# The threshold theorem:

A quantum circuit on *n* qubits and containing *k* gates can be successfully run with probability of error at most $\varepsilon$ using K total gates, where:

$$K \sim O(\log^c(\frac{k}{\varepsilon})k)$$

(for some constant *c*) on hardware whose components fail with probability at most *p*, provided *p* is below some constant *threshold*, $p < p_{th}$,and given reasonable assumptions about the noise in the underlying hardware.

# The threshold theorem:

A quantum circuit on *n* qubits and containing *k* gates can be successfully run with probability of error at most $\varepsilon$ using K total gates, where:

$$K \sim O(\log^c(\frac{k}{\varepsilon})k)$$

(for some constant *c*) on hardware whose components fail with probability at most *p*, provided *p* is below some constant *threshold*, $p < p_{th}$,and given reasonable assumptions about the noise in the underlying hardware.

In general, here “reasonable assumptions” can be interpreted as the standard independent and identically distributed (i.i.d.) assumption.

# The threshold theorem:

"The entire content of the Threshold Theorem is that you're correcting errors faster than they're created. That's the whole point, and the whole non-trivial thing that the theorem shows."

- Scott Aaronson

# What kind of errors can we correct?

Returning to the surface code as a concrete example, a distance $d = L$ code can handle up to (d-1)/2 errors on the data qubits, because it can measure the appropriate syndrome and perform the recovery operation.

# What kind of errors can we correct?

Returning to the surface code as a concrete example, a distance $d = L$ code can handle up to (d-1)/2 errors on the data qubits, because it can measure the appropriate syndrome and perform the recovery operation.

But what about an error on the syndrome measurement itself?

# What kind of errors can we correct?

Returning to the surface code as a concrete example, a distance $d = L$ code can handle up to (d-1)/2 errors on the data qubits, because it can measure the appropriate syndrome and perform the recovery operation.

But what about an error on the syndrome measurement itself?

If we measure the wrong syndrome, we'll perform the wrong recovery operation. We have no guarantee anymore that this won't flip the logical qubit, regardless of $d$. In other words, our error correcting code has no threshold for measurement errors.

# What kind of errors can we correct?

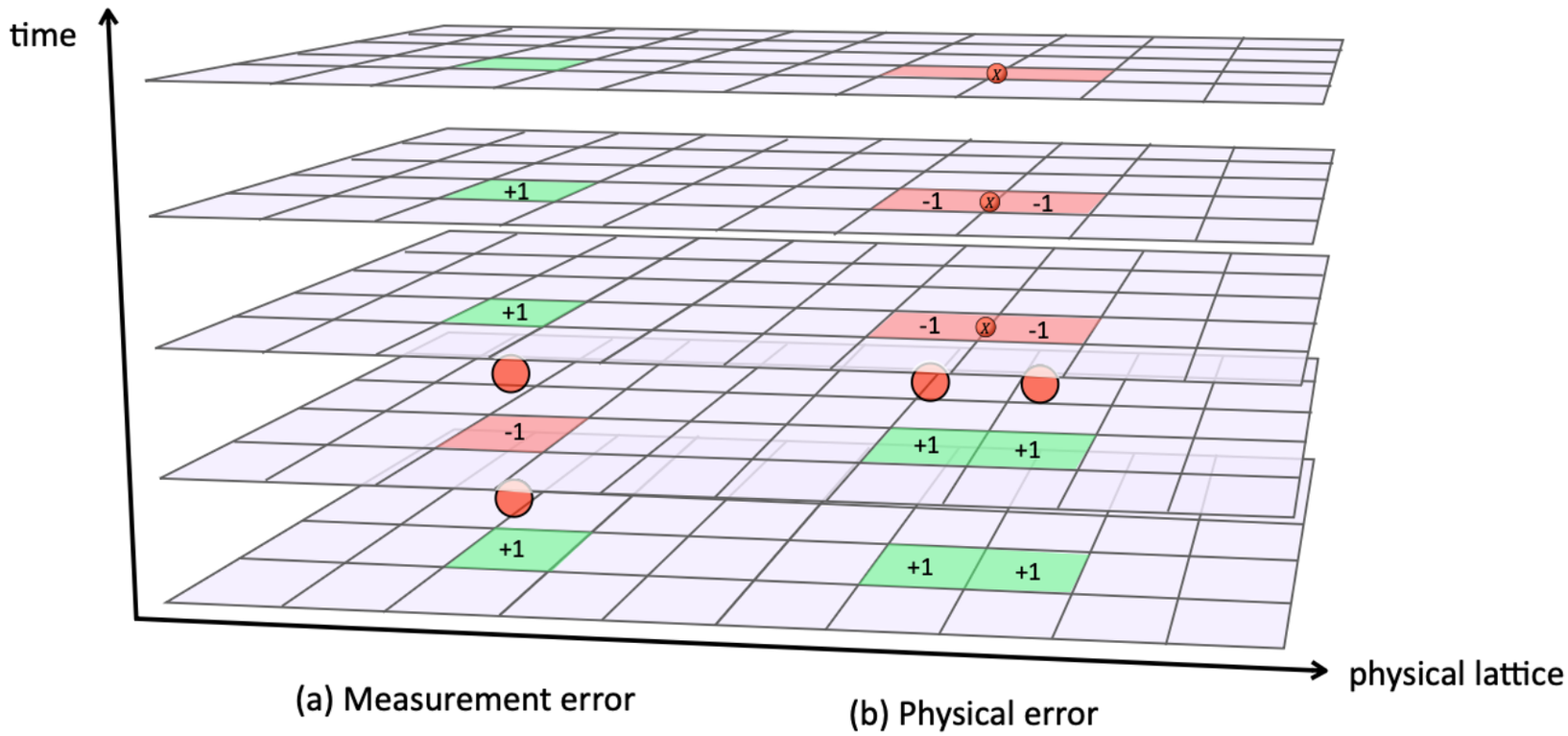
Returning to the surface code as a concrete example, a distance $d = L$ code can handle up to (d-1)/2 errors on the data qubits, because it can measure the appropriate syndrome and perform the recovery operation.

But what about an error on the syndrome measurement itself?

If we measure the wrong syndrome, we'll perform the wrong recovery operation. We have no guarantee anymore that this won't flip the logical qubit, regardless of $d$. In other words, our error correcting code has no threshold for measurement errors.

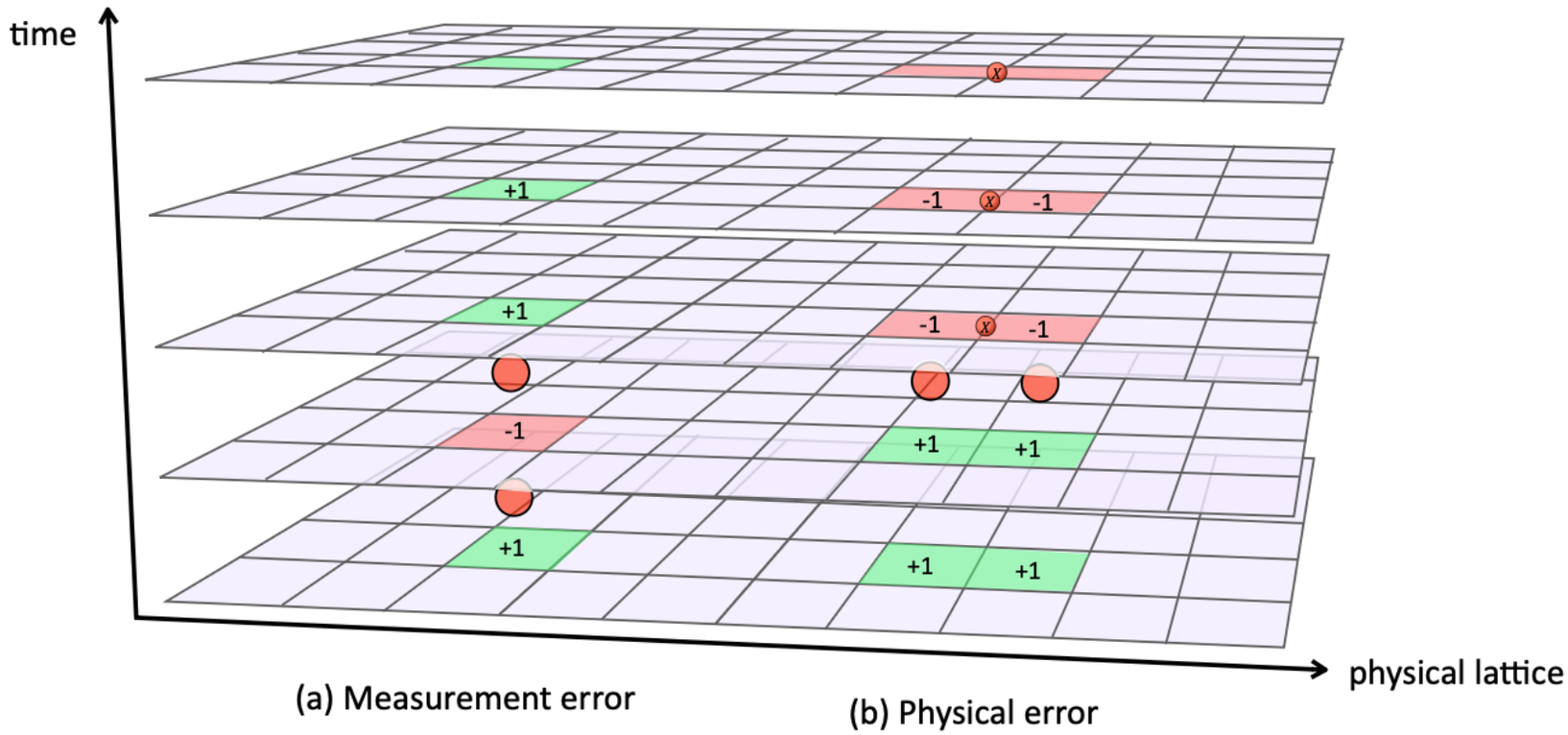
What should we do?

# Repeated measurements



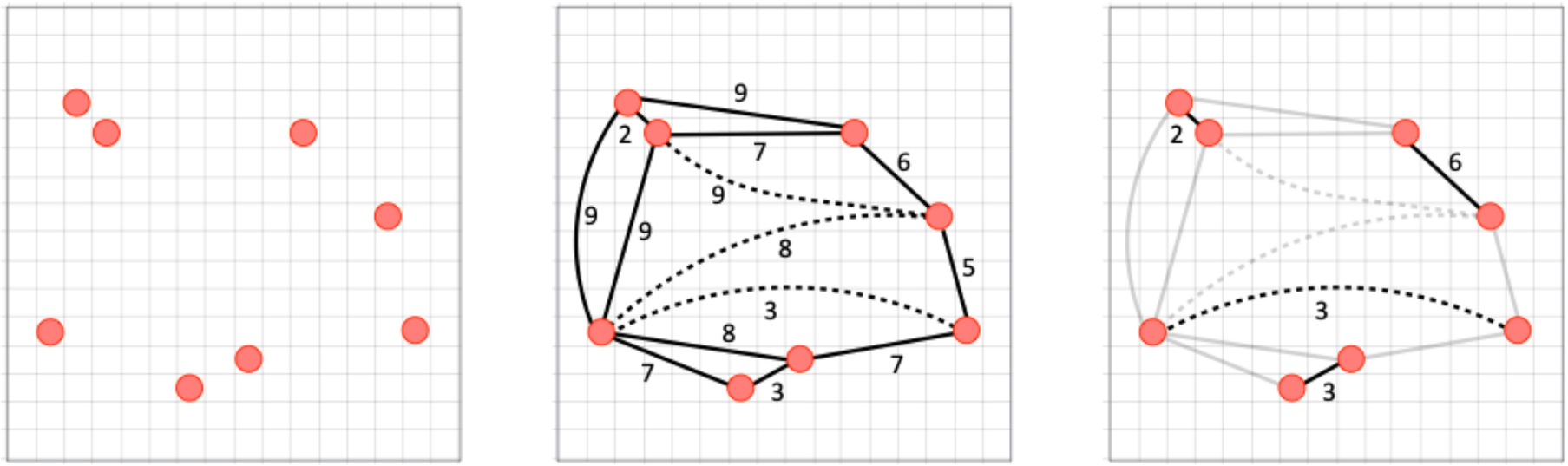

How many times should we measure?

# Repeated measurements



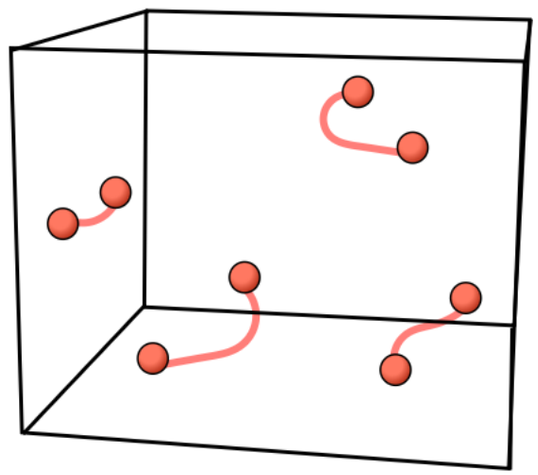

With one round of syndrome extraction, a single measurement error could flip the logical operator, regardless of the code distance *d*. With *D* rounds of repeated syndrome extraction, we could still get the same outcome from *D* faulty measurements in the same location. Therefore, to accommodate measurement errors on the same footing as data qubit errors, we require $D = d$ rounds.

# Repeated measurements




(a) (b) (c)

Our graph problem now becomes 3D instead of 2D:



# Hook errors and error propagation

A Z error on a data qubit will propagate through the circuit without inducing any additional errors (you can check this.) That's good!

