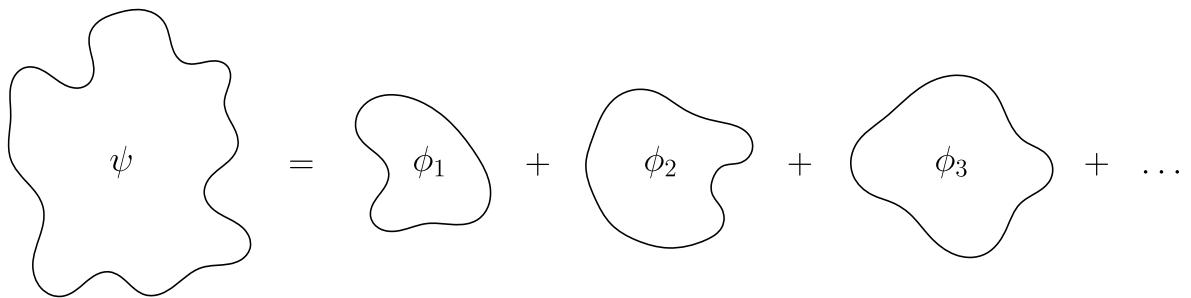


Graphical Stabilizer Decompositions for Multi-Control Toffoli Gate Dense Quantum Circuits

Maturitätsarbeit

Yves Vollmeier 

$$\psi = \phi_1 + \phi_2 + \phi_3 + \dots$$


Supervised by Dr. Riccardo Ferrario
Mathematisch Naturwissenschaftliches Gymnasium Rämibühl
January 6, 2025

Abstract

In this thesis, we study concepts in quantum computing using graphical languages, specifically using the ZX-calculus. The core of the research revolves around (graphical) stabilizer decompositions. The first major focus is on the decomposition of non-stabilizer states created from star edges. We discuss previous results and then present novel decompositions that yield a theoretical improvement. The second major focus is on weighting algorithms, applied to the special class of multi-control Toffoli gate dense quantum circuits. The representation of the corresponding gates is based on star edges. The applicability of known methods, such as CNOT-grouping, traditionally used for other classes, is examined in the context of this specific class. We then present a novel weighting algorithm that attempts to determine the best vertex to decompose. A refined version is implemented to simulate a known class of quantum querying algorithms, which is used to search for causal configurations of multiloop Feynman diagrams. For this case, as well as for a generalized benchmark consisting of randomly generated quantum circuits, we demonstrate occasional improvements in the final number of terms against traditional methods. These results are discussed by considering different simplification strategies. This thesis also provides a brief but broad outline of the important preliminaries.

Yves Vollmeier

Contents

Abstract	3
Contents	5
1 Introduction	1
1.1 Motivation	1
1.2 Structure of This Thesis	2
 PART I: PRELIMINARIES	 3
2 Quantum Computation	5
2.1 Postulates of Quantum Mechanics	6
2.2 Qubits, Quantum Gates and Quantum Circuits	11
2.3 Complexity and Classical Simulation	16
3 Graphical Languages	19
3.1 ZX-calculus	20
3.2 Related Calculi	23
3.3 Applications	25
 PART II: DECOMPOSITIONS OF NON-STABILIZER STATES	 27
4 Novel State Decompositions	29
4.1 Previous Work	29
4.2 This Work	33
 PART III: MULTI-CONTROL TOFFOLI GATE DENSE QUANTUM CIRCUITS	 39
5 Dynamic Decompositions	41
5.1 Overview	41
5.2 Comparison with State Decompositions	43
6 Weighting Algorithms	45
6.1 Basic Idea	45
6.2 Studied Approach	50
6.3 Application for Multiloop Feynman Diagrams	53
6.4 Importance of Simplification Strategies	59
 PART IV: CONCLUSION	 63
7 Conclusion	65
7.1 Summary	65
7.2 Outlook	65
Bibliography	67

1

Introduction

Yeah, but those things don't mean anything to me. [...] and I see in the Physics Review these idiot diagrams I cooked up [...]
— Richard Feynman

1.1 Motivation	1
1.2 Structure of This Thesis	2

1.1 Motivation

In 1966, Richard Feynman was asked about the awards he had received for his work. This provoked his annoyance, as he is often referred to as a humble physicist. We can only suspect that this annoyance was the reason for why he referred to *Feynman diagrams* as "idiot diagrams" [1]. The reality stands in stark contrast: Feynman diagrams are said to have revolutionized nearly every aspect of theoretical physics [2]. It was therefore a pleasant surprise when we found a graphical language that was reminiscent of these diagrams, and did not seem to require advanced knowledge about quantum field theory. We found out about this graphical language, which is referred to as the *ZX-calculus*, while learning about quantum mechanics and quantum computation and looking through research papers.

The initial goal set out for this **Maturitätsarbeit** was not only to learn about the ZX-calculus as a tool, but also to gain insight into the research field as a whole, as it was only introduced in 2008 [3]. This learning should be accompanied by active experimentation with the newly learned material.

The journey proved to be very fruitful and diverse, learning about many aspects of the ZX-calculus and related topics. We also got to experience scientific research in general. In order to create a common theme throughout this thesis and in order to remain brief, it was decided to only focus on so-called *stabilizer decompositions*. This was the topic we found particularly interesting and where we were able to find novel insights.

Stabilizer decompositions have first been introduced in 2012 [4] and have later become an important research direction within the ZX-calculus [5].

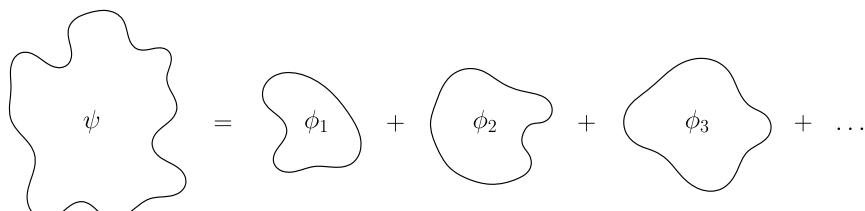


Figure 1.1: Simplified illustration of stabilizer decompositions.

The basic idea of stabilizer decompositions is illustrated in Figure 1.1. We have a mathematical object ψ that corresponds to a structure used in quantum computation, represented using the ZX-calculus. More precisely, it is a so-called *non-stabilizer state*, and has certain properties that make it difficult to simulate. The idea is now to *decompose* it into a sum of *stabilizer states* $\phi_1, \phi_2, \phi_3, \dots$, whose properties make their simulation easier.

1.2 Structure of This Thesis

This thesis consists of three parts:

- ▶ Part I: Preliminaries
- ▶ Part II: Decompositions of Non-Stabilizer States
- ▶ Part III: Multi-Control Toffoli Gate Dense Quantum Circuits

The first part aims to provide important preliminaries for the rest of this thesis, although notably, these will not be enough to get an in-depth understanding of all the concepts used throughout this thesis. Interested reader are therefore encouraged to consult the referenced sources for further study. Conversely, experienced readers may skip the first part and go directly to the second and third part.

Part I consists of Chapter 2, which introduces the basic ideas of *quantum mechanics*. This knowledge is then linked to *quantum computation*. Lastly, this chapter contains a brief overview of *computational complexity theory*. In Chapter 3, we give a general motivation for the use of *graphical languages*, as well as a basic introduction to the *ZX-calculus* and related calculi. Readers specifically interested in the applications of the *ZX-calculus* can find an extensive list of previous and current research topics compiled in Section 3.3.

Part II focuses on *state decompositions*. We start by reviewing previous work from the literature. Before showing our novel results, we also outline a few of our previous attempts.

Part III starts with Chapter 5, where we very briefly describe decompositions of *matrices* instead of *vectors*, that are furthermore *dynamic* in their size. Hence, we will refer to them as *dynamic decompositions*. These are then also compared to state decompositions in Section 5.2.

In Chapter 6, we build on these ideas, together with ideas from the literature, in order to create a *weighting algorithm* meant to decompose multi-control Toffoli gate dense quantum circuits with certain constraints as efficiently as possible. After very briefly explaining the *loop-tree duality formalism*, we demonstrate how our algorithm can be used to simulate a *quantum search algorithm*, which is used to find *causal configurations* of *multiloop Feynman diagrams*. Finally, the results are compared with the literature and discussed using further benchmarks.

The conclusion of this work features a summary of the research results and an outlook on future work.

PART I: PRELIMINARIES

2

Quantum Computation

God does not play dice

— Albert Einstein

Einstein, stop telling God what do!

— Niels Bohr

In 1927, perhaps one of the most iconic pictures in physics was taken (cf. Figure 2.1). Seventeen of the 29 attendees of the *fifth Solvay Conference on Physics* went down in history by winning a Nobel Prize. But not only that, many of them were integral to the development of **Quantum Mechanics** [7]. It is arguably one of the most successful, yet also one of the most mysterious scientific theories humanity has ever come up with [8].

In 1900, Max Planck postulated that electromagnetic energy is quantized, i.e. comes in discrete bundles of energy, which he called *quanta*¹. The energy of a *single* quanta could now be described by the equation

$$E = h\nu$$

where $h = 6.62610 \times 10^{-34}$ J s is a fit parameter called *Planck's constant*, and ν is the frequency of the electromagnetic wave. In 1905, based on Planck's work, Albert Einstein proposed the idea that light itself is made up of discrete quanta of energy². It is worth noting that these efforts originated from the *ultraviolet catastrophe*, an inconsistency between experimental data and *classical* theoretical predictions, which could be resolved using Planck's new ideas [9].

During the next two decades, there was an active back and forth between new discoveries³ and new theories⁴. This culmination of work ultimately led to what we now see as the origins of quantum mechanics [10].

Although this "first" period between 1900 and the 1920s provided profound insights into the natural world, it was only in the 1970s and 1980s that a shift in perspective allowed a greater understanding of quantum mechanics itself. This shift in perspective was led by pioneers such as Paul Benioff, Yuri Manin and Richard Feynman, who were now starting to think about *designed* systems, as opposed to *natural* systems. They realized that classical means of computation may not be optimal to study quantum mechanical systems and were thus asking questions like "What would a better suited machine look like?", "What is the space-time complexity of a certain quantum operation?", "How else can we exploit these properties?" and many more. This was the emergence of a new, highly interdisciplinary field of research, combining questions from physics, computer science, information theory and even engineering. It is what we now call *Quantum Information Theory* and *Quantum Computation* [8].

With these foundations in mind, Section 2.1 will focus on the concepts from the "first period", whereas Section 2.2 and Section 2.3 will focus on the concepts from the "second period". These three sections outline the information necessary for the rest of the thesis. They also aim to make the reader familiar with some key concepts from this fascinating field of research.

2.1 Postulates of Quantum Mechanics	6
2.2 Qubits, Quantum Gates and Quantum Circuits	11
2.3 Complexity and Classical Simulation	16

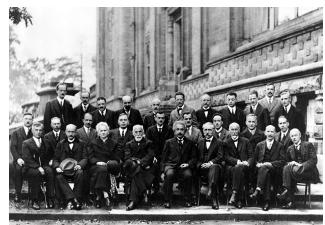


Figure 2.1: The Solvay Conference [6].

1: Hence the name *Quantum Mechanics*.

2: This ultimately led to the notion of *photons*.

3: Discovery of the atomic nucleus, compton scattering, electron diffraction, . . .

4: Postulation of wave-particle duality, electron spin, description of wave mechanics, matrix mechanics, . . .

2.1 Postulates of Quantum Mechanics

5: A famous example is that the speed of light c is constant. It is a key requirement to make special relativity work, whose predictions have been confirmed many times.

6: The definition of infinite-dimensional Hilbert spaces requires additional statements about limits and convergence.

Postulates are of great importance in physics⁵. They are requirements to make a certain theory work, even though they are not provable. Still, one may consider the validity of the resulting predictions [11].

Before starting with the *postulates of quantum mechanics*, it is important to realize that the mathematical tools to describe different quantum systems might vary. For example, in quantum computing, it suffices to use *finite-dimensional Hilbert spaces*, whereas the description of a hydrogen atom requires *infinite-dimensional Hilbert spaces*. For the purpose of this discussion, we will refrain from exploring the details of the latter⁶, only presenting a few basic use cases [12].

Definition 2.1.1 A *finite-dimensional Hilbert space* \mathcal{H} is a vector space over the complex numbers together with an *inner product*, that is, a mapping

$$\langle \cdot | \cdot \rangle : \mathcal{H} \times \mathcal{H} \rightarrow \mathbb{C}$$

such that the following conditions are satisfied:

1. *Linearity in the second argument*: $\forall \psi, \phi_1, \phi_2 \in \mathcal{H}, \forall \lambda_1, \lambda_2 \in \mathbb{C} :$

$$\langle \psi | \lambda_1 \phi_1 + \lambda_2 \phi_2 \rangle = \lambda_1 \langle \psi | \phi_1 \rangle + \lambda_2 \langle \psi | \phi_2 \rangle .$$

2. *Conjugate-symmetry*: $\forall \psi, \phi \in \mathcal{H} : \overline{\langle \phi | \psi \rangle} = \langle \psi | \phi \rangle .$

3. *Positive-definiteness*: $\forall \phi \in \mathcal{H} : \langle \psi | \psi \rangle \in \mathbb{R}, \langle \psi | \psi \rangle > 0 .$

Remark 2.1.2 By combining the first two conditions from Definition 2.1.1, we find that the inner product is *conjugate-linear in the first argument*, that means, $\forall \psi, \phi_1, \phi_2 \in \mathcal{H}, \forall \lambda_1, \lambda_2 \in \mathbb{C} :$

$$\langle \lambda_1 \psi_1 + \lambda_2 \psi_2 | \phi \rangle = \overline{\lambda_1} \langle \psi_1 | \phi \rangle + \overline{\lambda_2} \langle \psi_2 | \phi \rangle .$$

The notation used in Definition 2.1.1 is linked to the *Dirac notation*. Since it will commonly be used throughout this thesis, the following table should serve as a cheat sheet:

Dirac Notation Cheat Sheet [8]

7: The tensor product *in our context* is sometimes called Kronecker product.

z^*	Complex conjugate of the complex number z .
$ \psi\rangle$	Vector. Also known as a ket.
$\langle\psi $	Vector dual to $ \psi\rangle$. Also known as a bra.
$\langle\varphi \psi\rangle$	Inner product between the vectors $ \varphi\rangle$ and $ \psi\rangle$.
$ \varphi\rangle \otimes \psi\rangle$	Tensor product ⁷ of $ \varphi\rangle$ and $ \psi\rangle$.
$ \varphi\rangle \psi\rangle$	Abbreviated notation for tensor product of $ \varphi\rangle$ and $ \psi\rangle$.
A^*	Complex conjugate of the matrix A .
A^T	Transpose of the matrix A .
A^\dagger	Hermitian conjugate or adjoint of the matrix A , that is, $A^\dagger = (A^T)^*$.
$\langle\varphi A \psi\rangle$	Inner product between $ \varphi\rangle$ and $A \psi\rangle$.

Definition 2.1.3 A matrix U is said to be *unitary* if $U^\dagger U = I$, where I is the identity matrix.

The distinction between finite and infinite Hilbert spaces is typically accompanied by the distinction between *state vectors* and *wave functions*. In particular, given an n -dimensional Hilbert space \mathcal{H} equipped with an orthonormal⁸ basis of kets $\{|\psi_i\rangle\}$, we can express any element $|\psi\rangle$ of this Hilbert space as a linear combination (superposition) of the basis elements

$$|\psi\rangle = \sum_{i=1}^n c_i |\psi_i\rangle \quad (2.1)$$

with $c_i \in \mathbb{C}$. One refers to $|\psi\rangle$ as *state vector*. In the case of an infinite-dimensional Hilbert space, however, this sum becomes an integral

$$|\psi\rangle = \int dx \Psi(x) |x\rangle \quad (2.2)$$

where $\Psi(x)$ is referred to as *wave function*, essentially playing the same role as c_i in Eq. (2.1). Note that here we are using the position basis $\{|x\rangle\}$, hence the more precise term *position-space wave function*⁹ [14]. To be precise, $|x\rangle$ is an *improper vector*, whereas the $|\psi_i\rangle$ are *proper vectors*. The latter can be normalized to unity. The former, however, can only be normalized to the *Dirac delta function* [13]. *Heuristically*¹⁰, it is defined as

$$\delta(x) = \begin{cases} 0 & \text{if } x \neq 0 \\ \infty & \text{if } x = 0 \end{cases}$$

such that

$$\int_{-\infty}^{\infty} \delta(x) = 1.$$

As already mentioned, we have

$$\langle x' | x \rangle = \delta(x' - x)$$

and thus¹¹

$$\langle x' | \psi \rangle = \int dx \Psi(x) \langle x' | x \rangle = \Psi(x').$$

This is usually written more concisely as

$$\langle x | \psi \rangle = \Psi(x). \quad (2.3)$$

What will follow now are the four postulates of quantum mechanics as described in [8]. Since there are many different ways to formulate them, this resource was chosen as it focuses on quantum computation and thus uses the *state vector* representation.

8: *Orthonormal* means that the vectors are mutually orthogonal and are normalized.

9: We could also use the momentum basis, which would give us the *momentum-space wave function* [13].

10: In fact, there does not exist any function with these properties, which is why it can be seen as a "trick" used by physicists. However, there are rigorous definitions involving measure theory [15].

11: Again, even though this is the conventional way ([13]) to do it, these steps are not perfectly rigorous.

Postulate 1

Any isolated physical system is described by a *state space*, mathematically speaking a *Hilbert Space* \mathcal{H} , and the state of the system is completely described by its *state vector* $|\psi\rangle$, which we require to be of unit-length and therefore satisfy $\langle\psi|\psi\rangle = 1$.

Postulate 2

The *evolution* of the state in a closed quantum system is described by a *unitary transformation*. More precisely, the state $|\psi_1\rangle$ at time t_1 is related to the state $|\psi_2\rangle$ at time t_2 by a unitary operator U which depends only on the times t_1 and t_2 :

$$|\psi_2\rangle = U |\psi_1\rangle. \quad (2.4)$$

The case where only two specific times are considered can be recovered from the differential equation

$$i\hbar \frac{d|\psi(t)\rangle}{dt} = \hat{H}(t) |\psi(t)\rangle \quad (2.5)$$

where $\hbar = \frac{h}{2\pi}$ is the reduced Planck's constant and \hat{H} is the *Hamiltonian operator* of the system¹². This equation is called the *time-dependent Schrödinger equation*.

12: In essence, operators are functions over a space of physical states onto another state of states. One of them is the Hamiltonian operator, corresponding to the total energy of the system [16].

13: Note that there are different types of measurements, such as general measurements, projective measurements or POVMs [8].

Postulate 3

A quantum measurement is described by a collection M_m of *measurement operators* acting on the state space that is being measured, where m is a potential measurement outcome of the experiment¹³. The state right after a quantum measurement is given by

$$|\psi\rangle \xrightarrow{\text{measurement}} |\psi'\rangle = \frac{M_m |\psi\rangle}{\sqrt{p(m)}} = \frac{M_m |\psi\rangle}{\sqrt{\langle\psi|M_m^\dagger M_m|\psi\rangle}}$$

where $p(m)$ is the probability of getting the measurement outcome m . The measurement operators satisfy the *completeness equation*

$$\sum_m M_m^\dagger M_m = I$$

from which one can see that the total probability is indeed one $\sum_m p(m) = 1$.

Postulate 4

The state space of a composite physical system is formed by taking the tensor product of the state spaces from the individual subsystems. The state for the composite system is $|\psi_1\rangle \otimes |\psi_2\rangle \otimes |\psi_3\rangle \otimes \cdots \otimes |\psi_n\rangle$ if we number the systems 1 through n and $|\psi_i\rangle$ is the state for system i .

As already mentioned, most calculations, apart from those in quantum computation, involve the wave function representation. Using Eq. (2.3), we can rewrite Eq. (2.5) to get the *position-space Schrödinger equation*

$$i\hbar \frac{\partial \Psi(x, t)}{\partial t} = H(t)\Psi(x, t). \quad (2.6)$$

By making the Hamiltonian time-independent, $H(t) \sim E$, we get

$$i\hbar \frac{\partial \Psi(x, t)}{\partial t} = E\Psi(x, t). \quad (2.7)$$

We can now solve for $\Psi(x, t)$:

$$\begin{aligned} \frac{1}{\Psi(x, t)} \frac{\partial \Psi(x, t)}{\partial t} &= -\frac{iE}{\hbar} \\ \int \frac{1}{\Psi(x, t)} \partial \Psi(x, t) &= \int -\frac{iE}{\hbar} dt \\ \ln \Psi(x, t) &= -\frac{iEt}{\hbar} + \tilde{c} \\ \Psi(x, t) &= ce^{-iEt/\hbar}. \end{aligned}$$

Finally, since $\Psi(x, 0) = c$, we get

$$\Psi(x, t) = e^{-iEt/\hbar}\Psi(x, 0) \quad (2.8)$$

or, rewritten using Eq. (2.3),

$$|\psi(t)\rangle = e^{-iEt/\hbar} |\psi(0)\rangle. \quad (2.9)$$

This is the solution¹⁴ for the *time-dependent Schrödinger equation with constant Hamiltonian*. Plugging Eq. (2.8) back into Eq. (2.7), we get

$$\begin{aligned} i\hbar \frac{\partial}{\partial t} (e^{-iEt/\hbar} \Psi(x, 0)) &= He^{-iEt/\hbar} \Psi(x, 0) \\ i\hbar e^{-iEt/\hbar} \left(-\frac{iE}{\hbar}\right) \Psi(x, 0) &= He^{-iEt/\hbar} \Psi(x, 0) \\ E\Psi(x) &= H\Psi(x) \end{aligned} \quad (2.10)$$

14: Notice that this resembles Eq. (2.4).

The Hamiltonian H of such systems can be written as

$$H = -\frac{\hbar^2}{2m} \nabla^2 + V(x, y, z), \quad (2.11)$$

where $\nabla^2 = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} + \frac{\partial^2}{\partial z^2}$ is the Laplacian in Cartesian coordinates. Using Eq. (2.3), Eq. (2.10) becomes the *time-independent Schrödinger equation*

$$\hat{H} |\psi\rangle = E |\psi\rangle \quad (2.12)$$

and we can see that we have set up an *eigenvalue problem*.

We can now use this theory to study our first and last actual physical system. This is usually referred to as the "particle in a box" problem [17].

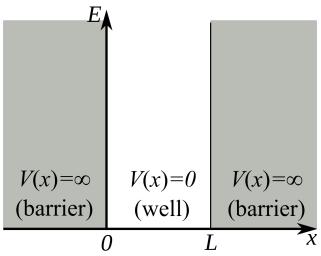


Figure 2.2: Visual representation of the "Particle in a box" problem [18].

