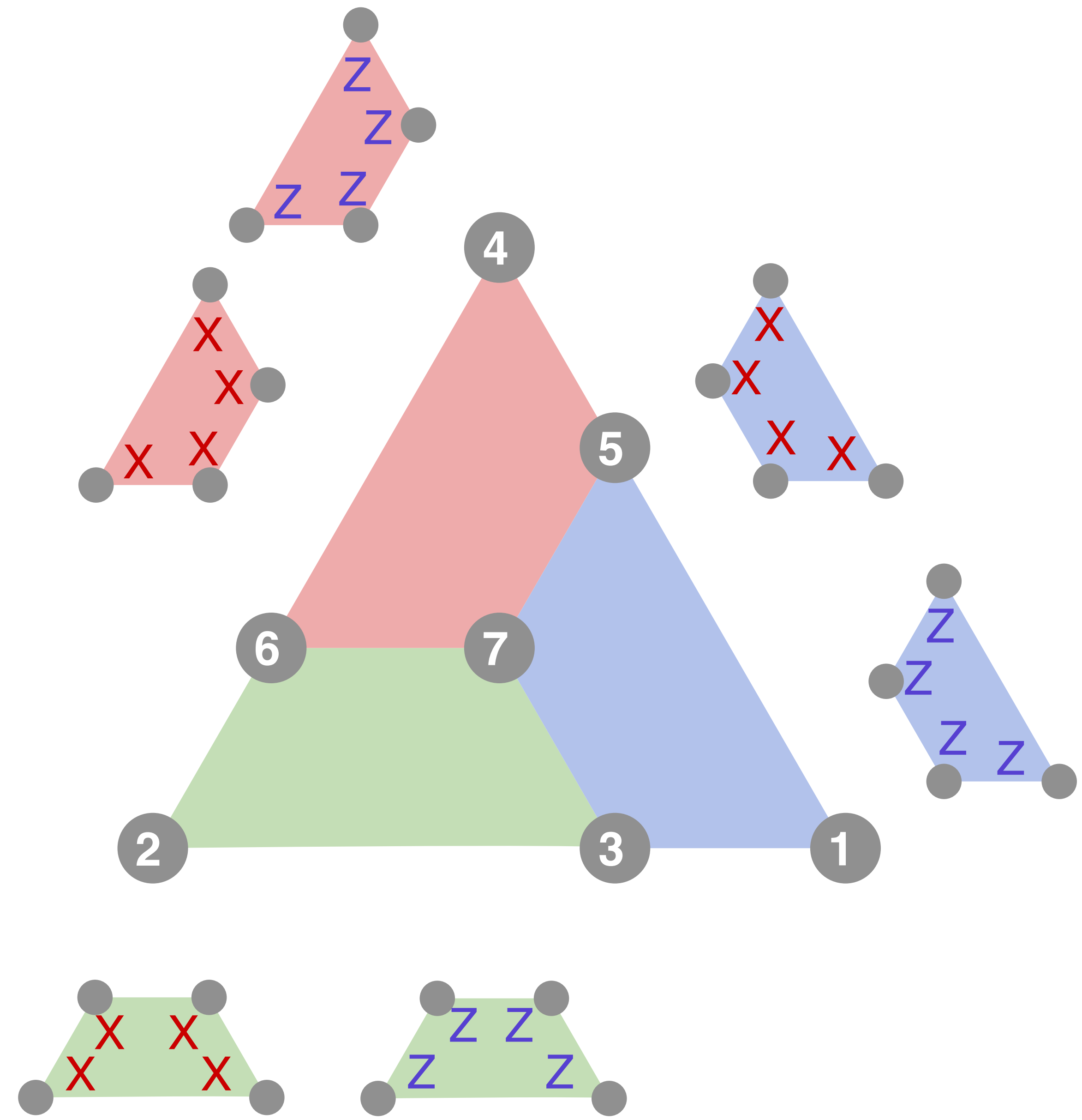
The Steane code (7-qubit code)

7 qubit Steane code
stabilizer generators:

S_1	X	X	I	X	X	I	I
S_2	X	I	X	X	I	X	I
S_3	I	X	X	X	I	I	X
S_4	Z	Z	I	Z	Z	I	I
S_5	Z	I	Z	Z	I	Z	I
S_6	I	Z	Z	Z	I	I	Z
Z_L	I	I	I	I	Z	Z	Z
X_L	I	I	I	I	X	X	X



The Steane code is an example of a CSS code (for Calder, Shor, and Steane,) meaning the stabilizers do X parity checks and Z parity checks separately.

CSS codes

9 qubit Shor code
stabilizer generators:

S_1	Z	Z							
S_2		Z	Z						
S_3				Z	Z				
S_4					Z	Z			
S_5							Z	Z	
S_6								Z	Z
S_7	X	X	X	X	X	X			
S_8				X	X	X	X	X	X
X_L	Z	Z	Z	Z	Z	Z	Z	Z	Z
Z_L	X	X	X	X	X	X	X	X	X

5 qubit code
stabilizer generators:

S_1	X	Z	Z	X	I
S_2	I	X	Z	Z	X
S_3	X	I	X	Z	Z
S_4	Z	X	I	X	Z
Z_L	Z	Z	Z	Z	Z
X_L	X	X	X	X	X

The Shor code and the surface code are also CSS codes, but the 5-qubit code is not.

The surface code “four-cycle”

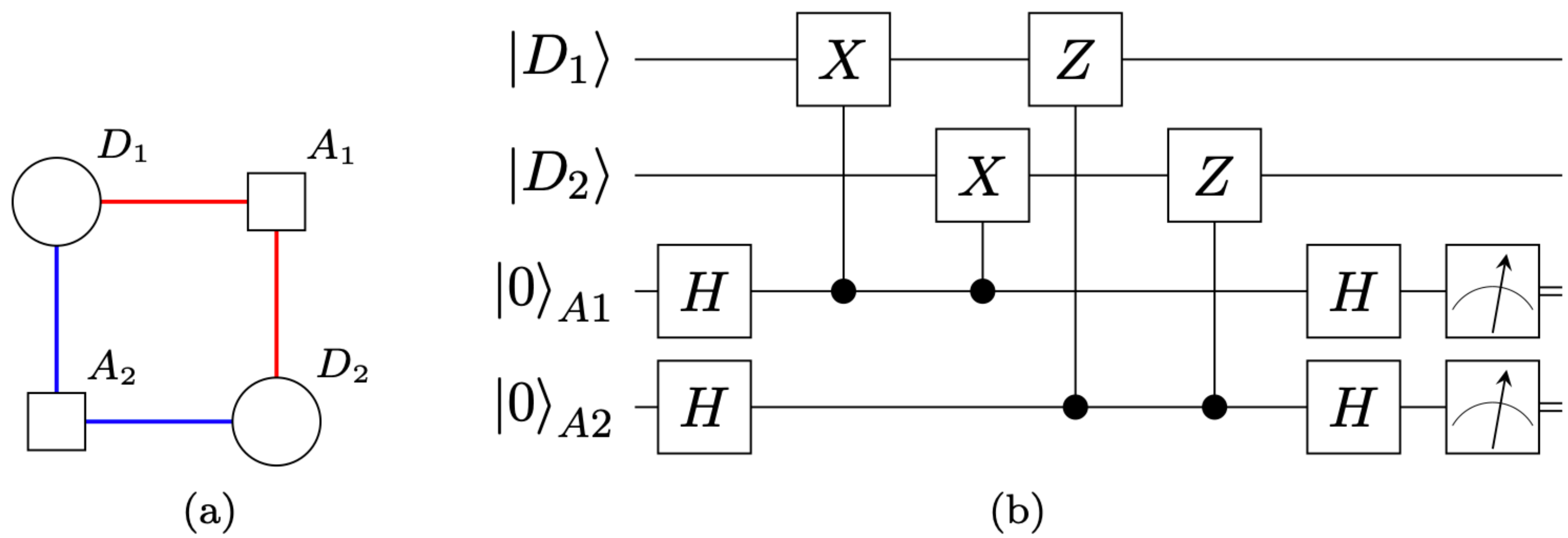
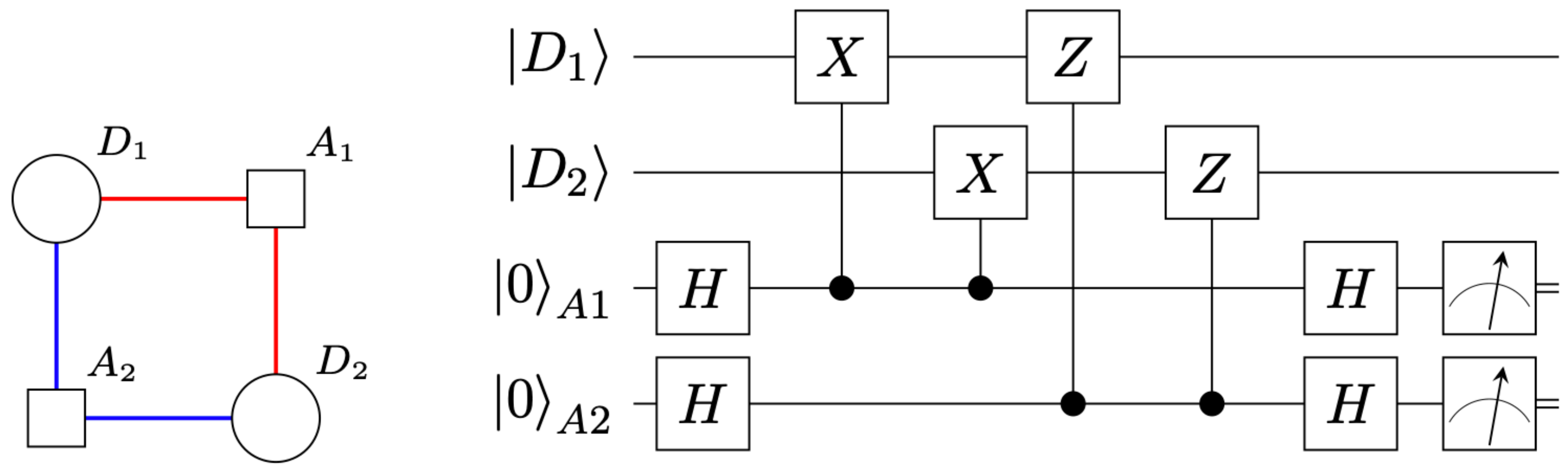


Figure 7. The surface code four-cycle. (a) Pictorial representation. The code qubits, D_1 and D_2, are represented by the circular nodes. The ancilla qubits, A_1 and A_2, are represented by the square nodes. The red and blue edges depict controlled-X and controlled-Z operations controlled on the ancilla qubits and acting on the code qubits. (b) An equivalent surface code four-cycle in circuit notation.

The surface code “four-cycle”

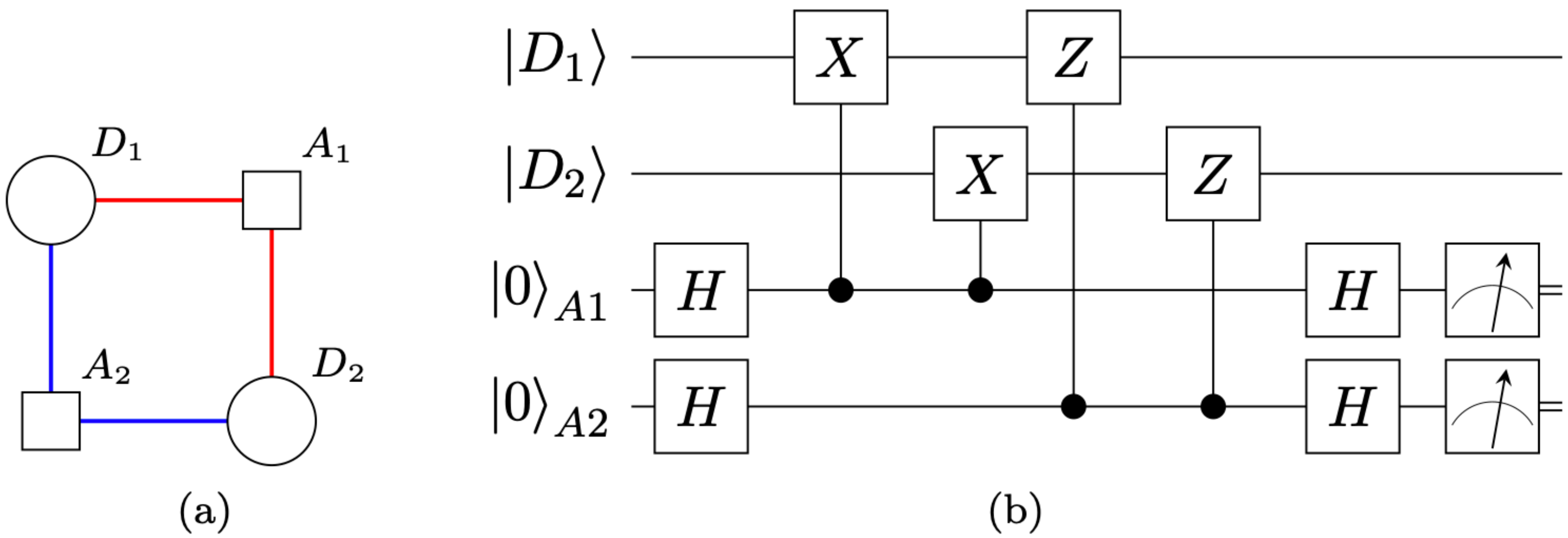


(a) (b)

$$X_iZ_i = -Z_iX_i$$

$$S_1 = X_1X_2$$
$$S_2 = Z_1Z_2$$

The surface code "four-cycle"



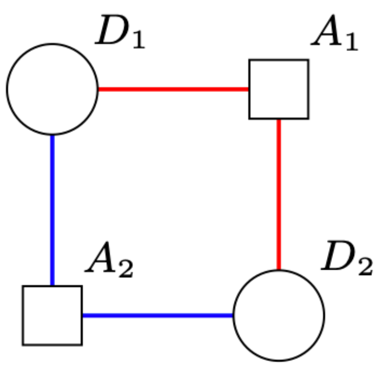
$$X_i Z_i = -Z_i X_i$$

$$S_1 = X_1 X_2$$
$$S_2 = Z_1 Z_2$$

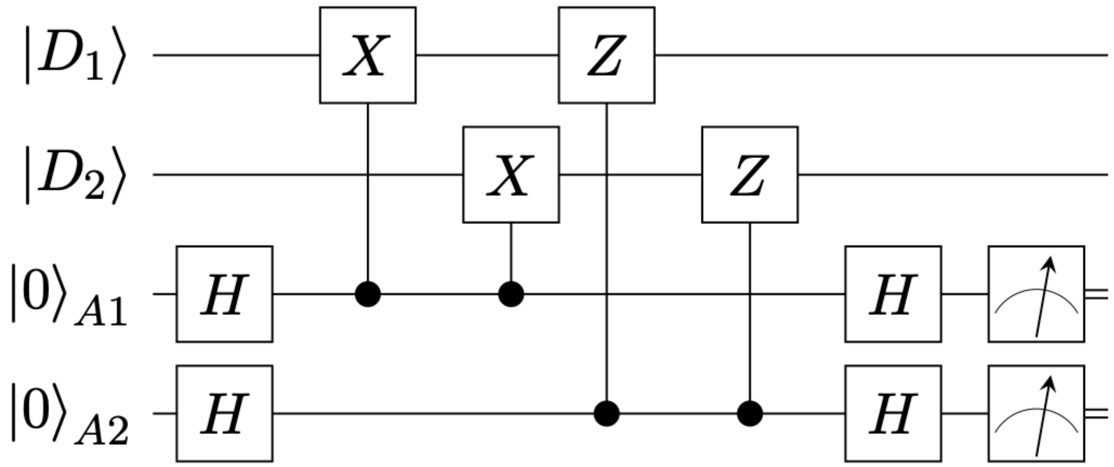
Do these commute?

$$[S_1, S_2] = [X_1 X_2, Z_1 Z_2]$$

The surface code "four-cycle"



(a)



(b)

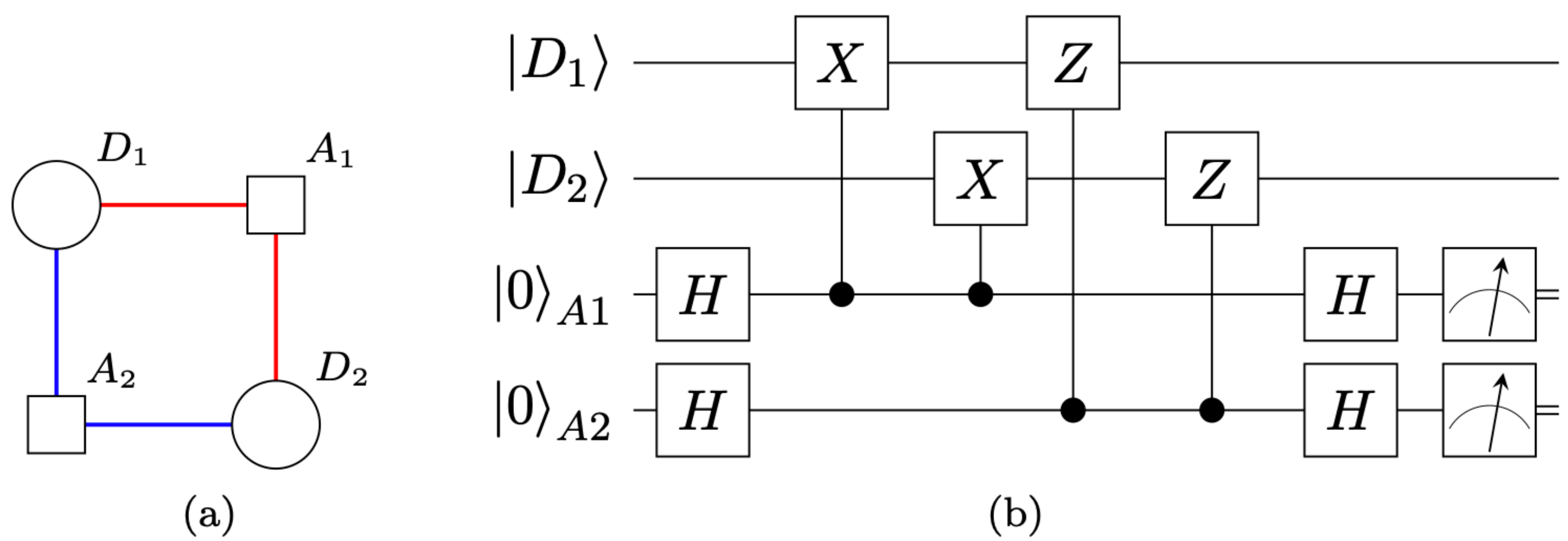
$$X_i Z_i = -Z_i X_i$$

$$S_1 = X_1 X_2$$
$$S_2 = Z_1 Z_2$$

Do these commute?

$$[S_1, S_2] = [X_1 X_2, Z_1 Z_2]$$
$$[X_1 X_2, Z_1 Z_2] = X_1 X_2 Z_1 Z_2 - Z_1 Z_2 X_1 X_2$$

The surface code "four-cycle"



