Quantum AI

The **FL**uid **A**llocation of **S**urface code **Q**ubits model for early fault-tolerant resource estimation

William J. Huggins, Tanuj Khattar, Amanda Xu, Matthew Harrigan, Christopher Kang, Guang Hao Low, Dmitri Maslov, Austin Fowler, Nicholas C. Rubin, Ryan Babbush

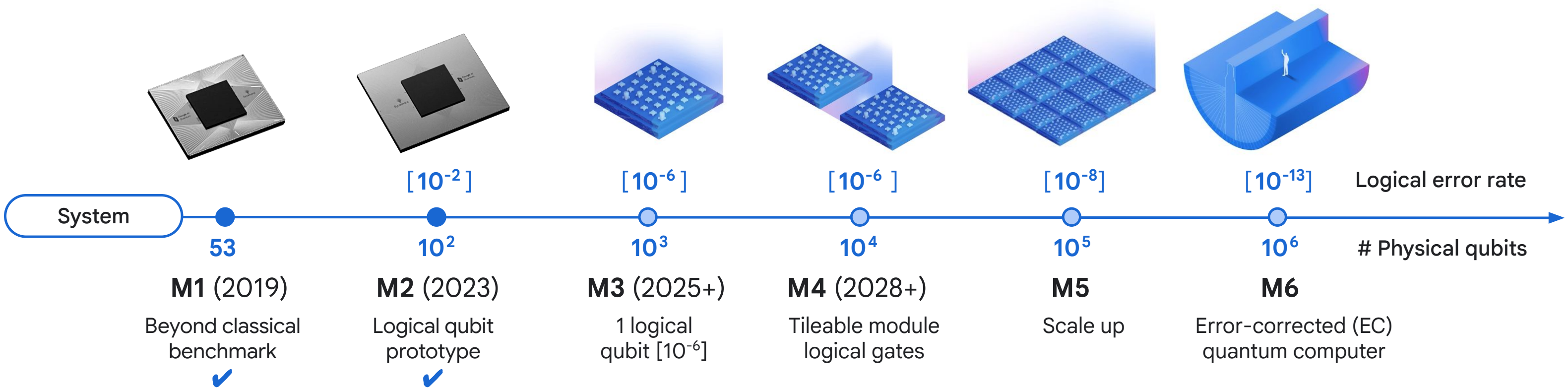
IPAM, Bridging the Gap Between NISQ and FTQC, February 2026

Google

What problem are we trying to solve?

μ = 1/π
Δ| = T(x) + e
F = m√x

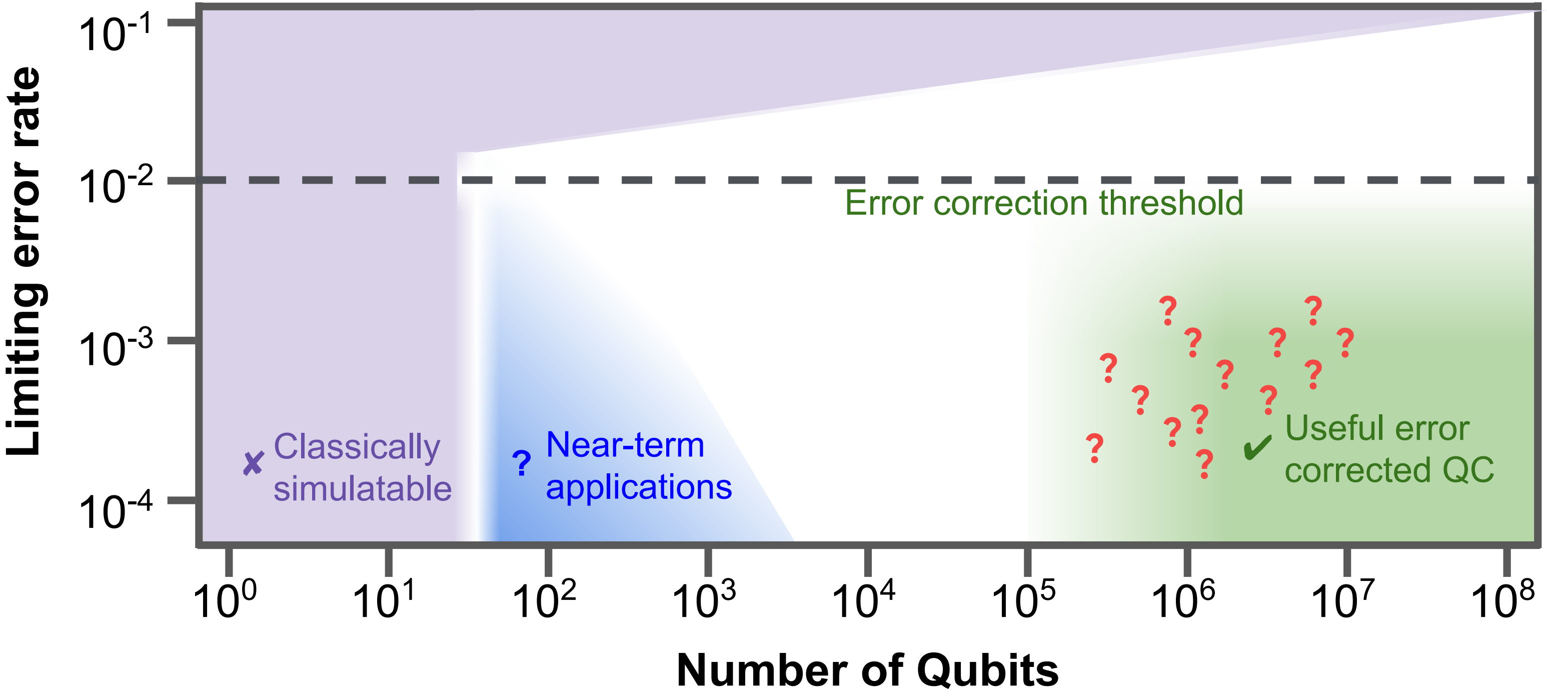
Understanding the transition to fault-tolerance



System	M1 (2019)	M2 (2023)	M3 (2025+)	M4 (2028+)	M5	M6
Logical error rate		$[10^{-2}]$	$[10^{-6}]$	$[10^{-6}]$	$[10^{-8}]$	$[10^{-13}]$
# Physical qubits	53	10^2	10^3	10^4	10^5	10^6
	Beyond classical benchmark	Logical qubit prototype	1 logical qubit $[10^{-6}]$	Tileable module logical gates	Scale up	Error-corrected (EC) quantum computer
	✓	✓				

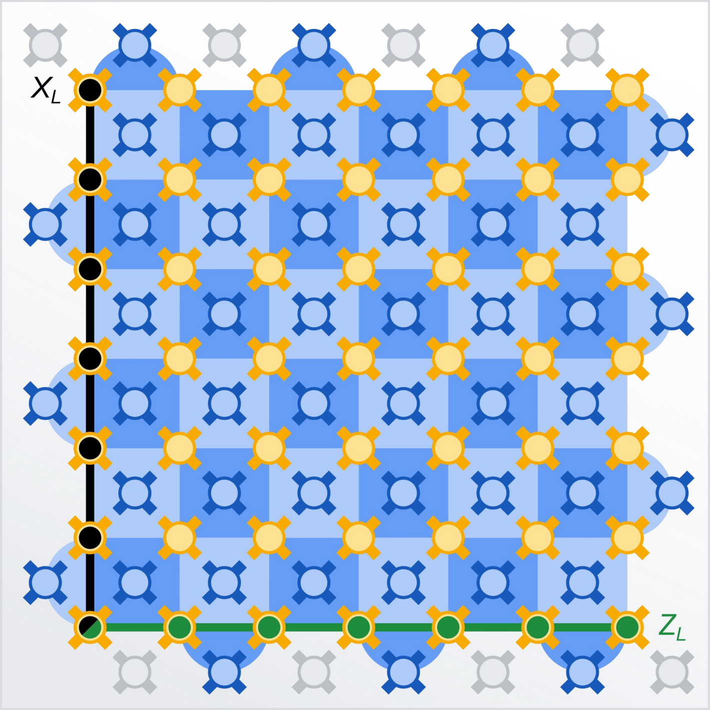
Quantum AI

Understanding the transition to fault-tolerance



Ingredients

Logical qubits in the surface code



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

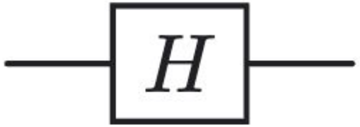
Logical Error Rate $\approx 0.1 * \Lambda^{-(d+1)/2}$

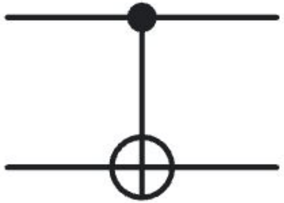
Surface code error correction gives exponentially more time for quadratically more space.

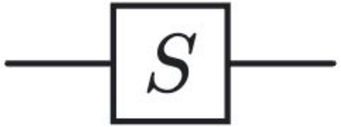
2D local connectivity is sufficient to store information and perform gates using lattice surgery