Example 2.1.4 Consider a particle in the system represented in Figure 2.2. The system is one-dimensional, and the following applies to the potential of the particle $V(x)$:

$$V(x) = \begin{cases} 0 & \text{if } 0 < x < L \\ \infty & \text{if } x \leq 0, x \geq L. \end{cases} \quad (2.13)$$

Since a particle cannot have infinite potential energy, it must stay in the well given by the box. This knowledge allows us to combine Eq. (2.10), Eq. (2.11) and Eq. (2.13) into

$$-\frac{\hbar^2}{2m} \frac{d^2\Psi(x)}{dx^2} = E\Psi(x).$$

We can rearrange this to get

$$\frac{d^2\Psi(x)}{dx^2} = -\frac{2mE}{\hbar^2}\Psi(x). \quad (2.14)$$

The roots of the characteristic polynomial $\lambda^2 + \frac{2mE}{\hbar^2}$ are $\lambda_{1,2} = \pm ik$, with $k := \sqrt{\frac{2mE}{\hbar^2}}$. Thus, the general solution of Eq. (2.14) is

$$\Psi(x) = c_1 \cos(kx) + c_2 \sin(kx). \quad (2.15)$$

We can now consider our boundary conditions

$$\begin{aligned} \Psi(x \leq 0) = 0 &\implies \Psi(0) = 0 \\ \Psi(x \geq L) = 0 &\implies \Psi(L) = 0. \end{aligned}$$

By plugging these values into Eq. (2.15), we obtain

$$c_1 = 0$$

and

$$c_2 \sin(kL) = 0 \Leftrightarrow kL = n\pi, \quad n \in \mathbb{Z}.$$

With this knowledge, we can find c_2 and E . To find c_2 , we need to use the *normalization condition*

$$\int_{-\infty}^{\infty} |\Psi(x)|^2 dx = 1, \quad (2.16)$$

which in our case becomes

$$\int_0^L (c_2)^2 \sin^2(kx) dx = 1.$$

Evaluating the definite integral yields

$$-\frac{(c_2)^2 \sin(2kL)}{4k} + \frac{(c_2)^2 L}{2} = 1 \Leftrightarrow c_2 = \sqrt{\frac{2}{L}}.$$

Thus, we get our final equation

$$\Psi_n(x) = \sqrt{\frac{2}{L}} \sin\left(\frac{n\pi}{L}x\right). \quad (2.17)$$

To find the energy eigenvalues, we simply need to compare $k = \sqrt{\frac{2mE}{\hbar^2}}$ with $kL = n\pi$ to get

$$E_n = \frac{n^2\pi^2\hbar^2}{2mL^2} = \frac{\hbar^2n^2}{8mL^2}, \quad (2.18)$$

where $n = 1, 2, 3, \dots$. This is because $\Psi_0(x) = 0$ cannot be normalized and furthermore because any $n < 0$ yields the same quantum state as its positive equivalent. These results are visualized in Figure 2.3.

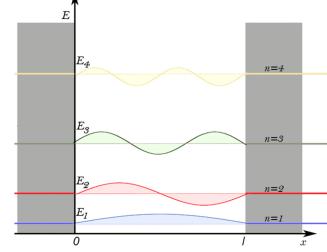


Figure 2.3: Visual representation of the different standing waves and corresponding quantized energy levels [19].

2.2 Qubits, Quantum Gates and Quantum Circuits

In 1982, the renowned physicist Richard Feynman published a paper about *quantum computers*.

Can physics be simulated by a universal computer? [...] the physical world is quantum mechanical, and therefore the proper problem is the simulation of quantum physics [...] Can you do it with a new kind of computer—a quantum computer?
— Richard Feynman [20]

The problem mentioned here is quite simple to understand. Imagine you want to study the molecular structure of drugs. For that purpose, one may consider the electron spin¹⁵. One electron may just be spin-up or spin-down, so there are two states to be considered. Looking at two electrons, we have four possible states. With ten electrons, there are already 1024 states. Fifty electrons are enough to have more than 1 quadrillion¹⁶. However, this is only the case when working with a *classical computer* that utilizes *bits* of information. We will later see that *qubits* — something inherently quantum mechanical — could resolve this problem.

To explain the basic ideas behind quantum computers, we will consider an analogy to classical computers. The reader is therefore assumed to have sufficient knowledge of classical computers.

15: In reality, of course, the task of studying such systems is much more intricate.

16: A one followed by fifteen zeros!

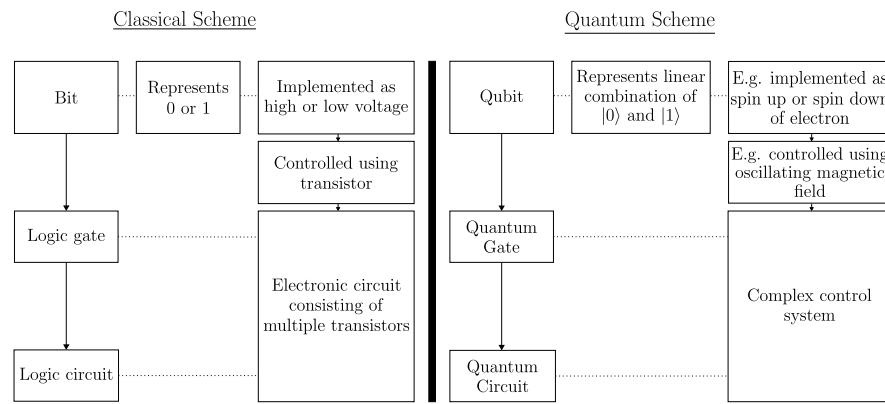


Figure 2.4: An analogy between classical computers and quantum computers.

The left sides of the two respective schemes in Figure 2.4, classical and quantum, represent the mathematical and information theoretic concepts, whereas the

17: Qubits are prone to *quantum decoherence*, a process which results in the loss of information due to loose energetic couplings. This is why one very often sees big cryostats cooling qubits, so that there is less thermal noise and thus less couplings.

right sides represent their physical implementation. Notice that it is a simplified description of how those systems actually work. For example, many types of quantum computers also have a classical part, which handles tasks like error-correction¹⁷ [21].

Definition 2.2.1 A *qubit* is an abstract representation of a single bit of information, represented as a state vector $|\psi\rangle$ in a two-dimensional Hilbert space

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle, \quad \alpha, \beta \in \mathbb{C}, \quad (2.19)$$

where the vectors $|0\rangle := \begin{pmatrix} 1 \\ 0 \end{pmatrix}$ and $|1\rangle := \begin{pmatrix} 0 \\ 1 \end{pmatrix}$ form an orthonormal basis and are thus called *computational basis states*.

We can now use the third postulate of quantum mechanics to study how a measurement on a qubit may look like. For reasons mentioned later on, the type of measurement we are going to use is called projective measurement. Let us define our measurement operators:

$$\begin{aligned} M_0 &= |0\rangle\langle 0| = \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix} \\ M_1 &= |1\rangle\langle 1| = \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix} \end{aligned}$$

We can thus compute the measurement probabilities

$$p(0) = \langle\psi|M_0^\dagger M_0|\psi\rangle = \alpha^* \alpha \langle 0|0 \rangle = |\alpha|^2 \quad (2.20)$$

$$p(1) = \langle\psi|M_1^\dagger M_1|\psi\rangle = \beta^* \beta \langle 1|1 \rangle = |\beta|^2 \quad (2.21)$$

and the state after the measurement

$$\frac{M_0|\psi\rangle}{\sqrt{p(0)}} = \frac{\alpha}{|\alpha|}|0\rangle \quad (2.22)$$

$$\frac{M_1|\psi\rangle}{\sqrt{p(1)}} = \frac{\beta}{|\beta|}|1\rangle. \quad (2.23)$$

From Eq. (2.20) and Eq. (2.21) we can deduce the *Born rule*

$$|\alpha|^2 + |\beta|^2 = 1. \quad (2.24)$$

The absolute values of $\alpha, \beta \in \mathbb{C}$ can be interpreted as real coordinates of points on the unit circle:

$$|\alpha| = \cos(\omega), \quad |\beta| = \sin(\omega)$$

for $\omega \in [0, 2\pi]$. Therefore, α and β can be written down in polar coordinates

$$\alpha = e^{i\phi_1} \cos(\omega), \quad \beta = e^{i\phi_2} \sin(\omega)$$

for $\phi_1, \phi_2 \in [0, 2\pi]$. Notice that Eq. (2.24) is preserved, since $|e^{i\phi_1}| = |e^{i\phi_2}| = 1$.

We now have

$$\begin{aligned} |\psi\rangle &= \alpha|0\rangle + \beta|1\rangle \\ &= e^{i\phi_1} \cos(\omega)|0\rangle + e^{i\phi_2} \sin(\omega)|1\rangle \\ &= e^{i\phi_1} (\cos(\omega)|0\rangle + e^{i(\phi_2-\phi_1)} \sin(\omega)|1\rangle), \end{aligned}$$

where $e^{i\phi_1}$ is the *global phase*, which can be omitted since it influences the probabilities of $|0\rangle$ and $|1\rangle$ the same way, and thus cannot be distinguished by measurements. For reasons that will become clear in a bit, we substitute $\omega := \frac{\theta}{2}$ and $\varphi := \phi_2 - \phi_1$. We thus obtain our final equation

$$|\psi\rangle = \cos\left(\frac{\theta}{2}\right)|0\rangle + e^{i\varphi} \sin\left(\frac{\theta}{2}\right)|1\rangle, \quad (2.25)$$

for $\theta \in [0, \pi]$, $\varphi \in [0, 2\pi)$. We call φ the *phase of the qubit*. This result can be visualized using the *Bloch sphere* (cf. Figure 2.5).

One can now see that plugging in $\theta = \pi$ produces a vector pointing straight down to $|1\rangle$ according to the visualization and $|1\rangle$ according to our formula too. This is why we substituted $\omega = \frac{\theta}{2}$.

Measuring the qubit state $|\psi\rangle$ projects it onto the Z-axis (either to $|0\rangle$ or $|1\rangle$). This is why the procedure is called a projective measurement.

We will demonstrate how Postulate 4 applies by considering a two-qubit systems:

$$|\psi\rangle = \alpha_{00}|00\rangle + \alpha_{01}|01\rangle + \alpha_{10}|10\rangle + \alpha_{11}|11\rangle.$$

Intuitively, there must be a normalization condition

$$\sum_{x \in \{0,1\}^2} |\alpha_x|^2 = 1, \quad (2.26)$$

where the notation $\{0,1\}^2$ means "the set of strings of length two with each letter being either zero or one" [8]. When measuring the first qubit alone, this will yield 0 with probability $p(m)$ and leave the post-measurement state

$$|\psi'\rangle = \frac{\alpha_{00}|00\rangle + \alpha_{01}|01\rangle}{\sqrt{p(m)}} = \frac{\alpha_{00}|00\rangle + \alpha_{01}|01\rangle}{\sqrt{|\alpha_{00}|^2 + |\alpha_{01}|^2}}.$$

Something very interesting happens when our two-qubit system is a so-called *Bell state* or *EPR Pair*, which is given by

$$\frac{|00\rangle + |11\rangle}{\sqrt{2}}. \quad (2.27)$$

Measuring the first qubit yields 0 with probability 0.5, leaving the post-measurement state $|00\rangle$. The probability for 1 is 0.5 and the post-measurement state is $|11\rangle$. However, notice what happens when measuring the second qubit of the post-measurement state: One will get the same value as in the first measurement. We thus say they are *correlated*. This phenomenon is called *quantum entanglement* [8].

In order to manipulate the state of our qubit(s), we can use *quantum gates*. There are *single-qubit gates* and *n-qubit gates*. Both are *unitary transformations*.

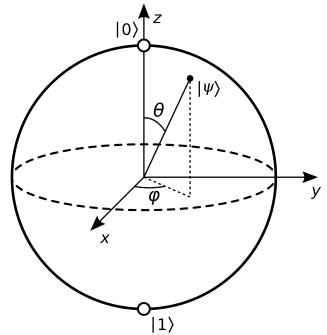


Figure 2.5: Bloch sphere [22].

18: If not mentioned otherwise, we always represent them in the computational basis.



Figure 2.6: Quantum circuit notation for NOT gate [8].

One of the simplest single-qubit gates is the *NOT gate*, *Pauli-X gate* or simply *X gate*, whose matrix representation¹⁸ is

$$X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}. \quad (2.28)$$

The corresponding *quantum circuit notation* is depicted in Figure 2.6.

Like the logical NOT gate, it "flips" the state:

$$X |0\rangle = |1\rangle \quad \text{and} \quad X |1\rangle = |0\rangle.$$

Another very important single-qubit gate is the *Hadamard gate*

$$H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}, \quad (2.29)$$

which is depicted in Figure 2.7. One can easily see how useful it is to prepare states in a superposition:



Figure 2.7: Quantum circuit notation for Hadamard gate [8].

In fact, it is possible to visualize all single-qubit gate transformations in the Bloch sphere according to Theorem 2.2.2.

Theorem 2.2.2 Any 2×2 unitary matrix can be decomposed as

$$U = e^{i\alpha} \begin{bmatrix} e^{-i\beta/2} & 0 \\ 0 & e^{i\beta/2} \end{bmatrix} \begin{bmatrix} \cos(\frac{\gamma}{2}) & -\sin(\frac{\gamma}{2}) \\ \sin(\frac{\gamma}{2}) & \cos(\frac{\gamma}{2}) \end{bmatrix} \begin{bmatrix} e^{-i\delta/2} & 0 \\ 0 & e^{i\delta/2} \end{bmatrix}, \quad (2.30)$$

where α, β, γ and δ are real-valued. The second matrix is an ordinary rotation, and the other two matrices are rotations in different planes. [8]

19: To be more precise, any *pure* state. There are also *mixed* states, but they are of no importance to us [8].



Figure 2.8: Quantum circuit notation for CNOT gate, where the filled black circle represents the control bit and the bottom one represents the target bit [8].

This should make sense intuitively, since any state¹⁹ can be represented by a vector pointing onto the surface of the Bloch sphere, and because any allowed transformation again creates such a vector, we know that they can be transformed into each other using rotations according to basic geometry.

One of the simplest n-qubit gates is the *CNOT gate* or *controlled NOT gate* (cf. Figure 2.8). As the name suggests, there is a *control bit* and a *target bit*. If the state for the control bit is $|0\rangle$, nothing happens. If it is $|1\rangle$, then a NOT gate is applied on the target bit. The corresponding matrix representation is

$$|0\rangle\langle 0| \otimes I + |1\rangle\langle 1| \otimes X = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}. \quad (2.31)$$

Notice that we could also let the control bit be below the target bit, which would result in

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}.$$

As we will see later, there could also be multiple qubits in-between the control and the target bit.

Another very important gate is the *Toffoli gate* or *CCNOT gate* (cf. Figure 2.9). It is a subcase of the *multi-control Toffoli gate*, where the number of control bits was set to two. The matrix representation is

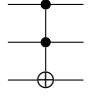
$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}. \quad (2.32)$$


Figure 2.9: Quantum circuit notation for Toffoli gate [8].

For the sake of completeness, we will now briefly list the rest of the most important quantum gates:

Common quantum gates		
Pauli-Y		$\begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix}$
Pauli-Z		$\begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$
S-gate / phase		$\begin{bmatrix} 1 & 0 \\ 0 & i \end{bmatrix}$
T-gate / $\frac{\pi}{8}$		$\begin{bmatrix} 1 & 0 \\ 0 & e^{i\pi/4} \end{bmatrix}$
SWAP		$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$
Controlled-Z		$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \end{bmatrix}$
Controlled-S		$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & i \end{bmatrix}$
Fredkin / Controlled-SWAP		$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$

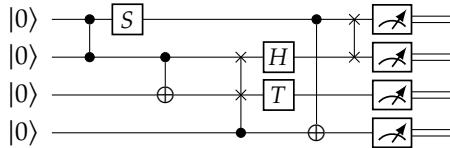
20: We use the following notation:

$$M_1 \otimes M_2 \otimes \dots \otimes M_n =: \bigotimes_{k=1}^n M_k$$

and

$$\bigotimes_{k=1}^n M =: M^{\otimes n}$$

Figure 2.10: An example of a whole quantum circuit.



Horizontal composition of gates represents the matrix product of the corresponding matrices, so a Hadamard gate on the left of a NOT gate transforms the input state $|\psi\rangle$ according to $XH|\psi\rangle$, in other words the Hadamard gate is applied first. Vertical composition of gates represents the tensor product of the corresponding matrices, so a Hadamard gate positioned above a NOT gate transforms the input state $|\psi \otimes \phi\rangle$ according to $H|\psi\rangle \otimes X|\phi\rangle$. Notice that we order ψ and ϕ from left to right, corresponding to top to bottom in the circuit. Finally, on the right-most side of the circuit, there might be symbols like in Figure 2.11. These represent quantum measurements, which result in classical bits, and are hence connected to classical wires, denoted as *double wires*.



Figure 2.11: Measurement symbol in a quantum circuit.

21: This means it may grow faster than any polynomial, but is still significantly smaller than an exponential [23].

2.3 Complexity and Classical Simulation

In 1994, Peter Shor published his work on an algorithm that is today known as *Shor's algorithm*. It is a *quantum algorithm*, meaning the computation is done on a quantum computer. The goal is to factor an integer N . The general, most-efficient classical algorithm, the *general number field sieve*, has a sub-exponential time-complexity²¹. Shor's algorithm on the other hand is polynomial in $\log(N)$. The key part here is that we do not know whether there exists a classical algorithm that could match or beat Shor's algorithm. There is strong evidence that the quantum algorithm is actually better, however, we do not know for certain [24]. Even more so, we *do not know for certain* whether quantum computers actually pose any advantage [8].

The field concerned with the difficulty of classical and quantum computational problems is called *computational complexity theory*. The most basic concept is the *complexity class*, a set of computational problems that require similar computational resources to be solved. **P** and **NP** are the most important classes. **P** is the set of all problems that can be solved quickly on a classical computer. **NP** is the set of all problems whose solutions can be quickly checked on a classical computer. As an example, consider the problem of factoring an integer N . As previously discussed, we think that there exists no algorithm capable of solving the problem quickly on a classical computer, and we thus think it is not in **P**. However, given a potential solution consisting of p_1, \dots, p_n , we can simply multiply them to see if the result equals N . Thus, it definitely is in **NP**. It is clear that **P** is a subset of **NP**. What remains one of the greatest unsolved problems in theoretical computer science is whether these two classes are different, in other words

$$\stackrel{?}{P} \neq NP. \quad (2.33)$$

Although we know that all problems in **P** can be solved efficiently using a quantum computer, we again do not know whether that is true for **NP** as well [8].

While such grand questions remain unsolved, smaller yet still very important statements have been proven. One of them is the **Gottesman-Knill theorem** [25]:

Theorem 2.3.1 Any quantum computer performing only: a) Clifford group gates, b) measurements of Pauli group operators, and c) Clifford group operations conditioned on classical bits, which may be the results of earlier measurements, can be perfectly simulated in polynomial time on a probabilistic classical computer.

In the next chapter, after having introduced new tools, we will consider this theorem in more detail. For now, it should suffice to say that there are **Clifford circuits** and **non-Clifford circuits**, where according to the theorem, it is possible to simulate the former efficiently on a classical computer²².

In order to see why one might be interested in **classical simulation**, we need to take a quick detour to experimental physics. Unsurprisingly, the actual physical implementation of quantum computers poses yet another big challenge. For instance, we do not know which approach²³ will "win the race". Each approach has certain advantages but also disadvantages, which are most often related to the number of qubits, the quality of the qubits and the circuit execution speed [26]. One of the problems that has to be solved is how to actually test whether the results a newly built quantum computer produces are correct. This can be done by simulating a quantum computer on a classical computer. Such a **quantum circuit simulator** might also be used to test or even come up with quantum algorithms [27].

Classical simulation can be split up into two domains [12]:

1. **Weak simulation** is concerned with sampling from the probability distribution that one would get when actually running the quantum circuit.
2. **Strong simulation** is concerned with calculating the actual probability of observing a certain measurement outcome.

Additionally, there exists a variety of different simulation methods based on ideas such as state vectors, density-matrices, matrix product states or tensor networks. We will focus on another approach based on *stabilizer decompositions*, which we will discuss in Chapter 4 and subsequent chapters. In Section 6.3 we will use a state vector approach implemented in Qiskit [28] in order to check our own simulation results. Interested readers may refer to [27] for a more detailed discussion of the various simulation methods.

22: And we think that the latter cannot be simulated efficiently on a classical computer, but they can be executed efficiently on a quantum computer.

23: Superconductors, ions, cold atoms, silicon dots, topological materials, photonic crystals, nitrogen vacancies, nuclear magnetic resonance etc.

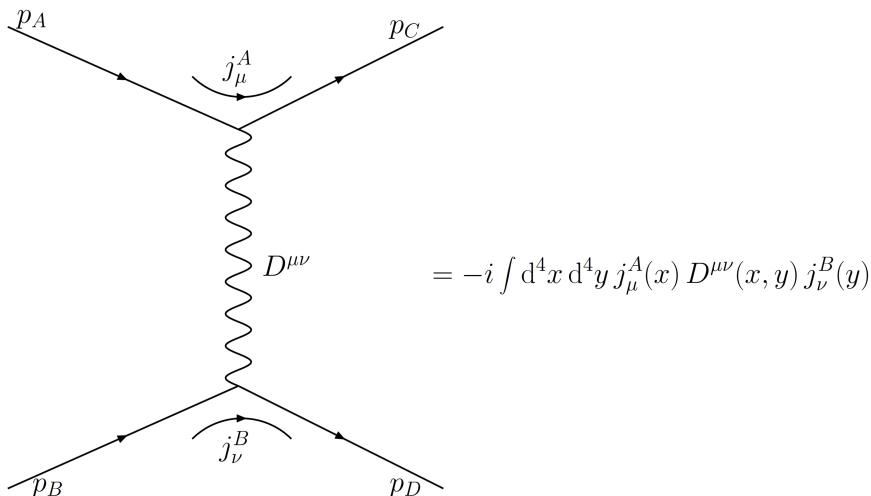
3

Graphical Languages

A diagram to help write down the mathematical expressions.

— Richard Feynman

In 1975, perhaps one of the most iconic cars, at least for a physicist, was bought (*cf.* Figure 3.1). The well-sighted reader might have spotted the rather peculiar drawings on the side of the van. Unsurprisingly, the owner of the van, Richard Feynman, came up with these types of diagrams himself. The first *Feynman diagram*, which can be seen in Figure 3.2, was published in 1949 in [30].



Since then, Feynman diagrams have become an essential part in the toolkit of physicists studying *Quantum Field Theory* (QFT). In brief, QFT combines classical field theory, special relativity and quantum mechanics into a powerful theoretical framework which is commonly used in particle physics and condensed matter physics. The current standard model of particle physics is based on QFT [35]. Feynman diagrams cannot only help to visualize formulas, they can also help to *come up* with formulas. In fact, there are purely graphical rules that define how a diagram may be rewritten, without having to deal with the underlying equations. Figure 3.3 shows a Feynman diagram and the according formula that represents a scattering process, for example electron-muan scattering, where particle A scatters into particle B. More precisely, both the diagram and the formula represent the *scattering amplitudes*. The details, however, lie outside the scope of this thesis [31].

The importance of *graphical languages* is apparent: Feynman diagrams simplify difficult QFT calculations, Penrose diagrams (*cf.* Figure 3.4) help represent causal relations in spacetime [36], logic circuits (*cf.* Figure 3.5) give a simple representation of complex compositions of Boolean functions [33] and tensor diagram notation (*cf.* Figure 3.6) allows for simple graphical manipulations of tensors [34].

