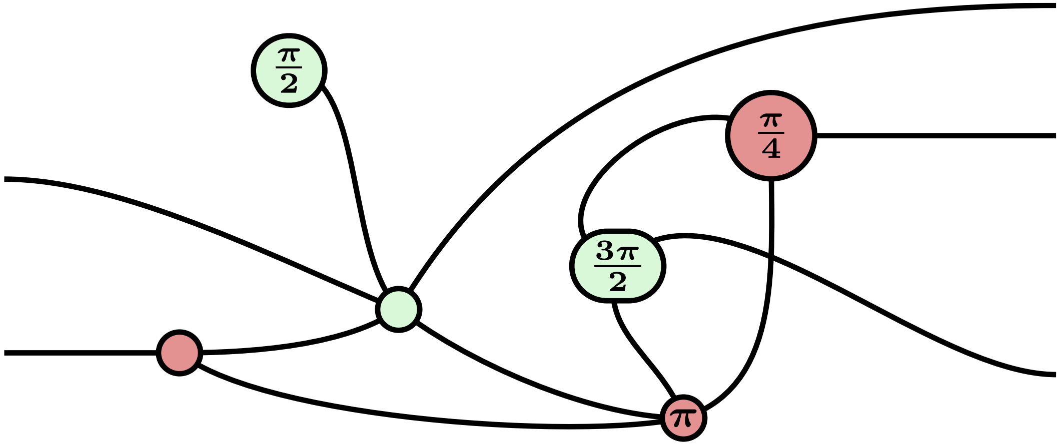
ZX-Calculus



Manuel Lerchner

01.06.2023

What is ZX-Calculus?

- A way to represent Quantum Circuits
- Graphical language
- Rules for simplifying the Diagram

Applications

- Quantum Circuit Optimization
  - T-Count Optimization
- Circuit Compilation

Quantum Circuit Optimization

- Idea: Transform circuits into equivalent circuits:
- Goal: Fewer or simpler Gates

- But why use ZX-Calculus for this?

