Hypergraph MBQC in the ZH-calculus

Yanbin Chen

A thesis submitted for the degree of

Master of Science in Computer Science

Trinity Term 2021

Acknowledgements

I wish to thank Aleks Kissinger for his inspirational ideas and willingness to supervise, and John van de Watering for his generous help in conducting this study, various suggestions and proofreading.

Abstract

Quantum computing studies how to perform computations within quantum systems where certain computations are much more efficient than in classical computers. Whereas we can describe quantum computations using an analogue of classical logic circuits, measurement-based quantum computation (MBQC) models focus on measuring qubits in an intricate resource state. The ZH-calculus provides a standard diagrammatic way to represent and reason about MBQC. In this thesis, we use the ZH-calculus to prove correctness of MBQC protocols, investigate the link between hypergraph and phase gadget state MBQC, construct a new MBQC model that achieves universal computations with a deterministic protocol, and prove correctness of our construction. First, we use the ZH-calculus to justify MBQC protocols of two hypergraph states: the GGM state presented by Gachechiladze et al., and the Union Jack state from Miller et al. Next, we study the link between hypergraph and phase gadget MBQC by applying Graphical Fourier theorem. Finally, we present a new MBQC model, and use the ZH-calculus to show that our MBQC model not only implements deterministic computations by allowing for error corrections in measurement patterns, but also achieves universal computations using only single-qubit Pauli X and Z measurements. Our proofs for protocols yield intuitive insights into hypergraph MBQC, our finding about the connection between hypergraph and phase gadget states may be used in future studies of relations between MBQC models, and the new construction of resource state provides an alternative idea to deterministically achieve universality in MBQC.

Contents

1 Introduction

1.1 Introduction

Quantum computing is a study of computations in quantum systems [1]. One of the reasons why people have been investing efforts into the study of quantum computing is that quantum computers can efficiently perform some tasks for which there are no known efficient algorithms on classical computers [2]. Measurement-based quantum computation (MBQC) describes an idea to implement quantum computations. In MBQC, computations are implemented by performing quantum measurements on a highly entangled resource state prepared in advance. One of the most well-studied MBQC models is the one-way model [3]. In the one-way model, a graph state is prepared as the resource state, where graph states are a well-know family of states taking the form of simple undirected graphs [4–7]. Then, single-qubit measurements are performed on the graph state to implement computations.

In this thesis, we focus on the following four aspects of MBQC: *universality*, *determinism*, *parallelism*, and *measurements*. For *universality*, we are interested in whether a MBQC model achieves universal computations. Since MBQC is a way to implement computations, it is natural to consider the set of computations that can be represented by the resource state and measurements performed on it. Ideally, we want this set to include the whole set of quantum computations we

care about. For example, the MBQC models proposed in [8], [9], [10], [11] and [12] can all achieve universal computations. For *determinism*, the problem we care about is whether a MBQC model has a deterministic protocol. Measurements cannot give deterministic outcomes, so undesired outcomes of measurements known as errors are introduced in the process of MBQC. Therefore, in order to provide a deterministic protocol and implement computation in a deterministic way, we need a scheme to correct measurement errors. For example, the MBQC model in [10] takes the idea of feed-forward to correct errors and achieves determinism. For *parallelism*, we want to know about the extent to which we can parallelize computations. In MBQC, it is normal that some measurements depend on others, so there may be constraints on the sequence of computations implemented in MBQC, which potentially influences parallelism of computations. For instance, the MBQC model proposed in [12] cannot implement all CCZ gates and Hadamard gates in parallel, because CZ errors cannot freely pass through Hadamard gates and have to be corrected before implementations of Hadamard gates. For *measurements*, we are interested in what kind of measurements are needed in a MBQC protocol. For example, if we only allow Pauli measurements in the one-way model, then by Gottesman-Knill theorem [13] universality cannot be achieved. We need non-Pauli measurements to achieve the full computational power of this model.

Resource states are of importance to MBQC. There are different ideas to construct resource states. In this thesis, we focus on two families of resource states: hypergraph states and phase gadget states. Hypergraphs are generalized simple undirected graphs. While the edge of a simple undirected graph only contains two vertices, the edge of a hypergraph is called a hyperedge and may contain more than two vertices. Hypergraph states are a family of states taking the form of hypergraphs, and a hypergraph state is implemented by preparing a qubit for each vertex and applying a C^nZ gate for each hyperedge [4]. The GGM state [12] and the Union Jack state [11] are both hypergraph states, and we sometimes call the MBQC model based on hypergraph states as *hypergraph MBQC* or *hypergraph state MBQC*. Phase gadget

states are a family of resource states which also have a structure of hypergraph. However, in phase gadget states each hyperedge corresponds to a phase gadget rather than a C^nZ gate. Phase gadgets [14] are special quantum gates, and they entangle qubits in a different way from C^nZ gates. We sometimes call the MBQC model based on phase gadget states as *phase gadget MBQC* or *phase gadget state MBQC*.

The ZX-calculus is a graphical language presented by Coecke and Duncan as a reasoning tool for quantum computations [15]. In [15], Duncan has shown the use of ZX-calculus in reasoning about MBQC. ZH-calculus presented in [16] is one of the variants of ZX-calculus. Compared to ZX-calculus, ZH-calculus makes it easier to reason about systems that contain C^nZ gates, the elements for hypergraph resource states. In [17], Graphical Fourier theory is presented and it links ZH-calculus with ZX-calculus by providing diagrammatic translation between these two graphical languages.

The aim of this thesis is threefold:

(i) Use ZH-calculus to give diagrammatic justifications for hypergraph state MBQC models in [11, 12]. Although justifications are given in [11, 12] for GGM state and Union Jack state based MBQC, they are neither intuitive nor easy to verify. In particular, for the GGM state model, the explanations given in [12] for measurement patterns deterministically implementing CZ wires involve a lot of matrix notations and text descriptions and thus are neither straightforward nor intuitive, and some explanations for transformations of states are missing. For the Union Jack state model, the measurement pattern implementing a CCZ gate contains many sub-measurement patterns and complex connections between them, but the justification in [11] for this measurement pattern is brief and not intuitive. Although diagrammatic justifications for the sub-measurement patterns are given in [18], there is no intuitive explanations for why these patterns compose together to implement a CCZ gate. So, we aim to give more intuitive justifications, and our contribution can increase interpretability and explainability of hypergraph MBQC.

(ii) Investigate the relation between hypergraph and phase gadget state MBQC. Although hypergraph states and phase gadget states both have a structure of hypergraphs, the relation between hypergraph state MBQC and phase gadget state MBQC has not yet been studied up to the time of this writing. Our contribution shows that these two families of MBQC are not totally independent to each other, and provides motivations for future study of relation between MBQC models.

(iii) Create a new MBQC model that achieves universal computations with a deterministic protocol using only Pauli X and Z measurements. Our construction can be seen as an application of the link between hypergraph and phase gadget state MBQC. Besides, the new MBQC model provides an alternative idea to achieve universality and correct measurement errors.

Related work In [10], a phase gadget state based MBQC model is presented where only Pauli X and Z measurements are needed to achieve deterministic universal computations. The authors apply the technique of feed-forward to correct measurement errors and implement deterministic computations. The authors develop a notation for depicting precisely how each measurement pattern implements its target gate and how it is adapted according to incoming Pauli Z and X errors to admit feed-forward. The resource state in this model is constructed by binary phase gadgets, while ours presented in chapter 5 is constructed by trinary phase gadgets.

In [18], the author uses ZH-calculus to show that measurement patterns in [11] can implement SWAP gates and special U_I gates. We will base our justification for the protocol of the Union Jack state model on the results in [18]. The author also provides diagrammatic descriptions and justifications of the action of Pauli measurements on quantum hypergraph states and hypergraph transformation rules, which are useful in studies of hypergraph MBQC.

1.2 Outline

In chapter 2, we give background information on quantum computing, ZX-calculus, ZH-calculus, and MBQC, and provide necessary preliminaries required to understand this thesis. We focus on introductions of the two graphical languages and MBQC, since all proofs in this thesis are described in ZH-calculus and MBQC is our topic.

In chapter 3, we use ZH-calculus to investigate the two MBQC models in [12] and [11]. For the GGM state based MBQC, we first define hexagon notations that make it easier to present the state, then justify measurement patterns defined on the GGM state, and finally prove the correctness of the protocol of the GGM model. For the Union Jack state based MBQC, we first briefly introduce the Union Jack state, and then use ZH-calculus to show that CCZ gates can be implemented by the measurement pattern presented in [11].

In chapter 4, we apply ZH-calculus and Graphical Fourier theorems to show the equivalence between the Union Jack state and a phase gadget state, which both serves as a motivation for our construction of a new resource state, and indicates a link between hypergraph and phase gadget state MBQC.

In chapter 5, we present a new MBQC model that achieves universal computations with a deterministic protocol via only Pauli X and Z measurements. First we introduce notations of boxes and ropes. Then we present the new resource state and define measurement patterns on fragments of it. Next, we use ZH-calculus to give justifications for the measurement patterns. Then we describe and justify how we use the technique of feed-forward to correct measurement errors in our MBQC model. Finally, we show that universal computations can be deterministically achieved by our MBQC model.

In chapter 6, we conclude our results and discuss some possible future work that might be of interest to readers.

2
Preliminaries

In section 2.1, we introduce basic concepts in quantum computing, namely qubits, quantum measurements and quantum gates, which will serve as basics to understand ZX-calculus, ZH-calculus, and MBQC. In sections 2.2 and 2.3, we give an informal introduction to ZX-calculus, a graphical language, and one of its most important variants, ZH-calculus, which will be used as a tool of reasoning in this thesis. In sections 2.4 and 2.5, phase gadgets and hypergraph states are introduced using ZH-calculus. They play important roles in MBQC models as resource states. In section 2.6, we introduce two special cases of Graphical Fourier theorems which build a bridge between phase gadgets and H-boxes. In section 2.7, we informally discuss about what universality of computations means and one idea to show that some quantum computation model can achieve universal computations. In sections 2.8 and 2.9, we introduce two models of quantum computation, the circuit model and the MBQC model. Our introduction to circuit models only serves as a contrast to MBQC models, since we focus on MBQC in this thesis.

2.1 Quantum computing

Qubits The *qubit* is an abbreviation for the *quantum qubit*, it is created as an analogy with the concept of a *bit* in classical computation [2]. However, unlike

a classical bit, the state of a qubit is represented as a vector in two-dimensional complex vector space. We define

$$|0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \quad |1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

then we can represent a qubit as [2]:

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$$

where the notation $|\cdot\rangle$ is used to indicate that the object is a column vector, $\alpha, \beta \in \mathbb{C}$, $|0\rangle$ and $|1\rangle$ are called computational basis states of the single-qubit system. For a column vector $|a\rangle$, we use $\langle a|$ to denote the conjugate transpose of $|a\rangle$. For vectors $|\phi\rangle$ and $|\psi\rangle$, we define $\langle\phi|\psi\rangle$ as the inner product of $|\phi\rangle$ and $|\psi\rangle$. The following two single-qubit states are well-known:

$$|+\rangle = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 \\ 1 \end{pmatrix}, \quad |-\rangle = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 \\ -1 \end{pmatrix}$$

In a general case, consider a n-qubit system. This system has 2^n computational basis states, namely $|00..0\rangle$, $|00..1\rangle$, ..., $|11..0\rangle$, $|11..1\rangle$, each of which is of the form $|x_1x_2...x_n\rangle$ [2]. The state of the n-qubit system is expressed as

$$|\psi\rangle = \alpha_{00...0}|00..0\rangle + \alpha_{00...1}|00..1\rangle + ... + \alpha_{11...0}|11..0\rangle + \alpha_{11...1}|11..1\rangle$$

where $\alpha_{00...0}, \alpha_{00...1}, ..., \alpha_{11...0}, \alpha_{11...1} \in \mathbb{C}$.

Quantum measurements For a qubit in the state $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$, to determine the value of α and β, we need to perform *measurements* on this qubit, since we cannot directly examine their values [1]. We can perform measurements with respect to any computational basis. For example, if we measure the state $|\psi\rangle$ with respect to the basis $|0\rangle$ and $|1\rangle$, we will get the outcome 0 with probability $|\alpha|^2$ and the outcome 1 with probability $|\beta|^2$ [2]. It is required that $|\alpha|^2 + |\beta|^2 = 1$. For a general case, if we measure $|\psi\rangle$ with respect to an orthonormal basis $|a\rangle$ and $|b\rangle$, and we express $|\psi\rangle$ as $|\psi\rangle = \alpha|a\rangle + \beta|b\rangle$, then we will get outcome a with probability $|\alpha|^2$ and the outcome b with probability $|\beta|^2$ [2].

Quantum gates *Quantum gates* or *gates* take a state and transform it to another state. Since states are represented by column vectors, gates are represented by matrices [2]. We also use the term *operation* to call a gate. For example, we can use two by two matrices to represent single-qubit gates. A NOT gate can be expressed as

$$\text{NOT} \equiv \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$$

In general, a n-qubit gate is represented by a n by n matrix. For example, CNOT gates, CZ gates, and SWAP gates are important two-qubit gates, and they have the following matrix representations:

$$\text{CNOT} \equiv \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}, \ \text{CZ} \equiv \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \end{pmatrix}, \ \text{SWAP} \equiv \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

However, not all matrices can be interpreted as quantum gates. For a matrix M, it can be interpreted as a quantum gate only if M is *unitary* [2]. A matrix U is unitary if $U^\dagger U = I$ where I is the identity matrix and $U^\dagger$ is the adjoint of U. Here, we give matrix representations of the Hadamard gate and the phase gate, which are important single-qubit gates that appear quite often in our latter discussions:

$$\text{Hadamard} \equiv \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}, \quad \phi \text{ phase} \equiv \begin{pmatrix} 1 & 0 \\ 0 & e^{i\phi} \end{pmatrix}$$

The *S gate* is a $\frac{\pi}{2}$ phase gate, the *T gate* is a $\frac{\pi}{4}$ phase gate, the $T^\dagger$ *gate* is a $-\frac{\pi}{4}$ phase gate, the *identity gate*, *identity operation* or the *identity wire* is a 0 phase gate, the *Pauli Z gate* is a π phase gate. By an identity operation, we sometimes mean a multi-qubit gate whose matrix representation is an identity matrix. Since the Pauli X and Pauli Z gates or operations are commonly used in this thesis, we give their matrices as below:

$$\text{Pauli X} \equiv \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \quad \text{Pauli Z} \equiv \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$$

There is a useful notation for linear operators [2]: Suppose $|v\rangle$ is a vector in an inner product space V and $|w\rangle$ is a vector in an inner product space W, we

define the linear operator $|w\rangle\langle v|$ from V to W as

$$(|w\rangle\langle v|)(|v'\rangle) \equiv |w\rangle\langle v|v'\rangle = \langle v|v'\rangle|w\rangle$$

For a n-qubit gate M on qubits x_1, x_2, ..., x_n, we use the notation $M^{(x_1,x_2,...,x_n)}$ to denote it.

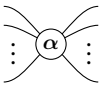
Controlled unitaries A *controlled unitary* U is a gate with a *control qubit* and *target qubits*, and it works as the following way [2]: if the control qubit is set to 0, then the identity operation will be performed on target qubits; otherwise the control qubit is set to 1, then the gate U will be performed on target qubits. For a controlled unitary U, its control qubit acts as a switch: the gate U takes effect on target qubits if and only if the switch is on (control bit is 1). For example, the CNOT gate and CZ gate are both controlled unitaries. In fact, their full names are the *controlled-NOT gate* and the *controlled-Z gate* respectively: if the control bit is 1, then the CNOT gate performs a NOT gate on the target qubit, the CZ gate performs a Pauli Z gate on the target qubit; if the control bit is 0, then the CNOT gate or CZ gate performs an identity gate on the target qubit.

2.2 ZX-calculus

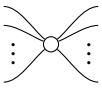
ZX-calculus is a graphical language introduced by Bob Coecke and Ross Duncan [19], and it has been known as one of the most well-known tools for tasks of derivations about multi-qubit systems in the area of quantum computing. Readers can find a detailed introduction to ZX-calculus in [1]. This graphical language consists of *ZX-diagrams* and their *rewrite rules*. A ZX-diagram is generated from a set of generators. We shortly introduce basic generators, namely white and grey spiders, in ZX-calculus.

White spiders This generator is referred to as the *white spider* or *Z-spider*. A white spider can have any number inputs and any number of outputs. By inputs and outputs, we mean input wires and output wires respectively, and if we draw

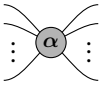
inputs as wires coming in from the left and outputs as wires coming out of the right, we can represent a white spider as:



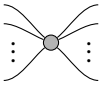
α is called the phase of the spider. When $\alpha = 0$ we drop the symbol and write a white spider as:



Grey spiders This generator is referred to as the *grey spider* or *X-spider*.

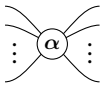


Similarly, α is called the phase of the spider. When $\alpha = 0$ we drop the symbol and write a grey spider as:

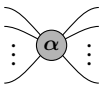


For simplicity, we sometimes call a spider with n inputs and m outputs as a $(n+m)$-arity spider or $(n+m)$-ary spider.

One of the most important reasons why ZX-calculus is useful for derivations about qubit systems is that we can interpret ZX-diagrams as linear maps or matrices, and what is more amazing, every linear map can be represented by a ZX-diagram [14]. The basic generators, white and grey spiders, are themselves ZX-diagrams, and we give the matrix interpretations for white and grey spiders as follows:



$$:= \quad |0\cdots0\rangle\langle 0\cdots0| + e^{i\alpha}\,|1\cdots1\rangle\langle 1\cdots1|\,,$$



$$:= \quad |+\cdots+\rangle\langle +\cdots+| + e^{i\alpha}\,|-\cdots-\rangle\langle -\cdots-|\,.$$

Next, we talk about how to put these generators together to form a ZX-diagram.

Compositions Two operations are needed to generate a ZX-diagram: the sequential composition and parallel composition. The *sequential composition* of two diagrams means connecting the outputs of one diagram to the inputs of another. The *parallel composition* of two diagrams is to place them side by side. Readers can read [1] for a detailed description of sequential and parallel compositions. We use symbol $\circ$ to denote sequential compositions and symbol $\otimes$ to denote parallel compositions. For a ZX-diagram D, we use $[\![D]\!]$ to denote the linear map representation of it. For ZX-diagrams D_1 and D_2, we have

$$[\![D_1 \circ D_2]\!] = [\![D_1]\!] \cdot [\![D_2]\!],\ [\![D_1 \otimes D_2]\!] = [\![D_1]\!] \otimes [\![D_2]\!]$$

where in the second equation, the $\otimes$ symbol on the right-hand side represents Kronecker product.

Symmetries One important property of ZX-diagrams is that we can treat ZX-diagrams as undirected graphs. In other words, only connectivity matters in ZX-diagrams. For example, see figure 2.1. This property is thoroughly discussed in [1] and [14].

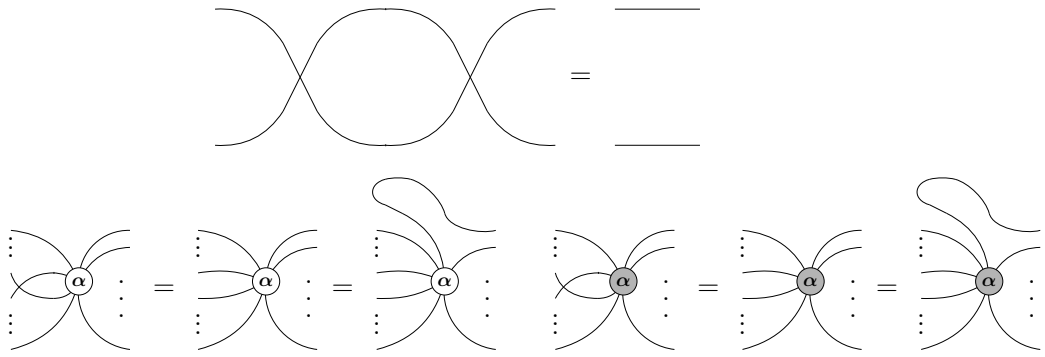
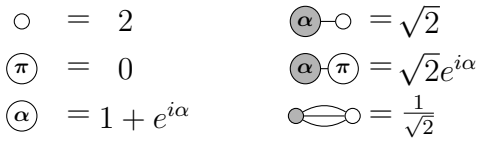


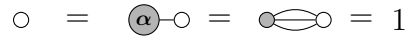
Figure 2.1: Only connectivity matters in ZX-diagrams.

Scalars In ZX-calculus, a *scalar* is a diagram with zero inputs and outputs [14]. The linear map or matrix interpretation for a scalar is a one by one matrix, which is

a complex number. Some instances of scalars presented in [14] are shown as follows:



In this thesis, if a scalar is non-zero, then unless specifically stated, we will ignore or drop it in our calculation. For example, if global scalars are ignored, then we have:



The justifications for ignoring global scalars can be found in [14] and [1].

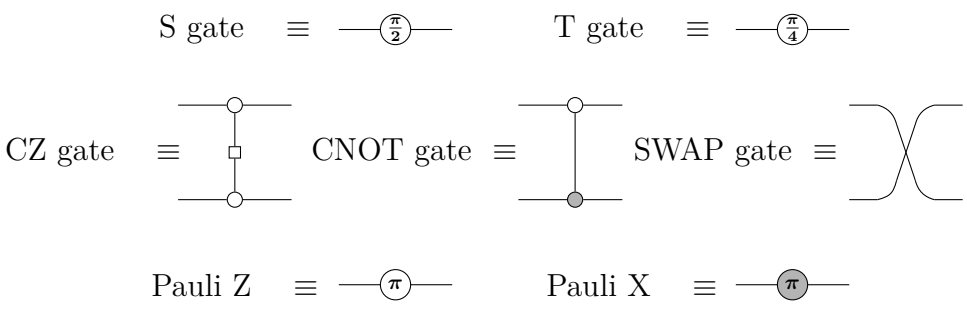
Common gates in ZX-diagram Recall that we have introduced Hadamard gates in section 2.1. In ZX-calculus, a Hadamard gate is represented by the following notation:

It is well-known that a Hadamard gate can be expressed by spiders in ZX-calculus, and the following equation (ED) defines one way to decompose the Hadamard gate into spiders:

(ED)

where the global scalar is ignored.

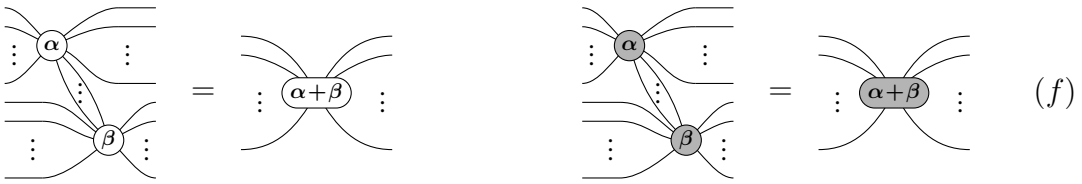
We give ZX notations for some other common gates as follows:



Sometimes we call CZ gates as CZ wires, since a CZ gate in ZX-diagrams looks like a wire connecting two vertices.

Rewrite rules Another reason why ZX-calculus is useful for derivations about qubit systems is that there are a set of *rewrite rules* allowing you to rewrite one ZX-diagram into one of its equivalent forms. With the help of ZX-calculus, properties of circuits and states can be proved in a diagrammatic way. Here we introduce some important rewrite rules in ZX-calculus that we will use later in our proofs. However, the rewrite rules introduced here are neither complete for ZX-calculus nor independent to each other. For a comprehensive introduction to ZX-calculus rewrite rules, see [1]. For formal discussions about the completeness of rewrite rules, see [20].

The rule of *spider fusion* is an important rewrite rule which will be often used in this thesis. In ZX-calculus, two connected spiders in the same colour can fuse together with their phases adding together. Diagrammatically, we have [14]:

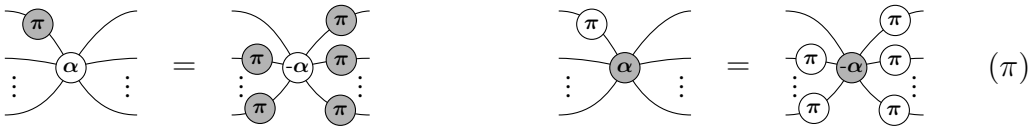


(f)

We use (f) to denote spider fusion rules. We may apply spider fusion rules without mentioning in our proofs.

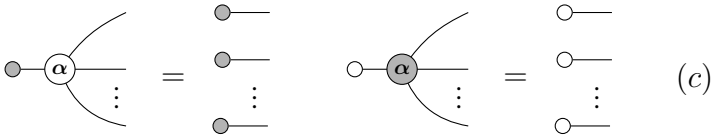
The following *identity removal* rule will also often be applied in this thesis without mentioning:

The following π-copy rules hold true:



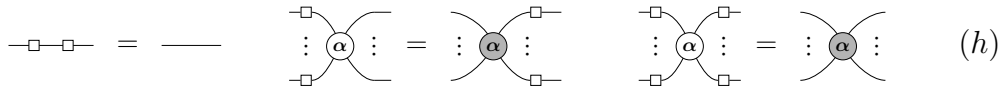
(π)

The following *copy* rules hold true:



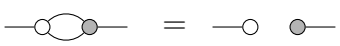
(c)

The following *colour changing* rules hold true:

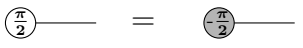


(h)

The *Hopf* rule tells us a white spider and a grey spider connected by two identity wires can disconnect as follows:



By rules of (h), (ED), (f) and (π), the following equation holds true [14]:

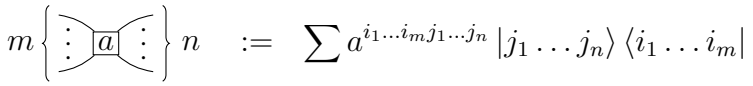


(π_2)

Clifford unitaries The *Clifford unitaries* are gates generated by compositions of CNOT, Hadamard and S gates [14]. In ZX-calculus, Clifford gates can be represented by ZX-diagrams that contain only phases that are multiple of $\frac{\pi}{2}$ [14].

2.3 ZH-calculus

ZH-calculus is another graphical language, which is presented by Miriam Backens and Aleks Kissinger and it is viewed as a variant of ZX-calculus [16]. ZH-calculus defines a superset of ZX-calculus and it extends the reasoning power of ZX-calculus [14]. To define ZH-calculus, we first introduce the concept of H-box, a generalized form of Hadamard gate. A *H-box* with m inputs and n outputs with parameter a is defined as follows:

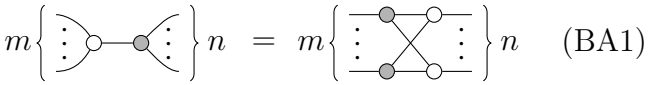


$$:= \sum a^{i_1 \ldots i_m j_1 \ldots j_n} |j_1 \ldots j_n\rangle \langle i_1 \ldots i_m|$$

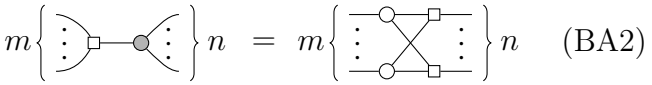
Therefore, a Hadamard gate is a special case of H-boxes, where the input and output number are both 1 and the parameter value is -1. If the parameter value for a H-box is -1, then we can drop the label inside the box:

For simplicity, we sometimes call a H-box with n inputs and m outputs as $(n+m)$-arity H-box or $(n+m)$-ary H-box.

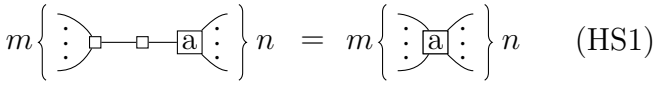
Similar to ZX-calculus, ZH-calculus consists of *ZH-diagrams* and *rewrite rules*. If we drop any non-zero global scalars, then the generators of ZH-calculus are white spiders and H-boxes, where white spiders have the same linear map interpretations as in the ZX-calculus. Grey spiders are defined as one of the derived generators in ZH-calculus and also have the same linear map interpretations as in the ZX-calculus. Any ZX-diagram is a valid ZH-diagram, and any rewrite rules in ZX-calculus also holds true in ZH-calculus. In addition, ZH-calculus defines some new rewrite rules. Here we only introduce the following additional rewrite rules that will be used in this thesis:



(BA1)

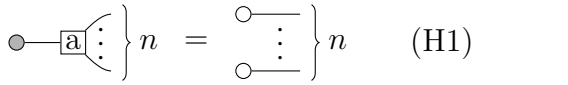


(BA2)

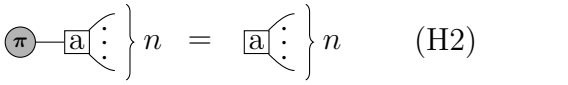


(HS1)

The rewrite rules introduced here are neither complete for ZH-calculus nor independent to each other. For a formal discussions about the completeness of rewrite rules of ZH-calculus, see [21]. The following equation holds true in ZH-calculus [21] [14]:

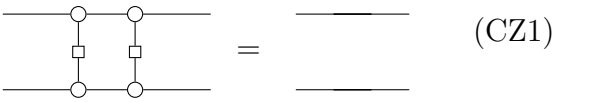


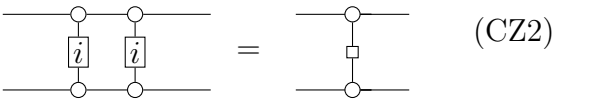
(H1)



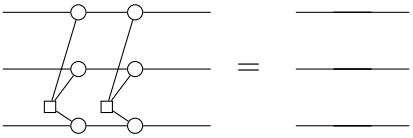
(H2)


(CS1)


(CZ1)

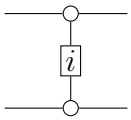


(CZ2)

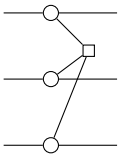
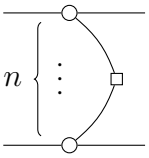


(CZ3)

By equation (CZ2), the following gate is sometimes referred to as $\sqrt{CZ}$ gate for obvious reasons:



We give ZH notations for some other common gates as follows:

CCZ gate $\equiv$  C^nZ gate $\equiv n$ 

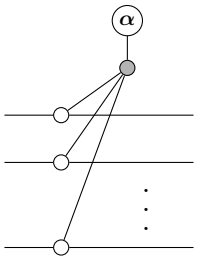
The CCZ gate and the C^nZ gate are both controlled unitaries, and the C^nZ gate generalizes both CZ gates and CCZ gates in the following way: the $C^{k+1}Z$ gate is a controlled-C^kZ gate for $k \in \mathbb{N}^+$; when $k = 1$, the C^kZ gate is a CZ gate.

For reasons explained above, we will treat all ZX-diagrams as ZH-diagrams and use all rewrite rules that are introduced in section 2.2 and in this section.

For a detailed introduction to ZH-calculus, see [14].

2.4 Phase gadgets

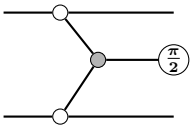
A ZH-diagram of the following structure is referred as a phase gadget [14]:



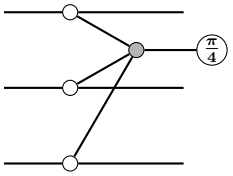
where $\alpha \in [0, 2\pi)$ is the phase value of the gadget. We think of phase gadgets as having the equal number input wires and output wires. Use P_α^n to denote a

phase gadget with n input wires, with α as the phase value. For example, a phase gadget with two inputs and phase value $\frac{\pi}{2}$ and another phase gadget with three inputs and phase value $\frac{\pi}{4}$ are shown as below:

$P^2_{\frac{\pi}{2}}$:=



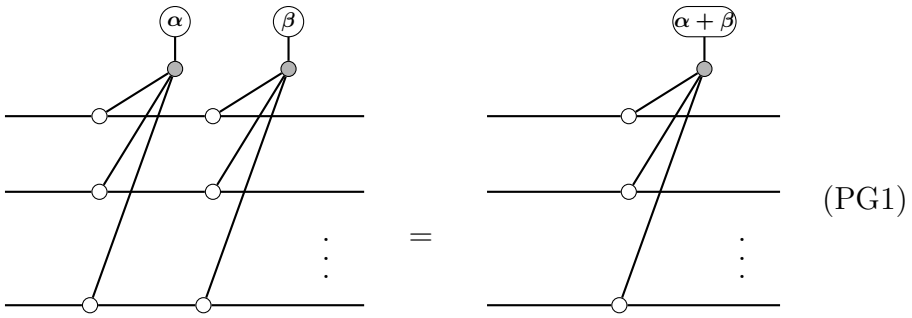
$P^3_{\frac{\pi}{4}}$:=



For simplicity, sometimes we call a phase gadget with two inputs as a binary phase gadget, and a phase gadget with three inputs as a trinary phase gadget, and we refer to P^n_α as α phase gadget if it doesn't cause any ambiguity.

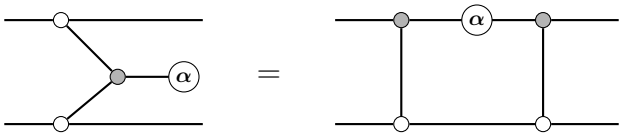
A phase gadget P^n_α can represent a function f of the form $f(x_1, ..., x_n) = \alpha \cdot (x_1 \oplus ... \oplus x_n)$, where $x_1, ..., x_n \in \mathbb{B}$ and $\oplus$ denotes a XOR operation [14]. This is because if we input a state $|x_1...x_n\rangle$ into a phase gadget P^n_α, then we get $e^{i\alpha(x_1 \oplus ... \oplus x_n)}|x_1...x_n\rangle$.

Two phase gadgets with the same number of inputs may fuse together as shown by the equation (PG1) [14].

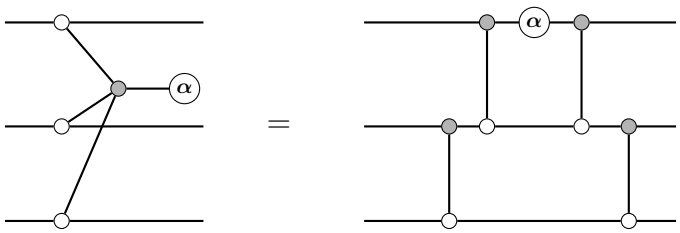


(PG1)

Phase gadgets can be decomposed to CNOT gates and a phase gate. For example, equations (PG2) and (PG3) hold true [14].

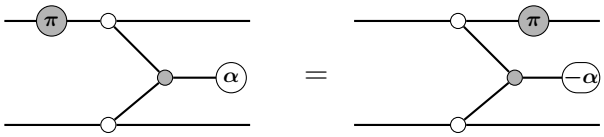


(PG2)

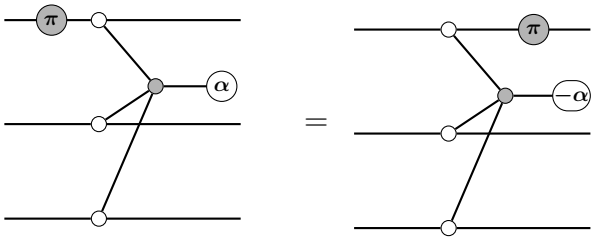


(PG3)

We can easily derive from (PG2), (PG3), and rules of (f) and (π) that the following equations (NPG2) and (NPG3) holds:

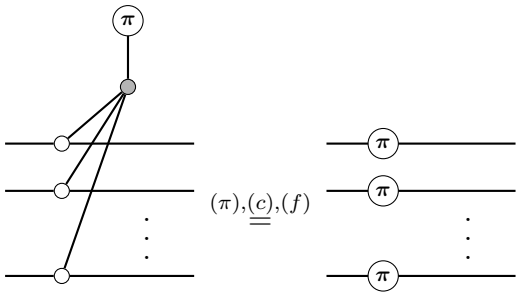


(NPG2)



(NPG3)

A π phase gadget is "unstable", since it "breaks down" to Pauli Z operations as shown by equation (PGπ):



(PGπ)

The phase gadget state We say that a state is a *phase gadget state* if all gates on it are phase gadgets. The phase gadget state is a candidate for the resource state in MBQC. We will introduce MBQC in section 2.9.

2.5 Graph states and Hypergraph states

In this section, we introduce two states that are well-known resource states in MBQC introduced in section 2.9. A *graph state* is a family of states of the shape of a simple undirected graph. To define a graph state, we start with a simple undirected graph $G = (V, E)$. Its corresponding graph state $|G\rangle$ can be constructed using the following steps: (i) prepare a state $|+...+\rangle$ where the number of qubits is equal to $|V|$, (ii) for each edge $(i, j) \in E$, apply a CZ gate on qubits i and j. For an example of graph states, see figure 2.2. Graph states are widely used in many

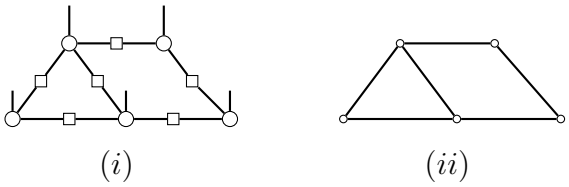


Figure 2.2: An example graph state. (i) The graph state. (ii) The corresponding undirected graph.

fields of quantum computing [5–7].