Nevertheless, it is important to keep in mind that graphical languages are simply a tool, the same way a drill is great for screws, but rather bad for nails. You could use it to hammer in a nail, but there are better tools like a hammer. This holds true for any tool a mathematician or scientist might use.

3.1 ZX-calculus	20
3.2 Related Calculi	23
3.3 Applications	25



Figure 3.1: Iconic Dodge Tradesman Maxivan [29].

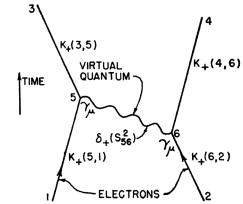


Figure 3.2: First published Feynman diagram [30].

Figure 3.3: Equivalence between Feynman diagram and formula [31].

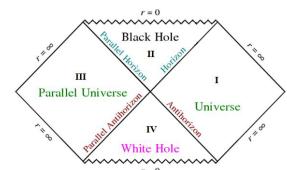


Figure 3.4: Example of a Penrose diagram [32].

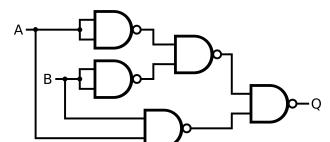


Figure 3.5: Example of a logic circuit [33].

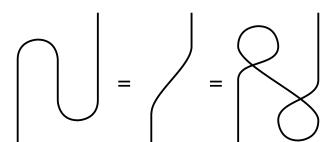
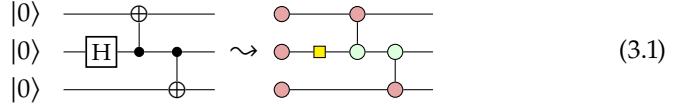


Figure 3.6: Example of tensor diagram notation [34].

3.1 ZX-calculus

1: In fact, a ZX-diagram is a *tensor network* [12].

A *ZX-diagram* is a graphical representation of a linear map between qubits. It is similar to the tensor diagram notation presented in the last section¹. Moreover, it is linked to the quantum circuit notation. As will soon be evident, we can convert *any* quantum circuit into a ZX-diagram:

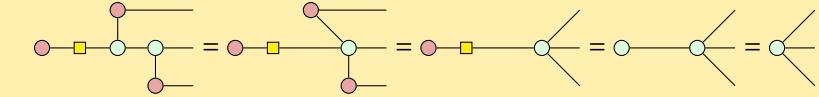


The *ZX-calculus* is a graphical language consisting of ZX-diagrams and *rewrite rules* that allow one to manipulate such diagrams [37]. It was introduced in 2008 by Coecke and Duncan [3].

Example 3.1.1 It is possible to diagrammatically prove that the circuit from Eq. (3.1) creates a so-called *GHZ state*

$$\frac{|000\rangle + |111\rangle}{\sqrt{2}} \quad (3.2)$$

using the rewrite rules²:



Readers interested in more such examples or a more detailed introduction to the ZX-calculus than the one to follow, may refer to [37], where Example 3.1.1 is taken from, or, alternatively, to [12].

In the remainder of this section, we will more formally introduce the ZX-calculus³. Section 3.2 will introduce certain variations of the ZX-calculus and finally, Section 3.3 will explore previous and current research areas by reviewing state-of-the-art applications.

3: If not noted otherwise, the primary source of information for the following is [12].

4: The process is analogous to how we treated the matrices corresponding to the gates in quantum circuits, in other words, matrix product for horizontal composition and tensor product for vertical composition.

ZX-diagrams are constructed by taking matrix products or tensor products⁴ of the following linear maps, so-called *generators*:

► *Z-spider*:

$$Z_m^n[\alpha] = m \left\langle \begin{array}{c} \diagup \\ \textcolor{green}{\circlearrowleft} \alpha \\ \diagdown \end{array} \right| n = |0\rangle^{\otimes n} \langle 0|^{\otimes m} + e^{i\alpha} |1\rangle^{\otimes n} \langle 1|^{\otimes m} \quad (3.3)$$

► *X-spider*:

$$X_m^n[\alpha] = m \left\langle \begin{array}{c} \diagup \\ \textcolor{red}{\circlearrowright} \alpha \\ \diagdown \end{array} \right| n = |+\rangle^{\otimes n} \langle +|^{\otimes m} + e^{i\alpha} |-\rangle^{\otimes n} \langle -|^{\otimes m} \quad (3.4)$$

Remark 3.1.2 These generators form two families of linear maps that are given by

$$\{Z_m^n[\alpha] : (\mathbb{C}^2)^{\otimes m} \rightarrow (\mathbb{C}^2)^{\otimes n} \mid m, n \in \mathbb{N}, \alpha \in [0, 2\pi)\}$$

and

$$\{X_m^n[\alpha] : (\mathbb{C}^2)^{\otimes m} \rightarrow (\mathbb{C}^2)^{\otimes n} \mid m, n \in \mathbb{N}, \alpha \in [0, 2\pi)\}.$$

Because of how they look, these generators are called ‘spiders’. Hence, the black lines are also referred to as *legs*. Alternatively, we can see ZX-diagrams as graphs, and thus call the spiders *vertices/nodes* and the legs *edges*. In analogy to quantum circuits, the edges might also be called *wires*. Since time flows from left to right, wires on the left side are *inputs* and wires on the right side are *outputs*. A ZX-diagram with no inputs and no outputs is referred to as *scalar diagram*. The number α in Eq. (3.3) and Eq. (3.4) is the *phase* of the spider. Finally, $n + m$ is the *arity* or (*vertex*) *degree* of the spider.

Consider a Z-spider with one input edge, one output edge and a phase of zero:



Per convention, a zero-phase is not written at all:

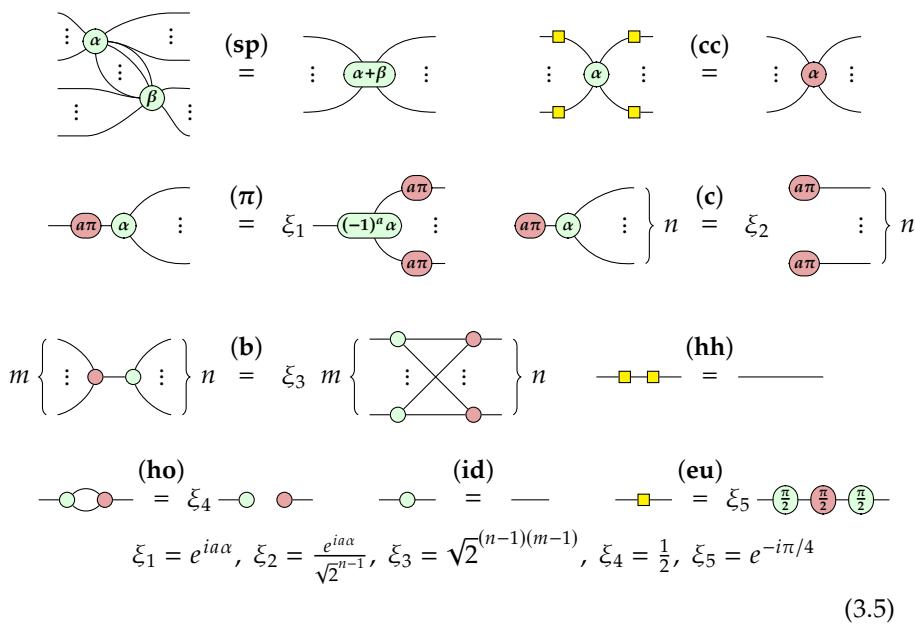


We can calculate the corresponding matrix

$$\begin{aligned}
 |0\rangle^{\otimes 1} \langle 0|^{\otimes 1} + e^{i0} |1\rangle^{\otimes 1} \langle 1|^{\otimes 1} &= |0\rangle \langle 0| + |1\rangle \langle 1| \\
 &= \begin{pmatrix} 1 \\ 0 \end{pmatrix} \begin{pmatrix} 1 & 0 \end{pmatrix} + \begin{pmatrix} 0 \\ 1 \end{pmatrix} \begin{pmatrix} 0 & 1 \end{pmatrix} \\
 &= \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix} + \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix} \\
 &= \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}
 \end{aligned}$$

which happens to be the identity matrix. We have thus derived one of the *nine* rewrite rules⁵ [12] [38] [39] [40]:

5: The remaining proofs are outside the scope of this thesis.



(sp)	spider fusion	(cc)	color change
(π)	π -commutation rule	(c)	state copy
(b)	bialgebra	(hh)	hadamard-hadamard cancellation
(ho)	hopf	(id)	identity
(eu)	euler decomposition of Hadamard		

It is worth mentioning that this set of rewrite rules is not minimal. In other words, there exists a smaller set of rewrite rules, with which one could derive (c) for instance. These ‘obsolete’ rules are still listed, as we will frequently encounter them throughout this thesis.

Alongside these rewrite rules, there exist two *meta rules*:

- ▶ All rewrite rules hold true when swapping red and green
- ▶ Only connectivity matters

The former should be clear. As for the latter, it simply means that one is allowed to ‘bend’ wires in any way.

For practical purposes, a new generator, the *Hadamard box*, might be introduced, even though it can be represented using more fundamental generators (*cf.* (eu)). Instead of using a yellow box, we sometimes use the equivalent *Hadamard edge* for notational convenience⁶:

$$\text{---} \square \text{---} \rightsquigarrow \text{---} \text{---} \quad (3.6)$$

6: It is especially used in programmatic implementations.

Remark 3.1.3 A scalar ξ_i in Eq. (3.5) is multiplied by the matrix representation of the diagram next to it. Since many applications in literature are not concerned with the global scalar of a diagram, one usually writes [41]:

$$\sqrt{2} \text{ ---} (\pi) \text{---} \approx \text{---} (\pi) \text{---}$$

In the *scalar-free ZX-calculus*⁷, one might even write [42]:

$$\sqrt{2} \text{ ---} (\pi) \text{---} = \text{---} (\pi) \text{---}$$

7: This is the first small example of a variation of the ZX-calculus (*cf.* Section 3.2).

8: The first two terms are used in [43]. The term ‘StabZX’ is used in [44], which for the sake of clarity we shall call *stabilizer-fragment* instead. The definitions refer to the same.

Definition 3.1.4 The *Clifford-fragment*, $\frac{\pi}{2}$ -*fragment* or *stabilizer-fragment*⁸ restricts the ZX-calculus by requiring the phase of spiders to be $\alpha = k\frac{\pi}{2}$, $k \in \mathbb{Z}$.

Definition 3.1.5 The *Clifford+T-fragment* or $\frac{\pi}{4}$ -*fragment* restricts the ZX-calculus by requiring the phase of spiders to be $\alpha = k\frac{\pi}{4}$, $k \in \mathbb{Z}$.

Remark 3.1.6 Accordingly, we can talk about *Clifford ZX-diagrams/stabilizer ZX-diagrams* and *non-Clifford ZX-diagrams/non-stabilizer ZX-diagrams* [12][44].

Definition 3.1.4 is of interest, since it can be shown that the *simplification routines* required to evaluate a Clifford ZX-diagram constructed from a Clifford circuit can be run in polynomial time [44]. As demonstrated in [37], one can even prove the Gottesman-Knill theorem using the ZX-calculus.

On the other hand, non-Clifford diagrams resulting from Definition 3.1.5 can be used to show *universality* of the ZX-calculus. In fact, this is one of the three main questions to ask when evaluating the sensibility of such a calculus with respect to its usefulness for reasoning about quantum systems [45]:

Is the ZX-calculus universal? Can ZX-diagrams be used to represent any linear map between finite-dimensional Hilbert spaces? The answer is yes. In the following, we will outline a short proof.

Proof. As a first step, we need to show that any scalar can be represented as a ZX-diagram. The following identities hold true:

$$\begin{array}{lll} \textcircled{1} = 2 & \textcircled{2} = \sqrt{2} \\ \textcircled{3} = 0 & \textcircled{3} = \sqrt{2}e^{i\alpha} \\ \textcircled{4} = 1 + e^{i\alpha} & \textcircled{5} = \frac{1}{\sqrt{2}} \end{array}$$

It is possible to show that one can construct any complex number using these scalars. As a next step, we will represent the CNOT, the Hadamard, the S and the T-gate as ZX-diagrams:



Since this is a known *universal quantum gate set* [46], we can thus, together with the correct scalars, represent any linear map between finite-dimensional Hilbert spaces [45]. \square

Is the ZX-calculus sound? Is the interpretation of any ZX-diagram invariant under all rewrite rules? The answer is again yes. The proof involves showing invariance for each rewrite rule, which is outside the scope of this thesis [45].

Is the ZX-calculus complete? Is it possible to prove any true equation for linear mappings using ZX-diagrams and a set of rewrite rules of the ZX-calculus? And again, the answer is yes. However, this result took many years until the most general version could be proven [47]. Simpler versions such as the completeness for the stabilizer-fragment were proven before that [48].

3.2 Related Calculi

In the previous section, we have seen that new generators like the Hadamard box can be introduced in order to make diagrams more concise. We will consider another example of such generators in Section 4.1, where we will see that the introduction of *triangles*, and for that matter also *stars*, can make our diagrams much more compact and allow us to make easy-to-write rules.

As it turns out, one can much more drastically change the ZX-calculus, by replacing the fundamental generator **X** with the **H** or **W** generator.

The first of these two generators is the *H-box*, which is a generalization of the Hadamard box. Formally, it is defined as

$$m \left\{ \begin{array}{c} \diagdown \quad \diagup \\ : \textcolor{blue}{a} : \\ \diagup \quad \diagdown \end{array} \right\} n = \sum a^{i_1 \dots i_m j_1 \dots j_n} |j_1 \dots j_n\rangle \langle i_1 \dots i_m| \quad (3.7)$$

where the sum is over all $i_1, \dots, i_m, j_1, \dots, j_n \in \{0, 1\}$ and $a \in \mathbb{C}$. It can thus be seen that all entries of the corresponding matrix are 1, except for the bottom right element, which is equal to a . For example, we have that

$$\text{---} \textcolor{blue}{a} \text{---} = \begin{pmatrix} 1 & 1 \\ 1 & a \end{pmatrix}.$$

Notice that for $a = -1$, we simply get a rescaled Hadamard box:

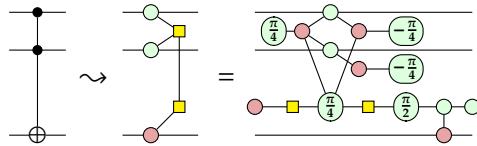
$$\text{---}[-1]\text{---} = \sqrt{2} \text{---}\square\text{---} \quad (3.8)$$

It is standard practice to omit the -1 label for H-boxes, which results in the same yellow box that is used for Hadamard boxes. Since this might lead to confusion, it is important to clarify which convention is being used, if it is not clear from the context.

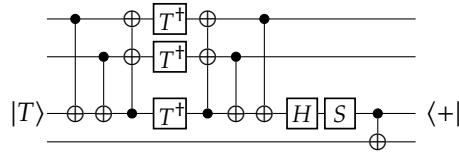
9: We refrain from showing the rewrite-rules, as they will not be used in this thesis.

This generator alongside with the according rewrite-rules⁹ results in the *ZH-calculus*, which was introduced in [49].

There are many applications of this new calculus, some of which we will see in Section 3.3. One common use-case is the representation of controlled unitaries. For instance, it is possible to show the following [37]:



We can extract the resulting quantum circuit from the last diagram by using so-called *phase gadgets*, giving us a particularly efficient construction:



The second generator is the *W-spider*, which is formally defined as

$$\begin{array}{c} \diagup \\ \diagdown \end{array} \bullet : = |10\dots0\rangle\langle0\dots0| + \dots + |0\dots01\rangle\langle0\dots0| \quad (3.9)$$

$$+ |0\dots0\rangle\langle10\dots0| + \dots + |0\dots0\rangle\langle0\dots01|. \quad (3.10)$$

where the sum is over all $|x_1\dots x_n\rangle\langle y_1\dots y_m|$ such that only one of x_i and y_j is 1. This generator creates the basis for a new calculus, the *ZW-calculus*, which was introduced in [50].

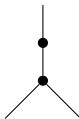
At this point it is worth noting that a translation between the ZX-calculus, the ZH-calculus and the ZW-calculus is in fact possible, as demonstrated in [51]. This means that, in theory, all three calculi are capable of the same, however, in practice, one calculus might be much easier to work with than the others. Furthermore, it is possible to combine aspects of the individual calculi, resulting in the *ZXH-calculus* and the *ZXW-calculus* [39] [52].

As demonstrated in [53], one might even consider a d -dimensional extension of the ZXW-calculus, which can be used to reason about *qudits*. One of the commonly used generators in that calculus is the *W node*:

$$\begin{array}{c} \nearrow \\ \nwarrow \end{array} = |00\rangle\langle0| + \sum_{i=1}^{d-1} (|0i\rangle + |i0\rangle)\langle i| \quad (3.11)$$

By setting $d = 2$, we can see that Eq. (3.11) becomes equivalent to the following

diagram involving W-spiders:



3.3 Applications

At the start of this chapter, we have seen many graphical languages used in a wide variety of research fields. In the following, we will provide an overview of what research directions have been pursued since the introduction of the ZX-calculus in 2008.

One of the first applications of the ZX-calculus was the description of graph states and measurement-based quantum computations [54] [55]. Another early use case was the verification of quantum protocols [56]. Verification of oracle-based quantum algorithms was later enabled through the development of the scalable ZX-calculus [57]. As it turns out, the ZX-calculus is also a great language to reason about lattice surgery on surface codes [58] [59]. It has also proved useful for finding and verifying the correctness of quantum error correcting codes [60] [61] [62]. It has even been used to study more exotic ideas like quantum natural language processing, which aims to study linguistic meanings and structures using quantum hardware [63]. Arguably one of the biggest research directions is concerned with classical simulation [5] [64] [65], quantum circuit optimization [66] [67] [68] and quantum circuit equality validation [69]. The ZX-calculus has been successfully applied to the boolean and counting satisfiability problem [70] [44]. In Section 4.2 we will see aspects of another research direction, concerned with the analysis of barren plateaus [71] [72]. Finally, the ZX-calculus enables us to study not only problems from computer science, but also many problems from physics. For example, it has been used in condensed matter physics to describe AKLT-states [73]. Even links to quantum field theory have been made, by creating a categorical description of Feynman diagrams¹⁰ [74], or more recently, by introducing Fock spiders, which allow for continuous-variable quantum computation. This could enable native simulation of quantum field theories in the future [75]. The ZXH-calculus has also been used to describe SU(2) representation theory, more precisely to represent Yutsis diagrams and Penrose spin-networks, which creates a connection to loop quantum gravity [39]. Lastly, [76] explored the relationship between the ZX-calculus and the Wolfram model, a fundamental theory of physics that tries to describe the fundamental structure of the universe using hypergraph rewriting systems [77].

10: We will briefly talk about quantum field theory and Feynman diagrams in Chapter 6.

PART II: DECOMPOSITIONS OF NON-STABILIZER STATES

4

Novel State Decompositions

In Section 2.3, we discussed different (classical) simulation methods. The state vector approach is a widely used method. However, when applied on n qubits, this approach leads to a computational complexity of $\mathcal{O}(2^n)$. One approach that tends to yield much better results both memory-wise and time-wise is the concept of **stabilizer decompositions**. It allows us to do strong simulation.

4.1 Previous Work

Consider any given quantum circuit. Furthermore, without loss of generality, assume the input to be $|0\rangle^{\otimes n}$. The idea of a stabilizer decomposition is to decompose the output $|\psi\rangle$ into a linear combination of Clifford states¹ $|\phi_1\rangle, \dots, |\phi_k\rangle$

$$|\psi\rangle = \sum_{i=1}^k a_i |\phi_i\rangle \quad (4.1)$$

for complex coefficients a_i . The probability of any output x is thus given by

$$\langle x|\psi\rangle = \sum_{i=1}^k a_i \langle x|\phi_i\rangle. \quad (4.2)$$

According to the Gottesman-Knill theorem, each term $\langle x|\phi_i\rangle$ can be computed in polynomial time, meaning the whole strong simulation has a complexity of $\mathcal{O}(k)$.

Remark 4.1.1 This procedure creates no conflicts with the theoretical complexity of the problem. This can be seen by decomposing a non-Clifford state into two Clifford states. If we then want to use this decomposition for three of the original non-Clifford states combined using a tensor product, we have to multiply the Clifford states individually, resulting in $2^3 = 8$ Clifford states. We conclude that we still have exponential scaling.

Consequently, minimizing the number of terms in such decompositions has become an active area of current research [78]. Such a minimization is limited by the *stabilizer rank*.

Definition 4.1.2 Given an arbitrary n -qubit state ψ , we define the *stabilizer rank* of ψ as the smallest non-negative integer χ , denoted $\chi(\psi)$, such that it is possible to write $|\psi\rangle = \sum_{i=1}^{\chi} a_i |\phi_i\rangle$.

One widely used example of a general stabilizer decomposition is the so-called **magic state** $|T\rangle := \frac{1}{\sqrt{2}}(|0\rangle + e^{i\frac{\pi}{4}}|1\rangle)$, which can be expressed using ZX-diagrams:

$$\textcircled{\frac{\pi}{4}}\text{---} = \frac{1}{\sqrt{2}} \left(\textcircled{\text{---}} + e^{i\frac{\pi}{4}} \textcircled{\pi} \textcircled{\text{---}} \right) \quad (4.3)$$

4.1 Previous Work	29
4.2 This Work	33

1: A (non-)Clifford state is a state obtained from a (non-)Clifford circuit (cf. Section 2.3).

Notice that such states can be unfused easily, a process similar to the magic state injection protocol used in other fields apart from the ZX-calculus:

$$\vdots \quad (2k+1)\frac{\pi}{4} \quad \vdots = \quad \vdots \quad \frac{\pi}{4} \quad \vdots \quad (4.4)$$

According to Remark 4.1.1, $\frac{\pi}{4}$ $^{\otimes t}$ would result in 2^t terms. A more sophisticated approach for $t = 2$ would be to use:

$$\begin{array}{c} \frac{\pi}{4} \\ \frac{\pi}{4} \end{array} = \begin{array}{c} \frac{\pi}{2} \\ \text{---} \end{array} + e^{i\frac{\pi}{4}} \begin{array}{c} \pi \\ \text{---} \end{array} \quad (4.5)$$

Notice that the scaling for these decompositions is given by $p^{t/r}$, where p is the number of terms and r represents the number of non-Clifford terms reduced, more concretely by how much the T-count was reduced. We will measure the scaling using the number α , defined by

$$p^{t/r} = 2^{\alpha t} \implies \alpha = \frac{\log_2(p)}{r} \quad (4.6)$$

for further comparisons. In Eq. (4.5), for example, the number α is equal to $\log_2(2)/2 = 0.5$. The objective is thus to find stabilizer decompositions with a small α [5].

