

Title: The Power of ZX Calculus: Compiling Quantum Codes

Speakers: Andrey Khesin

Series: Perimeter Institute Quantum Discussions

Date: December 04, 2023 - 2:30 PM

URL: <https://pirsa.org/23120029>

Abstract: Quantum error-correcting codes are a key pillar of quantum computing. They allow for the recovery of quantum information in the presence of noise. There are three main ways to depict a quantum code: by the list of codewords, by an encoding circuit, or by a stabilizer tableau. Although the latter is used most often, all three of these have pros and cons. In this talk, I will showcase ZX calculus, a graphical language that can be used to talk about quantum states, circuits, measurements, and importantly, codes. I will demonstrate its power and versatility and showcase a canonical form for states, circuits, and codes. The power of this theorem allows us to compile these quantum objects to a unique, elegant, and simple form which serves as a much better depiction than any of the three methods above.

Zoom link <https://pitp.zoom.us/j/91431057660?pwd=cW9xKzNXejdIekp3amFacFZ5OHN5dz09>

Reopen closed PDFs ✕
Acrobat closed abruptly. You can restore the PDFs you had open.
No, thanks **Restore**

The Power of ZX Calculus: Compiling Quantum Codes

Andrey Boris Khesin

MIT

Perimeter Institute

December 4, 2023

About me and Acknowledgements

Thank you for inviting me!

- Me: 5th year Math graduate student at MIT
- My research areas: Quantum computing and Quantum information
- My advisor: Peter Shor

In this talk:

- Graph work joint with Kevin Ren (2108.02686) and Alex Hu (2109.10210)
- ZX work joint with Peter Shor and Jonathan Lu (2301.02356)
- Ongoing ZX work joint with Alex Li (Ask me about this!)

Talk Outline

- 1 Motivate a problem in quantum code representation
- 2 Explain downsides of commonly used solutions
- 3 Introduce ZX Calculus and showcase uses
- 4 Solve the problem using ZX Calculus

Big Goal

Change the **language** we use to
discuss quantum codes

Big Goal

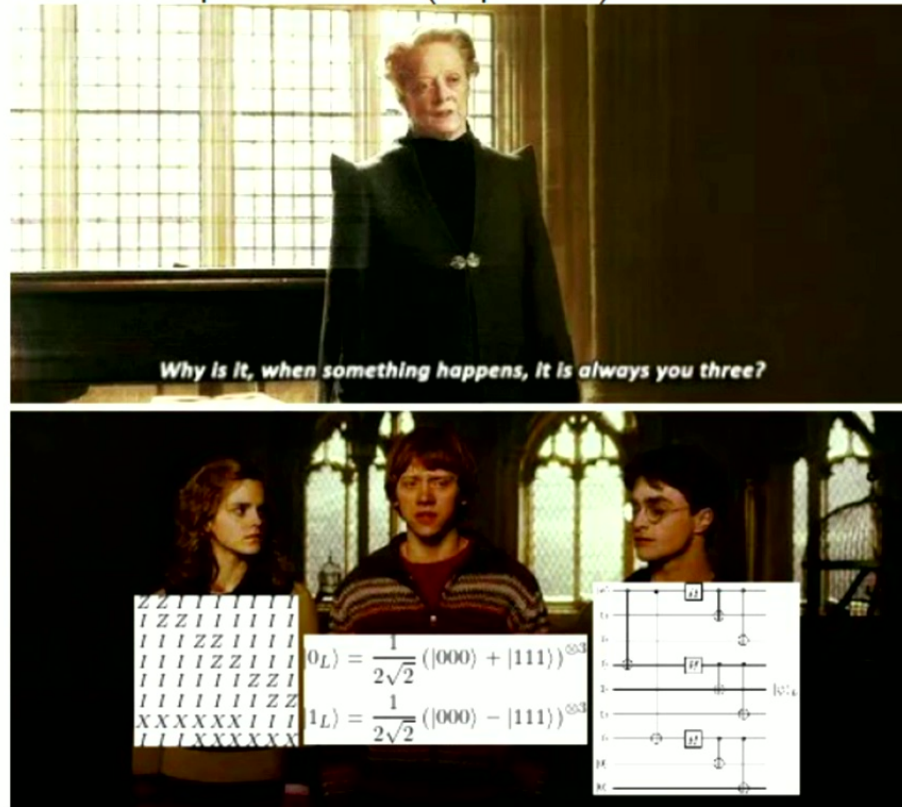
Change the **language** we use to
discuss quantum codes

Quantum Error-Correcting Codes

- Goal: Transmit quantum state in the presence of noise
 - State (2 qubits): $a|00\rangle + b|01\rangle + c|10\rangle + d|11\rangle$
 - Noise: Pauli matrices
$$X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \quad Y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix} \quad Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$$
- Problem: Want to copy quantum information, but cannot!
- Using entanglement, we can add redundancy without copying
- First QEC code: Shor's 9-qubit code in 1995
 - Encodes 1 qubit into 9 and can correct any single-qubit error

An Innocuous Question

1) Depict Shor's 9-qubit code. (5 points)



Andrey Boris Khesin (MIT)

The Power of ZX Calculus

Perimeter Institute
6 / 33

December 4, 2023

Express Codewords Directly (Bras and Kets)

$$|0_L\rangle = (|000\rangle + |111\rangle)^{\otimes 3} \quad |1_L\rangle = (|000\rangle - |111\rangle)^{\otimes 3}$$

Pros:

- 1 Tells you what you are getting
- 2 Unique presentation*
- 3 Shor code is compact

Cons:

- 1 Constructing? Decoding? Entanglement structure?