Hypergraph states generalize the concept of graph states. To define a hypergraph state, we first need to define a hyperedge. *Hyperedges* are generalization of edges. If we view a common edge as a binary tuple (x_1, x_2) where x_1 and x_2 are vertices, then a hyperedge is defined as a tuple $(x_1, \ldots, x_p)$, where $p \in \mathbb{N}^+$ and $x_1, \ldots, x_p$ are vertices. Note that a hyperedge may contain one vertex. A *hypergraph* is defined as $G =< V, E >$ where V is a set of vertices and E is a set of hyperedges. Given a hypergraph $G =< V, E >$, we can construct its corresponding hypergraph state $|G\rangle$ using the following steps: (i) prepare a state $|+...+\rangle$ with the number of its qubits equal to $|V|$, (ii) for each hyperedge $(v_1, \ldots, v_p) \in E$, perform a C^pZ gate on qubits $v_1, \ldots, v_p$. When $p = 2$, the hyperedge reduces to a common edge and the C^pZ operation reduces to a CZ gate. For an example of hypergraph states, see figure 2.3. For simplicity, we sometimes call a CCZ gate as a *CCZ hyperedge*. Readers can

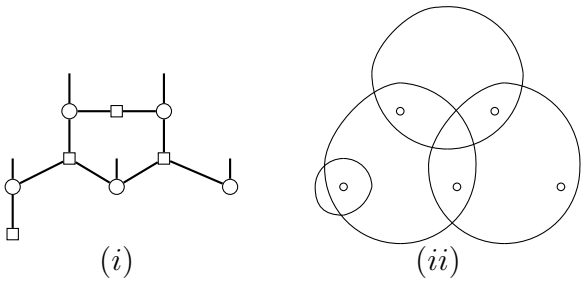
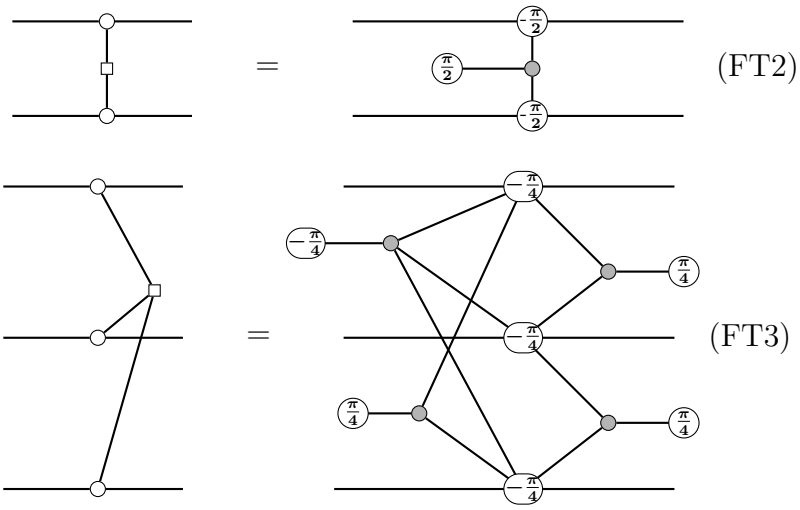


Figure 2.3: An example hypergraph state. (i) The hypergraph state. (ii) The corresponding hypergraph, where we use black circles to denote hyperedges.

find more details about graph states and hypergraph states in [4].

2.6 Graphical Fourier theory

It is often useful to decompose a H-box into some low-level gates. We have seen in section 2.2 that we can use equation (ED) to decompose a 2-ary H-box. In this section, we introduce a method to decompose general H-boxes. *Graphical Fourier theory* is proposed by Stach Kuijpers, John van de Wetering, and Aleks Kissinger as a link between ZX-calculus and ZH-calculus [17]. We will only use the following two special cases of Graphical Fourier theorems:



In this thesis, we refer to (FT2) and (FT3) as *graphical Fourier transforms*, *Fourier transforms*, or *Fourier decompositions*.

2.7 Universal computations

Universality of Computation is an important topic in the area of Quantum Computing. One of the basic question in this topic is whether we can draw gates from some gate set and construct an arbitrary unitary operation [22]. We recommend interested readers to read [22] and [23] for a more detailed introduction to this topic.

Roughly speaking, a gate set is said to be *universal* if we can decompose an arbitrary unitary operation into gates drawn from this gate set. This gate set is called a *universal gate set*. In fact, there are different types of universality. For example, readers can find definitions of *strict universality* and *computational universality* in

[23], and the author in [24] is interested in *encoded universality*. However, since universality is not our focus, we will ignore the difference among definitions of different universality and simply treat all kinds of universality as the same. In this thesis, we will use two universal gate sets: $\{CCZ, Hadamard\}$ and $\{Hadamard, S, CNOT, T\}$.

The concept of universal gate set gives us a way to prove that universal universal computations can be achieved: we first choose a universal gate set; then we show that we can implement all the gates in this set and compose any number of gates drawn from the set in any way we want. For example, in [10], in order to show that universality can be achieved, the authors show that S gates, T gates, CNOT gates, and Hadamard gates can be implemented, and they can compose gates from the universal gate set $\{Hadamard, S, CNOT, T\}$ in any desired manner if they tile their constructions in a regular pattern.

2.8 Quantum circuit model

The *quantum circuit model*, or *circuit model* defines a straightforward method to implement quantum computations. In this model, to implement any computation, we first prepare qubits in a certain state, then we build a circuit composed by quantum unitary gates and perform the circuit onto these qubits, and finally we perform measurements on all or some of the resulting qubits. For an example of circuit model see figure 2.4. Read [1] for a more detailed discussion about the quantum circuit model.

2.9 Measurement-based quantum computation

The *measurement-based quantum computation* (MBQC) is one of the well-known methods to achieve universal quantum computation. The *one-way model* [3] is the most well-studied MBQC model, and we use this model as an example to introduce MBQC. In contrast to the quantum circuit model introduced in section 2.8, MBQC defines a totally different method to implement computations. In circuit model, we

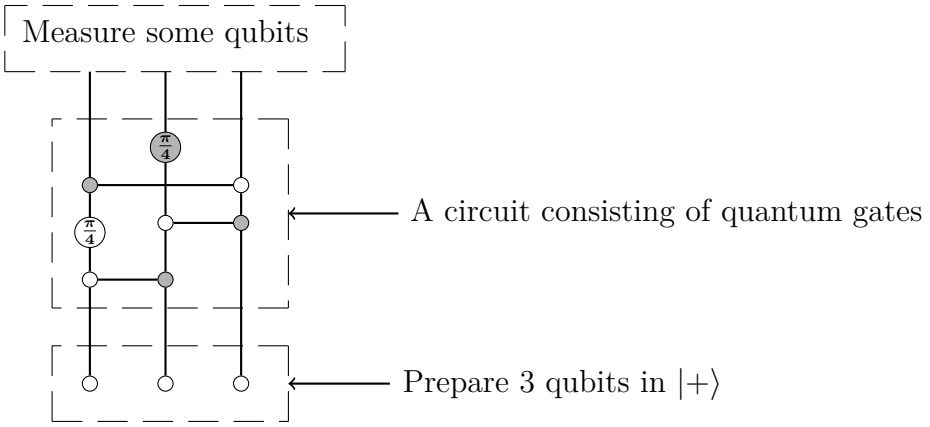


Figure 2.4: An example of circuit model computation.

start by preparing qubits in a fixed state and then perform a sequence of unitary gates. Any computation in circuit model is implemented by building a circuit piece by piece using individual gates. However, in MBQC, we start by preparing a highly entangled *resource state*. The resource state is prepared in a predefined way and thus does not depend on the computation to be implemented. For example, in the one-way model, the resource state is a graph state which is prepared in a standard process. After the resource state is prepared, we perform single-qubit measurements on this resource state. Each qubit can be measured in a different basis, and how later measurements are performed may depend on previous measurement outcomes. Since measurements non-deterministically produce different outcomes, undesired outcomes may appear. We call undesired outcomes as measurement errors. So, to implement a deterministic computation, we need to correct errors produced in measurements.

For a toy example, consider preparing a three-qubit resource state entangled by two CZ wires, and perform Pauli X and Z measurements on all of these qubits, as illustrated in figure 2.5.

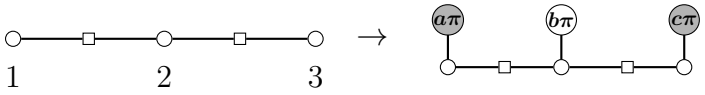


Figure 2.5: An toy example of MBQC. On the left-hand side of arrow, we prepare a three-qubit resource state. On the right-hand side, we show the result after performing Pauli Z measurements on qubits 1 and 3, and Pauli X measurement on qubit 2. $a, b, c \in \{0, 1\}$ depend on the measurement outcomes.

In MBQC model, a computation is implemented by cleverly planning how to perform measurements on the resource state. The plan of how measurements are performed is called the *measurement pattern*. We will introduce measurement patterns in more details in section 2.9.3.

In MBQC, we are allowed to perform measurements of which measurement basis live on the three principal axes of the Bloch sphere [14]. But in this thesis, we only care about two special cases of these allowed measurements: Pauli X and Pauli Z measurements. In section 2.9.1, we will discuss how we represent Pauli X and Z measurements in ZH-calculus. In section 2.9.2, we will discuss how we represent measurement errors in ZH-diagrams and how to correct them.

2.9.1 Single-qubit Pauli X and Z measurements in ZH-calculus

Performing a Pauli Z measurement on a qubit non-deterministically projects this qubit onto one of the eigenstates $\{|0\rangle, |1\rangle\}$. We say that the outcome of Z-measurement is 0 if the measurement projects the qubit onto $|0\rangle$ and we say the outcome is 1 if the measurement projects the qubit onto $|1\rangle$. Performing a Pauli X measurement on a qubit non-deterministically projects this qubit onto one of the eigenstates $\{|+\rangle, |-\rangle\}$. We say that the outcome of X-measurement is 0 if the measurement projects the qubit onto $|+\rangle$ and we say the outcome is 1 if the measurement projects the qubit onto $|-\rangle$. In this thesis, for a qubit q, terms like *measure q in Z*, *measure q in Z base*, and *perform a Z-measurement on q*, are used to denote that we perform a Pauli Z measurement on qubit q; terms like *measure q in X*, *measure q in X base*, and *perform a X-measurement on q*, are used to denote that we perform a Pauli X measurement on qubit q.

Single-qubit Pauli X and Z measurements can be concisely depicted in ZH-calculus [10]. In ZH-calculus, performing a Pauli X measurement on a qubit amounts to composing an effect $\langle a|$ to the qubit, where $a \in \{0, 1\}$ is the measurement outcome; performing a Pauli Z measurement on a qubit amounts to composing an effect $\left(b \cdot \left(\langle -| - \langle +|\right) + \langle +|\right)$ to the qubit, where $b \in \{0, 1\}$ is the measurement outcome. This is illustrated diagrammatically in figure 2.6.

(i) (ii)

Figure 2.6: Assume that qubits are prepared in $|+\rangle$. In (i), we perform a Pauli X measurement on the qubit on the left-hand side of the arrow. In (ii), we perform a Pauli Z measurement on the qubit on the left-hand side of the arrow. On the right-hand side of the arrow, we draw the results of measurements. $a \in \{0, 1\}$ and $b \in \{0, 1\}$ depend on the outcomes of measurements. In this diagram, we use the measurement pattern notation defined in 2.9.3

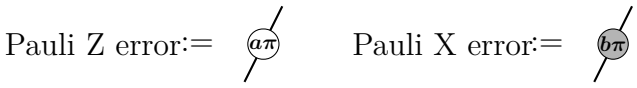
In this thesis, lower case letters with or without subscript such as a, b, c, a_1, b_2, c_3 appearing in ZH-diagrams represent measurement outcomes, and they only take value from the set $\{0, 1\}$.

2.9.2 Measurement errors in ZH-calculus

Since measurements introduce errors to our computations, we need to model measurement errors produced in MBQC, and in order to implement deterministic computations, we need ways to correct errors. We use terms *measurement errors* and *measurement byproducts* to denote errors of measurements interchangeably.

Pauli X and Z errors As described in section 2.9.1, a Pauli X measurement non-deterministically composes an effect $\langle 0|$ or $\langle 1|$ to a qubit. When the measurement outcome is 1 and thus the introduced effect is $\langle 1|$, we say that we produce a Pauli Z error. Similarly, a Pauli Z measurement non-deterministically composes an effect $\langle +|$ or $\langle -|$ to a qubit. When the measurement outcome is 1 and thus the introduced

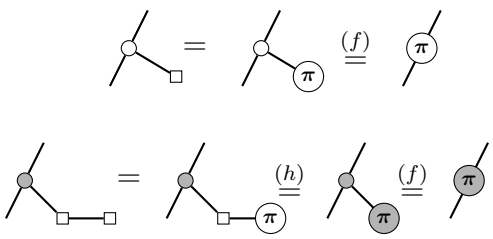
effect is $\langle -|$, we say that we produce a Pauli X error. Therefore, in ZH-calculus, the Pauli Z error and Pauli X error can be depicted as



where $a \in \mathbb{Z}$ and $b \in \mathbb{Z}$ depend on outcomes of measurements. If $a = (2k + 1)$ where $k \in \mathbb{Z}$, then it means there is a Pauli Z error; otherwise there is no Pauli Z error. If $b = (2k + 1)$ where $k \in \mathbb{Z}$, then it means there is a Pauli X error; otherwise there is no Pauli X error. Sometimes we also use a orange-dashed-line notation to denote Pauli X and Z errors as illustrated in figure 2.7.

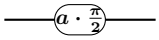
Figure 2.7: Orange dashed lines denote measurement errors. The part of diagram depicted by orange dashed line will appear if an error is produced. In (i), a Pauli Z error is depicted by a dashed orange H-box with 1 wire. In (ii), a Pauli X error is depicted by two connected dashed orange H-boxes.

This notation of orange dashed lines is used because the orange-dashed-line notation and phase-spider notation are equivalent: If there is no error and thus the part of diagram depicted by orange dashed lines will not appear, then obviously these two representation are equivalent; otherwise there is an error and thus the orange-dashed-line part appears, then we have



indicating that these two representations are equivalent.

S errors We define a S error to be the error taking the form of a S gate.



where $a \in \mathbb{Z}$ depends on measurement outcomes. If $a = 2k + 1$ where $k \in \mathbb{Z}$ then it means there is a S error; otherwise there is no S error. However, if the S gate is the computation we intend to implement, then the S gate should not be treated as an error anymore.

Multi-qubit errors Pauli X and Z errors and S errors are single-qubit errors, since they only have effect on some single qubit. However, in MBQC, multi-qubit errors might also be produced. One common multi-qubit errors are CZ errors illustrated in figure 2.8. In this thesis, we only use dashed lines to depict CZ errors.

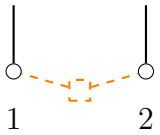


Figure 2.8: The CZ error on qubits 1 and 2 is depicted by a CZ gate depicted in orange dashed lines, the appearance of which depends on corresponding measurement outcomes.

Another multi-qubit error is the phase gadget error, which is a measurement error taking the form of a phase gadget, as illustrated in figure 2.9.

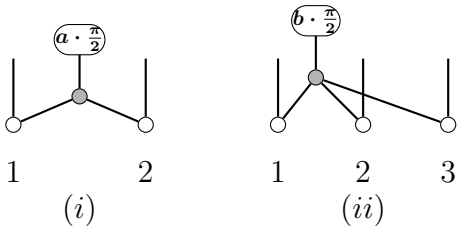
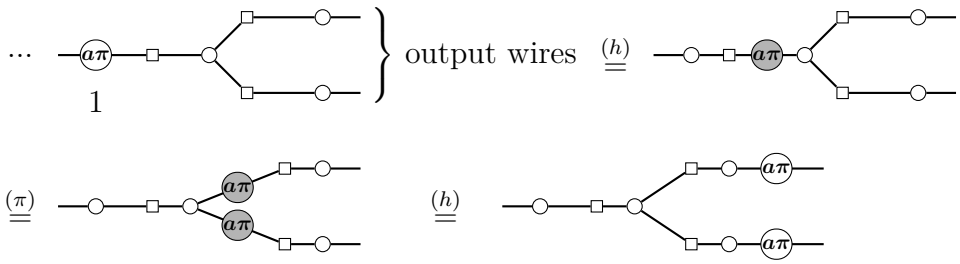


Figure 2.9: In (i), we depict a binary $\frac{\pi}{2}$ phase gadget error on qubits 1 and 2. In (ii), we depict a trinary $\frac{\pi}{2}$ phase gadget error on qubits 1, 2, and 3. $a \in \mathbb{Z}$ and $b \in \mathbb{Z}$ depend on measurement outcomes. In (i), if $a = (2k + 1)$ where $k \in \mathbb{Z}$, then it means there is a binary $\frac{\pi}{2}$ phase gadget error; otherwise there is no phase gadget error.

By rules of ZH-calculus and equation (PG1), we can easily show that a $\frac{-\pi}{2}$ phase gadget error is equal to a $\frac{\pi}{2}$ phase gadget error up to Pauli Z errors. So,

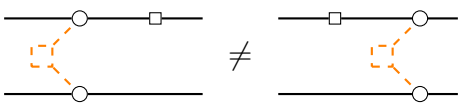
when we have X and Z Pauli errors and $\frac{\pm\pi}{2}$ phase gadget errors, we only need to consider $\frac{\pi}{2}$ phase gadget error.

Correction of measurement errors One technique to correct Pauli X and Z measurement errors on graph states is called *feed-forward* [1]. Feed-forward is the process where we push Pauli X and Z errors along edges of the graph state until they only appear on the output wires and then correct these errors by performing corresponding gates. For example, in the following figure, we illustrate the process where we push a Pauli Z error along the graph state to output wires:

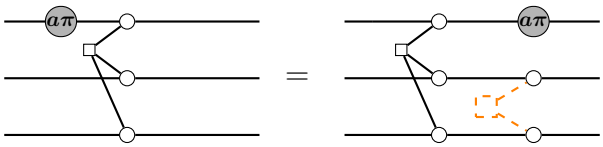


Then we can apply the process of post-selection to correct errors on output wires [1].

There are two reasons why the technique of feed-forward succeeds: (i) we only deal with Pauli X and Z errors, and (ii) the edges of graph states are CZ wires, and by the rules of (h), (π), (f), we can push around Pauli X and Z errors in graph states without producing errors other than Pauli X and Z errors. Issues can occur when directly applying this technique to correct errors in a resource state which is not a graph state or to correct errors other than Pauli X and Z errors. For instance, CZ errors cannot pass through edges in a graph state:

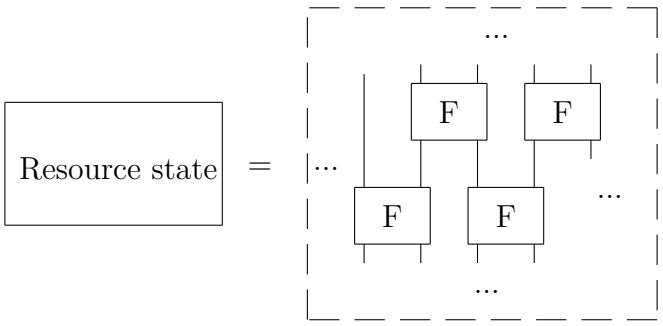


For another instance, when a Pauli X error passes through a CCZ hyperedge, we produce a CZ error:



The proof for the above equation is part of the proof for lemma 3.1.5.

However, it is still possible to apply the idea of feed-forward to correct non-Pauli errors in a resource state that is not a graph state. One way to achieve that is to view the resource state as being composed by small fragments. Connections between fragments are identity wires, and each fragment has inputs and outputs:



For each fragment, we ensure that it takes Pauli X and Z errors coming from its input wires and after measurements on this fragment are performed, only Pauli X and Z errors are produced on its output wires. The measurement performed on a fragment can depend on the Pauli X and Z errors on its input wires. This way, we ensure errors passed between fragments are Pauli X and Z errors, which enables us to push around Pauli X and Z errors in the resource state, and we use the idea of feed-forward to push all Pauli X and Z errors to output wires of MBQC and then correct them. In section 2.9.3, we will see how this idea of feed-forward can be included in measurement patterns.

2.9.3 Measurement patterns in ZH-calculus

Performing measurements is essential to implement computations in MBQC model, so we need measurement patterns to define how measurements are to be performed on a resource state. Readers can refer to [25] for a formal and more detailed description of measurement patterns. In this thesis, we only consider single-qubit Pauli X and Z measurements introduced in section 2.9.1, and we will use a simple notation to represent measurement patterns in ZH-calculus: In a ZH-diagram, we write a letter $M \in \{Z, X\}$ on some qubit to denote that we will perform a Pauli M measurement on this qubit.

Instead of defining measurement patterns on the whole resource state, we usually define measurement patterns on fragments of the resource state. In a measurement pattern defined on a fragment, we should set *input qubits* and *output qubits*. Each input qubit has an input wire and each output qubit has an output wire. These input and output wires connect to other parts of the resource state, therefore input qubits and output qubits will not be measured in a measurement pattern. In diagrams, these input and output wires are usually shown as free or dangling edges, but we omit them if no confusion is caused. For simplicity, we refer to input qubits as *inputs* and output qubits as *outputs*. For instances of measurement patterns, see figure 2.10.

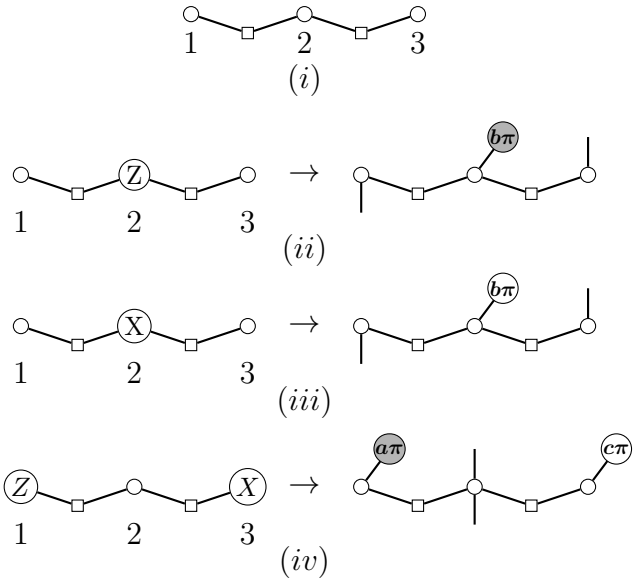
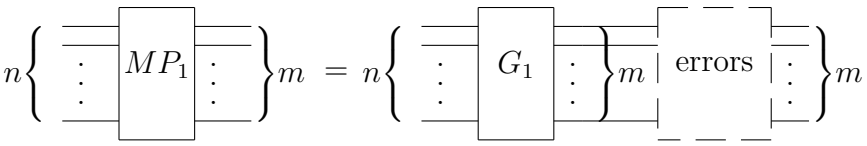


Figure 2.10: In (i), we have a three-qubit system with qubits 1, 2, and 3 where there are a CZ edge between qubits 1 and 2 and another CZ edge between qubits 2 and 3. In (ii), on the left-hand side of the arrow we define a measurement pattern on the system in (i), where we set qubit 1 as the input and 3 as the output and we measure qubit 2 in Z. In (iii), on the left-hand side we define another measurement pattern on on the system in (i), where we set qubit 1 as the input and 3 as the output and we measure qubit 2 in X. In (iv), on the left-hand side we define another measurement pattern on on the system in (i), where qubit 2 is both the input and the output and we measure qubit 1 in Z and qubit 3 in X. On the right-hand side of the arrow in (ii), (iii), and (iv), we calculate the result of the measurement pattern in the way described in section 2.9.1.

For a n-input-m-output measurement pattern MP_1, we say that MP_1 implements or achieves a n-input-m-output operation G_1 up to errors if the following

equation holds true:



Measurement errors are introduced in section 2.9.2. For an example, the measurement pattern defined in figure 2.11 implements an identity wire, up to Pauli errors.

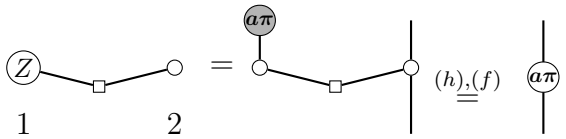
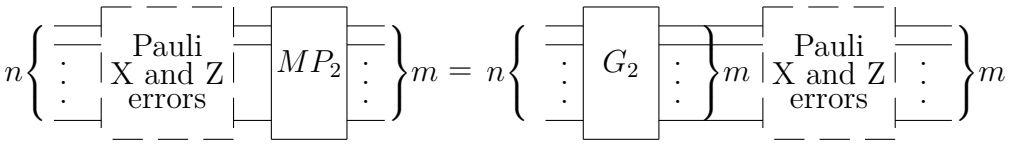
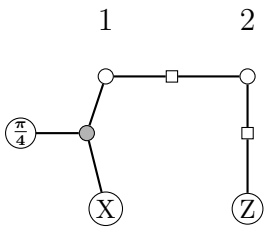


Figure 2.11: On the leftmost end of the equation, we define a measurement pattern on a two-qubit system where we set qubit 2 as both the input and the output. Calculating this measurement pattern we get the the identity wire on the rightmost end, up to Pauli Z errors.

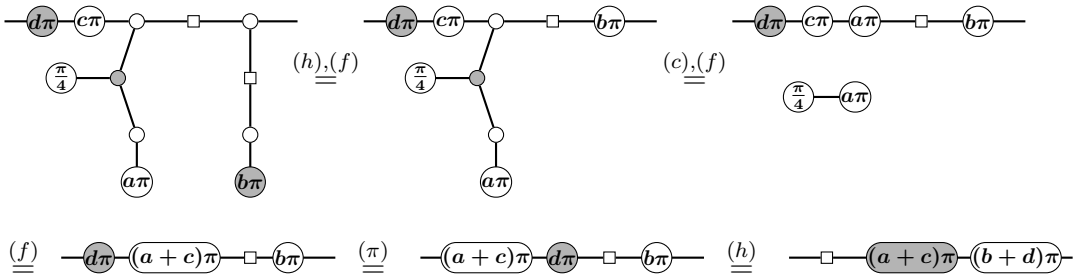
Now that we have seen how to define measurement patterns on fragments, it is time to discuss how to define a MBQC model that uses the idea of feed-forward to correct errors. To define such a MBQC model, we need to take Pauli X and Z errors into considerations in measurement patterns. We say that a n-input-m-output measurement pattern MP_2 implements a n-input-m-output gate G_2 and admits feed-forward if the following equation holds true:



For instance, the following measurement pattern, where qubit 1 is the input and qubit 2 is the output, implements a CZ wire between qubits 1 and 2, and admits feed-forward:



Since we have



3
Hypergraph MBQC Models

In this chapter, we use ZH-calculus to investigate and justify protocols of two hypergraph state MBQC models. One of the models is the GGM state based MBQC model proposed in [12] by Gachechiladze et al., where we use *GGM state* to call the resource state in this model. Another model is the Union Jack state based MBQC model proposed in [11], where the *Union Jack state* is the resource state. Although in [12] and [11], authors have given their justifications for proposed protocols, our diagrammatic proofs in ZH-calculus are more concise, intuitive, and easier to verify.

In section 3.1.1, we first introduce a hexagon notation and then use it to present the GGM state. In section 3.1.2, we prove useful lemmas in ZH-calculus. In section 3.1.3, we use ZH-calculus to describe and justify measurement patterns defined on fragments of the GGM state. In section 3.1.4, we use ZH-calculus and results in section 3.1.3 to describe and justify the protocol of the GGM state based MBQC model.

We briefly introduce the Union Jack state in section 3.2.1. Based on results in [18], in section 3.2.2 we use ZH-calculus to show that the CCZ gate can be implemented by the measurement pattern proposed in [11], which serves as the essential evidence for the fact that universality can be achieved by Union Jack state based MBQC model.

3.1 GGM state based MBQC model

3.1.1 The GGM state

The hexagon notation We introduce the following hexagon notation illustrated in figure 3.1. This hexagon notation is similar to the box notation introduced in [12]. The reason why we use hexagons instead of boxes is to distinguish this notation from that of a H-box. In figure 3.1 (a), a four-qubit hypergraph state where qubits

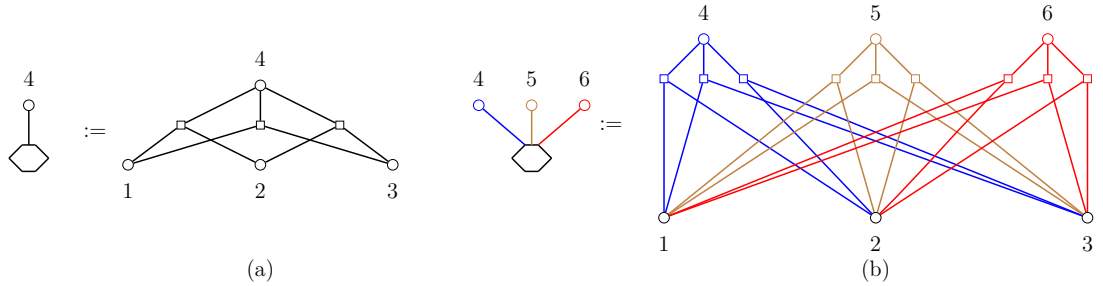


Figure 3.1: Hexagon notation in the GGM state. (a) A four-qubit hypergraph state with three CCZ hyperedges. (b) A six-qubit hypergraph state with nine CCZ hyperedges. Different colours are used to make the connections in the diagram clearer, and they have no syntactic meanings. On the left-hand side, three qubits and corresponding CCZ hyperedges are hidden/encapsulated inside the hexagon.

are numbered from 1 to 4 with three CCZ hyperedges $(1,2,4),(1,3,4),(2,3,4)$ is denoted by a hexagon connected to qubit 4. Each of these hyperedges include qubit 4 and two out of the three qubits $1,2,3$. In figure 3.1 (b), a six-qubit hypergraph state where qubits are numbered from 1 to 6 with nine CCZ hyperedges $(1,2,4)$, $(1,3,4)$, $(2,3,4)$, $(1,2,5)$, $(1,3,5)$, $(2,3,5)$, $(1,2,6)$, $(1,3,6)$, $(2,3,6)$ is denoted by a hexagon connected to qubits 4, 5, and 6.

We write a symbol $M \in \{Z, X\}$ on the hexagon notation to define a measurement pattern where qubits hidden inside the hexagon are measured in Pauli M and the three qubits attached on the outside are outputs or output qubits. Recall from section 2.9.3 that setting some qubit as the input or the output in a measurement pattern implies that this measurement pattern is defined on a fragment of a resource state, and when a qubit is set as the input or the output in a measurement pattern, this qubit will not be measured.

For example, in figure 3.2, qubits 4, 5, 6 are outputs, and qubits 1, 2, 3 are all measured in Pauli X. For simplicity, when there is a symbol $M \in \{Z, X\}$ on a

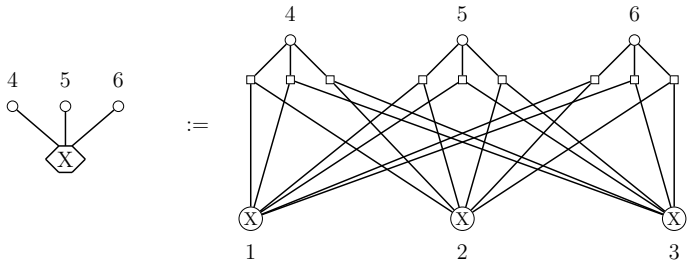


Figure 3.2: A measurement pattern on a six-qubit hypergraph state, where qubits 4, 5, 6 are outputs, and qubits 1, 2, 3 are measured in Pauli X. Here in the diagram, notations of measurement patterns defined in section 2.9.3 are used.

hexagon notation, we say that this hexagon is measured in M or this hexagon is measured in Pauli M.

GGM state With the help of hexagon notations, the ZH-diagram of the GGM state is presented in figure 3.3.

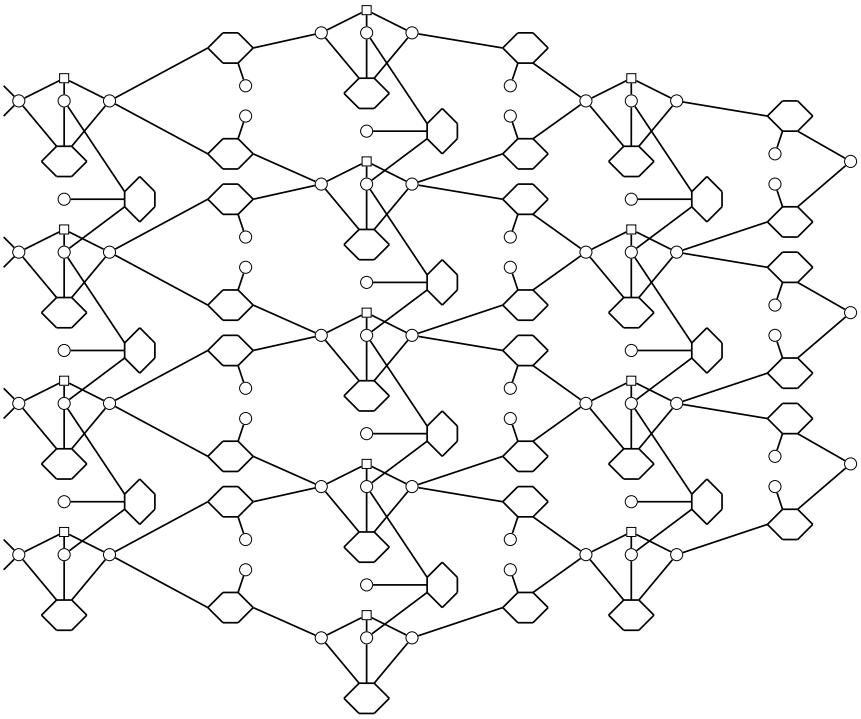
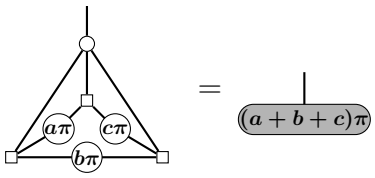


Figure 3.3: The GGM state. The hexagon notation is used.

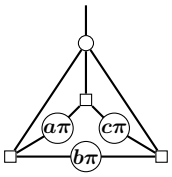
3.1.2 Lemmas

In this section, we prove a couple of lemmas that will be used in sections 3.1.3 and 3.1.4.

Lemma 3.1.1. For $a, b, c \in \{0, 1\}$, if $a + b + c \in \{0, 1, 3\}$, then the equation 3.1 holds true; if $a + b + c = 2$, then the equation 3.2 holds true.



