FRAUNHOFER INSTITUT (AISEC)

TECHNISCHE UNIVERSITÄT MÜNCHEN

Seminararbeit

ZX-Calculus

Author:	Manuel Lerchner
Submission Date:	July 4, 2023

Contents

ZX-Calculus

Manuel Lerchner
Technical University of Munich
Munich, Germany

***Abstract*—ZX-Calculus is a graphical language which extends classical quantum circuits by splitting up logic gates into even smaller building blocks. Those building blocks are called spiders and are represented by colored nodes in a graph. Together with edges connecting those nodes, they form a ZX-diagram. Using a set of rewrite rules, ZX-diagrams can be transformed into each other. This allows for a more intuitive way of reasoning about the optimization of quantum circuits, as there are fewer rewrite rules to remember than in the classical logic gate model. In this paper, I will introduce the ZX-Calculus and its rewrite rules, as well as some of its applications. I will give a special focus on the optimization of quantum circuits, as this is one of the main applications of the ZX-Calculus.**

***Index Terms*—quantum computing, ZX-Calculus, quantum circuits, circuit optimization**

I. Introduction

In the last few decades, quantum computing has become a very active field of research. The main reason for this is the fact that quantum computers promise to outperform classical computers in certain tasks. The most famous example of this is Shor's algorithm [9] which can factorize large numbers in polynomial time. This is a problem that is believed to be intractable for classical computers. Another example is Grover's algorithm [3] which can search an unsorted database in $\mathcal{O}(\sqrt{N})$ time. This is a quadratic speedup compared to the classical $\mathcal{O}(N)$ time.

Usually those algorithms described in a very high-level "language", the so called (quantum-) circuit model. This allows for a very intuitive way reasoning about quantum algorithms, because every unitary operation applied to the quantum state can be compactly represented by a single gate. However, this model does not take the restrictions of real quantum computers into account. [10]

A. Quantum Circuit Compilation

Real quantum computers come with a set of restrictions. For example, the set of gates is typically limited to a small set of universal gates (e.g. the Clifford+T gate set). Furthermore, the connectivity of the qubits is limited. This means that not every operation can be applied to every pair of qubits. As a consequence, the original circuits needs to be *compiled* into a circuit that can be executed on the specific quantum computer. This process is called *quantum circuit compilation*.

In circuit 1 the Toffoli gate is shown. However it is not represented in the Clifford+T gate set. In order to execute this circuit on a real quantum computer, we need to transform it into the Clifford+T gate set first, as this allows an efficient, and fault-tolerant implementation using surface code error correction. [6]

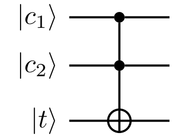


Fig. 1: The Toffoli gate.

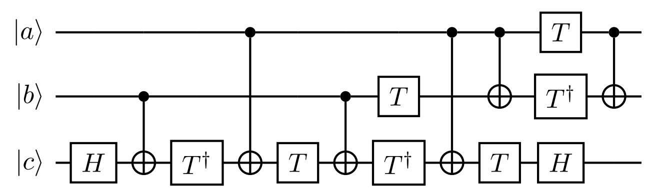


Fig. 2: Decomposition of the Toffoli gate in the Clifford+T gate set.

B. Quantum Circuit Optimization

Such a decomposition is shown in figure 2. Notice how the amount of gates increases significantly. This is a common problem in quantum circuit compilation. It means that the compiled circuit will be slower and therefore the execution time can exceed the coherence time of the qubits, and thus making the computation useless. [8]

This is where Circuit Optimizers come into play. They try to reduce the amount of gates in a circuit. This can be done by applying a set of rewrite rules to the circuit. The goal is to find a circuit that is equivalent to the original circuit, but has less gates. An important metric for simplifying quantum circuits is the so called *T-Count*. The T-Count is the amount of T gates in a circuit. Since the T-Gate is a non-Clifford gate it is very expensive to simulate and requires order of magnitudes more resources than the other clifford gates. [6]

C. Drawbacks of Classical Circuit Optimization

There are many different approaches to quantum circuit optimization. The most basic approach is to apply a set of rewrite rules directly to the logic-gate representation of the circuit. This approach is called *gate-level optimization*. [7]

However, this approach is typically very inefficient, as there exists a huge amount of possible rewrite rules (Some of them are shown in figure 3). Furthermore, rewrite rules are typically not independent of each other. This means that applying a rewrite rule can introduce new opportunities for other rewrite rules. This makes it very hard to find an optimal solution. [5]

