

the logical error probabilities versus the lattice size (L) and a (set of) noise parameter(s)—in App. C we discuss this process in more detail.

8.3 Results

The results are organized into three parts, where the noise models become increasingly complex. A summary of the most important thresholds found can be found in Fig. 8.6. Secs. 8.3.1 and 8.3.2 provide thresholds for various geometries under a phenomenological noise model. In these sections, we reproduce earlier-known results and investigate the influence of a lattice boundary. For the phenomenological noise model, cluster states based on lattices that have a lower bond-degree b are more resilient against errors. However, these cluster states typically require more two-qubit gates to construct. This aspect is not regarded by the phenomenological noise model.

To investigate the trade-off between noise resilience and noise introduced by constructing the cluster state, we investigate scenarios with circuit-level and network noise in Secs. 8.3.3 and 8.3.4. Sec. 8.3.3 discusses numerical thresholds for monolithic (*i.e.*, non-distributed) architectures. These results are compared with the circuit-based error models for cluster states on a distributed network in Sec. 8.3.4. In the last part of this section, in Sec. 8.3.5, we investigate how the GHZ success probability of distributed networks can be traded off against a higher erasure probability to achieve fault tolerance against larger infidelity of the entangled states used.

8.3.1 Phenomenological thresholds

Before considering more realistic noise models, we first numerically evaluate erasure and phenomenological thresholds for several known lattices, which were constructed using cell-vertex splitting. These are the cubic ($b = 6, v = 4$), diamond ($b = 4, v = 6$), triamond ($b = 3, v = 10$), and double-edge cubic ($b = 3$ on average, $v = 8$) cluster states. We go beyond the results in Ref. [3] by determining *fault-tolerant regions* against erasure and phenomenological errors. The results are depicted in Fig. 8.7. We consider a phenomenological error model. It corresponds to perfect state preparation of the cluster state followed by measurements that fail to report (*i.e.*, erase) an outcome with probability p_e or flip the outcome with probability p_m, both of which are i.i.d. In Fig. 8.7, we estimate a fault-tolerant region for the erasure probability and bit-flip probability. Data points of the fault-tolerant regions are calculated by sweeping over both error probabilities while keeping their ratio fixed. We use a different constant of proportionality at each data point on the fault-tolerant boundary.

The isolated thresholds found for both types of noise are included in Fig. 8.6a. These threshold values are slightly higher than those reported in Ref. [3]. This can be attributed to a different implementation of the Union-Find decoder—see Sec. 8.2.5 for more details. Because phenomenological noise does not take into account higher error probabilities that arise with increasingly complex preparations of a cluster state, a fairer comparison is made by weighing each qubit error probability with the cluster state valency v, producing i.i.d. noise with probabilities vp_e and vp_g. This is called the *weighted phenomenological* error model in Ref. [3]. Because the cubic, diamond, double-edge cubic, and triamond lattices are regular with valency $v \in \{4, 6, 8, 10\}$ on all qubits respectively, weighted thresholds may be calculated from unweighted thresholds by simply dividing by v.

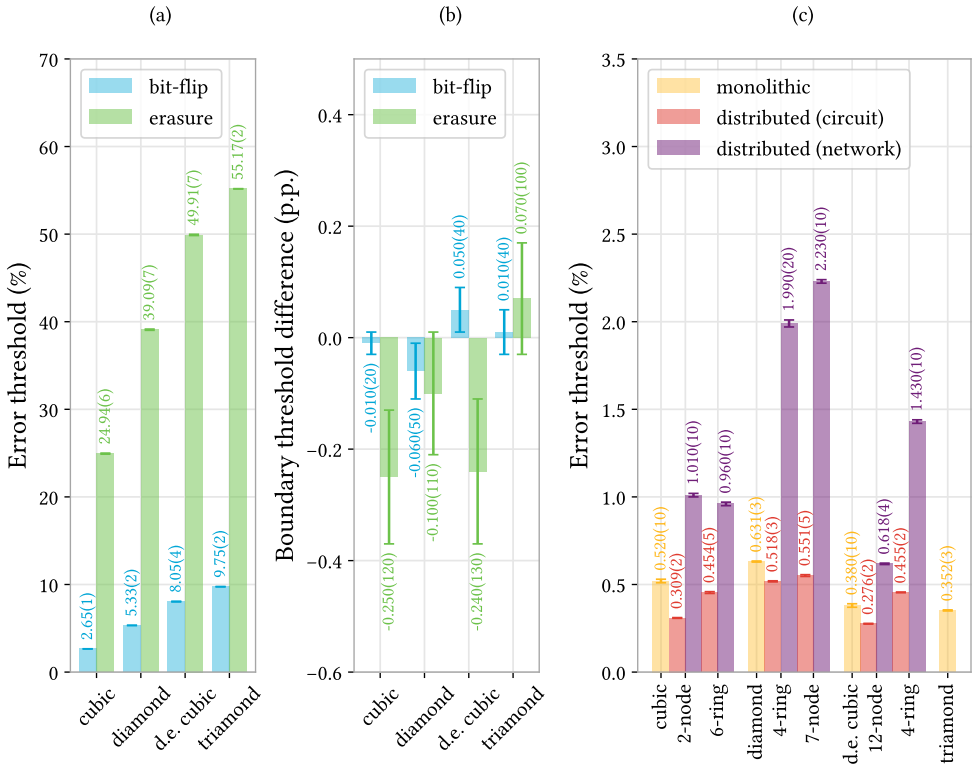


Figure 8.6: Overview of the threshold values found in our study, for different lattices and noise models considered. (a) Summary of numerical thresholds for bit-flip and erasure-type phenomenological noise models of the four lattices described in this work. (b) Differences of estimated threshold values for the same lattices and error models as in (a) with boundaries, expressed in percentage points (p.p.). (c) Monolithic thresholds are gauged against circuit-level errors with a common error probability as defined in Secs. 8.2.4 and 8.3.3, and similarly for distributed thresholds gauged under only circuit-level noise. Network thresholds are obtained from the model as defined in Sec. 8.2.4. For the monolithic thresholds, we show only the highest threshold value found with alternative gate orderings (see Sec. 8.3.3 in the main text for details). Numbers above the bars indicate the threshold, with 95% confidence intervals around the value indicated as error bars and as decimals in parentheses.

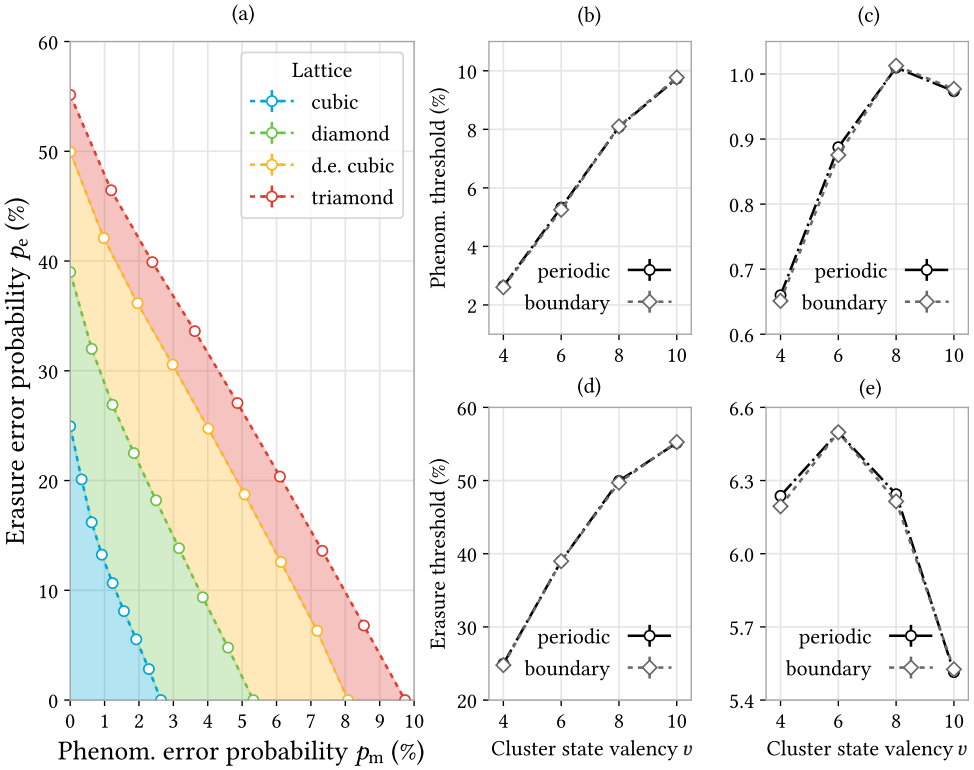


Figure 8.7: Fault-tolerant regions of the cubic, diamond, double-edge (d.e.) cubic, and triamond lattices under phenomenological bit-flip and erasure noise. (a) Regions are estimated based on individual threshold values. (b) Thresholds $p_{\mathrm{m,th}}$ under pure bit-flip noise as a function of cluster state valency. From left to right is the cubic, diamond, double-edge cubic, and triamond lattice. We provide thresholds with and without lattice boundaries—see Sec. 8.3.2 for more details. (c) Thresholds $p_{\mathrm{m,th}}/v$ under pure weighted bit-flip noise, where v is the cluster state valency. (d) Thresholds $p_{\mathrm{e,th}}$ under pure erasure noise. Results agree with known percolation thresholds for the given lattices. (e) Thresholds $p_{\mathrm{e,th}}/v$ under pure weighted erasure noise.

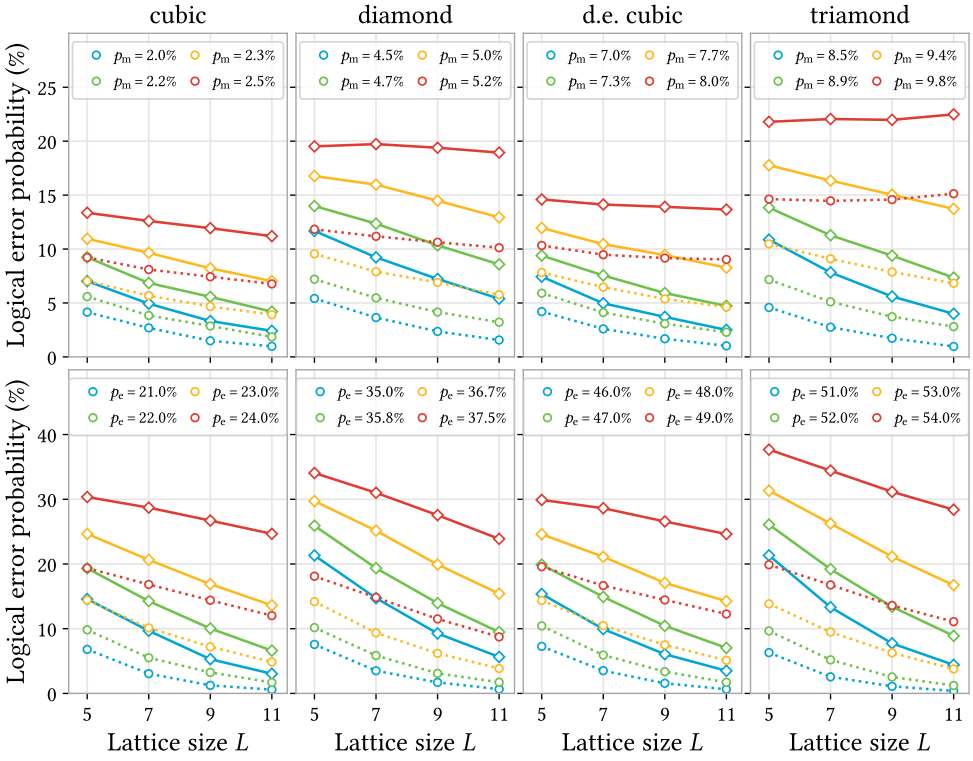


Figure 8.8: Sub-threshold logical error probability scaling of cubic, diamond, double-edge (d.e.) cubic, and triamond lattices with and without boundaries for phenomenological bit-flip (top row) and erasure (bottom row) noise models. Data for periodic conditions has round markers with a dashed line. Although the thresholds are minimally affected, sub-threshold error probabilities of any lattice with a rough and smooth boundary are about 1.5 to 2 times the probability for the same lattice under periodic boundary conditions.

8.3.2 Phenomenological thresholds with boundaries

The lattices considered above repeat periodically in all three spatial directions. It may be difficult to prepare such a cluster state if we take into account the connectivity of the qubits. We gauge how the performance of a cluster state is affected by the introduction of boundaries, under the same phenomenological noise model of both bit-flips with probability p_m and erasure errors with probability p_e. Each lattice is introduced to a smooth boundary along the $x = 0$ plane, and a rough boundary along the $y = 0$ plane. For the smooth boundary, we remove elements of the correlation surface defined by a *dual* logical membrane and its closure and introduce a boundary on the remaining dangling edges in the dual complex. The rough boundary is introduced in the same way, by swapping primal and dual notions.

In Figs. 8.6b and 8.7b-e, we show how the threshold values for the phenomenological and erasure noise models change with the introduction of the boundaries. The results show that differences in the phenomenological bit-flip threshold values are insignificant. For the erasure thresholds, only the cubic and double-edge cubic lattices are slightly affected. The results indicate that the introduction of boundaries affects the thresholds only minimally, at

least for the lattices and noise models considered here. However, we found that boundaries *do* impose significantly weaker *sub-threshold scaling*. The sub-threshold scaling is the rate at which the logical error probability is suppressed below the threshold. Fig. 8.8 shows that boundaryless architectures have a favorable error probability suppression that is about 50 to 100% as effective as the same architecture with boundaries. These results are consistent with recent work, where it was also shown that the introduction of boundaries leaves the threshold nearly invariant, but negatively impacts error rate scaling by roughly the same factor [18].

8.3.3 Monolithic thresholds

To compare distributed thresholds against monolithic implementations of the same geometry, we first benchmark monolithic architectures with the circuit-level noise models from Sec. 8.2.4. Below, we set all probabilities $p_{\mathrm{p}} = p_{\mathrm{g}} = p_{\mathrm{m}} \equiv p_{\mathrm{o}}$ equal (see Sec. 8.2.4 for the definition of the parameters), and sweep a threshold over the value of p_{o}. What remains is a specification of the ordering of CZ gates in the circuits, since pure CZ gates commute, whereas their noisy versions do not. Because no qubit can interact with two CZ gates simultaneously, a valid ordering may be extracted from an edge coloring of the corresponding ∂_2 boundary map of the cluster state. This graph is bipartite by definition. Therefore, the chromatic index (*i.e.*, the minimum number of colors needed for an edge-coloring) equals the maximum degree of any vertex in the graph—*i.e.*, the maximum valency v of the cluster state [19]. The cubic, diamond, double-edge cubic, and triamond lattices are all regular, and so their chromatic indices are 4, 6, 8, and 10, respectively.

For a chromatic index i and a given coloring, there are $i!$ different ways to order the edges and thus the CZ gates. Rotational and reflection symmetries of the cubic cluster state imply that there are only two colorings that correspond to a unique sequence of gates: a "(counter)clockwise" sequence going around a face, and a "zigzag" sequence jumping to opposite sides first. For diamond, double-edge-cubic and triamond lattices, the number of orderings quickly explodes as $6! = 720$, $8! \approx 4 \times 10^4$ and $10! \approx 3.6 \times 10^6$ (not taking into account symmetries). We have not included an exhaustive search of all orderings and their corresponding thresholds up to symmetries, but only investigate a subset of orderings.

Monolithic thresholds with the (counter)clockwise orderings are shown in Fig. 8.9. An overview of all monolithic thresholds is included in Fig. 8.6c. For the cubic lattice, a zigzag ordering of CZ gates slightly outperforms the (counter)clockwise ordering. The diamond lattice outperforms a cubic cluster state, whereas the double-edge cubic lattice drops in performance for the orderings considered. In all cases, (counter)clockwise orderings of the gates (at least in the primal complex) tend to produce lower thresholds than orderings that skip multiple edges at a time. We can intuitively understand this by considering that a single Pauli-X error on a face qubit spreads through all subsequent CZ gates as correlated Pauli-Z errors to neighboring edge qubits. In a (counter)clockwise ordering, the Pauli-Z errors produce a single strand that wraps around the face. In a zigzag orientation, Z error strings can form disconnected chains, such that a single Pauli-X error produces multiple syndrome pairs on different sides of the face. Initial numerical analysis shows that (counter)clockwise orderings indeed seem to result in, on average, longer error strings. The triamond lattice was gauged for a single ordering due to its complexity and performed poorly.

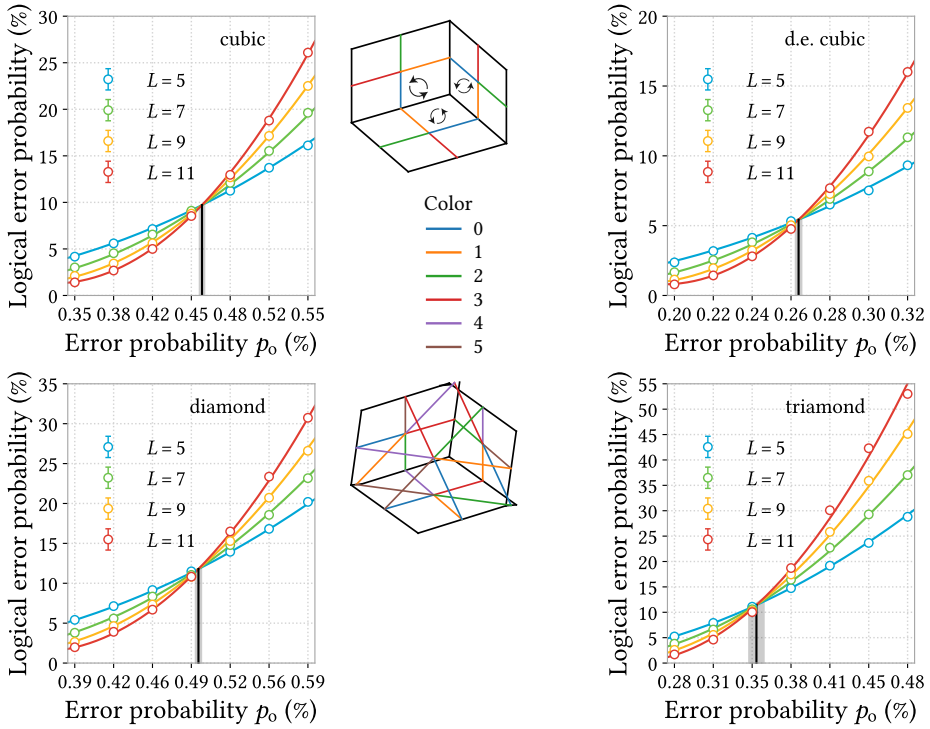


Figure 8.9: Monolithic thresholds $p_{o,th}$ for specific gate orderings. The thresholds are calculated with, for the cubic and diamond unit cell, a straightforward order of CZ gates according to a "(counter)clockwise" coloring of the diagram on the right (see arrows in the diagram for the cubic lattice). We omit details on (counter)clockwise orderings for the double-edge cubic and triamond lattice. Gates of a single color are performed simultaneously for all unit cells in the lattice. Each data point constitutes 50000 samples. Error bars of the logical error probability are given as 95% confidence intervals but are too small to be discernible in most cases. The threshold value is highlighted with a 95% confidence interval based on its least-squares estimate of a second-order polynomial of the logical error probability around the threshold value—see App. C for details.

In the limiting case that all gate errors are Pauli-Z errors, the relative impact of gate order disappears, and thresholds coincide with the weighted phenomenological thresholds considered above. In Ref. [10], Newman *et al.* consider an error model that, after each CZ gate, applies an X-type error on a (primal) face qubit with probability p_X and a Z-type error on a (primal) edge qubit with probability p_Z. Their results show that the best-performing lattice in terms of threshold moves from higher to lower valency as Pauli-X errors start to dominate. Our results show a similar tendency under depolarizing gate noise for the higher valent double-edge cubic lattice and triamond lattices when compared to the lower valent cubic and diamond lattices. We can summarize this finding by stating that as the cluster state valency increases, depolarizing noise incurs a larger cost on the threshold value. Under these noise models, there exists an optimal threshold resulting from a trade-off between the complexity of the geometry of the cluster state, and the structure of the noise created by the circuit. Noise bias is one example where this trade-off may be abused, by taking advantage of the architecture of the cluster state in either primal or dual lattice structures.

The monolithic cluster state thresholds do not compete with surface code monolithic thresholds, which are estimated at 0.90% and 0.95% under the same noise model [20]. It should be noted that the numbers for all non-cubic lattices provided here are likely sub-optimal, as we have only gauged the performance for a subset of CZ gate orderings. Nevertheless, estimates show that cluster states defined on the diamond lattice can outperform the cubic cluster state in the presence of circuit-level noise. For cluster states with even higher valencies, the cost of initialization negatively impacts the value of the threshold, consistent with the results under weighted phenomenological noise models. These results warrant further optimizations of the gate orderings and comparisons with other lattices.

8.3.4 Distributed thresholds

In this section, we investigate thresholds for distributed implementations of the cluster states. We use the face-edge splitting operation of Sec. 8.2.3 to split faces of the cubic, diamond, and double-edge cubic lattices. The cubic lattice is gauged for two different splits: one along the diagonals, producing network nodes with six cluster qubits in a ring and entangled through Bell pairs, and one on the entire face, producing 2-qubit nodes that are entangled through 4-partite GHZ states. Both these architectures also appear in the context of fusion-based quantum computation in Bartolucci *et al.* [6]. The structure of the diamond lattice is more intricate, and we produce two different architectures that contain only weight-3 GHZ states (the 4-ring) and a mixture of Bell pairs and weight-3 GHZ states. Architectures for the double-edge cubic lattice resemble the cubic lattice: the first contains only Bell pairs, whilst the other shares weight-4 GHZ states. All architectures are drawn schematically in Fig. 8.10.