Classical Optimization

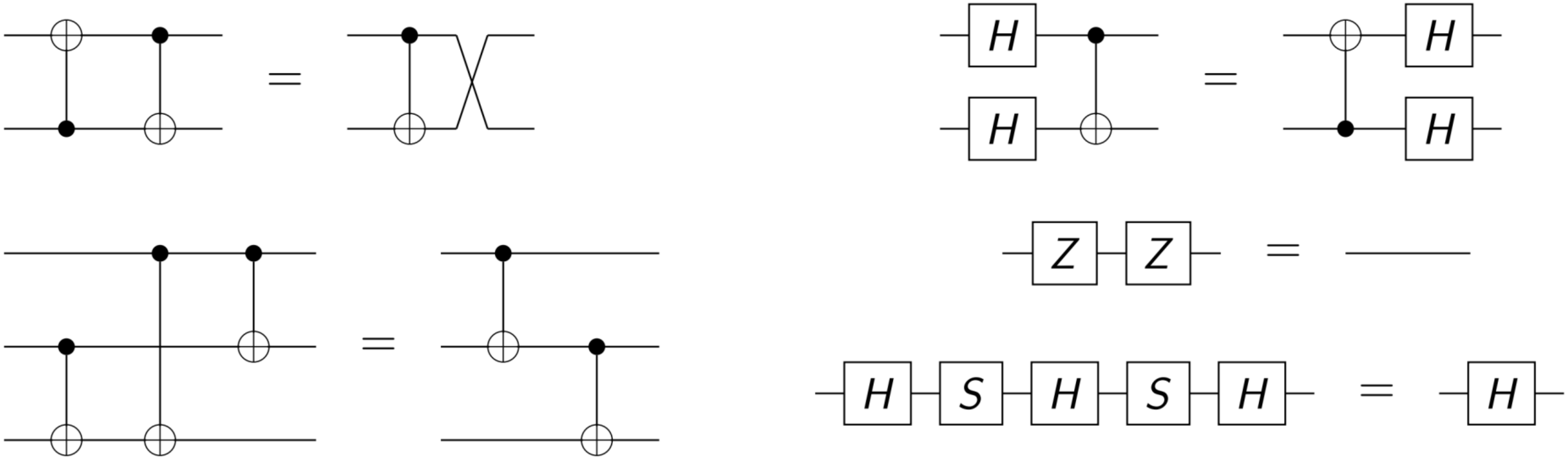


Fig. 2: Circuit Identities

Classical Optimization

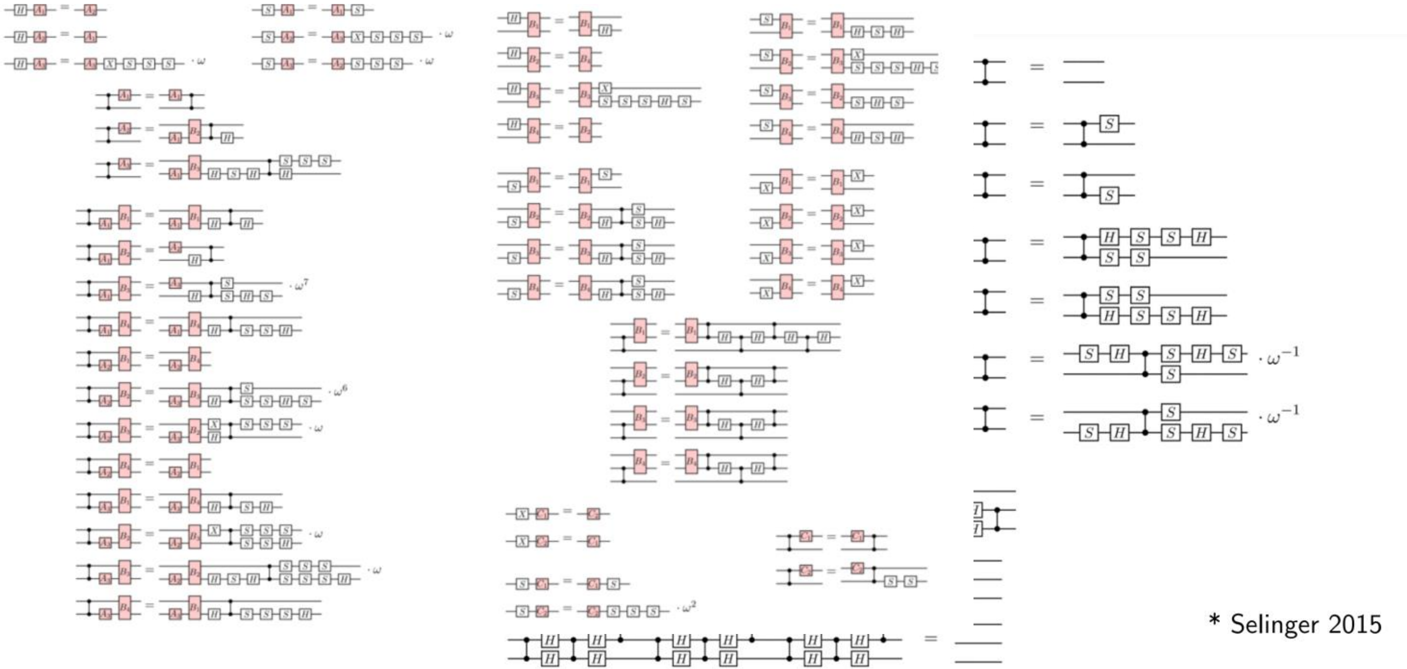
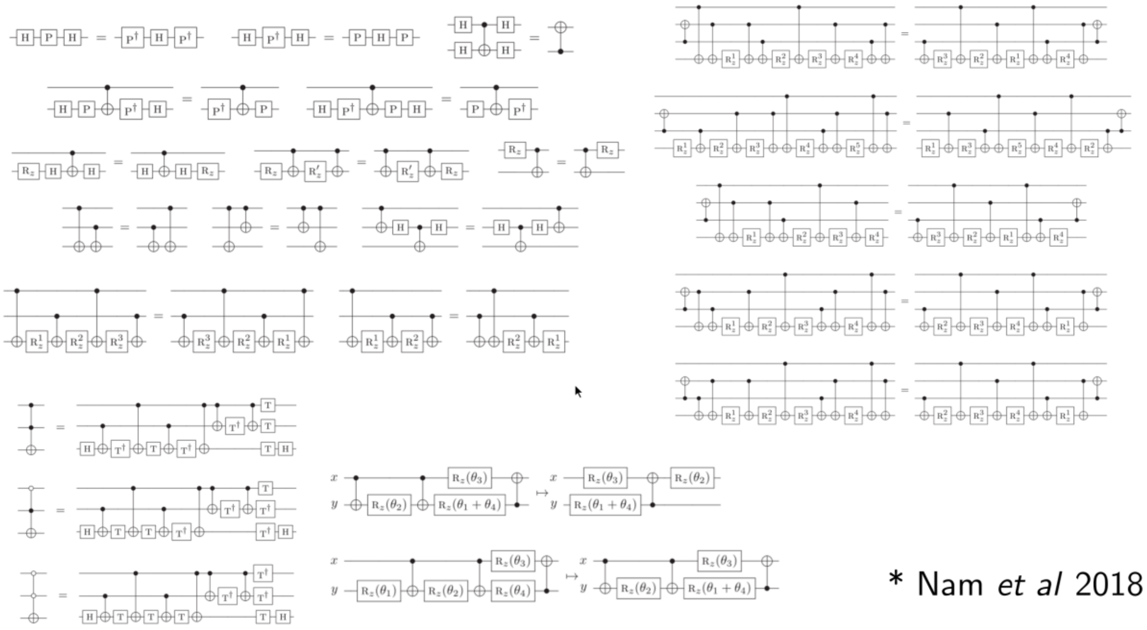