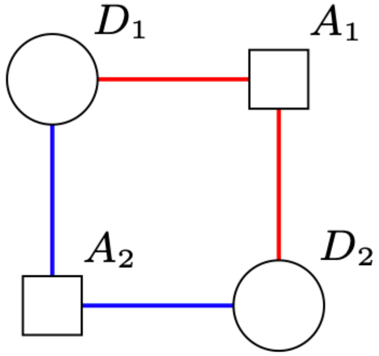
$$X_i Z_i = -Z_i X_i$$

$$S_1 = X_1 X_2$$
$$S_2 = Z_1 Z_2$$

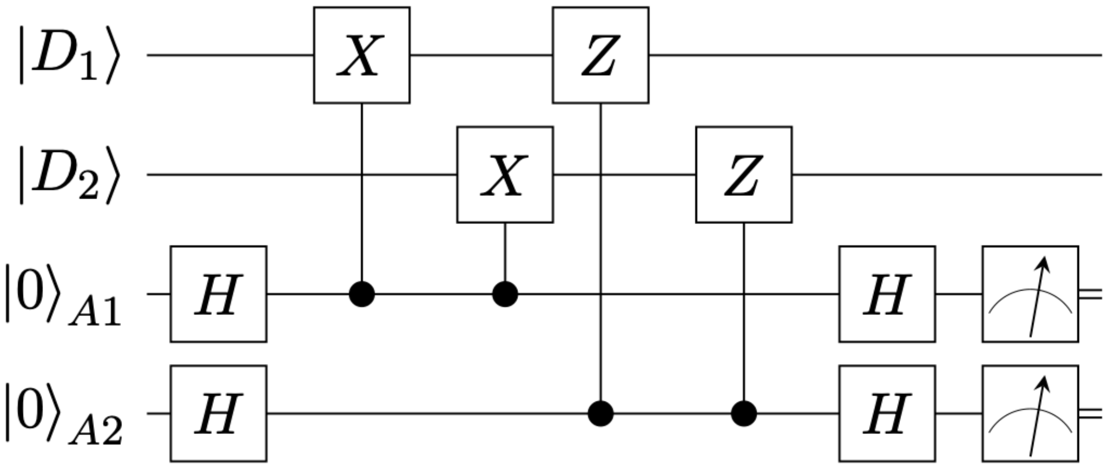
Do these commute?

$$[S_1, S_2] = [X_1X_2, Z_1Z_2]$$
$$[X_1X_2, Z_1Z_2] = X_1X_2Z_1Z_2 - Z_1Z_2X_1X_2$$
$$[X_1X_2, Z_1Z_2] = X_1Z_1X_2Z_2 - Z_1X_1Z_2X_2$$

The surface code “four-cycle”



(a)



(b)

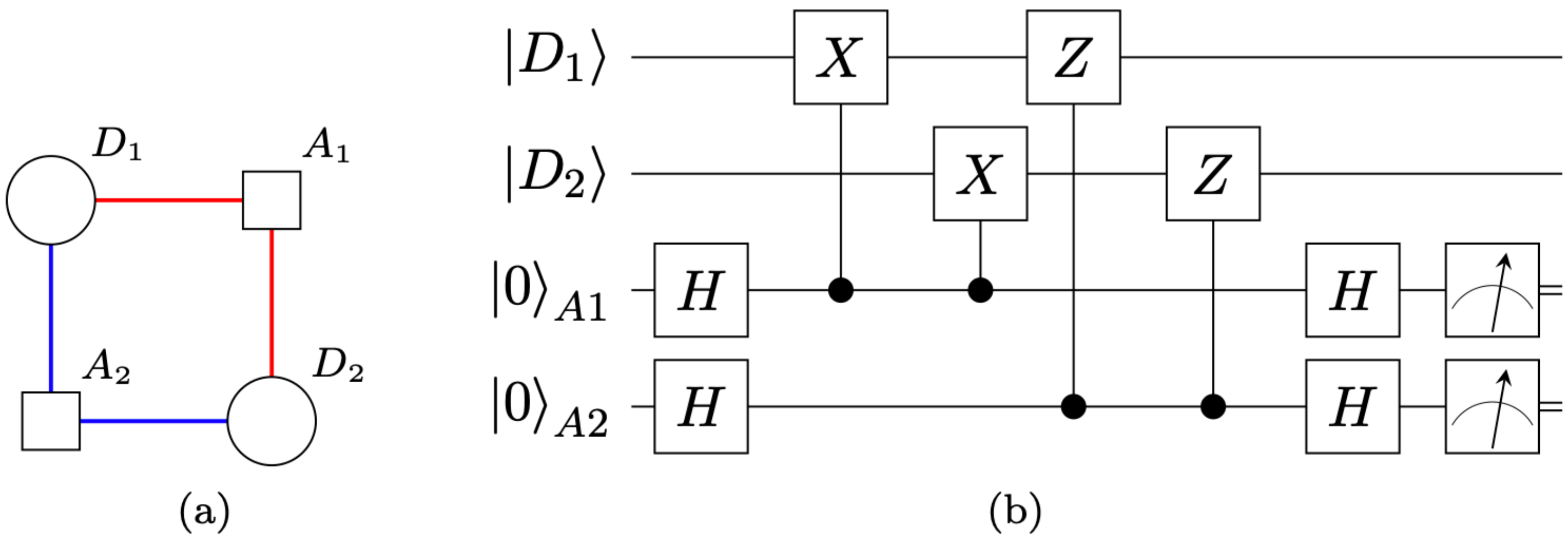
$$X_iZ_i = -Z_iX_i$$

$$S_1 = X_1X_2$$
$$S_2 = Z_1Z_2$$

Do these commute?

$$[S_1, S_2] = [X_1X_2, Z_1Z_2]$$
$$[X_1X_2, Z_1Z_2] = X_1X_2Z_1Z_2 - Z_1Z_2X_1X_2$$
$$[X_1X_2, Z_1Z_2] = X_1Z_1X_2Z_2 - Z_1X_1Z_2X_2$$
$$[X_1X_2, Z_1Z_2] = (-Z_1X_1)(-Z_2X_2) - Z_1X_1Z_2X_2$$
$$[X_1X_2, Z_1Z_2] = Z_1X_1Z_2X_2 - Z_1X_1Z_2X_2 = 0$$

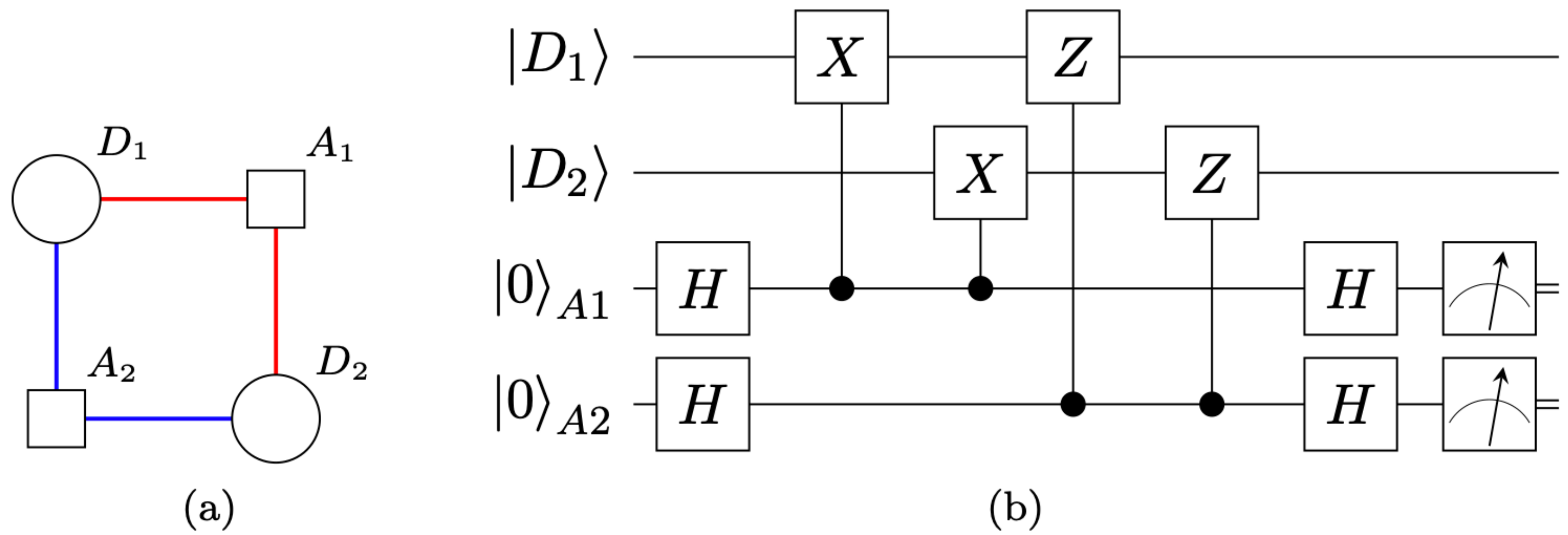
The surface code “four-cycle”



$$S_1 = X_1X_2$$
$$S_2 = Z_1Z_2$$

What are the eigenstates?

The surface code "four-cycle"



$$S_1 = X_1 X_2$$
$$S_2 = Z_1 Z_2$$

What are the eigenstates?

	$S_2 = Z_1 Z_2 = +1$	$Z_1 Z_2 = -1$
$S_1 = X_1 X_2 = +1$	$(\vert 00\rangle + \vert 11\rangle)/\sqrt{2}$	$(\vert 01\rangle + \vert 10\rangle)/\sqrt{2}$
$X_1 X_2 = -1$	$(\vert 00\rangle - \vert 11\rangle)/\sqrt{2}$	$(\vert 01\rangle - \vert 10\rangle)/\sqrt{2}$

The Bell states!

The surface code “four-cycle”

$$S_1 = X_1 X_2$$

$$S_2 = Z_1 Z_2$$

$$X_i Z_i = -Z_i X_i$$

The surface code “four-cycle”

$$S_1 = X_1 X_2$$

$$S_2 = Z_1 Z_2$$

$$X_i Z_i = -Z_i X_i$$

$$S_1 Z_1 |\psi\rangle = -Z_1 S_1 |\psi\rangle = -Z_1 |\psi\rangle$$

$$S_2 Z_1 |\psi\rangle = Z_1 S_2 |\psi\rangle = Z_1 |\psi\rangle$$

The surface code "four-cycle"

$$S_1 = X_1 X_2$$
$$S_2 = Z_1 Z_2$$
$$X_i Z_i = -Z_i X_i$$

$$S_1 Z_1 |\psi\rangle = -Z_1 S_1 |\psi\rangle = -Z_1 |\psi\rangle$$
$$S_2 Z_1 |\psi\rangle = Z_1 S_2 |\psi\rangle = Z_1 |\psi\rangle$$

$$\begin{array}{r|ccc}
 & S_2 = Z_1 Z_2 = +1 & & Z_1 Z_2 = -1 \\
\hline
S_1 = X_1 X_2 = +1 & (|00\rangle + |11\rangle)/\sqrt{2} & \xrightarrow{X} & (|01\rangle + |10\rangle)/\sqrt{2} \\
 & \downarrow Z & \searrow iY & \downarrow Z \\
X_1 X_2 = -1 & (|00\rangle - |11\rangle)/\sqrt{2} & \xrightarrow{X} & (|01\rangle - |10\rangle)/\sqrt{2}
\end{array}$$

The surface code "four-cycle"

$$S_1 = X_1X_2$$
$$S_2 = Z_1Z_2$$
$$X_iZ_i = -Z_iX_i$$

$$S_1 Z_1 |\psi\rangle = -Z_1 S_1 |\psi\rangle = -Z_1 |\psi\rangle$$
$$S_2 Z_1 |\psi\rangle = Z_1 S_2 |\psi\rangle = Z_1 |\psi\rangle$$

$$\begin{array}{r|ccc}
 & S_2 = Z_1 Z_2 = +1 & & Z_1 Z_2 = -1 \\
\hline
S_1 = X_1 X_2 = +1 & (|00\rangle + |11\rangle)/\sqrt{2} & \xrightarrow{X} & (|01\rangle + |10\rangle)/\sqrt{2} \\
 & \downarrow Z & \searrow iY & \downarrow Z \\
X_1 X_2 = -1 & (|00\rangle - |11\rangle)/\sqrt{2} & \xrightarrow{X} & (|01\rangle - |10\rangle)/\sqrt{2}
\end{array}$$

Can this be used as a logical qubit?

The surface code "four-cycle"

$$S_1 = X_1 X_2$$
$$S_2 = Z_1 Z_2$$
$$X_i Z_i = - Z_i X_i$$

$$S_1 Z_1 |\psi\rangle = -Z_1 S_1 |\psi\rangle = -Z_1 |\psi\rangle$$
$$S_2 Z_1 |\psi\rangle = Z_1 S_2 |\psi\rangle = Z_1 |\psi\rangle$$

$$\begin{array}{r|ccc}
 & S_2 = Z_1 Z_2 = +1 & & Z_1 Z_2 = -1 \\
\hline
S_1 = X_1 X_2 = +1 & (|00\rangle + |11\rangle)/\sqrt{2} & \xrightarrow{X} & (|01\rangle + |10\rangle)/\sqrt{2} \\
 & \downarrow Z & \searrow iY & \downarrow Z \\
X_1 X_2 = -1 & (|00\rangle - |11\rangle)/\sqrt{2} & \xrightarrow{X} & (|01\rangle - |10\rangle)/\sqrt{2}
\end{array}$$

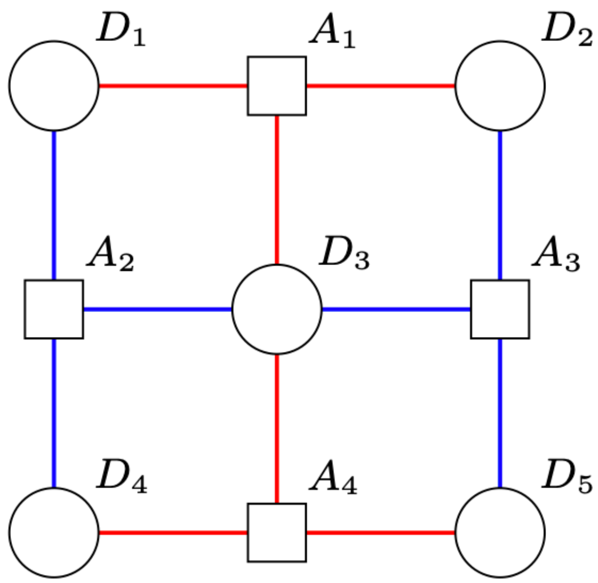
Can this be used as a logical qubit?

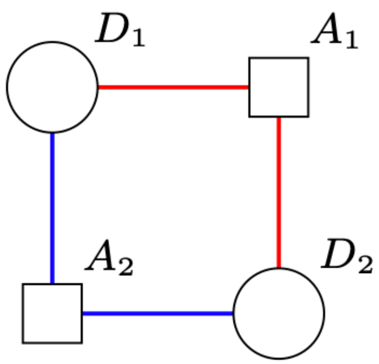
No! Only one state. Can preserve a state, but not a qubit.

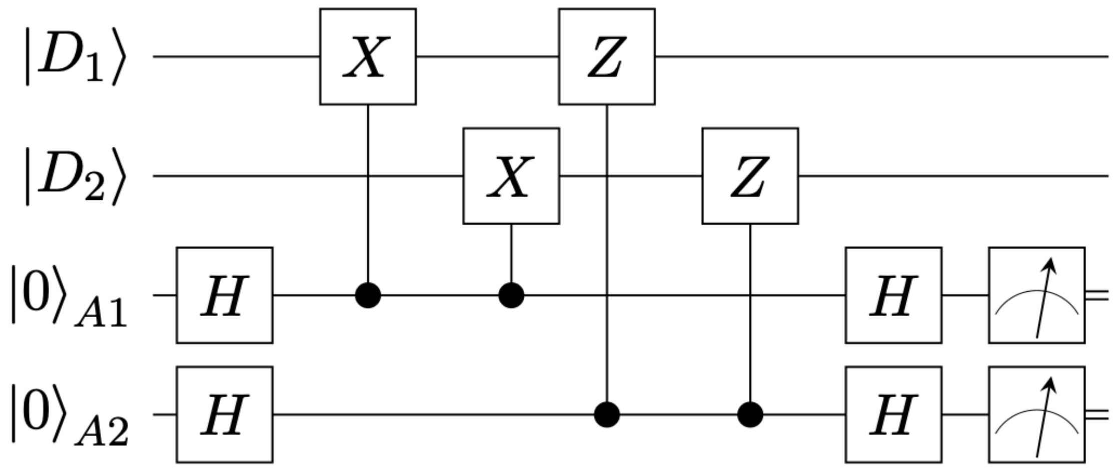
The [[5,1,2]] surface code

Controlled-X

Controlled-Z







The [[5,1,2]] surface code

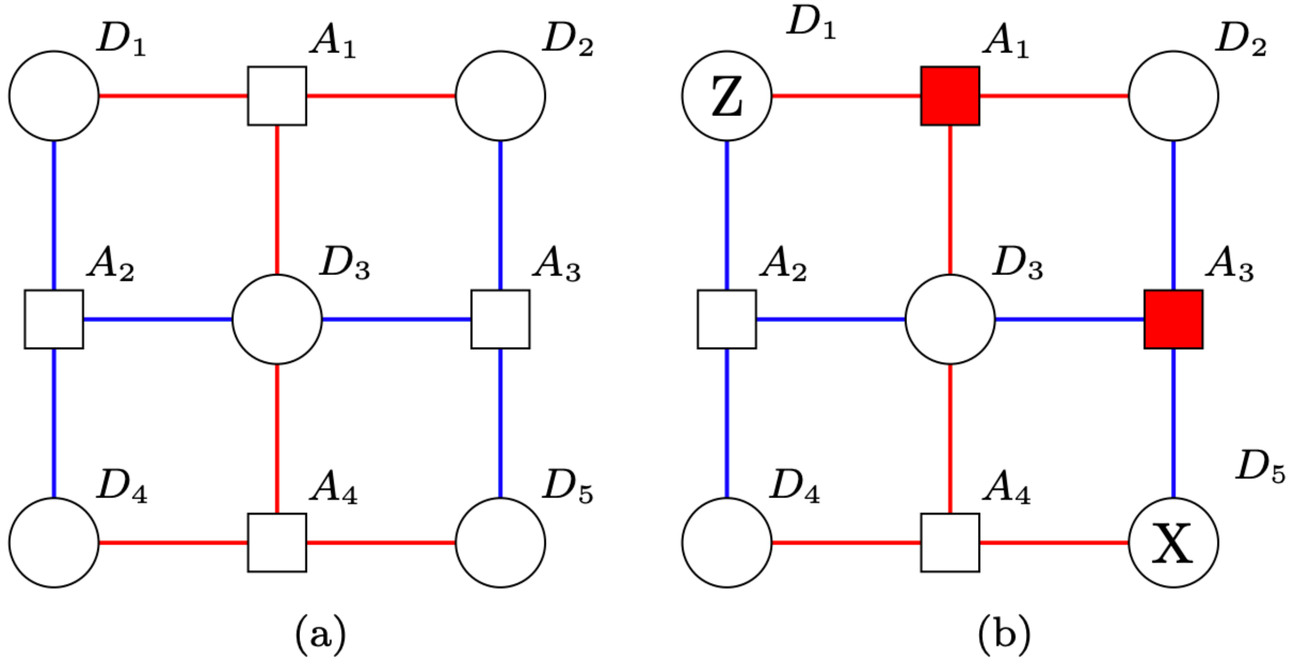


Figure 8. (a) The $[[5,1,2]]$ surface code formed by tiling together four four-cycles in a square lattice. (b) Examples of error detection in the $[[5,1,2]]$ surface code. The Z_{D_1} error on qubit D_1 anti-commutes with the stabilizer measured by ancilla qubit A_1. The A_1 qubit is coloured red to indicate it will measured as a '1'. Likewise, the X_{D_5} error on qubit D_5 is detected by the stabilizer measured by ancilla qubit A_3.