Everything* requires ancilla spacetime

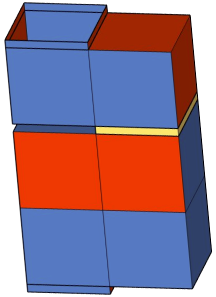
Gates in NISQ

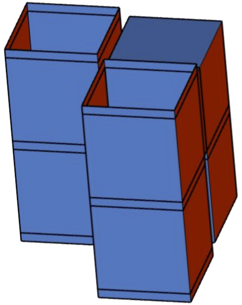


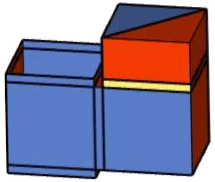




Gates in the surface code







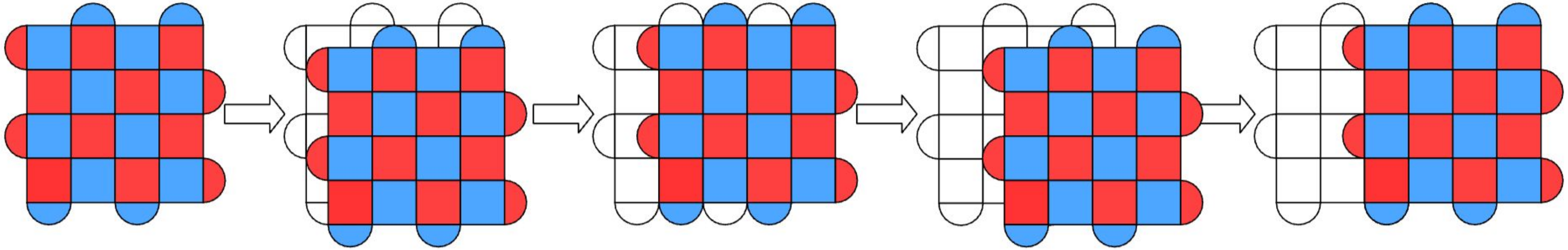
Walking surface codes

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

We will want to move ancilla around

Walking surface codes let us move logical qubits

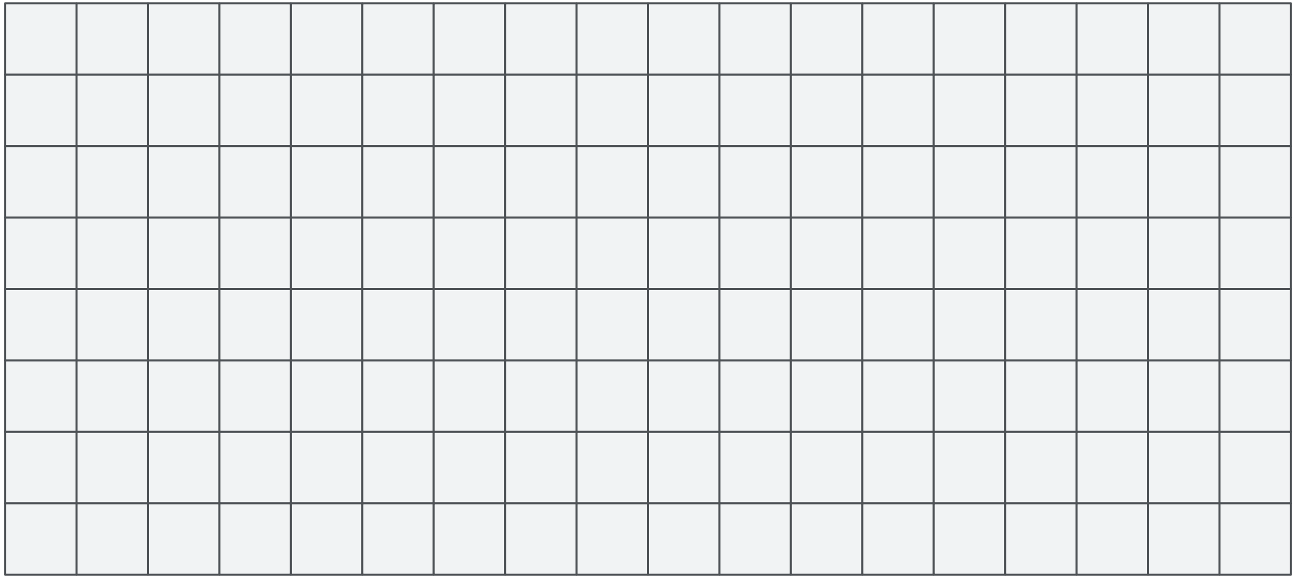
Whole patches can move together



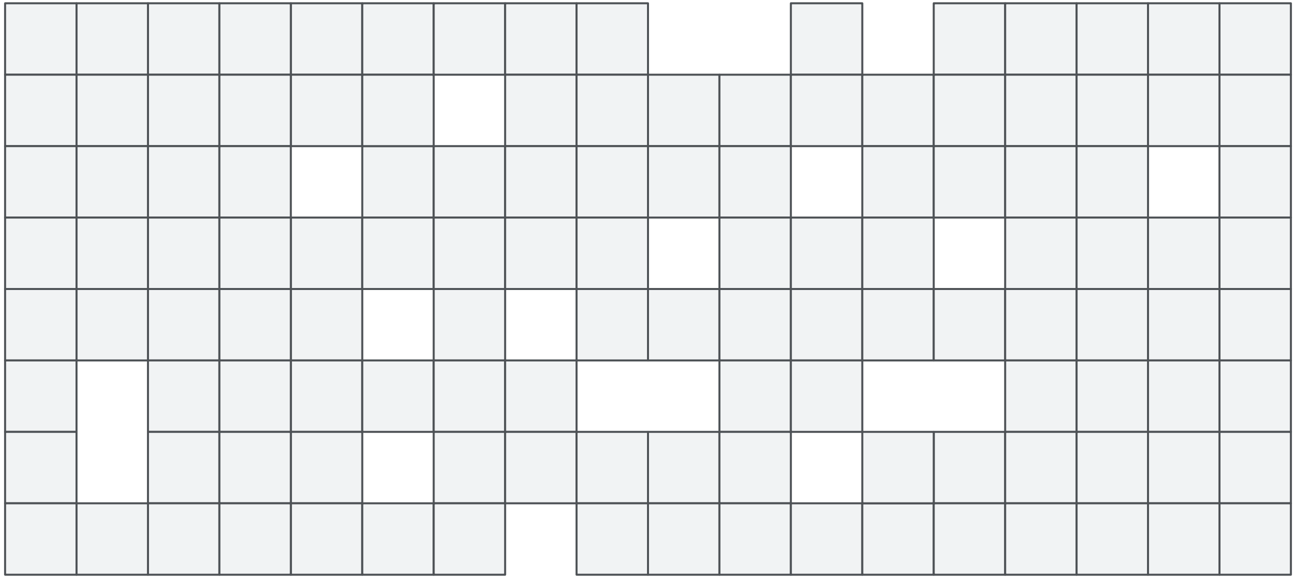
Quantum AI

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

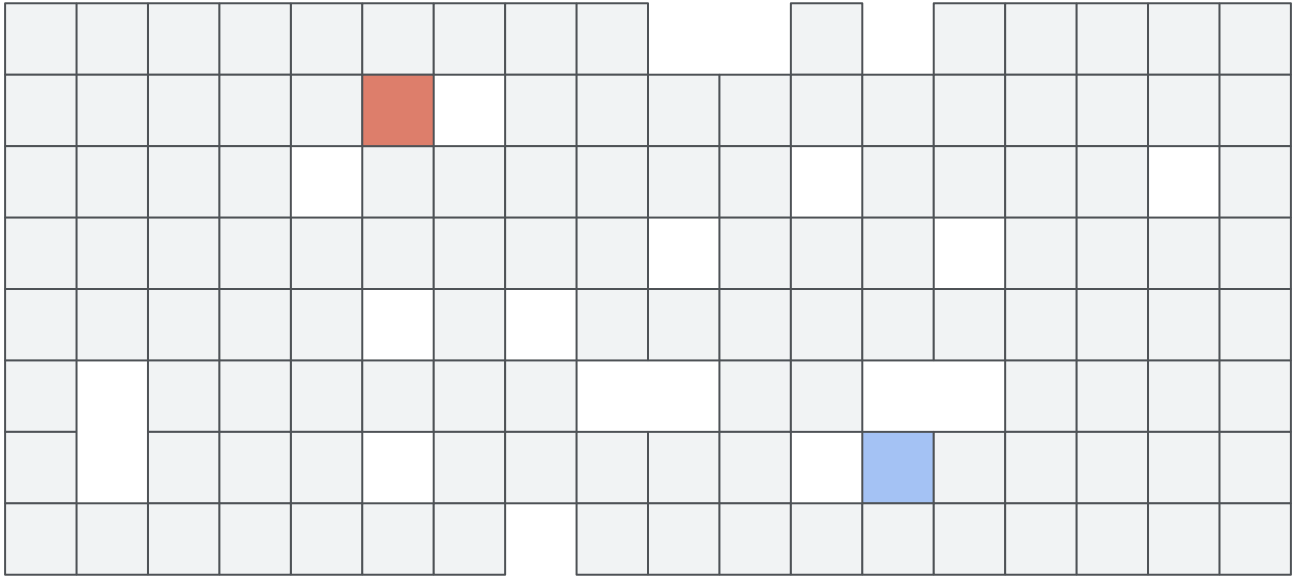
Walking surface codes in action



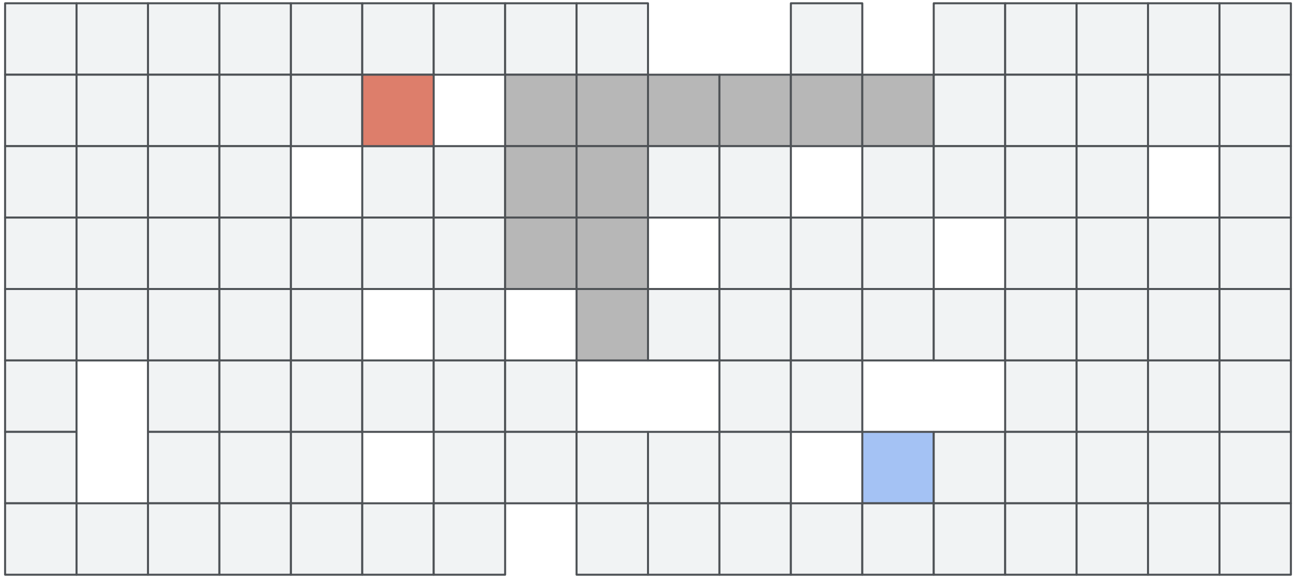
Walking surface codes in action



Walking surface codes in action

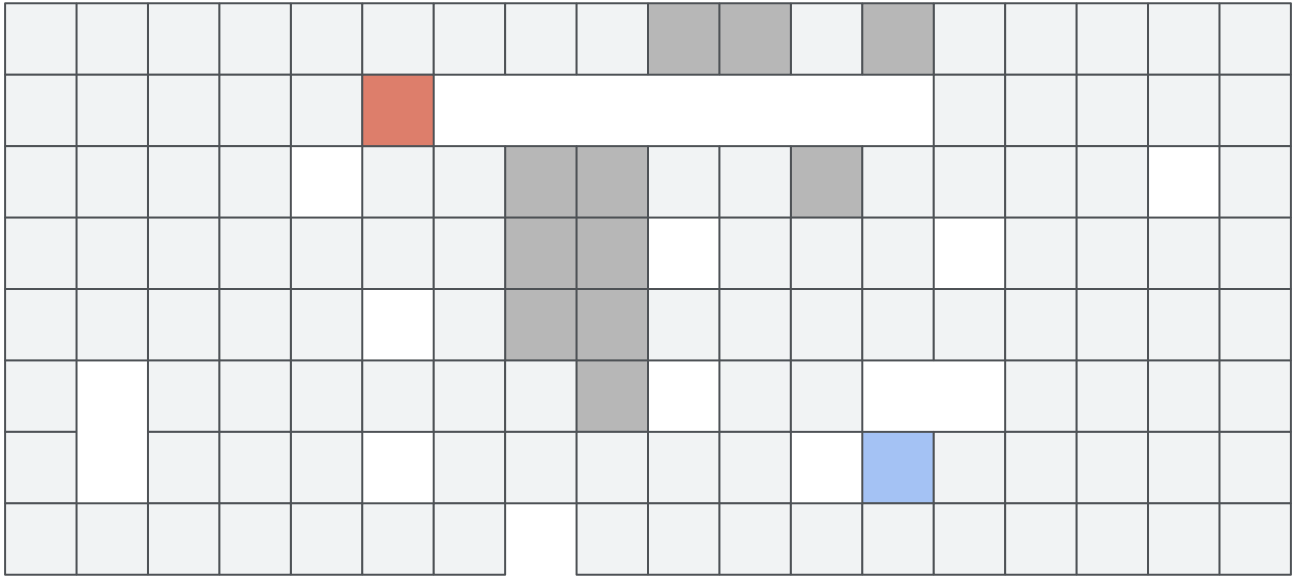


Walking surface codes in action



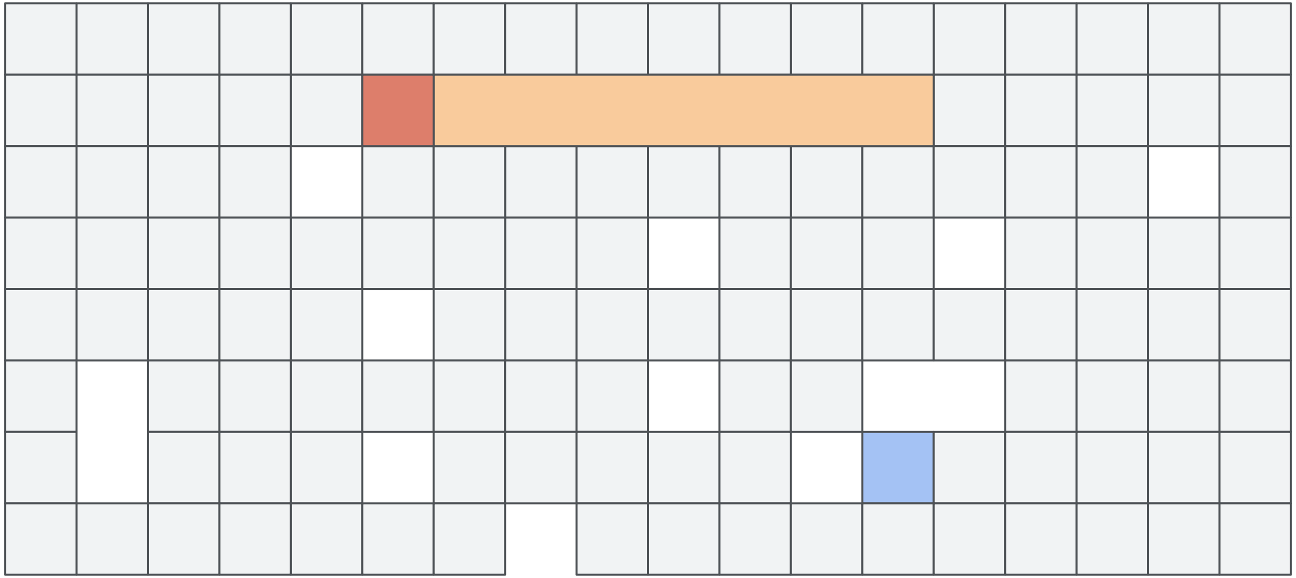
Time 0 → Time 1

Walking surface codes in action



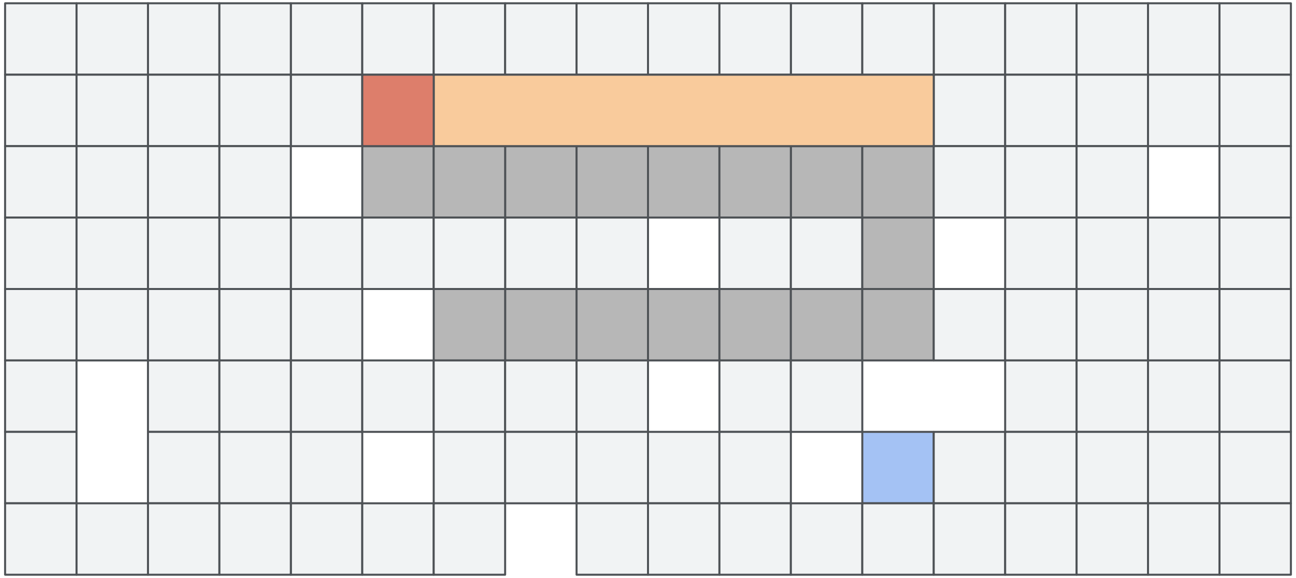
Time 0 → **Time 1**

Walking surface codes in action



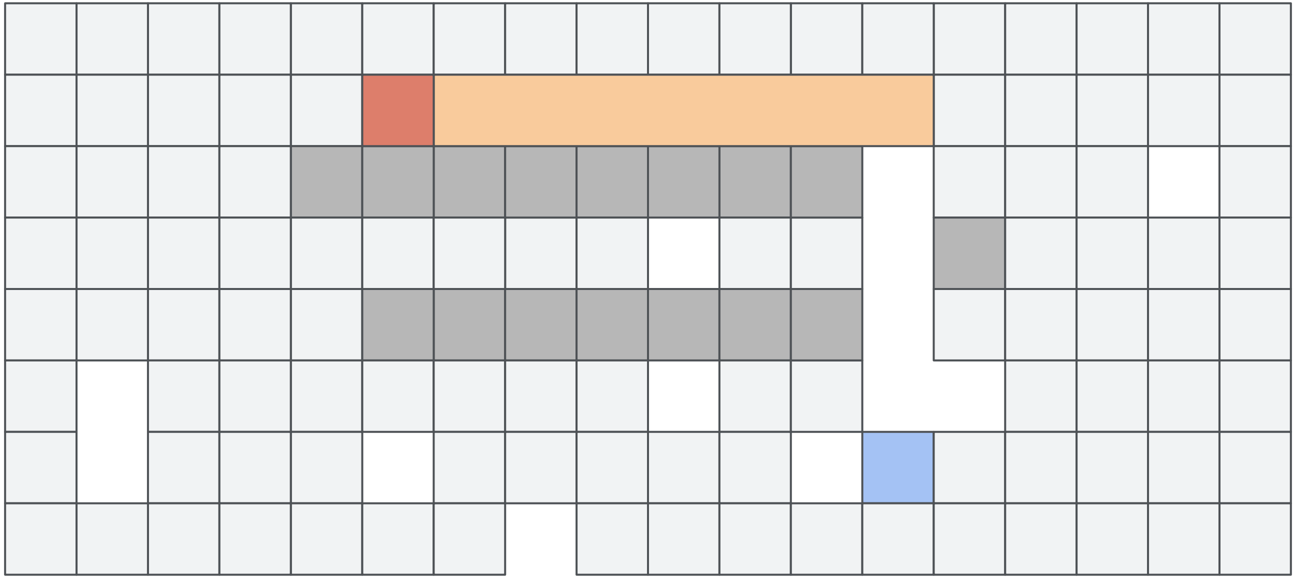
Time 1 → Time 2

Walking surface codes in action



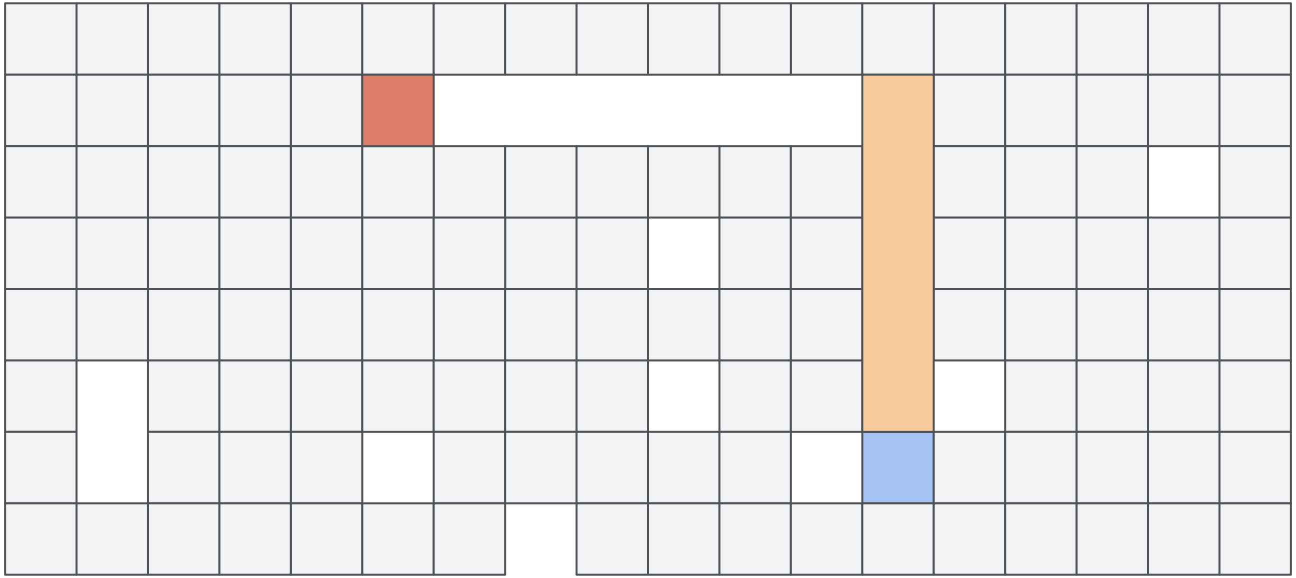
