Quantum AI

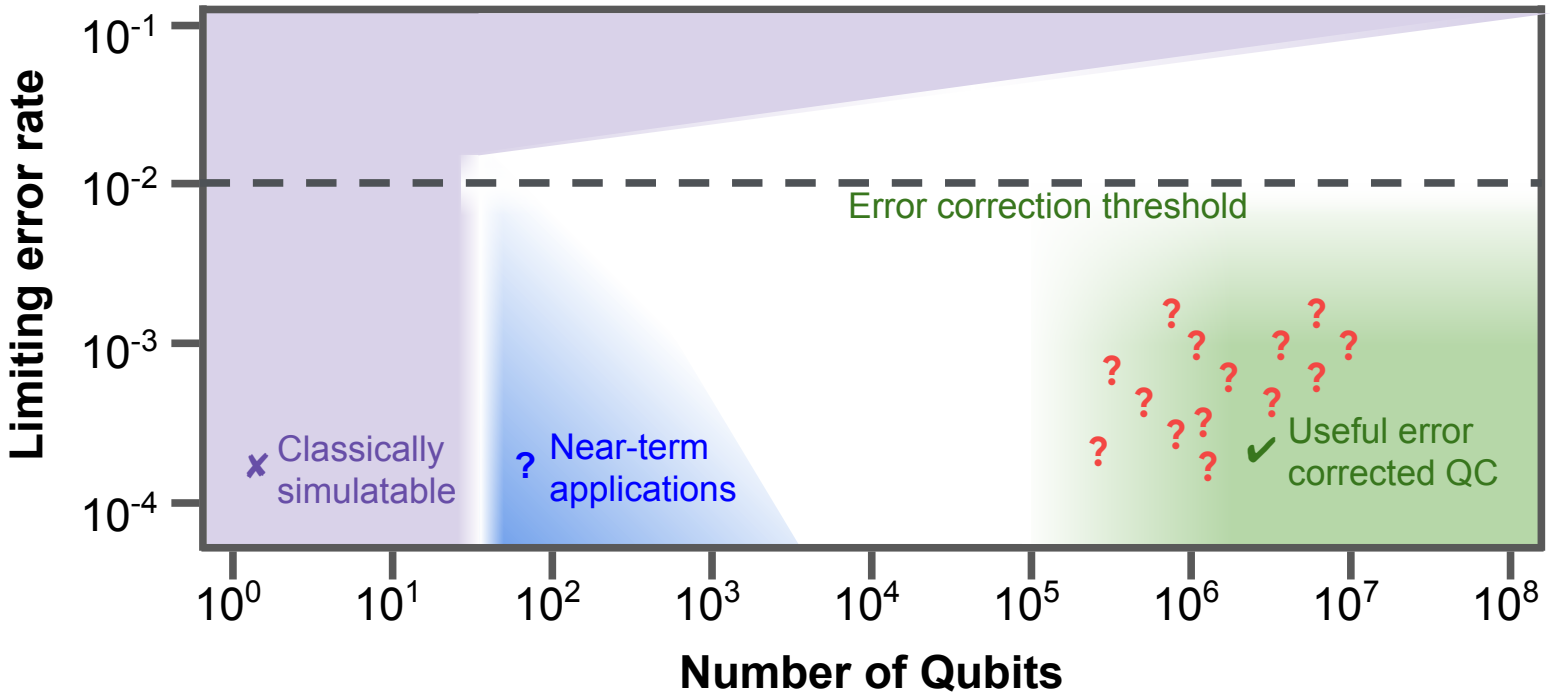
The FLuid Allocation of Surface code Qubits model for early fault-tolerant resource estimation

William J. Huggins

WERQSHOP - July 2025

Google

Understanding the transition to fault-tolerance



Error mitigation for today's quantum computers...

Time evolution of the 2D Ising model

$$H = J \sum_{\langle ij \rangle} Z_i Z_j + h \sum_i X_i$$

Second-order Trotter

$$N_{2q} \approx 2N^2 \frac{T}{\Delta t}$$

Target O(1) errors

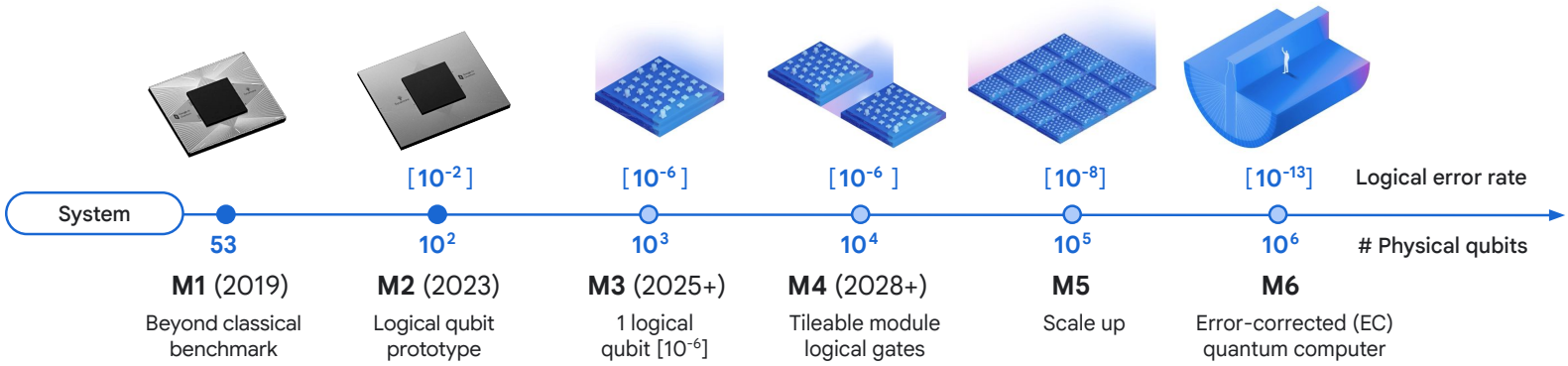
$$N_{2q} \lessapprox p_{2q}^{-1}$$

For example:

$$T = 1?$$
$$\delta T = .04$$
$$N = 10$$
$$p_{2q}^{-1} = .001$$

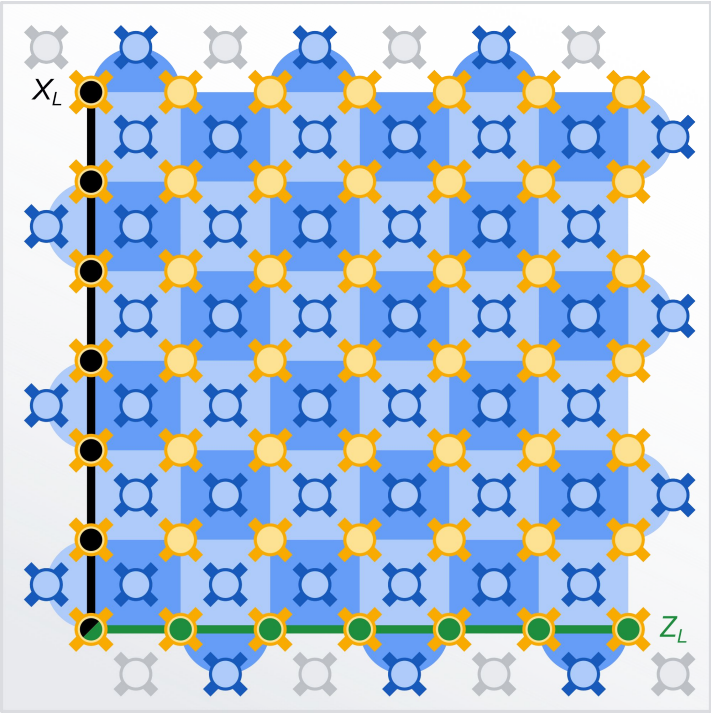
$$T \lessapprox .2$$

Understanding the transition to fault-tolerance



Ingredients

Surface code logical qubit



Physical qubit count = $2 * (d+1)^2$

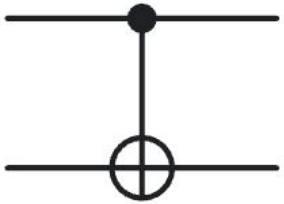
Logical Error Rate $\approx 0.1 * \Lambda^{-(d+1)/2}$

Surface code error correction gives exponentially more time for quadratically more space.

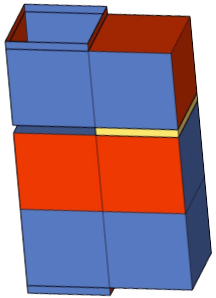
2D local connectivity is sufficient to store information and perform gates using lattice surgery

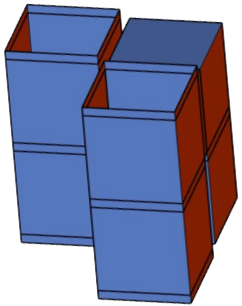
Everything* requires ancilla spacetime

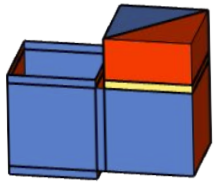
Gates in NISQ



Gates in the surface code







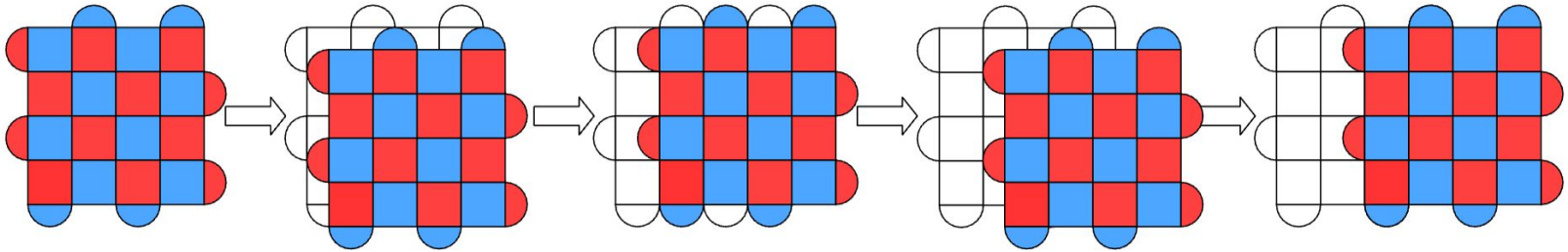
Rearranging the ancilla with walking codes