In the distributed model, we set $p_{\mathrm{p}} = p_{\mathrm{g}} \equiv p_{\mathrm{o}}$. On top of that, we use entangled states to connect the cluster states that are separated by the splits. For these states, we assume that nodes can prepare isotropic states with fidelity $1 - p_{\mathrm{n}}$ and that GHZ states are generated with the protocols of Fig. 8.5. In the absence of network links, this circuit-level model reduces to the model discussed in Sec. 8.3.3. Using this physical model, we numerically establish thresholds for the distributed lattices introduced above. In the same way as

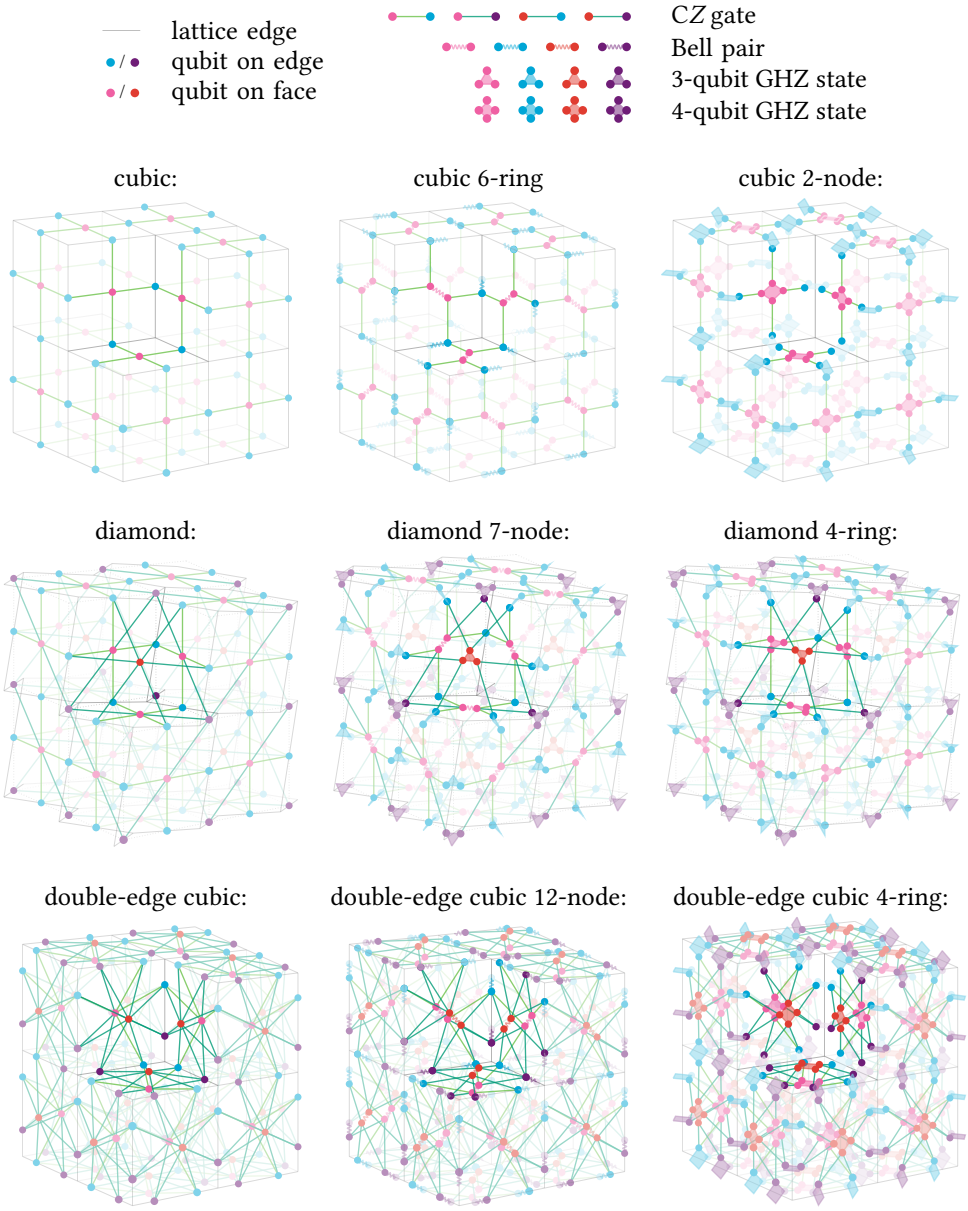


Figure 8.10: Distributed architectures for various lattices, obtained through splitting the faces of the monolithic cluster state. A single-face split leads to a Bell pair, whereas an n-split produces a GHZ state. They form connected components that are distinct nodes in a distributed network. We identify six different architectures: two each for the cubic, diamond, and double-edge cubic lattices. The left column shows purely monolithic cluster states. In the middle column, cluster states are initialized with Bell pairs on every face and edge, except for the diamond lattice that also contains 3-qubit GHZ states. Architectures in the right column are initialized with 4-qubit GHZ states for the cubic and double-edge cubic lattices, and 3-qubit states for the diamond lattice. Architectures in the middle column tend to have larger nodes than those in the right column.

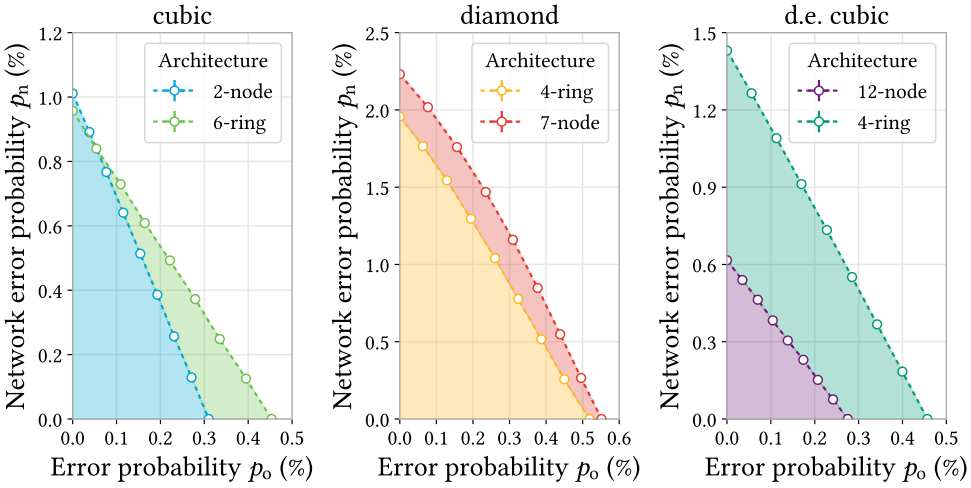


Figure 8.11: Fusion-based thresholds for the six distributed architectures of the cubic, diamond, and double-edge cubic lattices. The circuit-level noise parameter p_o and network error probability p_n are both swept to estimate a fault-tolerant region over thresholds of both parameters. Simulation details can be found in the main text.

phenomenological bit-flip and erasure thresholds, we estimate fault-tolerant regions for error probabilities p_o and p_n. The results are shown in Fig. 8.11 and in the overview of Fig. 8.6c.

These results show that network error probability thresholds for both the 2-node and 6-ring cubic designs are similar. Even though higher-valent GHZ states tend to produce lower-quality stabilizers, we suspect that the superior performance of the 2-node design against network errors p_n may be due to the size of its node. Because the 2-node architecture has only a single CZ gate per node, errors spread to only one adjacent qubit, as opposed to the 6-ring architecture that moves such an error to two adjacent qubits. Despite the more significant propagation of errors, the 6-ring architecture has a higher threshold against gate and measurement errors (the probability p_o) than the 2-node design. Most likely, this is because measurement errors have a higher influence in the 2-node architecture, which has more qubits. Both diamond architectures outperform the cubic lattice by a factor of roughly two in the network error probability threshold. This result is promising, especially considering that the depolarizing thresholds also outperform cubic architectures, as was already the case for the monolithic thresholds discussed above.

For the double-edge cubic architectures, network error probability thresholds drop again. Nevertheless, the 4-ring design with GHZ states outperforms the bigger 12-node design, which we may explain in the same way as in the cubic case: in the 4-ring lattice, errors propagate to fewer neighboring qubits compared to the 12-node architecture. An important difference with the cubic lattices is that the 4-ring design benefits enough from the lower error spreading to outperform the 12-node design under pure gate and measurement noise, despite the additional qubit measurements in the 4-ring implementation.

Thresholds for the circuit-level noise probability $p_{o,th}$ of these architectures are not optimal, due to the way that the ordering of CZ gates affects the threshold. We find that,

for all architectures considered, the values for $p_{\text{o,th}}$ without network noise are similar to the monolithic thresholds. Similar to the earlier analysis of distributed designs, comparing distributed to monolithic architectures involves trading off less error propagation in the distributed architectures versus fewer measurement errors in the monolithic architecture.

8.3.5 Trade-off GHZ success and erasure probability

The results in the previous sections indicate that fault tolerance can be achieved with network error probabilities below $\sim 2\%$—*i.e.*, with Bell pair fidelities above $\sim 98\%$. This condition is challenging from an experimental perspective. Fortunately, it is possible to boost the fidelity of entanglement before cluster state preparation through entanglement distillation. In this process, we model distillation failures as erasures on the corresponding qubit(s) and use a suitable decoder to deal with these qubit erasures. A suitable decoder is a decoder that can jointly correct errors and erasures—such as the Union-Find decoder. Phenomenological erasure thresholds can be roughly one order of magnitude higher than bit-flip error thresholds, so one may reasonably expect that trading in failed distillation attempts for higher-quality links is worth the cost. We make this argument more quantitative in Fig. 8.12, using the cubic 6-ring architecture as an example. In this architecture, every Bell pair is successfully heralded with probability $1-p_{\text{e}}$ and network error probability p_{n}, and discarded with probability p_{e}. The fault-tolerant region is simulated in the same way as before, this time sweeping over the network and erasure error probabilities, where circuit-level noise is not taken into account—*i.e.*, p_{o} is set to $p_{\text{o}} = 0$.

We can apply entanglement distillation to distill each Bell pair used in the cluster state by consuming multiple Bell pairs with initial fidelity $F_{\text{n}} = 1-p_{\text{n}}$. The distillation protocols we plot in Fig. 8.12 are (concatenated versions of) the DEJMPS protocol [21] (see Sec. 3.6.2) and the 5-to-1 bilocal Clifford distillation protocol of Sec. 3.6.4 based on the $[\![5,1,3]\!]$ error-correction code of Fig. 3.2. The output infidelities (on the p_{n} axis) and failure rates (on the p_{e} axis) of these distillation protocols are shown in the graph for two different values of the initial fidelity F_{n}. Here, the output infidelity is one minus the output fidelity and the failure rate is one minus the success probability of the distillation protocol. For the bilocal Clifford protocol, the (closed) data point on the bottom right of the two curves corresponds to a variant where, per Bell pair, only coinciding measurement outcomes are accepted. The data point on the top left of each curve corresponds to a variant where one accepts all measurement outcomes. Intermediate data points are protocols that accept a subset of non-coinciding measurement results. One can reach even more combinations of output infidelities and failure rates by using *interpolation* of two protocols [22]. This approach balances (or "mixes") the success probabilities and output fidelities of two protocols with a classical coin shared between the two parties. The coin outputs heads with probability r and tail with $1-r$, and the parties apply one of the two distillation protocols based on the outcome of the coin. For two protocols with success probabilities p_1 and p_2 and with output fidelities F_1 and F_2 this leads to a new protocol with success probability $p_{\text{succ}} = rp_1 + (1-r)p_2$ and output fidelity $(rp_1F_1 + (1-r)p_2F_2)/p_{\text{succ}}$.

It is clear that there is a trade-off for distillation between fidelity and success probability. Importantly, we can directly link this trade-off to the trade-off between erasure and network error probability—this makes it possible to move the state into the fault-tolerant region. For the distillation protocols considered, the first crossing with this region happens for an

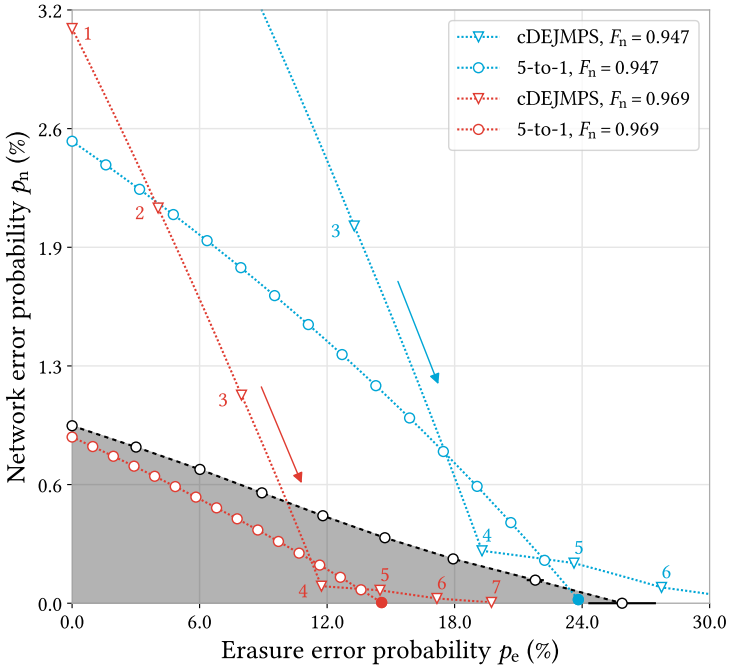


Figure 8.12: Trade-off between erasure and network error probability. The fault-tolerant region of the six-ring architecture is drawn (gray), where entangling links have a network error probability p_n and an independent probability p_e to fail. Gates and measurements are assumed to be noiseless. A failed link causes the edge to be erased—in the absence of network errors, we obtain the phenomenological erasure threshold of the cubic lattice. The colored lines reflect infidelities and failure rates of entanglement distillation protocols that use identical copies of an isotropic state with initial fidelity F_n to distill a single Bell pair with higher fidelity—*i.e.*, with lower network error probability. Distillation may bring a non-fault-tolerant architecture into the fault-tolerant regime. Data marked with triangles correspond to concatenated DEJMPS ("cDEJMPS") distillation protocols [21], where the numbers indicate how many Bell pairs are used to distill the final state. Data marked with colored circles is based on a bipartite Clifford distillation protocol that consumes five Bell pairs. This protocol is printed in Fig. 3.18, where, in this case, we use the encoding circuit of the error-correction code of Fig. 3.2 as the transformation U. Line segments between data points of the distillation protocols correspond to the interpolation of protocols on both sides of these segments—see the main text for more details.

initial network error probability of $p_n \approx 5\%$, which is a five-fold increase of the approximate 6-ring threshold value of 1% without distillation. These results are particularly promising for the diamond architecture, where both the network error threshold of roughly 2% and the erasure threshold of 40% are higher than the cubic lattice.

8.4 Conclusion

In this chapter, we provide several tools and numerical analyses to explore fault-tolerant measurement-based quantum computing architectures built from smaller units, in the context of modular, distributed, or networked computing. This is achieved with a method that allows us to distribute a fault-tolerant cluster state over multiple parties. This method augments the splitting procedure of previous work by Nickerson and Bombín [3]. In the distributed context, the resulting states lead to entanglement in the form of a Bell or GHZ state, such that existing methods for state generation and distillation can be used to design a fault-tolerant architecture.

The performance of various three-dimensional cluster state architectures is studied through numerical evaluation of their fault-tolerant thresholds, which quantify the probabilities of specific sources of error below which fault-tolerant computation is possible. We find that the diamond lattice outperforms the traditional cubic cluster state for monolithic architectures suffering standard noise models—*i.e.*, the noise models described in Sec. 8.2.4. It should be mentioned that even better results may be achieved with different permutations of the entangling CZ gates during cluster state preparation. In the cases we have considered, (counter)clockwise orderings of CZ gates tend to produce lower thresholds.

Furthermore, we gauge the performance of the same lattices in a distributed setting. For designs based on the cubic lattice, error thresholds of the network noise are around 1%. We consider two different designs of distributed cluster states defined on top of a diamond lattice, which outperform cubic thresholds roughly by a factor of two. Using a cubic architecture, we show how entanglement distillation may bring a non-fault-tolerant design into the fault-tolerant regime by trading in network noise for erasure errors in the cluster state. Combined with favorable erasure thresholds of the diamond lattice, these results indicate that distributed fault-tolerant cluster states may outperform topological error-correction codes, and warrant additional numerical simulations of these distributed networks in the presence of entanglement distillation.

There are several potential avenues for further investigation. On the one hand, our circuit-based qubit error models are limited to depolarizing and erasure-type noise, and it is interesting to estimate fault-tolerant thresholds for models that more accurately represent errors in state-of-the-art quantum hardware. Furthermore, it would be interesting to consider protocols that cannot be described as a simple sequence of instructions, but contain dependencies and branches that split based on intermediate decisions—as, *e.g.*, protocols that contain entanglement distillation. Previous results in the field apply pre-calculated error models (*i.e.*, quantum channels) on a unit cell level, and randomly sample from the error model during Monte Carlo threshold calculations [20, 23]. This is possible because the fault-tolerant protocol consists of identical rounds that are applied over time, where each round is split up and grouped into multiple sub-rounds that act on disjoint subsets of qubits, so that the entire protocol may be simulated as distinct sections that are separated in both space and time. This is not necessarily the case for the three-dimensional cluster

states that we consider here, in which case the entire cluster state should be simulated at a global level, *i.e.*, without pre-calculating error models of individual sections [24].

As an alternative to measurement-based quantum computation, it is possible to consider fusion-based quantum computation [6]. This paradigm combines a low circuit depth with the topological features of cluster states. The fundamental operations in fusion-based quantum computation are resource-state generation and fusion measurements. Fault-tolerant fusion-based architectures rely heavily on resilience against erasure errors, and the structures considered in this work are a natural candidate to consider in this alternative framework of computation.

Acknowledgments

The authors would like to thank Michael Newman, Naomi Nickerson, Tim Taminiau, and Barbara Terhal for helpful feedback and/or discussions. We gratefully acknowledge support from the joint research program "Modular quantum computers" by Fujitsu Limited and Delft University of Technology, co-funded by the Netherlands Enterprise Agency under project number PPS2007. This work was supported by the Netherlands Organization for Scientific Research (NWO/OCW), as part of the Quantum Software Consortium Program under Project 024.003.037/3368. We thank SURF (www.surf.nl) for the support in using the National Supercomputer Snellius.

References

[1] R. Raussendorf, J. Harrington, and K. Goyal, "A fault-tolerant one-way quantum computer," *Annals of Physics*, vol. 321, pp. 2242–2270, Sept. 2006.

[2] S. D. Barrett and T. M. Stace, "Fault tolerant quantum computation with very high threshold for loss errors," *Physical Review Letters*, vol. 105, p. 200502, Nov. 2010.

[3] N. Nickerson and H. Bombín, *Measurement Based Fault Tolerance beyond Foliation*. arXiv: 1810.09621 [quant-ph], Oct. 2018.

[4] J. M. Auger, H. Anwar, M. Gimeno-Segovia, T. M. Stace, and D. E. Browne, "Fault-tolerant quantum computation with nondeterministic entangling gates," *Physical Review A*, vol. 97, p. 030301, Mar. 2018.

[5] D. Herr, A. Paler, S. J. Devitt, and F. Nori, "A local and scalable lattice renormalization method for ballistic quantum computation," *npj Quantum Information*, vol. 4, pp. 1–8, June 2018.

[6] S. Bartolucci, P. Birchall, H. Bombín, H. Cable, C. Dawson, M. Gimeno-Segovia, E. Johnston, K. Kieling, N. Nickerson, M. Pant, F. Pastawski, T. Rudolph, and C. Sparrow, "Fusion-based quantum computation," *Nature Communications*, vol. 14, p. 912, Feb. 2023.

[7] D. D. Awschalom, R. Hanson, J. Wrachtrup, and B. B. Zhou, "Quantum technologies with optically interfaced solid-state spins," *Nature Photonics*, vol. 12, pp. 516–527, Sept. 2018.

[8] N. Delfosse and N. H. Nickerson, "Almost-linear time decoding algorithm for topological codes," *Quantum*, vol. 5, p. 595, Dec. 2021.

[9] K. Fujii, *Quantum Computation with Topological Codes: From Qubit to Topological Fault-Tolerance*, vol. 8 of *SpringerBriefs in Mathematical Physics*. Singapore: Springer, 2015.

[10] M. Newman, L. A. de Castro, and K. R. Brown, "Generating fault-tolerant cluster states from crystal structures," *Quantum*, vol. 4, p. 295, July 2020.

[11] D. E. Browne and T. Rudolph, "Resource-efficient linear optical quantum computation," *Physical Review Letters*, vol. 95, p. 010501, June 2005.

[12] W. P. Grice, "Arbitrarily complete Bell-state measurement using only linear optical elements," *Physical Review A*, vol. 84, p. 042331, Oct. 2011.

[13] F. Ewert and P. van Loock, "3/4-efficient Bell measurement with passive linear optics and unentangled ancillae," *Physical Review Letters*, vol. 113, p. 140403, Sept. 2014.

[14] N. Nickerson, *Practical Fault-Tolerant Quantum Computing*. Doctoral thesis, Imperial College London, London, United Kingdom, 2015.

[15] T. Ohno, G. Arakawa, I. Ichinose, and T. Matsui, "Phase structure of the random-plaquette Z2 gauge model: Accuracy threshold for a toric quantum memory," *Nuclear Physics B*, vol. 697, pp. 462–480, Oct. 2004.

[16] N. Delfosse and G. Zémor, "Linear-time maximum likelihood decoding of surface codes over the quantum erasure channel," *Physical Review Research*, vol. 2, p. 033042, July 2020.