Fig. 2: Circuit Identities

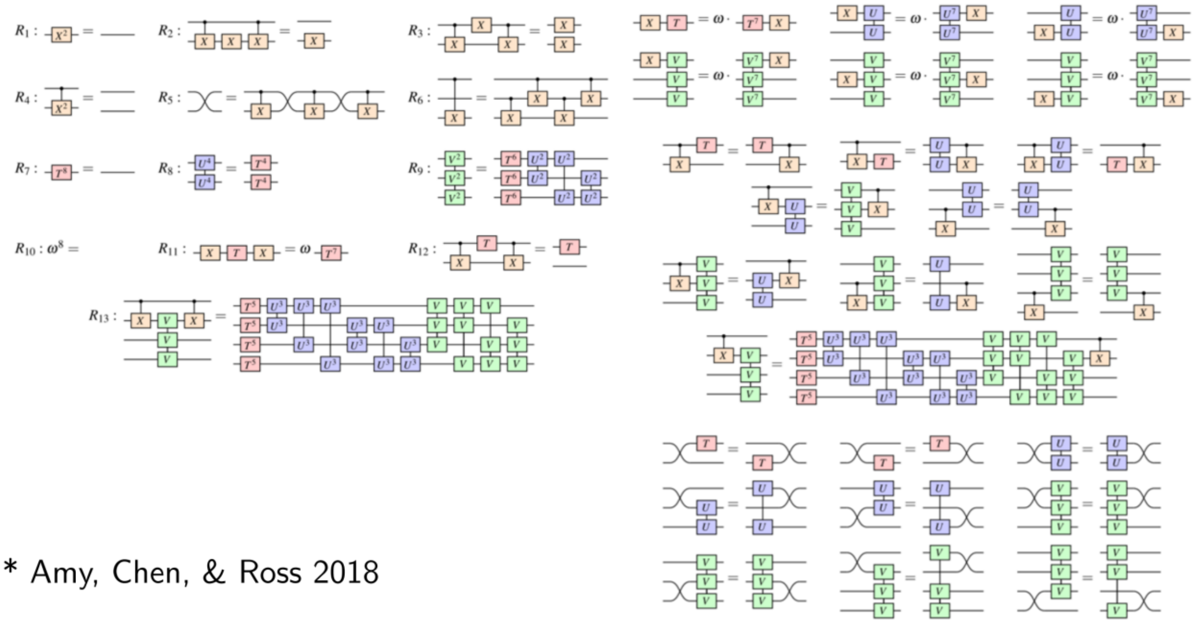
Classical Optimization



* Nam *et al* 2018

Fig. 2: Circuit Identities

Classical Optimization



* Amy, Chen, & Ross 2018

Fig. 2: Circuit Identities

Compilation of quantum circuits

- Circuits use many abstract gates
- Problems of real quantum computers:
  - Limited set of gates
  - Limited connectivity between qubits

