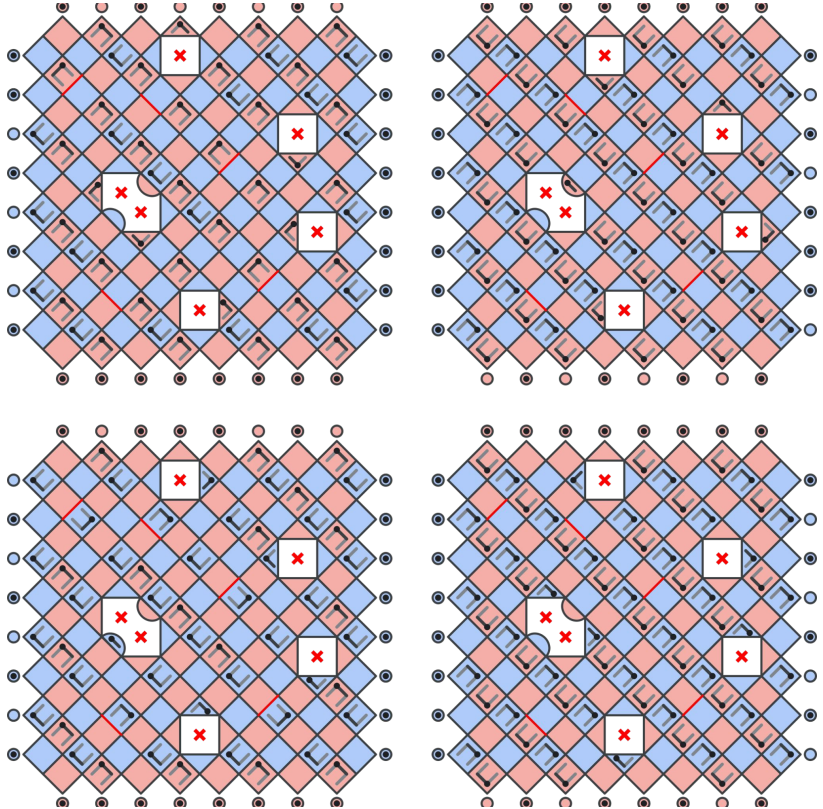
Quantum AI

Adapting QEC Circuits to Dropout Using LUCI

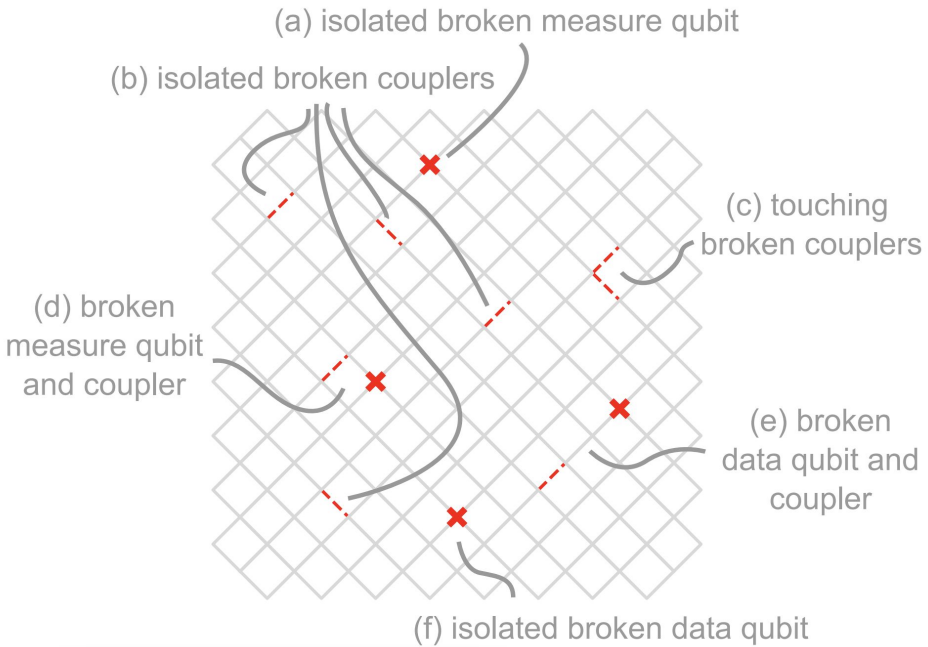
Dripto M. Debroy
dripto@google.com

IPAM Workshop, 2026
Los Angeles, CA
February 20th, 2026



Component dropouts from fabrication

Quantum AI



Our architecture has a square lattice of qubits, with couplers for entangling gates.

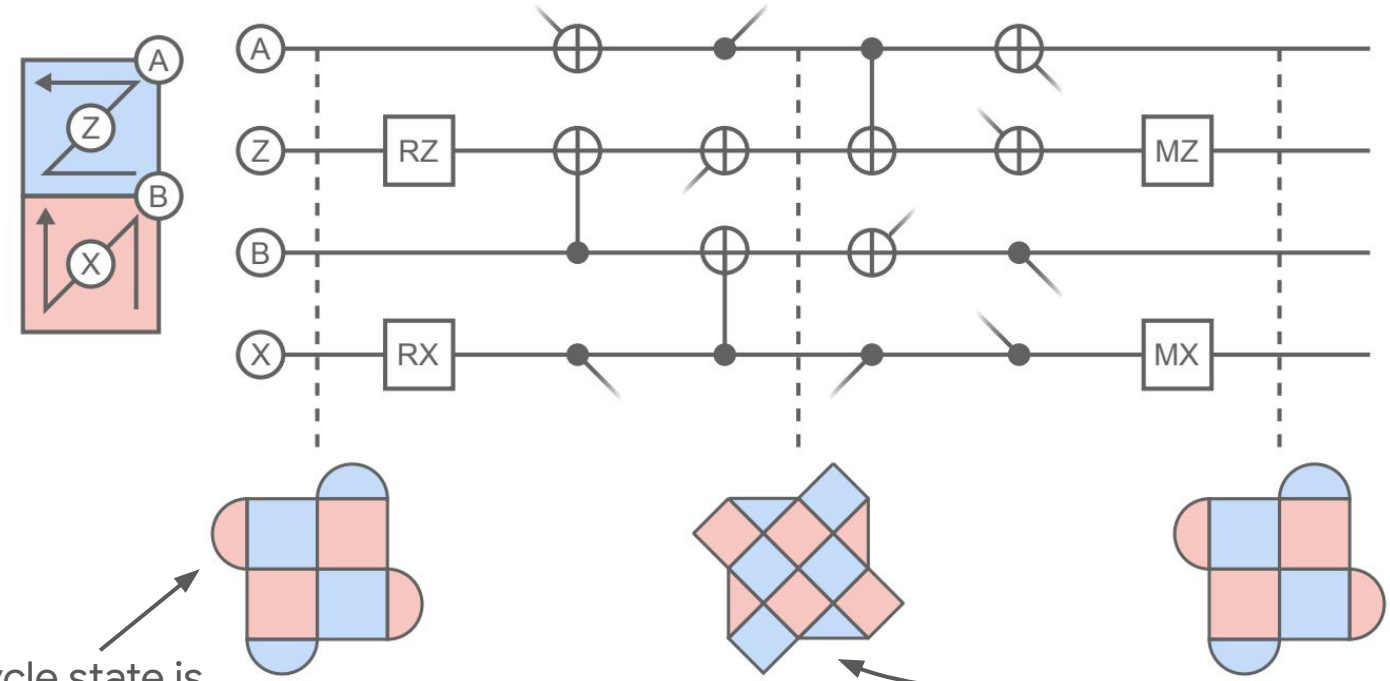
Adapting error correction protocols to grids with broken qubits and couplers is essential for building a scalable FTQC.

Current state of the art dropout handling methods handle missing couplers by dropping the attached data qubit.

They also struggle with clustered dropouts, as punctures can merge into large superstabilizers.

Surface codes and mid-cycle states

Quantum AI



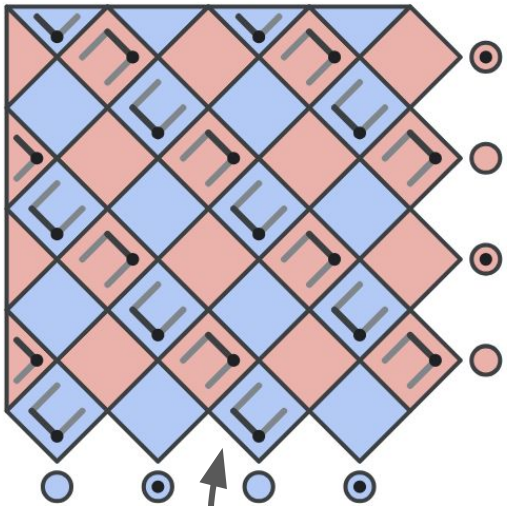
The end-cycle state is the standard rotated surface code state

The mid-cycle state is an *unrotated* surface code state

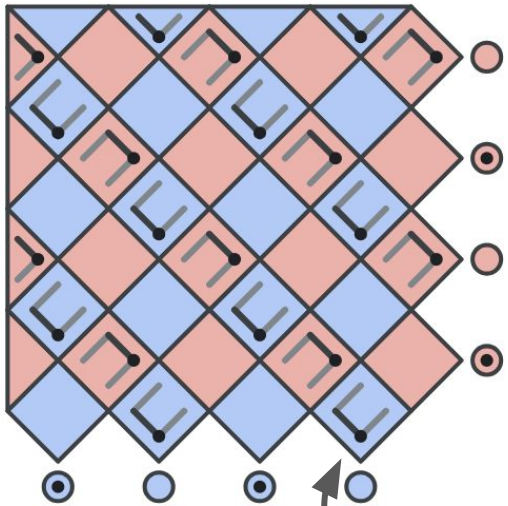
Figure adapted from McEwen, Bacon, Gidney, *Quantum,* (2023)