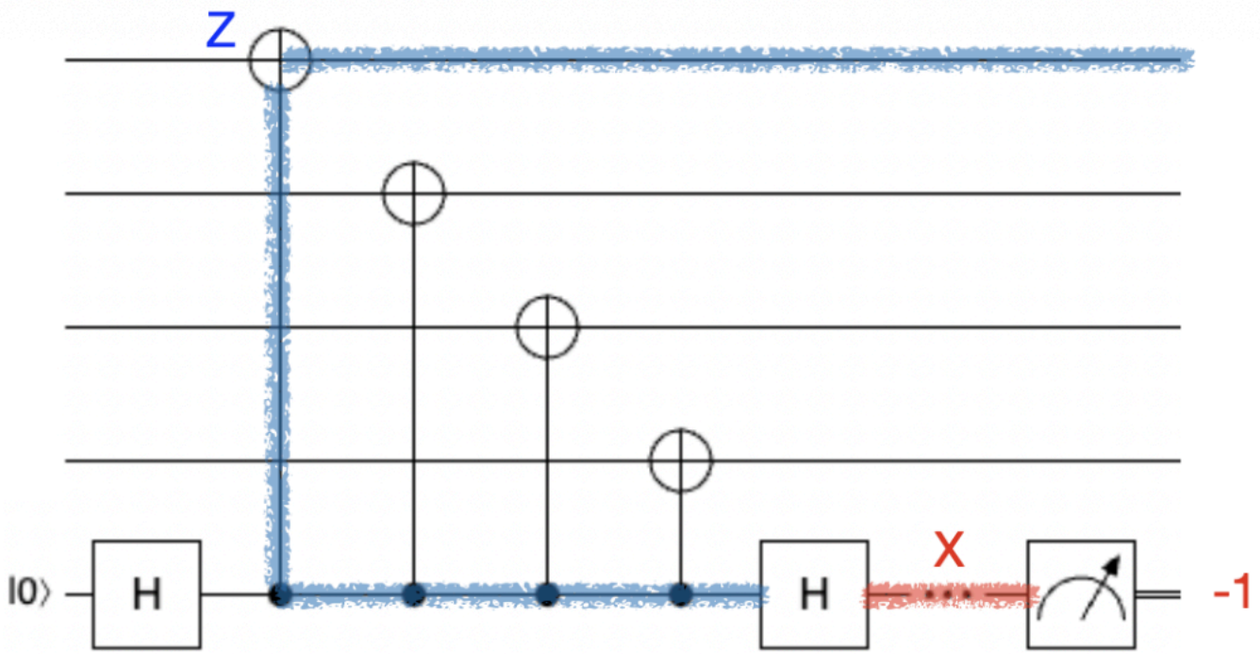
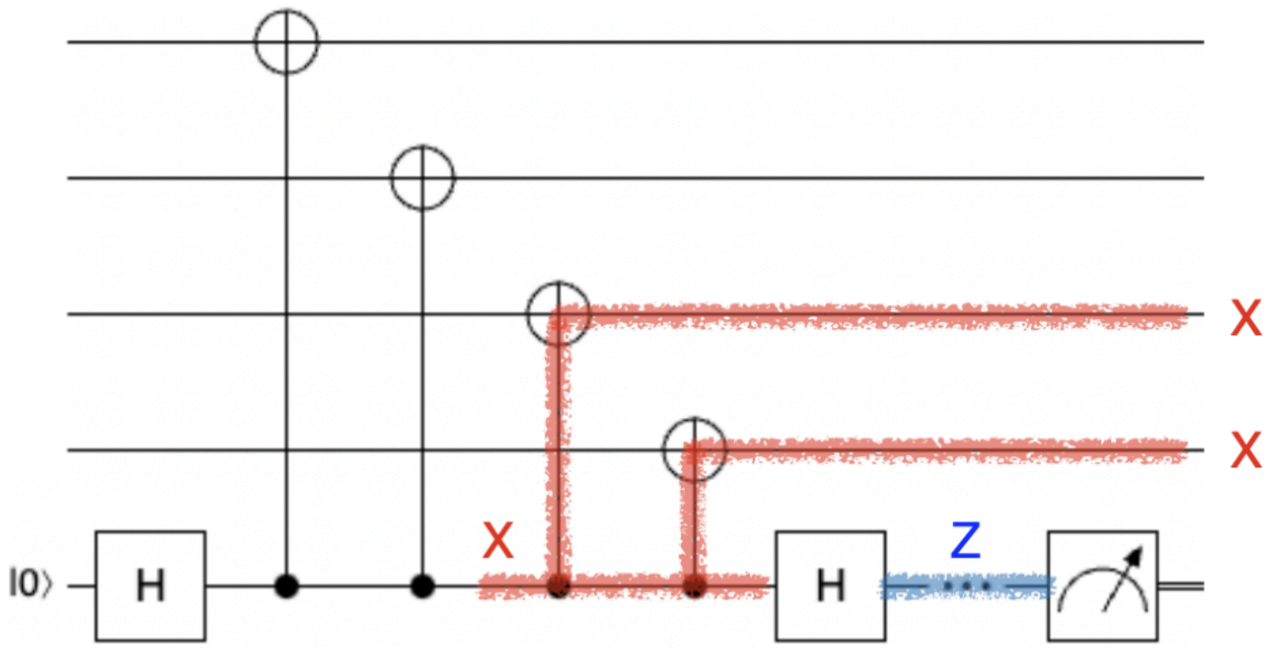



Figure credit: Jeff Thompson

# Hook errors and error propagation

But an X error on the ancilla in the location shown will back-propagate to become an X error on two data qubits. That's bad!




This is called a "Hook" error.

For a general code, you can avoid this by adding additional ancilla qubits for the parity measurement, or by using "flag" qubits to check for Hook errors.

Figure credit: Jeff Thompson

# Hook errors and error propagation

PHYSICAL REVIEW X **11,** 041058 (2021)

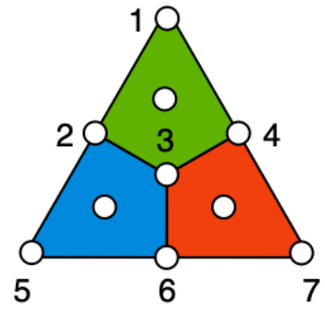
**Featured in Physics**

## Realization of Real-Time Fault-Tolerant Quantum Error Correction

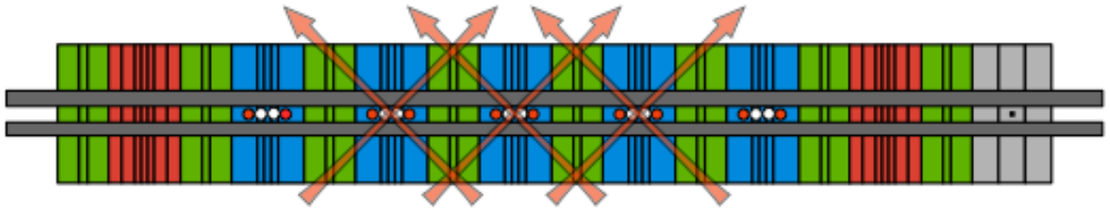
C. Ryan-Anderson, J. G. Bohnet, K. Lee,* D. Gresh, A. Hankin, J. P. Gaebler, D. Francois, A. Chernoguzov, D. Lucchetti, N. C. Brown, T. M. Gatterman, S. K. Halit, K. Gilmore, J. A. Gerber, B. Neyenhuis, D. Hayes, and R. P. Stutz

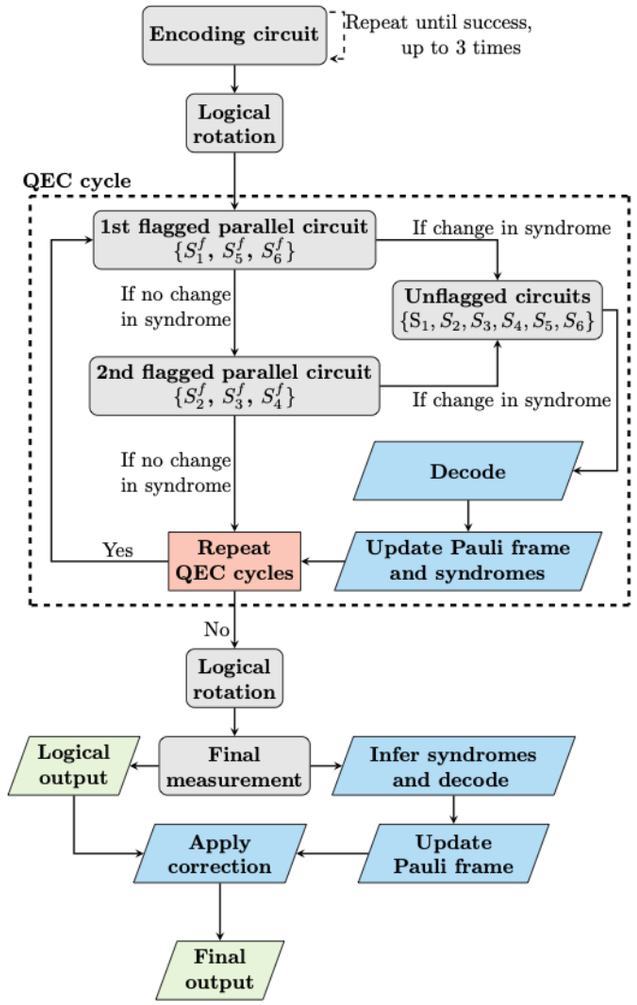
*Quantinuum, 303 South Technology Court, Broomfield, Colorado 80021, USA*

(Received 10 August 2021; revised 24 November 2021; accepted 7 December 2021; published 23 December 2021; corrected 11 January 2022)

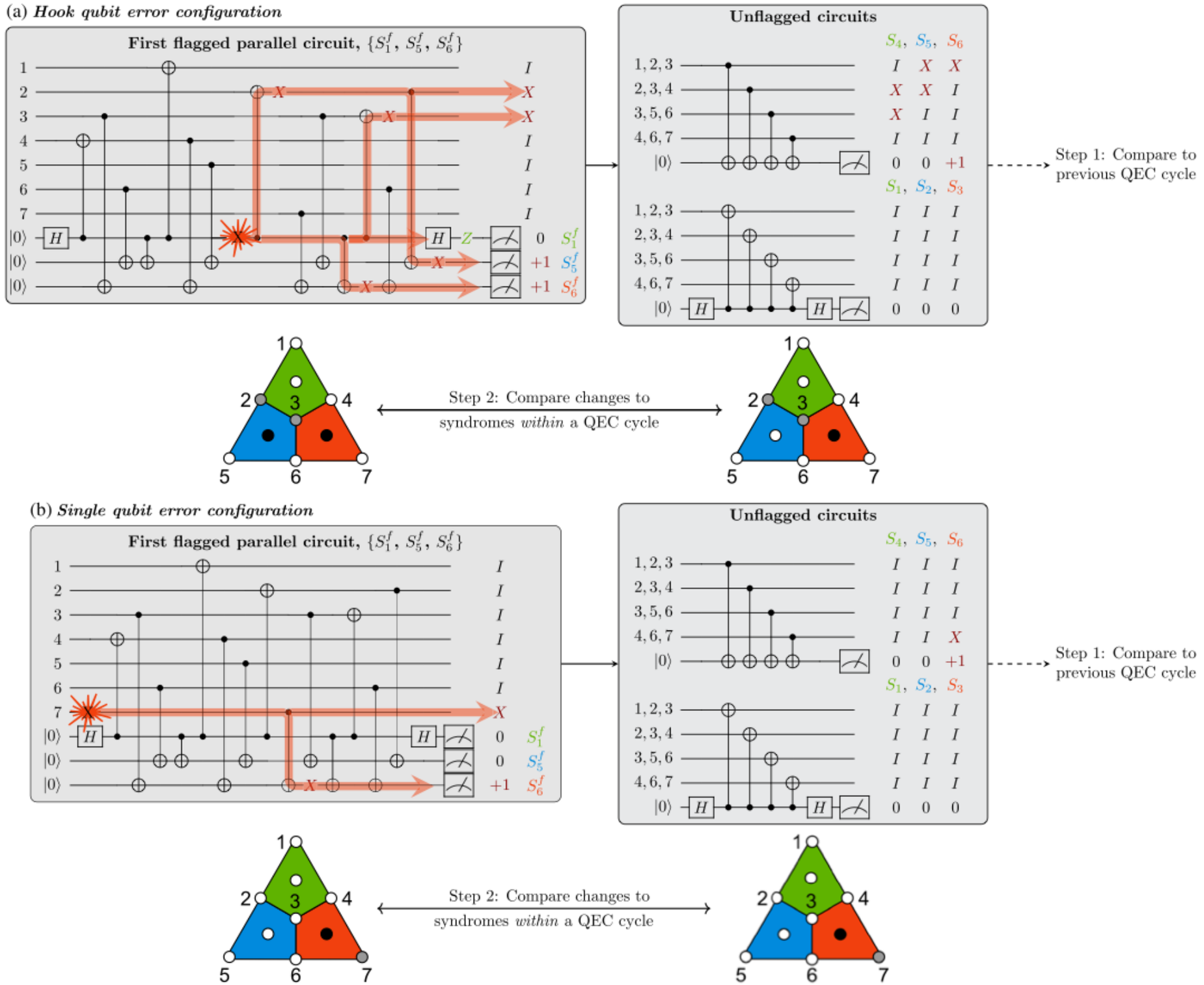



| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| $S_1$ | + | $X_1$ | $X_2$ | $X_3$ | $X_4$ | $I$ | $I$ | $I$ |
| $S_2$ | + | $I$ | $X_2$ | $X_3$ | $I$ | $X_5$ | $X_6$ | $I$ |
| $S_3$ | + | $I$ | $I$ | $X_3$ | $X_4$ | $I$ | $X_6$ | $X_7$ |
| $S_4$ | + | $Z_1$ | $Z_2$ | $Z_3$ | $Z_4$ | $I$ | $I$ | $I$ |
| $S_5$ | + | $I$ | $Z_2$ | $Z_3$ | $I$ | $Z_5$ | $Z_6$ | $I$ |
| $S_6$ | + | $I$ | $I$ | $Z_3$ | $Z_4$ | $I$ | $Z_6$ | $Z_7$ |
| $\overline{X}$ | + | $I$ | $I$ | $I$ | $I$ | $X_5$ | $X_6$ | $X_7$ |
| $\overline{Y}$ | − | $I$ | $I$ | $I$ | $I$ | $Y_5$ | $Y_6$ | $Y_7$ |
| $\overline{Z}$ | + | $I$ | $I$ | $I$ | $I$ | $Z_5$ | $Z_6$ | $Z_7$ |
| $\overline{H}$ | + | $H_1$ | $H_2$ | $H_3$ | $H_4$ | $H_5$ | $H_6$ | $H_7$ |
| $\overline{S}$ | + | $S_1^\dagger$ | $S_2^\dagger$ | $S_3^\dagger$ | $S_4^\dagger$ | $S_5^\dagger$ | $S_6^\dagger$ | $S_7^\dagger$ |



