Graphical CSS Code Transformation Using ZX Calculus

by

Sarah Meng Li

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Mathematics
in
Combinatorics and Optimization (Quantum Information)

Waterloo, Ontario, Canada, 2023

Author's Declaration

This thesis consists of material all of which I authored or co-authored: see Statement of Contributions included in the thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Statement of Contributions

- Chapter 1 reviews the background and the proceeding results for our research.
- Chapter 2 introduces the notions and techniques used to graphically transform different CSS codes using the ZX calculus. This is based on Kissinger's ZX normal form proposed in [57].
- Chapter 3 generalizes the ZX normal form for CSS stabilizer codes to CSS subsystem codes. It also provides bidirectional rewrite rules for any CSS encoder. This technique was proposed by Jiaxin Huang and myself.
- Chapter 4 provides explicit graphical derivations for morphing the Steane and the quantum Reed-Muller codes. This scheme was initially proposed by Michael Vasmer [92]. The graphical interpretation was proposed by Lia Yeh. The edge cases were proposed by myself.
- Chapter 5 focuses on the switching protocol between the Steane code and the quantum Reed-Muller code. Through the ZX calculus, we provide a graphical interpretation of this protocol as gauge-fixing the $[\![15, 1, 3, 3]\!]$ subsystem code, followed by syndrome-determined recovery operations. The idea of using a different fault-tolerant protocol (i.e., code-switching) rather than the magic state distillation was originally brought up by my supervisor, Michele Mosca. Aleks Kissinger encouraged us to explore the connections between code-switching and subsystem code gauge-fixing. To answer this question, I visualized both protocols using the ZX diagrams and showed that they are different interpretations of the same measurement-based procedure.
- Chapter 6 concludes our work and proposes future directions. This is based on the discussions among all co-authors.

The research deliverable in this thesis is the joint work of all co-authors.

Abstract

In this work, we present a generic approach to transform CSS codes by building upon their equivalence to phase-free ZX diagrams. Using the ZX calculus, we demonstrate diagrammatic transformations between encoding maps associated with different codes. As a motivating example, we give explicit transformations between the Steane code and the quantum Reed-Muller code, since by switching between these two codes, one can obtain a fault-tolerant universal gate set. To this end, we propose a bidirectional rewrite rule to find a (not necessarily transversal) physical implementation for any logical ZX diagram in any CSS code.

We then focus on two code transformation techniques: *code morphing*, a procedure that transforms a code while retaining its fault-tolerant gates, and *gauge fixing*, where complimentary codes can be obtained from a common subsystem code (e.g., the Steane and the quantum Reed-Muller codes from the $[\![15, 1, 3, 3]\!]$ code). We provide explicit graphical derivations for these techniques and show how ZX and graphical encoder maps relate several equivalent perspectives on these code-transforming operations.

Acknowledgements

First and foremost, I wish to thank my supervisor Michele Mosca for his guidance, insights, and unwavering support. When I joined Waterloo in 2021, I was a bit burned out from a very intensive undergrad and didn't know how to research as a graduate student. He never rushed me and allowed me to explore at my own pace. I could not think of a better time for me to stand on my own feet and navigate a research problem independently. It is a privilege to learn more from Michele and work with him. He trained me to present a research problem efficiently and showed me the importance of taking initiative. He challenged me with interesting problems and trusted me to conquer them. He taught me to research with a pragmatic mindset and nudged me to reason thoroughly. He encouraged me to see the bigger world and inspired me to stay grounded. Dear Michele, thank you for having faith in me. I am, and will continuously be driven by it.

I would like to express my deepest gratitude to my thesis readers, Professors David Gosset and William Sloftra. Thank you for taking the time to read my thesis and sharing the insightful feedback.

This thesis would not have existed without my collaboration with Jiaxin Huang, Aleks Kissinger, Michele Mosca, Michael Vasmer, and Lia Yeh. Dear Aleks, Michele, and Michael, thank you for your thought-provoking notes and numerous pointers throughout the project. I have learned so much from working with you. Deal Lia and Jacky, thank you for all the stimulating discussions and the wonderful collaborations.

My time at IQC and the University of Waterloo would not have been so enjoyable without the dedicated staff members. Thank you Chin for working so hard to take care of all the clerical work and for the uplifting chats. Thank you Melissa for helping me navigate the degree requirements.

To my amazing friends, far and near. Thank you for rooting for me. Without your company, I would not have had such a wonderful two years. Shout out to Xiaoning Bian, Zehui Chen, Arianne Meijer - van de Griend, Minyoung Kim, Andy Liu, Yifan Wang, and Yvonne Wu.

Finally, I wish to thank my parents for being my biggest fans who have never ceased to cheerlead for me. I am very proud to be your daughter, and I love you to the moon and back.

Table of Contents

List of Figures

Preface

Under the guidance of my supervisor Michele Mosca, my Master's research focuses on the algebraic and graphical methods for fault-tolerant quantum compilation. This encompasses three key ingredients. Firstly, characterize an ideal quantum system through the lens of algebra and the ZX calculus. Next, define and optimize a cost function for a logical quantum circuit. Based on the hardware and algorithm specifics, these insights help us tailor strategies to optimize resources in quantum computation.

A few months into the program, we started to realize that for me to really understand various aspects of quantum compiling, there is no alternative to solving different compilation-related problems and acquiring essential skills. I am very fortunate to learn from experts in the field. Over the past two years, our collaborations have led to novel and interesting results.

[1] Meijer-van de Griend, A., & **Li, S. M**. (2022). Dynamic Qubit Allocation and Routing for Constrained Topologies by CNOT Circuit Re-Synthesis. In Quantum Physics and Logic. Electronic Proceedings in Theoretical Computer Science. arXiv preprint arXiv:2205.00724.

[2] Amy, M., Glaudell, A. N., **Li, S. M.**, & Ross, N. J. (2023). Improved Synthesis of Toffoli-Hadamard Circuits. arXiv preprint arXiv:2305.11305.

[3] Huang, J., **Li, S. M.**, Yeh, L., Kissinger, A., Mosca, M., & Vasmer, M. (2023). Graphical CSS Code Transformation Using ZX Calculus. arXiv preprint arXiv:2307.02437.

[4] Kim, D., Kim, M., **Li, S. M.**, & Mosca, M. (2023). Improving the Fidelity of CNOT Circuits on NISQ Hardware (To appear in arXiv).

[5] **Li, S. M.**, Mosca, M., Ross, N. J., & Zhao, Y. (2024). Generators and Relations for N-Qutrit Clifford Operators (Manuscript in preparation).

[6] Huang, J., Kissinger, A., **Li, S. M.**, Mosca, M., Vasmer, M., & Yeh, L. (2024). A Unified Framework for Stabilizer Code Encoders Using ZX Calculus (Manuscript in preparation).

In the meantime, the training at Waterloo exposed me to fault-tolerant protocols based on different quantum error-correcting schemes. Lectures and seminars deepened my understanding of prevalent quantum algorithms and the fundamental complexity theory. Additionally, I learned algebraic and graph-theoretic techniques for quantum circuit optimization. My internship with PsiQuantum and numerous research discussions also helped me put everything in perspective. I started to appreciate quantum compilation beyond an isolated theoretical problem.

In this thesis, we focus on one of our published works in 2023, *Graphical CSS Code Transformation Using ZX Calculus*. This project is very special to me as it highlights my first adventure into the ZX calculus and quantum error-correcting codes. This is also the first project where I initiated the core ideas and took the lead throughout the collaboration.

Chapter 1

Introduction

Quantum computation has demonstrated its potential in speeding up large-scale computational tasks [4, 100] and revolutionizing multidisciplinary fields such as drug discovery [21], climate prediction [89], chemistry simulation [69], and the quantum internet [38]. However, in a quantum system, qubits are sensitive to interference and information becomes degraded [73]. To this end, quantum error correction [82, 84] and fault tolerance [47, 58] have been developed to achieve large-scale universal quantum computation [48].

Stabilizer theory [46] is a mathematical framework to describe and analyze properties of quantum error-correcting codes (QECC). It is based on the concept of stabilizer groups, which are groups of Pauli operators whose joint $+1$ eigenspace corresponds to the code space. Stabilizer codes are a specific type of QECC whose encoder can be efficiently simulated [1, 45]. As a family of stabilizer codes, Calderbank-Shor-Steane (CSS) codes permit simple code constructions from classical codes [17, 18, 84, 86].

As a language for rigorous diagrammatic reasoning of quantum computation, the ZX calculus consists of ZX diagrams and a set of rewrite rules [27, 91]. It has been used to relate stabilizer theory to graphical normal forms: notably, efficient axiomatization of the stabilizer fragments for qubits [5, 51, 72], qutrits [90, 95], and prime-dimensional qudits [12]. This has enabled various applications, such as measurement-based quantum computation [72, 83], quantum circuit optimization [30, 43] and verification [68], as well as classical simulation [25, 57]. Beyond these, ZX-calculus has been applied to verify QECC [36, 39], represent Clifford encoders [54], as well as study various QECC such as tripartite coherent parity check codes [22, 24] and surface codes [40, 41, 42, 81]. Specific to CSS codes, ZX-calculus has been used to visualize their encoders [55], code maps and code surgeries [33], their correspondence to affine Lagrangian relations [31], and their constructions in high-dimensional quantum systems [32].

In this thesis, we seek to answer some overarching questions about QECC constructions and fault-tolerant implementations. We focus on CSS codes and leverage the direct correspondence between phase-free ZX diagrams and CSS code encoders [55]. Given an arbitrary CSS code, based on its normal form, we propose a bidirectional rewrite rule to find a (not necessarily transversal) physical implementation for any logical ZX diagram. Furthermore, we demonstrate diagrammatic transformations between encoding maps associated with different codes. Here, we focus on two code transformation techniques: *code*

morphing, a procedure that transforms a code while retaining its fault-tolerant gates [92], and *gauge fixing*, where complimentary codes (such as the Steane and the quantum Reed-Muller codes) can be obtained from a common subsystem code [3, 74, 77, 94]. We provide explicit graphical derivations for these techniques and show how ZX and graphical encoder maps relate several equivalent perspectives on these code transforming operations.

Chapter 2

Preliminaries

We start with some definitions. The Pauli matrices are 2×2 unitary operators acting on a single qubit. Let i be the imaginary unit.

$$I = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \quad X = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}, \quad Z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}, \quad Y = iXZ = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix}.$$

Let $\mathcal{P}_1$ be the single-qubit Pauli group, $\mathcal{P}_1 = \langle i, X, Z \rangle$, $I, Y \in \mathcal{P}_1$.

2.1 The Stabilizer Formalism and CSS Codes

Definition 1. *Let $U \in \mathcal{U}(2)$. In a system over n qubits, $1 \leq i \leq n$,*

$$U_i = I \otimes \ldots \otimes I \otimes U \otimes I \otimes \ldots \otimes I$$

denotes U acting on the i-th qubit, and identity on all other qubits.

Let $\mathcal{P}_n$ be the n-qubit Pauli group. It consists of all tensor products of single-qubit Pauli operators.

$$\mathcal{P}_n = \langle i, X_1, Z_1, \ldots, X_n, Z_n \rangle.$$

The stabilizer formalism is a mathematical framework to describe and analyze the properties of certain QECC, called stabilizer codes [46, 47]. Consider n qubits and let $m \leq n$. A stabilizer group $\mathcal{S} = \langle S_1, \ldots, S_m \rangle$ is an Abelian subgroup of $\mathcal{P}_n$ that does not contain $-I$. The codespace of the corresponding stabilizer code, $\mathcal{C}$, is the joint $+1$ eigenspace of $\mathcal{S}$, i.e.,

$$\mathcal{C} = \{|\psi\rangle \in \mathbb{C}^{2^n};\ S|\psi\rangle = |\psi\rangle, \forall S \in \mathcal{S}\}.$$

The number of encoded qubits in a stabilizer code is $k = n - m$, where m is the number of independent stabilizer generators [46]. Moreover, we can define the *centralizer* of $\mathcal{S}$ as

$$\mathcal{N}(\mathcal{S}) = \{U \in \mathcal{P}_n;\ [U, S] = 0, \forall S \in \ \mathcal{S}\}.$$

One can check that $\mathcal{N}(\mathcal{S})$ is a subgroup of $\mathcal{P}_n$ and $\mathcal{S} \subset \mathcal{N}(\mathcal{S})$. We remark that the notions of normalizer and centralizer coincide for any stabilizer group. In what follows, we will use them interchangeably. As we will see later, $\mathcal{N}(\mathcal{S})$ provides an algebraic structure for the subsystem codes. The code distance, d, of a stabilizer code is the minimal weight of operators in $\mathcal{N}(\mathcal{S})/\langle iI \rangle$ that is not in $\mathcal{S}$. We summarize the properties of a stabilizer code with the shorthand $[\![n, k, d]\!]$.

Finally, we introduce some notation for subsets of n-qubit Pauli operators, which will prove useful for defining CSS codes.

Definition 2. *Let M be an $m \times n$ binary matrix and $P \in \mathcal{P}_1/\langle iI \rangle$. In the stabilizer formalism, M is called the* stabilizer matrix, *and M^P defines m P-type stabilizer generators.*

$$M^P := \left\{ \bigotimes_{j=1}^{n} P^{[M]_{ij}};\ 1 \leq i \leq m \right\}.$$

CSS codes are QECC whose stabilizers are defined by two orthogonal binary matrices G and H [17, 84]:

$$\mathcal{S} = \langle G^X, H^Z \rangle, \quad GH^\intercal = \mathbf{0},$$

$H^\intercal$ is the transpose of H. This means that the stabilizer generators of a CSS code can be divided into two types: X-type and Z-type. For example, the $[\![7, 1, 3]\!]$ Steane code [84] in Figure 2.1a is specified by

$$G = H = \begin{bmatrix} 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{bmatrix}_{3\times 7}. \tag{2.1}$$

Accordingly, the X-type and Z-type stabilizers are defined as

$$\begin{aligned} S_1^X &= X_1X_3X_5X_7, & S_2^X &= X_2X_3X_6X_7, & S_3^X &= X_4X_5X_6X_7 \\ S_1^Z &= Z_1Z_3Z_5Z_7, & S_2^Z &= Z_2Z_3Z_6Z_7, & S_3^Z &= Z_4Z_5Z_6Z_7. \end{aligned}$$

The logical operators $\overline{X}$ and $\overline{Z}$ are defined as

$$\overline{X} = X_1X_4X_5 \qquad \text{and} \qquad \overline{Z} = Z_1Z_4Z_5. \tag{2.2}$$

In Section 2.2, we define CSS subsystem codes. In Section 2.3, we define several CSS codes that will be used in subsequent sections. In Section 2.4, we introduce the basics of the ZX calculus and the phase-free ZX normal forms.

2.2 The CSS Subsystem Codes

Subsystem codes [60, 75] are QECC where some of the logical qubits are not used for information storage and processing. These logical qubits are called gauge qubits. By

fixing gauge qubits to some specific states, the same subsystem code may exhibit different properties, for instance, having different sets of transversal gates [10, 62, 63, 74, 99]. This provides a tool to circumvent restrictions on transversal gates such as the Eastin-Knill theorem [37].

Based on the construction proposed in [75], we describe a subsystem code using the stabilizer formalism.

Definition 3. *Given a stabilizer group $\mathcal{S}$, a gauge group $\mathcal{G}$ is a normal subgroup of $\mathcal{N}(\mathcal{S})$, such that $\mathcal{S} \subset \mathcal{G}$ and that $\mathcal{G}/\mathcal{S}$ contains anticommuting Pauli pairs. In other words, one can write*

$$\mathcal{S} = \langle S_1, \ldots, S_m \rangle, \quad \mathcal{G} = \langle S_1, \ldots, S_m, g_1^X, g_1^Z, \ldots, g_r^X, g_r^Z \rangle, \quad 1 \leq m + r \leq n.$$

$(\mathcal{S}, \mathcal{G})$ defines an $[\![n, k, r, d]\!]$ subsystem code where $n = m + k + r$. The logical operators are elements of the quotient group $\mathcal{L} = \mathcal{N}(\mathcal{S})/\mathcal{G}$.

Under this construction, n physical qubits are used to encode k logical qubits with r gauge qubits. Alternatively, we can think of the gauge group $\mathcal{G}$ as partitioning the code space $\mathcal{C}$ into two subsystems: $\mathcal{C} = \mathcal{A} \otimes \mathcal{B}$. Logical information is encoded in $\mathcal{A}$ and $\mathcal{L}$ serves as the group of logical operations. Gauge operators from $\mathcal{G}$ act trivially on subsystem $\mathcal{A}$, while operators from $\mathcal{L}$ act trivially on subsystem $\mathcal{B}$. Therefore, two states $\rho^{\mathcal{A}} \otimes \rho^{\mathcal{B}}$ and $\rho'^{\mathcal{A}} \otimes \rho'^{\mathcal{B}}$ are considered equivalent if $\rho^{\mathcal{A}} = \rho'^{\mathcal{A}}$, regardless of the states $\rho^{\mathcal{B}}$ and $\rho'^{\mathcal{B}}$. When $r = 0$, $\mathcal{G} = \mathcal{S}$. In that case, an $[\![n, k, 0, d]\!]$ subsystem code is essentially an $[\![n, k, d]\!]$ stabilizer code.