Understanding LUCI diagrams

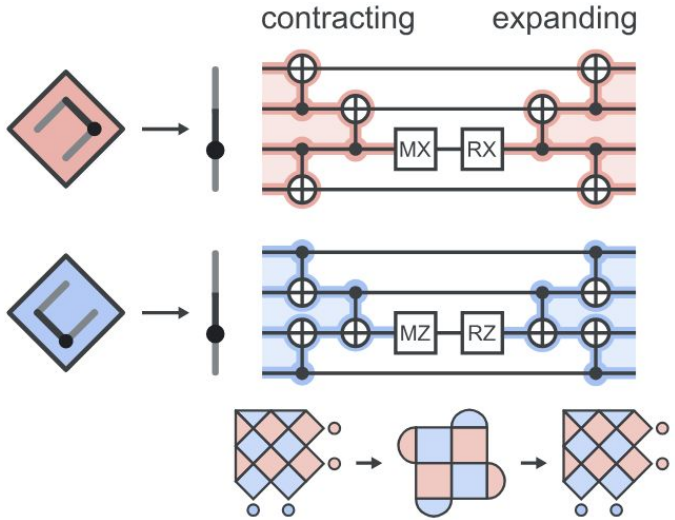
Quantum AI



Occupied squares show which mid-cycle stabilizers are being measured each round



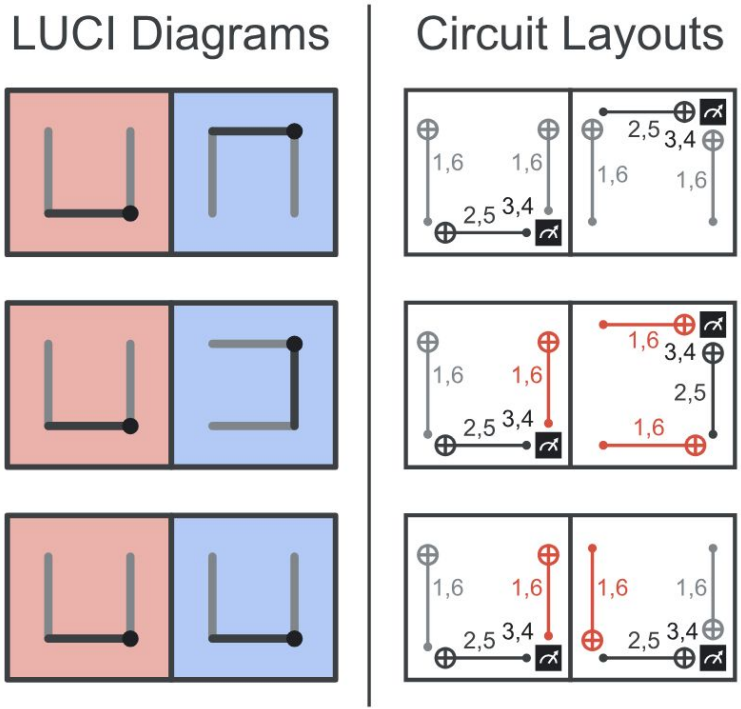
Dots indicate which qubits are being measured



Compilation of a shape depends on the basis of the underlying square.

Compatibility of LUCI shapes

Quantum AI



Neighboring shapes share qubits, so the operations on those qubits must fit together for the squares to be valid neighbors.

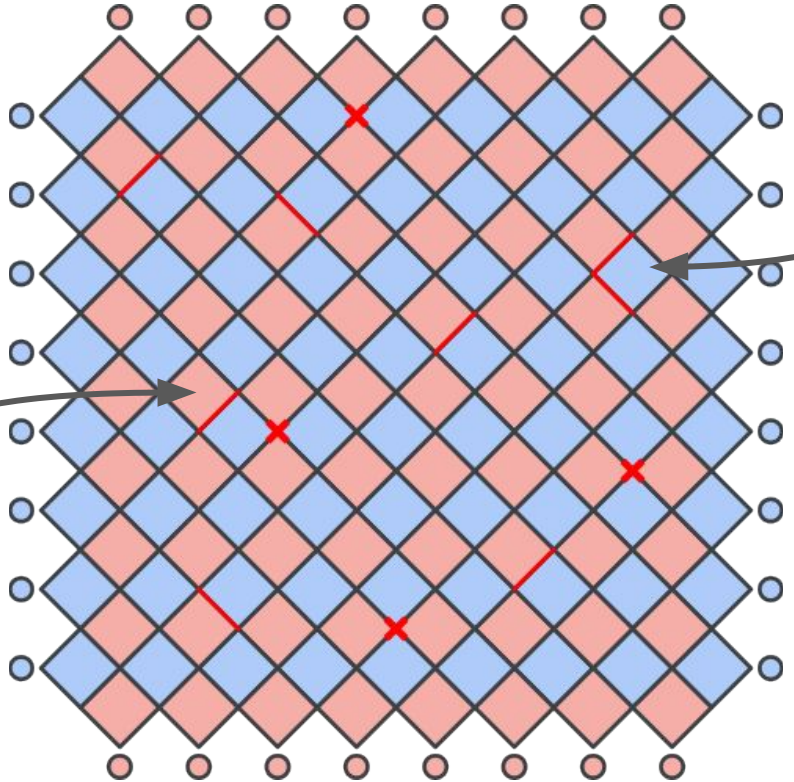
This constrains shapes to follow a “snake” pattern, like the LUCI diagram in the previous slide.

Any collisions indicate the two shapes cannot be adjacent in the same round, and would need to be separated in time.

If broken components require conflicting shapes, we must separate them in time.

Finding the mid-cycle state

Quantum AI

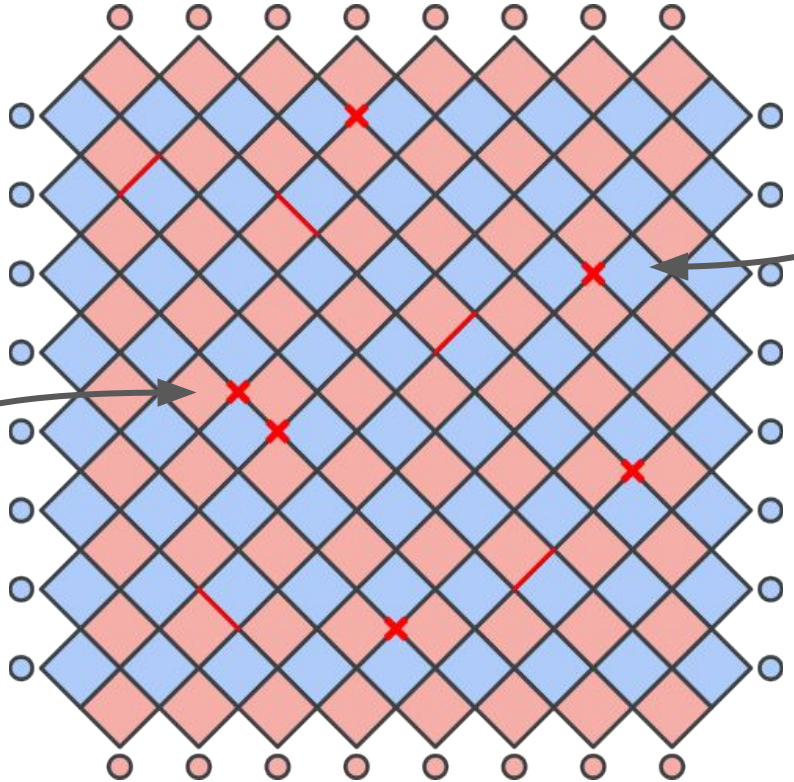


Remove qubit in case where two broken couplers touch a qubit.

Remove qubit in case where broken coupler is adjacent to broken qubit

Finding the mid-cycle state

Quantum AI

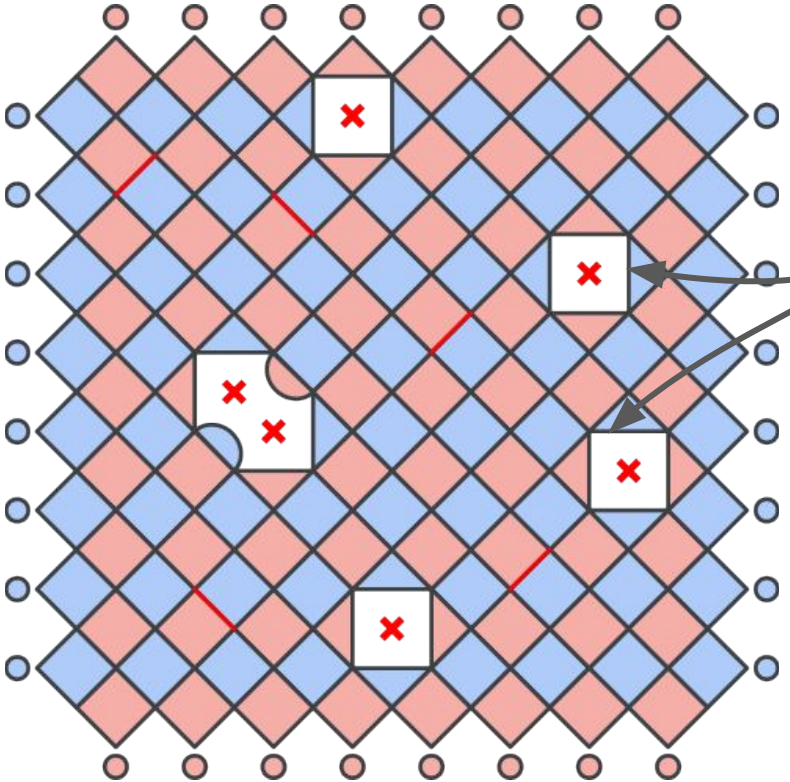


Remove qubit in case where two broken couplers touch a qubit.