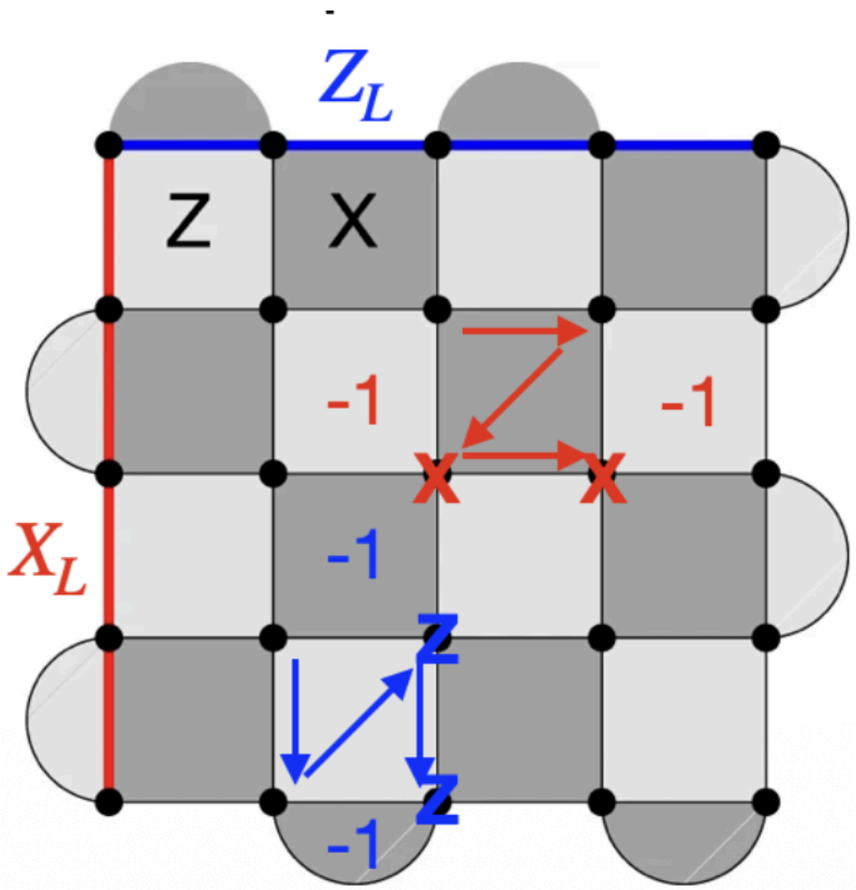


# Hook errors and error propagation

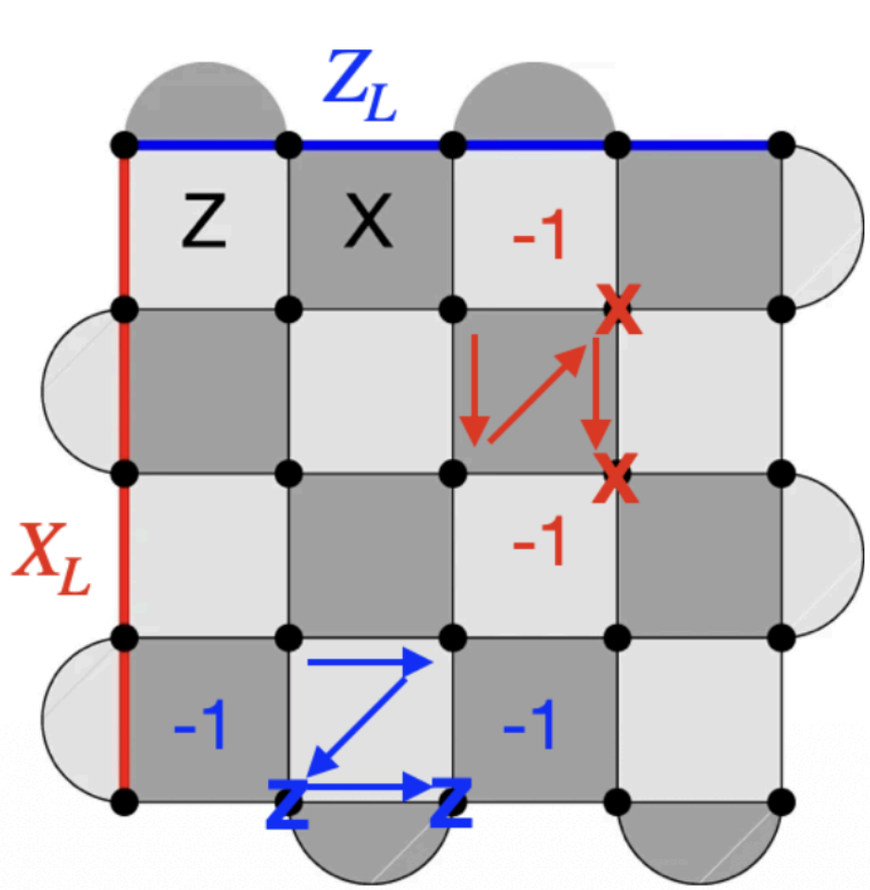


# Hook errors and error propagation

The surface code is a special case. If you order the syndrome extraction gates carefully, you can ensure that hook errors don't stack in the direction that is problematic:




Good order — hook errors perpendicular to respective logical operators




Bad order — hook errors parallel to respective logical operators

Figure credit: Jeff Thompson

# Error thresholds

| description | $[[n, k, d]]$ | threshold (units of $10^{-3}$) | transversal gate set |
|---|---|---|---|
| smallest code[140] | $[[5, 1, 3]]$ | 0.03 [150] | $PH, M_3$[27] |
| Steane/color | $[[7, 1, 3]]$ | $0.001 - 0.1$ | Clifford group |
| Shor | $[[9, 1, 3]]$ | 0.2 [150, 151] | CNOT[27] |
| 4.8.8 color | $[[17, 1, 5]]$ | 0.8 [148] | Clifford group |
| small surface code | $[[17, 1, 3]]$ | 0.8 [152] | Pauli group |
| Golay | $[[23, 1, 7]]$ | 1.3 [153] | Clifford group |

Table 12.3: Characteristics of some small to medium size codes. Each $[[n, k, d]]$ code has $k$ logical qubits encoded in $n$ physical qubits and has distance $d$.

# Error thresholds



**Table 8.** Various threshold estimates, associated geometric constraints and architectural considerations when performing the calculation. We have ordered the table with respect to the level of architectural detail that has been developed compatible with the threshold result. Note that this is not an exhaustive list.

| Relevant work | Code | Threshold | Geometric constraints | Architectural context |
|---|---|---|---|---|
| [ABO08] | [[7, 1, 3]] | $O(10^{-6})$ | A.I.[a] | None |
| [Got97] | [[7, 1, 3]] | $O(10^{-4} - 10^{-6})$ | A.I. | None |
| [KLZ96] | [[7, 1, 3]] | $O(10^{-6})$ | A.I. | None |
| [PR12] | [[23, 1, 7]] | $O(10^{-3})$ | A.I. | None |
| [Kni05] | 4- and 6-qubit detection | $O(10^{-2})$ | A.I. | None |
| [BAO+12] | Toric code | 18% | 2D NN[b] array, P.B.C.[c] | None, theoretical upper bound[d] |
| [MCT+04] | [[7, 1, 3]][e] | $O(10^{-4})$ | A.I. with movement penalty | Specific to ion-traps |
| [SFH08] | [[7, 1, 3]] | $O(10^{-6})$ | Bilinear NN array | Kane P : Si system [Kan98] |
| [SBF+06] | [[7, 1, 3]] | $O(10^{-7})$ | Variable width NN array | General NN systems |
| [SDT07] | [[7, 1, 3]] | $O(10^{-5})$ | 2D NN arrays | General NN systems |
| [FTY+07] | [[7, 1, 3]] | $O(10^{-6})$ | Bilinear NN array | Superconducting qubits |
| [BKSO05] | [[7, 1, 3]] | $O(10^{-9})$ | A.I. with movement penalty | Ion-traps |
| [RHG07] | Topological cluster | $O(10^{-2} - 10^{-3})$ | 3D NN array | Photonic qubits [DFS+09] |
| [WFSH10] | Surface codes | $O(10^{-2} - 10^{-3})$ | 2D NN array | Quantum dots, diamond [JMF+12, YJG+12] |

[a] Arbitrary interactions.
[b] Nearest neighbour.
[c] Periodic boundary conditions.
[d] This result is calculated assuming perfect quantum gates and is known as the code capacity.
[e] Only a preliminary estimate.

# Gottesman-Knill theorem

A quantum circuit using only the following elements can be simulated efficiently on a classical computer:

1) Preparation of qubits in computational-basis states.

2) Clifford gates (generated by the Hadamard gate, controlled NOT gate, and phase gate S ).

3) Measurements in the computational basis.

# The Clifford group $C_1$

$$H = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$

$$S = \begin{bmatrix} 1 & 0 \\ 0 & e^{i\frac{\pi}{2}} \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & i \end{bmatrix} = \sqrt{Z}$$

$$\mathrm{CNOT} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}.$$

| index | $x$ axis | $y$ axis | $z$ axis | $U$ | index | $x$ axis | $y$ axis | $z$ axis | $U$ |
|---|---|---|---|---|---|---|---|---|---|
| 1 | $I$ | $I$ | $I$ | $\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$ | 2 | $I$ | $I$ | $\pi/2$ | $e^{-i\pi/4}\begin{pmatrix} 1 & 0 \\ 0 & i \end{pmatrix}$ |
| 3 | $I$ | $I$ | $\pi$ | $-i\begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$ | 4 | $I$ | $I$ | $-\pi/2$ | $e^{i\pi/4}\begin{pmatrix} 1 & 0 \\ 0 & -i \end{pmatrix}$ |
| 5 | $I$ | $\pi$ | $I$ | $-1\begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix}$ | 6 | $I$ | $\pi$ | $\pi/2$ | $-e^{i\pi/4}\begin{pmatrix} 0 & 1 \\ i & 0 \end{pmatrix}$ |
| 7 | $\pi$ | $I$ | $I$ | $-i\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$ | 8 | $\pi$ | $I$ | $\pi/2$ | $e^{-i\pi/4}\begin{pmatrix} 0 & 1 \\ -i & 0 \end{pmatrix}$ |
| 9 | $\pi$ | $\pi/2$ | $I$ | $\frac{-i}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$ | 10 | $I$ | $-\pi/2$ | $I$ | $\frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ -1 & 1 \end{pmatrix}$ |
| 11 | $\pi/2$ | $I$ | $\pi/2$ | $\frac{e^{-i\pi/4}}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ -i & i \end{pmatrix}$ | 12 | $\pi/2$ | $\pi$ | $\pi/2$ | $-\frac{e^{i\pi/4}}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ i & -i \end{pmatrix}$ |
| 13 | $\pi$ | $-\pi/2$ | $I$ | $\frac{i}{\sqrt{2}}\begin{pmatrix} 1 & -1 \\ -1 & -1 \end{pmatrix}$ | 14 | $-\pi/2$ | $I$ | $\pi/2$ | $\frac{e^{-i\pi/4}}{\sqrt{2}}\begin{pmatrix} 1 & -1 \\ i & i \end{pmatrix}$ |
| 15 | $I$ | $\pi/2$ | $I$ | $\frac{1}{\sqrt{2}}\begin{pmatrix} 1 & -1 \\ 1 & 1 \end{pmatrix}$ | 16 | $-\pi/2$ | $\pi$ | $\pi/2$ | $\frac{e^{i\pi/4}}{\sqrt{2}}\begin{pmatrix} 1 & -1 \\ -i & -i \end{pmatrix}$ |
| 17 | $-\pi/2$ | $-\pi/2$ | $I$ | $\frac{e^{-i\pi/4}}{\sqrt{2}}\begin{pmatrix} 1 & i \\ -1 & i \end{pmatrix}$ | 18 | $-\pi/2$ | $\pi/2$ | $I$ | $\frac{e^{i\pi/4}}{\sqrt{2}}\begin{pmatrix} 1 & i \\ 1 & -i \end{pmatrix}$ |
| 19 | $-\pi/2$ | $\pi$ | $I$ | $\frac{i}{\sqrt{2}}\begin{pmatrix} 1 & i \\ -i & -1 \end{pmatrix}$ | 20 | $-\pi/2$ | $I$ | $I$ | $\frac{1}{\sqrt{2}}\begin{pmatrix} 1 & i \\ i & 1 \end{pmatrix}$ |
| 21 | $\pi/2$ | $-\pi/2$ | $I$ | $\frac{e^{i\pi/4}}{\sqrt{2}}\begin{pmatrix} 1 & -i \\ -1 & -i \end{pmatrix}$ | 22 | $\pi/2$ | $I$ | $I$ | $\frac{1}{\sqrt{2}}\begin{pmatrix} 1 & -i \\ -i & 1 \end{pmatrix}$ |
| 23 | $\pi/2$ | $\pi$ | $I$ | $\frac{-i}{\sqrt{2}}\begin{pmatrix} 1 & -i \\ i & -1 \end{pmatrix}$ | 24 | $\pi/2$ | $\pi/2$ | $I$ | $\frac{e^{-i\pi/4}}{\sqrt{2}}\begin{pmatrix} 1 & -i \\ 1 & i \end{pmatrix}$ |

