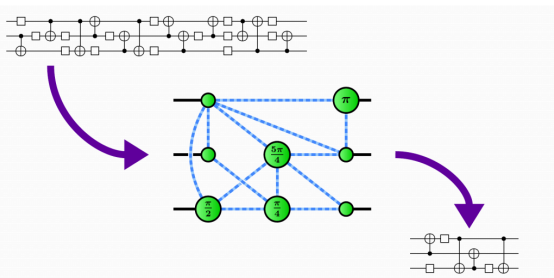
Fault-tolerance by construction

Aleks Kissinger

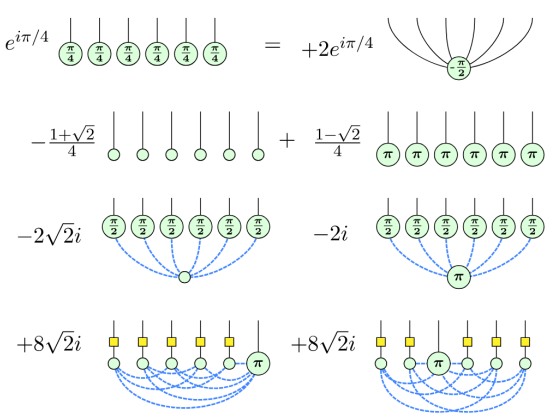
Fault-tolerant Quantum Technologies, Benasque 2026

ZX-calculus

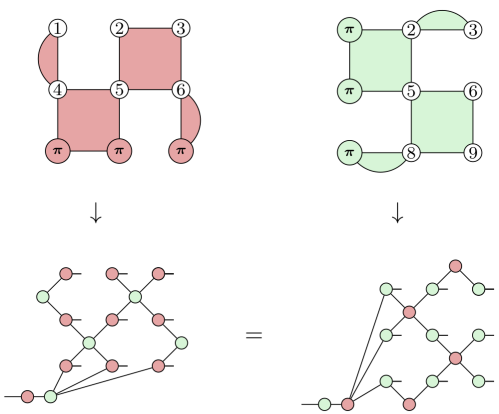
ZX := a handy tool for working with quantum computations using **graph rewriting**



Optimisation

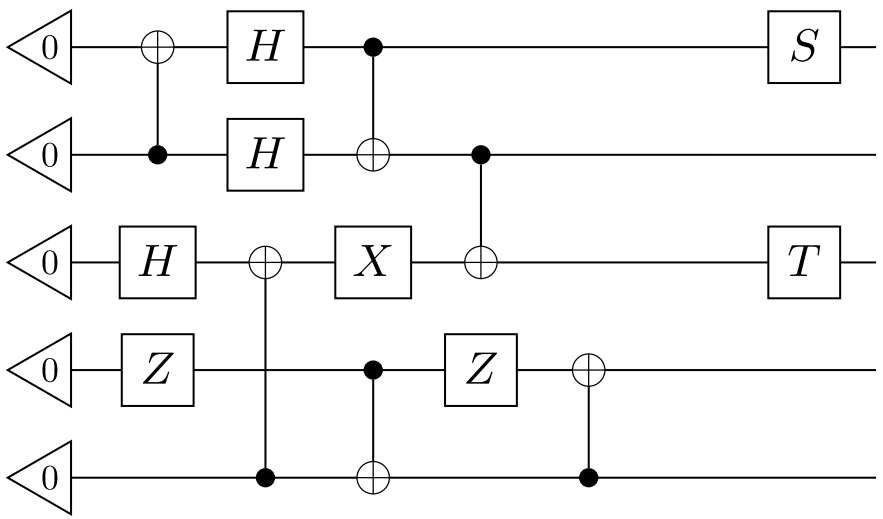


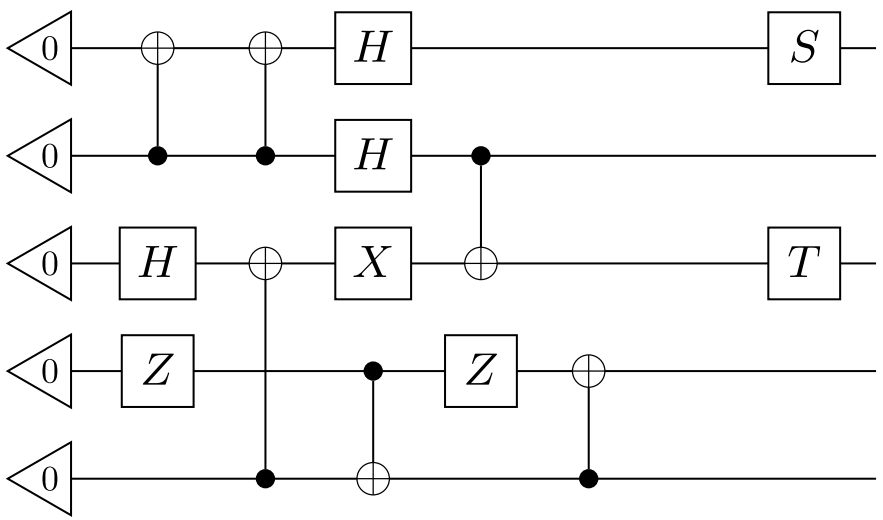
Simulation & Verification

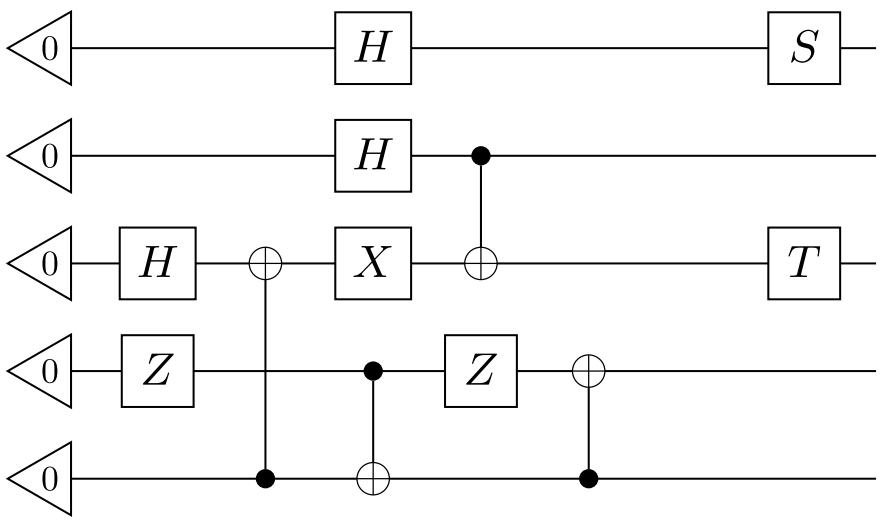


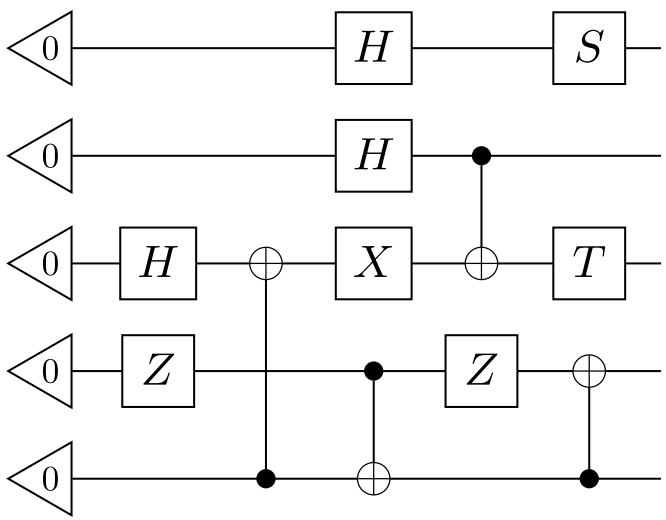
Error Correction

Optimising circuits the old-fashioned way

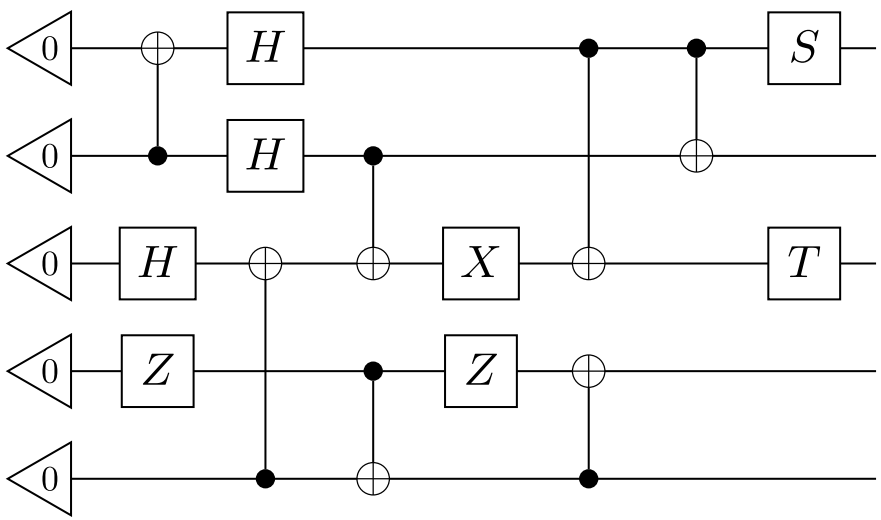

0
0
0
0
0
H
H
H
X
Z
Z
S
T


0
0
0
0
0
H
H
H
X
Z
Z
S
T

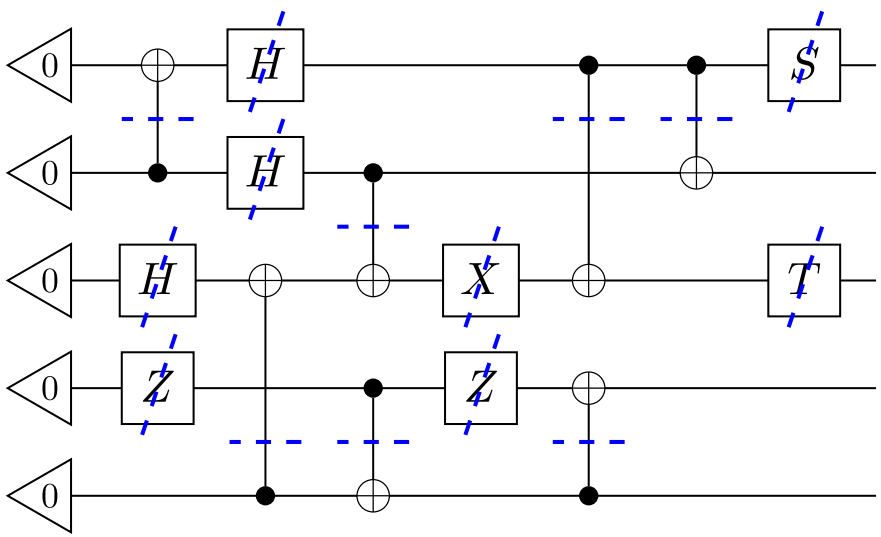

0
H
S
0
H
0
H
X
T
0
Z
Z
0


0
H
S
0
H
0
H
X
T
0
Z
Z
0

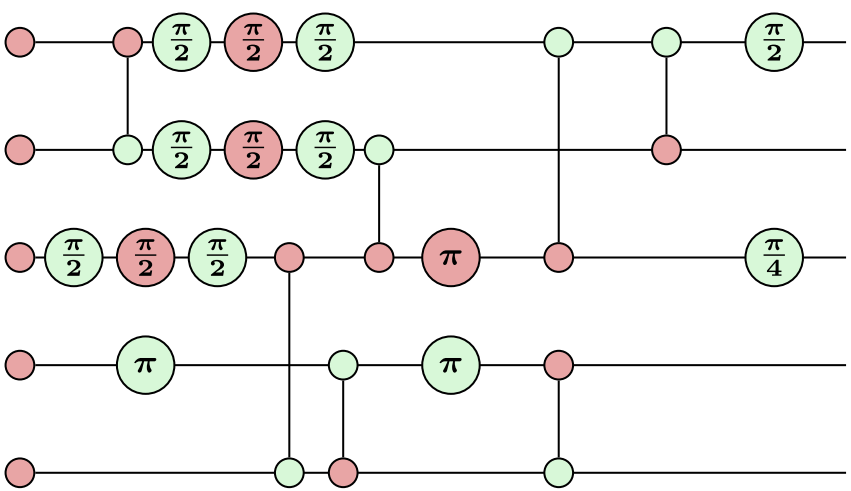
A better idea: decompose a circuit into a ZX diagram



A better idea: decompose a circuit into a ZX diagram

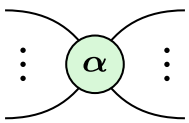


A better idea: decompose a circuit into a ZX diagram

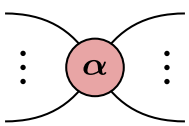


ZX diagrams

- Gates are represented using more basic building blocks, called **spiders**:



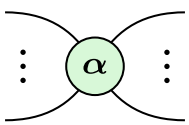
$$:= \quad |0...0\rangle\langle 0...0| + e^{i\alpha}|1...1\rangle\langle 1...1|$$



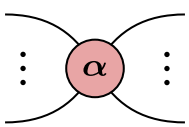
$$:= \quad |+...+\rangle\langle +...+| + e^{i\alpha}|-...-\rangle\langle -...-|$$

ZX diagrams

- Gates are represented using more basic building blocks, called **spiders**:

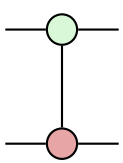


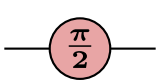
$$:= \quad |0...0\rangle\langle 0...0| + e^{i\alpha}|1...1\rangle\langle 1...1|$$

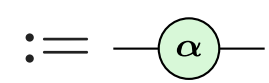


$$:= \quad |+...+\rangle\langle +...+| + e^{i\alpha}|-...-\rangle\langle -...-|$$

- E.g.

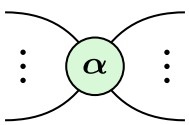
$$CNOT :=$$ 

$$\sqrt{X} :=$$ 

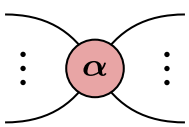
$$Z_\alpha :=$$ 

ZX diagrams

- Gates are represented using more basic building blocks, called **spiders**:

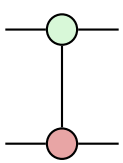


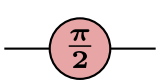
$$:= \quad |0...0\rangle\langle 0...0| + e^{i\alpha}|1...1\rangle\langle 1...1|$$

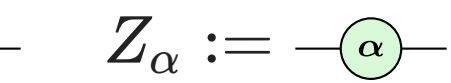


$$:= \quad |+...+\rangle\langle +...+| + e^{i\alpha}|-...-\rangle\langle -...-|$$

- E.g.

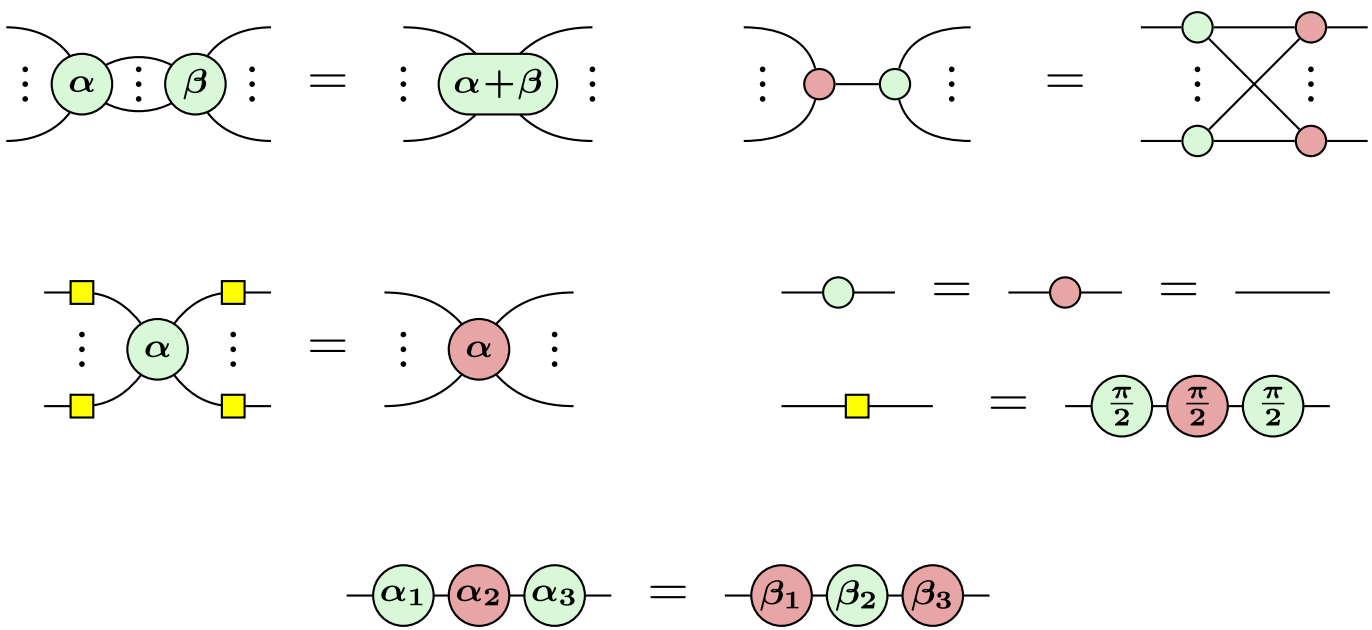
$$CNOT :=$$ 

$$\sqrt{X} :=$$ 

$$Z_\alpha :=$$ 

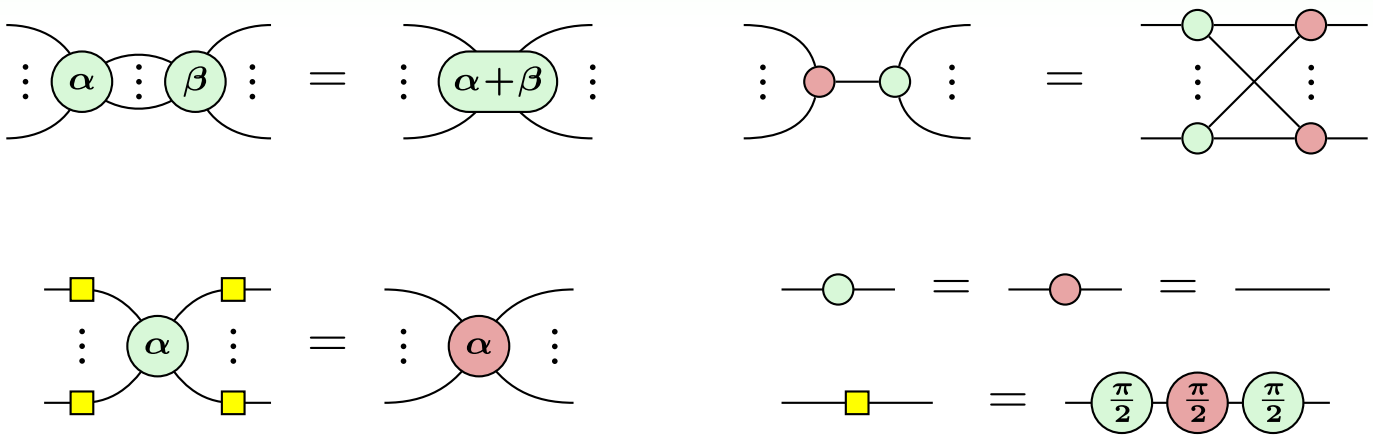
- Wire order doesn't matter $\Rightarrow$ treat ZX-diagrams as **undirected graphs**