Remove qubit in case where broken coupler is adjacent to broken qubit

Finding the mid-cycle state

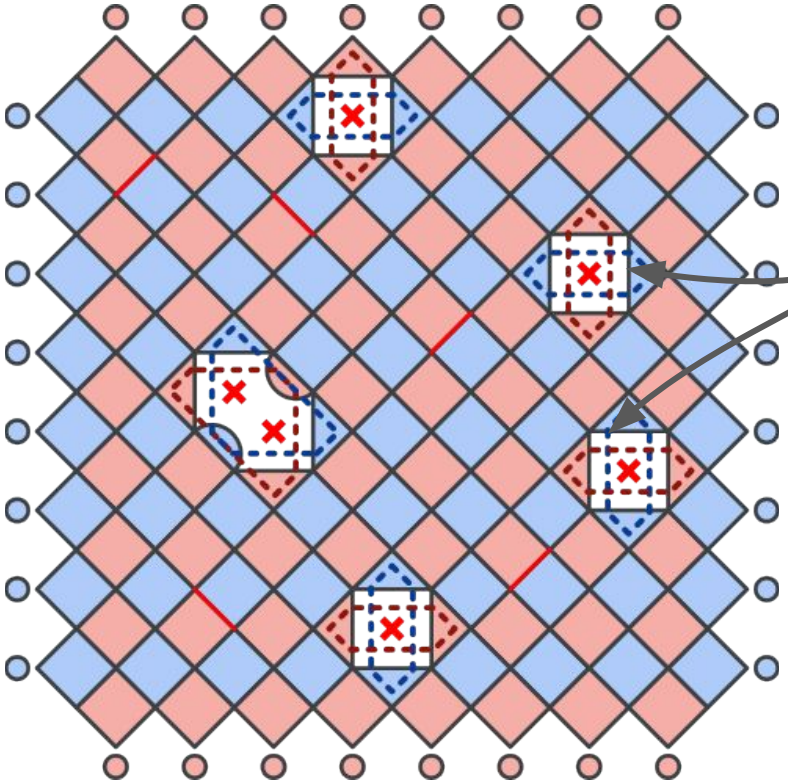
Quantum AI



Cut out broken qubits, converting stabilizer into gauges.

Finding the mid-cycle state

Quantum AI

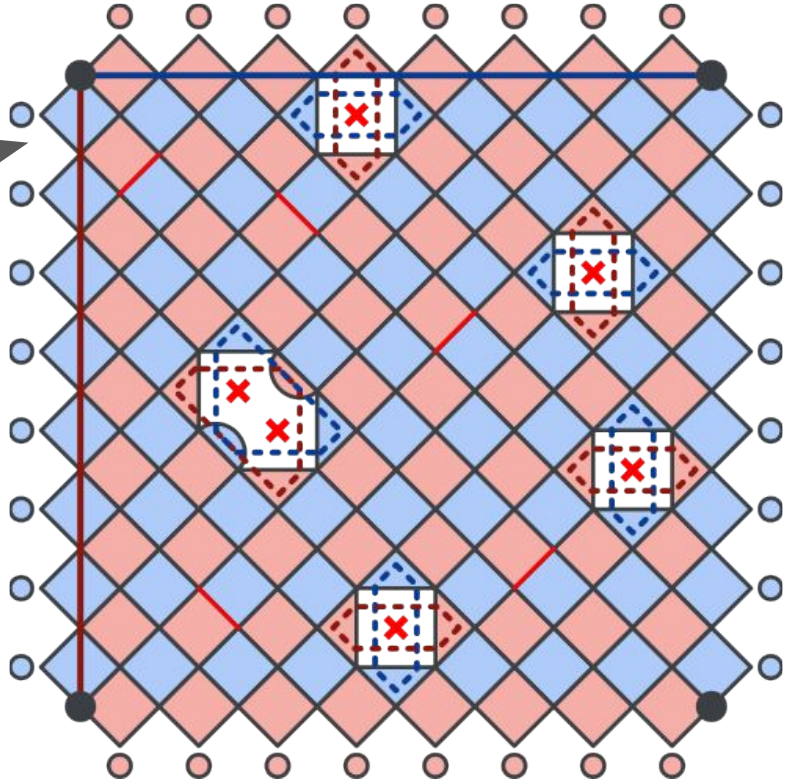


Group gauges together to form superstabilizers

Finding the mid-cycle state

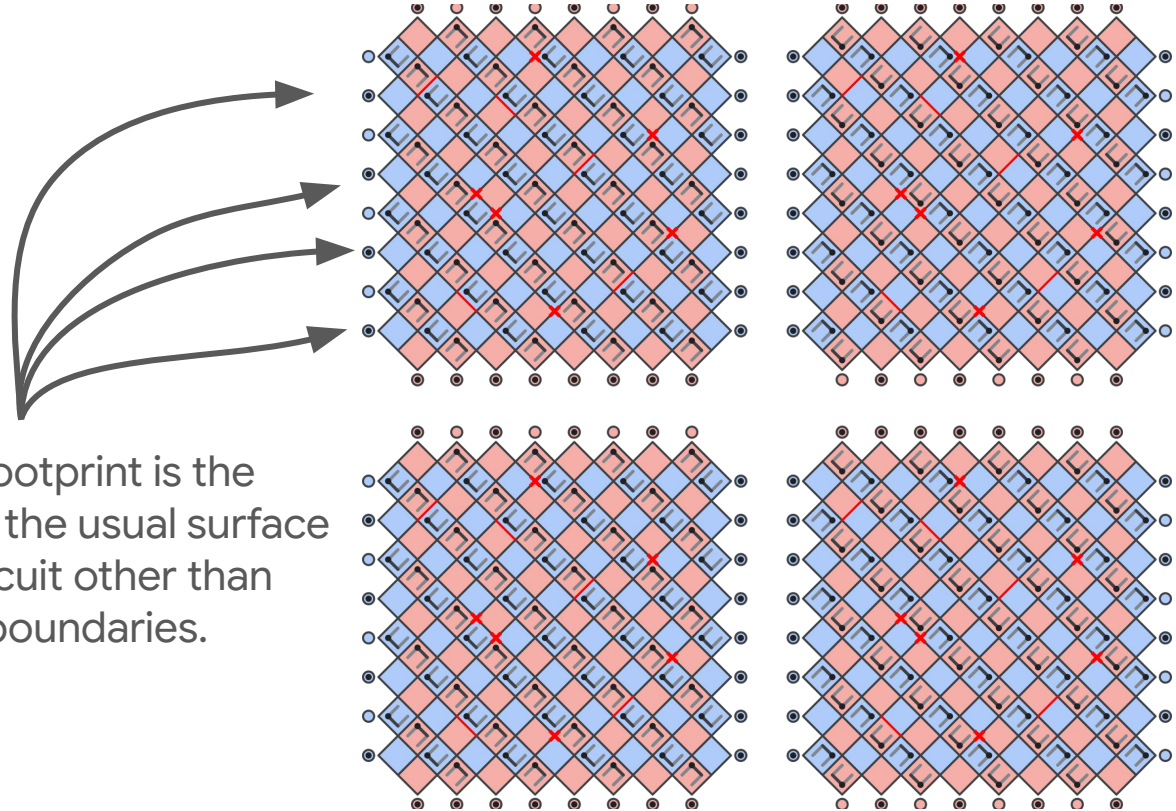
Quantum AI

Logical operators propagate over time to form a sheet, so finding them in the mid-cycle is sufficient.



Building the LUCI diagram

Quantum AI

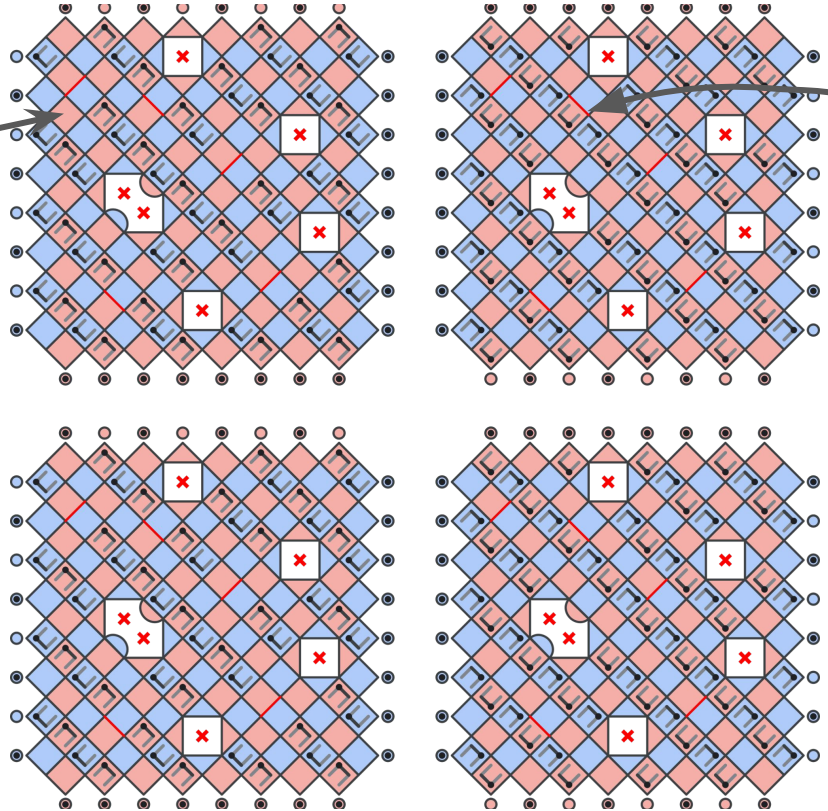


Same operations as the usual surface code, but reordered to reverse every other round: ABCD-DCBA-ABCD...

Circuit footprint is the same as the usual surface code circuit other than filled in boundaries.

Building the LUCI diagram

Quantum AI

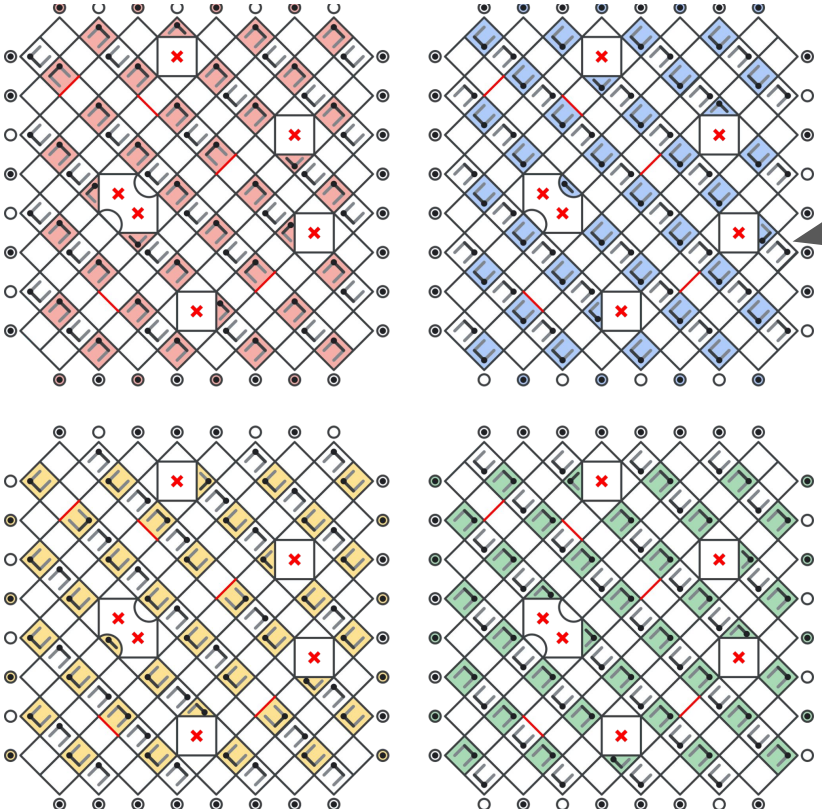


Depending on arrangement of dropouts a different base play may be preferable.