# STIM, crumbl, etc

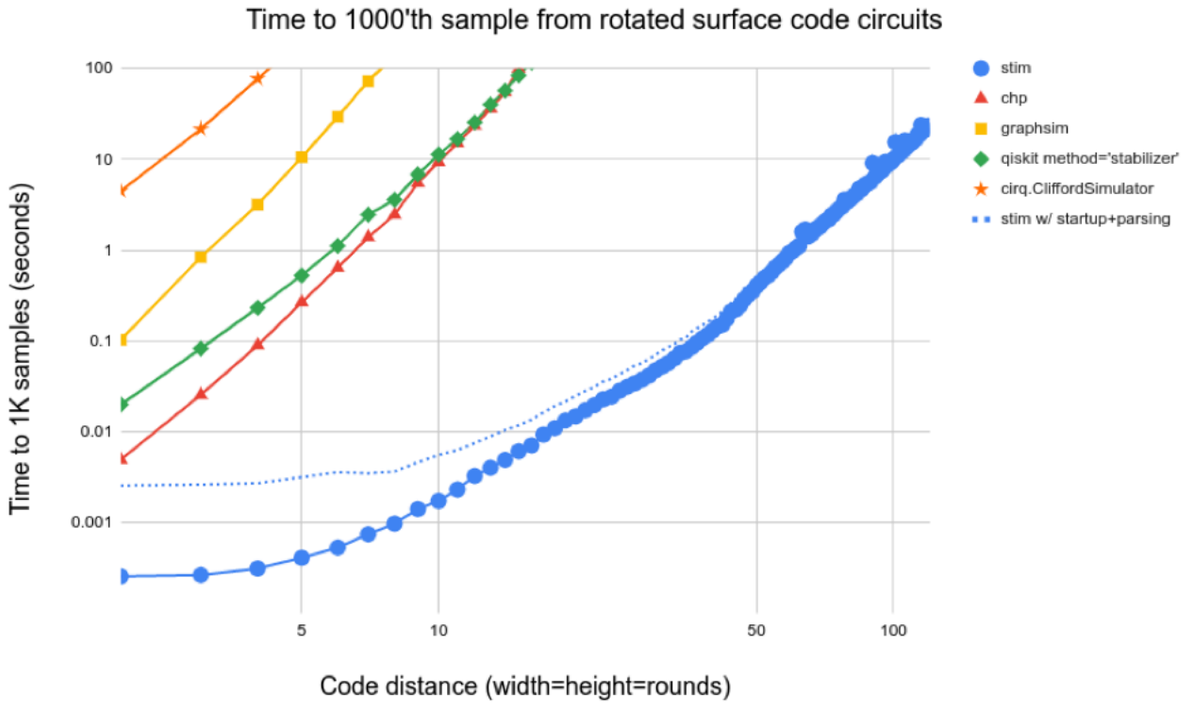
## Stim: a fast stabilizer circuit simulator

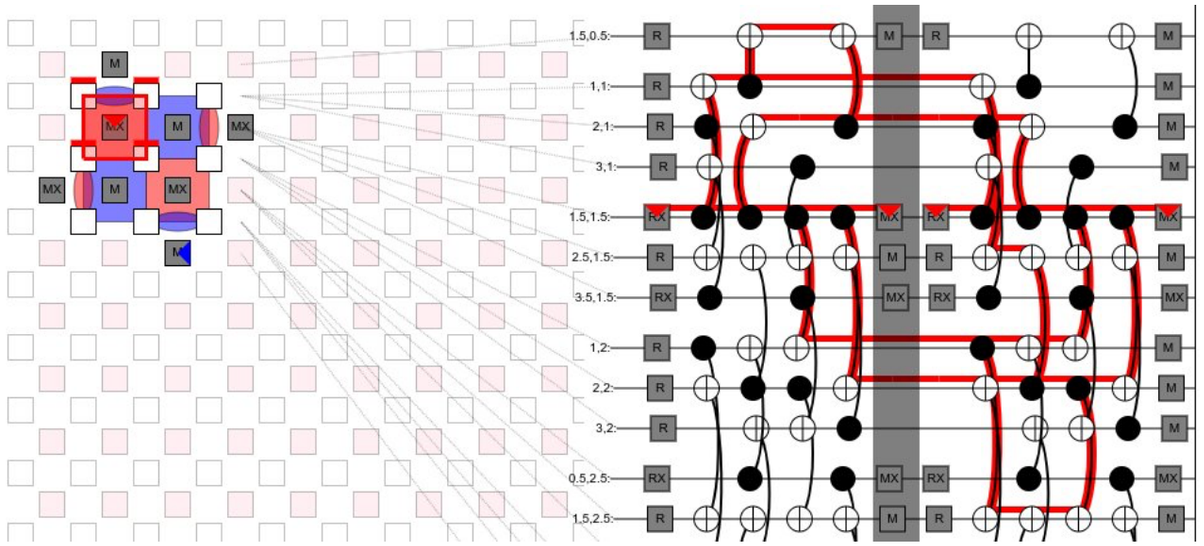
Craig Gidney

Google Inc., Santa Barbara, California 93117, USA

June 21, 2021

This paper presents “Stim", a fast simulator for quantum stabilizer circuits. The paper explains how Stim works and compares it to existing tools. With no foreknowledge, Stim can analyze a distance 100 surface code circuit (20 thousand qubits, 8 million gates, 1 million measurements) in 15 seconds and then begin sampling full circuit shots at a rate of 1 kHz. Stim uses a stabilizer tableau representation, similar to Aaronson and Gottesman's CHP simulator, but with three main improvements. First, Stim improves the asymptotic complexity of deterministic measurement from quadratic to linear by tracking the *inverse* of the circuit's stabilizer tableau. Second, Stim improves the constant factors of the algorithm by using a cache-friendly data layout and 256 bit wide SIMD instructions. Third, Stim only uses expensive stabilizer tableau simulation to create an initial reference sample. Further samples are collected in bulk by using that sample as a reference for batches of Pauli frames propagating through the circuit.



# Gottesman-Knill theorem

A quantum circuit using only the following elements can be simulated efficiently on a classical computer:

1) Preparation of qubits in computational-basis states.

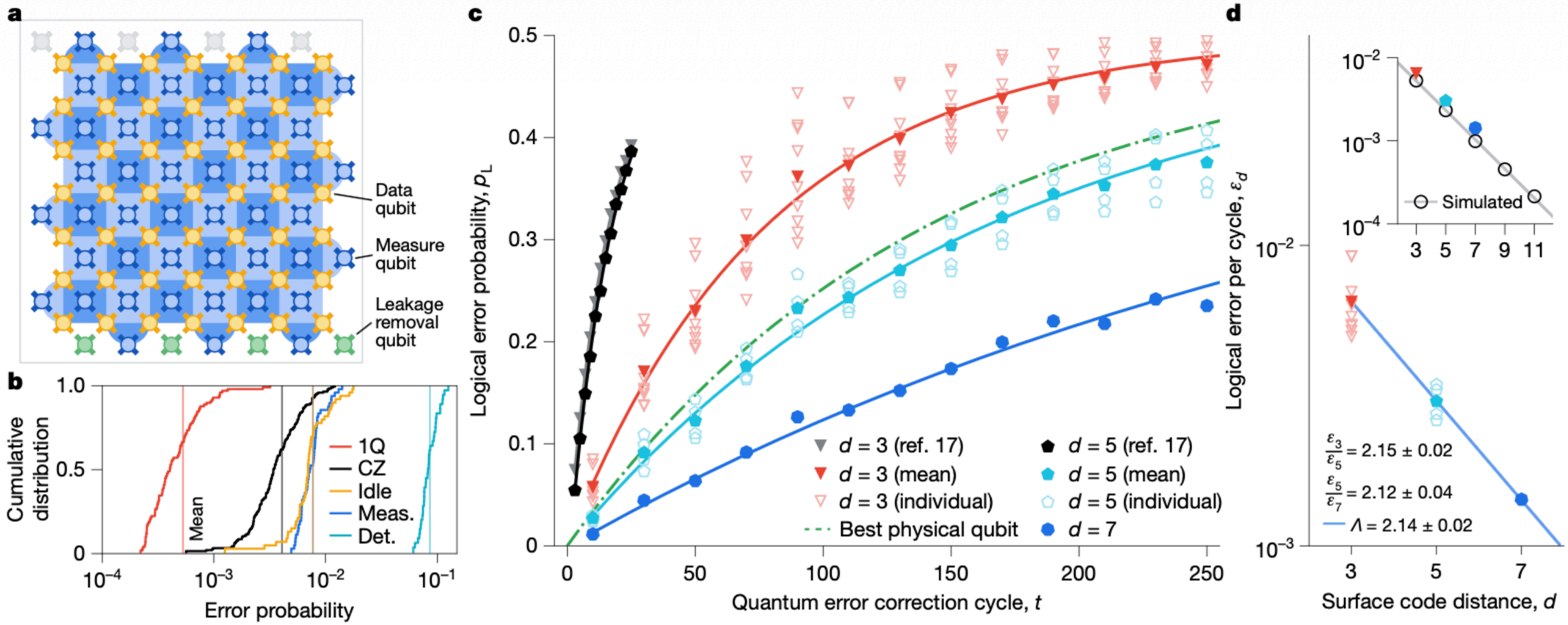
2) Clifford gates (generated by the Hadamard gate, controlled NOT gate, and phase gate S ).

3) Measurements in the computational basis.

This is actually pretty surprising! The Clifford gates will still give you superpositions and entanglement…

How is this compatible with the exponential advantage provided by quantum computers?

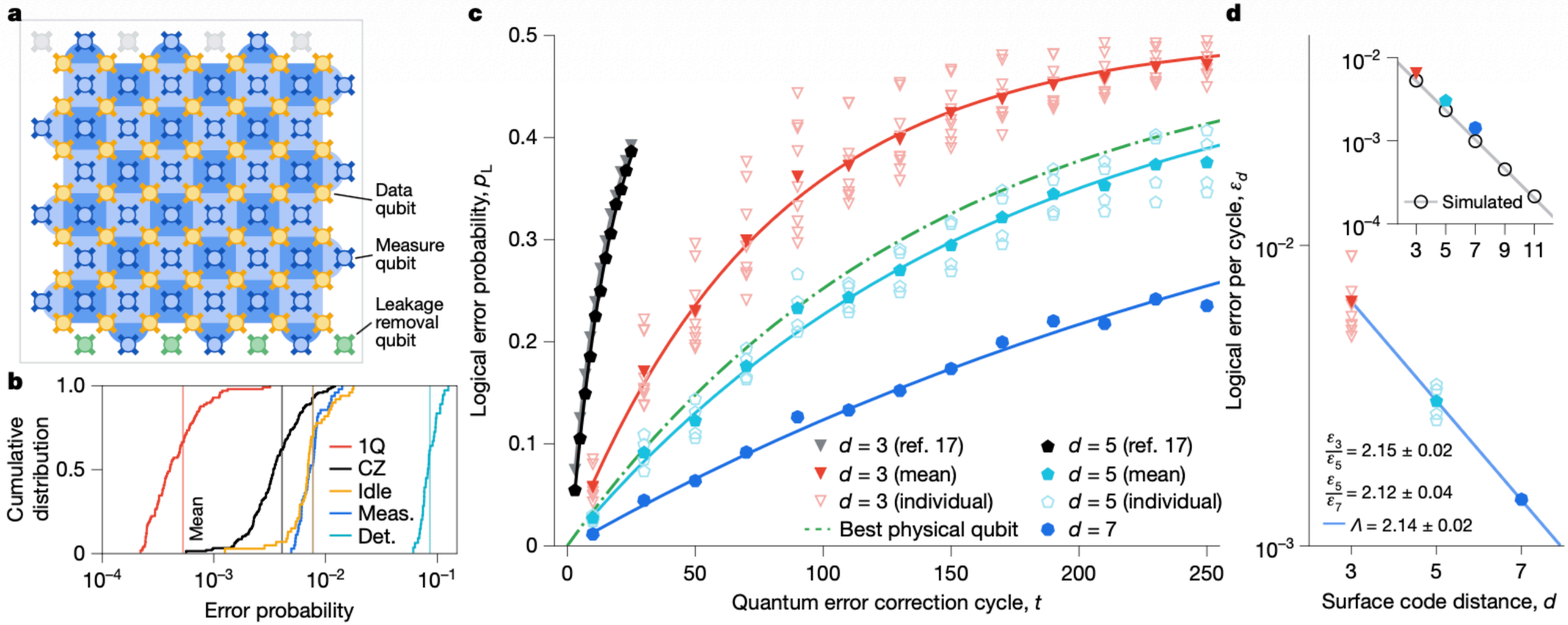
# Recent Google Surface Code QEC results




105 total qubits

What is still missing?

# Recent Google Surface Code QEC results




105 total qubits

What is still missing?

Logical gates!

# Logical gates

Suppose we want to implement a specific unitary U on a logical qubit. We could do this…