One such decomposition that turns out to be very useful was first introduced in [79] by Bravyi, Smith and Smolin, thus often being referred to as the **BSS decomposition**. It was later reformulated using ZX-diagrams² in [64]:

$$\begin{aligned} e^{i\pi/4} \begin{array}{c} \frac{\pi}{4} \\ \frac{\pi}{4} \\ \frac{\pi}{4} \\ \frac{\pi}{4} \\ \frac{\pi}{4} \\ \frac{\pi}{4} \end{array} &= +2e^{i\pi/4} \begin{array}{c} \frac{\pi}{2} \\ \text{---} \end{array} \\ -\frac{1+\sqrt{2}}{4} \begin{array}{c} \text{---} \\ \text{---} \\ \text{---} \\ \text{---} \\ \text{---} \\ \text{---} \end{array} &+ \frac{1-\sqrt{2}}{4} \begin{array}{c} \pi \\ \pi \\ \pi \\ \pi \\ \pi \\ \pi \end{array} \\ -2\sqrt{2}i \begin{array}{c} \text{---} \\ \text{---} \\ \text{---} \\ \text{---} \\ \text{---} \\ \text{---} \end{array} &-2i \begin{array}{c} \text{---} \\ \text{---} \\ \text{---} \\ \text{---} \\ \text{---} \\ \text{---} \end{array} \\ +8\sqrt{2}i \begin{array}{c} \text{---} \\ \text{---} \\ \text{---} \\ \text{---} \\ \text{---} \\ \text{---} \end{array} &+8\sqrt{2}i \begin{array}{c} \text{---} \\ \text{---} \\ \text{---} \\ \text{---} \\ \text{---} \\ \text{---} \end{array} \end{aligned} \quad (4.7)$$

Since $\alpha = \log_2(7)/6 \approx 0.468$, we call it a "6-to-7" decomposition.

Interestingly, there exist many other decomposition strategies that sometimes yield even better results. For example, **partial magic state decompositions** use non-stabilizer decompositions, meaning the terms in the linear combination

2: Note that whenever the diagrams are turned clockwise for space reasons, time evolution is from top to bottom.

might only have *less* T-spiders, but not necessarily *zero*:

$$\begin{array}{c} \text{---} \\ \text{---} \\ \text{---} \\ \text{---} \\ \text{---} \end{array} = 2 \begin{array}{c} \text{---} \\ \text{---} \\ \text{---} \\ \text{---} \\ \text{---} \end{array} + 2\sqrt{2}ie^{i\pi/4} \begin{array}{c} \text{---} \\ \text{---} \\ \text{---} \\ \text{---} \\ \text{---} \end{array} - 2\sqrt{2}e^{i\pi/4} \begin{array}{c} \text{---} \\ \text{---} \\ \text{---} \\ \text{---} \\ \text{---} \end{array} \quad (4.8)$$

So effectively, this is a 4-to-3 strategy, resulting in $\alpha = \log_2(3)/4 \approx 0.396$ [5].

Another approach introduced in [80] focuses on triangles instead of T-spiders:

$$\begin{aligned} \text{---} \rightarrow &= \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix} \\ \text{---} \rightarrow &= \text{---} \circlearrowleft \text{---} = \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix} \quad (4.9) \\ \text{---} \rightarrow^{-1} &= \text{---} \circlearrowleft \text{---} = \begin{pmatrix} 1 & -1 \\ 0 & 1 \end{pmatrix} \end{aligned}$$

However, since these triangles essentially create directed graphs, one can use the symmetric **star** instead:

$$\text{---} \star := \text{---} \circlearrowleft \text{---} = \text{---} \circlearrowright \text{---} = \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix} \quad (4.10)$$

Similar as for Hadamard edges, we will use an orange dashed line to represent a **star edge**:

$$\text{---} \star \text{---} \rightsquigarrow \text{---} \text{---} \quad (4.11)$$

Technically, we could write triangles in terms of four T-spiders:

$$\text{---} \rightarrow = \frac{1}{2} \text{---} \circlearrowleft \text{---} \circlearrowright \text{---} \quad (4.12)$$

This is suboptimal, since given n T-spiders and m triangles, the scaling is $2^{\alpha n + \beta m}$, where $\beta = 4\alpha$. This results in a value greater than $\beta = 1$, which would already result from its naive decomposition. What will follow now is this decomposition, along with more efficient ones, but formulated using star edges instead of triangles³:

3: We will refer to Eq. (4.13), Eq. (4.14) and Eq. (4.15) as *star decompositions*.

$$\boxed{\text{---}} = \sqrt{2} \begin{array}{c} \text{---} \\ \text{---} \end{array} + 2 \begin{array}{c} \text{---} \\ \text{---} \end{array} \quad (4.13)$$

$$\boxed{\text{---}} = \frac{1}{\sqrt{2}} \begin{array}{c} \text{---} \\ \text{---} \end{array} + \frac{1}{\sqrt{2}} \begin{array}{c} \text{---} \\ \text{---} \end{array} + 4 \begin{array}{c} \text{---} \\ \text{---} \end{array} \quad (4.14)$$

$$\begin{aligned} \boxed{\text{---}} &= \frac{1}{2\sqrt{2}} \begin{array}{cc} \text{---} & \text{---} \\ \text{---} & \text{---} \end{array} + \frac{1}{2\sqrt{2}} \begin{array}{cc} \text{---} & \text{---} \\ \text{---} & \text{---} \end{array} + \frac{1}{2\sqrt{2}} \begin{array}{cc} \text{---} & \text{---} \\ \text{---} & \text{---} \end{array} \\ &+ \frac{1}{\sqrt{2}} \begin{array}{c} \text{---} \\ \text{---} \end{array} + 8 \begin{array}{c} \text{---} \\ \text{---} \end{array} \end{aligned} \quad (4.15)$$

4: We will refer to such states as *star states*.

The corresponding scalings for Eq. (4.13), Eq. (4.14) and Eq. (4.15) are given by $\beta = \log_2(2)/1 = 1$, $\beta = \log_2(3)/2 \approx 0.792$ and $\beta = \log_2(5)/3 \approx 0.774$, respectively. This can be further improved by considering special cases, namely when star edges are connected to Z-spiders⁴, creating **non-stabilizer states**, which can be decomposed into **stabilizer states**.

$$\boxed{\text{---} \text{---} \text{---}} = 3 \begin{array}{c} \text{---} \\ \text{---} \end{array} - \begin{array}{ccc} \text{---} \\ \text{---} \end{array} + \frac{3}{\sqrt{2}} \begin{array}{ccc} \text{---} & \text{---} & \text{---} \\ \text{---} & \text{---} & \text{---} \end{array} - \frac{3}{2\sqrt{2}} \begin{array}{ccc} \text{---} & \text{---} & \text{---} \\ \text{---} & \text{---} & \text{---} \end{array} \quad (4.16)$$

$$\begin{aligned} \boxed{\left(\pm\frac{\pi}{2}\right) \left(\pm\frac{\pi}{2}\right) \left(\pm\frac{\pi}{2}\right)} &= \frac{1 \pm 3i}{2} \begin{array}{c} \text{---} \\ \text{---} \end{array} + \frac{1 \mp i}{2} \begin{array}{c} \text{---} \\ \text{---} \end{array} - \frac{3-i}{2\sqrt{2}} \begin{array}{ccc} \text{---} & \text{---} & \text{---} \\ \text{---} & \text{---} & \text{---} \end{array} \\ &+ \frac{1 \mp i}{2\sqrt{2}} \begin{array}{ccc} \text{---} & \text{---} & \text{---} \\ \text{---} & \text{---} & \text{---} \end{array} \end{aligned} \quad (4.17)$$

The corresponding scaling here is $\beta = \log_2(4)/3 \approx 0.667$ for both decompositions. Note that we do need to consider a π -phase, since the resulting state is in fact Clifford:

$$\boxed{\text{---} \text{---}} = \frac{1}{\sqrt{2}} \boxed{\text{---} \text{---}} \quad (4.18)$$

At this point, it is worth mentioning that although this *star formalism* was introduced in [80], the underlying linear map is equivalent up to a scalar to the zero-labelled H-box in the ZH-calculus:

$$\boxed{\text{---} \text{---}} = \frac{1}{2} \boxed{\text{---} \text{---}} \quad (4.19)$$

5: Note that they use a slightly different, but equivalent notation to the more traditional notation described in Section 3.2.

In fact, such stabilizer decompositions utilizing the H-box formalism had already been introduced in [44], most notably they included one decomposition⁵ similar to the stabilizer state decompositions from before, but with a better scaling of $\beta = \log_2(5)/4 \approx 0.580$, which can be seen in Figure 4.1:

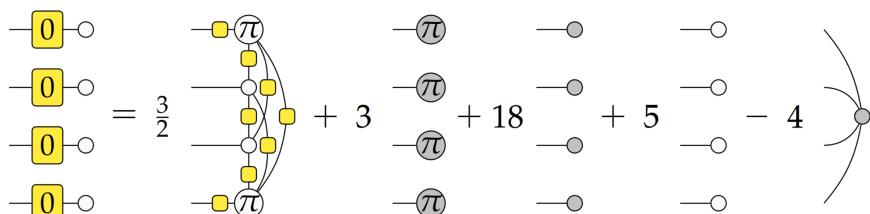


Figure 4.1: Four states tensored together with their according decomposition taken from [44].

4.2 This Work

One of the first experiments we conducted after having started learning about the ZX-calculus was an attempt at proving certain theorems required to check the validity of a quantum query algorithm (*cf.* Section 6.3) introduced in an undergraduate thesis⁶. It involved the calculation of the expectation value $\mathbb{E}_\theta(f)$ given by the integral

$$\mathbb{E}_\theta(f) = \left(\frac{1}{2\pi}\right)^n \int_{[0,2\pi]^n} f(\theta) d\theta, \quad (4.20)$$

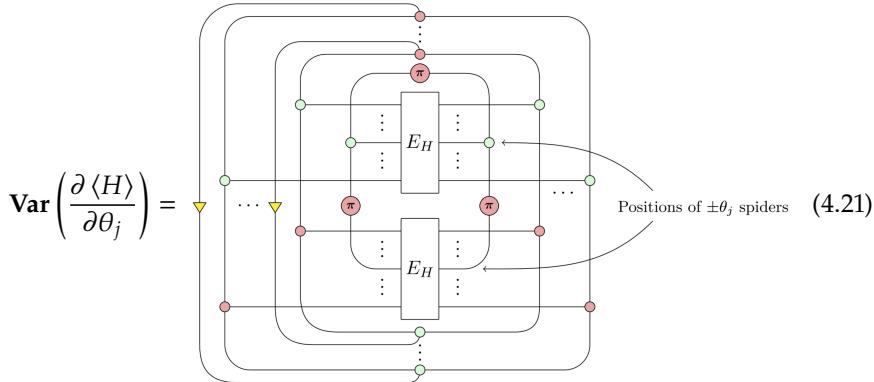
where n is the number of qubits and f is a placeholder for a matrix composed of permutation and rotation matrices, which are dependent on θ .

Thus having found [72] proved beneficial, as it included a theorem regarding *diagrammatic integration*. Later, however, we developed a classical algorithm to solve the same problem as the quantum query algorithm. Even for simple starting configurations, the classical algorithm heavily outperformed the quantum algorithm. Of course, the theoretical study could still be of interest, but it was decided to investigate another, more practical aspect of [72], namely the concept of *barren plateau detection*.

The work in [72] introduced a theorem regarding *diagrammatic variance calculation*⁷. It shows that given a certain type of Hamiltonian H , and therefore also its expectation value E_H , the equality in Eq. (4.21) holds. Note that since E_H is not only dependent on the Hamiltonian, but also on the *circuit ansatz* $U(\theta)$ for the n -tuple $\theta = (\theta_1, \theta_2, \dots)$, the resulting ZX-diagram contains spiders with phase θ_j .

6: As it has not been published yet, we cannot cite it here.

7: The variance can be calculated using the expectation value, which again can be calculated using diagrammatic integration.



The reason for the appearance of the Hamiltonian is that the type of quantum circuits being studied here is part of a field called *quantum machine learning*. In *quantum chemistry*, a *circuit ansatz* might be used to find the ground state of a Hamiltonian. Such ansätze are often optimized using gradient-based techniques, much like how it is done in classical machine learning. However, it has been shown that many ansätze suffer from an exponentially flat training landscape, making gradient descent impossible. This type of landscape is referred to as *barren plateau*. One common approach to detect barren plateaus is to check whether $\text{Var}\left(\frac{\partial \langle H \rangle}{\partial \theta_j}\right) \approx 0$, in which case a barren plateau is likely to start and thus disturb the learning progression [81].

For the next project, we attempted to brute force linear combinations resulting in decompositions that could be used to simulate Eq. (4.21). More precisely, the

goal was to decompose the non-Clifford cycle:



8: The code is very similar to the implementation in [82].

9: In retrospect, it is clear that we had basically found the standard decomposition of triangles/star edges as in Eq. (4.13), only that it had been embedded in the diagram of the cycle. This was not obvious at the time, as we did not know of these decompositions.

10: It can be proven that the optimal decomposition of a real state consists only of real Cliffords [84].

11: The coefficients are written separately for space reasons.

For this purpose, a big set of potential decomposition terms was generated, hence also obtaining the according matrices. These matrices were combined in a linear combination in Mathematica⁸ [83]. By a process of elimination, a decomposition consisting of two terms was indeed found⁹. One could thus compute the variance using a sum of 2^{m-1} terms, where m is the number of parameters in the considered ansatz. At the time this seemed to be a nice improvement over the approach introduced in [71], which resulted in a sum of 3^{m-1} terms.

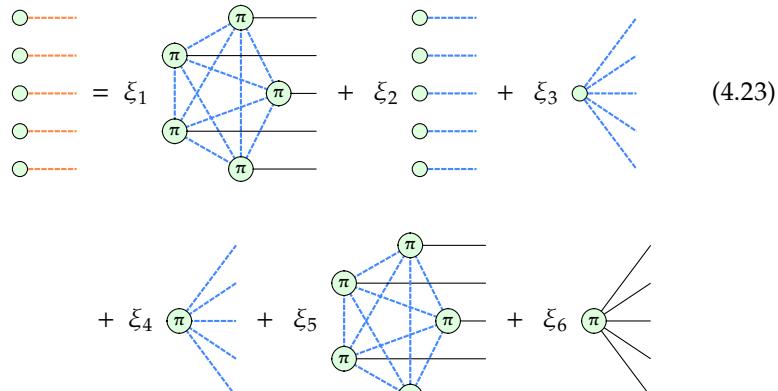
It was only later that we realized that [80] had already introduced a better approach utilizing decompositions of star edges as described in Section 4.1, resulting in a scaling of $2^{\beta(m-1)}$.

Since finding better decompositions than the ones introduced in [80] and [44] would most likely require a better approach than our brute force method, we investigated the techniques that were originally used to find such decompositions. As it turns out, there exists an open-source implementation [84] of the technique used in [79] to find the (asymptotic) 6-to-7 stabilizer decomposition.

The algorithm uses *simulated annealing* to search over the solution space of the real Cliffords¹⁰. Simulated annealing is a probabilistic optimization algorithm that explores the solution space using a random walk and probabilistically accepting worse solutions in order to escape local optima, whilst gradually reducing this acceptance probability, aiming to converge on a global optimum [79] [85].

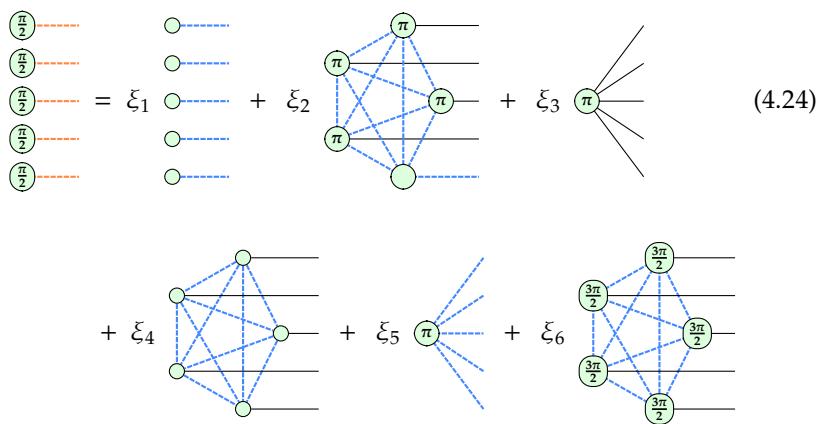
By configuring the program to search for specific states using different annealing parameters, we were in fact able to find novel decompositions. It is worth noting that the parameters are crucial, since the solution is easily overlooked when reducing the acceptance probability too quickly or sometimes even too slowly. As the program did not provide the coefficients of the decomposition, we created a small routine to find them. This routine, together with a program to check the validity of the decompositions, can be found on GitHub [82].

These efforts led to the following five novel decompositions¹¹:



$$\xi_1 = -192, \quad \xi_2 = \frac{15\sqrt{2}}{8}, \quad \xi_3 = 10\sqrt{2}, \quad \xi_4 = 20\sqrt{2}, \quad \xi_5 = 48\sqrt{2}, \quad \xi_6 = 15$$

Eq. (4.23) yields $\beta = \frac{\log_2 6}{5} \approx 0.517$.



$$\begin{aligned}\xi_1 &= -\frac{5i}{4\sqrt{2}}, \quad \xi_2 = 56 + 8i, \quad \xi_3 = -5, \quad \xi_4 = -(64 + 32i), \quad \xi_5 = -(7\sqrt{2} - \sqrt{2}i), \\ \xi_6 &= -(16 - 48i)\end{aligned}$$

Eq. (4.24) yields $\beta = \frac{\log_2 6}{5} \approx 0.517$.

$$\begin{array}{c} \text{---} \\ (-\frac{\pi}{2}) \end{array} \quad \begin{array}{c} \text{---} \\ (-\frac{\pi}{2}) \end{array} \quad \begin{array}{c} \text{---} \\ (-\frac{\pi}{2}) \end{array} = \xi_1 \begin{array}{c} \text{---} \\ (-\frac{\pi}{2}) \end{array} + \xi_2 \begin{array}{c} \text{---} \\ (-\frac{\pi}{2}) \end{array} + \xi_3 \begin{array}{c} \text{---} \\ (-\frac{\pi}{2}) \end{array} \quad (4.25)$$

$$+ \xi_4 \begin{array}{c} \text{---} \\ (-\frac{\pi}{2}) \end{array} + \xi_5 \begin{array}{c} \text{---} \\ (\pi) \end{array} + \xi_6 \begin{array}{c} \text{---} \\ (\frac{3\pi}{2}) \end{array}$$

$$\begin{aligned} \xi_1 &= \frac{5i}{4\sqrt{2}}, \quad \xi_2 = 56 - 8i, \quad \xi_3 = -5, \quad \xi_4 = -(64 - 32i), \quad \xi_5 = -(\sqrt{2} + 3\sqrt{2}i), \\ \xi_6 &= -(16 + 48i) \end{aligned}$$

Eq. (4.25) yields $\beta = \frac{\log_2 6}{5} \approx 0.517$.

$$\begin{array}{c} \text{---} \\ (\frac{\pi}{2}) \end{array} \quad \begin{array}{c} \text{---} \\ (\frac{\pi}{2}) \end{array} \quad \begin{array}{c} \text{---} \\ (\frac{\pi}{2}) \end{array} = \xi_1 \begin{array}{c} \text{---} \\ (\frac{\pi}{2}) \end{array} + \xi_2 \begin{array}{c} \text{---} \\ (\frac{\pi}{2}) \end{array} + \xi_3 \begin{array}{c} \text{---} \\ (\frac{\pi}{2}) \end{array} \quad (4.26)$$

$$+ \xi_4 \begin{array}{c} \text{---} \\ (\frac{\pi}{2}) \end{array} + \xi_5 \begin{array}{c} \text{---} \\ (\pi) \end{array}$$

$$\xi_1 = -6 + 2i, \quad \xi_2 = -\frac{5 + 5i}{2}, \quad \xi_3 = -(3\sqrt{2} - \sqrt{2}i), \quad \xi_4 = -(6 - 18i), \quad \xi_5 = \frac{7 + 9i}{2}$$

Eq. (4.26) yields $\beta = \frac{\log_2 5}{4} \approx 0.580$.

$$\begin{array}{c} \text{---} \\ (-\frac{\pi}{2}) \end{array} \quad \begin{array}{c} \text{---} \\ (-\frac{\pi}{2}) \end{array} \quad \begin{array}{c} \text{---} \\ (-\frac{\pi}{2}) \end{array} = \xi_1 \begin{array}{c} \text{---} \\ (-\frac{\pi}{2}) \end{array} + \xi_2 \begin{array}{c} \text{---} \\ (-\frac{\pi}{2}) \end{array} + \xi_3 \begin{array}{c} \text{---} \\ (-\frac{\pi}{2}) \end{array} \quad (4.27)$$

$$+ \xi_4 \begin{array}{c} \text{---} \\ (\frac{\pi}{2}) \end{array} + \xi_5 \begin{array}{c} \text{---} \\ (\pi) \end{array}$$

$$\xi_1 = -6 + 18i, \quad \xi_2 = -\frac{5 + 5i}{2}, \quad \xi_3 = -(3\sqrt{2} + 11\sqrt{2}i), \quad \xi_4 = -(6 - 2i), \quad \xi_5 = -\frac{1 + 3i}{2}$$

Eq. (4.27) yields $\beta = \frac{\log_2 5}{4} \approx 0.580$.

As of writing this thesis, we have not come across comparable decompositions.

To test the limitations of this approach, the program was run on an AWS EC2 C6a instance (c6a.32xlarge) [86] to search for more complex decompositions. Although using parallelization on 128 vCPUs, together with 256 GiB of memory, we did not find any novel decompositions. It is presumed that one fundamental limitation of this approach is that it either finds a result relatively quickly, or it never finds one, thus not being able to take advantage of high-performance computers.

The practicality of the novel decompositions alongside with the star edge decompositions listed in Section 4.1 will be discussed in Section 5.2.

PART III: MULTI-CONTROL TOFFOLI GATE DENSE QUANTUM CIRCUITS

Dynamic Decompositions

In Chapter 4, we discussed state decompositions. Mathematically, these correspond to decomposing a *vector* into a sum of other *vectors*. In the following, we will demonstrate that we can also decompose *matrices*.