[17] Y. van Montfort, S. de Bone, and D. Elkouss, "Data/software underlying the publication: Fault-tolerant structures for measurement-based quantum computation on a network." 4TU.ResearchData, https://doi.org/10.4121/929e24f9-31fa-4816-99fa-3356e272df43, Jan. 2024.

[18] H. Bombín, C. Dawson, R. V. Mishmash, N. Nickerson, F. Pastawski, and S. Roberts, "Logical blocks for fault-tolerant topological quantum computation," *PRX Quantum*, vol. 4, p. 020303, Apr. 2023.

[19] D. König, "Graphok es matrixok," *Matematikai és Fizikai Lapok*, vol. 38, pp. 116–119, 1931.

[20] N. H. Nickerson, Y. Li, and S. C. Benjamin, "Topological quantum computing with a very noisy network and local error rates approaching one percent," *Nature Communications*, vol. 4, p. 1756, Dec. 2013.

[21] D. Deutsch, A. Ekert, R. Jozsa, C. Macchiavello, S. Popescu, and A. Sanpera, "Quantum privacy amplification and the security of quantum cryptography over noisy channels," *Physical Review Letters*, vol. 77, no. 13, pp. 2818–2821, 1996.

[22] F. Rozpędek, T. Schiet, L. P. Thinh, D. Elkouss, A. C. Doherty, and S. Wehner, "Optimizing practical entanglement distillation," *Physical Review A*, vol. 97, June 2018.

[23] N. H. Nickerson, J. F. Fitzsimons, and S. C. Benjamin, "Freely scalable quantum technologies using cells of 5-to-50 qubits with very lossy and noisy photonic links," *Physical Review X*, vol. 4, p. 041041, Dec. 2014.

[24] C. Gidney, M. Newman, and M. McEwen, "Benchmarking the planar honeycomb code," *Quantum*, vol. 6, p. 813, Sept. 2022.

9

Conclusions and outlook

In this chapter, we summarize the main results of the research presented in this thesis. On top of that, we list improvements and additions to this work. Lastly, we provide an overview of interesting future research directions.

9.1 Summary of results

Creation and distillation of GHZ states. The first part of the contributions relates to the creation and distillation of GHZ states. We use a heuristic dynamic programming algorithm to identify protocols that create GHZ states out of non-perfect Bell pairs. In each stage of the algorithm, a new protocol is created by combining two well-performing protocols identified at earlier stages. The GHZ generation protocols produced by this algorithm can therefore be represented as directed binary trees. The elementary Bell pairs used in the protocol are always located on the leaves of the tree. The intermediate nodes of the tree contain either entanglement fusion or entanglement distillation operations. The fusion operation creates an entangled state of higher weight from two smaller-weight states that overlap in a network node. The distillation operations consume an entangled state to non-locally measure a stabilizer operator of the target state.

The dynamic program can be used to create GHZ states of arbitrary weight. However, we are specifically interested in weight-4 GHZ states because of their use in the distributed toric surface code. At this stage, our goal is to minimize the minimum number of Bell pairs needed to create a GHZ state of a specific fidelity. Compared to previously known GHZ generation protocols, the new weight-4 protocols produce GHZ states with higher output fidelities and require roughly half the number of Bell pairs to achieve a similar fidelity in a scenario without memory decoherence and operation noise. We consider two variants of the dynamic program: a standard variant and a randomized variant. In most regimes, the randomized version finds the best GHZ protocols.

Evaluating GHZ generation protocols in the presence of memory decoherence. We use another heuristic approach to convert binary tree protocols generated by the dynamic program into a protocol recipe—*i.e.*, a practical set of ordered instructions. We develop an open-source circuit simulator to evaluate GHZ protocols and protocol recipes in the presence of memory decoherence, gate noise, and measurement errors. The simulator can be utilized with general hardware models. We use this simulator to evaluate protocol recipes for the use-case of realistic nitrogen-vacancy center hardware that suffers memory decoherence. We show that, under these conditions, protocol recipes generated by the dynamic programming algorithm use lower completion times to produce GHZ states with higher fidelities compared to previously known protocols. One of the best-performing protocols is a protocol that uses a minimum of four Bell pairs to generate a weight-4 GHZ state. We refer to this protocol as the Modicum protocol.

Calculating error thresholds for the distributed surface code in the presence of memory decoherence. We adapt an existing surface code simulator to perform stabilizer measurements with more general superoperator channels. This makes it possible to simulate the non-local measurement of the code's stabilizers with the aid of weight-4 GHZ states. Measuring the code's stabilizer in such a way represents a scenario where each data qubit of the surface code is part of a separate network node. We combine this simulator with the standard variant of the dynamic program and the circuit simulator. This enables us to optimize over GHZ generation protocols used to non-locally measure the stabilizers of the distributed surface code for a hardware model of choice. This, in turn, makes it possible to calculate error probability thresholds for the distributed surface code for a specific combination of a hardware model and a GHZ protocol. We limit ourselves to calculating thresholds for a logical memory channel over L time layers with the Union-Find

error syndrome decoder, where $L \times L$ is the size of the surface code lattice.

We use these methods with hardware models based on realistic nitrogen-vacancy centers, but find that it is not possible to operate state-of-the-art nitrogen-vacancy center devices below their error thresholds—*i.e.*, at the moment, these nitrogen-vacancy centers introduce more errors than the code can correct for. We calculate toric surface code thresholds with models that represent more optimistic diamond color center hardware and find a threshold of $4 \cdot 10^2$ in the ratio between entanglement generation and memory decoherence rates. This is more than one order of magnitude above state-of-the-art. We identify that improving the entanglement generation rate with better photon detection probabilities seems to be the main challenge in obtaining hardware that can be operated fault-tolerantly. Our analysis shows that we approach fault tolerance in state-of-the-art hardware for photon detection probabilities 50 times higher than state-of-the-art and up to one order of magnitude improvement in coherence times during entanglement generation. A GHZ generation protocol consuming a minimum of seven Bell pairs is the best-performing protocol in a large number of regimes considered. We refer to this protocol as the Septimum protocol.

Fault-tolerant cluster states for distributed quantum computing. The surface code memory channel, with time as the third dimension, can be converted to a fault-tolerant cubic cluster state. This opens the door for considering fault-tolerant cluster state channels based on other three-dimensional lattices. By splitting the edges and faces of the lattice on which the cluster state is defined, we introduce a method to distribute the qubits of a large cluster state over multiple network parties connected by entanglement. This method extends on an approach by Nickerson and Bombín for constructing new three-dimensional lattices by splitting edges and cells of the cubic lattice. We present a stabilizer simulator for evaluating logical error rates for (distributed) cluster states defined on a general lattice.

With this simulator, we show that for cluster states based on four different three-dimensional lattices, the diamond lattice has the highest thresholds under a standard circuit-level noise model. This model applies depolarizing noise for every CZ gate used to construct the cluster state and incorporates bit-flip errors while measuring out the cluster state qubits. Additionally, we find that distributed versions of the diamond lattice have the highest thresholds against both standard circuit-level and network noise, where we also consider noise on Bell and GHZ states. These calculations do not include memory decoherence. For a distributed variant of the cubic cluster state, we analyze how a lower GHZ success probability can be exchanged for a higher erasure probability. By exploiting this trade-off, we can achieve fault tolerance with lower elementary entanglement fidelities.

9.2 Future work

Generating and evaluating GHZ generation protocols. The circuit simulator we use to evaluate GHZ generation protocols in the presence of operation noise and memory decoherence makes use of density matrix calculations. This means that, typically, using this simulator to calculate the average GHZ state is a time-consuming affair. A concrete proposal to make these calculations faster is the use of the stabilizer formalism to track states during protocol simulation—even more concretely, this can be achieved by combining this circuit simulator with our cluster state simulator, in which the stabilizer formalism is implemented. Faster evaluation of GHZ protocols can be beneficial since it means the

dynamic program can consider more protocols in the same amount of time. A calculation speed-up also makes using the randomized dynamic program more feasible in situations with operation noise and memory decoherence. The randomized variant gives better results in most of the scenarios without operation noise. However, because the standard dynamic program is faster, we currently restrict to the standard variant in all scenarios that include operation noise and memory decoherence.

An alternative approach for generating well-performing GHZ protocols could be found by employing reinforcement-learning techniques to identify protocols or optimize over the space of GHZ protocols [1–3]. Less drastically, alternative GHZ protocols can be generated with the existing dynamic programming algorithm if one also includes distillation operations that can not be represented as non-local stabilizer measurements. We include possible extensions with this type of distillation in Sec. 3.6 of this thesis—specifically, they are introduced Secs. 3.6.3 and 3.6.4 and Refs. [4–6] for the bipartite case and in Sec. 3.6.8 and Ref. [4] for the multipartite case.

Further possibilities for improvement can be found in the scheduling of GHZ creation protocols. For example, the starting time of operations can be optimized to increase the probability of parallel branches finishing at the same time. On top of that, during the execution of a protocol, it might be beneficial to implement a dynamic strategy that re-evaluates the best execution plan after each failed distillation attempt. These optimizations can possibly be achieved with artificial intelligence.

Interpret failed distillation attempts as erasure errors. Both for the distributed surface code and for fault-tolerant cluster states, it is possible to interpret failed distillation attempts as lost qubits—*i.e.*, as qubit erasures. This requires using an error syndrome decoder able to decode erased qubits, like the Union-Find decoder. This approach weakens the restrictions for the minimum fidelity of entangled states, as resilience against erasure allows us to probabilistically increase entanglement fidelities by incorporating more steps of entanglement distillation. In this thesis, we investigate a straightforward example of the distributed cubic cluster state, but it would be interesting to also incorporate this trade-off in other distributed cluster state architectures and the distributed surface code.

Distributed surface code implementations. There are alternative implementations possible for the distributed surface code. For example, one could think of variants that use more than one data qubit per node—*e.g.*, more than one data qubit per diamond color center [7]. Another variant worth investigating is a version where nodes measure stabilizer operators through ancillary nodes that do not hold data qubits themselves. In this variant, CZ and CX gates are carried out non-locally—*i.e.*, with the aid of Bell pairs—to measure the code's stabilizer—similarly to how, in a monolithic architecture, stabilizers can be measured with ancillary qubits in $|+\rangle$. Next to that, promising results have been obtained with the so-called $XZZX$ variant of the surface code [8] and it would be interesting to investigate this code in a distributed context. Given its high resilience against dephasing noise, which is typically the most prominent noise source in diamond color centers, this code is particularly interesting for implementations with diamond color center hardware. Lastly, it is important to study the performance of time-efficient error syndrome decoders for distributed error-correction codes. This is because, in practical realizations with large lattice sizes, error syndromes must be decoded within the cycle time of one round of error-detection measurements to prevent additional errors caused by memory decoherence.

Distributed error-correction codes with diamond color centers. Our tools can be readily employed in future scenarios with further experimental advancements in diamond color centers reaching higher photon detection probabilities. This possibly includes combining nitrogen-vacancy centers with microcavities, or further characterization of tin-vacancy and silicon-vacancy nanophotonic devices. Additionally, the use of tin-vacancy centers or silicon-vacancy centers unlocks the possibility of generating GHZ states directly with photon scattering or transmission, without the need to fuse Bell pairs [9]. Initial numerical investigations show this approach is likely to boost surface code thresholds, as it seems to reduce the influence of noisy and time-consuming SWAP gates [7].

Fault-tolerant cluster states for distributed quantum computing. For the research on fault-tolerant cluster states, one remaining open question considers the optimal order of noisy CZ gates applied during cluster state construction. This question is relevant for the construction of general graph and cluster states.

Next to that, follow-up research could focus on expanding the functionalities of the simulator, by, *e.g.*, implementing non-deterministic logic during Monte Carlo simulations. This makes it possible to explicitly include entanglement distillation in these simulations—both for distilling the Bell pairs and GHZ states that connect the cluster states as for distilling the cluster states themselves with the methods discussed in Sec. 3.6.7. Additionally, the implementation of more detailed noise models, such as, *e.g.*, memory decoherence, could further catalyze interesting research. That is because this makes it possible to better investigate scenarios where the (distributed) cluster state is split into different time-separated slabs or blocks, and where one relies on keeping qubits "alive" between consecutive time steps—similarly to how data qubits of the surface code are kept alive over different time layers. This applies to qubits that sit on the boundary of time layers as well as qubits that are teleported to the next layer in the original cluster state. For example, this idea could be used to "exfoliate" the (distributed) diamond cluster state into (non-identical) two-dimensional time slabs. Partitions that contain even smaller cluster state blocks are also imaginable, resulting in situations in which space and time are no longer associated with exclusive dimensions of the full cluster state. This line of research is particularly interesting since fault-tolerant channels that use fewer qubits seem more realistic in the near future.

The implementation of more realistic noise models naturally facilitates comparisons between different hardware suitable for distributed cluster state implementation. Next to diamond color centers, this includes combinations of matter-based qubits and photons in the context of fusion-based quantum computation [10]. In fusion-based quantum computing, the entangled states of distributed cluster states can be directly replaced by (type-II) fusion measurements.

9.3 Outlook

Looking further into the future, from the viewpoint of theoretical research, it could be fruitful to recast cluster state channels as so-called *subsystem codes* [11, 12]. In such a code, some of the operators that would otherwise be used as stabilizer operators or logical operators stay unused. These operators are then referred to as *gauge (check) operators*, and we say that they describe *gauge qubits*. This concept can be used to design a code in which the stabilizer measurements can be constructed by classically combining lower-

weight parity measurements—similar to how the error syndrome is constructed for cluster state channels. Among other proposals, this idea has been used to transform surface codes into *subsystem surface codes* with weight-three parity measurements [13]. Related to this idea, a promising implementation with weight-two parity measurements is the *Floquet code* [14–16]. This code is similar to a dynamically changing version of the two-dimensional hexagonal toric code. A disadvantage of this approach is the increase in the number of measurement operations, which introduces more errors. Another disadvantage of introducing gauge qubits is that, typically, this leads to an increase in the qubit *overhead*—*i.e.*, the number of physical qubits required to encode logical information with the code.

Both the concepts of subsystem codes [17] and the earlier-mentioned $XZZX$ surface code [8] can be used to tailor an error-correction code based on noise characteristics—*i.e.*, by optimizing error correction based on existing noise biases in the hardware [18, 19]. Generalizing non-foliated cluster state channels under biased noise is a natural next step in this line of research [20]. In general, fault-tolerant cluster states could play an important role in designing tailor-made error-correction channels, as "time exfoliation/partitioning" could help in making these channels more practical.

Further generalization of three-dimensional cluster state channels in the direction of so-called *low-density parity-check* (LDPC) codes [21, 22] may lead to even more fruitful results. The error-correction codes considered in this thesis show a vanishing *decoding rate* K/N as $N \to \infty$. However, recent years saw the development of similar codes with constant decoding rates. A code with a constant decoding rate has the property that expanding the size of the code (and thus increasing its number of physical qubits N) leads to a proportional increase in the number of encoded logical qubits K—*i.e.*, it has a lower qubit overhead. Some of the constant-rate LDPC codes discovered also have favorable scaling with respect to the distance of the code—*e.g.*, their distance scales as a function of $\sqrt{N}$ [23, 24] or as a function of N [25]. Unfortunately, it turns out that these codes lose their two-dimensional [26] and nearest-neighbor characteristics: typically, the better the properties of these codes, the more long-range the interactions between the data qubits become, if one were to place them on a two-dimensional lattice [27]. Luckily, from the perspective of diamond color centers, more complex connectivities are not problematic, since entanglement between remote centers can—in principle—be realized with photonic links. With ion trap modules, long-range photonic connections are also possible [28]. Just as for the surface code and cluster state channels, the existence of efficient error syndrome decoders remains an open question for constant-rate LDPC codes.

On top of that, further theoretical research is necessary on realizing logical operators and universal quantum computing with fault-tolerant distributed channels [29, 30]. In an ideal scenario, to prevent local errors from spreading out, we would perform all logical operations transversally. This means we do not have to resort to methods such as magic state distillation. However, according to the Eastin-Knill theorem, codes that can do all gates transversally do not exist. This is another area where the use of the subsystem formalism is beneficial since subsystem codes can be used to perform *gauge fixing* [31, 32]. The idea of gauge fixing is that, during operation, one moves encoded information between different quantum error-correction codes that together hold a universal set of transversal logical gates. Switching between different codes can be achieved by (temporarily) adding or removing gauge operators to the set of stabilizer operators of the code. An example

of a topological code that can perform universal logical gates with gauge fixing is the three-dimensional *gauge color code* [33, 34].

Possibly in combination with these advancements in error-correction code design, it would be interesting to bolster numerical simulations with noise models that (at least schematically) resemble more precise hardware implementations—*i.e.*, noise models based on detailed engineering efforts of distributed quantum computers. This could, *e.g.*, be based on a photonic chip with integrated diamond color centers connected with waveguides. Future noise models should feature the characteristics of such an architecture and the imperfections and heterogeneities therein. This direction brings along challenges, as in our current two-level numerical implementation it is, *e.g.*, not straightforward to implement cross-talk between different nodes. However, perhaps it is not strictly necessary to simulate these architectures with the same level of detail as we did in Ch. 7. Eventually, one could argue that it is not *that* important what the exact value of the threshold is, as long as we have a rough idea of its order of magnitude and the logical error rates that can approximately be reached at certain error probabilities and lattice sizes. That is because, in practice, we want to run these codes significantly below the threshold in order to reach low logical error rates without having to resort to enormous code sizes. As such, only the most prominent noise sources should arguably be included in numerical investigations of this kind.

From a pure experimental perspective, it would be exciting to perform demonstrations of either distributed error-correction codes or (one of) their ingredients. For example, an experimental demonstration of the Modicum or Septimum GHZ generation protocol would constitute an important milestone in realizing practical fault-tolerant distributed quantum computing.

References

[1] S. Krastanov, V. V. Albert, and L. Jiang, "Optimized entanglement purification," *Quantum*, vol. 3, p. 123, Feb. 2019.

[2] F. Ferreira da Silva, A. Torres-Knoop, T. Coopmans, D. Maier, and S. Wehner, "Optimizing entanglement generation and distribution using genetic algorithms," *Quantum Science and Technology*, vol. 6, p. 035007, June 2021.

[3] X. Zhao, B. Zhao, Z. Wang, Z. Song, and X. Wang, "Practical distributed quantum information processing with LOCCNet," *npj Quantum Information*, vol. 7, pp. 1–7, Nov. 2021.

[4] S. Glancy, E. Knill, and H. M. Vasconcelos, "Entanglement purification of any stabilizer state," *Physical Review A*, vol. 74, p. 032319, Sept. 2006.

[5] S. Jansen, K. Goodenough, S. de Bone, D. Gijswijt, and D. Elkouss, "Enumerating all bilocal Clifford distillation protocols through symmetry reduction," *Quantum*, vol. 6, p. 715, May 2022.

[6] K. Goodenough, S. de Bone, V. Addala, S. Krastanov, S. Jansen, D. Gijswijt, and D. Elkouss, "Near-term n to k distillation protocols using graph codes," *IEEE Journal on Selected Areas in Communications*, vol. 42, pp. 1830–1849, July 2024.

[7] S. Singh, F. Gu, S. de Bone, E. Villaseñor, D. Elkouss, and J. Borregaard, *Modular Architectures and Entanglement Schemes for Error-Corrected Distributed Quantum Computation.* arXiv: 2408.02837 [quant-ph], Aug. 2024.

[8] J. P. Bonilla Ataides, D. K. Tuckett, S. D. Bartlett, S. T. Flammia, and B. J. Brown, "The XZZX surface code," *Nature Communications*, vol. 12, p. 2172, Apr. 2021.

[9] H. K. Beukers, M. Pasini, H. Choi, D. Englund, R. Hanson, and J. Borregaard, "Remote-entanglement protocols for stationary qubits with photonic interfaces," *PRX Quantum*, vol. 5, p. 010202, Mar. 2024.

[10] S. Bartolucci, P. Birchall, H. Bombín, H. Cable, C. Dawson, M. Gimeno-Segovia, E. Johnston, K. Kieling, N. Nickerson, M. Pant, F. Pastawski, T. Rudolph, and C. Sparrow, "Fusion-based quantum computation," *Nature Communications*, vol. 14, p. 912, Feb. 2023.

[11] D. Poulin, "Stabilizer formalism for operator quantum error correction," *Physical Review Letters*, vol. 95, p. 230504, Dec. 2005.

[12] D. Bacon, "Operator quantum error-correcting subsystems for self-correcting quantum memories," *Physical Review A*, vol. 73, p. 012340, Jan. 2006.

[13] S. Bravyi, G. Duclos-Cianci, D. Poulin, and M. Suchara, *Subsystem Surface Codes with Three-Qubit Check Operators.* arXiv: 1207.1443 [quant-ph], Dec. 2013.

[14] M. B. Hastings and J. Haah, "Dynamically generated logical qubits," *Quantum*, vol. 5, p. 564, Oct. 2021.

[15] C. Vuillot, *Planar Floquet Codes.* arXiv: 2110.05348 [quant-ph], Dec. 2021.

[16] J. R. Wootton, *Measurements of Floquet Code Plaquette Stabilizers.* arXiv: 2210.13154 [quant-ph], Oct. 2022.

[17] O. Higgott and N. P. Breuckmann, "Subsystem codes with high thresholds by gauge fixing and reduced qubit overhead," *Physical Review X*, vol. 11, p. 031039, Aug. 2021.

[18] D. K. Tuckett, S. D. Bartlett, and S. T. Flammia, "Ultrahigh error threshold for surface codes with biased noise," *Physical Review Letters*, vol. 120, p. 050505, Jan. 2018.

[19] D. K. Tuckett, A. S. Darmawan, C. T. Chubb, S. Bravyi, S. D. Bartlett, and S. T. Flammia, "Tailoring surface codes for highly biased noise," *Physical Review X*, vol. 9, p. 041031, Nov. 2019.