Express Codewords Directly (Bras and Kets)

$$|0_L\rangle = (|000\rangle + |111\rangle)^{\otimes 3} \quad |1_L\rangle = (|000\rangle - |111\rangle)^{\otimes 3}$$

Pros:

- 1 Tells you what you are getting
- 2 Unique presentation*
- 3 Shor code is compact

Cons:

- 1 Constructing? Decoding? Entanglement structure?
- 2 $|0_L\rangle = \frac{1}{4}(|00000\rangle + |10010\rangle + |01001\rangle + |10100\rangle + |01010\rangle - |11011\rangle - |00110\rangle - |11000\rangle - |11101\rangle - |00011\rangle - |11110\rangle - |01111\rangle - |10001\rangle - |01100\rangle - |10111\rangle + |00101\rangle),$
 $|1_L\rangle = \frac{1}{4}(|11111\rangle + |01101\rangle + |10110\rangle + |01011\rangle + |10101\rangle - |00100\rangle - |11001\rangle - |00111\rangle - |00010\rangle - |11100\rangle - |00001\rangle - |10000\rangle - |01110\rangle - |10011\rangle - |01000\rangle + |11010\rangle)$

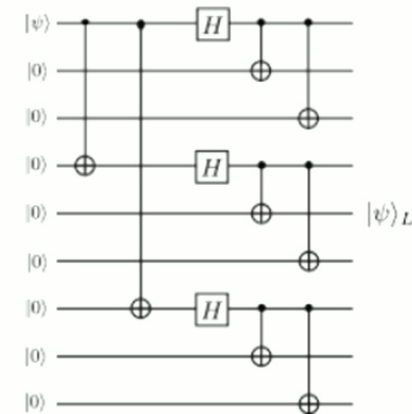
Circuit Diagram

Pros:

- 1 Construction ✓
- 2 Also tells you what you are getting
- 3 Partial recipe for recovery map
- 4 Some entanglement structure visible

Cons:

- 1 Not unique
- 2 Arbitrarily long/complicated
- 3 Identical qubits look asymmetric



Stabilizer Presentation

Pros:

- 1 Compact ✓
- 2 Construction*
- 3 Tells us how to measure and correct ✓

```
ZZIIIIIII  
IZZIIIIII  
IIIZZIIII  
IIIIZZIII  
IIIIIIZZI  
IIIIIIIZZ  
XXXXXIII  
IIIXXXXX
```

Cons:

- 1 Poor entanglement structure
- 2 Identical qubits still look asymmetric
- 3 Construction involves post-selection
- 4 Not unique!

```
-IIIIYXYXX  
XYYIZZXY  
ZIZYYXYXY  
-XXXYYXIII  
ZZIYYXYXY  
YXYIZZYYX  
YYXXYYZZI  
-XXXZIZYXY
```

There are more presentations

- We covered methods taught in quantum classes
- Tanner graphs are nice, but are for a specific code family
- Could use graphs to express states in other ways (Kheisin and Ren) (Ask me about this!)
- Surface codes give us nice visuals, but can we generalize them?

What is ZX Calculus?

Spiders!

- Tensor network made of just two types of spiders, “Z” and “X”
 - Align with familiar Pauli Z and X
 - Drawn as green and red nodes, resp.
 - Nodes have a single phase, α (0 if blank)
 - Edges are inner products
 - Ignores all scalars! (Ask me about this!)
 - If a spider has n input legs and m output legs...

What is ZX Calculus?

Spiders!

- Tensor network made of just two types of spiders, “ Z ” and “ X ”
- Align with familiar Pauli Z and X
- Drawn as green and red nodes, resp.
- Nodes have a single phase, α (0 if blank)
- Edges are inner products
- Ignores all scalars! (Ask me about this!)
- If a spider has n input legs and m output legs...

What is ZX Calculus?

Spiders!

- Tensor network made of just two types of spiders, “Z” and “X”
- Align with familiar Pauli Z and X
- Drawn as green and red nodes, resp.
- Nodes have a single phase, α (0 if blank)
- Edges are inner products
- Ignores all scalars! (Ask me about this!)
- If a spider has n input legs and m output legs...