5.1 Overview	41
5.2 Comparison with State Decompositions	43

5.1 Overview

Using the concept of decompositions for quantum computation may seem like a rather exotic idea, however, one of the most elementary definitions already allows us to obtain a very useful decomposition:

Theorem 5.1.1

$$m \left\{ \begin{array}{c} : \\ \diagup \alpha \diagdown \\ \text{---} \end{array} \right\} n = \frac{1}{\sqrt{2^{n+m}}} m \left\{ \begin{array}{c} : \\ \diagup \textcolor{red}{\pi} \diagdown \\ \text{---} \end{array} \right\} n + \frac{e^{i\alpha}}{\sqrt{2^{n+m}}} m \left\{ \begin{array}{c} : \\ \diagup \textcolor{red}{\pi} \diagdown \\ \text{---} \end{array} \right\} n \quad (5.1)$$

Proof. From Eq. (3.3) we have

$$m \left\{ \begin{array}{c} : \\ \diagup \alpha \diagdown \\ \text{---} \end{array} \right\} n = |0\rangle^{\otimes n} \langle 0|^{\otimes m} + e^{i\alpha} |1\rangle^{\otimes n} \langle 1|^{\otimes m}. \quad (5.2)$$

Furthermore, we know that

$$|0\rangle = \frac{1}{\sqrt{2}}(|+\rangle + |-\rangle) = \frac{1}{\sqrt{2}} \textcolor{red}{\circ} \text{---} \quad (5.3)$$

and

$$|1\rangle = \frac{1}{\sqrt{2}}(|+\rangle - |-\rangle) = \frac{1}{\sqrt{2}} \textcolor{red}{\pi} \text{---}. \quad (5.4)$$

Therefore, plugging in Eq. (5.3) and Eq. (5.4) into Eq. (5.2), we get:

$$m \left\{ \begin{array}{c} : \\ \diagup \alpha \diagdown \\ \text{---} \end{array} \right\} n = \frac{1}{\sqrt{2^{n+m}}} m \left\{ \begin{array}{c} : \\ \diagup \textcolor{red}{\pi} \diagdown \\ \text{---} \end{array} \right\} n + \frac{e^{i\alpha}}{\sqrt{2^{n+m}}} m \left\{ \begin{array}{c} : \\ \diagup \textcolor{red}{\pi} \diagdown \\ \text{---} \end{array} \right\} n$$

□

Remark 5.1.2 Theorem 5.1.1 can be rewritten using the (c)-rule, resulting in

$$m \left\{ \begin{array}{c} : \\ \diagup \alpha \diagdown \\ \text{---} \end{array} \right\} n = \frac{1}{\sqrt{2}} m \left\{ \begin{array}{c} : \\ \diagup \textcolor{red}{\circ} \diagdown \\ \text{---} \end{array} \right\} n + \frac{e^{i\alpha}}{\sqrt{2}} m \left\{ \begin{array}{c} : \\ \diagup \textcolor{red}{\pi} \diagdown \\ \text{---} \end{array} \right\} n. \quad (5.5)$$

This form is particularly handy for programmatic implementations, as one only has to attach a single state to the center node and multiply by a scalar¹.

1: And then also do a simplification that would be necessary in either case.

In [80], Theorem 5.1.1 was used to derive the following decomposition:

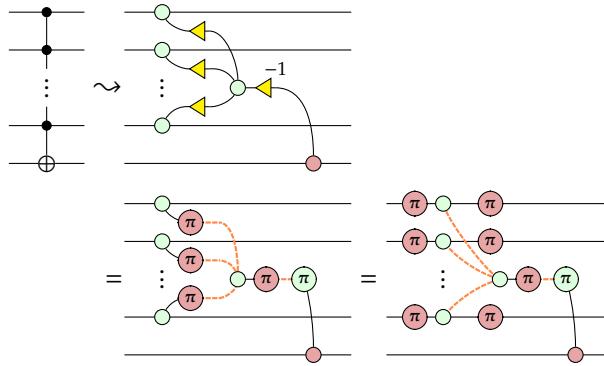
Theorem 5.1.3

$$\begin{array}{c} \text{Diagram showing a multi-control Toffoli gate with controls } \beta_1, \dots, \beta_n \text{ and target } \gamma_1, \dots, \gamma_m. \text{ A central control } \alpha \text{ is connected to all targets.} \\ \vdots \\ \beta_1 \quad \alpha \quad \gamma_1 \\ \vdots \quad \vdots \quad \vdots \\ \beta_n \quad \alpha \quad \gamma_m \end{array} = \frac{1}{\sqrt{2^n}} \begin{array}{c} \text{Diagram showing a multi-control Toffoli gate with controls } \beta_1, \dots, \beta_n \text{ and target } \gamma_1, \dots, \gamma_m. \\ \vdots \\ \beta_1 \quad \gamma_1 \\ \vdots \quad \vdots \\ \beta_n \quad \gamma_m \end{array} + \frac{e^{i\alpha}}{\sqrt{2^{n+m}}} \begin{array}{c} \text{Diagram showing a multi-control Toffoli gate with controls } \beta_1 + \pi, \dots, \beta_n + \pi \text{ and target } \gamma_1, \dots, \gamma_m. \\ \vdots \\ \beta_1 + \pi \quad \gamma_1 \\ \vdots \quad \vdots \\ \beta_n + \pi \quad \gamma_m \end{array} \quad (5.6)$$

2: The term does not occur in literature, but we will be referring to this decomposition a lot, so we have thus introduced it.

This yields a scaling of $\beta = \frac{1}{m}$ for m star edges. Since this decomposition holds for any m , we may call this a *dynamic decomposition*².

We can use these theorems to find the stabilizer rank of a multi-control Toffoli gate with n control bits. The first step is to translate the according quantum circuit into a ZX-diagram [80]:



For the sake of simplicity, we will use Theorem 5.1.1 instead of Theorem 5.1.3:

$$\begin{array}{c} \text{Diagram showing a multi-control Toffoli gate with controls } \pi \text{ and target } \pi. \\ \vdots \\ \pi \quad \pi \\ \vdots \quad \vdots \\ \pi \quad \pi \end{array} = \frac{1}{\sqrt{2^{n+1}}} \begin{array}{c} \text{Diagram showing a multi-control Toffoli gate with controls } \pi \text{ and target } \pi. \\ \vdots \\ \pi \quad \pi \\ \vdots \quad \vdots \\ \pi \quad \pi \end{array} + \frac{1}{\sqrt{2^{n+1}}} \begin{array}{c} \text{Diagram showing a multi-control Toffoli gate with controls } \pi \text{ and target } \pi. \\ \vdots \\ \pi \quad \pi \\ \vdots \quad \vdots \\ \pi \quad \pi \end{array} \quad (5.7)$$

At this point, it is worth introducing a few simple to derive equalities:

$$\text{Diagram with a red circle and a dashed orange line} = \text{Diagram with a red circle} \quad (5.8)$$

$$\text{Diagram with a red circle and a dashed orange line} = \text{Diagram with a green circle} \quad \sqrt{2} \quad (5.9)$$

$$\text{Diagram with a red circle and a dashed orange line} = \text{Diagram with a red circle} \quad \frac{1}{\sqrt{2}} \quad (5.10)$$

One can therefore observe that both terms in Eq. (5.7) are stabilizer terms, since all non-trivial leftover states can be brought into the form of Eq. (5.8), Eq. (5.9) or Eq. (5.10). We can conclude that the stabilizer rank of a multi-control Toffoli gates is two.

In Chapter 6, we will use the elementary decomposition from Theorem 5.1.1 to do more sophisticated procedures.

5.2 Comparison with State Decompositions

One of the authors of [80], Mark Koch, was kind enough to share the source code for their algorithm, including the benchmark for barren plateau detection.

We repeated the benchmark after having slightly modified the code, which allowed us to capture information regarding which decomposition was applied for a given ansatz. The results³ show that around two-thirds of the decompositions were star decompositions, and only one-third utilized dynamic decompositions. State decompositions did not occur.

3: The log file can be found in [87].

It is evident that decompositions of matrices occur much more often than state decompositions, as they can be used more generally. Since not even the state decompositions introduced in the original paper occurred, we can conclude that our novel decompositions are of little use for this concrete application. Nevertheless, it is probable to assume that other applications may involve such star states to a greater extent.

In the next chapter, we will focus on quantum circuits, where, according to our testing, even star decompositions are exceedingly rare or nonexistent, despite the presence of numerous star edges. Consequently, dynamic decompositions will occur more frequently, which will be beneficial for the concept we will introduce, namely *weighting algorithms*.

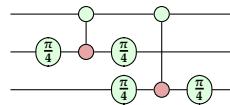
Weighting Algorithms

Whilst exploring further areas of the ZX-calculus, we encountered an interesting paper [38]. They propose and demonstrate a new approach for strong simulation. Traditionally, programs would check which decompositions¹ currently apply, and then greedily choose the one with the smallest scaling factor. The novel algorithm, in short, uses a weighting algorithm that considers more information about the graph than previously done. They implemented a Python Proof-of-Concept (PoC) [88]. The results can be seen in Figure 6.1 and Figure 6.2.

In the following, we will briefly explain their method, and then we will show whether these ideas can be translated into a similar problem, namely the strong simulation of quantum circuits consisting of many multi-control Toffoli gates, rather than many T-gates.

6.1 Basic Idea

Consider the following ZX-diagram taken from [38]:



The usual approach would be to unfuse four $\frac{\pi}{4}$ -states and then use decompositions similar to the ones reviewed in Section 4.1. However, one might instead use Theorem 5.1.1 in order to *cut* a Z-spider:

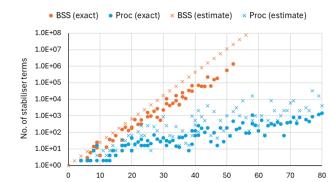
$$\begin{aligned}
 & \text{Diagram 1: } \text{Z-spider with four legs, each } \frac{\pi}{4} \\
 & = \text{Diagram 2: } \text{Z-spider with four legs, each } \frac{\pi}{4}, \text{ with a red dot on the top wire} \\
 & \approx \text{Diagram 3: } \text{Z-spider with four legs, each } \frac{\pi}{4}, \text{ with a red dot on the top wire} \\
 & \quad + \text{Diagram 4: } \text{Z-spider with four legs, each } \frac{\pi}{4}, \text{ with a red dot on the bottom wire} \\
 & = \text{Diagram 5: } \text{Two red dots on the top wire} \\
 & \quad + e^{i\frac{\pi}{2}} \text{Diagram 6: } \text{Two red dots on the bottom wire}
 \end{aligned}$$

This should be very compelling, since we have just seen that using a decomposition with a worse scaling than traditional decompositions can yield a better *effective scaling*, i.e. the scaling calculated from the final number of terms, in this case² $\alpha = 0.25$.

The above example is part of an idea called *CNOT-grouping*. More generally, in [38] it was realized that CNOTs can *block* $\pm \frac{\pi}{4}$ -spiders from fusing, which would result in a phase of either 0 or $\frac{\pi}{2}$, hence being reduced to a Clifford (sub-)diagram. By defining strict rules, one can classify each spider by considering its neighbors and counting how many T-gates could potentially be fused, up until some depth³.

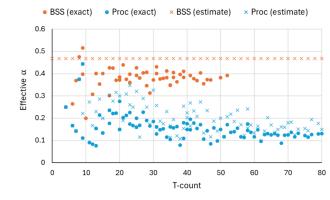
6.1 Basic Idea	45
6.2 Studied Approach . .	50
6.3 Application for Multi-loop Feynman Diagrams	53
6.4 Importance of Simplification Strategies	59

1: For example, the BSS decomposition.



(a) Number of stabiliser terms versus T-count

Figure 6.1: Results from PoC [38].



(b) Effective efficiency α versus T-count

Figure 6.2: Results from PoC [38].

2: By extending the diagram to include more separated T-gates whilst maintaining a common Z-spider, one cut would still suffice, thus giving $\alpha \rightarrow 0$.

3: The actual procedure employed is slightly more sophisticated, utilizing so-called *tiers*.

The results in Figure 6.1 and Figure 6.2 clearly show that the new approach can yield a benefit over traditional methods. Nevertheless, a few important things need to be mentioned:

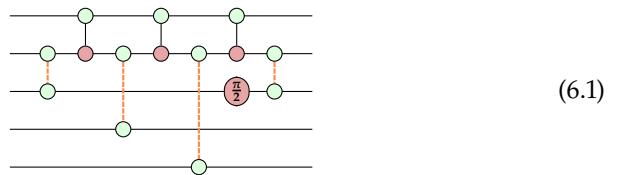
1. The experiments were conducted using "random" circuits. Here, the challenge is to achieve a natural balance between highly random circuits, which may prevent the algorithm from demonstrating its effectiveness, and highly structured circuits, where the results may be overly biased.
2. By only implementing a PoC, runtime comparisons are not possible with state-of-the-art implementations like [89]. This only leaves the final number of terms as a usable metric for comparison. Nevertheless, they did provide a rough upper-bound for the computational complexity of $\mathcal{O}(V^2)$ for V being the number of non-trivial vertices. It is thus argued to be negligible compared to the exponential scaling produced by stabilizer terms.
3. The algorithm only uses partial simplifications⁴, as this avoids the risk of losing relevant patterns. Later, however, it is argued that this might have been the reason for not even better results.

⁴: This means that they, for example, do not use *pivoting* or *local complementation*.

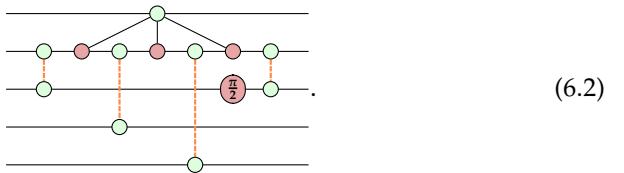
Similarly, the following points should be noted for our own algorithm, as described in Section 6.2 and Section 6.3:

1. The multiloop Feynman diagram application might be a good or a bad fit, which is why we will conduct experiments using "more random" circuits too.
2. We will also only implement a PoC in Python. We will not consider the space-time complexity in more detail, however, some further comments on the runtime will be made.
3. Our algorithm will also only use a partial simplification strategy to mimic the original algorithm and potentially avoid the same problem of losing relevant patterns. This choice will be further discussed in Section 6.4.

We will now demonstrate an attempt to replicate the idea of decomposing spiders that block further improvements in diagrams consisting of star edges instead of T-gates. To start with, consider the following diagram:

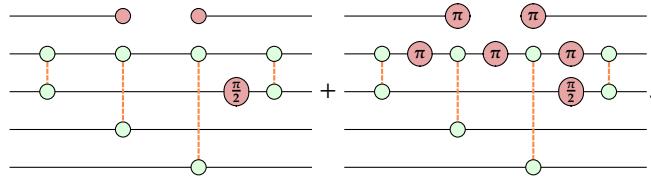


By fusing the control bits, we get



It is now possible to decompose the resulting spider according to Theorem 5.1.1,

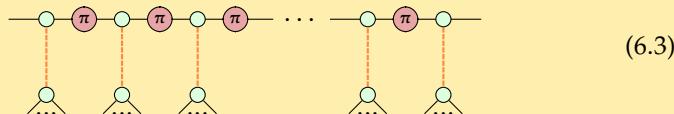
which gives us



The left term can be simplified by fusing the four Z-spiders. According to Theorem 5.1.3, we can decompose the resulting node, which gives us two terms.

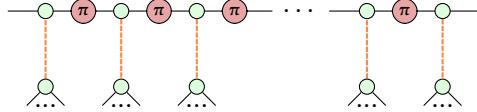
The right term is a bit more difficult due to the new obstructions caused by NOT gates. However, it can be shown that the right term can also be decomposed into two terms, according to Lemma 6.1.1. Therefore, Eq. (6.1) can be decomposed into four terms. But, what would be the final number of terms if a traditional method was used? At first, it might seem that Eq. (4.14) is the best choice. Applying it twice, one obtains nine terms. While this may be true in theory, algorithms that greedily choose the decomposition with the smallest scaling factor will always simplify the diagram after having applied one decomposition. This is exemplified after the following lemma and proof.

Lemma 6.1.1 Any (sub-)diagram of the form



with m star edges can be rewritten as

Proof. The diagram



with m star edges, can be rewritten by using Eq. (4.10), giving us

$$\begin{aligned}
 & \left. \begin{array}{c} \text{Diagram 1: } m \text{ odd} \\ \text{Diagram 2: } m \text{ even} \end{array} \right\} , \text{ if } m \text{ is odd} \\
 & \left. \begin{array}{c} \text{Diagram 1: } m \text{ even} \\ \text{Diagram 2: } m \text{ odd} \end{array} \right\} , \text{ if } m \text{ is even} \\
 = & \left. \begin{array}{c} \text{Diagram 1: } m \text{ odd} \\ \text{Diagram 2: } m \text{ even} \end{array} \right\} , \text{ if } m \text{ is odd} \\
 & \left. \begin{array}{c} \text{Diagram 1: } m \text{ even} \\ \text{Diagram 2: } m \text{ odd} \end{array} \right\} , \text{ if } m \text{ is even.}
 \end{aligned} \tag{6.5, 6.6}$$

The star edges will always be connected to one of the two Z-spiders. We will sometimes refer to these two sides as *stacks*. By decomposing both stacks using Theorem 5.1.3, one obtains four terms, two of which cancel out as they will include an isolated X-spiders with a phase of π , which is equivalent to a zero scalar. The final two terms (for both m being odd and even) are given by

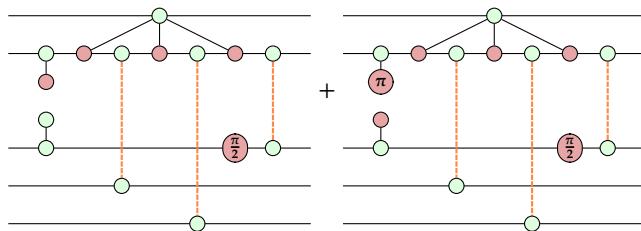
$$\begin{aligned}
 & \text{Diagram 1: } \\
 & \text{Diagram 2: } \\
 & + \quad \text{Diagram 3: } \\
 & + \quad \text{Diagram 4: } \\
 & + \quad \text{Diagram 5: } \\
 & + \quad \text{Diagram 6: } \\
 & , \text{ if } m \text{ is odd} \quad (6.7)
 \end{aligned}$$

and

$$\begin{aligned}
 & \text{---} \bullet \text{---} \bullet \text{---} \pi \text{---} \bullet \text{---} \bullet \text{---} 0 \\
 + & \text{---} \bullet \text{---} \bullet \text{---} \bullet \text{---} \bullet \text{---} \bullet \text{---} \dots \text{---} \bullet \text{---} \bullet \text{---} \bullet \text{---} \bullet \text{---} \bullet \text{---} \dots \\
 + & \text{---} \bullet \text{---} \pi \text{---} \bullet \text{---} \pi \text{---} \bullet \text{---} \pi \text{---} \bullet \text{---} \bullet \text{---} 0 \\
 + & \text{---} \bullet \text{---} \bullet \text{---} \bullet \text{---} \bullet \text{---} \bullet \text{---} \dots \text{---} \bullet \text{---} \bullet \text{---} \bullet \text{---} \bullet \text{---} \bullet \text{---} \dots \\
 + & \text{---} \bullet \text{---} \bullet \text{---} \bullet \text{---} \bullet \text{---} \bullet \text{---} \dots \text{---} \bullet \text{---} \bullet \text{---} \bullet \text{---} \bullet \text{---} \bullet \text{---} \dots \\
 + & \text{---} \bullet \text{---} \bullet \text{---} \bullet \text{---} \bullet \text{---} \bullet \text{---} \dots \text{---} \bullet \text{---} \bullet \text{---} \bullet \text{---} \bullet \text{---} \bullet \text{---} \dots \\
 + & \text{---} \bullet \text{---} \pi \text{---} \bullet \text{---} \pi \text{---} \bullet \text{---} \bullet \text{---} 0, \text{ if } m \text{ is even} \quad (6.8) \\
 + & \text{---} \bullet \text{---} \bullet \text{---} \bullet \text{---} \bullet \text{---} \bullet \text{---} \dots \text{---} \bullet \text{---} \bullet \text{---} \bullet \text{---} \bullet \text{---} \bullet \text{---} \dots \\
 + & \text{---} \bullet \text{---} \bullet \text{---} \bullet \text{---} \bullet \text{---} \bullet \text{---} \dots \text{---} \bullet \text{---} \bullet \text{---} \bullet \text{---} \bullet \text{---} \bullet \text{---} \dots
 \end{aligned}$$

which can be brought into the desired form of Lemma 6.1.1.

Getting back to the reason why such CNOT blockades cannot be exploited like in the T-gate case, let us decompose the first star edge in Eq. (6.2) using Eq. (4.13):



Here, the left term can be further simplified to the following:

\equiv

$(ho), (sp)$ =

$= \sqrt{2}$

5: One has to use the (π) -rule in order to handle the X-spider with π -phase.

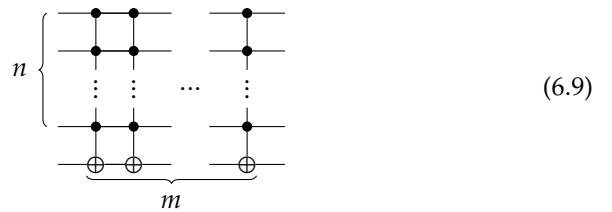
Note that this simplification pattern could continue indefinitely, if more CNOTs and star edges were to be attached. It would always end up with one Z-spider connecting all star edges, thus it is also possible to get two terms for the left term. The right term can be simplified analogously⁵, also resulting in two terms. We can conclude that using the CNOT-grouping technique for star edges does not improve upon traditional methods.

It is worth reminding ourselves that this was only one part of the original paper. The other focus was on a weighting algorithm that takes the graph beyond the immediate neighbors into account. As already mentioned, our own implementation will only use partial simplifications, and therefore we will see that Lemma 6.1.1 is still of importance.

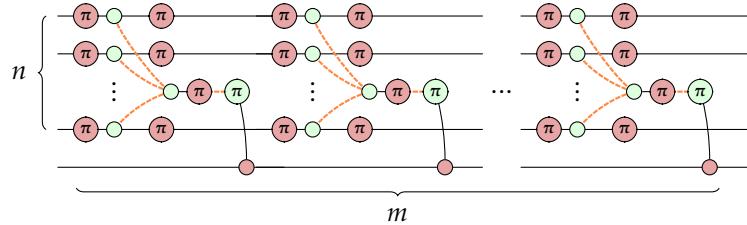
6.2 Studied Approach

6: In fact, since all the gates in this configuration apply a (invertible) NOT-gate on the same qubit, it is in fact trivial. We will still show this example for illustrative purposes.

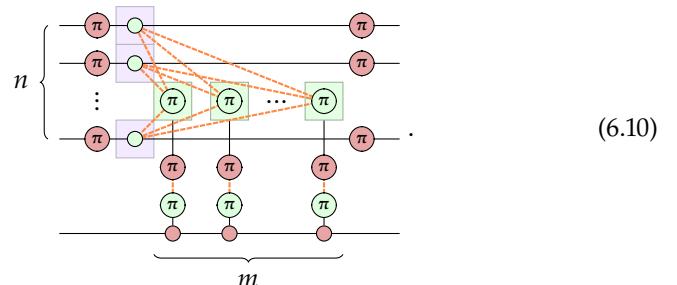
Let us begin by defining the concept of *multi-control Toffoli gate dense quantum circuits*. In general, these are simply quantum circuits, where multi-control Toffoli gates have a high probability of occurring across the whole circuit. We will start off by considering the simplest case⁶



for m multi-control Toffoli gates with n control bits each. According to Eq. (2), we can give the alternative ZX-diagram representation



which can be simplified to



7: In either case, the star edges at the bottom will be simplified automatically.

Here, there exist two possibilities for what we can decompose⁷, highlighted using pink and green boxes. We will refer to these individual spiders as *potential master nodes*. For such simple cases, one can create a simple heuristic: If $n \leq m$, then we decompose all pink potential master nodes, else, if $n > m$, we decompose all green potential master nodes.

Of course, such circuits are in practice of little interest due to their low complexity and because they would only occur in rare cases⁸. In Section 6.3, we will look at a concrete example of a class of quantum circuits with a high density of multi-control Toffoli gates. One pleasant property of this class is that there are no other gates apart from NOT gates⁹.