[20] J. Claes, J. E. Bourassa, and S. Puri, "Tailored cluster states with high threshold under biased noise," *npj Quantum Information*, vol. 9, p. 9, Jan. 2023.

[21] M. B. Hastings, *Decoding in Hyperbolic Spaces: LDPC Codes with Linear Rate and Efficient Error Correction.* arXiv: 1312.2546 [quant-ph], Dec. 2013.

[22] L. Guth and A. Lubotzky, "Quantum error correcting codes and 4-dimensional arithmetic hyperbolic manifolds," *Journal of Mathematical Physics*, vol. 55, p. 082202, Aug. 2014.

[23] J.-P. Tillich and G. Zémor, "Quantum LDPC codes with positive rate and minimum distance proportional to the square root of the blocklength," *IEEE Transactions on Information Theory*, vol. 60, pp. 1193–1202, Feb. 2014.

[24] A. Krishna and D. Poulin, "Fault-tolerant gates on hypergraph product codes," *Physical Review X*, vol. 11, p. 011023, Feb. 2021.

[25] P. Panteleev and G. Kalachev, "Asymptotically good quantum and locally testable classical LDPC codes," in *Proceedings of the 54th Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2022, (New York, NY, USA), pp. 375–388, Association for Computing Machinery, June 2022.

[26] S. Bravyi, D. Poulin, and B. Terhal, "Tradeoffs for reliable quantum information storage in 2D systems," *Physical Review Letters*, vol. 104, p. 050503, Feb. 2010.

[27] N. Baspin and A. Krishna, "Quantifying nonlocality: How outperforming local quantum codes Is expensive," *Physical Review Letters*, vol. 129, p. 050505, July 2022.

[28] C. Monroe, R. Raussendorf, A. Ruthven, K. R. Brown, P. Maunz, L.-M. Duan, and J. Kim, "Large scale modular quantum computer architecture with atomic memory and photonic interconnects," *Physical Review A*, vol. 89, p. 022317, Feb. 2014.

[29] J. Ramette, J. Sinclair, N. P. Breuckmann, and V. Vuletić, "Fault-tolerant connection of error-corrected qubits with noisy links," *npj Quantum Information*, vol. 10, pp. 1–6, June 2024.

[30] H. Bombín, C. Dawson, R. V. Mishmash, N. Nickerson, F. Pastawski, and S. Roberts, "Logical blocks for fault-tolerant topological quantum computation," *PRX Quantum*, vol. 4, p. 020303, Apr. 2023.

[31] A. Paetznick and B. W. Reichardt, "Universal fault-tolerant quantum computation with only transversal gates and error correction," *Physical Review Letters*, vol. 111, p. 090505, Aug. 2013.

[32] C. Vuillot, L. Lao, B. Criger, C. G. Almudéver, K. Bertels, and B. M. Terhal, "Code deformation and lattice surgery are gauge fixing," *New Journal of Physics*, vol. 21, p. 033028, Mar. 2019.

[33] H. Bombín, "Gauge color codes: Optimal transversal gates and gauge fixing in topological stabilizer codes," *New Journal of Physics*, vol. 17, Aug. 2015.

[34] B. J. Brown, N. H. Nickerson, and D. E. Browne, "Fault-tolerant error correction with the gauge color code," *Nature Communications*, vol. 7, p. 12302, July 2016.

A

Protocol recipe construction and simulation

A.1 Algorithm for protocol recipe construction

In Sec. 6.3, we discuss the main principles of the algorithms used to construct a protocol recipe from a binary tree GHZ generation protocol. Here, we elaborate on some additional details of these algorithms.

A.1.1 Algorithm for operation identification and ordering

Using terminology from Ch. 6, we define $n_o \subseteq \{\nu^{(i)}\}_{i=1}^{N}$ as the subset of network nodes in which an operation o takes place. For each elementary link operation $e \in \{e_i\}_{i=1}^{K}$ considered, we identify the set $\mathbb{E}_e \subset \{e_i\}_{i=1}^{K}$, that for $e' \in \mathbb{E}_e$ fulfills $n_{e'} \cap n_e = \emptyset$, where $\emptyset$ is the empty set. The operations in $\mathbb{E}_e$ are all elementary link operations that can be carried out simultaneously with e. We store e and the operations in $\mathbb{E}_e$ together as a set s_e in the list l_{link}. The elementary links e are ordered with the recursive binary tree approach of Sec. 6.3.1.

Consequently, we identify a list l'_{link} with subsets of link generation operations that can be carried out simultaneously. For each element $o \in l_{\text{link}}$, we identify a subset s_o of link generation operations that can be carried out simultaneously with o. This subset can only contain one operation per node in the network—*i.e.*, there can, *e.g.*, be only one link generation operation in s_o that involves network node A, one link generation operation that involves network node B, *etc.*. The operation o is, itself, also part of s_o. The identified subset s_o is added to l'_{link}. Each operation in l_{link} is only added to one subset of l'_{link}.

Next, we create a list of operations l_{ops} that contains all operations that need to be carried out—*i.e.*, not just the elementary link operations considered so far. This list is created by looping over the subsets $s_o \in l'_{\text{link}}$ and by keeping track of the nodes' qubit occupancy during the operations that we add to l_{ops} (to identify the qubits on which the operations should take place). For each subset $s_o \in l'_{\text{link}}$, the link generation operations $o \in s_o$ are added to l_{ops}. Subsequently, for each $o \in s_o$, we check if we can perform its direct parent operation p in the binary tree. This operation p is either a distillation operation or a fusion operation. We check if, at this point, both direct children of p are contained

in l_{ops}. If this is the case, we add p to l_{ops}, and move on to the direct parent of p and do the same—until we stumble upon a parent node that does not have its two direct children contained in l_{ops} yet. For each operation added to l_{ops}, we first check if it is necessary to free up the communication qubits of the involved nodes, or swap back a state to the nodes' communication qubits. If one of those steps is necessary, we add SWAP gate operations to l_{ops} before the actual operation. For each fusion operation added to l_{ops}, we add information about what correction operations we need to perform in case of an odd measurement result. By default, this is a Pauli-X operator on all qubits of the right child in the tree that are not part of the network node in which the fusion operation takes place—so, *e.g.*, for a fusion operation between qubits in nodes ABC and CDE, this is a Pauli-X operator on the qubits in nodes D and E.

A.1.2 Algorithm for operation scheduling

In this section, we describe how we group the operations in the list l_{ops} into different time steps $\{s^{(\tau)}\}_\tau$ and, for each time step $s^{(\tau)}$, a set of time blocks $\{b_i^{(\tau)}\}_i$. Every time block $b \in \{b_i^{(\tau)}\}_i$ has an associated subset n_b of network nodes in which its operations are carried out. The time blocks are created by looping over the operations $o \in l_{\text{ops}}$, for which we identify the subset n_o of network nodes in which operations of o take place:

1. We start at the last time step at this stage of construction process—*i.e.*, the time step $s^{(\tau')}$ labeled with $\tau' \leftarrow |\{s^{(\tau)}\}_\tau|$. In case $|\{s^{(\tau)}\}_\tau| = 0$ holds, we create time step $s^{(1)}$ and set $\tau' \leftarrow 1$.

2. We check if there is a time block $b \in \{b_i^{(\tau')}\}_i$ that fulfills $n_o \subseteq n_b$.

    a. If this is the case, we add o to this time block b, select the next operation in l_{ops} and move to step 1.

    b. If that is not the case, we check if there is at least one time block $b \in \{b_i^{(\tau')}\}_i$ that fulfills $n_o \cap n_b \neq \emptyset$.

        i. If this is the case, we create or move to the next time step $s^{(\tau'+1)}$ and create a new time block b'' in time step $s^{(\tau'+1)}$. We set $n_{b''} \leftarrow n_o$, add o to b'', select the next operation in l_{ops} and move to step 1.

        ii. If that is not the case, we must have that $n_o \cap n_b = \emptyset$ holds for all $b \in \{b_i^{(\tau')}\}_i$. If $\tau' > 1$ holds, we decrease τ' by one and move back to the start of step 2. If $\tau' = 1$ holds, we create a new time block b' in time step $s^{(1)}$, set $n_{b'} \leftarrow n_o$, add o to b', select the next operation in l_{ops} and move to step 1.

To every time step $s^{(\tau)}$, we add a list $l_{\text{corr}}^{(\tau)}$ of fusion corrections and a list $l_{\text{eval}}^{(\tau)}$ of distillation operations that need to be applied and evaluated after the time step. Fusion operators need corrections when their full measurement outcome is odd. Typically, each correction is not applied in the nodes where the fusion operation itself takes place. Because the nodes in which the correction has to be applied are typically part of a different time block of the time step, we collect all fusion corrections and delay them until the end of a time step. Sometimes, fusion corrections influence the outcome of distillation measurements. This can occur if the fusion correction in a different time block does not commute with one of the measurements of the distillation operation. If this is the case, the success or failure of a

distillation operation cannot be determined in the time block itself. In that case, we also delay the decision on whether or not distillation was successful to the end of a time step, where all fusion measurement outcomes are available. Similarly, it can occur that a fusion correction is altered by operations applied between its associated fusion operation and the end of the time step. We use standard commutation rules to modify these corrections (and their dependencies) in the instructions contained in $l_{\text{corr}}^{(\tau)}$.

In more technical terms, the above is achieved by using a standard format for adding fusion correction operations that are newly added to time step $s^{(\tau)}$ in the list $l_{\text{corr}}^{(\tau)}$—see the end of Sec. A.1.1 for this standard format. This is possible since, by default, every operation is always added to the end of a time block. Every time we add a new operation o to one of the time blocks of a certain time step $s^{(\tau')}$, we have to evaluate its influence on the existing fusion corrections in $l_{\text{corr}}^{(\tau'')}$ for all time steps $s^{(\tau'')}$ in the range $\tau' \leq \tau'' \leq |\{s^{(\tau)}\}_\tau|$. If o is a distillation operation that does not commute with at least one fusion correction in $l_{\text{corr}}^{(\tau'')}$ for $\tau' \leq \tau'' \leq |\{s^{(\tau)}\}_\tau|$, o is added to $l_{\text{eval}}^{(\tau''')}$ and the non-commuting fusion operation(s) are added to the outcomes required to determine the success of o. Here, $s^{(\tau''')}$ corresponds to the latest time step in the range $\tau' \leq \tau''' \leq |\{s^{(\tau)}\}_\tau|$ that contains fusion corrections that do not commute with the distillation operation o.

A.2 Algorithm for protocol recipe simulation

In this section, we expand on the description in Sec. 6.4 regarding protocol recipe simulation. The implementation discussed below includes Alg. 3 for identifying all operations that need to be reapplied in case of a failed distillation operation.

A.2.1 General description

In Sec. 6.4, we mention that we have to make use of a structure that keeps track of operations that should be skipped while executing the protocol recipe. This is because recreating a state after a failed distillation operation typically only requires reapplying a subset of all operations prior to the failed distillation operation. Specifically, we use two objects for this task: l_{recreate} and l_{clear}. During protocol recipe execution, we only apply operations contained in the last object of l_{recreate}, and skip all other operations that we encounter. The structure l_{clear} contains information on when the last object in l_{recreate} can be removed—*e.g.*, if we have successfully recreated a state with a distillation operation that failed at an earlier stage of the protocol, we remove the associated object from l_{recreate}. The second-to-last object in l_{recreate} then becomes the last object and determines which operations are executed and skipped. At the start of the simulation process, the structure l_{recreate} is empty—with an empty l_{recreate} all operations are applied.

In reconstruction mode, we deterministically reapply all time blocks $\{b_i^{(\tau')}\}_i$ in this time step $s^{(\tau')}$ until they reach time t_{fail}. Note that, because a distillation operation contains a set of operations that need to be carried out locally in the network nodes—and not necessarily simultaneously in all nodes—it could occur that a full distillation operation is only executed partially before t_{fail} occurs. If this is the case, we typically finish the rest of the operation when we reach this stage in the protocol again after fully recreating the failed state. Every time we enter reconstruction mode, an empty list l_{skip} is initialized. In this list, we collect all operations in the time step that are skipped because they fall outside the time t_{fail}.

Algorithm 3: Pseudo-code used to identify the failure-reset-level and a list of operations that need to be re-applied in case of a failed distillation attempt.

Data: List l_{d} of operations that need to be re-applied
Dictionary δ_{qubits} with states currently stored on qubits
Binary tree of the protocol
Protocol recipe
Result: Failure-reset-level of operations in l_{d}
List of operations that have to be reapplied

$l_{\mathrm{reset}} \leftarrow \emptyset$
while $l_{\mathrm{d}} \neq \emptyset$ **do**
  for $o \in l_{\mathrm{d}}$ **do**
    Add o to l_{reset}
    Add all children of o to l_{reset}
    Add all parents of o that are currently present in δ_{qubits} to l_{reset}, including all their children in the binary tree
  Identify all operations l_{occ} that sit on qubits in δ_{qubits} that need to be empty if we want to reapply the operations in l_{reset}
  $l_{\mathrm{d}} \leftarrow l_{\mathrm{occ}}$
Identify the first operation $o_{\mathrm{frl}} \in l_{\mathrm{reset}}$ in the protocol recipe
return o_{frl}, l_{reset}.

When, in reconstruction mode, the end of the time step is reached, we calculate a list l_{reset} with all operations that have to be reapplied because of failed operation(s) in the last time step. This list is calculated using Alg. 3. It contains all operations in the sub-trees of the failed distillation operation(s). On top of that, it can occur that one or more of their parents are also executed at this point. In that case, these parents also need to be reapplied to recover the state at the end of this time step. To evaluate what parent operations need to be reapplied, we keep track of a "real-time" dictionary δ_{qubits} that describes what binary tree states currently sit on which network node qubits—*i.e.*, what operations in our original binary tree are successfully created at the current stage of the protocol recipe execution. On top of that, we also add all operations in l_{skip} to l_{reset}, but we exclude SWAP operations that act on states not contained in l_{reset}. Lastly, we also make sure to recreate states that are present on network node qubits that need to be empty to reapply the operations in l_{reset}: these states—including their sub-trees and (possibly) parents—are also added to l_{reset}. We define the failure reset level as the operation $o_{\mathrm{frl}} \in l_{\mathrm{reset}}$ that appears first in the protocol recipe. We add l_{reset} to the end of l_{recreate} and store information in l_{clear} that tells us we can remove this list from l_{recreate} as soon as we reach the end of this time step again. After that, we tell the algorithm to go back to operation o_{frl} in the protocol recipe and continue normal execution from there.

A.2.2 Step-by-step description

Below, we present a more precise description of the execution of a protocol recipe in the presence of a GHZ cycle time t_{GHZ}. We denote the "local" time in a network node $v \in \{v^{(i)}\}_{i=1}^{N}$ by t_v.

1. We set $t_v \leftarrow 0$ in all nodes $v \in \{v^{(i)}\}_{i=1}^{N}$. We create empty structures l_{recreate} and l_{clear}. We select execution mode.

2. We select $\tau' \leftarrow 1$ and select the time step $s^{(\tau')}$. We set $i \leftarrow 1$ and select the time block $b_i^{(\tau')}$ of this time step $s^{(\tau')}$.

3. We select the first operation o in time block $b_i^{(\tau')}$.

4. The operation o is only applied if it is contained in the last list of l_{recreate} or if l_{recreate} is empty. Without loss of generality, we assume that o performs operations in network nodes $n_o = \{\mu_j^{(o)}\}_j$ with $t_{\mu_1^{(o)}} \leq t_{\mu_2^{(o)}} \leq \ldots$, for $\mu_j^{(o)} \in \{v^{(i)}\}_{i=1}^{N}$.

   a. If, for any of the involved nodes $\mu \in n_o$, $t_\mu \geq t_{\text{GHZ}}$ holds *after* o would have taken place, the operations of o in node μ are not executed, and we "switch off" node μ, as it has now reached the GHZ cycle time. If all nodes $v \in \{v^{(i)}\}_{i=1}^{N}$ have reached the GHZ cycle time, the full protocol is aborted.

   b. If the operation creates an entangled link between network nodes $\mu \equiv \mu_1^{(o)}$ and $\mu' \equiv \mu_2^{(o)}$, we set $t_\mu \leftarrow t_{\mu'}$.

   c. If we are in reconstruction mode, we check if $t_\mu \geq t_{\text{fail}}$ holds for all of the involved nodes $\mu \in n_o$. If that is the case, we skip (part of) the operation o in node μ and add o to l_{skip}.

   d. We carry out o in the nodes $\mu \in n_o$ that are not yet switched off and (in case we are in reconstruction mode) have not yet reached t_{fail}. We add the time it takes to perform this operation to t_μ of the involved network node μ.

   e. As long as o is the last element of l_{clear}, we remove the last elements of l_{clear} and l_{recreate}.

   f. If o is a distillation operation and its success can be evaluated here, we do so:

      i. If the distillation operation succeeds, we proceed as if nothing happened.

      ii. If the distillation operation fails, we calculate l_{reset} and o_{frl} for just the operation o using the version of δ_{qubits} at the current stage of the protocol and Alg. 3.

      iii. In case of a failed distillation operation, if o_{frl} is an operation in the same time block $b_i^{(\tau')}$ as o, we add o at the end of l_{clear} and add l_{reset} at the end of l_{recreate}. We then set $o \leftarrow o_{\text{flr}}$ and move back to the start of step 4.

      iv. In case of a failed distillation operation, if o_{frl} is in a different time block as $b_i^{(\tau')}$, and we are not in reconstruction mode, we set t_{fail} to the time when the full measurement result of o was known. We create a list l_{fail} and add o to this list. The list l_{fail} contains failed distillation operations in the current instance of reconstruction mode. We set l_{skip} to the empty list. We reset l_{recreate}, l_{clear} and δ_{qubits} to their values at the start of this time step $s^{(\tau')}$,

set $i \leftarrow 1$ to select the first time block $b_i^{(\tau')}$ of this time step $s^{(\tau')}$. We enter reconstruction mode and move back to step 3.

v. In case of a failed distillation operation, if o_{frl} is in a different time block as $b_i^{(\tau')}$, and we are already in reconstruction mode, we check if o is in l_{fail}. If that is the case, we proceed as if nothing happened. If that is not the case and the time at which the full measurement result of o was known is smaller than t_{fail}, we reset t_{fail} to this earlier time, add o to l_{fail}, reset l_{recreate}, l_{clear} and δ_{qubits} to their values at the start of this time step $s^{(\tau')}$, set $i \leftarrow 1$ to select the first time block $b_i^{(\tau')}$ of this time step $s^{(\tau')}$, and move back to step 3. If o is not in l_{fail}, but the time at which the full measurement result of o was known exceeds t_{fail}, we add o to l_{fail} and proceed as if nothing happened.

If o is not the last operation in $b_i^{(\tau')}$, we select the next operation in $b_i^{(\tau')}$ as the new operation o and move back to start of step 4. If o is the last operation in $b_i^{(\tau')}$ and $i < |\{b_j^{(\tau')}\}_j|$ holds, we increase i by one and move back to step 3. If o is the last operation in $b_i^{(\tau')}$ and $i = |\{b_j^{(\tau')}\}_j|$ holds, we move to step 5.

5. We have reached the end of the time step $s^{(\tau')}$. As long as this time step is the last element of l_{clear}, we remove the last objects of l_{clear} and l_{recreate}. Subsequently, if we are not in reconstruction mode, we move to step 6. Otherwise—*i.e.*, if we *are* in reconstruction mode—we add this time step at the end of l_{clear}. On top of that, we calculate l_{reset} and o_{frl} with Alg. 3, using l_{fail} as the list of operations in Alg. 3. Lastly, we add each operation $o' \in l_{\text{skip}}$ to l_{reset} as well. Here, we exclude operations $o' \in l_{\text{skip}}$ that are SWAP gates swapping a state not contained in l_{reset}. We enter execution mode, set $o \leftarrow o_{\text{frl}}$, and move back to step 4.

6. We evaluate the full results of distillation operations $o'' \in l_{\text{eval}}^{(\tau')}$. Here, if l_{recreate} is not empty, we skip distillation operations that are not part of the last element of l_{recreate}.

   a. As soon as one distillation operation o'' fails, we add this location in the protocol recipe at the end of l_{clear}, calculate l_{reset} and o_{frl} with Alg. 3 for just operation o'', and add l_{reset} at the end of l_{recreate}. We set $o \leftarrow o_{\text{frl}}$ and move back to step 4.

   b. If all distillation evaluations succeed, we move to step 7.

7. We evaluate and carry out the fusion corrections in $l_{\text{corr}}^{(\tau')}$.

8. If $\tau' < |\{s^{(\tau)}\}_\tau|$ holds, we increase τ' by one, set $i \leftarrow 1$, set $t_v \leftarrow \max_{v'} t_{v'}$ for all $v, v' \in \{v^{(i)}\}_{i=1}^N$, and move to step 3. If $\tau' = |\{s^{(\tau)}\}_\tau|$ holds, we set $t_v \leftarrow \max_{v'} t_{v'}$ for all $v, v' \in \{v^{(i)}\}_{i=1}^N$ and stop the protocol.

B

Superoperator calculation and convergence

B.1 Superoperator calculation

In this section, we describe how we calculate the superoperator that we use in the surface code simulations. Separately calculating a superoperator has the advantage that it breaks up the process of calculating GHZ state creation from the threshold simulations: this drastically decreases the complexity of the full calculation.

Following earlier work by Nickerson *et al.* [1, 2], we assume that only Pauli errors occur on the data qubits during the toric code simulations. This simplifies the simulation, as every stabilizer measurement now deterministically measures either $+1$ or -1, and measurement results can be calculated by simply considering commutativity between Pauli errors and the stabilizer operators. In most situations, the stochastic Pauli error model can be considered as a good approximation for coherent errors described by continuous rotations [3]. On top of that, since the nuclear spin qubits (*i.e.*, the memory qubits) of NV centers have no states to leak to, it is believed that a depolarizing channel (*i.e.*, Pauli noise) is a good approximation for noise on these qubits.