The [[5,1,2]] surface code

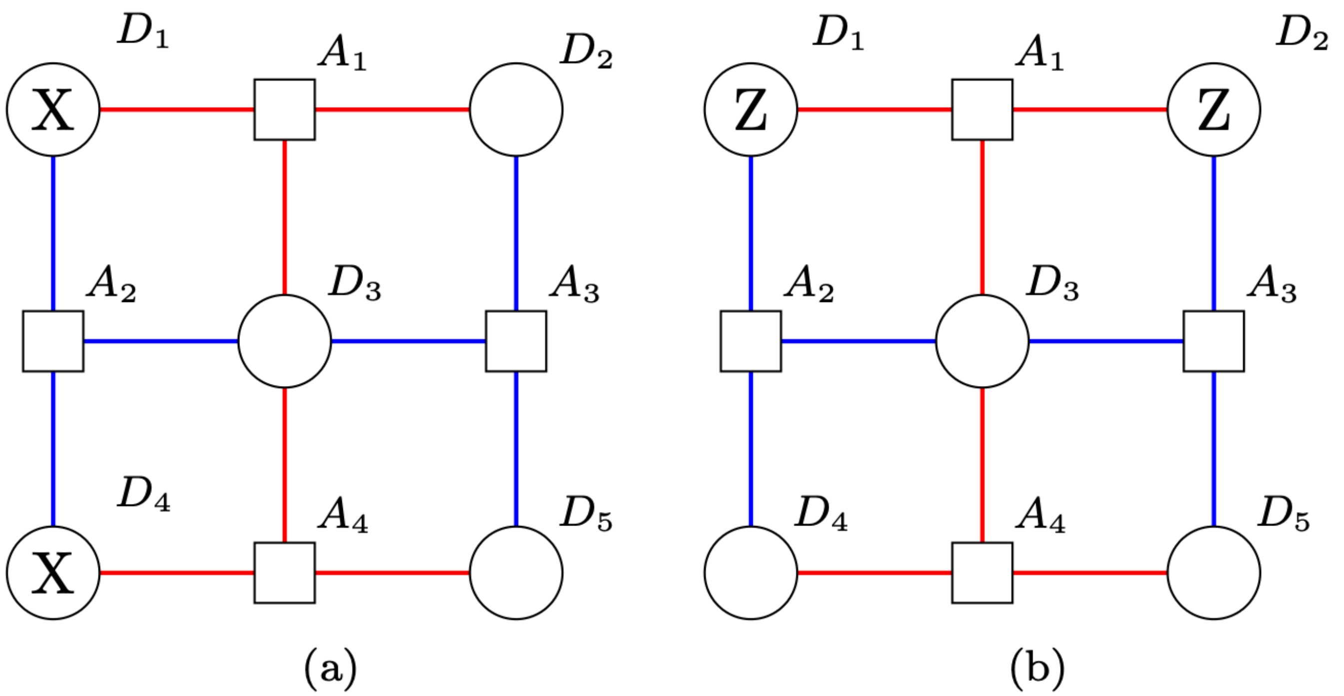


Figure 9. The logical operators of a surface code can be defined as chains of Pauli operations that act along the boundaries of the lattice. (a) The Pauli-X logical operator $\bar{X} = X_{D_1}X_{D_4}$ acts along the boundary along which Z-type stabilizers are measured. (b) The Pauli-Z logical operator $\bar{Z} = Z_{D_1}Z_{D_2}$ acts along the boundary along which X-type stabilizers are measured. The two logical operators anti-commute with one another.

The [[13,1,3]] surface code

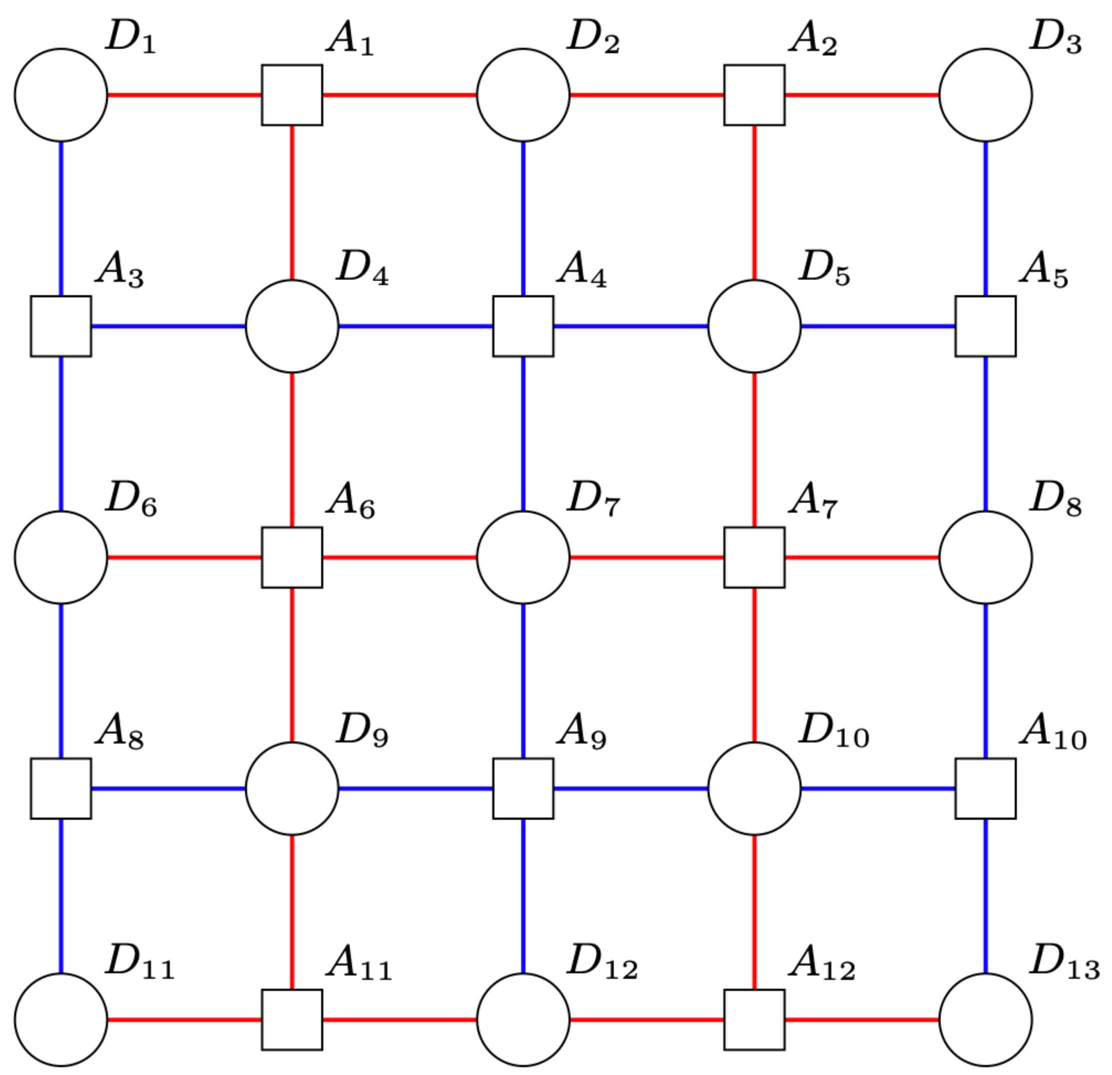
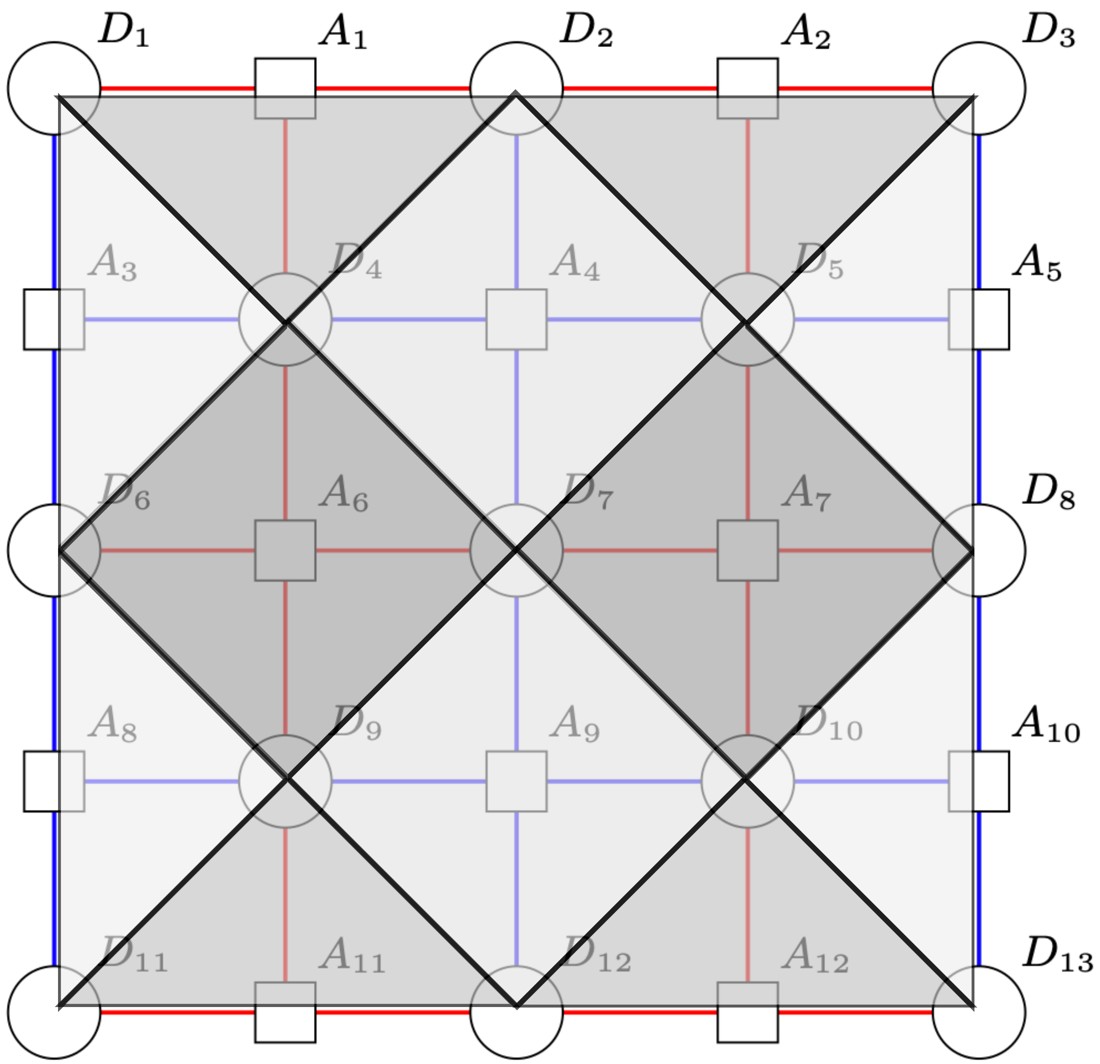
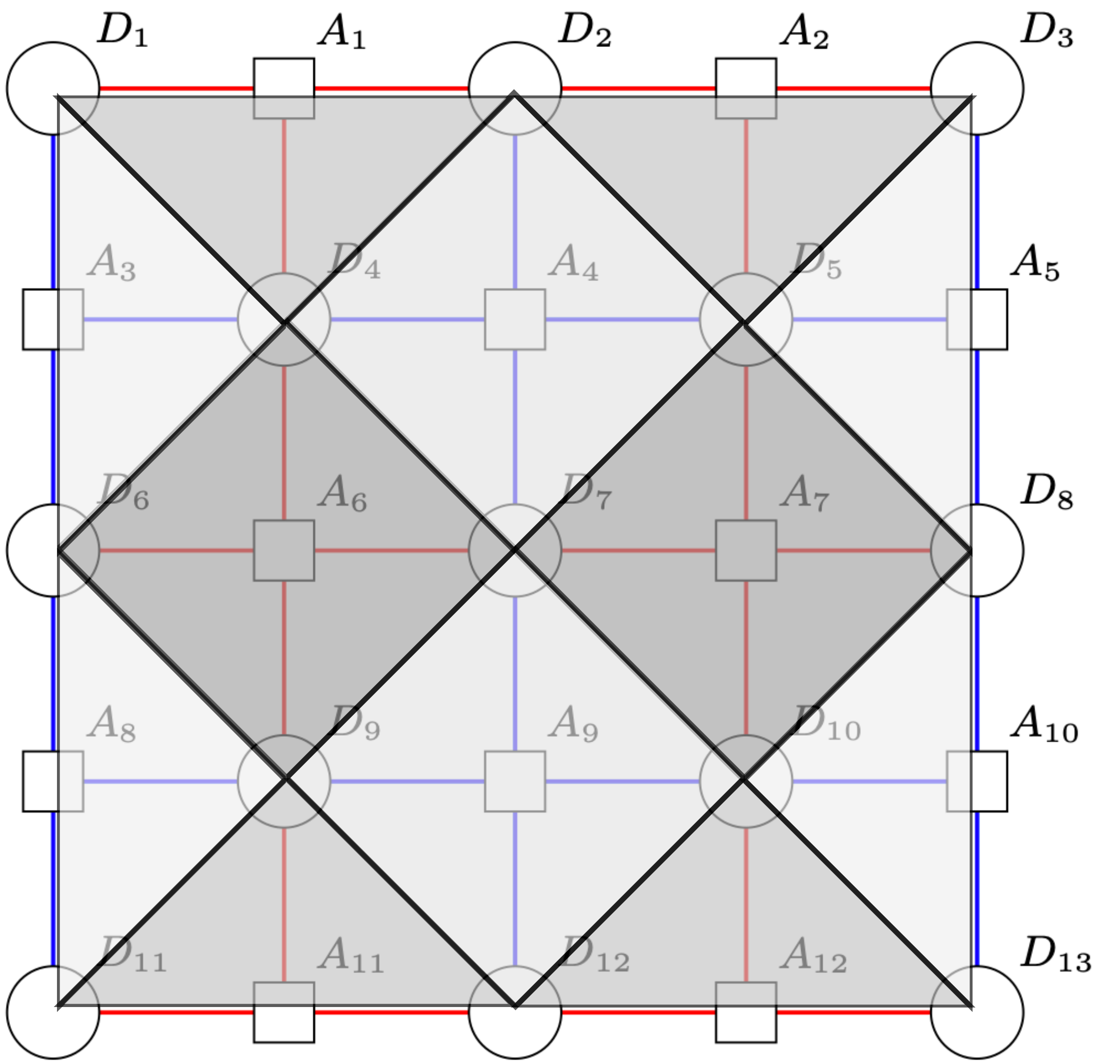


Figure 10. A distance-three surface code with parameters $[[13, 1, 3]]$. A possible choice for the logical operators of this code would be $\bar{X} = X_{D_1} X_{D_6} X_{D_{11}}$ and $\bar{Z} = Z_{D_1} Z_{D_2} Z_{D_3}$.