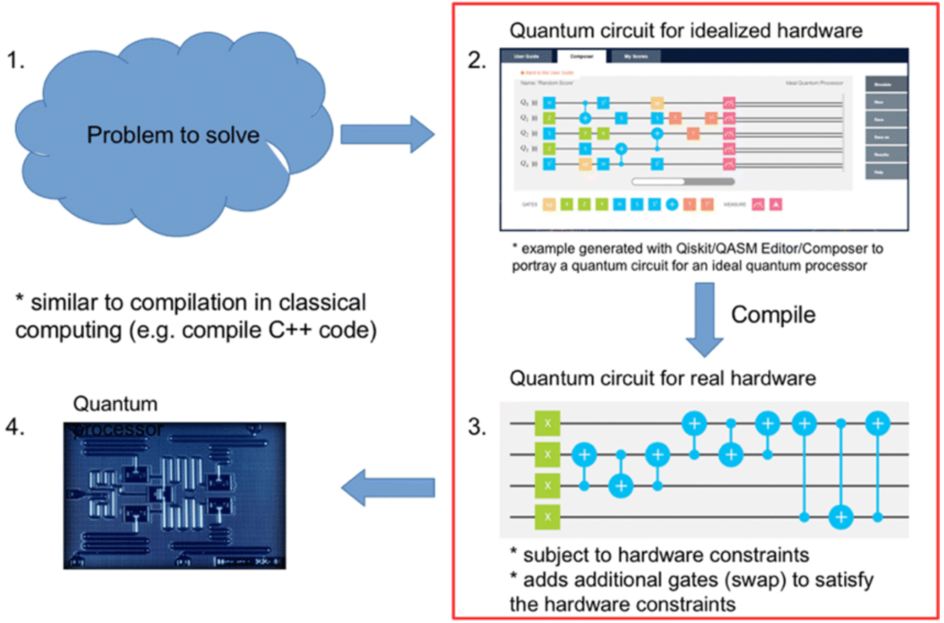


Fig. 3: Quantum Compilation

T-Count Optimization

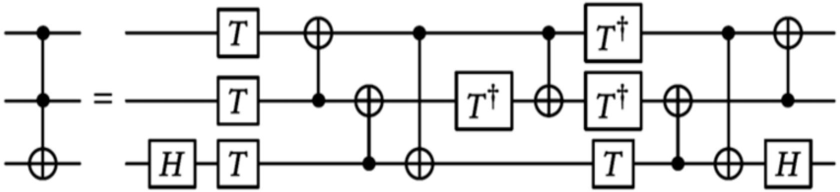


Fig. 4: Fault tolerant Toffoli Gate

- Quantum computers are affected by noise
- Clifford+T Circuits can be made tolerant to noise
  - Idea: Introduce Error Correcting Codes
  - Problem: Many new T-Gates need to be introduced
  - Difficult to simulate (Hardware Limits)
- ZX-Calculus can simplify such circuits

Mathematical Background: Category Theory

- Consists of **objects** and **arrows** (Morphisms)
  - Objects: $\{A, B, C\}$
  - Morphisms: $f: A \to B,\ g: B \to C,\ h: C \to D$
- Identity: $\forall A \in ob(\mathcal{C})\ .\ id_{\mathrm{A}}: A \to A$
- Associative Composition $\circ$: $g \circ f\ :\ A \to C$
  - Composition with id does nothing

$$h \circ g \circ f : A \to D$$

Monoidal Category

- Category $\mathcal{C}$ with:
  - Bifunctor: $\otimes : \mathcal{C} \times \mathcal{C} \to \mathcal{C}$
  - $\otimes$ is associative
  - Unit Object: $I \in \mathcal{C}$

$$f \otimes g \otimes h : A \otimes B \otimes C \to B \otimes C \otimes D$$

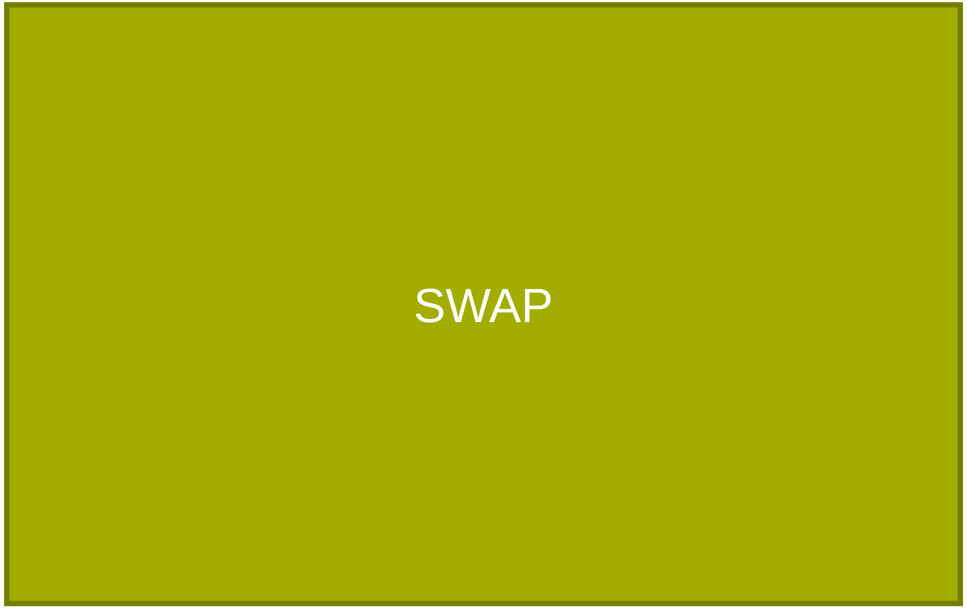
Monoidal Category

- Preparing States:
  - $\nu : I \to A$ (Ket)
- Erasing States:
  - $\phi^\dagger : A \to I$ (Bra)
- Combination:
  - $\phi^\dagger \circ \nu : I \to I$