With limited qubits, we expect to have $N_{ancilla} \ll N_{data}$

We will want to move ancilla around

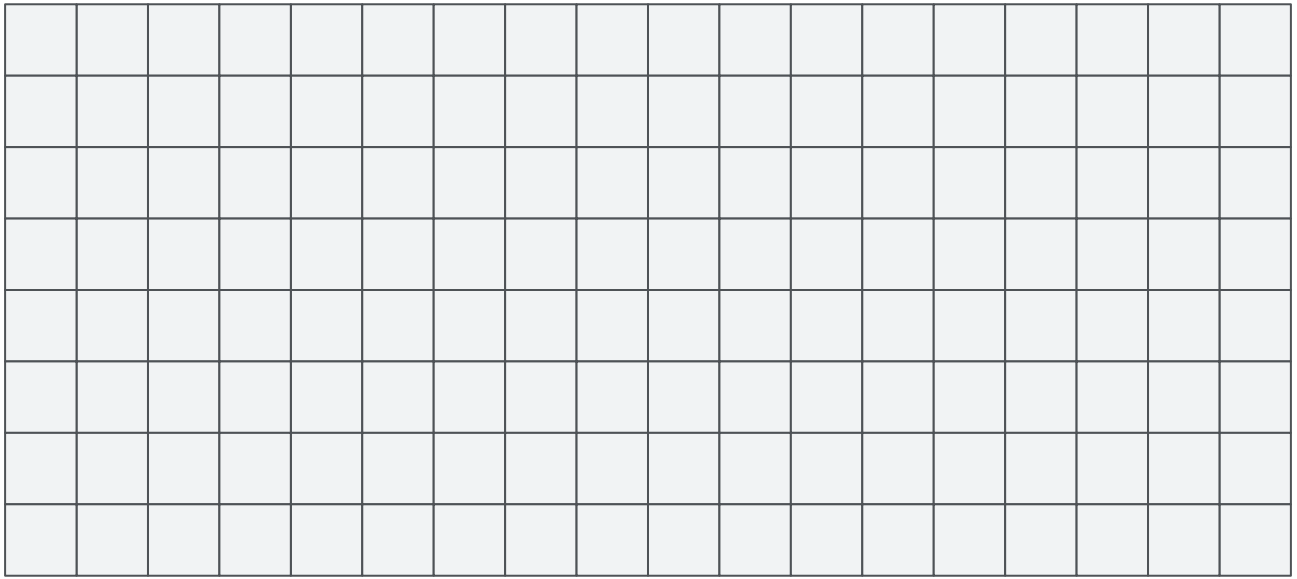
Walking surface codes let us move logical qubits

Whole patches can move together

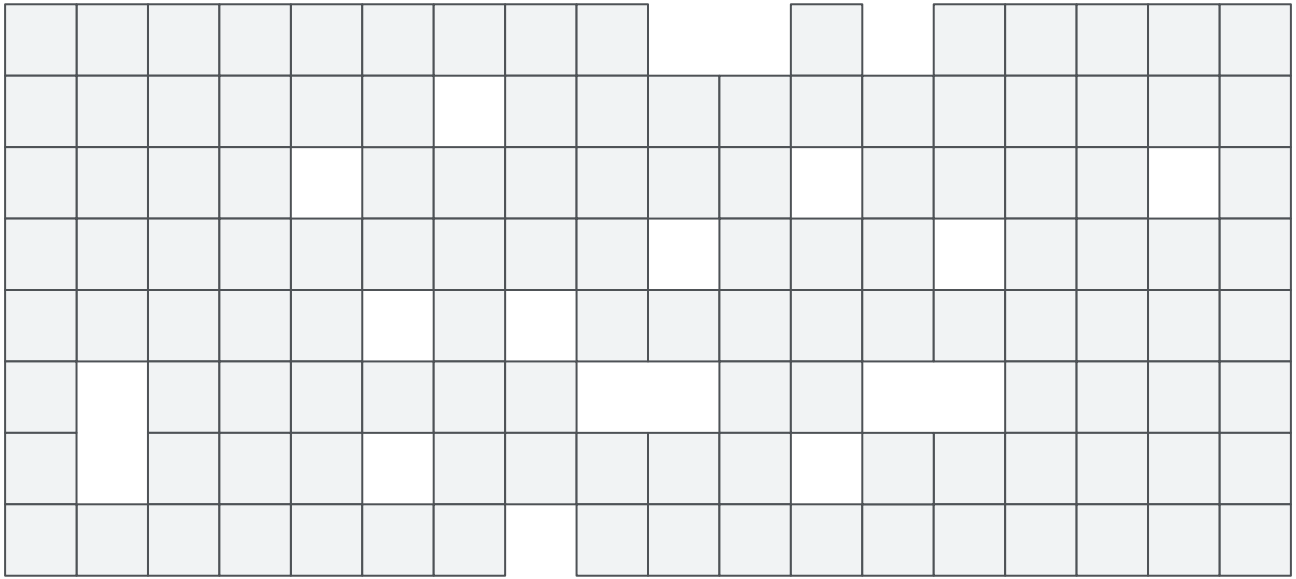


McEwen, M., Bacon, D. & Gidney, C. Relaxing hardware requirements for surface code circuits using time-dynamics. *Quantum* **7**, 1172 (2023).