Our characterization of the toric code stabilizer measurements is carried out with density matrix calculations that *do* include more general errors. To align these calculations with the toric code calculations themselves, the stabilizer measurement channel is twirled over the Pauli group [4–6]. This makes sure the superoperators describing the channel only contain Pauli errors. Each superoperator is constructed via the channel's Choi state—*i.e.*, by using the GHZ state created by the concerning protocol to non-locally perform the stabilizer measurement on half of the maximally entangled state.

To explain this process in more detail, we consider the states $|\Psi_\pm\rangle$ that follow from projecting half of the maximally entangled state $|\Psi\rangle$ on the $+P^{\otimes 4}$ and $-P^{\otimes 4}$ subspaces with projectors $\Pi_\pm = (\mathbb{I}^{\otimes 8} \pm P^{\otimes 4} \otimes \mathbb{I}^{\otimes 4})/2$. Here, $P \in \{X, Z\}$ describes the two types of stabilizer measurements of the toric code, and $|\Psi\rangle$ is the eight-qubit maximally entangled state describing the four data qubits of the code. We also define states $|\Psi_\pm^{(m)}\rangle$ that describe Pauli errors $P_m \in \{\mathbb{I}, X, Y, Z\}^{\otimes 4}$ occuring on the first half of $|\Psi_\pm\rangle$ after the projection with $\Pi_\pm$. We

define the these states in the following way:

$$\begin{aligned}|\Psi\rangle &= \frac{1}{\sqrt{2^4}}\sum_{j=0}^{2^4-1}|j\rangle\otimes|j\rangle,\\ |\Psi_\pm\rangle &= \frac{\mathbb{I}^{\otimes 8}\pm P^{\otimes 4}\otimes\mathbb{I}^{\otimes 4}}{\sqrt{2}}|\Psi\rangle,\\ |\Psi_\pm^{(m)}\rangle &= (P_m\otimes\mathbb{I}^{\otimes 4})|\Psi_\pm\rangle.\end{aligned} \tag{B.1}$$

Later in the analysis, we only consider the subset of Pauli operators P_m that lead to orthogonal states $|\Psi_\pm^{(m)}\rangle$, *i.e.*, we only use P_m that make sure we have

$$\langle\Psi_s^{(m)}|\Psi_{s'}^{(n)}\rangle = \delta_{mn}\delta_{ss'}, \tag{B.2}$$

with $(s,s')\in\{+,-\}^2$. We call this subset $\mathcal{E}\subseteq\{\mathbb{I},X,Y,Z\}^{\otimes 4}$.

We define two versions of the full noisy stabilizer measurement channel: $\mathcal{N}_+$, which projects with Π_+, and $\mathcal{N}_-$, which projects with Π_-:

$$\mathcal{N}_s(\rho) = \sum_i \mathcal{K}_s^{(i)}\Pi_s\rho\Pi_s(\mathcal{K}_s^{(i)})^\dagger. \tag{B.3}$$

Here, $s\in\{+,-\}$. The Kraus operators $\mathcal{K}_s^{(i)}$ describe the noise on the data qubits. Each of them can be decomposed into Pauli matrices:

$$\mathcal{K}_s^{(i)} = \sum_{P_q\in\{\mathbb{I},X,Y,Z\}^{\otimes 4}} \xi_{s,q}^{(i)}P_q. \tag{B.4}$$

Using this decomposition, the channel's Choi state can be expressed in the following way:

$$\begin{aligned}\rho_{\text{Choi}} &= \sum_s(\mathcal{N}_s\otimes\mathbb{I}^{\otimes 4})(|\Psi\rangle\langle\Psi|)\\ &= \sum_s\sum_i\sum_{P_q,P_{q'}}\xi_{s,q}^{(i)}\left(\xi_{s,q'}^{(i)}\right)^*|\Psi_s^{(q)}\rangle\langle\Psi_s^{(q')}|.\end{aligned} \tag{B.5}$$

We now focus on ρ_{Choi} as the post-measurement state for stabilizer measurement outcome $+1$. The influence of noise can cause measurement errors, meaning ρ_{Choi} can contain terms projected with Π_-. One can extract coefficients $p_s^{(m)}$ from ρ_{Choi} by constructing the states $|\Psi_s^{(m)}\rangle$ from Pauli operators $P_m\in\mathcal{E}$ via:

$$p_s^{(m)} = \langle\Psi_s^{(m)}|\rho_{\text{Choi}}|\Psi_s^{(m)}\rangle = \sum_i\left|\xi_{s,m}^{(i)}\right|^2. \tag{B.6}$$

These are the coefficients of the Pauli operators that act as the stabilizer measurement channel's Kraus operators after the channel is twirled over the Pauli group—see Sec. 2.4.4 for more details.

We see that this procedure gives us the probabilities required to construct the superoperator of the channel. If ρ_{Choi} is constructed by preparing the post-measurement

state according to a +1 measurement outcome, the coefficients $p_+^{(m)}$ give rise to the Pauli errors $P_m \in \mathcal{E}$ without a measurement error on the stabilizer measurement, whereas the coefficients $p_-^{(m)}$ describe Pauli errors P_m accompanied with a measurement error. If ρ_{Choi} is constructed with a -1 measurement outcome, the role of $p_+^{(m)}$ and $p_-^{(m)}$ is inverted, but the parameter values themselves are the same.

The stabilizer fidelity is defined as the coefficient $p_s^{(m)}$ corresponding to $P_m = \mathbb{I}^{\otimes 4} \otimes \mathbb{I}^{\otimes 4}$ (*i.e.*, no errors on the data qubits) and no stabilizer measurement error. In our search for well-performing GHZ creation protocols, a good reason for comparing two protocols by using the stabilizer fidelity over, *e.g.*, the GHZ state fidelity is the fact that the surface code data qubits undergo more decoherence for protocols that take longer to finish. This aspect of the optimization problem is not taken into account if we just use the GHZ fidelity to compare the protocols.

B.2 Convergence of the average superoperator

The construction of a superoperator that we use in the surface code simulator requires averaging over a large number of Monte Carlo simulations. In this section, we investigate the convergence of the average superoperator over an increasing number of Monte Carlo samples. In Fig. B.1, we calculate the average Choi state $\overline{\rho}_{\text{Choi}}$ over $3 \cdot 10^5$ Monte Carlo iterations and calculate the trace distance of this state with the average Choi state $\overline{\rho}_{\text{Choi}}^{(i)}$ after a smaller number of i iterations. As explained in more detail below, this figure suggests that, after 10^5 Monte Carlo samples, errors in the average superoperator elements are on the order of 10^{-4}.

For $s \in \{+,-\}$ and $P_m \in \mathcal{E}$, the superoperator used in threshold simulations is calculated as the average $\overline{\mathcal{S}} = \{\overline{p}_s^{(m)}\}_{s,m}$ of the individual superoperators $\mathcal{S} = \{p_s^{(m)}\}_{s,m}$ calculated with the method of Sec. B.1. Alternatively, this average superoperator can be constructed by calculating the average Choi state $\overline{\rho}_{\text{Choi}}$, as defined in Sec. B.1, and using Eq. (B.6) to calculate $\{\overline{p}_s^{(m)}\}_{s,m}$ from this average Choi state.

The trace distance $\mathcal{T}(\rho, \rho')$ between two density matrices ρ and ρ' is defined in Eq. (2.16). For an operator $\mathcal{O}$ with eigenvalues $0 \leq \xi_j \leq 1$, we can show that the following holds [7]:

$$\left|\text{Tr}\big(\mathcal{O}(\rho - \rho')\big)\right| \leq \mathcal{T}(\rho, \rho'). \tag{B.7}$$

This means that, for $\overline{\rho}_{\text{Choi}}^{(i)}$ after a certain iteration i, $\mathcal{T}(\overline{\rho}_{\text{Choi}}, \overline{\rho}_{\text{Choi}}^{(i)})$ provides an upper bound on the difference between the superoperator $\overline{\mathcal{S}}^{(i)}$ after iteration i and the superoperator $\overline{\mathcal{S}}$ after the full number of iterations. This is because, for the difference in the elements of $\overline{\mathcal{S}}$ and $\overline{\mathcal{S}}^{(i)}$, the following holds:

$$\begin{aligned}
\Delta\overline{p}_s^{(m)} &= \left|\langle\Psi_s^{(m)}|\overline{\rho}_{\text{Choi}}|\Psi_s^{(m)}\rangle - \langle\Psi_s^{(m)}|\overline{\rho}_{\text{Choi}}^{(i)}|\Psi_s^{(m)}\rangle\right| \\
&= \left|\text{Tr}\left(|\Psi_s^{(m)}\rangle\langle\Psi_s^{(m)}|\left(\overline{\rho}_{\text{Choi}} - \overline{\rho}_{\text{Choi}}^{(i)}\right)\right)\right| \\
&\leq \mathcal{T}\left(\overline{\rho}_{\text{Choi}}, \overline{\rho}_{\text{Choi}}^{(i)}\right).
\end{aligned} \tag{B.8}$$

Calculating $\mathcal{T}(\overline{\rho}_{\text{Choi}}, \overline{\rho}_{\text{Choi}}^{(i)})$, therefore, gives us information about the convergence of the average superoperator elements after i iterations.

B

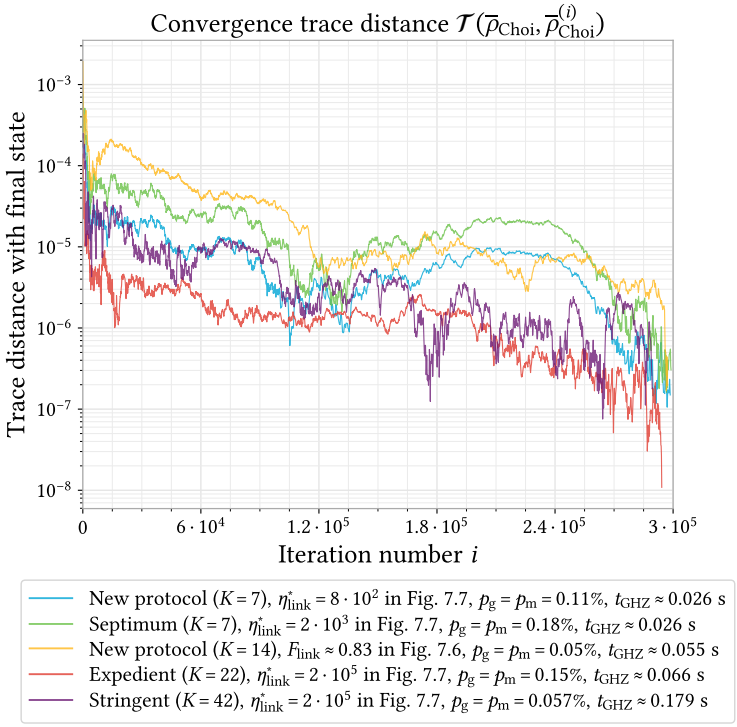


Figure B.1: Convergence of the trace distance between $\overline{\rho}_{\text{Choi}}$ (the average Choi state after $3 \cdot 10^5$ Monte Carlo iterations) and $\overline{\rho}^{(i)}_{\text{Choi}}$ (the average Choi state after i iterations). We track changes in the trace distances by varying i on the x-axis of the plot. For each data point, we add on the order of 100 new iterations to $\overline{\rho}^{(i)}_{\text{Choi}}$. In the plot, we include five GHZ protocols with a varying number of distillation steps K. We only include Choi states based on $X^{\otimes 4}$ stabilizer measurements and exclude iterations that did not finish within the GHZ cycle time t_{GHZ}.

In our simulations, as shown schematically in Fig. 7.5, the superoperator $\{\overline{p}_s^{(m)}\}_{s,m}$ is subsequently used in a second level of Monte Carlo simulations that emulates the operation of the surface code with this specific superoperator. In App. C, we discuss how, for a specific $\{\overline{p}_s^{(m)}\}_{s,m}$, the statistical uncertainty in the Monte Carlo simulations of the surface code leads to uncertainty in the calculated threshold value.

References

[1] N. H. Nickerson, Y. Li, and S. C. Benjamin, "Topological quantum computing with a very noisy network and local error rates approaching one percent," *Nature Communications*, vol. 4, p. 1756, Dec. 2013.

[2] N. H. Nickerson, J. F. Fitzsimons, and S. C. Benjamin, "Freely scalable quantum technologies using cells of 5-to-50 qubits with very lossy and noisy photonic links," *Physical Review X*, vol. 4, p. 041041, Dec. 2014.

[3] D. Greenbaum and Z. Dutton, "Modeling coherent errors in quantum error correction," *Quantum Science and Technology*, vol. 3, p. 015007, Dec. 2017.

[4] W. Dür, M. Hein, J. I. Cirac, and H.-J. Briegel, "Standard forms of noisy quantum operations via depolarization," *Physical Review A*, vol. 72, p. 052326, Nov. 2005.

[5] M. R. Geller and Z. Zhou, "Efficient error models for fault-tolerant architectures and the Pauli twirling approximation," *Physical Review A*, vol. 88, p. 012314, July 2013.

[6] Z. Cai and S. C. Benjamin, "Constructing smaller Pauli twirling sets for arbitrary error channels," *Scientific Reports*, vol. 9, p. 11281, Aug. 2019.

[7] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information*. Cambridge University Press, 2000.

C

Fitting procedure threshold plots

C.1 Regression model

In this section, we describe how we determine threshold values for the toric surface code considered in Ch. 7 and the fault-tolerant cluster states of Ch. 8. This involves varying one or more of the error probabilities. The fitting procedure described below is implemented with the `optimize.curve_fit` function of the SciPy package [1] for Python in Ch. 7 and the `minimize` function of the LMFIT package [2] for Python in Ch. 8.

For each combination of error probabilities p, we calculate an average superoperator or noise channel using the methods described in Apps. B and F. At a specific p, we then use Monte Carlo simulations to calculate the logical success probability r of the toric surface code for multiple lattice sizes L. The logical error probability is defined as $1-r$. We denote the observed logical success probability of a certain input combination (p_i, L_i) by r_i. We make use of n_C to describe the total number of input combinations: $\{(p_i, L_i)\}_{i=1}^{n_C}$. For a single (p_i, L_i) combination, the logical success probability is defined as the number of error-correction iterations M_i that do not induce a logical error divided by the full number of error-correction iterations N_i. In this context, N_i can be considered as the number of Monte Carlo iterations used for surface code calculations for a certain (p_i, L_i) and the exact hardware configuration used. We assume that the uncertainty in the observed logical success probabilities is described by the binomial distribution. This means that the standard deviation can be estimated via

$$\sigma_i = \sqrt{\frac{r_i(1-r_i)}{N_i}}, \quad \text{where } r_i = \frac{M_i}{N_i}. \tag{C.1}$$

We see that a logical success probability r_i determined with a higher number of iterations N_i generally has lower uncertainty compared to a probability determined with a lower number of iterations.

Following Wang *et al.* [3], we fit the logical success probabilities $\{r_i\}_i$ or the logical error probabilities $\{1-r_i\}_i$ with either the model

$$\hat{r} \equiv \hat{a} + \hat{b}(p - \hat{p}_{\text{th}})L^{1/\hat{\kappa}} + \hat{c}(p - \hat{p}_{\text{th}})^2 L^{2/\hat{\kappa}} + \hat{d}L^{-1/\hat{\zeta}}, \tag{C.2}$$

or with the model

$$\hat{r} \equiv \hat{a} + \hat{b}(p - \hat{p}_{\text{th}})L^{1/\hat{\kappa}} + \hat{c}(p - \hat{p}_{\text{th}})^2 L^{2/\hat{\kappa}}. \tag{C.3}$$

The model of Eq. (C.3) can be considered a simplified version of the model of Eq. (C.2) with less fitting parameters. Specifically, we use Eq. (C.2) for the fits in Ch. 7 and Eq. (C.3) for the fits in Ch. 8. Using one of these models, we find *estimates* $\{\hat{r}_i\}_i$ of the logical success probabilities for all input combinations $\{(p_i, L_i)\}_i$. For a certain (p_i, L_i), the *residual* $\hat{\epsilon}_i$ is defined as the difference between the observed logical success probability and the estimated value: $\hat{\epsilon}_i = r_i - \hat{r}_i$. Values for the fitting parameters $\hat{a}$, $\hat{b}$, $\hat{c}$, $\hat{p}_{\text{th}}$, $\hat{\kappa}$, ($\hat{d}$, and $\hat{\zeta}$) are found by identifying their (local) minimum with respect to the sum $\mathcal{W}$ of the "weighted" squared residuals. This sum is defined in the following way:

$$\mathcal{W} \equiv \sum_{i=1}^{n_{\text{C}}} \left(\frac{\hat{\epsilon}_i}{\sigma_i} \right)^2 = \sum_{i=1}^{n_{\text{C}}} \left(\frac{r_i - \hat{r}_i}{\sigma_i} \right)^2. \tag{C.4}$$

We see that this approach makes sure that residuals of data points that are determined with high uncertainty (*i.e.*, with a high standard deviation σ_i) are given less priority in the least-squares fit than data points with low uncertainty.

C.2 Weighted least-squares fitting procedure

To understand how the confidence intervals in the values of the fitting parameters are determined, we delve a bit deeper into how one could determine fitting parameters for a non-linear regression like Eq. (C.2). In line with convention, we denote our input configuration as $\mathcal{X}_i = (p_i, L_i)$, and we write $\hat{r}_i$ as $\hat{r}_i = f(\hat{\beta}, \mathcal{X}_i)$. Here, the function f is the function of Eqs. (C.2) or (C.3), and $\hat{\beta} = (\hat{a}, \hat{b}, \hat{c}, \hat{d}, \hat{p}_{\text{th}}, \hat{\kappa}, \hat{\zeta})$ or $\hat{\beta} = (\hat{a}, \hat{b}, \hat{c}, \hat{p}_{\text{th}}, \hat{\kappa})$ describes the converged values for the fitting parameters after the optimization. We define $n_{\text{P}} = 7$ for Eq. (C.2) or $n_{\text{P}} = 5$ for Eq. (C.3) as the number of fitting parameters of the model. Furthermore, we use the notation $\beta^{(t)}$ to indicate the fitting parameter values at a certain step t during the optimization. We can now write $r_i = f(\beta^{(t)}, \mathcal{X}_i) + \epsilon_i^{(t)}$ for each t. Here, the residuals also contain the superscript (t) to denote that they depend on the exact values of $\beta^{(t)}$.

Finding the least-squares fit is now achieved with the *Gauss-Newton algorithm* [4, 5]. This method is a variant of Newton's method for finding the minimum of a non-linear function. We start with a guess $\beta^{(1)}$ for the fitting parameter values. These values are then iteratively updated by using the fact that we want to minimize the parameter $\mathcal{W}$ of Eq. (C.4) until they converge. To go from a certain $\beta^{(t)}$ to a new improved version $\beta^{(t+1)}$, we write a new estimation $r_i = f(\beta^{(t+1)}, \mathcal{X}_i) + \epsilon_i^{(t+1)}$ in terms of the old fitting parameter values $\beta^{(t)}$. In the expression with $\beta^{(t+1)}$, we Taylor expand $f(\beta^{(t+1)}, \mathcal{X}_i)$ around $\beta^{(t)}$, and

neglect the second and higher order terms:

$$r_i \approx f(\beta^{(t)}, \mathcal{X}_i) + \left.\frac{\partial f}{\partial \beta_1}\right|_{(\beta^{(t)}, X_i)} \left(\beta_1^{(t+1)} - \beta_1^{(t)}\right) + \ldots + \left.\frac{\partial f}{\partial \beta_{n_\mathrm{P}}}\right|_{(\beta^{(t)}, X_i)} \left(\beta_{n_\mathrm{P}}^{(t+1)} - \beta_{n_\mathrm{P}}^{(t)}\right) + \epsilon_i^{(t+1)}. \tag{C.5}$$

Here, β_1 is the first fitting parameter, β_2 the second, *etc.* The vertical lines next to the derivatives indicate that, after taking the derivatives, the functions must be evaluated at the old values $(\beta^{(t)}, \mathcal{X}_i)$.

At this point, we recognize that we can express the full system of equations of Eq. (C.5) as a matrix equation—*i.e.*, for each input and output combination of $\mathcal{X}_i$ and r_i as a row of this equation. To this end, we put the fitting parameter values at step t of the optimization process into a $n_\mathrm{P} \times 1$ column vector $\boldsymbol{\beta}^{(t)}$. For our model, we have $\boldsymbol{\beta}^{(t)} = [a^{(t)} \quad b^{(t)} \quad c^{(t)} \quad d^{(t)} \quad p_\mathrm{th}^{(t)} \quad \kappa^{(t)} \quad \zeta^{(t)}]^T$ or $\boldsymbol{\beta}^{(t)} = [a^{(t)} \quad b^{(t)} \quad c^{(t)} \quad p_\mathrm{th}^{(t)} \quad \kappa^{(t)}]^T$. We do the same for the values of $\{\mathcal{X}_i\}_i$ and $\{\epsilon_i^{(t)}\}_i$, and write them as $n_\mathrm{C} \times 1$ column vectors $\boldsymbol{\mathcal{X}}$ and $\boldsymbol{\epsilon}^{(t)}$, respectively. On top of this, we define $\Delta\boldsymbol{\beta}^{(t+1)}$ as $\Delta\boldsymbol{\beta}^{(t+1)} = \boldsymbol{\beta}^{(t+1)} - \boldsymbol{\beta}^{(t)}$. These parameters tell us how the new values for $\boldsymbol{\beta}^{(t+1)}$ depend on the old values $\boldsymbol{\beta}^{(t)}$. The full system of equations of Eq. (C.5) can now be written as [4, 5]

$$\boldsymbol{\epsilon}^{(t+1)} = \boldsymbol{\epsilon}^{(t)} - \boldsymbol{J}^{(t)} \Delta\boldsymbol{\beta}^{(t+1)}. \tag{C.6}$$

In Eq. (C.6), we have used that $r_i - f(\beta^{(t)}, \mathcal{X}_i) = \epsilon_i^{(t)}$. Furthermore, the matrix $\boldsymbol{J}^{(t)}$ is an $n_\mathrm{C} \times n_\mathrm{P}$ matrix that contains the derivatives of f with respect to the fitting parameters, evaluated at $\beta^{(t)}$ and the inputs $\{\mathcal{X}_i\}_i$. More specifically, the jth column of $\boldsymbol{J}^{(t)}$ contains derivatives with respect to β_j, and the ith row of $\boldsymbol{J}^{(t)}$ contains these derivatives evaluated at the input $\mathcal{X}_i$. This matrix is also known as the *Jacobian* matrix.

