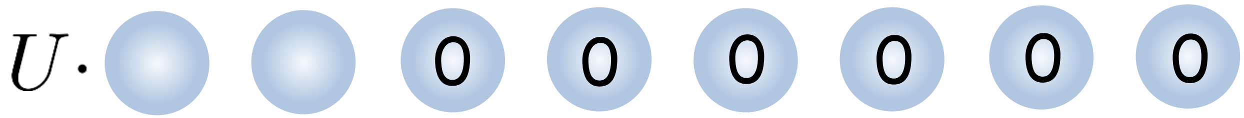
Fault-Tolerant Quantum Computing by Code Deformation

Sergey Bravyi

IBM Watson Center

QIP Tutorial
2016

Stabilizer codes: summary



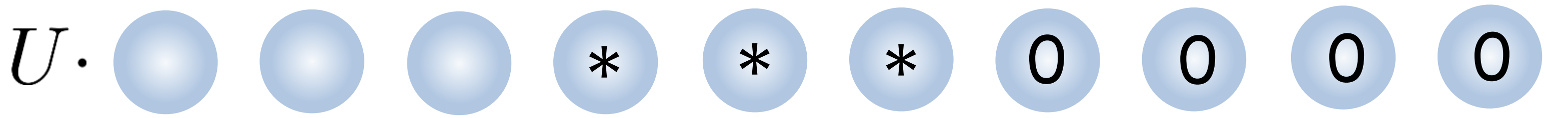
Encode k logical qubits into a subspace of n physical qubits. Encoded states are eigenvectors of commuting Pauli operators called stabilizers.

Errors are diagnosed by their syndromes (stabilizers whose eigenvalue has been flipped).

The code distance d quantifies how well the code protects encoded information.

Any error of weight at most $(d-1)/2$ is correctable.

Subsystem codes: summary



Stabilizer codes with extra "gauge qubits". Needed to describe a conversion between stabilizer codes.

Gauge qubits do not store any information.

The purpose of gauge qubits is to describe stabilizers whose eigenvalue can be flipped as a result of the code conversion itself.

Homological CSS codes: logical operators can be described by cycles in a graph. The code distance can be computed efficiently by Dijkstra's algorithm.

Code deformation: summary

Computation is driven by syndrome measurements, error correction, and transversal gates.

Elementary code deformations

1. Stop measuring some existing stabilizers.
   Use them to make new logical or gauge qubits.
2. Start measuring some new stabilizers.
   A new stabilizer can be made of logical or gauge operators.
3. Transversal logical gates

4. Choose a new basis set of generators for syndrome/gauge/logical subsystems

Code deformation: summary

Computation is driven by syndrome measurements, error correction, and transversal gates.

At each step the logical state is encoded into a subsystem quantum code.

A code deformation is fault-tolerant (corrects t errors per step) if each intermediate subsystem code has large enough distance (at least $2t + 1$).

Outline

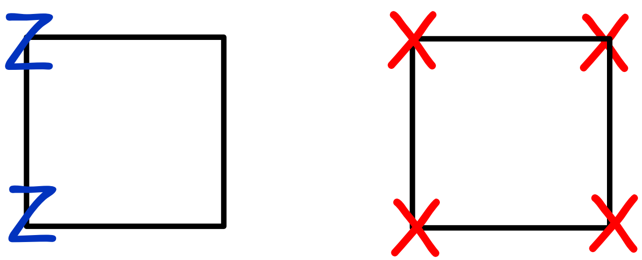
- Stabilizer codes
- The decoding problem and code distance
- Fault-tolerant code deformation
- Example 1: Shor's 4-qubit code
- Example 2: lattice surgery
- Maximum likelihood decoding

Shor's 4-qubit code

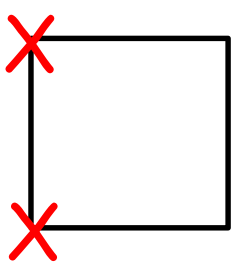
Code parameters: [[4,1,2]]

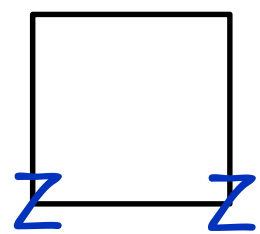
Smallest distance-2 code

Stabilizers:

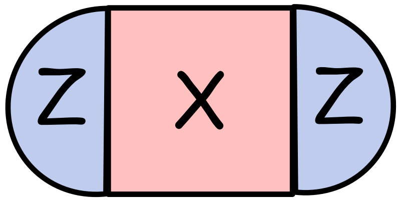


Logical operators:

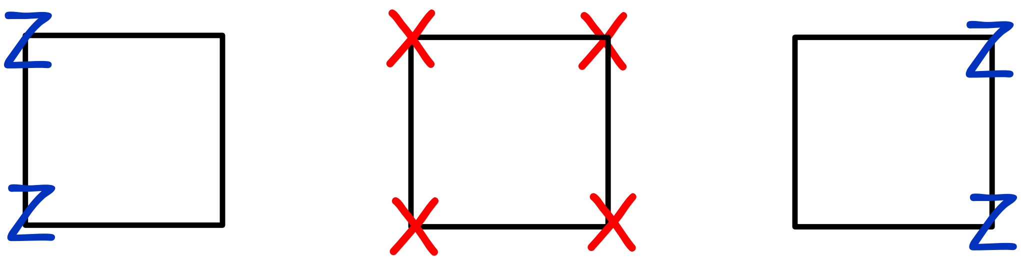
$\overline{X} =$ 

$\overline{Z} =$ 

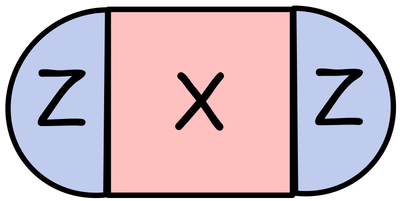
Pictorial notations for stabilizers:



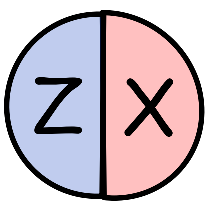
Qubits = vertices
Stabilizers = faces



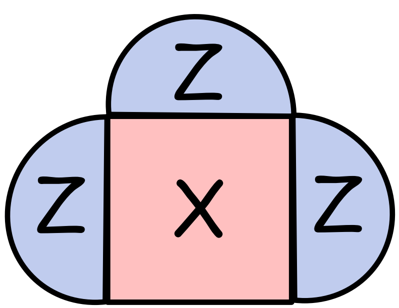
Pictorial notations for stabilizers:



Qubits = vertices
Stabilizers = faces

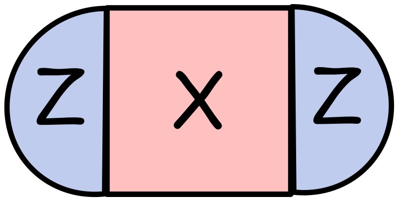


EPR state $|00\rangle + |11\rangle$

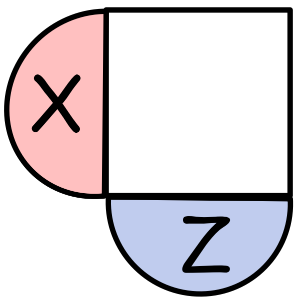


GHZ state $|0000\rangle + |1111\rangle$

Pictorial notations for stabilizers:



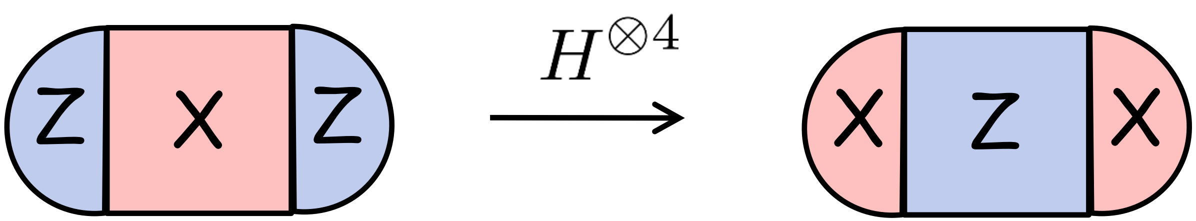
Qubits = vertices
Stabilizers = faces



logical operators
of the Shor's code

Goal: implement logical Hadamard gate for the Shor's code by code deformation

Let's try transversal Hadamard gate:

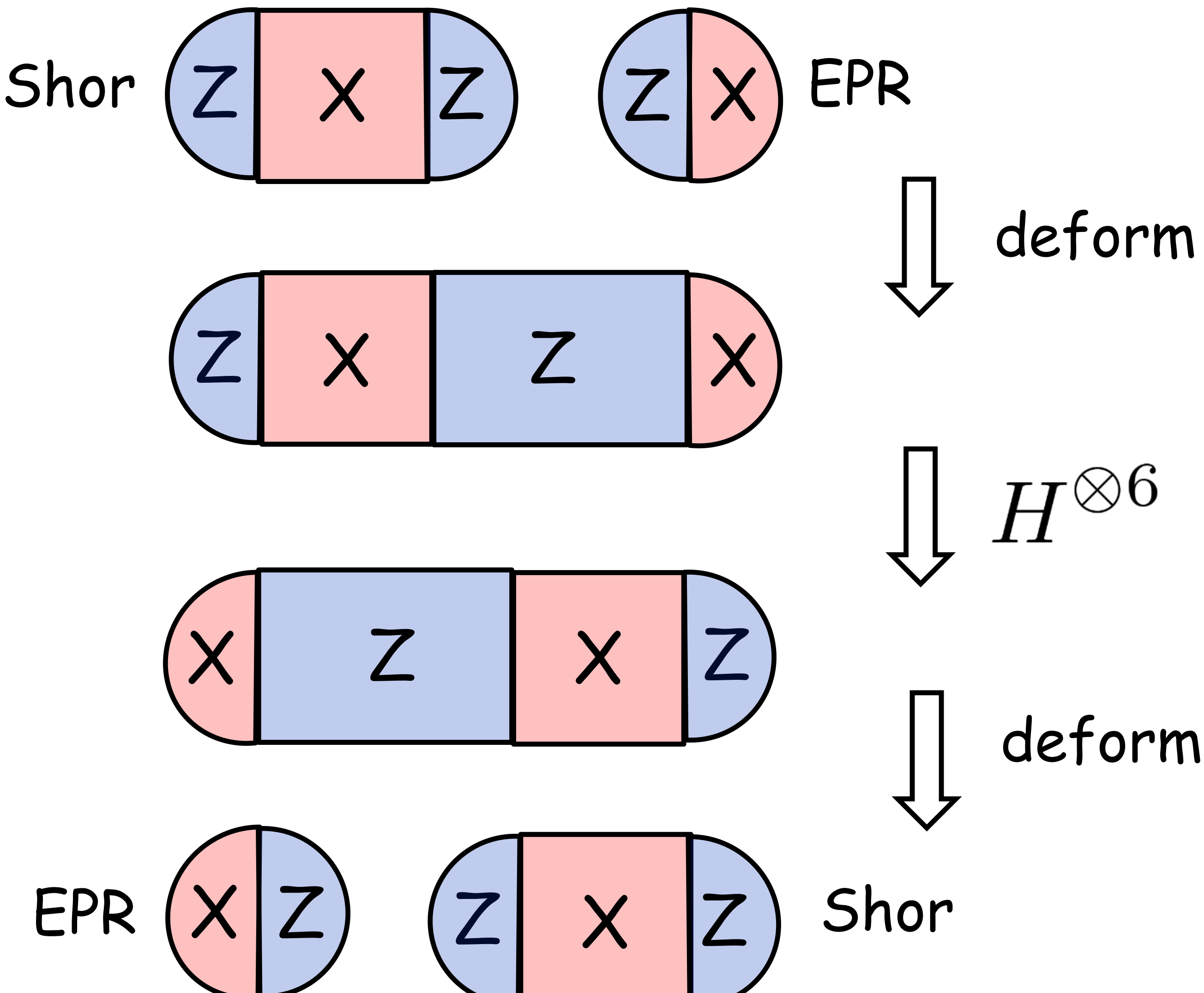


Wrong number of X and Z stabilizers

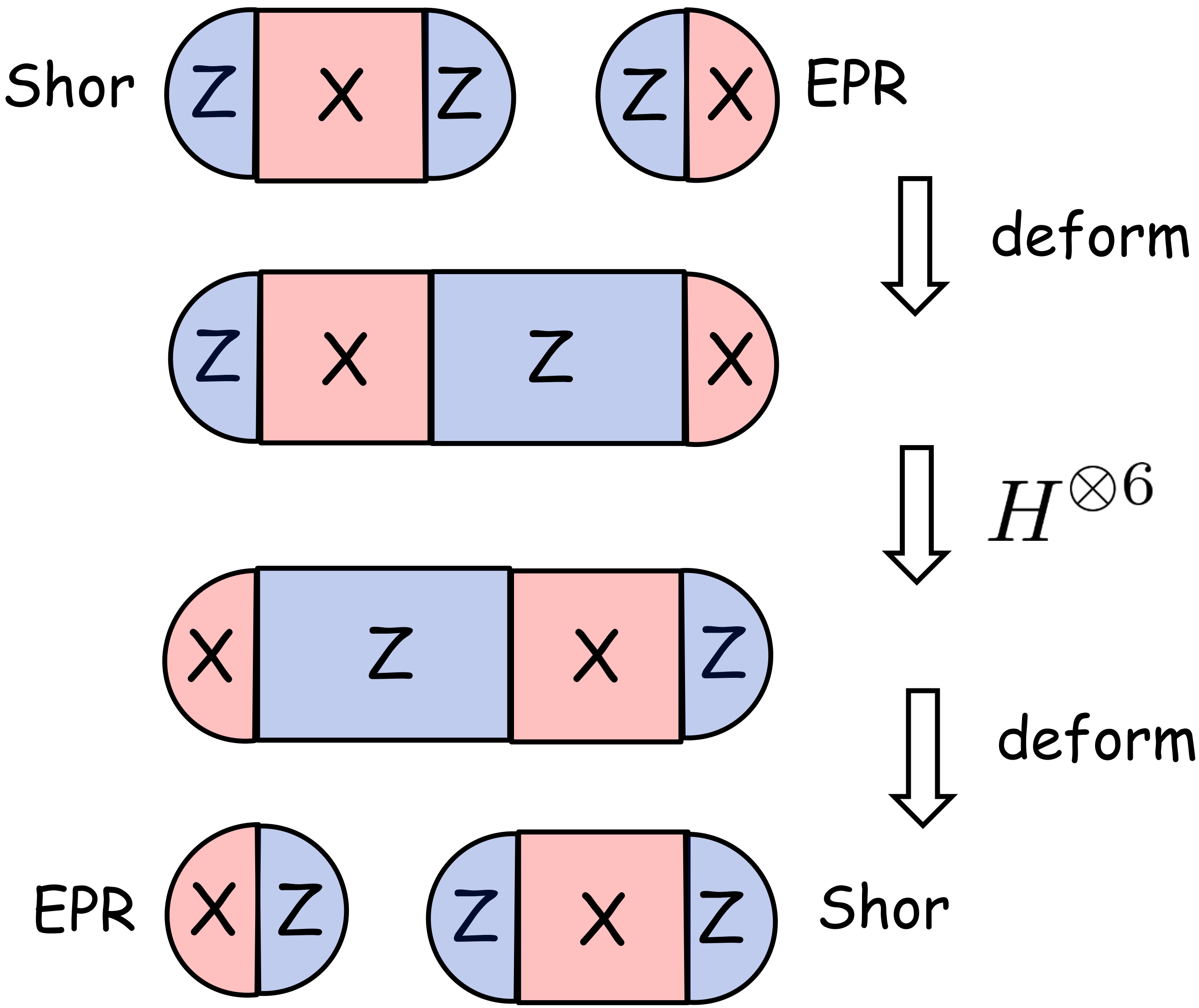
Transversal Hadamard does not preserve the code space.

Applying Hadamard to a subset of qubits doesn't help.

Logical Hadarmard: outline



For the next Hadamard gate use the reverse order of deformations to avoid shifting the code



We shall choose the deformations that implement a logical identity gate:

$$\overline{X} \quad \overline{Z}$$

$\Downarrow$ deform

$$\overline{X} \quad \overline{Z}$$

$\Downarrow$ $H^{\otimes 6}$

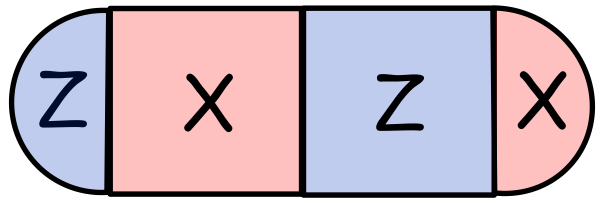
$$\overline{Z} \quad \overline{X}$$

$\Downarrow$ deform

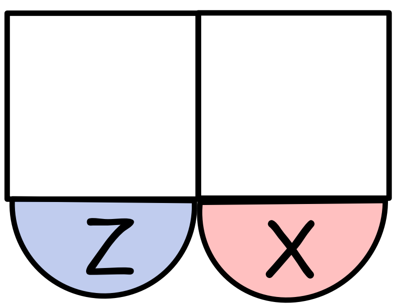
$$\overline{Z} \quad \overline{X}$$

The net effect is a logical Hadamard.

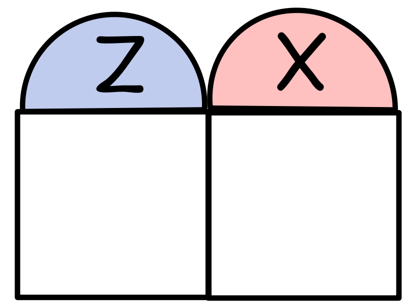
The intermediate code is a subsystem code with one gauge qubit:



Stabilizers



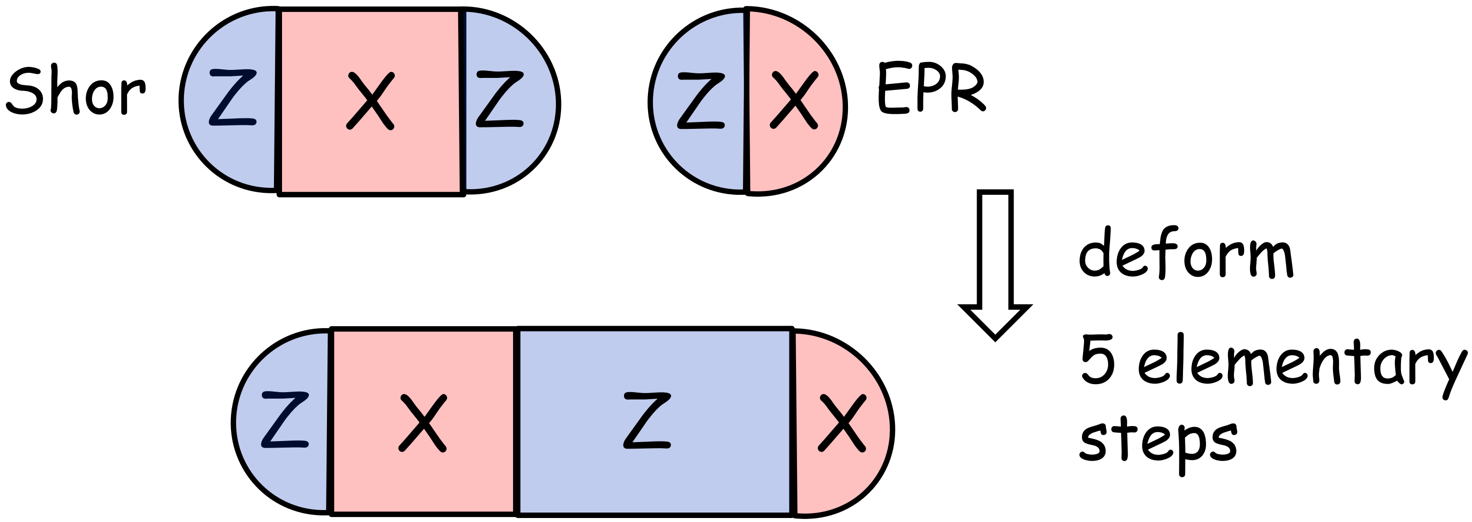
Logical operators



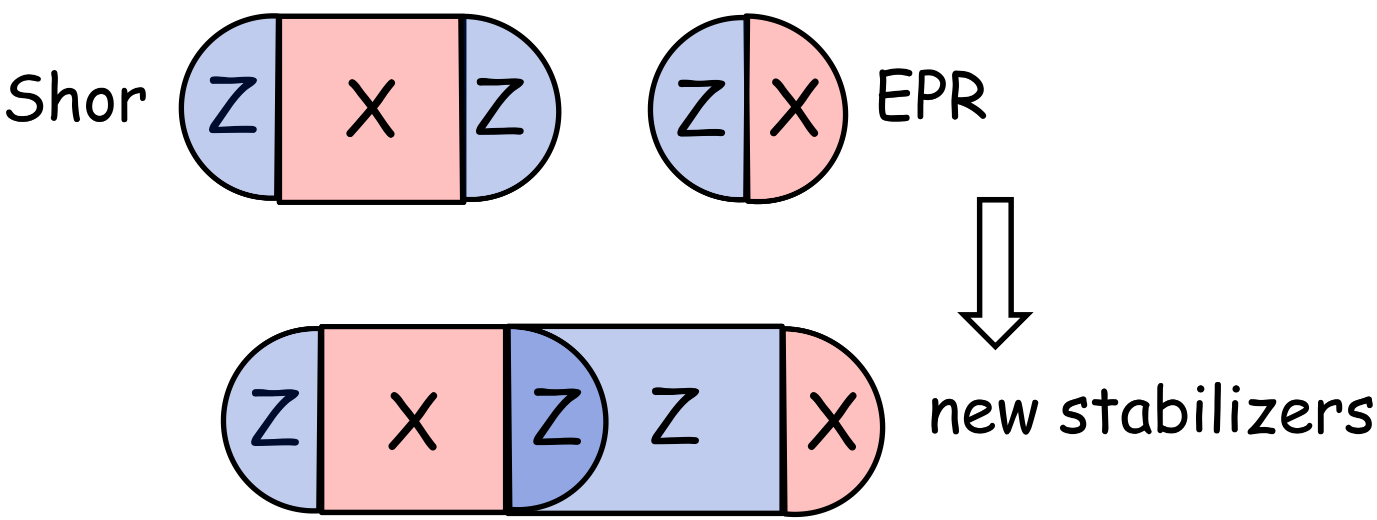
Gauge operators

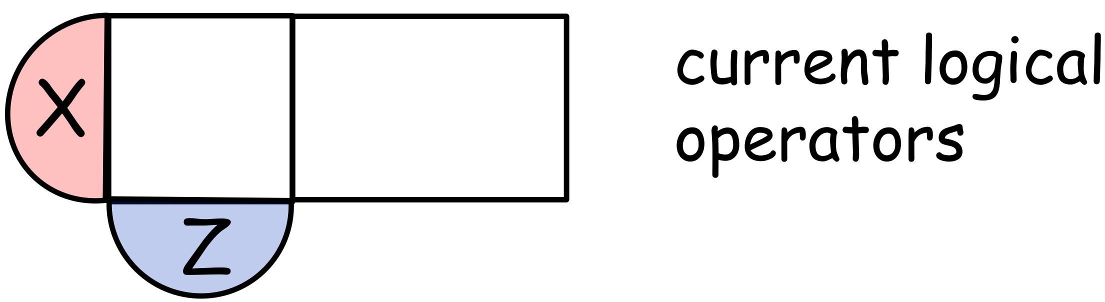
This code has distance d=2 because each qubit is touched by both X and Z stabilizers.

How to implement the deformation ?

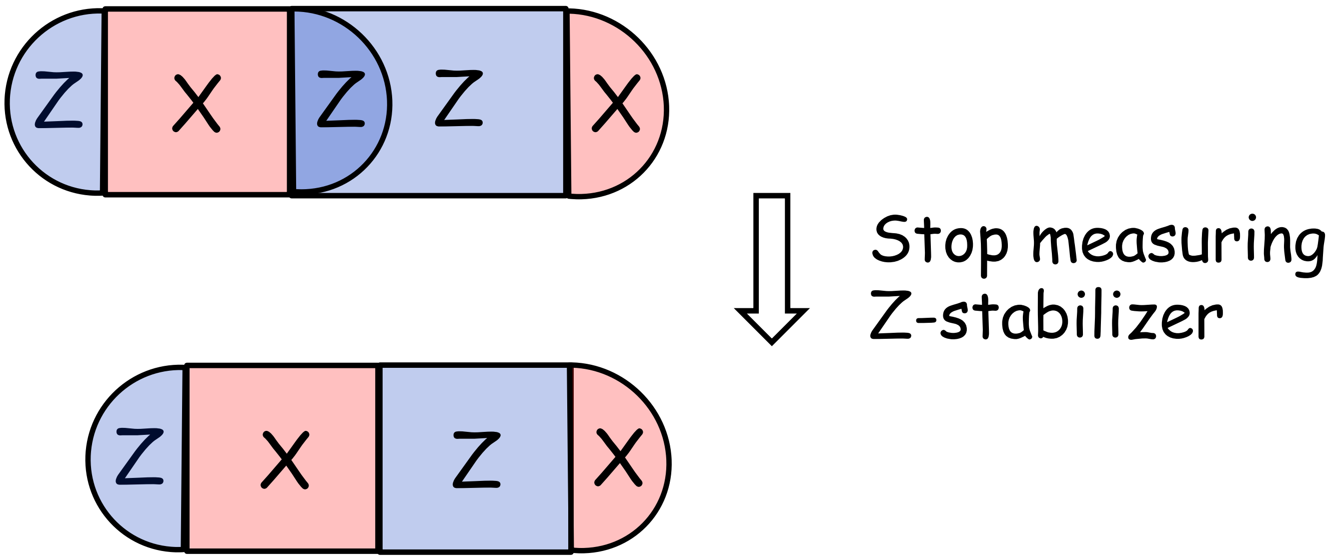


Step 1: choose a new basis set of stabilizers.