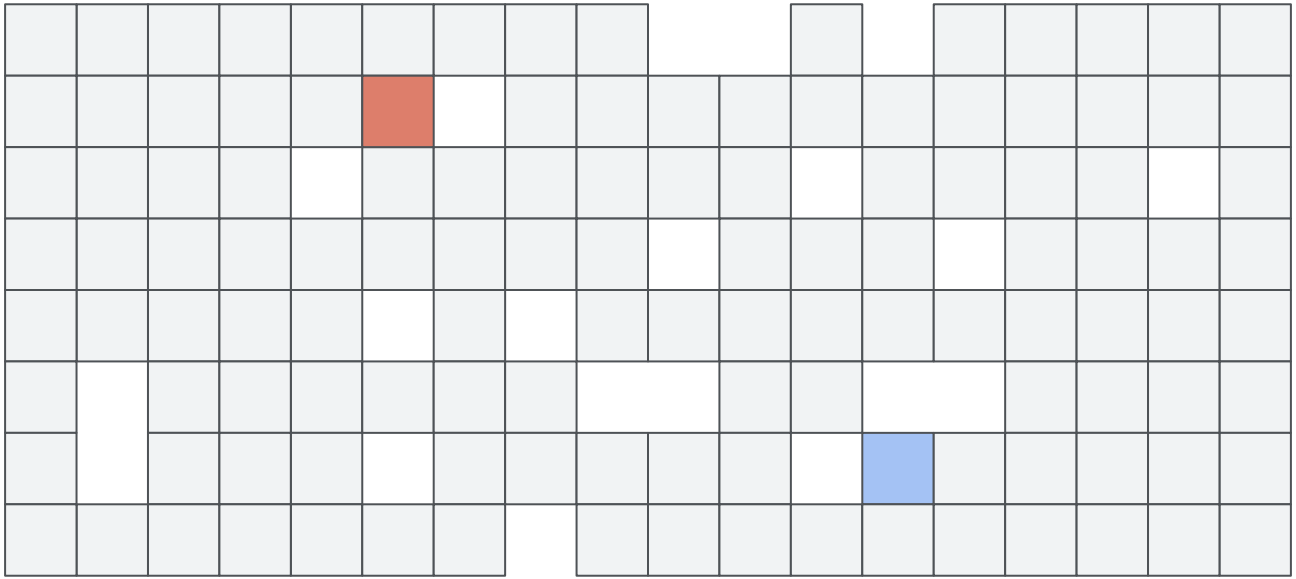
Walking codes in action



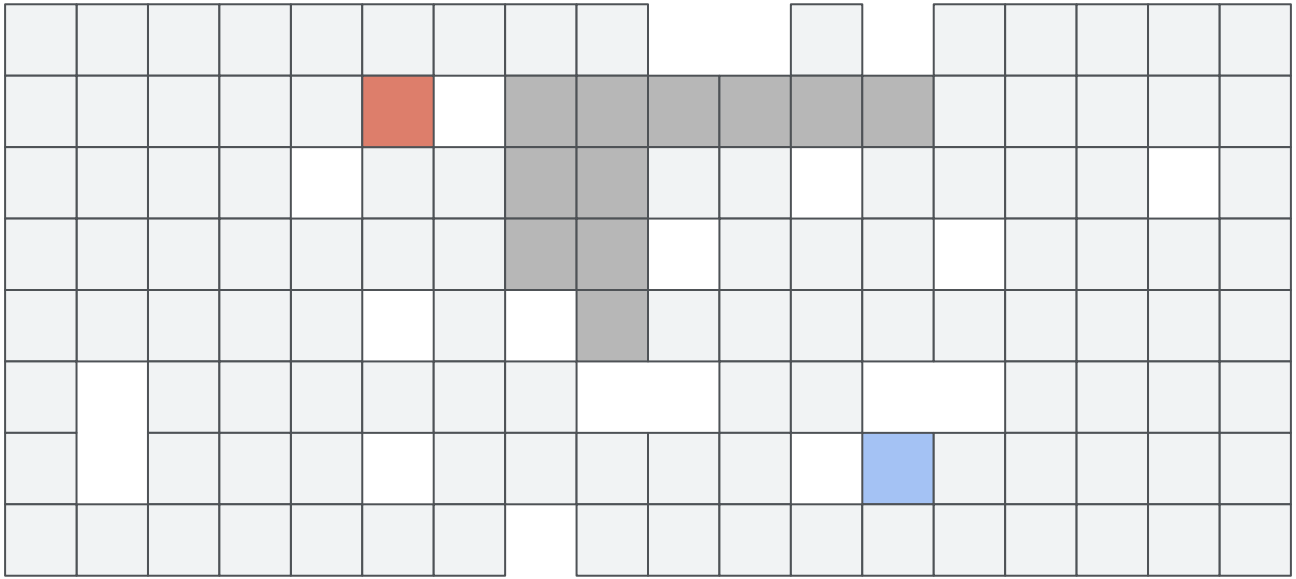
Walking codes in action



Walking codes in action

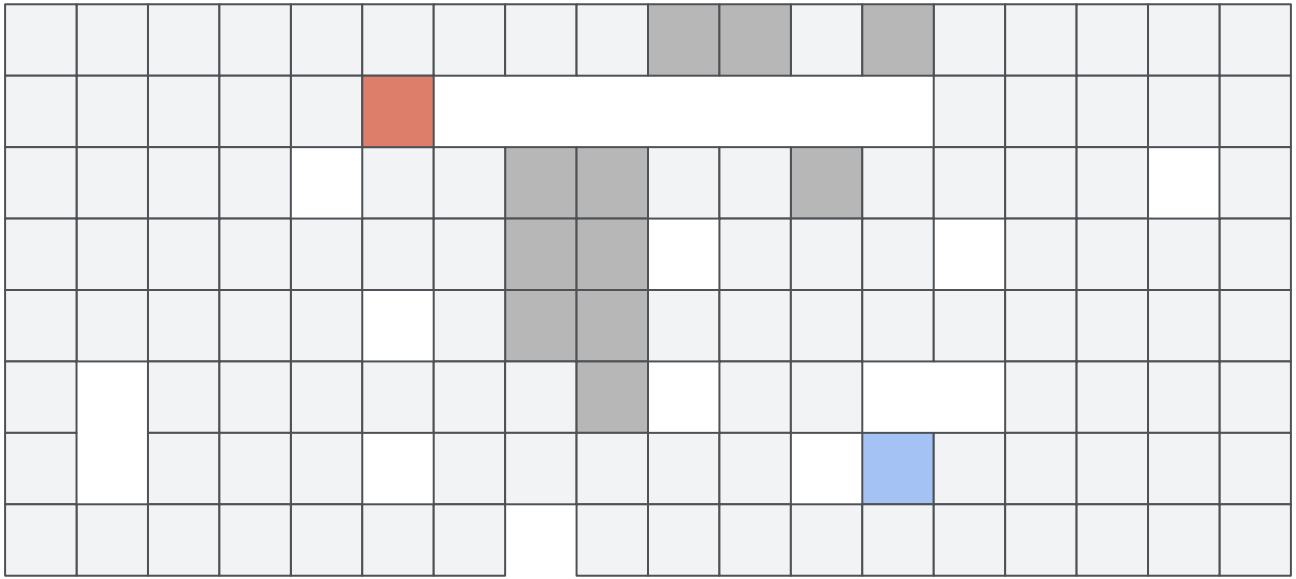


Walking codes in action



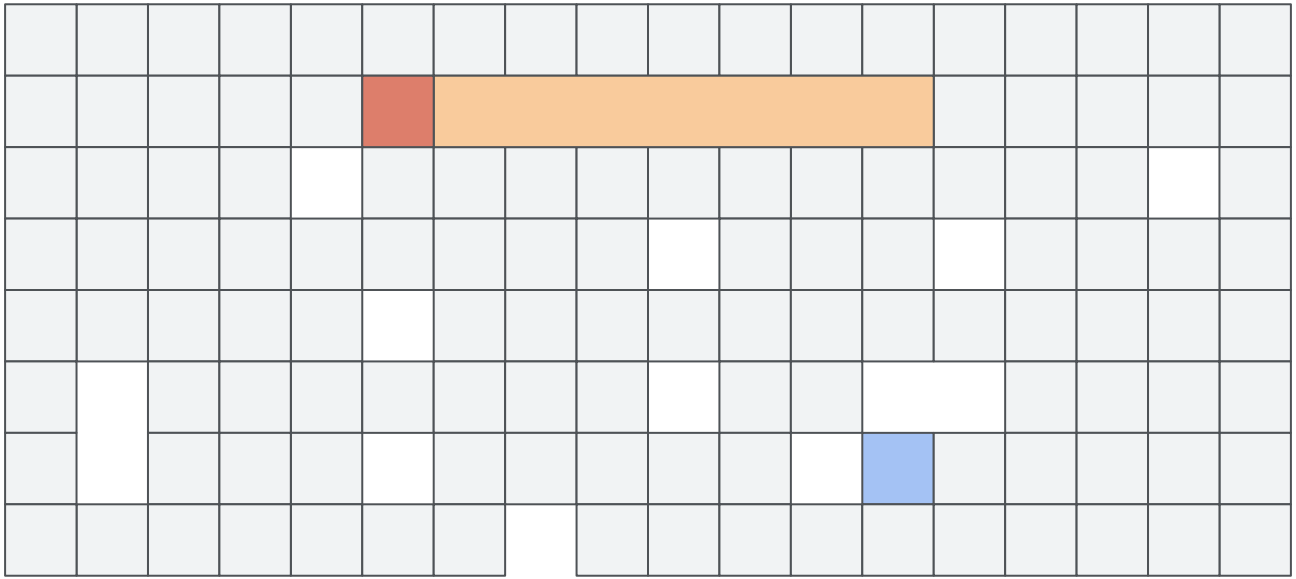
Time 0 → Time 1

Walking codes in action



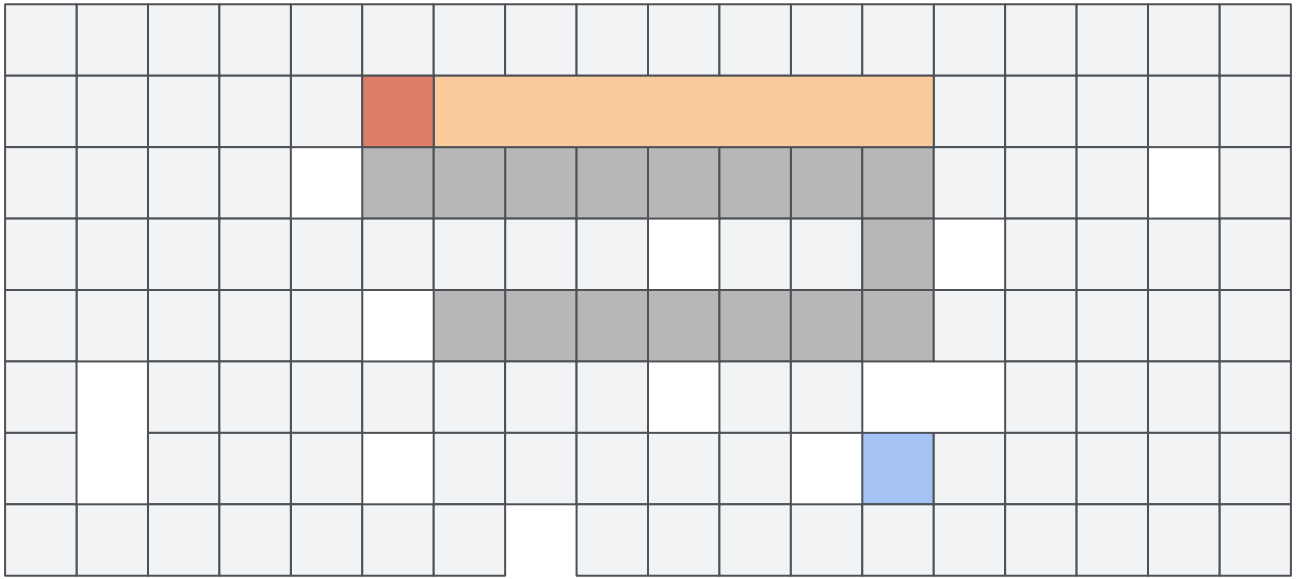
Time 0 → **Time 1**

Walking codes in action



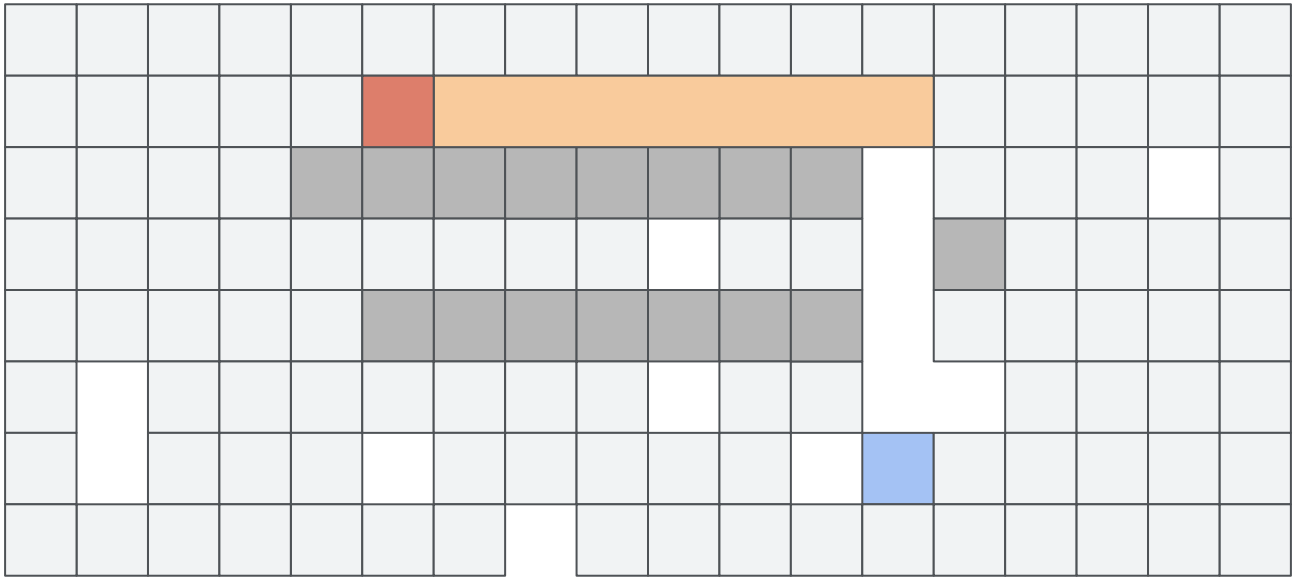
Time 1 → Time 2

Walking codes in action



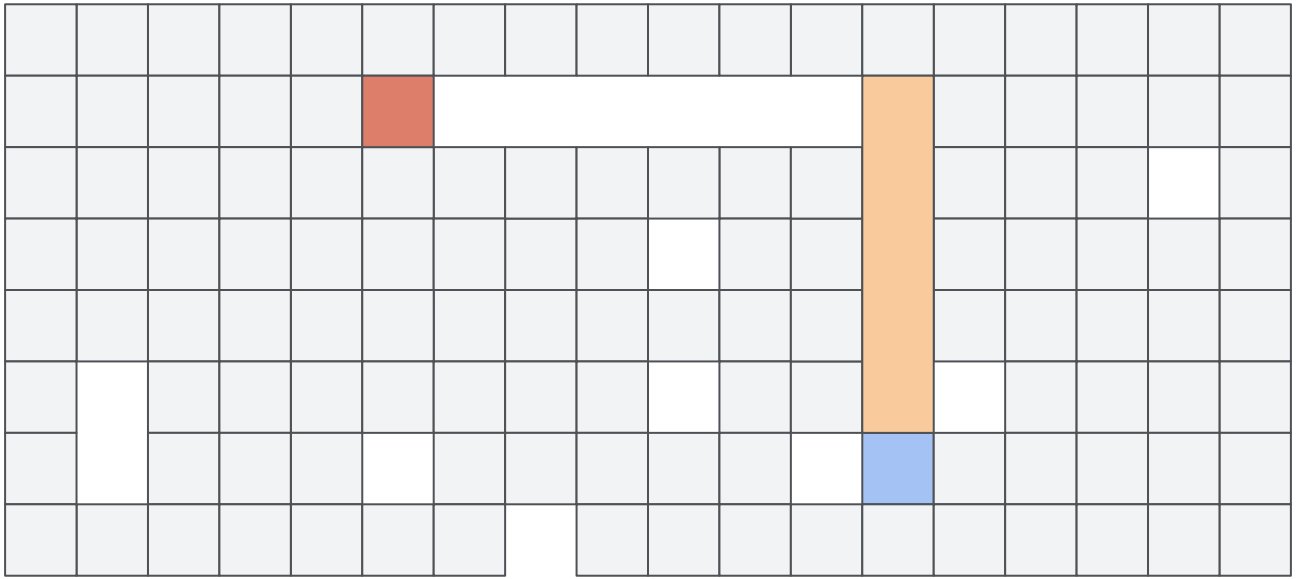
Time 1 → Time 2

Walking codes in action



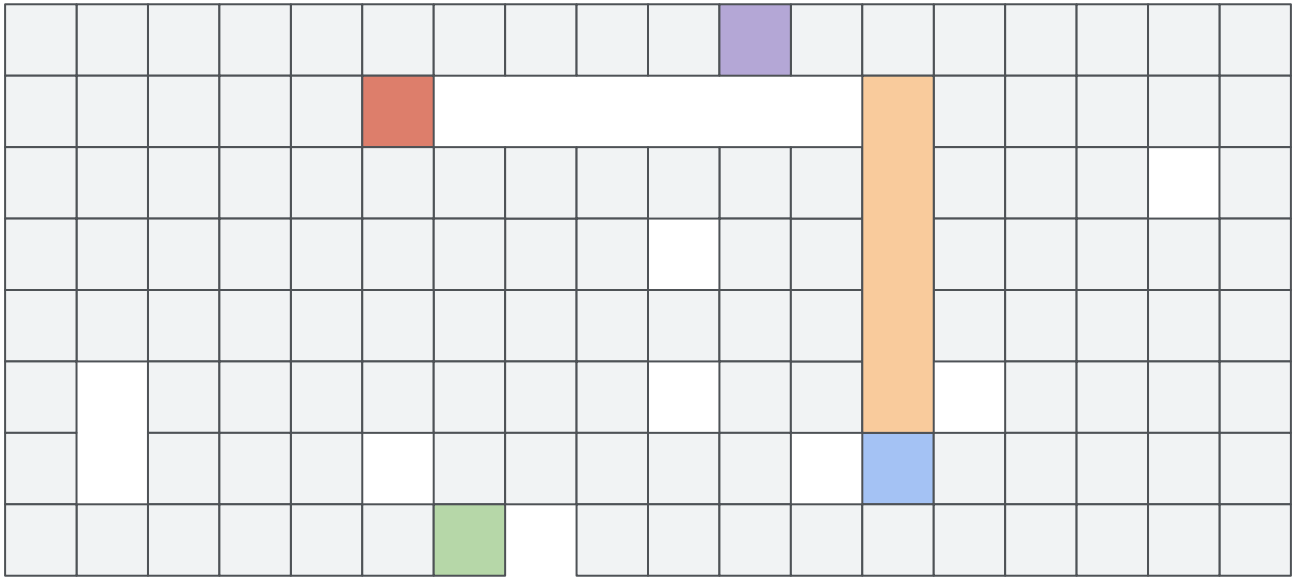
Time 1 → **Time 2**

Walking codes in action



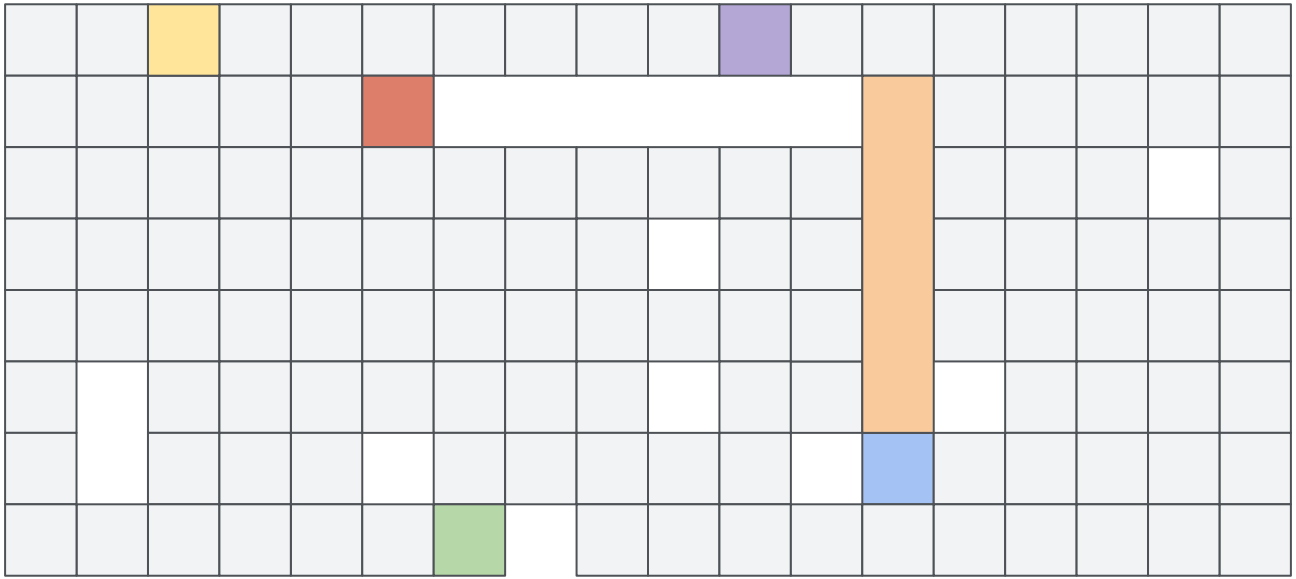
Time 2 → Time 3

Walking codes in action



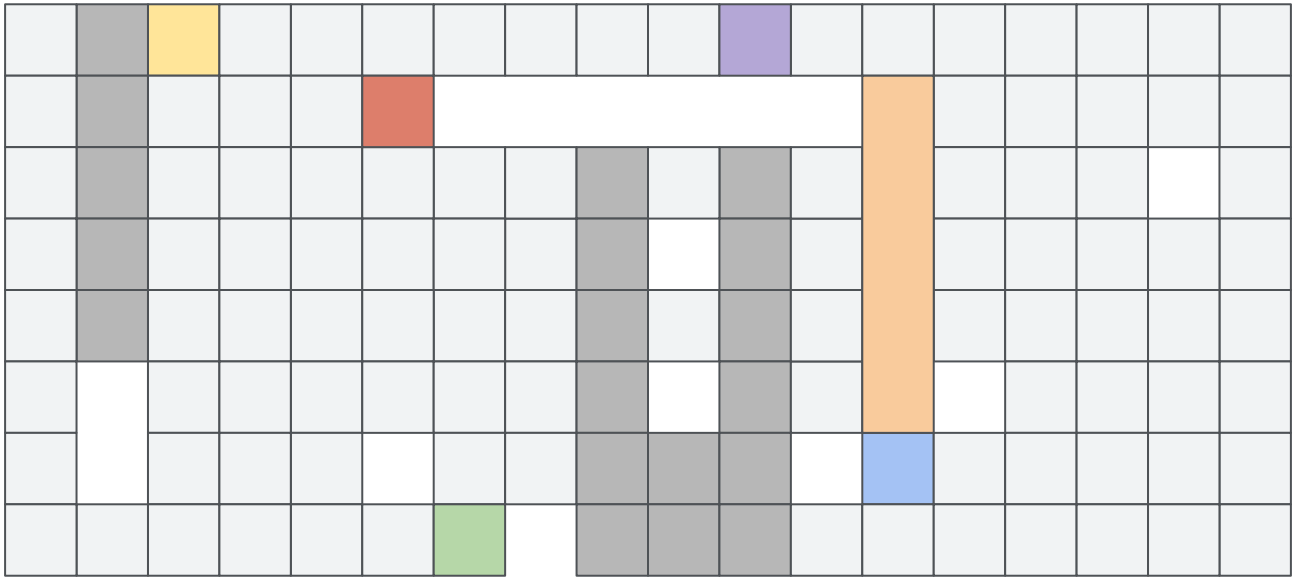
Time 2 → Time 3

Walking codes in action



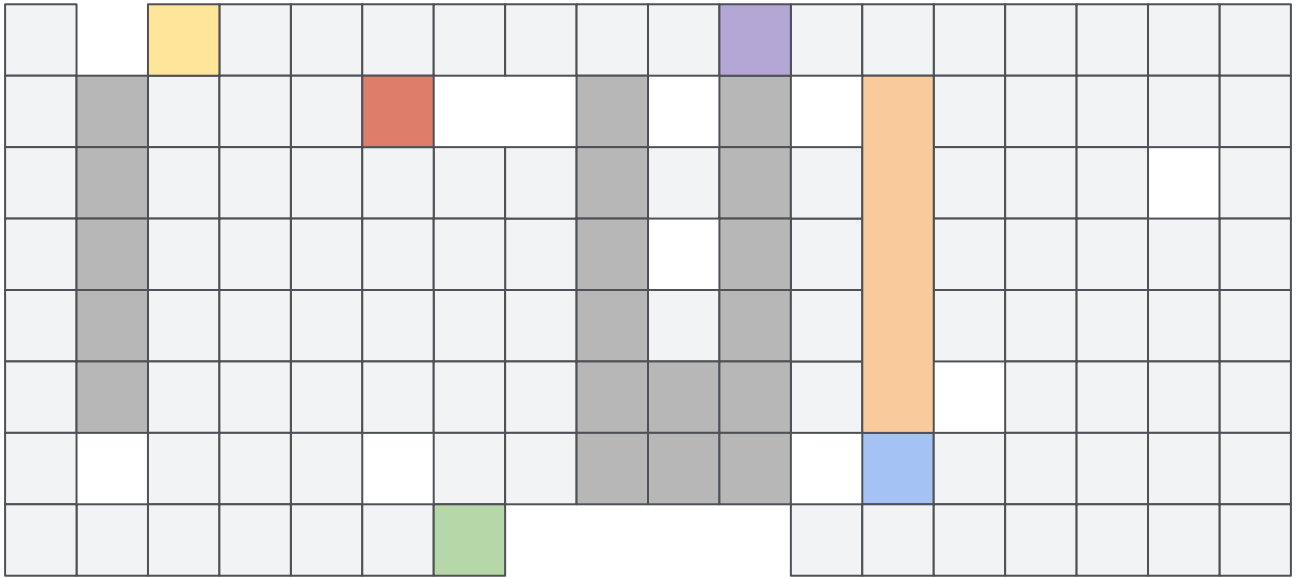
Time 2 → Time 3

Walking codes in action



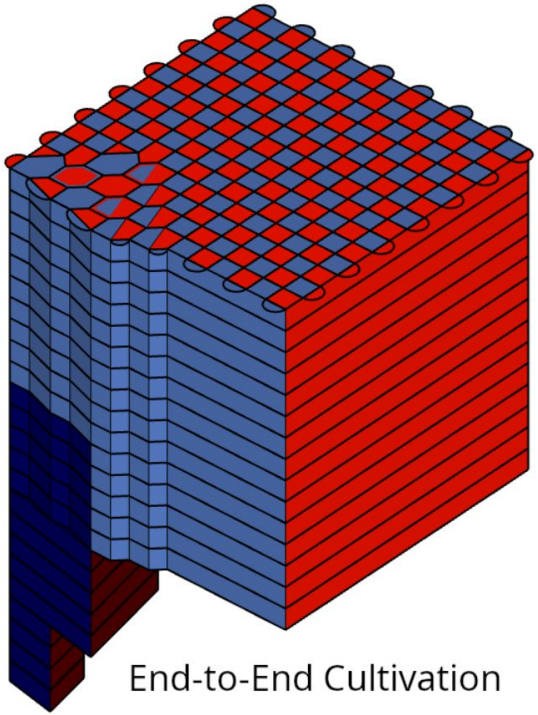
Time 2 → Time 3

Walking codes in action



Time 2 → **Time 3**

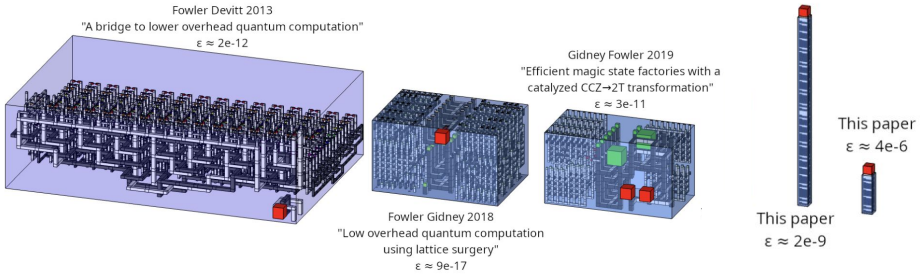