The quantity that we want to minimize can now be expressed in terms of $\boldsymbol{\epsilon}^{(t+1)}$. To realize this, we define a diagonal matrix containing the variances of the observed values r_i as $\Sigma = \mathrm{diag}(\sigma_1^2, \sigma_2^2, \ldots, \sigma_{n_\mathrm{C}}^2)$. This allows us to reformulate $\mathcal{W}$ as

$$\begin{aligned} \mathcal{W} &= \left(\boldsymbol{\epsilon}^{(t+1)}\right)^T \Sigma^{-1} \boldsymbol{\epsilon}^{(t+1)} \\ &= \left(\boldsymbol{\epsilon}^{(t)} - \boldsymbol{J}^{(t)} \Delta\boldsymbol{\beta}^{(t+1)}\right)^T \Sigma^{-1} \left(\boldsymbol{\epsilon}^{(t)} - \boldsymbol{J}^{(t)} \Delta\boldsymbol{\beta}^{(t+1)}\right). \end{aligned} \tag{C.7}$$

To minimize $\mathcal{W}$ with respect to $\Delta\boldsymbol{\beta}^{(t+1)}$, one sets $\partial\mathcal{W}/\partial\Delta\boldsymbol{\beta}^{(t+1)}$ to zero. After expanding the brackets and taking the partial derivative, we end up with an expression that we can solve for $\Delta\boldsymbol{\beta}^{(t+1)}$. This results in [5]

$$\Delta\boldsymbol{\beta}^{(t+1)} = \left(\left(\boldsymbol{J}^{(t)}\right)^T \Sigma^{-1} \boldsymbol{J}^{(t)}\right)^{-1} \left(\boldsymbol{J}^{(t)}\right)^T \Sigma^{-1} \boldsymbol{\epsilon}^{(t)}. \tag{C.8}$$

This expression tells us how to get from the existing $\boldsymbol{\beta}^{(t)}$ fitting parameter values to a new set of values $\boldsymbol{\beta}^{(t+1)}$ by using the Jacobian evaluated at the old $\boldsymbol{\beta}^{(t)}$, the matrix Σ that describes the uncertainty in the observations, and the residuals $\{\epsilon_i^{(t)}\}_i$ associated with the old $\boldsymbol{\beta}^{(t)}$. The new values $\boldsymbol{\beta}^{(t+1)}$ generally lead to a lower value for $\mathcal{W}$. In deriving the

expression of Eq. (C.8), when taking the partial derivative with respect to $\Delta\boldsymbol{\beta}^{(t+1)}$, we have assumed that $\boldsymbol{J}^{(t)}$ does not depend on $\Delta\boldsymbol{\beta}^{(t+1)}$. For a general non-linear regression like Eqs. (C.2) and (C.3), this assumption is incorrect. It is exactly the incorrectness of this assumption that makes that, for a non-linear regression model, the least-squares fitting parameters have to be found iteratively. The technique is borrowed from linear regressions, where the Jacobian is independent of the fitting parameters, and only one iteration is required to find the least-squares fit.

As a final note, we emphasize that including the full covariance matrix for the observed quantities as Σ, if this information is available, leads to the same equations.

C.3 Uncertainty in fitting parameter values

The idea of the fitting procedure is that, after sufficient iterations $t \gg 1$, $\boldsymbol{\beta}^{(t+1)}$ describes the final (converged) parameter values $\hat{\boldsymbol{\beta}}$. Of course, in theory, in the limit $n_C \to \infty$, the fitting parameter values $\hat{\boldsymbol{\beta}}$ would converge to the *true* values, which we indicate with $\boldsymbol{\beta}$ or β. We can use a Taylor expansion procedure similar to the one executed in Eqs. (C.5), (C.6), and (C.8) to, up to first order errors, express $\hat{\boldsymbol{\beta}}$ in terms of the true set of values $\boldsymbol{\beta}$. This leads to

$$\hat{\boldsymbol{\beta}} \approx \boldsymbol{\beta} + \left(\hat{\boldsymbol{J}}^T \Sigma^{-1} \hat{\boldsymbol{J}}\right)^{-1} \hat{\boldsymbol{J}}^T \Sigma^{-1} \boldsymbol{\epsilon}. \tag{C.9}$$

Here, $\boldsymbol{\epsilon}$ describes the *true* residuals $\epsilon_i = r_i - f(\beta, \mathcal{X}_i)$, *i.e.*, the residuals with respect to the true values of the fitting parameter. Furthermore, $\hat{\boldsymbol{J}}$ indicates the Jacobian evaluated with the calculated values $\hat{\boldsymbol{\beta}}$. To get Eq. (C.9), one has to assume that, for the final $\boldsymbol{\beta}^{(t+1)} = \hat{\boldsymbol{\beta}}$ of the optimization procedure, $\hat{\boldsymbol{\beta}} = \boldsymbol{\beta}^{(t)}$, $\hat{\boldsymbol{J}} = \boldsymbol{J}^{(t)}$ and $\hat{\boldsymbol{\epsilon}} = \boldsymbol{\epsilon}^{(t)}$ hold—*i.e.*, the system of equations has fully converged.

Strictly speaking, in the limit $n_C \to \infty$, we have $\hat{\boldsymbol{\beta}} = \boldsymbol{\beta}$ and $\hat{\boldsymbol{\beta}}$ does not have a distribution, since $\boldsymbol{\beta}$ contains constant values. For finite n_C, however, we can argue that $\hat{\boldsymbol{\beta}}$ *does* have a distribution, and we use the last term of Eq. (C.9) to estimate the uncertainty in the fitting parameters $\hat{\boldsymbol{\beta}}$. This estimation again involves the assumption that $\hat{\boldsymbol{J}}$ is a constant matrix, and does not depend on the fitting parameters. The correctness of this assumption depends on how close the model is to a linear regression. Under this assumption for $\hat{\boldsymbol{J}}$, we can make use of the fact that, for a general constant matrix $\boldsymbol{A}$, the variance of $\boldsymbol{A}\boldsymbol{\mathcal{Y}}$ is given by $\mathrm{Var}(\boldsymbol{A}\boldsymbol{\mathcal{Y}}) = \boldsymbol{A}\,\mathrm{Var}(\boldsymbol{\mathcal{Y}})\boldsymbol{A}^T$. Together with the assumption that $\mathrm{Var}(\boldsymbol{\epsilon})$ can be estimated by $\mathrm{Var}(\boldsymbol{\epsilon}) = \Sigma$, the covariance matrix of $\hat{\boldsymbol{\beta}}$ can be expressed as [4, 5]

$$\mathrm{Var}\left(\hat{\boldsymbol{\beta}}\right) \approx \left(\hat{\boldsymbol{J}}^T \Sigma^{-1} \hat{\boldsymbol{J}}\right)^{-1}. \tag{C.10}$$

The fact that we have a good idea of the uncertainties $\{\sigma_i\}_i$ of the observed $\{r_i\}_i$ means that we are able to estimate the quality of the obtained fit. For a good fit, we expect the uncertainty in the observed $\{r_i\}_i$ to match the uncertainty in the true residuals $\{\epsilon_i\}_i$ and the uncertainty in the residuals $\{\hat{\epsilon}_i\}_i$ of the fitting model. After all, in that case, the observed variance would be perfectly described by the theoretically expected variance. This means that the quality of the fit can be evaluated with the *reduced chi-squared* metric, which is defined as

$$\chi^2_\nu \equiv \frac{\mathcal{W}}{\nu} = \frac{1}{\nu} \sum_{i=1}^{n_C} \left(\frac{r_i - \hat{r}_i}{\sigma_i}\right)^2. \tag{C.11}$$

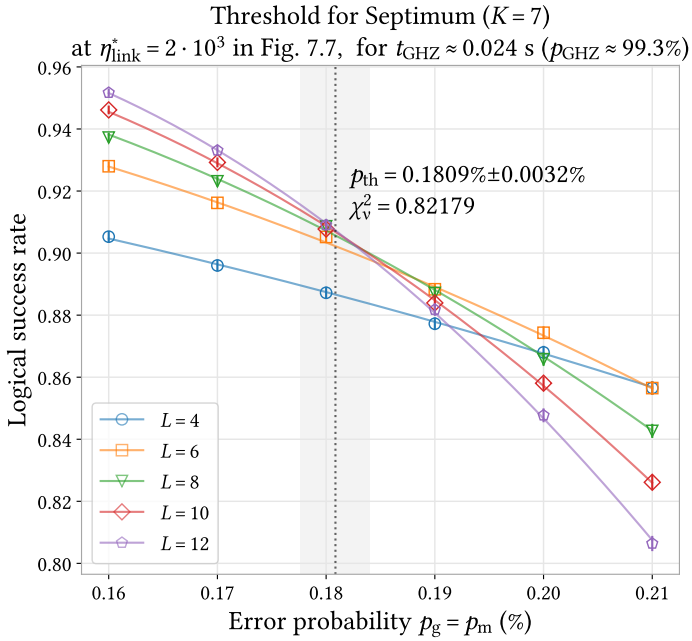


Figure C.1: Example of a threshold plot for the Septimum protocol, including a 95% confidence interval calculated with the method described in Sec. C.3.

Here, $\nu = n_C - n_P$ describes the number of degrees of freedom of the fitting model. A χ^2_ν value of approximately one corresponds to the variance in the observations matching the variance of the residuals. On the other hand, $\chi^2_\nu < 1$ indicates that the uncertainty of the model is too small to describe the data (indicating that the number of fitting parameters might be too large), whereas $\chi^2_\nu > 1$ indicates that the model does not describe the data well enough.

For the fits in this thesis, we are predominantly interested in the fitting parameter $\hat{p}_{\text{th}}$ that indicates the threshold value of a certain configuration. We find that the regression model of Eq. (C.2) works relatively well in a close range around the true threshold value (see, *e.g.*, Fig. C.1). If using data over a larger range of p values, we would typically find fits with $\chi^2_\nu > 1$. In those situations, to make sure the uncertainty in our threshold value estimation $\hat{p}_{\text{th}}$ would contain the true value, we assumed that the standard deviation in the $\{r_i\}_i$ values was not given by $\{\sigma_i\}_i$, but instead, was scaled up by the constant, positive factor χ_ν. This means we estimate $\text{Var}(\boldsymbol{\epsilon}) = \chi^2_\nu \Sigma$ and

$$\text{Var}\left(\hat{\boldsymbol{\beta}}\right) = \chi^2_\nu \left(\hat{\boldsymbol{J}}^T \Sigma^{-1} \hat{\boldsymbol{J}}\right)^{-1}, \quad \text{if } \chi^2_\nu > 1. \tag{C.12}$$

If one is in possession of $\text{Var}(\hat{\boldsymbol{\beta}})$, the standard deviation of the least-squares fitting parameter value $\hat{p}_{\text{th}}$ can be obtained from the square root of the corresponding diagonal element in $\text{Var}(\hat{\boldsymbol{\beta}})$. One can then calculate confidence intervals for the fitting parameters by identifying with what factor the standard deviations should be multiplied to ensure the requested

level of confidence. For this, we made use of the probability distribution f_{PD} of Student's t-distribution:

$$f_{\mathrm{PD}}(t_{\mathrm{ci}}) \equiv \Gamma'(\nu)\left(1+\frac{t_{\mathrm{ci}}^2}{\nu}\right)^{-(\nu+1)/2},$$
$$\Gamma'(\nu) \equiv \begin{cases} \dfrac{(\nu-1)(\nu-3)\cdots 5\cdot 3}{2\sqrt{\nu}(\nu-2)(\nu-4)\cdots 4\cdot 2}, & \text{if } \nu>1 \text{ even}, \\ \dfrac{(\nu-1)(\nu-3)\cdots 4\cdot 2}{\pi\sqrt{\nu}(\nu-2)\nu-4)\cdots 5\cdot 3}, & \text{if } \nu>1 \text{ odd}. \end{cases} \tag{C.13}$$

C

The t-distribution has mean 0 and variance 1 and converges to the normal distribution for $\nu\to\infty$. However, smaller degrees of freedom ν give rise to heavier tails that describe larger error ranges. Specifically, confidence intervals can be calculated by finding the t_{ci} factor that corresponds to the confidence interval of choice for the distribution of Eq. (C.13): for a confidence interval of I_{ci}, we find the corresponding t_{ci} via $\int_{-t_{\mathrm{ci}}}^{t_{\mathrm{ci}}} f_{\mathrm{PD}}(\tau)\mathrm{d}\tau = I_{\mathrm{ci}}$. In the plots in this thesis, we show 95% confidence intervals. For large ν and a confidence interval of $I_{\mathrm{ci}} = 95\%$, we have $t_{\mathrm{ci}} \approx 1.96$. Smaller values of ν lead to t_{ci} values that are slightly bigger. In Fig. C.1, we see an example of a threshold plot with a 95% confidence interval in the value found for the threshold fitting parameter.

References

[1] P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, S. J. van der Walt, M. Brett, J. Wilson, K. J. Millman, N. Mayorov, A. R. J. Nelson, E. Jones, R. Kern, E. Larson, C. J. Carey, İ. Polat, Y. Feng, E. W. Moore, J. VanderPlas, D. Laxalde, J. Perktold, R. Cimrman, I. Henriksen, E. A. Quintero, C. R. Harris, A. M. Archibald, A. H. Ribeiro, F. Pedregosa, P. van Mulbregt, and SciPy 1.0 Contributors, "SciPy: Fundamental algorithms for scientific computing in python." `https://scipy.org/`.

[2] M. Newville, T. Stensitzki, D. B. Allen, and A. Ingargiola, "LMFIT: Non-Linear Least-Square Minimization and Curve-Fitting for Python." Zenodo, Sept. 2014.

[3] C. Wang, J. Harrington, and J. Preskill, "Confinement-Higgs transition in a disordered gauge theory and the accuracy threshold for quantum memory," *Annals of Physics*, vol. 303, pp. 31–58, Jan. 2003.

[4] A. Ruckstuhl, *Introduction to Nonlinear Regression.* IDP Institut für Datenanalyse und Prozessdesign, ZHAW Zürcher Hochschule für Angewandte Wissenschaften, Oct. 2010.

[5] D. Constales, G. S. Yablonsky, D. R. D'hooge, J. W. Thybaut, and G. B. Marin, "Chapter 9 - Experimental Data Analysis: Data Processing and Regression," in *Advanced Data Analysis & Modelling in Chemical Engineering* (D. Constales, G. S. Yablonsky, D. R. D'hooge, J. W. Thybaut, and G. B. Marin, eds.), pp. 285–306, Amsterdam: Elsevier, Jan. 2017.

D

Best-performing GHZ generation protocols

At the end of Sec. 7.5.2, we discuss the Septimum protocol (depicted in Fig. 7.3): a GHZ generation protocol found with the dynamic program of Sec. 7.2.2 that gives rise to the highest thresholds for the simulation parameters used in several segments of Figs. 7.6, 7.7 and 7.8. In this appendix, we identify four additional GHZ generation protocols that perform the best in multiple segments of the figures in Sec. 7.5.2: the protocols *Sextimum*, *Decimum*, *Undecum*, and *Duodecum*. We depict the timed binary trees of these four protocols in Figs. D.1, D.2, and D.3, and provide more information on their performance in Table D.1. All four protocols are found with the dynamic program of Sec. 7.2.2 for the simulation model and parameters used in Sec. 7.5.2. The associated protocol recipes used with these protocols can be found in the repository of Ref. [1].

	K	Q	**Fig. 7.6**	**Fig. 7.7**	**Fig. 7.8**
Sextimum	6	3		$[4\cdot10^2, 5\cdot10^2]$	$4.7\cdot10^2$
Septimum	7	3	0.96	$[5\cdot10^2, 2\cdot10^3]$	$[6.3\cdot10^2, 1.1\cdot10^3]$
Decimum	10	3	$[0.9, 0.93]$	$[2\cdot10^3, 2\cdot10^5]$	$2\cdot10^3$
Undecum	11	3	$[0.85, 0.9]$		
Duodecum	12	4	$[0.78, 0.82]$		

Table D.1: Details about the GHZ generation protocols depicted in Figs. 7.3, D.1, D.2, and D.3. These protocols are found with the dynamic program of Sec. 7.2.2. The numbers K and Q denote the minimum number of Bell pairs and the maximum number of qubits per node required to generate the GHZ state, respectively. The last three columns of the table denote locations or ranges on the x-axes of Figs. 7.6, 7.7 and 7.8 in which these protocols are either the best-performing protocol or one of the best-performing protocols.

D

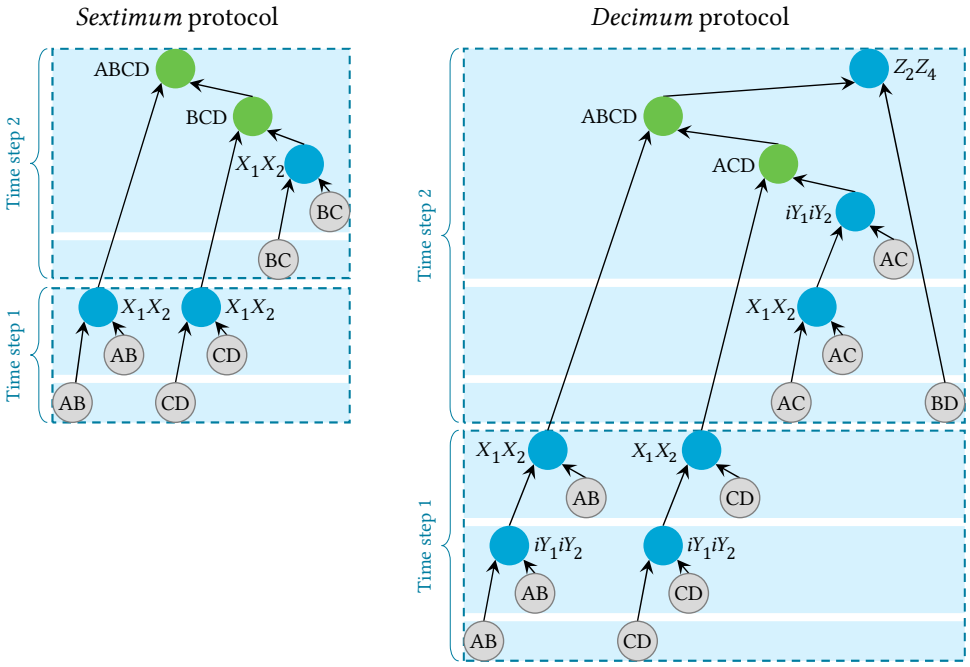


Figure D.1: Timed binary trees of two well-performing protocols in the Bell pair fidelity and the link efficiency sensitivity studies of Sec. 7.5.2. Clarification on the notation can be found in Fig. 7.3.

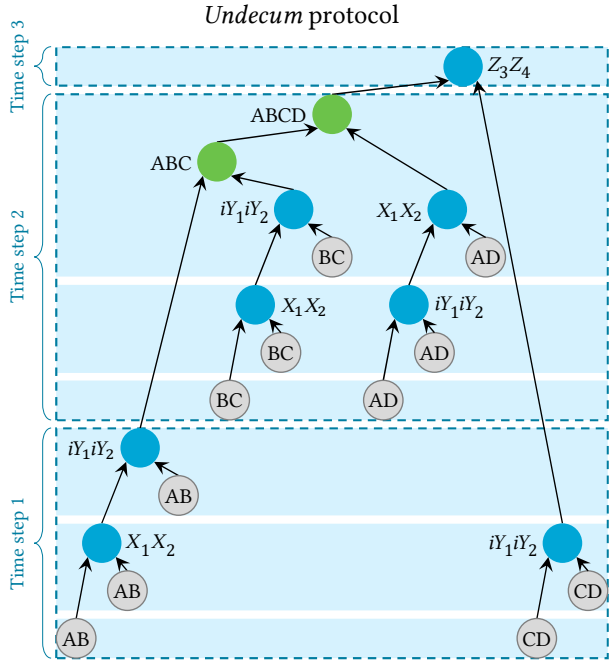


Figure D.2: Timed binary tree of a well-performing protocol in the Bell pair fidelity and the link efficiency sensitivity studies of Sec. 7.5.2. Clarification on the notation can be found in Fig. 7.3.