CSS subsystem codes are subsystem codes whose stabilizer generators can be divided into X-type and Z-type operators. In what follows, we provide an example to illustrate their construction.

2.3 Some Interesting CSS Codes

We start by defining the stabilizer groups for the $[\![7, 1, 3]\!]$ Steane code, the $[\![15, 1, 3]\!]$ extended Steane code [3], and the $[\![15, 1, 3]\!]$ quantum Reed-Muller code [59]. They are derived from the family of $[\![2^m - 1, 1, 3]\!]$ quantum Reed-Muller codes, with a recursive construction of stabilizer matrices [87]. The Steane code has transversal logical Clifford operators, and the quantum Reed-Muller code has a transversal logical T gate. Together these operators form a universal set of fault-tolerant gates. In Chapter 5, the relations between these codes are studied from a diagrammatic perspective.

For brevity, their corresponding stabilizer groups are denoted as $\mathcal{S}_{steane}$, $\mathcal{S}_{ex}$, and $\mathcal{S}_{qrm}$. As per Definition 2, consider three stabilizer matrices F, H, and J. Note that G is defined in Equation (2.1). $\mathbf{0}$ and $\mathbf{1}$ denote blocks of 0s' and 1s' respectively. Their dimensions can be inferred from the context.

$$F = \begin{bmatrix} G & 0 & G \\ \mathbf{0} & 1 & \mathbf{1} \end{bmatrix}_{4\times 15}, \quad H = \begin{bmatrix} G & \mathbf{0} \end{bmatrix}_{3\times 15},$$
$$J = \begin{bmatrix} 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \end{bmatrix}_{3\times 15}.$$

Then, the stabilizer groups are defined as

$$\mathcal{S}_{steane} = \langle G^X, G^Z \rangle, \quad \mathcal{S}_{ex} = \langle F^X, F^Z, H^X, H^Z \rangle, \quad \mathcal{S}_{qrm} = \langle F^X, F^Z, H^Z, J^Z \rangle. \tag{2.3}$$

Geometrically, one can define $\mathcal{S}_{steane}$ and $\mathcal{S}_{qrm}$ with the aid of Figure 2.1. In Figure 2.1a, the Steane code is visualized on a 2D lattice. Since the Steane code is self-dual, every coloured face corresponds to an X-type and Z-type stabilizer. In Figure 2.1b, the quantum Reed-Muller code is visualized on a 3D lattice. Every coloured face corresponds to a weight-4 Z-type stabilizer. Every coloured cell corresponds to a weight-8 X-type stabilizer. For the Steane code, the logical operators defined in Equation (2.2) correspond to an edge in the triangle. For the quantum Reed-Muller code, the logical X operator corresponds to a weight-7 triangular face, and the logical Z operator corresponds to a weight-3 edge of the entire tetrahedron. An example is shown below.

$$\overline{X} = X_1X_2X_3X_4X_5X_6X_7 \qquad \text{and} \qquad \overline{Z} = Z_1Z_4Z_5 \tag{2.4}$$

Given such representations, the Steane code and the quantum Reed-Muller code are also special cases of colour codes [7, 8, 62].

(a) Visualize the Steane code. (b) Visualize the quantum Reed-Muller code.

Figure 2.1: Each vertex represents a physical qubit. Each edge serves as an aid to the eye. They do not imply any physical interactions or inherent structures. In Figure 2.1a, the Steane code is visualized as a 2D colour code. In Figure 2.1b, the quantum Reed-Muller code is visualized as a 3D colour code.

From Equation (2.3), the extended Steane code is self-dual, and its encoded state is characterized by the lemma below. It shows that $\mathcal{S}_{ex}$ and $\mathcal{S}_{steane}$ are equivalent up to some auxiliary state.

Lemma 1 ([3]). *Any codeword $|\psi\rangle$ of the extended Steane code can be decomposed into a codeword $|\phi\rangle$ of the Steane code and a fixed state $|\eta\rangle$. That is,*

$$|\psi\rangle = |\phi\rangle \otimes |\eta\rangle,$$

where $|\eta\rangle = \frac{1}{\sqrt{2}}(|0\rangle|\overline{0}\rangle + |1\rangle|\overline{1}\rangle)$, $|\overline{0}\rangle$ *and* $|\overline{1}\rangle$ *are the logical 0 and 1 encoded in the Steane code.*

Since the logical information $|\phi\rangle$ encoded in the Steane code is not entangled with $|\eta\rangle$, to switch between the Steane code and the extended Steane code, one may simply add or discard the auxiliary state $|\eta\rangle$. This property will prove useful in Chapter 5.

Next, we define the $[\![15, 1, 3, 3]\!]$ CSS subsystem code [94]. As per Definition 3, let $\mathcal{S}_{sub}$ and $\mathcal{G}$ be its stabilizer group and gauge group respectively.

$$\mathcal{S}_{sub} = \langle F^X, F^Z, H^Z \rangle, \quad \mathcal{G} = \langle F^X, F^Z, H^X, H^Z, J^Z \rangle. \tag{2.5}$$

Let $\mathcal{L}_g = \mathcal{G}/\mathcal{S}$ and $\mathcal{L} = \mathcal{N}(\mathcal{S})/\mathcal{G}$. One can verify that

$$\mathcal{L}_g = \langle H^X, J^Z \rangle, \quad \mathcal{L} = \langle \overline{X}, \overline{Z} \rangle. \tag{2.6}$$

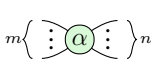
Thus, the CSS subsystem code has one logical qubit and three gauge qubits, and they are acted on by $\mathcal{L}_g$ and $\mathcal{L}$ respectively. From Chapter 3 onwards, we call operators in $\mathcal{L}_g$ as *gauge operators*.

Moreover, $\mathcal{S}_{sub}$ can be viewed as the stabilizer group of a $[\![15, 4, 3]\!]$ CSS code, with logical operators $\mathcal{L}'$. This code appears in an intermediary step of the gauge fixing process in Chapter 5.

$$\mathcal{L}' \coloneqq \mathcal{L}_g \cup \mathcal{L} = \langle H^X, J^Z, \overline{X}, \overline{Z} \rangle. \tag{2.7}$$

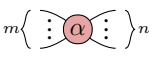
2.4 The ZX Calculus

The qubit ZX-calculus [26, 27, 28, 91] is a quantum graphical calculus for diagrammatic reasoning of any qubit quantum computation. Every diagram in the calculus is composed of two types of generators: Z spiders, which sum over the eigenbasis of the Pauli Z operator:



$$\coloneqq |0\rangle^{\otimes n}\langle 0|^{\otimes m} + e^{i\alpha}|1\rangle^{\otimes n}\langle 1|^{\otimes m}, \tag{2.8}$$

and X spiders, which sum over the eigenbasis of the Pauli X operator:



$$\coloneqq |+\rangle^{\otimes n}\langle +|^{\otimes m} + e^{i\alpha}|-\rangle^{\otimes n}\langle -|^{\otimes m}. \tag{2.9}$$

The ZX-calculus is *universal* [27] in the sense that any linear map from m qubits to n qubits corresponds exactly to a ZX diagram, by the construction of Equations (2.8) and (2.9) and the composition of linear maps.

Furthermore, the ZX-calculus is *complete* [49, 53]: Any equality of linear maps on any number of qubits derivable in the Hilbert space formalism, is derivable using only a finite set of rules in the calculus. The smallest complete rule set to date [93] is shown in Figure 2.2.

Some additional rules, despite being derivable from this rule set, will be convenient to use in this paper. They are summarized in Figure 2.3.

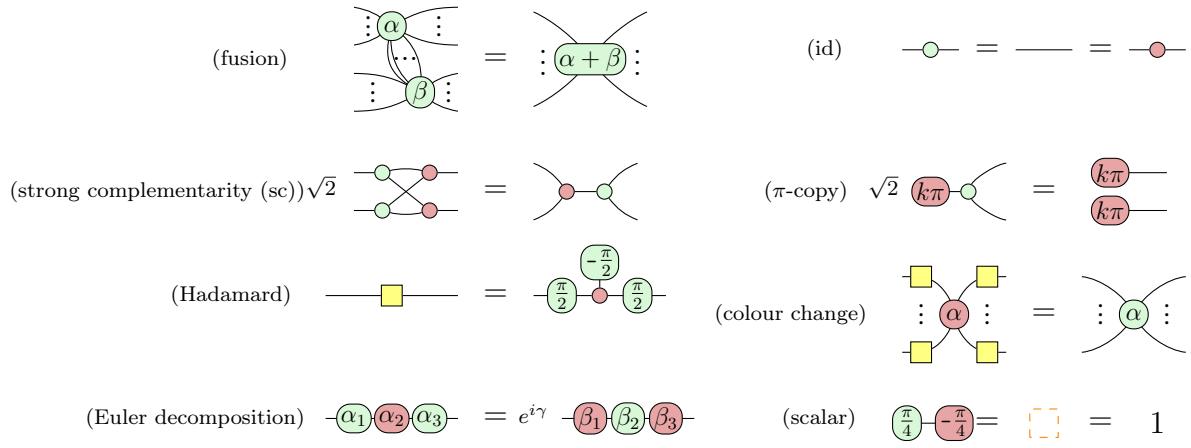


Figure 2.2: These eight equations suffice to derive all other equalities of linear maps on qubits [93]. $k \in \mathbb{Z}_2$. α_i, β_i and γ are real numbers satisfying the trigonometric relations derived in [29]. Each equation still holds when we replace all spiders with their corresponding spiders of the opposite colour. Whenever there are any two wires with ... between them, the rule holds when replacing this with any number of wires (i.e., 0 or greater).

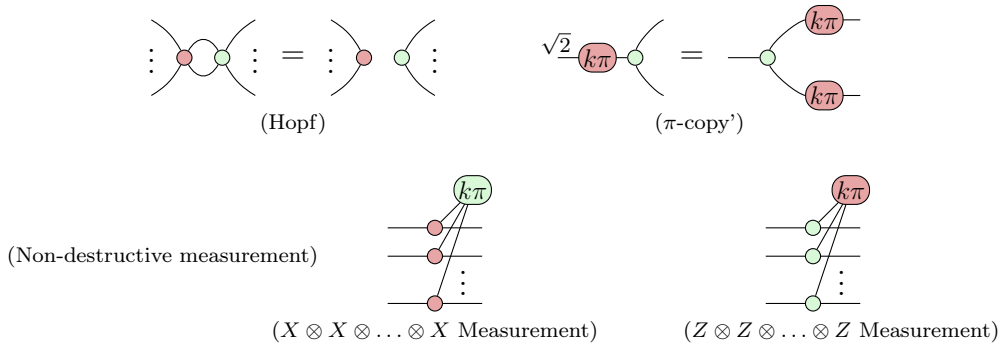


Figure 2.3: Some other useful rewrite rules, each derivable from the rules in Figure 2.2. $k \in \mathbb{Z}_2$. Each equation still holds when we interchange X and Z spiders.

When a spider has phase zero, we omit its phase in the diagram, as shown below. A ZX diagram is *phase-free* if all of its spiders have zero phases. For more discussions on phase-free ZX diagrams, we refer readers to consult [55].

2.5 The ZX Normal Form for CSS Codes

Due to the universality of the ZX calculus, quantum error-correcting code encoders, as linear isometries, can be drawn as ZX diagrams [54]. Moreover, the encoder for a CSS code corresponds exactly to the phase-free *ZX (and XZ) normal form* [55].

Definition 4. *For a CSS stabilizer code defined by $\mathcal{S}$, let $\left\{S_i^X; 1 \leq i \leq m_1\right\} \subset \mathcal{S}$ be the X-type stabilizer generators and $\left\{\overline{X_j}; 1 \leq j \leq k\right\}$ be the logical X operators, $m_1 + k < n$. Its ZX normal form can be found via the following steps:*

(a) For each physical qubit, introduce an X spider.

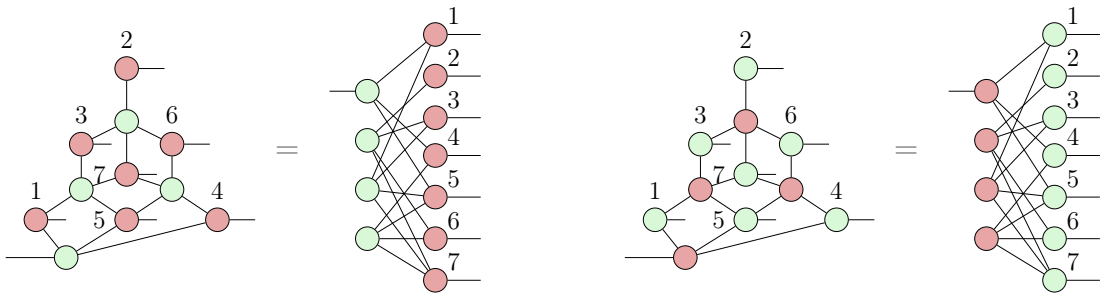
(b) For each X-type stabilizer generator S_i^X and logical operator $\overline{X_j}$, introduce a Z spider and connect it to all X spiders where this operator has support.

(c) Give each X spider an output wire.

(d) For each Z spider representing $\overline{X_j}$, give it an input wire.

As an example, the ZX normal form for the Steane code is drawn in Figure 2.4a. The XZ normal form can be constructed based on the Z-type stabilizer generators $\left\{S_i^Z; 1 \leq i \leq m_2\right\} \subset \mathcal{S}$ and the logical Z operators $\left\{\overline{Z_j}; 1 \leq j \leq k\right\}$ by inverting the roles of X and Z spiders in Definition 4. Note that $m_1 + m_2 + k = n$. Correspondingly, the XZ normal form for the Steane code is drawn in Figure 2.4b.

In [55], Kissinger gave an algorithm to rewrite any phase-free ZX diagram into both the ZX and XZ normal forms. In Appendix A, we prove that it is sufficient to represent a CSS code encoder using either one of the forms.



(a) The Steane code encoder in the ZX normal form. (b) The Steane code encoder in the XZ normal form.

Figure 2.4: The XZ and ZX normal forms of the Steane code.

Chapter 3

The Graphical Construction of CSS Encoders

3.1 The ZX Normal Form for CSS Subsystem Codes

We generalize the ZX normal form for CSS stabilizer codes to CSS subsystem codes as follows.

Definition 5. *For an $[\![n, k, r, d]\!]$ CSS subsystem code defined by $(\mathcal{S}, \mathcal{G})$, let $\left\{S_i^X; 1 \leq i \leq m\right\}$ be the X-type stabilizer generators, $\left\{L_{g_t}^X; 1 \leq t \leq r\right\}$ be the X-type gauge operators, and $\left\{\overline{X_j}; 1 \leq j \leq k\right\}$ be the logical X operators, $m + k + r < n$. Its ZX normal form can be found via the following steps:*

(a) For each physical qubit, introduce an X spider.

(b) For each stabilizer generator S_i^X, logical operator $\overline{X_j}$ and gauge operator $L_{g_t}^X$, introduce a Z spider and connect it to all X spiders where this operator has support.

(c) Give each X spider an output wire.

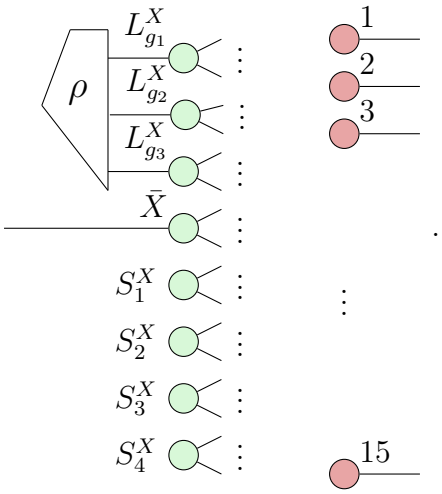
(d) For each Z spider representing $\overline{X_j}$, give it an input wire.

(e) For all Z spiders representing $L_{g_t}^X$, attach to them a joint arbitrary input state (i.e., a density operator ρ).

Similar to CSS stabilizer codes, CSS subsystem codes also have an equivalent XZ normal form, which can be found by inverting the role of Z and X in the above procedure.

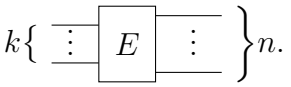
For $n > 3$, below we exemplify the ZX normal form for an $[\![n, 1, 2, d]\!]$ CSS subsystem code with three X-type stabilizers generators $\left\{S_1^X, S_2^X, S_3^X\right\}$, two X-type gauge operators $\left\{L_{g_1}^X, L_{g_2}^X\right\}$, and one logical operator $\left\{\overline{X}\right\}$. For simplicity, we substitute wires connecting Z and X spiders by $\lessdot$ $\vdots$. The detailed connectivities are omitted here, but they should be

clear following step (b) in Definition 5. This notation will be used in the remainder of this paper.