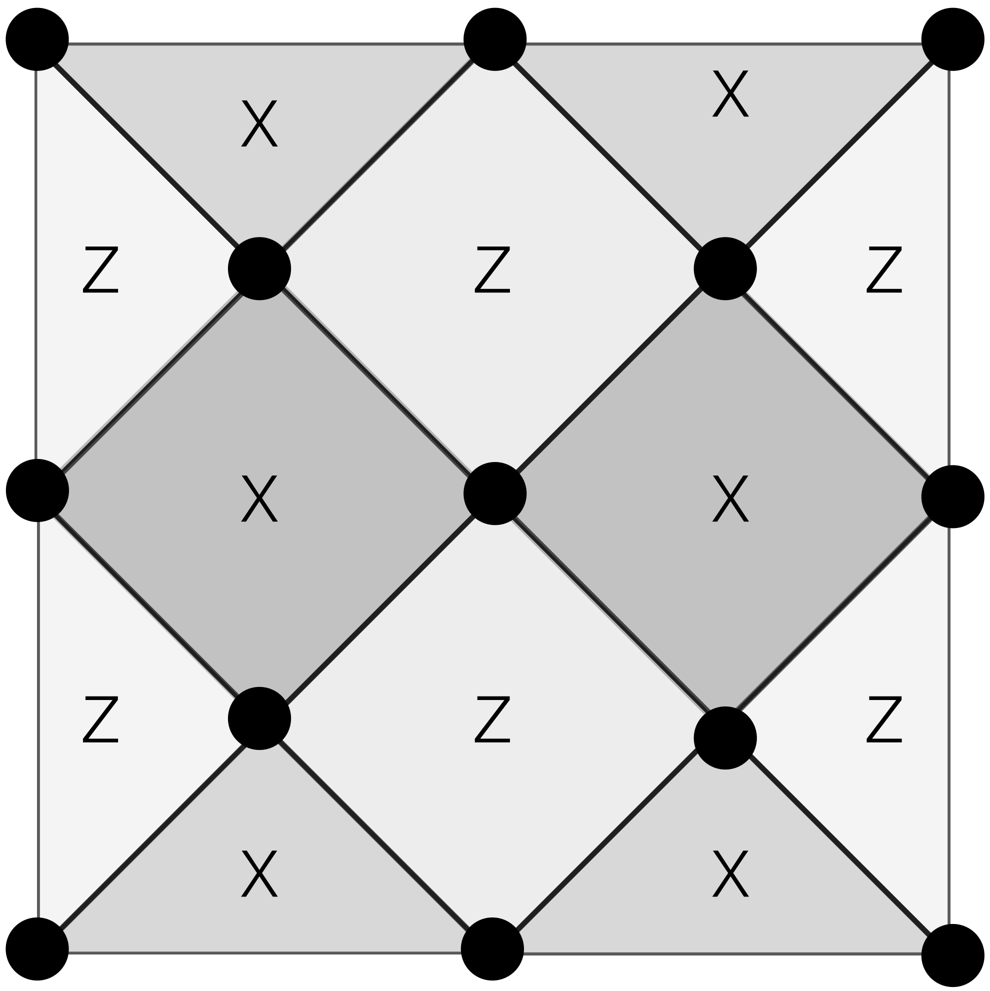
The [[13,1,3]] surface code



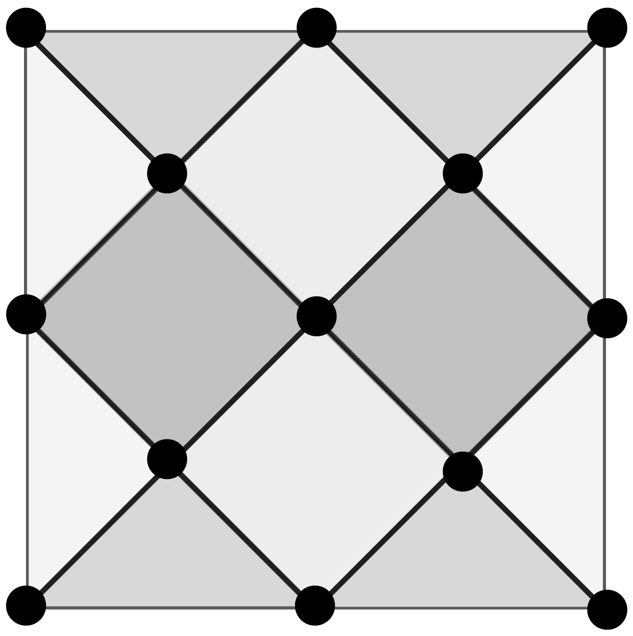
The [[13,1,3]] surface code



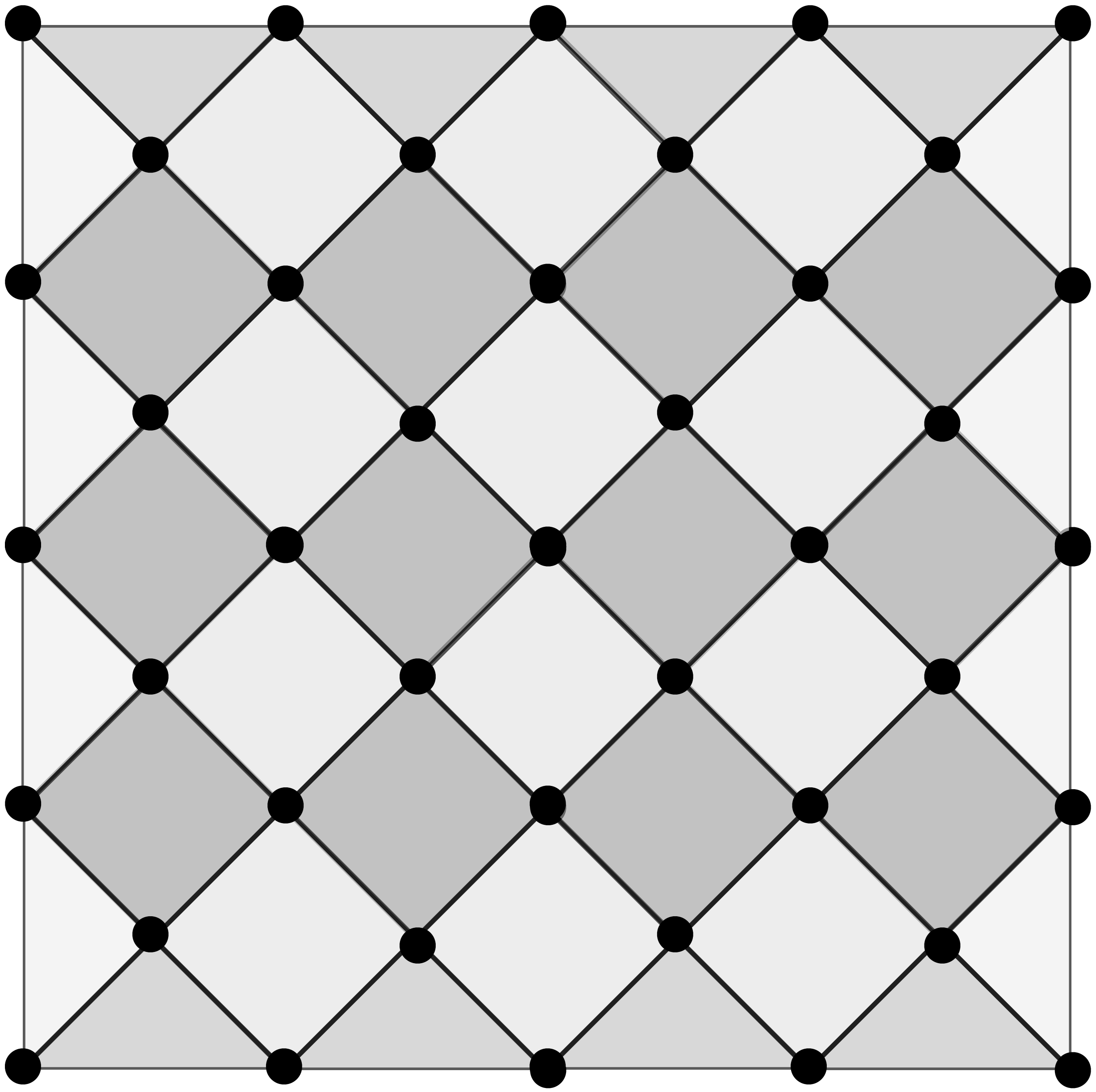
=



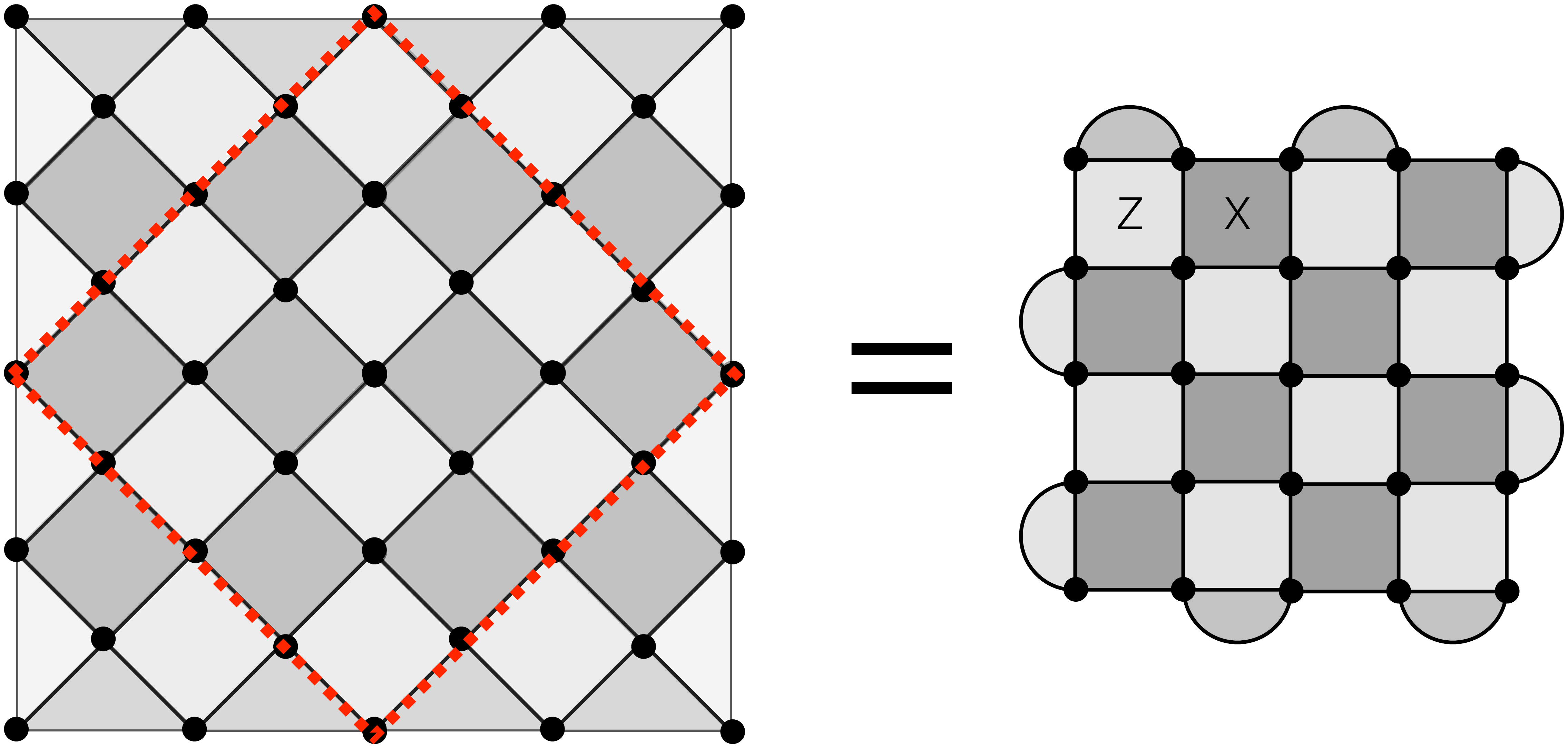
Scaling up the distance by tiling



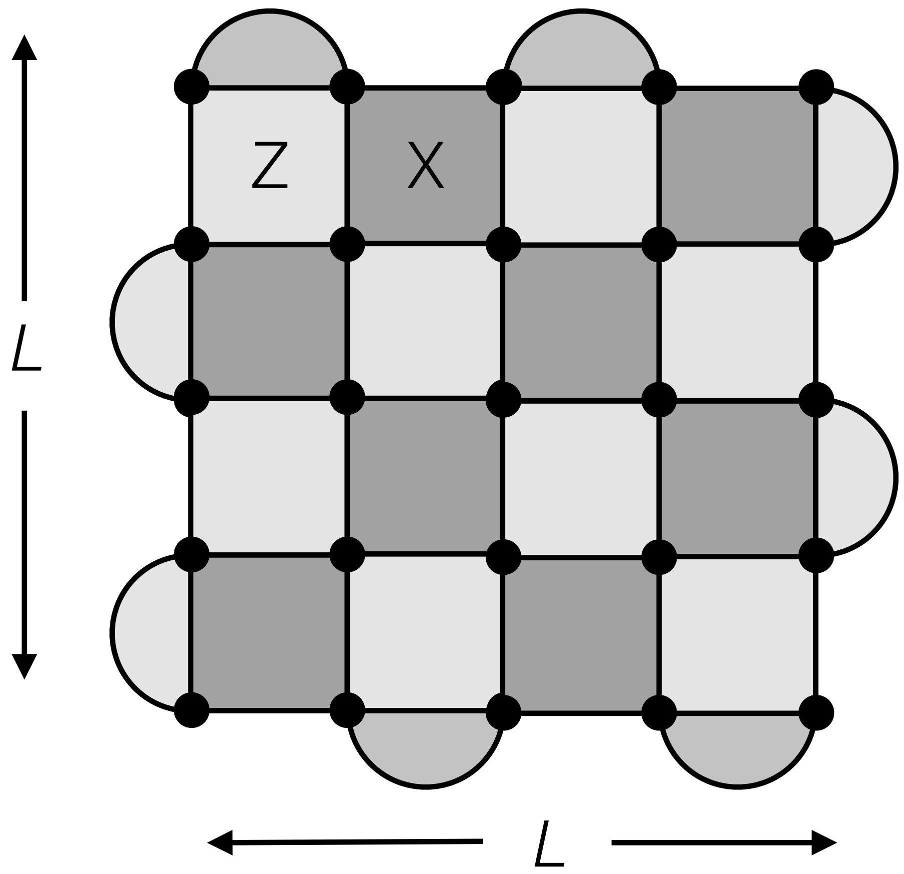
$$\times 4 =$$



“Rotated” surface code



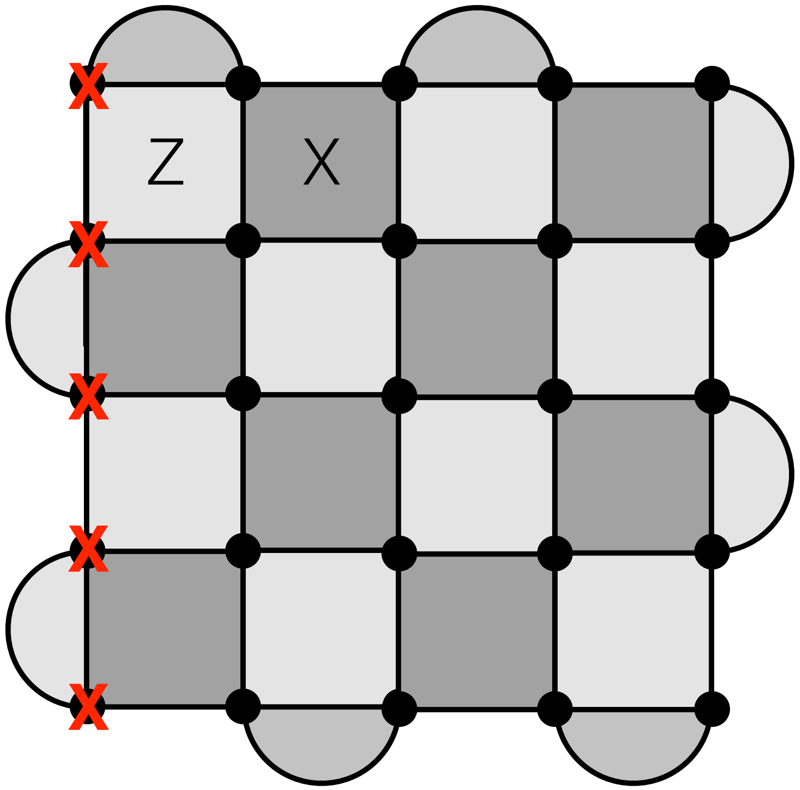
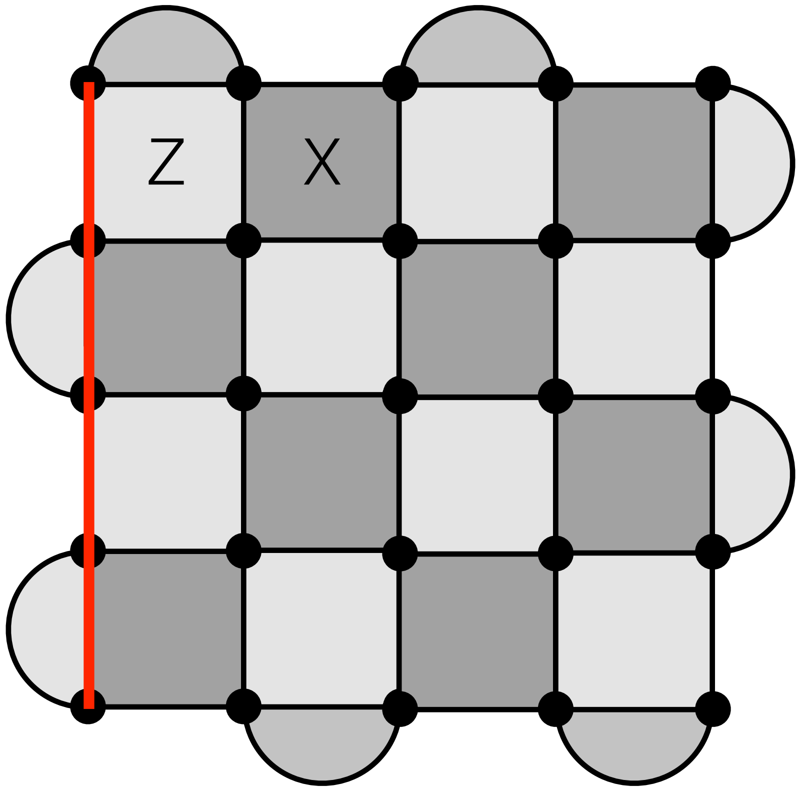
“Rotated” surface code



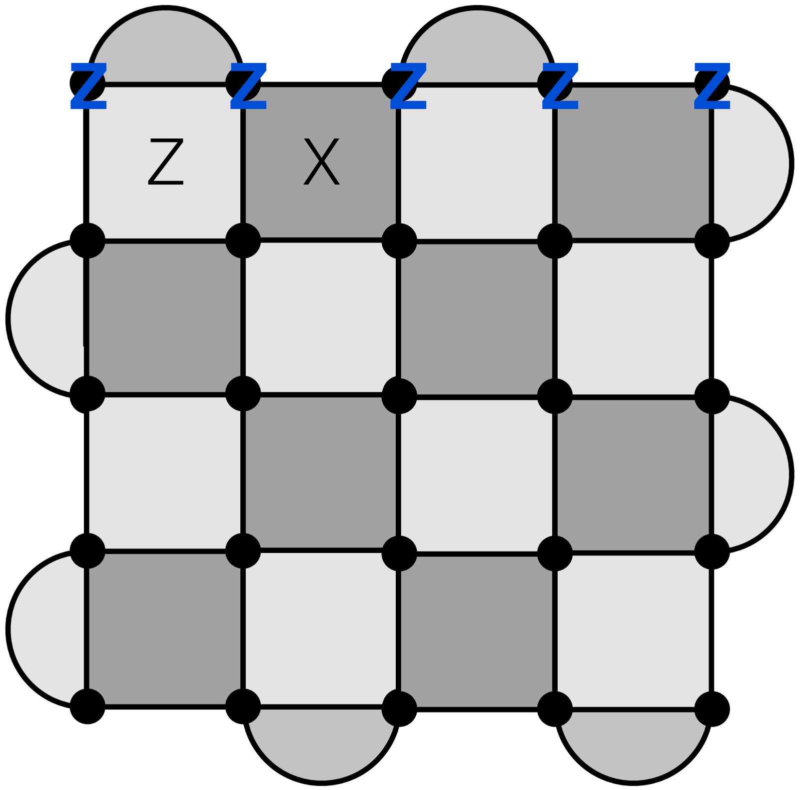
This is a $[[L^2,1,L]]$ code. It has distance L, and requires L^2 data qubits (and roughly the same number of ancilla qubits.)

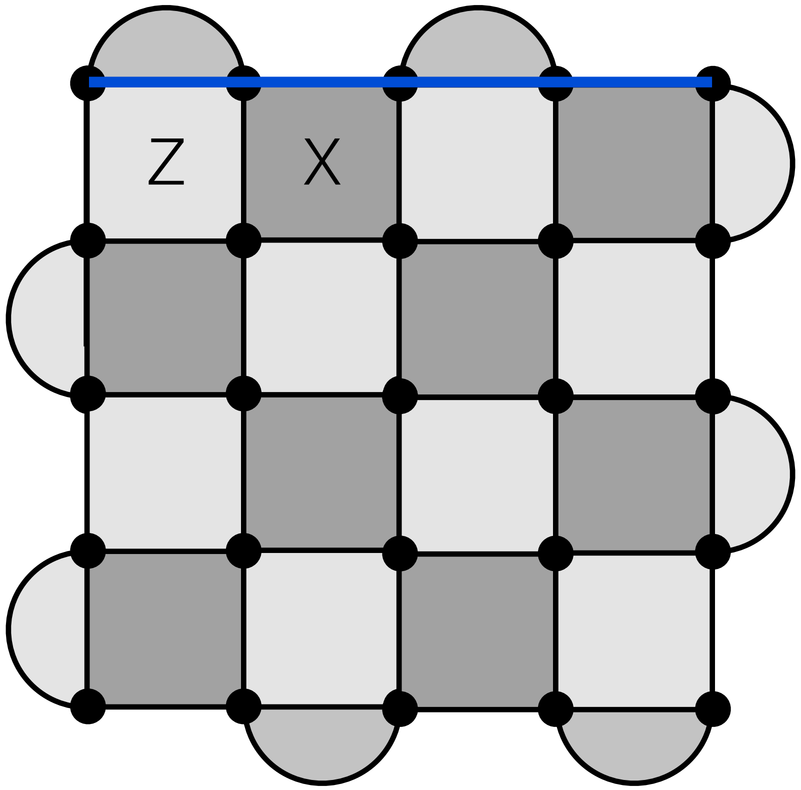
Here $L = 5$, requires 25 data qubits and 24 ancilla qubits.