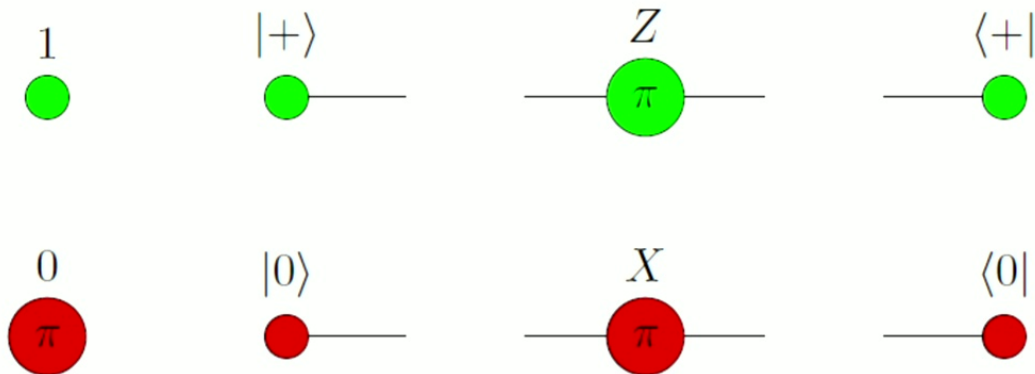
$$\text{Green Spider } \alpha = \begin{cases} |0\rangle^{\otimes n} \mapsto |0\rangle^{\otimes m} \\ |1\rangle^{\otimes n} \mapsto e^{i\alpha} |1\rangle^{\otimes m}, \end{cases}$$

$$\text{Red Spider } \alpha = \begin{cases} |+\rangle^{\otimes n} \mapsto |+\rangle^{\otimes m} \\ |-\rangle^{\otimes n} \mapsto e^{i\alpha} |-\rangle^{\otimes m}, \end{cases}$$

$$\text{H Spider} = \begin{cases} |0\rangle \mapsto |+\rangle \\ |1\rangle \mapsto |-\rangle. \end{cases}$$

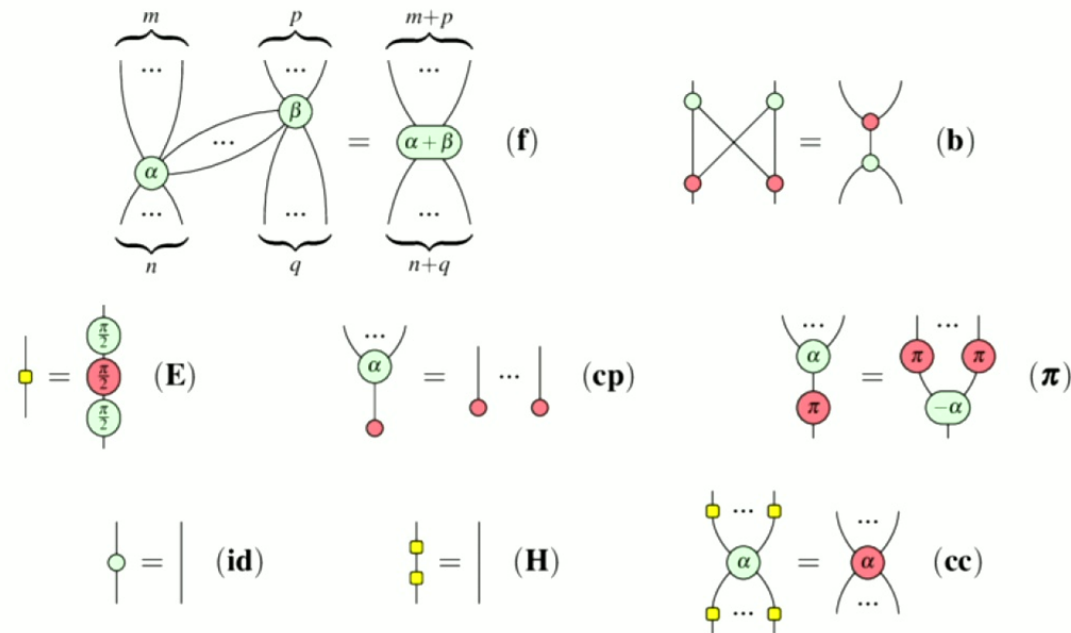
Building Blocks

- Green Z nodes: $|0 \dots 0\rangle \langle 0 \dots 0| + e^{i\alpha} |1 \dots 1\rangle \langle 1 \dots 1|$
- Red X nodes: $|+\dots+\rangle \langle +\dots+| + e^{i\alpha} |-\dots-\rangle \langle -\dots-|$
- $|+\rangle = \frac{|0\rangle+|1\rangle}{\sqrt{2}}$ $|-\rangle = \frac{|0\rangle-|1\rangle}{\sqrt{2}}$