(3.1)



(3.2)

Proof. In this proof, we always ignore global scalars. Let $|\psi\rangle$ be the state on the left-hand side of the equation 3.1 and 3.2, then we have:

$$|\psi\rangle = |0\rangle \iff \langle 0|\psi\rangle = 1 \text{ and } \langle 1|\psi\rangle = 0$$

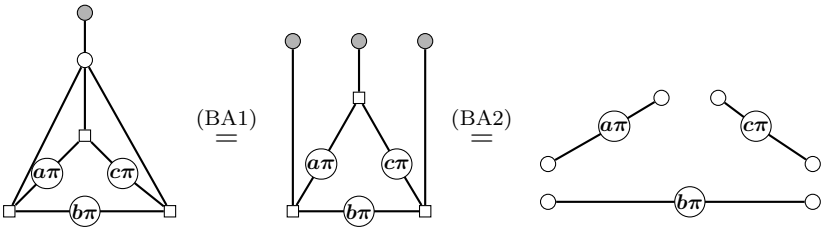
$$|\psi\rangle = |1\rangle \iff \langle 1|\psi\rangle = 1 \text{ and } \langle 0|\psi\rangle = 0$$

$$|\psi\rangle = 0 \iff \langle 1|\psi\rangle = 0 \text{ and } \langle 0|\psi\rangle = 0$$

By definition of matrix interpretation of states, we have (ignoring global scalars):

$|0\rangle$ =  , $|1\rangle$ = 

So, we can calculate the value of $\langle 0|\psi\rangle$ as follow:

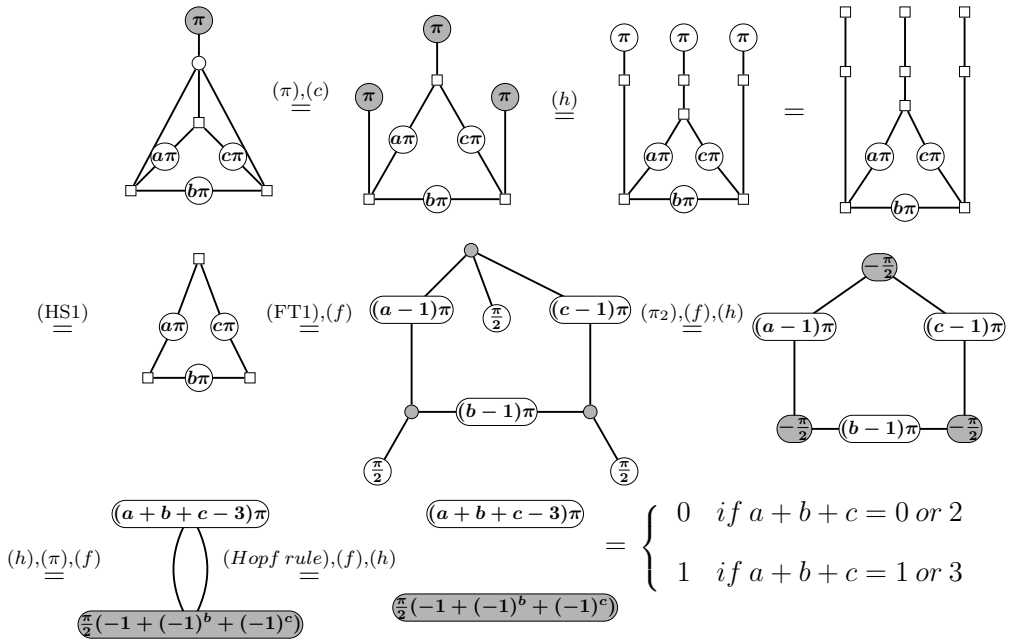


$$\overset{(f)}{=} \quad \begin{matrix} a\pi & & c\pi \\ & b\pi & \end{matrix} \quad = \quad \begin{cases} \boxed{\circ \quad if\ a+b+c=0} \\ \boxed{(\pi) \quad if\ a+b+c>0} \end{cases} = \quad \begin{cases} 1 & if\ a+b+c=0 \\ 0 & if\ a+b+c>0 \end{cases}$$

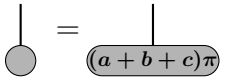
In the last step, we ignore global scalars and use the following fact:

$$\circ \ = \ 1, \qquad (\pi) \ = \ 0$$

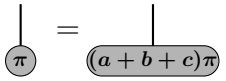
Similarly, we can calculate the value of $\langle 1|\psi\rangle$ as follow:



Therefore, if $a+b+c=0$, then we have $\langle 0|\psi\rangle = 1$ and $\langle 1|\psi\rangle = 0$, which indicates

$$|\psi\rangle \quad = |0\rangle \quad =$$  $=$

If $a+b+c=1$ or 3, then we have $\langle 0|\psi\rangle = 0$ and $\langle 1|\psi\rangle = 1$, which indicates

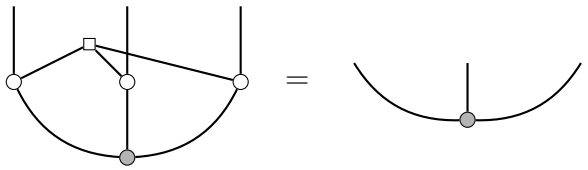
$$|\psi\rangle \quad = |1\rangle \quad =$$  $=$

If $a+b+c=2$, then we have $\langle 0|\psi\rangle = 0$ and $\langle 1|\psi\rangle = 0$, which indicates

$$|\psi\rangle = 0.$$

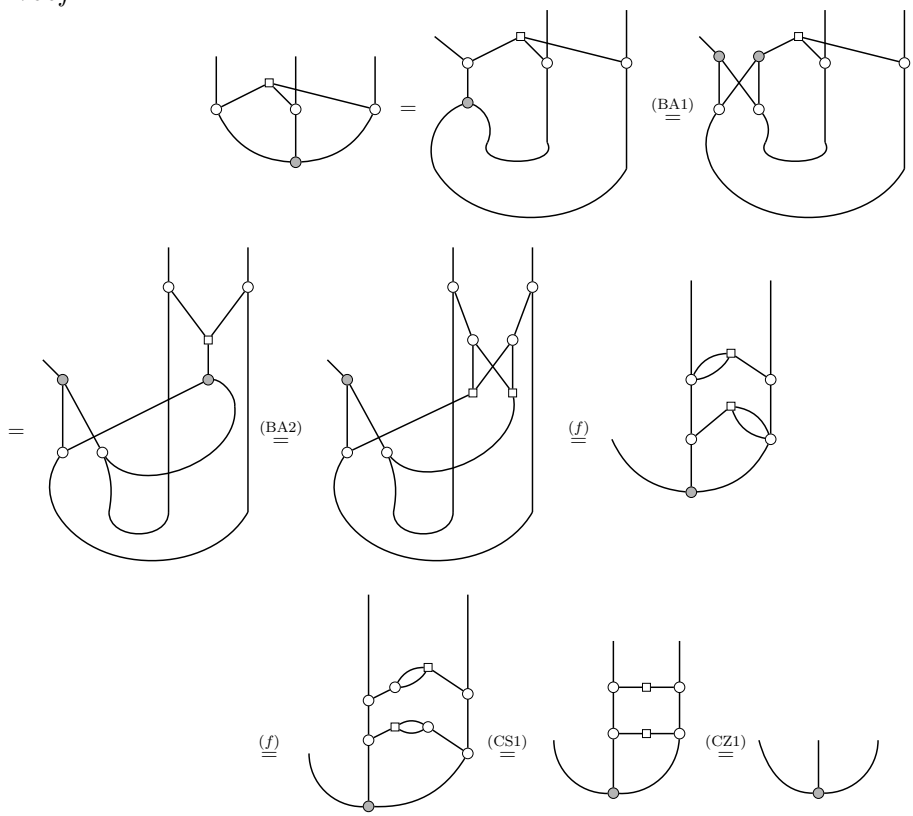
■

Lemma 3.1.2. The equation 3.3 holds true.



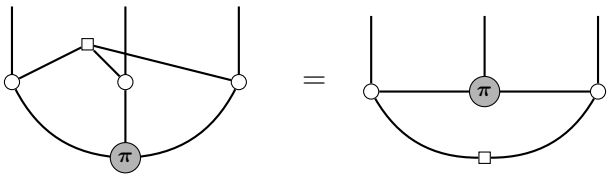
(3.3)

Proof.



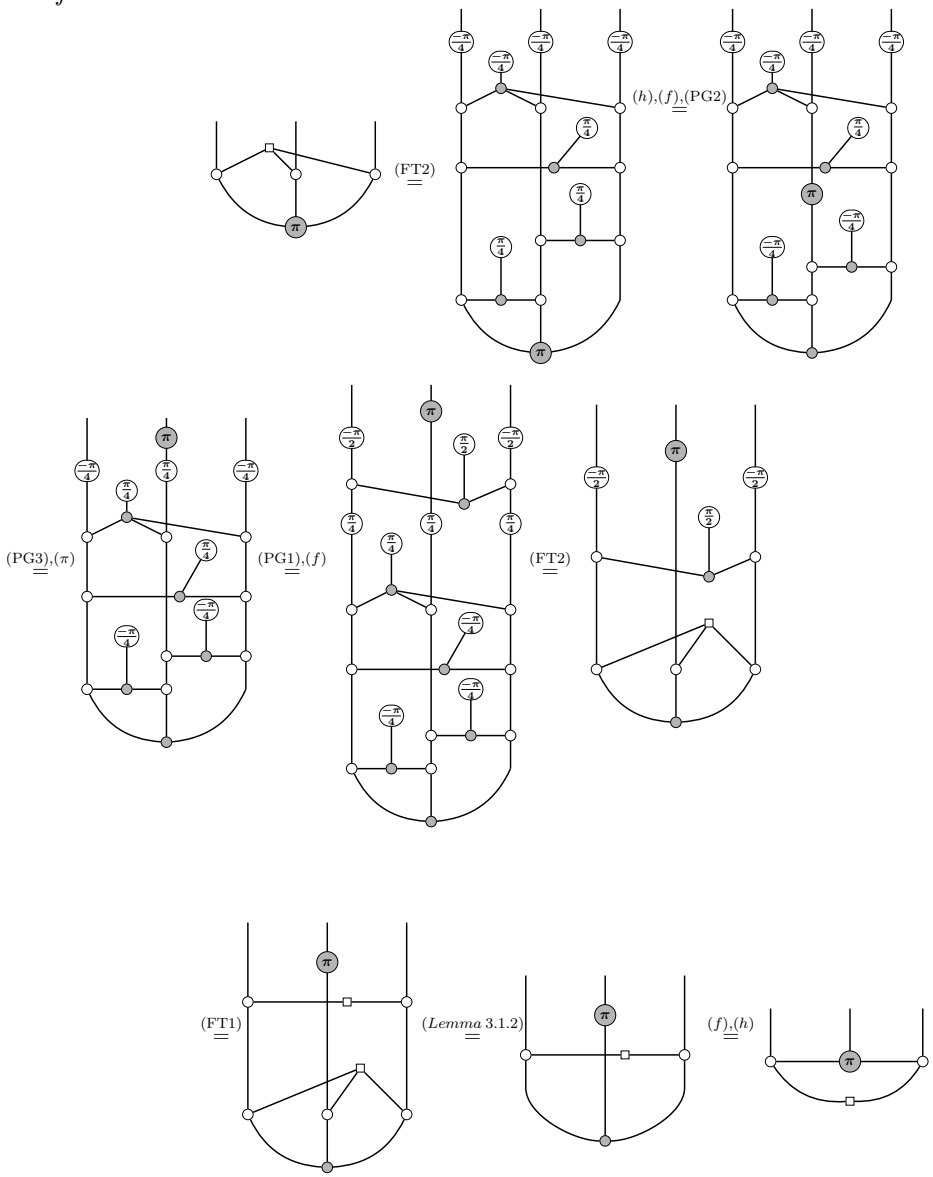
■

Lemma 3.1.3. The equation 3.4 holds true.



(3.4)

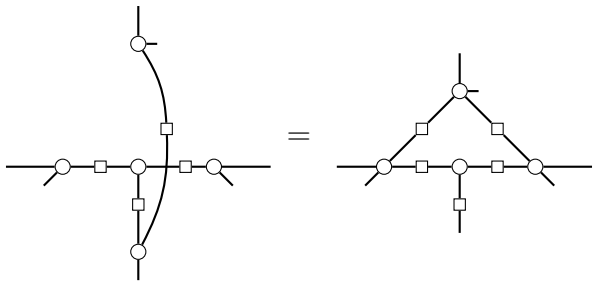
Proof.



In the proof above, we first use rule of (FT2) decompose the 3-ary H-box on the left-hand side of the equation and we get a bunch of phase gadgets. Then, by (π)

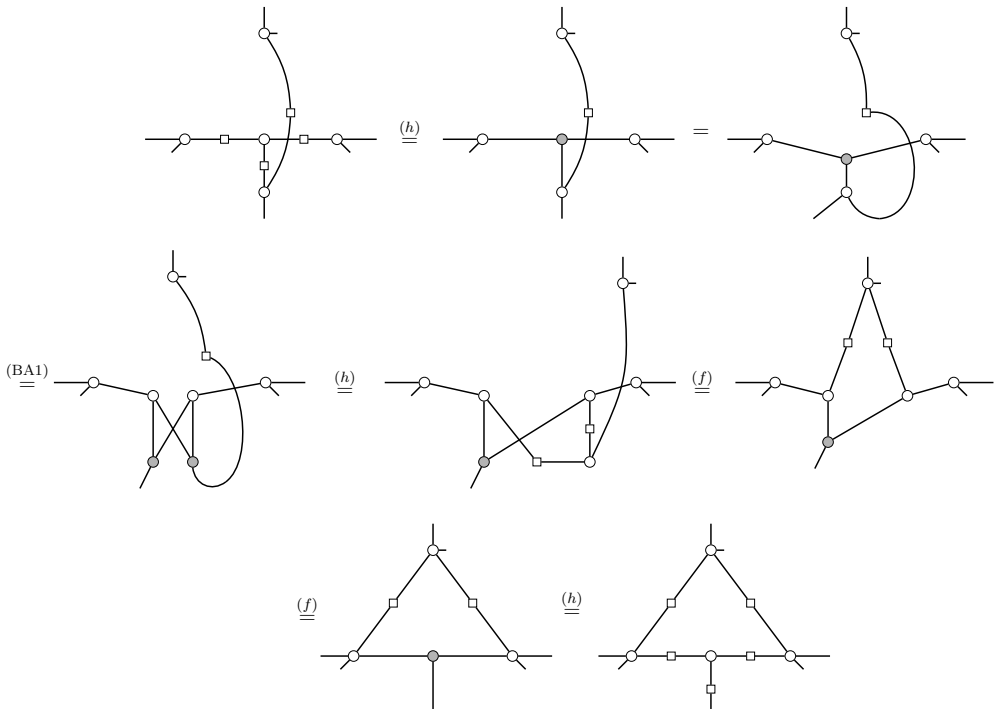
and (NPG), we pass the 2-ary X-spider of phase π through all these phase gadgets and flip their phases. By spider fusion, phase gadgets can freely pass through each other. By phase gadget fusion rule (PG1), we decompose a $\frac{\pi}{4}$ phase gadget into a $\frac{\pi}{2}$ phase gadget and another $\frac{-\pi}{4}$ phase gadget. Spider fusion allows us to split a spider of phase $\frac{-\pi}{4}$ into a spider of phase $\frac{\pi}{4}$ and another with $\frac{-\pi}{2}$. Next, apply (FT1) and (FT2) to compress phase gadgets and X-spiders into H-boxes, and apply the equation in lemma 3.1.2 to remove the 3-ary H-box. Finally, we fuse spiders together. ■

Lemma 3.1.4. The equation 3.5 holds true.



(3.5)

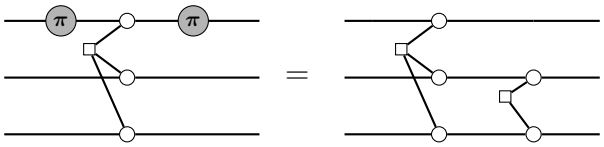
Proof.



■

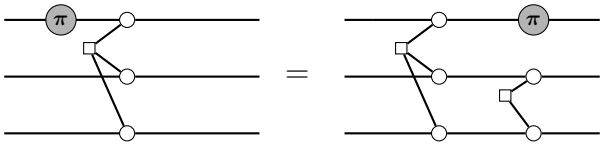
The next lemma will be used in the proof of lemma 3.1.9.

Lemma 3.1.5. The equation 3.6 holds true.

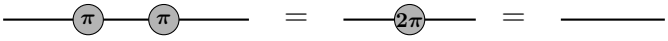


(3.6)

Proof. From the proof of lemma 3.1.3, we know that when a Pauli-X operation passes through a CCZ gate, we will get another CZ gate:



Combining this with the fact that

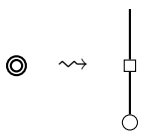


we can easily conclude that the equation 3.6 holds true. ■

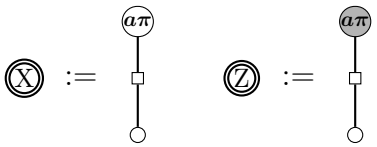
3.1.3 Measurement patterns

In this section, we discuss measurement patterns defined on fragments of the GGM state.

Double circle notation In this chapter, we draw a qubit in a resource state as a double circle to mean that there is a Hadamard gate on this qubit:



We also make this double circle notation compatible with the notation of Pauli X and Z measurement patterns defined in section 2.9.3 in the following way:



When there is a multi-qubit gate on a double circle qubit and another qubit, the Hadamard gate is always the first gate to be performed on the double circle qubit:

One of the most important features of GGM state is that we can deterministically obtain CZ gates using Pauli X measurements, which is described in more detail in lemma 3.1.6 and lemma 3.1.7.

Lemma 3.1.6. As shown in figure 3.4, by performing the measurement pattern defined on the left-hand side of the arrow, we implement two CZ wires and a Hadamard gate on the middle output qubit, up to Pauli Z errors.

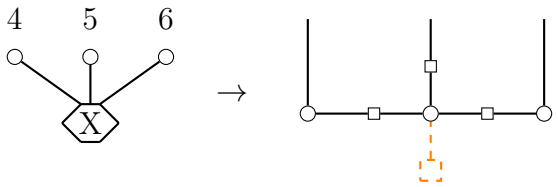
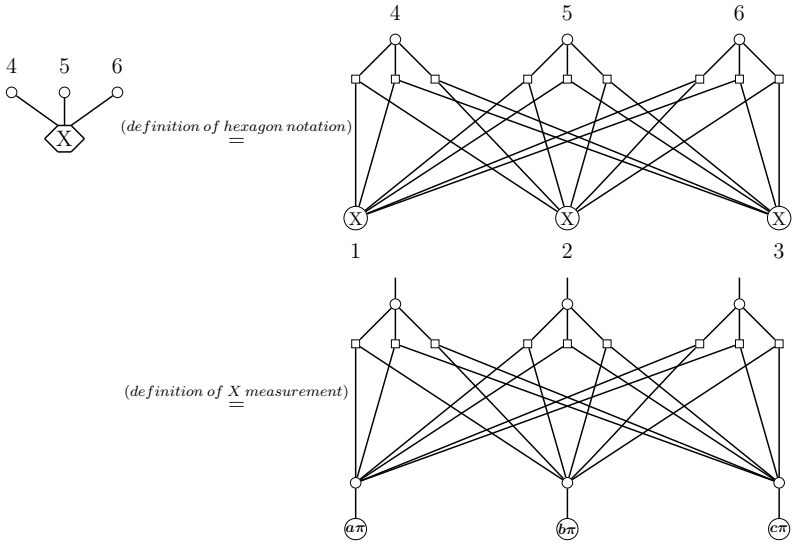


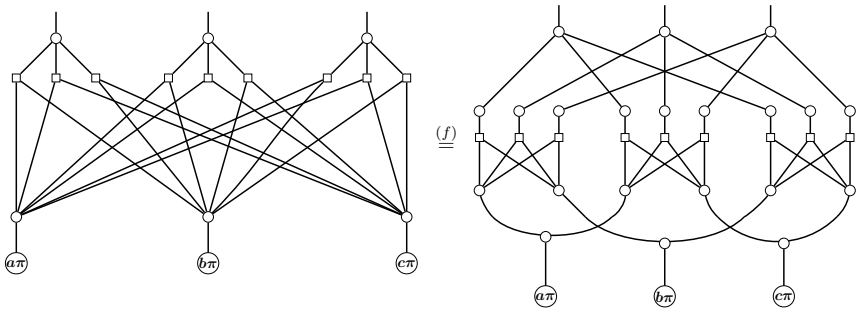
Figure 3.4: On the left-hand side of the arrow, we define a measurement pattern using the hexagon notation introduced in section 3.1.1. Qubits 4, 5, 6 are outputs. On the right-hand side we give the result of calculating this measurement pattern. The orange dashed lines is the notation introduced in section 2.9.2 depicting byproducts or errors.

Proof. First, we expand the hexagon notation on the left-hand side by its definition, and then calculate the measurement pattern using the method introduced in section

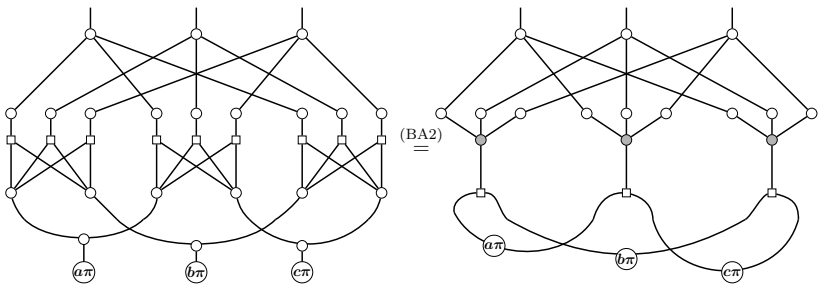
2.9.3.



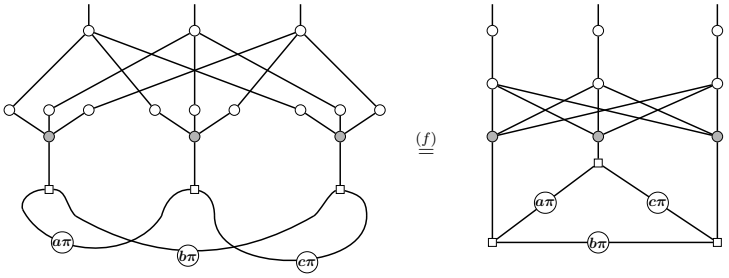
where a, b and c are measurement outcomes, and $a, b, c \in \{0, 1\}$. Next, we use the spider fusion rule to split the state into roughly three parts.



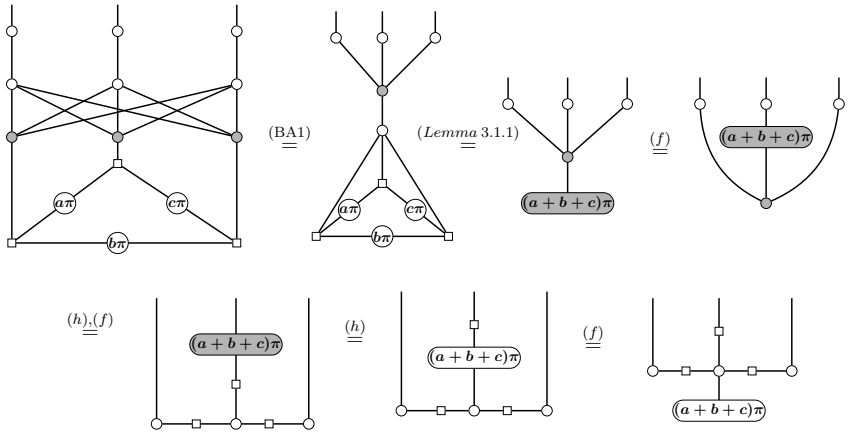
In each part, we apply the rule of (BA2).



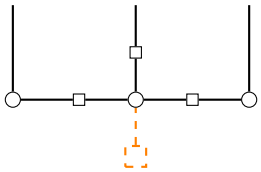
Then we again use spider fusion to combine three parts together



Continue to apply other rules of ZH-calculus and lemma 3.1.1, and we have



Here in the last state of the above derivations, we see that there is a Pauli Z error on the middle qubit. So, rewrite this error into dashed line representation, and we get our target state:



■

Lemma 3.1.7. As shown in figure 3.5, by performing the measurement pattern defined on the left-hand side of the arrow, we implement two CZ wires and a Hadamard gate on the middle output qubit, up to Pauli Z errors and CZ errors.

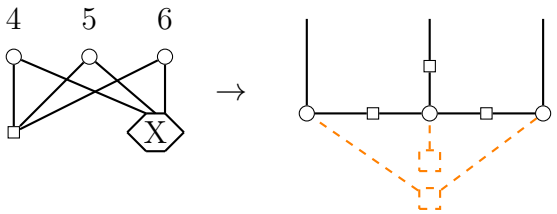
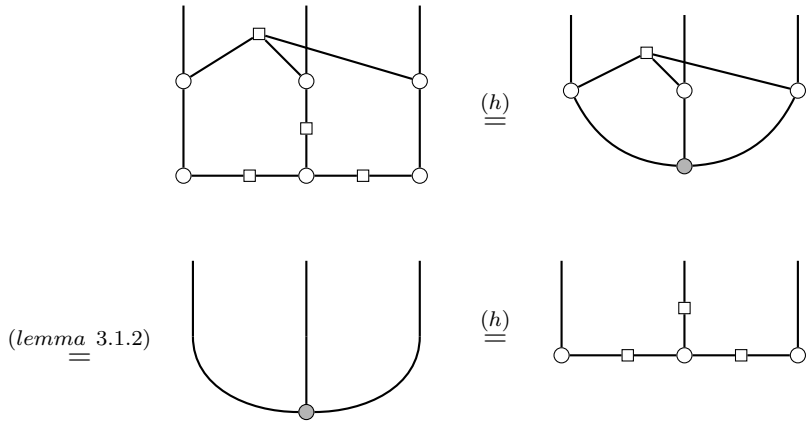


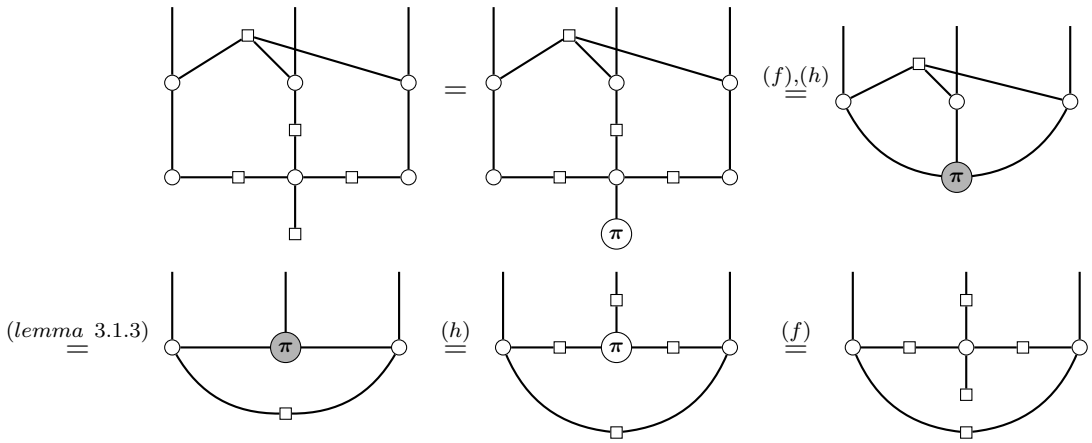
Figure 3.5: On the left-hand side of the arrow, we define a measurement pattern where qubits 4, 5, 6 are outputs. The notation introduced in section 3.1.1 is used. On the right-hand side we give the result of calculating this measurement pattern. The orange dashed lines is the notation introduced in section 2.9.2 depicting byproducts or errors.

Proof. This lemma follows immediately from lemma 3.1.6, lemma 3.1.2, and lemma 3.1.3. First, apply lemma 3.1.6 to calculate the result of measuring the hexagon in

Pauli X. If no Pauli Z error is produced in this measurement on the hexagon, then apply (h) and lemma 3.1.2 to the result and we get:



Otherwise, a Pauli Z error is produced in this measurement on the hexagon, then apply (f), (h), and lemma 3.1.3 to the result and we get:



This case, we have a Pauli Z error and a CZ error. Rewrite errors into dashed line representation, then we get our target state. ■

If we measure the hexagon in Z, then all relevant CZ wires are disconnected, which is explained in more detail in the following lemma.

Lemma 3.1.8. By performing the measurement pattern defined in figure 3.6, we disconnect all CZ wires and produce Pauli Z errors.

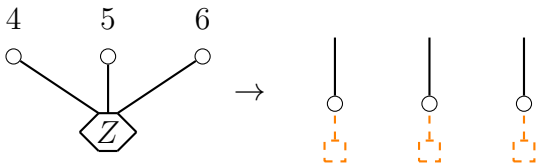
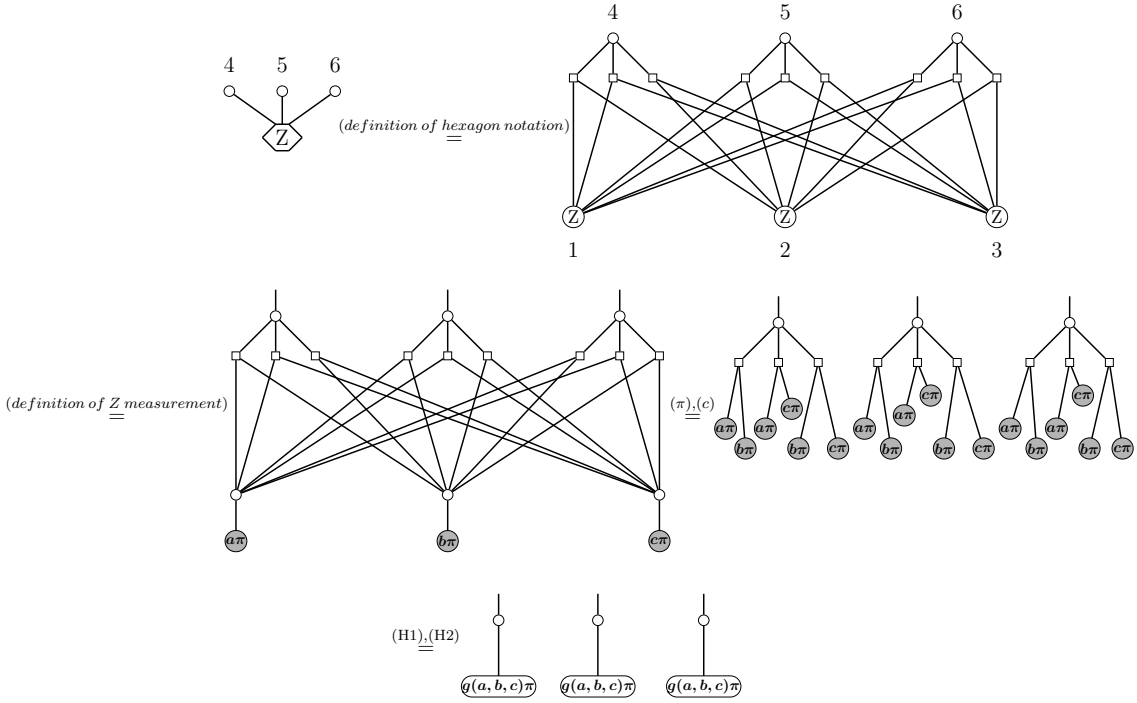


Figure 3.6: On the left-hand side of the arrow, we define a measurement pattern where qubits 4, 5, 6 are outputs. The notation introduced in section 3.1.1 is used. On the right-hand side we give the result of calculating this measurement pattern. The orange dashed lines is the notation introduced in section 2.9.2 depicting byproducts or errors.

Proof.



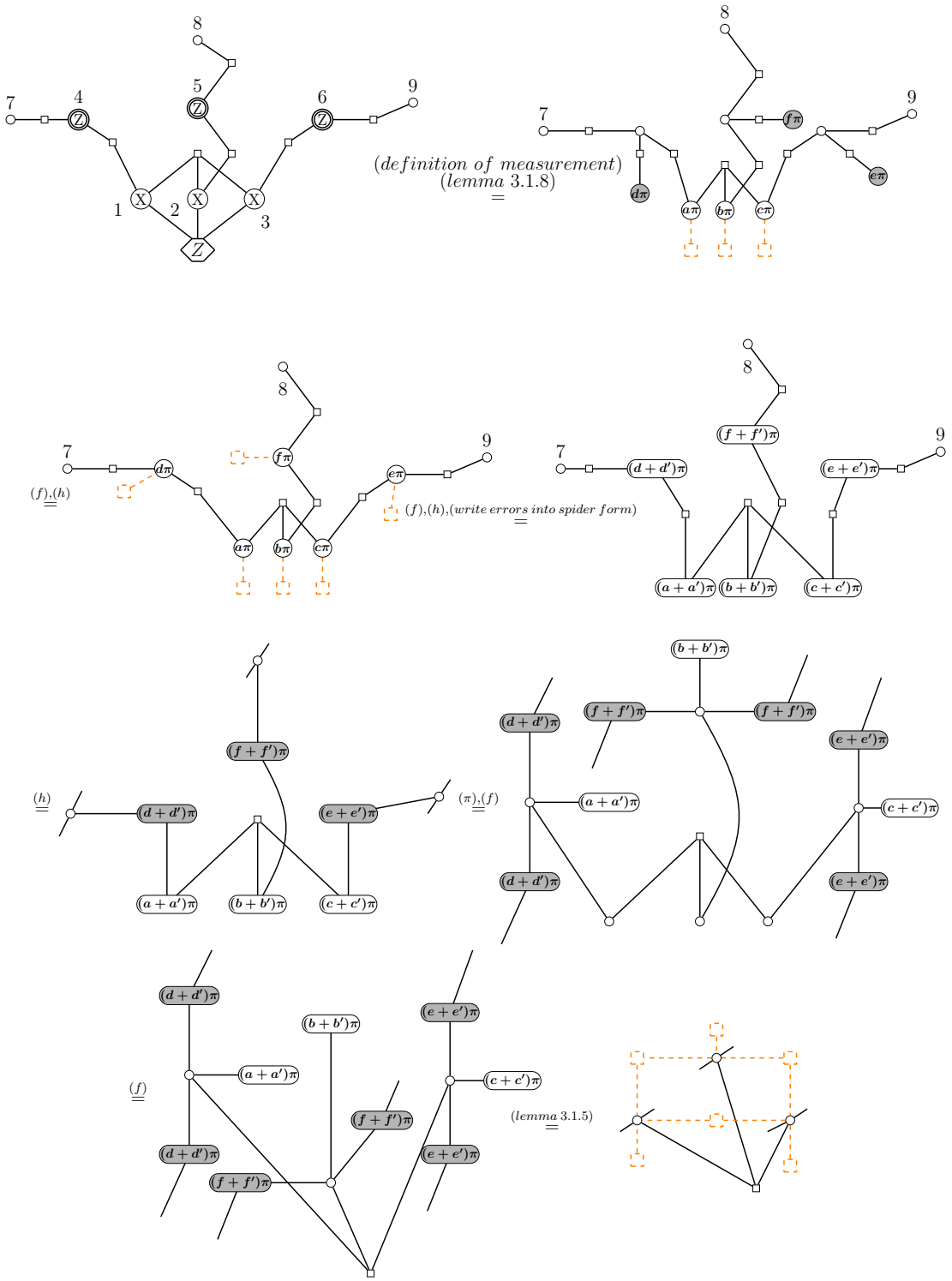
where $g(a,b,c) \in \{0,1\}$ and the value of $g(a,b,c)$ depends on a, b, and c. Rewrite errors into dashed line representation, then we get our target state. ■

In the next lemma, we show that we can implement a nearest neighbuor CCZ gate by performing Pauli X and Z measurements on a fragment.

Lemma 3.1.9. The measurement pattern defined in figure 3.7 implements a CCZ gate on qubits 7, 8, and 9, up to CZ and Pauli Z errors.

Proof. By definition of measurements, lemma 3.1.8, and rules of ZH-calculus, we

have:



In the last step of derivations, we apply lemma 3.1.5 three times, and rewrite errors into dashed line notation. ■

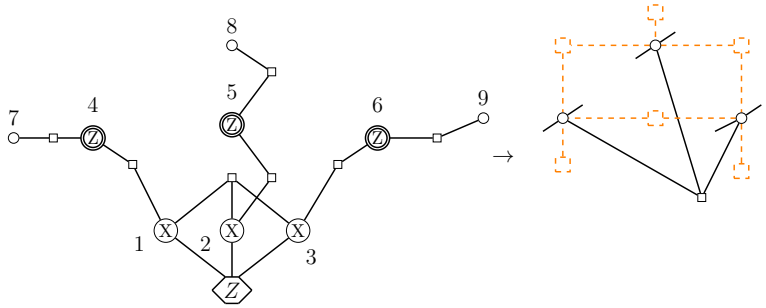


Figure 3.7: On the left-hand side of the arrow, we define a measurement pattern where qubits 7, 8, 9 are both inputs and outputs. The hexagon notation introduced in section 3.1.1, the notation of double circle defined in the beginning of this section, and the notation of measurement patterns in section 2.9.3 are used. On the right-hand side we give the result of calculating this measurement pattern. The orange dashed lines is the notation introduced in section 2.9.2 depicting byproducts or errors.

3.1.4 Correctness of the protocol

In this section, we show that the protocol of GGM state based MBQC model is correct. The protocol includes two essential parts: the implementation of the hexagonal lattice state in figure 3.15 and the implementation of the nearest neighbour CCZ gate. These two implementations are done by performing Pauli X and Z measurements on the GGM state.

The first step The first step of the protocol is to measure some hexagons in X, as shown in figure 3.8. In the figure 3.8, all the hexagons, to which three qubits connecting do not belong to the same CCZ hyperedge, are measured in X.

By lemma 3.1.6, the resulting state after performing these measurements is shown in figure 3.9. Getting this state is important, since we will implement all gates we need by performing Pauli X and Z measurements on this state.

Implementation of hexagonal lattice Now we will show how to get the lattice state in figure 3.15 step by step.

First, we need to measure the rest of hexagons in the state in figure 3.9 in Pauli X, as shown in figure 3.10. By lemma 3.1.7, after these measurements, we get the state in figure 3.11.

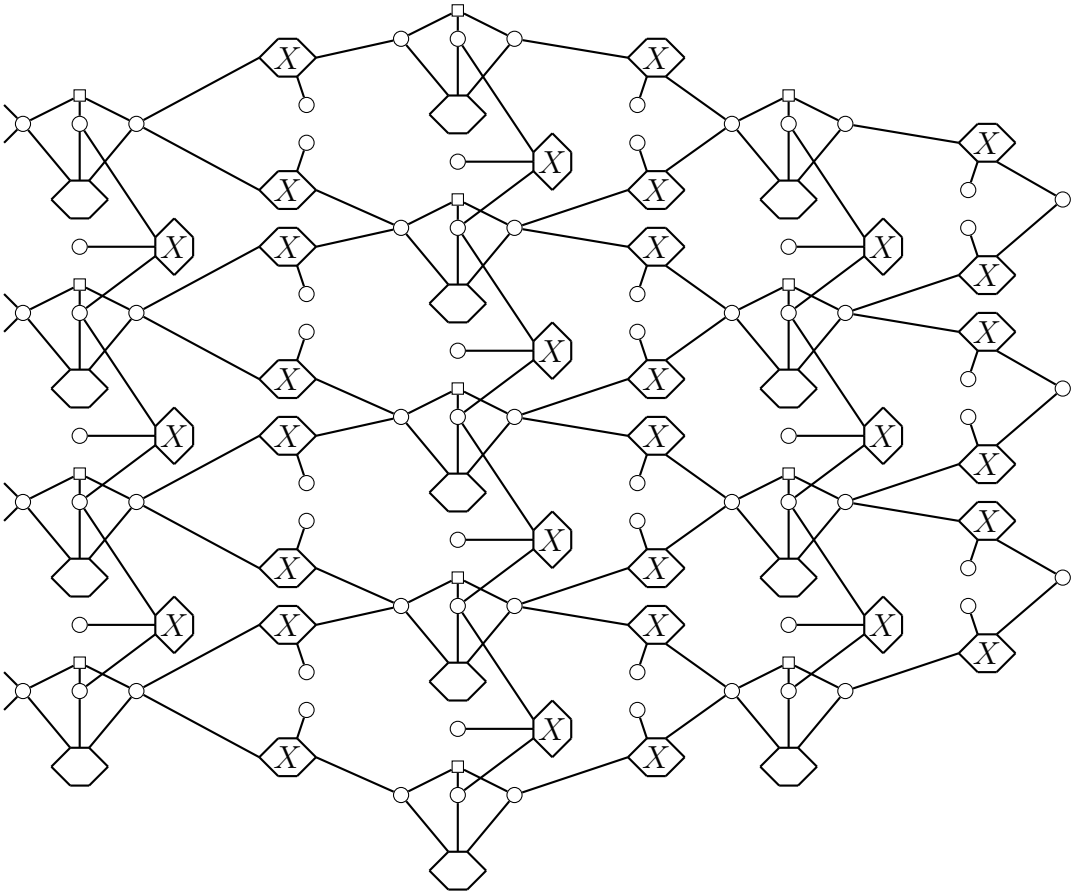
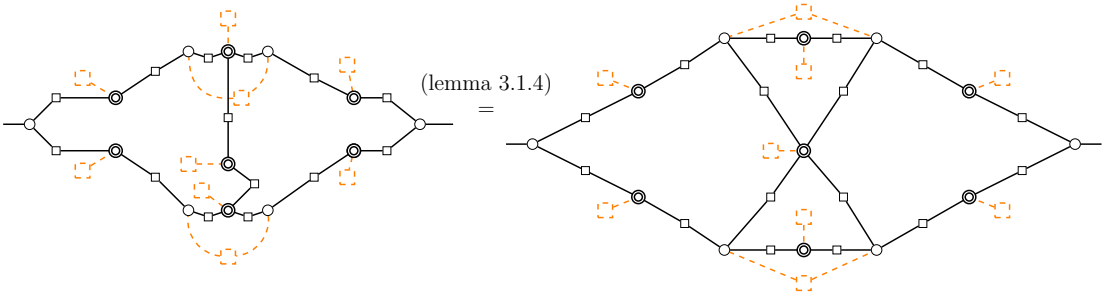


Figure 3.8: The first step of GGM measurement protocol. Some hexagons are measured in X.