3.2 Pushing through the Encoder

For any $[\![n, k, d]\!]$ CSS code, its encoder map E is of the form:



Definition 6. *Let $\overline{X_i}$ and $\overline{Z_i}$ be the X and Z operators acting on the i-th logical qubit. Let $\overline{\mathcal{X}_i}$ and $\overline{\mathcal{Z}_i}$ be the physical implementation of $\overline{X_i}$ and $\overline{Z_i}$ respectively. Diagrammatically, they can be represented as*

In other words, pushing $\overline{X_i}$ (or $\overline{Z_i}$) through E yields $\overline{\mathcal{X}_i}$ (or $\overline{\mathcal{Z}_i}$). Using ZX rewrite rules along with the ZX (or XZ) normal form, we can prove Lemma 2. Although this is a well-known fault-tolerant property of the CSS codes, this graphical proof reveals how logical Pauli operators are propagated through the encoder.

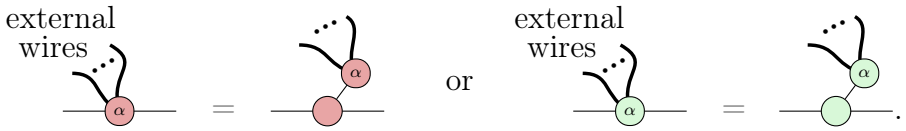
Lemma 2. *For any CSS code, all $\overline{X_i}$ and $\overline{Z_i}$ are implementable by multiple single-qubit Pauli operators. In other words, all CSS codes have* transversal *$\overline{X_i}$ and $\overline{Z_i}$.*

Proof. Consider an arbitrary CSS code. Without loss of generality, represent its encoder E in the ZX normal form following Definition 4. Then proceed by applying the π-copy' rule on every $\overline{X_i}$ (the X spider with a phase π on the left-hand side of the encoder E). □

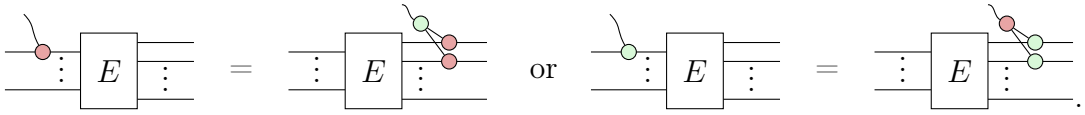
Beyond just X or Z spiders, one can push *any* logical ZX diagram through the encoder. Such pushing is bidirectional, and the left-to-right direction is interpreted as finding a physical implementation for a given logical operator.

Proposition 1. Let E be the encoder of a CSS code. For any ZX diagram L on the left-hand side of E, one can write down a corresponding ZX diagram P on the right-hand side of E, such that $EL = PE$. In other words, P is a valid physical implementation of L for that CSS code.

Proof. We proceed as follows. First, unfuse all spiders on the logical qubit wires of L, whenever they are not phase-free or have more than one external wire:



For each X (or Z) spider on the logical qubit wire, rewriting E to be in ZX (or XZ) normal form and applying the strong complementarity (sc) rule yields:



On the left-hand side, a phase-free X (or Z) spider acts on the i-th logical qubit; on the right-hand side, phase-free X (or Z) spiders act on all physical qubits wherever $\overline{X}_i$ (or $\overline{Z}_i$) has support. Therefore, any type of L can be pushed through E, resulting in a diagram P which satisfies $EL = PE$. □

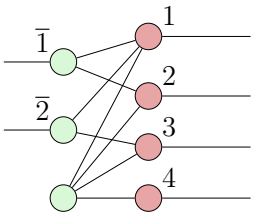
In [39], it was proved that a physical implementation P of a logical operator L satisfies $L = E^\dagger PE$. This is implied by $EL = PE$ as $E^\dagger E = I$.

3.3 Examples

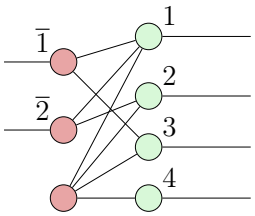
To illustrate Lemma 2 and proposition 1, we use the $[\![4, 2, 2]\!]$ code for an example. This code is defined by the stabilizer group $\mathcal{S} = \langle X_1X_2X_3X_4, Z_1Z_2Z_3Z_4\rangle$ and a set of logical operators $\mathcal{L}$:

$$\mathcal{L} = \{\overline{X_1}, \overline{Z_1}, \overline{X_2}, \overline{Z_2}\},\ \overline{X_1} = X_1X_2,\ \overline{Z_1} = Z_1Z_3,\ \overline{X_2} = X_1X_3,\ \overline{Z_2} = Z_1Z_2.$$

Its ZX and XZ normal forms are shown in Figure 3.1. Figure 3.1a is the ZX normal form defined by the logical X operators and the X-type stabilizer generator. Figure 3.1b is the XZ normal form defined by the logical Z operators and the Z-type stabilizer generator.



(a) The ZX normal form.



(b) The XZ normal form.

Figure 3.1: Represent the encoder of the $[\![4, 2, 2]\!]$ code in two equivalent normal forms.

Example 1 (Implement a Logical X Gate)**.** *In Figure 3.2, we show how to find the physical implementation of a logical X gate by pushing a logical X spider through the encoder. We use this example to articulate the proof of Lemma 2.*

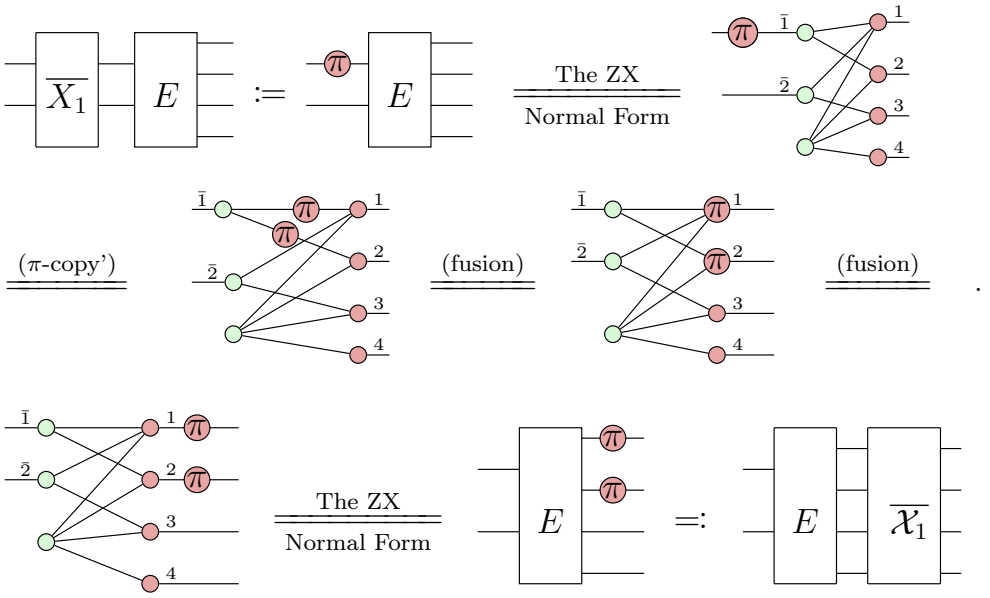


Figure 3.2: For the $[\![4, 2, 2]\!]$ code, $\overline{X_1} = X_1X_2$.

In what follows, we use Examples 2 to 5 to articulate the proof of Proposition 1.

Example 2 (Implement a Logical XX Measurement)**.** *The fundamental building block for lattice surgery is a multi-qubit Pauli measurement across two lattices. In Figure 3.3, we show that a non-destructive logical-qubit measurement in (i) can be implemented by a non-destructive physical-qubit measurement in (v).*

To start, substitute the $[\![4, 2, 2]\!]$ *encoder by Figure 3.1a, as shown in (ii). Then, consider the bottom lattice (i.e., the second code block) and the Z spider with a* $k\pi$ *phase as an exterior attachment to the X spider in the top lattice (i.e., the first code block). Applying the strong complementarity (sc) rule to the top lattice yields (iii), which is the partially fixed ZX diagram. After that, consider the top-fixed lattice as an attachment to the bottom lattice and apply the sc rule. This results in (iv), which is equivalently expressed in (v). This transformation agrees with the graphical derivation for the colour code lattice surgery [50].*

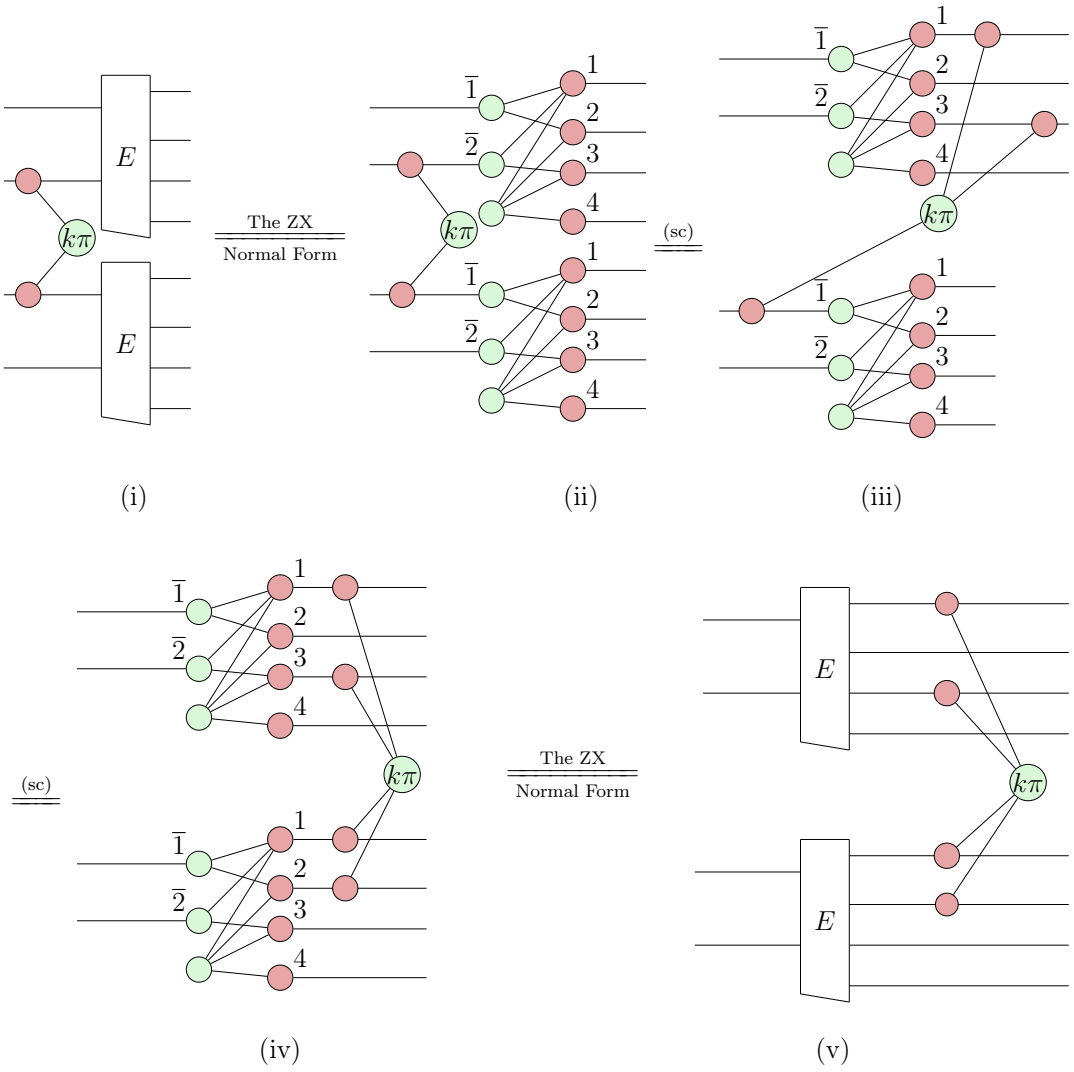


Figure 3.3: By pushing the logical XX measurement through the encoder, we show that a Pauli measurement of two logical blocks can be implemented by a Pauli measurement of multiple physical qubits.

Example 3 (Implement a Logical CNOT Gate)**.** *To push a logical CNOT through the encoder, we proceed by case distinctions.*

Case 1: *When a CNOT acts on two logical qubits within one code block.*

In Figure 3.4, we push the CNOT through the encoder by applying Proposition 1. Equivalently, we find the physical implementation of a logical CNOT for the $[\![4, 2, 2]\!]$ *code.*

Case 2: *When a CNOT acts on two logical qubits across two code blocks.*

In Figure 3.5, after applying a sequence of rewrite rules, a logical CNOT is pushed through the encoder. As a result, we find the physical implementation of a logical CNOT acting on two code blocks. Alternatively, in Figure 3.6, we apply Proposition 1 and derive the same result.

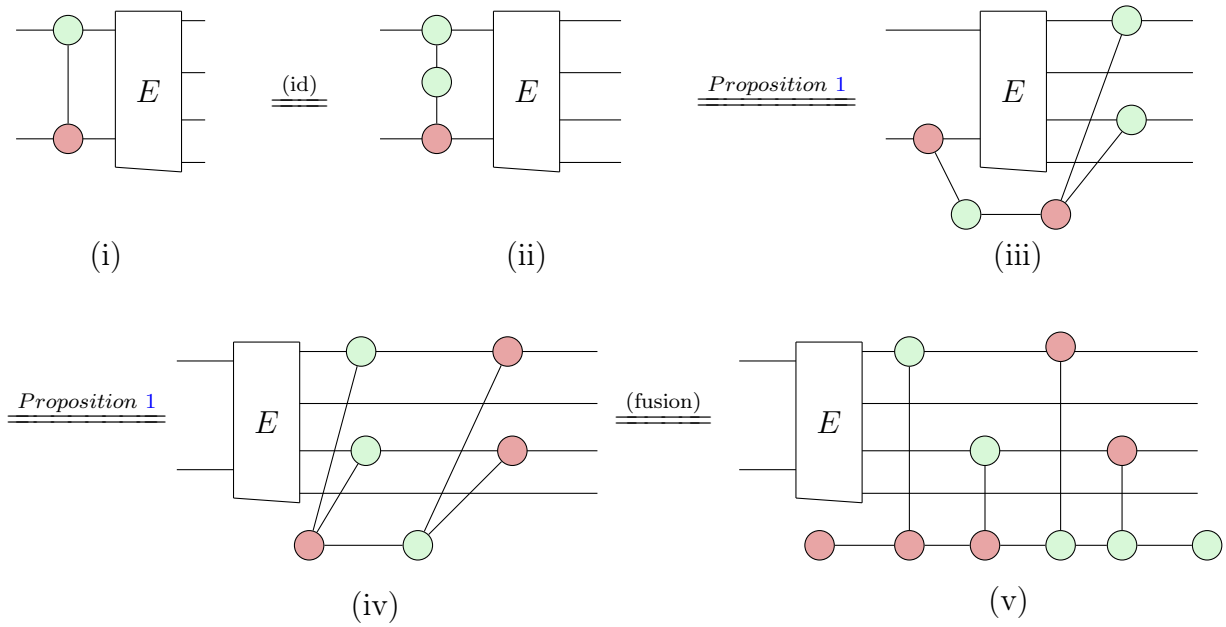


Figure 3.4: When a logical CNOT acts within one code block, push the CNOT through the encoder by applying Proposition 1. As a result, we find a non-transversal implementation of this logical CNOT gate, as shown in (v).