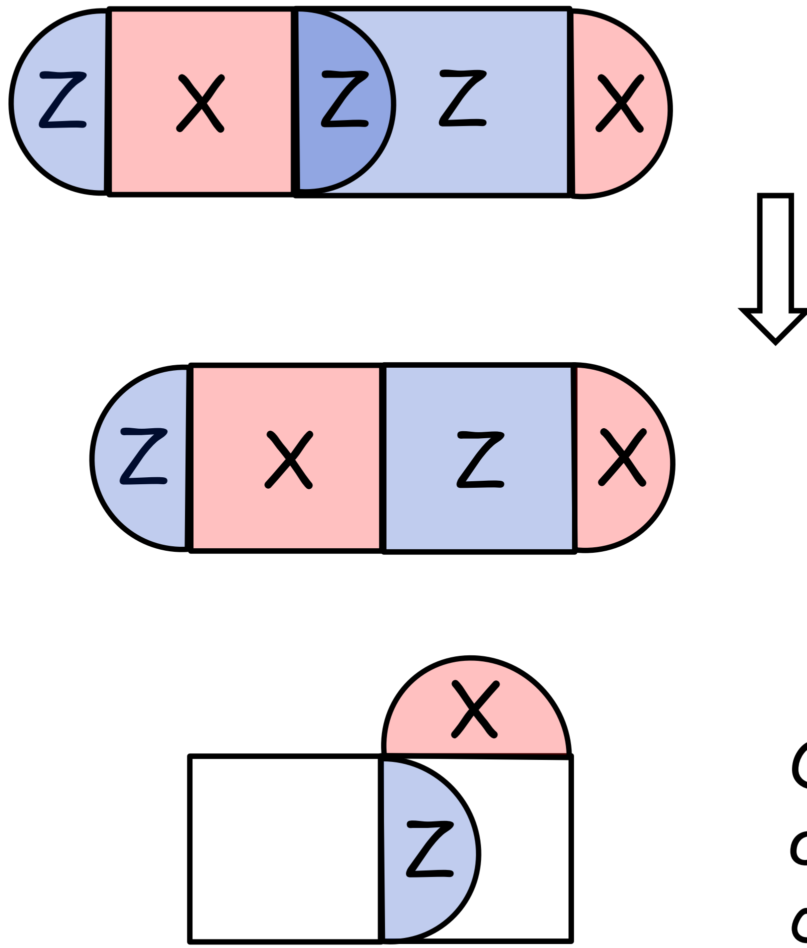




Step 2: change the stabilizer group



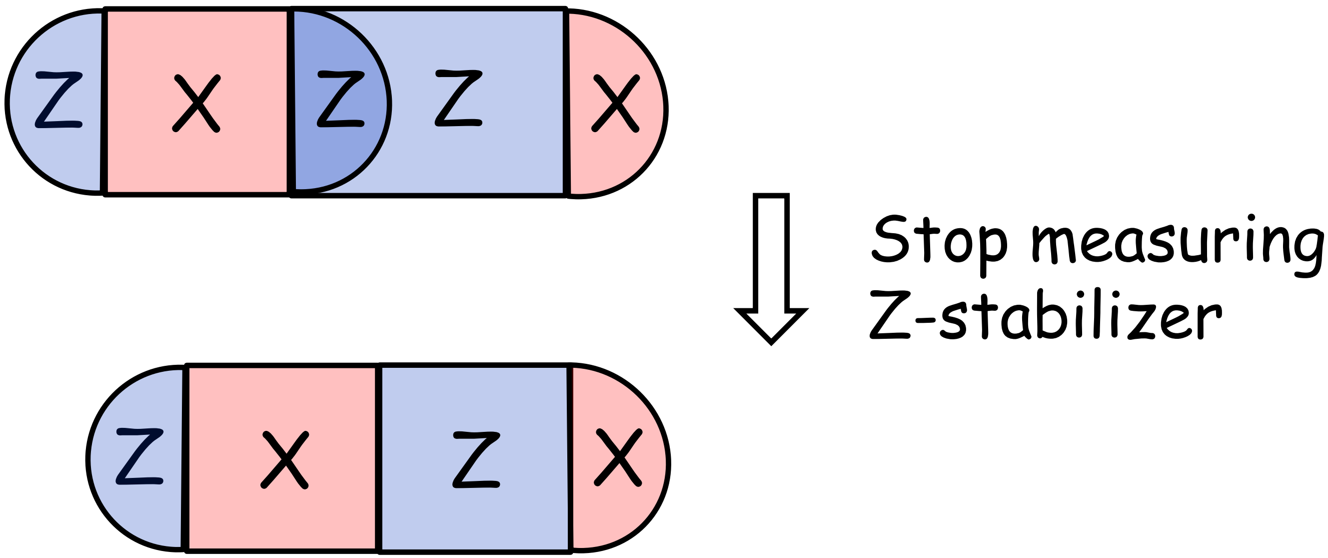
Step 2: change the stabilizer group



Stop measuring Z-stabilizer

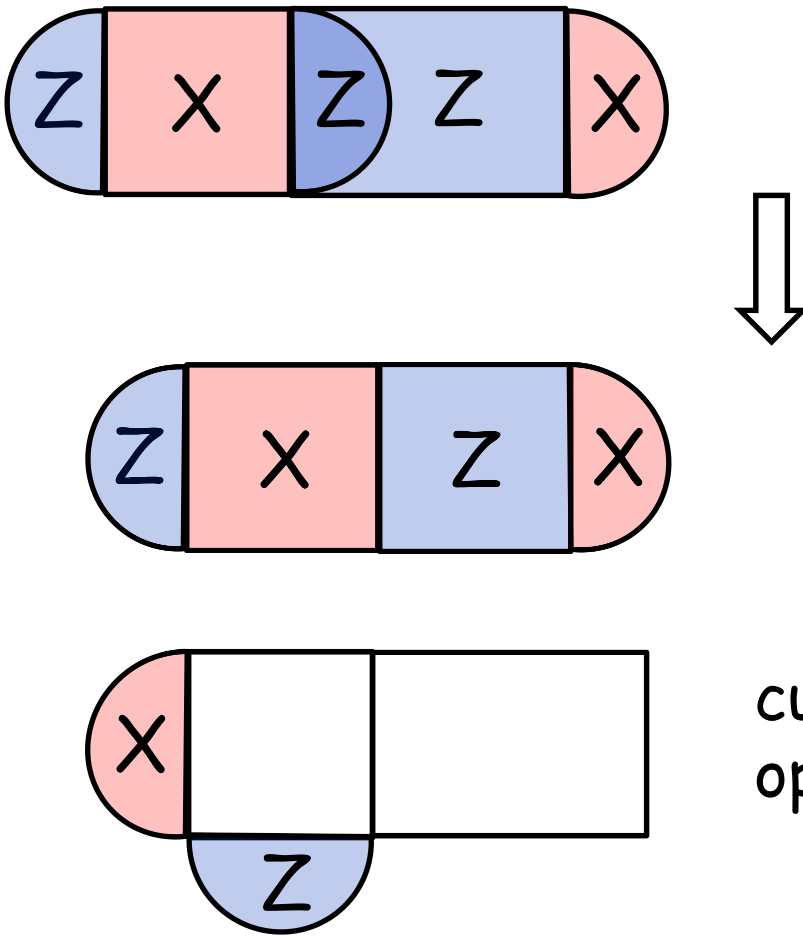
One gauge qubit has been created

Step 2: change the stabilizer group



We are not done yet...
The new code has the desired stabilizers, but wrong logical/gauge operators.