Don’t need to remove shape in cases where dropouts are respected.

Building the LUCI diagram

Quantum AI

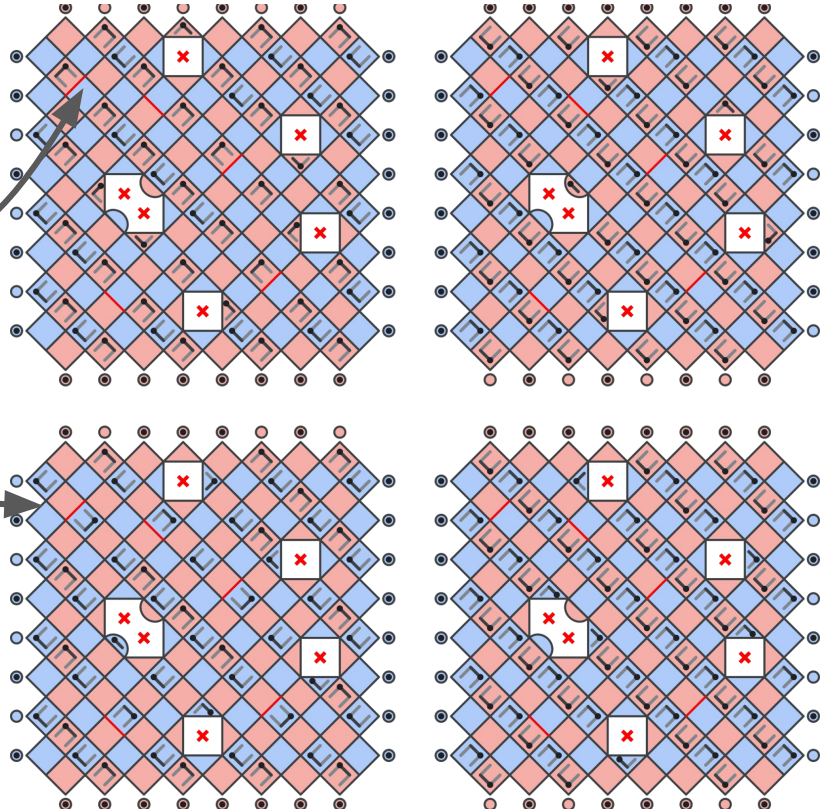


When a collision occurs we remove the shape that is is not colored.

Four-coloring guarantees that every square is measured without any compatibility issues.

Building the LUCI diagram

Quantum AI

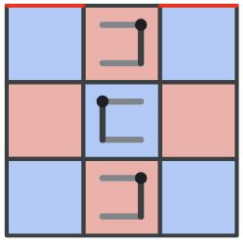


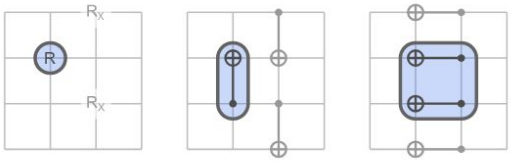
This method does **not** guarantee an optimal LUCI diagram for the dropout configuration.

Areas with big gaps will lead to high error rate detectors and weaknesses in the decoder graph.

Lifecycle of a detecting region

Quantum AI



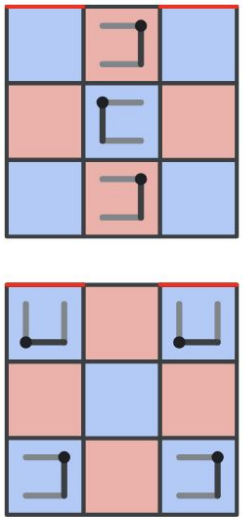


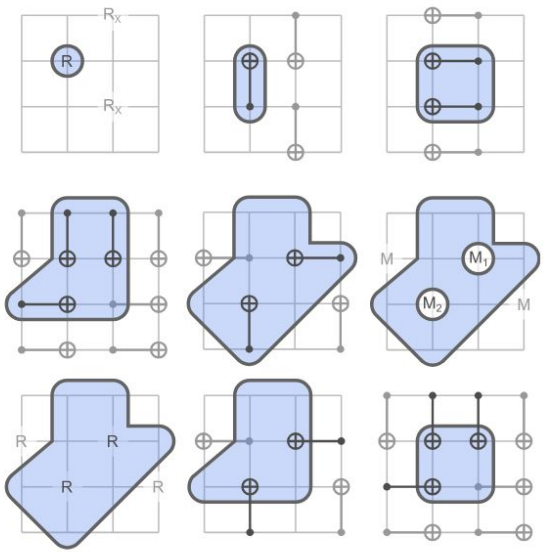
A detecting region goes through a number of stages over the rounds:

1. Initialize from a reset, and then expand into a mid-cycle stabilizer.

Lifecycle of a detecting region

Quantum AI



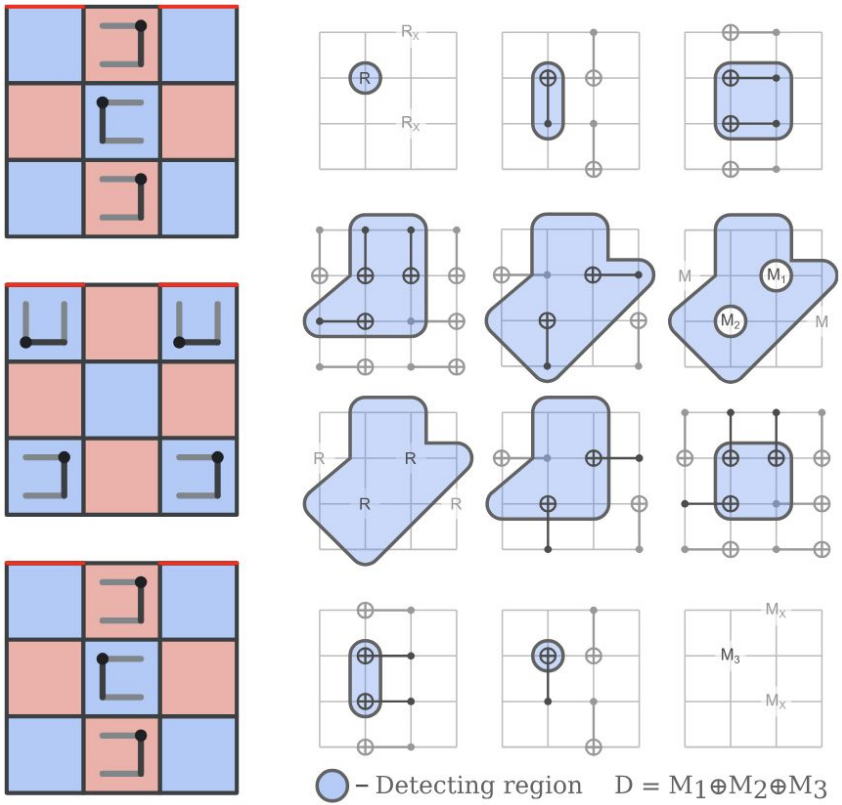


A detecting region goes through a number of stages over the rounds:

1. Initialize from a reset, and then expand into a mid-cycle stabilizer.
2. Pass through a few measurements in round(s) where it is not measured directly, including both the measurement and reset so that it returns to its usual footprint at each mid-cycle.