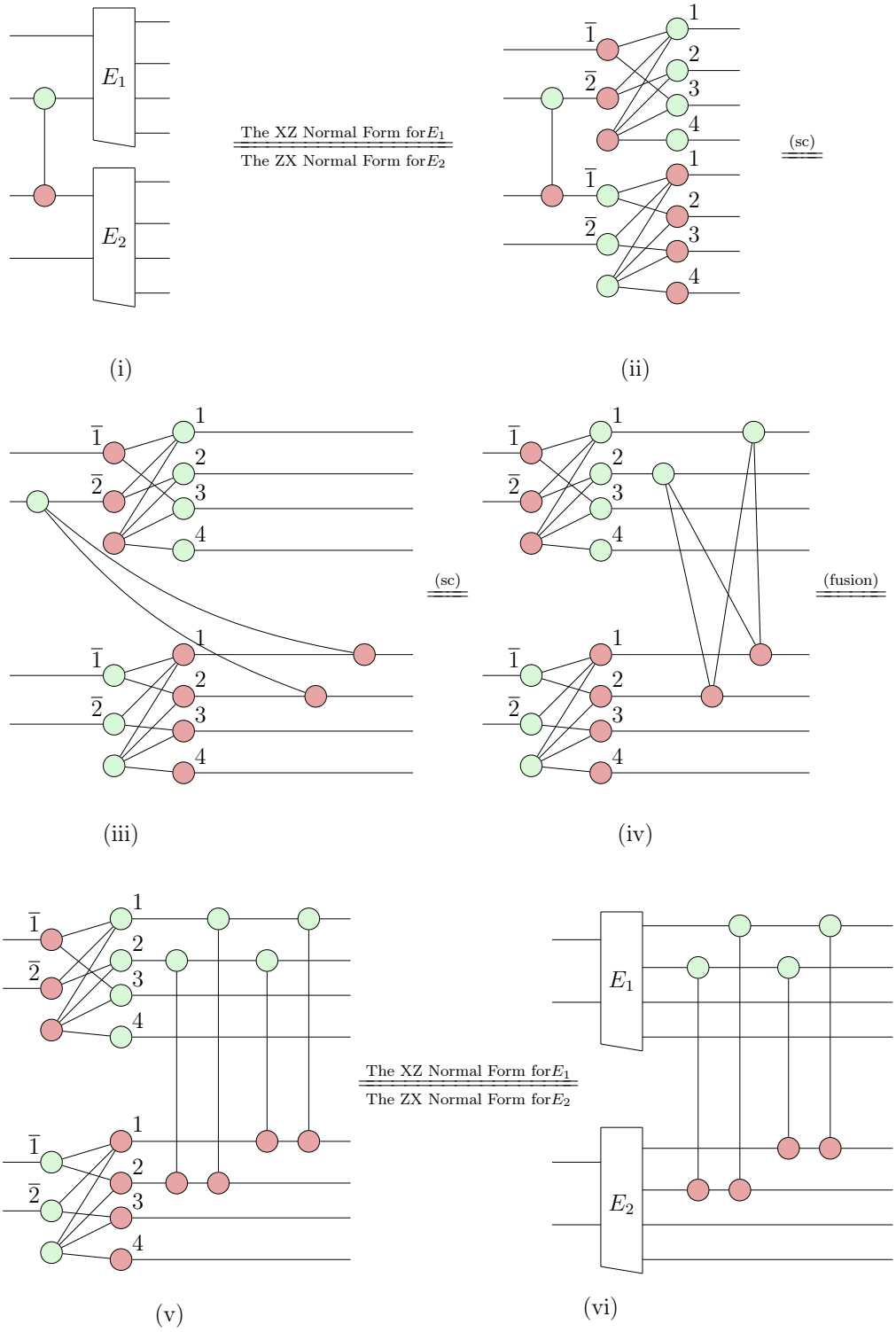


Figure 3.5: When a logical CNOT acts on two code blocks, push the CNOT through the encoder by applying a sequence of rewrite rules. As a result, we find a transversal implementation of this logical CNOT gate, as shown in (vi).

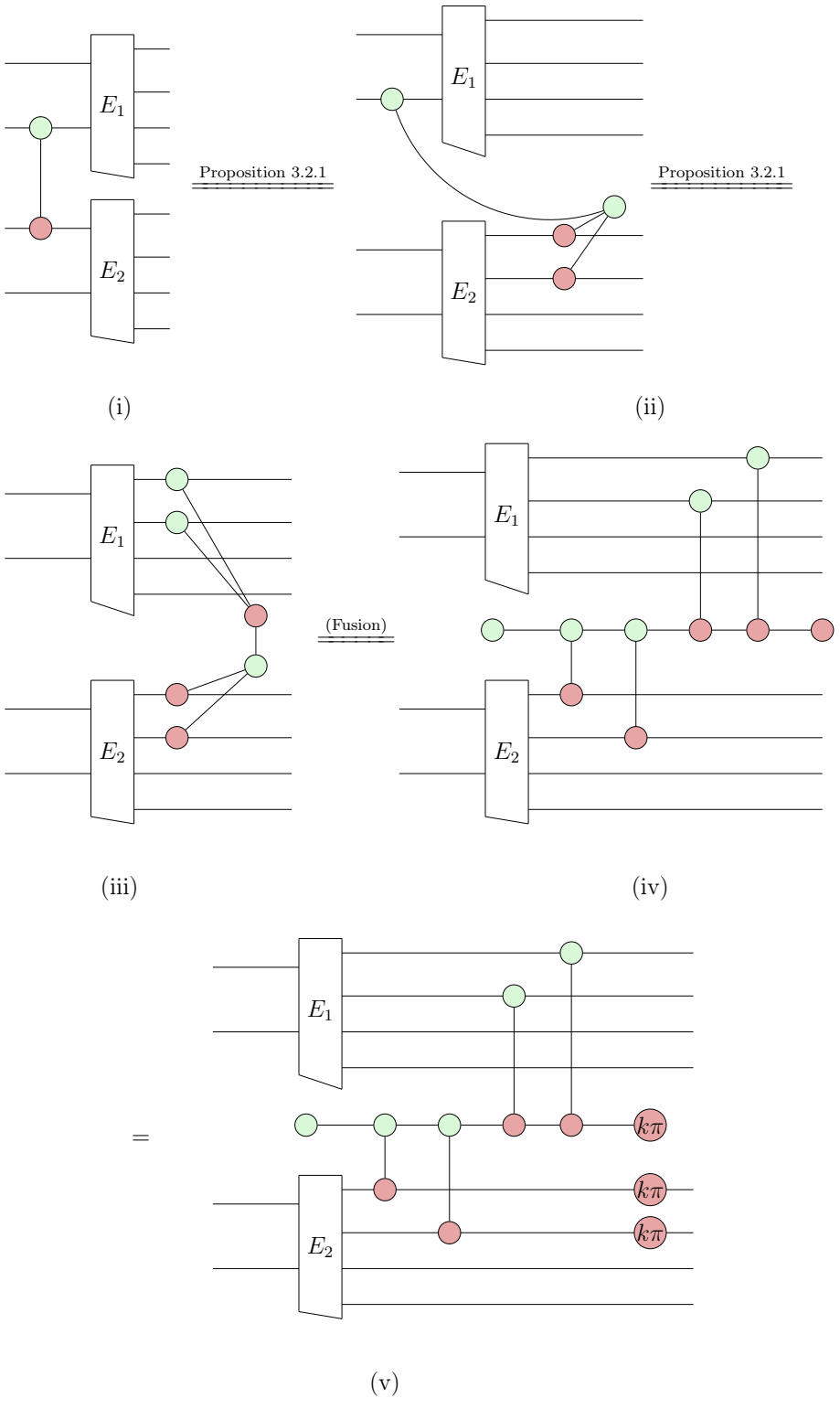


Figure 3.6: When a logical CNOT acts on two code blocks, push the CNOT through the encoder by applying Proposition 1. In (v), $k \in \mathbb{Z}_2$. We can interpret the physical implementation in terms of the following steps. First, prepare a logical ancilla in the $|+\rangle$ state. For each physical qubit where $\overline{X_2} = X_1X_3$ or $\overline{Z}_1 = Z_1Z_3$ has support, apply a CNOT on this qubit and the ancilla. Next, measure the ancilla in the Pauli Z basis. Finally, if the measurement output is $k = 1$, apply a Pauli X correction on the physical qubits where $\overline{X_2} = X_1X_3$ has its support.

Example 4 (Implement a Logical T Gate)**.** *In Figure 3.7, we push a logical T gate through the encoder by applying a sequence of rewrite rules. This helps visualize the techniques used in proving Proposition 1.*

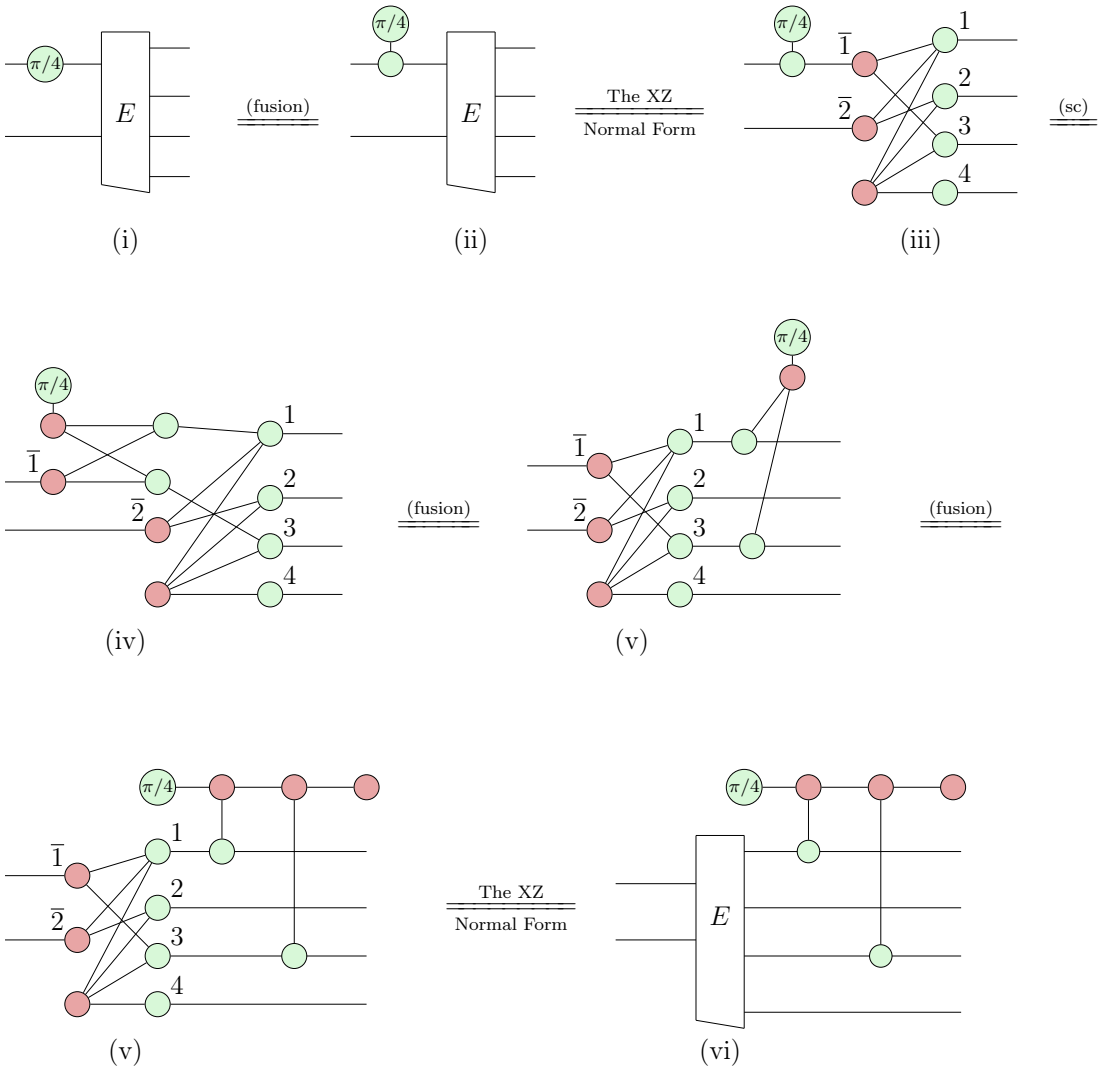


Figure 3.7: A logical T gate is pushed through the encoder by applying a sequence of rewrite rules. As a result, we find a non-transversal implementation of this logical T gate, as shown in (vi).

Example 5 (Implement a Logical Hadamard Gate)**.** *In Figure 3.8, we show how to find a physical implementation of a logical Hadamard gate by applying Proposition 1.*

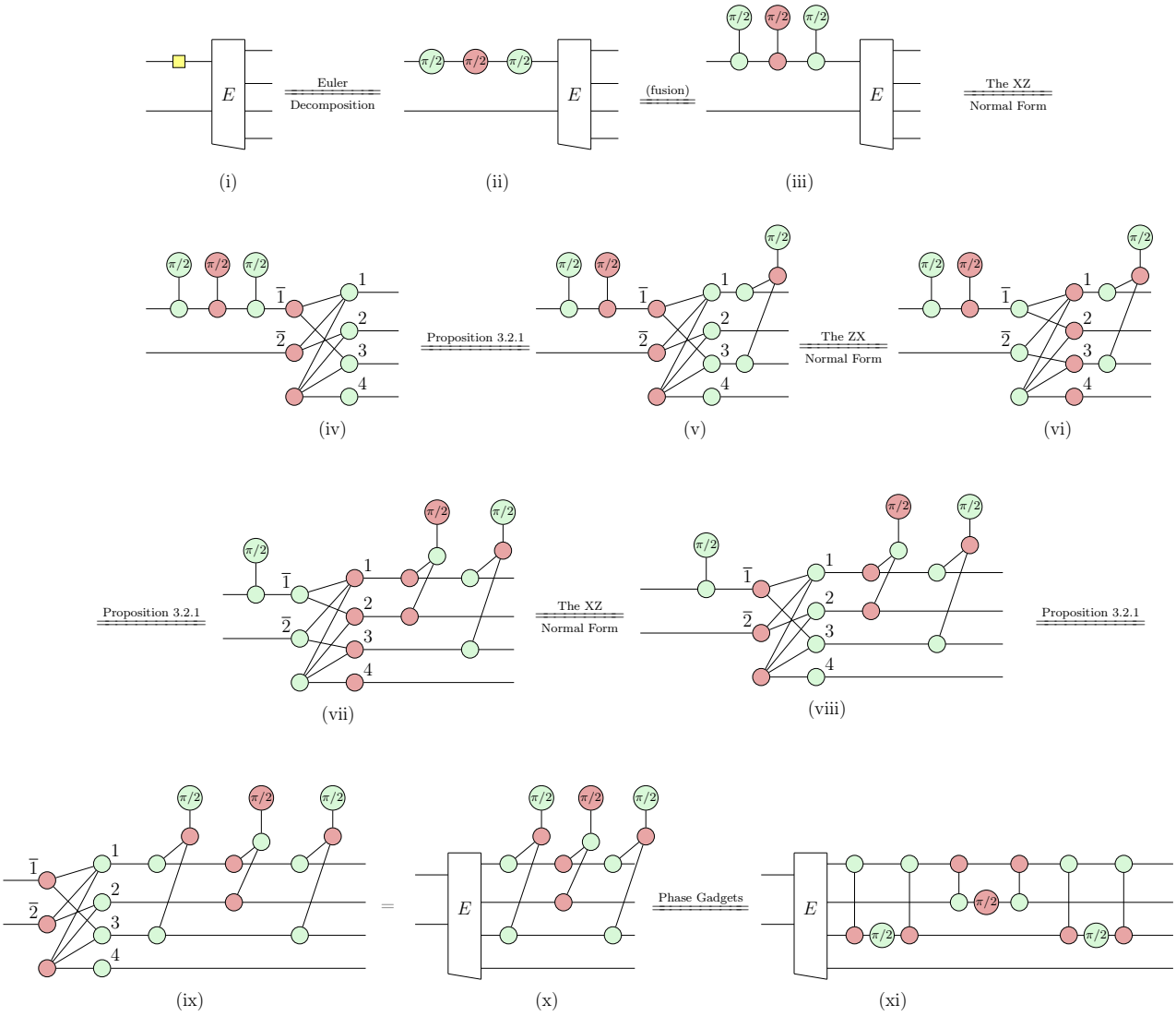


Figure 3.8: To push a logical Hadamard gate through the encoder, we first apply the Euler decomposition (ii). After unfusing logical spiders (iii), apply Proposition 1 three times and each time with a convenient normal form. As a result, we find a non-transversal implementation of this logical Hadamard gate, as shown in (xi).

Chapter 4

Graphical Morphing of CSS Codes

One way to transform CSS codes is known as *code morphing*. It provides a systematic framework to construct new codes from an existing code while preserving the number of logical qubits in the morphed code. Here, we present this procedure by rewriting the encoder diagram using the ZX calculus. We start by revisiting the code morphing definition in [92].

Definition 7. *Let $\mathcal{S}$ be a stabilizer group and $\mathcal{C}$ be its joint $+1$ eigenspace. $\mathcal{C}$ is called the* parent code. *Let Q denote the set of physical qubits of $\mathcal{C}$ and $R \subseteq Q$. Then $\mathcal{S}(R)$ is a subgroup of $\mathcal{S}$ generated by all stabilizers of $\mathcal{S}$ that are fully supported on R. Let $\mathcal{C}(R)$ be the joint $+1$ eigenspace of $\mathcal{S}(R)$, and $\mathcal{C}(R)$ is called the* child code. *Given the parent code encoder $E_{\mathcal{C}}$, concatenate it with the inverse of the child code encoder $E^{\dagger}_{\mathcal{C}(R)}$. This gives the* morphed code $\mathcal{C}_{\setminus R}$.