Logical operators

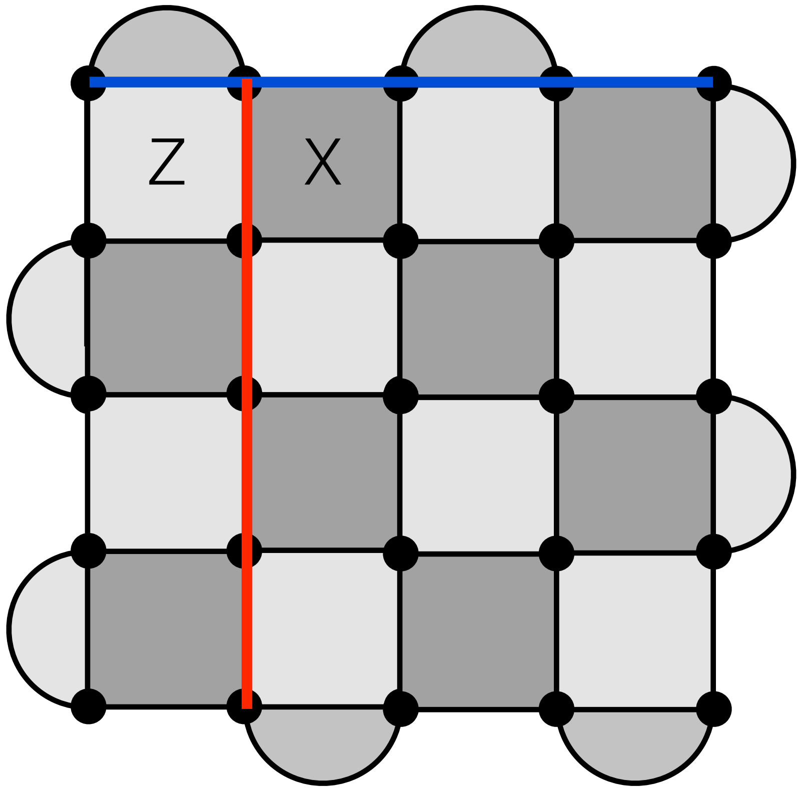
$X_L =$  $=$ 

Logical operators

$$Z_L = $$ 

$=$ 

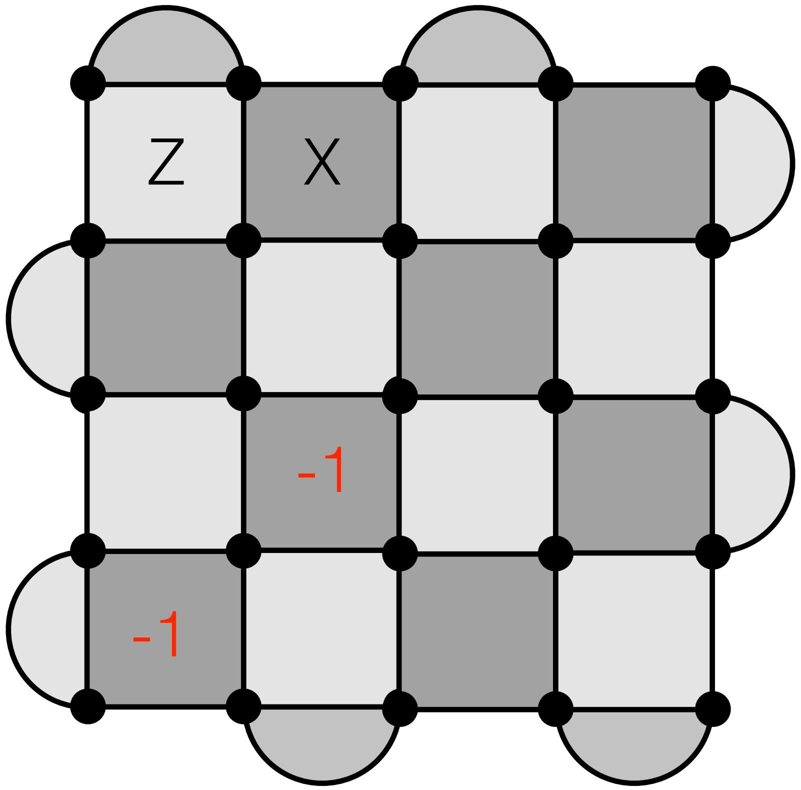
Logical operators



Any vertical line of X operators connecting the top/bottom is a logical X_L.

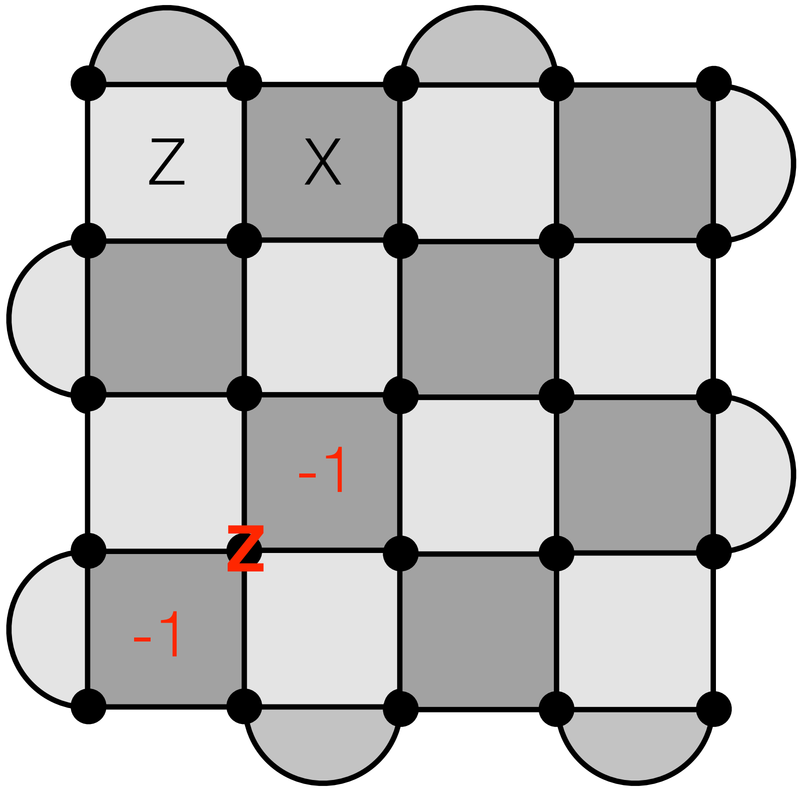
Any horizontal line of Z operators connecting the top/bottom is a logical Z_L.

Errors



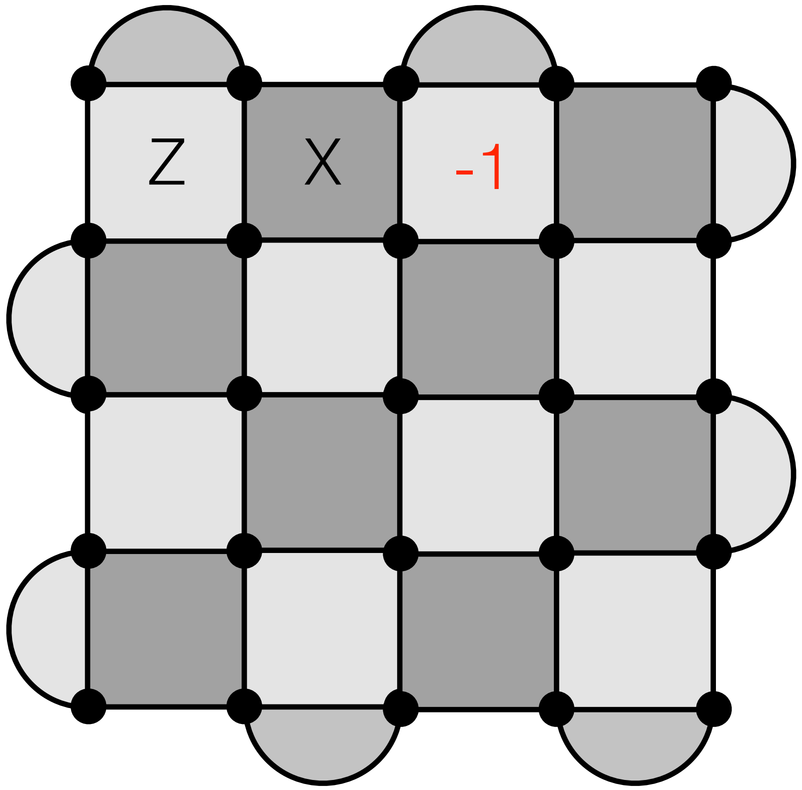
Most single qubit errors will give unique syndromes (as long as they're in the bulk of the lattice.)

Errors



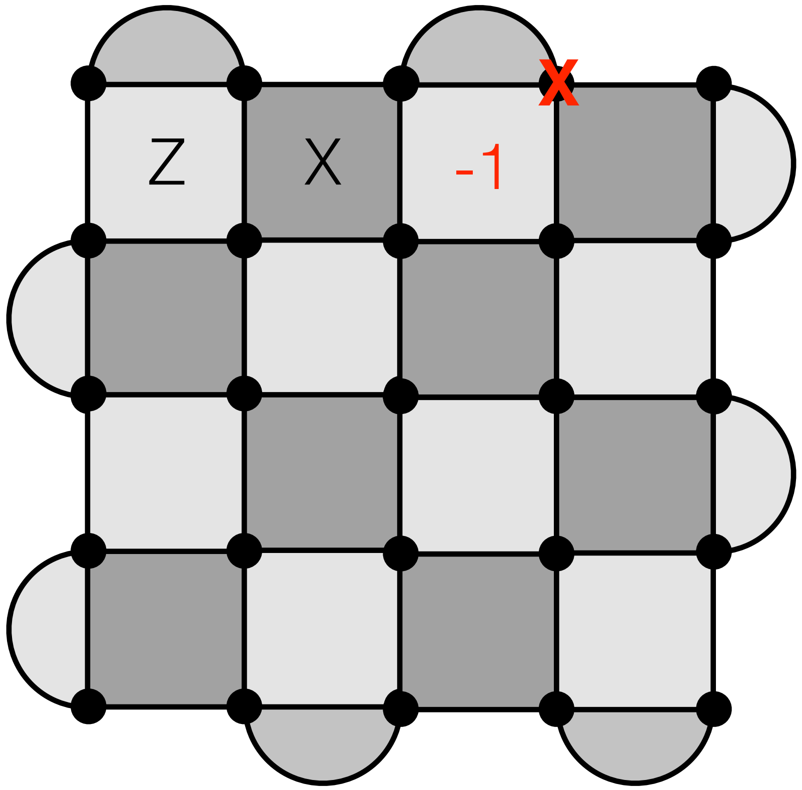
Most single qubit errors will give unique syndromes (as long as they're in the bulk of the lattice.)

Errors



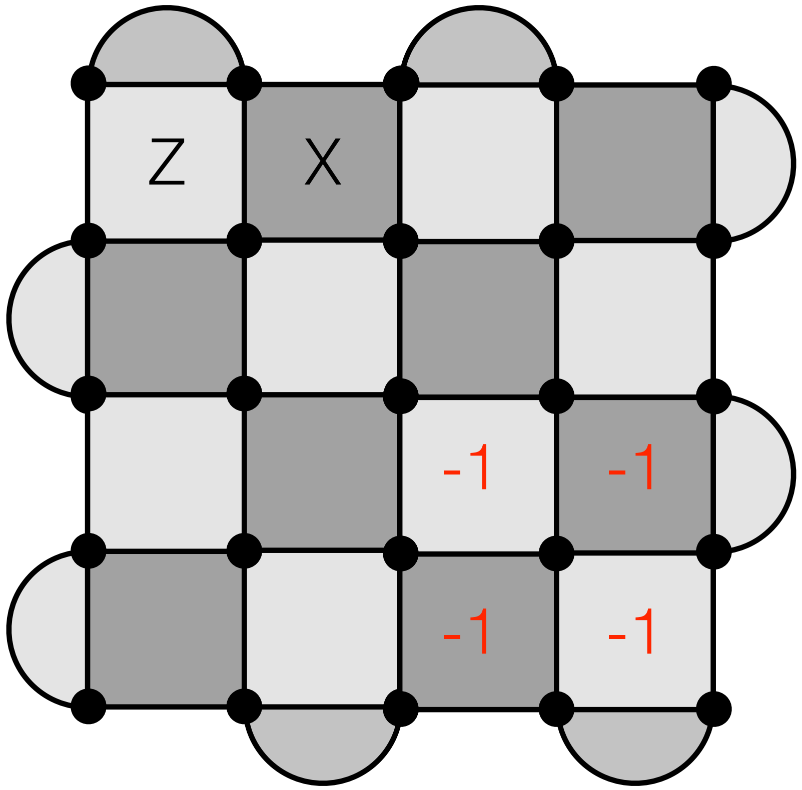
But some single qubit errors on the edges can be degenerate.

Errors



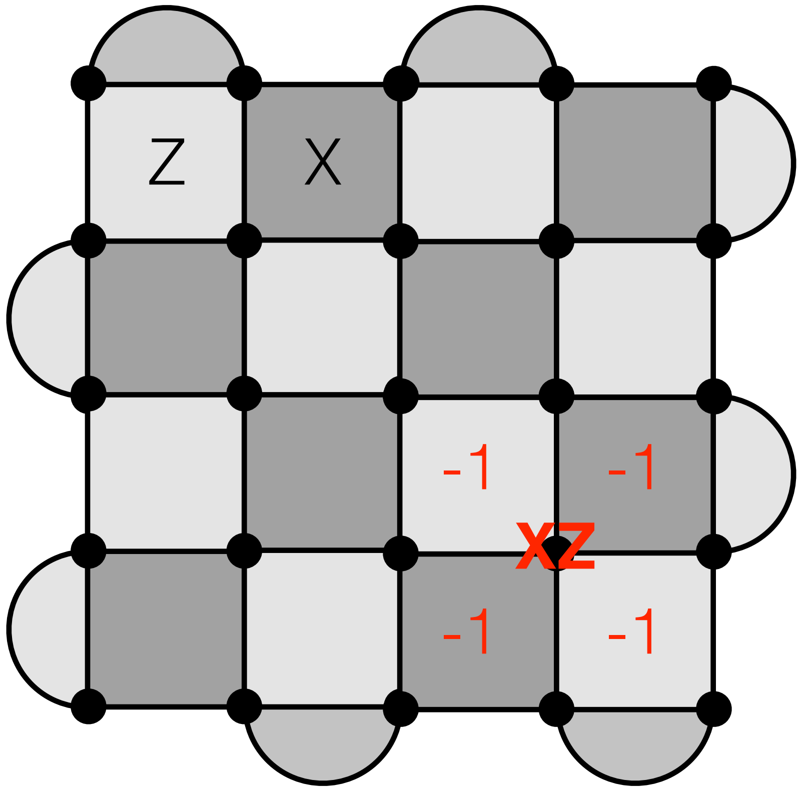
But some single qubit errors on the edges can be degenerate.

Errors



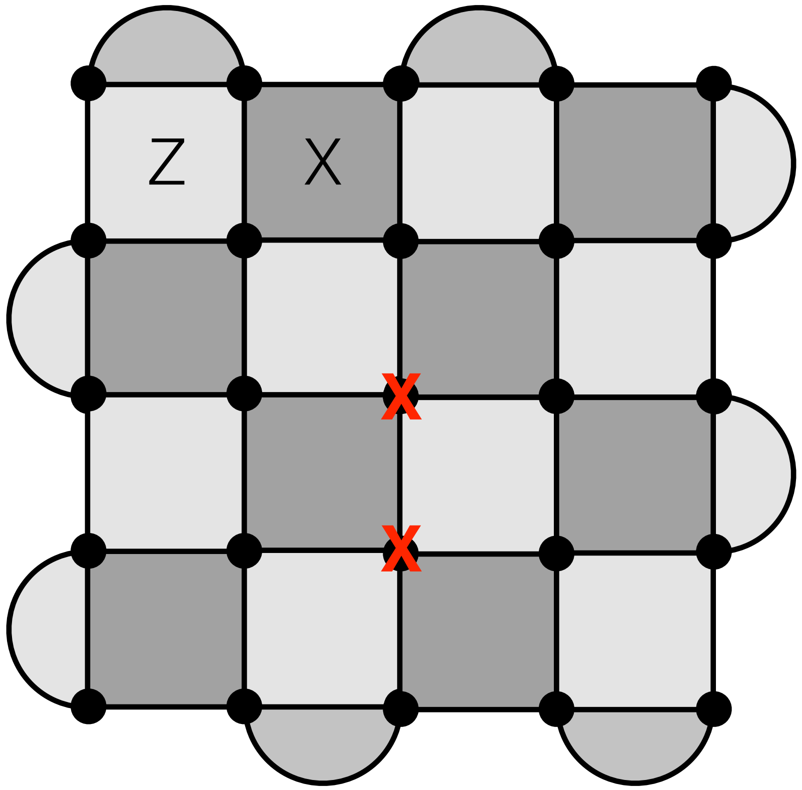
Most single qubit errors will give unique syndromes (as long as they're in the bulk of the lattice.)

Errors



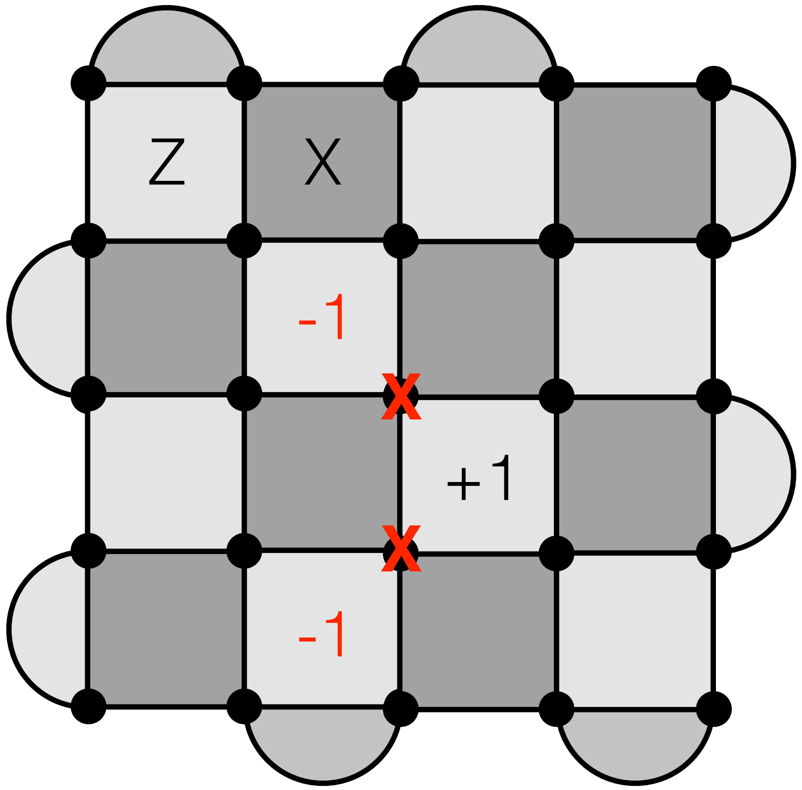
Y (or XZ) errors are especially easy to detect and decode for the surface code.

Errors



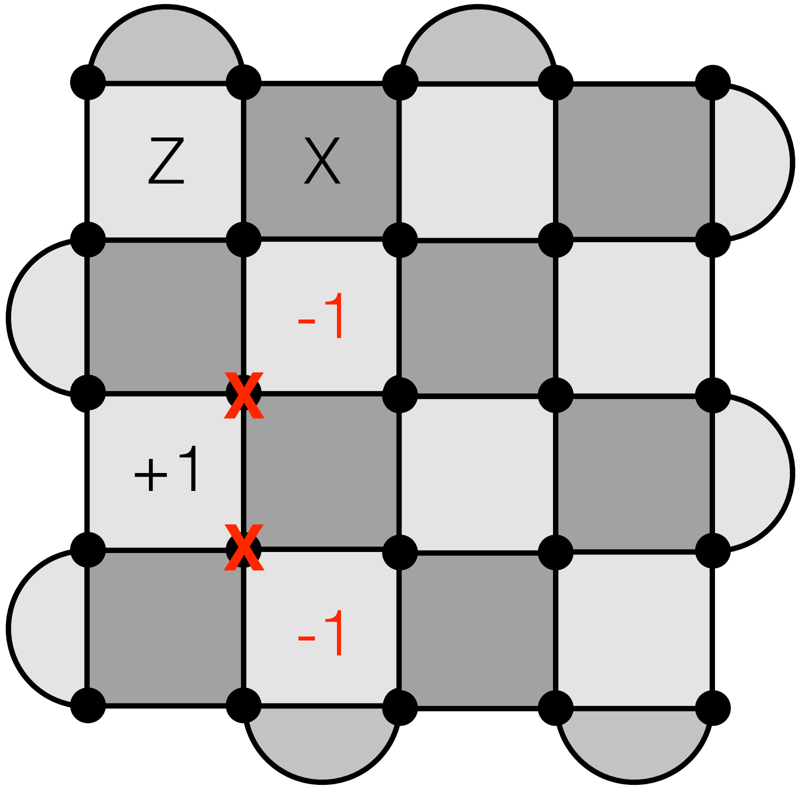
Two qubit errors are usually degenerate though.

Errors



Two qubit errors are usually degenerate though.