Taking these properties into account, a weighting algorithm was developed. Contrary to traditional algorithms, it takes neighbors of neighbors into account. Additionally, it considers patterns that could be helpful for further simplifications (see "extra_weight"), something which has not been done in [38]. As mentioned already, we only utilize a partial simplification strategy, that is, the iterative application of (sp) and (c). Furthermore, we make use of certain preliminary simplifications, such as applying the stack representation, which we used for the proof of Lemma 6.1.1. Therefore, this algorithm is specifically meant for quantum circuits only consisting of multi-control Toffoli gates and NOT gates.

The main part of the weighting algorithm is the function `get_master_v_weight`, which is part of the PoC¹⁰ implemented for the concrete application discussed in Section 6.3:

```

1 def get_master_v_weight(self, g, v):
2     extra_weight = 2
3     total_weight = 0
4     for l1_nv in g.neighbors(v):
5         if g.phase(l1_nv) != star_phase:
6             continue
7         total_weight += 1
8         l2_nv = list(g.neighbors(l1_nv))
9         l2_nv.remove(v)
10        if len(l2_nv) > 1:
11            continue
12        l2_nv = l2_nv[0]
13        if g.type(l2_nv) != Z:
14            continue
15        l3_nv = list(g.neighbors(l2_nv))
16        l3_nv.remove(l1_nv)
17        for cur_v in l3_nv:
18            if g.phase(cur_v) == star_phase:
19                total_weight += 1
20            elif not g.type(cur_v) in [Z,X]:
21                continue
22            elif g.vertex_degree(cur_v) > 2:
23                continue
24            else:
25                extra_weight_valid = False
26                if g.type(cur_v) == X and g.phase(cur_v) == 1:
27                    extra_weight_valid = True
28                cur_v = list(g.neighbors(cur_v))
29                cur_v.remove(l2_nv)
30                if len(cur_v) > 1:
31                    continue
32                cur_v = cur_v[0]
33                if g.phase(cur_v) != star_phase:
34                    continue
35                total_weight += (extra_weight if extra_weight_valid
36                                else 1)
37
return total_weight

```

8: Additionally, if they were to occur, such a heuristic is of little use as this would automatically be implemented by a greedy algorithm as described before.

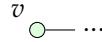
9: Technically, a *diffusion operator* is part of it, but, as we will see, it can be treated separately.

10: The code can be found in [90].

The algorithm assigns a weight to each potential master node, and will then decompose the one with the highest weight. It performs a depth-first search. In other words, it starts with a new branch only after having finished the previous branch. Consider the following visualization:

11: The other branches/edges are not shown.

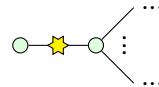
The potential master node v has a branch we want to check¹¹:



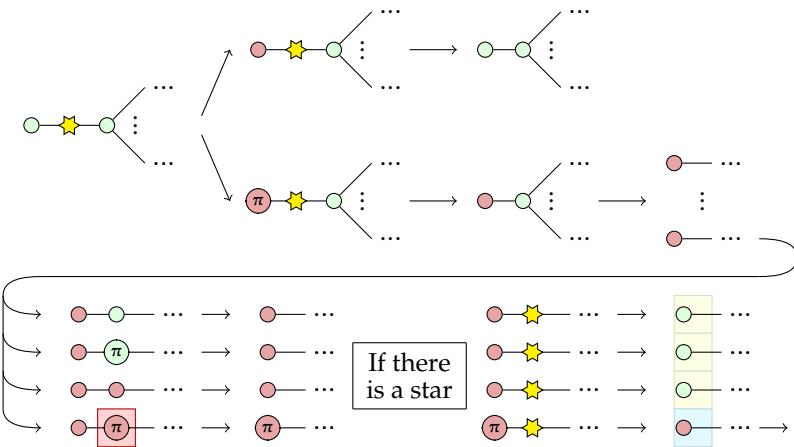
We will only continue searching if the immediate neighbor is a star, in which case the weight is increased by one.



The searching will only continue if there is a Z-spider as its only neighbor, that can potentially have multiple neighbors. It only considers Z-spiders as other patterns were not observed sufficiently enough.



Here, each neighbor is checked. The weight is increased by one if it is a star. Only if the neighbor has a degree of two, the search on that branch is not terminated. If it is an X-spider with phase 0, a Z-spider with phase 0 or π followed by a star, the weight is increased by one. If it is an X-spider with phase π , then the weight is increased by *extra_weight*, which was chosen to be two. The reason for this can be seen in the following illustration, where v is decomposed using Theorem 5.1.1:



In the end, it is clear why an extra weight is useful: An X-spider could remove stars that might follow, whereas a Z-spider could not.

6.3 Application for Multiloop Feynman Diagrams

In a quest to find examples of multi-control Toffoli gate dense quantum circuits with concrete applications, we came across a whole class of such circuits in [91]. One such example can be seen in Figure 6.3.

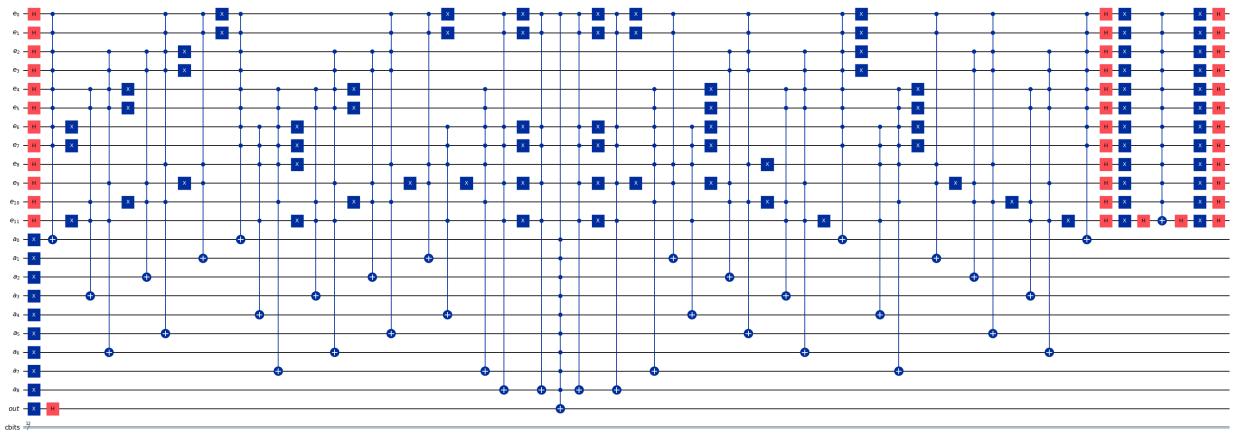
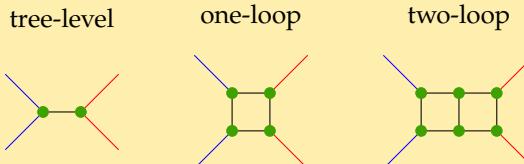


Figure 6.3: Multi-control Toffoli gate dense quantum circuit found in [91].

Such quantum circuits implement a quantum algorithm based on *Grover's algorithm*, a famous quantum search algorithm, which beats the classical space-time complexity of $\mathcal{O}(N)$, only requiring $\mathcal{O}(\sqrt{N})$, for N being the number of elements searched. More concretely, it searches for *causal singular configurations of multiloop Feynman diagrams* [91]. The exact details of the theory behind the algorithm lie beyond the scope of this thesis. However, the following will provide a brief overview.

In Chapter 3, we very briefly introduced QFT and scattering processes. Calculating the scattering amplitude \mathcal{A} of an experiment conducted using a particle accelerator is one of the cornerstones of modern particle physics. One common approach is to use perturbation theory, which expresses the amplitude as a power series, where each term corresponds to increasingly more complex interactions during the scattering process. This allows physicists to approximate scattering processes by considering only the most significant contributions, typically the first few terms in the series [92]. These terms can be represented and computed as Feynman diagrams. However, such computations get cumbersome for more complex *topologies*, in other words, when the number of *external legs* in the Feynman diagrams increases, or when the *loop order* increases [93].

Example 6.3.1 The following illustrates the loop order, increasing from left to right:



In this case, the number of external legs (colored red and blue) was kept constant. The number of couplings (colored green) did change, however [94].

12: Of course, all the following definitions are only listed for completeness, and the reader is not expected to give them more thought. Interested readers should refer to [92], [91] and [95].

It is therefore an active area of research to find more efficient methods for such computations. One proposed alternative is based on the *loop-tree duality formalism*. The basic idea is to reinterpret complicated loop diagrams in terms of simpler tree-level diagrams [93]. More mathematically¹², one would start with *generic loop integrals* and *scattering amplitudes*, which are written as

$$\mathcal{A}_F^{(L)} = \int_{\ell_1 \dots \ell_L} \mathcal{N} \left(\{\ell_s\}_L, \{p_j\}_P \right) \prod_{i=1}^n G_F(q_i) \quad (6.11)$$

where n is the number of propagators, P is the number of external legs, L is the number of loop momenta, ℓ_s with $s \in \{1, \dots, L\}$ are the primitive loop momenta, p_j with $j \in \{1, \dots, P\}$ are the external momenta, q_i are the momenta flowing through the Feynman propagator G_F and \mathcal{N} is the numerator given by the specific theory being used (cf. [96]).

By applying *Cauchy's residue theorem*, it is then possible to obtain the *loop-tree duality causal representation* [91], which is written as

$$\mathcal{A}_D^{(L)} = \int_{\vec{\ell}_1 \dots \vec{\ell}_L} \frac{1}{x_n} \sum_{\sigma \in \Sigma} \frac{\mathcal{N}_{\sigma(i_1, \dots, i_{n-L})}}{\lambda_{\sigma(i_1)}^{h_{\sigma(i_1)}} \dots \lambda_{\sigma(i_{n-L})}^{h_{\sigma(i_{n-L})}}} + (\lambda_p^+ \leftrightarrow \lambda_p^-), \quad (6.12)$$

where, x_n is a normalization factor, a given λ is a causal propagator and a given σ corresponds to a specific permutation, which is an element of the set Σ , which in turn is determined by the causal configurations. This set is therefore crucial to ensure that only physical contributions are included in the calculation.

Although the details are very advanced, the essence of Cauchy's residue theorem can be explained using a simple example, taken from [97].

Theorem 6.3.2 [97] Suppose $f(z)$ is analytic in the region A except for a set of isolated singularities. Also suppose C is a simple, closed curve in A that does not go through any of the singularities of f and is oriented counterclockwise. Then

$$\int_C f(z) dz = 2\pi i \sum_{k=1}^n \text{residue of } f \text{ at } a_k, \quad (6.13)$$

where the $a_k \in \mathbb{C}$ are the poles of the complex function f lying inside C .

Example 6.3.3 We want to compute the following integral:

$$\int_{|z|=2} \frac{5z-2}{z(z-1)} dz.$$

Let

$$f(z) = \frac{5z-2}{z(z-1)}.$$

In order to use Theorem 6.3.2, we need to find the residue of f at each of its poles. If f has a *simple pole*, then we can use a relatively simple formula for the residue [92], given by

$$R(z_0) = \lim_{z \rightarrow z_0} (z - z_0) f(z). \quad (6.14)$$

In our case we have two simple poles at $z = 0$ and $z = 1$, which, when plugged into f , would result in undefined behavior. Theorem 6.3.2 requires us only to

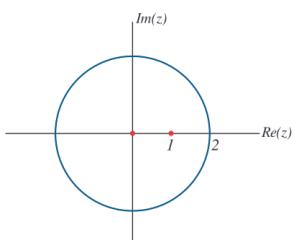


Figure 6.4: Contour of f taken from [97]. Note that the contour has to be considered in counterclockwise direction.

use the residue at the poles lying inside the contour, which is true for both, as illustrated in Figure 6.4.

At $z = 0$, we have

$$R(0) = \lim_{z \rightarrow 0} (z - 0) \frac{5z - 2}{z(z - 1)} = \lim_{z \rightarrow 0} \frac{5z - 2}{z - 1} = 2.$$

And at $z = 1$, we have

$$R(0) = \lim_{z \rightarrow 1} (z - 1) \frac{5z - 2}{z(z - 1)} = \lim_{z \rightarrow 1} \frac{5z - 2}{z} = 3.$$

We can now apply Theorem 6.3.2:

$$\int_C \frac{5z - 2}{z(z - 1)} dz = 2\pi i [R(0) + R(1)] = 10\pi i.$$

That being said, obtaining the causal representation in Eq. (6.12) requires finding causal configurations. This is exactly what the algorithm searches for using the proposed quantum circuits.

The paper outlines how to create such a quantum circuit corresponding to a given topology using certain boolean functions. For example, the two eloop topology with six edges corresponds to the following quantum circuit:

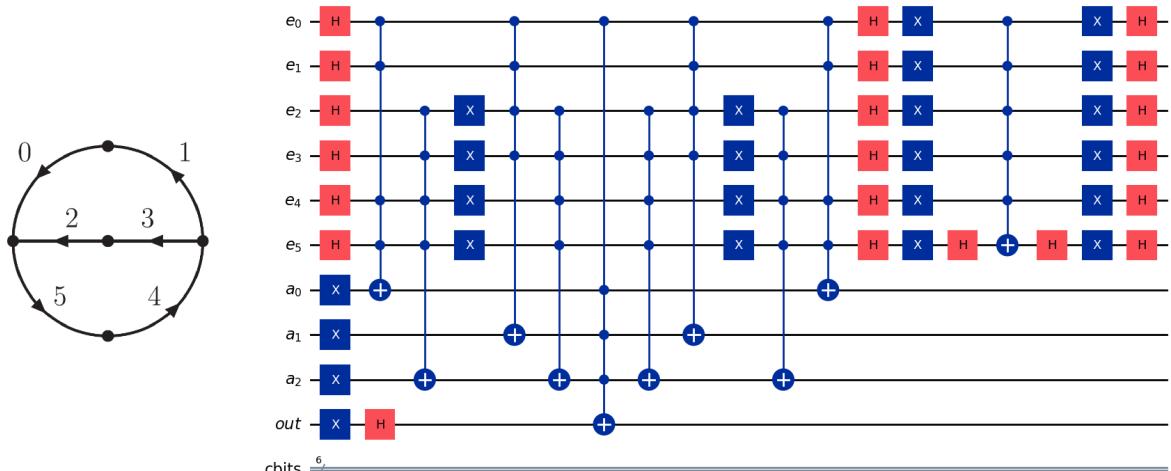


Figure 6.5: Two eloop topology with six edges (taken from [91]) converted to a quantum circuit (generated in [90]).

Note that the distinct right part of the quantum circuit¹³ is the *diffusion operator*, a key aspect of Grover's algorithm [98].

In brief, Grover's algorithm is a *quantum query algorithm* for unstructured search. This means that the search algorithm is oracle-based: Given a function satisfying certain properties, we want to recover some information about the function¹⁴. This should be done using a minimal number of queries to the corresponding *quantum oracle*¹⁵.

13: We are referring to the right-most multi-control Toffoli gate surrounded by Hadamard and NOT-gates.

14: But only with a certain probability.

15: This is similar to how the Oracle at Delphi would never reveal everything she knows, but only what she is being asked.

Problem statement for Grover's algorithm

Given a function $f : \{0, 1\}^n \rightarrow \{0, 1\}$, we can query it using the following quantum oracle given by the unitary operator U_ω :

$$\begin{cases} U_\omega|x\rangle = -|x\rangle & \text{for } x = \omega, \text{ that is, } f(x) = 1 \\ U_\omega|x\rangle = |x\rangle & \text{for } x \neq \omega, \text{ that is, } f(x) = 0 \end{cases}$$

Alternatively, we can write

$$U_\omega|x\rangle = (-1)^{f(x)}|x\rangle. \quad (6.15)$$

We assume that only one x satisfies $f(x) = 1$. The goal is to identify ω with high probability [98].

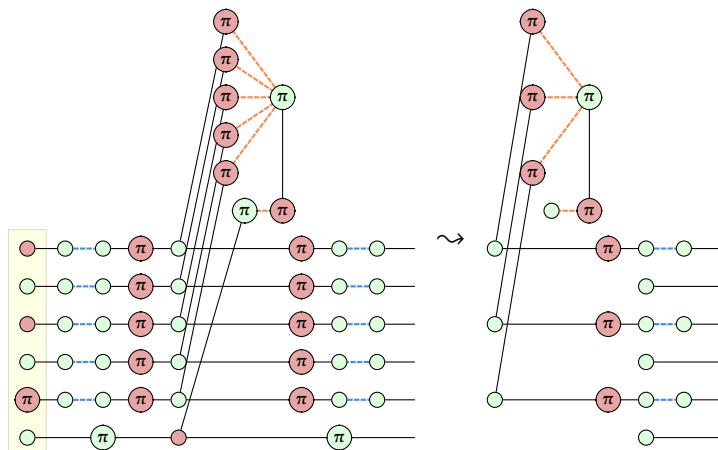
It can be seen that Eq. (6.15) is analogous to the oracle defined in [91], which is given by

$$U_\omega|e\rangle|a\rangle|out\rangle = (-1)^{f(x)}|e\rangle|a\rangle|out\rangle$$

Grover's algorithm consists of the quantum oracle and the diffusion operator, which are applied t times on a superposition of all x . With high probability, we can say that the resulting state is ω . The algorithm presented in [91] exhibits certain nuances. For example, they demonstrated that the most effective results for most multiloop topologies are obtained when $t = 1$. This means that we can simulate the quantum circuits shown in Figure 6.3 and Figure 6.5 as they are, without the need to repeat the circuits.

The quantum circuit in Figure 6.5 alongside with the quantum circuit in Figure 6.3, which corresponds to a four eloop topology with twelve edges, were implemented using Qiskit [28] in order to verify the results from [91]. Indeed, 23 causal configurations were found for the two eloop topology, and 1199 causal configurations for the four eloop topology, matching the obtained values from the paper. The implementation can be found in [90].

The Proof-of-Concept for our algorithm described in Section 6.2 was implemented in [90] as well. The program uses a ZX-diagram representation of the two quantum circuits from before. It then applies the algorithm on the main part, that is, excluding the diffusion operator. When it arrives at a fully simplified Clifford representation of all terms, highlighted in yellow, it will connect it to the diffusion operator. This will then get simplified and decomposed if necessary. An example can be seen here:



In this case, a decomposition would be necessary.

Once we are left with only Clifford terms, we have found our final decomposition. All terms will be converted to statevectors, which will then get summed up, in order to obtain the final statevector. We can compute the probability for each state as described in Chapter 2. The resulting probability distribution can be seen in Figure 6.6.

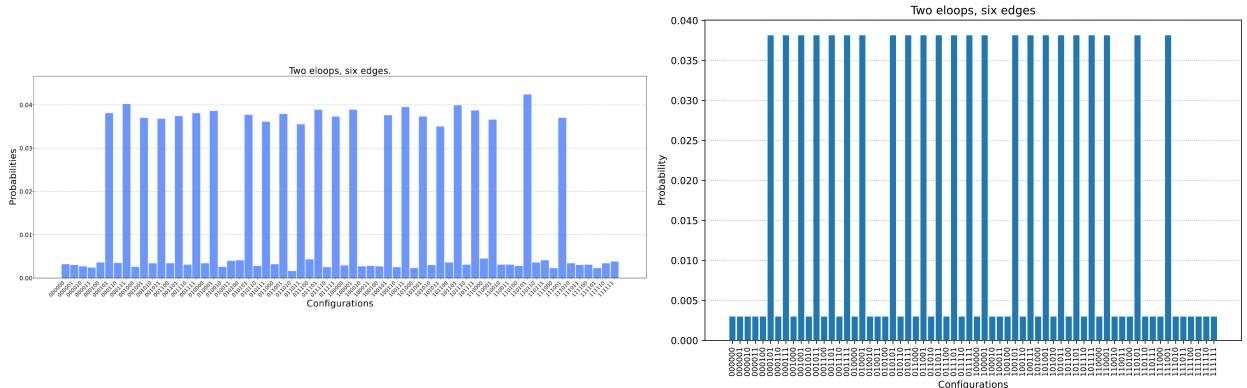


Figure 6.6: Left: Probability distribution from [91]. Right: Probability distribution generated using [90]. Both are for the two eloop topology with six edges.

By counting the number of peaks¹⁶, one can determine the number of causal configurations. Indeed, the plot matches the probability distribution from the original paper.

Readers interested in the steps taken before applying the weighting algorithm should refer to the Jupyter notebook in [90], where by setting the first parameter of `qalgo.run(show_diagrams=True, check_decomposition=True)` to `True`, one can see each major step of the program graphically. In brief, all the NOT gates in the quantum circuit are pushed to the sides by applying the (c)-rule. Since this can alter the form of the multi-control Toffoli gates¹⁷, we apply the stack representation described in Eq. (6.6), used for the proof of Lemma 6.1.1. This makes the weighting process easier, as all the edges of interest connected to the potential master nodes are of the same form¹⁸. Finally, the (c)-rule is applied on all present states as a preliminary simplification routine.

It should be noted, however, that the implementation utilizes W nodes (*cf.* Section 3.2) to represent the stars, and placeholder numbers as phases, which is a programmatic trick to mark potential master nodes. This is deprecated, and was therefore replaced in the subsequent implementation. The better alternative is to use zero-labelled H-boxes (*cf.* Eq. (4.19)) and `vdata`. Both features can be easily implemented using PyZX [99], the main library used for both implementations.

Now, in order to compare our weighting algorithm to the algorithm developed in [80], the quizx version by Mark Koch was modified to also work with non-scalar diagrams. This made it possible to decompose the quantum circuits for the two and four eloop topologies using their algorithm. The implementation can also be found in [90].

16: This was done computationally by calculating a threshold given by the average of the maximum and the minimum probability, and then checking which states were above this threshold.

17: It will create a structure similar to the one found in Eq. (6.5).

18: This will result in two stacks per row. Since a decomposition applied to one of the two stacks causes the same effect on the other, the two weights are treated as one weight. Readers interested in more details should refer to the code implementation.

The measured runtimes compare as follows:

	Modified quizx	Our implementation
Two eloop topology with six edges	0.0111416 s	≈ 1.2 s (0.035086 s for weighting algorithm)
Four eloop topology with twelve edges	2.0793388 s	≈ 81.9 s (1.807702 s for weighting algorithm)

It is clear that in this domain, no improvements could be made. This outcome is anticipated, given that our implementation is written in Python, utilizing PyZX as the main library, whereas the modified quizx implementation is written in Rust, utilizing quizx. Both Rust and quizx are known to achieve much faster performance than their Python counterpart [100] [89]. For instance, certain graph manipulations¹⁹ in quizx might be 5700 times faster than the corresponding implementation in PyZX [89]. Additionally, our implementation is a very unoptimized Proof-of-Concept.

As mentioned in [38], it is difficult to give an exact formula for the space-time complexity of such weighting algorithms, but the rough upper-bound will be similar to the polynomial one from the original paper, which should be insignificant in comparison to the exponential process of producing stabilizer terms. This also matches the measurements of the time taken only for the weighting algorithm. We conclude that the obtained results were expected and are thus not unsatisfactory.