Definition 8. *Let M be a stabilizer generator of a CSS code. Let $\mathsf{supp}(M)$ be the set of physical qubits where M acts non-trivially. For brevity, we call $\mathsf{supp}(M)$ the* support *of M. There are three types of qubit coverage of R concerning $\mathsf{supp}(M)$.*

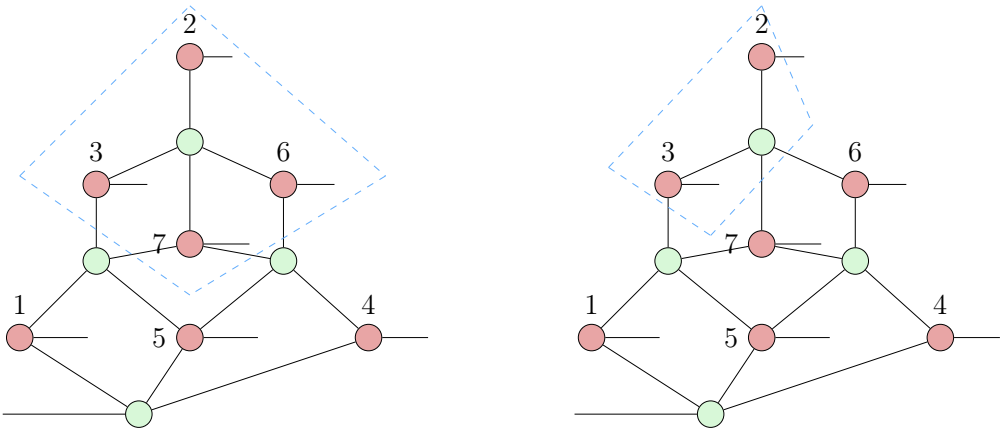
- *When $|\mathsf{supp}(M) \cap R| = |\mathsf{supp}(M)|$, M is* fully-supported *on R.*
- *When $0 < |\mathsf{supp}(M) \cap R| < |\mathsf{supp}(M)|$, M is* partially-supported *on R.*
- *When $|\mathsf{supp}(M) \cap R| = 0$, M is* not supported *on R.*

In Figure 4.1 we use the $[\![7, 1, 3]\!]$ Steane code to illustrate different types of qubit coverage. Without loss of generality, consider its ZX normal form. Its X-type stabilizer generators and their respective supports are as follows.

$$\begin{aligned} S_1^X &= X_1X_3X_5X_7, & \mathsf{supp}(S_1^X) &= \{1,3,5,7\},\\ S_2^X &= X_2X_3X_6X_7, & \mathsf{supp}(S_2^X) &= \{2,3,6,7\},\\ S_3^X &= X_4X_5X_6X_7, & \mathsf{supp}(S_3^X) &= \{4,5,6,7\}. \end{aligned}$$

In Figure 4.1a, take $R_1 = \{2,3,6,7\} \subset Q$. Since $|\mathsf{supp}(S_1^X) \cap R_1| = |\{3,7\}| = 2 < 4$, $|\mathsf{supp}(S_3^X) \cap R_1| = |\{6,7\}| = 2 < 4$, S_1^X and S_3^X are partially supported on R_1. Since $|\mathsf{supp}(S_2^X) \cap R_1| = |\{2,3,6,7\}| = 4$, S_2^X is fully supported on R_1.

In Figure 4.1b, take $R_2 = \{2,3\} \subset Q$. Since $|\mathsf{supp}(S_1^X) \cap R_2| = |\{3\}| = 1 < 4$, $|\mathsf{supp}(S_2^X) \cap R_2| = |\{2,3\}| = 2 < 4$, S_1^X and S_3^X are partially supported on R_2. Since $|\mathsf{supp}(S_3^X) \cap R_2| = |\emptyset| = 0$, S_3^X is not supported on R_2.



(a) $R_1 = \{2,3,6,7\}$ corresponds to the 4 X spiders in the blue dashed box.

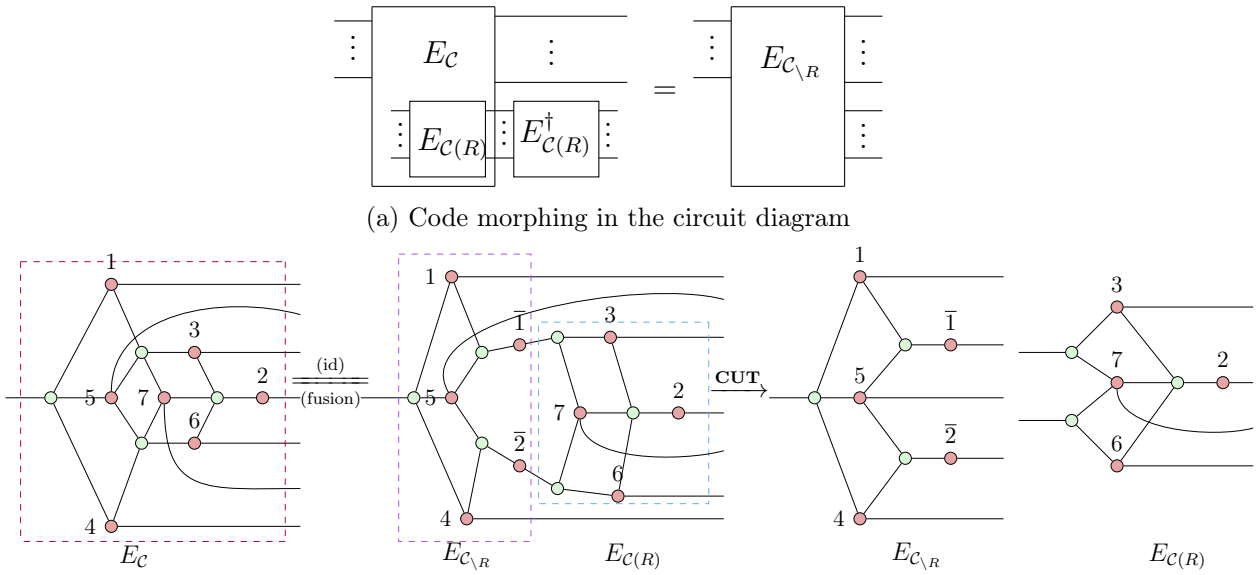
(b) $R_2 = \{2,3\}$ corresponds to the 2 X spiders in the blue dashed box.

Figure 4.1: Given the ZX normal form of the Steane code, $Q = \{1,2,3,4,5,6,7\}$. Different choices of $R \subset Q$ provide different coverage for an X-type stabilizer generator.

Figure 4.2 provides two equivalent interpretations for the code morphing process. In Figure 4.2a, Definition 7 is depicted by the circuit diagram. Since $E_{\mathcal{C}(R)}$ is an isometry, $E_{\mathcal{C}(R)}^\dagger E_{\mathcal{C}(R)} = I$. By construction, the equation shown in Figure 4.2a holds [92]. Moreover, the parameters of $\mathcal{C} = [\![n,k,d]\!]$, $\mathcal{C}(R) = [\![n_1,k_1,d_1]\!]$, and $\mathcal{C}_{\setminus R} = [\![n_2,k_2,d_2]\!]$ are characterized below. Let m, m_1, m_2 be the number of stabilizer generators for $\mathcal{C}$, $\mathcal{C}(R)$, and $\mathcal{C}_{\setminus R}$ respectively. Then

$$n_2 = n - n_1 + k_1, \quad k_2 = k, \quad m_2 = (n-k) - (n_1 - k_1) = m - m_1, \quad d_1, d_2 \in \mathbb{N}.$$

Figure 4.2b provides a concrete example of applying Definition 7 to the $[\![7,1,3]\!]$ Steane code, where $S = \{1,2,3,4,5,6,7\}$ and $R = \{2,3,6,7\}$. As a result, the $[\![5,1,2]\!]$ code is morphed from the parent code along with the $[\![4,2,2]\!]$ child code. This morphed code inherits a fault-tolerant implementation of the Clifford group from the $[\![7,1,3]\!]$ code, which has a transversal implementation of the logical Clifford operators. This morphing process is represented in the ZX diagram by cutting the edges labelled by $\overline{1}$ and $\overline{2}$ adjacent to the X spider. This is equivalent to concatenating the ZX diagram of $E_{[\![4,2,2]\!]}^\dagger$ in Figure 4.2a.



(a) Code morphing in the circuit diagram

(b) Code morphing of the Steane code in the ZX diagram

Figure 4.2: Code morphing can be visualized using both circuit and ZX diagrams. In Figure 4.2a, code morphing is viewed as a concatenation of the parent code encoder $E_{\mathcal{C}}$ and the inverse of the child code encoder $E^{\dagger}_{\mathcal{C}(R)}$. In Figure 4.2b, the encoder $E_{\mathcal{C}}$ of the Steane code is represented in the ZX normal form. As described in Proc. 1, by applying ZX rules (id) and (fusion) in Figure 2.2, we can perform code morphing by bipartitioning it into the encoder $E_{\mathcal{C}_{\setminus R}}$ of the morphed code $\mathcal{C}_{\setminus R} = [\![5, 1, 2]\!]$, and the encoder $E_{\mathcal{C}(R)}$ of the child code $\mathcal{C}(R) = [\![4, 2, 2]\!]$.

Next, we generalize the notion of code morphing and show how ZX calculus could be used to study these relations between the encoders of different CSS codes. More precisely, we provide an algorithm to morph a new CSS code from an existing CSS code.

Procedure 1. *Given a parent code $\mathcal{C}$ and a child code $\mathcal{C}(R)$ satisfying Definition 7, construct the encoder of $\mathcal{C}$ in the ZX normal form. Then the code morphing proceeds as follows:*

(a) Unfuse every Z spider which is supported on c qubits within R and f qubits outside R, $c \neq 0$, $f \neq 0$.

(b) Add an identity X spider between each pair of Z spiders being unfused in step (a).

(c) Cut the edge between every identity X spider and the Z spiders supported on the f qubits in R.

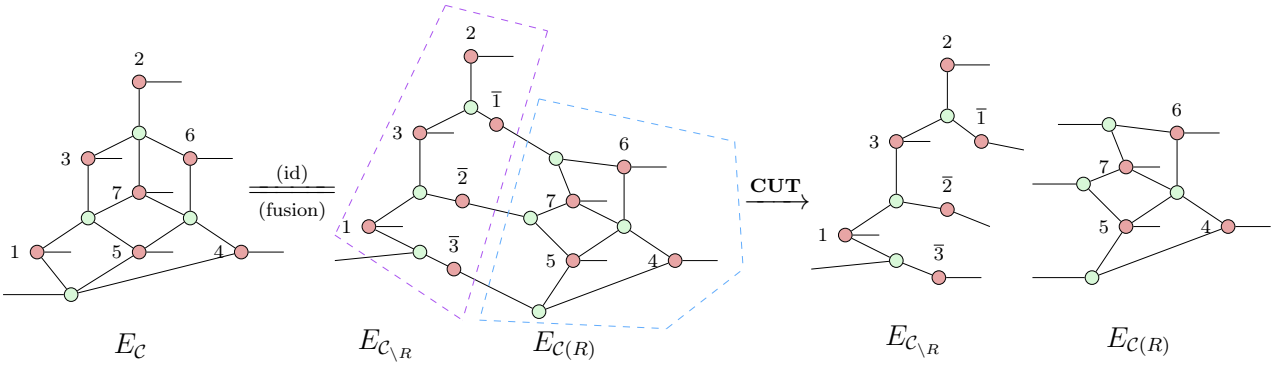
It follows that the subdiagram containing R corresponds to the ZX normal form of $E_{\mathcal{C}(R)}$. It has the same number of X spiders as R, so $n_1 = |R|$. Suppose that there are h Z spiders being unfused. Then h must be bounded by the number of Z spiders in the ZX normal form of $E_{\mathcal{C}}$. As each spider unfusion introduces a logical qubit to $\mathcal{C}(R)$, $k_1 = h$. On the other hand, the complement subdiagram contains $n - n_1 + k_1$ X spiders as each edge cut introduces a new X spider into the complement subdiagram. It also contains k

logical qubits as the input edges in the ZX normal form of $E_{\mathcal{C}}$ are invariant throughout the spider-unfusing and edge-cutting process. This gives the ZX normal form for the encoder of the morphed code $\mathcal{C}_{\backslash\mathcal{R}} = [\![n_2, k_2, d_2]\!]$, where $n_2 = n - n_1 + k_1$, $k_2 = k$, $d_2 \in \mathbb{N}$. As a result, the ZX normal form of $E_{\mathcal{C}}$ is decomposed into the ZX normal forms of $E_{\mathcal{C}(R)}$ and $E_{\mathcal{C}_{\backslash\mathcal{R}}}$ respectively.

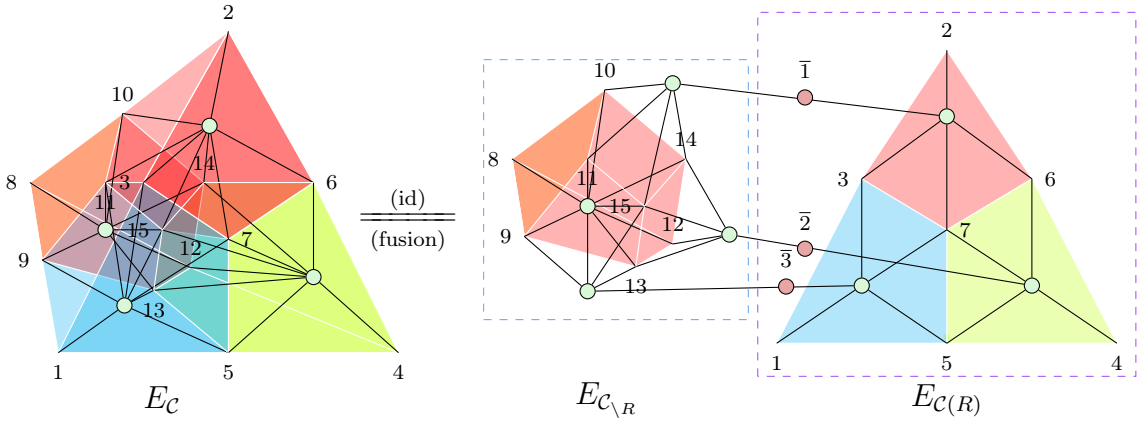
As the XZ and ZX normal forms are equivalent for CSS codes, Proc. 1 can be carried out for the XZ normal form by inverting the roles of Z and X at each step.

Here, we exemplify the application of Proc. 1 by morphing two simple CSS codes. Unlike Figure 4.2b, Example 6 chooses a different subset of qubits, $R = \{4, 5, 6, 7\}$, to obtain the $[\![6, 1, 1]\!]$ morphed code. In Example 7, we visualize the $[\![10, 1, 2]\!]$ code morphing from the $[\![15, 1, 3]\!]$ quantum Reed-Muller code. The $[\![10, 1, 2]\!]$ code is interesting because it inherits a fault-tolerant implementation of the logical T gate from its parent code, which has a transversal implementation of the logical T gate.

Example 6. *Let the parent code $\mathcal{C}$ be the Steane code and the child code be $C(R) = [\![4, 3, 1]\!]$. By Proc. 1, we obtain the morphed code $\mathcal{C}_{\backslash R} = [\![6, 1, 1]\!]$. Note that for $C(R)$, there is one X-type stabilizer generator and no Z-type stabilizer generator. This means that $C(R)$ cannot detect a single-qubit X error, so it has a distance of 1. In $\mathcal{C}_{\backslash R}$, the physical qubit labelled $\overline{3}$ is not protected by any X-type stabilizer. Therefore, $\mathcal{C}_{\backslash R}$ is of distance 1.*



Example 7. *Let the parent code $\mathcal{C}$ be the quantum Reed-Muller and the child code be $\mathcal{C}(R) = [\![8, 3, 2]\!]$. By Proc. 1, we obtain the morphed code $\mathcal{C}_{\backslash R} = [\![10, 1, 2]\!]$. For brevity, the X spiders representing physical qubits and the logical qubit wires inputting to the Z spiders are omitted.*



Chapter 5

Graphical Code Switching of CSS Codes

Another way to transform CSS codes is known as *code switching*. It is a widely studied technique in quantum error correction. Codes with complementary fault-tolerant gate sets are switched between each other to realize a universal set of logical operations. As a case study, we focus on the code switching protocol between the Steane code and the quantum Reed-Muller code [3, 74, 77]. Since this process is bidirectional, the reasoning for one direction can be simply adjusted for the opposite direction. Recall in Lemma 1, we showed that the extended Steane code is equivalent to the Steane code up to some auxiliary state. In what follows, we focus on the *backward switching* from the quantum Reed-Muller code to the extended Steane code.

Using the ZX calculus, we provide a graphical interpretation for the backward code switching. More precisely, it is visualized as gauge-fixing the $[\![15, 1, 3, 3]\!]$ subsystem code, followed by a sequence of syndrome-determined recovery operations.

We first characterize the relations between the quantum Reed-Muller code, the extended Steane code, and the $[\![15, 1, 3, 3]\!]$ subsystem code. For brevity, we denote these codes as $\mathcal{C}_{qrm}$, $\mathcal{C}_{ex}$ and $\mathcal{C}_{sub}$, and their respective encoders as E_{qrm}, E_{ex}, and E_{sub}.