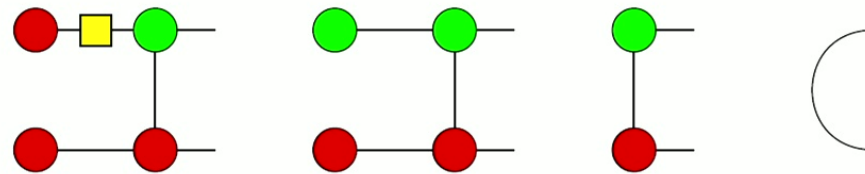
Rewrite rules and corollaries

Like Reidemeister moves for knots

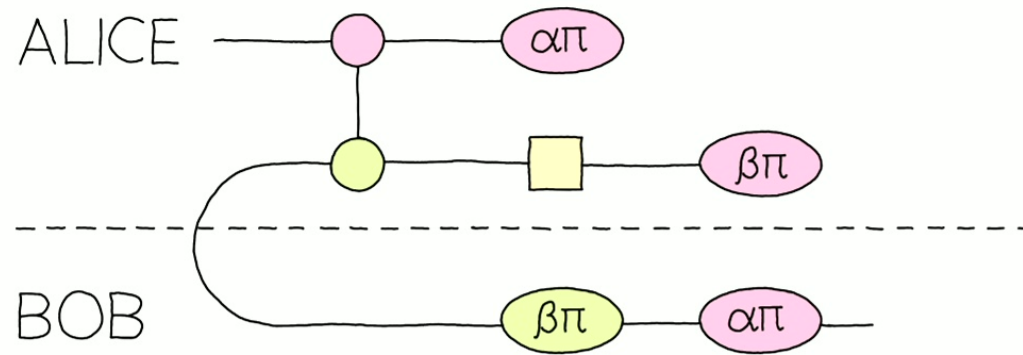


Constructing a Bell Pair

- 1 Start with our usual circuit
 - Two qubits initialized to $|0\rangle$
 - A Hadamard gate on qubit 1
 - A controlled- X gate with qubit 1 as the control
- Hadamard on all incident edges \rightarrow basis change
- Merge same colour nodes and add phases ($0+0=0$)
- Delete phase 0 nodes with exactly 2 incident edges

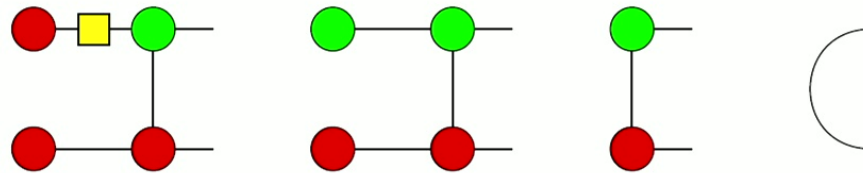


Quantum Teleportation

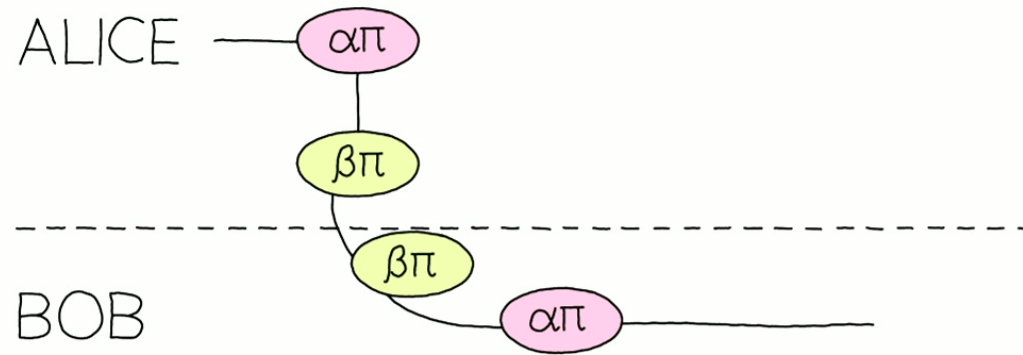


Constructing a Bell Pair

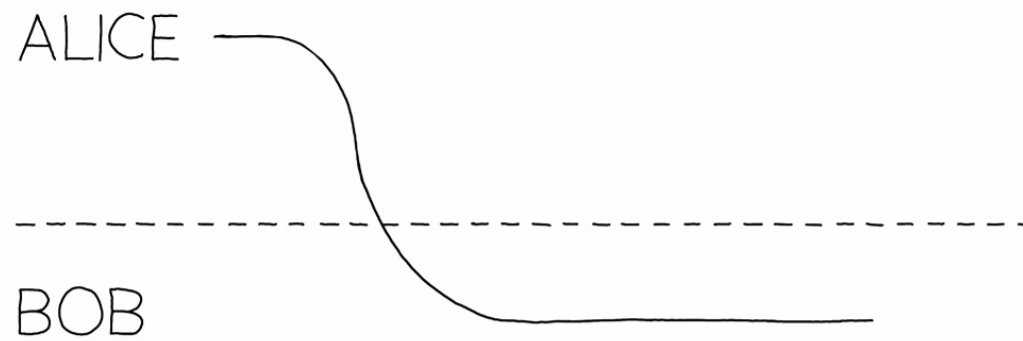
- 1 Start with our usual circuit
 - Two qubits initialized to $|0\rangle$
 - A Hadamard gate on qubit 1
 - A controlled- X gate with qubit 1 as the control
- 2 Hadamard on all incident edges \rightarrow basis change
- 3 Merge same colour nodes and add phases ($0+0=0$)
- 4 Delete phase 0 nodes with exactly 2 incident edges