...then use the ZX calculus



A complete set of equations for qubit QC

Clifford ZX calculus

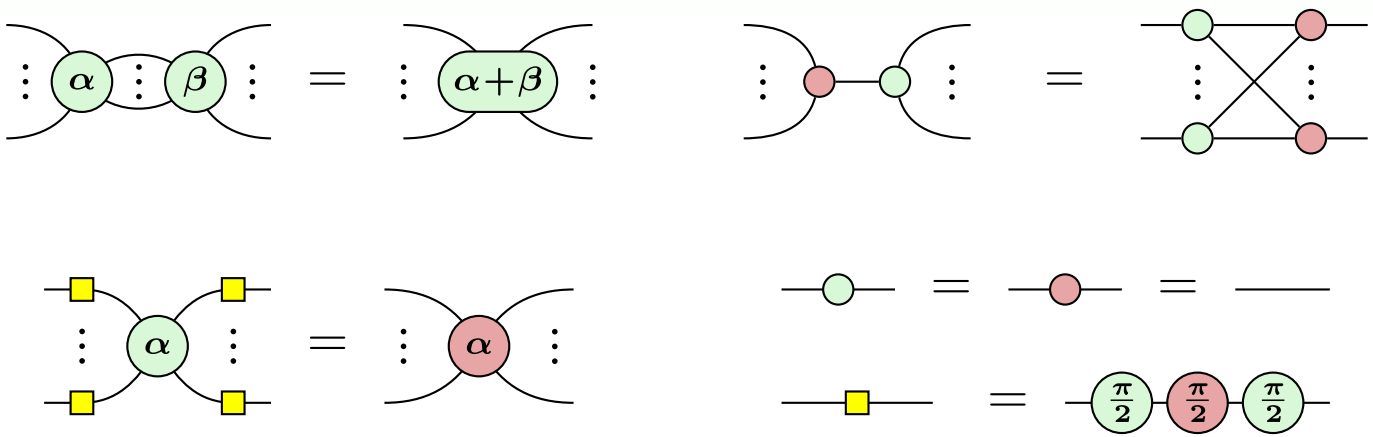


$$\alpha, \beta \in \{0, \tfrac{\pi}{2}, \pi, -\tfrac{\pi}{2}\}$$

A complete set of equations for qubit **Clifford** QC

efficient synthesis, equality checking, classical simulation, ...

Clifford ZX calculus

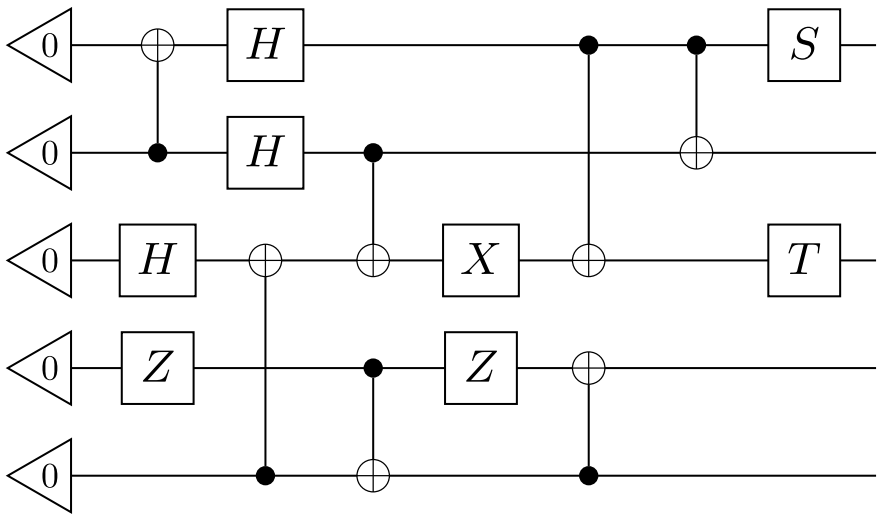


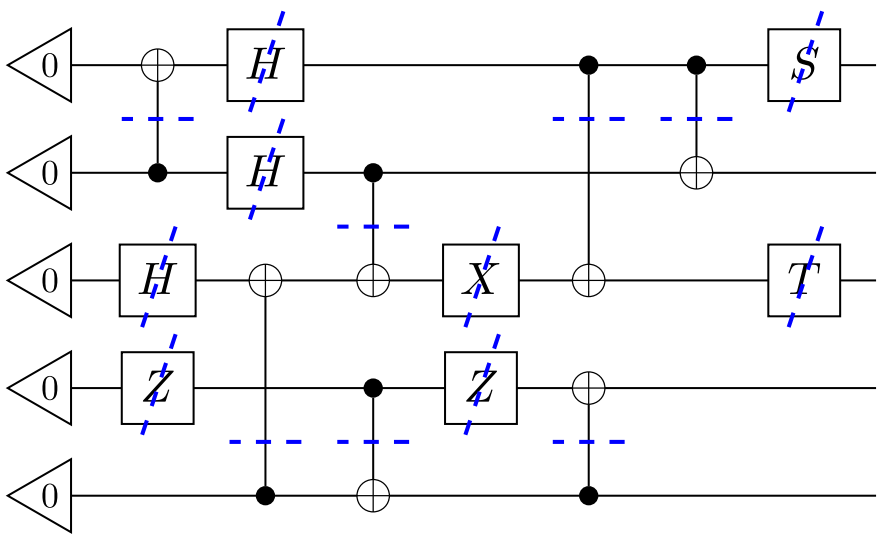
$$\alpha, \beta \in \{0, \tfrac{\pi}{2}, \pi, -\tfrac{\pi}{2}\}$$

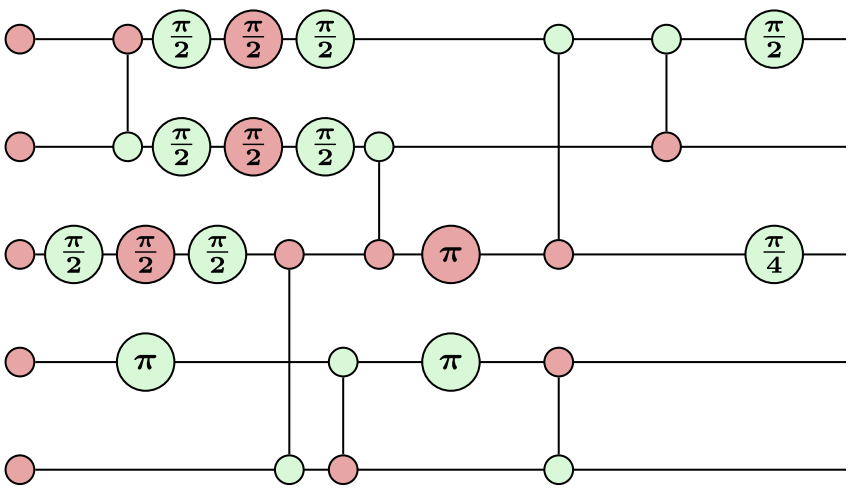
A complete set of equations for qubit **Clifford** QC

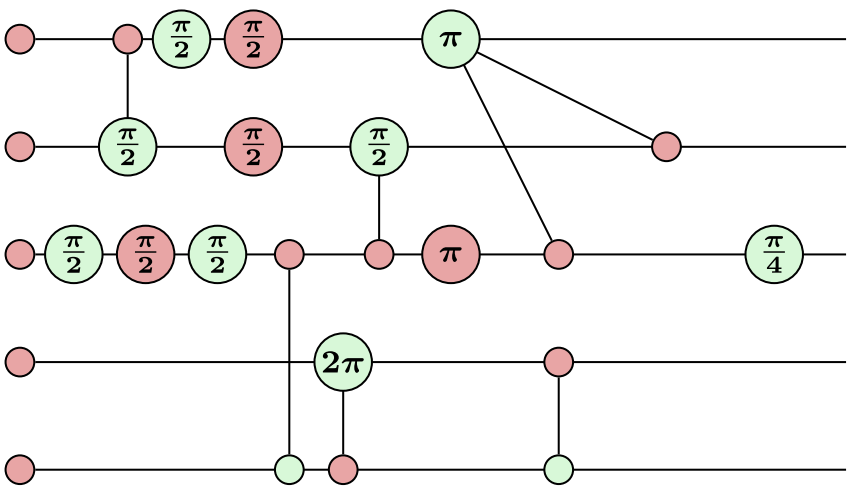
efficient synthesis, equality checking, classical simulation, ...

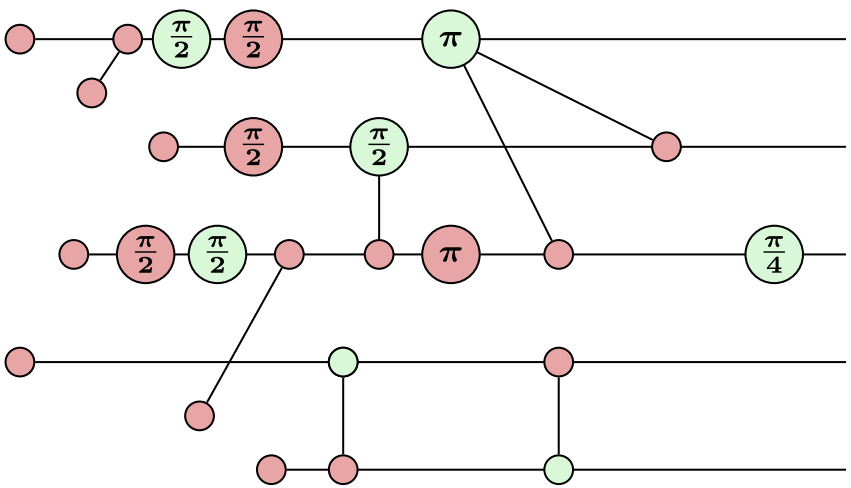
$\approx$ "graphical **stabiliser theory**"

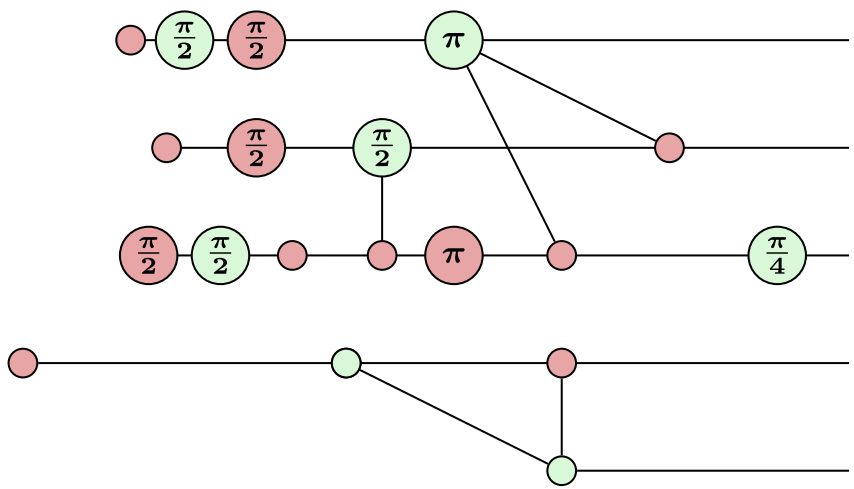

0
0
0
0
0
H
H
H
Z
X
Z
S
T

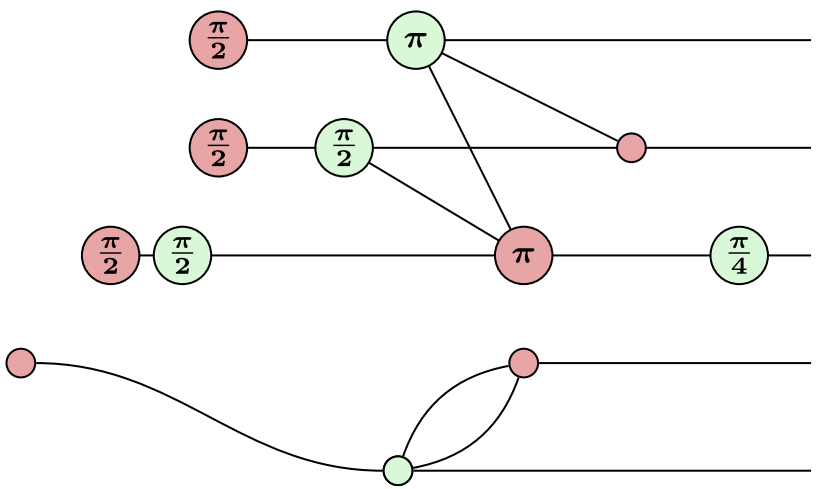

0
0
0
0
0
H
H
H
Z
X
Z
S
T

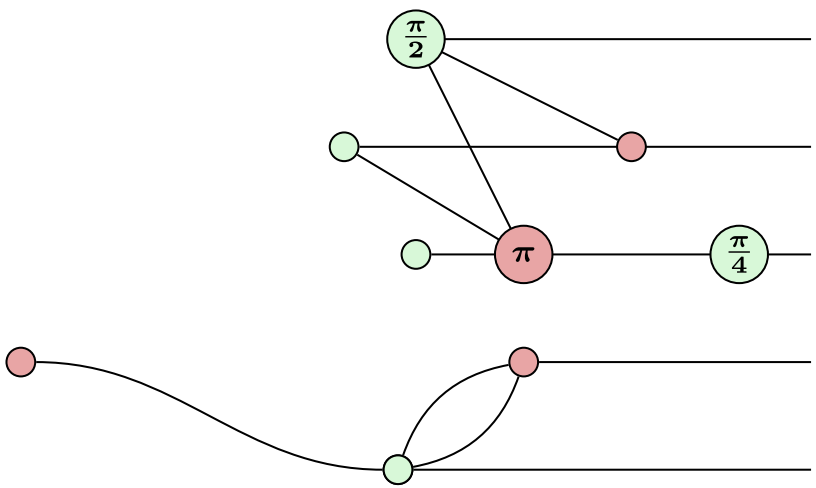

$\frac{\pi}{2}$
$\frac{\pi}{2}$
$\frac{\pi}{2}$
$\frac{\pi}{2}$
$\frac{\pi}{2}$
$\frac{\pi}{2}$
$\frac{\pi}{2}$
$\frac{\pi}{2}$
$\frac{\pi}{2}$
π
$\frac{\pi}{4}$
π
π

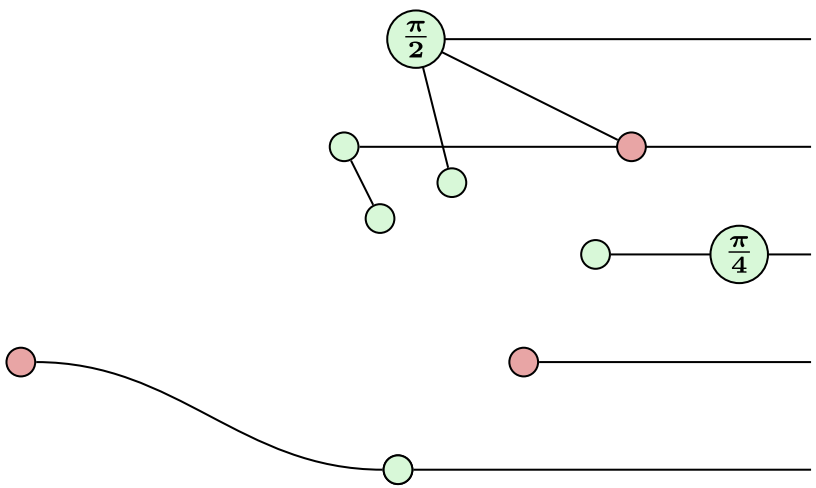

$\frac{\pi}{2}$
$\frac{\pi}{2}$
π
$\frac{\pi}{2}$
$\frac{\pi}{2}$
$\frac{\pi}{2}$
$\frac{\pi}{2}$
$\frac{\pi}{2}$
$\frac{\pi}{2}$
π
$\frac{\pi}{4}$
2π

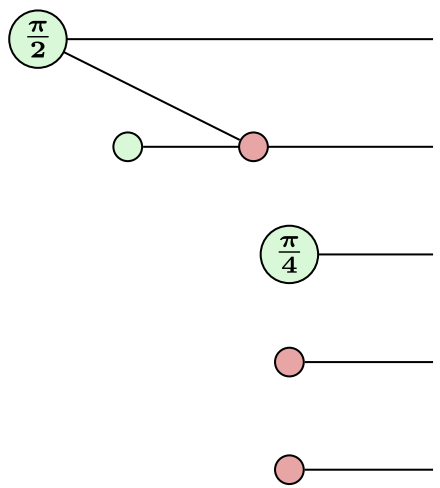

$\frac{\pi}{2}$
$\frac{\pi}{2}$
π
$\frac{\pi}{2}$
$\frac{\pi}{2}$
$\frac{\pi}{2}$
$\frac{\pi}{2}$
π
$\frac{\pi}{4}$