Lemma 3. *When the three gauge qubits are in the* $|\overline{+++}\rangle$ *state,* $\mathcal{C}_{sub}$ *is equal to* $\mathcal{C}_{ex}$*, as shown in Figure 5.1.*

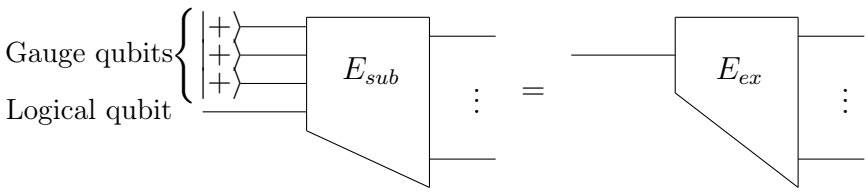
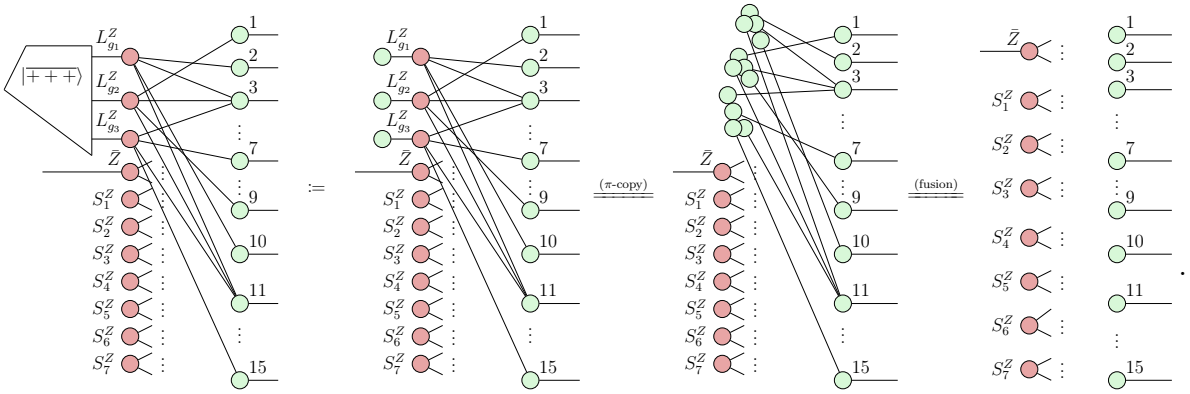


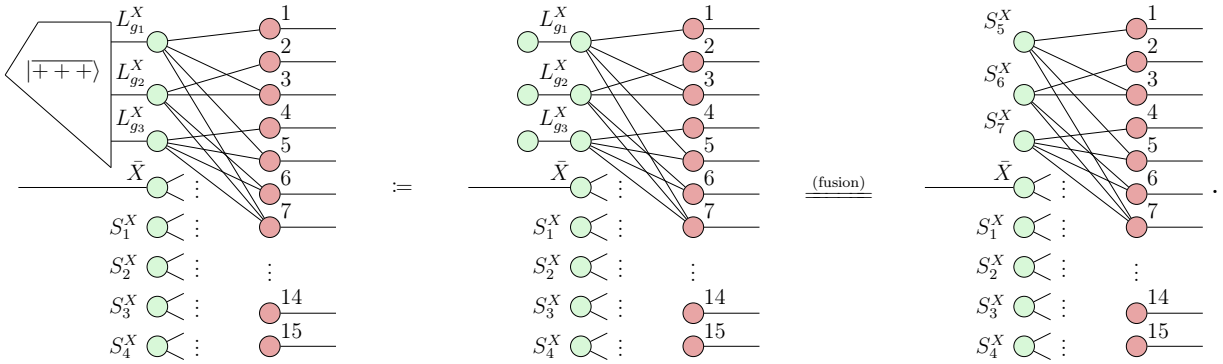
Figure 5.1: $\mathcal{C}_{sub}$ is equivalent to $\mathcal{C}_{ex}$ up to a fixed state of gauge qubits.

Proof. According to Definition 3, represent E_{sub} in the XZ normal form. With a sequence of rewrite rules, we obtain exactly the XZ normal form for E_{ex} which captures both the

Z-type stabilizer generators S_i^Z and the logical Z operators $L_{g_j}^Z$, $1 \leq i \leq 4$, $1 \leq j \leq 3$.



Alternatively, if one chooses to represent E_{sub} in the ZX normal form, the proof proceeds by applying the (fusion) rule to the Z spiders and identifying the gauge operators $L_{g_1}^X$, $L_{g_2}^X$, $L_{g_3}^X$ of $\mathcal{C}_{sub}$ as the stabilizers S_5^X, S_6^X, S_7^X of $\mathcal{C}_{ex}$, respectively:



□

Corollary 2. When the three gauge qubits are in the $|\overline{000}\rangle$ state, $\mathcal{C}_{sub}$ is equal to $\mathcal{C}_{qrm}$.

In [3, 74], code switching is described as a *gauge fixing* process. Further afield, [94] provides a generic recipe to gauge-fix a CSS subsystem code. Here, we generalize Lemma 3 and describe how to gauge-fix $\mathcal{C}_{sub}$ to $\mathcal{C}_{ex}$ using the ZX calculus.

Proposition 3. Gauge-fixing $\mathcal{C}_{sub}$ in the following steps results in $\mathcal{C}_{ex}$, as shown in Figure 5.2.

(a) Measure three X-type gauge operators $L_{g_i}^X$ and obtain the corresponding outcomes $k_1, k_2, k_3 \in \mathbb{Z}_2$.

(b) When $k_i = 1$, the gauge qubit i has collapsed to the wrong state $|\overline{-}\rangle$. Apply the Z-type recovery operation $L_{g_i}^Z$.

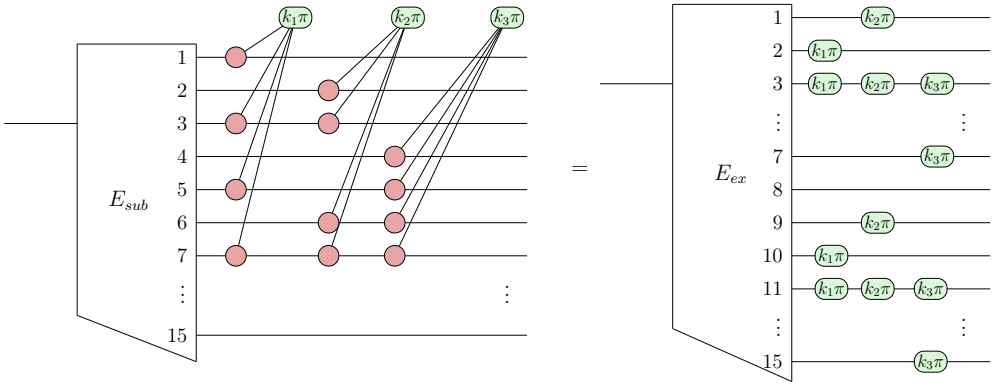


Figure 5.2: Gauge-fixing $\mathcal{C}_{sub}$ to $\mathcal{C}_{ex}$ in the circuit diagram.

Proof. In Figure 5.3, we proceed by applying a sequence of ZX rewrite rules. By Definition 5, construct the ZX normal form of E_{sub} in the blue dashed box of (i). Then the three gauge operators $L_{g_i}^X$ are measured in step (a). The subsequent equalities follow from Figures 2.2 and 2.3. Next, we observe that the purple dashed box in (iii) is exactly the encoder of the $[\![15, 4, 3]\!]$ stabilizer code. By Lemma 3.2 in [55], it can be equivalently expressed in the XZ normal form, as in (iv). By Proposition 1, pushing each Z spider with the phase $k_i\pi$ across $E_{[\![15,4,3]\!]}$ results in (v). In step (b), Pauli Z operators are applied based upon the measurement outcome k_i, which corresponds to the recovery operations in the red dashed box of (v). After that, the gauge qubits of $\mathcal{C}_{sub}$ are set to the $|\overline{+++}\rangle$ state. By Lemma 3, we obtain the XZ normal form for E_{ex}, as shown in the orange dashed box of (vi). Therefore, the equation in Figure 5.2 holds. □

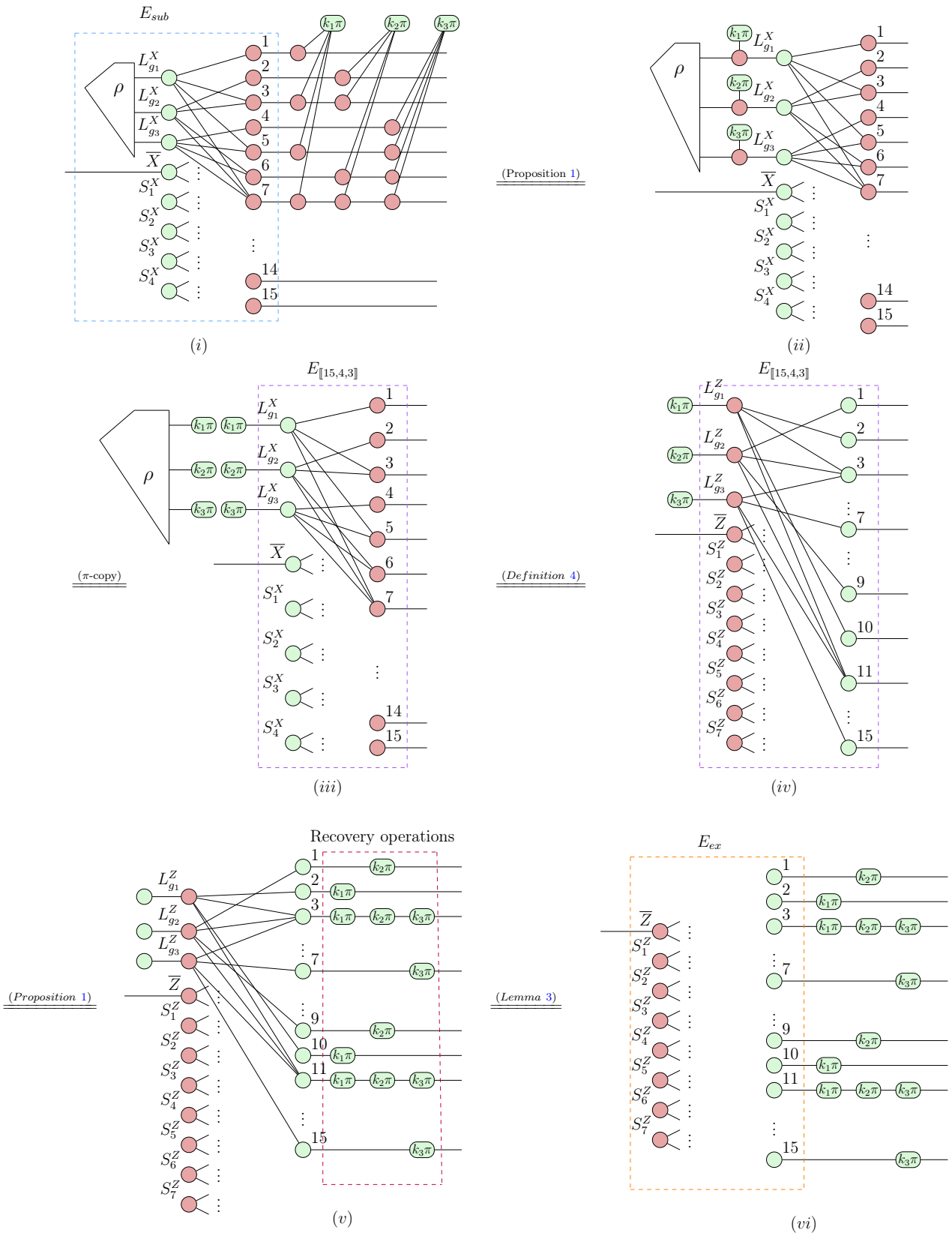


Figure 5.3: The graphical proof for Proposition 3.

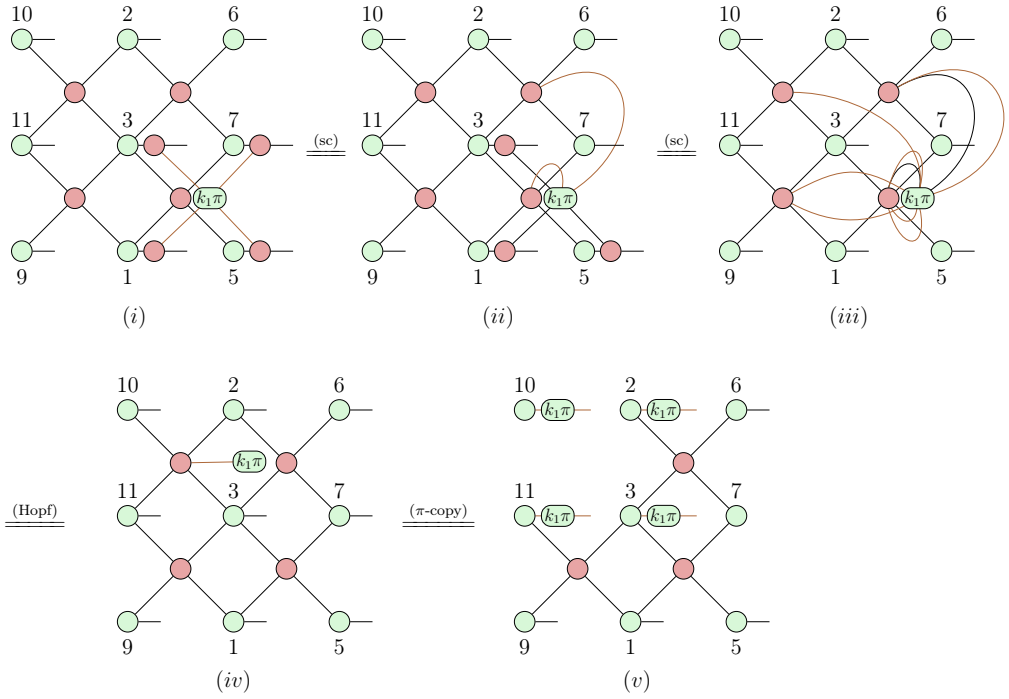


Figure 5.4: The switching from $\mathcal{C}_{qrm}$ to $\mathcal{C}_{ex}$ provides an alternative interpretation of Proposition 3. After measuring $L^X_{g_1}$, $L^Z_{g_1}$ is removed from the stabilizer group $\mathcal{S}_{qrm}$ and the recovery operation is performed based on the measurement syndrome. Note that unrelated X and Z spiders are omitted from the ZX diagrams.

We sum up by explaining how to obtain $\mathcal{C}_{ex}$ and $\mathcal{C}_{qrm}$ by gauge-fixing $\mathcal{C}_{sub}$. In Proposition 3, we showed that measuring the X-type gauge operators $L^X_{g_i}$ followed by the Z-type recovery operations $L^Z_{g_i}$ is equivalent to adding $L^X_{g_i}$ to the stabilizer group $\mathcal{S}_{sub}$. This results in the formation of $\mathcal{C}_{ex}$. Analogously, measuring the Z-type gauge operators $L^Z_{g_i}$ followed by the X-type recovery operations $L^X_{g_i}$ is equivalent to adding $L^Z_{g_i}$ to $\mathcal{S}_{sub}$. Thus, we obtain $\mathcal{C}_{qrm}$.

Alternatively, gauge-fixing $\mathcal{C}_{sub}$ can be viewed as a way of switching between $\mathcal{C}_{ex}$ and $\mathcal{C}_{qrm}$ [3, 77]. As an example, in Figure 5.4, we visualize the measurement of $L^X_{g_1} \coloneqq X_1X_3X_5X_7$ in order to switch from $\mathcal{C}_{qrm}$ to $\mathcal{C}_{ex}$. The effect of measuring other X-type gauge operators can be reasoned analogously.

By Definition 5, construct the XZ normal form of E_{qrm} in (i). Then measure $L^X_{g_1}$ and apply a sequence of rewrite rules to the ZX diagram. In (v), the stabilizer $L^Z_{g_1} \coloneqq Z_2Z_3Z_{10}Z_{11}$ is removed from the stabilizer group $\mathcal{S}_{qrm}$. Meanwhile, the recovery operation can be read off from the graphical derivation: $(Z_2Z_3Z_{10}Z_{11})^{k_1} = \left(L^Z_{g_1}\right)^{k_1}$, $k_1 \in \mathbb{Z}_2$.

Overall, ZX visualization provides a deeper understanding of the gauge fixing and code switching protocols. On top of revealing the relations between different CSS codes' encoders, it provides a simple yet rigorous test for various fault-tolerant protocols. Beyond this, it will serve as an intuitive guiding principle for the implementation of various logical operations.

Chapter 6

Conclusion

In this thesis, we generalize the notions in [55] and describe a normal form for CSS subsystem codes. Built upon the equivalence between CSS codes and the phase-free ZX diagrams, we provide a bidirectional rewrite rule to establish a correspondence between a logical ZX diagram and its physical implementation. With these tools in place, we provide a graphical representation of two code transformation techniques: code morphing, a procedure that transforms a code through unfusing spiders for the stabilizer generators, and gauge fixing, where different stabilizer codes can be obtained from a common subsystem code. These explicit graphical derivations show how the ZX calculus and graphical encoder maps relate several equivalent perspectives on these code transforming operations, allowing potential utilities of ZX to simplify fault-tolerant protocols and verify their correctness.

Looking ahead, many questions remain. It is still not clear how to present the general code deformation of CSS codes using phase-free ZX diagrams. Besides, understanding code concatenation through the lens of ZX calculus may help derive new and better codes. In addition, it would be interesting to look at other code modification techniques derived from the classical coding theory [71].

References

[1] Scott Aaronson and Daniel Gottesman. Improved simulation of stabilizer circuits. *Physical Review A*, 70(5):052328, 2004.