Magic state cultivation



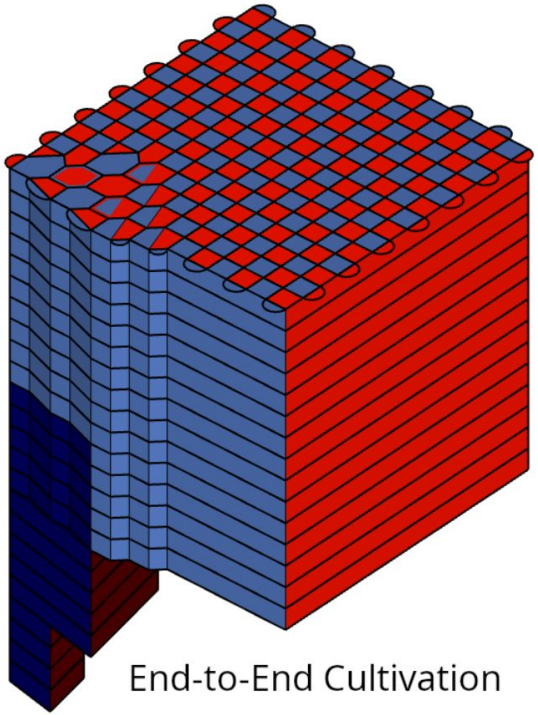
Magic state cultivation prepares a resource state we can use to perform a T gate.

The key ideas:

- Check the T state in a low-distance color code
- Rapidly grow to a larger code and postselect
- Low but finite error rate with much lower spacetime volume than previous techniques



Magic state cultivation



Magic state cultivation prepares a resource state we can use to perform a T gate.

The key ideas:

- Check the T state in a low-distance color code
- Rapidly grow to a larger code and postselect
- Low but finite error rate with much lower spacetime volume than previous techniques

Magic state cultivation: growing T states as cheap as CNOT gates

Craig Gidney, Noah Shutty, and Cody Jones

Google Quantum AI, California, USA

September 27, 2024

We refine ideas from [KLZ96; JBH16; CN20; Bom+24; GJ23; Gid+23; HIF24] to efficiently prepare good $|T\rangle$ states. We call our construction "magic state cultivation"

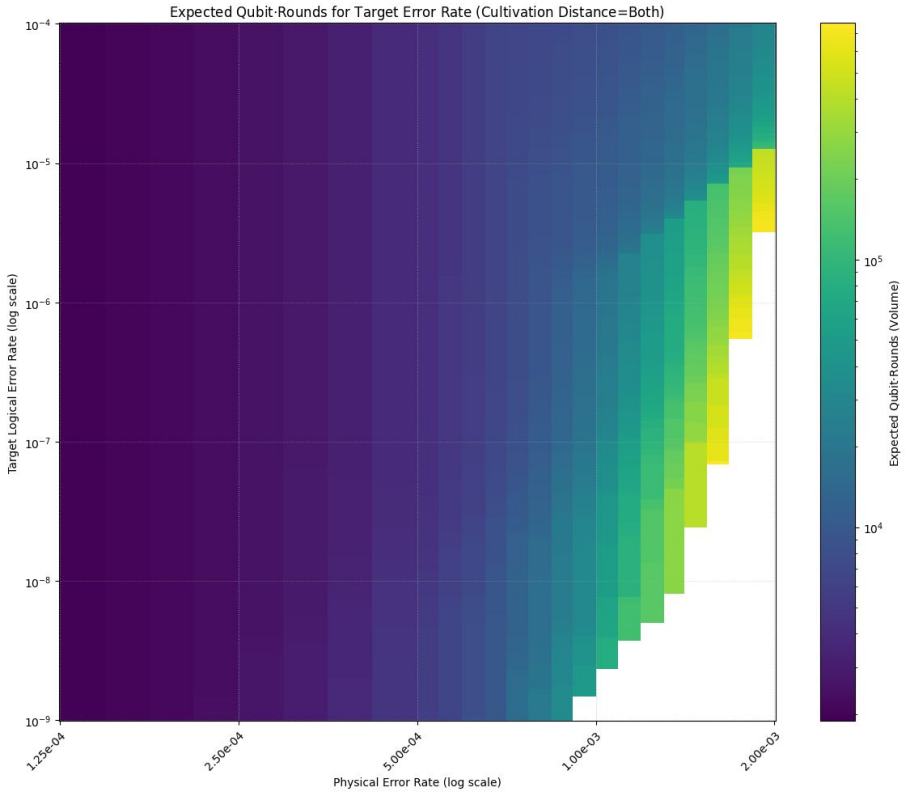
Magic state cultivation