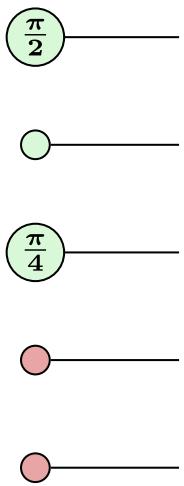

$\frac{\pi}{2}$
$\frac{\pi}{2}$
π
$\frac{\pi}{2}$
$\frac{\pi}{2}$
$\frac{\pi}{2}$
$\frac{\pi}{2}$
π
$\frac{\pi}{4}$


$\frac{\pi}{2}$
π
$\frac{\pi}{2}$
$\frac{\pi}{2}$
$\frac{\pi}{2}$
$\frac{\pi}{2}$
π
$\frac{\pi}{4}$


$\frac{\pi}{2}$
π
$\frac{\pi}{4}$

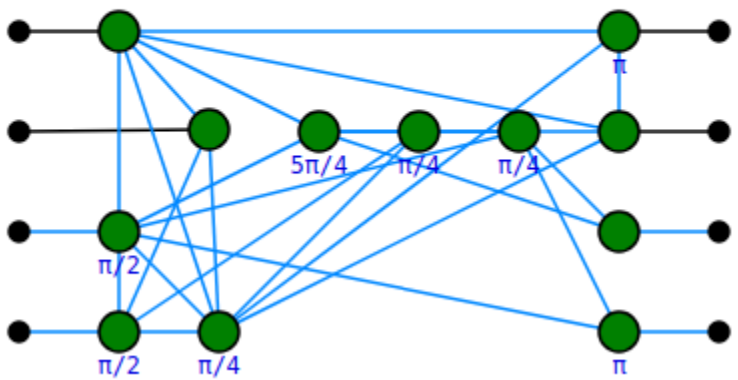

$\frac{\pi}{2}$
$\frac{\pi}{4}$


$\frac{\pi}{2}$
$\frac{\pi}{4}$


$\frac{\pi}{2}$
$\frac{\pi}{4}$

PyZX

```
In [11]: zx.simplify.clifford_simp(g)
         g.normalise()
         zx.d3.draw(g)
```



- Open source Python library for circuit optimisation, experimentation, and education using ZX-calculus

https://github.com/zxcalc/pyzx

QuiZX

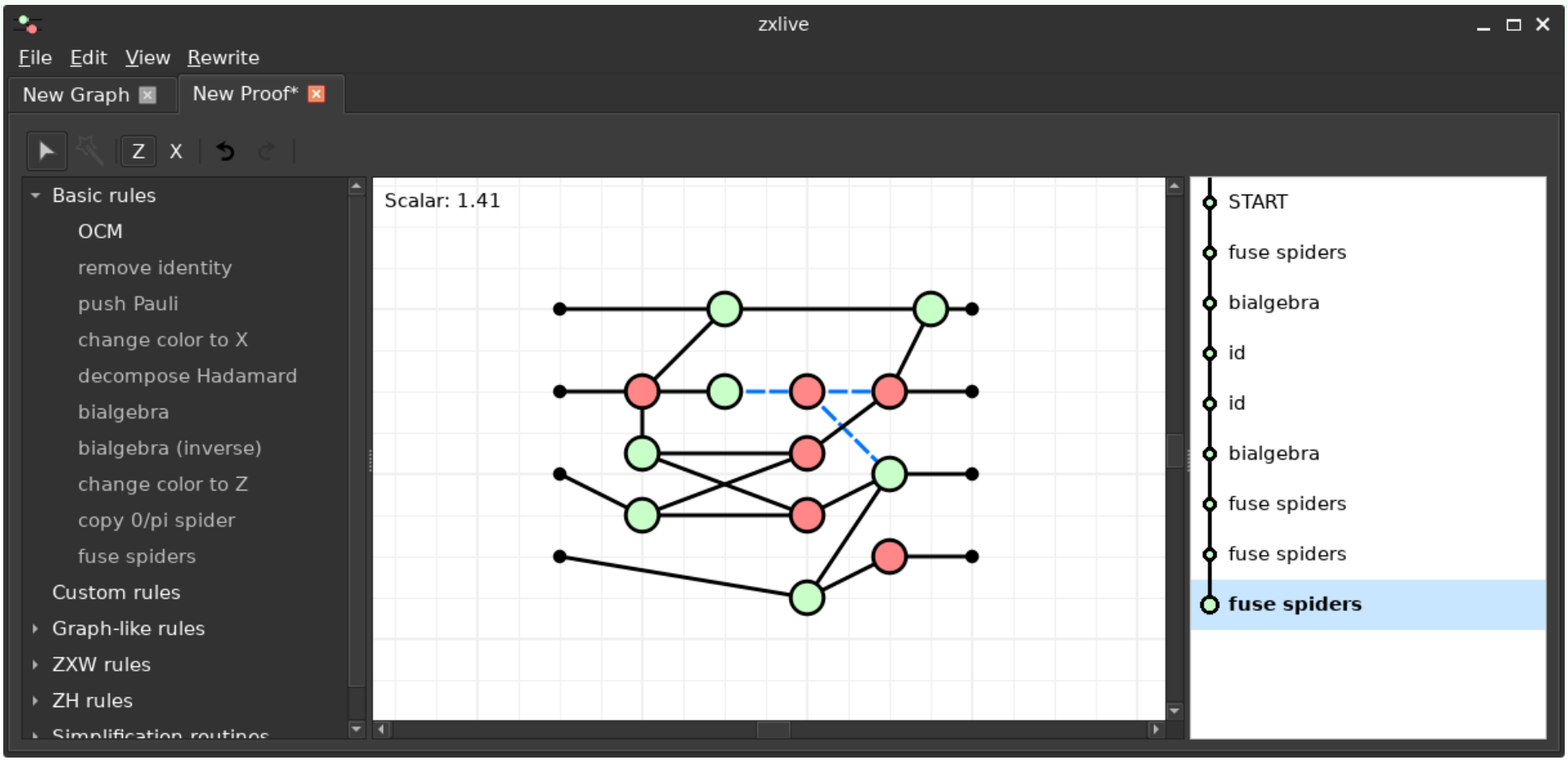
```rust
use quizx::basic_rules::*;

loop {
    match g.find_edge(|v0,v1,_| check_spider_fusion(&g, v0, v1)) {
        Some((v0,v1,_)) => spider_fusion_unsafe(&mut g, v0, v1),
        None => break,
    };
}
```

- **Large scale** circuit optimisation and classical simulation library for ZX-calculus

https://github.com/zxcalc/quizx

ZXLive



- GUI tool based on PyZX

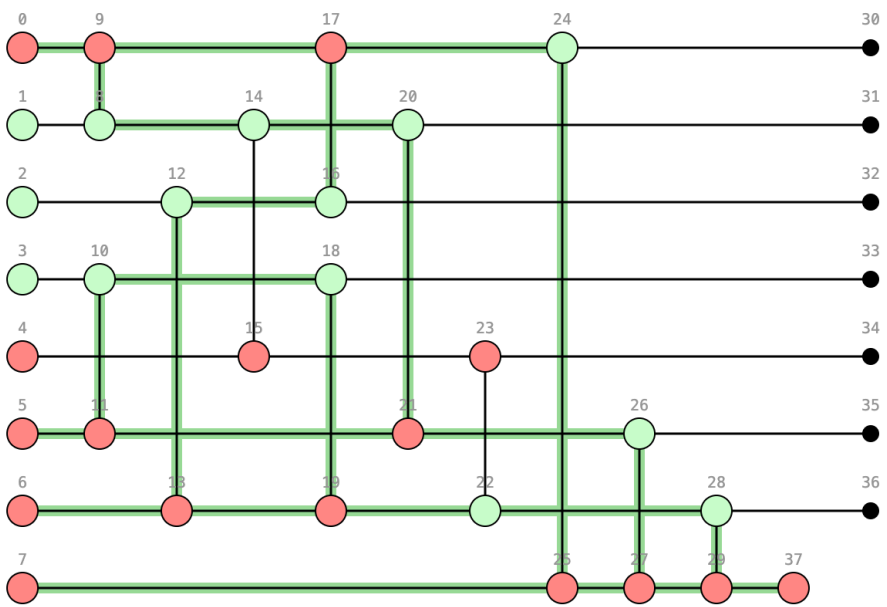
https://github.com/zxcalc/zxlive

Paritea

```
from paritea.equivalence import is_fault_equivalence

stabilisers, regions = compute_pauli_webs(d_circuit)
draw(d_circuit, web=regions[0])
is_fault_equivalence(nm_state, nm_circuit, quiet=True)
```

[17] ✓ 0.0s



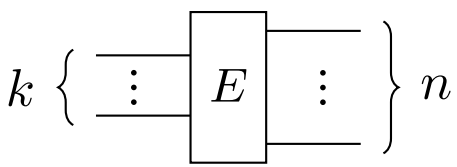
True

FT Simulation/Rewriting Utilities: github.com/paritea/paritea

Quantum error correction

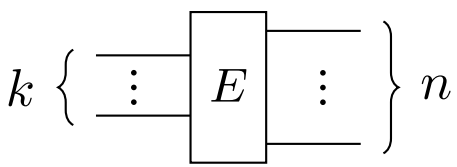
Quantum error correction

...is done by encoding some space of **logical** qubits
into a bigger space of **physical** qubits:



Quantum error correction

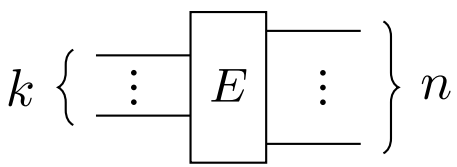
...is done by encoding some space of **logical** qubits
into a bigger space of **physical** qubits:



- E (or just $\mathrm{Im}(E)$) defines a **quantum error correcting code**

Quantum error correction

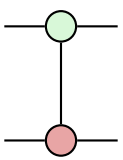
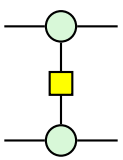
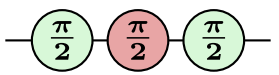
...is done by encoding some space of **logical** qubits
into a bigger space of **physical** qubits:

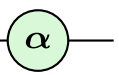
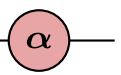


- E (or just $\mathrm{Im}(E)$) defines a **quantum error correcting code**
- Fault-tolerant quantum computing (FTQC) requires methods for:
  - encoding/decoding logical states and measurements
  - measuring physical qubits to detect/correct errors
  - doing fault-tolerant computations on encoded qubits

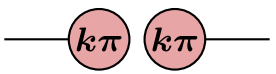
QEC in ZX 101...

Quantum Gates

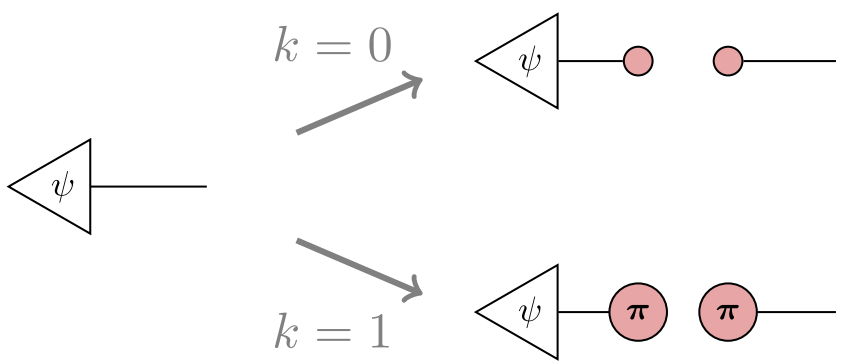
$CNOT =$ 
$CZ =$ 
$H =$ $:=$ 

$R_Z[\alpha] =$ 
$R_X[\alpha] =$ 

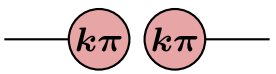
Quantum Measurements

$|k\rangle\langle k| \propto$ 

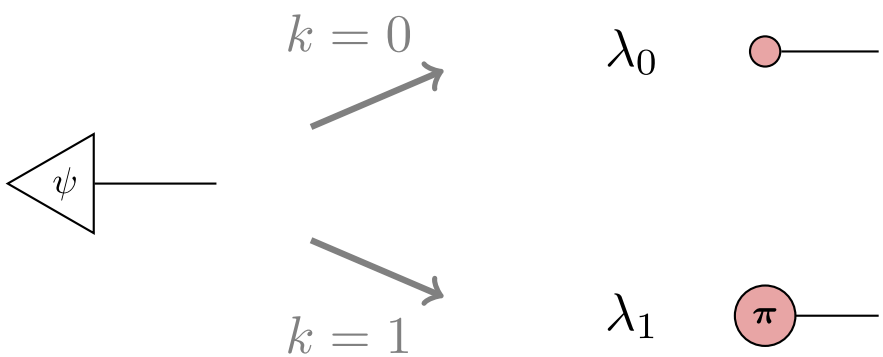
...collapse the quantum state to a fixed one,
depending on the **outcome** $k \in \{0, 1\}$:



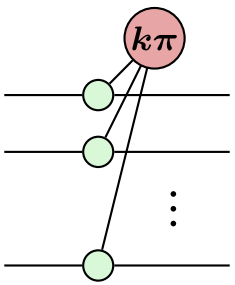
Quantum Measurements

$$|k\rangle\langle k| \propto$$ 

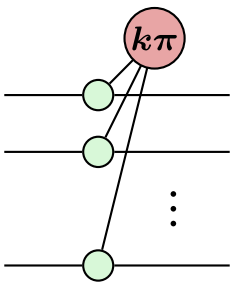
...collapse the quantum state to a fixed one, depending on the **outcome** $k \in \{0, 1\}$:



Multi-qubit measurements

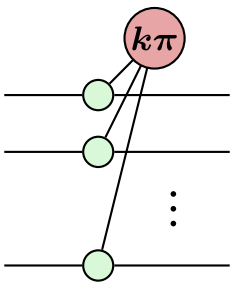


Multi-qubit measurements



$$k = 0 \quad \Rightarrow \quad +1 \text{ eigenspace of } Z \otimes \ldots \otimes Z$$
$$k = 1 \quad \Rightarrow \quad -1 \text{ eigenspace of } Z \otimes \ldots \otimes Z$$

Multi-qubit measurements



$$k = 0 \quad \Rightarrow \quad +1 \text{ eigenspace of } Z \otimes \ldots \otimes Z$$
$$k = 1 \quad \Rightarrow \quad -1 \text{ eigenspace of } Z \otimes \ldots \otimes Z$$

Generalises the single-qubit case, thanks to the "copy" rule:

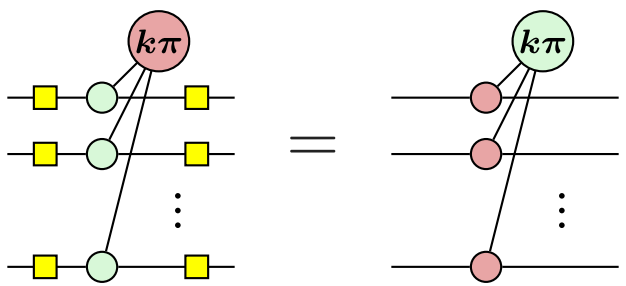
Multi-qubit measurements

Any other multi-qubit measurement can be obtained by conjugating with local Cliffords, e.g.

Multi-qubit measurements

Any other multi-qubit measurement can be obtained by conjugating with local Cliffords, e.g.