[2] Dorit Aharonov and Michael Ben-Or. Fault-tolerant quantum computation with constant error. In *ACM symposium on Theory of computing*, pages 176–188, 1997.

[3] Jonas T Anderson, Guillaume Duclos-Cianci, and David Poulin. Fault-tolerant conversion between the steane and reed-muller quantum codes. *Physical Review Letters*, 113(8):080501, 2014.

[4] Frank Arute, Kunal Arya, Ryan Babbush, Dave Bacon, Joseph C Bardin, Rami Barends, Rupak Biswas, Sergio Boixo, Fernando GSL Brandao, David A Buell, et al. Quantum supremacy using a programmable superconducting processor. *Nature*, 574(7779):505–510, 2019.

[5] Miriam Backens. The ZX-calculus is complete for stabilizer quantum mechanics. *New Journal of Physics*, 16(9):093021, September 2014.

[6] Darren Banfield and Alastair Kay. Implementing logical operators using code rewiring, 2022.

[7] H. Bombín and M. A. Martin-Delgado. Topological quantum distillation. *Physical Review Letters*, 97:180501, Oct 2006.

[8] H. Bombín and M. A. Martin-Delgado. Topological computation without braiding. *Physical Review Letters*, 98:160502, Apr 2007.

[9] Héctor Bombín. Clifford gates by code deformation. *New Journal of Physics*, 13(4):043005, 2011.

[10] Héctor Bombín. Gauge color codes: optimal transversal gates and gauge fixing in topological stabilizer codes. *New Journal of Physics*, 17(8):083002, August 2015.

[11] Héctor Bombín and Miguel Angel Martin-Delgado. Quantum measurements and gates by code deformation. *Journal of Physics A: Mathematical and Theoretical*, 42(9):095302, 2009.

[12] Robert I. Booth and Titouan Carette. Complete ZX-calculi for the stabiliser fragment in odd prime dimensions. ISSN: 1868-8969.

[13] Coen Borghans. *Zx-calculus and quantum stabilizer theory*. PhD thesis, Master's thesis, Radboud University, 2019.

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

[15] Sergey Bravyi, Barbara M Terhal, and Bernhard Leemhuis. Majorana fermion codes. *New Journal of Physics*, 12(8):083039, aug 2010.

[16] Hans J Briegel, David E Browne, Wolfgang Dür, Robert Raussendorf, and Maarten Van den Nest. Measurement-based quantum computation. *Nature Physics*, 5(1):19–26, 2009.

[17] A. R. Calderbank and Peter W. Shor. Good quantum error-correcting codes exist. *Physical Review A*, 54:1098–1105, Aug 1996.

[18] A Robert Calderbank, Eric M Rains, Peter W Shor, and Neil JA Sloane. Quantum error correction and orthogonal geometry. *Physical Review Letters*, 78(3):405, 1997.

[19] Earl T Campbell. The smallest interesting colour code, 2016.

[20] ChunJun Cao and Brad Lackey. Quantum lego: building quantum error correction codes from tensor networks. *PRX Quantum*, 3:020332, May 2022.

[21] Yudong Cao, Jhonathan Romero, and Alán Aspuru-Guzik. Potential of quantum computing for drug discovery. *IBM Journal of Research and Development*, 62(6):6–1, 2018.

[22] Titouan Carette, Dominic Horsman, and Simon Perdrix. SZX-calculus: scalable graphical quantum reasoning. In *44th International Symposium on Mathematical Foundations of Computer Science (MFCS 2019)*, volume 138 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 55:1–55:15, Dagstuhl, Germany, 2019. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

[23] Christopher Chamberland, Guanyu Zhu, Theodore J. Yoder, Jared B. Hertzberg, and Andrew W. Cross. Topological and subsystem codes on low-degree graphs with flag qubits. *Phys. Rev. X*, 10:011022, Jan 2020.

[24] Nicholas Chancellor, Aleks Kissinger, Joschka Roffe, Stefan Zohren, and Dominic Horsman. Graphical structures for design and verification of quantum error correction, 2023.

[25] Julien Codsi. Cutting-edge graphical stabiliser decompositions for classical simulation of quantum circuits. Master's thesis, University of Oxford, 2022.

[26] Bob Coecke and Ross Duncan. Interacting quantum observables. In *ICALP*, Lecture Notes in Computer Science, pages 298–310, 07 2008.

[27] Bob Coecke and Ross Duncan. Interacting quantum observables: categorical algebra and diagrammatics. *New Journal of Physics*, 13(4):043016, Apr 2011.

[28] Bob Coecke and Aleks Kissinger. *Picturing quantum processes*. Cambridge University Press, 2017.

[29] Bob Coecke and Quanlong Wang. Zx-rules for 2-qubit clifford+t quantum circuits. In Jarkko Kari and Irek Ulidowski, editors, *Reversible Computation*, pages 144–161, Cham, 2018. Springer International Publishing.

[30] Oliver Cole. *Quantum circuit optimisation through stabiliser reduction of Pauli exponentials*. PhD thesis, Master's thesis, University of Oxford, 2022.

[31] Cole Comfort. The algebra for stabilizer codes, 2023.

[32] Alexander Cowtan. Qudit lattice surgery, 2022.

[33] Alexander Cowtan and Simon Burton. Css code surgery as a universal construction. *arXiv preprint arXiv:2301.13738*, 2023.

[34] Alexander Cowtan, Silas Dilkes, Ross Duncan, Will Simmons, and Seyon Sivarajah. Phase Gadget Synthesis for Shallow Circuits. In Bob Coecke and Matthew Leifer, editors, *Proceedings 16th International Conference on Quantum Physics and Logic, Chapman University, Orange, CA, USA., 10-14 June 2019*, volume 318 of *Electronic Proceedings in Theoretical Computer Science*, pages 213–228. Open Publishing Association, 2020.

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

[36] Ross Duncan and Maxime Lucas. Verifying the steane code with quantomatic. *Electronic Proceedings in Theoretical Computer Science*, 171:33–49, dec 2014.

[37] Bryan Eastin and Emanuel Knill. Restrictions on transversal encoded quantum gate sets. *Physical Review Letters*, 102(11):110502, 2009.

[38] Kun Fang, Jingtian Zhao, Xiufan Li, Yifei Li, and Runyao Duan. Quantum network: from theory to practice. *arXiv preprint arXiv:2212.01226*, 2022.

[39] Liam Garvie and Ross Duncan. Verifying the smallest interesting colour code with quantomatic. *arXiv preprint arXiv:1706.02717*, 2017.

[40] Craig Gidney. A pair measurement surface code on pentagons, 2022.

[41] Craig Gidney and Austin G. Fowler. Efficient magic state factories with a catalyzed |ccz> to 2|t> transformation. *Quantum*, 3:135, apr 2019.

[42] Craig Gidney and Austin G. Fowler. Flexible layout of surface code computations using autoccz states, 2019.

[43] Jonathan Gorard, Manojna Namuduri, and Xerxes D. Arsiwalla. ZX-calculus and extended Wolfram model systems II: fast diagrammatic reasoning with an application to quantum circuit simplification. *arXiv preprint arXiv:2103.15820*, 2021.

[44] Hayato Goto. Step-by-step magic state encoding for efficient fault-tolerant quantum computation. *Scientific Reports*, 4(1):7501, Dec 2014.

[45] D Gottesman. The heisenberg representation of quantum computers. In *Group22: Proceedings of the XXII International Colloquium on Group Theoretical Methods in Physics*, pages 32–43, 6 1999.

[46] Daniel Gottesman. *Stabilizer codes and quantum error correction.* PhD thesis, California Institute of Technology, 1997.

[47] Daniel Gottesman. Theory of fault-tolerant quantum computation. *Physical Review A*, 57:127–137, Jan 1998.

[48] Daniel Gottesman. Opportunities and challenges in fault-tolerant quantum computation. *arXiv preprint arXiv:2210.15844*, 2022.

[49] Amar Hadzihasanovic, Kang Feng Ng, and Quanlong Wang. Two complete axiomatisations of pure-state qubit quantum computing. In *ACM/IEEE Symposium on Logic in Computer Science*, LICS '18, page 502–511, New York, NY, USA, 2018. Association for Computing Machinery.

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

[51] Alexander Tianlin Hu and Andrey Boris Khesin. Improved graph formalism for quantum circuit simulation. *Physical Review A*, 105(2), feb 2022.

[52] Emmanuel Jeandel, Simon Perdrix, and Renaud Vilmart. A complete axiomatisation of the zx-calculus for clifford+ t quantum mechanics. In *Proceedings of the 33rd Annual ACM/IEEE Symposium on Logic in Computer Science*, pages 559–568, 2018.

[53] Emmanuel Jeandel, Simon Perdrix, and Renaud Vilmart. Completeness of the ZX-calculus. *Logical Methods in Computer Science*, Volume 16, Issue 2, June 2020.

[54] Andrey Boris Khesin, Jonathan Z. Lu, and Peter W. Shor. Graphical quantum clifford-encoder compilers from the zx calculus, 2023.

[55] Aleks Kissinger. Phase-free zx diagrams are css codes (... or how to graphically grok the surface code). *arXiv preprint arXiv:2204.14038*, 2022.

[56] Aleks Kissinger and John van de Wetering. PyZX: Large Scale Automated Diagrammatic Reasoning. In Bob Coecke and Matthew Leifer, editors, Proceedings 16th International Conference on *Quantum Physics and Logic,* Chapman University, Orange, CA, USA., 10-14 June 2019, volume 318 of *Electronic Proceedings in Theoretical Computer Science*, pages 229–241. Open Publishing Association, 2020.

[57] Aleks Kissinger and John van de Wetering. Simulating quantum circuits with ZX-calculus reduced stabiliser decompositions. *Quantum Science and Technology*, 7(4):044001, 2022.

[58] Emanuel Knill and Raymond Laflamme. Theory of quantum error-correcting codes. *Physical Review A*, 55(2):900, 1997.

[59] Emanuel Knill, Raymond Laflamme, and Wojciech Zurek. Threshold accuracy for quantum computation. *arXiv preprint quant-ph/9610011*, 1996.

[60] David Kribs, Raymond Laflamme, and David Poulin. Unified and generalized approach to quantum error correction. *Physical Review Letters*, 94:180501, May 2005.

[61] Aleksander 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*. PhD thesis, California Institute of Technology, 2018.

[62] Aleksander Kubica and Michael E. Beverland. Universal transversal gates with color codes - a simplified approach. *Physical Review A*, 91(3):032330, March 2015.

[63] Aleksander Kubica and Michael Vasmer. Single-shot quantum error correction with the three-dimensional subsystem toric code. *Nature Communications*, 13(1):6272, October 2022.

[64] Andrew J. Landahl, Jonas T. Anderson, and Patrick R. Rice. Fault-tolerant quantum computing with color codes, 2011.

[65] Andrew J. Landahl and Ciaran Ryan-Anderson. Quantum computing by color-code lattice surgery, 2014.

[66] Andrew J Landahl and Ciaran Ryan-Anderson. Quantum computing by color-code lattice surgery. *http://www.arXiv.org and Physical Review A*, 7 2014.

[67] Adrian Lehmann, Ben Caldwell, and Robert Rand. Vyzx : A vision for verifying the zx calculus, 2022. accepted for publication.

[68] Adrian Lehmann, Ben Caldwell, and Robert Rand. VyZX: a vision for verifying the ZX calculus. *arXiv preprint arXiv:2205.05781*, 2022.

[69] Sebastian Leontica, F Tennie, and T Farrow. Simulating molecules on a cloud-based 5-qubit ibm-q universal quantum computer. *Communications Physics*, 4(1):112, 2021.

[70] Daniel Litinski. A Game of Surface Codes: Large-Scale Quantum Computing with Lattice Surgery. *Quantum*, 3:128, March 2019.

[71] F. J. MacWilliams and N. J. A. Sloane. *The theory of error-correcting codes*. Elsevier, 1977.

[72] Tommy McElvanney and Miriam Backens. Complete flow-preserving rewrite rules for mbqc patterns with pauli measurements, 2022.

[73] Michael A. Nielsen and Isaac L. Chuang. *Quantum computation and quantum information*. Cambridge University Press, 10th anniversary ed edition, 2010.

[74] Adam Paetznick and Ben W. Reichardt. Universal fault-tolerant quantum computation with only transversal gates and error correction. *Physical Review Letters*, 111:090505, Aug 2013.

[75] David Poulin. Stabilizer formalism for operator quantum error correction. *Physical Review Letters*, 95(23):230504, 2005.

[76] Boldizsár Poór, Quanlong Wang, Razin A. Shaikh, Lia Yeh, Richie Yeung, and Bob Coecke. Completeness for arbitrary finite dimensions of zxw-calculus, a unifying calculus, 2023.

[77] Dong-Xiao Quan, Li-Li Zhu, Chang-Xing Pei, and Barry C Sanders. Fault-tolerant conversion between adjacent reed–muller quantum codes based on gauge fixing. *Journal of Physics A: Mathematical and Theoretical*, 51(11):115305, 2 2018.

[78] R Raussendorf, J Harrington, and K Goyal. Topological fault-tolerance in cluster state quantum computation. *New Journal of Physics*, 9(6):199, jun 2007.

[79] Robert Raussendorf and Hans J. Briegel. A one-way quantum computer. *Physical Review Letters*, 86:5188–5191, May 2001.

[80] Robert Raussendorf and Jim Harrington. Fault-tolerant quantum computation with high threshold in two dimensions. *Physical Review Letters*, 98(19):190504, 2007.

[81] Alexis T. E. Shaw, Michael J. Bremner, Alexandru Paler, Daniel Herr, and Simon J. Devitt. Quantum computation on a 19-qubit wide 2d nearest neighbour qubit array, 2022.

[82] Peter W. Shor. Scheme for reducing decoherence in quantum computer memory. *Physical Review A*, 52:R2493–R2496, Oct 1995.

[83] Will Simmons. Relating measurement patterns to circuits via pauli flow. *Electronic Proceedings in Theoretical Computer Science*, 343:50–101, sep 2021.

[84] Andrew Steane. Multiple-particle interference and quantum error correction. *Proceedings of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences*, 452(1954):2551–2577, 1996.

[85] Andrew M Steane. Error correcting codes in quantum theory. *Physical Review Letters*, 77(5):793, 1996.

[86] Andrew M Steane. Simple quantum error-correcting codes. *Physical Review A*, 54(6):4741, 1996.

[87] Andrew M Steane. Quantum reed-muller codes. *IEEE Transactions on Information Theory*, 45(5):1701–1703, 1999.

[88] R. Tanner. A recursive approach to low complexity codes. *IEEE Transactions on Information Theory*, 27(5):533–547, 1981.

[89] Felix Tennie and Tim Palmer. Quantum computers for weather and climate prediction: the good, the bad and the noisy. *arXiv preprint arXiv:2210.17460*, 2022.

[90] Alex Townsend-Teague and Konstantinos Meichanetzidis. Classifying complexity with the zx-calculus: Jones polynomials and potts partition functions. In *Quantum Physics and Logic*, Electronic Proceedings in Theoretical Computer Science (EPTCS 287), page 313–344, Oxford, 2022.

[91] John van de Wetering. Zx-calculus for the working quantum computer scientist. *arXiv preprint arXiv:2012.13966*, 2020.

[92] Michael Vasmer and Aleksander Kubica. Morphing quantum codes. *PRX Quantum*, 3(3), aug 2022.

[93] Renaud Vilmart. A near-minimal axiomatisation of zx-calculus for pure qubit quantum mechanics. In *LICS 2019*, pages 1–10, 2019.

[94] Christophe Vuillot, Lingling Lao, Ben Criger, Carmen García Almudéver, Koen Bertels, and Barbara M Terhal. Code deformation and lattice surgery are gauge fixing. *New Journal of Physics*, 21(3):033028, 2019.

[95] Quanlong Wang. Qutrit ZX-calculus is complete for stabilizer quantum mechanics. *Electronic Proceedings in Theoretical Computer Science*, 266:58–70, February 2018.

[96] George Watkins, Hoang Minh Nguyen, Varun Seshadri, Keelan Watkins, Steven Pearce, Hoi-Kwan Lau, and Alexandru Paler. A high performance compiler for very large scale surface code computations, 2023.

[97] Paul Webster and Stephen D Bartlett. Braiding defects in topological stabiliser codes of any dimension cannot be universal. *arXiv preprint arXiv:1811.11789*, 2018.

[98] Paul Webster, Michael Vasmer, Thomas R. Scruby, and Stephen D. Bartlett. Universal fault-tolerant quantum computing with stabilizer codes. *Phys. Rev. Res.*, 4:013092, Feb 2022.

[99] Theodore J. Yoder. Universal fault-tolerant quantum computation with Bacon-Shor codes, May 2017.

[100] Qingling Zhu, Sirui Cao, Fusheng Chen, Ming-Cheng Chen, Xiawei Chen, Tung-Hsun Chung, Hui Deng, Yajie Du, Daojin Fan, Ming Gong, et al. Quantum computational advantage via 60-qubit 24-cycle random circuit sampling. *Science bulletin*, 67(3):240–245, 2022.