Next, to get one step closer to the hexagonal lattice, we consider transforming each unit in the state in figure 3.11. By lemma 3.1.4, we transform each unit of this state to an equivalent shape as shown in the following equation:



On the right-hand side of the above equation, the central qubit is connected to four surrounding qubits by CZ wires.

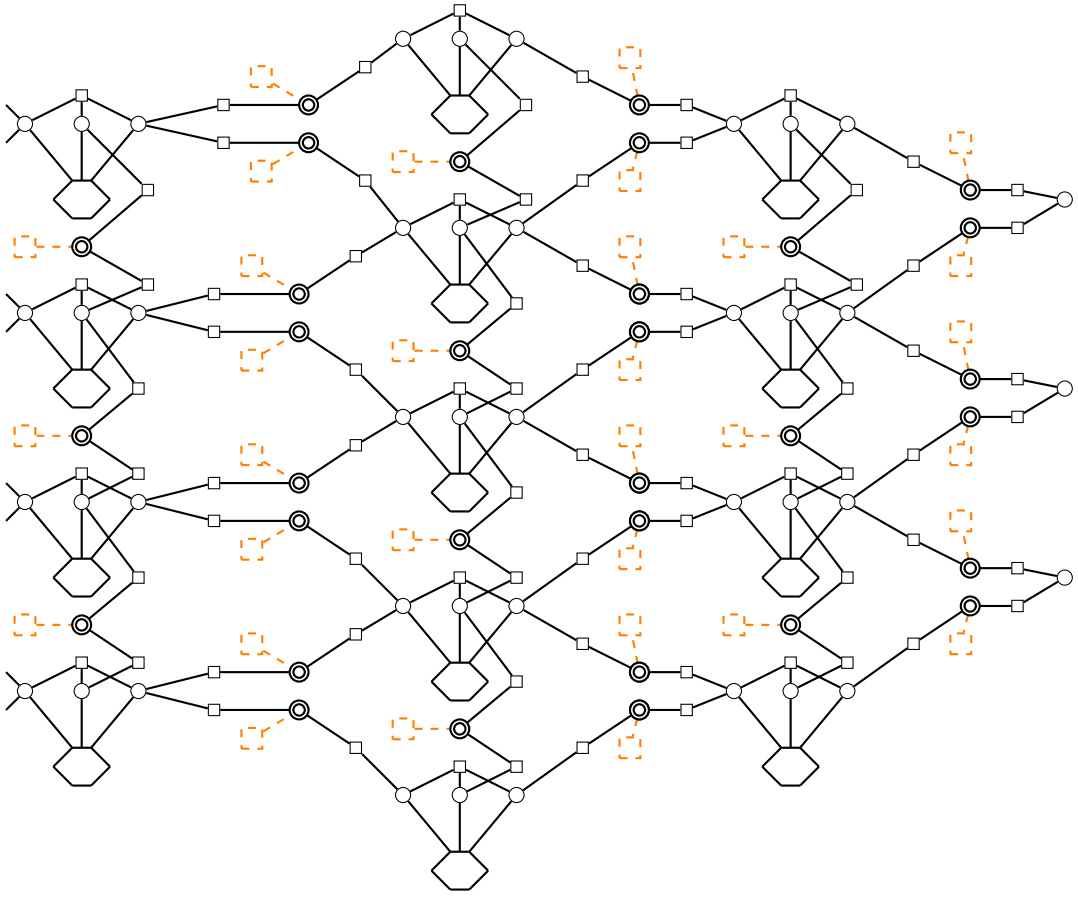


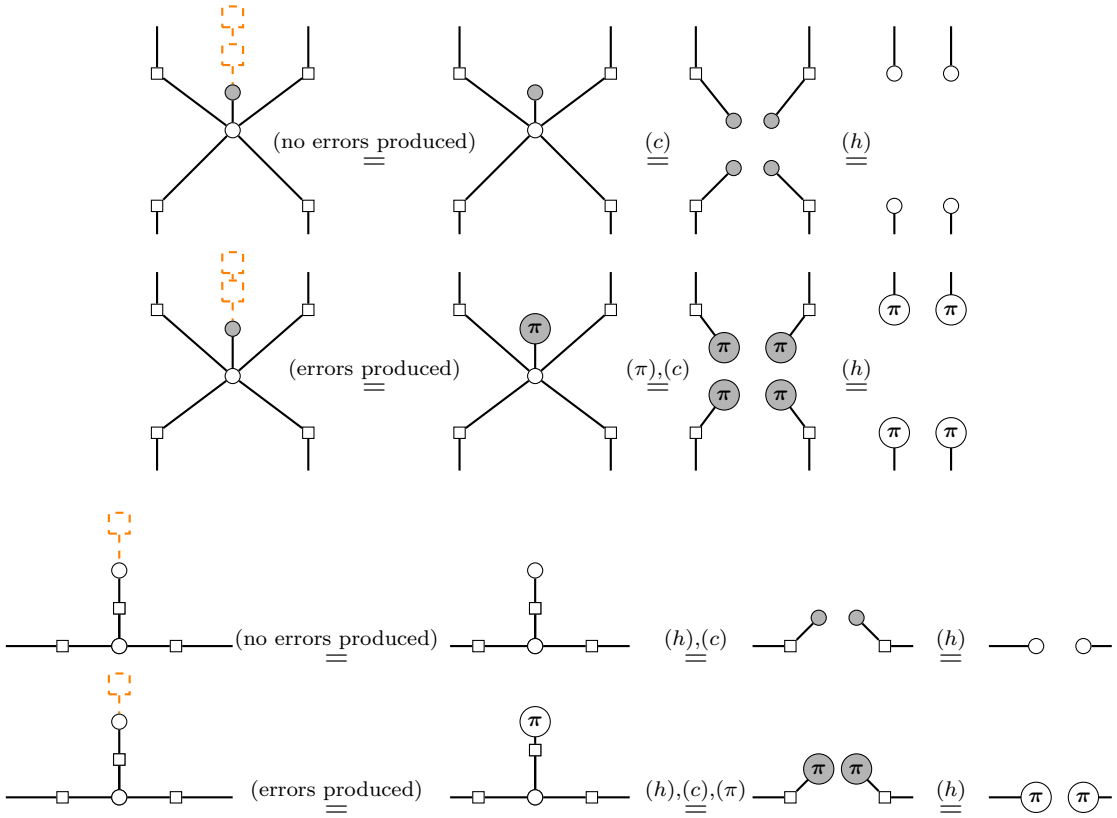
Figure 3.9: The resulting state after performing the first step of the GGM state based measurement protocol. The double circle qubit notation defined in section 3.1.3 is used.

By applying this transformation to each unit of the state in figure 3.11, we get the state in figure 3.12.

The final step is to perform measurements in Pauli Z and X so as to shape the state in figure 3.12 into the hexagonal lattice state in figure 3.15. We use pink to colour qubits that are to be measured in Pauli Z, and green to colour qubits that are to be measured in Pauli X, as shown in figure 3.13. Among all the coloured qubits in the figure above, those qubits on which there is a Hadamard gate are coloured with green.

After performing Pauli X and Z measurements on coloured qubits, by method of calculating measurements introduced in section 2.9.3, we get the state in figure 3.14. In figure 3.14, we write measurement errors in orange-dashed-line notations. Then by applying rules of (h), (π), (f), and (c), the state in figure 3.14 is equal

to the hexagonal lattice in figure 3.15, up to Pauli Z and CZ errors, as largely explained by the following derivations:



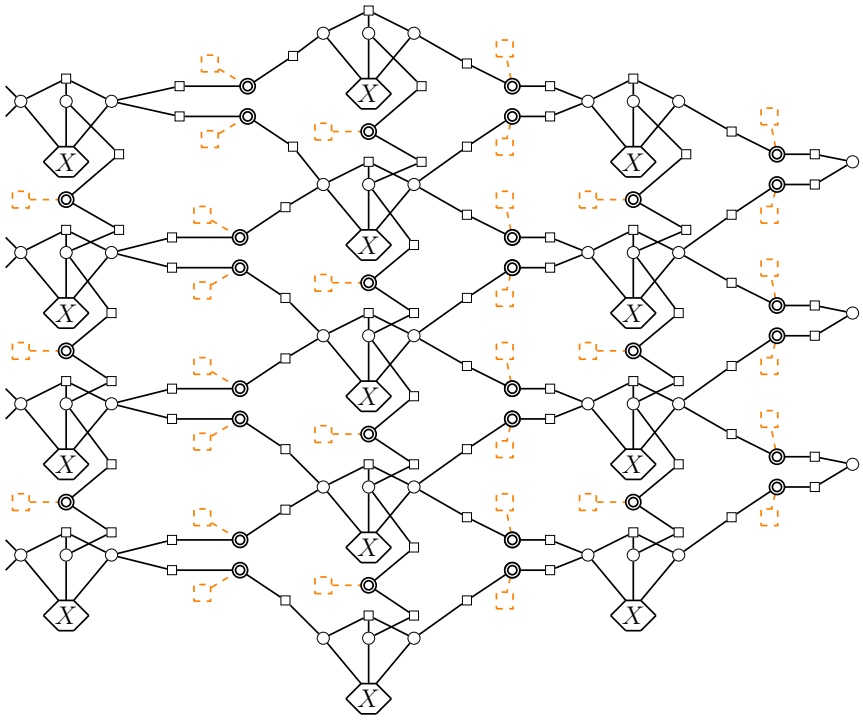


Figure 3.10: Performing X measurements on all the hexagons in the state in figure 3.9.

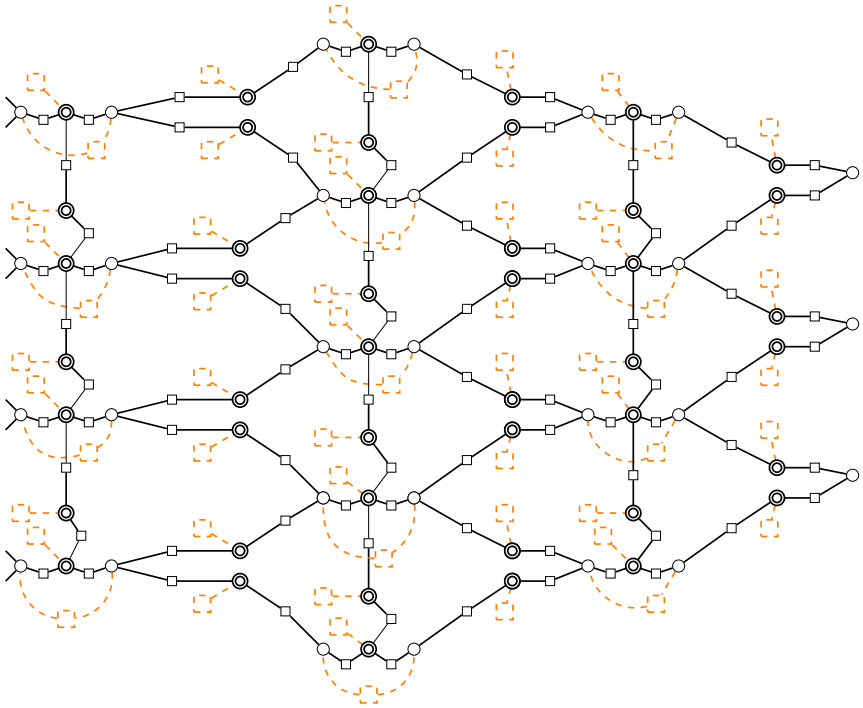


Figure 3.11: The resulting state after performing X measurements on all the hexagons in the state shown in figure 3.9. The double circle qubit notation defined in section 3.1.3 is used.

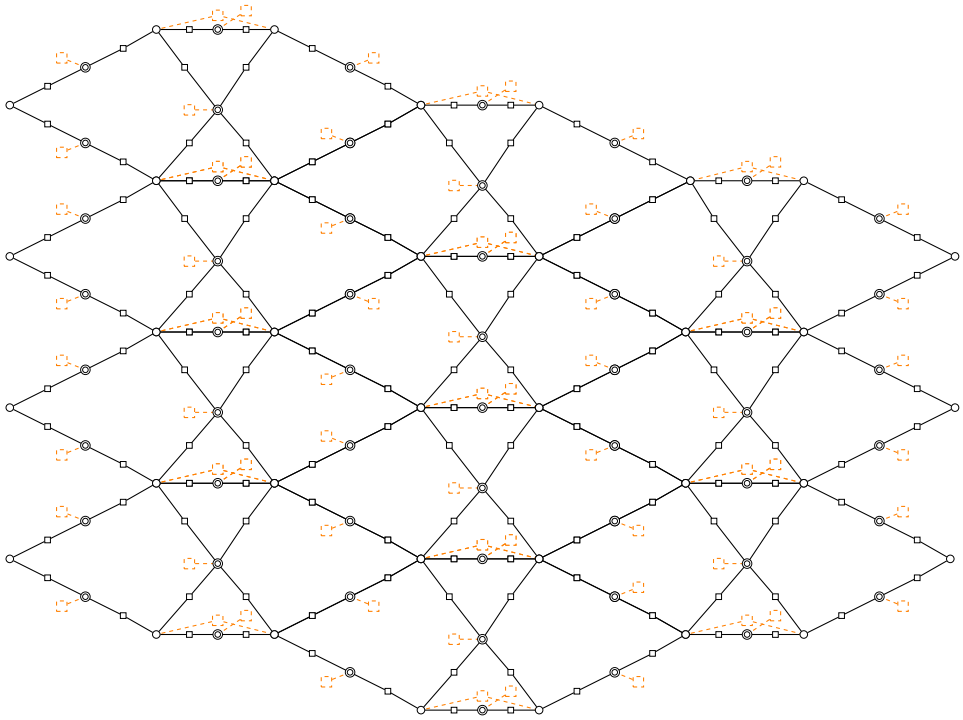


Figure 3.12: The resulting state after transforming each unit in the state 3.11.

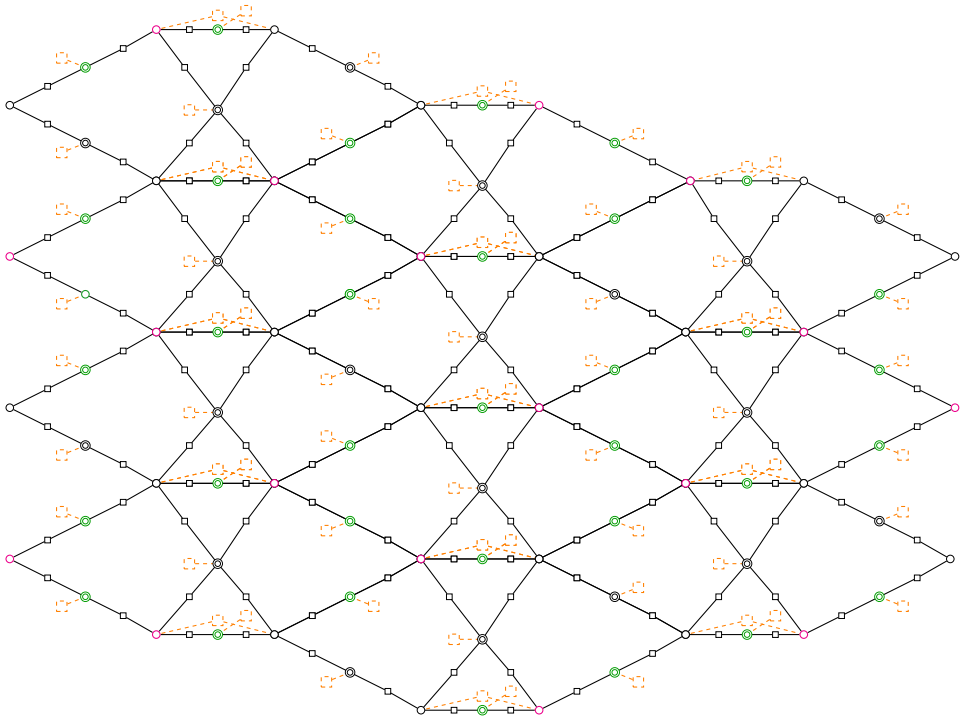


Figure 3.13: Colouring qubits in the state in figure 3.12 that are to be measured in Pauli Z and X: pink is used to colour qubits that are to be measured in Pauli Z; green is used to colour qubits that are to be measured in Pauli X.

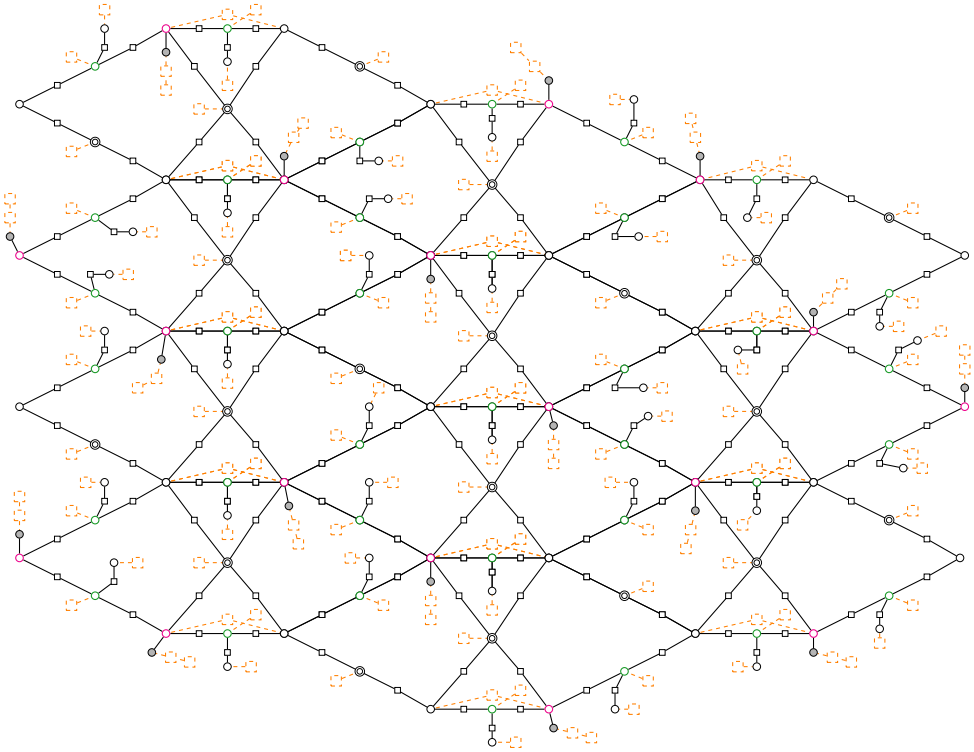


Figure 3.14: The resulting state after performing Pauli X and Z measurements defined in figure 3.13.

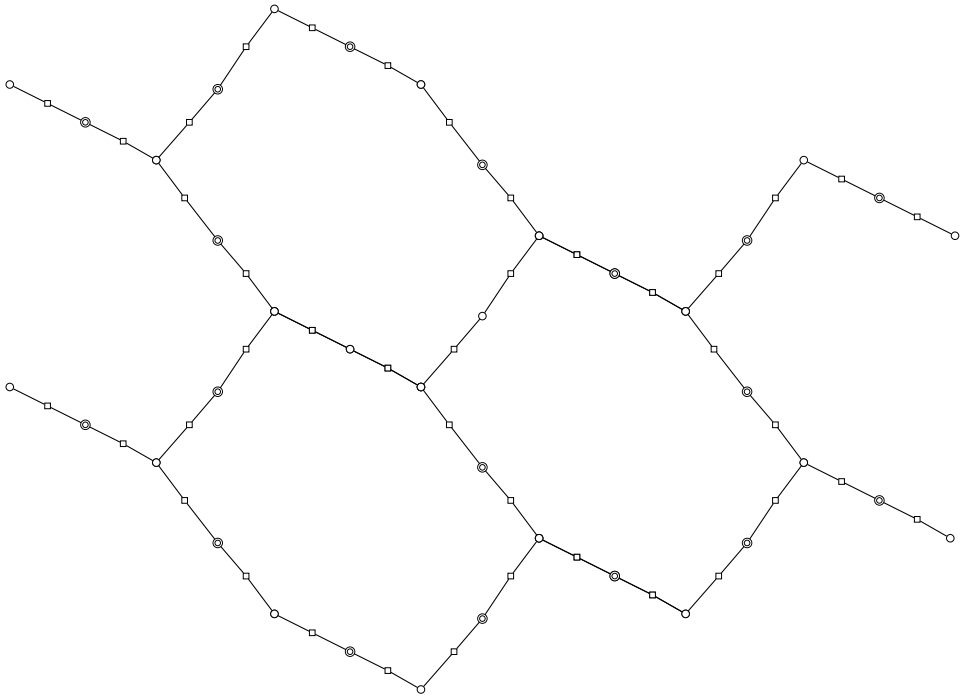


Figure 3.15: The hexagonal lattice state that can achieve any Clifford operation via X and Z measurements.

Implementation of nearest neighbour CCZ gates By lemma 3.1.9, we can implement nearest neighbour CCZ gates, up to CZ an Pauli Z errors, by performing Pauli X and Z measurements on the state in figure 3.9, since the measurement pattern given in lemma 3.1.9 is exactly defined on a fragment of this state.

Proposition 3.1.10. The GGM state based MBQC protocol is correct.

Proof. The GGM state based MBQC protocol is correct, since we have shown that we can implement the hexagonal lattice state in figure 3.15 and nearest neighbour CCZ gates by this protocol. ■

Based on our discussions, we can also easily verify the following three features of the GGM model included in theorem 1 in [12]:

1) *We can achieve universal computations using only Pauli measurements.* We use the idea introduced in section 2.7 to show this fact. We choose $\{CCZ, Hadamard\}$ as the universal gate set. It has been shown that we can implement the hexagonal lattice in figure 3.15 up to CZ errors and Pauli Z errors. By [9], any Clifford gate can be implemented in parallel by Pauli measurements on the state in figure 3.15, up to Pauli X and Z errors. Since the SWAP gate and the Hadamard gate are both Clifford gates, it follows immediately that we can implement SWAP and Hadamard gates up to Pauli X and Z and CZ errors. In addition, we have also shown that nearest neighbour CCZ gates can be implemented up to Pauli Z and CZ errors. Combining nearest neighbour CCZ gates with SWAP gates, we can implement CCZ gates on arbitrary three qubits as illustrated in figure 3.16. Then, let the information flows from the bottom to the top in the GGM state, we can implement CCZ gates and Hadamard gates and compose them in any manner. Therefore, universal computations can be achieved.

2) *The scheme is deterministic.* This fact is now obvious. Errors produced in this protocol are Pauli X and Z and CZ errors. Pauli X and Z errors can pass through Hadamard gates by rule (h) producing only Pauli X and Z errors. In proof for lemma 3.1.5 we see a Pauli X error can pass through a CCZ gate and produce a CZ error. CZ errors are Clifford and thus can be corrected by Pauli measurements on the hexagonal

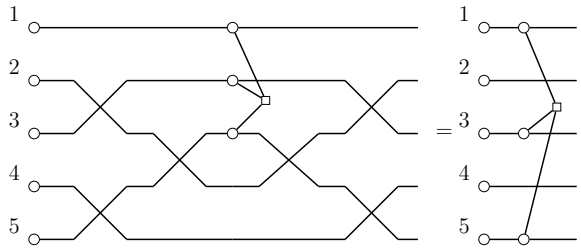


Figure 3.16: We can use SWAP gates to "slide" the nearest neighbour CCZ gate onto our desired qubits and use additional SWAP gates to cancel out all SWAP gates. Five qubits 1, 2, 3, 4, and 5 are prepared. To achieve a CCZ gate on qubits 1, 3, and 5 on the right-hand side, we compose a nearest neighbour CCZ gate on 1, 2, and 3 and multiple SWAP gates on the left-hand side.

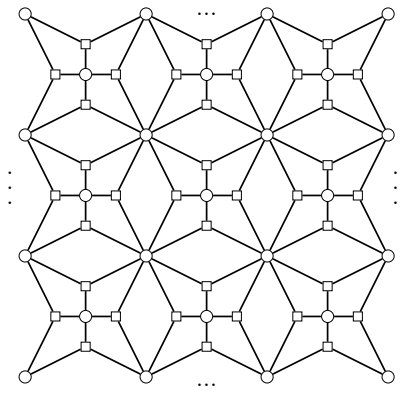
lattice state in figure 3.15. Therefore, we can use the technique of feed-forward introduced in section 2.9.2 to correct errors to implement deterministic computations.

3) *All CCZ gates and SWAP gates can be implemented in parallel.* This fact holds true, because measurements to implement CCZ gates or SWAP gates are independent to each other, and Pauli X and Z errors and CZ errors can pass through SWAP or CCZ gates without producing errors other than Pauli X and Z errors and CZ errors. For justifications of errors passing through gates: Obviously Pauli Z and X errors can freely pass through SWAP gates, and by (f) Pauli Z errors can freely pass through CCZ gates; in proof for lemma 3.1.5 we see a Pauli X can error passes through a CCZ gate and produce a CZ error.

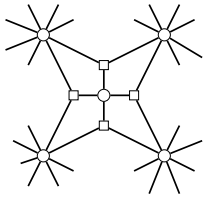
3.2 Union Jack state based MBQC model

3.2.1 Introduction to the Union Jack state

The ZH-diagram of the Union Jack state is presented as follows:



The Union Jack state is constructed by tiling the following small bricks up:



The CCZ hyperedges in the Union Jack state are arranged in a regular pattern.

We can see that the construction of Union Jack state is much simpler and more elegant compared with that of a GGM state. However, the justification for the construction of Union Jack state is profound. The idea of *symmetry protected topological order* is used in the construction of the Union Jack state, and it is also responsible for the fact that universal computations can be achieved using single-qubit Pauli measurements on this state [11]. We recommend interested readers to read [11] for a detailed description of the Union Jack state MBQC protocol.

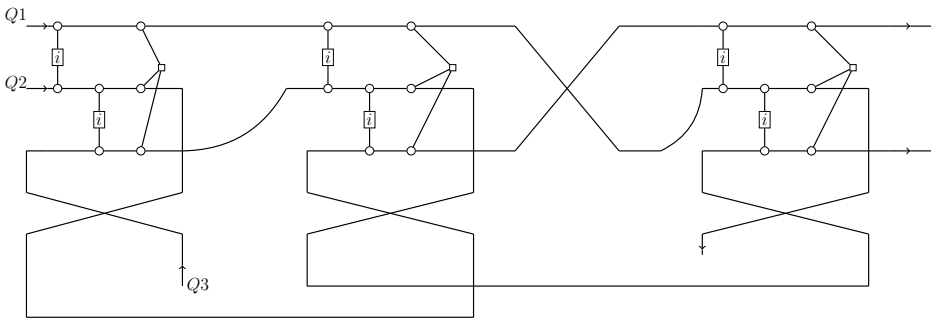
3.2.2 Correctness of the protocol

One essential part of the protocol is the implementation of CCZ gates. CCZ gates belong to the universal gate set $\{CCZ, Hadamard\}$ and thus serve as an essential evidence for universality of the Union Jack state model. Now we use ZH-calculus to

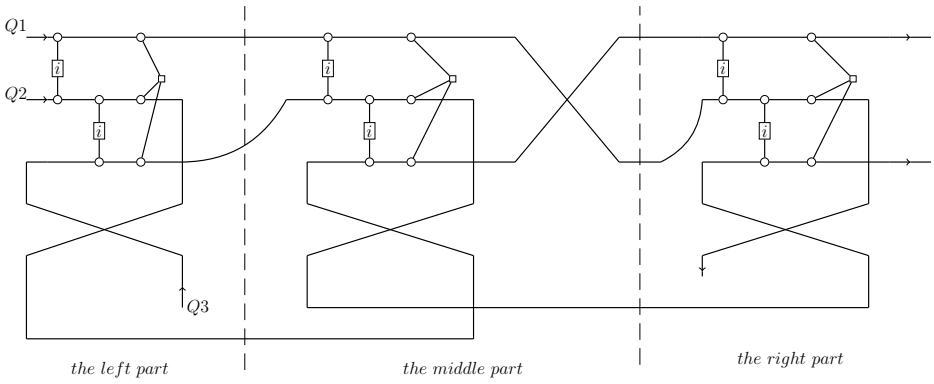
show that CCZ gate can be implemented by Pauli measurements on the Union Jack state. The measurement pattern implementing a CCZ gate is illustrated in figure A.1. It has been shown in [18] using ZH-calculus that the gate U_I and the SWAP gate can be implemented by performing corresponding measurement patterns on fragments of Union Jack state. So, we only need to prove that by composing U_I and the SWAP gates in the way shown in figure A.1, we implement a CCZ gate.

Proposition 3.2.1. The measurement pattern in figure A.1 implements a CCZ gate, up to CZ errors.

Proof. Translate the measurement pattern in figure A.1 into a ZH-diagram and we get:

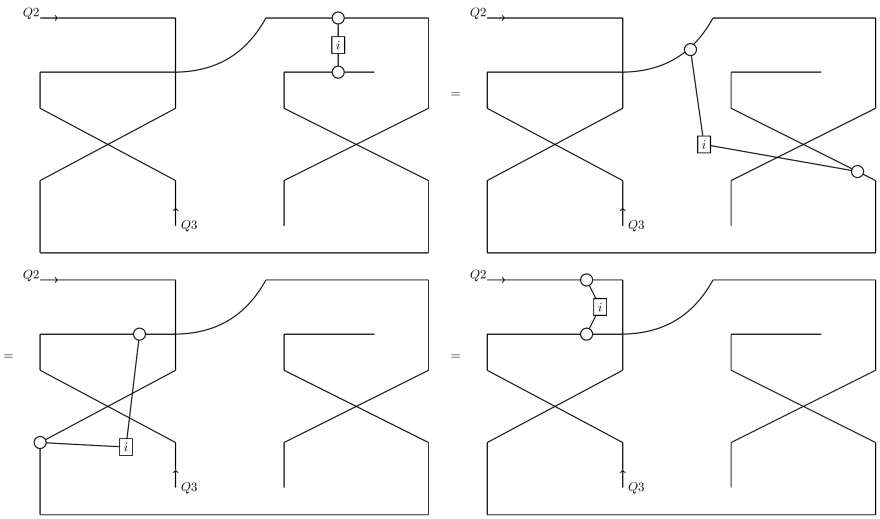


We can roughly divide this diagram into the left part, the middle part and the right part according to the distribution of gates:

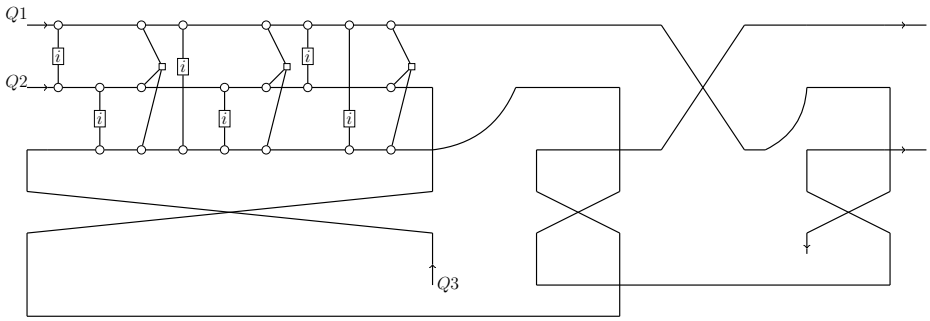


By connections of wires and spider fusion law (f), we can easily "slide" gates in the middle and the right part to the left part of the diagram. For example, the second

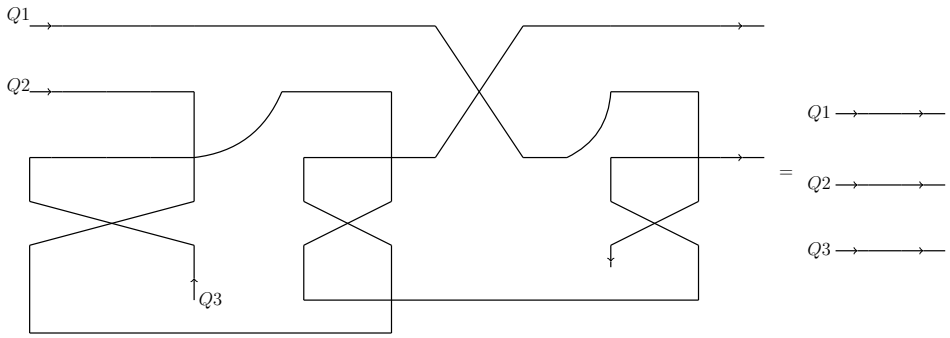
$\sqrt{CZ}$ gate in the middle part "slide" to the left part in the following way:



After sliding all the gates in the middle and the right part to the left part, we get:



By equations (CZ1), (CZ2), and (CZ3), two $\sqrt{CZ}$ gates merge into one CZ gate, two CZ gates cancel out with each other, and two CCZ gates cancel out with each other. In addition, we have the following fact:



Therefore, we conclude that the measurement pattern in figure A.1 implements a CCZ gate, up to CZ errors. ■

4

From Hypergraph to Phase Gadget MBQC

In this chapter, we investigate the link between hypergraph and phase gadget state MBQC. We have seen in chapter 3 two hypergraph state based MBQC models, namely GGM state model and Union Jack state model. Although their resource states are constructed in different ways, they share two features: (i) they both use CCZ gates as the unique building blocks: qubits in both states are entangled purely by CCZ gates; (ii) both models can achieve universal computations by only single-qubit Pauli measurements. Therefore, one of the obvious questions to ask is: Is it possible to choose a gate P other than the CCZ gate, and use P as the unique building blocks to construct a resource state for MBQC where universal computations can be achieved by performing only Pauli measurements? The answer turns out to be "Yes". Our discussions in this chapter will justify this answer and will also serve as motivations for our construction in chapter 5.

In sections 4.1 and 4.2, we apply Graphical Fourier theory introduced in 2.6 and transform the Union Jack state to an equivalent state composed purely by trinary $\frac{\pi}{4}$ phase gadgets. This equivalence serves both as an inspiration that we can use trinary $\frac{\pi}{4}$ phase gadgets to construct a new universal resource state and as a bridge between the two families of MBQC models: hypergraph state MBQC and phase gadget MBQC.