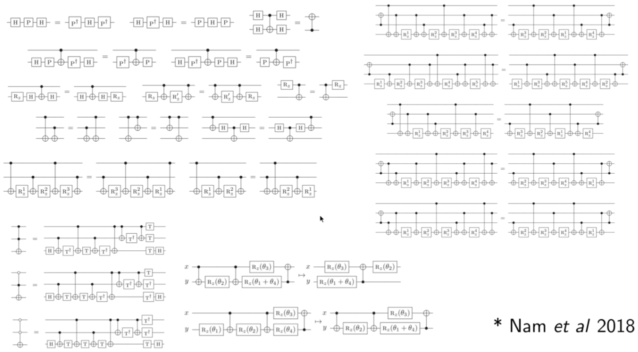


Fig. 3: Small subset of rewrite rules for classical circuit optimization [5]

D. Quantum Circuit Optimization using the ZX-Calculus

A more efficient approach for quantum circuit optimization is to use the ZX-Calculus. The ZX-Calculus is a graphical language for quantum computing, differing from the logic-gate representation by using connected nodes to represent operations on qubits. Since this new representation utilizes way fewer *gate-types* than the classical representation, there exist way fewer rewrite rules.

ZX-Calculus has been kickstarted by Coecke and Duncan in 2008 [1]. Since then, it has been used to prove a lot of interesting results in quantum computing. It was initially used in the field of Measurement Based Quantum Computing (MBQC) [2]. But recently it has found wide application in quantum circuit optimization and verification [11].

II. Mathematical Background: Category Theory

The ZX-Calculus is based on the mathematical model of Category-theory. A category is an abstract mathematical model which consists of objects and morphisms. Objects are the building blocks of the category, whereas morphisms act like mappings between objects. In particular a ZX-diagram is a strict compact closed symmetric monoidal categor [4].

Given such a category $\mathcal{C}$ with objects $\{A, B, C, D\}$ and morphisms $\{f : A \to B, g : B \to C, h : C \to D\}$, there exists a notion of composition . This means that we can compose morphisms to get new morphisms. In our example, we can compose f and g to get a new morphism $g \circ f : A \to C$. This composition is associative, meaning that $(h \circ g) \circ f = h \circ (g \circ f)$. Additionally, there exists an identity morphism $id_A : A \to A$ for each object A. This identity morphism is neutral with respect to composition, meaning that $f \circ id_A = f = id_B \circ f$.

There also exists a Bifunctor $\otimes : \mathcal{C} \times \mathcal{C} \to \mathcal{C}$ which is used to combine multiple *parallel* morphisms into a single morphism . In our example, we can combine f and g to get a new morphism $f \otimes g : A \otimes B \to C \otimes D$. This Bifunctor is associative, meaning that $(f \otimes g) \otimes h = f \otimes (g \otimes h)$. Additionally, there exists a unit object I which acts as a neutral element with respect to the Bifunctor, meaning that $f \otimes I = f = I \otimes f$ for each morphism f.

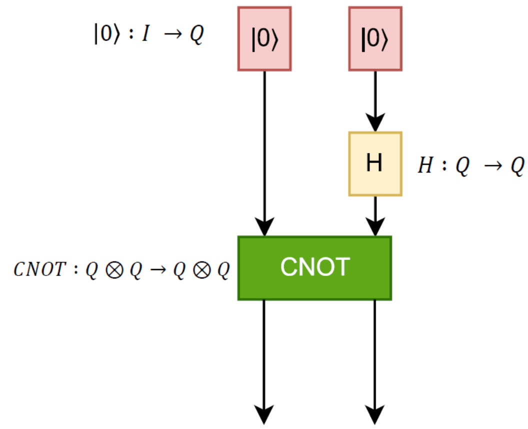


Fig. 4: Circuit for creating Bell pairs

Furthermore, there exists a natural isomorphism $\sigma_{A,B} : A \otimes B \to B \otimes A$ for each pair of objects A and B. This means that we can always swap the order of parallel morphisms without changing their meaning. This is also called the *swap* rule.

Finally, there also exists a notion of dual objects. Given an object A, there exists a dual object A^*. Using the special morphisms $\eta_A : I \to A \otimes A^*$ and $\epsilon_A : A^* \otimes A \to I$, we can introduce a notion of *curved* morphism. This special morphisms are called *CAP* and *CUP* respectively.