Lifecycle of a detecting region

Quantum AI

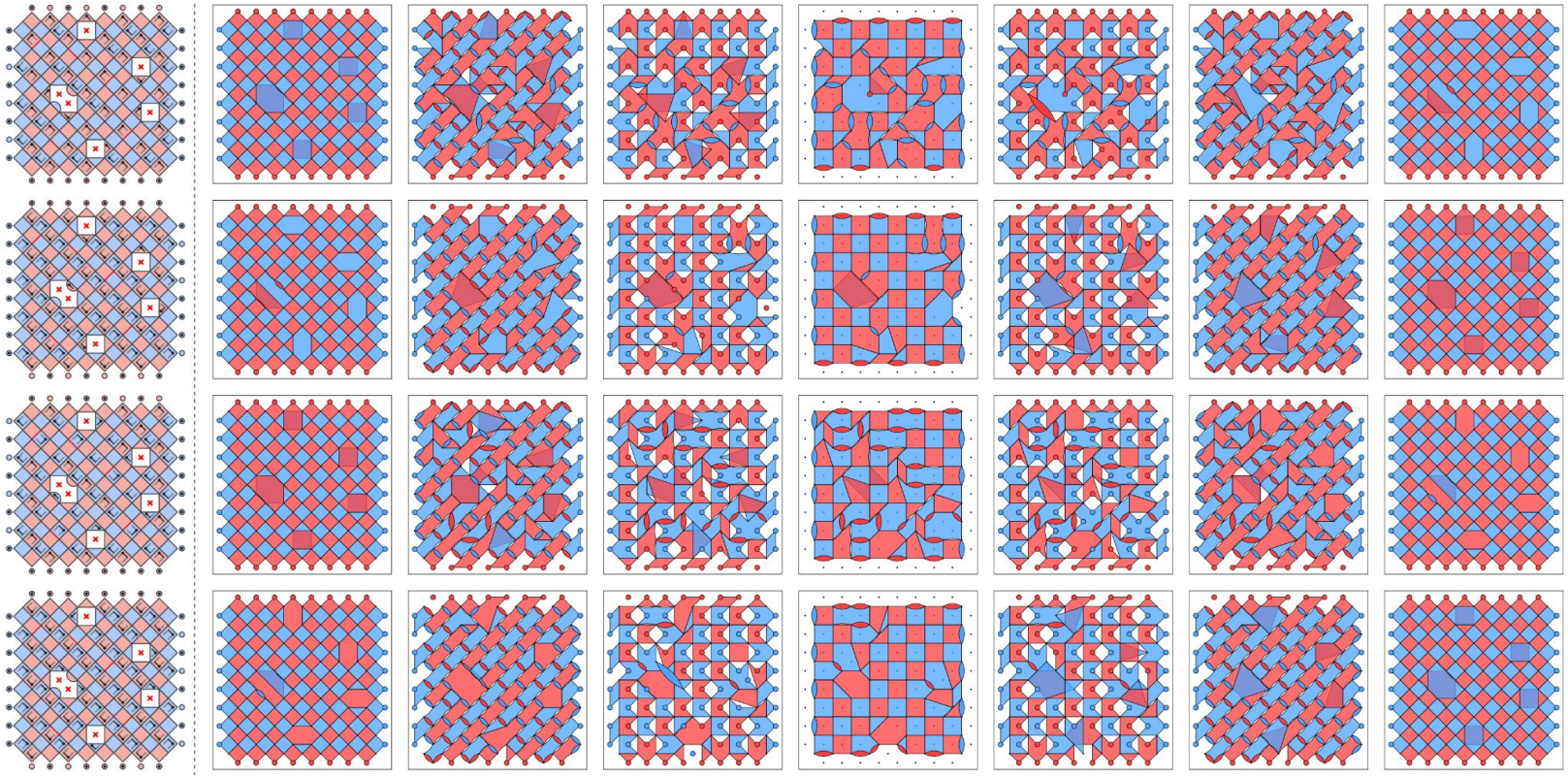


A detecting region goes through a number of stages over the rounds:

1. Initialize from a reset, and then expand into a mid-cycle stabilizer.
2. Pass through a few measurements in round(s) where it is not measured directly, including both the measurement and reset so that it returns to its usual footprint at each mid-cycle.
3. Measure out the mid-cycle stabilizer the next time that square is measured.

Detector slices for a LUCI circuit

Quantum AI



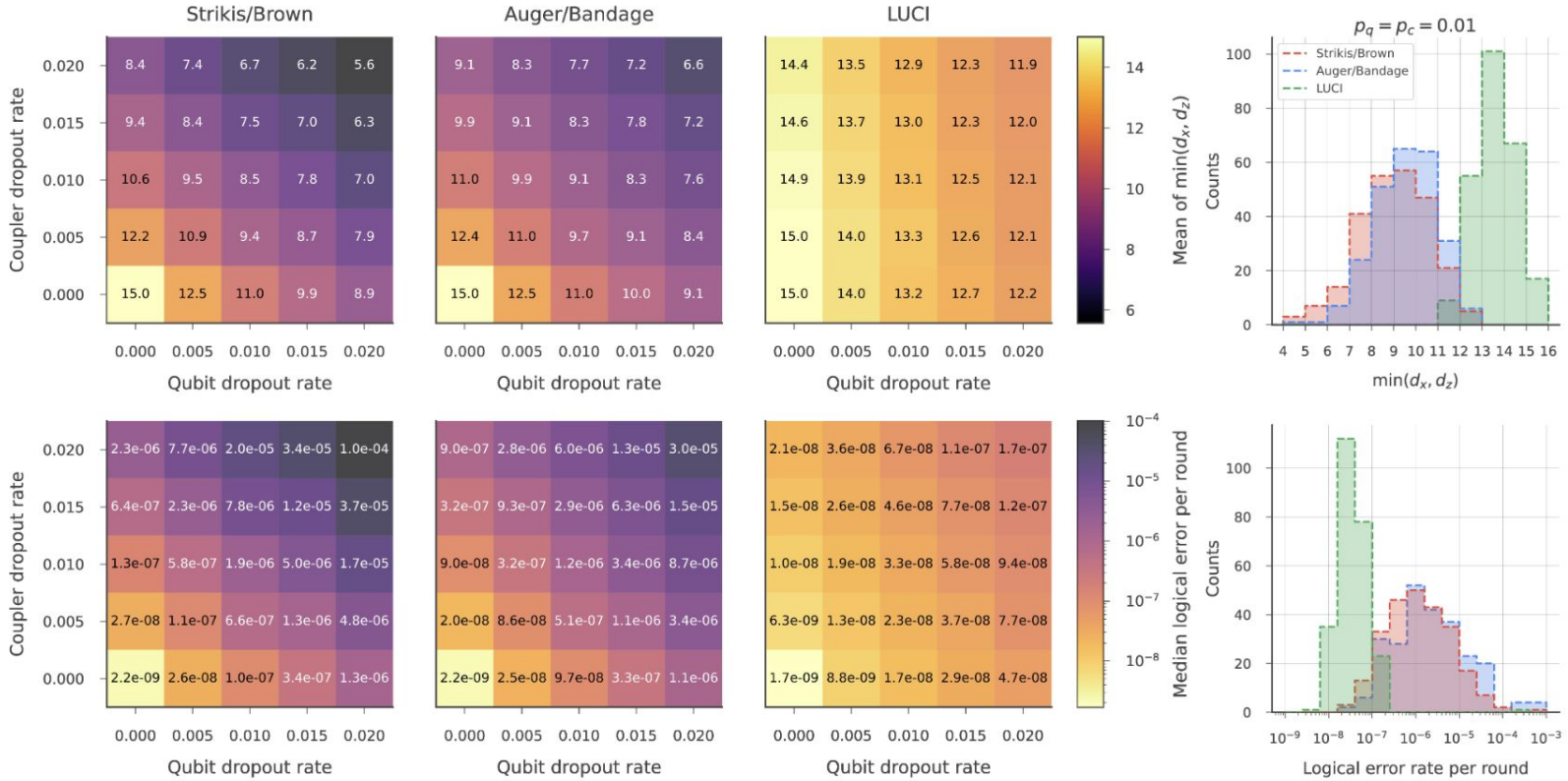
Mid-cycle is well structured and acts as the transition, while end-cycle states are messy.

Performance for ensembles of dropouts

Quantum AI

Distance

Logical Error Rae



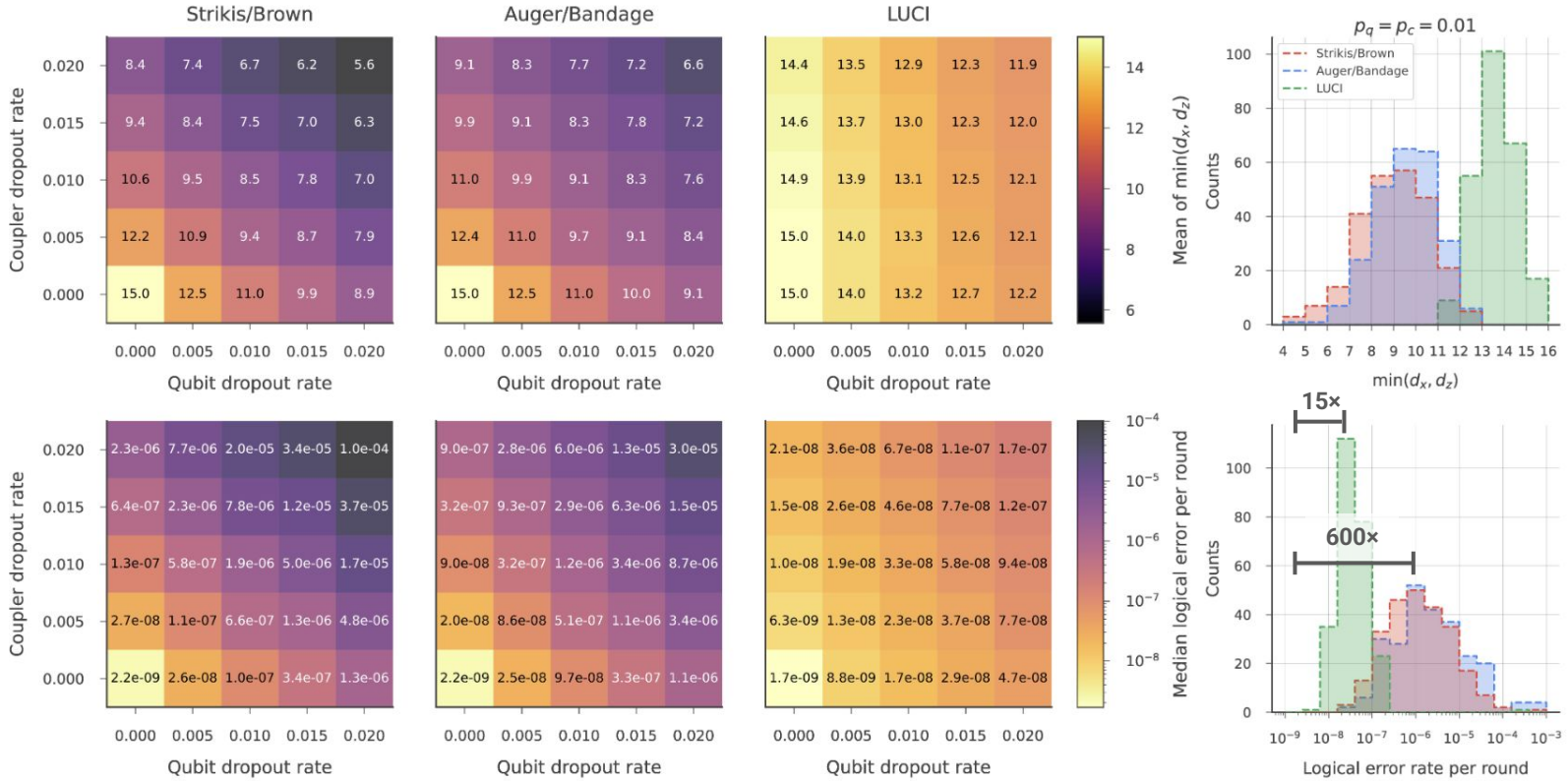
Other methods: Strikis et al., PRApplied (2023), Lin et al., ASPLOS (2024), Auger et al., PRA (2017), Wei et al., arXiv (2024)