IMG Preparing / erasing

Symmetric Monoidal Category

- Monoidal Category $\mathcal{C}$ with:
  - Swap-Isomorphism
  - $\sigma_{A,B} : A \otimes B \to B \otimes A$
- $\sigma_{B,A} \circ \sigma_{A,B} = id_{A,B}$
- Operations can be pushed trough



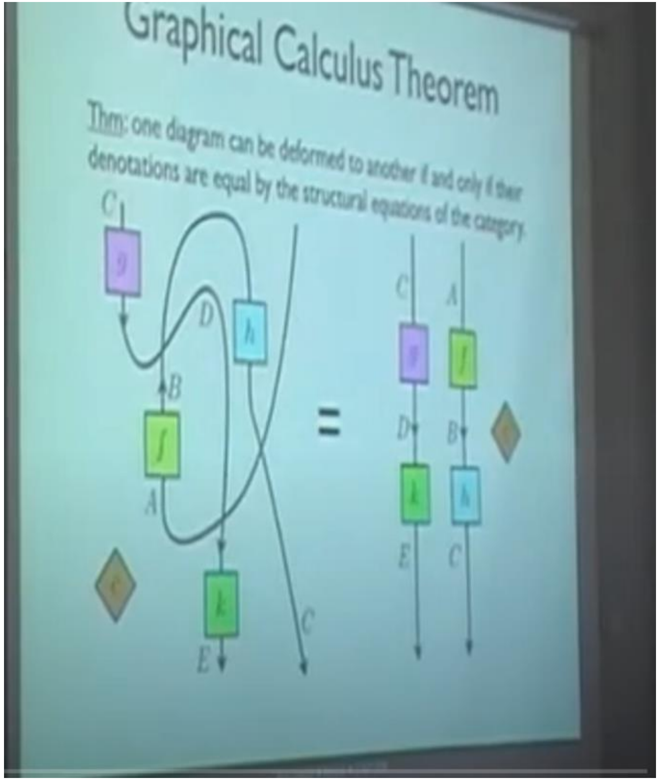
Compact Monoidal Category

- Symmetric Monoidal Category $\mathcal{C}$ where:
    - Every object A has a dual object A^*
    - Morphism Unit: $\eta_A : I \rightarrow \mathrm{A}^* \otimes A$
    - Morphism Counit: $\epsilon_A : A \otimes A^* \rightarrow I$
- Combining them yields id_A