A. Quantum Circuits based on Category Theory

Using the mathematical model of category theory, we can now define a quantum circuit. For this we are going to restrict our category to the category of finite dimensional Hilbert spaces and linear maps. This means that the objects of our category are finite dimensional Hilbert spaces and the morphisms are linear maps between those Hilbert spaces. We are going to denote this category as $\mathbf{FDHilb}$. Furthermore we define the composition of morphisms as the matrix multiplication of linear maps. The Bifunctor $\otimes$ is defined as the tensor/kronecker product respectively.

We can now define an actual circuit, using this model: For this we are going to look at an entanglement circuit, which creates Bell pairs. This circuit is shown in figure 4. Note that it looks very similar to the normal gate-based circuit. The only difference is that we are translating it in such a way, that we are using the morphisms of our category instead of the gates.

It uses the following morphisms:

- $|0\rangle$ is the morphism $I \to Q$ which initializes a qubit to the state $|0\rangle$.
- H is a morphism of the form $Q \to Q$. It is a linear map from one qubit to one qubit.
- $CNOT$ is the controlled-not gate, which is a linear map from two qubits to two qubits. It is a morphism of the form $Q \otimes Q \to Q \otimes Q$.

At the moment we are only able to define the circuit abstractly. We can not yet define the actual circuit, because we have not yet defined specifically what the objects and morphisms of our category are. We are going to do this in the next section, where we are going to define the basic building Blocks of ZX-Calculus.

III. Introduction to ZX-Calculus

As stated in the previous section, the ZX-Calculus represents the **FDHilb** category. This means that the objects of our category are finite dimensional Hilbert spaces and the morphisms are linear maps between those Hilbert spaces. We are going to represent the morphisims as *Spiders*. So in total the whole classical quantum circuit is represented as a graph of morphism/spiders. In this representation it is very easy to to apply simplification rules to the circuit.

A. Spiders

Spiders are the *atoms* of a ZX-Diagram. They represent the decomposition of quantum gates into even smaller and more fundamental operations. Some important decompositions are shown in figure 7.

Spiders can have an arbitrary number of incoming and outgoing edges and appear in two flavors: *Z-Spiders* and *X-Spiders*. This distinction is shown visually using the green and red color respectively when drawing the spiders. Additionally spiders may carry a phase value $\alpha \in [0, 2\pi)$. Which can be ommited if it is zero.

In figure 5 the spiders are shown visually. The first spider is a *Z-Spider* and the second one is an *X-Spider*. Both have n incoming and m outgoing edges and carry a phase value α.

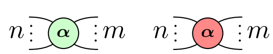
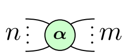


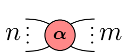
Fig. 5: Fundamental spiders with n incoming and m outgoing edges

It is important to remember that each spider represents a morphism in the **FDHilb** category. This means that each spider represents a linear map from the incoming n to the outgoing m qubits. Those linear maps are shown in figure 6.



$$:= |\underbrace{0\dots0}_{m}\rangle\langle\underbrace{0\dots0}_{n}| + e^{i\alpha}|\underbrace{1\dots1}_{m}\rangle\langle\underbrace{1\dots1}_{n}|$$

$$= \begin{pmatrix} 1 & 0 & \dots & 0 \\ 0 & 0 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & e^{i\alpha} \end{pmatrix}$$



$$:= |\underbrace{+\cdots+}_{m}\rangle\langle\underbrace{-\cdots-}_{n}| + e^{i\alpha}|\underbrace{-\cdots-}_{m}\rangle\langle\underbrace{+\cdots+}_{n}|$$

Fig. 6: Linear maps represented by individual spiders

Note that the linear maps in figure 6 do not have to be unitary, nor do they have to square. The dimension of the linear map depends only on the number of incoming and outgoing edges. For example, a spider with n incoming and m outgoing edges represents a linear map in $\mathbb{C}^{2^m \times 2^n}$.

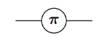
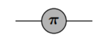
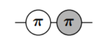
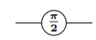
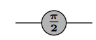
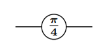
Name	Diagram	Matrix
identity		$\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$
Pauli Z		$\begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$
Pauli X NOT gate		$\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$
Pauli Y	i 	$\begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}$
Hadamard gate		$\frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$
S gate		$\begin{pmatrix} 1 & 0 \\ 0 & i \end{pmatrix}$
V gate		$\frac{1}{2}\begin{pmatrix} 1+i & 1-i \\ 1-i & 1+i \end{pmatrix}$
T gate		$\begin{pmatrix} 1 & 0 \\ 0 & e^{i\frac{\pi}{4}} \end{pmatrix}$

Fig. 7: Spiders representing quantum gates [11][P.87]

B. Classical Quantum Gates as Spiders

By just using the single spiders shown in figure 5 we can already represent a lot of quantum gates. The most important ones are shown in figure 7. Note that we are going to ignore the global scalar values of ZX-Diagrams in the following sections, since they are negligible for most cases [10]. Furthermore, when working with unitary circuits, the scalar value can be restored at the end of the calculation as the resulting circuit-matrix is proportional to the unitary matrix [11].