We use simulations to determine the expected spacetime volume given a physical error rate and a target logical error rate

We optimize by varying:

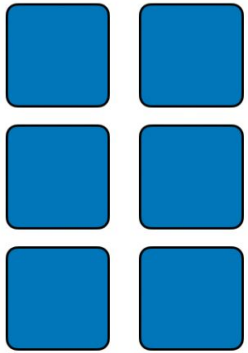
- The amount of postselection
- The choice of color code distance

(Performed using the code and methodology of Gidney *et al.*)



The model

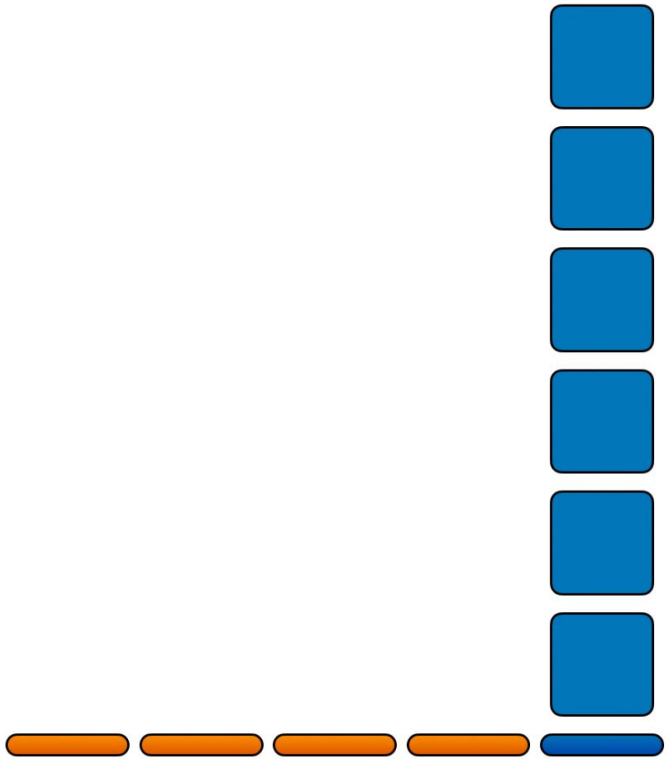
FLuid **A**llocation of **S**urface Code **Q**ubits



Each gate gets a cost that represents the extra spacetime volume it requires.

$$\text{COST} = \sum_g c_g$$

FLuid **A**llocation of **S**urface Code **Q**ubits



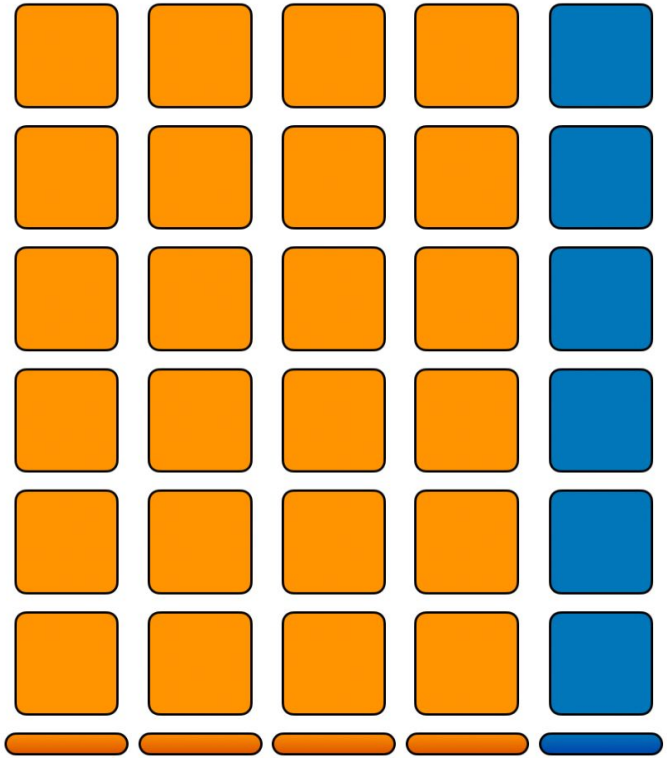
Each gate gets a cost that represents the extra spacetime volume it requires.

$$\text{COST} = \sum_g c_g$$

We make sure we have enough extra spacetime for all the gates.

$$\text{DEPTH} \geq \frac{\text{COST}}{\text{NUMBER OF FLUID ANCILLA}}$$

FLuid **A**llocation of **S**urface Code **Q**ubits



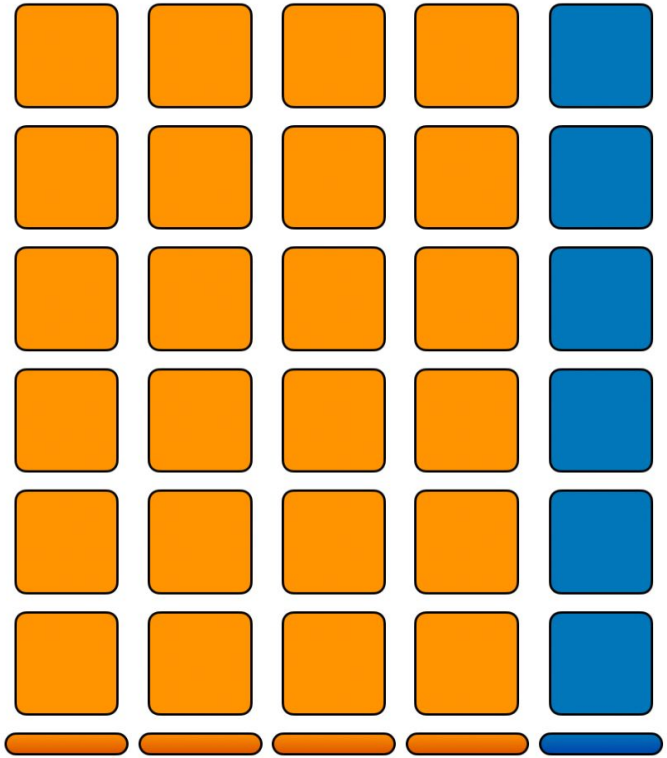
Each gate gets a cost that represents the extra spacetime volume it requires.