Time 1 → Time 2

Walking surface codes in action



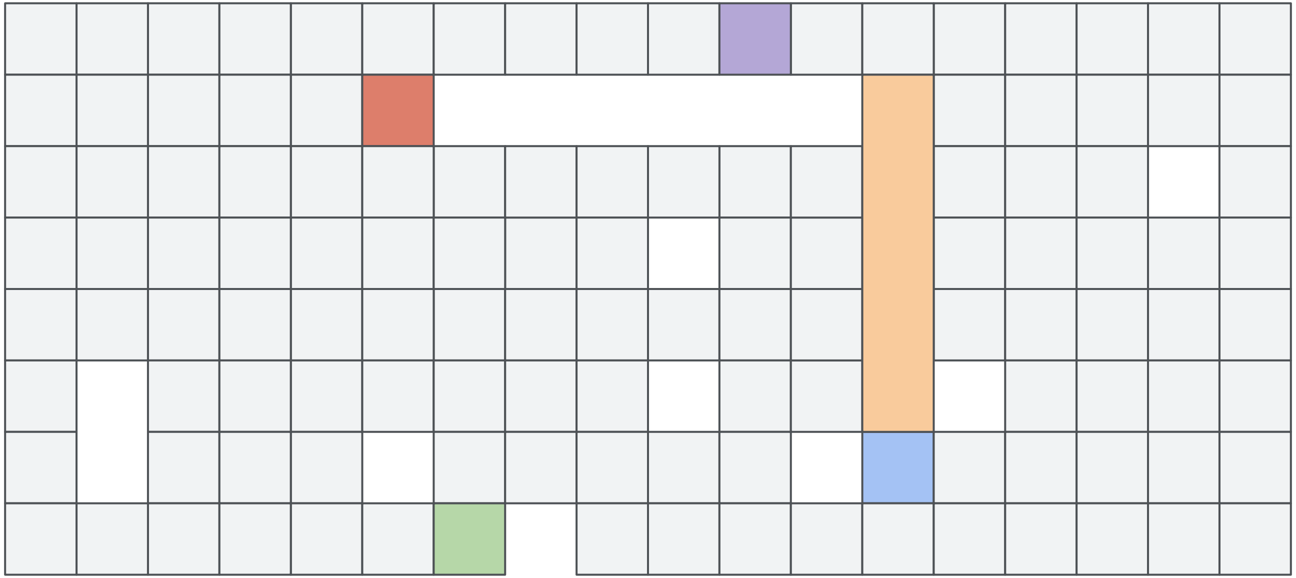
Time 1 → **Time 2**

Walking surface codes in action



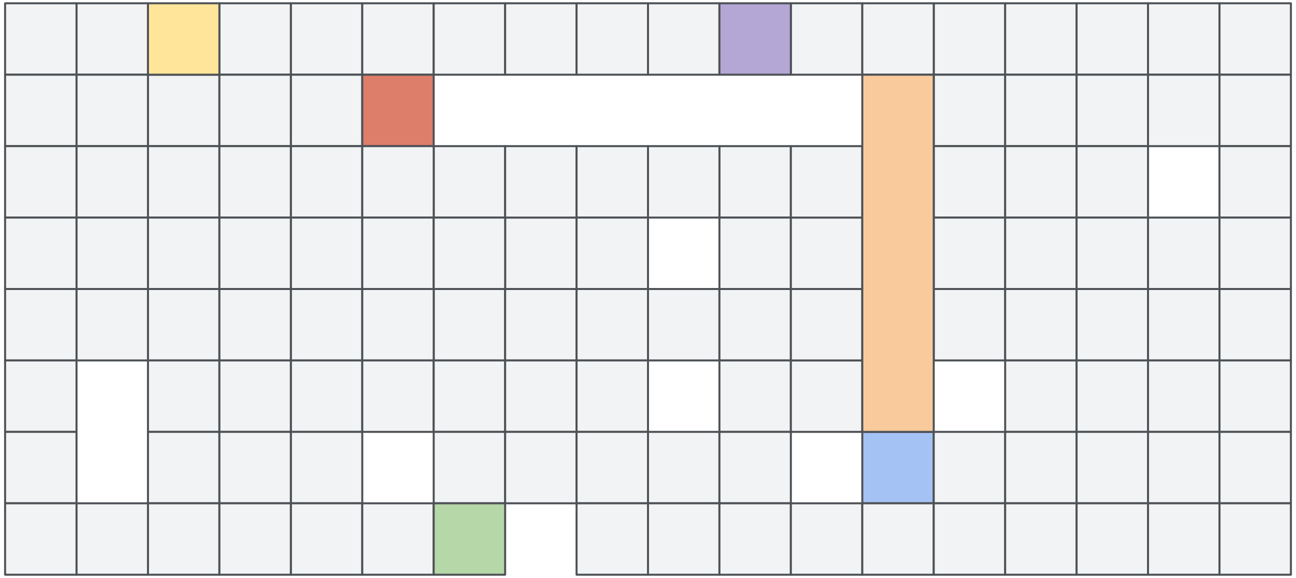
Time 2 → Time 3

Walking surface codes in action



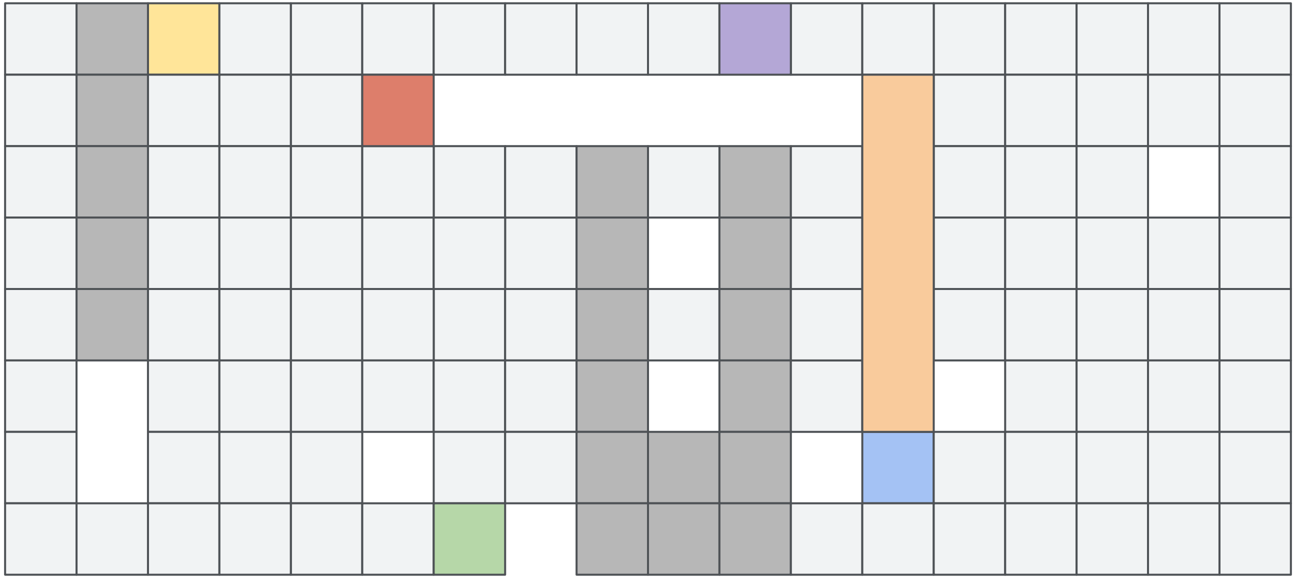
Time 2 → Time 3

Walking surface codes in action



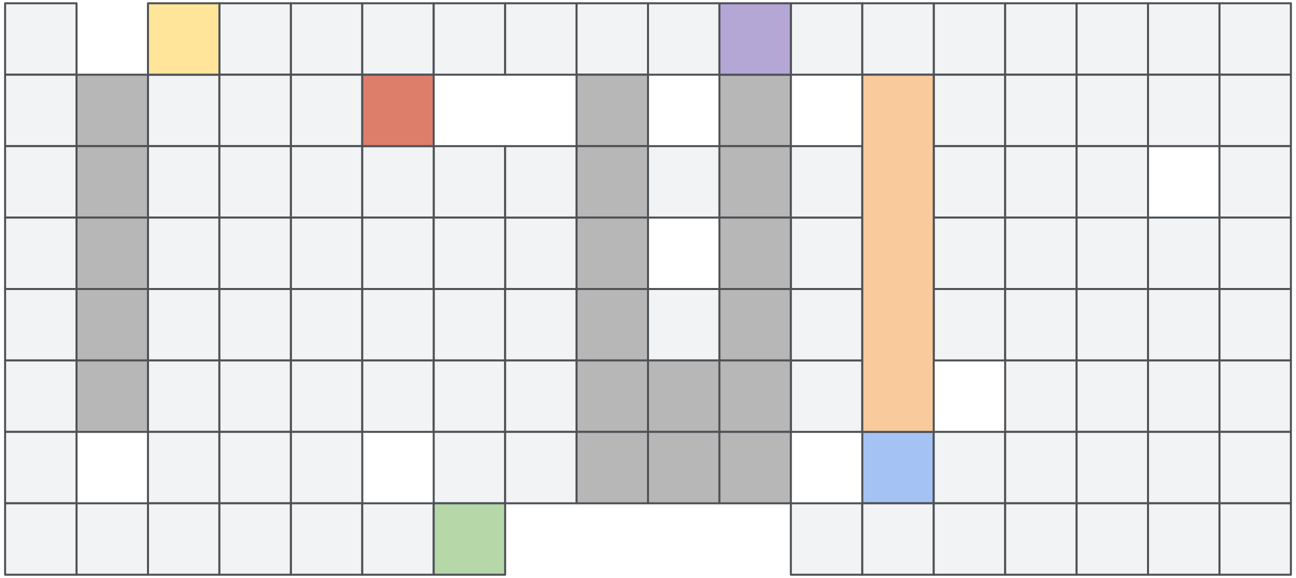
Time 2 → Time 3

Walking surface codes in action



Time 2 → Time 3

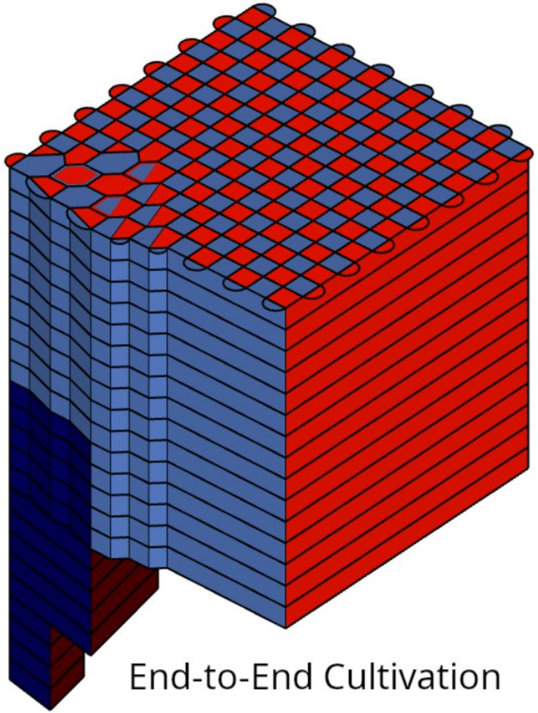
Walking surface codes in action



Time 2 → **Time 3**

Quantum AI

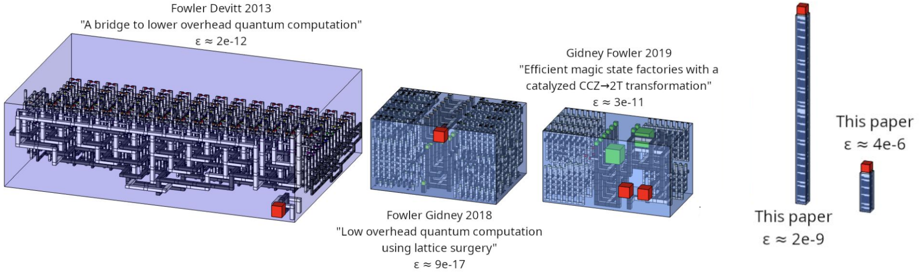
Magic state cultivation



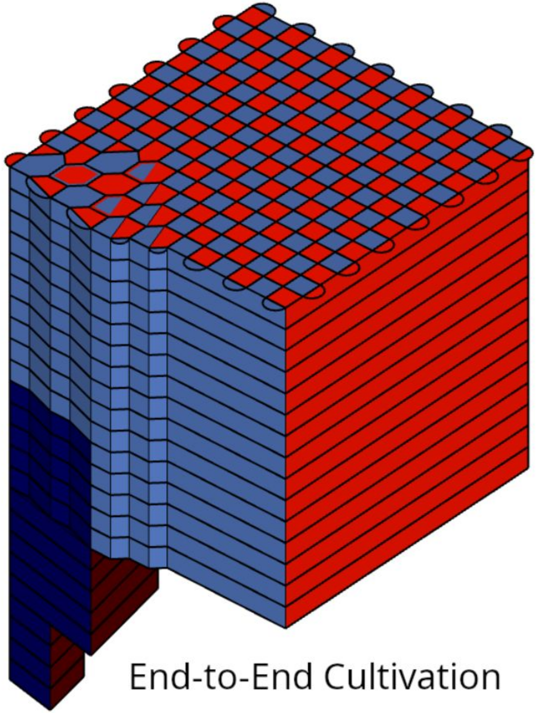
Magic state cultivation prepares a resource state we can use to perform a T gate.

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


Fowler Devitt 2013
"A bridge to lower overhead quantum computation"
ε ≈ 2e-12
Gidney Fowler 2019
"Efficient magic state factories with a catalyzed CCZ→2T transformation"
ε ≈ 3e-11
This paper
ε ≈ 4e-6
Fowler Gidney 2018
"Low overhead quantum computation using lattice surgery"
ε ≈ 9e-17
This paper
ε ≈ 2e-9

Magic state cultivation



End-to-End 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"