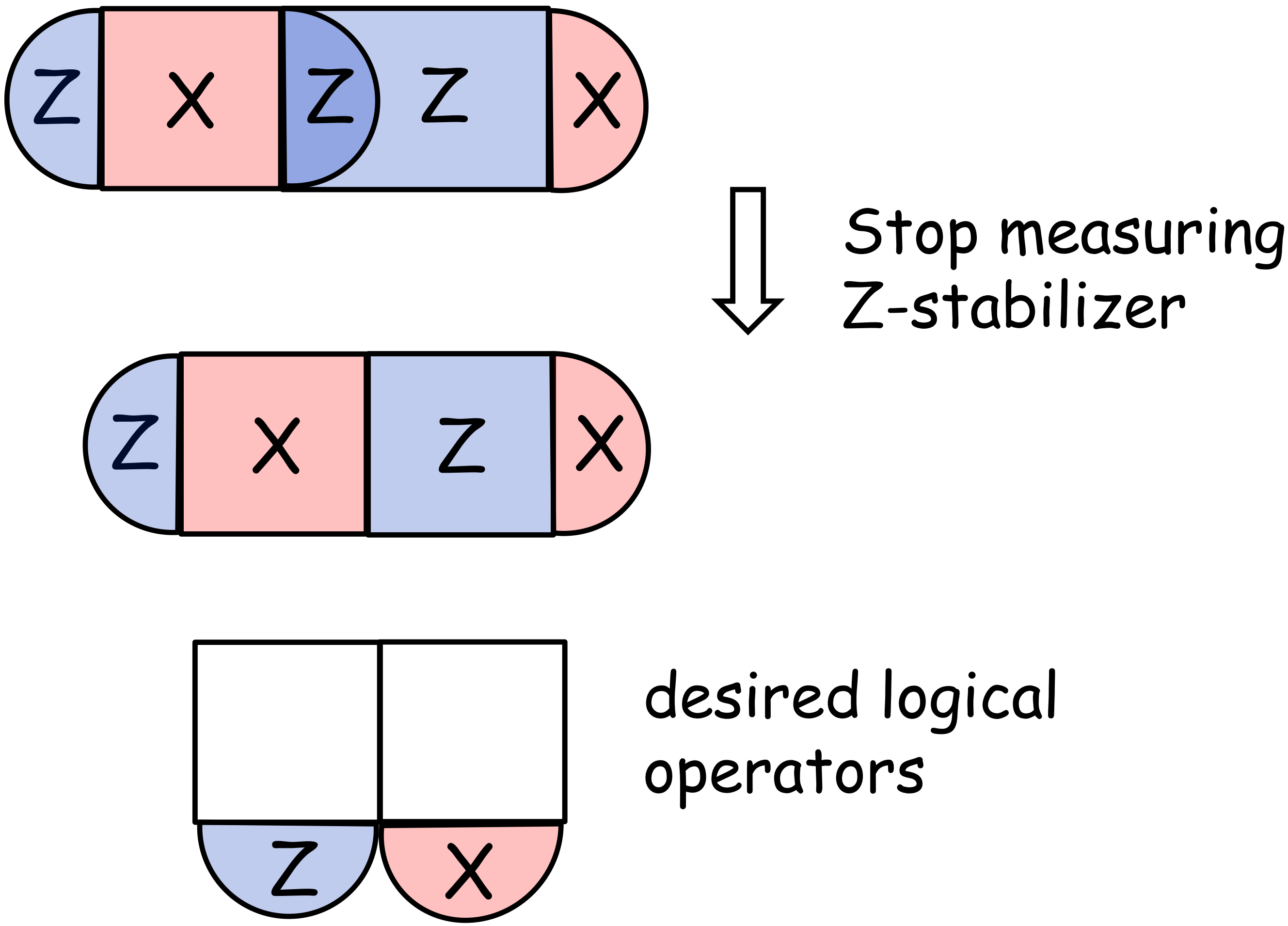
Step 2: change the stabilizer group



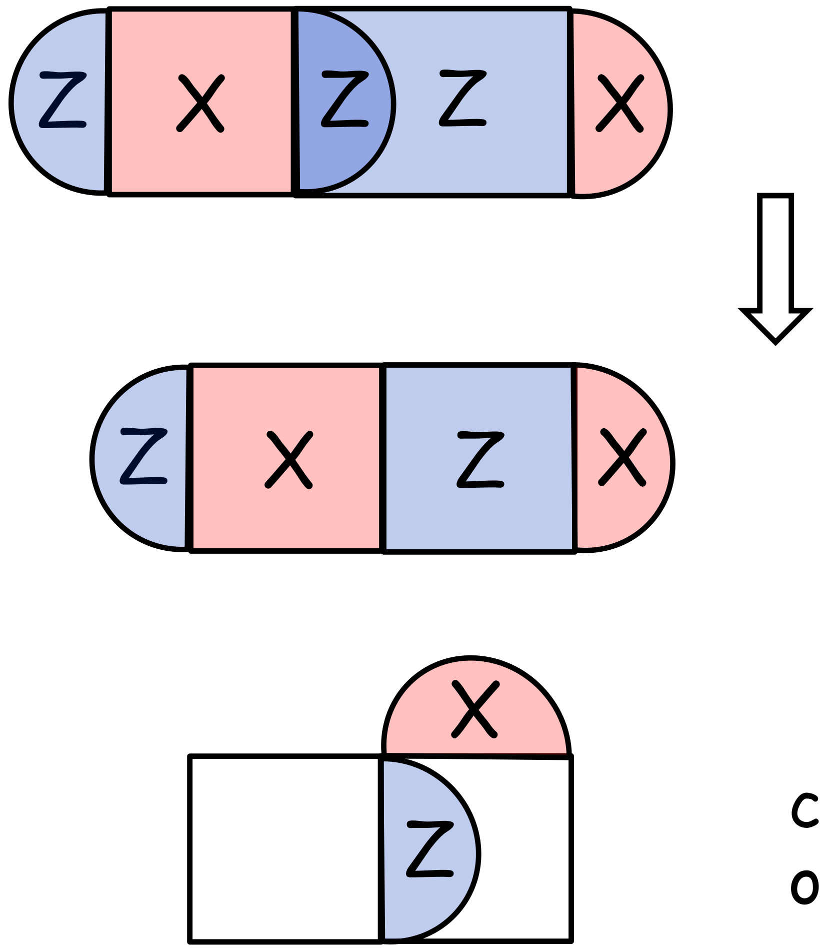
Stop measuring
Z-stabilizer

current logical
operators

Step 2: change the stabilizer group



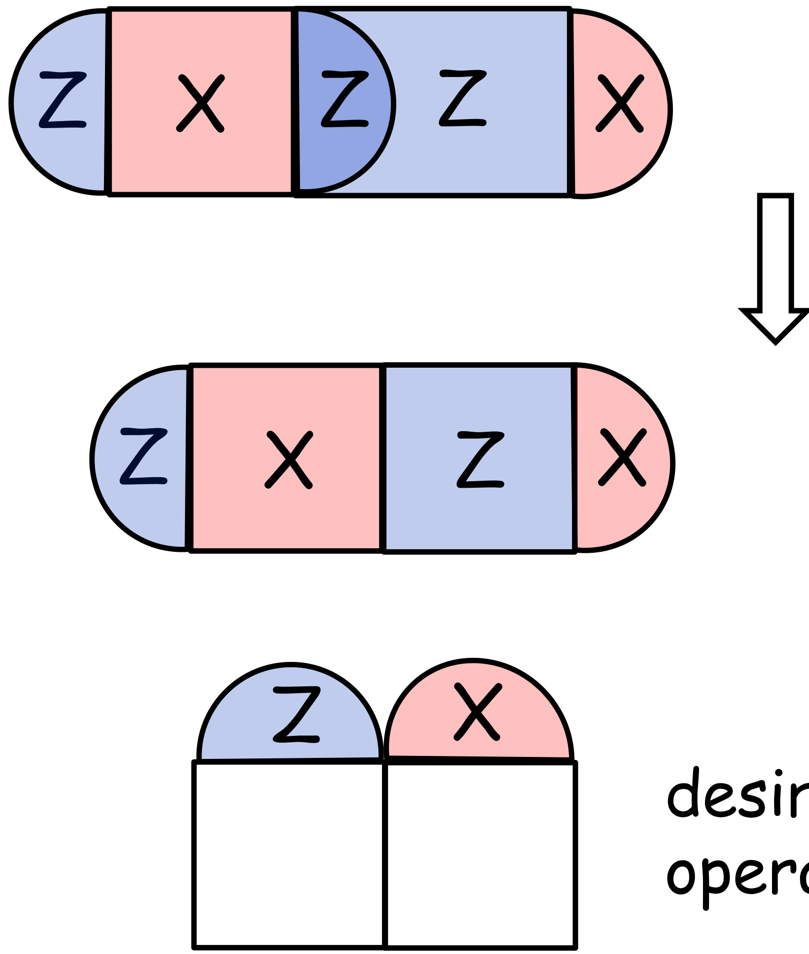
Step 2: change the stabilizer group



Stop measuring
Z-stabilizer

current gauge
operators

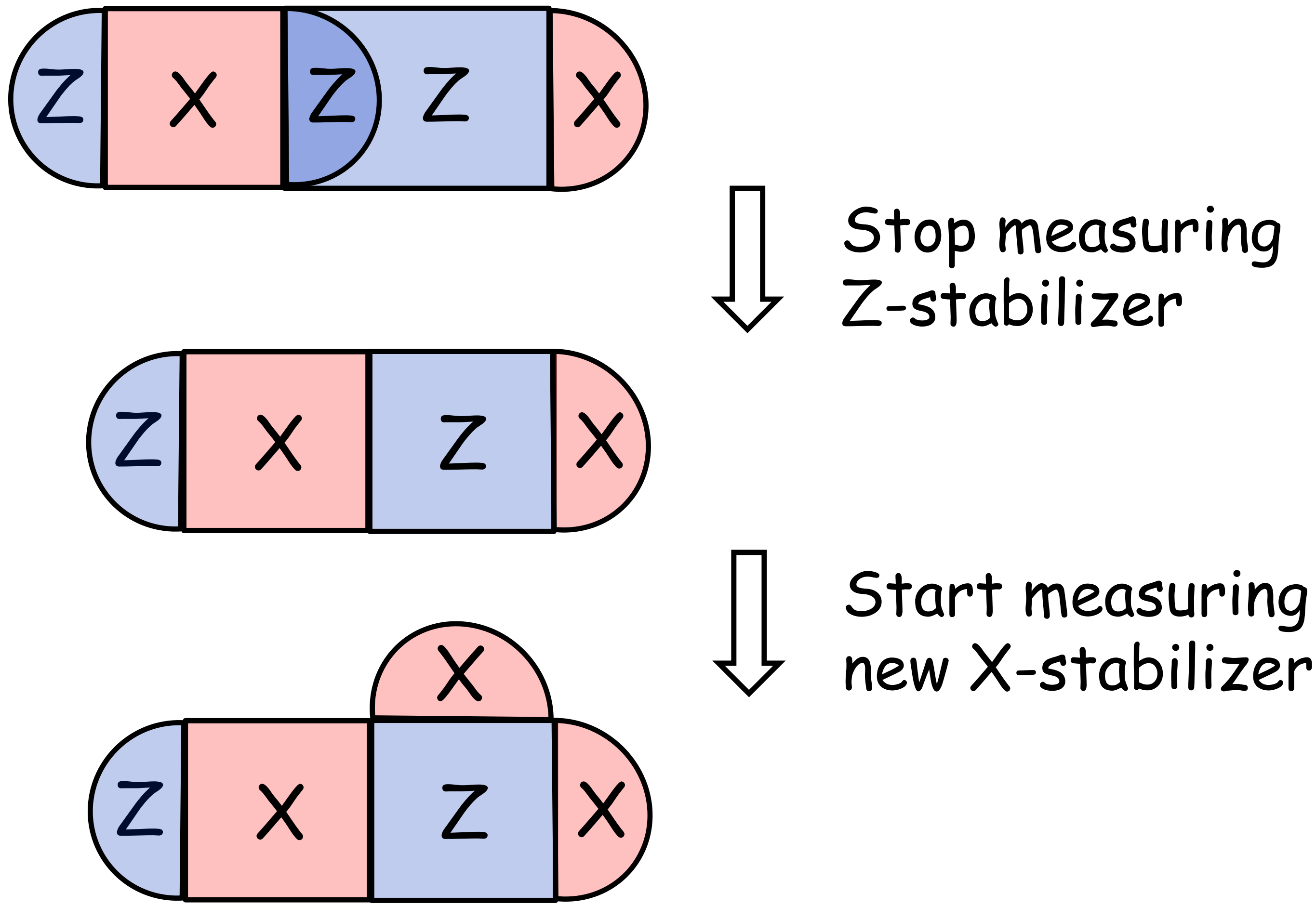
Step 2: change the stabilizer group



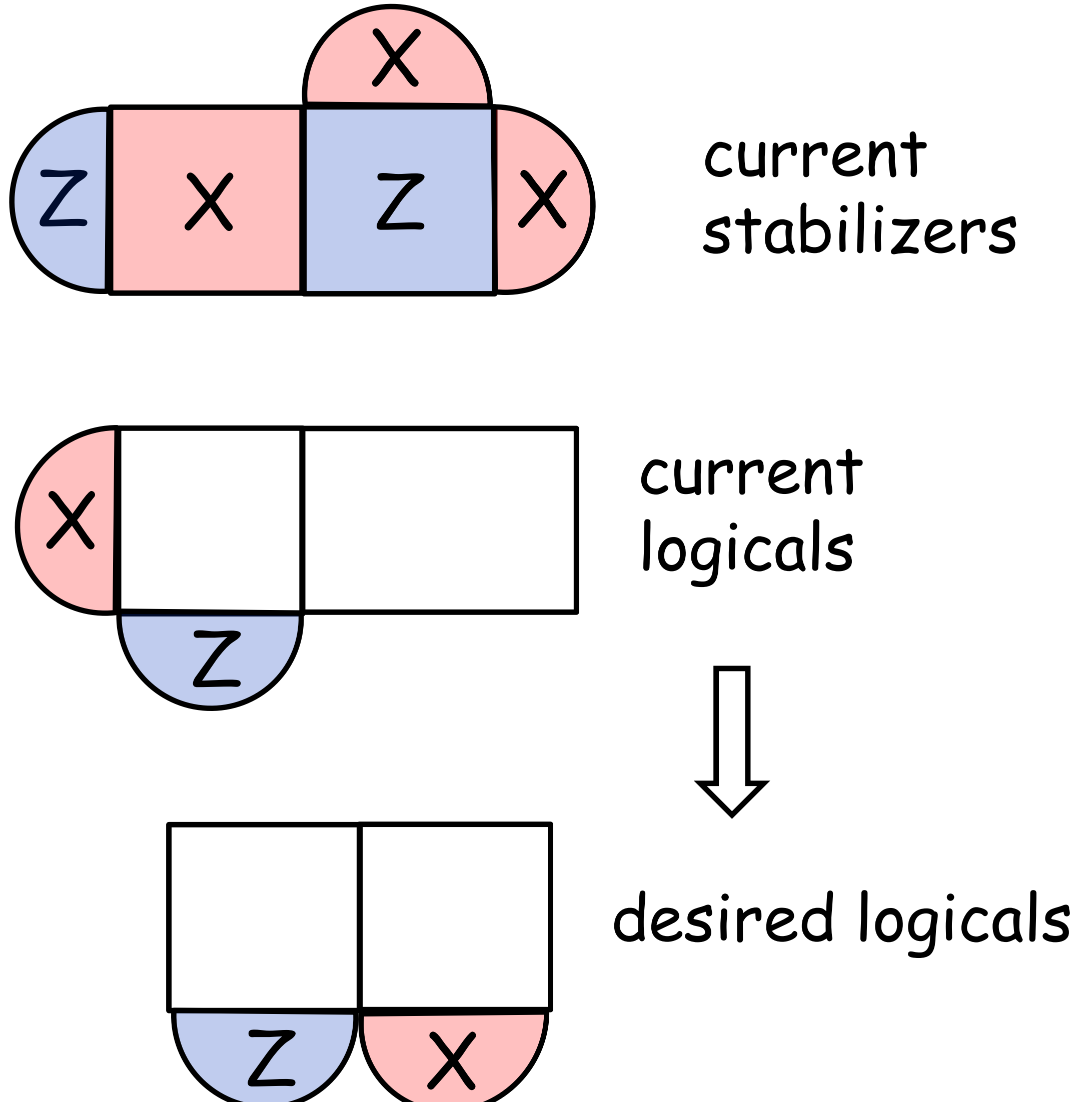
Stop measuring Z-stabilizer

desired gauge operators

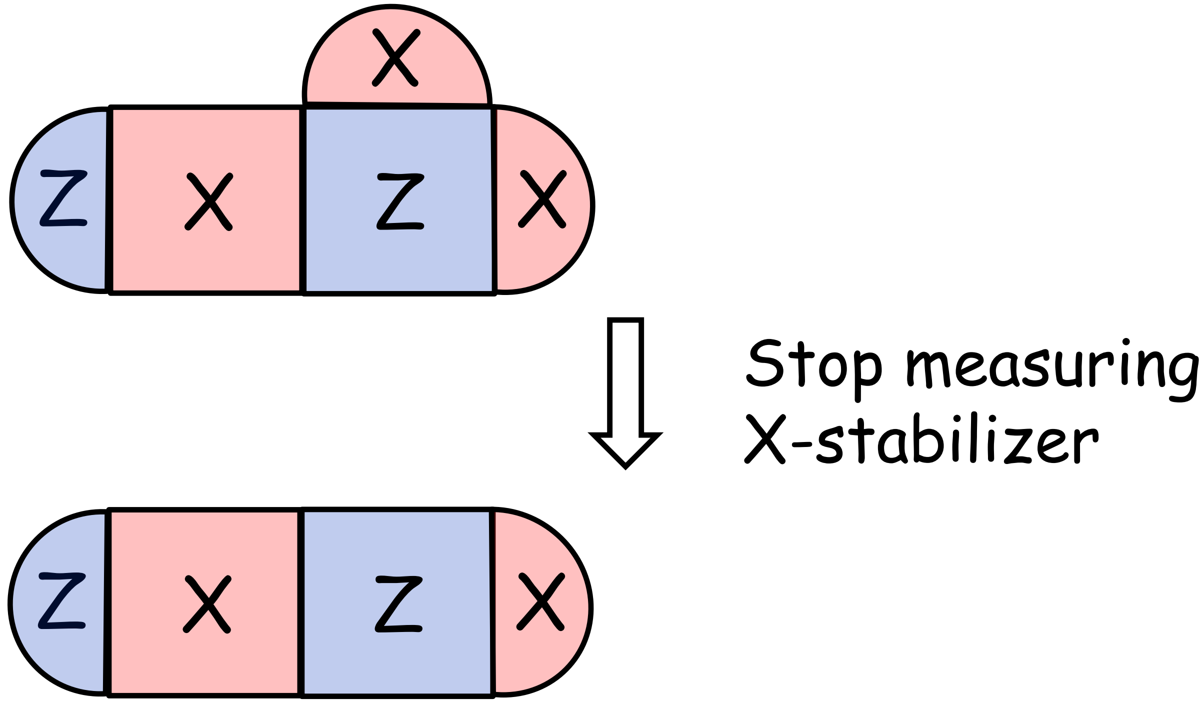
Step 3: change the stabilizer group



Step 4: choose a new basis set of logical operators

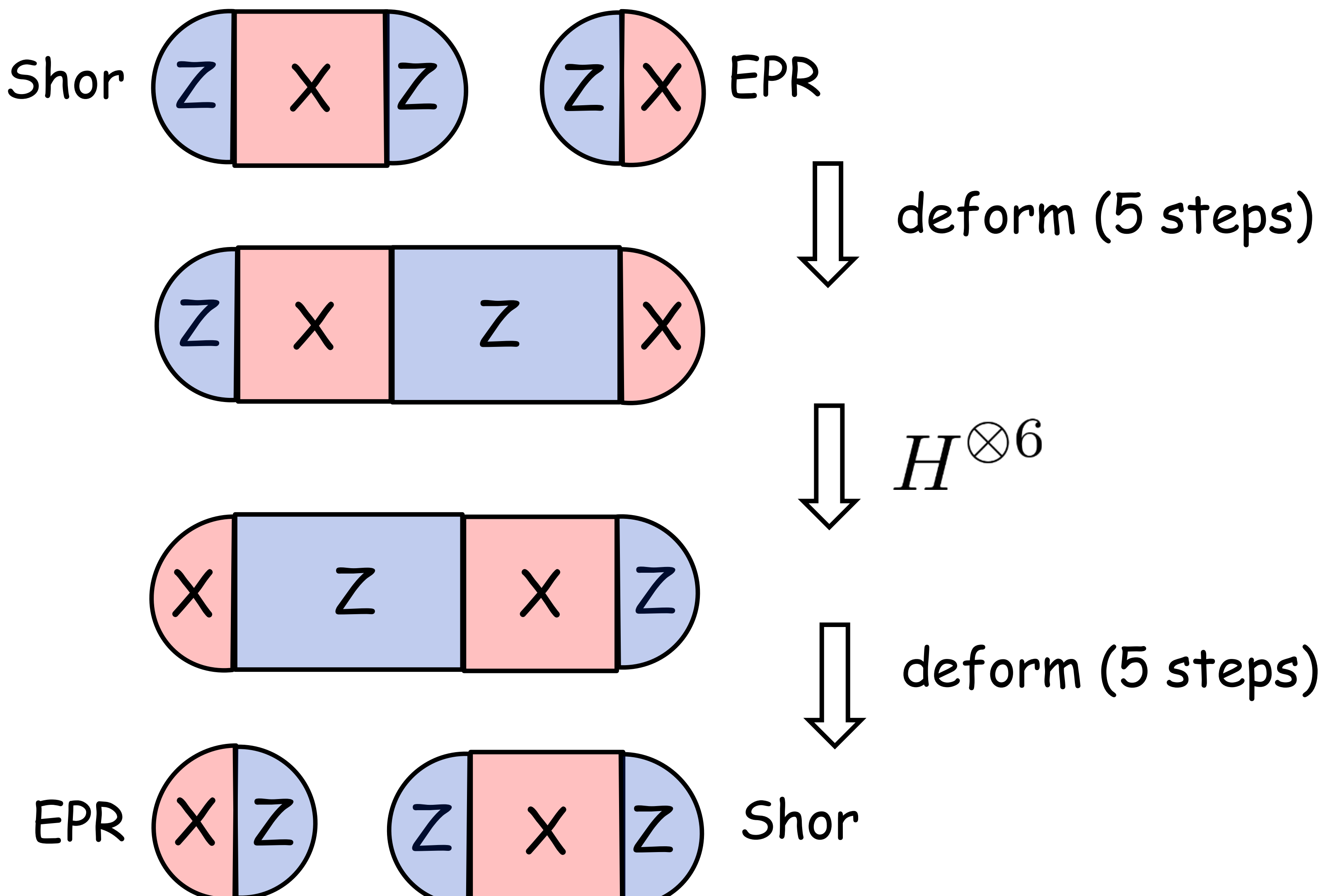


Step 5: change the stabilizer group



Now we have the desired stabilizer/logical/gauge operators.

Logical Hadamard: summary

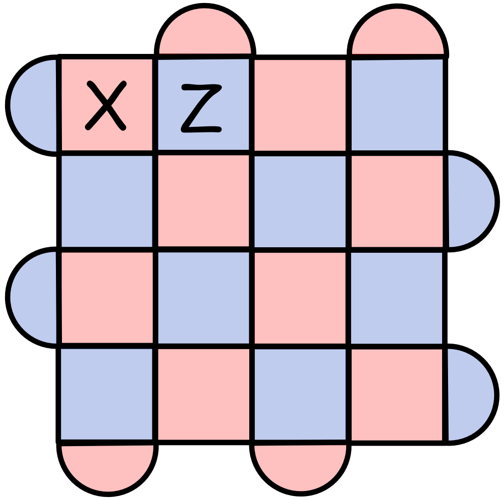


Outline

- Stabilizer codes
- The decoding problem and code distance
- Fault-tolerant code deformation
- Example 1: Shor's 4-qubit code
- Example 2: lattice surgery
- Maximum likelihood decoding