4.1 Sign-related decomposition

If we view a qubit as a vertex and a CCZ gate as a triangle, then we can imagine that a Union Jack state is composed by many triangles. For convenience, we do not consider the existence of a border in the Union Jack state. Then, it is clear that there are two types of vertices: we say that a vertex shared by four triangles is of type I and colour it in red, and a vertex shared by eight triangles is of type II and colour it in blue, as illustrated in figure 4.1. We now apply the Graphical Fourier

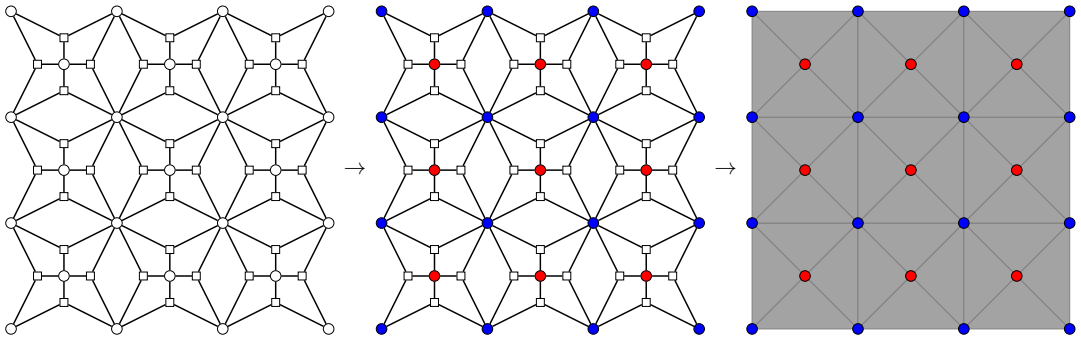
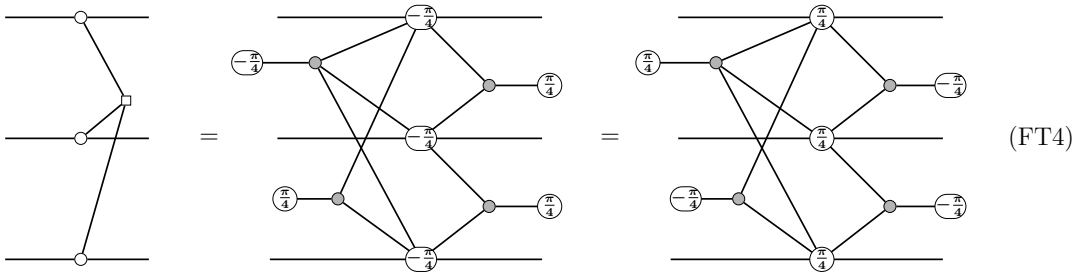


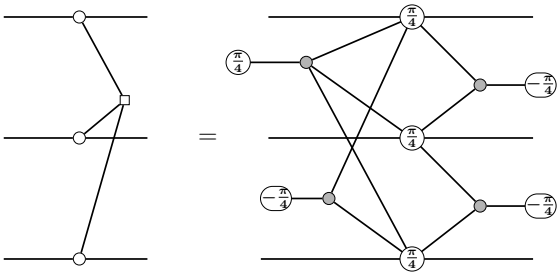
Figure 4.1: Viewing the Union Jack state as being composed by triangles, where type I vertices are coloured in red and type II vertices are coloured in blue. Type I vertices are vertices that are shared by four triangles and type II vertices are the ones shared by eight triangles.

theorem to decompose each CCZ gate in the Union Jack state. One important idea to decompose CCZ gates in Union Jack state is that the Fourier decomposition of a CCZ gate is not unique. Since a CCZ gate is self-adjoint, the outcome of taking the conjugate of it is still a CCZ gate, which means we can flip all the phases in its Fourier decomposition and still retain the same matrix. Expressing this idea in

ZH-calculus, the following equations (NFT3) and (FT4) hold true.

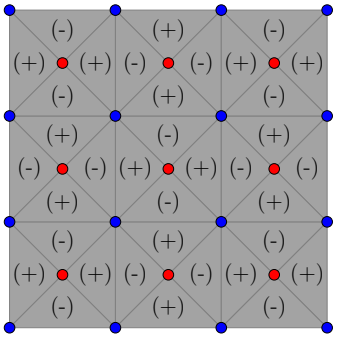


(FT4)



(NFT3)

We say that two triangles are adjacent if they share two vertices. Next, we use (+) and (−) to mark each triangle in Union Jack state such that any two adjacent triangles are marked with different signs, illustrated as follows:



It is easy to check that this can be done. In addition, it is not hard to check the following facts:

Fact 4.1.1. Type I vertices are shared by exactly 2 triangles marked with (+) and 2 triangles marked with (−).

Fact 4.1.2. Type II vertices are shared by exactly 4 triangles marked with (+) and 4 triangles marked with (−).

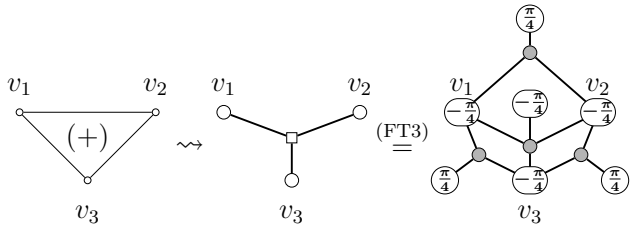
Fact 4.1.3. For any two different vertices that are shared by two triangles, they are shared exactly by a triangle marked with (+) and another marked with (−).

Fact 4.1.4. For any two different vertices that belong to the same triangle, they are shared exactly by two triangles.

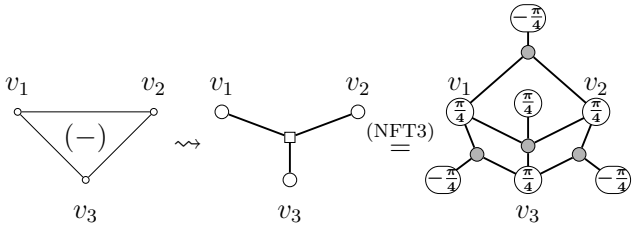
We define a *sign-related decomposition*: if a triangle is marked with (+), then we use (FT3) to decompose the corresponding CCZ gate; if a triangle is marked with (−), then we use (NFT3) to decompose the corresponding CCZ gate.

Recall that vertices and qubits are treated as the same, and triangles and CCZ gates are also treated as the same in our discussion in this section.

For a CCZ gate or triangle on vertices v_1, v_2, and v_3, if we apply (FT3) to decompose it, then after decomposition we will have a phase of $\frac{-\pi}{4}$ on vertex v_1, a phase of $\frac{-\pi}{4}$ on vertex v_2, a phase of $\frac{-\pi}{4}$ on vertex v_3, a $\frac{\pi}{4}$ binary phase gadget on vertices v_1 and v_2, a $\frac{\pi}{4}$ binary phase gadget on vertices v_1 and v_3, a $\frac{\pi}{4}$ binary phase gadget on vertices v_2 and v_3, a $\frac{-\pi}{4}$ trinary phase gadget on vertices v_1, v_2, and v_3:



For a CCZ gate or triangle on vertices v_1, v_2, and v_3, if we apply (NFT3) to decompose it, then after decomposition we will have a phase of $\frac{\pi}{4}$ on vertex v_1, a phase of $\frac{\pi}{4}$ on vertex v_2, a phase of $\frac{\pi}{4}$ on vertex v_3, a $\frac{-\pi}{4}$ binary phase gadget on vertices v_1 and v_2, a $\frac{-\pi}{4}$ binary phase gadget on vertices v_1 and v_3, a $\frac{-\pi}{4}$ binary phase gadget on vertices v_2 and v_3, a $\frac{\pi}{4}$ trinary phase gadget on vertices v_1, v_2, and v_3:



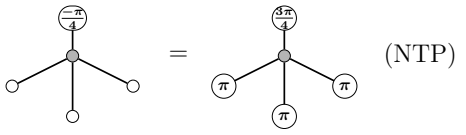
4.2 Equivalence to phase gadget states

Now, we apply the sign-related decomposition to each triangle or CCZ gate in the Union Jack state. By above facts 4.1.1 and 4.1.2, after sign-related decompositions, on each type I vertex we will have two phases of $\frac{-\pi}{4}$ and two phases of $\frac{\pi}{4}$, and on each type II vertex we will have four phases of $\frac{-\pi}{4}$ and four phases of $\frac{\pi}{4}$. Therefore, after sign-related decompositions, the phase on each vertex is zero, since opposite phases cancel out with each other by spider fusion (f).

By the fact 4.1.3, we see that for any two different vertices v_1 and v_2 that are shared by two triangles, after decomposition, there will be exactly a binary phase gadget $P^2_{\frac{-\pi}{4}}$ and another binary phase gadget $P^2_{\frac{\pi}{4}}$ on v_1 and v_2. These two opposite binary phase gadgets will cancel out with each other by the phase gadget fusion rule (PG1). Then, by fact 4.1.4, all binary phase gadgets are cancelled out.

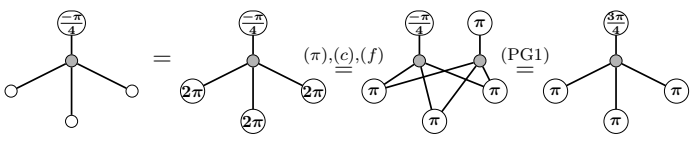
Therefore, only trinary phase gadgets are left in the result of sign-related decompositions. The resulting state after performing the above sign-related Fourier decompositions is shown in figure 4.2. This resulting state is a phase gadget state. Although it seems that two kinds of phase gadgets are contained in this phase gadget state, we can transform this state to an equivalent state constructed by only $\frac{\pi}{4}$ phase gadgets. Next, we will show how to do that.

The following equation (NTP) holds true.



(NTP)

This is because we have:



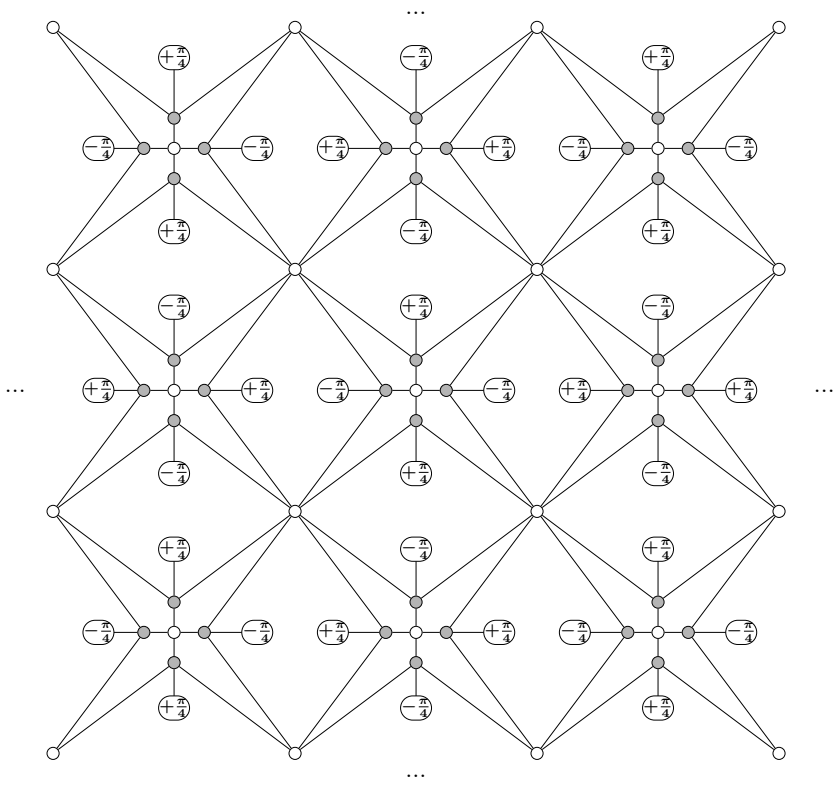
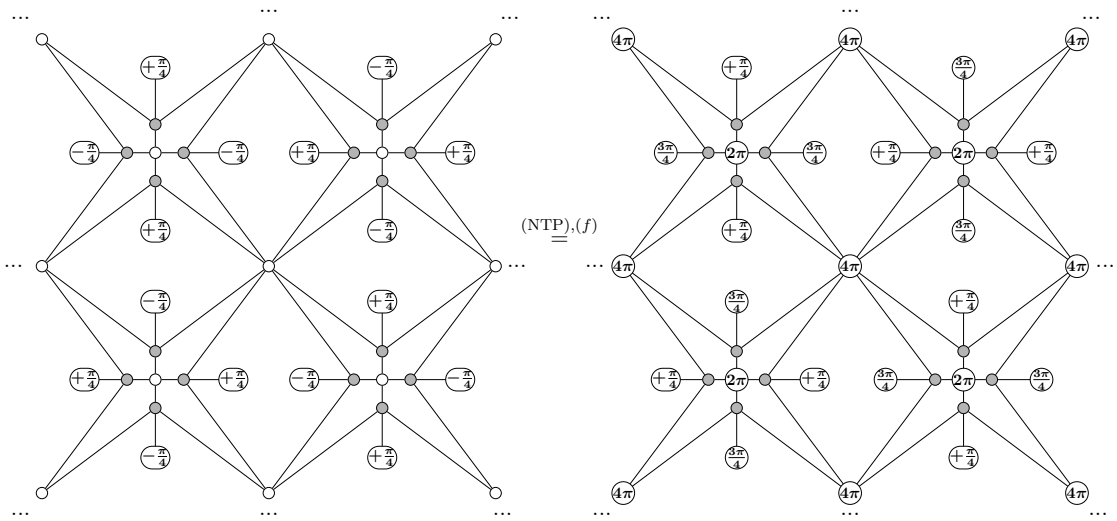
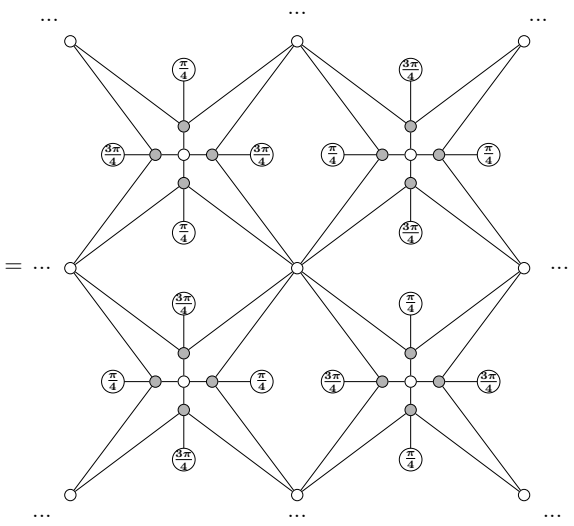


Figure 4.2: Resulting state after applying sign-related Fourier decomposition to each CCZ gate in the Union Jack state

We apply the equation (NTP) to each $\frac{-\pi}{4}$ phase gadget in the state in figure 4.2, and we have:





here we get a state composed by $\frac{3\pi}{4}$ and $\frac{\pi}{4}$ trinary phase gadgets. Recall that a $\frac{3\pi}{4}$ phase gadget can be composed by three $\frac{\pi}{4}$ phase gadgets using rule of (PG1). So, the state in figure 4.2 is equivalent to a phase gadget state composed purely by $\frac{\pi}{4}$ phase gadgets. Therefore, the Union Jack state is equivalent to a phase gadget state which is composed purely by trinary $\frac{\pi}{4}$ phase gadgets.

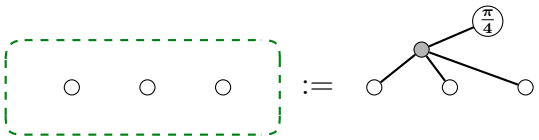
Discussions This equivalence indicates a link between the two families of MBQC models, namely hypergraph state MBQC and phase gadget MBQC. The Union Jack state based MBQC model is universal, and the Union Jack state is originally constructed by CCZ gates. Amazingly, by applying sign-related Fourier decompositions to decompose each CCZ gate in the Union Jack state, we get an equivalent phase gadget state composed by 3-ary $\frac{\pi}{4}$ phase gadgets. Since the Union Jack state can be used as a resource state to achieve universal computations, using this equivalent phase gadget state as a resource state can also achieve universal computations, which indicates that $\frac{\pi}{4}$ trinary phase gadgets are promising building blocks for a universal MBQC resource state. Following this idea, in chapter 5, we will use 3-ary $\frac{\pi}{4}$ phase gadgets as basic elements to build a new universal MBQC model.

5
A New MBQC Model

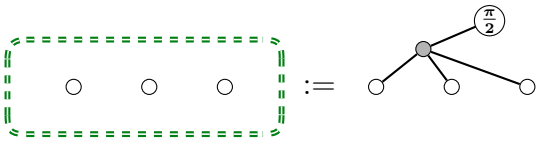
In this chapter, we present a new MBQC model that achieves universal computations with a deterministic protocol using only Pauli X and Z measurements, and we use ZH-calculus to prove correctness of our model. Motivated by the idea discussed in chapter 4, we use trinary $\frac{\pi}{4}$ phase gadgets to construct the resource state. In section 5.1, we introduce new notations that help us better present the new resource state. In section 5.2, we present our construction of the new resource state. In section 5.3, we define important measurement patterns on fragments of our resource state. In section 5.4, we use ZH-calculus to justify all the measurement patterns defined in section 5.3. For conciseness of proofs, we assume in sections 5.3 and 5.4 that there are no incoming Pauli Z and X errors on input wires of fragments. But we take these incoming errors into account in the following sections to fully justify that our MBQC model can correct measurement errors. In section 5.5, we discuss how we adapt our measurement patterns so as to correct measurement errors. In section 5.6, we put everything together and show that we can deterministically achieve universal computations with our MBQC model.

5.1 Notations

For simplicity, we use the following green dashed rounded rectangle to represent a trinary $\frac{\pi}{4}$ phase gadget:



When we fuse two $\frac{\pi}{4}$ trinary phase gadgets together by (PG1), we get a $\frac{\pi}{2}$ trinary gadget, which we use the following double green dashed rounded rectangle to denote:



Orange, red, and grey boxes To avoid drawing a lot of green dashed lines, we define the notations of *orange boxes*, *red boxes* and *grey boxes* in figure 5.1, figure 5.2, and figure 5.3 respectively.

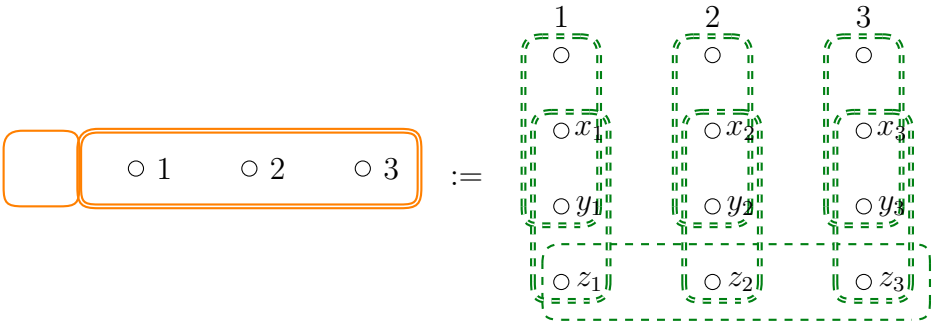


Figure 5.1: The *orange box* notation. The notation on the left-hand side defines a twelve-qubit system with six $\frac{\pi}{2}$ trinary phase gadgets, and one $\frac{\pi}{4}$ trinary phase gadgets shown on hte right-hand side. 9 qubits and phase gadgets are encapsulated or hidden in the box.

Remark 5.1.1. When we introduce notations, we usually number all qubits using characters, which serves as making the relative positions of qubits clearer. Numberings have no syntactic meaning.

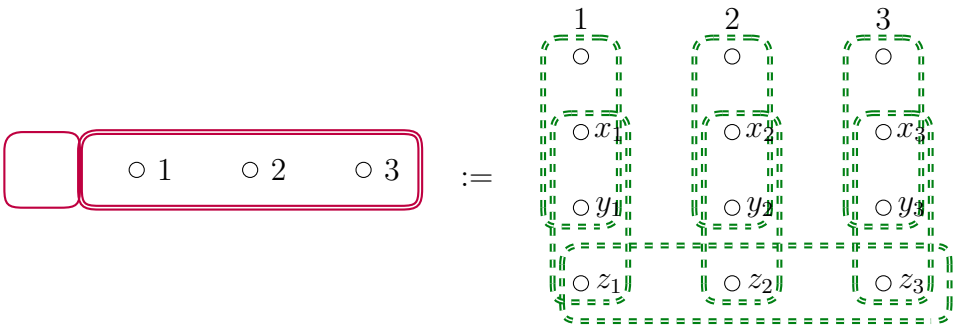


Figure 5.2: The *red box* notation. The notation on the left-hand side defines a twelve-qubit system with seven $\frac{\pi}{2}$ trinary phase gadgets. 9 qubits and phase gadgets are encapsulated or hidden in the box. The only difference between this notation and the orange box notation in figure 5.1 is that here in the red box notation, the trinary phase gadget on the bottom is a $\frac{\pi}{2}$ phase gadget rather than a $\frac{\pi}{4}$ phase gadget.

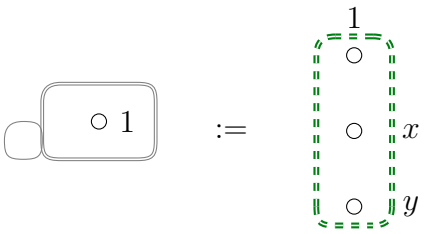


Figure 5.3: The *grey box* notation. The notation on the left-hand side defines a three-qubit system with one $\frac{\pi}{2}$ trinary phase gadget. 2 qubits and the phase gadget are encapsulated or hidden in the box.

Remark 5.1.2. In the orange box, red box, and grey box notation, there is a small box attached to the larger box. The reason why we create this small box in these notations will become clear in section 5.3. The position of the small box has no syntactic meaning as illustrated in figure 5.4. That is, we can attach the small box to the left, to the right, to the bottom, or on the top, of the larger box.

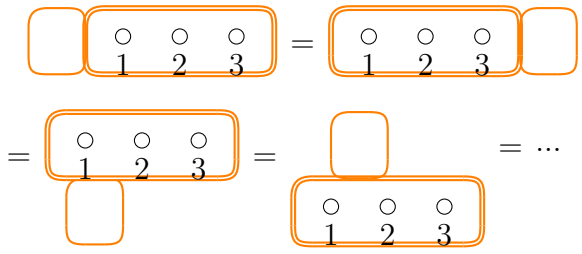
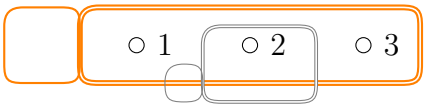


Figure 5.4: In the orange, the red, and the grey box notation, the position where the little box is attached to the larger one has no syntactic meaning.

Remark 5.1.3. Two different box (orange, red, or grey) would never share any hidden qubits, even if they overlap each other. In other words, if a qubit is hidden by a box notation, then it is hidden by this box only. For example, the following diagram defines a 14-qubit system:



There are three visible qubits 1, 2, and 3, nine qubits hidden by the orange box, and two qubits hidden by the grey box.

Filled orange boxes and ropes Now we are ready to define a *filled orange box* notation and a *rope* notation, which will be used to present our new resource state. The filled orange box notation is defined in figure 5.5. This filled orange box system is composed by one orange box defined in figure 5.1, one red box defined in figure 5.2, and three grey boxes defined in figure 5.3. The rope notation is defined in figure 5.6.

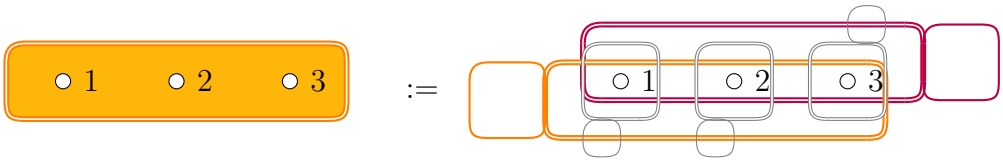


Figure 5.5: The *filled orange box* notation. This notation defines a 27-qubit system with 17 phase gadgets. Note that as mentioned in remark 5.1.3, no hidden qubits are shared by different boxes.

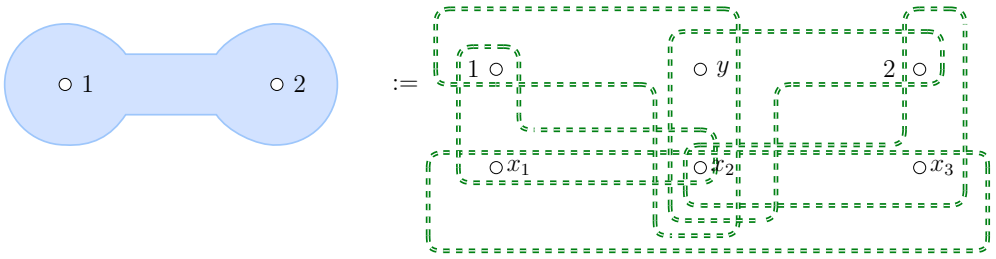
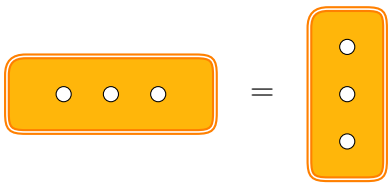


Figure 5.6: The *rope notation*. This notation defines a six-qubit system with five phase gadgets.

Remark 5.1.4. There are also hidden qubits in the rope notation. Similar to the case of orange/red/grey box notations, qubits hidden by a rope notation are "private" to this rope, and will never be shared with other ropes or orange/red/grey boxes.

Remark 5.1.5. Whether we draw our notations horizontally or vertically has no syntactic meaning. For example, we have:



5.2 The new resource state

The resource state is presented in figure 5.7. In our resource state, the boxes are filled orange systems defined in figure 5.5. We can think of these boxes as bricks. We connect these bricks with rope systems defined in figure 5.6. It is clear that our state has a vertical layered structure. In each layer, bricks are parallel to each other. Between two adjacent layers, there is always an offset of one qubit in the arrangement of bricks. The reason why we arrange boxes and ropes in the way shown in figure 5.7 is to allow gates implemented by measurement patterns on these boxes and ropes to compose together in any manner.

5.3 Measurement patterns

In this section, we define measurement patterns on the box and rope systems introduced in section 5.1. Since a system can have multiple measurement patterns, to differentiate them, we write different symbols in the little box of the box notations or in the centre of the rope notation to represent different measurement patterns. Notations for measurement patterns introduced in section 2.9.3 is used in this section.

Let us recall a point that is introduced in section 2.9.3. When we define a measurement pattern on a fragment of a resource state, we set inputs and outputs. In a measurement pattern, if a qubit is set as an input or an output, then this

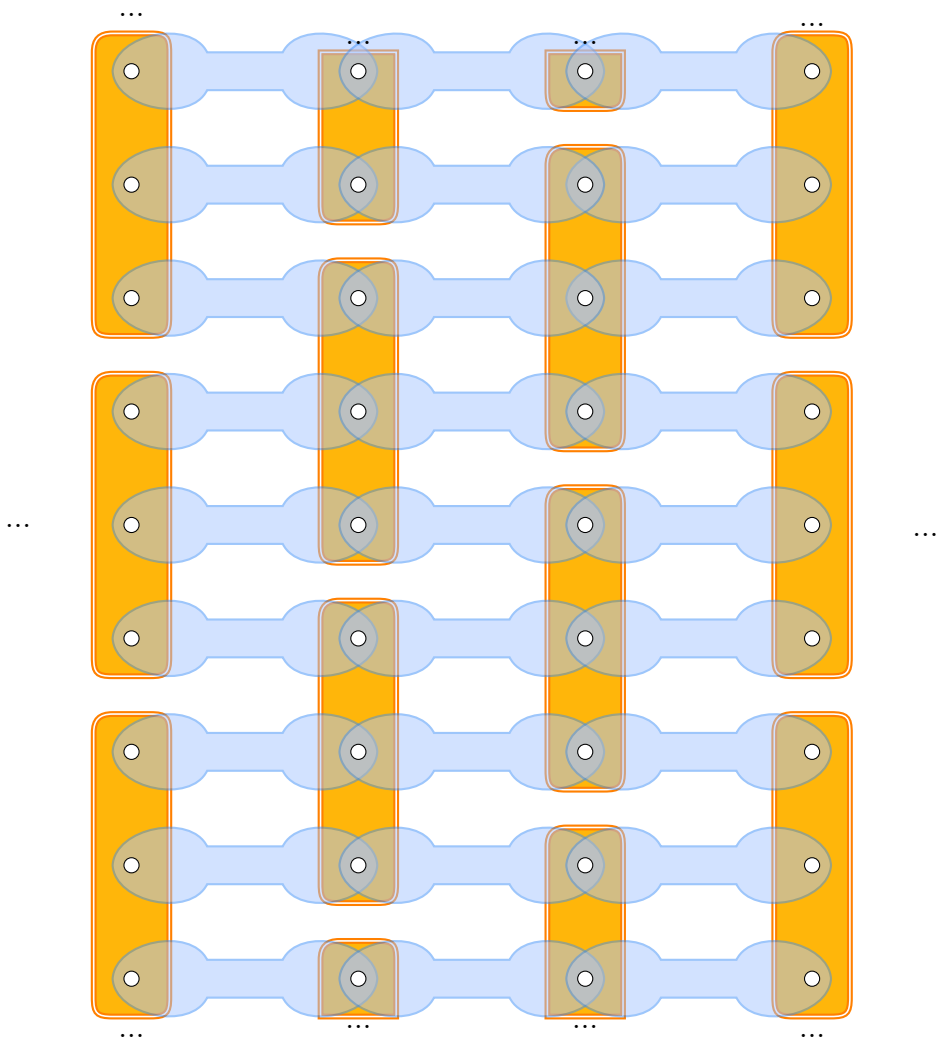


Figure 5.7: The resource state. This resource state is composed purely by $\frac{\pi}{4}$ phase gadgets. The filled orange box and rope notations defined in section 5.1 are used.

qubit will not be measured. There is an input wire for each input qubit and an output wire for each output qubit. These input and output wires are free or dangling edges, but we often omit them in diagrams.

Measurement patterns on the orange and red box system First, we define several measurement patterns on the orange box and the red box system. In figure 5.8, we set qubits 1, 2, and 3 as both inputs and outputs and define a T_1 measurement pattern on the orange box system. This measurement pattern implements a T gate

on qubit 1. Symmetrically, in figures A.2 and A.3, we define a T_2 measurement

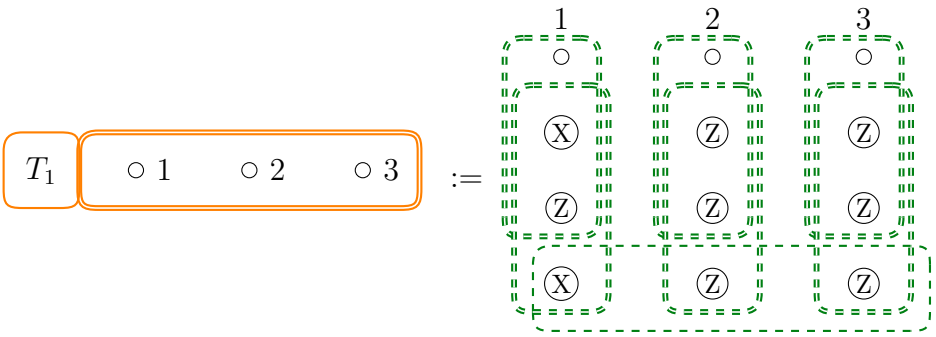


Figure 5.8: A measurement pattern on the orange box notation. This measurement pattern implements a T gate on qubit 1. Qubits 1, 2, and 3 are both inputs and outputs.

pattern and a T_3 measurement pattern on the orange box system.

In figure 5.9, we set qubits 1, 2, and 3 as both inputs and outputs and define a S_* measurement pattern on the orange box system. This measurement pattern implements S gates on qubits 1, 2, and 3. Similarly, we define in figure A.4 a S_*

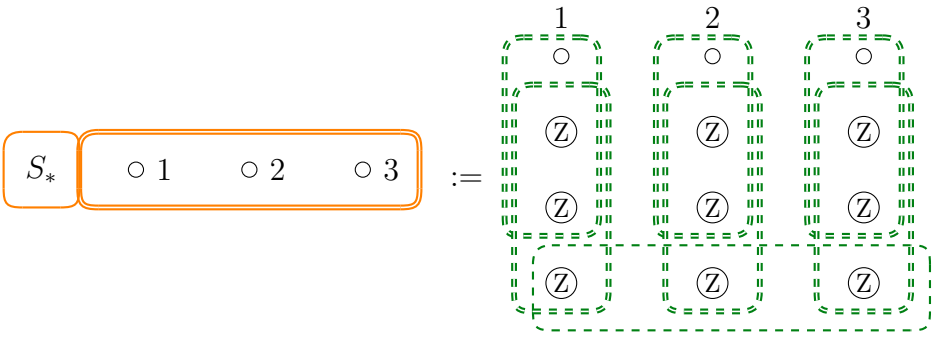


Figure 5.9: A measurement pattern on the orange box notation. This measurement pattern implements S gates on qubits 1, 2, and 3. Qubits 1, 2, and 3 are both inputs and outputs.

measurement pattern on the red box system.

In figure 5.10, we set qubits 1, 2, and 3 as both inputs and outputs and define a $P_{1,2}$ measurement pattern on the orange box system. This measurement pattern implements a binary $\frac{\pi}{4}$ phase gadget on qubits 1 and 2. Symmetrically, we define in figures A.5 and A.6 a $P_{1,3}$ measurement pattern and a $P_{2,3}$ measurement pattern on the orange box system. Similarly, we define in figures A.7, A.8, and A.9 a

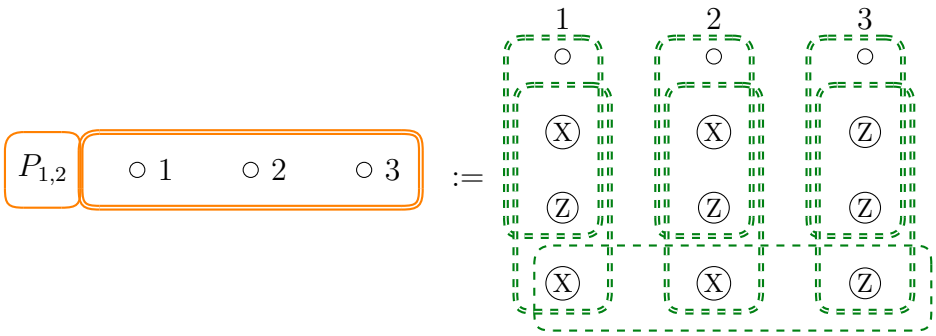


Figure 5.10: A measurement pattern on the orange box notation. This measurement pattern implements a binary $\frac{\pi}{4}$ phase gadget on qubits 1 and 2. Qubits 1, 2, and 3 are both inputs and outputs.

$P_{1,2}$ measurement pattern, a $P_{1,3}$ measurement pattern, and a $P_{2,3}$ measurement pattern on the red box system.

In figure 5.11, we set qubits 1, 2, and 3 as both inputs and outputs and define a P_* measurement pattern on the orange box system. This measurement pattern implements a trinary $\frac{\pi}{4}$ phase gadget on qubits 1, 2, and 3. Similarly, we define in

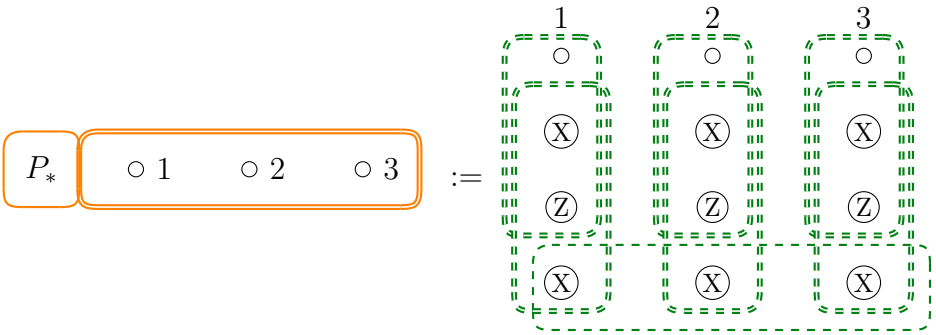


Figure 5.11: A measurement pattern on the orange box notation. This measurement pattern implements a trinary $\frac{\pi}{4}$ phase gadget on qubits 1, 2, and 3. Qubits 1, 2, and 3 are both inputs and outputs.

figure A.10 a P_* measurement pattern on the red box system.

Measurement patterns on the grey box system Next, we define two measurement patterns on the grey box system. In figure 5.12, we set qubit 1 as both the input and the output and define a S measurement pattern on the grey box system. This measurement pattern implements a S gate on qubit 1. In figure 5.13, we set qubit 1

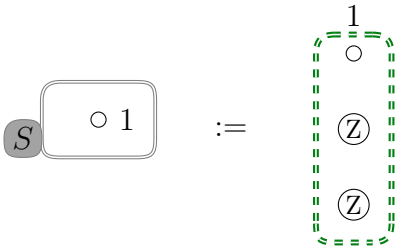


Figure 5.12: A measurement pattern on the grey box notation. This measurement pattern implements a S gate on qubit 1. Qubit 1 is both the input and the output.

as both the input and the output and define an I measurement pattern on the grey box system. This measurement pattern implements an identity gate on qubit 1.

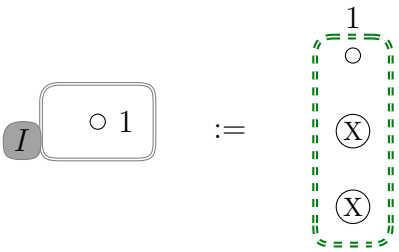


Figure 5.13: A measurement pattern on the grey box notation. This measurement pattern implements an identity operation on qubit 1. Qubit 1 is both the input and the output.

Measurement patterns on the rope system Next, we define two measurement patterns on the rope system. In figure 5.14, we set qubit 1 as the input and qubit 2 as the output and define an I measurement pattern on the rope system. This measurement pattern implements an identity wire between qubits 1 and 2. In

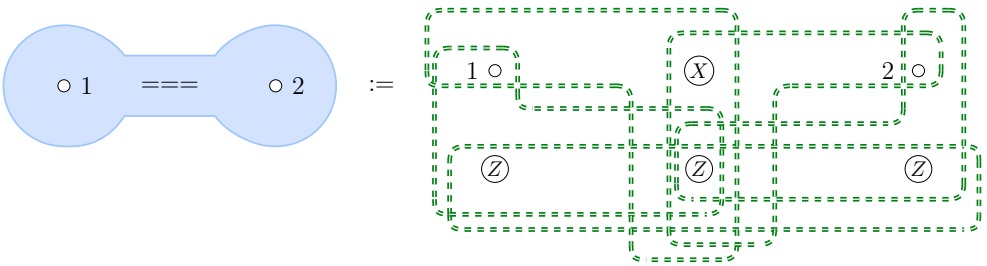


Figure 5.14: A measurement pattern on the rope notation. This measurement pattern implements an identity wire between qubits 1 and 2. Qubit 1 is the input and qubit 2 is the output.

figure 5.15, we set qubit 1 as the input and qubit 2 as the output and define a H measurement pattern on the rope system. This measurement pattern implements a CZ wire between qubits 1 and 2.