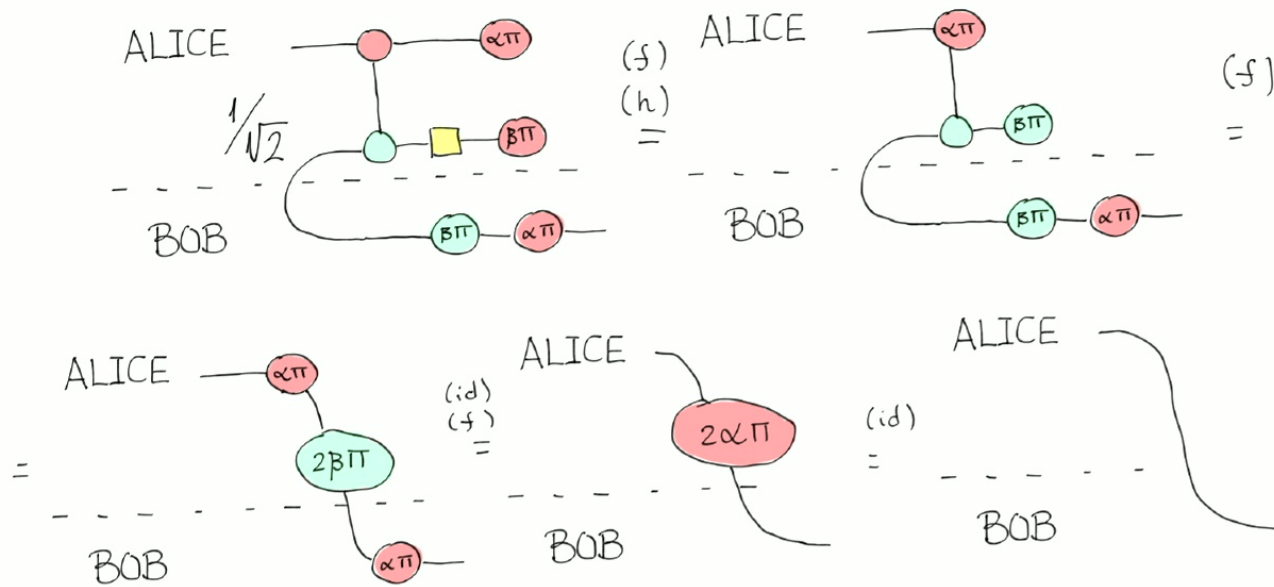
Quantum Teleportation



Quantum Teleportation



Quantum Teleportation



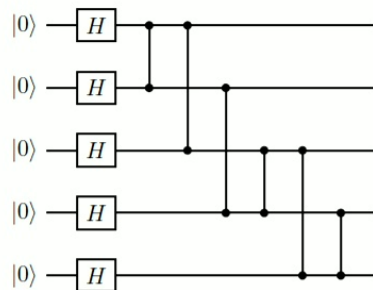


Graph States

- Some quantum states are *graph states*
- State: For $G = (V, E)$,

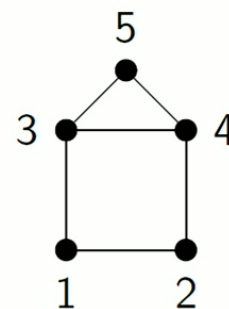
$$|G\rangle = \prod_{(a,b) \in E} CZ_{a,b} |+\rangle^{\otimes n}$$

- Circuit:


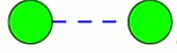


- Stabilizer tableau:


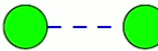
X	Z	Z	I	I
Z	X	I	Z	I
Z	I	X	Z	Z
I	Z	Z	X	Z
I	I	Z	Z	X

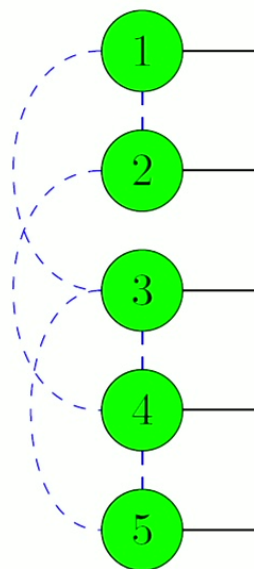
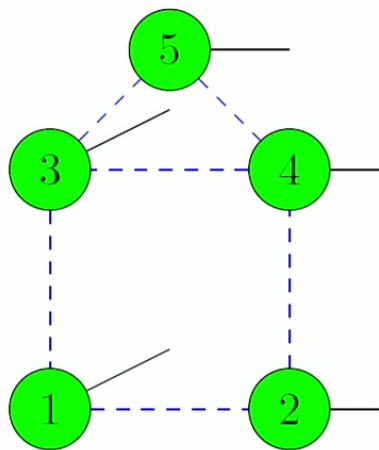