$$\text{COST} = \sum_g c_g$$

We make sure we have enough extra spacetime for all the gates.

$$\text{DEPTH} \geq \frac{\text{COST}}{\text{NUMBER OF FLUID ANCILLA}}$$

FLuid **A**llocation of **S**urface Code **Q**ubits



Each gate gets a cost that represents the extra spacetime volume it requires.

$$\text{Cost} = \sum_g c_g$$

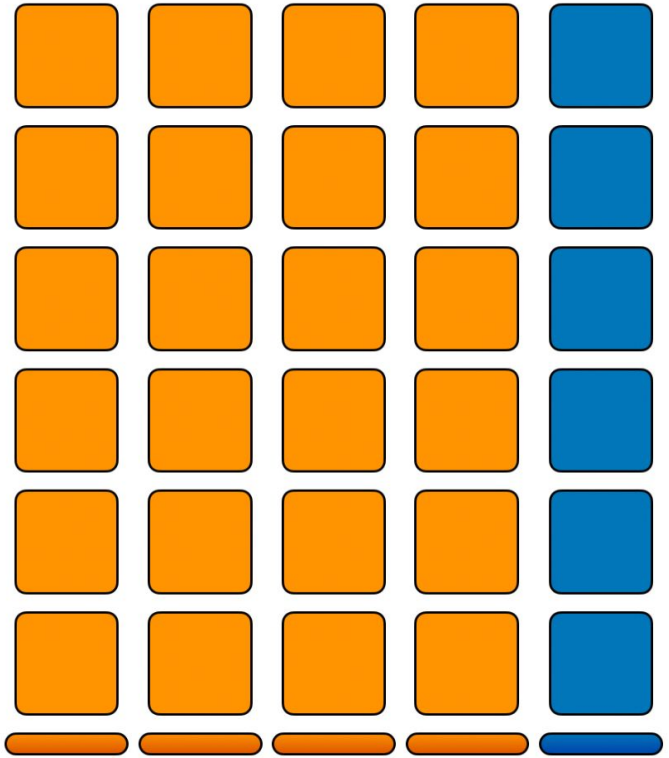
We make sure we have enough extra spacetime for all the gates.

$$\text{Depth} \geq \frac{\text{Cost}}{\text{Number of Fluid Ancilla}}$$

(We also make sure the depth is large enough to account for measurements that have to happen sequentially.)

$$\text{Depth} \geq \text{Reaction Time} \cdot \text{Measurement Depth}$$

FLuid **A**llocation of **S**urface Code **Q**ubits



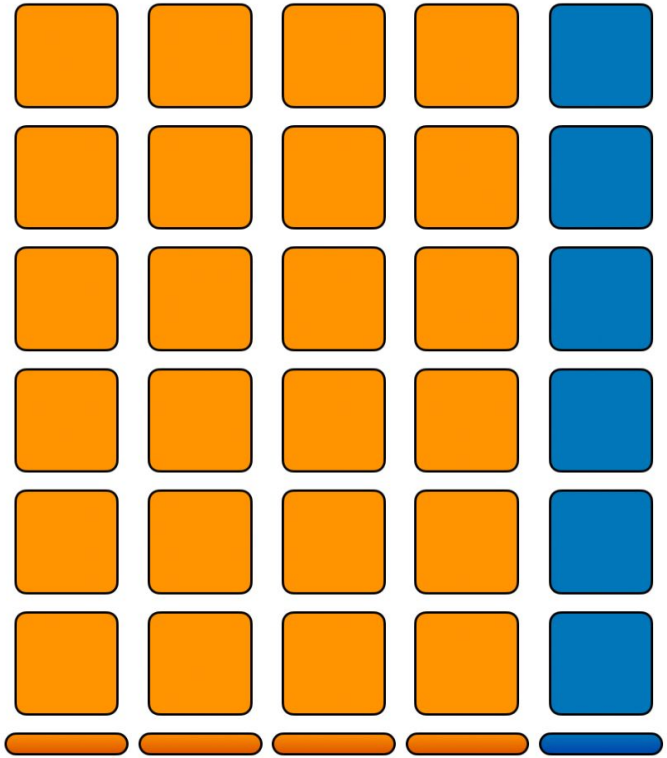
Each gate gets a cost that represents the extra spacetime volume it requires.

$$\text{COST} = \sum_g c_g$$

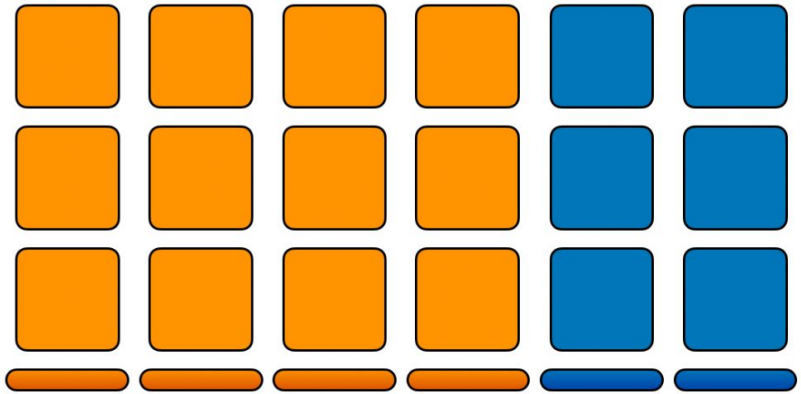
We make sure we have enough extra spacetime for all the gates.

$$\text{DEPTH} \geq \frac{\text{COST}}{\text{NUMBER OF FLUID ANCILLA}}$$

FLuid **A**llocation of **S**urface Code **Q**ubits



The depth shrinks as we increase the number of fluid ancilla qubits



FLuid Allocation of Surface Code Qubits

Each gate gets a cost that represents the extra spacetime volume it requires.

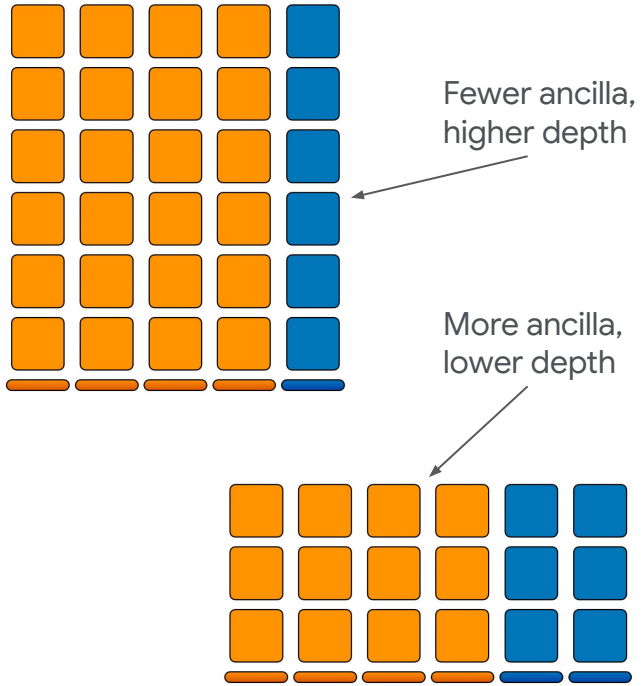
$$\textsc{Cost} = \sum_g c_g$$

We make sure we have enough extra spacetime for all the gates.

$$\textsc{Depth} \geq \frac{\textsc{Cost}}{\textsc{Number of Fluid Ancilla}}$$

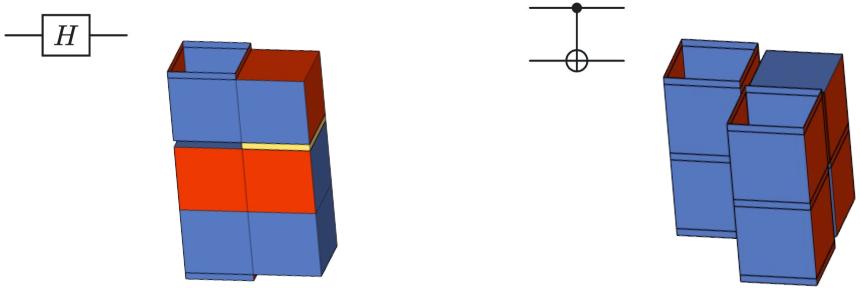
(We also make sure the depth is large enough to account for measurements that have to happen sequentially.)

$$\textsc{Depth} \geq \textsc{Reaction Time} \cdot \textsc{Measurement Depth}$$



Gate costs in the FLASQ model

Basic gates	Conservative cost	Optimistic cost	Notes
X / Y / Z	0	—	Implemented in software
X / Z basis measurement (or initialization)	0	—	
H	3	1.5	Patch rotation each time (patch rotation half the time)
S / $S^\dagger$	1.5	1	Gate performed statically (performed in motion half the time)
T / $T^\dagger$	$1.5v(p_{phys}, p_{cult}) + 2$	$v(p_{phys}, p_{cult}) + 2$	Buffer to account for packing constraints (no buffer) Depends on physical error rate (p_{phys}) and target logical error rate (p_{cult})
$CNOT$ / CZ	$2p(q_1, q_2) + 3$	$p(q_1, q_2)$	Buffer to account for extra routing (no buffer)



Cost in units of d^3, represents volume required above and beyond memory.

$p(q_1, ..., q_k)$ denotes the minimum length (measured in the manhattan distance) required to connect the qubits.

The volume required to cultivate a magic state depends on the underlying physical error rate p_{phys} and the target logical error rate p_{cult}.