CAP CUP

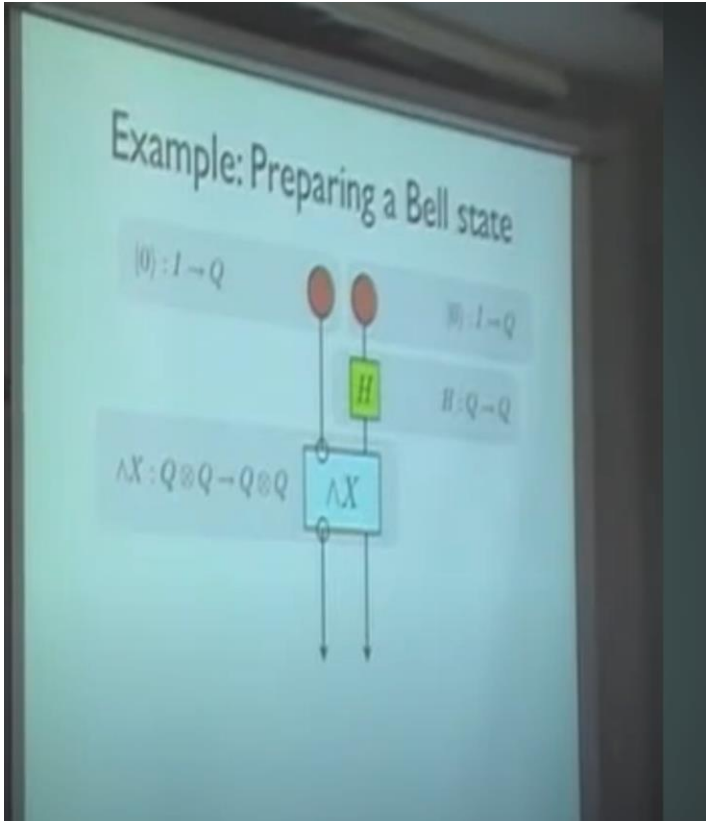
Graphical Calculus

- Visually combine Elements of the Category
  - Process Theory
- Main Idea of ZX-Calculus:
  - Represent Circuit as Network of Processes
  - Apply Simplifications on the Network
- “Only topology Matters”
  - If it looks like the same graph its the same thing
  - Guaranteed by the rules of the Category



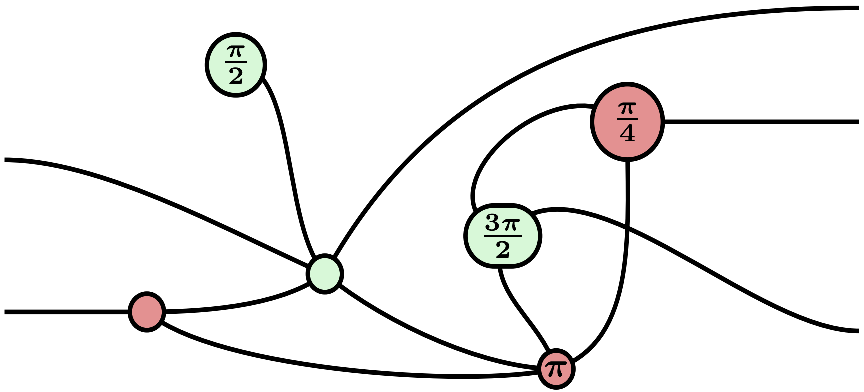
Example Network: CNOT

1. Prepare Qubits
2. Apply Hadamard
3. Apply CNTOT



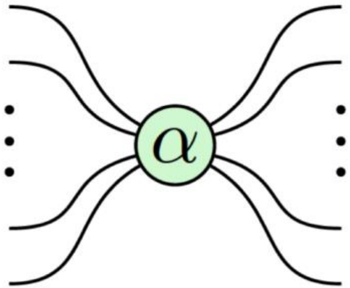
ZX-Notation

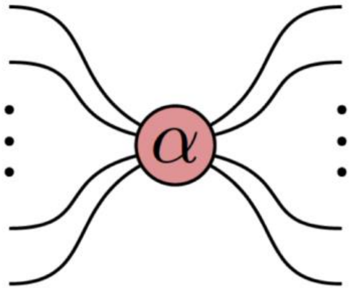
- Circuits can be represented visually
  - Everything is based on mathematical rules
- The representation is very simple:
  - Spiders
  - Lines
- We will see ZX-Calculus is universal



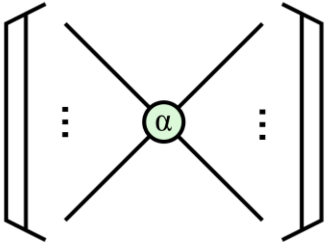
Spiders

- Nodes in the graph
- Arbitrary number of inputs / outputs
- Two Colors:
  - Green (Z-Basis)
  - Red (X-Basis)
- Phase angle α possible





Spiders as linear maps



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

- Each spider is a linear map
- $GreenSpider(n,m)_\alpha = |\underbrace{0 \dots 0}_{m}\rangle\langle\underbrace{0 \dots 0}_{n}| + e^{i\alpha}|\underbrace{1 \dots 1}_{m}\rangle\langle\underbrace{1 \dots 1}_{n}|$
- $RedSpider(n,m)_\alpha = \underbrace{| + \cdots +\rangle}_{m}\underbrace{\langle + \cdots + |}_{n} + e^{i\alpha}\underbrace{| - \cdots -\rangle}_{m}\underbrace{\langle - \cdots - |}_{n}$
- Example:
  - $GreenSpider(5,3)$ is associated with a $2^3 \times 2^5 = 8 \times 32$ matrix
  - Not unitary, not even square

Example Spiders: Basis States

- $GreenSpider(0,1)_0 = |0\rangle \cdot 1 + e^{i \cdot 0}|1\rangle \cdot 1 = \begin{bmatrix} 1 \\ 1 \end{bmatrix} \propto |+\rangle$

- $GreenSpider(0,1)_\pi = |0\rangle \cdot 1 + e^{i \cdot \pi}|1\rangle \cdot 1 = \begin{bmatrix} 1 \\ -1 \end{bmatrix} \propto |-\rangle$