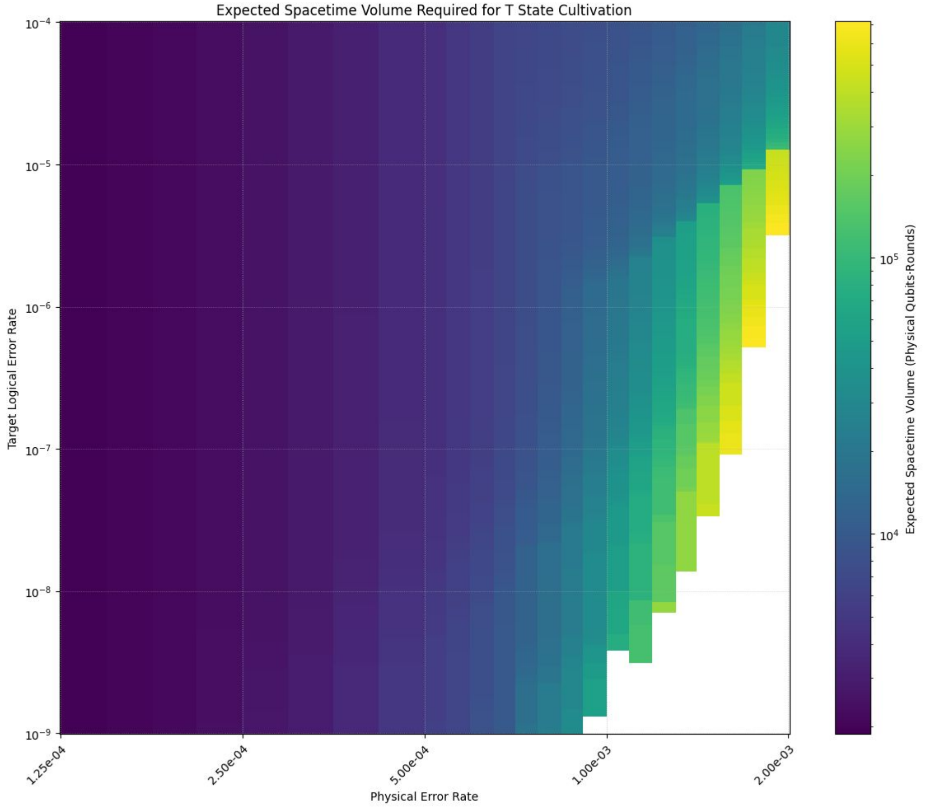
Magic state cultivation

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

We optimize by varying:

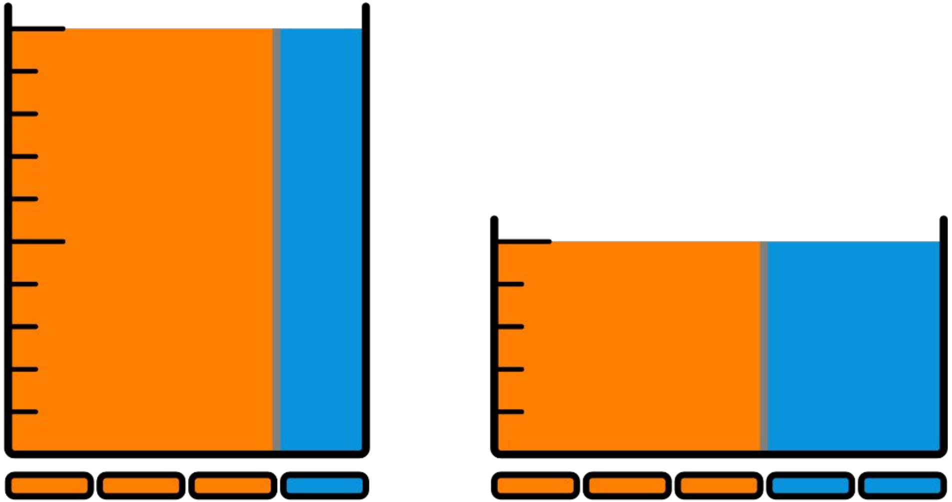
- The amount of postselection
- The choice of color code distance

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



The model

Fluid allocation of surface code qubits



A simple cartoon

- Space is horizontal and time is vertical
- Data qubits are orange, fluid ancilla are blue

We make sure we have enough ancilla volume

$$\textsc{Total Ancilla Volume} = \sum_g \textsc{Ancilla Volume}(g)$$

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

With more fluid ancilla...

- The ancilla volume (blue) is conserved
- The overall spacetime volume is reduced (orange and blue)

Gate costs in the FLASQ model

Basic gates	FLASQ ancilla volume	Measurement depth	Notes
X / Y / Z	0	0	Implemented in software
X / Z basis measurement (or initialization)	0	0	
H	7	0	Includes the cost of a patch rotation
S / $S^\dagger$	5.5	0	
T / $T^\dagger$	$1.5v(p_{phys}, p_{cult}) + t_{react} + 6$	1	Depends on physical error rate, p_{phys}, and target logical error rate, p_{cult}
Move	$5p(q_1, q_2)$	0	Moves a qubit to an empty patch
$CNOT$ / CZ	$5p(q_1, q_2)$	0	

The reaction limit

Most operations can be rearranged freely in space and time

Some measurements have to be performed serially

- The choice of measurement basis depends on earlier measurement outcomes
- So the control software has to catch up before the measurement can be made

This is the ”reaction limit”

- We often assume 10 microseconds
- We don’t expect this to be the limiting factor in early fault-tolerance

Just in case, we ensure that

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

Quantum AI

Applications

Ising model time dynamics

Time evolution

$$H = -J \sum_{\langle i,j \rangle} Z_i Z_j + g \sum_i X_i$$

Estimate correlation functions

- Time-evolve after a high-entanglement quench
- Measure the expectation value of $Z_{tot}^2 = \frac{1}{N^2} \sum_{j,k} Z_j Z_k$
- Target an absolute error ≤ 0.01 with high probability

Classically challenging

- Canonical benchmark for quantum many-body physics
- Recent tensor network heuristics (Mandra 2025) struggle to perfectly converge and bound errors at an 11×11 system size

FLASQ lets us estimate the volume (V)