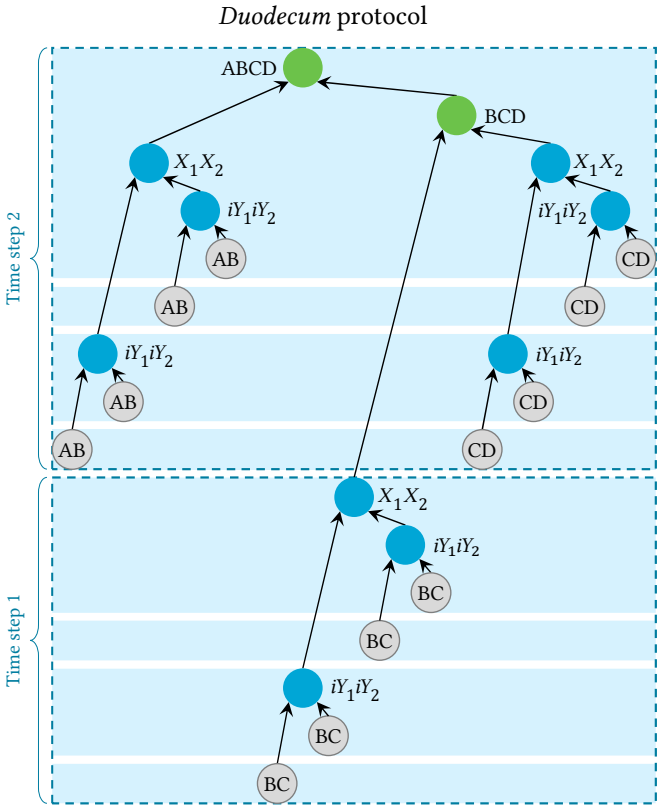


Figure D.3: Timed binary tree of a well-performing protocol in the Bell pair fidelity and the link efficiency sensitivity studies of Sec. 7.5.2. Clarification on the notation can be found in Fig. 7.3.

References

[1] S. de Bone, P. Möller, and D. Elkouss, "Data/software underlying the publication: Thresholds for the distributed surface code in the presence of memory decoherence." 4TU.ResearchData, https://doi.org/10.4121/708d4311-49b1-4ec2-b3cb-292d267df6be, Jan. 2024.

E

Quotient boundaries in unit cell complex and crystal embeddings

E.1 Quotient boundaries in unit cell complex

By the prescription in Sec. 8.2.1, the unit cell complex is described as a sequence of vector spaces

$$Q_3 \xrightarrow{\partial_3^{[\mathbf{r}]}} Q_2 \xrightarrow{\partial_2^{[\mathbf{r}]}} Q_1 \xrightarrow{\partial_1^{[\mathbf{r}]}} Q_0,$$

with each Q_i over the field $\mathbb{Z}_2$ and with quotient boundaries $\partial_i^{[\mathbf{r}]} : Q_i \mapsto Q_{i-1}$. Similar to the equivalence between $\overline{C}_i$ and C_{D-i}, we define $\overline{Q}_i \cong Q_{D-i}$. One may verify that the dual quotient boundaries $\overline{\partial}_i^{[\mathbf{r}]} : \overline{Q}_i \mapsto \overline{Q}_{i-1}$ are related to primal boundaries as

$$\overline{\partial}_i^{[\mathbf{r}]} = \left(\partial_{D+1-i}^{[\bar{\mathbf{r}}]}\right)^T. \tag{E.1}$$

Importantly, the translation vector $\mathbf{r}$ is also reversed to $\bar{\mathbf{r}}$. By dualizing boundary maps of unit cell complex directly, one may construct a representation of the dual unit cell without redefining it from the dual crystal. The zero map conditions $\partial_{i-1}\partial_i = 0$ take the form of

$$\sum_{\mathbf{p}} \partial_{i-1}^{[\mathbf{p}]} \partial_i^{[\mathbf{r}-\mathbf{p}]} = 0 \quad \forall \mathbf{r}. \tag{E.2}$$

The proof is given in Sec. E.2.

It is convenient to represent the underlying unit cell complex as a *labeled* graph, which is essentially a sparse representation of its boundaries as *arcs* and basis elements as *nodes*. (We use nomenclature *nodes* and *arcs* for such a graph, to make the distinction between vertices and edges of the chain complex.) Every basis element $(\mathbf{q}_i)_n \in Q_i$ is mapped to a node $q_{i,n}$, with two nodes $q_{i,n} \to_{[\mathbf{r}]} q_{i-1,m}$ connected by an $[\mathbf{r}]$-labelled arc if the mnth matrix

element of the quotient boundary $\partial_i^{[\mathbf{r}]}$ equals one. The maps $\partial_i^{[\mathbf{r}]}$ thus form the *biadjacency* matrices between the nodes of Q_i and Q_{i-1}. We note that this description, including its labeling, resembles the *vector method* for describing three-periodic networks as a quotient graph [1], except that the nodes of our quotient graph also represent higher-dimensional elements in a chain complex, such as edges, faces, and cells for a three-dimensional complex. Examples of the square, triangular, and cubic lattice are given in Fig. E.1.

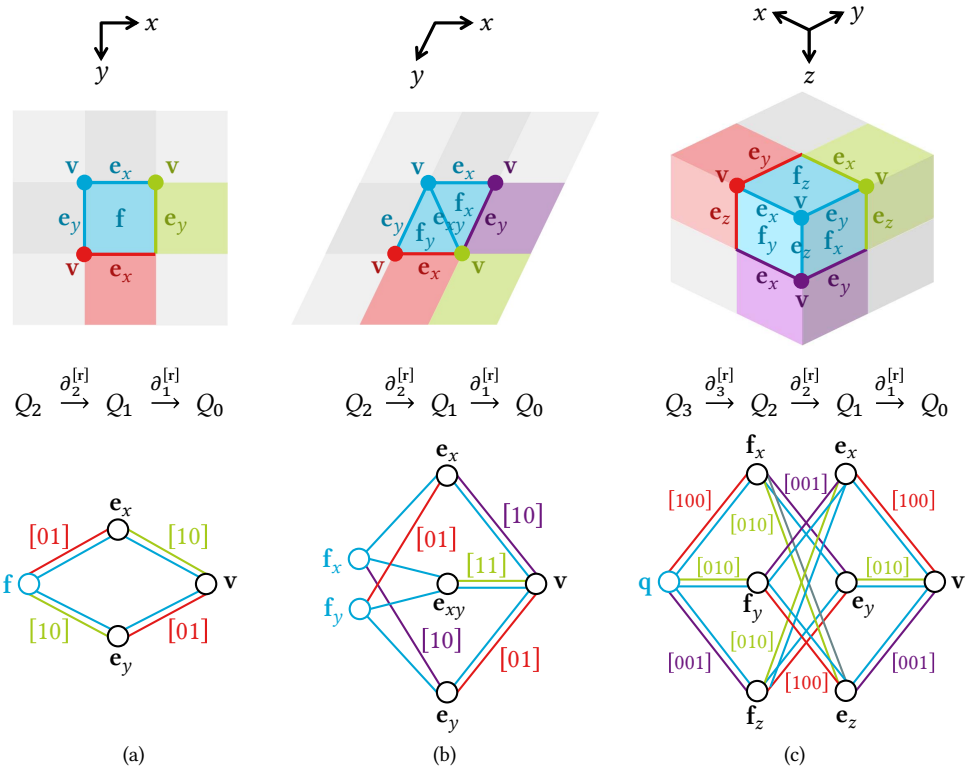


Figure E.1: Unit cell complexes as a labeled graph. Unlabeled edges correspond to a Miller index containing only zeros. (a) A square lattice. There is one face, two edges, and one vertex. The face $\mathbf{f}$ is connected twice to both $\mathbf{e}_x$ and $\mathbf{e}_y$ within ([00]) and outside ([01] and [10], respectively) the unit cell. Because the complex is self-dual, similar relationships hold for its vertex v. (b) A triangular lattice. This lattice can be created by splitting the faces of the square lattice: per unit cell, the triangular lattice has one extra face and one extra edge compared to the square lattice. From the asymmetry in the quotient boundary maps it is clear that this lattice is not self-dual. (c) A cubic lattice. Per unit cell, there is one cell $\mathbf{q}$; three faces $\mathbf{f}_x$, $\mathbf{f}_y$, and $\mathbf{f}_z$; three edges $\mathbf{e}_x$, $\mathbf{e}_y$, and $\mathbf{e}_z$; and one vertex $\mathbf{v}$.

E.2 Crystal embeddings

In Fig. E.2, we show an intuitive interpretation of how the vector spaces C_i of the full crystalline chain complex are constructed with the vector spaces Q_i of the unit cell complex of Sec. E.1. To construct the boundary maps ∂_i of the full crystal from the quotient boundary maps $\partial_i^{[\mathbf{r}]}$, we let $\partial_i^{(\mathbf{n},\mathbf{m})} : C_i \mapsto C_{i-1}$ be the boundary map that applies $\partial_i^{[\mathbf{n}-\mathbf{m}]}$ from cell $Q_i^{(m)}$

to cell $Q_{i-1}^{(n)}$ and is zero everywhere else:

$$\partial_i^{(\mathbf{n},\mathbf{m})} = \partial_i^{[\mathbf{n}-\mathbf{m}]} \otimes e_{nm}. \tag{E.3}$$

Here, $e_{nm} : L' \mapsto L'$ is a *matrix unit*, *i.e.*, a matrix with a one at indices n,m and zero elsewhere. Then the boundary maps of the embedding are given as a sum

$$\partial_i = \sum_{m,n} \partial_i^{(\mathbf{n},\mathbf{m})} = \sum_{m,n} \left(\partial_i^{[\mathbf{n}-\mathbf{m}]} \otimes e_{nm} \right). \tag{E.4}$$

Because most $\partial_i^{[\mathbf{n}-\mathbf{m}]}$ are zero, we can substitute $\mathbf{r} = \mathbf{n} - \mathbf{m}$ and sum $\mathbf{r}$ only over the non-zero maps $\partial_i^{[\mathbf{r}]}$, leading to

$$\partial_i = \sum_{m,r} \left(\partial_i^{[\mathbf{r}]} \otimes e_{m+r,m} \right) = \sum_r \partial_i^{[\mathbf{r}]} \otimes \left(\sum_m e_{m+r,m} \right). \tag{E.5}$$

In the second equality, the sum over $\mathbf{m}$ may be carried over to the right by distributivity of the tensor product over addition. This last term is a permutation matrix with a single one in each row and column; it represents a *translation* of a lattice point $\mathbf{m}$ to $\mathbf{m}+\mathbf{r}$. Denote this term as $L_\mathbf{r} \equiv \sum_m e_{m+r,m}$, such that the embedding is given as

$$\partial_i = \sum_\mathbf{r} \partial_i^{[\mathbf{r}]} \otimes L_\mathbf{r}. \tag{E.6}$$

Intuitively, the crystal boundary is formed by "gluing" the boundaries between the unit cells $Q_i^{(\mathbf{m})}$ and $Q_{i-1}^{(\mathbf{m}+\mathbf{r})}$ at lattice points $\mathbf{m}$ and $\mathbf{m}+\mathbf{r}$ according to the map $\partial_i^{[\mathbf{r}]}$, and repeating this process for every non-trivial quotient boundary map. The zero map conditions for quotient boundaries (Eq. (E.2)) follow trivially. Because matrix units multiply as $e_{ij}e_{kl} = \delta_{jk}e_{il}$, the multiplication of two permutation matrices

$$\begin{aligned} L_\mathbf{p} L_\mathbf{q} &= \sum_{m,n} e_{m+p,m} e_{n+q,n} = \sum_{m,n} \delta_{m,n+q} e_{m+p,n} \\ &= \sum_n e_{n+p+q,n} = L_{\mathbf{p}+\mathbf{q}} \end{aligned} \tag{E.7}$$

represents the sum of their translations. Combining this result with the embedded boundaries (Eq. (E.6)) directly, the composition of two maps equals

$$\partial_{i-1}\partial_i = \sum_{p,q} \partial_{i-1}^{[\mathbf{p}]} \partial_i^{[\mathbf{q}]} \otimes L_{\mathbf{p}+\mathbf{q}} = \sum_r \left(\sum_p \partial_{i-1}^{[\mathbf{p}]} \partial_i^{[\mathbf{r}-\mathbf{p}]} \right) \otimes L_\mathbf{r}, \tag{E.8}$$

where we have substituted $\mathbf{r} = \mathbf{p}+\mathbf{q}$ in the last equality. This map is the zero map if and only if the term in brackets is zero for all $\mathbf{r}$, which is exactly the result stated in Eq. (E.2).

The above recipe for a crystal embedding may be expressed as a composition of *direct products* between two graphs, given the following correspondences:

1. The matrix $\partial_i^{[\mathbf{r}]}$ is the biadjacency matrix of the subgraph $G[\partial_i^{[\mathbf{r}]}]$ of the unit cell complex induced by edges with label $[\mathbf{r}]$. This definition is consistent with the graph description given in Sec. E.1.

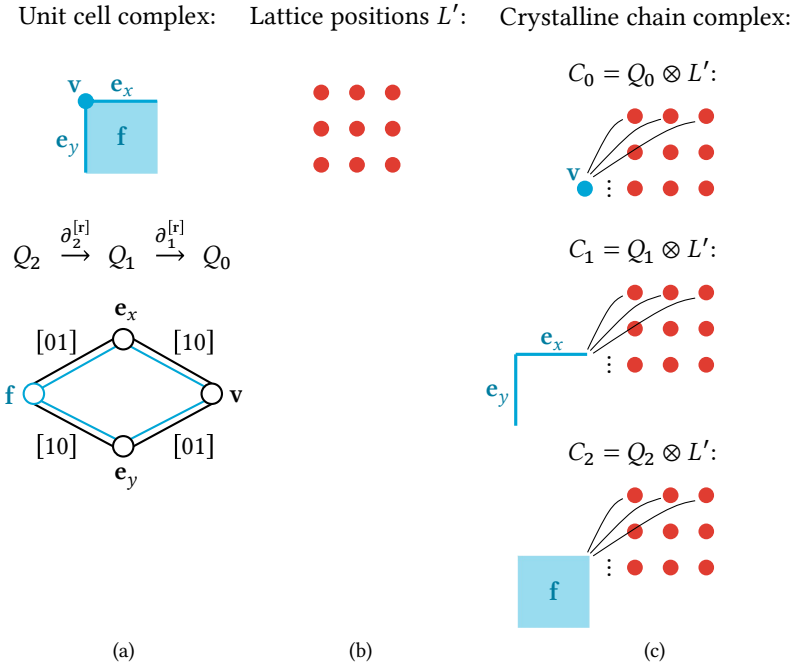


Figure E.2: Abstract interpretation of how the full crystalline chain complex is created out of the unit cell complex introduced in Sec. E.1. (a) We use the example of the unit cell of the square lattice—see Fig. E.1 for more details. (b) As described in Sec. 8.2.1 of the main text, we use an N_{cells}-dimensional vector space $L' \equiv \mathbb{Z}_2^{\oplus N_{\text{cells}}}$ with as the basis vectors the lattice positions of the lattice. (c) The vector spaces C_i of the full crystalline chain complex are realized with the *graph product* between L' and the vector spaces Q_i of the unit cell complex. The full boundary maps ∂_i are constructed from the quotient boundary maps $\partial_i^{[\mathbf{r}]}$ according to Eq. (E.5).

2. The matrix $L_{\mathbf{r}}$ is the biadjacency matrix of the subgraph of the lattice $G'[L_{\mathbf{r}}]$ induced by edges that translate each lattice point by $\mathbf{r}$. That is, each lattice point is represented by a node $\mathbf{m}$ and connected by an arc to the translated node $\mathbf{m}+\mathbf{r}$.

The tensor products $\partial_i^{[\mathbf{r}]} \otimes L_{\mathbf{r}}$ inside the embedding (Eq. (E.6)) are direct products of the corresponding edge-induced subgraphs $G[\partial_i^{[\mathbf{r}]}] \times G'[L_{\mathbf{r}}]$. The sum over labels $\mathbf{r}$, which adds together adjacency matrices of the products modulo 2, composes the edge sets of the corresponding graphs as a disjunctive union. In this way, the entire crystal complex may be constructed directly as a graph from a given unit cell and a lattice of arbitrary size.

References

[1] S. J. Chung, T. Hahn, and W. E. Klee, "Nomenclature and generation of three-periodic nets: The vector method," *Acta Crystallographica Section A: Foundations of Crystallography*, vol. 40, pp. 42–50, Jan. 1984.

F

Characterization of noisy channels

The general process in the simulations of Ch. 8 can be described as an N-qubit quantum circuit $\mathcal{U}$ composed of Clifford operations, ending in a projective Pauli basis measurement $\mathcal{P}^{\mathbf{m}}(\rho) = \Pi_{\mathbf{m}} \rho \Pi_{\mathbf{m}}$ with measurement projectors $\Pi_{\mathbf{m}} \equiv \prod_{m_k \in \mathbf{m}} \Pi_{m_k}$ for outcomes $\mathbf{m} = \{m_1, m_2, \ldots m_l\}$ of (part of) the evolved state. For the sake of completeness, we also assume that the circuit operates on an ancillary input system A with a stabilizer state $|\psi\rangle$. Such a circuit might represent an entanglement distillation circuit, operating on a mixed Bell pair ρ and an ancillary Bell pair to distill it with. Alternatively, it can represent the action of a measurement-based fault-tolerant channel, where ρ is the input code space, $|\psi\rangle$ is the state of all ancillary qubits in the channel, $\mathcal{U}$ represents the CZ gates of the cluster state and $\mathcal{P}^{\mathbf{m}}$ is the final Pauli-X measurement of every ancillary qubit. The output code space is then (up to normalization) given by $\mathcal{N}^{\mathbf{m}}(\rho)$.

We consider three sources of noise, depicted schematically in Fig. F.1. First of all, noisy ancillary input may not be a pure stabilizer state $|\psi\rangle$, but a mixture ρ_A of possibly non-stabilizer states. Second, the circuit $\mathcal{U}$ consists of imperfect operations, which we assume as ideal operations followed by a mixture of Pauli gates. Lastly, the projection $\mathcal{P}^{\mathbf{m}}$ may produce a "wrong" outcome $\widetilde{\mathbf{m}}$, which we model as a perfect operation $\mathcal{P}^{\mathbf{m}}$ followed by classical bit-flips on $\mathbf{m}$.

Arbitrary noisy ancillary input states ρ_A that differ from the noiseless input $|\psi\rangle$ cannot be simulated efficiently. If, on the other hand, ρ_A may somehow be decomposed as a Pauli channel $\mathcal{N}_P$ acting on $|\psi\rangle$, the Pauli operators may be pushed through the circuit in the same way as Pauli noise coming from imperfect gates. The key idea is to pre-process ρ_A by twirling with the stabilizers $P^{(s)} \in \mathcal{J}$. Here, $\mathcal{J}$ is the stabilizer group describing the state $|\psi\rangle$, with associated destabilizer group $\overline{\mathcal{J}}$. In Sec. 2.4.4 it is shown that this allows us to write the state $\mathrm{Tw}(\rho_A)$ after twirling state ρ_A as

$$\mathrm{Tw}(\rho_A) = \frac{1}{|\mathcal{J}|} \sum_{P^{(s)} \in \mathcal{J}} P^{(s)} \rho_A P^{(s)} = \sum_{P^{(k)} \in \overline{\mathcal{J}}} \lambda'_k P^{(k)} |\psi\rangle\langle\psi| P^{(k)} = \mathcal{N}_P(|\psi\rangle\langle\psi|). \tag{F.1}$$

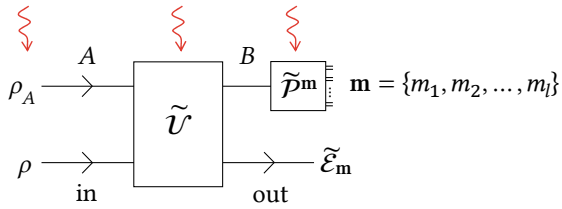


Figure F.1: A noisy channel that can be characterized efficiently, given that ρ_A is a convex combination of stabilizer states, $\widetilde{\mathcal{U}}$ is the ideal circuit $\mathcal{U}$ with Pauli noise, and $\widetilde{\mathcal{P}}^{\mathbf{m}}$ are ideal projectors followed by classical bit-flips on the outcome $\mathbf{m}$. Under suitable assumptions of the noise models used, the $\widetilde{\mathcal{N}}^{\mathbf{m}}$ operation may be expressed as a mixture of the ideal $\mathcal{N}^{\mathbf{m}}$ and Pauli operations.

Here, the prefactors λ'_k are given by $\lambda'_k = \lambda_{kk} = \langle\psi| P^{(k)} \rho_A P^{(k)} |\psi\rangle$. This shows how we can approximate a noisy ancillary input state ρ_A as a mixture of the ideal state $|\psi\rangle$ that is depolarized by a Pauli channel $\mathcal{N}_P$. In the same way, we can twirl non-Pauli noisy processes occurring during the application of the Clifford circuit $\mathcal{U}$ to a Pauli form. Operators $\{\tilde{P}^{(s)}\}_s$ can now be propagated through the circuit $\mathcal{U}$, forming another set of Pauli strings $\{P^{(s)} = \mathcal{U}\tilde{P}^{(s)}\mathcal{U}^\dagger\}_s$. Each Pauli operator $P^{(s)}$ that appears after $\mathcal{U}$ can each be split into a string $P_B^{(s)}$ appearing on the ancillary B system and a string $P_{\text{out}}^{(s)}$ appearing on the output system as $P^{(s)} \equiv P_B^{(s)} \otimes P_{\text{out}}^{(s)}$. The string $P_B^{(s)}$ commutes with some projectors $\Pi_{m_k} P_B^{(s)} = P_B^{(s)} \Pi_{m_k}$ (where $m_k \in \mathbf{m}$), but anti-commutes with others as $\Pi_{m_{k'}} P_B^{(s)} = P_B^{(s)} \Pi_{-m_{k'}}$. We summarize both cases as a commutation relation $\Pi_{\mathbf{m}} P_B^{(s)} = P_B^{(s)} \Pi_{\mathbf{m}\odot\mathbf{m}_s}$, where $\mathbf{m}_s \in \{+1, -1\}^{|\mathbf{m}|}$ is a string of errors that represents the bit-flips due to the individual commutation relations above and $\odot$ corresponds to the element-wise product.