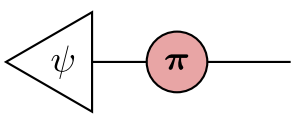
$$X \otimes \ldots \otimes X \quad \rightsquigarrow$$

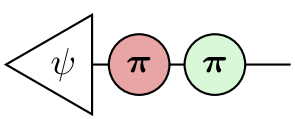


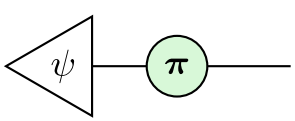
Modelling errors

Modelling errors

Pauli errors are modelled by introducing X, Y, or Z flips on edges in a ZX-diagram:

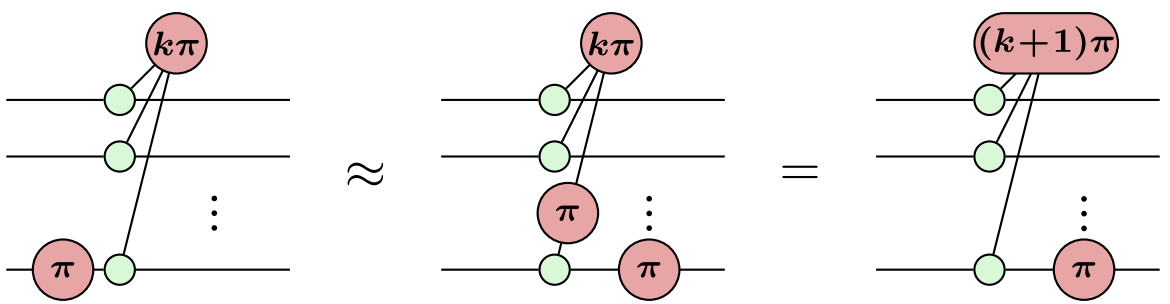
X-flip: 

Y-flip: 

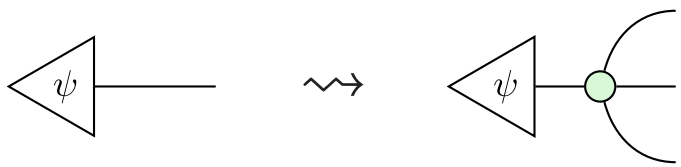
Z-flip: 

Modelling errors

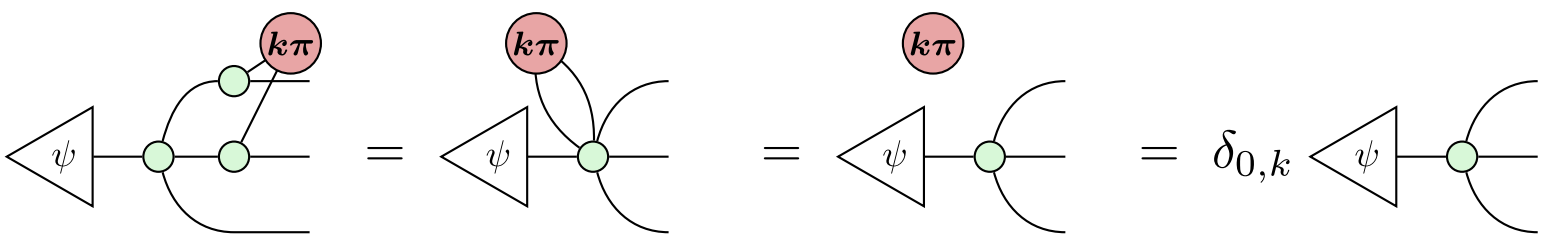
Anti-commuting errors flip measurement outcomes:



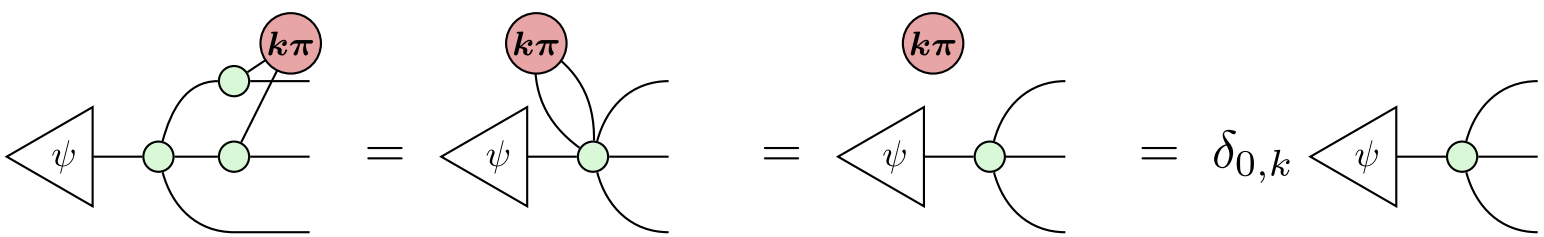
Example: GHZ code



Example: GHZ code

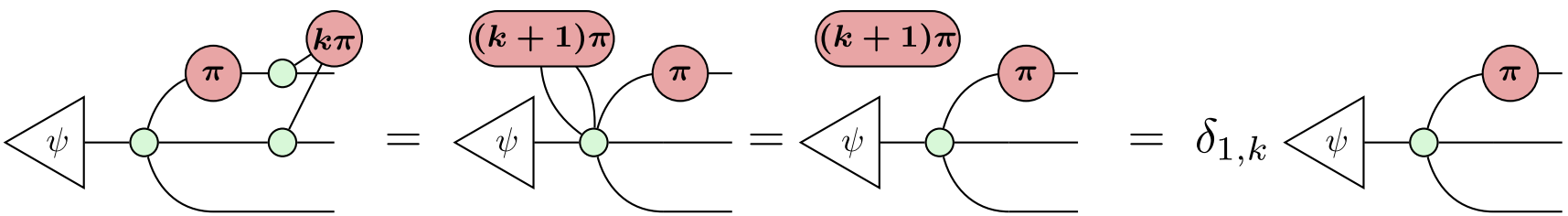


Example: GHZ code



This is a **stabiliser** measurement. In the absence of errors, it doesn't do anything!

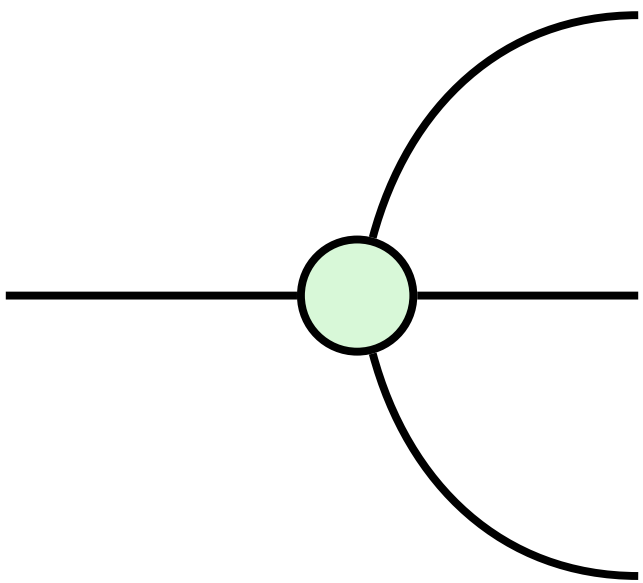
Example: GHZ code



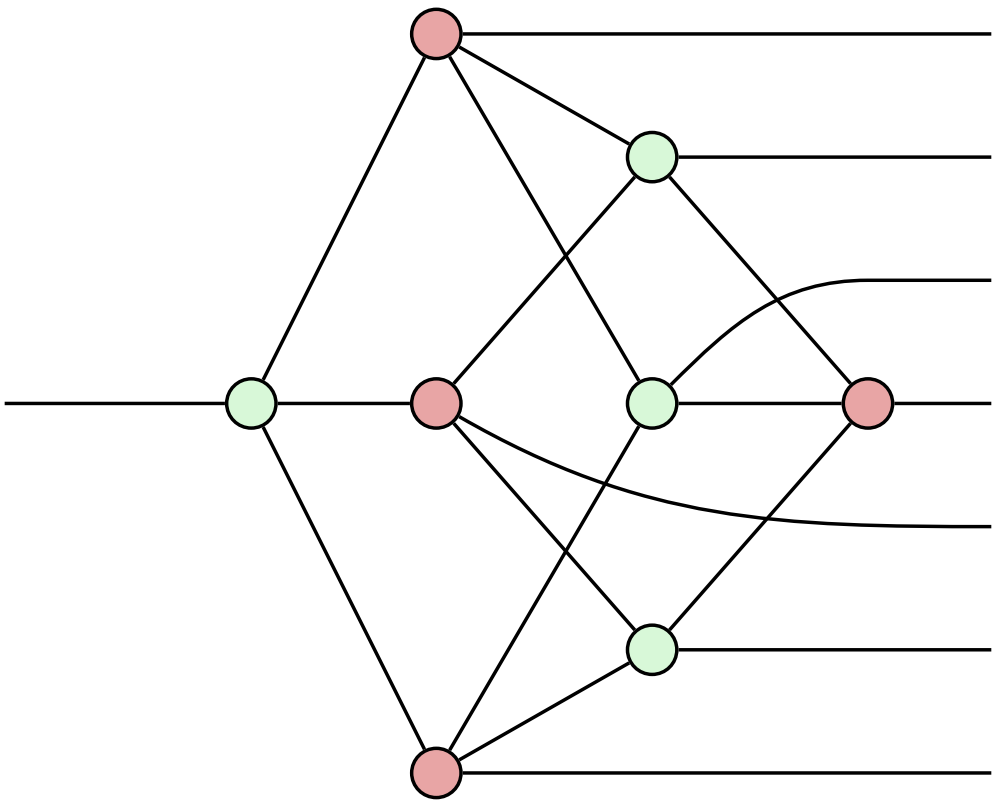
...but some errors will flip the measurement outcome,
giving an **error syndrome**.

Graphical encoders

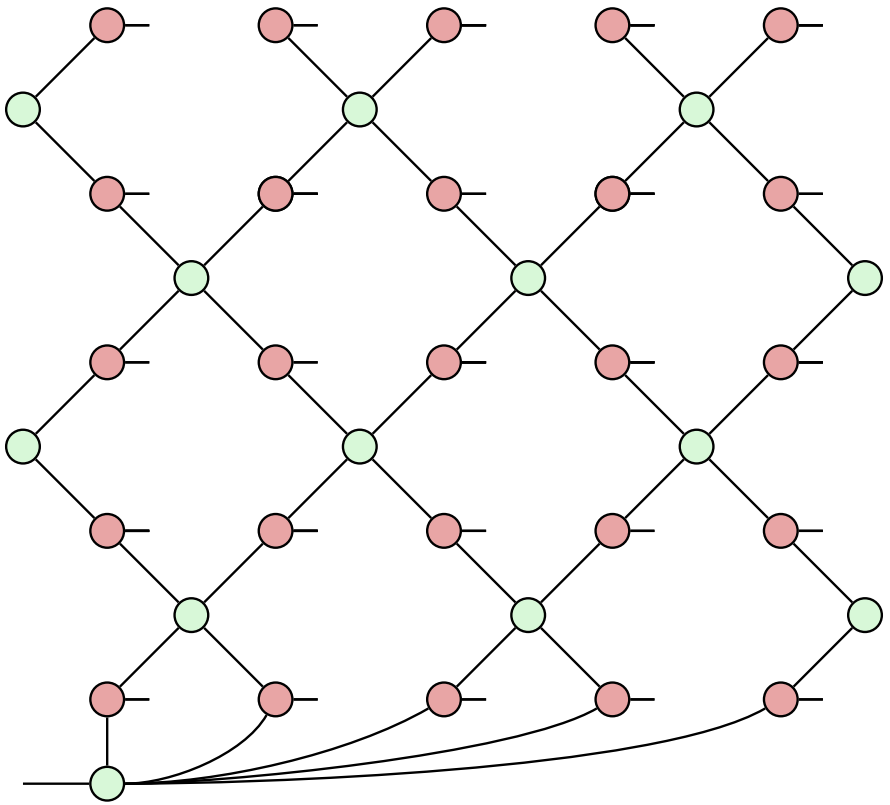
...give us a simple dictionary between QEC codes and ZX-diagrams, e.g. the GHZ code can be fully represented as:



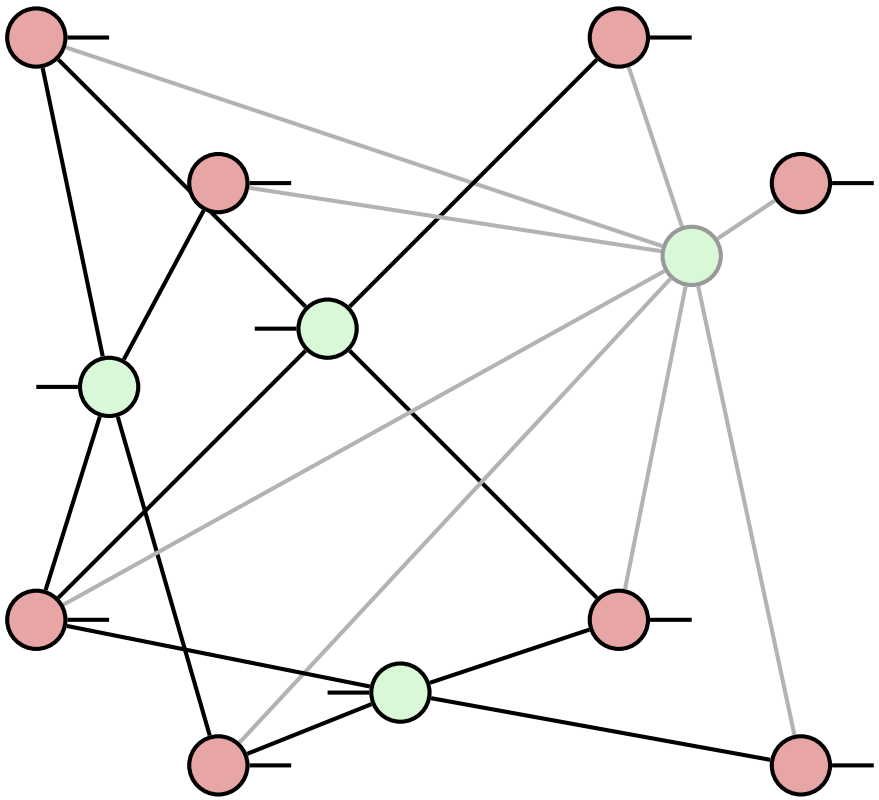
Example: Steane code



Example: Surface code



Example: [[8, 3, 2]] Colour code



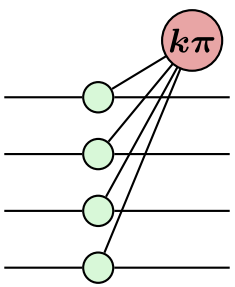
Fault-tolerant computation

But codes are only half the story in FTQC. We also need to know how to implement operations fault-tolerantly.

Fault-tolerant computation

But codes are only half the story in FTQC. We also need to know how to implement operations fault-tolerantly.

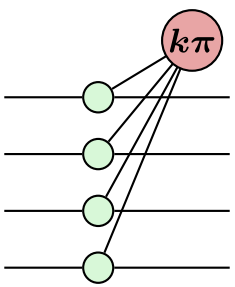
Consider a 4-qubit $Z \otimes Z \otimes Z \otimes Z$ measurement:



Fault-tolerant computation

But codes are only half the story in FTQC. We also need to know how to implement operations fault-tolerantly.

Consider a 4-qubit $Z \otimes Z \otimes Z \otimes Z$ measurement:

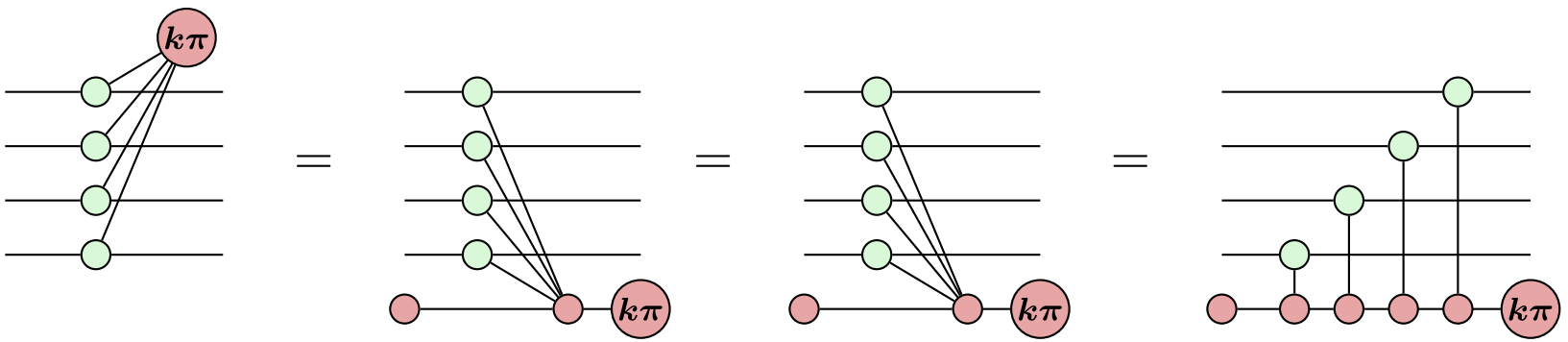


This isn't a basic operation (for most quantum computers).

How can we implement this?

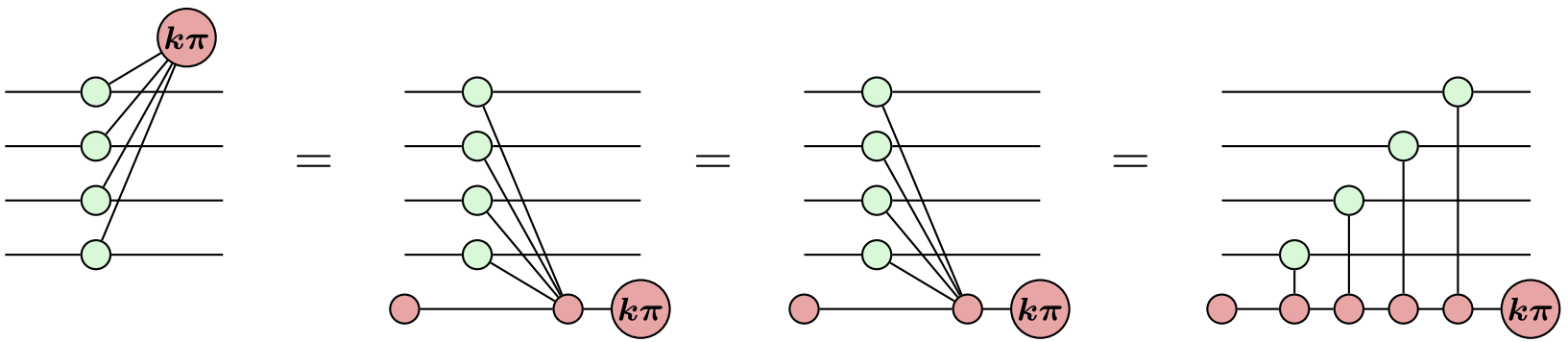
Example: measurement circuits

ZX can give us an answer:



Example: measurement circuits

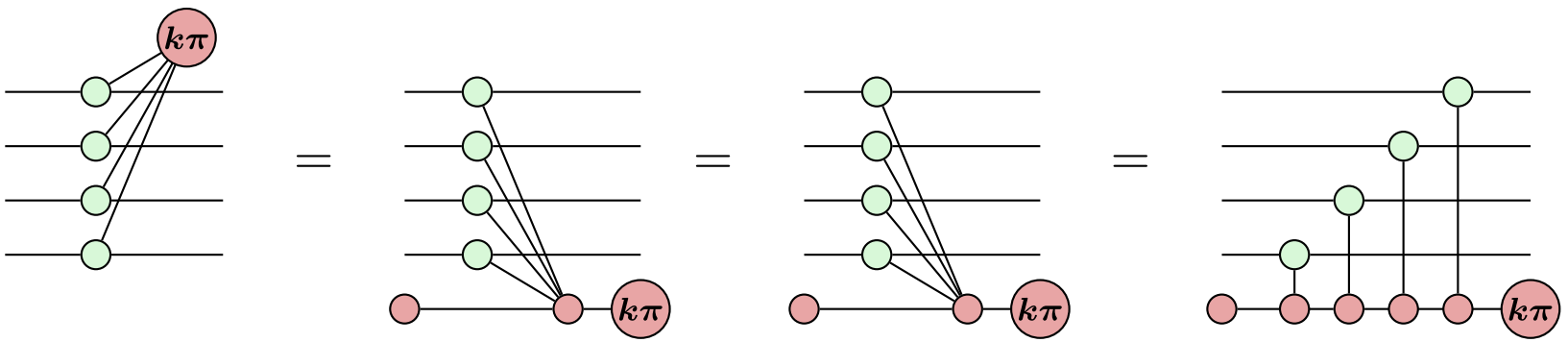
ZX can give us an answer:



Q: Is the LHS really equivalent to the RHS?