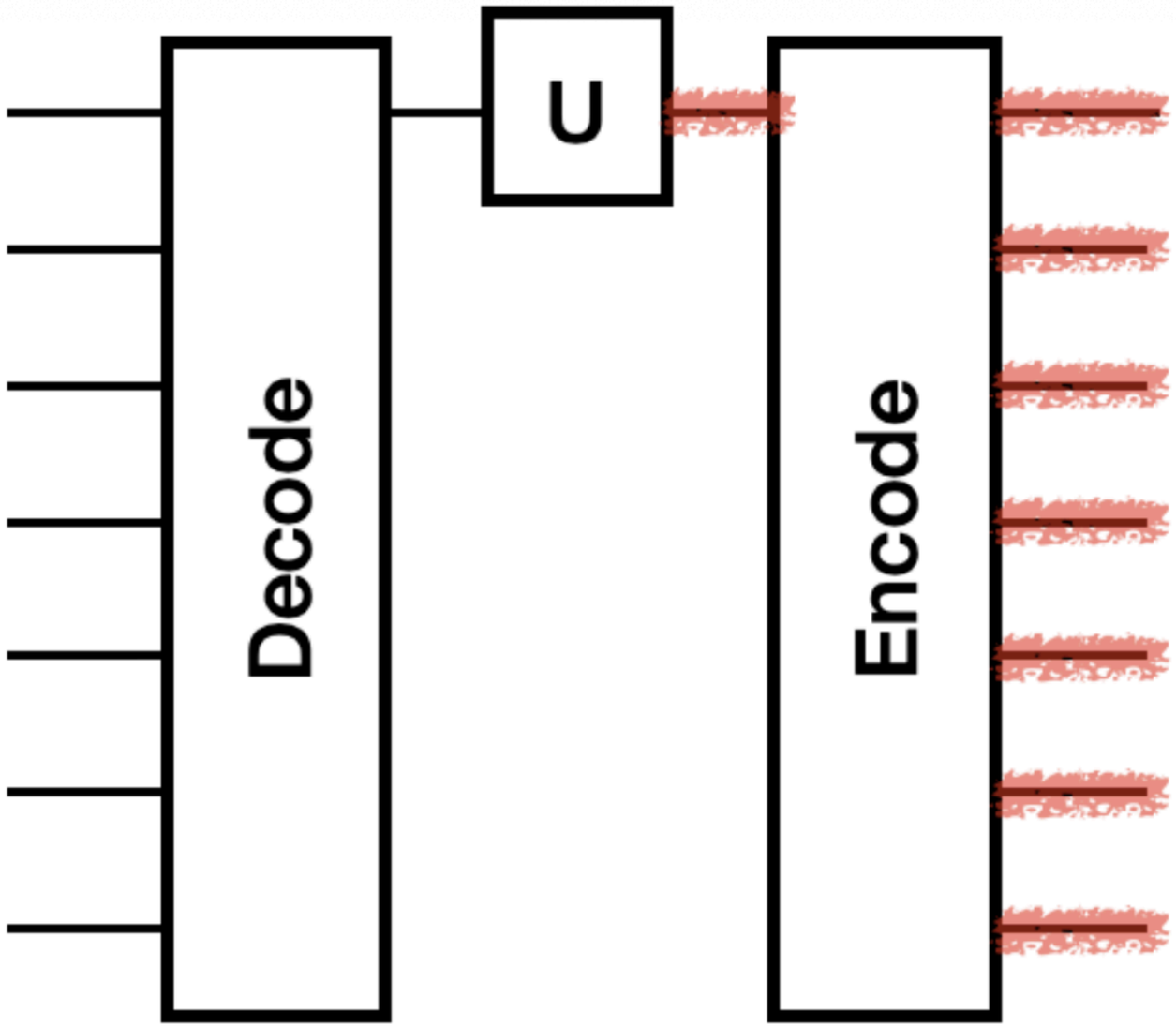
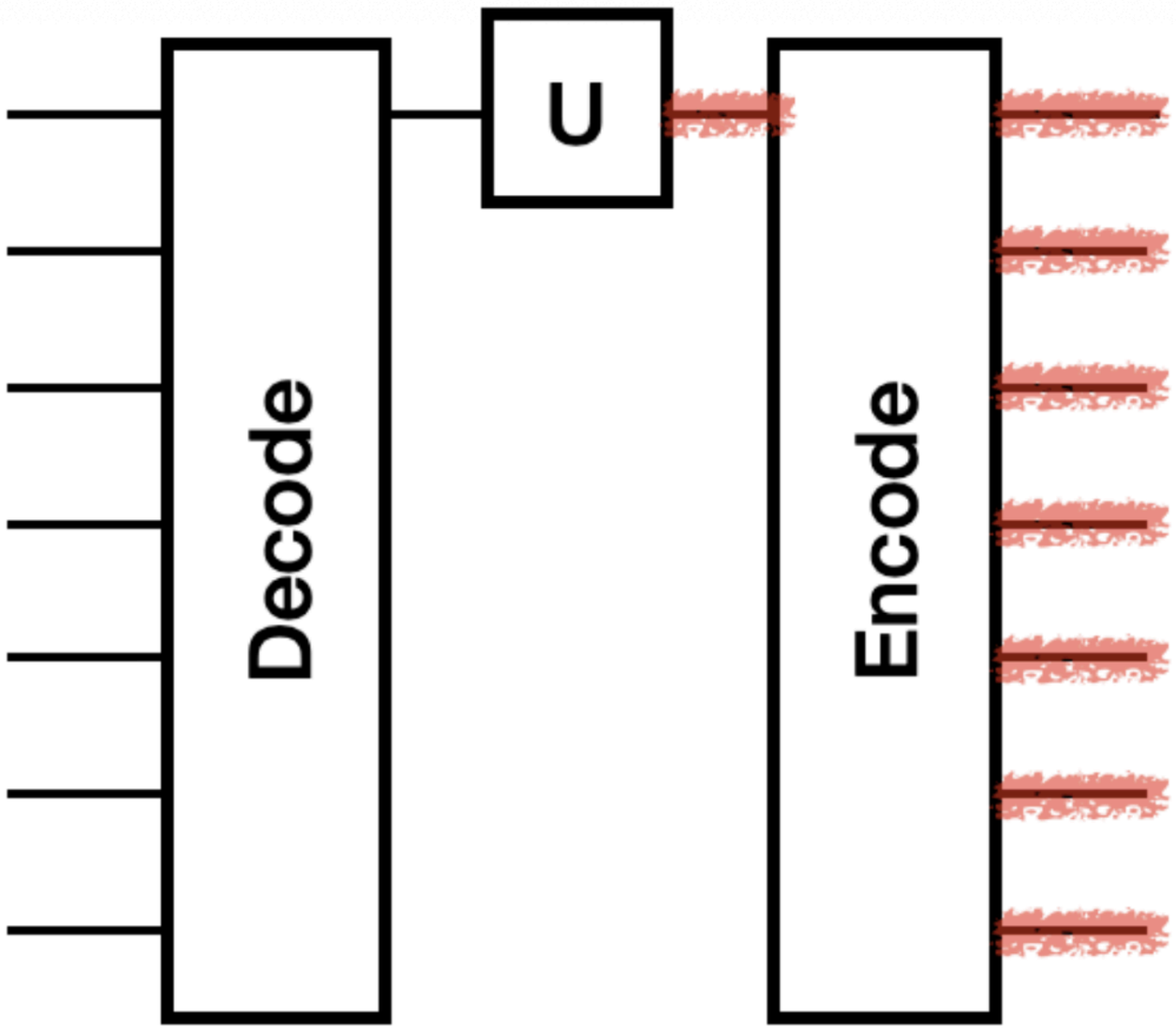



Figure credit: Jeff Thompson

# Logical gates

Suppose we want to implement a specific unitary U on a logical qubit. We could do this…

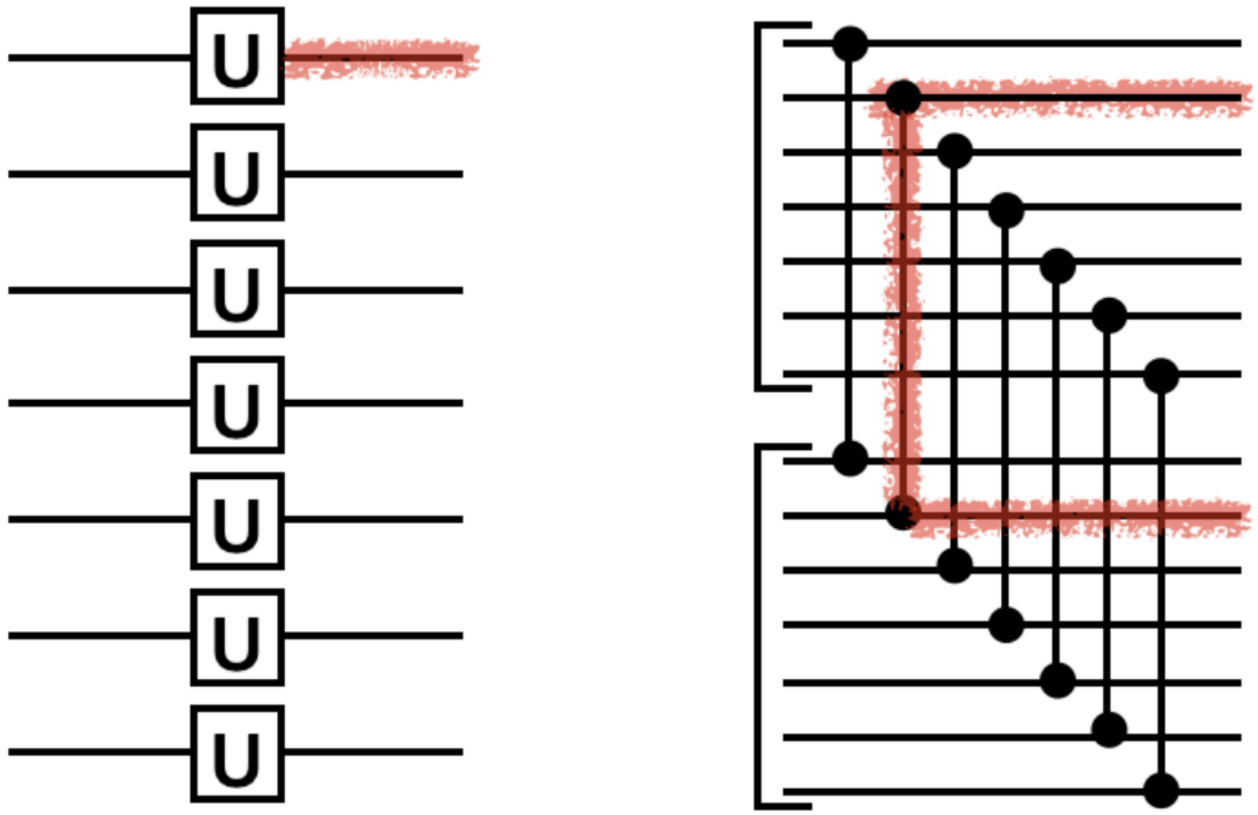



For obvious reasons, this is a terrible idea.

Figure credit: Jeff Thompson

# Transversal logical gates

Instead, it would be much better if we could implement our logical gates like this:




This is called a transverse, or transversal, gate.

Figure credit: Jeff Thompson

# Eastin-Knill Theorem

Unfortunately, the Eastin-Knill theorem says that you can't have an error correcting code with universal set of transversal gates.

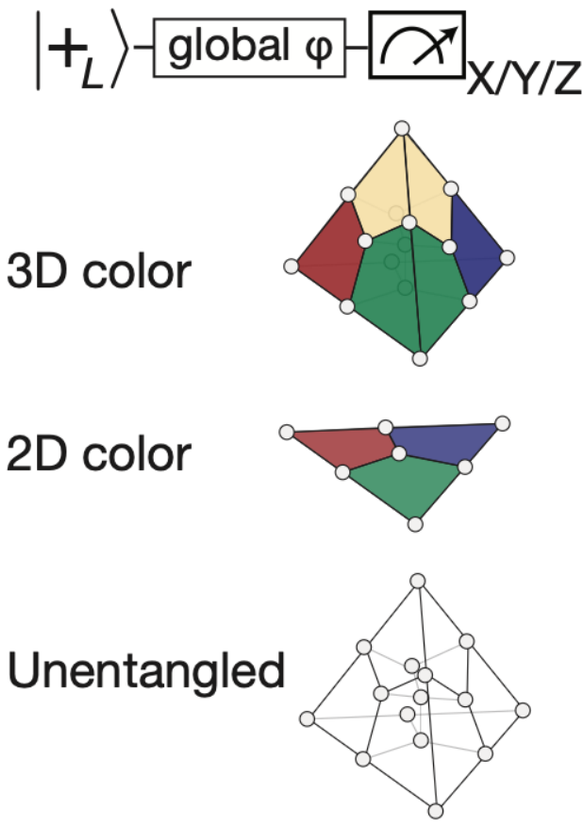
Intuition: a universal gate set includes infinitesmal rotations like $e^{i\varepsilon X}$

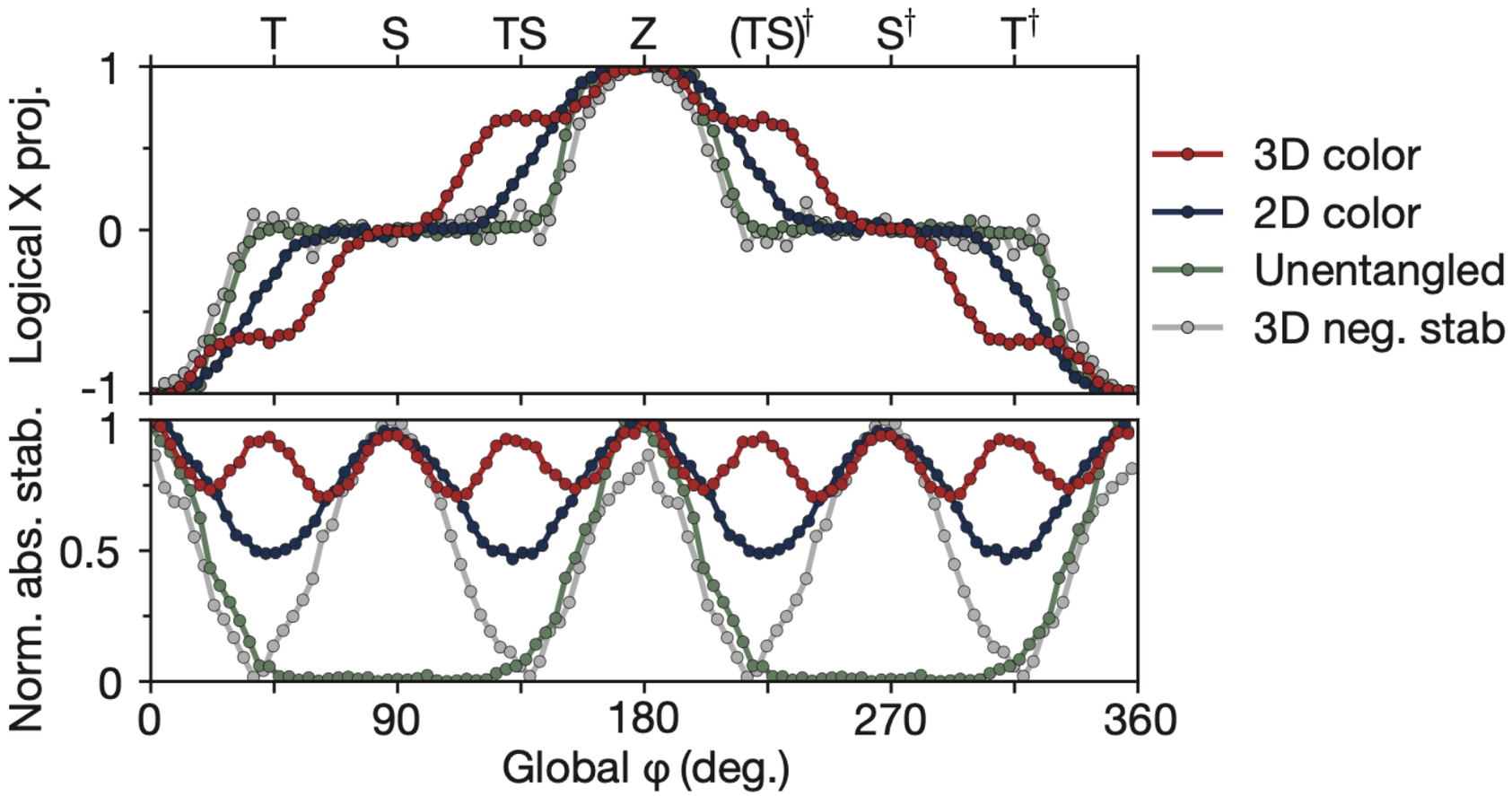
But it is clear that such rotations look like X errors on every qubit, and will create superpositions of many stabilizer eigenspaces!