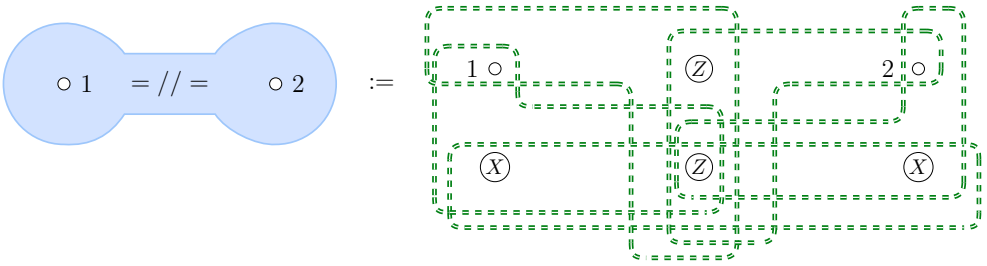


Figure 5.15: A measurement pattern on the rope notation. This measurement pattern implements a CZ wire between qubits 1 and 2. Qubit 1 is the input and qubit 2 is the output.

Measurement patterns on the filled orange box system Next, we define measurement patterns on the filled orange box system. In figure 5.16, we set qubits 1, 2, and 3 as both inputs and outputs, and define a T_1 measurement pattern on the filled orange box system. This measurement pattern implements a T gate on qubit 1.

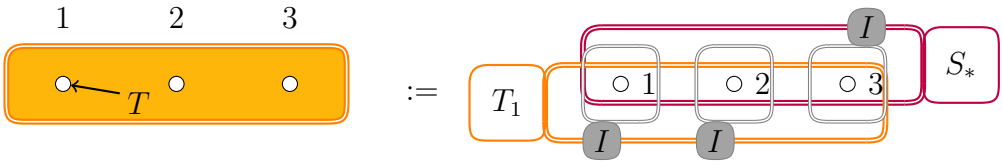


Figure 5.16: A measurement pattern on the filled orange box notation. This measurement pattern implements a T gate on qubit 1. Qubits 1, 2, and 3 are both inputs and outputs.

Remark 5.3.1. One important reason why we compose orange, red, and grey boxes together to make the filled orange box system is to form a scheme to correct measurement errors. This T_1 measurement pattern in figure 5.16 does additional work to correct errors, while the other T_1 measurement pattern defined in figure 5.8 does not. Similar arguments apply to other measurement patterns defined on the filled orange box system.

Symmetrically, we define in figures A.11 and A.12 a T_2 measurement pattern and a a T_3 measurement pattern on the filled orange box system.

In figure 5.17, we set qubits 1, 2, and 3 as both inputs and outputs, and define a S_* measurement pattern on the filled orange box system. This measurement pattern implements S gates on qubits 1, 2, and 3. For convenience, next we define a

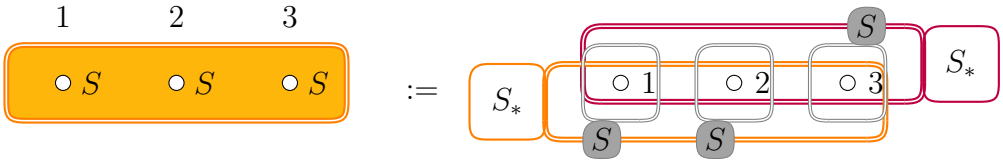


Figure 5.17: A measurement pattern on the filled orange box notation. This measurement pattern implements S gates on qubits 1, 2, and 3. Qubits 1, 2, and 3 are both inputs and outputs.

generalized form for measurement patterns like S_*. In figure 5.18, we set qubits 1, 2, and 3 as both inputs and outputs, and define a series of measurement patterns on the filled orange box system. One special case of the measurement patterns defined

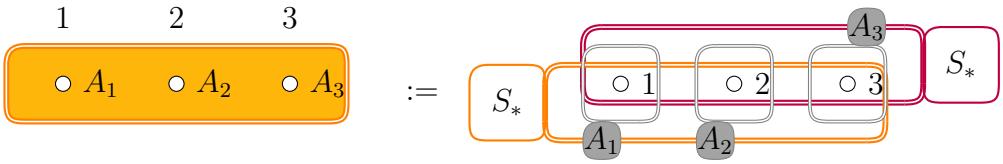


Figure 5.18: A measurement pattern defined on the filled orange box system. For $i \in \{1,2,3\}$, $A_i \in \{S, I\}$. For $i \in \{1,2,3\}$, this measurement pattern implements a S gate on qubit i if $A_i = S$; otherwise it implements an identity gate on qubit i. Qubits 1, 2, and 3 are both inputs and outputs.

in figure 5.18 is an I measurement pattern shown in figure 5.19. This I measurement pattern implements identity gates on qubits 1, 2, and 3.

In figure 5.20, we set qubits 1, 2, and 3 as both inputs and outputs, and define a $P_{\frac{\pi}{4}}^{(1,2)}$ measurement pattern on the filled orange box system. Note that this $P_{\frac{\pi}{4}}^{(1,2)}$ measurement pattern is performed in two steps, where the measurements in the second step depend on the measurement outcomes of the first step. In the first step, we perform on the orange box the $P_{1,2}$ measurement pattern defined in figure

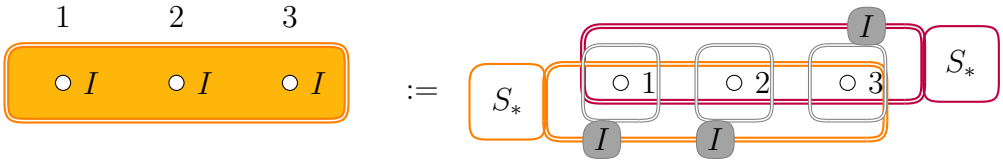


Figure 5.19: A measurement pattern on the filled orange box notation. This measurement pattern implements identity gates on qubits 1, 2, and 3. Qubits 1, 2, and 3 are both inputs and outputs.

5.10. The values of $\Phi_{1,2}$ and Θ are then determined by whether a phase gadget error is produced in this measurement. If a phase gadget error is produced, then $\Phi_{1,2} = P_{1,2}$ and $\Theta = I$; otherwise, $\Phi_{1,2} = S_*$ and $\Theta = S$. In the second step, we perform measurement patterns defined on the red box and grey box system. This measurement pattern implements a binary $\frac{\pi}{4}$ phase gadget on qubits 1 and 2. Symmetrically, we define in figures A.13 and A.14 a $P^{(1,3)}_{\frac{\pi}{4}}$ measurement pattern and

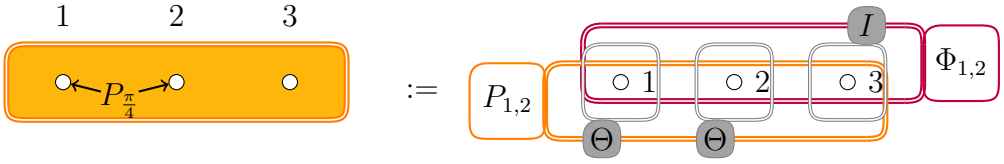


Figure 5.20: A measurement pattern on the filled orange box system. First, we perform on the orange box $P_{1,2}$ measurement pattern. If a phase gadget error is produced then $\Phi_{1,2} = P_{1,2}$ and $\Theta = I$; otherwise $\Phi_{1,2} = S_*$ and $\Theta = S$. Then we perform all the rest of measurements defined in this figure. This measurement pattern implements a binary $\frac{\pi}{4}$ phase gadget on qubits 1 and 2. Qubits 1, 2, and 3 are both inputs and outputs.

a $P^{(2,3)}_{\frac{\pi}{4}}$ measurement pattern on the filled orange box system.

In figure 5.21, we set qubits 1, 2, and 3 as both inputs and outputs, and define a $P_{\frac{\pi}{4}}$ measurement pattern on the filled orange box system. This measurement pattern is also performed in two steps, where the values of Φ_* and Θ are determined by whether a phase gadget error is produced in the first step. This measurement pattern implements a trinary $\frac{\pi}{4}$ phase gadget on qubits 1, 2, and 3.

Similarly, in figure 5.22, we set qubits 1, 2, and 3 as both inputs and outputs, and define a $P_{\frac{\pi}{2}}$ measurement pattern on the filled orange box system. This measurement pattern implements a trinary $\frac{\pi}{2}$ phase gadget on qubits 1, 2, and 3.

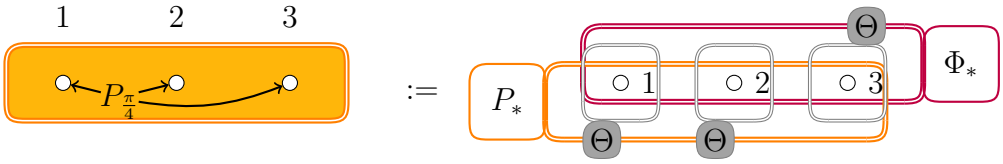


Figure 5.21: A measurement pattern on the filled orange box system. First, we perform on the orange box P_* measurement pattern defined in figure 5.11. If a phase gadget error is produced, then $\Phi_* = P_*$ and $\Theta = I$; otherwise $\Phi_* = S_*$ and $\Theta = S$. Then we perform all the rest of measurements defined in this figure. This measurement pattern implements a trinary $\frac{\pi}{4}$ phase gadget on qubits 1, 2, and 3. Qubits 1, 2, and 3 are both inputs and outputs.

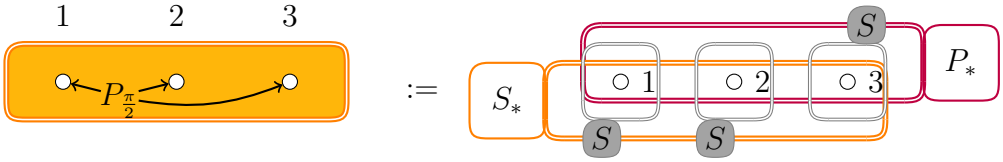


Figure 5.22: A measurement pattern on the filled orange box notation. This measurement pattern implements a trinary $\frac{\pi}{2}$ phase gadget on qubits 1, 2, and 3. Qubits 1, 2, and 3 are both inputs and outputs.

In figure A.17, we set qubits 1, 2, and 3 as both inputs and outputs, and define a $P^{(1,2)}_{\frac{\pi}{2}}$ measurement pattern on the filled orange box system. Symmetrically, we define in figures A.15 and A.16 a $P^{(1,3)}_{\frac{\pi}{2}}$ measurement pattern and a $P^{(2,3)}_{\frac{\pi}{2}}$ measurement pattern on the filled orange box system.

To provide more flexibility, we also allow measurement patterns like those defined in figures A.18, A.19, A.20, A.21. It should also be clear how to define measurement patterns that are symmetric to those in figures A.18, A.19, A.20, A.21.

5.4 Proofs for measurement patterns

In this section, we prove that the measurement patterns defined in section 5.3 implement their corresponding target gates or operations.

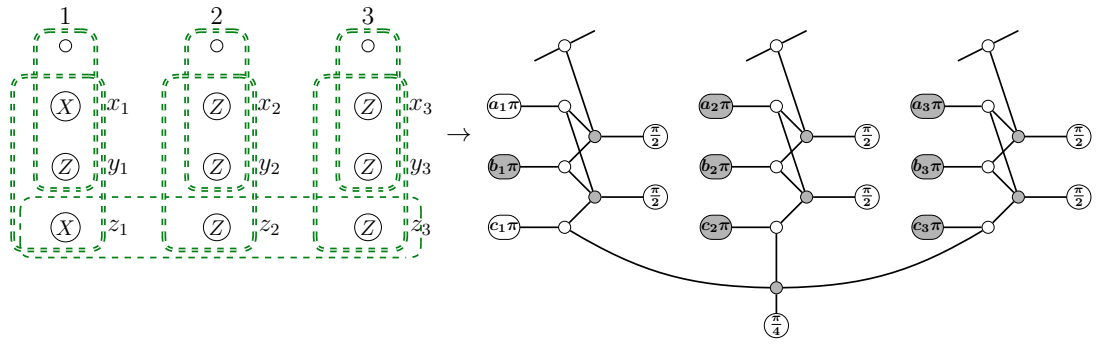
Proofs for measurement patterns on orange, red, and grey boxes The following lemma 5.4.1 justifies measurement patterns implementing T gates on

the orange box system.

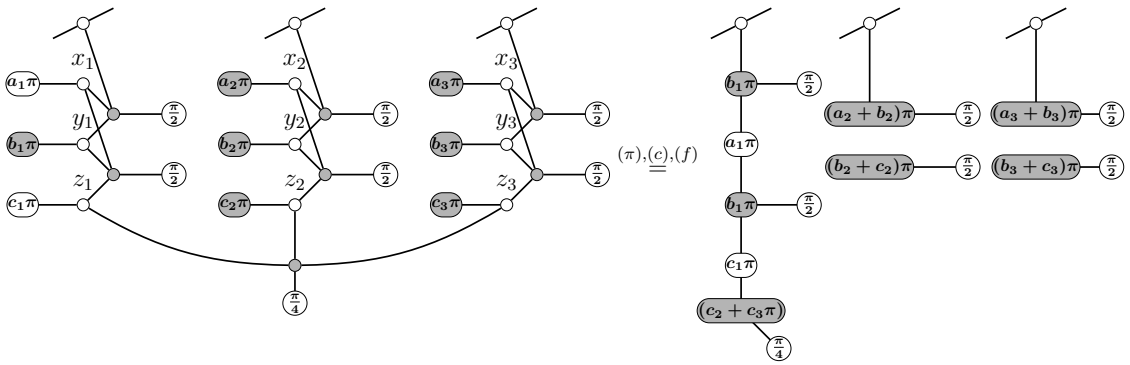
Lemma 5.4.1. The following three statements hold true:

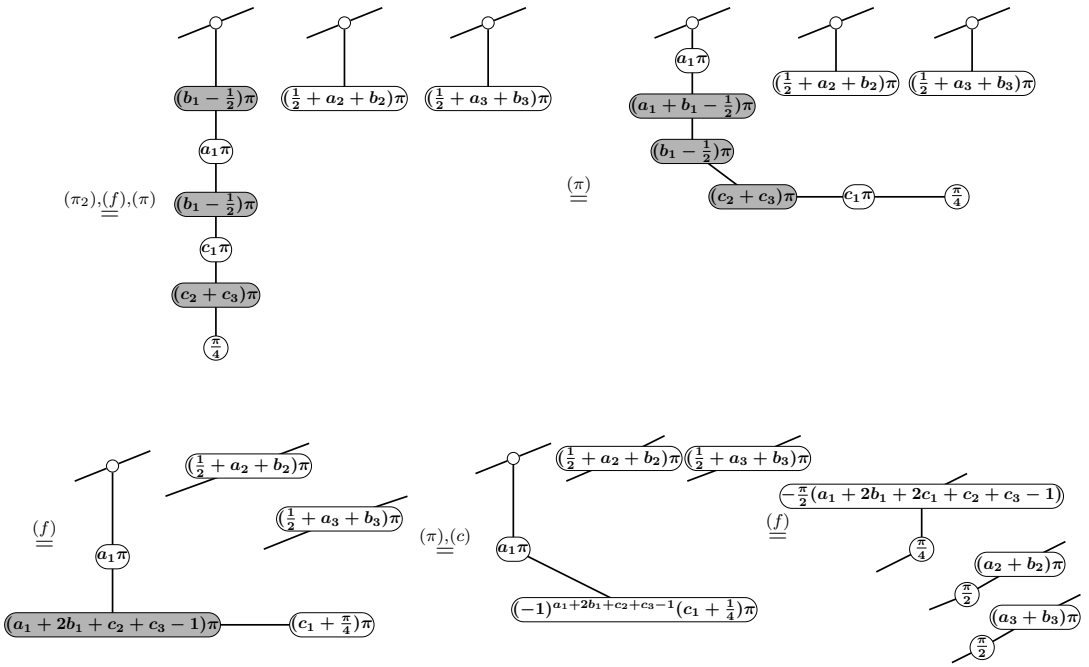
- The measurement pattern defined in figure 5.8 implements a T gate on qubit 1, up to Pauli Z errors and S errors.
- The measurement pattern defined in figure A.2 implements a T gate on qubit 2, up to Pauli Z errors and S errors.
- The measurement pattern defined in figure A.3 implements a T gate on qubit 3, up to Pauli Z errors and S errors.

Proof. We only need to prove the first part of the lemma, since the three parts are symmetric. We first use rules of measurement patterns introduced in section 2.9.3 to translate the measurement pattern into a ZH-diagram as follows:



Then, apply rules of ZH-calculus and we have:

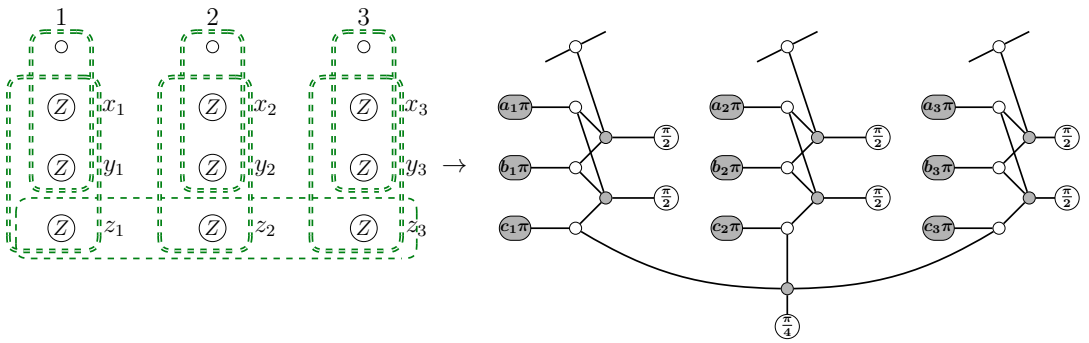




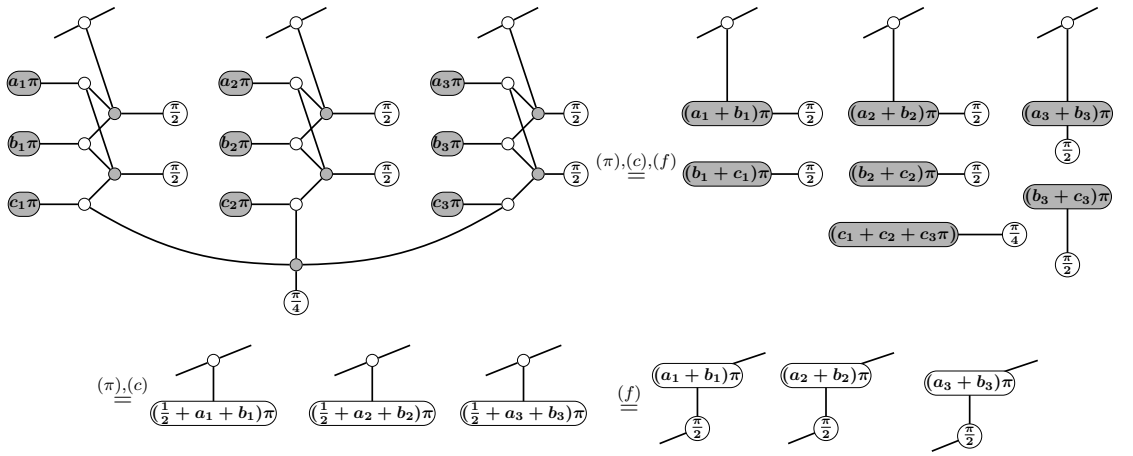
The following lemma 5.4.2 justifies measurement patterns implementing S gates on the orange box system.

Lemma 5.4.2. The measurement pattern defined in figure 5.9 implements S gates on qubits 1, 2 and 3, up to Pauli Z errors.

Proof. Translate the measurement pattern into a ZH-diagram as follows:



Then, apply rules of ZH-calculus and we have:



■

Remark 5.4.3. In this case, the S gate is the computation we intend to implement or our target gate, so we do not treat it as an error.

The following lemma 5.4.4 is similarly to lemma 5.4.2, and so is its proof. It justifies measurement patterns implementing S gates on the red box system.

Lemma 5.4.4. The measurement pattern defined in figure A.4 implements S gates on qubits 1, 2 and 3, up to Pauli errors.

Proof. Similar to the proof for lemma 5.4.2. ■

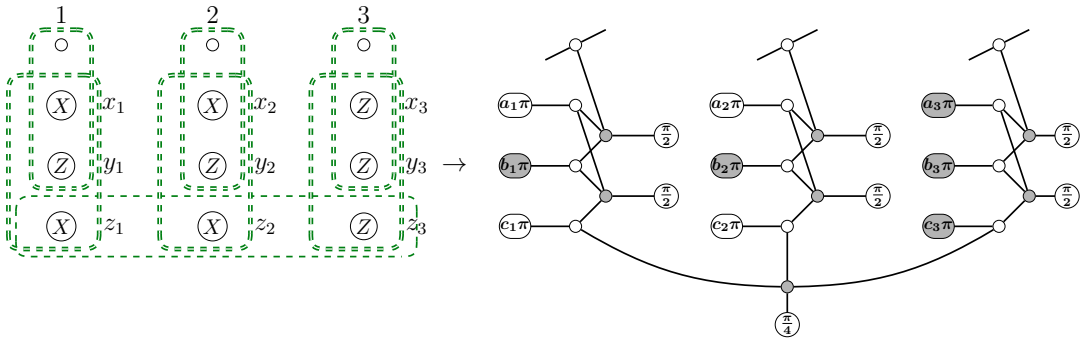
The following lemma 5.4.5 justifies measurement patterns implementing binary $\frac{\pi}{4}$ phase gadgets on the orange box system.

Lemma 5.4.5. The following three statements hold true:

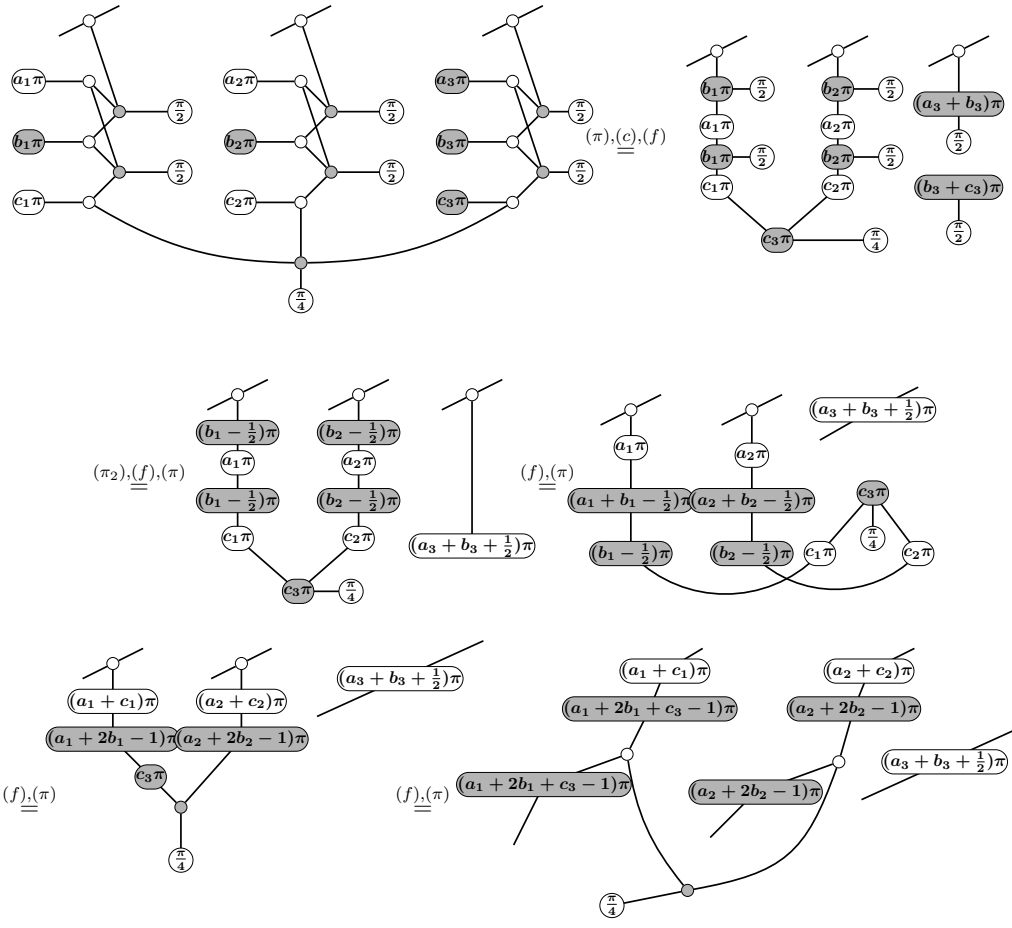
- The measurement pattern defined in figure 5.10 implements a binary $\frac{\pi}{4}$ phase gadget on qubits 1 and 2, up to Pauli Z errors, binary $\frac{\pi}{2}$ phase gadget errors, and S errors.
- The measurement pattern defined in figure A.5 implements a binary $\frac{\pi}{4}$ phase gadget on qubits 1 and 3, up to Pauli Z errors, binary $\frac{\pi}{2}$ phase gadget errors, and S errors.

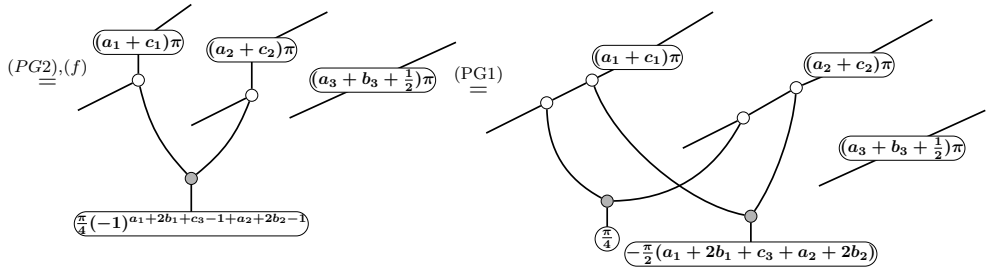
- The measurement pattern defined in figure A.6 implements a binary $\frac{\pi}{4}$ phase gadget on qubits 2 and 3, up to Pauli Z errors, binary $\frac{\pi}{2}$ phase gadget errors, and S errors.

Proof. We only need to prove the first part of the lemma, since the three parts are symmetric. Translate the measurement pattern into a ZH-diagram as follows:



Then, apply rules of ZH-calculus and we have:





According to the above derivations, we can see that a $\frac{\pi}{2}$ phase gadget error will appear if $a_1+2b_1+c_3+a_2+2b_2 = (2k+1)$, and it will not appear if $a_1+2b_1+c_3+a_2+2b_2 = 2k$, where $k \in \mathbb{Z}$. ■

The following lemma 5.4.6 is similar to the above lemma 5.4.5. It justifies measurement patterns implementing binary $\frac{\pi}{2}$ phase gadgets on the red box system.

Lemma 5.4.6. The following three statements hold true:

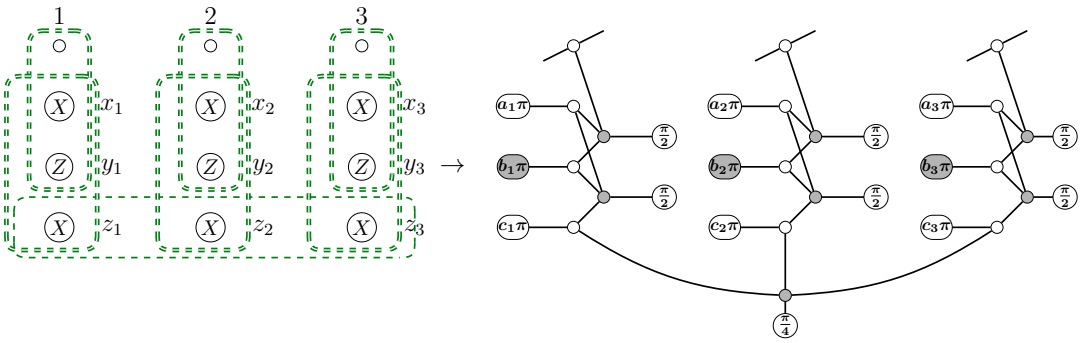
- The measurement pattern defined in figure A.7 implements a binary $\frac{\pi}{2}$ phase gadget on qubits 1 and 2, up to Pauli Z errors and S errors.
- The measurement pattern defined in figure A.8 implements a binary $\frac{\pi}{2}$ phase gadget on qubits 1 and 3, up to Pauli Z errors and S errors.
- The measurement pattern defined in figure A.9 implements a binary $\frac{\pi}{2}$ phase gadget on qubits 2 and 3, up to Pauli Z errors and S errors.

Proof. The proof is similar to the proof for lemma 5.4.5. There is only one thing to note: this measurement pattern in figure A.10 will not produce any phase gadget errors. This is because a π phase gadget error will "break down" into Pauli Z errors by equation (PGπ). ■

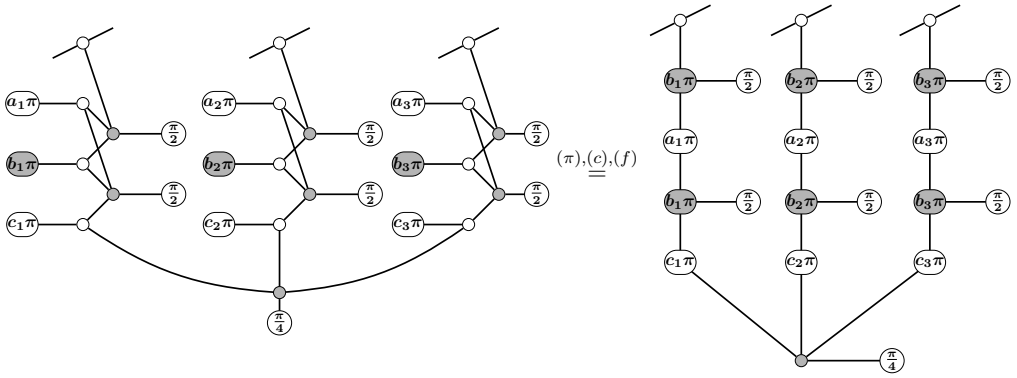
The following lemma 5.4.7 justifies the measurement pattern implementing trinary $\frac{\pi}{4}$ phase gadgets on the orange box system.

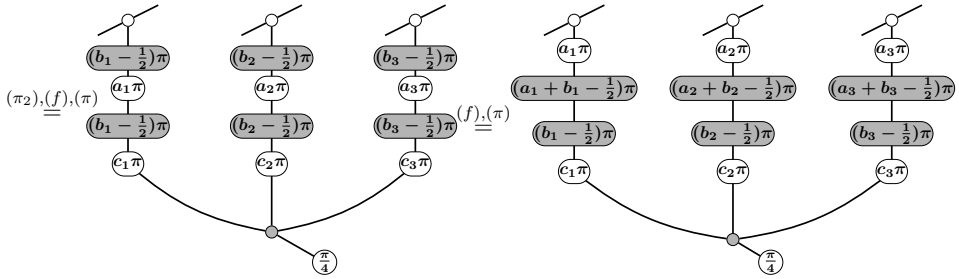
Lemma 5.4.7. The measurement pattern defined in figure 5.11 implements a trinary $\frac{\pi}{4}$ phase gadget on qubits 1, 2, and 3, up to Pauli Z and trinary $\frac{\pi}{2}$ phase gadget errors.

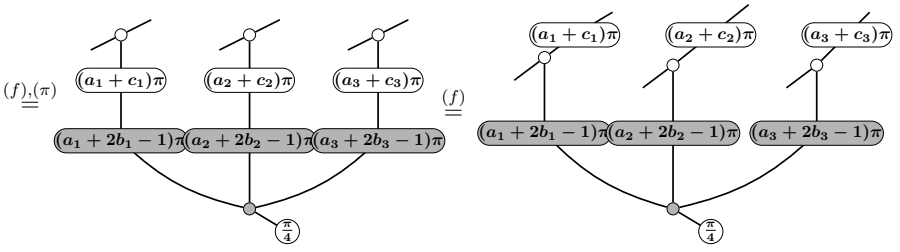
Proof. Translate the measurement pattern into a ZH-diagram as follows:

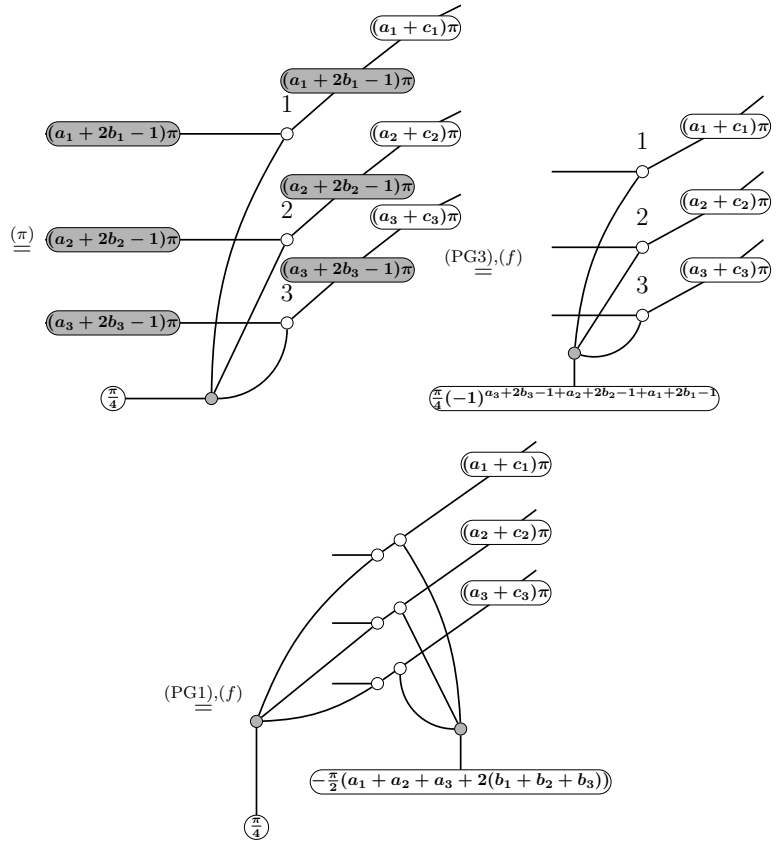


Then, we apply rules of ZH-calculus and we have:









■

The following lemma is similar to lemma 5.4.7 and its proof is also similar. It justifies the measurement pattern implementing trinary $\frac{\pi}{2}$ phase gadgets on the red box system.

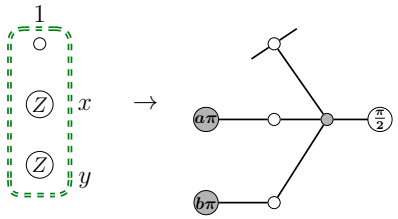
Lemma 5.4.8. The measurement pattern defined in figure A.10 implements a trinary $\frac{\pi}{2}$ phase gadget on qubits 1, 2, and 3, up to Pauli errors.

Proof. The proof is very similar to the proof for lemma 5.4.7. Again note that this measurement pattern in figure A.10 will not produce phase gadget errors. This is because by rules of ZH-calculus, a π phase gadget error will "break down" into Pauli Z errors as shown by equation (PGπ). ■

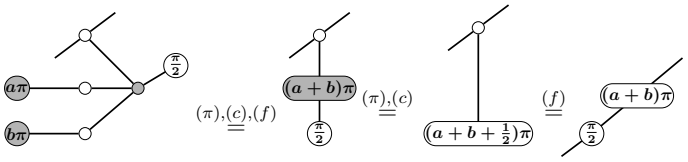
The following lemma 5.4.9 justifies the measurement pattern implementing S gates on the grey box system.

Lemma 5.4.9. The measurement pattern defined in figure 5.12 implements a S gate on qubit 1, up to Pauli Z errors.

Proof. Translate the measurement pattern into a ZH-diagram as follows:



Then, we apply rules of ZH-calculus and we have

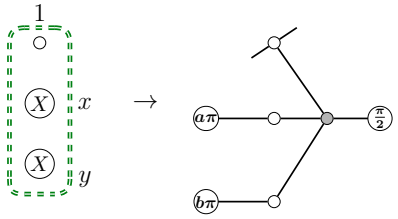


■

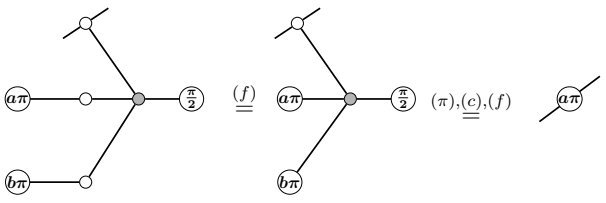
The following lemma 5.4.10 justifies the measurement pattern implementing identity gates on the grey box system.

Lemma 5.4.10. The measurement pattern defined in figure 5.13 implements an identity gate on qubit 1, up to Pauli Z errors.