Errors

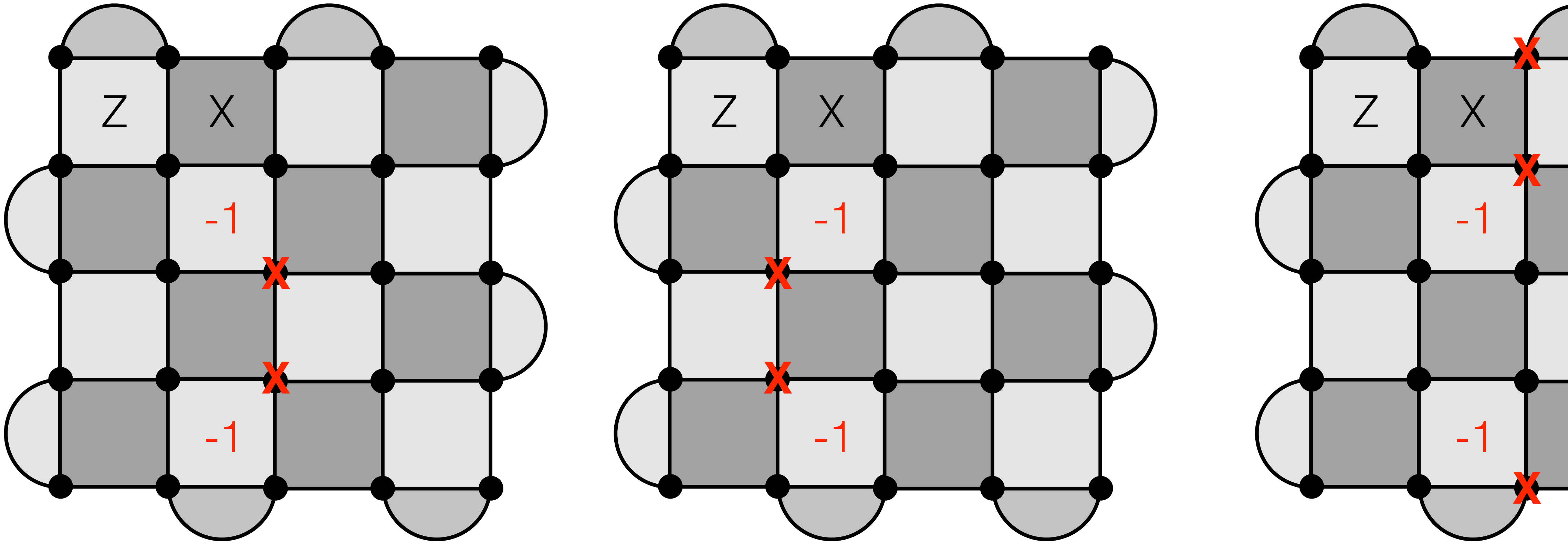


Two qubit errors are usually degenerate though.

Decoding errors

Decoding errors can be reduced to an efficiently solvable graph problem.

Consider these three sets of errors that all result in the same syndrome:

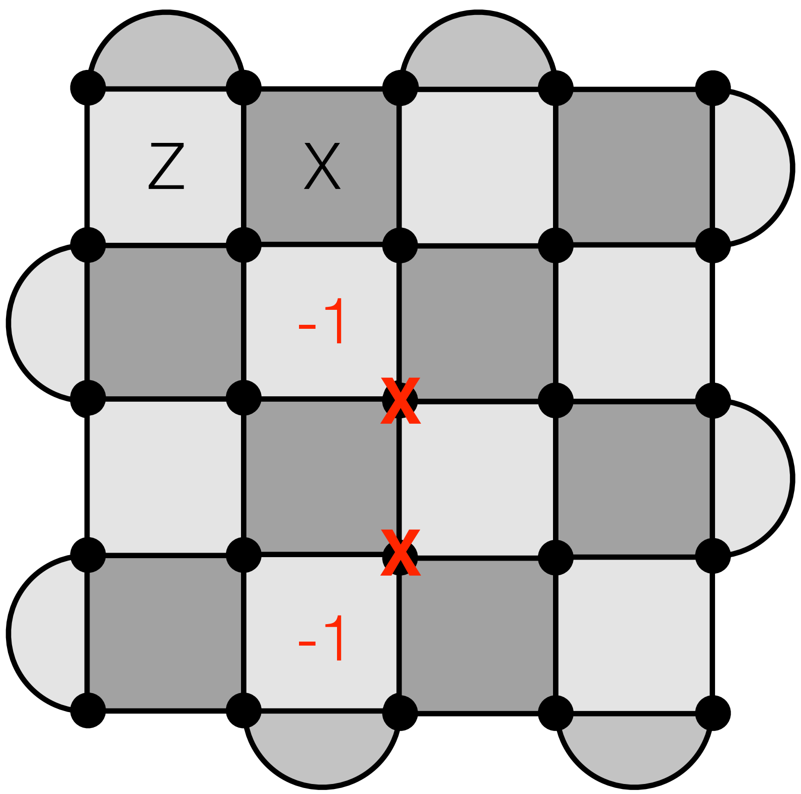


E_2 E_2' E_3

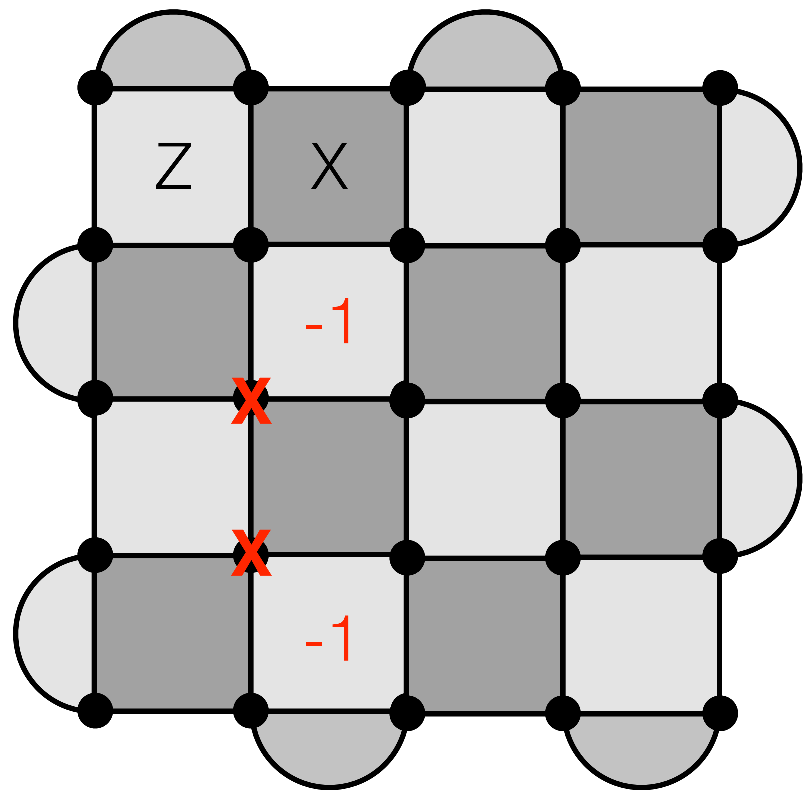
Decoding errors

Decoding errors can be reduced to an efficiently solvable graph problem.

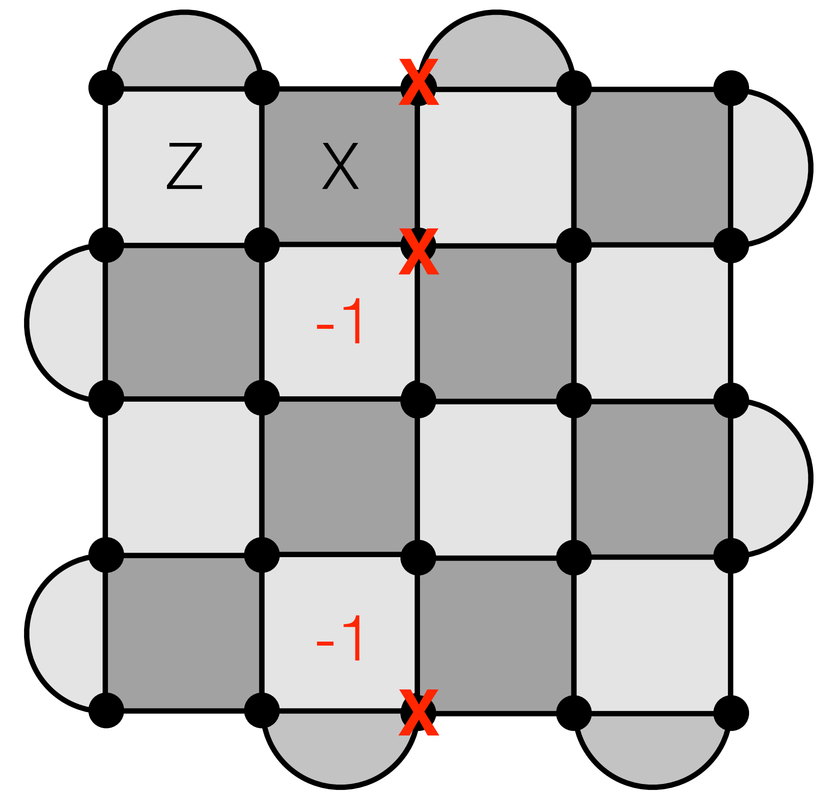
Consider these three sets of errors that all result in the same syndrome:



E_2



E_2'



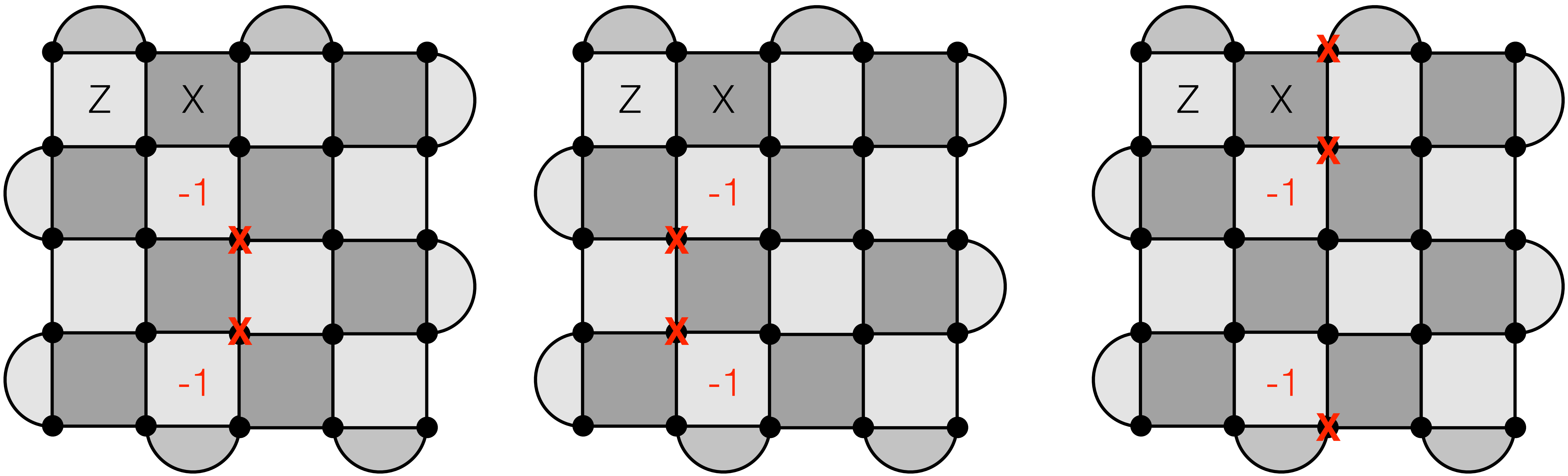
E_3

Which one should you pick, and what should you do?

Decoding errors

Decoding errors can be reduced to an efficiently solvable graph problem.

Consider these three sets of errors that all result in the same syndrome:



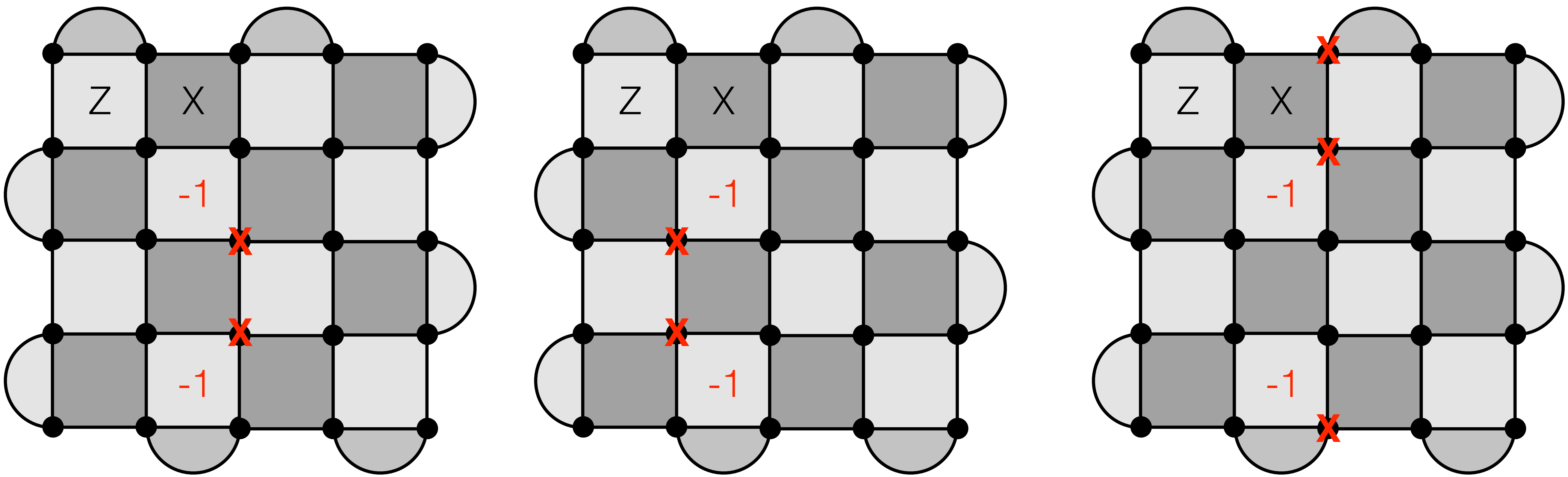
E_2 E_2' E_3

E_2 and E_2' are more likely (by a factor of p,) so pick them!
But which one?

Decoding errors

Decoding errors can be reduced to an efficiently solvable graph problem.

Consider these three sets of errors that all result in the same syndrome:



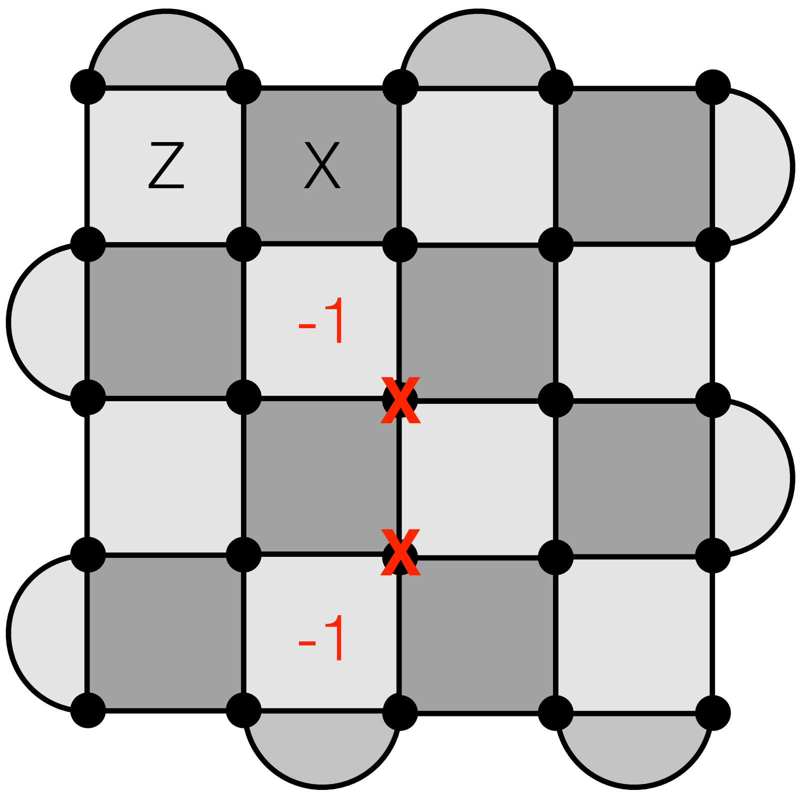
E_2 E_2' E_3

It doesn't matter. The corrective actions of fixing E_2 and E_2' are equivalent.
The surface code is degenerate.

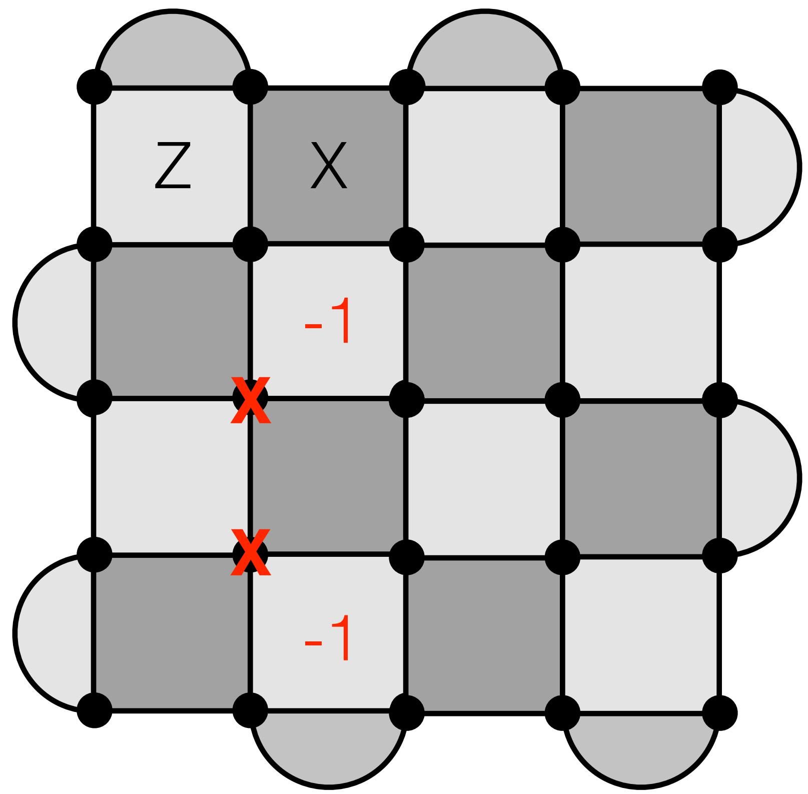
Decoding errors

Decoding errors can be reduced to an efficiently solvable graph problem.