It is easy to verify that the gates from the figure 7 can indeed be represented using just the basic spiders. This boils down just inserting the corresponding spiders into the formulas from figure 6 and to compare the resulting matrices with the matrices of the Pauli gates. These proofs are shown in figures A, B and C.

C. Bigger Circuits

Using the rules from figure 7 we can only represent circuits consisting of a single gate. For bigger circuits we need to combine multiple spiders into a single diagram. This is done by connecting the spiders leg to leg. This means that the outgoing legs of one spider are connected to the incoming legs of another spider.

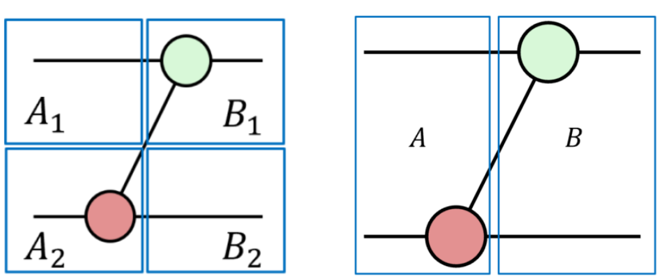
But before we can work with such circuits we need to define some more rules to calculate the matrix representation of a circuit.

1) **Only Topology Matters**
The first rule is that we can move spiders around freely, as long as we maintain the correct order of the incoming andoutgoing legs. This rule corresponds to the *Only Topology Matters* mantra of the ZX-Calculus.
2) **Parallel Composition** The second rule allows to calculate the matrix representation of spiders acting in parallel. We have seen this rule bevore, when we looked at the category representation of the ZX-Calculus (II). In **FDHilb** this rule corresponds to the kroncker product of matrices for the individual spiders.
3) **Sequential Composition** The third rule allows to calculate the matrix representation of spiders acting in sequence. This rule corresponds to the sequential composition (II) in the category representation of the ZX-Calculus. In particular it allows to calculate the matrix representation such a circuit by multiplying the matrices of the individual spiders.

D. *Example: CNOT Gate*

An example for a circuit consisting of multiple spiders is the classical CNOT gate, which can be represented by the ZX-Diagram shown in figure 8.

In order to calculate the matrix representation of this circuit we first need to split the circuit into the individual spiders. The result this process is already shown in figure 8a.



(a) CNOT split into 4 Regions (b) CNOT split into 2 Regions

Fig. 8: Splitting the CNOT gate into regions

Using the rules we just defined we can now calculate the matrix representation of the circuit. We start by applying the *Parallel Composition* rules, to combine sections A_1 and A_2 into a new section A. The same is done for sections B_1 and B_2 to create section B. This reduction results in the ZX-Diagram shown in figure 8b.

The calculation looks as follows:

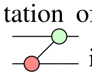
$$
\begin{aligned}
A &= A_1 \otimes A_2 \\
&= id \otimes \text{RedSpider}(n=1,\, m=2) \\
&= id \otimes |+\rangle\langle++| + |-\rangle\langle--| \\
&= \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \otimes \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 1 \\ 1 & 0 \end{pmatrix} \\
&= \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}
\end{aligned}
$$

$$
\begin{aligned}
B &= B_1 \otimes B_2 \\
&= \text{GreenSpider}(n=2,\, m=1) \otimes id \\
&= \begin{pmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{pmatrix} \otimes \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \\
&= \begin{pmatrix} 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \end{pmatrix}
\end{aligned}
$$

Now we have calculated all Regions from figure 8b and can combine them into a single matrix. For this we apply the *Sequential Composition* rule to combine sections A and B into a new section R.

The calculation looks as follows:

$$
\begin{aligned}
R &= A \circ B \\
&= B \cdot A \\
&= \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix} \\
&\propto \text{CNOT}
\end{aligned}
$$

The resulting matrix is proportional to the matrix representation of the CNOT gate. This means that the ZX-Diagram  is indeed a valid representation of the CNOT gate.