Example: measurement circuits

ZX can give us an answer:

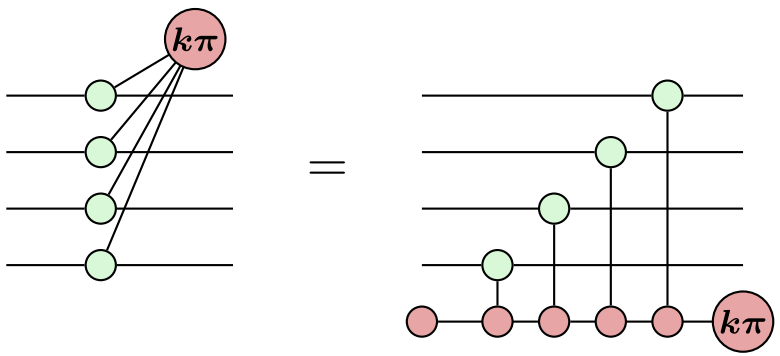


Q: Is the LHS really equivalent to the RHS?

A: It depends on what "equivalent" means.

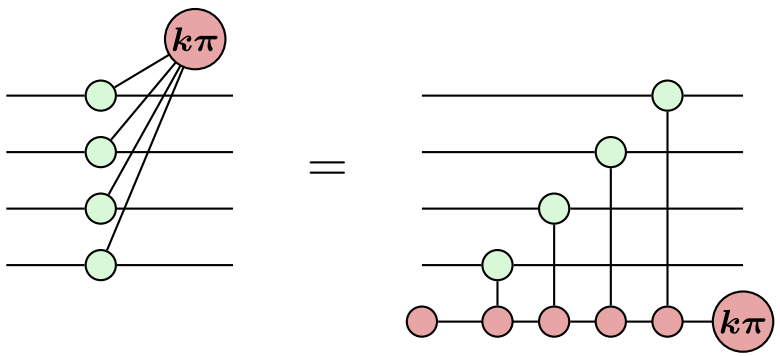
Example: measurement circuits

They both give the same linear map, i.e. the behave the same in the absence of errors.

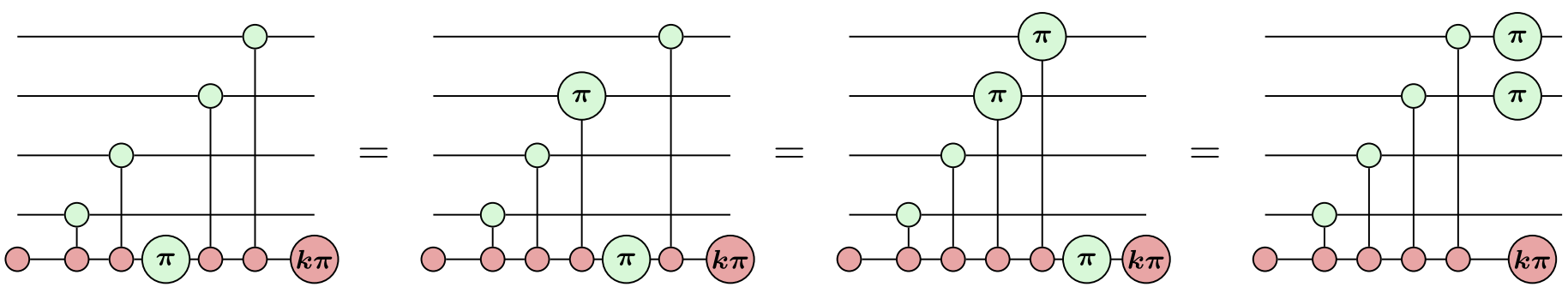


Example: measurement circuits

They both give the same linear map, i.e. the behave the same in the absence of errors.



But they behave differently in the **presence** of errors, e.g.



Solution: Fault equivalence

What we need is a notion of equivalence that captures the behaviour of circuits (or ZX-diagrams) in the presence of errors, **fault-equivalence**:

$$D \mathrel{\hat{=}} E$$

Solution: Fault equivalence

What we need is a notion of equivalence that captures the behaviour of circuits (or ZX-diagrams) in the presence of errors, **fault-equivalence**:

$$D \hat{=} E$$

This is a **finer-grained** notion of equivalence than the usual one:

$$D \hat{=} E \implies D = E$$

$$D = E \not\implies D \hat{=} E$$

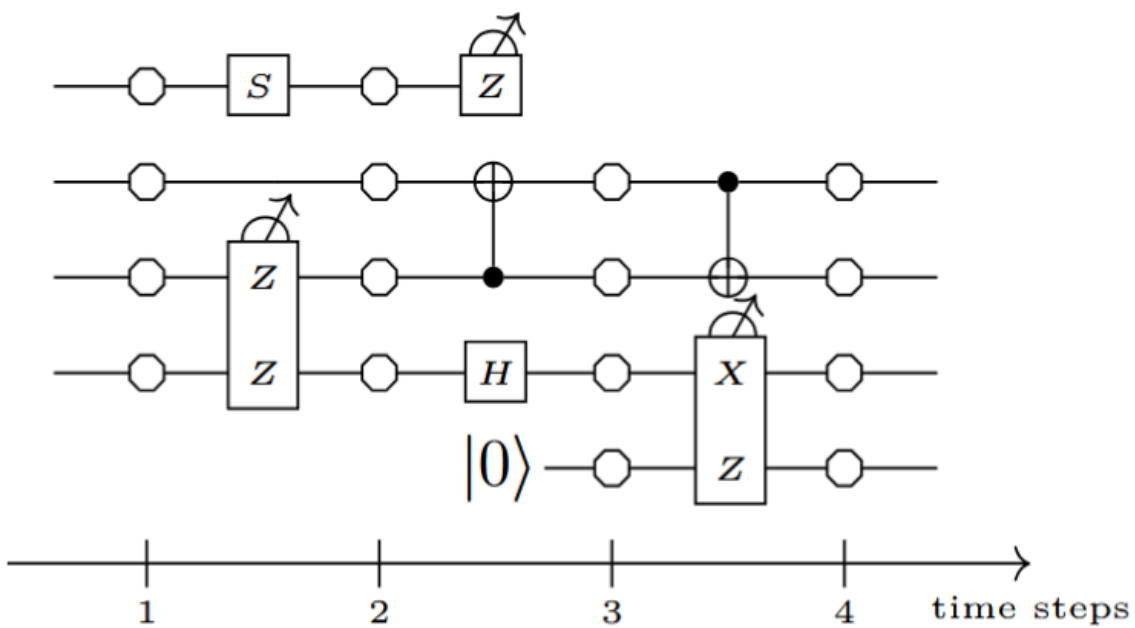
Error locations

Similar to space-time codes, we can model faults by tracking their locations in a circuit or ZX-diagram.

Error locations

Similar to space-time codes, we can model faults by tracking their locations in a circuit or ZX-diagram.

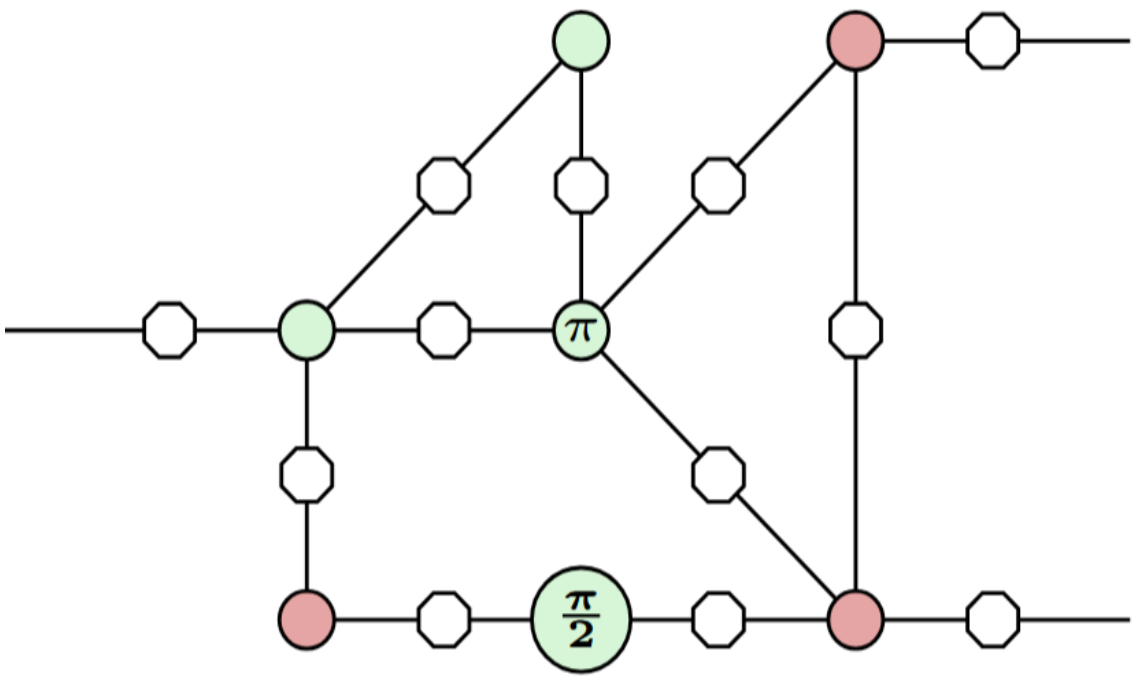
For circuits, fault locations are qubits $\times$ time steps:



Error locations

Similar to space-time codes, we can model faults by tracking their locations in a circuit or ZX-diagram.

For ZX-diagrams, locations are time-agnostic, we allow them at any edge in the diagram:



Applying a Pauli to a Circuit/Diagram

- For a circuit/diagram with a set $\mathcal{L}$ of fault locations, a fault is a Pauli

$$F \in \mathcal{P}^{|\mathcal{L}|}$$

Applying a Pauli to a Circuit/Diagram

- For a circuit/diagram with a set $\mathcal{L}$ of fault locations, a fault is a Pauli

$$F \in \mathcal{P}^{|\mathcal{L}|}$$

- We can **apply** any Pauli $F \in \mathcal{P}^{|\mathcal{L}|}$ to a diagram D by plugging it in, to make a linear map $D[F]$
  - *convention: interpret measurements as projection onto $+1$, and assume $D[I] \neq 0$*

Applying a Pauli to a Circuit/Diagram

- For a circuit/diagram with a set $\mathcal{L}$ of fault locations, a fault is a Pauli

$$F \in \mathcal{P}^{|\mathcal{L}|}$$

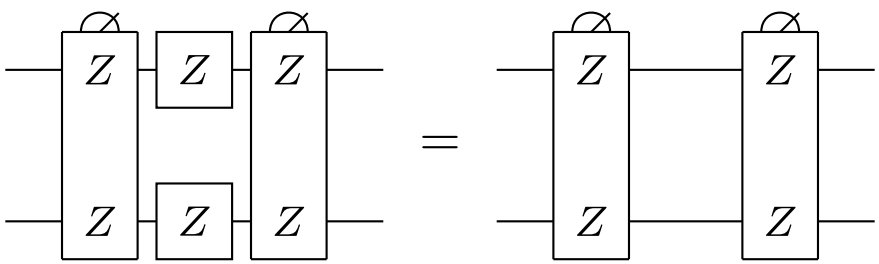
- We can **apply** any Pauli $F \in \mathcal{P}^{|\mathcal{L}|}$ to a diagram D by plugging it in, to make a linear map $D[F]$
  - *convention: interpret measurements as projection onto $+1$, and assume $D[I] \neq 0$*

- **There are 3 possibilities...**

F does nothing

$$D[F] = D[I]$$

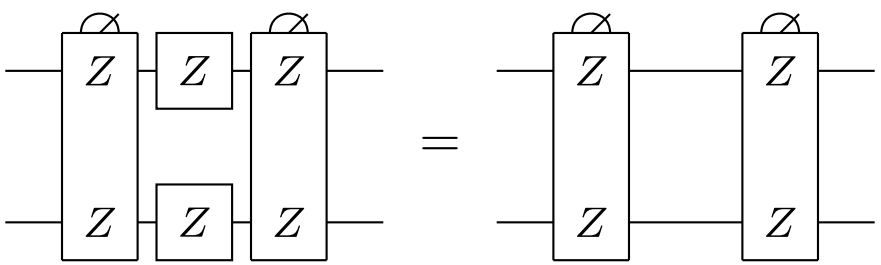
i.e. it is a **gauge** of the diagram

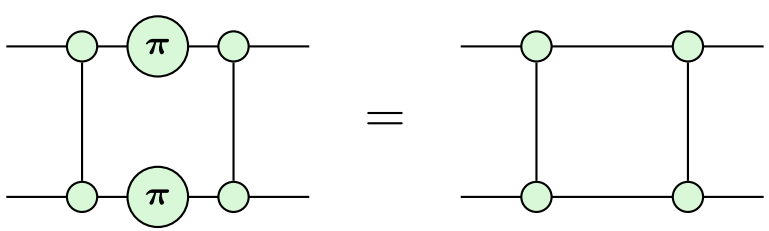


F does nothing

$$D[F] = D[I]$$

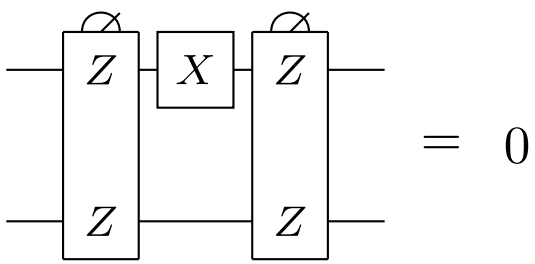
i.e. it is a **gauge** of the diagram



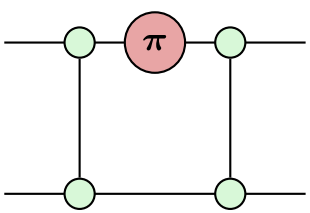


F is detectable

$$D[F] = 0$$



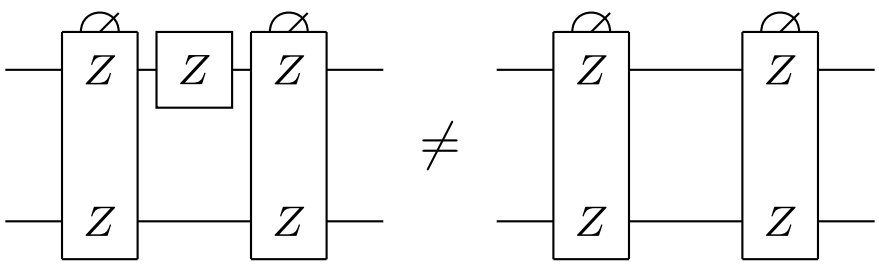
$= \quad 0$

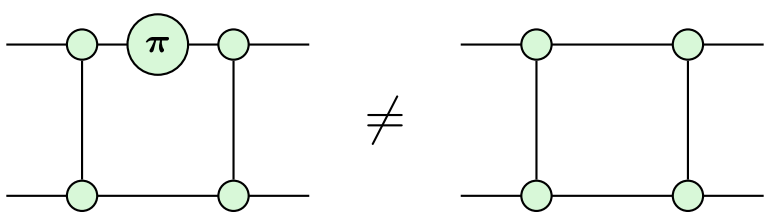


$= \quad 0$

F is undetectable

$D[F] \neq 0$ and $D[F] \neq D[I]$





Pauli error models

- A **Pauli error model** is a choice of basic faults $F_1, \ldots, F_b$. The **weight** of a fault is the number of basic faults in its product.

Pauli error models

- A **Pauli error model** is a choice of basic faults $F_1, \ldots, F_b$. The **weight** of a fault is the number of basic faults in its product.

- Ex: **Phenominological/naive model**:

$$\text{basic faults} := \text{all single-qubit Paulis}$$

Pauli error models

- A **Pauli error model** is a choice of basic faults $F_1, \ldots, F_b$. The **weight** of a fault is the number of basic faults in its product.

- Ex: **Phenominological/naive model**:

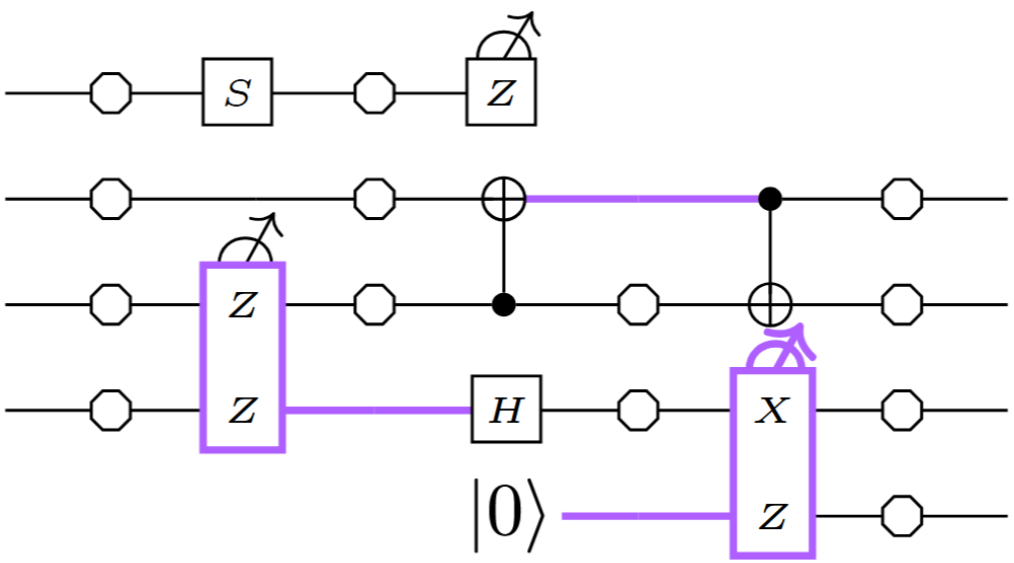
  basic faults $:=$ all single-qubit Paulis

- Ex: **Circuit-level noise model**:

  basic faults $:=$ all single-qubit Paulis (memory errors) +
  some multi-qubit Paulis (correlated gate/measurement errors)

Pauli error models

Ex: **sub-models**, where we remove some basic faults to represent components we assume are noise-free:



Fault-equivalence

Definition: Two circuits (or ZX-diagrams) C, D are called *fault-equivalent*:

$$C \mathrel{\hat{=}} D$$

if for any undetectable fault F of weight w on C, there exists an undetectable fault F' of weight $\leq w$ on D such that $C[F] = D[F']$ (and vice-versa).