The only way for a continuously generated logical gate to work is for all of these amplitudes to destructively interfere and leave us back in logical codespace.

This can only happen at special, finite rotation angles. If it held for all angles, it doesn't seem like we would be able to detect errors.

# Logical gates



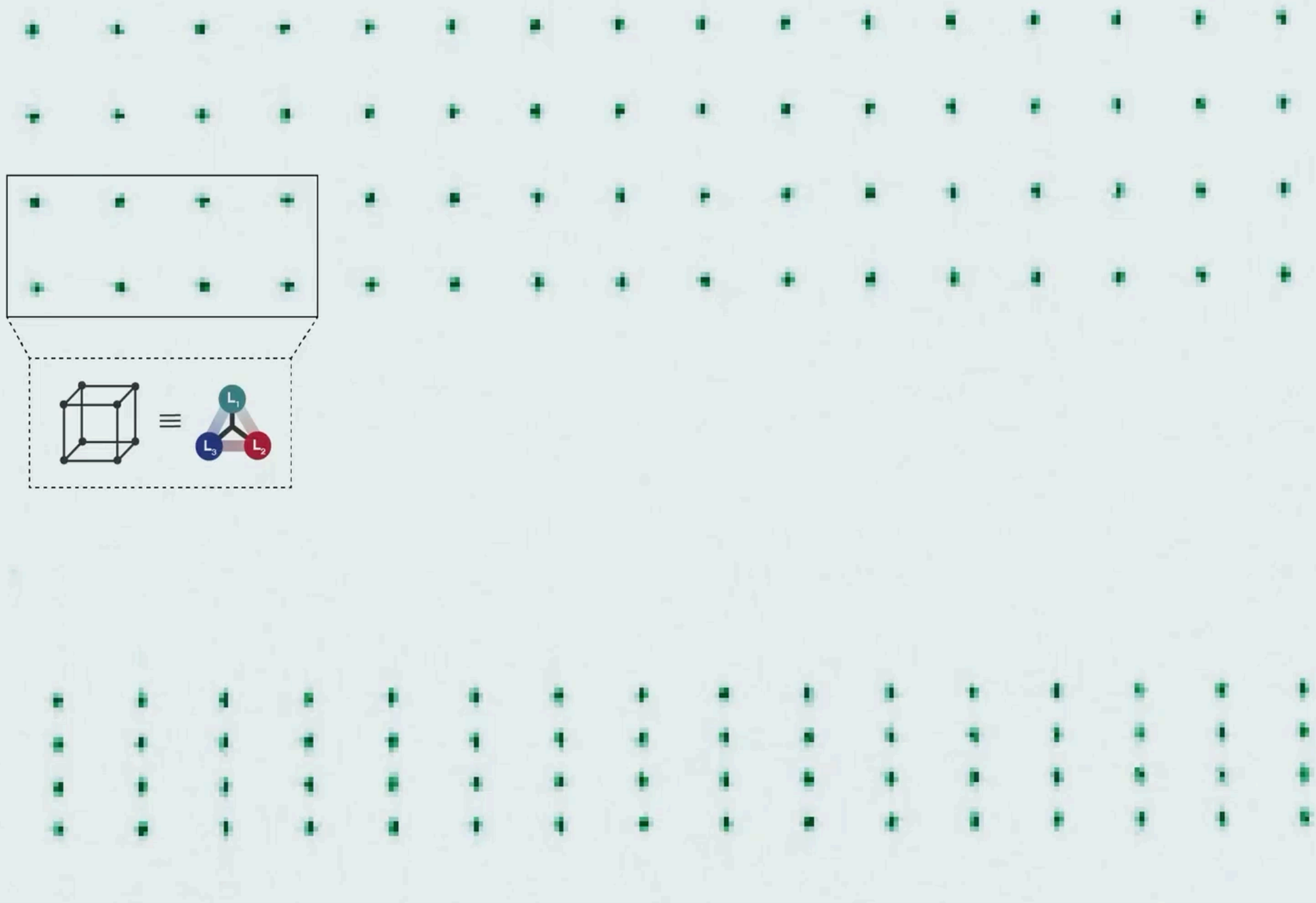
# Transversal logical gates

Eastin-Knill doesn't tell us which gates can be transverse and which ones can't be, and it varies depending on the given code:

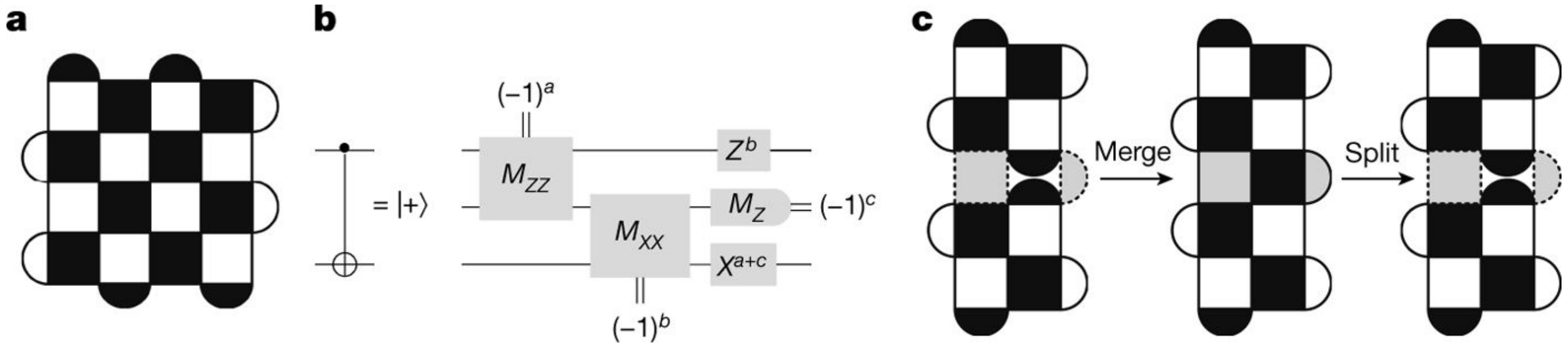
| description | $[[n, k, d]]$ | threshold (units of $10^{-3}$) | transversal gate set |
|---|---|---|---|
| smallest code[140] | $[[5, 1, 3]]$ | 0.03 [150] | $PH, M_3$[27] |
| Steane/color | $[[7, 1, 3]]$ | $0.001 - 0.1$ | Clifford group |
| Shor | $[[9, 1, 3]]$ | 0.2 [150, 151] | CNOT[27] |
| 4.8.8 color | $[[17, 1, 5]]$ | 0.8 [148] | Clifford group |
| small surface code | $[[17, 1, 3]]$ | 0.8 [152] | Pauli group |
| Golay | $[[23, 1, 7]]$ | 1.3 [153] | Clifford group |

Table 12.3: Characteristics of some small to medium size codes. Each $[[n, k, d]]$ code has $k$ logical qubits encoded in $n$ physical qubits and has distance $d$.

# Transversal CNOT gates



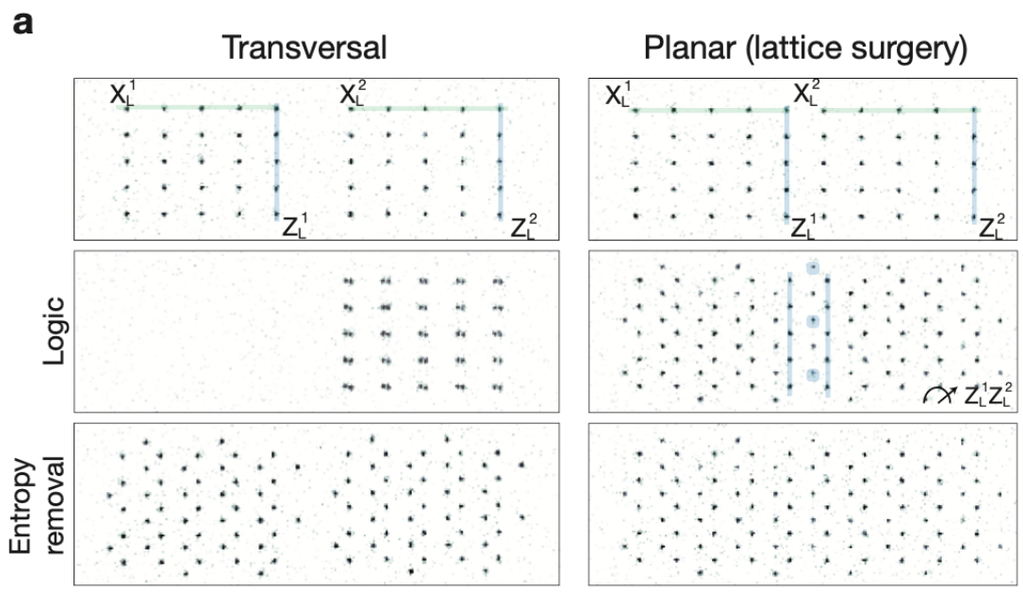
# Lattice surgery




**a**, One logical qubit is encoded in a surface code sheet consisting of $d^2$ physical qubits at the vertices of the lattice ($d=5$ here). Black faces represent $X$ parity checks and white faces represent $Z$ parity checks on the qubits. **b**, A CNOT circuit is equivalent to performing non-destructive $X \otimes X$ and $Z \otimes Z$ measurements for the control and target qubits, respectively, with an ancillary qubit. Lattice code surgery provides a method for non-destructively measuring a logical $Z \otimes Z$ or $X \otimes X$ operator between two encoded qubits. M$xx$, M$zz$ and M$z$ represent these measurements and $b$, $a$ and $c$ their respective outcomes. Initially, the ancilla is in the state $|+\rangle = (|0\rangle + |1\rangle)/\sqrt{2}$. **c**, Lattice code surgery between two surface code sheets realizes a logical $Z \otimes Z$ measurement. By measuring and merging the parity checks between two surface code sheets, we merge the two sheets into one. At the same time, we learn the value of the logical $Z \otimes Z$ operator because it is equal to the product of the newly measured (grey) faces. The two sheets can then be split again for further operations.

Campbell, E., Terhal, B. & Vuillot, C. Roads towards fault-tolerant universal quantum computation. *Nature* **549**, 172–179 (2017)

# Lattice surgery vs transversal CNOT



# S and T gates

The surface code lacks transversal S and T gates, which are necessary for a universal gate set:

$$S = \begin{bmatrix} 1 & 0 \\ 0 & e^{i\frac{\pi}{2}} \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & i \end{bmatrix} = \sqrt{Z}$$

$$T = \begin{bmatrix} 1 & 0 \\ 0 & e^{i\frac{\pi}{4}} \end{bmatrix} = \sqrt{S} = \sqrt[4]{Z}.$$

# S and T gates

The surface code lacks transversal S and T gates, which are necessary for a universal gate set:

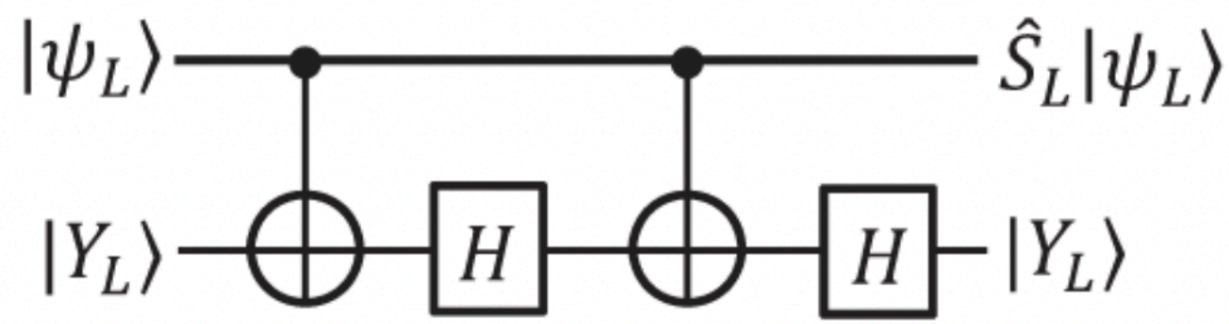
$$S = \begin{bmatrix} 1 & 0 \\ 0 & e^{i\frac{\pi}{2}} \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & i \end{bmatrix} = \sqrt{Z}$$

$$T = \begin{bmatrix} 1 & 0 \\ 0 & e^{i\frac{\pi}{4}} \end{bmatrix} = \sqrt{S} = \sqrt[4]{Z}.$$

What can we do?