Performance for ensembles of dropouts

Quantum AI

Distance

Logical Error Rae

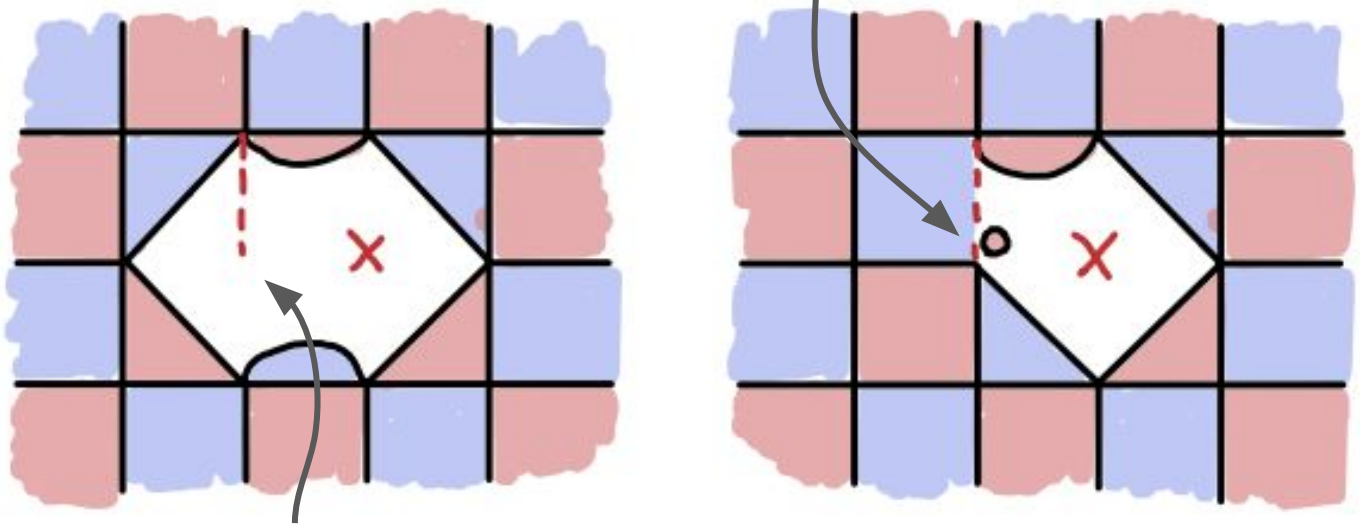


Other methods: Strikis et al., PRApplied (2023), Lin et al., ASPLOS (2024), Auger et al., PRA (2017), Wei et al., arXiv (2024)

Improving mid-cycle gauge groups

Quantum AI

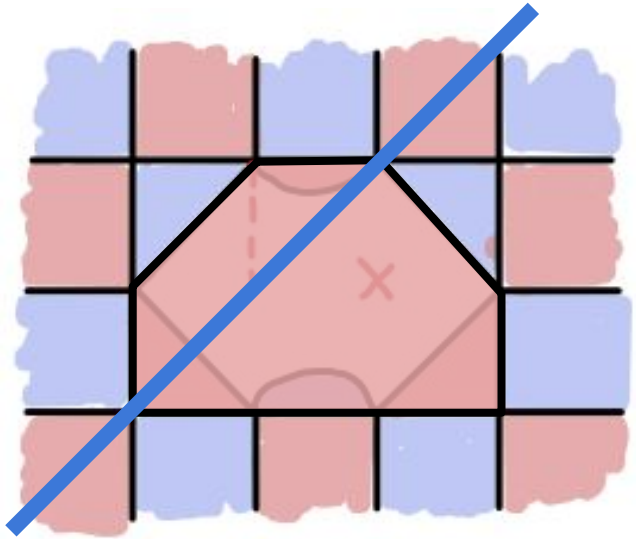
By introducing weight-1 gauge operators this qubit can be kept



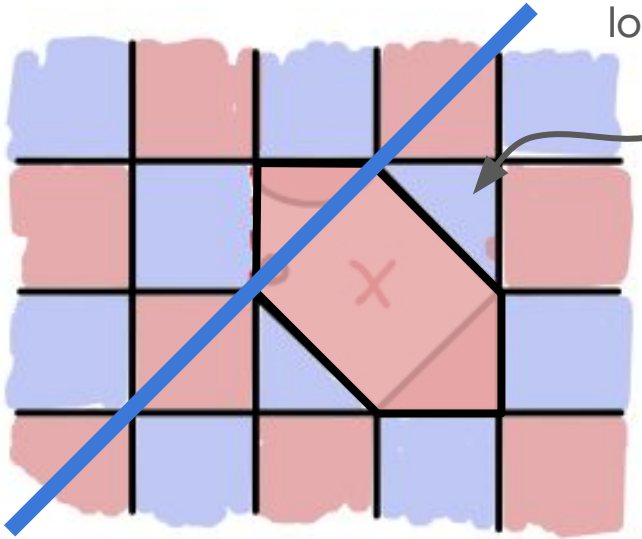
Previously thought that a qubit needed at least one coupler to access the qubits around it for each mid-cycle stabilizer.

Improving mid-cycle gauge groups

Quantum AI



$d_z - 1$

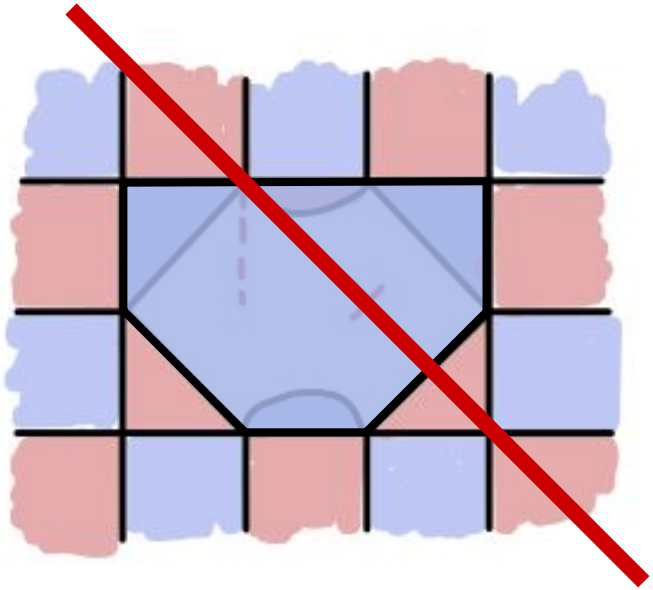


$d_z - 0$

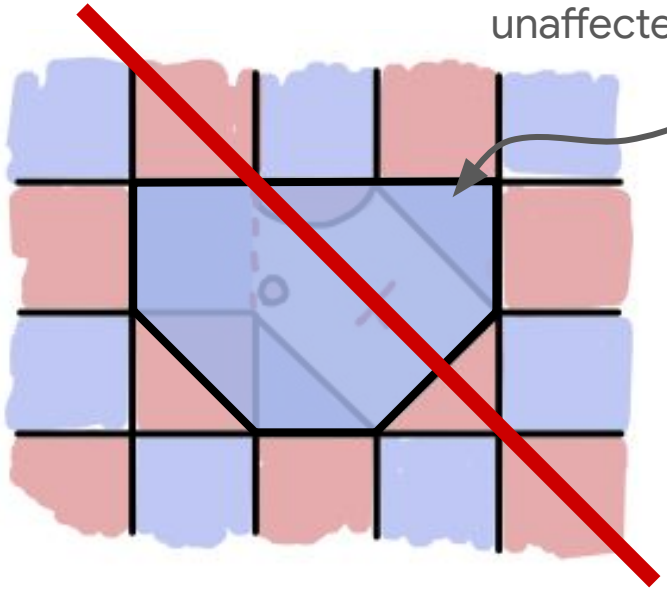
In half of cases we would still lose distance

Improving mid-cycle gauge groups

Quantum AI



$d_x - 1$



Other basis essentially unaffected by change

$d_x - 1$

Circuit refinement with ILP

Quantum AI

Integer Linear Programming (ILP) is an optimization framework where a solver is given a cost function with both hard constraints and rewards.

Effective solvers can be found off the shelf, so we can take advantage of the work being done in the classical optimization space.

These ideas have been expanded to color codes and BB codes by [Wolanski, arXiv: 2512.01943].

Can we find methods to generate performant 2+1 dimensional spacetime tilings beyond codes?

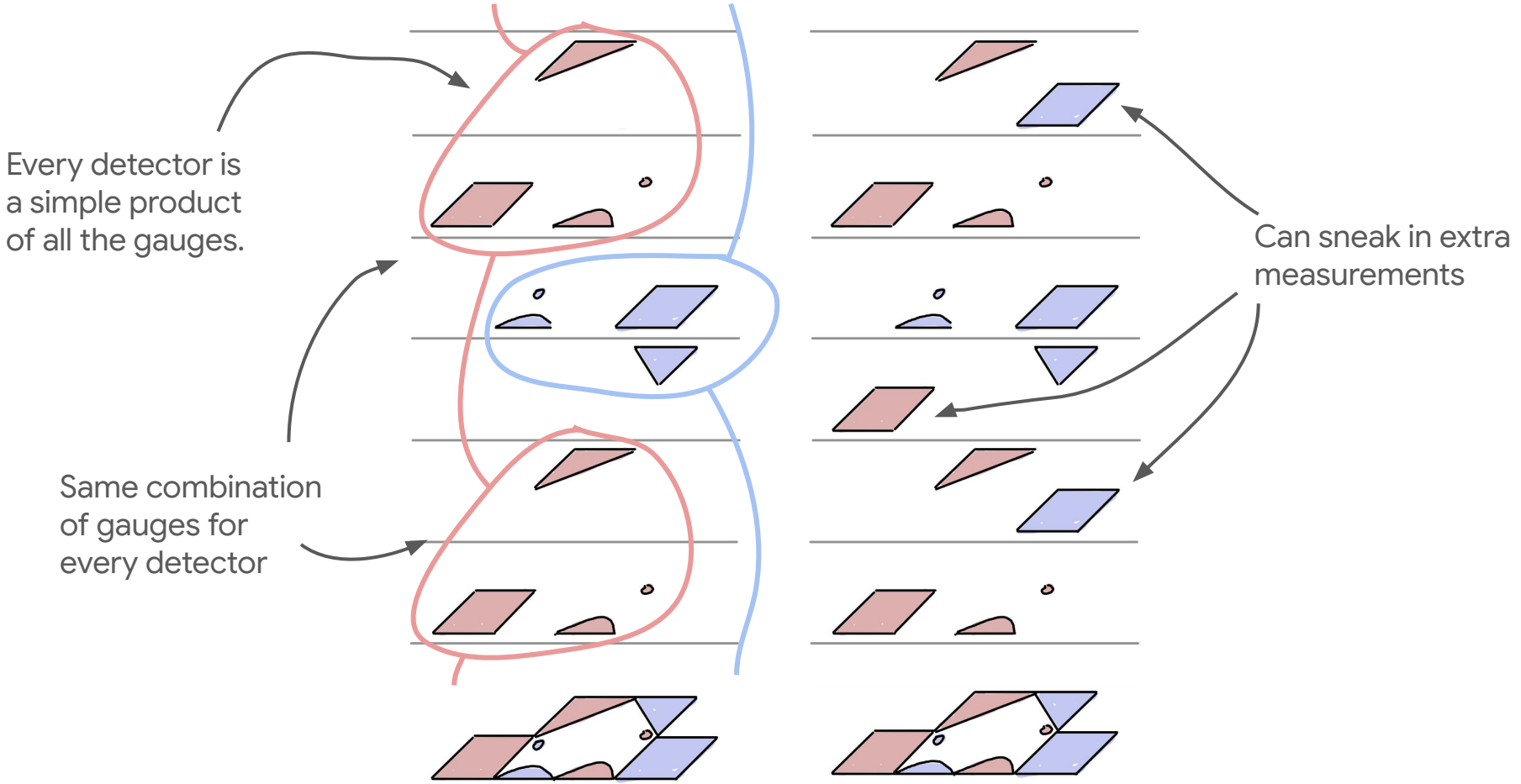
We can describe the problem of generating a valid LUCI diagram as an ILP problem:

Hard constraints:
- Avoiding gate collisions
- Measure each stabilizer once over 4 rounds
- Measure gauges in a valid order

Rewards and Penalties:
- Deterministic measurements (+1 detector)
- Skip measuring a given Pauli for two rounds
- Skip measuring a given Pauli for three rounds
- Detecting region “stretching” in multiple directions simultaneously
- etc.

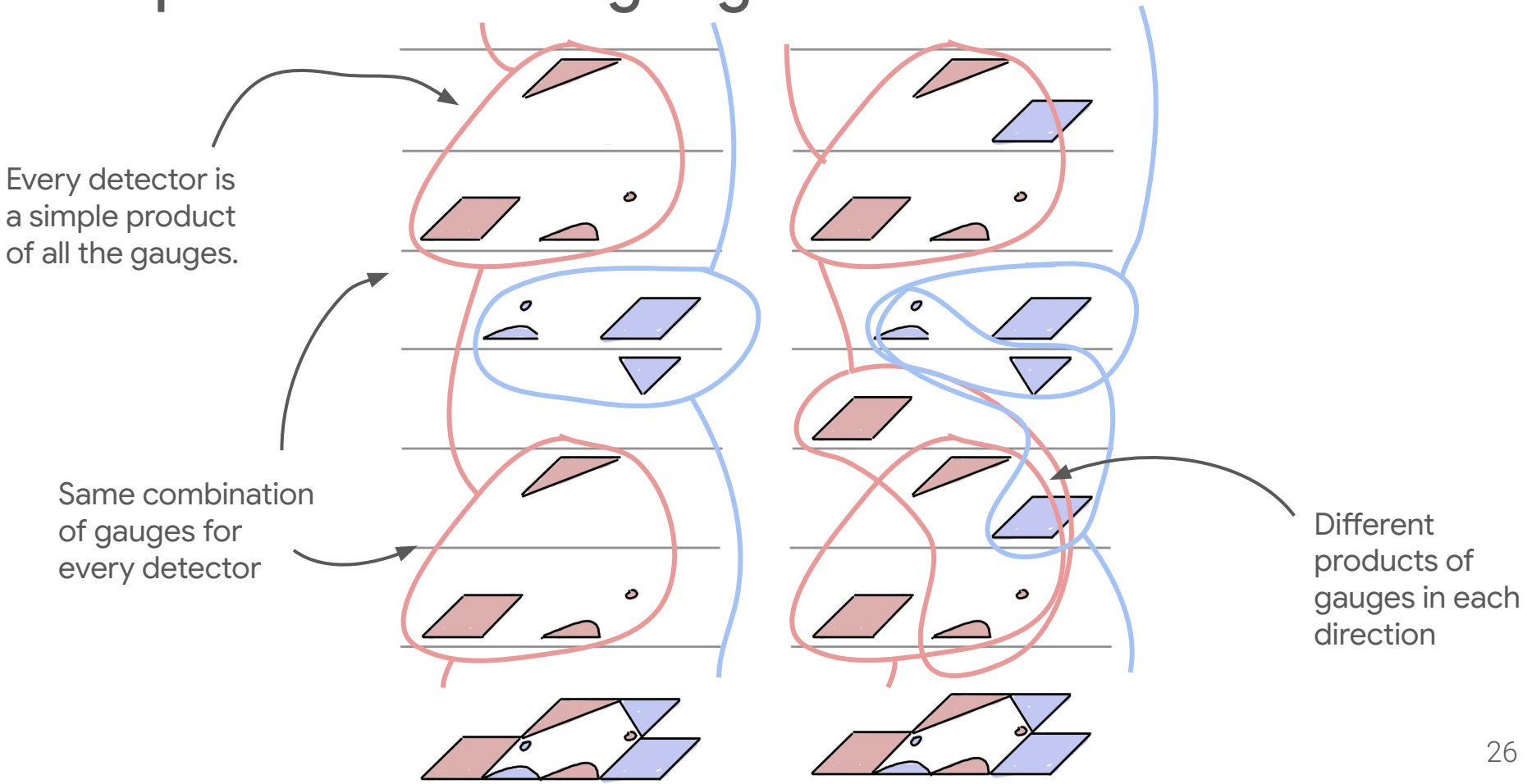
Improved detecting regions

Quantum AI



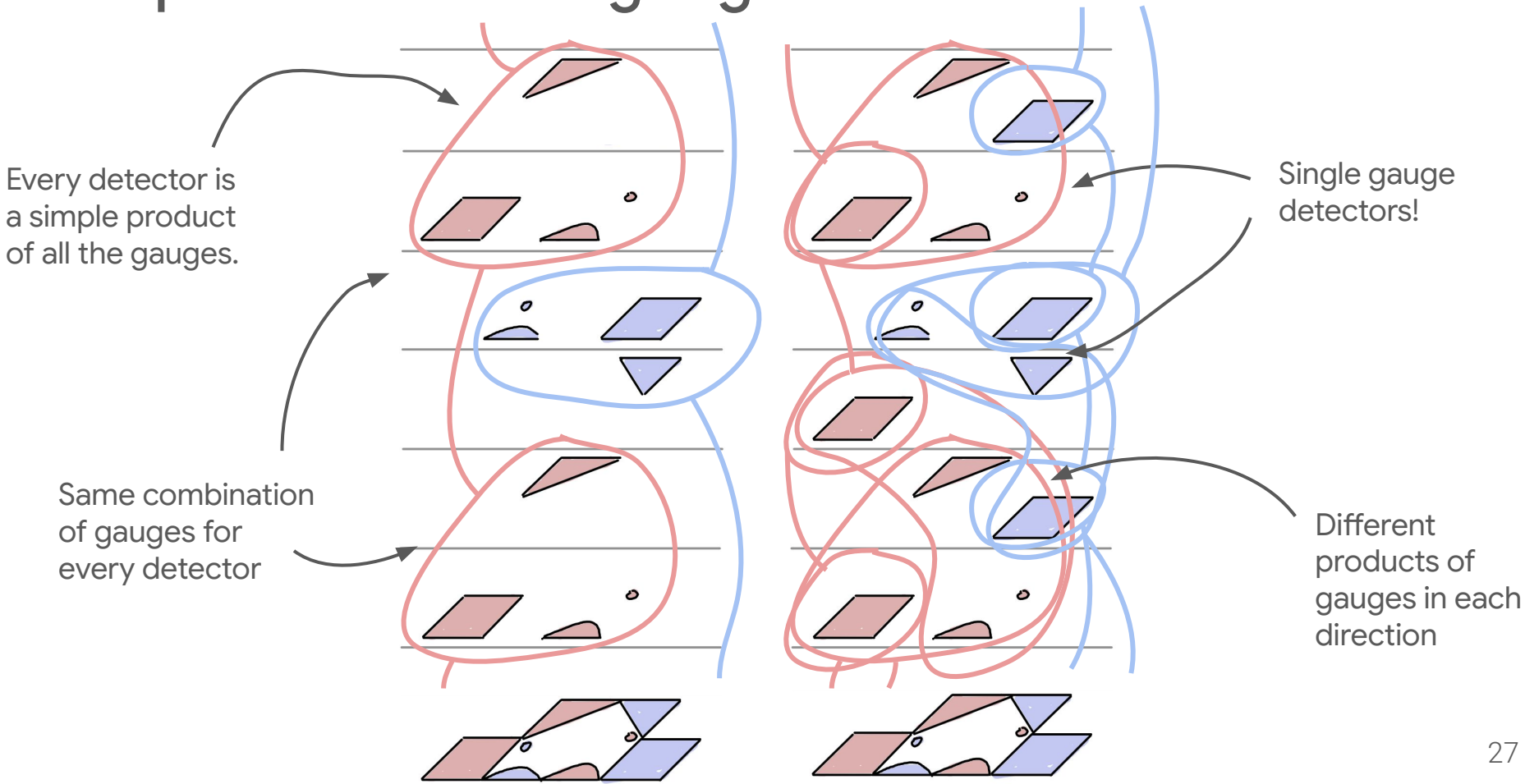
Improved detecting regions

Quantum AI

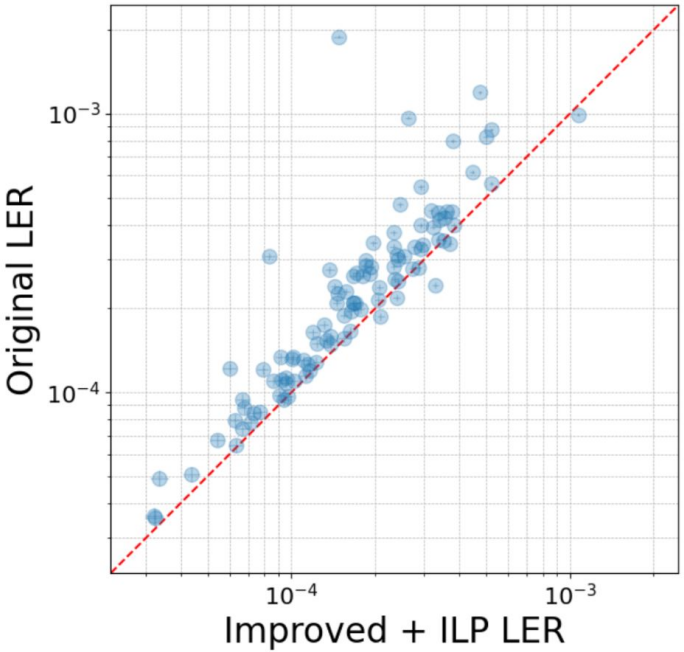


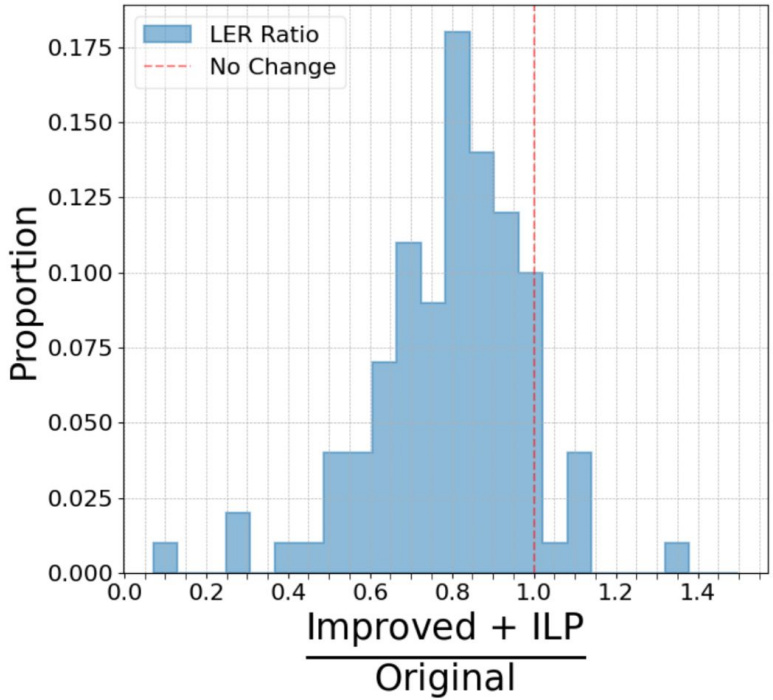
Improved detecting regions

Quantum AI



Quantifying improvement with simulation

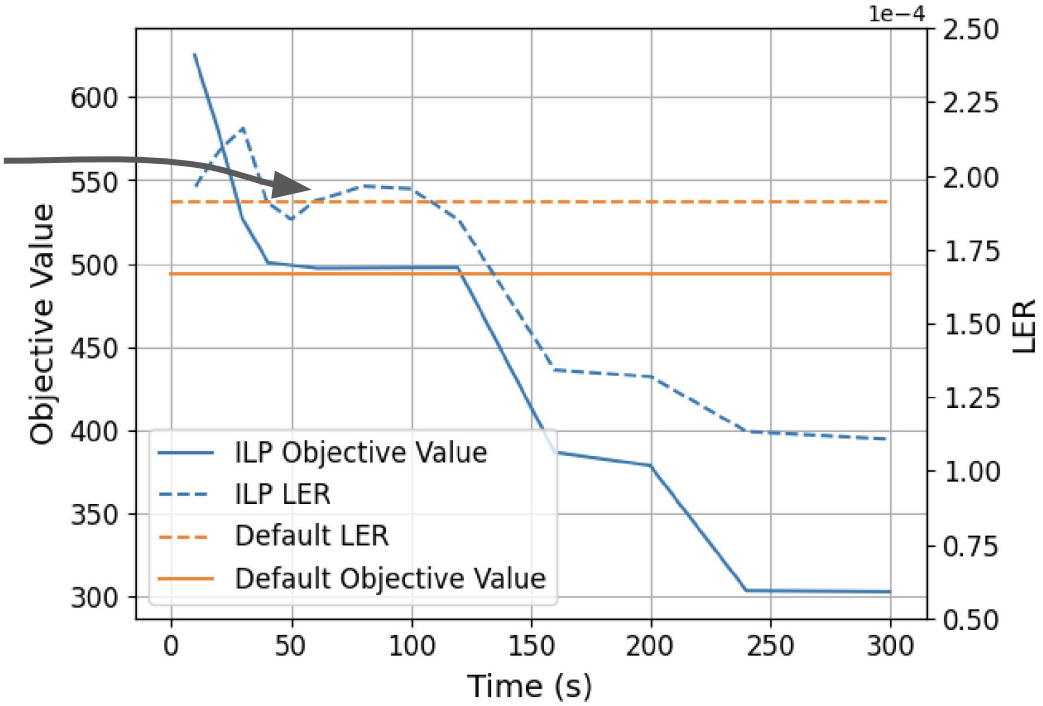




Simulations for distance-11 footprints with 3% qubit and coupler dropout rates and a SI1000(0.001) noise model.

Can our cost function be improved?

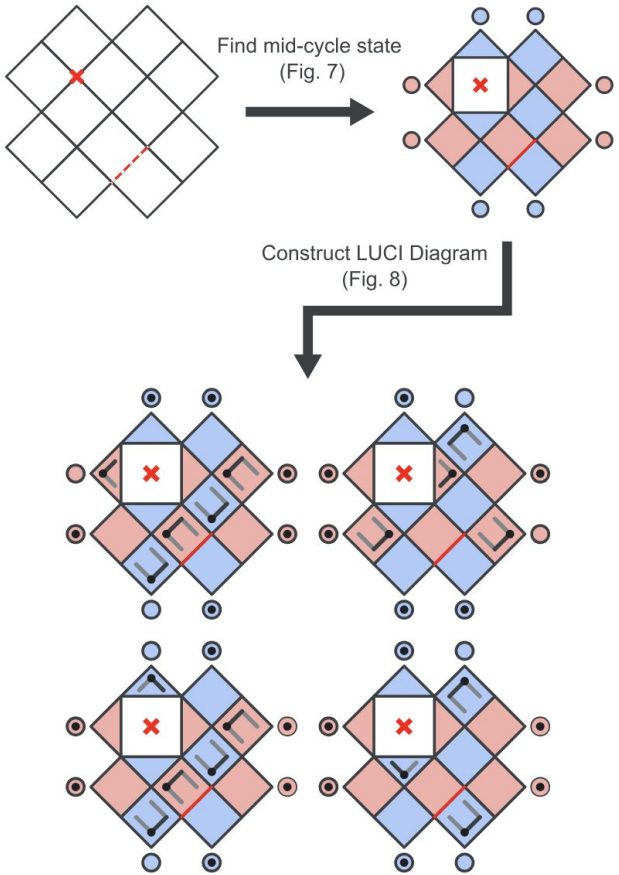
Non-monotonic behavior implies that the cost model does not perfectly reflect LER.



Overall LER and objective value track closely, so we are aligned, but may be missing some details.

Conclusions

Quantum AI



LUCI is a flexible framework which describes a family of fault-tolerant circuits, including circuits with implications beyond yield.