One can now derive an expression for the noisy operation $\widetilde{\mathcal{N}}^{\mathbf{m}}$ as a mixture of ideal operations with the addition of probabilistic Pauli strings:

$$\widetilde{\mathcal{N}}^{\mathbf{m}}(\rho) = \sum_s p_s P_{\text{out}}^{(s)} \mathcal{N}^{\mathbf{m}\odot\mathbf{m}_s}(\rho) P_{\text{out}}^{(s)}. \tag{F.2}$$

The noisy channel $\widetilde{\mathcal{N}}^{\mathbf{m}}$ is a mixture of ideal operations $\mathcal{N}^{\mathbf{m}\odot\mathbf{m}_s}$ and Pauli noise, with the mixture arising due to classical bit-flips $\mathbf{m}_s$ that act on $\mathbf{m}$. Because we assumed each operation $\mathcal{N}^{\mathbf{m}}$ to be efficiently simulatable, the mixture can also be simulated efficiently.

On top of that, to account for faulty measurements, we assume that each projector Π_{m_k} has a fixed probability p_{m} of reporting the wrong outcome $\widetilde{m}_k \equiv -m_k$, such that the noisy measurement channel $\widetilde{\mathcal{P}}^{m_k}$ is given by a mixture

$$\widetilde{\mathcal{P}}^{m_k}(\rho) = (1 - p_{\text{m}}) \Pi_{m_k} \rho \Pi_{m_k} + p_{\text{m}} \Pi_{-m_k} \rho \Pi_{-m_k}. \tag{F.3}$$

For all measurement outcomes $\mathbf{m} = \{m_1, m_2, \ldots m_l\}$ this corresponds to the channel

$$\begin{aligned} \widetilde{\mathcal{P}}^{\mathbf{m}}(\rho) &= \sum_s p_f \Pi_{\mathbf{m}\odot\mathbf{m}_f} \rho \Pi_{\mathbf{m}\odot\mathbf{m}_f}, \\ p_f &\equiv (p_{\text{m}})^{h_f} (1 - p_{\text{m}})^{l - h_f}, \end{aligned} \tag{F.4}$$

where h_f corresponds to the number of -1 terms in $\mathbf{m}_f$—*i.e.*, the *Hamming weight* in case we interpret $\mathbf{m}_f$ as a binary bit string. The additional mixing of measurement outcomes

does not change the form of the noise channel as in Eq. (F.2) but introduces additional terms $\mathcal{N}^{\mathbf{m}\odot\mathbf{m}_s\odot\mathbf{m}_f}$ with prefactors that reflect the probability of applying a particular configuration of measurement errors:

$$\widetilde{\mathcal{N}}^{\mathbf{m}}(\rho) = \sum_{s,f} p_s p_f P_{\text{out}}^{(s)} \mathcal{N}^{\mathbf{m}\odot\mathbf{m}_s\odot\mathbf{m}_f}(\rho) P_{\text{out}}^{(s)}. \tag{F.5}$$

Acknowledgments

It is difficult to put into words how grateful I am for the opportunity to work on a Ph.D. thesis at QuTech and Delft University of Technology. Without a single doubt, I consider it the best period of my life. Besides meeting so many amazing and like-minded people it allowed me to develop myself on a personal and professional level. I want to thank everyone who contributed to this—the people mentioned below as well as the many people not mentioned. As someone who sometimes struggles to express himself, the text below began as an effort to ensure I did not appear ungrateful for all the support I have received over the past few years. However, as I was writing, I realized this provides an ideal opportunity to genuinely express my gratitude and let people know how much I appreciate them.

In the first place, I want to thank my supervisors and (co)promotors for the opportunity to work on this Ph.D. project and for supervising me. When I started the project, I had little knowledge of key concepts in this research field. Now, years later, I'm even more aware of my lack of knowledge. **David**, I have learned a lot from you in terms of presenting, writing efficiently and clearly, and effectively communicating our results. It has made me more capable as a scientist—even though, from that perspective, this thesis shows there is still room for improvement. My Ph.D. defense will be the first time we meet in person after more than two years. Luckily, it does not feel that way thanks to our many online meetings during the one hour of overlap on workdays on opposite sides of the world. I often smile when I recall a student asking whether you had already graded his exam, just as you finally had time to join us for a drink after work; or thinking about me having to correct and scan over 200 exams for your course and the massive pile of paper overheating every scanner and all-in-one printer in Delft and its near vicinity. (Of course, the second event is completely unrelated to the first.) **Stacey**, I am profoundly grateful for how you welcomed me to the group in Amsterdam, for providing me with well-appreciated advice, for organizing wonderful open mic events, and for thoroughly explaining papers on shallow circuits. I regret that the pandemic set back our collaboration and our results did not qualify for publication. **Stephanie**, I truly appreciate your valuable feedback and advice at several crucial moments and for incorporating me in the group meetings of your group in the first years of my Ph.D. project.

I want to express my gratitude to my committee members: to **Prof. Browne**, for traveling all the way to Delft for the defense and for creating the insightful lecture notes that sparked my journey in this field; to **Prof. DiCarlo** and **Prof. Terhal**, as full-time committee members of QuTech defenses; and to **Dr. Santra** and **Dr. Wong**, for taking the time to be part of the defense and providing useful feedback on the thesis. Many other highly learned, knowledgeable, and honorable people have contributed to the results presented in this thesis. **Tim T.**, ik heb veel gehad aan je waardevolle en diepgaande feedback tijdens ons project. **Conor**, your efforts in helping me understand nitrogen-vacancy centers and how to operate them are hugely appreciated, just as our pleasant collaboration. **Johannes**, my deepest gratitude for your valuable feedback, fantastic knowledge, and ideas, for organizing

and inviting me to the workshop last year, and for offering to adopt me in your group after David left. **Siddhant**, it was great to collaborate so closely. My apologies for the somewhat convoluted Python scripts you had to work with. **Fenglei**, I truly value your explanations of quantum optics and the direct entanglement schemes, your hard work on modeling said schemes, and the fact that you always seem to find new ways to cheer me up. You are a fascinating person and I appreciate all our encounters. **Doutzen**, elk evenement van het Quantum Software Consortium is erg goed georganiseerd en dat is voornamelijk dankzij jouw inspanningen. **Jenny**, **Esther**, **Shannon**, and **Sara**, thank you for taking care of many administrative questions and problems. Ik ben veel dank verschuldigd aan **Charlotte**, **Hawar**, **Elke**, **Martin**, **Stefan** en de rest van mijn huidige collega's voor hun begrip en steun tijdens het afronden van dit proefschrift. Of course, I also deeply value the (other) people mentioned in the acknowledgments at the end of the different chapters in the thesis.

One of my most cherished experiences at QuTech came from supervising students for their internships, or B.Sc. or M.Sc. end projects: **Runsheng**, whom I came to know as extremely pragmatic, smart, and friendly; **Paul**, who is incredibly hardworking, smart, dedicated, and a great programmer; **Sarah**, who picked up topics that were completely unfamiliar to her at the start of the project and wrote a well-received paper about her B.Sc. end project; **Yves**, who taught me more than I taught him and is a master at writing efficient software with as few symbols and lines as possible; and **Yorgos**, whom I did not really supervise in the end but still enjoyed listening to. **Yves**, ik beschouw je inmiddels als een vriend en vind dat je je meer moet realiseren dat iemand met jouw capaciteiten tot ontzettend veel in staat is. Veel dank voor de gezellige dagen en avonden in Delft, Leiden en Eindhoven.

Two people with a hugely important role during a Dutch Ph.D. defense ceremony are the *paranymphs*. **Julius**, bij jou kan ik om één of andere reden altijd helemaal mezelf zijn. Sinds ik je ken ben je één van mijn grootste voorbeelden. Ik neem binnenkort graag het eerste prototype van ons gitaarpedaal in ontvangst—hoewel je nu meer bezig lijkt te zijn met piano spelen. Veel dank, aan ook Lous, voor de gezellige middagen en avonden in Amstelveen de laatste tijd. Ondanks de onzekere tijden tijdens mijn bezoek in Boston kijk ik ook daar met veel plezier op terug. **Francisco**, ik kan altijd bij jou aankloppen voor hulp of een leuk gesprek. Er is niemand met wie ik vaker geluncht heb dan met jou, Guus, en Carlo—en dat was vaak het hoogtepunt van mijn dag. Je verbindt en bent met recht een sociale router! Diepe excuses voor mijn onhandige bijdragen aan je D&D-campagnes: ondanks de uitstekende uitleg en inspirerende verhaallijnen ben ik er nog steeds niet heel goed in. Ik kijk in ieder geval met heel veel genoegen uit naar je eerste boek!

There are many other people at QuTech to whom I am deeply appreciative. **Kenneth**, we spreken elkaar de laatste tijd helaas niet zo vaak meer, maar er zijn weinig mensen die zo'n grote invloed op mij gehad hebben als jij. Je was ontzettend aardig vanaf het eerste keer dat we elkaar ontmoetten en dat raakte me meteen. Je motiveerde me op een aantal vlakken om er gewoon voor te gaan en je advies betekent heel veel voor me. We zaten jarenlang naast elkaar in het kantoor en het was een eer om deel te nemen aan je willekeurige, maar altijd heel systematisch geformuleerde gedachtespinsels, muzikale intermezzo's en wiskundige curiositeiten. Het blijft eeuwig zonde dat, op twee niet nader te noemen artiesten na, onze muziekvoorkeuren zo drastisch uiteenlopen. **KC**, the dinners

you organized in the Choorstraat were fantastic. I vividly remember the great times we had after work on Fridays in pre-pandemic times and I vaguely remember my birthday during the pandemic. I still sometimes laugh about your jokes—mostly because they were so silly—and I cannot believe I still have not visited you after you left. **Carlo**, who still remembers the workshop on making cocktails? Just one of the many highlights during our long, successful stint as organizers of social activities for our research groups. You are definitely a yellow sheep yourself, and I could not have wished for a nicer colleague to share my Ph.D. experience with from beginning to end. I am confident we will someday find ourselves in the right situation to actually make music together. **Tim C.**, je bent een geboren academicus, professor en groepsleider: bescheiden, communicatief sterk, ontzettend vriendelijk en erg slim. Je bent altijd een soort supervisor voor me geweest—vanaf het moment dat ik bij David kwam solliciteren. Samen met Kenneth doorgrondden we het elusieve fenomeen destillatie in onze geheime journal club—of deden een poging daartoe in ieder geval! **Guus**, het was altijd heel erg gezellig met jou, en ik kwam met veel plezier langs in Rotterdam: of het nu was voor bordspelletjes of met oud en nieuw. Ik hoop dat Eva en jij het net zo gezellig hebben in Boston. **Luise**, you have been a great desk neighbor during the last part of my time at QuTech: a wonderful conversation partner during coffee and lunch breaks, and you never distracted me once with clapping polyrhythms while I was trying to work. I also truly appreciate your hospitality during game nights and post-dinner gatherings. **Stefan**, you were another much-appreciated office roommate. Thank you for providing me with some furniture when I arrived in Delft and you were about to leave. **Gayane**, you single-handedly realized strong entanglement between quantum information research in Delft and Amherst. It was a joy to get to know you a little bit. **Kian**, veel dank voor de leuke en interessante gesprekken de laatste tijd. Ik kan veel leren van jouw aanpak en hoe jij tegen bepaalde zaken aankijkt. **Hemant**, it is always great spending time with you. Even though I just realized I still have no idea where your office is exactly, it would mean a lot to me if we can somehow collaborate on our shared research interests in the future.

Furthermore, it has been such a pleasure to interact with the likes of **Álvaro**, **Bart**, **Bethany**, **Thomas**, **Ravi**, **Ravi**, **Anta**, **Hana**, **Tzula**, **Luca**, **Kaushik**, **Margrete**, **Scarlett**, **Soubhadra**, **Niv**, **Wesley**, **Benjamin**, **Nic**, **Christina**, **Jeroen**, **Eric**, **Aram**, **Héctor**, **Daniel**, **Cansın**, **Vicky**, **Ingmar**, **Mick**, **Julian**, **Matthijs**, **Emlyn**, **Leon**, **Matt**, **Matt**, **David**, **David**, **Aletta**, **Sjoerd**, **Mohammad**, **Francesco**, **Hany**, **Josh**, **Arian**, **Julia**, **Siddharth**, **Hans**, **Hans**, **Sophie**, **Wojciech**, **Mark**, **Mark**, **Lars**, **Antal**, **Gláucia**, **Axel**, **Jérémy**, **Victoria**, **Bas**, and many, many others during periods of my time at QuTech, as well as **Philip**, **Alvaro**, **Subha**, **Peter**, **Florian**, **Farrokh**, **Arjan**, **Joran**, **Ido**, **Joris**, **Sander**, **Freek**, **Yfke**, **Koen**, and many others during my visits at QuSoft. I had a great time organizing "Thuisje" with **Medina**, **David**, **Marcel**, **Kian**, and **Hridya** during the pandemic. **Ben**, thank you for so patiently helping me understand error correction and for sharing secret textbooks. **Jonas**, ontzettend veel dank voor de discussies, uitleg, ideeën, en leuke gesprekken door de jaren heen. **Filip**, you are one of the friendliest and most helpful people I have met. Thank you for looking at the DEJMPS protocol together. **Gao**, it was a delight having you in our group as a guest student. **André**, you were a much-appreciated group member of our group—lighting up our lunch breaks and social activities. I am sorry I never really got to know you.

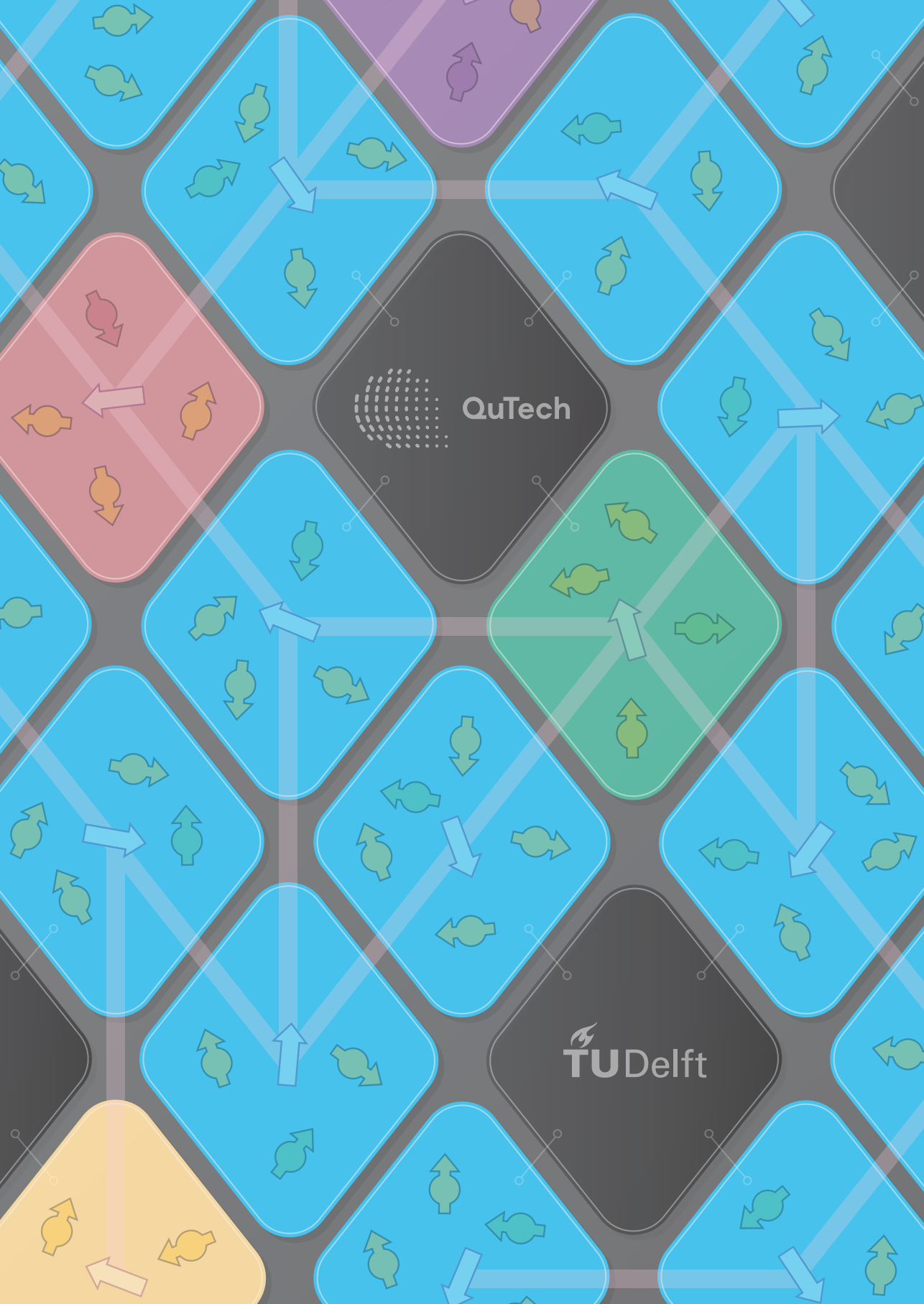
The importance of a good balance between work and free time should not be overrated and I should also acknowledge the people who made this possible during my Ph.D. project. I am very thankful for the movie club that I am part of: it has been one endless stream of great movies so far—really one after the other. **Giulia**, you should know that I never once misspelled your name: the "i" after the "G" makes all the sense in the world! **Janice**, voor twee mensen van het platteland zijn we allebei al best ver gekomen. Veel dank voor de uitnodigingen en gastvrijheid de laatste tijd. **Sam**, I greatly appreciate your support and encouraging words during the final stages of this thesis. I also owe a great deal to my friends and (former) teammates in **T10** and **T6** of futsal association **FC Tutor**, and the people I made music with over the last few years: **Manolo**, **Richella**, **Heleen**, **Niek**, **Andrea**, **Anne**, **Anish**, **Sanne**, and others. Veel dank aan mijn mede-redactieleden bij het **Nederlands Tijdschrift voor Natuurkunde**. Daarnaast waardeer ik natuurlijk mijn studievrienden uit Enschede: **Fenna**, **Viola**, **Jeroen**, **Joris**, **Sander**, **Jorrit**, en de rest. **Jelte**, **Maaike** en **Caspar**, heel veel dank voor de leuke middagen, avonden, verjaardagen en Bevrijdingsdagen, de herinnering aan de Batavierenrace, en mijn leuke bezoek aan Philippine. **Roeland**, ik kijk met ontzettend veel plezier terug op onze tijd als huisgenoten. **Gerben**, ik ben ontzettend dankbaar dat we nog steeds af en toe bijpraten. Vanaf het moment dat ik je leerde kennen tijdens onze introductieweek ben ik enorm op je gesteld. Ik herinner me de gouden tijd als hoofd- en eindredactieduo van de Focus, op stand dineren in Amsterdam tijdens één van de vele rondzwervingen langs Nederlandse musea, mijn onverwachte logeerbezoek na het kwijtraken van mijn huissleutels, en vele andere gewone momenten die om die reden misschien wel juist heel bijzonder waren: ik kan altijd op je rekenen. **Emma met een e**, waar moet ik ook maar beginnen. Ontzettend bedankt voor diep geluk, leven in het moment, onwerkelijke momenten waarop de tijd leek stil te staan, en het zien van de wereld door jouw ogen.

Finally, a few words to the people who shaped me more than anyone else. **Mariska**, ik ben ontzettend trots om jou als kleine zus te hebben. Ik vind dat je erg goed bent in wat je doet: je geeft nooit op, je bent erg creatief en ook heel loyaal. Daarnaast maak je mijn leven leuker en mis ik je dynamische aanwezigheid als je er niet bent. Weet dat je me altijd kan bereiken als dat nodig is—of juist niet nodig is. **Peter**, wat kan jij niet! Ik heb veel waardering voor je veelzijdigheid en je vriendelijke persoonlijkheid. **Papa**, **mama**, **Belle**, **Tygo**, **Iko**, **Mareille**, **Rosita**, **Kaatje**, **Isaac** en **alle andere dieren**, jullie zijn een erg goed team waarbinnen iedereen zijn of haar eigen taken heeft: het gras maaien, het nieuws voorlezen, luchtballonnen verjagen, de tuin omspitten, schuurtjes bouwen, eten voorschotelen, de pan uitlikken, het huis bewaken, het hek dichtdoen, eieren leggen, plantjes opeten, balken (op maat zagen), op de bank hangen, schuurtjes slopen, luid snurken, het fietspad bewaken, *etc.* Ik geniet erg van de momenten waarop we met z'n allen samen zijn. Het is onmogelijk om alles wat jullie voor mij gedaan hebben ooit terug te geven, maar ik beloof hier altijd mijn best voor te zullen (blijven) doen.

Curriculum Vitæ

Sébastian Wicher de Bone

2023 - present	Senior Technical Specialist Quantum Technology NLD Ministry of Defence Maasland, The Netherlands
2018 - 2024	Ph.D. in Quantum Information Delft University of Technology Delft, The Netherlands *Thesis:* "Reducing Hardware Requirements for Fault-Tolerant Distributed Quantum Computers" *Copromotor:* Dr. D. Elkouss Coronas *Promotor:* Prof. dr. S. D. C. Wehner
2015 - 2018	Project Assistant Royal Netherlands Society of Engineers The Hague, The Netherlands
2012 - 2017	M.Sc. in Applied Physics University of Twente Enschede, The Netherlands *Thesis:* "Theoretical Model for Superconducting *SIsFS* Josephson Devices" *Supervisor:* Dr. A. A. Golubov *Internship:* "Density of States and Magnetization in Long-Range Superconductor-Ferromagnetic Interfaces in the Clean Limit" *Supervisor:* Prof. dr. M. Eschrig (Royal Holloway, University of London)
2009 - 2012	B.Sc. in Applied Physics (*with honors* and *cum laude*) University of Twente Enschede, The Netherlands *Thesis:* "Growth of $PbTiO_3$ and $BiFeO_3$ Thin Films on Flat and Three-Dimensional $SrRuO_3$ Surface Morphologies" *Supervisor:* Dr. B. Smith *Supervisor:* Prof. dr. ir. G. Koster
1990/12/21	Born in Meppel, The Netherlands


QuTech
TU Delft