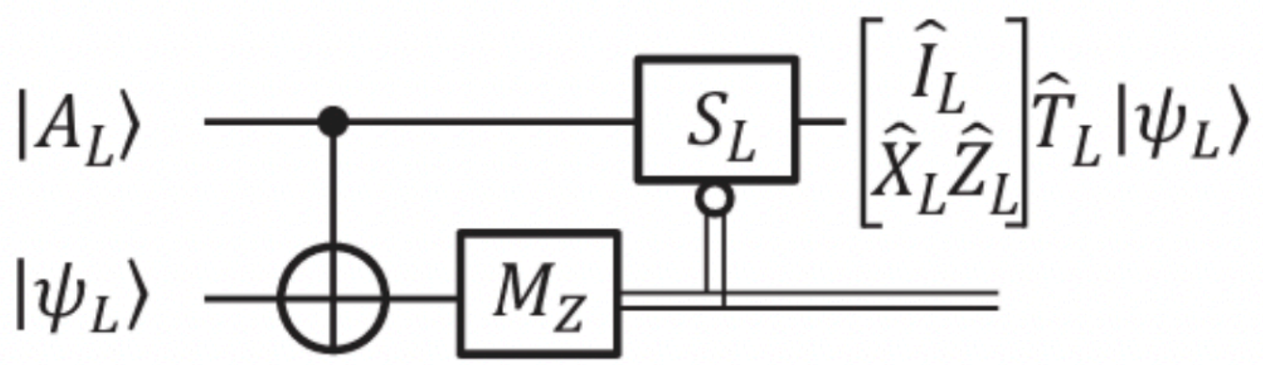
# *S* gate injection

$$|Y_L\rangle = |0_L\rangle + i|1_L\rangle$$



# *T* gate injection

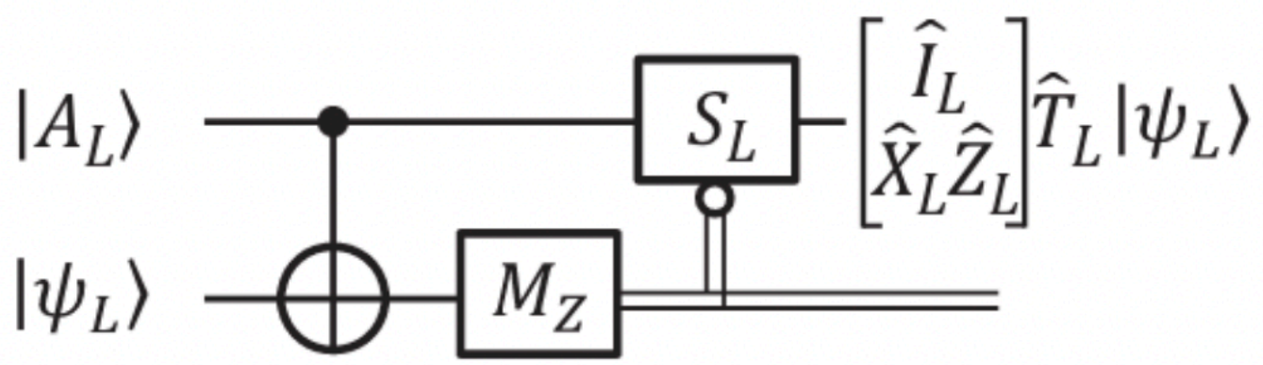
$$|A_L\rangle = |0_L\rangle + e^{i\pi/4}|1_L\rangle$$




$$\frac{1}{\sqrt{2}}(|0\rangle + e^{i\pi/4}|1\rangle)(\alpha|0\rangle + \beta|1\rangle)$$

# *T* gate injection

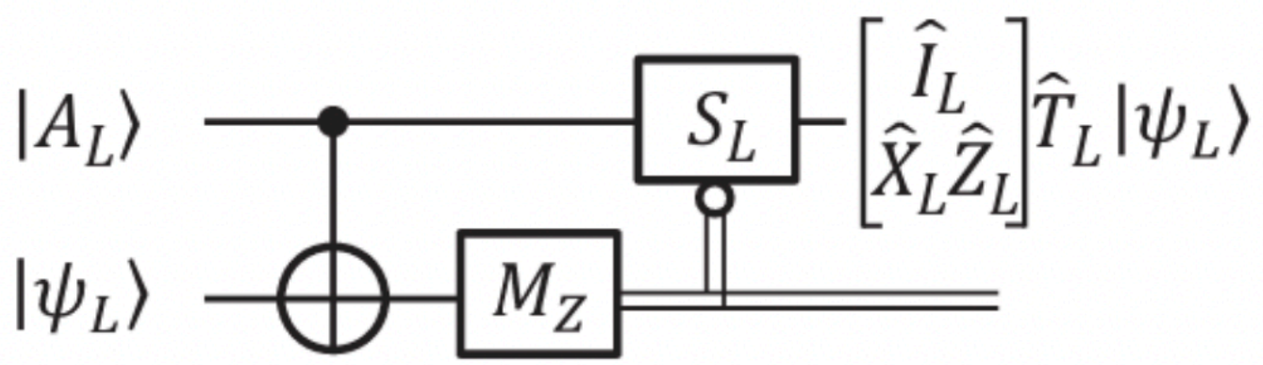
$$|A_L\rangle = |0_L\rangle + e^{i\pi/4}|1_L\rangle$$




$$\frac{1}{\sqrt{2}}(|0\rangle + e^{i\pi/4}|1\rangle)(\alpha|0\rangle + \beta|1\rangle)$$

# *T* gate injection

$$|A_L\rangle = |0_L\rangle + e^{i\pi/4}|1_L\rangle$$

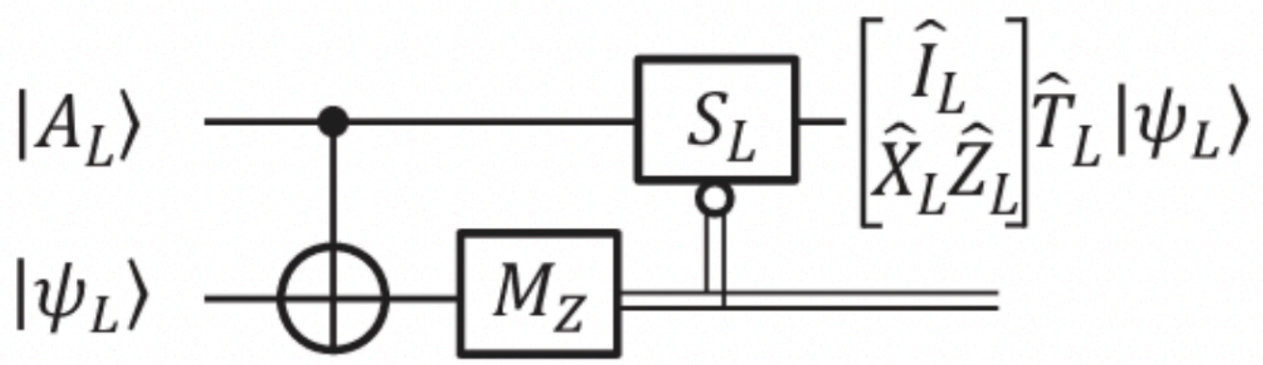



$$\frac{1}{\sqrt{2}}(|0\rangle + e^{i\pi/4}|1\rangle)(\alpha|0\rangle + \beta|1\rangle)$$

$$\xrightarrow{CNOT} \frac{1}{\sqrt{2}}(\alpha|00\rangle + \alpha e^{i\pi/4}|11\rangle + \beta|01\rangle + \beta e^{i\pi/4}|10\rangle)$$

# *T* gate injection

$$|A_L\rangle = |0_L\rangle + e^{i\pi/4}|1_L\rangle$$



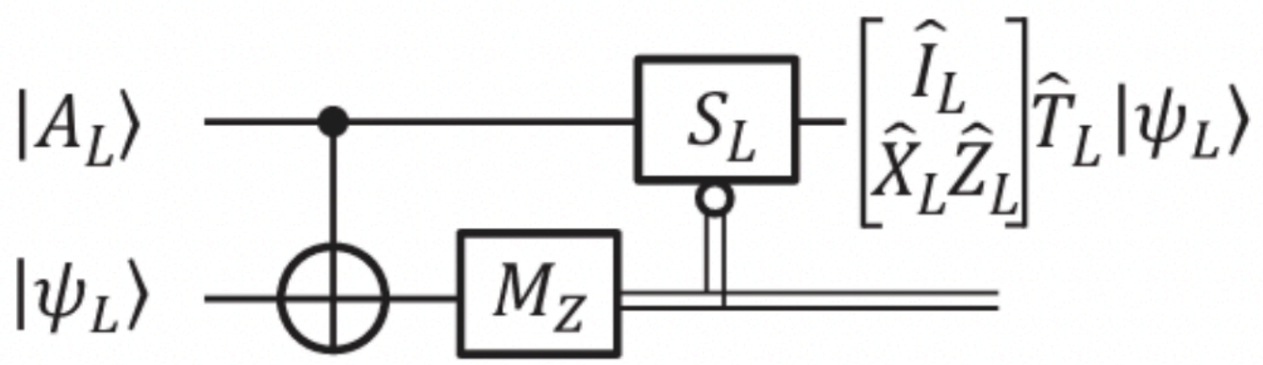

$$\frac{1}{\sqrt{2}}(|0\rangle + e^{i\pi/4}|1\rangle)(\alpha|0\rangle + \beta|1\rangle)$$

$$\xrightarrow{CNOT} \frac{1}{\sqrt{2}}(\alpha|00\rangle + \alpha e^{i\pi/4}|11\rangle + \beta|01\rangle + \beta e^{i\pi/4}|10\rangle)$$

$$= (\alpha|0\rangle + \beta e^{i\pi/4}|1\rangle)|0\rangle + (e^{i\pi/4}\alpha|0\rangle + \beta|1\rangle)|1\rangle$$

# *T* gate injection

$$|A_L\rangle = |0_L\rangle + e^{i\pi/4}|1_L\rangle$$




$$\frac{1}{\sqrt{2}}(|0\rangle + e^{i\pi/4}|1\rangle)(\alpha|0\rangle + \beta|1\rangle)$$

$$\xrightarrow{CNOT} \frac{1}{\sqrt{2}}(\alpha|00\rangle + \alpha e^{i\pi/4}|11\rangle + \beta|01\rangle + \beta e^{i\pi/4}|10\rangle)$$

$$= (\alpha|0\rangle + \beta e^{i\pi/4}|1\rangle)|0\rangle + (e^{i\pi/4}\alpha|0\rangle + \beta|1\rangle)|1\rangle$$

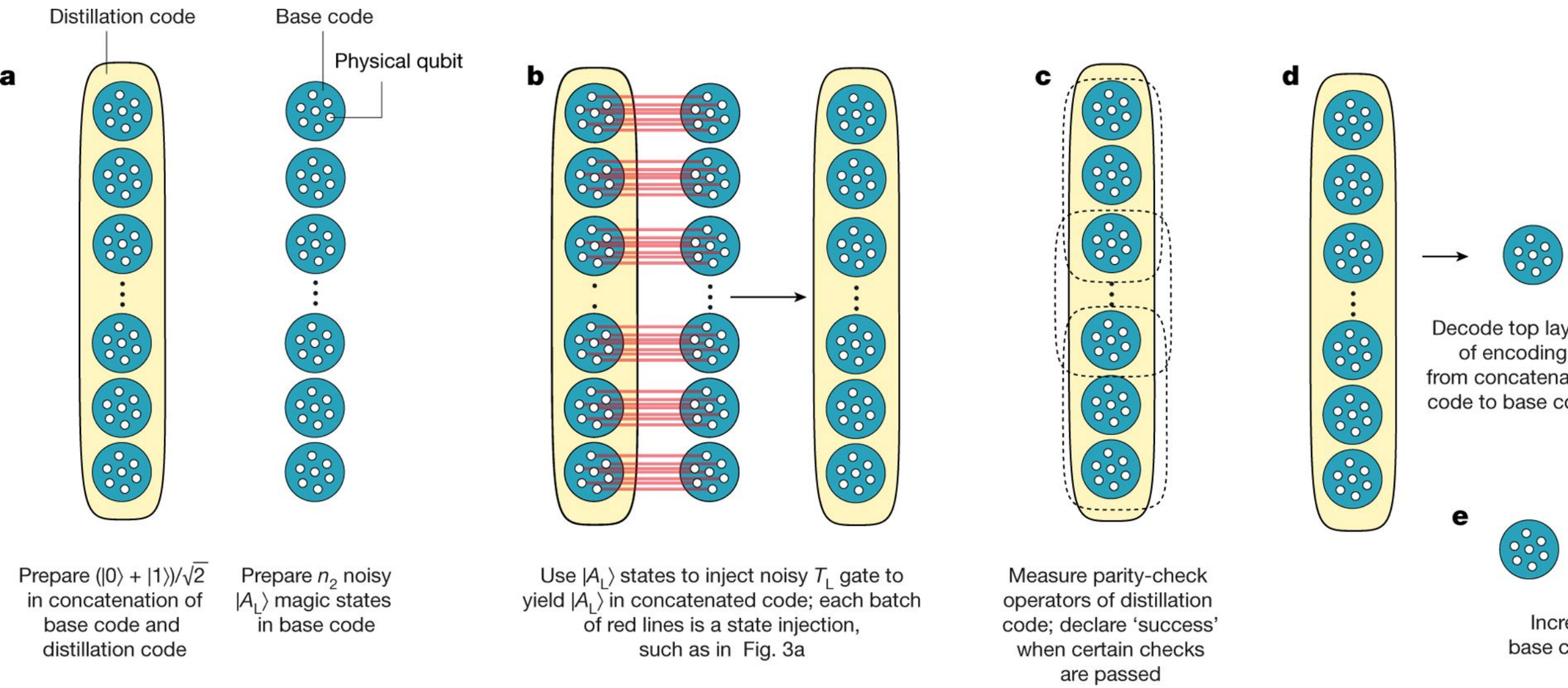
$$= T|\psi_L\rangle|0\rangle + iT^\dagger|\psi_L\rangle|1\rangle$$

# Magic state distillation

Of course, so far we haven't actually solved anything, because where did the initial T states come from?

# Magic state distillation

Of course, so far we haven't actually solved anything, because where did the initial T states come from?