Graph States in ZX Calculus

- In ZX Calculus, $|+\rangle$ is , while CZ is .
- Double edges “cancel” (Hopf rule) so we can “toggle” edges

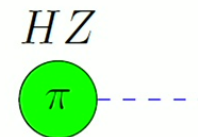
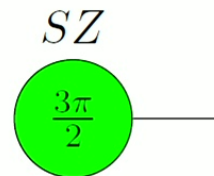
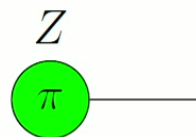
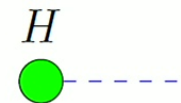
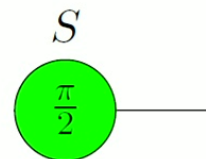
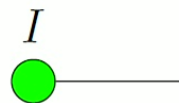
Graph States in ZX Calculus

- In ZX Calculus, $|+\rangle$ is , while CZ is .
- Double edges “cancel” (Hopf rule) so we can “toggle” edges



Big Idea

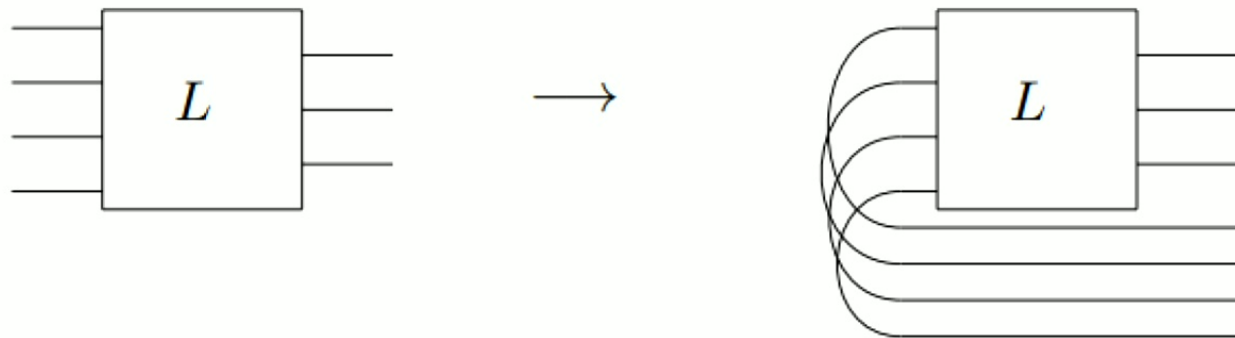
- Can get *any Clifford state* by adding operations to output edges!
- Operations need only be one of I , Z , S , SZ , H , or HZ



Theorem: States have a unique canonical form where no H can be “moved down”! (Khesin and Hu)

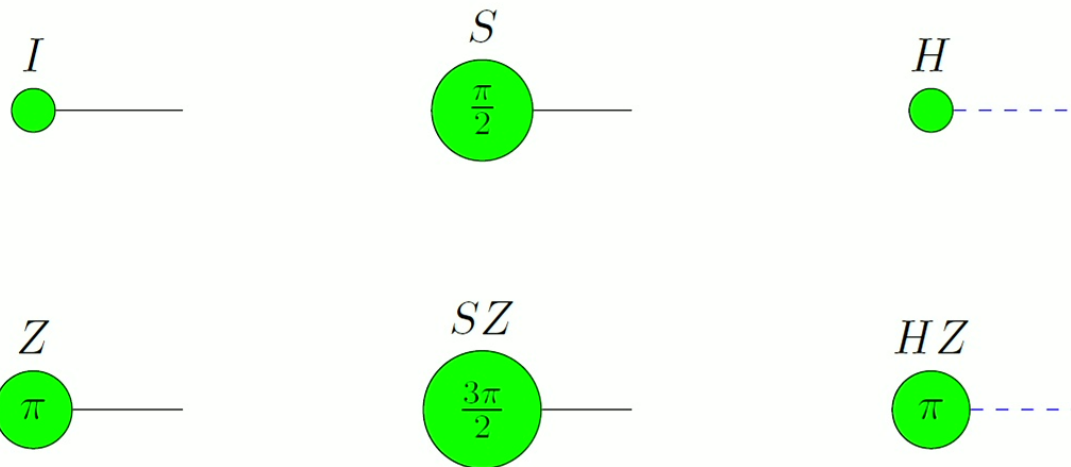
Another big idea

- **Circuits are states!** (and vice versa)
- Turn wires around/turn bras into kets/Choi-Jamiołkowski isomorphism
- Have canonical form for *Clifford circuits*!
- Also have efficient compiler!