Proof. Translate the measurement pattern into a ZH-diagram as follows:



Then, we apply rules of ZH-calculus and we have:



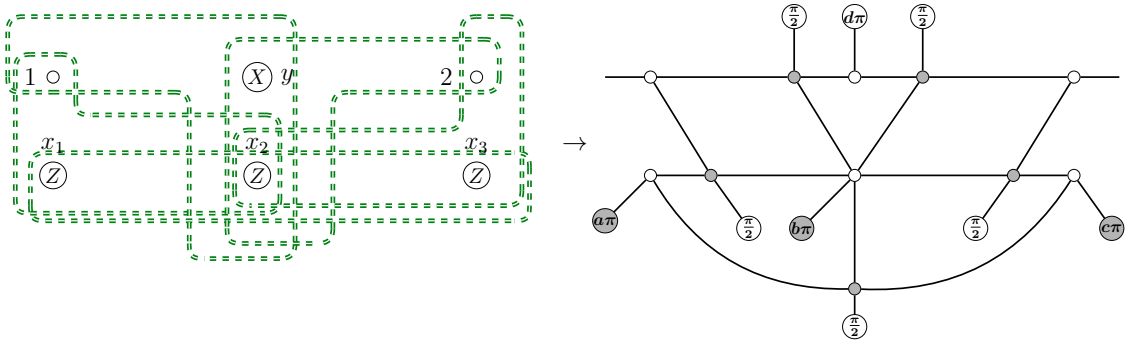
■

Remark 5.4.11. In the above derivations, we see that with some measurement outcomes, the result of calculation is zero. Getting a zero means that the probability of producing the corresponding measurement outcomes is zero, so we simply omit such discussions in our proofs.

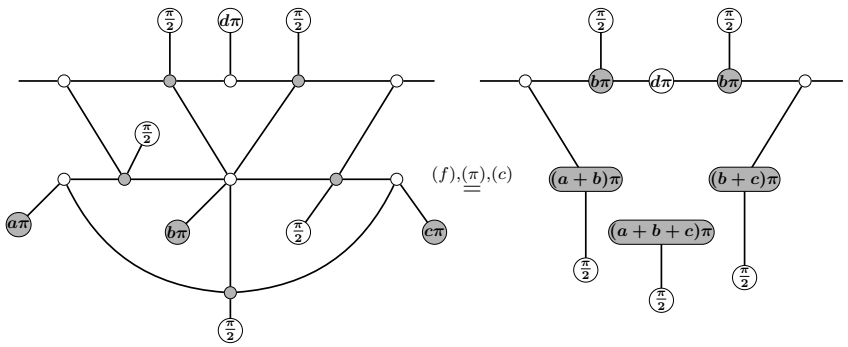
Proofs for measurement patterns on ropes The following lemma 5.4.12 justifies the measurement pattern implementing identity gates on the rope structure.

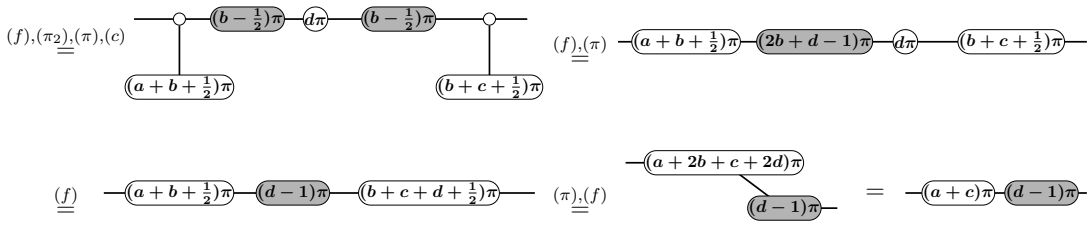
Lemma 5.4.12. The measurement pattern defined in figure 5.14 implements an identity wire between qubits 1 and 2, up to Pauli X and Z errors.

Proof. Translate the measurement pattern into a ZH-diagram as follows:



Then, we apply rules of ZH-calculus and we have:



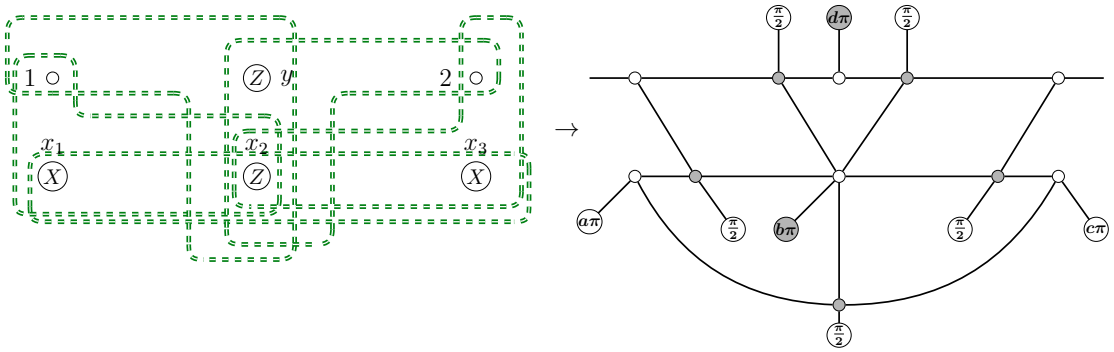


■

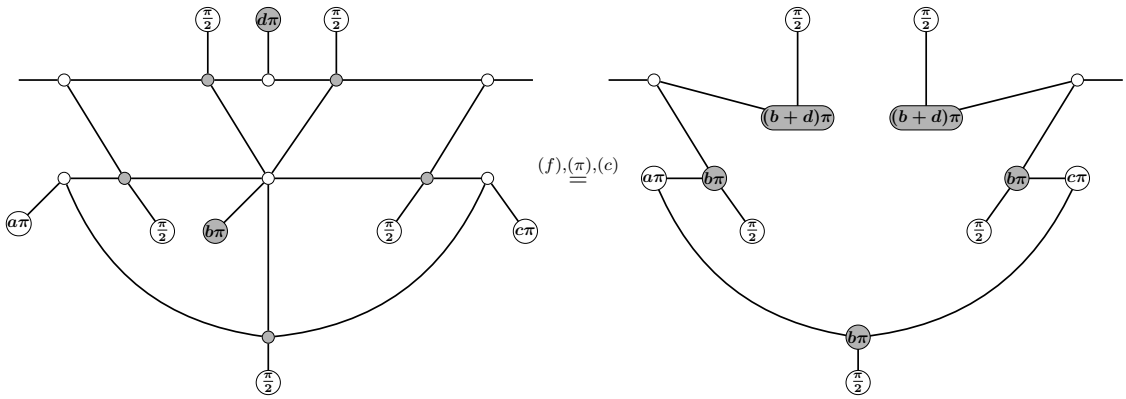
The following lemma 5.4.13 justifies the measurement pattern implementing Hadamard gates on the rope structure.

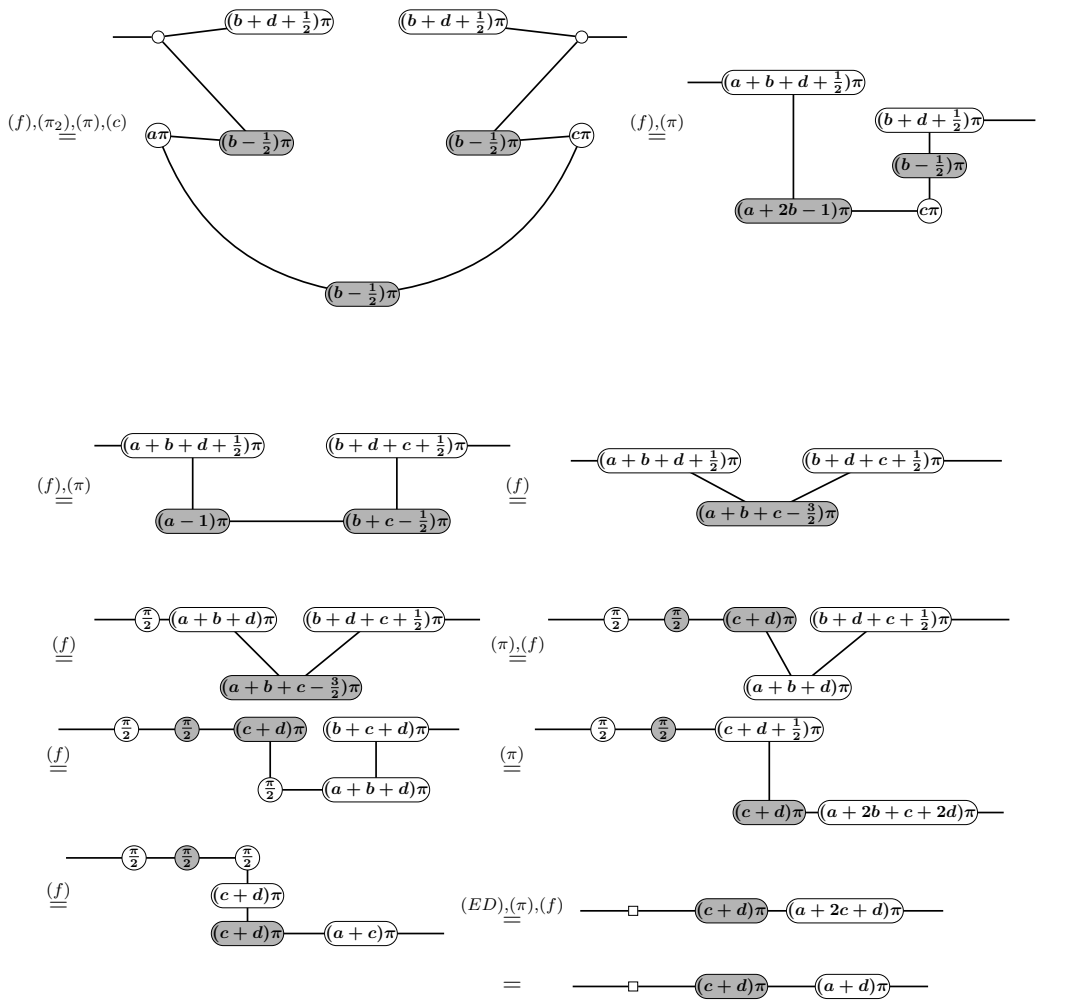
Lemma 5.4.13. The measurement pattern defined in figure 5.15 implements a CZ wire between qubits 1 and 2, up to Pauli X and Z errors.

Proof. Translate the measurement pattern into a ZH-diagram as follows:



Then, we apply rules of ZH-calculus and we have:





■

Proofs for measurement patterns on filled orange boxes The following lemma 5.4.14 justifies measurement patterns implementing T gates on the filled orange box system.

Lemma 5.4.14. The following three statements hold true:

- The measurement pattern defined in figure 5.16 implements a T gate on qubit 1, up to Pauli Z errors.
- The measurement pattern defined in figure A.11 implements a T gate on qubit 2, up to Pauli Z errors.

- The measurement pattern defined in figure A.12 implements a T gate on qubit 3, up to Pauli Z errors.

Proof. We only need to prove the first part of the lemma, since the three parts are symmetric. Note in this measurement pattern, we treat S gates as errors. By lemmas 5.4.1, 5.4.4, and 5.4.10, after performing measurements defined in figure 5.16, we will have a T gate on qubit 1, two S errors on qubit 1, two S errors on qubit 2, two S errors on qubit 3, and Pauli Z errors on qubits 1, 2, and 3. By the spider fusion rule (f), two S errors fuse up to become a Pauli Z error. Therefore, this measurement pattern implements a T gate on qubit 1 up to Pauli Z errors. ■

The following lemma 5.4.15 justifies measurement patterns implementing S or identity gates on the filled orange box system.

Lemma 5.4.15. For $i \in \{1, 2, 3\}$, the measurement pattern defined in figure 5.18 implements a S gate on qubit i if $A_i = S$; otherwise it implements an identity gate on qubit i.

Proof. By lemmas 5.4.2, 5.4.4, 5.4.9, 5.4.10, after performing measurements defined in figure 5.18, for $i \in \{1, 2, 3\}$, if $A_i = S$, then there will be three S gates on qubit i; otherwise there will be two S gates on qubit i. By the spider fusion rule (f), two S gates fuse up to become a Pauli Z error. Therefore, on qubit $i \in \{1, 2, 3\}$, this measurement implements a S gate up to Pauli Z errors if $A_i = S$; otherwise it implements a identity gate up to Pauli Z errors. ■

The following lemma 5.4.16 justifies measurement patterns implementing binary $\frac{\pi}{4}$ phase gadgets on the filled orange box system.

Lemma 5.4.16. The following three statements hold true:

- The measurement pattern defined in figure 5.20 implements a binary $\frac{\pi}{4}$ phase gadget on qubits 1 and 2, up to Pauli Z errors.

- The measurement pattern defined in figure A.13 implements a binary $\frac{\pi}{4}$ phase gadget on qubits 1 and 3, up to Pauli Z errors.
- The measurement pattern defined in figure A.14 implements a binary $\frac{\pi}{4}$ phase gadget on qubits 2 and 3, up to Pauli Z errors.

Proof. We only need to prove the first part of the lemma, since the three parts are symmetric.

This measurement pattern is performed in two steps, in the first step we perform measurement defined on the orange box, and the values of $\Phi_{1,2}$ and Θ are determined by whether a phase gadget error is produced in this measurement. See the proof of lemma 5.4.5 for details about how to determine whether a phase gadget error appears in the measurement.

If no phase gadget error is produced, then $\Phi_{1,2} = S_*$ and $\Theta = S$. By lemmas 5.4.5, 5.4.4, and 5.4.9, after performing measurements in the second step, we will have a binary $\frac{\pi}{4}$ phase gadget on qubits 1 and 2, two S gates on qubit 1, two S gates on qubit 2, two S gates on qubit 3, and Pauli Z errors on qubits 1, 2, and 3. By the spider fusion rule (f), two S gates are equal to an identity operation up to Pauli Z errors. So, in this case, The measurement pattern implements a binary $\frac{\pi}{4}$ phase gadget on qubits 1 and 2, up to Pauli Z errors.

Otherwise, a $\frac{\pi}{2}$ phase gadget error is produced, then we have $\Phi_{1,2} = P_{1,2}$ and $\Theta = I$. By lemmas 5.4.5, 5.4.6, and 5.4.10, after performing measurements in the second step, we will have a binary $\frac{\pi}{4}$ phase gadget on qubits 1 and 2, a binary $\frac{\pi}{2}$ phase gadget on qubits 1 and 2, two S errors on qubit 3, a $\frac{\pi}{2}$ phase gadget error on qubits 1 and 2, and Pauli Z errors on qubits 1, 2, and 3. By the phase gadget fusion rule (PG1), the binary $\frac{\pi}{2}$ phase gadget and the $\frac{\pi}{2}$ phase gadget error fuse into a π phase gadget, and by equation (PGπ) the π phase gadget "break down" into Pauli Z errors. By the spider fusion rule (f), two S errors fuse up to one Pauli Z error. So, in this case, the measurement pattern implements a binary $\frac{\pi}{4}$ phase gadget on qubits 1 and 2, up to Pauli Z errors. ■

The following lemma 5.4.17 justifies measurement patterns implementing binary $\frac{\pi}{2}$ phase gadgets on the filled orange box system.

Lemma 5.4.17. The following three statements hold true:

- The measurement pattern defined in figure A.17 implements a binary $\frac{\pi}{2}$ phase gadget on qubits 1 and 2, up to Pauli Z errors.
- The measurement pattern defined in figure A.15 implements a binary $\frac{\pi}{2}$ phase gadget on qubits 1 and 3, up to Pauli Z errors.
- The measurement pattern defined in figure A.16 implements a binary $\frac{\pi}{2}$ phase gadget on qubits 2 and 3, up to Pauli Z errors.

Proof. We only need to prove the first part of the lemma, since the three parts are symmetric.

By lemmas 5.4.2, 5.4.6, 5.4.9, and 5.4.10, after performing the measurements defined in A.17, we will have a binary $\frac{\pi}{2}$ phase gadget on qubits 1 and 2, three S errors on qubit 3, two S errors on qubit 1, two S errors on qubit 2, and Pauli Z errors on qubits 1, 2, and 3. By the spider fusion rule (f), two S errors fuse up to one Pauli Z error. Therefore, this measurement pattern implements a binary $\frac{\pi}{2}$ phase gadget on qubits 1 and 2, up to Pauli Z errors. ■

The following lemma 5.4.18 justifies the measurement pattern implementing trinary $\frac{\pi}{4}$ phase gadgets on the filled orange box system.

Lemma 5.4.18. The measurement pattern defined in figure 5.21 implements a trinary $\frac{\pi}{4}$ phase gadget on qubits 1, 2, and 3, up to Pauli Z errors.

Proof. This measurement pattern is performed in two steps, in the first step we perform measurement defined on the orange box, and the values of Φ_* and Θ are determined by whether a phase gadget error is produced in this measurement.

If no phase gadget is produced, then $\Phi_* = S_*$ and $\Theta = S$. By lemmas 5.4.7, 5.4.4, and 5.4.9, after performing measurements in the second step, we will have a trinary

$\frac{\pi}{4}$ phase gadget on qubits 1, 2, and 3, two S gates on qubit 1, two S gates on qubit 2, two S gates on qubit 3, and Pauli Z errors on qubits 1, 2, and 3. By the spider fusion rule (f), two S errors fuse up to one Pauli Z error. In this case, The measurement pattern implements a trinary $\frac{\pi}{4}$ phase gadget on qubits 1, 2 and 3, up to Pauli Z errors.

Otherwise, a trinary $\frac{\pi}{2}$ phase gadget error is produced, then we have $\Phi_* = P_*$ and $\Theta = I$. By lemmas 5.4.7, 5.4.8, and 5.4.10, after performing measurements in the second step, we will have a trinary $\frac{\pi}{4}$ phase gadget on qubits 1, 2 and 3, a trinary $\frac{\pi}{2}$ phase gadget on qubits 1, 2, and 3, a trinary $\frac{\pi}{2}$ phase gadget error on qubits 1, 2, and 3, and Pauli Z errors on qubits 1, 2, and 3. By the phase gadget fusion law (PG1), the trinary $\frac{\pi}{2}$ phase gadget and the trinary $\frac{\pi}{2}$ phase gadget error fuse into a π phase gadget, and by equation (PGπ) π phase gadget "break down" into Pauli Z errors. Therefore, the measurement pattern implements a trinary $\frac{\pi}{4}$ phase gadget on qubits 1, 2, and 3, up to Pauli Z errors. ■

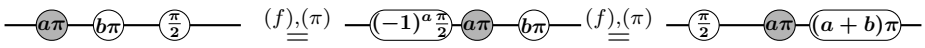
5.5 Correction of measurement errors

In this section, we show how we make small changes to measurement patterns introduced in section 5.3, so that we can use the idea of feed-forward to correct measurement errors. The feed-forward technique is introduced in section 2.9.2 and how to equip measurement patterns with feed-forward is introduced in section 2.9.3. Recall that in order to apply feed-forward to correct errors, we need to ensure that all defined measurement patterns admit feed-forward. A measurement pattern admits feed-forward, if it can take Pauli Z and X errors coming on its input wires, and after the measurement pattern is performed, only Pauli Z and X errors will appear on its output wires.

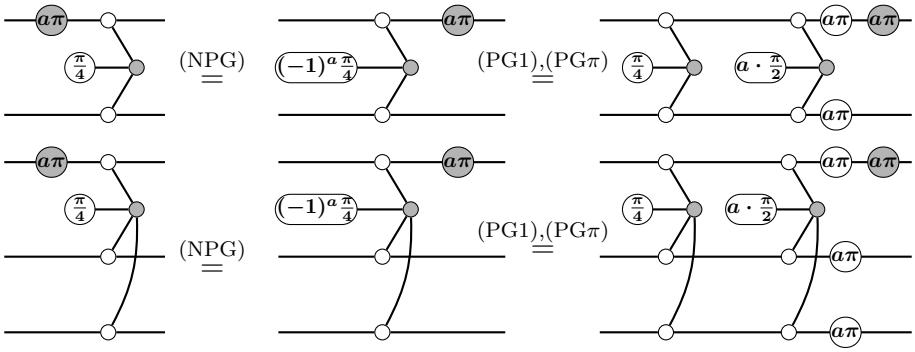
To begin with, let us consider what kind of non-Pauli errors could be produced if Pauli X and Z errors pass through our measurement patterns.

Fact 5.5.1. For measurement patterns that implement a T gate, Pauli Z errors can freely pass through the implementation by rule (f), but when the Pauli X error passes through a T gate, a Pauli Z error and a S error are produced, where the S error is non-Pauli:

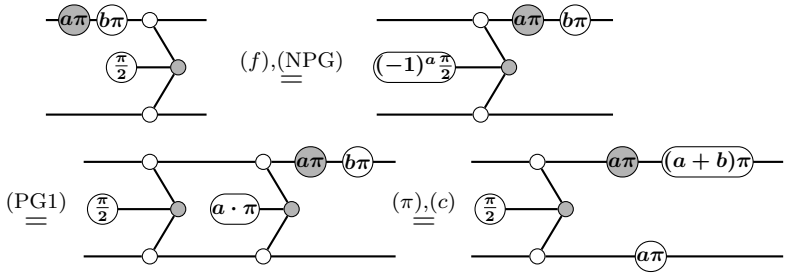
Fact 5.5.2. For measurement patterns that implement S or identity gates, Pauli X and Z errors can both freely pass through the implementation without producing any errors other than Pauli Z and X errors, since we have:

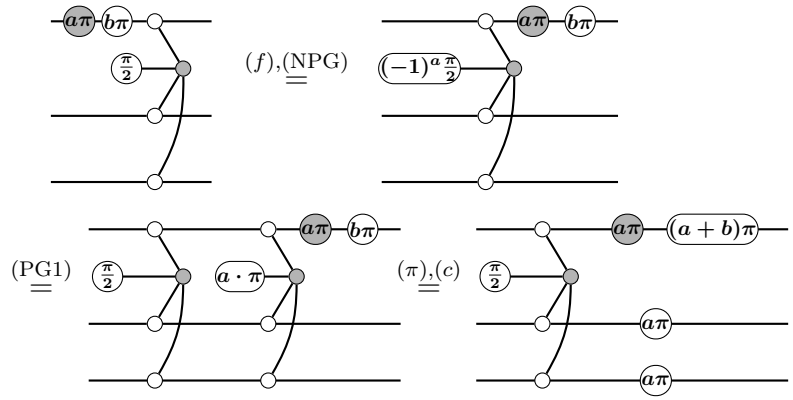


Fact 5.5.3. For measurement patterns that implement a binary or trinary $\frac{\pi}{4}$ phase gadget, Pauli Z errors can freely pass through the implementation by rule (f), but when Pauli X error passes through a binary or trinary $\frac{\pi}{4}$ phase gadget, a binary or trinary $\frac{\pi}{2}$ phase gadget error and Pauli Z errors are produced:



Fact 5.5.4. For measurement patterns that implement a binary or trinary $\frac{\pi}{2}$ phase gadget, Pauli X and Z errors can both freely pass through the implementation without producing any errors other than Pauli Z and X errors, since we have:





We will use the above facts in our following discussions.

Now, we make changes to measurement patterns defined in section 2.9.3 so as to make them admit feed-forward. The concept of measurement patterns admitting feed-forward is introduced in section 2.9.3. Our idea of making a measurement pattern admit feed-forward is to adjust measurement patterns on subsystems like orange, red, and grey boxes according to the Pauli X and Z errors coming on input wires in a way where all undesired errors like S errors and phase gadget errors produced within this measurement pattern are cancelled out or transformed to Pauli X and Z errors, and thus only Pauli X and Z errors could appear on output wires.

Measurement patterns in figures 5.16, A.11, and A.12 For the measurement pattern defined in figure 5.16, we change it to the measurement pattern illustrated in figure 5.23.

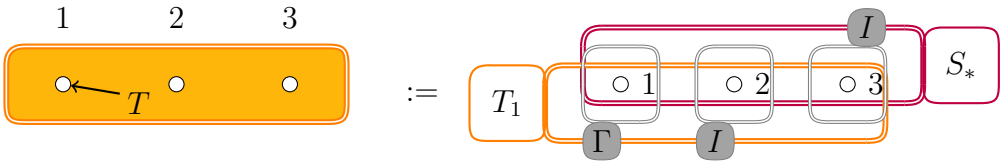


Figure 5.23: A measurement pattern defined on the filled orange box system that admits feed-forward. The value of Γ in this pattern is determined by whether there is a Pauli X error coming to qubit 1: $\Gamma = S$ if there is a Pauli X error coming to qubit 1; otherwise $\Gamma = I$. Qubits 1, 2, 3 are both inputs and outputs.

Proposition 5.5.5. The measurement pattern defined in figure 5.23 implements a T gate on qubit 1 and admits feed-forward.

Proof. We assume that there are Pauli X and Z errors on input wires, since if there are no Pauli X or Z errors on input wires, then the case is discussed in lemma 5.4.14. By fact 5.5.2, and lemmas 5.4.1, 5.4.4, and 5.4.9, after measurements are performed, Pauli X and Z errors can pass through qubits 2 and 3 and only Pauli Z and X errors appear on output wires. By fact 5.5.1, and lemmas 5.4.1, 5.4.4, and 5.4.9, after measurements are performed, Pauli Z errors can freely pass through qubit 1, but when a Pauli X error passes through qubit 1, an additional S error will be produced, since there is a T gate on qubit 1.

Therefore, by facts 5.5.2 and 5.5.1, and lemmas 5.4.1, 5.4.4, and 5.4.9, after measurements are performed and Pauli X and Z errors pass through qubits 1, 2, and 3, there will be a T gate on qubit 1, four S errors on qubit 1, and Pauli Z and X errors on qubits 1, 2, and 3. By rule of (f), four S errors fuse up to an identity wire. So, this measurement pattern implements a T gate on qubit 1 and admits feed-forward. ■

We can make similar changes to measurement patterns defined in figures A.11 and A.12 so as to make them admit feed-forward, since these measurement patterns are symmetric to the one defined in figure 5.16.

Measurement pattern in figure 5.18 This measurement pattern admits feed-forward on its own, since by fact 5.5.2, Pauli X and Z errors can pass through S or identity gates without producing errors other than Pauli X and Z errors. So, we do not need to make any change to it. Recall that this measurement pattern is defined in a general form that includes cases like those defined in figures 5.17 and 5.19

Measurement patterns in figures 5.20, A.13, and A.14 Next, we will use the notation Err-$X_{1,2}$ to describe Pauli X errors coming on qubits 1 and 2: If there are no Pauli X error coming on qubits 1 or 2, or there are a Pauli X error coming on qubit 1 and a Pauli X error coming on qubit 2, then Err-$X_{1,2} = even$; otherwise

Err-$X_{1,2} = odd$. For the measurement pattern defined in figure 5.20, we change it to the measurement pattern defined in figure 5.24.

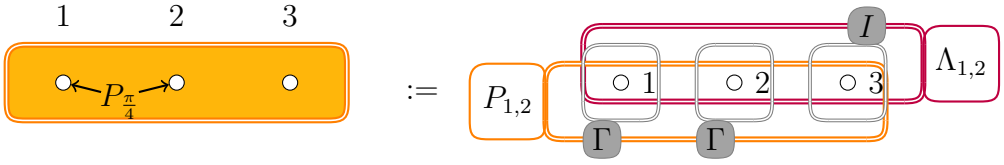


Figure 5.24: A measurement pattern on the filled orange box system that admits feed-forward. This pattern is performed in two steps. First, we perform on the orange box $P_{1,2}$ measurement pattern. The values of $\Lambda_{1,2}$ and Γ depend on whether a phase gadget error is produced in this measurement and whether there are Pauli X errors coming on qubits 1 and 2: If a phase gadget error is produced and Err-$X_{1,2} = even$, then $\Lambda_{1,2} = P_{1,2}$ and $\Gamma = I$; if a phase gadget error is produced and Err-$X_{1,2} = odd$, then $\Lambda_{1,2} = S_*$ and $\Gamma = S$; if no phase gadget error is produced and Err-$X_{1,2} = even$, then $\Lambda_{1,2} = S_*$ and $\Gamma = S$; if no phase gadget error is produced and Err-$X_{1,2} = odd$, then $\Lambda_{1,2} = P_{1,2}$ and $\Gamma = I$. Then we perform all the rest of measurement patterns. Qubits 1, 2, and 3 are both inputs and outputs.

Proposition 5.5.6. The measurement pattern defined in figure 5.24 implements a binary $\frac{\pi}{4}$ phase gadget on qubits 1 and 2 and admits feed-forward.

Proof. We assume that there are Pauli X and Z errors on input wires, since if there are no Pauli X or Z errors on input wires, then the case is discussed in lemma 5.4.16. By fact 5.5.2 and lemmas 5.4.16, 5.4.4, 5.4.6, and 5.4.10, after measurements are performed, Pauli X and Z errors can pass through qubit 3 and only Pauli Z and X errors appear on the output wire of qubit 3. For qubits 1 and 2, we have the following cases:

- (i) there is no Pauli X error coming on qubit 1 or 2. In this case, this measurement pattern becomes that defined in figure 5.20. By lemma 5.4.16 and fact 5.5.3, after the measurement pattern is performed and Pauli Z errors pass through qubits 1 and 2, there will be a binary $\frac{\pi}{4}$ phase gadget on qubits 1 and 2 and Pauli Z errors on qubits 1 and 2.

- (ii) there are one Pauli X error coming on qubit 1 and another Pauli X error coming on qubit 2. In this case, this measurement pattern becomes the

one defined in figure 5.20. By lemma 5.4.16, facts 5.5.3 and 5.5.4, after the measurement pattern is performed and Pauli X and Z errors pass through qubits 1 and 2, there will be a binary $\frac{\pi}{4}$ phase gadget, Pauli X and Z errors, and phase gadget errors on qubits 1 and 2. If no phase gadget error is produced in the first step, then there will be two $\frac{\pi}{2}$ phase gadget errors on qubits 1 and 2; otherwise, there will be four $\frac{\pi}{2}$ phase gadget errors on qubits 1 and 2.

- (iii) there is one Pauli X error coming on qubit 1 but no Pauli X error coming on qubit 2. If a phase gadget error is produced in the first step, then we have $\Lambda_{1,2} = S_*$ and $\Gamma = S$. By lemmas 5.4.4, 5.4.9, facts 5.5.3 and 5.5.4, after measurements are performed and Pauli X and Z errors pass through qubits 1, 2, and 3, there will be a binary $\frac{\pi}{4}$ phase gadget on qubits 1 and 2, two $\frac{\pi}{2}$ phase gadget errors on qubits 1 and 2, two S errors on qubit 1, two S errors on qubit 2, and Pauli Z and X errors on qubits 1, 2, and 3. Otherwise, no phase gadget error is produced in the first step, then we have $\Lambda_{1,2} = P_{1,2}$ and $\Gamma = I$. By lemmas 5.4.6, 5.4.10, facts 5.5.3 and 5.5.4, after measurements are performed and Pauli X and Z errors pass through qubits 1, 2, and 3, there will be a binary $\frac{\pi}{4}$ phase gadget on qubits 1 and 2, two $\frac{\pi}{2}$ phase gadget errors on qubits 1 and 2, and Pauli X and Z errors on qubits 1, 2, and 3.

- (iv) there is one Pauli X error coming on qubit 2 but no Pauli X error coming on qubit 1. This case is symmetric to (iii).

In each case, on qubits 1 and 2, there are even number of $\frac{\pi}{2}$ phase gadget errors, and on any qubit, there are even number of S errors. Two $\frac{\pi}{2}$ phase gadget errors fuse up to π phase gadgets by (PG1) and then "break down" to Pauli Z errors by equation (PGπ), and two S errors fuse up to be Pauli Z errors by rule of (f). Therefore, this measurement pattern implements a binary $\frac{\pi}{4}$ phase gadget on qubits 1 and 2 and admits feed-forward. ■

We can make similar changes to other measurement patterns on filled orange box such as those in figures A.13 and A.14 so as to make them admit feed-forward, since these measurement patterns are symmetric to that defined in figure 5.20.

Measurement pattern in figure 5.21 For the measurement pattern defined in figure 5.21, we change it to the measurement pattern illustrated in figure 5.25, where the notation Err-X_* to describe Pauli X errors coming on qubits 1, 2, and 3: If there are no Pauli X error coming on qubits 1, 2, and 3, or there are two Pauli errors on two out of the three qubits, then Err-$X_* = even$; otherwise Err-$X_* = odd$. The

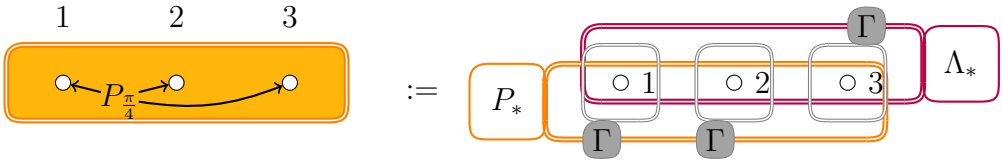


Figure 5.25: A measurement pattern on the filled orange box system that admits feed-forward. This pattern is performed in two steps. First, we perform on the orange box P_* measurement pattern. The value of Λ_* and Γ depend on whether there is a phase gadget is produced in this measurement and whether there are Pauli X errors coming on qubits 1 and 2: If a phase gadget error is produced and Err-$X_* = even$, then $\Lambda_* = P_*$ and $\Gamma = I$; if a phase gadget error is produced and Err-$X_* = odd$, then $\Lambda_* = S_*$ and $\Gamma = S$; if no phase gadget error is produced and Err-$X_* = even$, then $\Lambda_* = S_*$ and $\Gamma = S$; if no phase gadget error is produced and Err-$X_* = odd$, then $\Lambda_* = P_*$ and $\Gamma = I$. Then we perform all the rest of measurement patterns. Qubits 1, 2, and 3 are both inputs and outputs.

justification for this pattern is similar to the proof for proposition 5.5.6.

Measurement patterns in figures A.17, A.15, A.16, and 5.22 These measurement patterns admit feed-forward themselves, since by fact 5.5.4, Pauli X and Z errors can pass through $\frac{\pi}{2}$ phase gadgets without producing errors other than Pauli X and Z errors, so we do not need to make any change to them.

Other measurement patterns on filled orange boxes We can use similar techniques to make measurement patterns like those defined in figures A.18, A.21, and A.19 admit feed-forward.

Measurement patterns on ropes For measurement patterns defined on rope system, namely the ones defined in figures 5.14 and 5.15, they admit feed-forward on their own, since by rule (h) Pauli X and Z errors can pass through Hadamard gates without producing errors other than Pauli X and Z errors. Therefore, we do not need to make any change to them either.

Now, we put everything together.

Proposition 5.5.7. Our MBQC model can correct measurement errors by feed-forward.

Proof. Our new resource state is composed by filled orange boxes and ropes. After making changes discussed in this section, all measurement patterns defined on fragments (filled orange boxes and ropes) of our resource state admit feed-forward. Therefore, our MBQC model can correct errors by feed-forward. ■

5.6 Universality and determinism

In this section, we show that our new resource state based MBQC model can achieve universal computations with a deterministic protocol. The idea of proving universality introduced in section 2.7 is used.

Proposition 5.6.1. Our MBQC model can achieve universal computations with a deterministic protocol.

Proof. We show that we can implement gates from the universal gate set $\{S$, $Hadamard$, $CNOT$, $T\}$ and compose them arbitrarily.

By lemmas 5.4.15, 5.4.13, and 5.4.14, we can implement S gates, Hadamard gates, and T gates, up to Pauli X and Z errors. By lemmas 5.4.15, 5.4.17, 5.4.12, fact 5.5.2, and equation (FT2), we can implement a CZ gate on two neighbouring qubits up to Pauli X and Z errors, using the measurement pattern shown in figure 5.26. We can

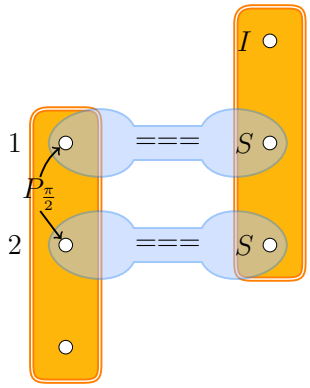
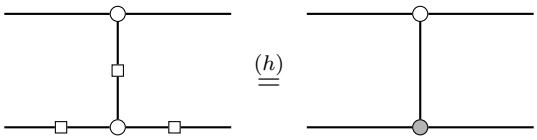
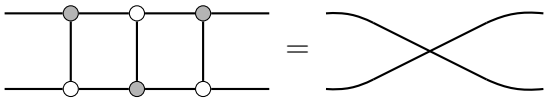


Figure 5.26: A measurement pattern that implements a CZ gate on qubits 1, 2, up to Pauli X and Z errors.

compose Hadamard gates and a CZ gate on two neighbouring qubits to implement a CNOT gate on two neighbouring qubits, since we have:



By how we construct the resource state as illustrated in figure 5.7, we can compose implemented gates in any way we want. By sequential composing three implementations of the CNOT gate on neighbouring qubits, we can implement a SWAP gate on neighbouring two qubits, since we have: [10]



By composing CNOT and SWAP gates on neighbouring qubits together, we can implement CNOT gates on arbitrary two qubits using the technique illustrated in figure 5.27. Since we can (i) implement all gates from the universal gate set up to Pauli Z and X errors, (ii) compose them in an arbitrary way, and (iii) correct Pauli X and Z errors by applying feed-forward technique as discussed in section 5.5, we thus conclude that our new resource state based MBQC model can achieve universal computations with a deterministic protocol.