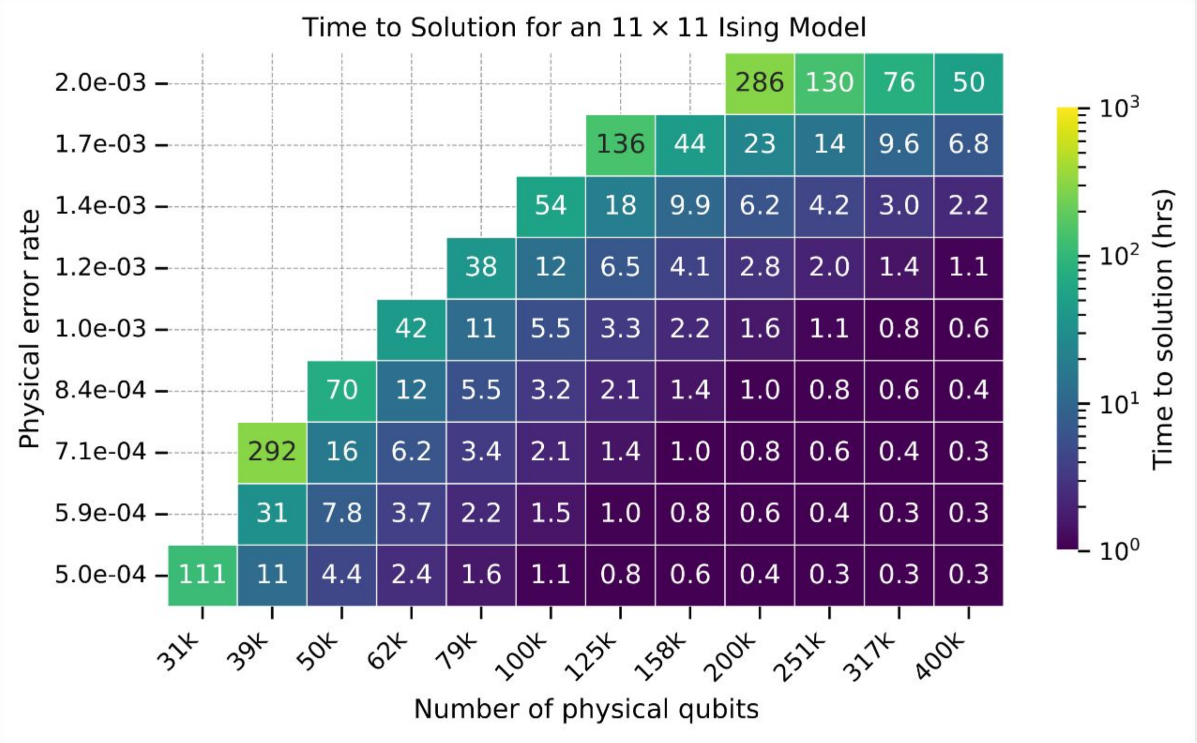
Phenomenological error model

- Surface code suppression: $\Lambda = 0.01/p_{phys}$
- Logical error rate: $p_{cyc} \approx 0.03\Lambda^{-(d+1)/2}$
- Magic state error: p_{mag} from cultivation

Probabilistic error cancellation

- We can mitigate residual errors to get an unbiased estimator
- Sampling overhead: $\Gamma^2 \approx \exp\left(4p_{cyc} d S_{cliff} + 4p_{mag} M\right)$

Ising model resource estimates



Time to Solution for an 11 × 11 Ising Model

Physical error rate	31K	39K	50K	62K	79K	100K	125K	158K	200K	251K	317K	400K
2.0e-03									286	130	76	50
1.7e-03							136	44	23	14	9.6	6.8
1.4e-03						54	18	9.9	6.2	4.2	3.0	2.2
1.2e-03					38	12	6.5	4.1	2.8	2.0	1.4	1.1
1.0e-03				42	11	5.5	3.3	2.2	1.6	1.1	0.8	0.6
8.4e-04			70	12	5.5	3.2	2.1	1.4	1.0	0.8	0.6	0.4
7.1e-04		292	16	6.2	3.4	2.1	1.4	1.0	0.8	0.6	0.4	0.3
5.9e-04		31	7.8	3.7	2.2	1.5	1.0	0.8	0.6	0.4	0.3	0.3
5.0e-04	111	11	4.4	2.4	1.6	1.1	0.8	0.6	0.4	0.3	0.3	0.3

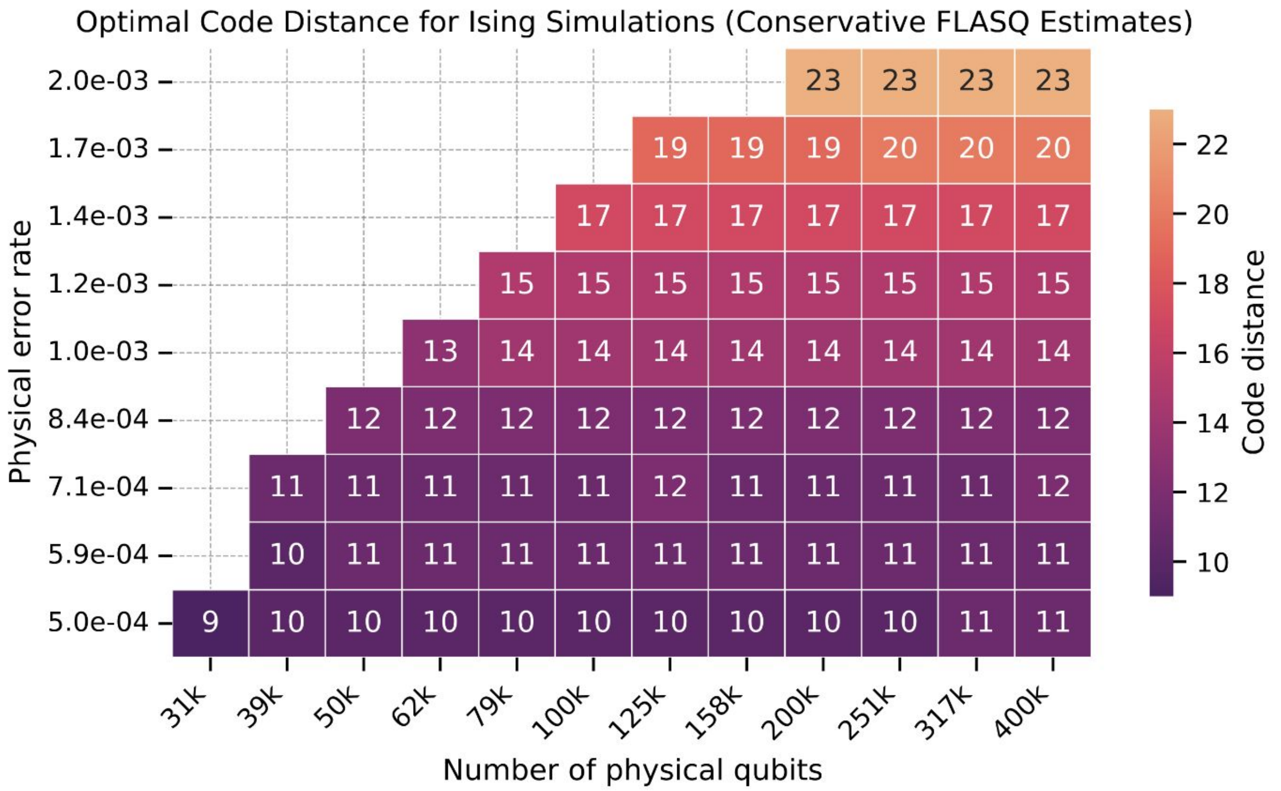
Time-to-solution

- 20 second-order Trotter steps
- Optimized over choice of code distance and cultivation parameters
- Assumes target standard deviation of $\sigma = 0.0045$

Cost drivers

- Large number of samples for target precision
- $\approx 7,400$ arbitrary rotations

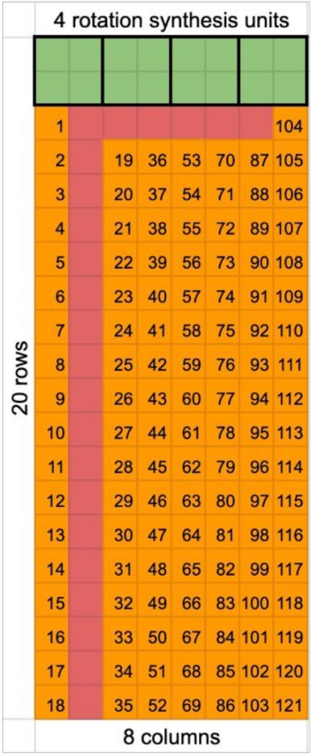
Optimal code distance



Optimal Code Distance for Ising Simulations (Conservative FLASQ Estimates)

Physical error rate	31K	39K	50K	62K	79K	100K	125K	158K	200K	251K	317K	400K
2.0e-03									23	23	23	23
1.7e-03							19	19	19	20	20	20
1.4e-03						17	17	17	17	17	17	17
1.2e-03					15	15	15	15	15	15	15	15
1.0e-03				13	14	14	14	14	14	14	14	14
8.4e-04			12	12	12	12	12	12	12	12	12	12
7.1e-04		11	11	11	11	11	12	11	11	11	11	12
5.9e-04		10	11	11	11	11	11	11	11	11	11	11
5.0e-04	9	10	10	10	10	10	10	10	10	10	11	11

Validating against a hand compilation

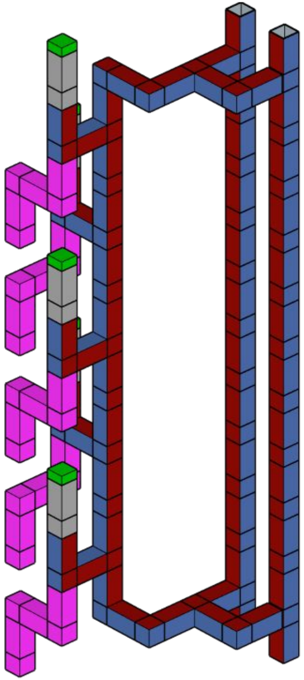


4 rotation synthesis units

20 rows

🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩
🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩
1	🟥	🟥	🟥	🟥	🟥	🟥	104
2	🟥	19	36	53	70	87	105
3	🟥	20	37	54	71	88	106
4	🟥	21	38	55	72	89	107
5	🟥	22	39	56	73	90	108
6	🟥	23	40	57	74	91	109
7	🟥	24	41	58	75	92	110
8	🟥	25	42	59	76	93	111
9	🟥	26	43	60	77	94	112
10	🟥	27	44	61	78	95	113
11	🟥	28	45	62	79	96	114
12	🟥	29	46	63	80	97	115
13	🟥	30	47	64	81	98	116
14	🟥	31	48	65	82	99	117
15	🟥	32	49	66	83	100	118
16	🟥	33	50	67	84	101	119
17	🟥	34	51	68	85	102	120
18	🟥	35	52	69	86	103	121