- $RedSpider(0,1)_0 = |+\rangle \cdot 1 + e^{i \cdot 0}|-\rangle \cdot 1 = \begin{bmatrix} 2 \\ 0 \end{bmatrix} \propto |0\rangle$

- $RedSpider(0,1)_\pi = |+\rangle \cdot 1 + e^{i \cdot \pi}|-\rangle \cdot 1 = \begin{bmatrix} 0 \\ 2 \end{bmatrix} \propto |1\rangle$

Example Spiders: Pauli Matrices

- $GreenSpider(1,1)_{\pi} = |0\rangle\langle 0| + e^{i\cdot\pi}|1\rangle\langle 1| = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} = Z$

- $RedSpider(1,1)_{\pi} = |+\rangle\langle +| + e^{i\cdot\pi}|-\rangle\langle -| = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} = X$

Example Spiders: Identity Matrix

- $GreenSpider(1,1)_0 = |0\rangle\langle 0| + e^{i\cdot 0}|1\rangle\langle 1| = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = id_2$
- $RedSpider(1,1)_0 = |+\rangle\langle +| + e^{i\cdot 0}|-\rangle\langle -| = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = id_2$

IMG

ID

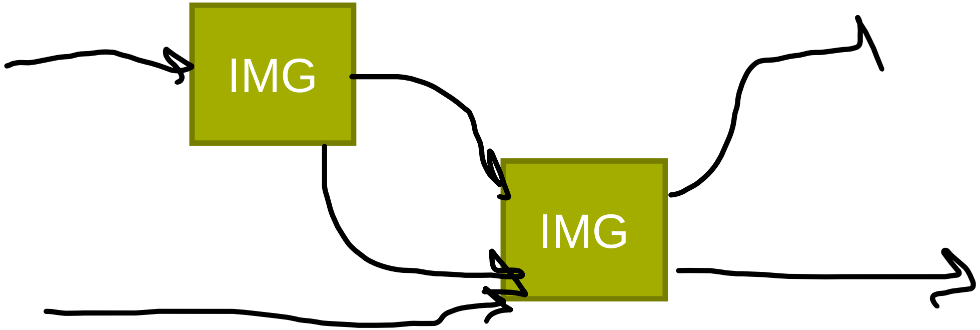
Red IMG

Example Spiders: Bell State

- Spiders can generate entangled States
- $GreenSpider(0,2)_0 = |00\rangle \cdot 1 + e^{i\cdot 0}|11\rangle \cdot 1 = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 1 \end{bmatrix} \propto |\Phi^+\rangle$

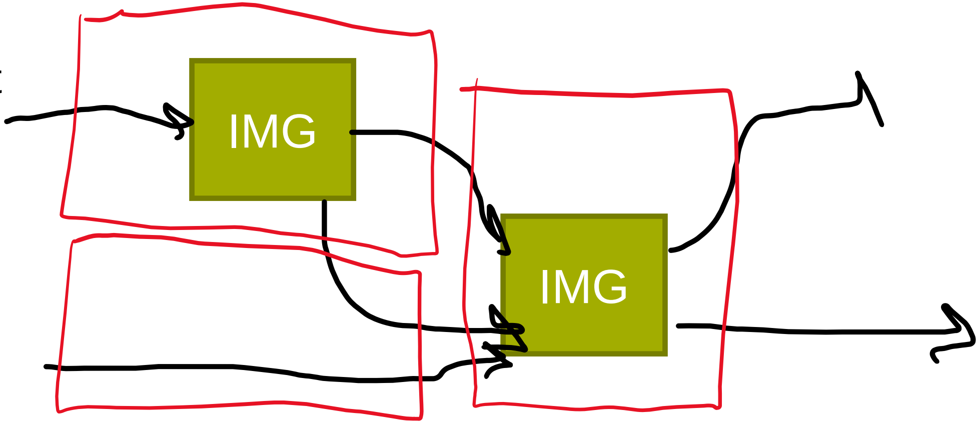
Combining Spiders

- Connect output lines of a spider with input lines of another spider
  - Again: Only the topology matters
- The resulting Graph can represent a Quantum Circuit



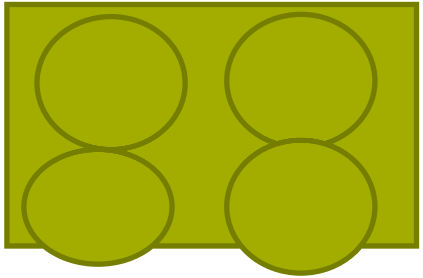
Evaluating a Graph of Spiders

- We divide the graph into regions
  - Each region must contain exactly one spider
- Like normal quantum circuits:
  - „parallel“ Parts are combined using the tensors product ($\otimes$)
  - „serial” Parts are combined using matrix product ($\circ$)
- We get a matrix representation of the circuit
- Based on category theory