We are primarily interested in the final count of stabilizer terms, as it is the only valid metric for concrete comparisons, being independent of implementation details and thus ensuring an objective evaluation. The obtained results are as follows:

	Modified quizx	Our implementation
Two eloop topology	52 terms	48 terms
Four eloop topology	1810 terms	1948 terms

Using our algorithm, it was indeed possible to improve the final number of terms for the simpler topology, although only by a rather small margin. For the more complex topology, the modified quizx version outperformed our algorithm by a similar relative margin. We suspect that the main reason for not getting better results is our partial simplification strategy. The modified quizx version utilizes a full simplification strategy. Since the smaller topology corresponds to a smaller quantum circuit, simplifications will not have a great influence. For bigger topologies and thus also bigger circuits, simplifications can outweigh the expected improvement of a weighting algorithm. In Section 6.4, we will see compelling evidence for this hypothesis by considering randomly generated quantum circuits with similar properties to those studied in this section.

19: The part that took the longest to execute in the four eloop topology case was the translation of ZX-diagrams into statevectors and subsequent vector computations.

6.4 Importance of Simplification Strategies

At the beginning of Section 6.1, we outlined the reasoning behind our decision to utilize a partial simplification strategy instead of full simplification approach. The goal was to mimic certain aspects of the original algorithm introduced in [38]. It was thought that their reasoning would also apply to our case, namely that full simplification strategies could destroy potentially useful patterns. Our partial decomposition strategy was therefore able to use the idea of stacks, as described in the last section. In [38], the authors speculated that more advanced simplification routines could have further contributed to reducing the final number of terms, but due to a lack of evidence for this speculation, we decided to start with partial simplifications.

Despite this, in the last section we have seen preliminary evidence that our partial simplification strategy hinders us from getting a reliable improvement over the modified quizx version. To strengthen the case for our hypothesis, we will discuss a benchmark in the subsequent discussion.

The benchmark will apply our algorithm and the modified quizx version on randomly generated multi-control Toffoli gate dense quantum circuits with variable amounts of NOT gates, CNOT gates and multi-control Toffoli gates²⁰. Additionally, the number of qubits will be varied. Each configuration will be sampled 50 times with different random seeds. If the execution of either algorithm takes more than three minutes, it will be interrupted. Although the benchmark used parallelization, the computation took more than 20 hours.

20: The target bits of the multi-control Toffoli gates are distributed over the bottom qubits, in order to create a similar pattern to those found in the quantum circuits from Section 6.3.

Remark 6.4.1 For the two considered topologies from Section 6.3 we checked that no star decompositions had affected the results. This benchmark, on the other hand, occasionally resulted in star decompositions being used. Since this gives a more realistic comparison between our algorithm and the state-of-the-art approach, we still accepted it. Additionally, it should be noted that the benchmark uses scalar diagrams, in order to compare our algorithm to the original quizx version by Mark Koch, and not to the modified version.

Due to the three minute timeout, certain configurations contain less than 50 samples. The statistical relevance can therefore be distinguished according to the marker used in the following plots: a cube for (relatively) high relevance, a big circle for medium relevance, a small circle for low relevance and a cross for no sample obtained.

Two plots were generated. The first one shows the average performance ratio per datapoint. This represents the ratio of the final number of terms given by quizx to the final number of terms given by our algorithm. Therefore, a high performance ratio is favorable. The second plot shows the percentage of improvements relative to the total number of relevant samples collected.

The source code for the benchmark can be found in [101].

The results can be seen in Figure 6.7 and in Figure 6.8.

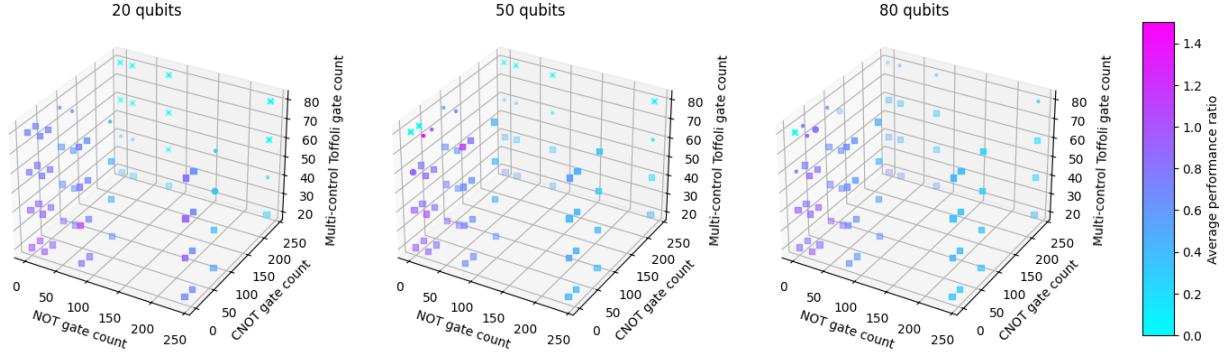


Figure 6.7: Benchmark results regarding average performance ratio, generated using [101].

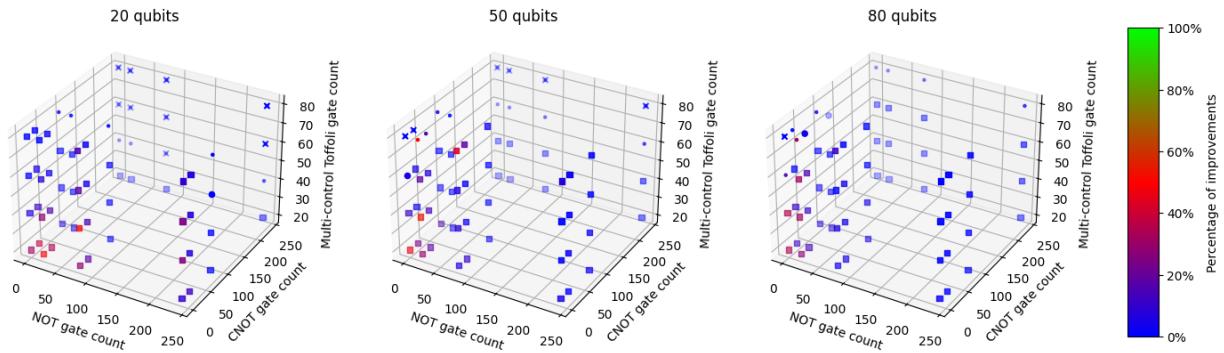


Figure 6.8: Benchmark results regarding percentage of improvements, generated using [101].

Examining Figure 6.7, we can see that overall, there are only a few occurrences where the average performance ratio is greater than one. Nevertheless, we can see that according to Figure 6.8, our algorithm is *capable* of delivering an advantage across various configurations, at times yielding improvements for up to 50% of the samples.

21: We are referring to the configurations that have either 0 or 240 NOT gates, and 240 or 0 CNOT gates, respectively.

By looking at the vertical edges of the graphs²¹, one can see that a high NOT count combined with a low CNOT count yields on average a better performance than a high CNOT count combined with a low NOT count. This aligns with our hypothesis, as partial simplifications can handle NOT gates relatively well, whereas only a full simplification strategy could reduce the added complexity of CNOT gates effectively.

Overall, low counts of NOT and CNOT gates tend to yield the best results. Furthermore, increasing the number of multi-control Toffoli gates does not strongly affect the average performance ratio. This is expected, since to our algorithm, increasing this number simply means increasing the degree of the potential master nodes, assuming obstructions are handled sufficiently well by the simplification strategy.

Our algorithm was able to yield improvements for particularly many configurations in the 20 qubit case. This can be seen as further support for our hypothesis, as more qubits allow for more simplifications to apply.

Finally, Figure 6.8 shows a very similar distribution to Figure 6.7, which demonstrates that there were no significant outliers affecting the average performance ratio.

To see what it would mean to apply a full simplification strategy, we first have to clarify what we mean by a full simplification. PyZX offers two main simplification routines, the first one being *clifford_simp* and the second one being *full_reduce*. The latter applies *clifford_simp* alongside certain routines that make use of *gadgetization* [99]. As the quizx implementation we are comparing against utilizes *clifford_simp* and furthermore *full_reduce* does not work well with H-boxes that are used to represent the star edges, we will use *clifford_simp* for the following demonstration.

Consider the following randomly generated 17-qubit scalar diagram in Figure 6.9, consisting of 25 NOT and CNOT gates, and six multi-control Toffoli gates.

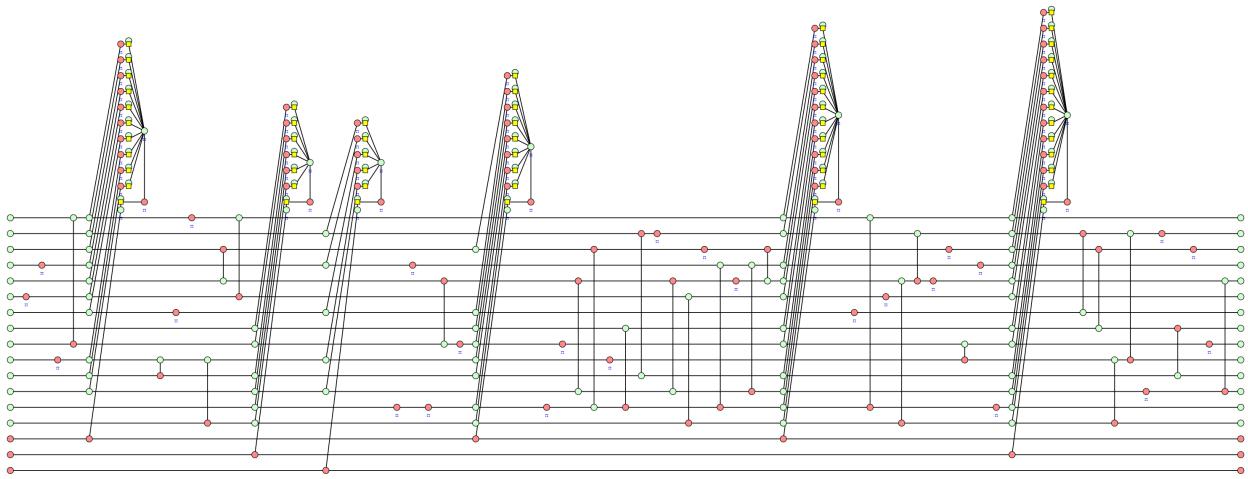


Figure 6.9: The initial 17-qubit scalar diagram before applying any simplification strategy, generated in [101].

After applying our partial simplification strategy employed thus far, which includes an unsuccessful attempt at creating two stacks, we get the diagram in Figure 6.10.

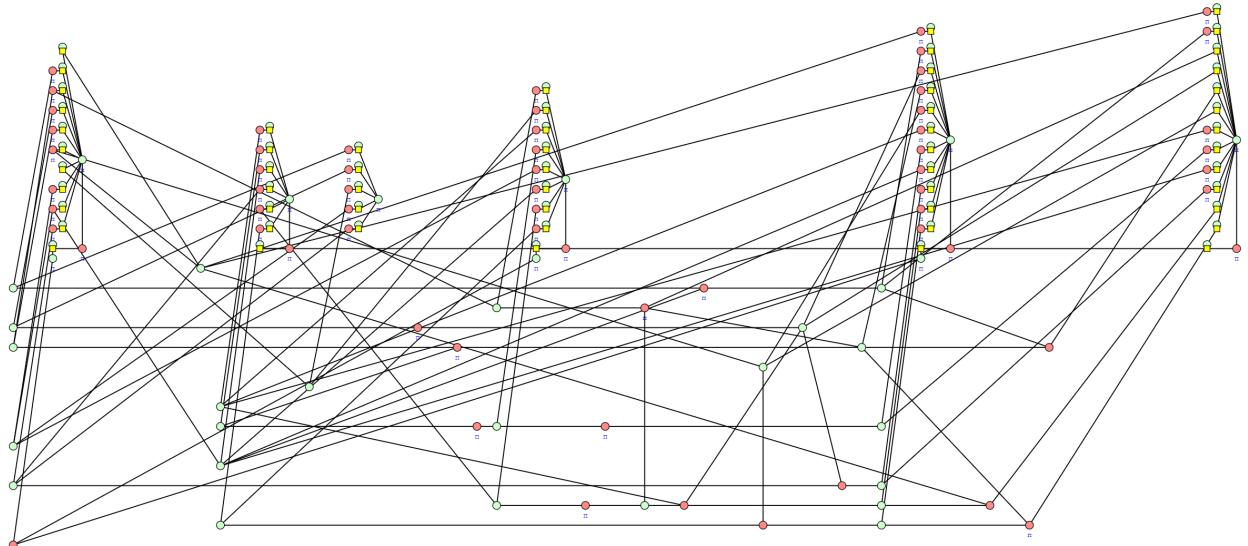


Figure 6.10: The 17-qubit scalar diagram after applying our partial simplification strategy, generated in [101].

Comparing this to the diagram obtained in Figure 6.11 after applying `clifford_simp`, we can observe significant differences.

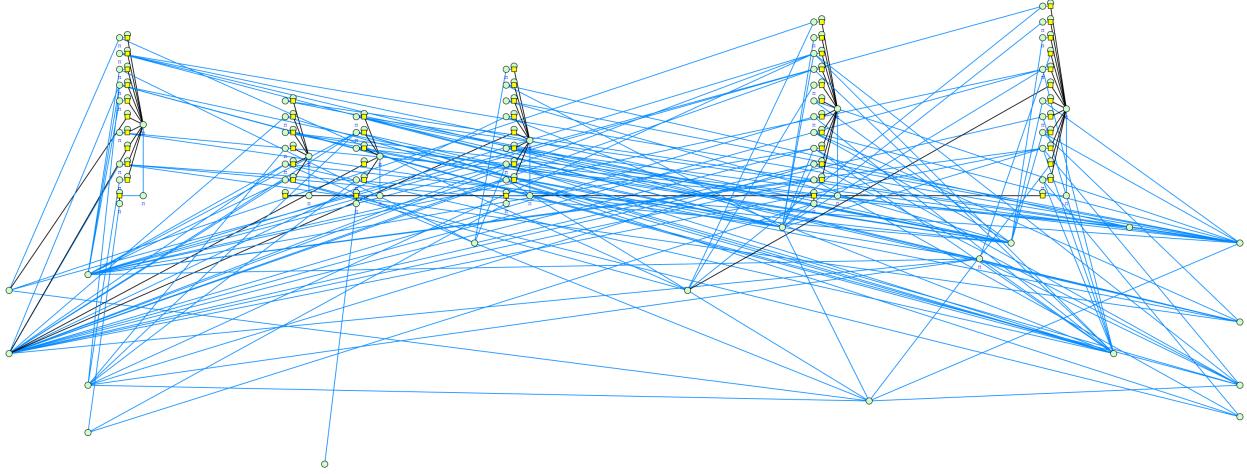


Figure 6.11: The 17-qubit scalar diagram after applying the full simplification strategy, generated in [101].

Most notably, the new diagram contains a lot more Hadamard edges. Additionally, there is a lot more clustering. In other words, this means that there are less spiders, but higher vertex degrees. This is very important, as it allows a weighting algorithm to make a better prediction due to a bigger coverage of the diagram, whilst not having to change the search depth. Nevertheless, this might require clarification, as it is easy to think that our partial simplification performs very poorly. Concretely, our simplification strategy was made for a class of quantum circuits capable of searching for causal configurations, for which we know that CNOT gates can practically never occur. This is why the diagram in Figure 6.10 is only poorly simplified. We added a lot of CNOT gates in order to artificially increase the complexity of the circuit²².

22: Instead of seeing the addition of CNOT gates as a tool to change the complexity without having to change the number of qubits, we could see this as changing the class we initially set out to study, in order for it to now include different gates.

23: This would again require a detailed analysis of post-simplification patterns.

Given this evidence, we conclude that changing our algorithm such that it is capable of obtaining reliable improvements in the final number of terms would require the use of `clifford_simp`. This would then require changes to the weighting algorithm, as it was not intended to be used with Hadamard edges. A similar analysis to the one conducted in Section 6.2 would need to reveal which patterns (that may include Hadamard edges) can be attributed to which weight. Ideally, this would be implemented in our modified quizx version, in order to make the obtained speed-ups more practical and easier to compare. Additionally, this would also allow the embedding of star decompositions and potentially even state decompositions into the weighting algorithm²³. Finally, it is worth mentioning that it is not clear how big the actual improvement would be if we were to use such a refined algorithm. As of right now, the improvements are significantly less than the ones observed in [38]. This could be a general property of our algorithm, or it might be because the benchmarks were conducted in a certain regime that is not suitable to demonstrate bigger improvements, similar to how in Figure 6.1 and Figure 6.2 the obtained improvements for low T-counts is relatively small. In either case, it would be interesting to investigate the effects of increasing the search depth. Since in practice the cost of searching is not negligible, it might be an option to start with a big search depth and then only use more shallow searches afterwards. This might prevent going down a suboptimal branch in the *tree* of possible vertex decompositions.

PART IV: CONCLUSION

Conclusion

7

7.1 Summary

In this thesis, we have explored two primary research directions. The first direction concerns the state decompositions of non-stabilizer states made up of star edges, a topic previously studied in the literature [80] [44]. Using simulated annealing, we were able to find five novel decompositions: three 5-to-6 decompositions and two 4-to-5 decompositions, with phases constrained to 0 , $\frac{\pi}{2}$ or $-\frac{\pi}{2}$. We tested the software [84] on an AWS EC2 C6a instance [86] and observed that results were either found relatively quickly, or not at all. Therefore, we could not take advantage of the high-performance computer.

The second direction focused on dynamic decompositions and their application to weighting algorithms, drawing inspiration from [38]. We demonstrated that their proposed idea of CNOT-grouping for the reduction of T-gates cannot be transferred to the reduction of star edges. Nevertheless, we described a new weighting algorithm designed for multi-control Toffoli gate dense quantum circuits. This algorithm was tailored to a specific class of quantum circuits, which is used to find causal configurations of multiloop Feynman diagrams. Thus, the algorithm only needs to take special care of NOT-gates, which is accomplished by using a "stack" representation, that we introduced whilst studying the CNOT-grouping technique.

Our implementation successfully simulated the two tested topologies. A comparison to the implementation from [80] revealed that our algorithm could only provide an improvement in the final number of terms for the simpler topology. We hypothesized that this is due to our partial simplification strategy, whereas they used a full simplification strategy.

In order to test our hypothesis, we generated 192 different types of random quantum circuits and sampled each type 50 times. The resulting data seems to support our hypothesis.

7.2 Outlook

The results from working with a high-performance computer seem to suggest that future work on methods to find novel state decompositions would require the use of a different approach than simulated annealing.

Similarly, the benchmarks of our weighting algorithm provide convincing evidence that future work should primarily consider using full simplification strategies. Additionally, it would be interesting to see the effects of increasing the search depth, at least in the beginning of the procedure.

Whilst finishing this project, we made a final discovery: It turns out that the quizz implementation from [80] can produce reliable improvements when not using star decompositions. In other words, using star decompositions makes sense for certain quantum circuits such as those used for barren plateau detection, but might be disadvantageous for the tested multi-control Toffoli gate dense

7.1 Summary	65
7.2 Outlook	65

quantum circuits. However, it might be possible that a more clever algorithm could embed star decompositions such that no disadvantages are created, and potentially even create advantages by incorporating it into a weighting algorithm. More information can be found in [102].

Bibliography

Here are the references in citation order.

- [1] American Institute of Physics. *Richard Feynman - Session IV*. Jan. 27, 2015. URL: <https://www.aip.org/history-programs/nie.../oral-histories/5020-4> (visited on 12/18/2024) (cited on page 1).
- [2] David Kaiser. 'Physics and Feynman's Diagrams'. In: (2005) (cited on page 1).
- [3] Bob Coecke and Ross Duncan. 'Interacting Quantum Observables'. In: *Automata, Languages and Programming*. Ed. by Luca Aceto et al. Vol. 5126. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 298–310. doi: [10.1007/978-3-540-70583-3_25](https://doi.org/10.1007/978-3-540-70583-3_25) (cited on pages 1, 20).
- [4] Hector J. Garcia, Igor L. Markov, and Andrew W. Cross. *Efficient Inner-product Algorithm for Stabilizer States*. Aug. 7, 2013. doi: [10.48550/arXiv.1210.6646](https://arxiv.org/abs/1210.6646). Pre-published (cited on page 1).
- [5] Aleks Kissinger, John van de Wetering, and Renaud Vilmart. 'Classical Simulation of Quantum Circuits with Partial and Graphical Stabiliser Decompositions'. In: (2022), 13 pages, 867977 bytes. doi: [10.4230/LIPICS.TQC.2022.5](https://doi.org/10.4230/LIPICS.TQC.2022.5) (cited on pages 1, 25, 30, 31).
- [6] *Solvay_conference_1927*. URL: https://upload.wikimedia.org/wikipedia/commons/6/6e/Solvay_conference_1927.jpg (visited on 09/15/2024) (cited on page 5).
- [7] *Solvay Conference*. In: *Wikipedia*. Aug. 8, 2024. (Visited on 09/15/2024) (cited on page 5).
- [8] Michael A. Nielsen and Isaac L. Chuang. *Quantum Computation and Quantum Information: 10th Anniversary Edition*. 1st ed. Cambridge University Press, June 5, 2012 (cited on pages 5–8, 13–17).
- [9] *Ultraviolet Catastrophe*. In: *Wikipedia*. May 18, 2024. (Visited on 09/15/2024) (cited on page 5).
- [10] *18.2: Brief Summary of the Origins of Quantum Theory*. Physics LibreTexts. Nov. 11, 2017. URL: [https://phys.libretexts.org/Bookshelves/Classical_Mechanics/Variational_Principles_in_Classical_Mechanics_\(Cline\)/18%3A_The_Transition_to_Quantum_Physics/18.02%3A_Brief_summary_of_the_origins_of_quantum_theory](https://phys.libretexts.org/Bookshelves/Classical_Mechanics/Variational_Principles_in_Classical_Mechanics_(Cline)/18%3A_The_Transition_to_Quantum_Physics/18.02%3A_Brief_summary_of_the_origins_of_quantum_theory) (visited on 09/15/2024) (cited on page 5).
- [11] *Postulate*. In: *Simple English Wikipedia, the Free Encyclopedia*. Jan. 6, 2024. (Visited on 09/16/2024) (cited on page 6).
- [12] Aleks Kissinger and John van de Wetering. *Picturing Quantum Software: An Introduction to the ZX-calculus and Quantum Compilation*. Preprint, 2024 (cited on pages 6, 17, 20–22).
- [13] *Wave Function*. In: *Wikipedia*. Aug. 18, 2024. (Visited on 10/16/2024) (cited on page 7).
- [14] Juan Rojo. 'QuantumMechanics2-LectureNotes'. In: *Quantum Mechanics* (2021) (cited on page 7).
- [15] *Dirac Delta Function*. In: *Wikipedia*. Oct. 13, 2024. (Visited on 10/16/2024) (cited on page 7).
- [16] *Operator (Physics)*. In: *Wikipedia*. July 21, 2024. (Visited on 10/15/2024) (cited on page 8).
- [17] *Schrödinger Equation*. In: *Wikipedia*. Aug. 10, 2024. (Visited on 09/17/2024) (cited on page 9).
- [18] *Particle in a Box*. In: *Wikipedia*. July 3, 2024. (Visited on 09/19/2024) (cited on page 10).
- [19] *A Graphical Representation of the 'Particle in a Box' as Solution Of...* ResearchGate. URL: https://www.researchgate.net/figure/A-graphical-representation-of-the-particle-in-a-box-as-solution-of-the-time-independent_fig5_337259777 (visited on 09/21/2024) (cited on page 11).
- [20] Richard P. Feynman. 'Simulating Physics with Computers'. In: *International Journal of Theoretical Physics* 21.6 (June 1, 1982), pp. 467–488. doi: [10.1007/BF02650179](https://doi.org/10.1007/BF02650179) (cited on page 11).
- [21] *Quantum Decoherence*. In: *Wikipedia*. July 9, 2024. (Visited on 09/22/2024) (cited on page 12).
- [22] *Bloch Sphere*. In: *Wikipedia*. July 16, 2024. (Visited on 09/22/2024) (cited on page 13).
- [23] *Time Complexity*. In: *Wikipedia*. Aug. 11, 2024. (Visited on 09/29/2024) (cited on page 16).