IV. Motivation

V. Problem Statement

VI. Solution

VII. Evaluation

VIII. Results

IX. Future Work

X. Conclusion

References

[1] Bob Coecke and Ross Duncan. A graphical calculus for quantum observables. *Preprint*, 2007.

[2] Ross Duncan. A graphical approach to measurement-based quantum computing, 2012.

[3] Lov K. Grover. A fast quantum mechanical algorithm for database search, 1996.

[4] Emmanuel Jeandel. Completeness of the zx-calculus, 2020.

[5] Aleks Kissinger. An introduction to the zx-calculus. https://www.cs.ox.ac.uk/people/aleks.kissinger/slides/fullstack-25mins.pdf, 2020. accessed: 30.06.2023.

[6] Aleks Kissinger and John van de Wetering. Reducing t-count with the zx-calculus, 2020.

[7] Yunseong Nam, Neil J. Ross, Yuan Su, Andrew M. Childs, and Dmitri Maslov. Automated optimization of large quantum circuits with continuous parameters, may 2018.

[8] Michael A. Nielsen and Isaac L. Chuang. Quantum computation and quantum information, 2010.

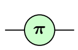
[9] Peter W. Shor. Algorithms for quantum computation: discrete logarithms and factoring, 1994.

[10] Robert Wille Tom Peham, Lukas Burgholzer. Equivalence checking of quantum circuits with the zx-calculus. https://www.cda.cit.tum.de/files/eda/2022_jetcas_equivalence_checking_of_quantum_circuits_with_the_zx_calculus.pdf, 2022. accessed: 30.06.2023.

[11] John van de Wetering. Zx-calculus for the working quantum computer scientist, 2020.

Appendix

A. ZX-Representation of the Pauli Z-Gate

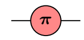


$$
\begin{aligned}
&\equiv |\underbrace{0\dots0}_{1}\rangle\langle\underbrace{0\dots0}_{1}| + e^{i\pi}|\underbrace{1\dots1}_{1}\rangle\langle\underbrace{1\dots1}_{1}| \\
&= |0\rangle\langle0| - |1\rangle\langle1| \\
&= \begin{pmatrix}1\\0\end{pmatrix}\begin{pmatrix}1 & 0\end{pmatrix} - \begin{pmatrix}0\\1\end{pmatrix}\begin{pmatrix}0 & 1\end{pmatrix} \\
&= \begin{pmatrix}1 & 0\\0 & -1\end{pmatrix} \\
&= Z
\end{aligned}
$$

Fig. 9: Pauli-Z gate represented as a ZX-Diagram

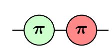
B. ZX-Representation of the Pauli X-Gate

C. ZX-Representation of the Pauli Y-Gate

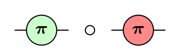


$$
\begin{aligned}
&\equiv |\underbrace{+\cdots+}_{1}\rangle\langle\underbrace{+\cdots+}_{1}| + e^{i\pi}|\underbrace{-\cdots-}_{1}\rangle\langle\underbrace{-\cdots-}_{1}| \\
&= |+\rangle\langle+| - |-\rangle\langle-| \\
&= \frac{1}{\sqrt{2}}\begin{pmatrix}1\\1\end{pmatrix}\frac{1}{\sqrt{2}}\begin{pmatrix}1 & 1\end{pmatrix} - \frac{1}{\sqrt{2}}\begin{pmatrix}1\\-1\end{pmatrix}\frac{1}{\sqrt{2}}\begin{pmatrix}1 & -1\end{pmatrix} \\
&= \frac{1}{2}\begin{pmatrix}1 & 1\\1 & 1\end{pmatrix} - \frac{1}{2}\begin{pmatrix}1 & -1\\-1 & 1\end{pmatrix} \\
&= \begin{pmatrix}0 & 1\\1 & 0\end{pmatrix} \\
&= X
\end{aligned}
$$

Fig. 10: Pauli-X gate represented as a ZX-Diagram



$\equiv$



$$
\begin{aligned}
&\equiv Z \circ X \\
&= XZ \\
&= \begin{pmatrix}0 & 1\\1 & 0\end{pmatrix}\begin{pmatrix}1 & 0\\0 & -1\end{pmatrix} \\
&= \begin{pmatrix}0 & -1\\1 & 0\end{pmatrix} \\
&= Y/i \\
&\propto Y
\end{aligned}
$$

Fig. 11: Pauli-Y gate represented as a ZX-Diagram