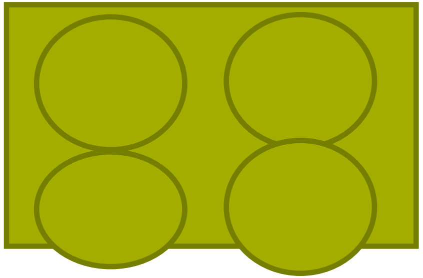
Example: CNOT

1. Evaluate parallel Sections
   - A= $GreenSpider(1,1) \otimes RedSpider(2,1) = id_2 \otimes RedSpider(2,1)$
   - B= $GreenSpider(1,2) \otimes RedSpider(1,1) = GreenSpider(2,1) \otimes id_2$



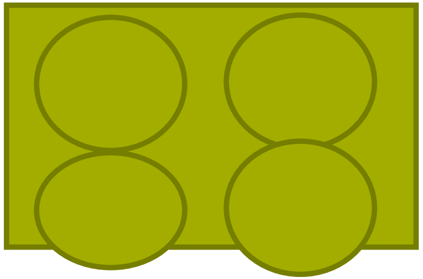
2. Combine sequential Regions
   - $R = A \circ B$

Example: CNOT



- Combine parallel Sections:
  - $A = \mathrm{id}_2 \otimes (\, |+\rangle\langle + \; +|+|-\rangle\langle - \; -|\,) = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \otimes \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{bmatrix} = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \end{bmatrix}$
  - $B = (\, |00\rangle\langle 0|+|11\rangle\langle 1|\,) \otimes id_2 = \begin{bmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 1 \end{bmatrix} \otimes \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$

Example: CNOT



- Combine sequential Sections:
  - $R = A \circ B = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \end{bmatrix} \circ \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \propto CNOT$

- It works! But evaluating the circuit this way is just as bad as a classical matrix approach

Remark: Where to draw Regions?

- For larger Graphs there are multiple ways of drawing the regions
  - Obvious as you are allowed to move the components around (“Only topology mattes”)
- This leads to different matrices in the calculation process
- But the final matrices are always equivalent up to a scalar factor

Res matrix

Simplification Rules

- We don’t want to calculate the graph using its matrix form
- There exist many rules to simplify ZX-Graphs
  - But far fewer rules as for classical circuits
- We can apply the rules anywhere in the graph aslong:
  - The pattern for the substitution matches
  - The order of the input / output wires of regions are unchanged

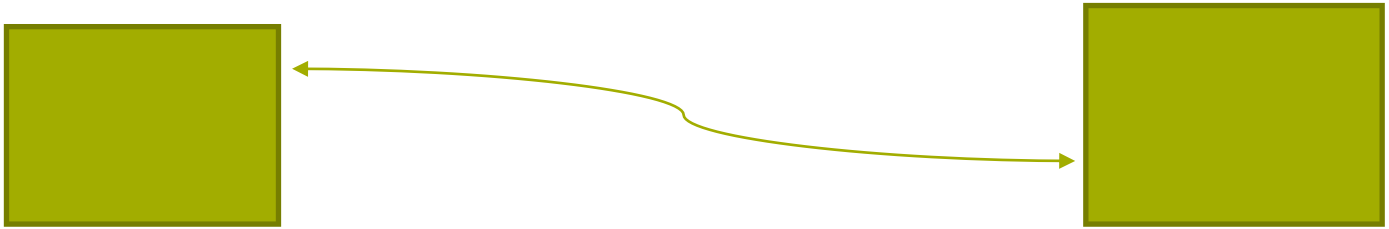


Image Sources

- Fig. 1: ZX-Circuit https://upload.wikimedia.org/wikipedia/commons/5/50/Zx-diagram-example.svg
- Fig. 2: Circuit Identities https://www.cs.ox.ac.uk/people/aleks.kissinger/slides/qnlp-40mins.pdf
- Fig. 3: Quantum Compilation https://www.researchgate.net/figure/Quantum-circuit-compilation-47_fig15_348930917
- Fig. 4: Fault tolerant Toffoli Gate https://www.researchgate.net/figure/The-fault-tolerant-Clifford-T-implementations-of-quantum-logic-gates-used-in-this-work_fig1_322049116