Rotated surface code

X.-G. Wen (2006)



$d \times d$

Qubits = sites.

Stabilizers = faces.

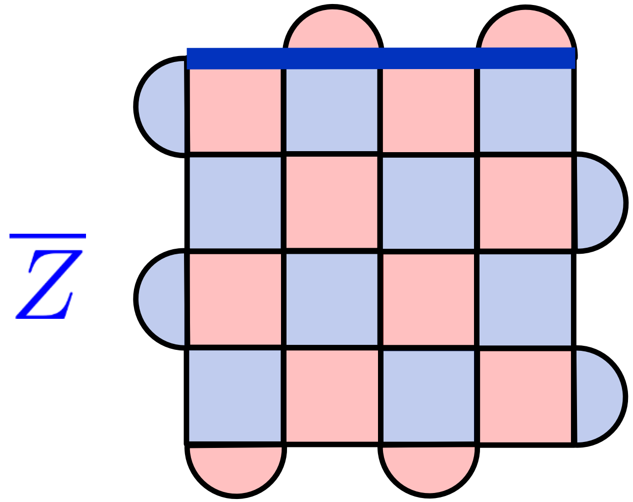
$$[[d^2, 1, d]]$$

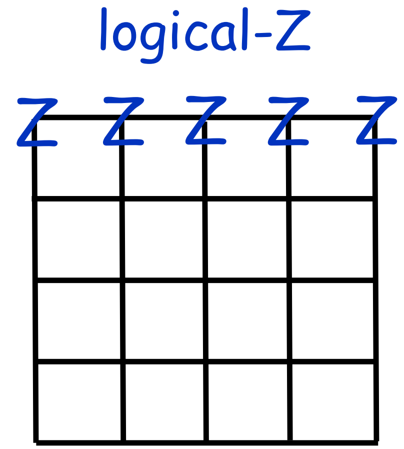
Generalization of the Shor's 4-qubit code.

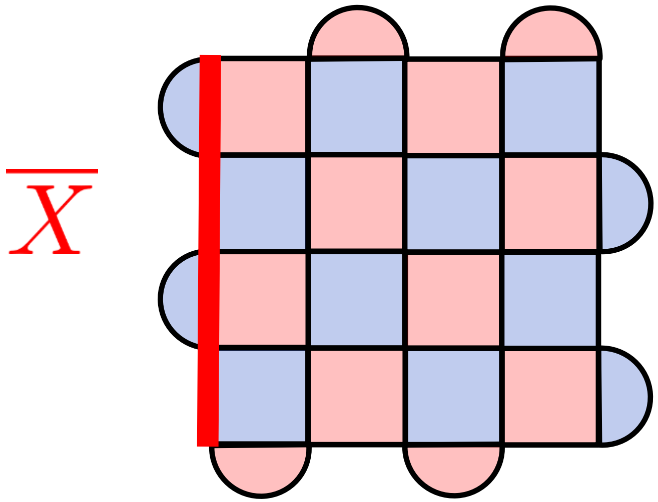
Homological CSS code for any boundary conditions.

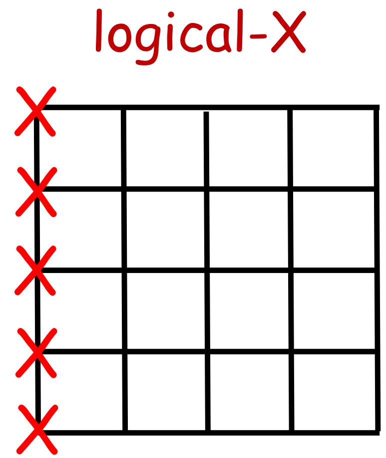
Achieves the same distance as the standard surface code with twice as less qubits.

Pictorial notations for the logical operators

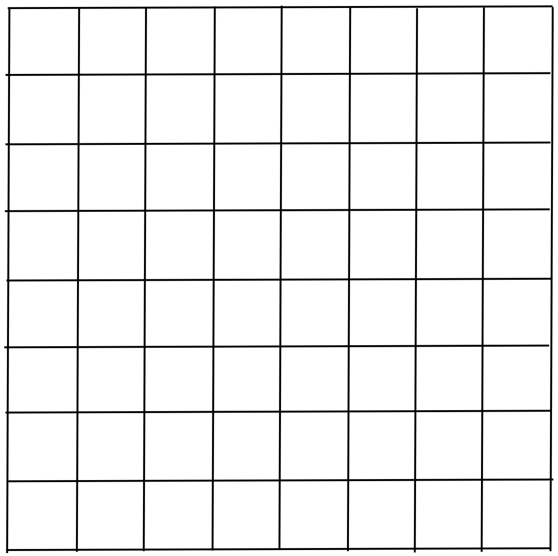




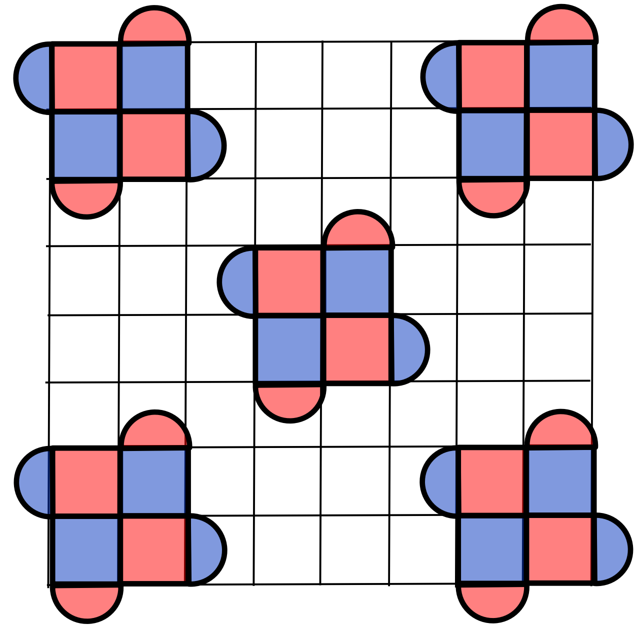




Multiple logical qubits: planar layout



9x9 physical qubits

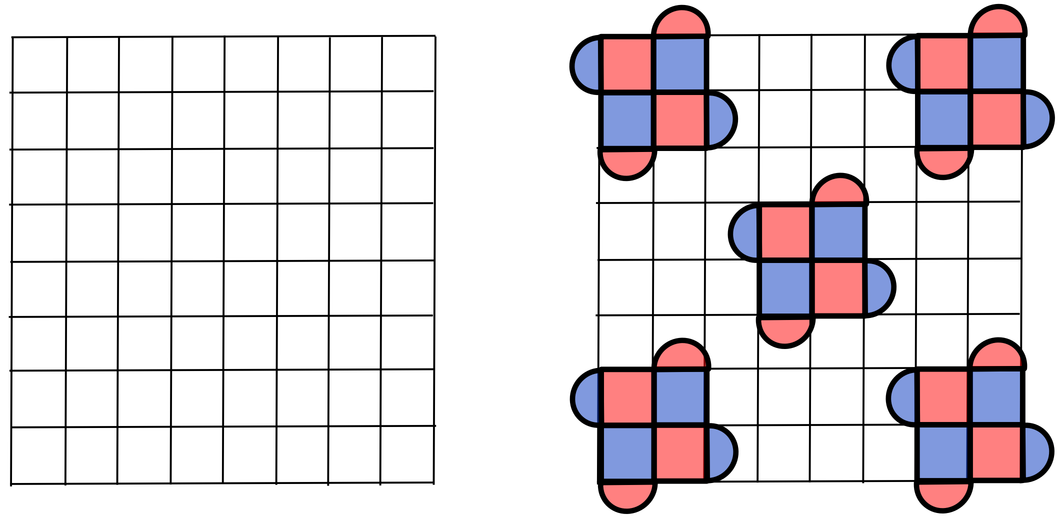


5 logical qubits

The empty space between the logical patches is filled by connector qubits.

Connector qubits mediate interactions between logical qubits and provide space for code deformation

Multiple logical qubits: planar layout



Locality restriction: any stabilizer measured in the protocol must be face-like, or edge-like, or a single site.