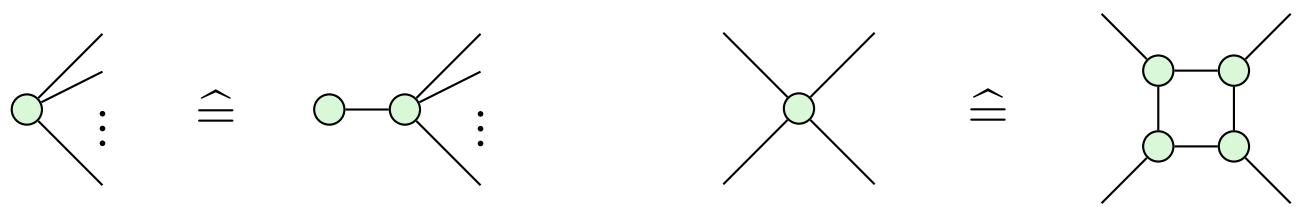
Fault-equivalence

Fault-equivalence

- While all the ZX rules preserve map-equivalence, only some rules preserve fault-equivalence.

Fault-equivalence

- While all the ZX rules preserve map-equivalence, only some rules preserve fault-equivalence.
- It turns out the ones that do, e.g.

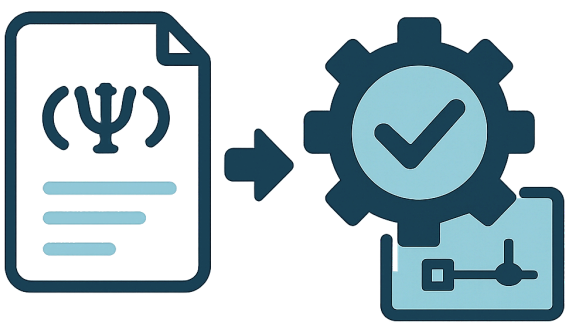


...are very useful for compiling fault-tolerant circuits!

Paradigm: Fault-tolerance by construction

arXiv:2506.17181

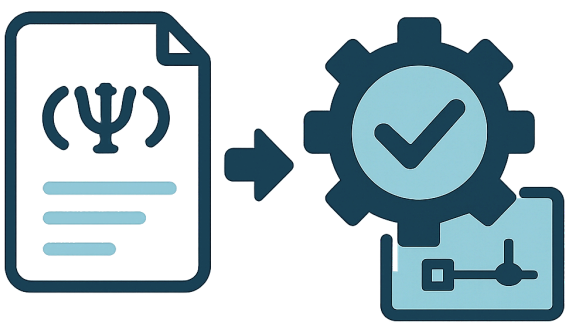
Idea: start with an idealised computation (i.e. specification) and refine it with fault-equivalent rewrites until it is implementable on hardware.



Paradigm: Fault-tolerance by construction

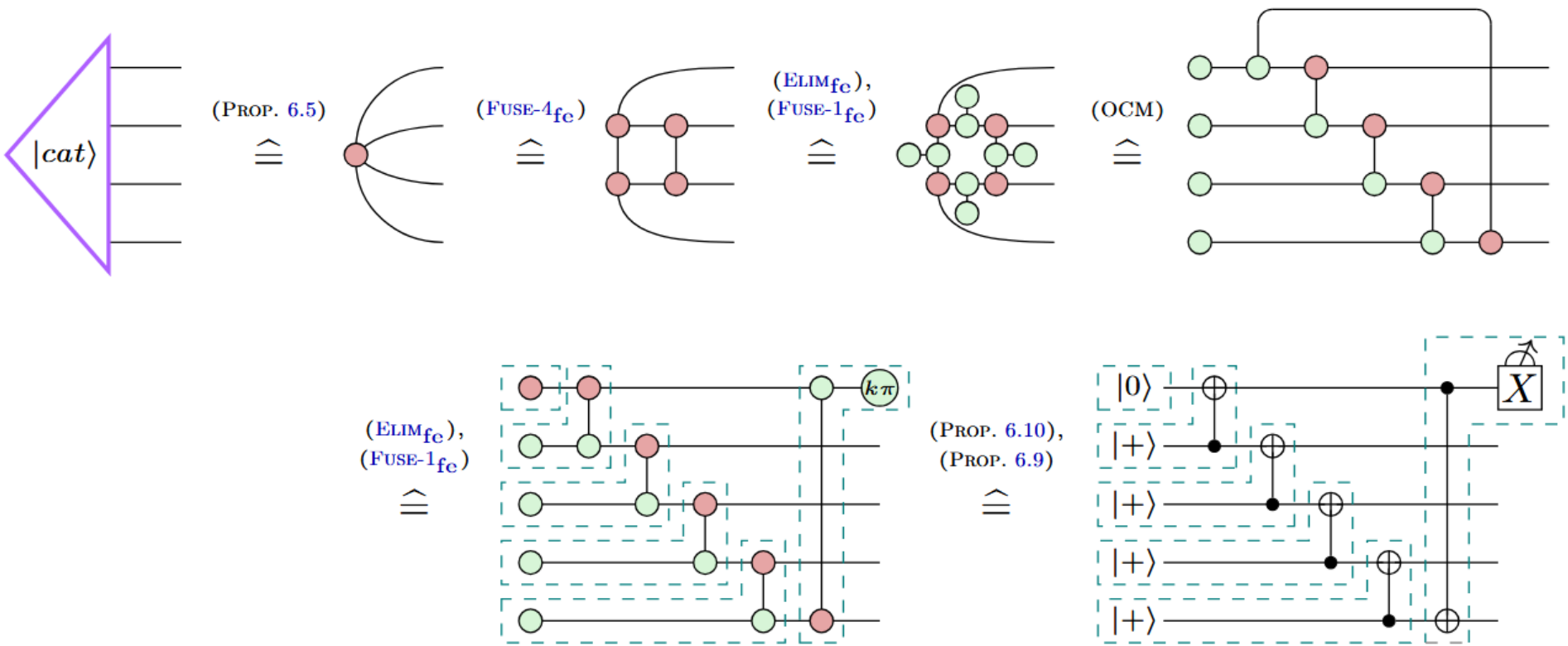
arXiv:2506.17181

Idea: start with an idealised computation (i.e. specification) and refine it with fault-equivalent rewrites until it is implementable on hardware.

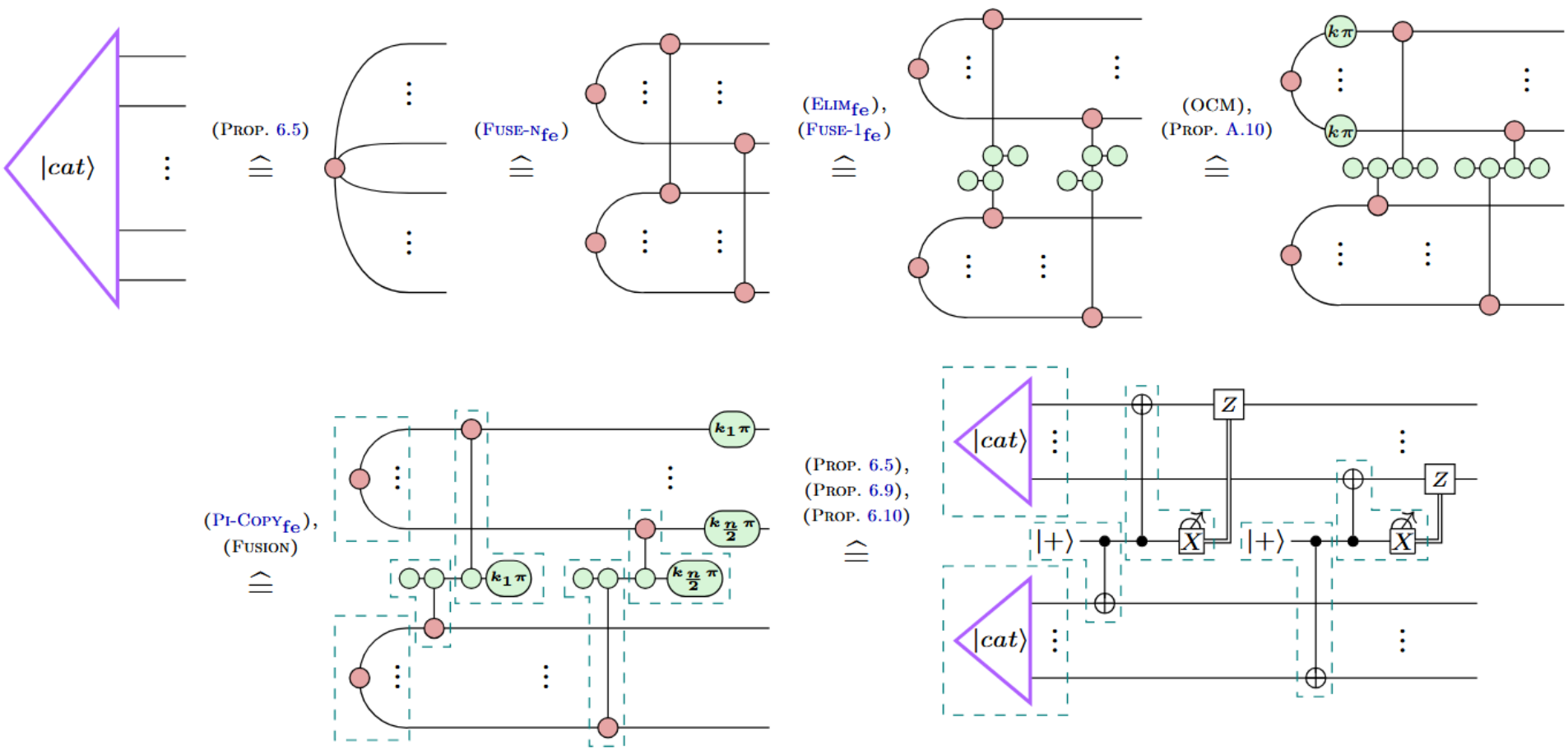


Specification/refinement has been used in formal methods for classical software dev since the 1970s. **Why not for FTQC?**