8 columns



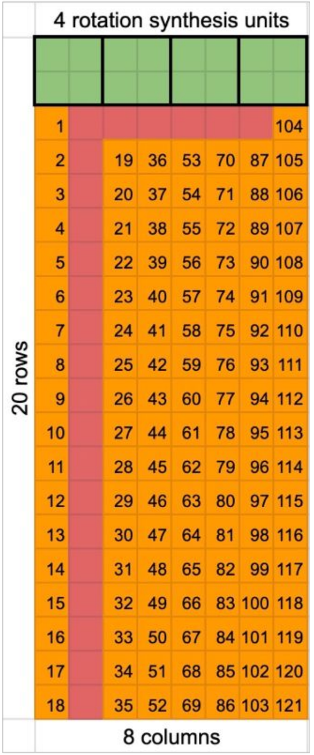
Static layout

- We couple the data qubits (orange) to the rotation synthesis areas (green)
- We use walking surface codes to shift the red access hallway

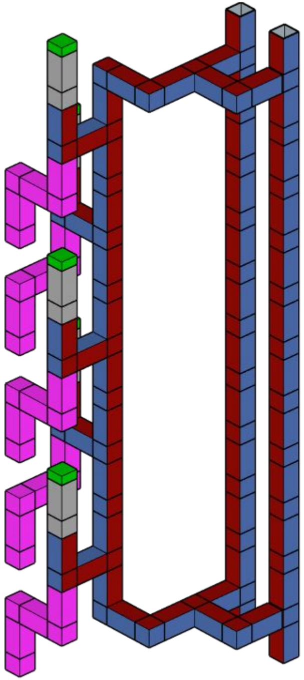
Rotation synthesis gadget

- Each 2×2 region can implement an arbitrary rotation
- They consume one T state every four logical timesteps

Validating against a hand compilation



🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩
🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩
1	🟥	🟥	🟥	🟥	🟥	🟥	104
2	🟥	19	36	53	70	87	105
3	🟥	20	37	54	71	88	106
4	🟥	21	38	55	72	89	107
5	🟥	22	39	56	73	90	108
6	🟥	23	40	57	74	91	109
7	🟥	24	41	58	75	92	110
8	🟥	25	42	59	76	93	111
9	🟥	26	43	60	77	94	112
10	🟥	27	44	61	78	95	113
11	🟥	28	45	62	79	96	114
12	🟥	29	46	63	80	97	115
13	🟥	30	47	64	81	98	116
14	🟥	31	48	65	82	99	117
15	🟥	32	49	66	83	100	118
16	🟥	33	50	67	84	101	119
17	🟥	34	51	68	85	102	120
18	🟥	35	52	69	86	103	121



Static layout

- We couple the data qubits (orange) to the rotation synthesis areas (green)
- We use walking surface codes to shift the red access hallway

Rotation synthesis gadget

- Each 2×2 region can implement an arbitrary rotation
- They consume one T state every four logical timesteps

Comparison with FLASQ estimates

Logical timesteps (FLASQ)	Logical timesteps (by hand)	Ratio (FLASQ / hand)
55995	73810	0.76

Probing the crossover between NISQ and fault-tolerance

Two modes of operation for a quantum processor

NISQ Mode

- Treat the device as a large, noisy processor
- Run many parallel copies of the simulation using all available space

Simple NISQ error model

- Single-qubit depolarizing noise after every two-qubit gate
- Allows for arbitrary rotations in a single timestep
- Mitigated using Probabilistic Error Cancellation (PEC)

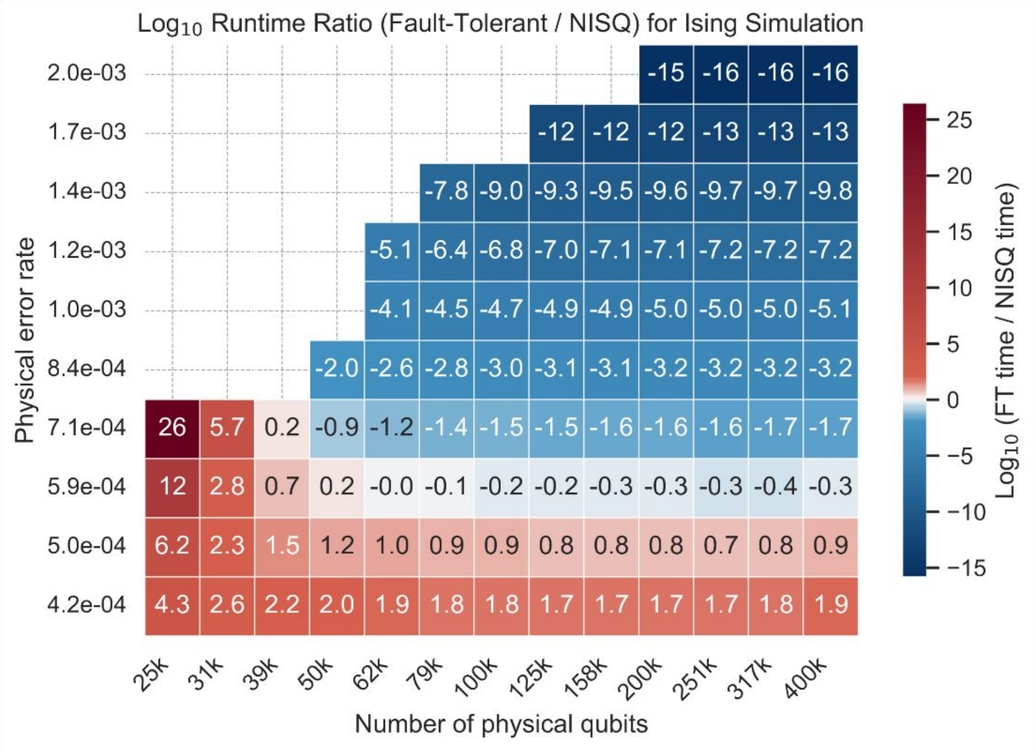
Fault-Tolerant Mode

- Allocate all physical qubits to a single, error-corrected simulation
- (Does not account for the possibility of Heisenberg-limited measurement)

Phenomenological FT error model

- Rotations compiled into Clifford + T gate set
- Same as other Ising simulation (single-qubit X and Z errors)
- Use PEC to mitigate residual errors

Moderately deep circuits (20 Trotter Steps)



Physical error rate	25k	31k	39k	50k	62k	79k	100k	125k	158k	200k	251k	317k	400k
2.0e-03										-15	-16	-16	-16
1.7e-03								-12	-12	-12	-13	-13	-13
1.4e-03						-7.8	-9.0	-9.3	-9.5	-9.6	-9.7	-9.7	-9.8
1.2e-03					-5.1	-6.4	-6.8	-7.0	-7.1	-7.1	-7.2	-7.2	-7.2
1.0e-03					-4.1	-4.5	-4.7	-4.9	-4.9	-5.0	-5.0	-5.0	-5.1
8.4e-04				-2.0	-2.6	-2.8	-3.0	-3.1	-3.1	-3.2	-3.2	-3.2	-3.2
7.1e-04	26	5.7	0.2	-0.9	-1.2	-1.4	-1.5	-1.5	-1.6	-1.6	-1.6	-1.7	-1.7
5.9e-04	12	2.8	0.7	0.2	-0.0	-0.1	-0.2	-0.2	-0.3	-0.3	-0.3	-0.4	-0.3
5.0e-04	6.2	2.3	1.5	1.2	1.0	0.9	0.9	0.8	0.8	0.8	0.7	0.8	0.9
4.2e-04	4.3	2.6	2.2	2.0	1.9	1.8	1.8	1.7	1.7	1.7	1.7	1.8	1.9

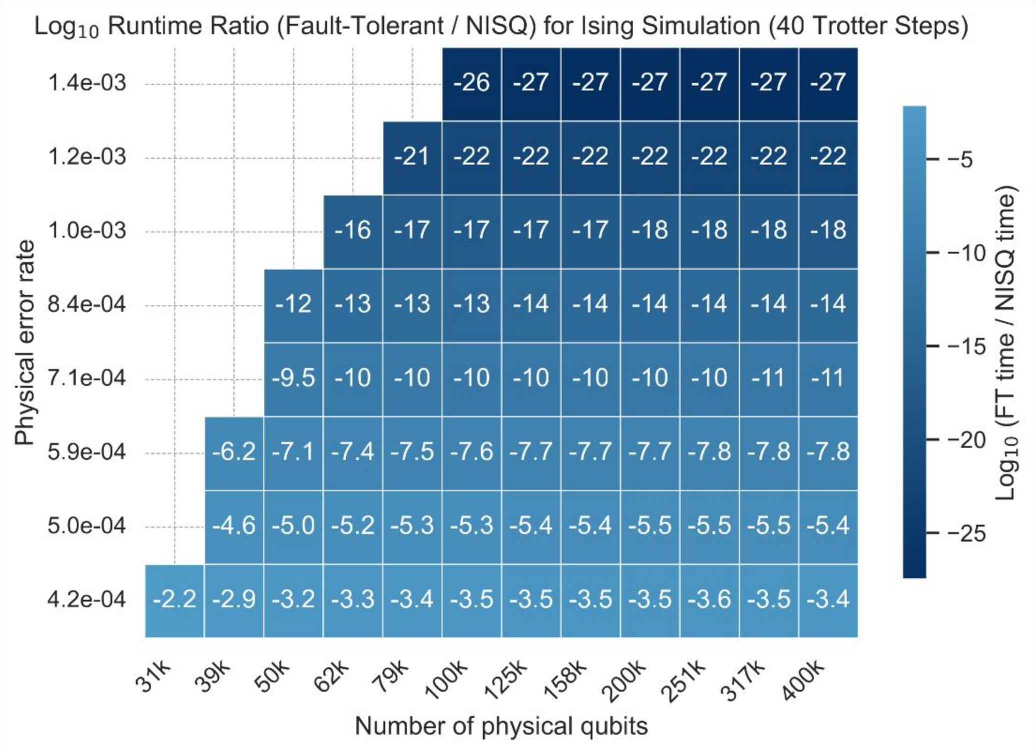
Reading the heatmap

- We plot the log-ratio of the runtimes: $\log_{10}(T_{FT}/T_{NISQ})$
- **Red regions:** NISQ is faster
- **Blue regions:** Fault-Tolerant is faster

The tradeoff

- NISQ wins at low error rates where the exponential overhead is not too large
- The FT mode is preferable at higher error rates (provided there is enough space)

Deeper circuits (40 Trotter Steps)