Gate costs in the FLASQ model

Basic gates	Conservative cost	Optimistic cost	Notes
X / Y / Z	0	—	Implemented in software
X / Z basis measurement (or initialization)	0	—	
H	3	1.5	Patch rotation each time (patch rotation half the time)
S / $S^\dagger$	1.5	1	Gate performed statically (performed in motion half the time)
T / $T^\dagger$	$1.5v(p_{phys}, p_{cult}) + 2$	$v(p_{phys}, p_{cult}) + 2$	Buffer to account for packing constraints (no buffer) Depends on physical error rate (p_{phys}) and target logical error rate (p_{cult})
$CNOT$ / CZ	$2p(q_1, q_2) + 3$	$p(q_1, q_2)$	Buffer to account for extra routing (no buffer)
Additional primitive gates	**Cost**		**Notes**
$T_X = HTH$ / $T_X^\dagger$	$\textsc{Cost}(T)$	—	Can be implemented directly using cultivation
$R_Z(\theta)$	$\approx (4.86 + .53 \log_2(\epsilon_{rot}))\,\textsc{Cost}(T)$ $+2\,\textsc{Cost}(S) + 2\,\textsc{Cost}(H)$	$\approx (4.86 + .53 \log_2(\epsilon_{rot}))\,\textsc{Cost}(T)$ $+\frac{2}{6}\,\textsc{Cost}(S) + \frac{2}{3}\,\textsc{Cost}(H)$	Depends on target rotation synthesis error (ϵ_{rot}, in the diamond norm)
Move	$p(q_1, q_2)$	—	Swaps the state of two qubits, one of which must be in the $\|0\rangle$ state
Toffoli	$4\,\textsc{Cost}(T) + 2p(q_1, q_2, q_3) + 10$	$4\,\textsc{Cost}(T) + p(q_1, q_2, q_3) + 10$	Buffer to account for extra routing (no buffer)
k-target $CNOT$	$2p(q_0, \cdots, q_k) + 3$	$p(q_0, \cdots, q_k)$	Buffer to account for extra routing (no buffer)
k-parity-controlled X gate	$2p(q_0, \cdots, q_k) + 3$	$p(q_0, \cdots, q_k)$	Buffer to account for extra routing (no buffer) Controlled based on parity of multiple control qubits
Y basis measurement (or initialization)	.5	—	

Cost in units of d^3, represents volume required above and beyond memory.

$p(q_1, \ldots, q_k)$ denotes the minimum length (measured in the manhattan distance) required to connect the qubits.

The volume required to cultivate a magic state depends on the underlying physical error rate p_{phys} and the target logical error rate p_{cult}.

Applications

Error mitigation overhead with probabilistic error cancellation

$$\text{NUMBER OF SAMPLES} \propto \Gamma^2 \frac{\|O\|}{\epsilon^2}$$

$$p_{cyc} = .1\Lambda^{(d+1)/2}, \qquad \Lambda = \frac{.01}{p_{phys}}$$

$$V = N_{active} \cdot \text{DEPTH} + \text{COST}$$

$$\Gamma^2 \approx (1 - p_{cyc})^{-4dV}$$

We can reduce the resource requirements by combining error correction and error mitigation.

Here we use probabilistic error cancellation and a simple noise model.

Our FLASQ model lets us estimate how many opportunities we have for errors.

We can convert this into an estimate of the error mitigation overhead

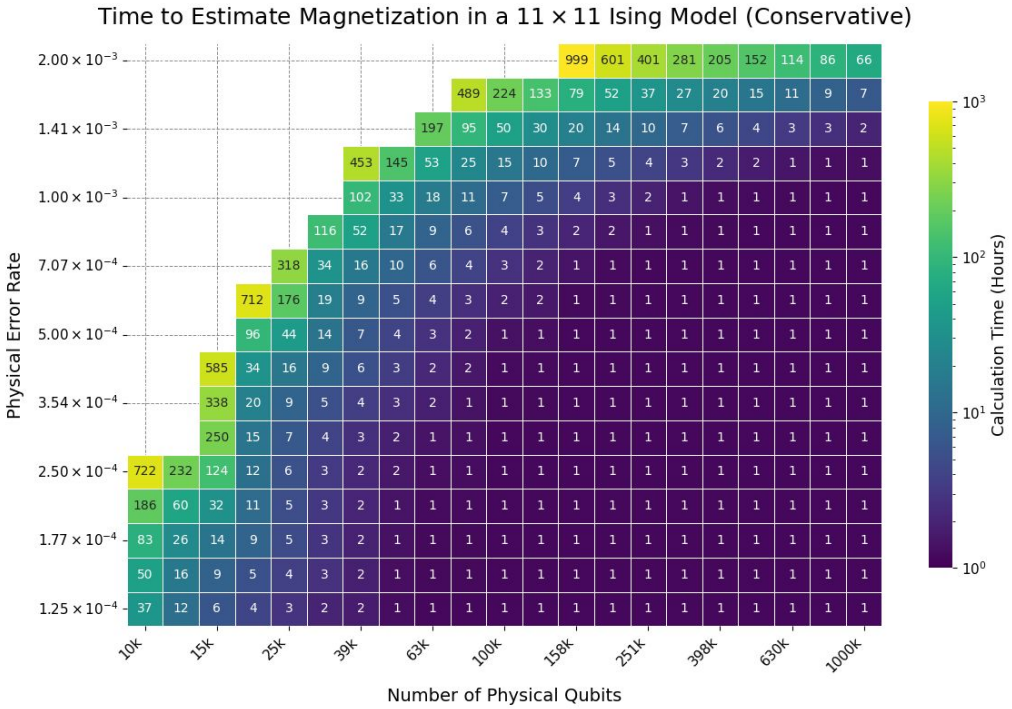
Resource estimates for lattice models

- 2D Ising model

$$H = J \sum_{\langle ij \rangle} Z_i Z_j + h \sum_i X_i$$

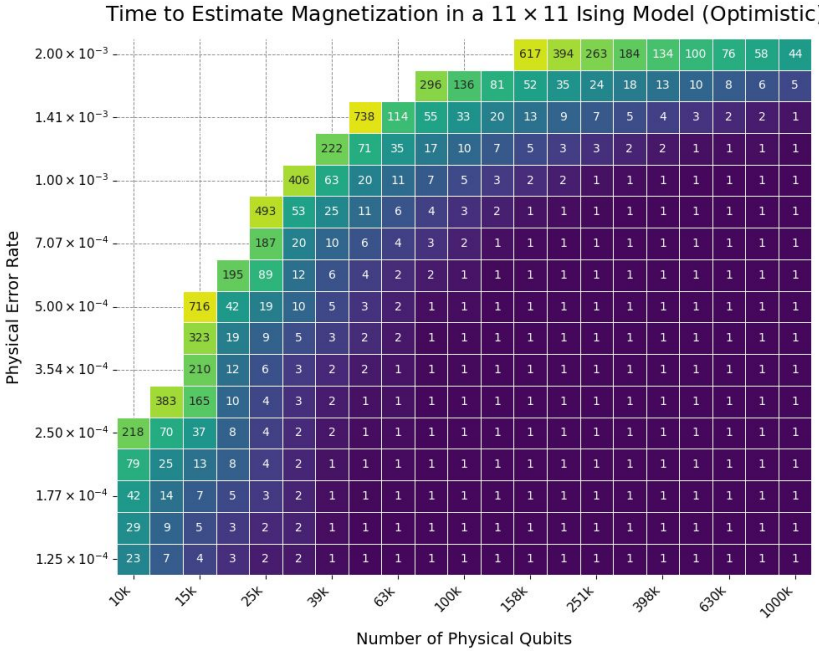
  - Time-evolve after a quench and measure magnetization

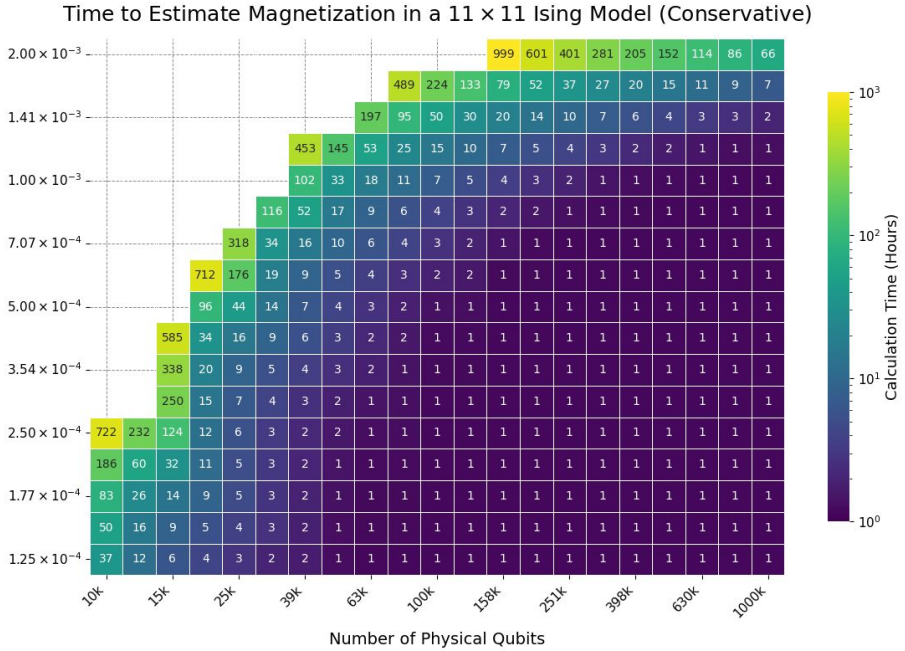
- Probing difficult regime
  - Near phase transition
  - T = 1, dt=.05, 2nd-order Trotter
  - .002 standard deviation (comparable with disagreement between classical methods)[1]



[1] Begušić, T. & Chan, G. K.-L. Real-time operator evolution in two and three dimensions via sparse Pauli dynamics. *PRX quantum* **6**, (2025).

Resource estimates for lattice models





Synthesizing rotations is expensive

Even the best methods for synthesizing a single-qubit Z rotation require many T gates[2]:

$$\mathsf{T}\left(R_Z\right) = 4.86 + 0.53 \log_2 \epsilon^{-1}$$

This is the dominant cost for many applications.

Can we reduce it?

Hamming weight phasing lets us implement K identical R_Z rotations in parallel using only $\log_2(K)$ rotations and ≈ 4K additional T gates.

It works by calculating the Hamming weight into an ancilla register and applying rotations to each of the ancilla.

Standard compilations use additional ancilla to minimize the number of T gates.