Big Idea

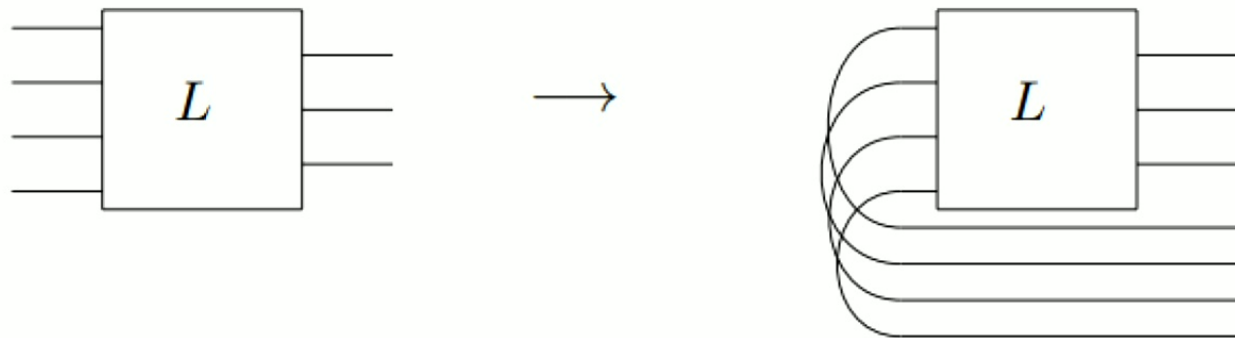
- Can get *any Clifford state* by adding operations to output edges!
- Operations need only be one of I , Z , S , SZ , H , or HZ



Theorem: States have a unique canonical form where no H can be “moved down”! (Khesin and Hu)

Another big idea

- **Circuits are states!** (and vice versa)
- Turn wires around/turn bras into kets/Choi-Jamiołkowski isomorphism
- Have canonical form for *Clifford circuits*!
- Also have efficient compiler!



Quantum Codes

- But what about **quantum codes**?
- A code is a set of codewords
- A code is a stabilizer tableau
- Applying arbitrary unitary operations to the *input* nodes does nothing!
- Can we find a canonical form for this object?

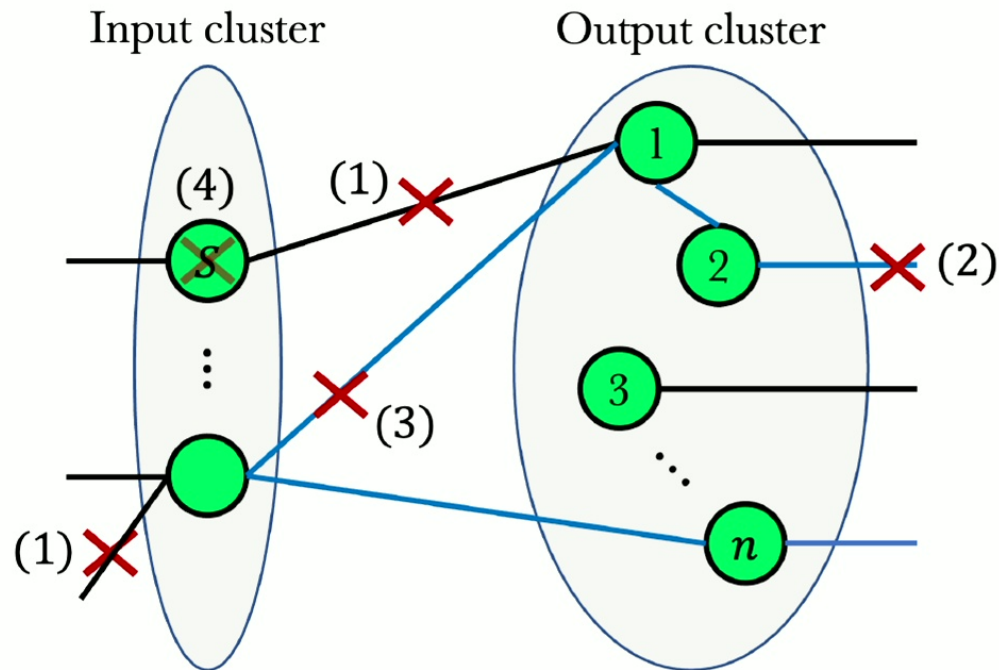
Main Result

Theorem: Codes have a canonical form! (Khesin, Lu, and Shor)

- ① Edge Rule: One Z node per free edge, every internal edge Hadamarded
- ② Hadamard Rule: No Hadamard output connected to lower-numbered node or input
- ③ RREF Rule: Adjacency matrix from inputs to outputs must be in RREF
 - Suppose we have 3 inputs and 5 outputs: $\begin{pmatrix} 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \end{pmatrix}$
 - Columns 1, 2, and 4 are *pivot columns*
- ④ Clifford Rule: No operations (including edges) on input nodes or pivot nodes