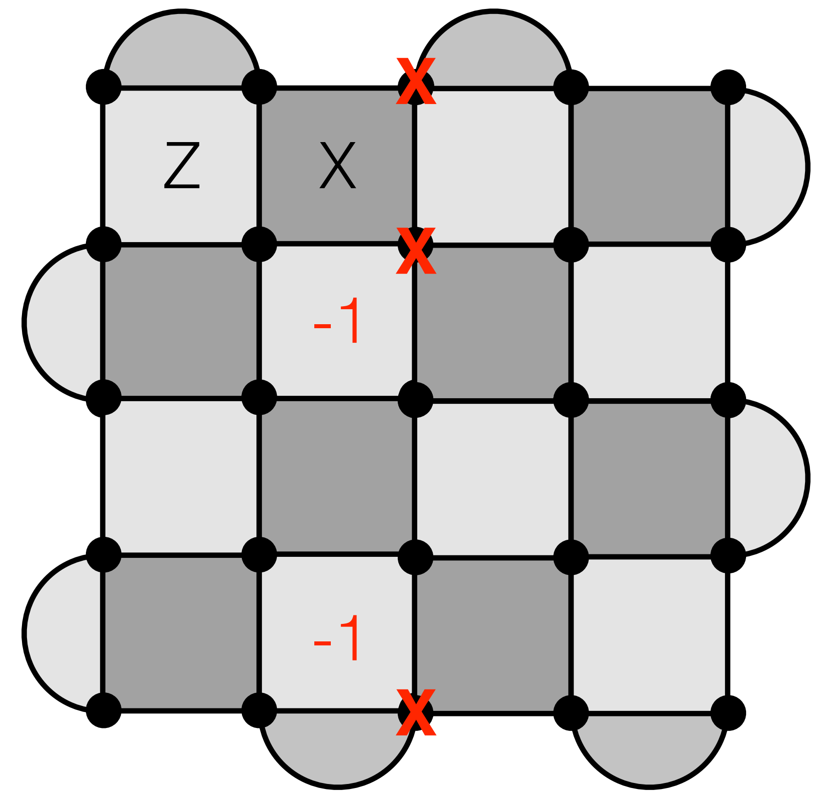
Consider these three sets of errors that all result in the same syndrome:



E_2



E_2'



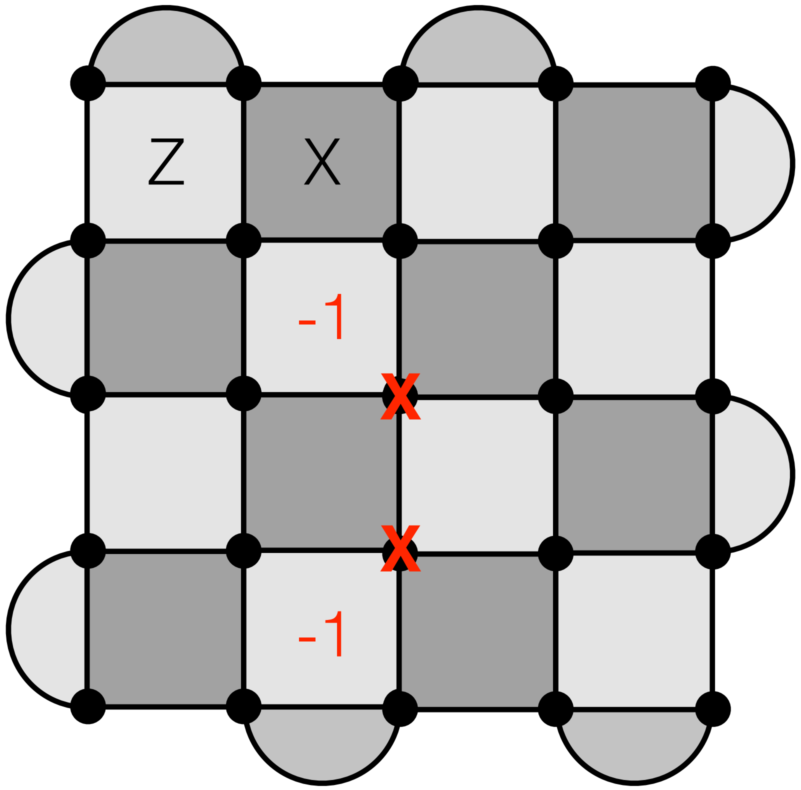
E_3

But what if it was E_3?

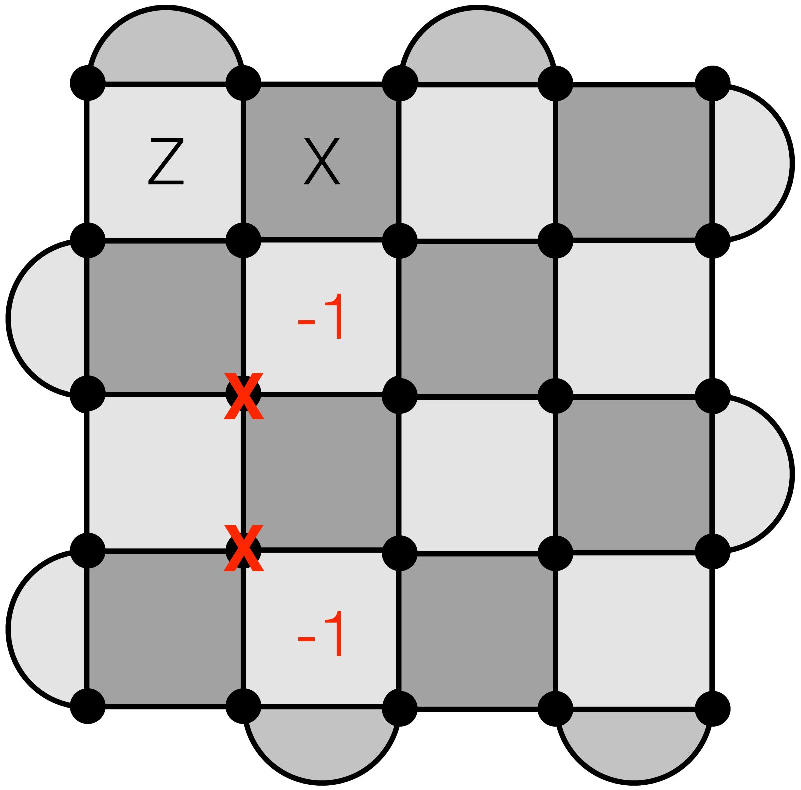
Decoding errors

Decoding errors can be reduced to an efficiently solvable graph problem.

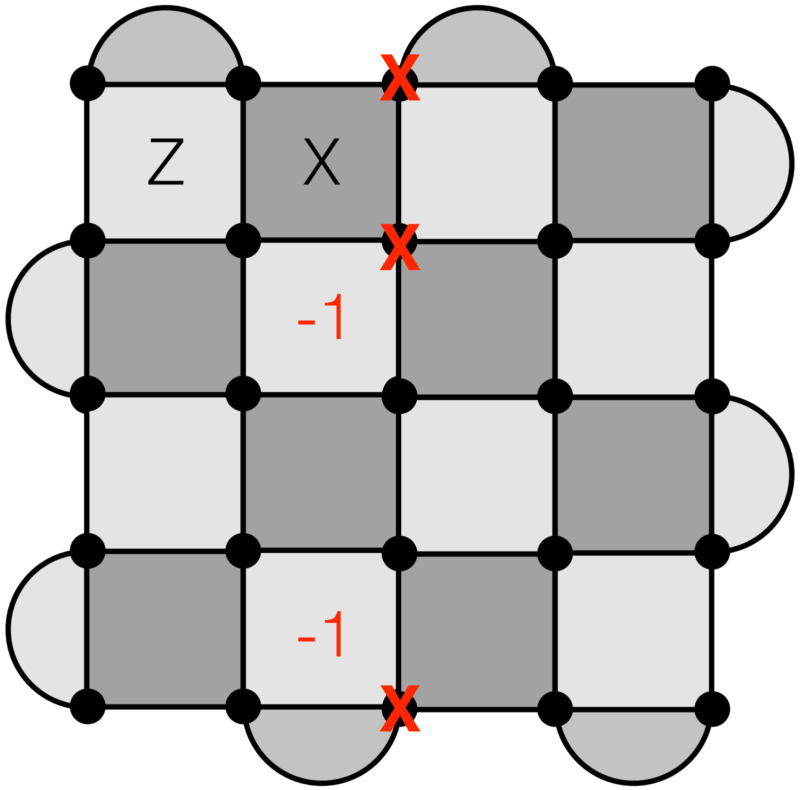
Consider these three sets of errors that all result in the same syndrome:



E_2



E_2'



E_3

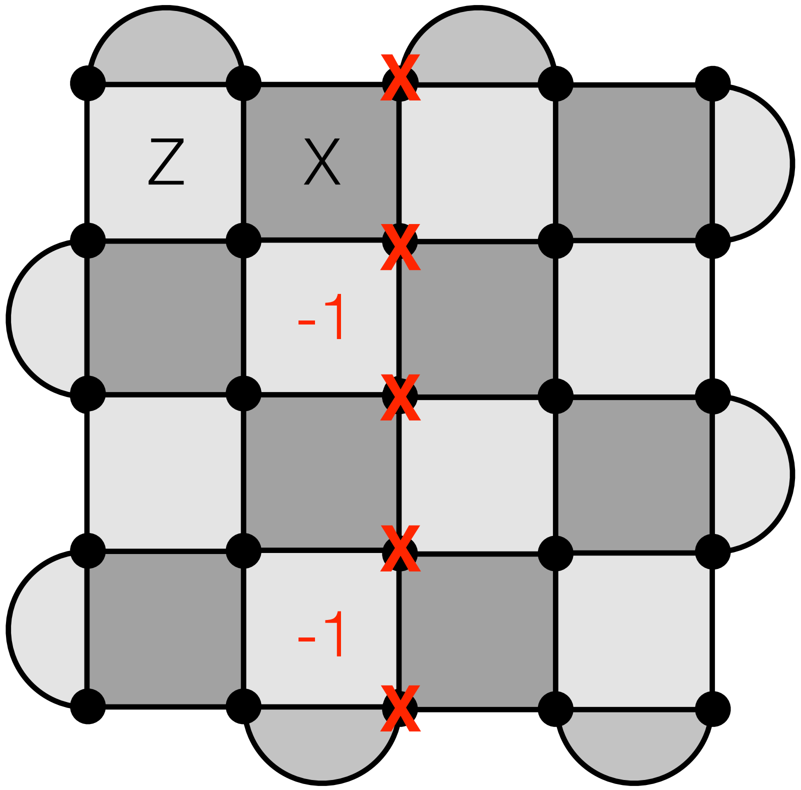
But what if it was E_3?

Well, our decoder will pick E_2, and will apply E_2 as a correction. This is bad, because $E_2 \otimes E_3$ is a logical X_L!

Decoding errors

Decoding errors can be reduced to an efficiently solvable graph problem.

Consider these three sets of errors that all result in the same syndrome:



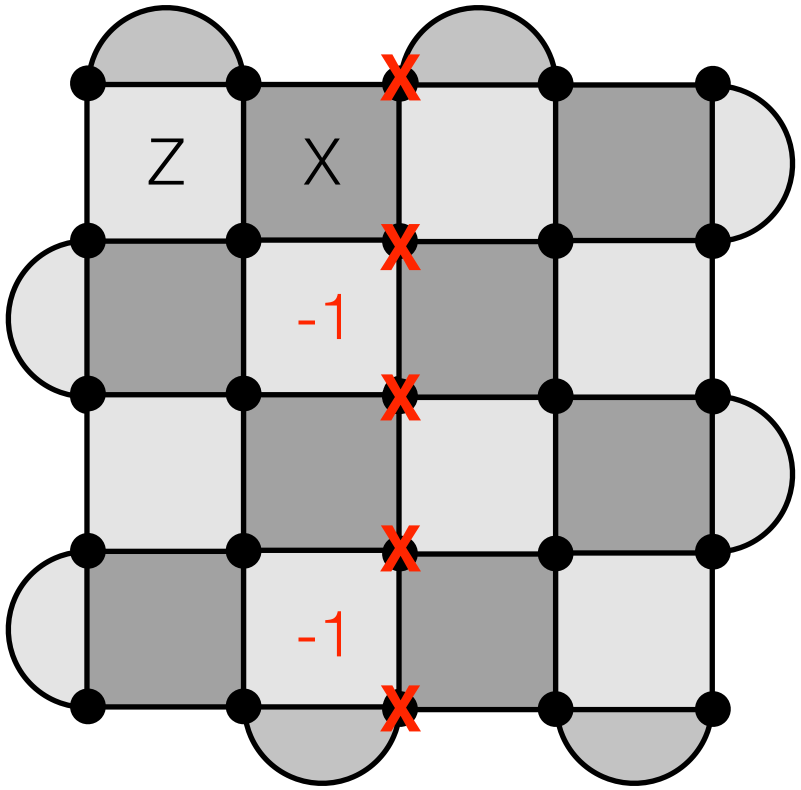
$$E_2 \otimes E_3$$

But what if it was E_3?
Well, our decoder will pick E_2, and will apply E_2 as a correction. This is bad, because $E_2 \otimes E_3$ is a logical X$_L$!

Decoding errors

Decoding errors can be reduced to an efficiently solvable graph problem.

Consider these three sets of errors that all result in the same syndrome:



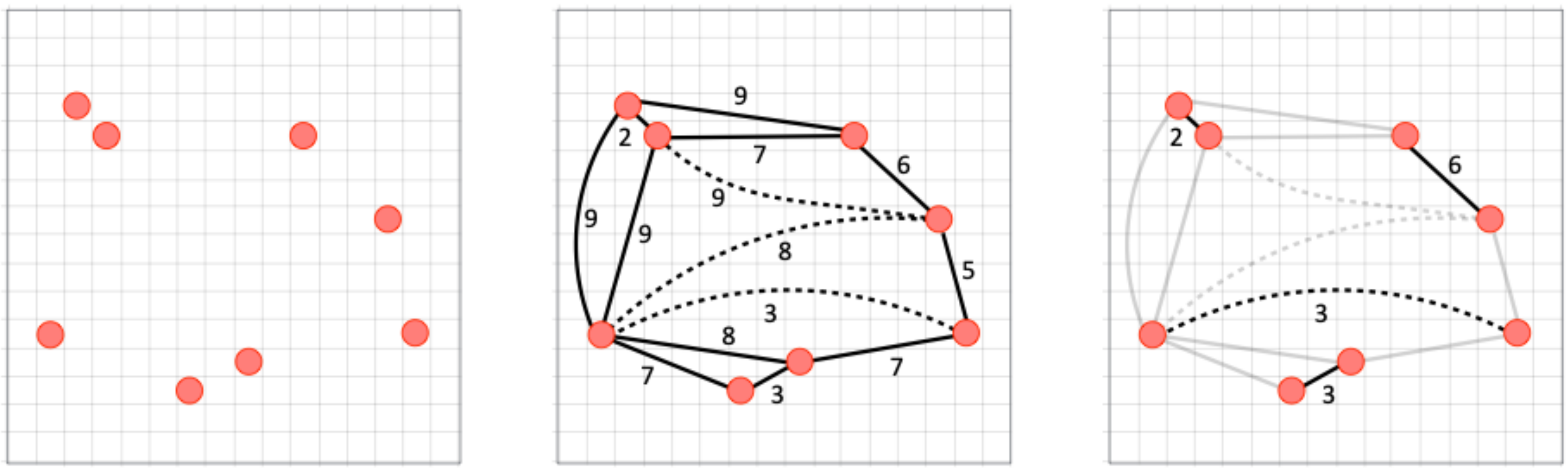
$$E_2 \otimes E_3$$

This shouldn't come as a surprise. This code is distance d=5, meaning it can correct up to (d-1)/2=2 errors successfully.

E_3 involved three bit flips, so we can't expect to correct it.

Decoders

We can formalize this procedure by making a graph where the nodes are the flipped syndromes, and the edges between them are weights of possible error chains giving rise to those syndromes.



(a) (b) (c)

This can be solved using an algorithm called minimum-weight perfect matching (MWPM), which is essentially linear time in the size of the decoding graph.

Decoders

Because of the degeneracy of the code, a better strategy would have been to make a list of all possible error strings consistent with the observed syndromes, then group them into classes by whether they flip the logical operator, and pick the most likely class instead of committing to any specific error chain.

This decoder is called the most likely error (MLE) decoder, and while it is optimal, it is generally NP-hard as it requires enumerating all possible errors. In the special case of the surface code with local connectivity, an efficient and accurate approximate version using tensor network states has been developed:

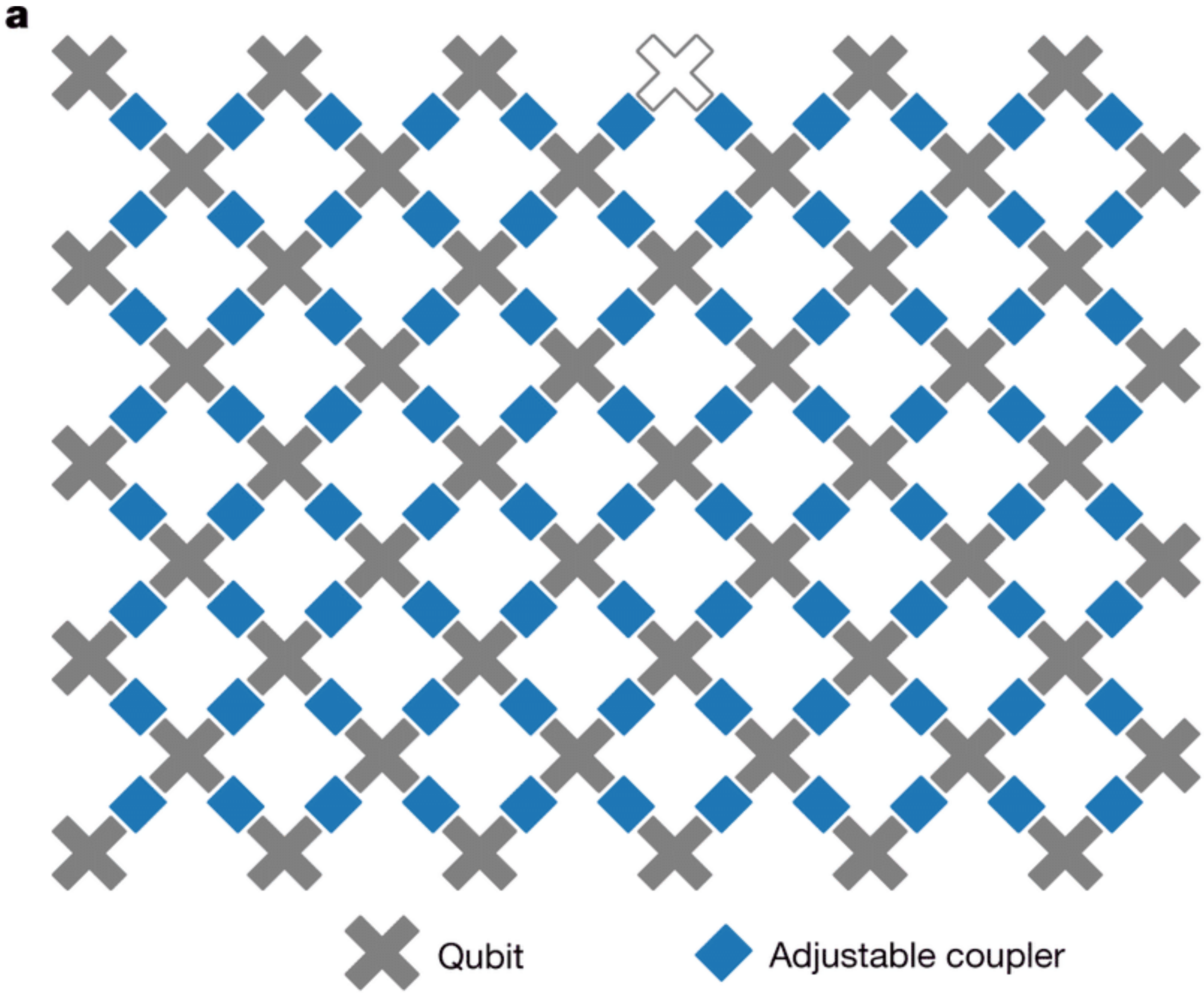
Sergey Bravyi, Martin Suchara, and Alexander Vargo. “Efficient Algorithms for Maximum Likelihood Decoding in the Surface Code”. Physical Review A 90.3 (Sept. 2014), p. 032326. arXiv: 1405.4883 [quant-ph].