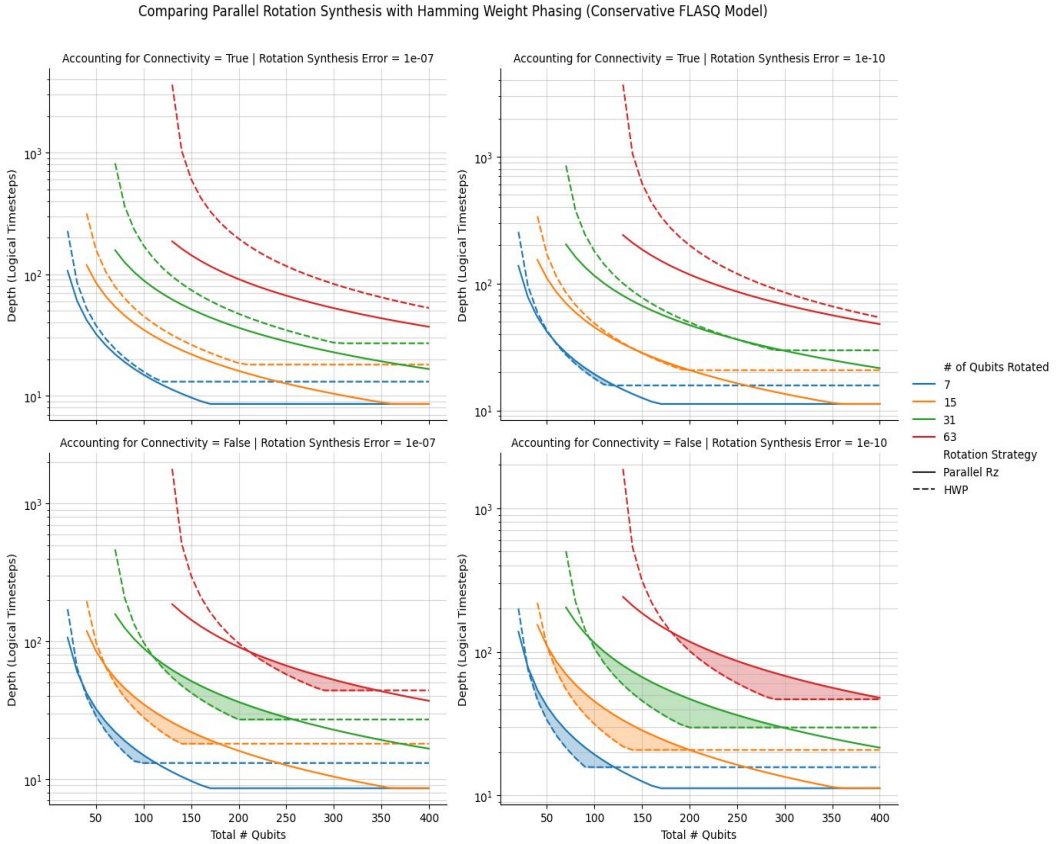
[2] Kliuchnikov, V., Lauter, K., Minko, R., Paetznick, A. & Petit, C. Shorter quantum circuits via single-qubit gate approximation. *arXiv:2203.10064* (2022).

Lowering the cost with Hamming weight phasing?

We compare parallel rotations synthesis with Hamming weight phasing.

- Top: We lay the circuits out on a 2D grid using a heuristic.
- Bottom: We neglect the cost of long-range interactions.

We shade the region where Hamming weight phasing yields an improvement in the circuit depth.



Thank you

Magic state cultivation simulations (expected number of retries)

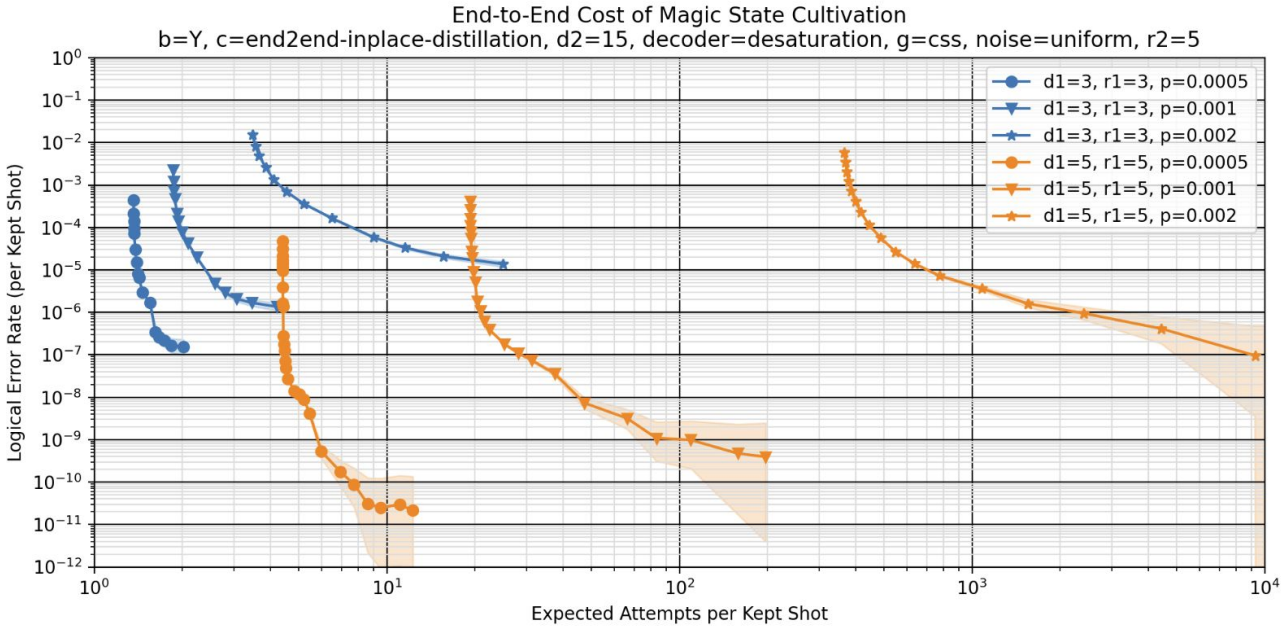
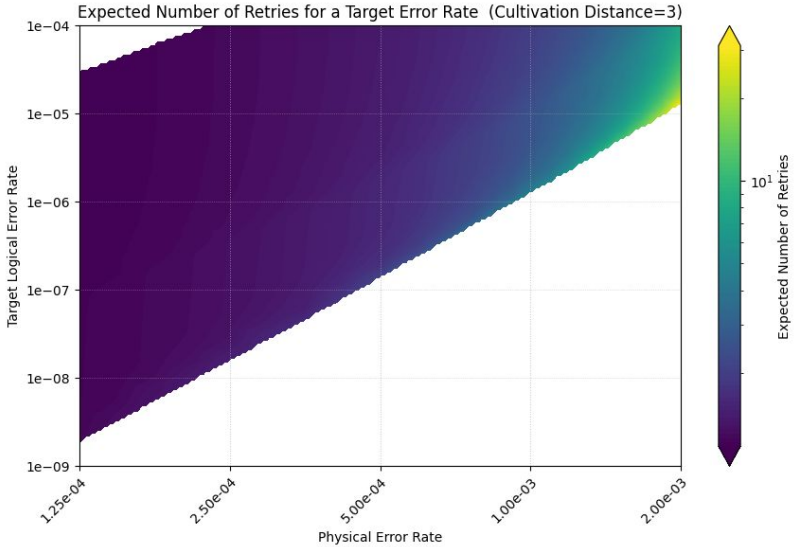
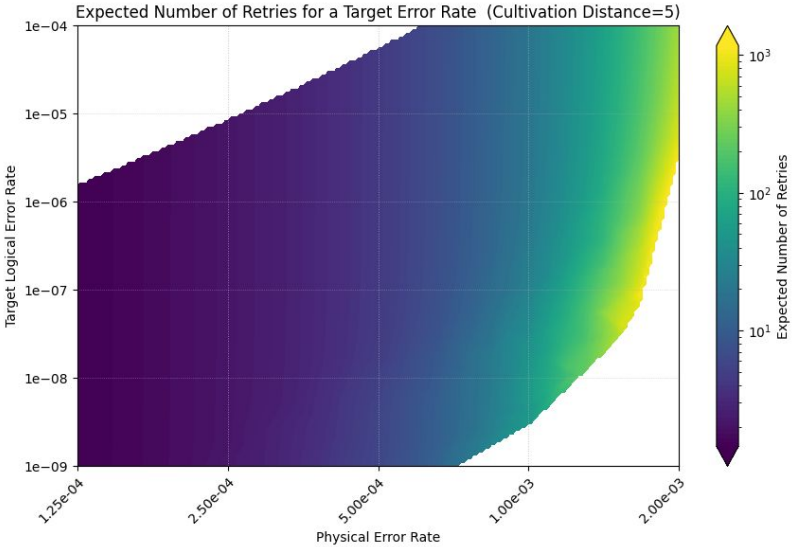


Figure 14: **End to end simulation of magic state cultivation.** Curves show the effect of varying the complementary gap cutoff for rejecting shots, decreasing logical error rate at the cost of increasing expected attempts. Click here to open an example end-to-end circuit in Crumble. Shaded regions indicate error rate hypotheses with a likelihood within a factor of 1000 of the max likelihood hypothesis, given the sampled data.

Magic state cultivation simulations (expected number of retries)





- 21 different error rates
- 5 billion distance 3
- 50 billion distance 5

- Same methodology as cultivation paper
- Trimmed data with high uncertainty

Magic state cultivation simulations (expected spacetime volume)

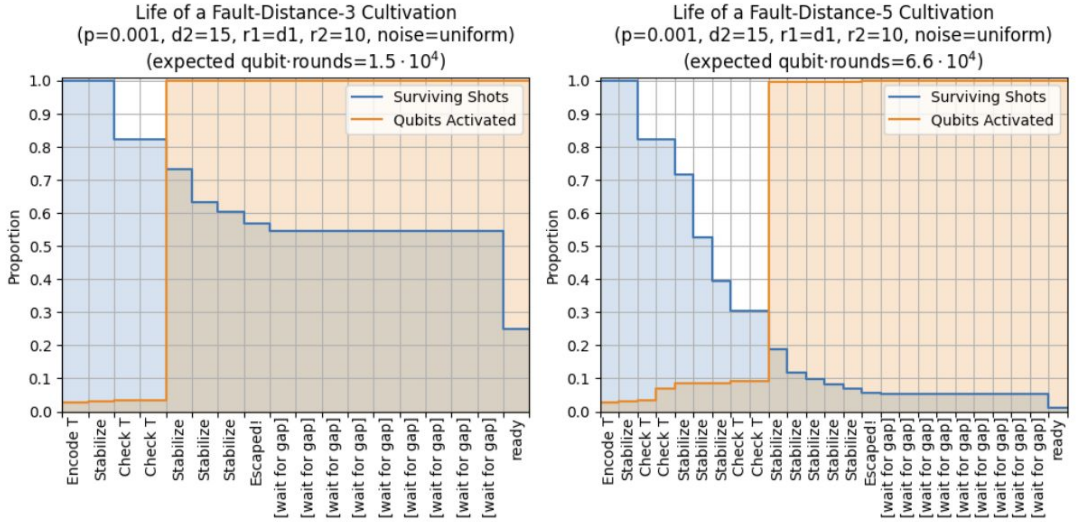


Figure 15: **Growth and survival during magic state cultivation**. Each column is a labelled circuit cycle, which performs roughly ten layers of one/two qubit gates and ends with one or two layers of dissipative gates. The proportion of activated qubits starts small during the injection stage, gradually increases during the cultivation stage, and then jumps for the escape stage. Survival rates drop continuously due to postselected detectors failing, then hold steady for 10 cycles while waiting for the complementary gap to be computed. Expected qubit · rounds is computed by integrating the product of the survival rate times the qubit proportion, then multiplying by the total number of qubits and dividing by the final survival rate. This is slightly optimistic, due to ignoring factors like packing efficiency.