Physical error rate	31k	39k	50k	62k	79k	100k	125k	158k	200k	251k	317k	400k
1.4e-03						-26	-27	-27	-27	-27	-27	-27
1.2e-03					-21	-22	-22	-22	-22	-22	-22	-22
1.0e-03				-16	-17	-17	-17	-17	-18	-18	-18	-18
8.4e-04			-12	-13	-13	-13	-14	-14	-14	-14	-14	-14
7.1e-04			-9.5	-10	-10	-10	-10	-10	-10	-10	-11	-11
5.9e-04		-6.2	-7.1	-7.4	-7.5	-7.6	-7.7	-7.7	-7.7	-7.8	-7.8	-7.8
5.0e-04		-4.6	-5.0	-5.2	-5.3	-5.3	-5.4	-5.4	-5.5	-5.5	-5.5	-5.4
4.2e-04	-2.2	-2.9	-3.2	-3.3	-3.4	-3.5	-3.5	-3.5	-3.5	-3.6	-3.5	-3.4

Reading the heatmap

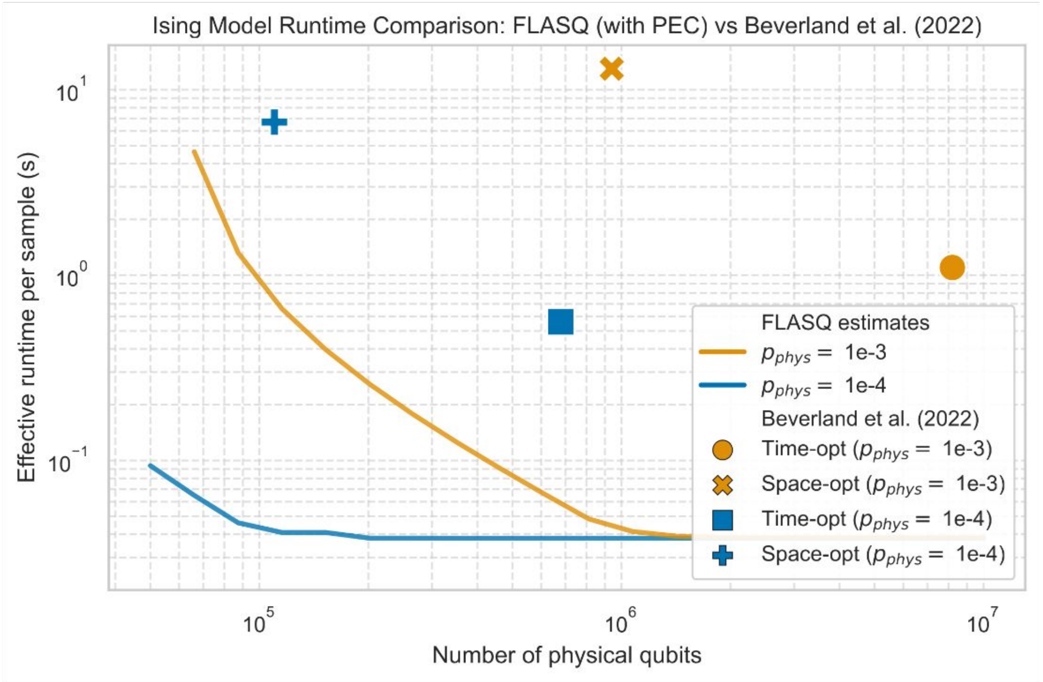
- We plot the log-ratio of the runtimes: $\log_{10}(T_{FT}/T_{NISQ})$
- **Red regions:** NISQ is faster
- **Blue regions:** Fault-Tolerant is faster

Doubling the depth shifts the balance

- The NISQ calculations hit an exponential wall
- The fault-tolerant cost increases only marginally

Comparing FLASQ to previous FT estimates

Reduced resource estimates compared to prior work



Previous state-of-the-art

- Beverland et al. (2022) analyzed a 10×10 Ising model (4th-order Trotter)
- Relied purely on QEC + traditional Magic State Distillation
- Strict target error budget of $\epsilon = 0.001$ (No error mitigation)

FLASQ predicts much lower runtimes

- Same error budget, but PEC allows for smaller code distances
- Cultivation and error mitigation together reduce the cost of non-Clifford gates
- **More than an order of magnitude reduction in space and time costs at realistic physical error rates** ($p_{phys} = 10^{-3}$)

Case Study: Hamming Weight Phasing

The cost of synthesizing parallel rotations

The Problem

- Synthesizing many arbitrary rotations can dominate the cost of an algorithm, even with cultivation
- Many applications (including the Ising model simulations) require executing many identical $R_Z(\theta)$ rotations in parallel

One approach: Hamming weight phasing (HWP)

- Calculate the Hamming weight of the N target qubits into an ancilla register of size $\approx \log_2(N)$
- Apply scaled rotations only to the register: $R_Z(\theta)$ to bit 1, $R_Z(2\theta)$ to bit 2, etc.
- Uncompute the Hamming weight
- T-count drops to $\approx 4N + \mathcal{O}(\log N \cdot \log \epsilon^{-1})$

The cost of synthesizing parallel rotations

The Problem

- Synthesizing many arbitrary rotations can dominate the cost of an algorithm, even with cultivation
- Many applications (including the Ising model simulations) require executing many identical $R_Z(\theta)$ rotations in parallel

One approach: Hamming weight phasing (HWP)

- Calculate the Hamming weight of the N target qubits into an ancilla register of size $\approx \log_2(N)$
- Apply scaled rotations only to the register: $R_Z(\theta)$ to bit 1, $R_Z(2\theta)$ to bit 2, etc.
- Uncompute the Hamming weight
- T-count drops to $\approx 4N + \mathcal{O}(\log N \cdot \log \epsilon^{-1})$

Theoretical T-counts for precision $\epsilon = 10^{-5}$

Qubits to Rotate	HWP T-Count	Parallel R_Z T-Count	T-Count Ratio (R_Z/HWP)
15	≈ 112	≈ 270	$\approx 2.4\times$
43	≈ 252	≈ 774	$\approx 3.1\times$
121	≈ 576	$\approx 2{,}178$	$\approx 3.8\times$

If we only count T gates...

- HWP looks very favorable
- But this is missing two important factors

The other costs of Hamming Weight Phasing

Ancilla qubit overhead

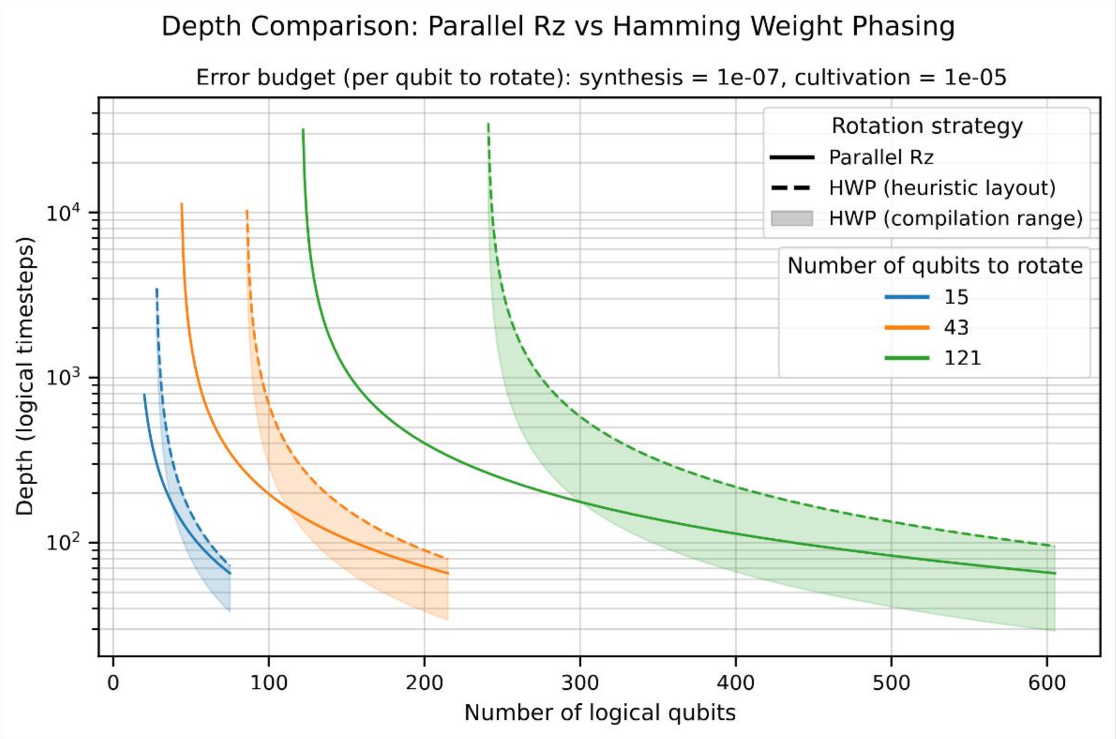
- Computing the Hamming weight requires a sophisticated arithmetic circuit
- Prior work minimizes the T count but uses $\approx N$ extra ”algorithmic” ancilla qubits
- These ancilla are not available to mediate other operations

Clifford and routing complexity

- The arithmetic circuits require a significant number of Clifford operations
- Compiling the multi-qubit Clifford and non-Clifford operations to a 2D grid adds more overhead

Evaluating the impact of these tradeoffs requires a detailed cost model

Evaluating HWP with the FLASQ model



Comparing depth (in terms of logical timesteps)

- **Solid lines:** Naive Parallel R_Z approach
- **Dotted lines:** HWP with an explicit (but not optimal) layout on a 2D grid
- **Shaded region:** HWP neglecting some fraction of the routing overhead

Takeaways

- The other costs of HWP signficantly reduce its potential advantage
- A careful analysis, with an explicit layout, would be required to benefit at all (even with abundant space)

arxiv.org/abs/2511.08508

FLuid Allocation of Surface code Qubits (FLASQ)

Goal: estimate the resources required to implement a quantum circuit ...

- ... In a two-dimensional surface code architecture
- ... With an (currently non-existent) compilation stack

Key assumption: Operations can be freely rearranged in space and time

- Their ”extra” ancilla spacetime volume is conserved
- This is justified by the inherent flexibility of lattice surgery and the use of walking surface codes
- We separately account for the reaction depth

This assumption makes it easy to estimate the overall spacetime volume

- This enables estimates of error rates, wall-clock times, etc
- Can be done programmatically (ask me for the code!)
- We can use this ability to balance tradeoffs

But it also ignores some important factors

Going forward

Early fault-tolerance is a moving target

- Magic state cultivation
- Developments in quantum error correction
- Careful combinations of existing techniques can have a large impact

Just counting T gates may not be sufficient

- If you are dominated by small angle rotations it might be okay
- But the cost of routing and Clifford operations can be large
- Even determining this requires a good model!

Compilation for early fault-tolerance is a big project

- FLASQ doesn’t actually solve the problem, it just guesses the cost
- This work is necessary and may lead to surprising cost reductions

Thank you

μ = 1/π
Δ| = T(x) + e
F = m√x
g = 1/√e