Appendix A

The Complementary Proof

A.1 The ZX and XZ Normal Forms are Equivalent for a CSS Code

An $[\![n, k, d]\!]$ CSS code $\mathcal{C}$ is generated by orthogonal subspaces $V = \mathsf{Span}\{v_1, \ldots, v_{m_1}\}$ and $W = \mathsf{Span}\{w_1, \ldots, w_{m_2}\}$. $V,\ W \subset \mathbb{F}_2^n$, $m_1 = \dim V$ and $m_2 = \dim W$. m_1 and m_2 denote the number of independent X-type and Z-type stabilizer generators respectively. Then, the stabilizer generators of $\mathcal{C}$ can be expressed as

$$S_i^X = \bigotimes_{t=1}^{n} X^{(v_i)_t}, \quad S_j^Z = \bigotimes_{t=1}^{n} Z^{(w_j)_t}.$$

Let $\mathcal{S}$ be the stabilizer group of $\mathcal{C}$. Then

$$\mathcal{S} = \left\langle S_i^X, S_j^Z; 1 \leq i \leq m_1, 1 \leq j \leq m_2 \right\rangle.$$

Let E be the encoder of $\mathcal{C}$. Since E is an isometry from the k logical qubit space to the n physical qubit space, $E^\dagger E = I_k$ and $k = n - m_1 - m_2$ [46]. When a CSS code is not maximal, $m_1 + m_2 < n$. Then $k > 0$ and its logical operators can be expressed as

$$\overline{X}_i = \bigotimes_{t=1}^{n} X^{(v_i)_t}, \quad m_1 \leq i \leq m_1 + k$$

$$\overline{Z}_j = \bigotimes_{t=1}^{n} Z^{(w_j)_t}, \quad m_2 \leq j \leq m_2 + k$$

When a CSS code is *maximal*, $\mathcal{S}$ is maximal and $V \oplus U = \mathbb{F}_2^n$. Thus $m_1 + m_2 = n$. $k = 0$ means that there is no degree of freedom for the encoded logical information. Therefore, the maximal CSS code is uniquely determined by $\mathcal{S}$. More precisely, it is the joint $+1$ eigenstate of the n independent stabilizer generators.

In Appendix A.1.1, we visualize a stabilizer generator of a CSS code using the ZX diagrams. In Appendix A.1.2, we explain the map-duality of a ZX diagram. This allows us to move spiders around and bend wires. In Appendix A.1.3, we leverage this property and prove Proposition 4, which states that it is sufficient to represent a CSS code using one of its ZX and XZ normal forms.

A.1.1 Represent a Stabilizer Generator Using the ZX Diagrams

In [55], a pictorial way was introduced to represent non-destructive Pauli measurements.

$$\mathcal{M}_{X\dots X} = \left\{ \prod_{X\dots X}^{k} \coloneqq \frac{1}{2}(I + (-1)^k X \otimes \cdots \otimes X) \right\}_{k=0,1}$$

$$\mathcal{M}_{Z\dots Z} = \left\{ \prod_{Z\dots Z}^{k} \coloneqq \frac{1}{2}(I + (-1)^k Z \otimes \cdots \otimes Z) \right\}_{k=0,1}$$

Definition 9. *When $k = 0$, we obtain a phase-free version of the diagrams in Figure 2.3. They represent the X- and Z-type stabilizer generators, which are the projections onto the $+1$ eigenspace of a Pauli operator.*

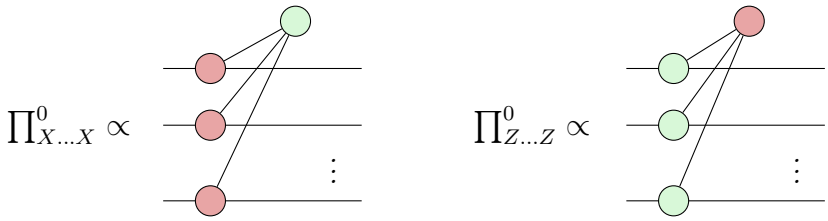
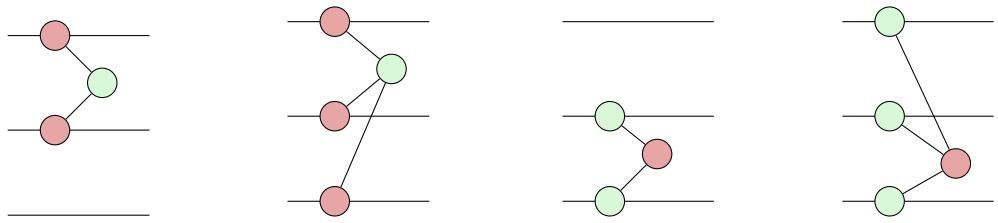


Figure A.1 use four examples to illustrate this notion. We can read off a Pauli projection from each ZX diagram. In Figure A.1a, since the X spiders are supported on the first two qubits, Pauli X acts on them and we get $I + XXI$. In Figure A.1b, since the X spiders are supported on all three qubits, we get $I + XXX$. Similarly, in Figures A.1c and A.1d, Pauli Z acts on qubits where Z spiders are supported.



(a) $I + XXI$ (b) $I + XXX$ (c) $I + IZZ$ (d) $I + ZZZ$

Figure A.1: Up to a scalar, each projection in $\{I + XXI, I + XXX, I + IZZ, I + ZZZ\}$ is represented by Figures A.1a to A.1d respectively. Here, I is short for $I^{\otimes 3}$.

A.1.2 The Map-State Duality of a ZX Diagram

Next, we introduce wires that are commonly used in a ZX diagram. In Figure A.2, a single wire corresponds to a one-qubit identity operator. A crossing wire corresponds to a SWAP operation. Wires in the shape of a cup or cap allow us to "bend" inputs around to become outputs and vice-versa. This corresponds to performing the partial transpose in the computational basis.

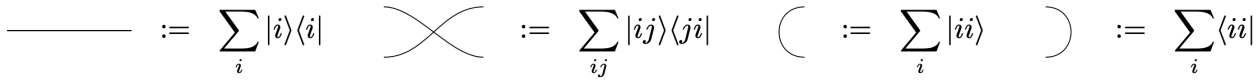


Figure A.2: In addition to spiders, a ZX diagram could contain identity wires, swaps, cups, and caps [55].

Cups and caps help establish a bijection between a linear map $E : \mathbb{C}_2^{\otimes k} \to \mathbb{C}_2^{\otimes n}$ and the state $|E\rangle \in \mathbb{C}_2^{\otimes(k+n)}$, as shown in Figure A.3. This property is known as the *map-state duality*. It allows us to deform a ZX diagram arbitrarily by moving spiders around, bending, and unbending the wires. So long as the order of the diagram's inputs and outputs is preserved, the deformed ZX diagram can be recovered to represent the same matrix [91].

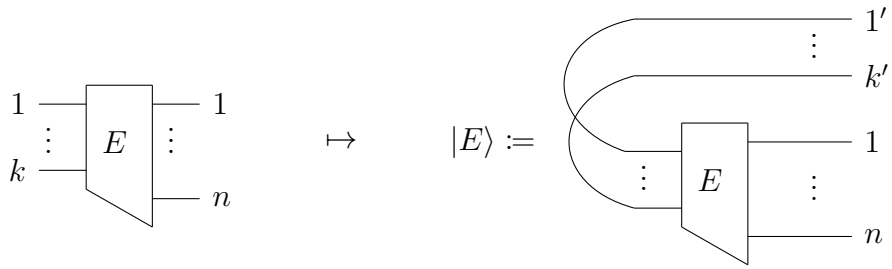


Figure A.3: Given some linear map E from k qubits to n qubits, we can transform it into a $(k+n)$-qubit state $|E\rangle$ by applying the Choi-Jamiołkowski isomorphism.

A.1.3 The ZX and XZ Normal Forms for a CSS Code

Recall in [55], a CSS code permits a ZX normal form E_X which is constructed from the X-type stabilizer generators and the logical X operators. It also permits an XZ normal form E_Z which is constructed from the Z-type stabilizer generators and the logical Z operators.

We start by considering the maximal CSS code, which is uniquely determined by the X- and Z-type stabilizer generators.

Lemma 4. *For any maximal CSS code $|E\rangle$ over n physical qubits, its ZX normal form E_X is equivalent to its XZ normal form E_Z. In other words, E_X and E_Z correspond to the same linear map.*

Proof. Let $\mathcal{S}$ be the maximal stabilizer group that uniquely fixes $|E\rangle$,

$$\mathcal{S} = \langle S_1^X, \ldots, S_{m_1}^X, S_1^Z, \ldots, S_{m_2}^Z \rangle, \quad 0 \le m_1, m_2 \le n, \quad m_1 + m_2 = n.$$

By Definition 4,

$$E_X = \prod_{i=1}^{m_1} \Big(I + S_i^X\Big)|0\rangle^{\otimes n}, \quad E_Z = \prod_{j=1}^{m_2} \Big(I + S_j^Z\Big)|+\rangle^{\otimes n}.$$

Note that $Z|0\rangle = |0\rangle$ and thus $S_j^Z|0\rangle^{\otimes n} = |0\rangle^{\otimes n}$. For all $1 \leq j \leq m_2$, since $\left[S_j^Z, S_i^X\right] = 0$ for all $1 \leq i \leq m_1$, we have

$$\begin{aligned} S_j^Z E_X &= S_j^Z \left(\prod_{i=1}^{m_1} \Big(I + S_i^X\Big)|0\rangle^{\otimes n} \right) \\ &= \prod_{i=1}^{m_1} S_j^Z \Big(I + S_i^X\Big)|0\rangle^{\otimes n} \\ &= \prod_{i=1}^{m_1} \Big(I + S_i^X\Big) S_j^Z |0\rangle^{\otimes n} \\ &= \prod_{i=1}^{m_1} \Big(I + S_i^X\Big)|0\rangle^{\otimes n} = E_X. \end{aligned}$$

Hence, E_X is stabilized by every Z-type stabilizer generator of $\mathcal{S}$. It follows that E_X is in the joint $+1$ eigenspace of $\mathcal{S}$.

Note that $X|+\rangle = |+\rangle$ and thus $S_i^X|+\rangle^{\otimes n} = |+\rangle^{\otimes n}$. Reasoning analogously, we have

$$\begin{aligned} S_i^X E_Z &= S_i^X \left(\prod_{j=1}^{m_2} \Big(I + S_j^Z\Big)|+\rangle^{\otimes n} \right) \\ &= \prod_{j=1}^{m_2} S_i^X \Big(I + S_j^Z\Big)|+\rangle^{\otimes n} \\ &= \prod_{j=1}^{m_2} \Big(I + S_j^Z\Big) S_i^X |+\rangle^{\otimes n} \\ &= \prod_{j=1}^{m_2} \Big(I + S_j^Z\Big)|+\rangle^{\otimes n} = E_Z. \end{aligned}$$

Hence, E_Z is stabilized by every X-type stabilizer generator of $\mathcal{S}$. It follows that E_Z is in the joint $+1$ eigenspace of $\mathcal{S}$. By the uniqueness of the maximal CSS code, $E_X = E_Z = |E\rangle$. □

As an exercise, we use Example 8 to illustrate this proof.

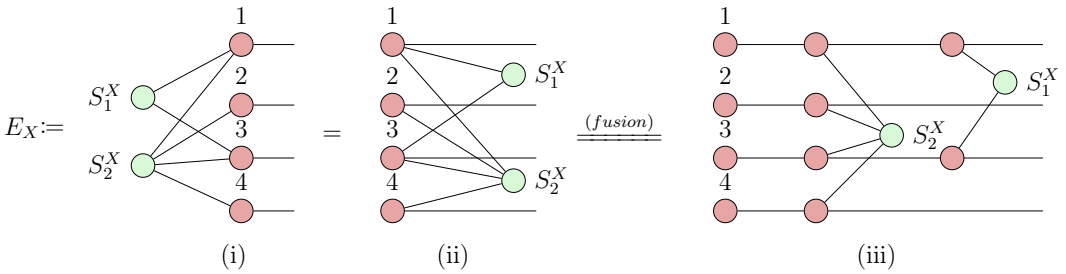
Example 8. *Consider a maximal stabilizer group* $\mathcal{S} = \langle S_1^X, S_2^X, S_1^Z, S_2^Z \rangle$*, where*

$$S_1^X = X_1X_3, \quad S_2^X = X_1X_2X_3X_4, \quad S_1^Z = Z_2Z_4, \quad S_2^Z = Z_1Z_2Z_3Z_4.$$

Let $|E\rangle$ *be the maximal CSS code uniquely fixed by* $\mathcal{S}$*. Let* E_X *and* E_Z *be its ZX and XZ normal forms respectively. Then* $E_X = E_Z$*.*

Proof. According to Definition 4, E_X is defined by the X-type stabilizer generators. In Figure A.4, after relocating the Z spiders, E_X in (i) is transformed into a semantically equivalent diagram in (ii). After unfusing the wires and applying Definition 9,

$$E_X = \sum_{b \in V} |b\rangle, \quad V = \mathsf{Span}\{v_1, v_2\}, \quad v_1 = \begin{bmatrix} 1 \\ 0 \\ 1 \\ 0 \end{bmatrix}, \quad v_2 = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}.$$

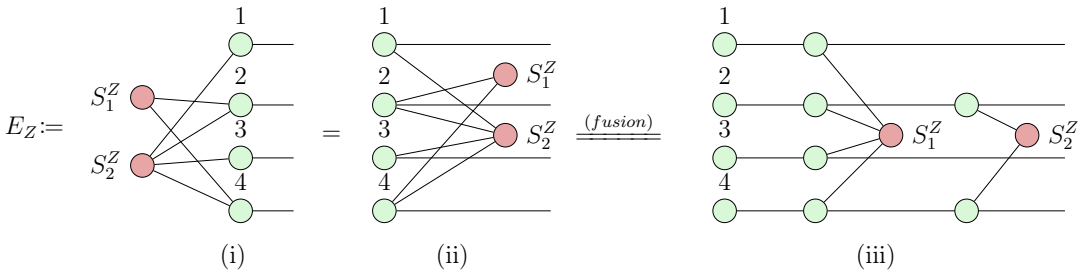


$$\overset{Definition}{=\!=\!=} (I + S_2^X)(I + S_1^X)|0000\rangle = \textstyle\sum_{i,j=0}^{1} (S_2^X)^j (S_1^X)^i |0000\rangle = \sum_{b \in V} |b\rangle$$

Figure A.4: The ZX normal form of the maximal CSS code fixed by $\mathcal{S}$. The equalities hold up to a scalar.

Similarly, E_Z is defined by the Z-type stabilizer generators. In Figure A.5, after relocating the X spiders, E_Z in (i) is transformed into a semantically equivalent diagram in (ii). After unfusing the wires and applying Definition 9,

$$E_Z = \sum_{b \in W} |b\rangle, \quad W = \mathsf{Span}\{w_1, w_2\}, \quad w_1 = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}, \quad w_2 = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 1 \end{bmatrix}.$$



$$\overset{Definition}{=\!=\!=} (I + S_2^Z)(I + S_1^Z)|++++\rangle = \textstyle\sum_{i,j=0}^{1} (S_2^Z)^j (S_1^Z)^i |++++\rangle = \sum_{b \in W} |b\rangle$$

Figure A.5: The XZ normal form of a maximal CSS code fixed by $\mathcal{S}$. The equalities hold up to a scalar.

By direct computation, we can show that up to a scalar, $E_Z = E_X$:

$$\begin{aligned}
E_Z &= (I + S_2^Z)(I + S_1^Z)|++++\rangle \\
&= (I + S_2^Z)(I + S_1^Z) \sum_{v \in \mathbb{F}_2^4} |v\rangle \\
&= (I + S_2^Z) \sum_{v \perp w_1} |v\rangle, \quad \text{where } S_1^Z = \bigotimes_{t=1}^{n} Z^{(w_1)_t} \\
&= \sum_{v \perp w_1, v \perp w_2} |v\rangle, \quad \text{where } S_2^Z = \bigotimes_{t=1}^{n} Z^{(w_2)_t} \\
&= \sum_{c \perp c'} |c\rangle, \quad \forall c' \in W, \quad \text{where } W = \mathsf{Span}\{w_1, w_2\} \\
&= \sum_{b \in V} |b\rangle, \quad \text{where } V = \mathsf{Span}\{v_1, v_2\} \\
&= E_X.
\end{aligned}$$

□

Proposition 4. For any non-maximal CSS code $\mathcal{C}$ over n physical qubits, its ZX normal form E_X is equivalent to its XZ normal form E_Z. In other words, E_X and E_Z correspond to the same linear map.

Proof. Suppose towards contradiction that E_X and E_Z correspond to different linear maps. After bending the input wires of E_X and E_Z and turning them into the output wires, we obtain a maximal CSS code whose ZX and XZ normal forms are E'_X and E'_Z respectively. Since $E_X \neq E_Z$, $E'_X \neq E'_Z$. By Lemma 4, $E'_X = E'_Z$. This yields a contradiction. Therefore, $E_X = E_Z$. □