Promising architecture for platforms based on superconducting qubits
(no qubit movement, no long-range interactions)

Horsman et al (2011); Gambetta, Chow, Steffen (2015)

Target logical operations:

Z-Prep	X-Prep
Z-Meas	X-Meas
CNOT	H

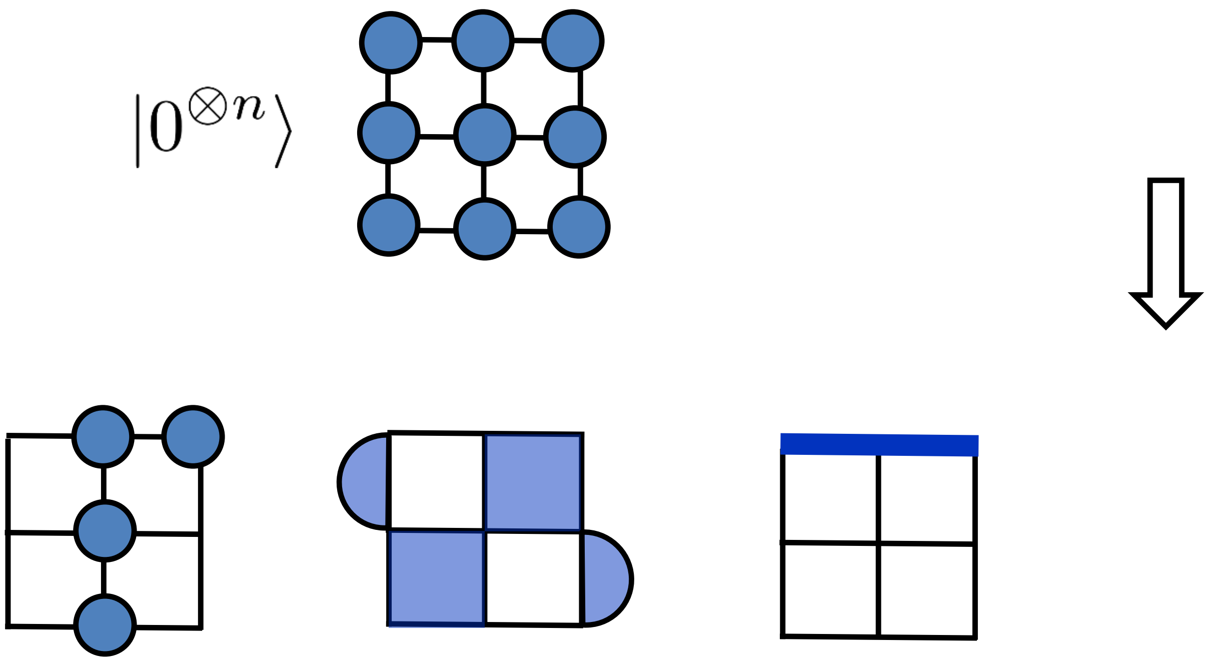
1. Prepare a new logical qubit in $|0\rangle$ or $|+\rangle$
2. Measure a logical qubit in Z or X basis
3. Logical Hadamard
4. Logical CNOT

Goal: implement 1-4 by code deformation satisfying the locality and the fault-tolerance constraints.

Z-Prep

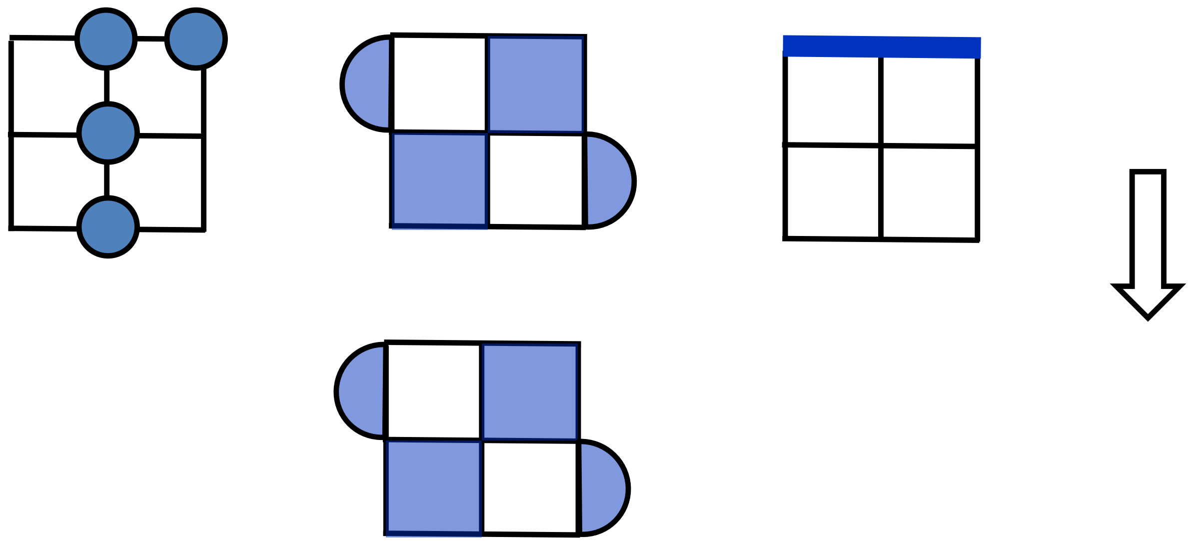
1. Initialize each physical qubit in $|0\rangle$
2. Measure syndrome
3. Use syndromes of Z-type stabilizers to correct X-type errors
4. Use syndromes of X-type stabilizers for gauge fixing.

Step 1: choose a new basis set of stabilizers.

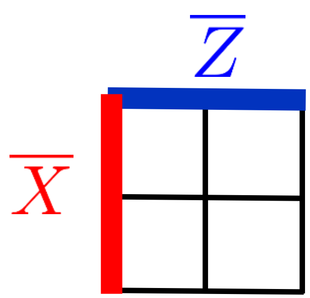


The code has no logical/gauge qubits

Step 2: stop measuring stabilizers of type 1,3



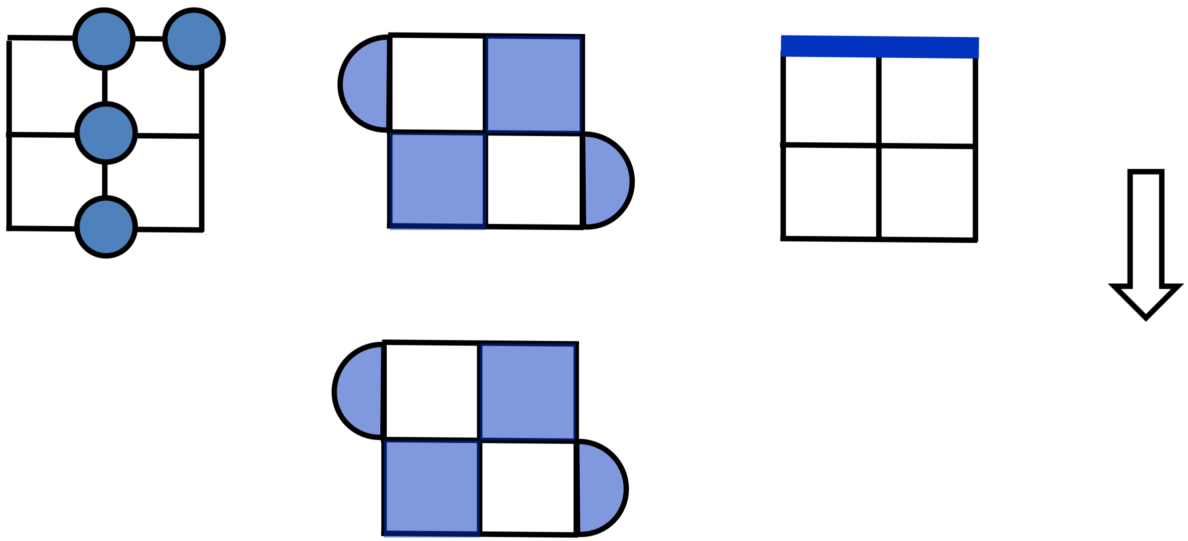
Use stabilizers of type 3 to create a logical qubit:



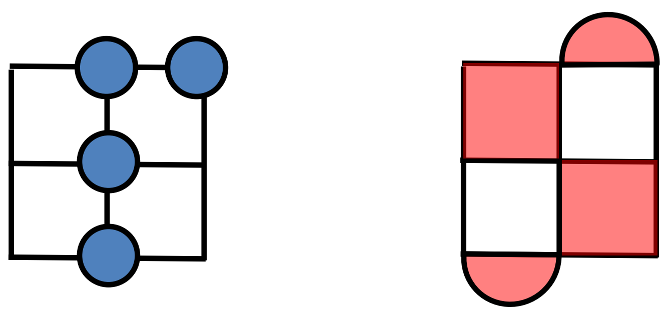
new logical operators

$$\overline{Z} = 1$$

Step 2: stop measuring stabilizers of type 1,3

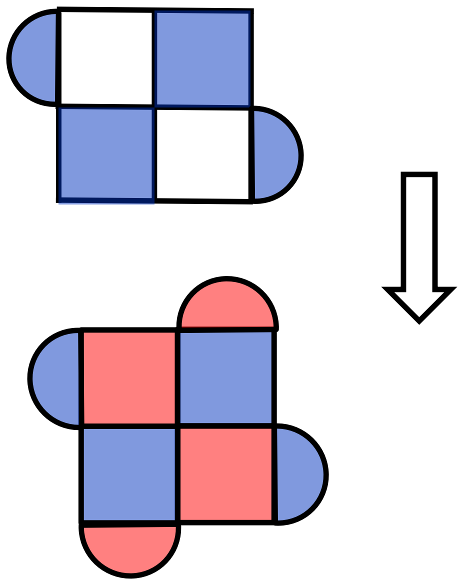


Use stabilizers of type 1 to create 4 gauge qubits:

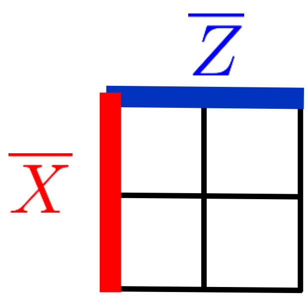


new gauge operators

Step 3: start measuring new stabilizers.
Gauge operators of X-type become stabilizers.



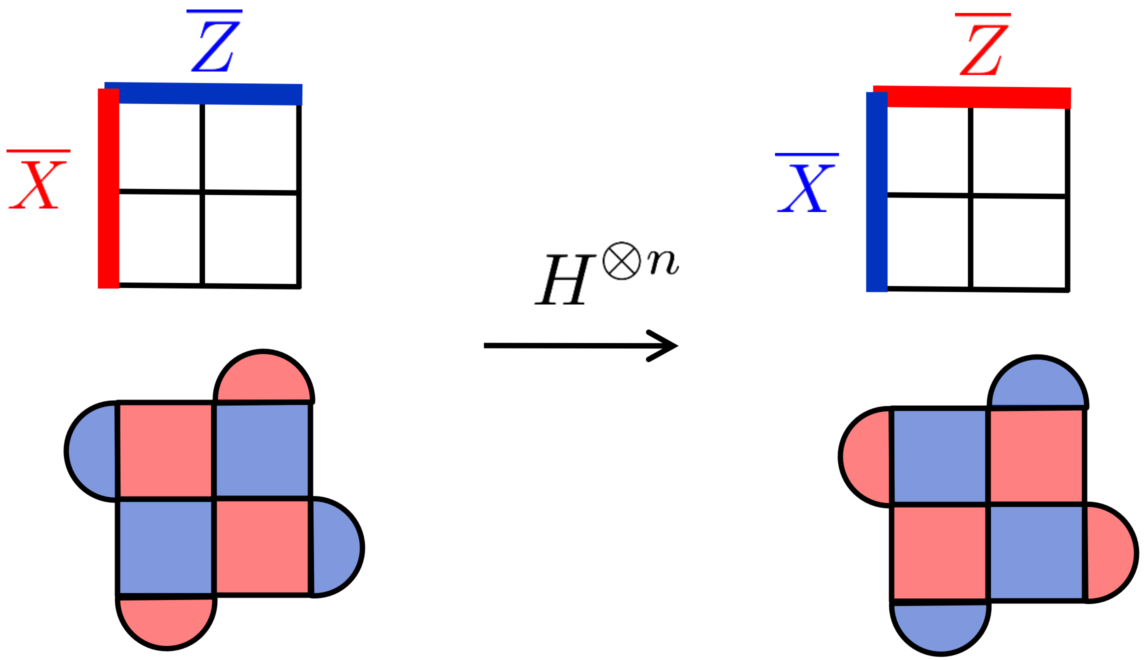
The final code has no gauge qubits and
one logical qubit with the logical operators



$$\overline{Z} = 1$$

Logical Hadamard

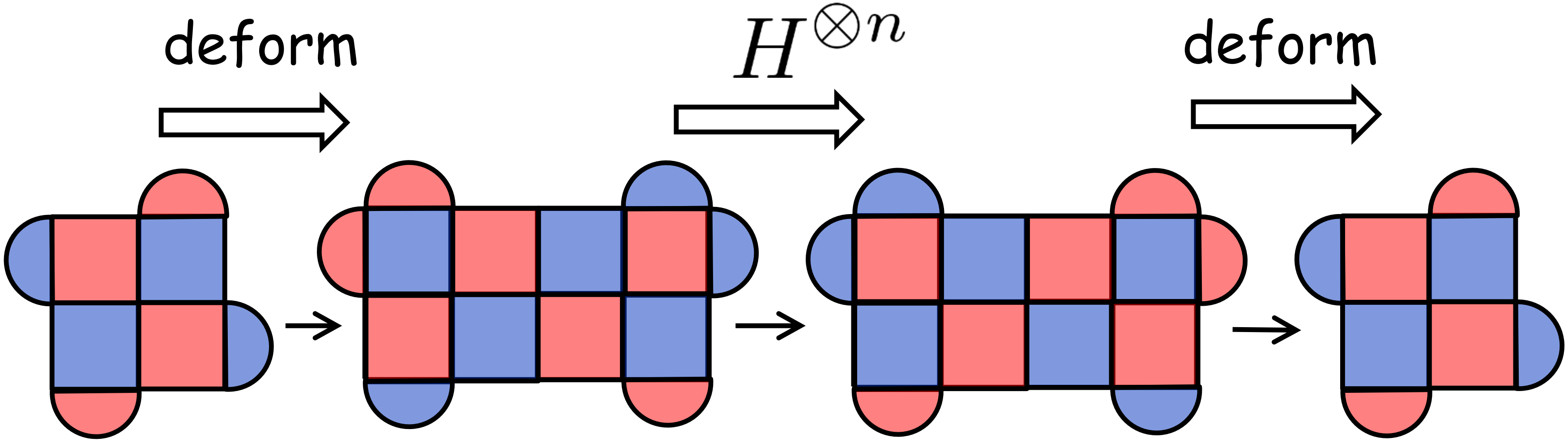
Naive implementation:



Lattice rotation is needed to get the original code.