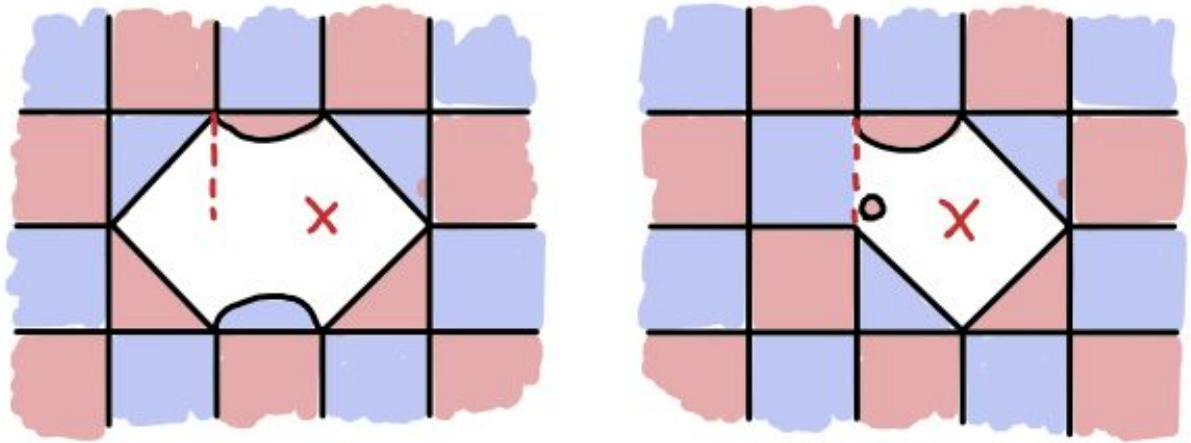
These circuits dramatically outperform previous techniques, lowering the cost of fabrication failures and outliers.

The success of this framework shows that there is value in considering other mid-cycle constructions.

These techniques could be extended to find with resource efficient spacetime tilings not represented by codes.

Thank you!

Quantum AI



Papers Referenced:

[1] *LUCI in the Surface Code with Defects*, Debroy, et al., Quantum (2025)

[2] *Quantum Computing is Scalable on a Planar Array of Qubits with Fabrication Defects,* Strikis, et al., PRApplied (2023)

[3] *Codesign of quantum error-correcting codes and modular chiplets in the presence of defects*, Lin, et al., ASPLOS (2024)

[4] *Fault-tolerance thresholds for the surface code with fabrication errors*, Auger, et al., PRA (2017)

[5] *Low-Overhead Defect-Adaptive Surface Code with Bandage-Like Super-Stabilizers,* Wei, et al., arXiv (2024)

[6] *Handling fabrication defects in hex-grid surface codes*, Higgott, et al., arXiv (2025)

[7] *Optimized Measurement Schedules for the Surface Code with Dropout,* Anker, et al., arXiv (2025)

[8] *Automated Compilation Including Dropouts: Tolerating Defective Components in Stabiliser Codes,* Wolanski, arXiv (2025)