■

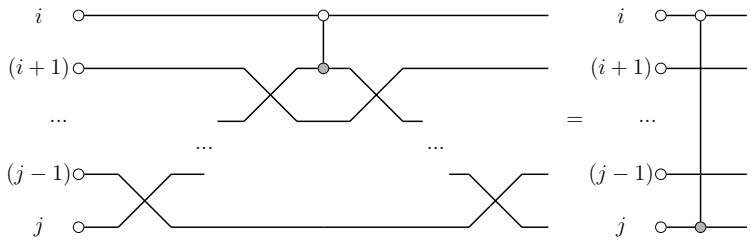


Figure 5.27: Technique to implement a CNOT gate on two qubits over arbitrary distance by composing CNOT and SWAP gates on neighbouring qubits. Assume we want to implement a CNOT gate on qubits i and j $(i < j)$. We first implement a CNOT gate on qubits i and $(i+1)$, then use SWAP gates to "slide" the CNOT gate to the right place and apply additional SWAP gates so that all SWAP gates cancel out with each other.

Examples We provide an example of using our MBQC model.

Proposition 5.6.2. The measurement pattern defined in figure 5.28 implements a CCZ gate on qubits 1, 2, and 3, up to X and Z Pauli errors.

Remark 5.6.3. This measurement pattern faithfully implement each gate on the right-hand side of the equation (FT3).

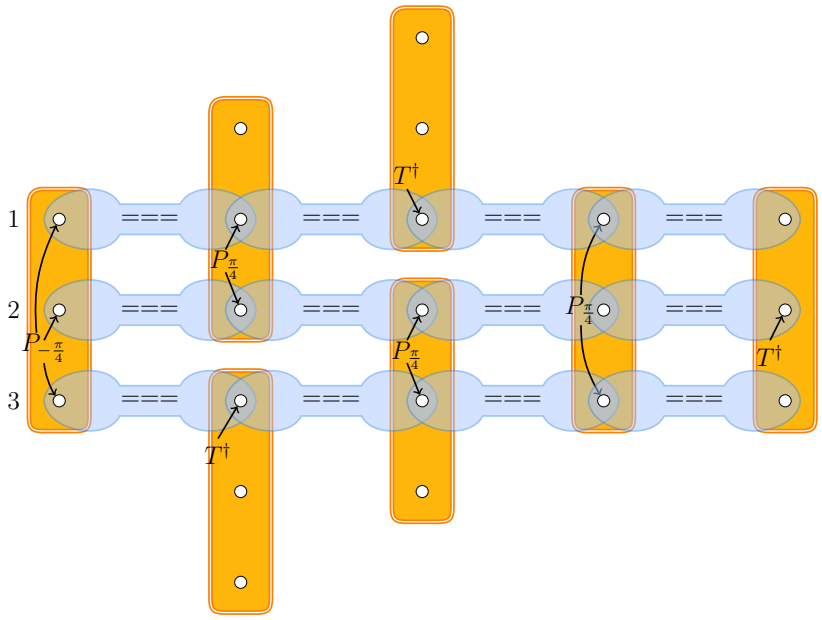


Figure 5.28: A measurement pattern that implements a CCZ gate on qubits 1, 2, and 3, up to X and Z Pauli errors.

6 Conclusion and Future Work

6.1 Conclusions

In this thesis, we have given graphical proofs in ZH-calculus for MBQC protocols based on the GGM state and Union Jack state, investigated the link between hypergraph and phase gadget state MBQC, and presented a new MBQC model that achieves universal computations with a deterministic scheme.

As we saw, our proofs for hypergraph MBQC protocols are not only concise and intuitive, but also easy to understand and validate. Our contribution provides more explainability and interpretability for hypergraph MBQC models. In [12], the authors have used a lot of matrix notations to formulate Pauli Z and X measurements on the GGM state, while we have diagrammatically justified measurement patterns step by step, and each single step of our derivations only uses simple and concise rules of the ZH-calculus. In particular, we have clearly proved the correctness of measurement patterns where several Pauli X measurements are performed to deterministically implement CZ wires. Since a single-qubit Pauli X measurement on a hypergraph state yields a state that is not a hypergraph state [18], it is confusing at first glance how several Pauli X measurements interact with each other to implement a state that belongs to the family of hypergraph states. Our diagrammatic proofs interpret the process of performing measurement patterns into

multiple stages where transformations from one stage to another follow simple rules of ZH-calculus and our intuitions. In [11], the authors have used matrix notations and text descriptions to justify that a CCZ gate can be implemented by composing U_I and SWAP gates together, while we have translated the large measurement pattern into a ZH-diagram and seen that gates can "slide" along wires and cancel out with each other, which also follows our intuitions.

Although hypergraph states and phase gadget states both have a structure of hypergraphs, the relation between hypergraph and phase gadget state MBQC has not been studied before. We have built a bridge between hypergraph state MBQC and phase gadget state MBQC by applying Graphical Fourier theory to demonstrate the equivalence between the Union Jack state and a phase gadget state, and our work also provides motivations for future studies of relations between MBQC models. Besides, this equivalence we have shown also confirms that it is possible to use only trinary $\frac{\pi}{4}$ phase gadgets to build a universal resource state, which provides inspirations for future constructions of MBQC models.

Inspired by the equivalence between Union Jack states and phase gadget states, we have used trinary $\frac{\pi}{4}$ phase gadgets to construct a new resource state. Our MBQC model provides an alternative idea to construct a universal MBQC model with a scheme to deterministically correct errors.

We have introduced notations of boxes and ropes to better present our resource state and describe measurement patterns. These notations enable us to represent complex systems in an elegant way by systematically encapsulating qubits and phase gadgets into boxes or ropes. More importantly, our notations allow us to easily define a lot of measurement patterns on one system or one fragment of resource states. However, our notations have not modelled the errors and dependency between measurements in an ideal way, so we need additional descriptions in our proofs to make our scheme of error correction clear. As a contrast to our notation, the notation given in [10] elegantly describes errors passing among fragments, though it does not model the dependency between measurements.

It turns out that the idea of using trinary $\frac{\pi}{4}$ phase gadgets leads us to a method of generalizing the construction of phase gadget states. By a Pauli Z measurement, we can transform a trinary $\frac{\pi}{4}$ phase gadget to a binary $\frac{\pi}{4}$ phase gadget (up to Pauli errors). Hence any resource state constructed by binary $\frac{\pi}{4}$ phase gadgets can be implemented by some phase gadget state composed by trinary $\frac{\pi}{4}$ phase gadgets. So, our resource state seems to belong a more generalized family of phase gadget states, compared to the resource state presented in [10] which is constructed by binary $\frac{\pi}{4}$ phase gadgets.

6.2 Future work

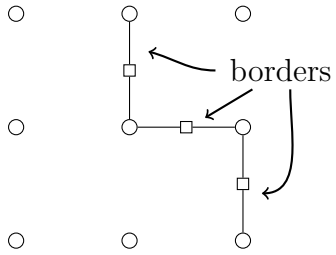
Although trinary phase gadgets seem to be more generalized building blocks than binary ones, trinary-phase-gadget constructions potentially cost much more qubits than binary-phase-gadget constructions. Therefore, it would be interesting to investigate whether we can use trinary phase gadgets as building blocks and construct a more compact universal resource state.

Besides, it is not clear whether using phase gadgets with more input wires brings us additional expressive power. It might be the case that for a phase gadget state constructed by k-ary phase gadgets where $k > 2$, we could construct an equivalent or similar resource state using phase gadgets with less input wires. So, some possible future work could be a study of the relation between the number of inputs of phase gadgets and their expressive power in MBQC.

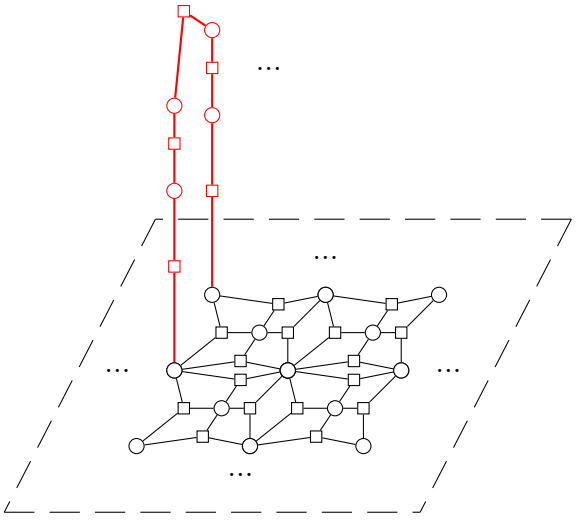
In another direction, we could also move away from using only one type of gates and try using two or more types of phase gadgets to construct universal resource states. For example, we might consider using both binary and trinary phase gadgets as building blocks. Relaxing the restriction on the type of phase gadgets may add flexibility to our work of constructing.

It would be interesting to consider modifying the structure of Union Jack state so that we get a new resource state that admits a deterministic method to correct errors in an easy way while maintaining universality achieved by the Union Jack state. We have tried to construct 3-D structured states where the bottom layer is

exactly the Union Jack state, and additional structures are added above the bottom layer. By adding additional structures, it is not hard to eliminate side effects caused by errors of Pauli Z measurements on the Union Jack state. It turns out that Pauli Z measurements in protocols of Union Jack state MBQC are used to form desired "borders" which are CZ wires or edges between adjacent qubits [11]:

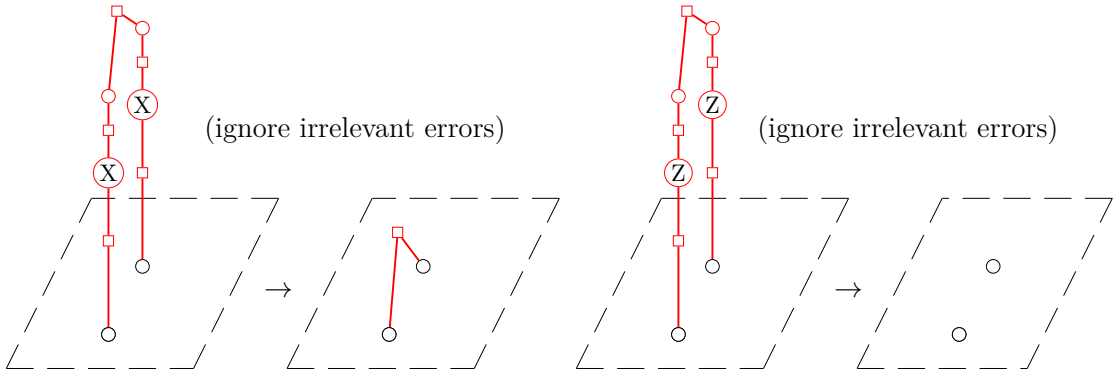


Errors produced by Pauli Z measurements can lead to failure to form desired CZ edges or appearance of undesired CZ edges. To eliminate this side effect, we could consider the following toy construction:



The additional structure composed above the layer of Union Jack state is used to deterministically select desired CZ edges on two neighbouring or adjacent qubits. We can select whether to compose an additional CZ gate on two neighbouring qubits by

choice of Pauli X or Z measurements on two middle qubits of the additional structure:



If a "border" has not appeared because of errors, then we compose an additional CZ gate on the corresponding position; if there is an undesired CZ edge, then we also compose an additional CZ gate on the two qubits, since they cancel out each other by equation (CZ1). With this toy construction, we can deterministic select "borders" in the Union Jack state, which could potentially help us shape the state into some interesting structure. It is worth studying how to design and use additional structures to cleverly correct all measurement errors produced in the Union Jack state based MBQC.

Appendices

A

Additional measurement patterns

A.1 Measurement patterns for chapter 3

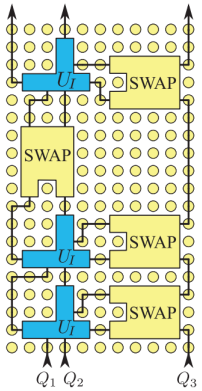


Figure A.1: The measurement pattern proposed in [11] that implements a CCZ gate. U_I and SWAP in this figure are gates implemented by corresponding measurement patterns given in [11]. U_I gate is a three-qubit operation such that $U_I^{(i,j,k)} = CCZ^{(i,j,k)}\sqrt{CZ}^{(i,j)}\sqrt{CZ}^{(j,k)}$.

A.2 Measurement patterns for chapter 5

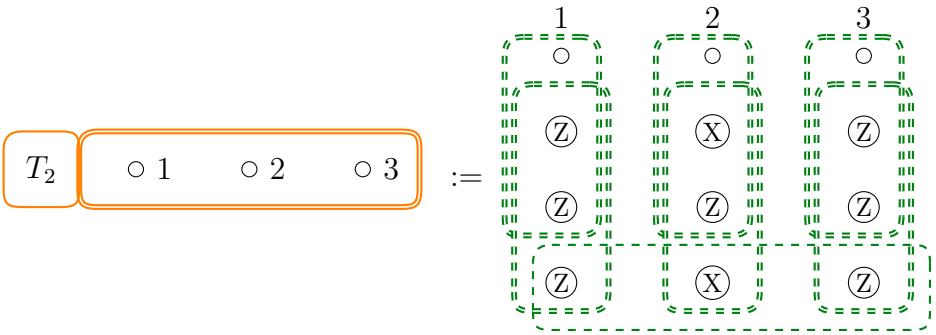


Figure A.2: A measurement pattern on the orange box notation. This measurement pattern implements a T gate on the qubit 2. Qubits 1, 2, and 3 are both inputs and outputs.

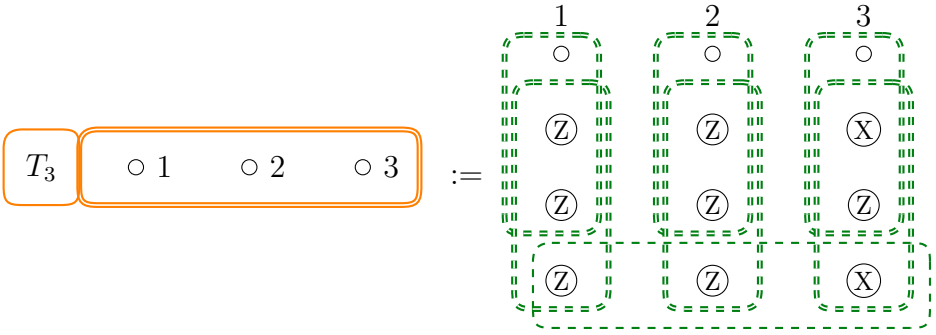


Figure A.3: A measurement pattern on the orange box notation. This measurement pattern implements a T gate on the qubit 3. Qubits 1, 2, and 3 are both inputs and outputs.

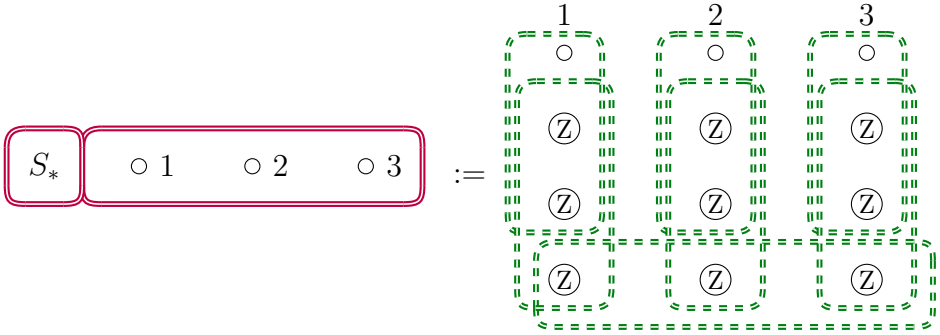


Figure A.4: A measurement pattern on the red box notation. This measurement pattern implements S gates on qubits 1, 2, and 3. Qubits 1, 2, and 3 are both inputs and outputs.

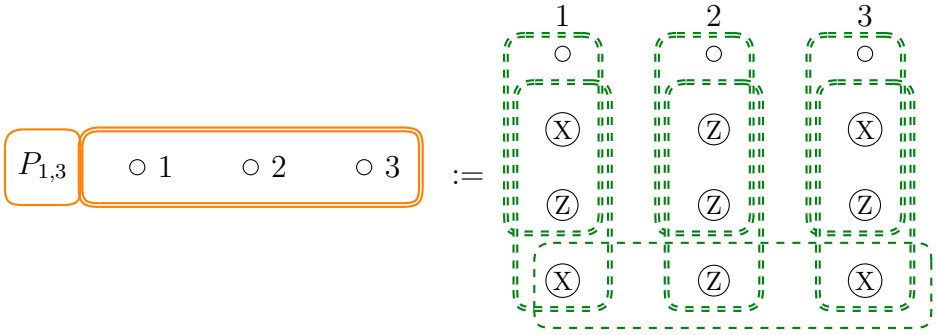


Figure A.5: A measurement pattern on the orange box notation. This measurement pattern implements a binary $\frac{\pi}{4}$ phase gadget on qubits 1 and 3. Qubits 1, 2, and 3 are both inputs and outputs.

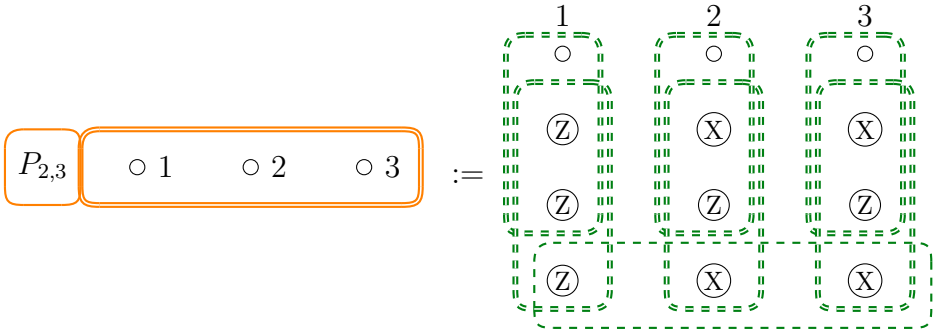


Figure A.6: A measurement pattern on the orange box notation. This measurement pattern implements a binary $\frac{\pi}{4}$ phase gadget on qubits 2 and 3. Qubits 1, 2, and 3 are both inputs and outputs.

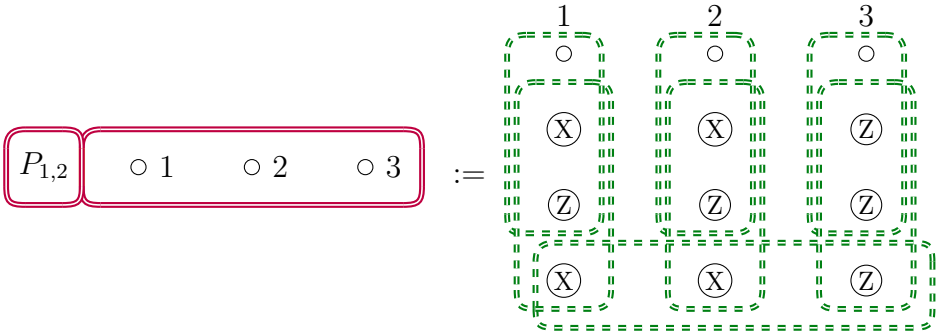


Figure A.7: A measurement pattern on the red box notation. This measurement pattern implements a binary $\frac{\pi}{2}$ phase gadget on qubits 1 and 2. Qubits 1, 2, and 3 are both inputs and outputs.

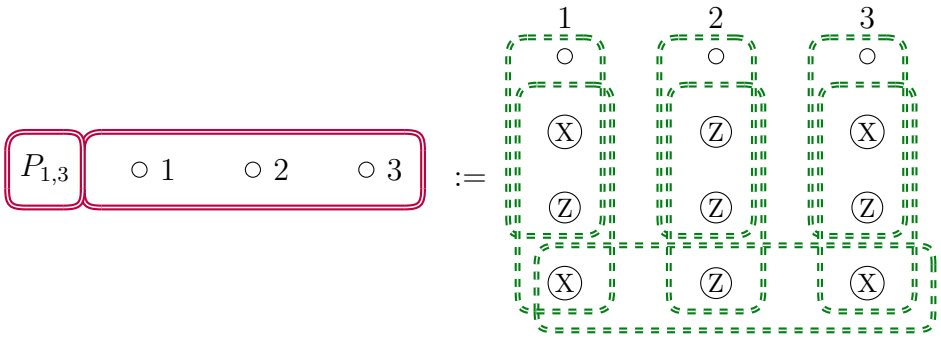


Figure A.8: A measurement pattern on the red box notation. This measurement pattern implements a binary $\frac{\pi}{2}$ phase gadget on qubits 1 and 3. Qubits 1, 2, and 3 are both inputs and outputs.

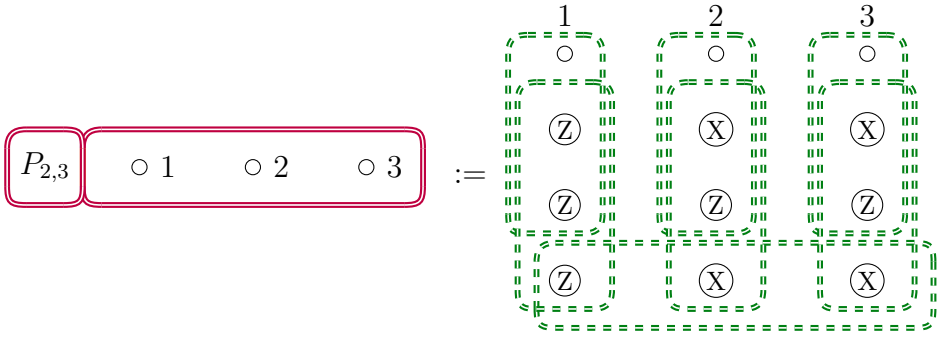


Figure A.9: A measurement pattern on the red box notation. This measurement pattern implements a binary $\frac{\pi}{2}$ phase gadget on qubits 2 and 3. Qubits 1, 2, and 3 are both inputs and outputs.

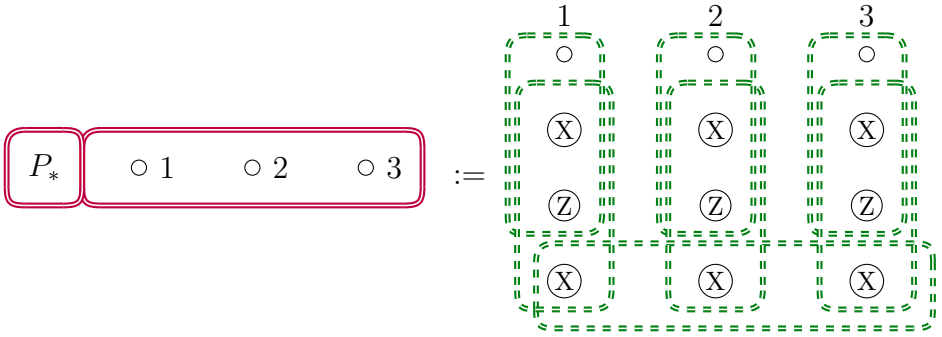


Figure A.10: A measurement pattern on the red box notation. This measurement pattern implements a trinary $\frac{\pi}{2}$ phase gadget on qubits 1, 2, and 3. Qubits 1, 2, and 3 are both inputs and outputs.

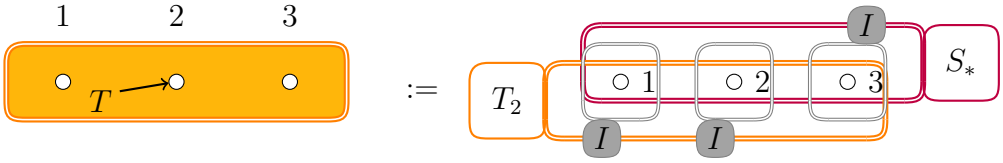


Figure A.11: A measurement pattern on the filled orange box notation. This measurement pattern implements a T gate on qubit 2. Qubits 1, 2, and 3 are both inputs and outputs.

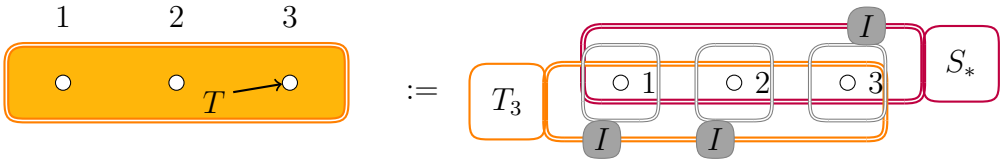


Figure A.12: A measurement pattern on the filled orange box notation. This measurement pattern implements a T gate on qubit 3. Qubits 1, 2, and 3 are both inputs and outputs.

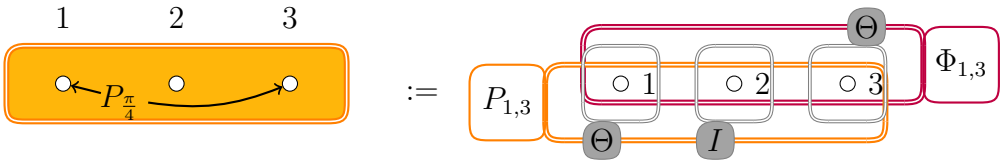


Figure A.13: A measurement pattern on the filled orange box notation. First, we perform on the orange box $P_{1,3}$ measurement pattern. If a phase gadget error is produced then $\Phi_{1,3} = P_{1,3}$ and $\Theta = I$; otherwise $\Phi_{1,3} = S_*$ and $\Theta = S$. Then we perform all the rest of measurement patterns. This measurement pattern implements a binary $\frac{\pi}{4}$ phase gadget on qubits 1 and 2. Qubits 1, 2, and 3 are both inputs and outputs.

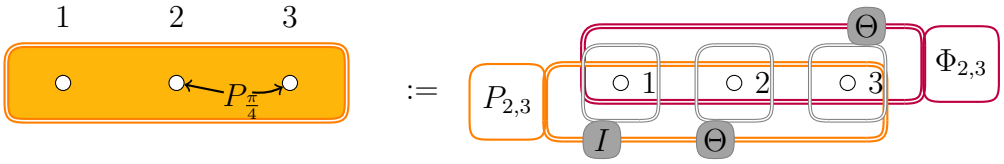


Figure A.14: A measurement pattern on the filled orange box notation. First, we perform on the orange box $P_{2,3}$ measurement pattern. If a phase gadget error is produced then $\Phi_{2,3} = P_{2,3}$ and $\Theta = I$; otherwise $\Phi_{2,3} = S_*$ and $\Theta = S$. Then we perform all the rest of measurement patterns. This measurement pattern implements a binary $\frac{\pi}{4}$ phase gadget on qubits 1 and 2. Qubits 1, 2, and 3 are both inputs and outputs.

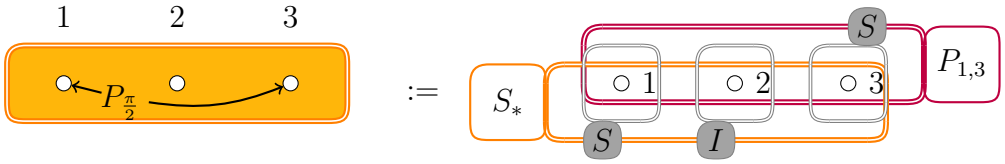


Figure A.15: A measurement pattern on the filled orange box notation. This measurement pattern implements a binary $\frac{\pi}{2}$ phase gadget on qubits 1 and 3. Qubits 1, 2, and 3 are both inputs and outputs.

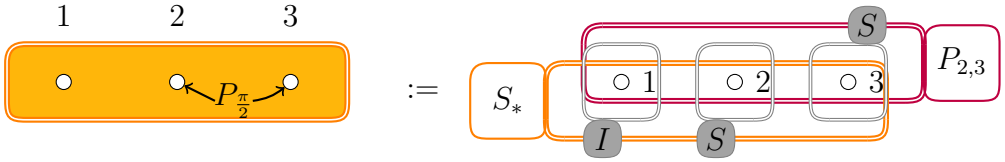


Figure A.16: A measurement pattern on the filled orange box notation. This measurement pattern implements a binary $\frac{\pi}{2}$ phase gadget on qubits 2 and 3. Qubits 1, 2, and 3 are both inputs and outputs.

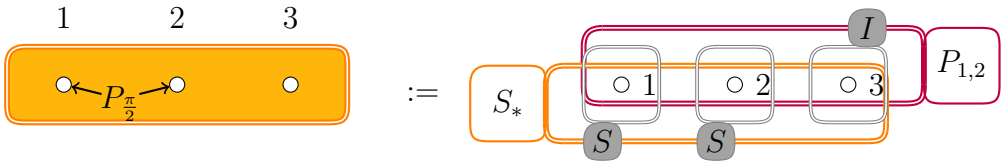


Figure A.17: A measurement pattern on the filled orange box notation. This measurement pattern implements a binary $\frac{\pi}{2}$ phase gadget on qubits 1 and 2. Qubits 1, 2, and 3 are both inputs and outputs.

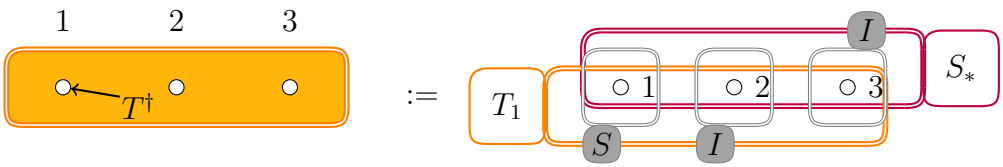


Figure A.18: A measurement pattern on the filled orange box notation. This measurement pattern implements a $T^\dagger$ gate qubits 1. Qubits 1, 2, and 3 are both inputs and outputs.

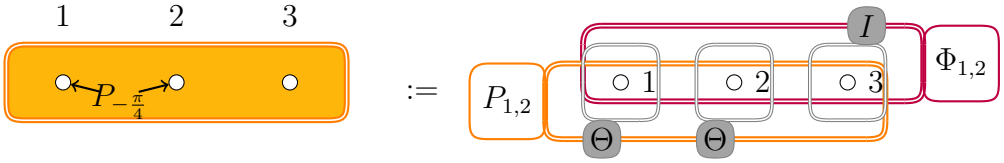


Figure A.19: A measurement pattern on the filled orange box notation. First, we perform on the orange box $P_{1,2}$ measurement pattern. If a phase gadget error is produced then $\Phi_{1,2} = S_*$ and $\Theta = S$; otherwise $\Phi_{1,2} = P_{1,2}$ and $\Theta = I$. Then we perform all the rest of measurement patterns. This measurement pattern implements a binary $-\frac{\pi}{4}$ phase gadget on qubits 1 and 2. Qubits 1, 2, and 3 are both inputs and outputs.

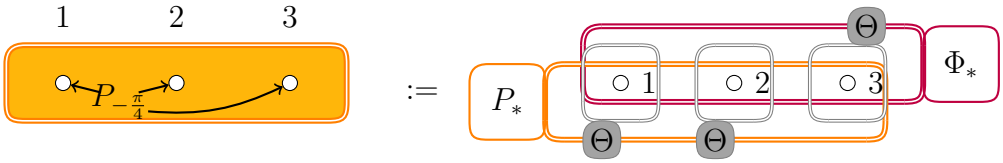


Figure A.20: A measurement pattern on the filled orange box notation. First, we perform on the orange box P_* measurement pattern. If a phase gadget error is produced then $\Phi_* = S_*$ and $\Theta = S$; otherwise $\Phi_* = P_*$ and $\Theta = I$. Then we perform all the rest of measurement patterns. This measurement pattern implements a trinary $-\frac{\pi}{4}$ phase gadget on qubits 1, 2, and 3. Qubits 1, 2, and 3 are both inputs and outputs.

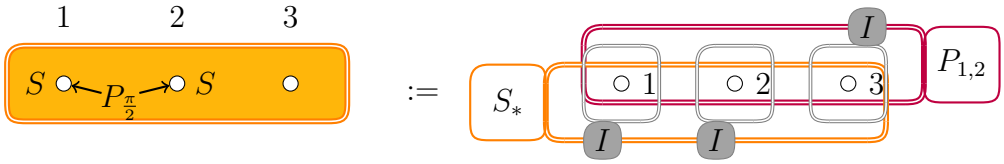


Figure A.21: A measurement pattern on the filled orange box notation. This measurement pattern implements a binary $\frac{\pi}{2}$ phase gadget on qubits 1 and 2, a S gate on qubit 1 and another S gate on qubit 2. Qubits 1, 2, and 3 are both inputs and outputs.

References

[1] Bob Coecke and Aleks Kissinger. *Picturing Quantum Processes: A First Course in Quantum Theory and Diagrammatic Reasoning*. Cambridge University Press, 2017.

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

[3] Robert Raussendorf and Hans J. Briegel. "A One-Way Quantum Computer". In: *Phys. Rev. Lett.* 86 (22 May 2001), pp. 5188–5191. URL: https://link.aps.org/doi/10.1103/PhysRevLett.86.5188.

[4] M Rossi et al. "Quantum hypergraph states". In: *New Journal of Physics* 15.11 (Nov. 2013), p. 113022. URL: http://dx.doi.org/10.1088/1367-2630/15/11/113022.

[5] Simon Anders and Hans J. Briegel. "Fast simulation of stabilizer circuits using a graph-state representation". In: *Physical Review A* 73.2 (Feb. 2006). URL: http://dx.doi.org/10.1103/PhysRevA.73.022334.

[6] Nathan Shettell and Damian Markham. "Graph States as a Resource for Quantum Metrology". In: *Physical Review Letters* 124.11 (Mar. 2020). URL: http://dx.doi.org/10.1103/PhysRevLett.124.110502.

[7] S R Clark, C Moura Alves, and D Jaksch. "Efficient generation of graph states for quantum computation". In: *New Journal of Physics* 7 (May 2005), pp. 124–124. URL: http://dx.doi.org/10.1088/1367-2630/7/1/124.

[8] Yuki Takeuchi, Tomoyuki Morimae, and Masahito Hayashi. *Quantum computational universality of hypergraph states with Pauli-X and Z basis measurements*. 2019. arXiv: 1809.07552 [quant-ph].

[9] Maarten Van den Nest et al. "Universal Resources for Measurement-Based Quantum Computation". In: *Phys. Rev. Lett.* 97 (15 Oct. 2006), p. 150504. URL: https://link.aps.org/doi/10.1103/PhysRevLett.97.150504.

[10] Aleks Kissinger and John van de Wetering. "Universal MBQC with generalised parity-phase interactions and Pauli measurements". In: *Quantum* 3 (Apr. 2019), p. 134. URL: http://dx.doi.org/10.22331/q-2019-04-26-134.

[11] Jacob Miller and Akimasa Miyake. "Hierarchy of universal entanglement in 2D measurement-based quantum computation". In: *npj Quantum Information* 2.1 (Nov. 2016). URL: http://dx.doi.org/10.1038/npjqi.2016.36.

[12] Mariami Gachechiladze, Otfried Gühne, and Akimasa Miyake. "Changing the circuit-depth complexity of measurement-based quantum computation with hypergraph states". In: *Physical Review A* 99.5 (May 2019). URL: http://dx.doi.org/10.1103/PhysRevA.99.052304.

[13] Scott Aaronson and Daniel Gottesman. "Improved simulation of stabilizer circuits". In: *Phys. Rev. A* 70 (5 Nov. 2004), p. 052328. URL: https://link.aps.org/doi/10.1103/PhysRevA.70.052328.

[14] John van de Wetering. *ZX-calculus for the working quantum computer scientist*. 2020. arXiv: 2012.13966 [quant-ph].

[15] Ross Duncan. *A graphical approach to measurement-based quantum computing*. 2012. arXiv: 1203.6242 [quant-ph].

[16] Miriam Backens and Aleks Kissinger. "ZH: A Complete Graphical Calculus for Quantum Computations Involving Classical Non-linearity". In: *Electronic Proceedings in Theoretical Computer Science* 287 (Jan. 2019), pp. 23–42. URL: http://dx.doi.org/10.4204/EPTCS.287.2.

[17] Stach Kuijpers, John van de Wetering, and Aleks Kissinger. *Graphical Fourier Theory and the Cost of Quantum Addition*. 2019. arXiv: 1904.07551 [quant-ph].

[18] Mateusz Piotr Kupper. "Analysis of quantum hypergraph states in the ZH-calculus". MA thesis. University of Edinburgh, 2019.

[19] Bob Coecke and Ross Duncan. "Interacting quantum observables: categorical algebra and diagrammatics". In: *New Journal of Physics* 13.4 (Apr. 2011), p. 043016. URL: http://dx.doi.org/10.1088/1367-2630/13/4/043016.

[20] Quanlong Wang. *An algebraic axiomatisation of ZX-calculus*. 2021. arXiv: 1911.06752 [quant-ph].

[21] Miriam Backens et al. *Completeness of the ZH-calculus*. 2021. arXiv: 2103.06610 [quant-ph].

[22] Brett Giles and Peter Selinger. "Exact synthesis of multiqubit Clifford+Tcircuits". In: *Physical Review A* 87.3 (Mar. 2013). URL: http://dx.doi.org/10.1103/PhysRevA.87.032332.

[23] D. Aharonov. "A Simple Proof that Toffoli and Hadamard are Quantum Universal". In: *arXiv: Quantum Physics* (2003).

[24] D Bacon et al. "Encoded universality in physical implementations of a quantum computer". In: *arXiv preprint quant-ph/0102140* (2001).

[25] Vincent Danos, Elham Kashefi, and Prakash Panangaden. *The Measurement Calculus*. 2007. arXiv: 0704.1263 [quant-ph].