- [24] *Shor's Algorithm*. In: *Wikipedia*. Sept. 12, 2024. (Visited on 09/29/2024) (cited on page 16).
- [25] Daniel Gottesman. *The Heisenberg Representation of Quantum Computers*. July 1, 1998. URL: <http://arxiv.org/abs/quant-ph/9807006> (visited on 05/04/2024). Pre-published (cited on page 17).
- [26] Sieglinde M.-L. Pfaendler, Konstantin Konson, and Franziska Greinert. 'Advancements in Quantum Computing—Viewpoint: Building Adoption and Competency in Industry'. In: *Datenbank-Spektrum* 24.1 (Mar. 1, 2024), pp. 5–20. doi: [10.1007/s13222-024-00467-4](https://doi.org/10.1007/s13222-024-00467-4) (cited on page 17).
- [27] Xiaosi Xu et al. *A Herculean Task: Classical Simulation of Quantum Computers*. Feb. 17, 2023. doi: [10.48550/arXiv.2302.08880](https://doi.org/10.48550/arXiv.2302.08880). Pre-published (cited on page 17).
- [28] Ali Javadi-Abhari et al. *Quantum Computing with Qiskit*. 2024. doi: [10.48550/arXiv.2405.08810](https://doi.org/10.48550/arXiv.2405.08810) (cited on pages 17, 56).
- [29] *FeynmanDsonVan.Jpg* (JPEG Image, 800 × 529 Pixels). URL: <http://woodahl.physics.iupui.edu/Astro105/FeynmanDsonVan.jpg> (visited on 10/16/2024) (cited on page 19).
- [30] R. P. Feynman. 'Space-Time Approach to Quantum Electrodynamics'. In: *Physical Review* 76.6 (Sept. 15, 1949), pp. 769–789. doi: [10.1103/PhysRev.76.769](https://doi.org/10.1103/PhysRev.76.769) (cited on page 19).
- [31] Michael H Seymour. *The Meaning of Feynman Diagrams*. URL: <https://indico.cern.ch/event/421552/sessions/170249/attachments/884224/1242865/feynman.pdf> (visited on 10/16/2024) (cited on page 19).
- [32] *Figure 2. The Four Different Regions of the Penrose Diagram*. ResearchGate. URL: https://www.researchgate.net/figure/The-four-different-regions-of-the-Penrose-diagram_fig2_346519045 (visited on 10/16/2024) (cited on page 19).
- [33] *Logic Gate*. In: *Wikipedia*. Oct. 15, 2024. (Visited on 10/16/2024) (cited on page 19).
- [34] *Penrose Graphical Notation*. In: *Wikipedia*. Sept. 9, 2024. (Visited on 10/16/2024) (cited on page 19).
- [35] *Quantum Field Theory*. In: *Wikipedia*. Oct. 8, 2024. (Visited on 10/16/2024) (cited on page 19).
- [36] *Penrose Diagram*. In: *Wikipedia*. Aug. 30, 2024. (Visited on 10/16/2024) (cited on page 19).
- [37] John van de Wetering. *ZX-calculus for the Working Quantum Computer Scientist*. Dec. 27, 2020. doi: [10.48550/arXiv.2012.13966](https://doi.org/10.48550/arXiv.2012.13966). Pre-published (cited on pages 20, 22, 24).
- [38] Matthew Sutcliffe and Aleks Kissinger. *Procedurally Optimised ZX-Diagram Cutting for Efficient T-Decomposition in Classical Simulation*. Mar. 16, 2024. URL: <http://arxiv.org/abs/2403.10964> (visited on 05/27/2024). Pre-published (cited on pages 21, 45, 51, 58, 59, 62, 65).
- [39] Richard D. P. East, Pierre Martin-Dussaud, and John Van de Wetering. *Spin-Networks in the ZX-calculus*. Nov. 18, 2022. URL: <http://arxiv.org/abs/2111.03114> (visited on 10/18/2024). Pre-published (cited on pages 21, 24, 25).
- [40] Romain Moyard. 'Introduction to the ZX-calculus'. In: *PennyLane Demos* (June 6, 2023). (Visited on 10/19/2024) (cited on page 21).
- [41] Boldizsar Poor. *A Unique Normal Form for Prime-Dimensional Qudit Clifford ZX-calculus*. URL: https://www.cs.ox.ac.uk/michael.benedikt/msctheses/2022/1059275_97382463_1.pdf (cited on page 22).
- [42] Tom Peham, Lukas Burgholzer, and Robert Wille. 'Equivalence Checking of Quantum Circuits With the ZX-Calculus'. In: *IEEE Journal on Emerging and Selected Topics in Circuits and Systems* 12.3 (Sept. 2022), pp. 662–675. doi: [10.1109/JETCAS.2022.3202204](https://doi.org/10.1109/JETCAS.2022.3202204) (cited on page 22).
- [43] Emmanuel Jeandel, Simon Perdrix, and Renaud Vilmart. 'Completeness of the ZX-Calculus'. In: *Logical Methods in Computer Science* Volume 16, Issue 2 (June 4, 2020) (cited on page 22).
- [44] Tuomas Laakkonen. 'Graphical Stabilizer Decompositions For Counting Problems'. PhD thesis. University of Oxford, 2022. (Visited on 04/08/2024) (cited on pages 22, 25, 32, 34, 65).
- [45] Gina Muuss. 'Linear Combinations of ZX-diagrams for Parameterized Quantum Circuits'. In: () (cited on pages 22, 23).
- [46] *Quantum Logic Gate*. In: *Wikipedia*. Aug. 16, 2024. (Visited on 10/22/2024) (cited on page 23).

- [47] Boldizsár Poór, Razin A. Shaikh, and Quanlong Wang. *ZX-calculus Is Complete for Finite-Dimensional Hilbert Spaces*. May 17, 2024. URL: <http://arxiv.org/abs/2405.10896> (visited on 10/23/2024). Pre-published (cited on page 23).
- [48] Miriam Backens. ‘The ZX-calculus Is Complete for Stabilizer Quantum Mechanics’. In: *New Journal of Physics* 16.9 (Sept. 2014), p. 093021. doi: [10.1088/1367-2630/16/9/093021](https://doi.org/10.1088/1367-2630/16/9/093021) (cited on page 23).
- [49] 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. 31, 2019), pp. 23–42. doi: [10.4204/EPTCS.287.2](https://doi.org/10.4204/EPTCS.287.2) (cited on page 24).
- [50] Amar Hadzihasanovic. *A Diagrammatic Axiomatisation for Qubit Entanglement*. Jan. 28, 2015. doi: [10.48550/arXiv.1501.07082](https://doi.org/10.48550/arXiv.1501.07082). Pre-published (cited on page 24).
- [51] Titouan Carette and Emmanuel Jeandel. ‘A Recipe for Quantum Graphical Languages’. In: *47th International Colloquium on Automata, Languages, and Programming (ICALP 2020)*. Ed. by Artur Czumaj, Anuj Dawar, and Emanuela Merelli. Vol. 168. Leibniz International Proceedings in Informatics (LIPIcs). Dagstuhl, Germany: Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2020, 118:1–118:17. doi: [10.4230/LIPIcs.ICALP.2020.118](https://doi.org/10.4230/LIPIcs.ICALP.2020.118) (cited on page 24).
- [52] Razin A. Shaikh, Quanlong Wang, and Richie Yeung. ‘How to Sum and Exponentiate Hamiltonians in ZXW Calculus’. In: *Electronic Proceedings in Theoretical Computer Science* 394 (Nov. 16, 2023), pp. 236–261. doi: [10.4204/EPTCS.394.14](https://doi.org/10.4204/EPTCS.394.14) (cited on page 24).
- [53] Boldizsár Poór et al. ‘Completeness for Arbitrary Finite Dimensions of ZXW-calculus, a Unifying Calculus’. In: *2023 38th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*. June 26, 2023, pp. 1–14. doi: [10.1109/LICS56636.2023.10175672](https://doi.org/10.1109/LICS56636.2023.10175672) (cited on page 24).
- [54] Ross Duncan and Simon Perdrix. *Graphs States and the Necessity of Euler Decomposition*. Vol. 5635. 2009 (cited on page 25).
- [55] Ross Duncan and Simon Perdrix. ‘Rewriting Measurement-Based Quantum Computations with Generalised Flow’. In: *Automata, Languages and Programming*. Ed. by Samson Abramsky et al. Berlin, Heidelberg: Springer, 2010, pp. 285–296. doi: [10.1007/978-3-642-14162-1_24](https://doi.org/10.1007/978-3-642-14162-1_24) (cited on page 25).
- [56] A. Hillebrand. ‘Quantum Protocols Involving Multiparticle Entanglement and Their Representations in the Zx-Calculus.’ In: 2011. (Visited on 12/17/2024) (cited on page 25).
- [57] Titouan Carette, Yohann D’Anello, and Simon Perdrix. ‘Quantum Algorithms and Oracles with the Scalable ZX-calculus’. In: *Electronic Proceedings in Theoretical Computer Science* 343 (Sept. 18, 2021), pp. 193–209. doi: [10.4204/EPTCS.343.10](https://doi.org/10.4204/EPTCS.343.10) (cited on page 25).
- [58] Niel de Beaudrap and Dominic Horsman. *The ZX Calculus Is a Language for Surface Code Lattice Surgery*. Version 1. Apr. 27, 2017. doi: [10.48550/arXiv.1704.08670](https://doi.org/10.48550/arXiv.1704.08670). Pre-published (cited on page 25).
- [59] Niel de Beaudrap et al. *Pauli Fusion: A Computational Model to Realise Quantum Transformations from ZX Terms*. Version 1. Apr. 29, 2019. doi: [10.48550/arXiv.1904.12817](https://doi.org/10.48550/arXiv.1904.12817). Pre-published (cited on page 25).
- [60] Nicholas Chancellor et al. *Graphical Structures for Design and Verification of Quantum Error Correction*. Version 3. Jan. 12, 2018. doi: [10.48550/arXiv.1611.08012](https://doi.org/10.48550/arXiv.1611.08012). Pre-published (cited on page 25).
- [61] Ross Duncan and Maxime Lucas. ‘Verifying the Steane Code with Quantomatic’. In: *Electronic Proceedings in Theoretical Computer Science* 171 (Dec. 27, 2014), pp. 33–49. doi: [10.4204/EPTCS.171.4](https://doi.org/10.4204/EPTCS.171.4) (cited on page 25).
- [62] Liam Garvie and Ross Duncan. ‘Verifying the Smallest Interesting Colour Code with Quantomatic’. In: *Electronic Proceedings in Theoretical Computer Science* 266 (Feb. 27, 2018), pp. 147–163. doi: [10.4204/EPTCS.266.10](https://doi.org/10.4204/EPTCS.266.10) (cited on page 25).
- [63] Bob Coecke et al. *Foundations for Near-Term Quantum Natural Language Processing*. Dec. 7, 2020. doi: [10.48550/arXiv.2012.03755](https://doi.org/10.48550/arXiv.2012.03755). Pre-published (cited on page 25).
- [64] Aleks Kissinger and John van de Wetering. ‘Simulating Quantum Circuits with ZX-calculus Reduced Stabiliser Decompositions’. In: *Quantum Science and Technology* 7.4 (Oct. 1, 2022), p. 044001. doi: [10.1088/2058-9565/ac5d20](https://doi.org/10.1088/2058-9565/ac5d20) (cited on pages 25, 30).

- [65] Matthew Sutcliffe and Aleks Kissinger. *Fast Classical Simulation of Quantum Circuits via Parametric Rewriting in the ZX-calculus*. Mar. 11, 2024. doi: [10.48550/arXiv.2403.06777](https://doi.org/10.48550/arXiv.2403.06777). Pre-published (cited on page 25).
- [66] Alexander Cowtan et al. ‘Phase Gadget Synthesis for Shallow Circuits’. In: *Electronic Proceedings in Theoretical Computer Science* 318 (May 1, 2020), pp. 213–228. doi: [10.4204/EPTCS.318.13](https://doi.org/10.4204/EPTCS.318.13) (cited on page 25).
- [67] Niel De Beaudrap, Xiaoning Bian, and Quanlong Wang. ‘Techniques to Reduce Pi/4-Parity-Phase Circuits, Motivated by the ZX Calculus’. In: *Electronic Proceedings in Theoretical Computer Science* 318 (May 1, 2020), pp. 131–149. doi: [10.4204/EPTCS.318.9](https://doi.org/10.4204/EPTCS.318.9) (cited on page 25).
- [68] Ross Duncan et al. ‘Graph-Theoretic Simplification of Quantum Circuits with the ZX-calculus’. In: *Quantum* 4 (June 4, 2020), p. 279. doi: [10.22331/q-2020-06-04-279](https://doi.org/10.22331/q-2020-06-04-279) (cited on page 25).
- [69] Aleks Kissinger and John van de Wetering. ‘Reducing the Number of Non-Clifford Gates in Quantum Circuits’. In: *Physical Review A* 102.2 (Aug. 11, 2020), p. 022406. doi: [10.1103/PhysRevA.102.022406](https://doi.org/10.1103/PhysRevA.102.022406) (cited on page 25).
- [70] Niel De Beaudrap, Aleks Kissinger, and Konstantinos Meichanetzidis. ‘Tensor Network Rewriting Strategies for Satisfiability and Counting’. In: *Electronic Proceedings in Theoretical Computer Science* 340 (Sept. 6, 2021), pp. 46–59. doi: [10.4204/EPTCS.340.3](https://doi.org/10.4204/EPTCS.340.3) (cited on page 25).
- [71] Chen Zhao and Xiao-Shan Gao. ‘Analyzing the Barren Plateau Phenomenon in Training Quantum Neural Networks with the ZX-calculus’. In: *Quantum* 5 (June 4, 2021), p. 466. doi: [10.22331/q-2021-06-04-466](https://doi.org/10.22331/q-2021-06-04-466) (cited on pages 25, 34).
- [72] Quanlong Wang, Richie Yeung, and Mark Koch. ‘Differentiating and Integrating ZX Diagrams with Applications to Quantum Machine Learning’. 2022 (cited on pages 25, 33).
- [73] Richard D.P. East et al. ‘AKLT-States as ZX-Diagrams: Diagrammatic Reasoning for Quantum States’. In: *PRX Quantum* 3.1 (Jan. 4, 2022), p. 010302. doi: [10.1103/PRXQuantum.3.010302](https://doi.org/10.1103/PRXQuantum.3.010302) (cited on page 25).
- [74] Razin A. Shaikh and Stefano Gogioso. *Categorical Semantics for Feynman Diagrams*. May 1, 2022. doi: [10.48550/arXiv.2205.00466](https://doi.org/10.48550/arXiv.2205.00466). Pre-published (cited on page 25).
- [75] Razin A. Shaikh, Lia Yeh, and Stefano Gogioso. *The Focked-up ZX Calculus: Picturing Continuous-Variable Quantum Computation*. June 5, 2024. doi: [10.48550/arXiv.2406.02905](https://doi.org/10.48550/arXiv.2406.02905). Pre-published (cited on page 25).
- [76] Jonathan Gorard, Manojna Namuduri, and Xerxes D. Arsiwalla. *ZX-Calculus and Extended Hypergraph Rewriting Systems I: A Multiway Approach to Categorical Quantum Information Theory*. Oct. 5, 2020. doi: [10.48550/arXiv.2010.02752](https://doi.org/10.48550/arXiv.2010.02752). Pre-published (cited on page 25).
- [77] *The Wolfram Physics Project: Finding the Fundamental Theory of Physics*. URL: <https://www.wolframphysics.org/> (visited on 12/17/2024) (cited on page 25).
- [78] Julien Codsí. *Cutting-Edge Graphical Stabiliser Decompositions for Classical Simulation of Quantum Circuits*. URL: <https://www.maths.ox.ac.uk/system/files/inline-files/J%20Codsí%202021-22.pdf> (cited on page 29).
- [79] Sergey Bravyi, Graeme Smith, and John Smolin. ‘Trading Classical and Quantum Computational Resources’. In: *Physical Review X* 6.2 (June 29, 2016), p. 021043. doi: [10.1103/PhysRevX.6.021043](https://doi.org/10.1103/PhysRevX.6.021043) (cited on pages 30, 34).
- [80] Mark Koch, Richie Yeung, and Quanlong Wang. *Speedy Contraction of ZX Diagrams with Triangles via Stabiliser Decompositions*. July 4, 2023. URL: <http://arxiv.org/abs/2307.01803> (visited on 02/26/2024). Pre-published (cited on pages 31, 32, 34, 42, 43, 57, 65).
- [81] Jarrod R. McClean et al. ‘Barren Plateaus in Quantum Neural Network Training Landscapes’. In: *Nature Communications* 9.1 (Nov. 16, 2018), p. 4812. doi: [10.1038/s41467-018-07090-4](https://doi.org/10.1038/s41467-018-07090-4) (cited on page 33).
- [82] *Tix3Dev/Novel_star_state_decompositions*. URL: https://github.com/Tix3Dev/novel_star_state_decompositions (visited on 11/17/2024) (cited on page 34).

- [83] Wolfram Research Inc. *Mathematica, Version 14.1*. URL: <https://www.wolfram.com/mathematica> (cited on page 34).
- [84] Tuomas Laakkonen. *Tuomas56/Cliffs*. Apr. 29, 2024. (Visited on 11/17/2024) (cited on pages 34, 65).
- [85] *Simulated Annealing*. In: *Wikipedia*. Sept. 14, 2024. (Visited on 11/17/2024) (cited on page 34).
- [86] *Amazon EC2 C6a Instances - Amazon Web Services*. Amazon Web Services, Inc. URL: <https://aws.amazon.com/ec2/instance-types/c6a/> (visited on 11/17/2024) (cited on pages 37, 65).
- [87] *Novel_star_state_decompositions/Zxbarren-Private/Result.Txt at Main · Tix3Dev/Novel_star_state_decompositions*. URL: https://github.com/Tix3Dev/novel_star_state_decompositions/blob/main/zxbarren-private/result.txt (visited on 12/10/2024) (cited on page 43).
- [88] mjsutcliffe99. *Mjsutcliffe99/ProcOptCut*. Sept. 24, 2024. (Visited on 11/26/2024) (cited on page 45).
- [89] *Zxcalc/Quizx*. ZX Calculus Projects, Nov. 1, 2024. (Visited on 11/27/2024) (cited on pages 46, 58).
- [90] *Tix3Dev/Feynman_loop_diagram_qsim*. GitHub. URL: https://github.com/Tix3Dev/feynman_loop_diagram_qsim (visited on 11/30/2024) (cited on pages 51, 55–57).
- [91] Selomir Ramírez-Uribe, Andrés E. Rentería-Olivo, and Germán Rodrigo. *Quantum Querying Based on Multicontrolled Toffoli Gates for Causal Feynman Loop Configurations and Directed Acyclic Graphs*. Apr. 4, 2024. doi: [10.48550/arXiv.2404.03544](https://arxiv.org/abs/2404.03544). Pre-published (cited on pages 53–57).
- [92] Tom Lancaster and Stephen J. Blundell. *Quantum Field Theory for the Gifted Amateur*. Oxford University Press, Apr. 2014 (cited on pages 53, 54).
- [93] William J. Torres Bobadilla. ‘Lotty – The Loop-Tree Duality Automation’. In: *The European Physical Journal C* 81.6 (June 14, 2021), p. 514. doi: [10.1140/epjc/s10052-021-09235-0](https://doi.org/10.1140/epjc/s10052-021-09235-0) (cited on pages 53, 54).
- [94] William J. Torres Bobadilla. *Off-Shell Jacobi Currents within the Loop-Tree Duality*. URL: https://indico.cern.ch/event/577856/contributions/3420312/attachments/1878617/3094415/EPS-2019_ckd.pdf (cited on page 53).
- [95] Selomir Ramírez-Uribe et al. ‘Quantum Algorithm for Feynman Loop Integrals’. In: *Journal of High Energy Physics* 2022.5 (May 16, 2022), p. 100. doi: [10.1007/JHEP05\(2022\)100](https://doi.org/10.1007/JHEP05(2022)100) (cited on page 54).
- [96] *List of Quantum Field Theories*. In: *Wikipedia*. Aug. 11, 2024. (Visited on 12/28/2024) (cited on page 54).
- [97] *9.5: Cauchy Residue Theorem*. Mathematics LibreTexts. Sept. 5, 2017. URL: [https://math.libretexts.org/Bookshelves/Analysis/Complex_Variables_with_Applications_\(Orloff\)/09%3A_Residue_Theorem/9.05%3A_Cauchy_Residue_Theorem](https://math.libretexts.org/Bookshelves/Analysis/Complex_Variables_with_Applications_(Orloff)/09%3A_Residue_Theorem/9.05%3A_Cauchy_Residue_Theorem) (visited on 12/01/2024) (cited on page 54).
- [98] *Grover’s Algorithm*. In: *Wikipedia*. Sept. 19, 2024. (Visited on 09/29/2024) (cited on pages 55, 56).
- [99] Aleks Kissinger and John van de Wetering. ‘PyZX: Large Scale Automated Diagrammatic Reasoning’. In: *Electronic Proceedings in Theoretical Computer Science* 318 (May 1, 2020), pp. 229–241. doi: [10.4204/EPTCS.318.14](https://doi.org/10.4204/EPTCS.318.14) (cited on pages 57, 61).
- [100] *Python VS Rust Benchmarks, Which Programming Language or Compiler Is Faster*. URL: <https://programming-language-benchmarks.vercel.app/python-vs-rust> (visited on 12/06/2024) (cited on page 58).
- [101] *Tix3Dev/Rand_multi_ctrl_toff_dense_qcirc_sim*. URL: https://github.com/Tix3Dev/rand_multi_ctrl_toff_dense_qcirc_sim (visited on 12/10/2024) (cited on pages 59–62).
- [102] *Tix3Dev/Unexpected_improv_of_speedy_algo*. URL: https://github.com/Tix3Dev/unexpected_improv_of_speedy_algo (visited on 12/29/2024) (cited on page 66).