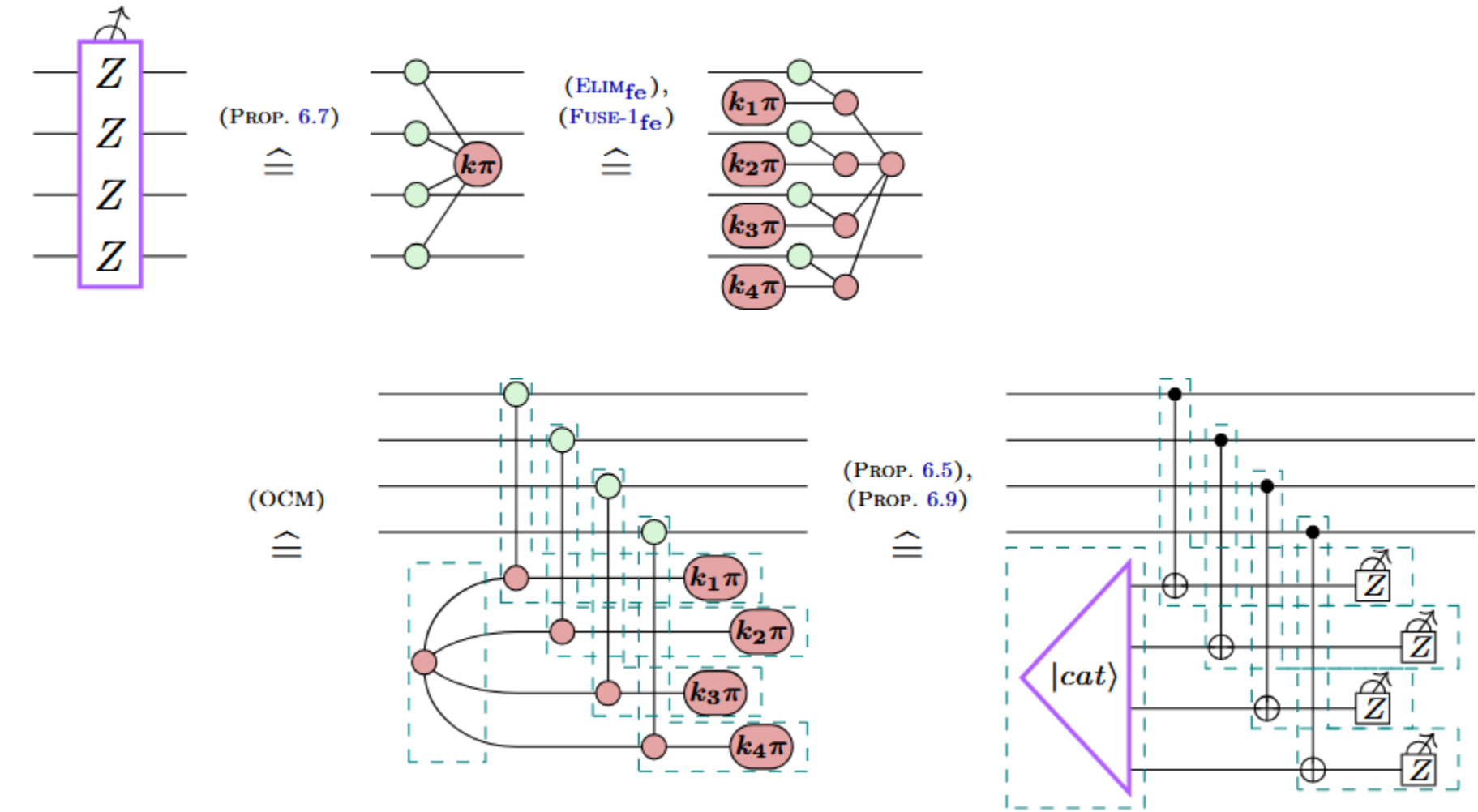
Example: Cat state preparation



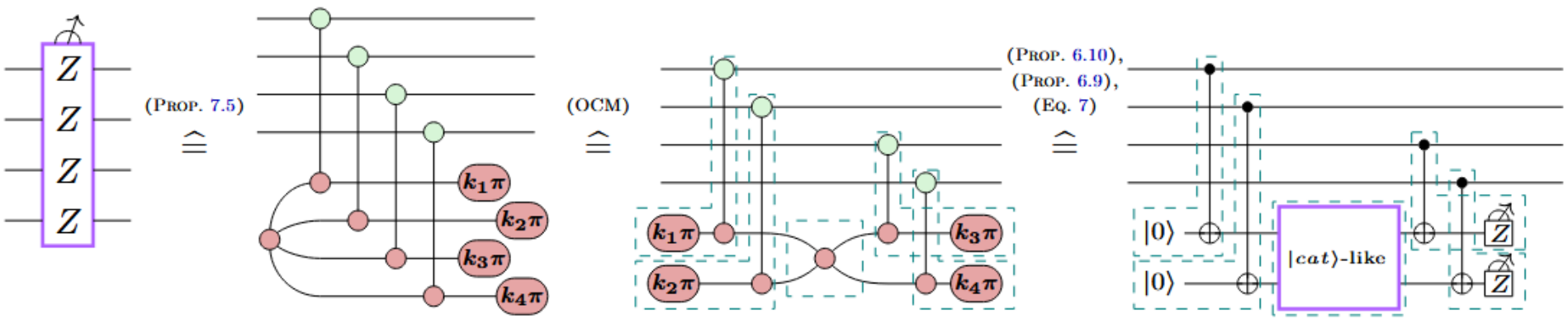
Example: Cat state preparation



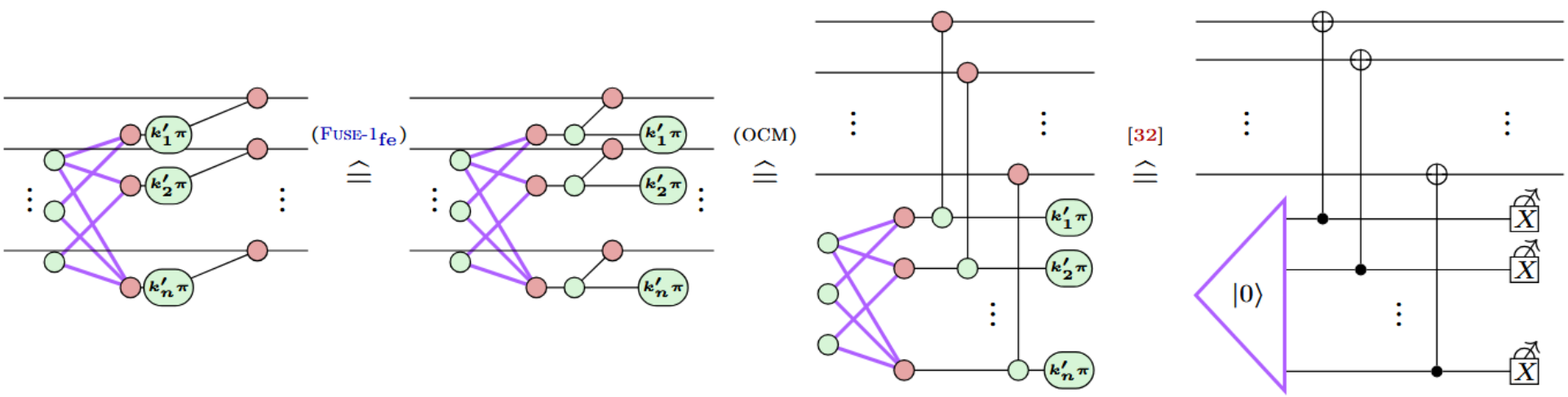
Example: Shor-style syndrome extraction



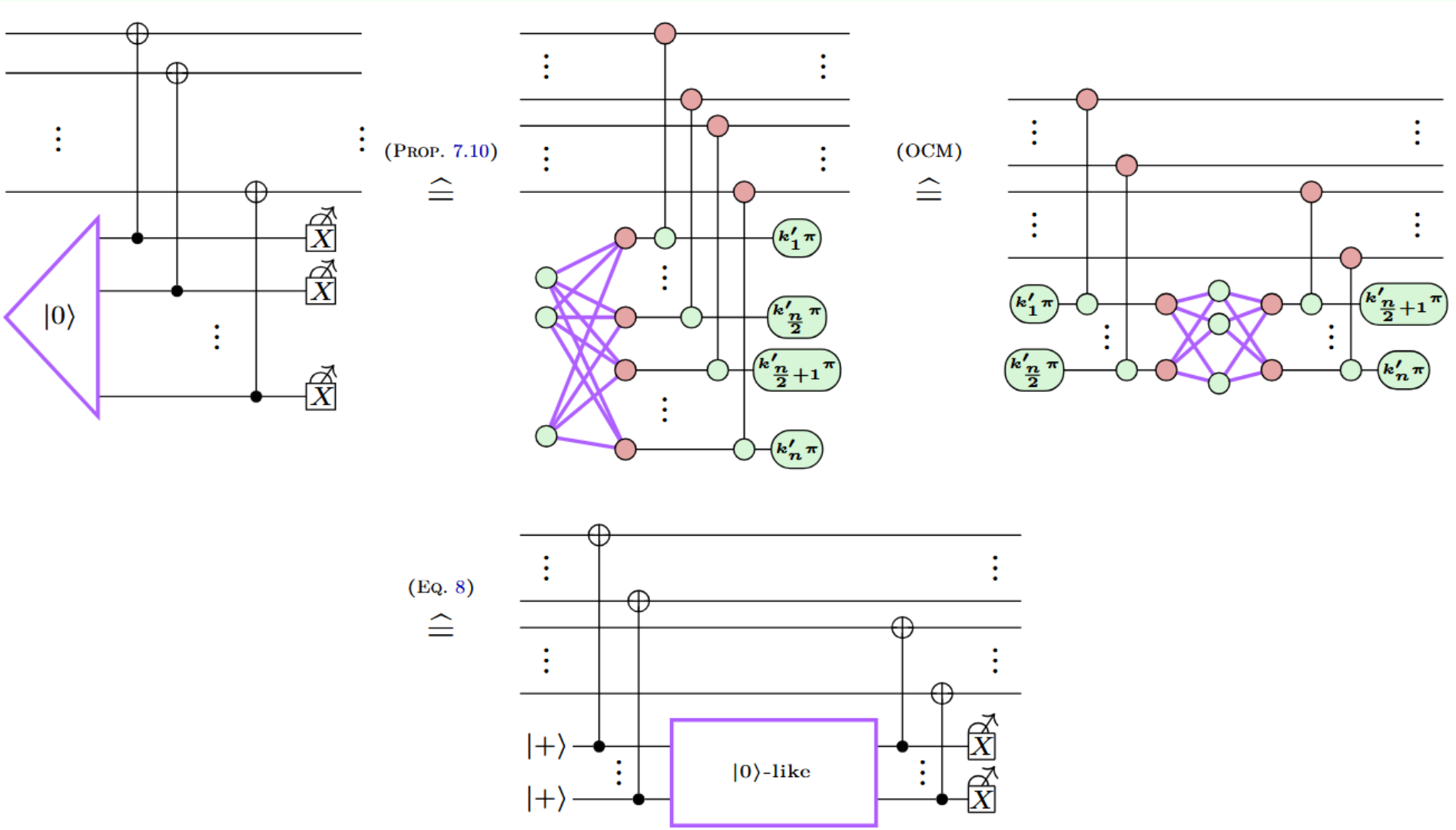
Example: A new variation on Shor



Example: Steane-style syndrome extraction



Example: A new variation on Steane (the same trick)

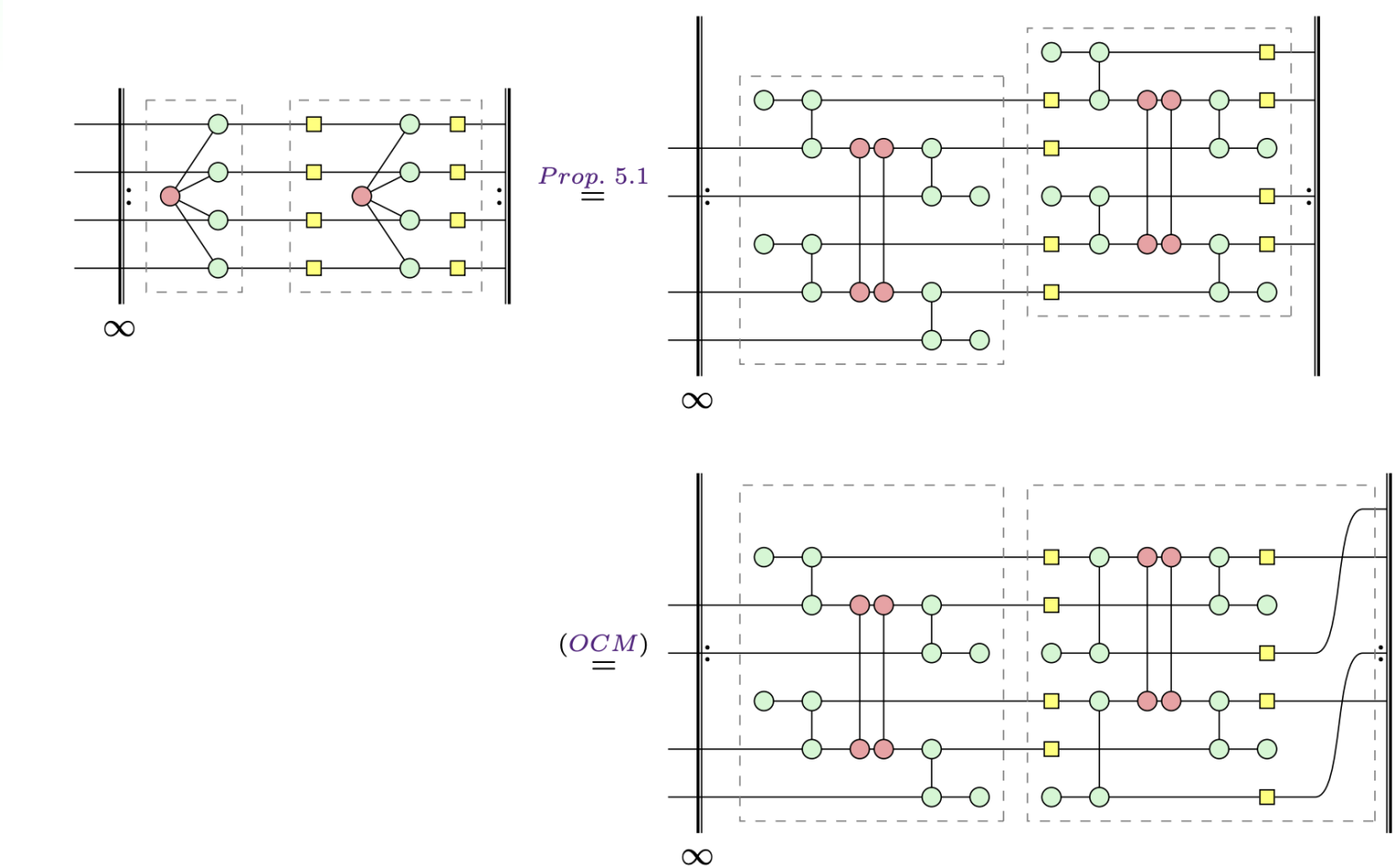


...with fewer qubits and a lower logical error rate!

FTbC Survey

Floquetification

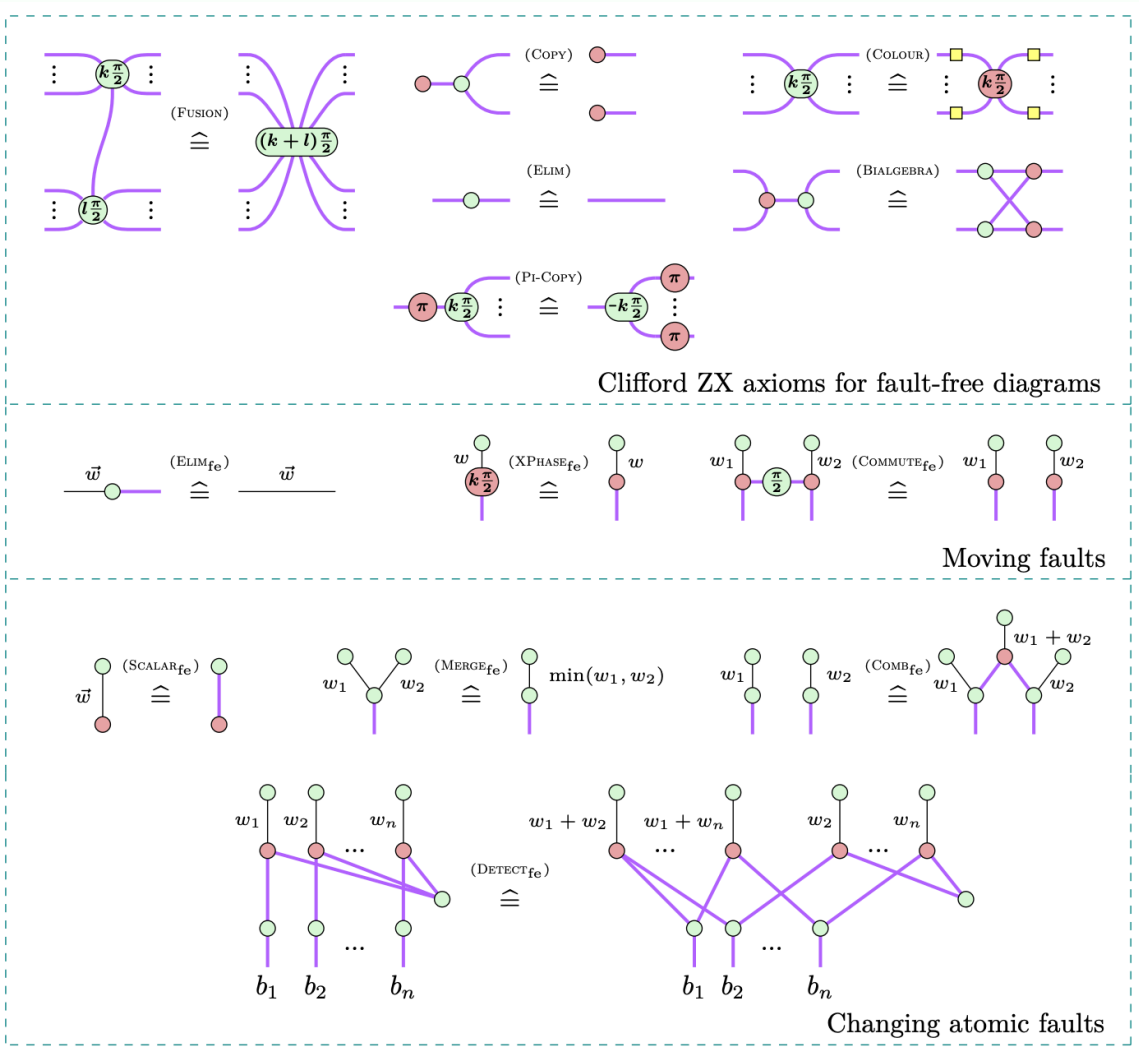
B Rodatz, B Poór, AK. arXiv:2410.17240



See also: A. Townsend-Teague, J. Magdalena de la Fuente, M. Kesselring. arXiv:2307.11136

Completeness for Fault-equivalent Rewriting

M Rüsch, AK, B Rodatz. arXiv:2510.08477

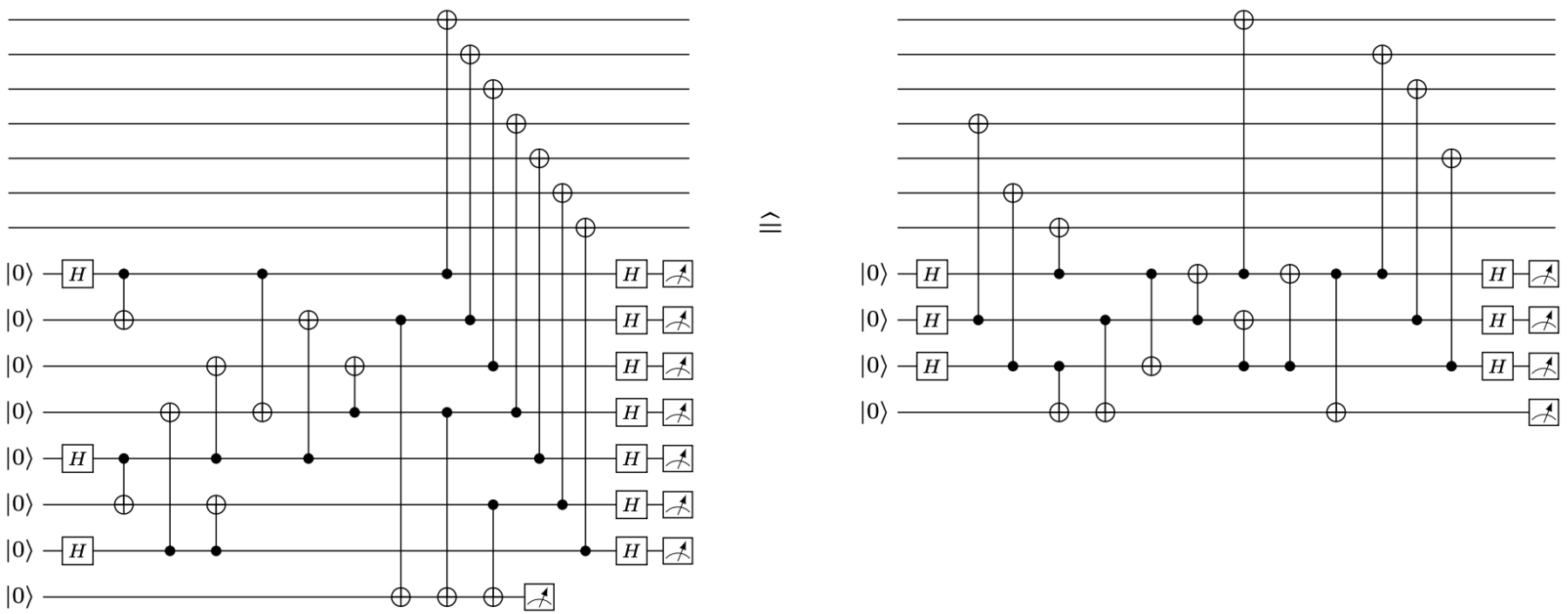


Ultra-low overhead syndrome extraction for the Steane Code

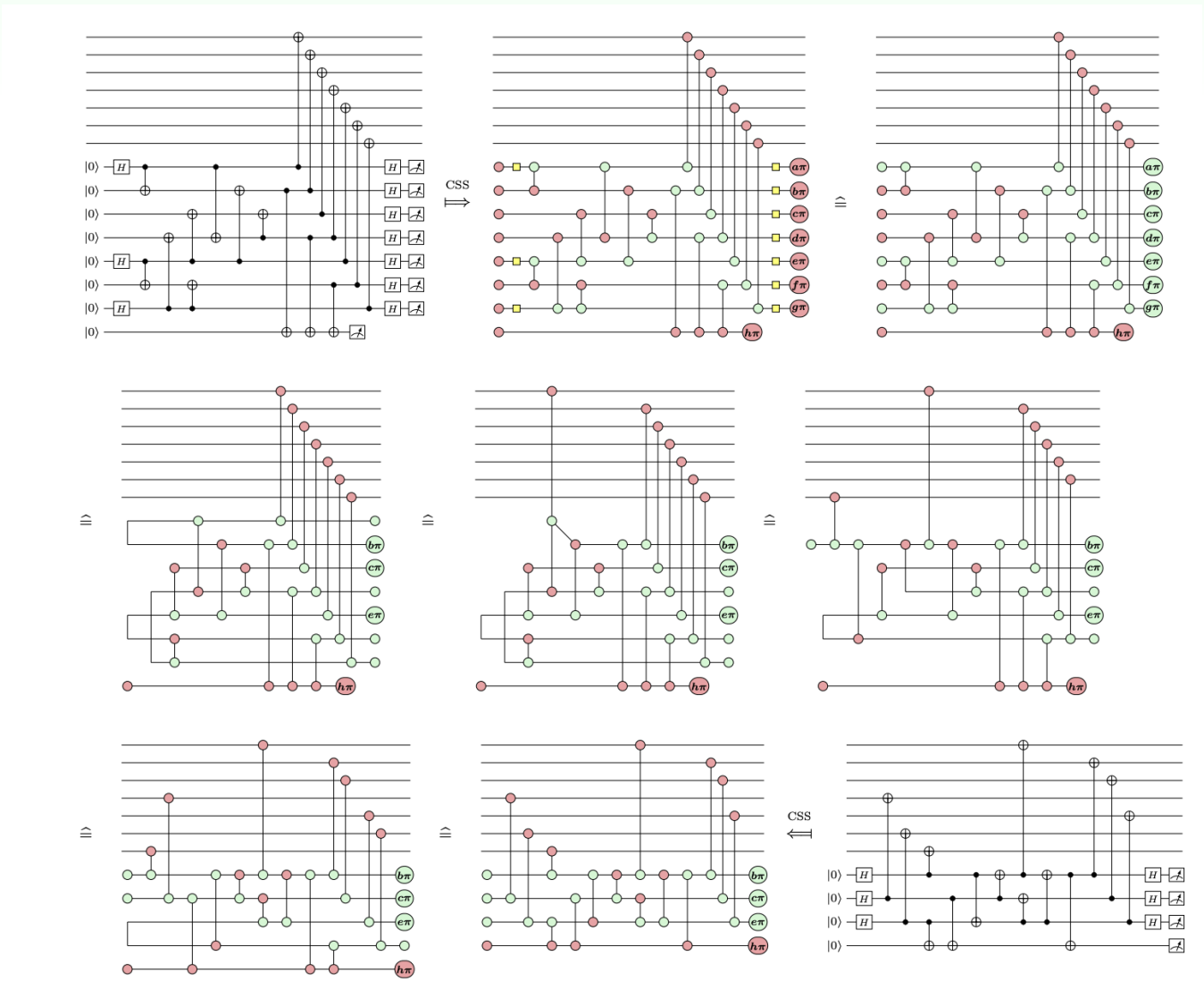
B Poór, B Rodatz, AK. arXiv:2511.13700

an 11-CNOT (optimal) syndrome extraction circuit for the Steane code*

Theorem 1. The following transformation is fault-equivalent:



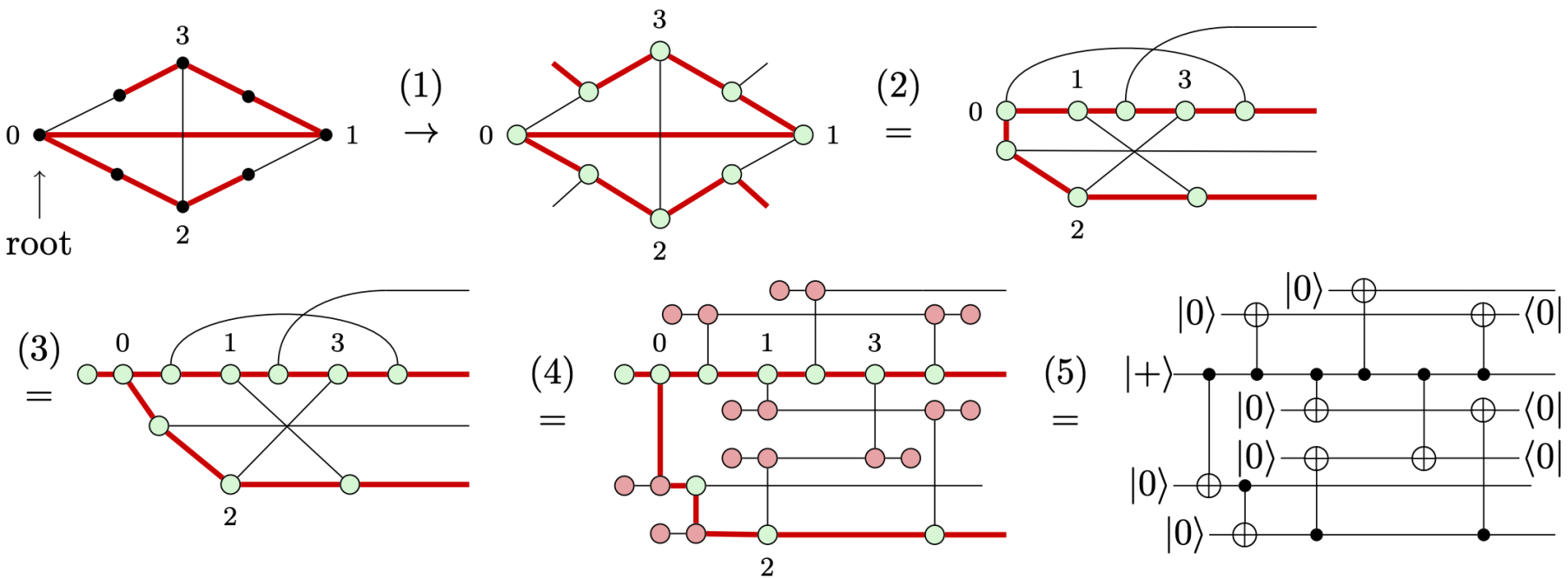
Ultra-low overhead syndrome extraction for the Steane Code



SpiderCat: Optimal FT Cat State Preparation

A Khesin, S Li, B Poór, B Rodatz, J van de Wetering, R Yeung.

arXiv:2603.05391



gives CNOT lower bounds for all sizes n and max FT error weight t

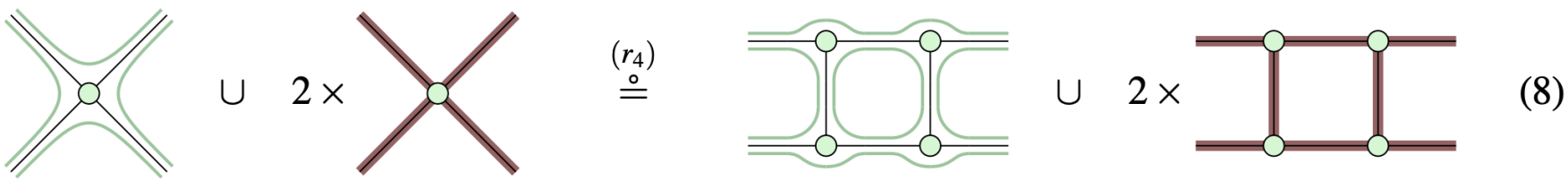
+ optimal explicit constructions for most pairs

$(n \leq 100, t \leq 5)$ and $(n \leq 50, t \leq 7)$

MWPM-Decodability Preserving Rewriting

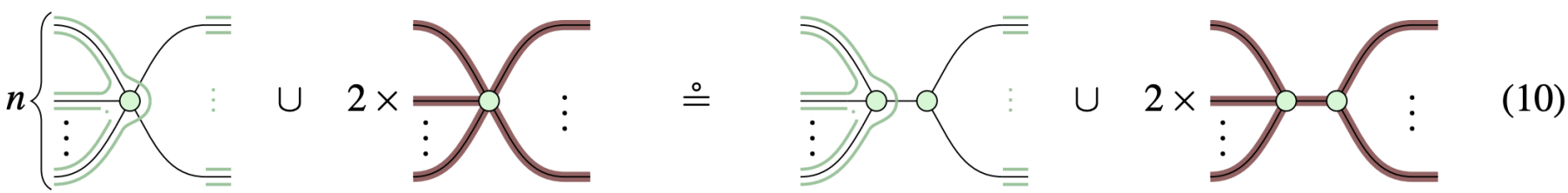
M Schweikart, L Grans-Samuelsson, AK, B Rodatz. arXiv:2603.19522

Proposition 3.13 (r_4). *The following rewrite is CSS matchability preserving:*



(8)

Proposition 4.3. *Let $C(D)$ have a circuit distance of more than n. Then, the following rewrite is matchability-preserving:*



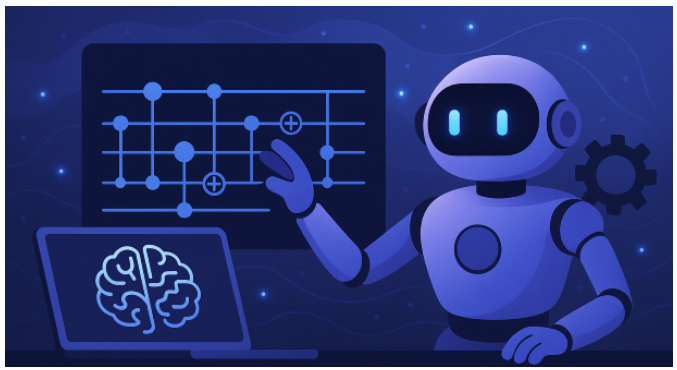
(10)

matchable QEC codes ⇒ FT syndrome extraction circuits with matchable DEMs

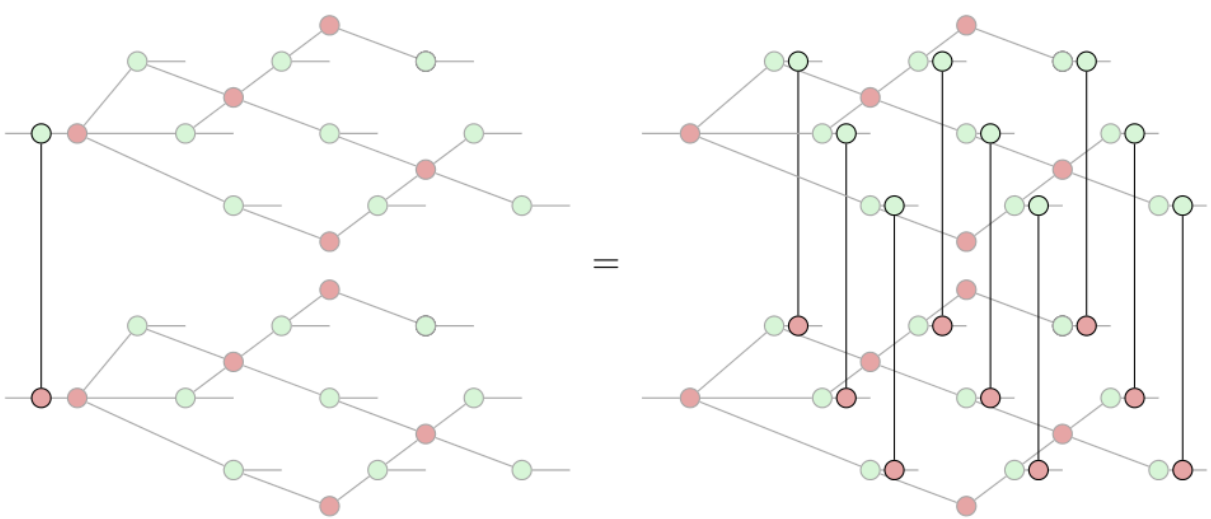
Lots to do!

Lots to do!

- Automatically building/optimising FT circuits
  (e.g. via heuristic search or AI)

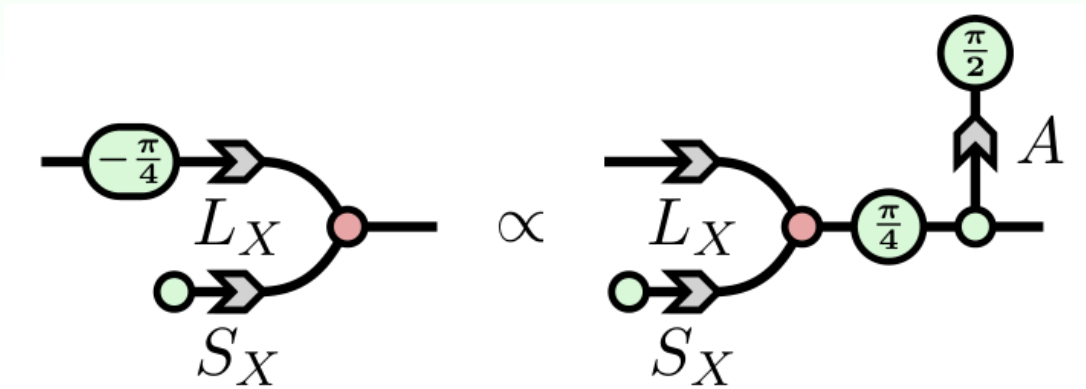


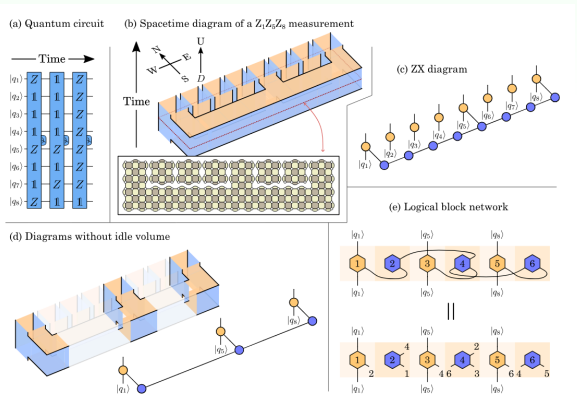
- Logical computation and measurement
  (esp. in non-traditional QEC paradigms, like dynamical codes)



Lots to do!

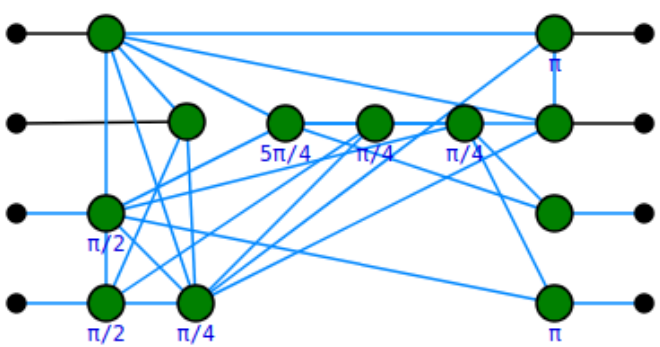
- Scalability, compositionality, and FT architectures

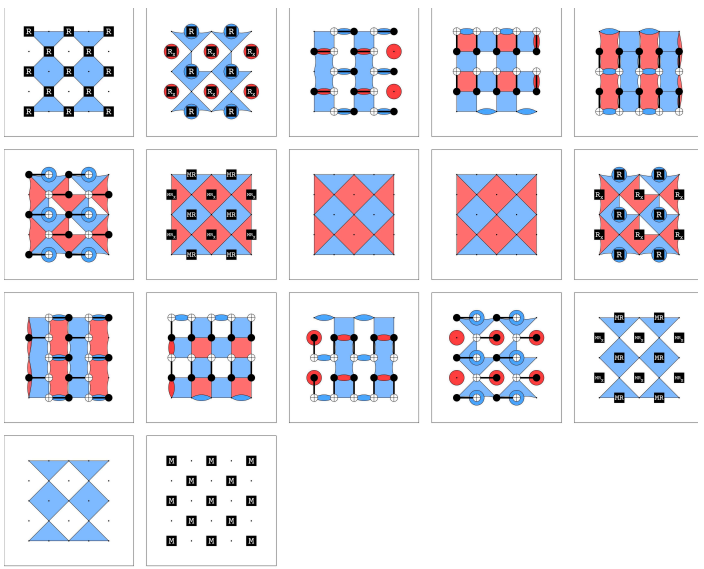




- Tooling, automation, and integration
  (PyZX/QuiZX/ZXLive/Paritea/Stim)

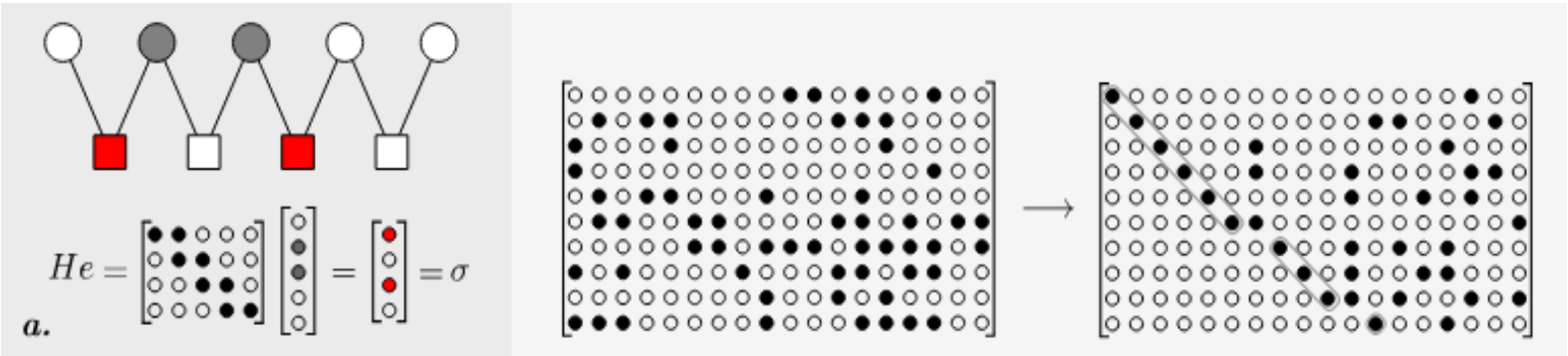
```
In [11]: zx.simplify.clifford_simp(g)
         g.normalise()
         zx.d3.draw(g)
```





Lots to do!

- Decoding errors and preserving efficient decodability
  (beyond matchable codes/circuits)



- Reasoning about stochastic noise
  (fault-equiv. = adversarial/worst-case noise behaviour)

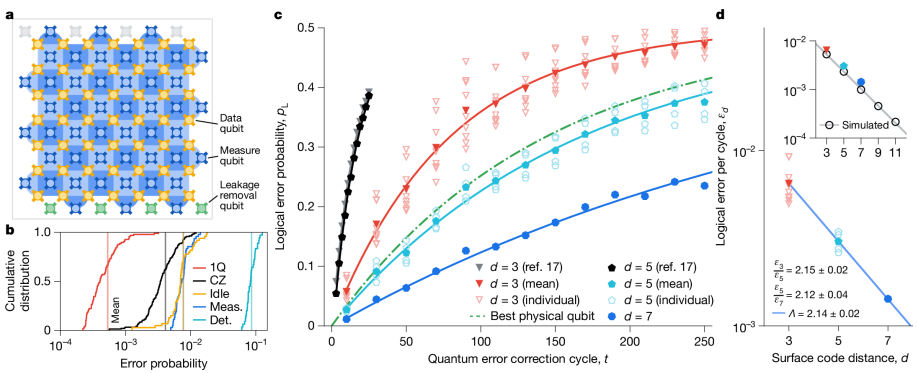
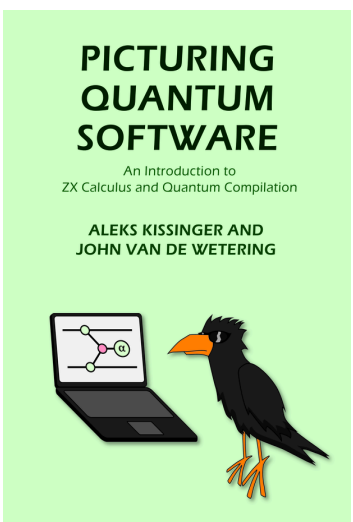


Image credit: Riverlane and Google Quantum AI

Thanks!

Fault Tolerance by Construction » arXiv:2506.17181

Collabs: Andrey Khesin, Sarah Meng Li, Boldiszár Poór, Benjamin Rodatz, John van de Wetering, Richie Yeung, Max Schweikart, Max Rüsch, Linnea Grans-Samuelsson



https://zxcalc.github.io/book
(free book! Ch 12 = ZX + QEC)

https://zxcalculus.com
(350+ ZX papers, ~10% tagged QEC, online seminars, Discord)