# Magic state distillation

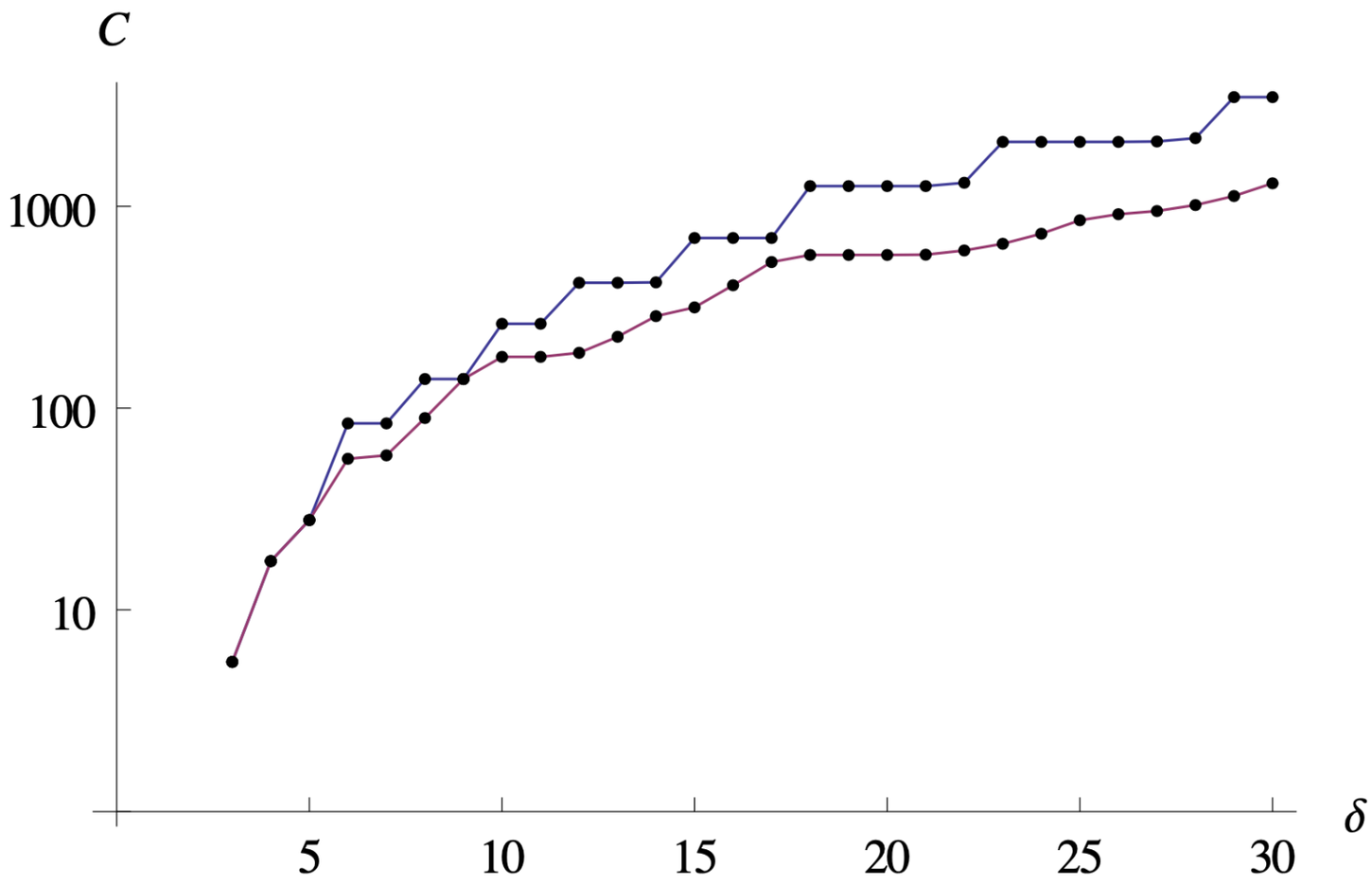
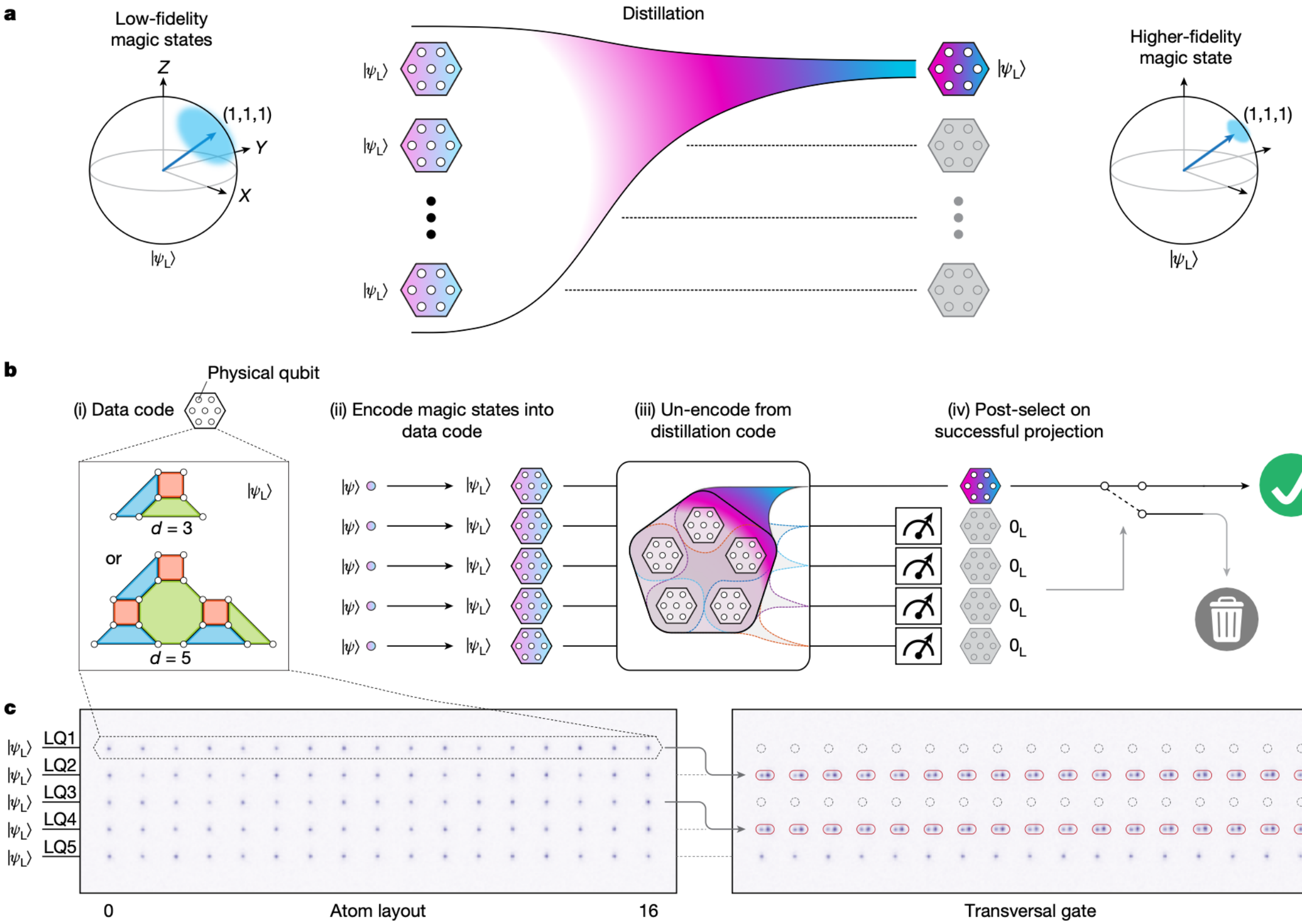
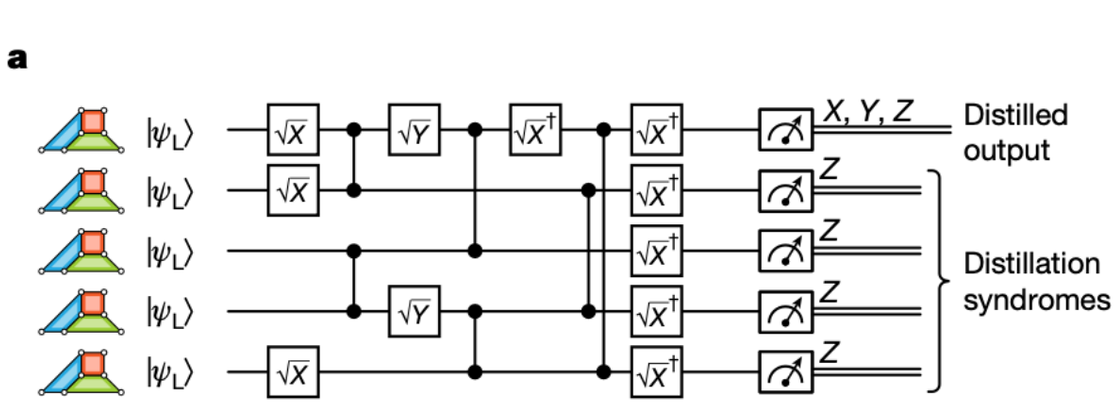


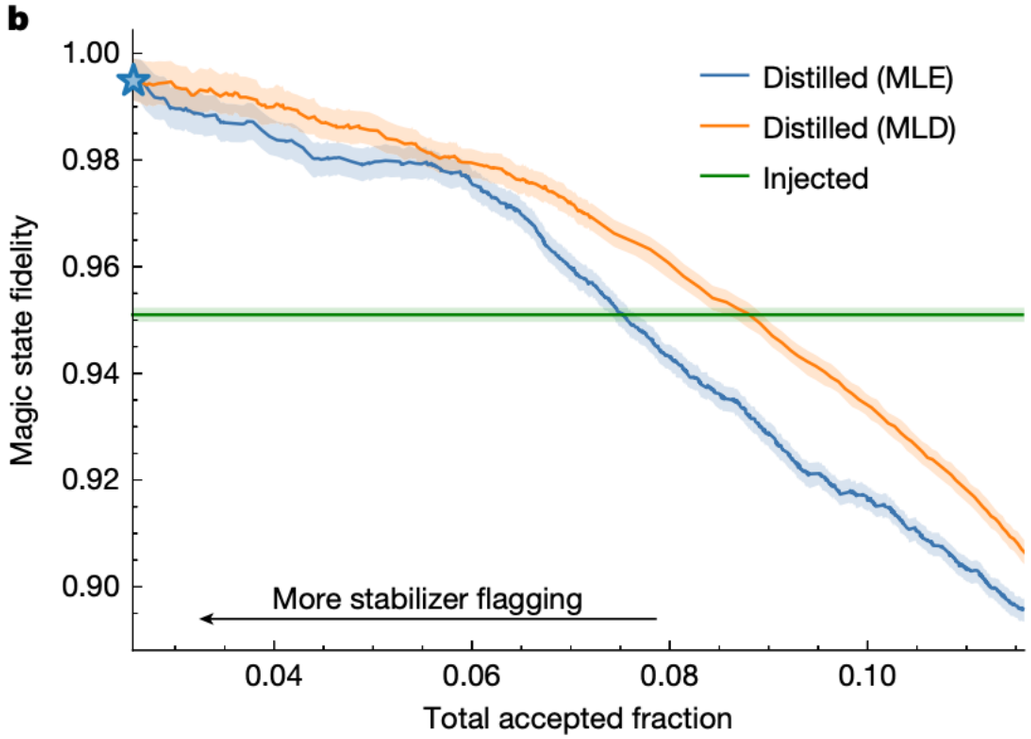

FIG. 3. Distillation cost $C$ as a function of the target error rate $\epsilon = 10^{-\delta}$ for a fixed input error rate $p = 0.01$. The upper curve is obtained from [17], and the lower curve from the optimization using the triorthogonal matrices $G(k)$. The lines are mere guide to eyes.

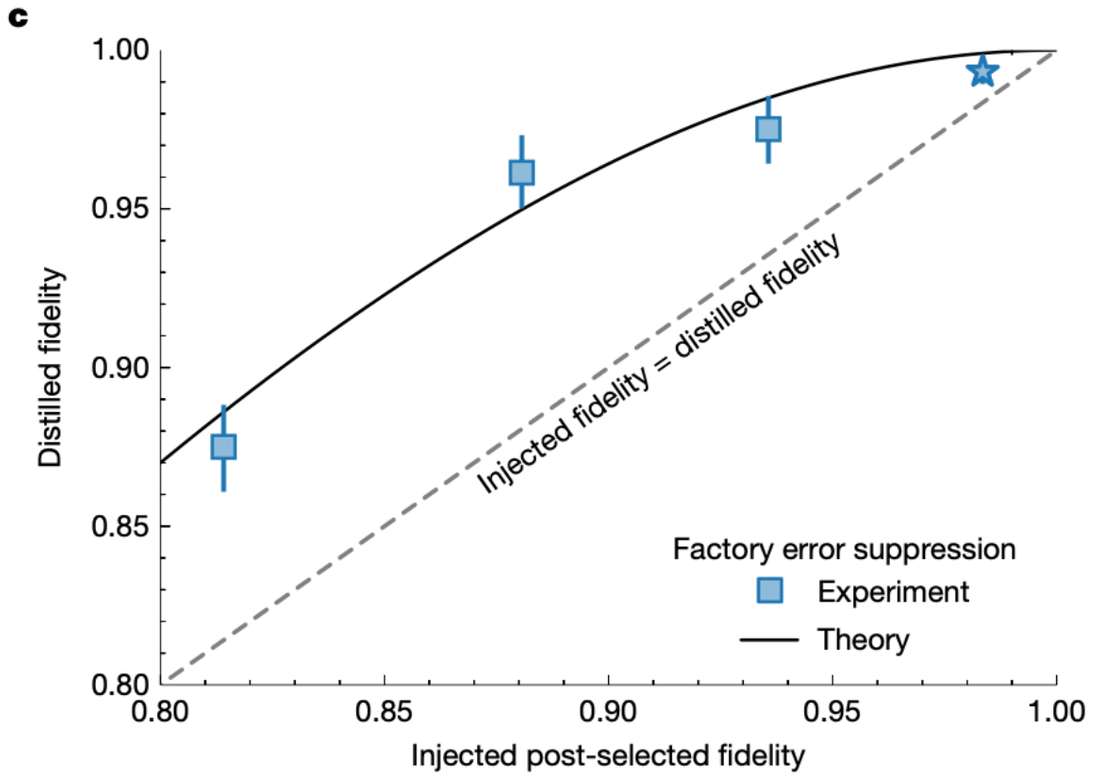
# Magic state distillation

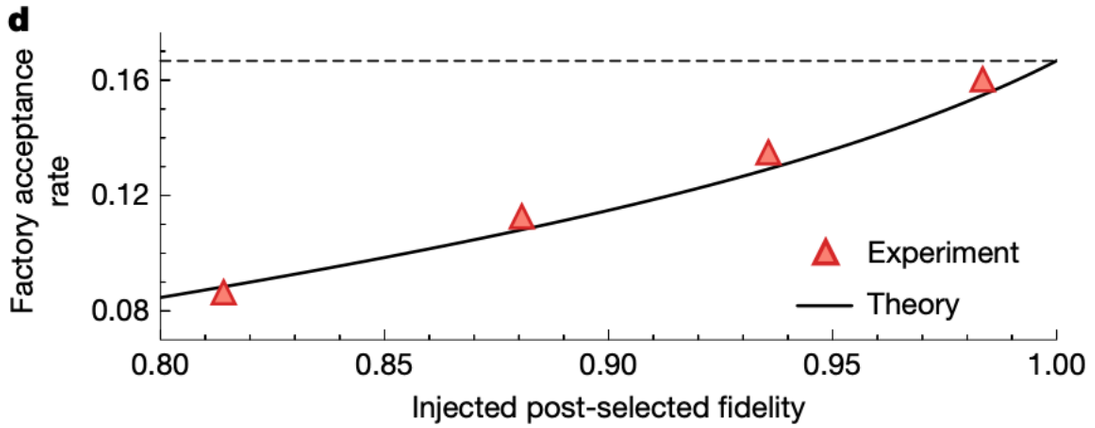


# Magic state distillation



# Logical gates with T state injection