Decoders

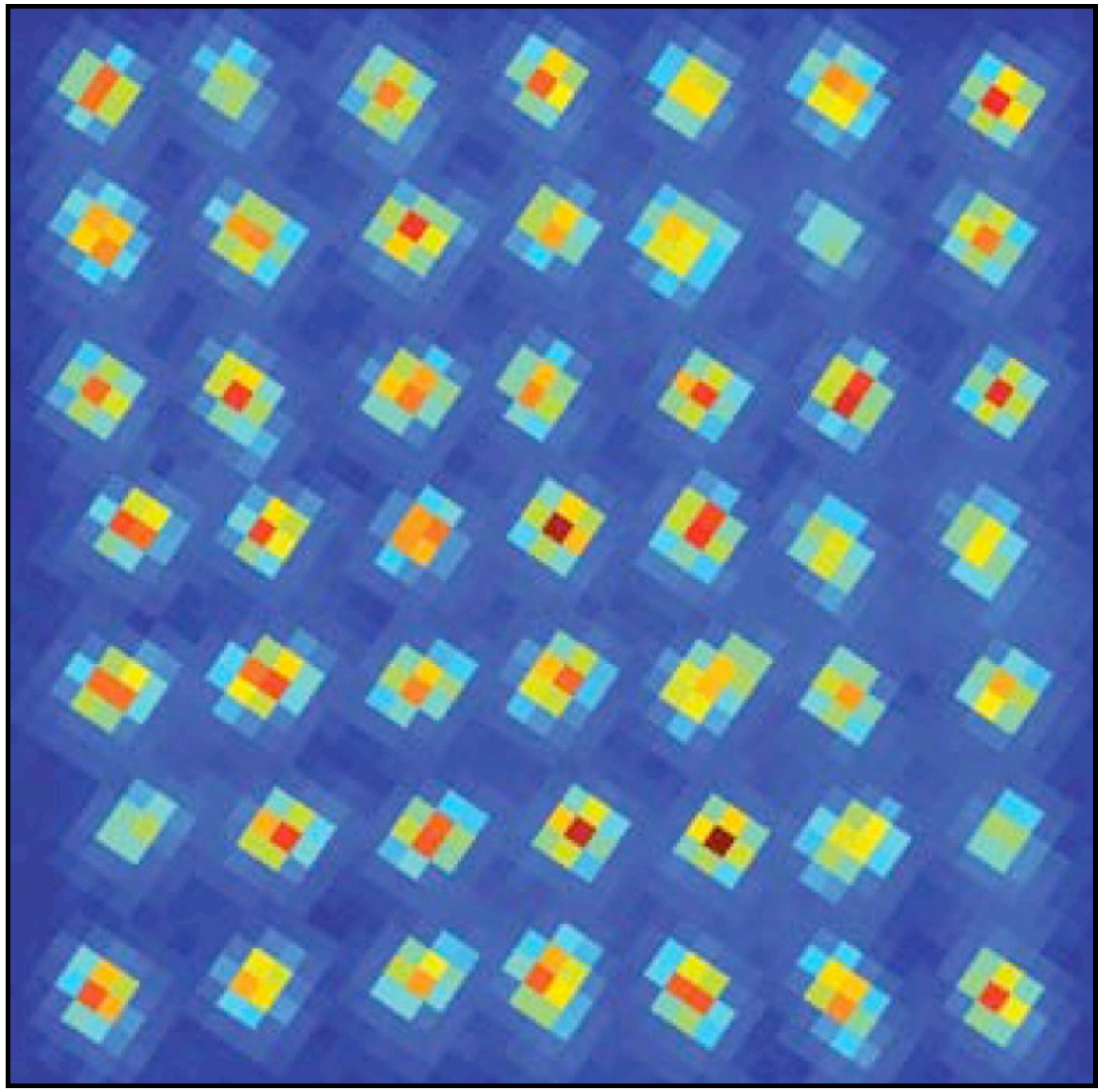
The surface code is particularly popular for two reasons:

1) It only requires nearest neighbor connectivity between qubits, which is much easier to achieve than the long range or all-to-all connectivity required for some other QEC codes.

Google Sycamore processor



Atoms in lattices



(Saffman group)

Decoders

The surface code is particularly popular for two reasons:

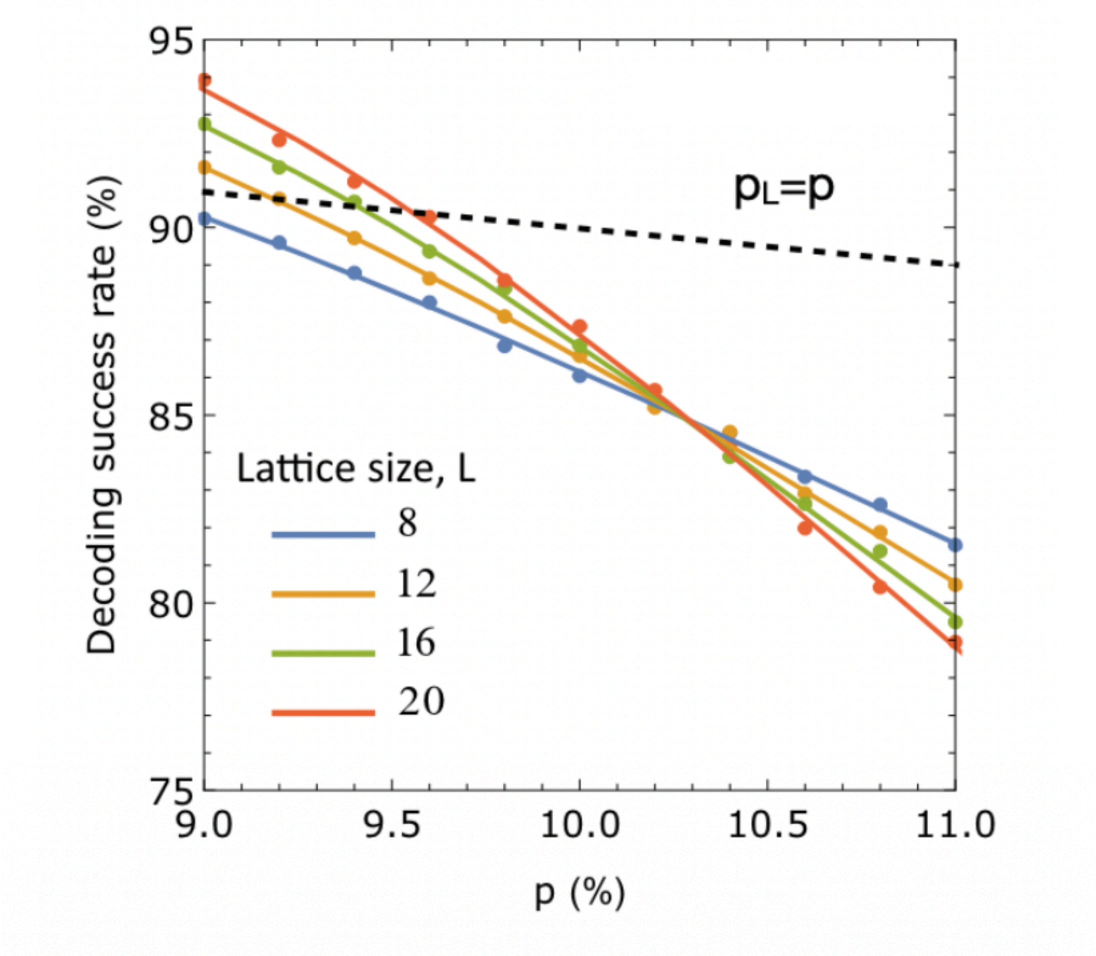
1) It only requires nearest neighbor connectivity between qubits, which is much easier to achieve than the long range or all-to-all connectivity required for some other QEC codes.

2) The error threshold for the code is relatively high.

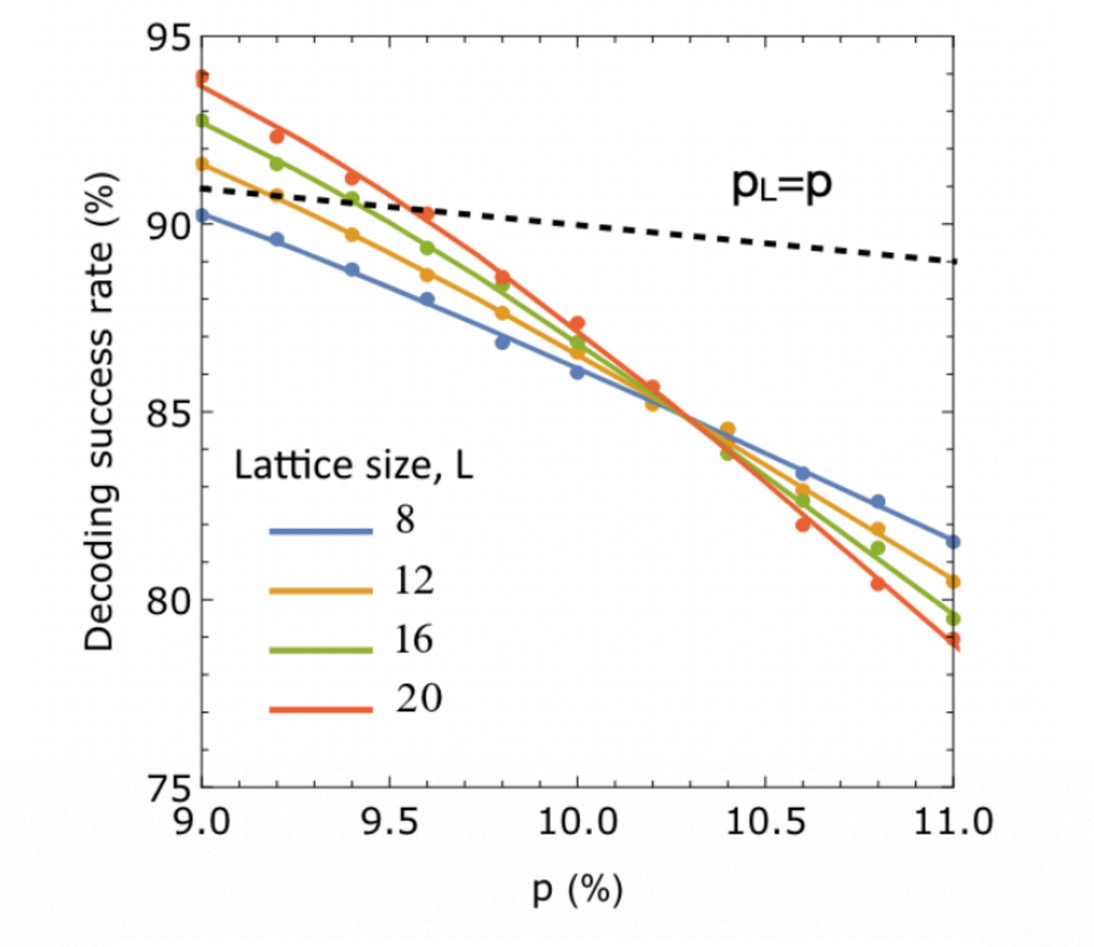
Thresholds

An important property of a code family, error model and decoder is the threshold. The threshold specifies the error rate below which the logical error probability is suppressed with d.

The threshold is usually demonstrated by sampling errors stochastically and running the decoder to see if a correct decoding was obtained:



Thresholds



$$p_L = A(d)\left(\frac{p}{p_{th}}\right)^{\nu}$$

$$\nu = (d+1)/2$$

$$A(d) \sim \mathrm{poly}(d)$$

Thresholds

For surface code, the threshold is 10.3% for the MWPM decoder.

The surface code threshold increases to 10.93% with the optimal decoder.

You may have been expecting to hear $p_{th} \approx 1\%$ for the surface code – this is the circuit threshold including noisy operations. Here, we are assuming perfect operations and measurements to derive the quantum memory or code capacity threshold. We will discuss the circuit threshold and why it is lower in the next lecture.

Recent Google Surface Code QEC results

Article

Quantum error correction below the surface code threshold

https://doi.org/10.1038/s41586-024-08449-y

Received: 24 August 2024

Accepted: 25 November 2024

Published online: 9 December 2024

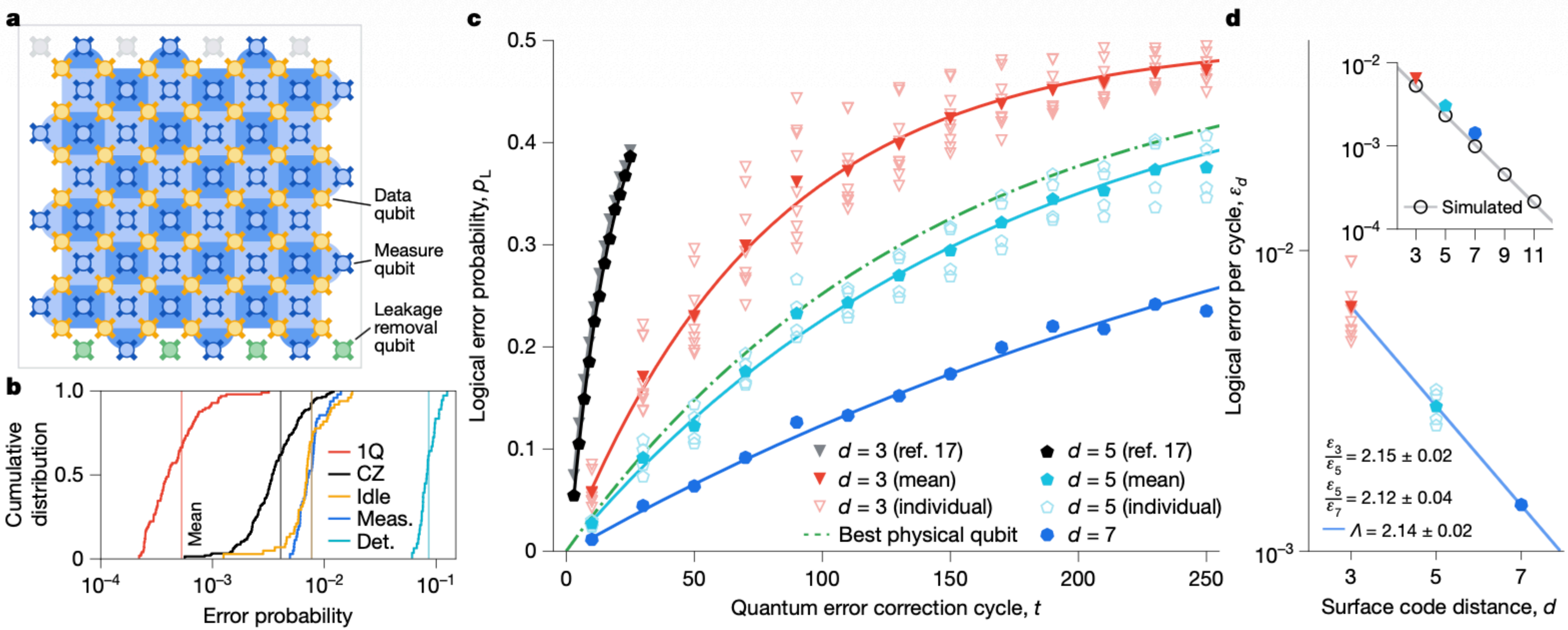
Open access

Check for updates

Google Quantum AI and Collaborators*

Quantum error correction[1–4] provides a path to reach practical quantum computing by combining multiple physical qubits into a logical qubit, in which the logical error rate is suppressed exponentially as more qubits are added. However, this exponential suppression only occurs if the physical error rate is below a critical threshold. Here we present two below-threshold surface code memories on our newest generation of superconducting processors, Willow: a distance-7 code and a distance-5 code integrated with a real-time decoder. The logical error rate of our larger quantum memory is suppressed by a factor of $\Lambda = 2.14 \pm 0.02$ when increasing the code distance by 2, culminating in a 101-qubit distance-7 code with $0.143\% \pm 0.003$ per cent error per cycle of error correction. This logical memory is also beyond breakeven, exceeding the lifetime of its best physical qubit by a factor of 2.4 ± 0.3. Our system maintains below-threshold performance when decoding in real time, achieving an average decoder latency of 63 microseconds at distance 5 up to a million cycles, with a cycle time of 1.1 microseconds. We also run repetition codes up to distance 29 and find that logical performance is limited by rare correlated error events, occurring approximately once every hour or 3×10^9 cycles. Our results indicate device performance that, if scaled, could realize the operational requirements of large-scale fault-tolerant quantum algorithms.

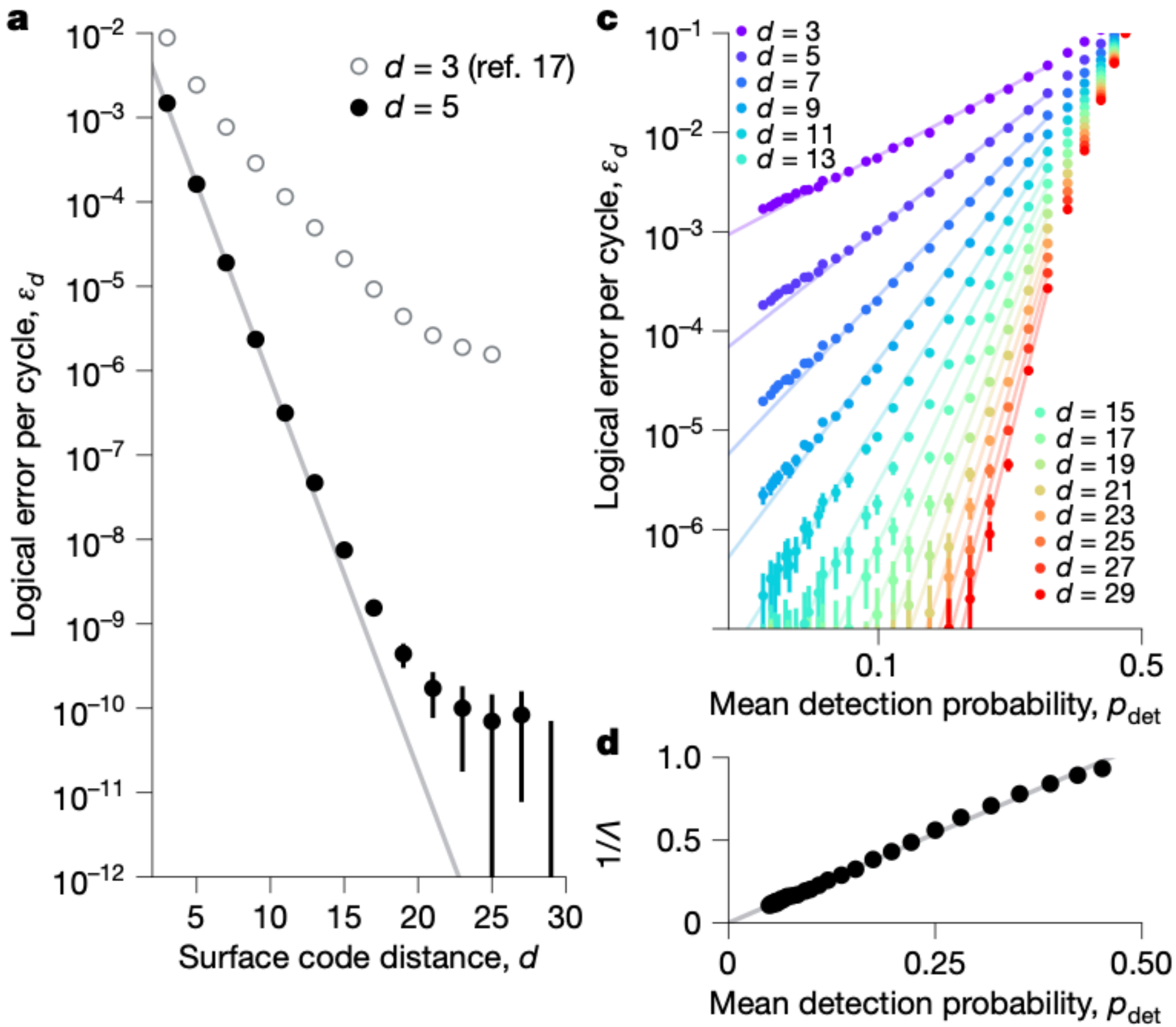
Recent Google Surface Code QEC results



105 total qubits

$$\Lambda = \varepsilon_d / \varepsilon_{d+2} \approx p_{\mathrm{thr}} / p$$

Recent Google Surface Code QEC results



(Repetition code)