Lattice rotation by code deformation is too expensive...

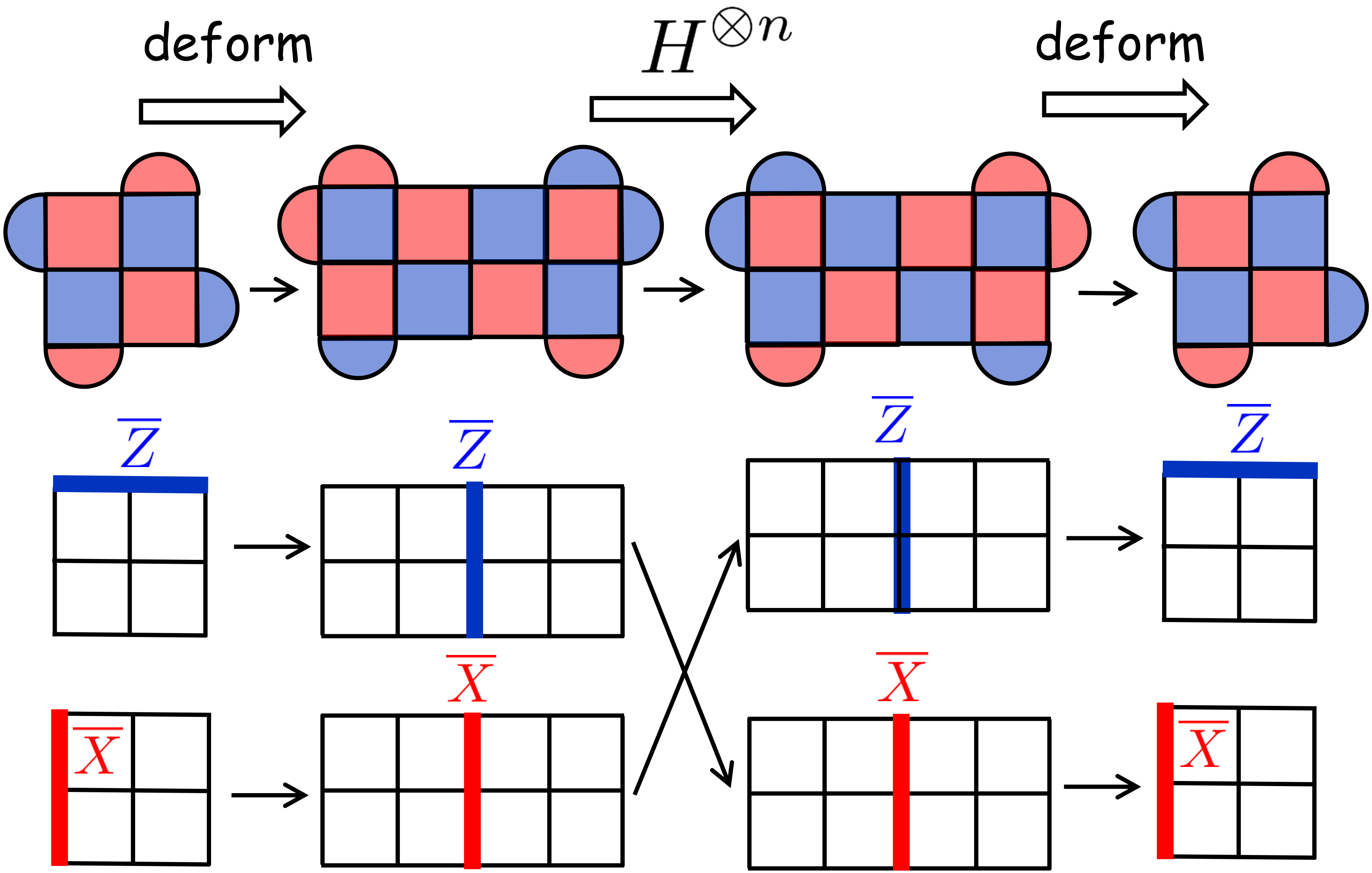
Hadamard without lattice rotation: sketch



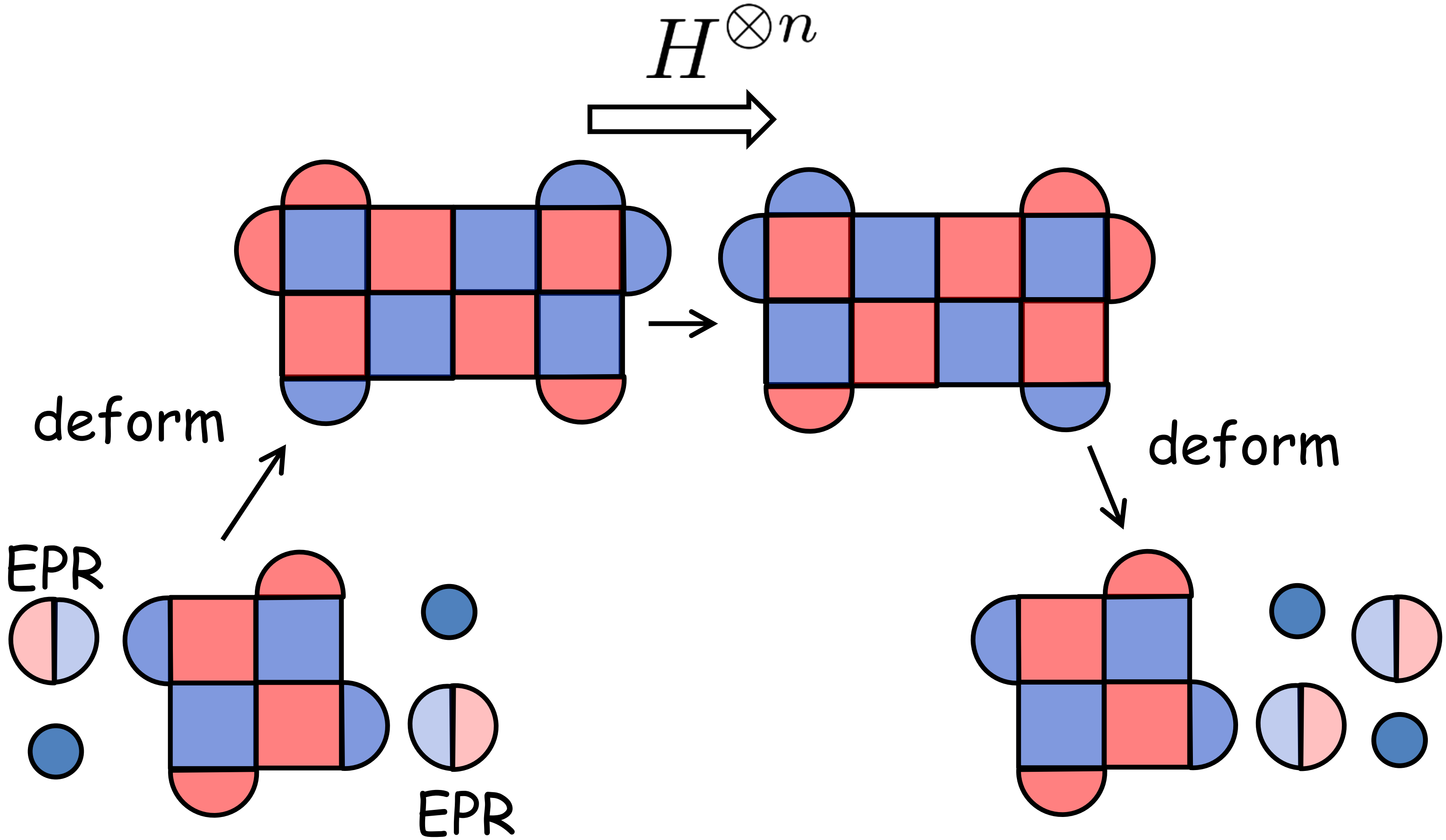
extend the lattice;
deform boundary
stabilizers

deform boundary
stabilizers;
contract the lattice

Hadamard without lattice rotation: sketch

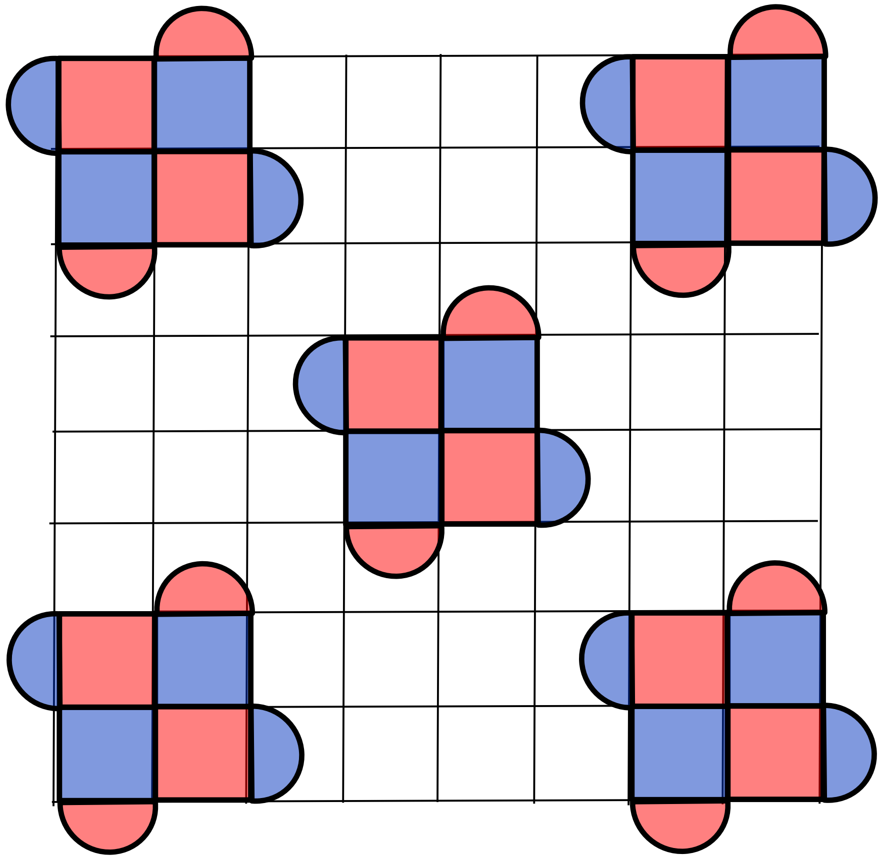


Hadamard without lattice rotation: sketch

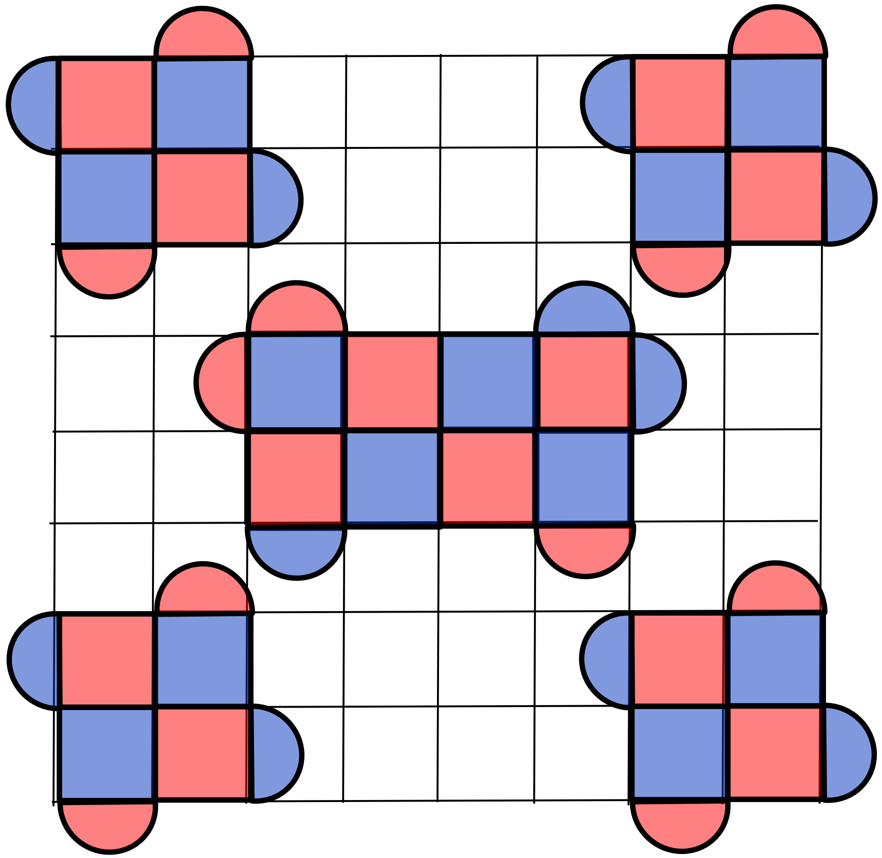


Lattice extension requires ancillary qubits

Does it fit into the chosen planar layout ?