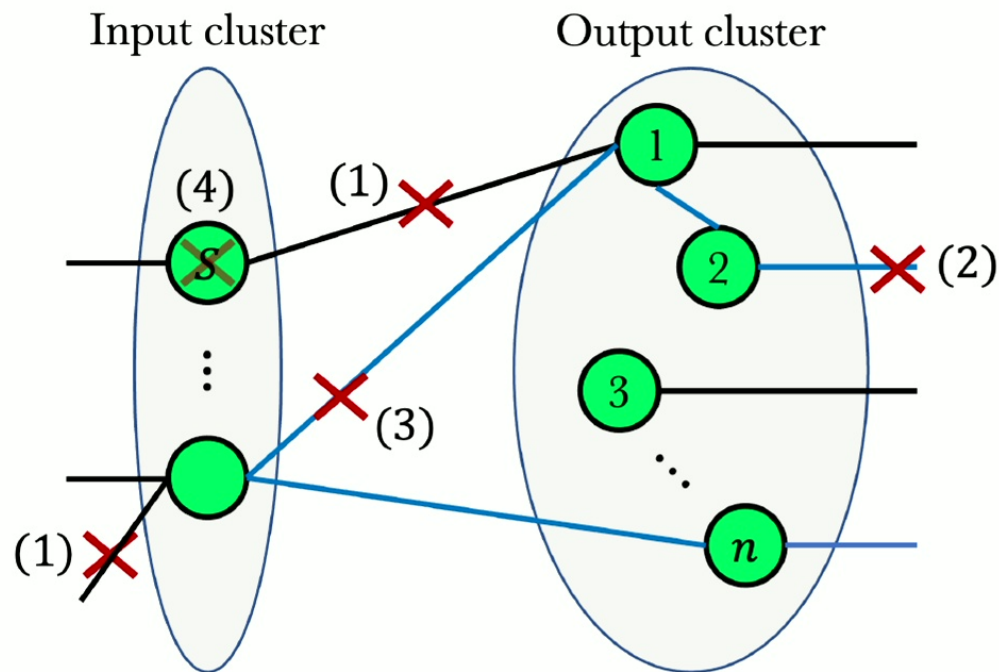
A non-example

- 1) Edge 2) Hadamard 3) RREF 4) Clifford



A non-example

- 1) Edge 2) Hadamard 3) RREF 4) Clifford



A proof

The number of tableaux with k stabilizers and n qubits is

$$\frac{\prod_{i=1}^k 2 \cdot 4^n / 2^{i-1} - 2 \cdot 2^{i-1}}{\prod_{i=1}^k 2^k - 2^{i-1}} = \prod_{i=1}^k \frac{2^{2n-i+2} - 2^i}{2^k - 2^{i-1}}$$

The number of canonical forms is

$$f(n, k, p, o) = \begin{cases} 1 & \text{if } n = k = 0 \\ A_{n,k,p,o} + B_{n,k,p,o} & \text{else} \end{cases}$$

$$A_{n,k,p,o} = 2^o f(n-1, k, p+1, o)$$

if $n \neq k$ and $A_{n,k,p,o} = 0$ if $n = k$

$$B_{n,k,p,o} = (2^{2p+o+2} + 2) f(n-1, k-1, p, o+1)$$

if $k \neq 0$ and $B_{n,k,p,o} = 0$ if $k = 0$

Solving the double recursion

$$f(n, k, p, o) = 2^{o(n-k)} \prod_{i=1}^k \frac{(2^{n+1} - 2^i)(2^{n-i+1+2p+o} + 1)}{2^k - 2^{i-1}}$$

Plug in $p = o = 0$ and get $\prod_{i=1}^k \frac{2^{2n-i+2} - 2^i}{2^k - 2^{i-1}}$, the same number as the number of tableaux!

Together with the fact that each code can be transformed into canonical form, we are done!

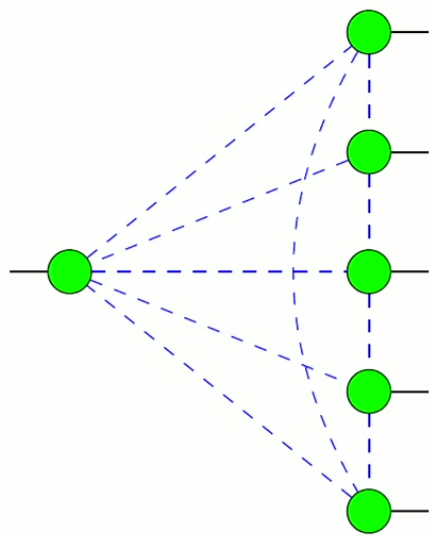
7 qubit code

Andrey Boris Khesin (MIT)

The Power of ZX Calculus

Perimeter Institute December 4, 2023
28 / 33

5 qubit code



Andrey Boris Khesin (MIT)

The Power of ZX Calculus

Perimeter Institute December 4, 2023
27 / 33

Thank you!

Thank you!

Email: khesin@mit.edu

Sign up to chat: tinyurl.com/andreyatperimeter

(Here through Wednesday, Office 359)

The ZX Solution

1) Depict Shor's 9-qubit code. (5 points)