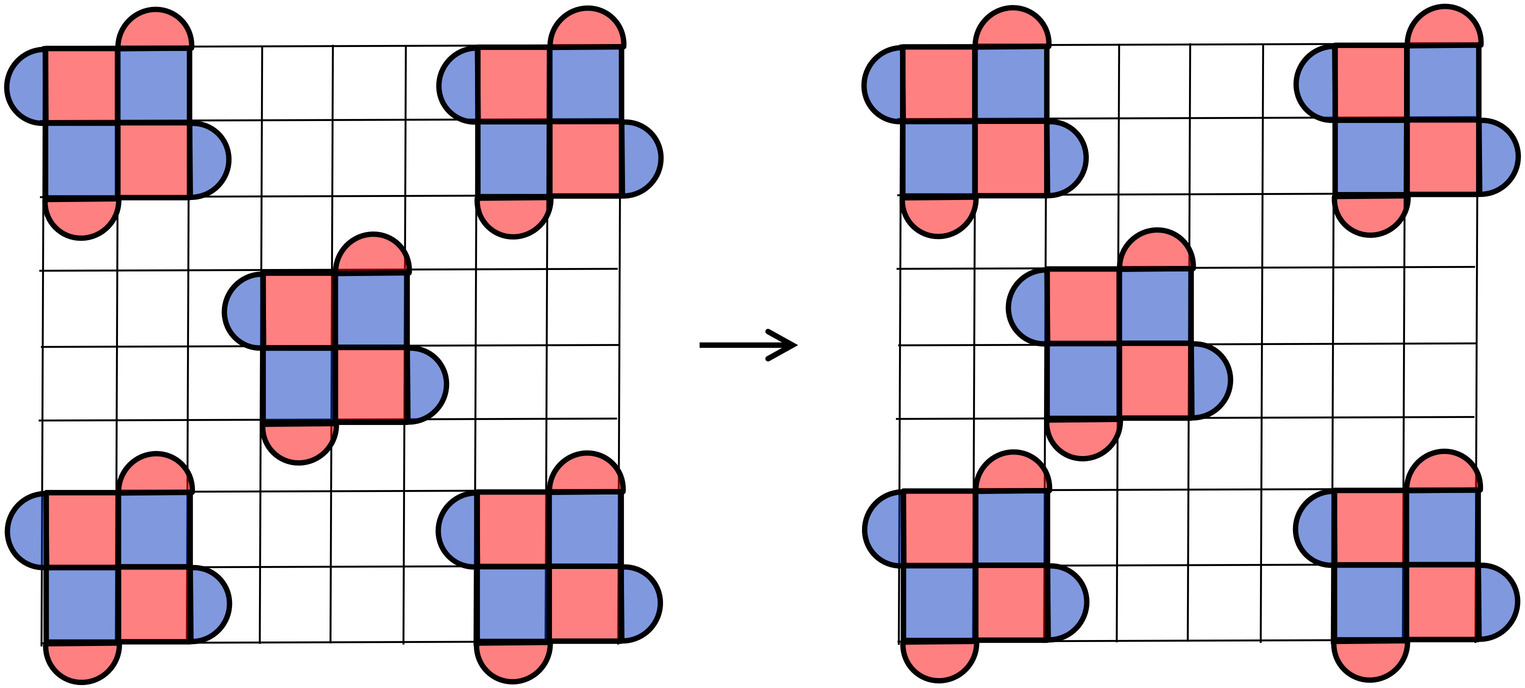


Does it fit into the chosen planar layout ?



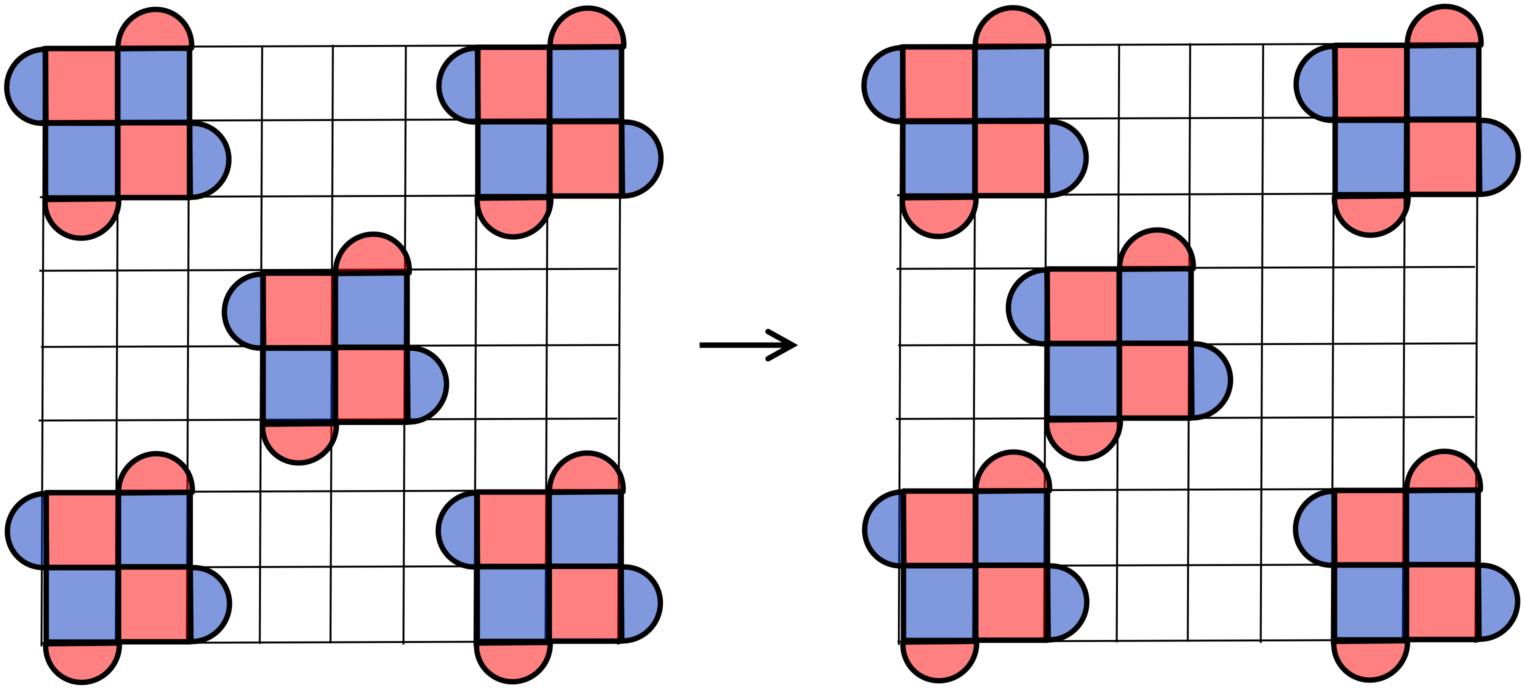
Logical Hadamards on adjacent logical qubits do not interfere with each other

Does it fit into the chosen planar layout ?



Problem: the logical patch is shifted by one lattice period after each Hadamard.

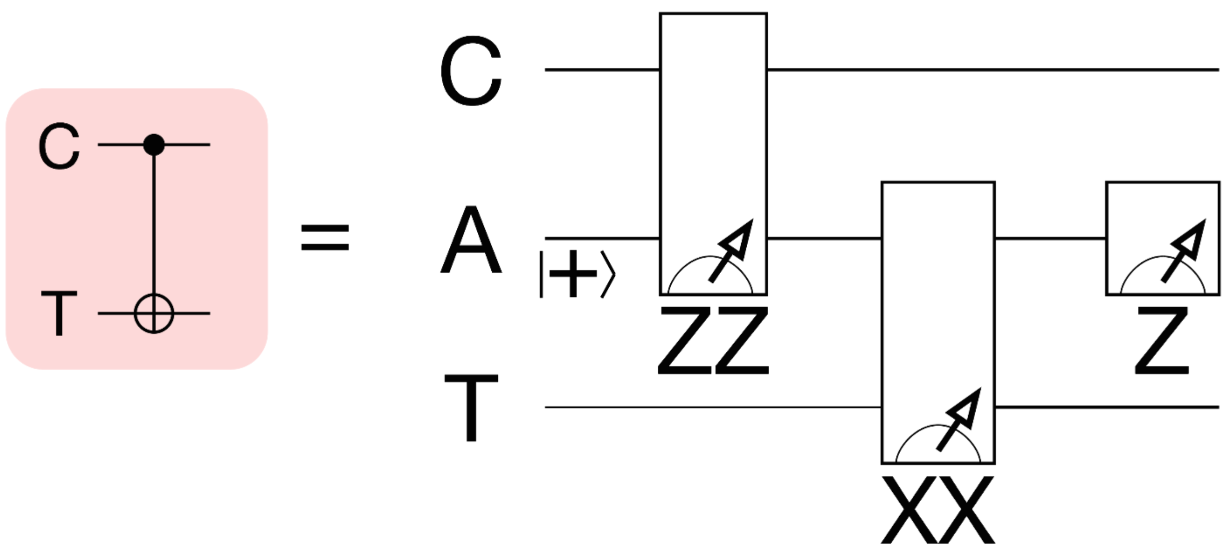
Does it fit into the chosen planar layout ?



Solution: alternate between left and right shifts

Better solution: use syndrome readout ancillas (not shown) to shift the logical patch back

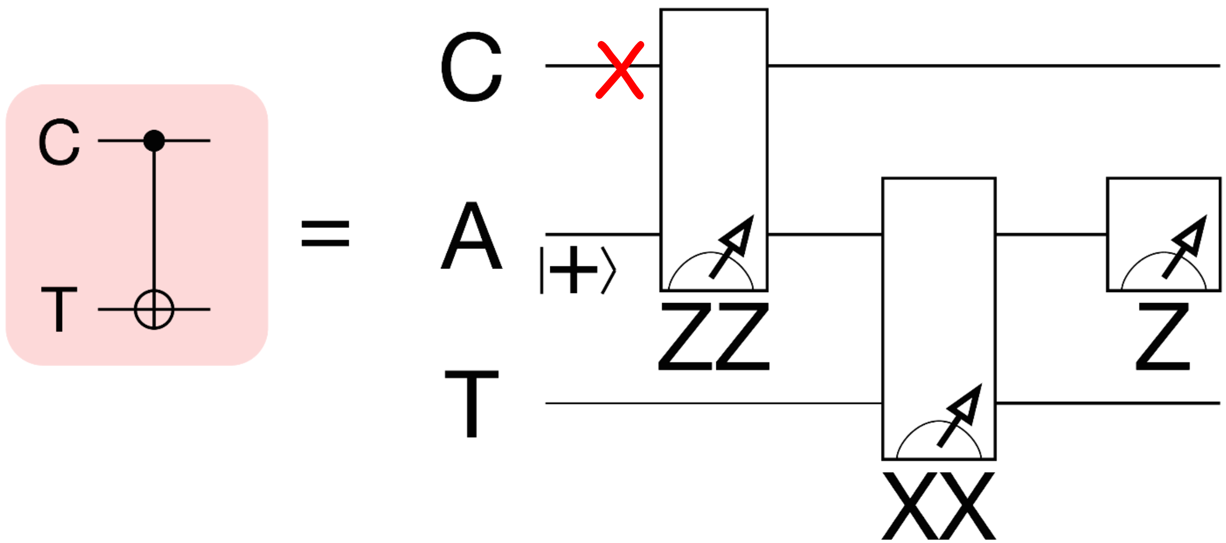
Logical CNOT



CNOT truth table:

in	out
XI	XX
IX	IX
ZI	ZI
IZ	ZZ

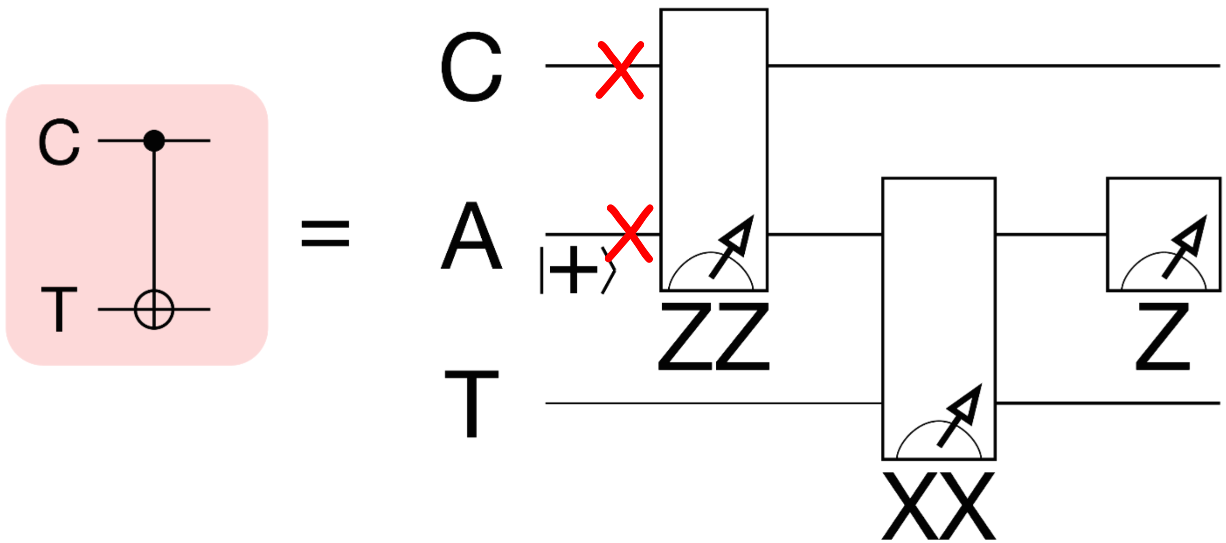
Logical CNOT



CNOT truth table:

in	out
XI	XX
IX	IX
ZI	ZI
IZ	ZZ

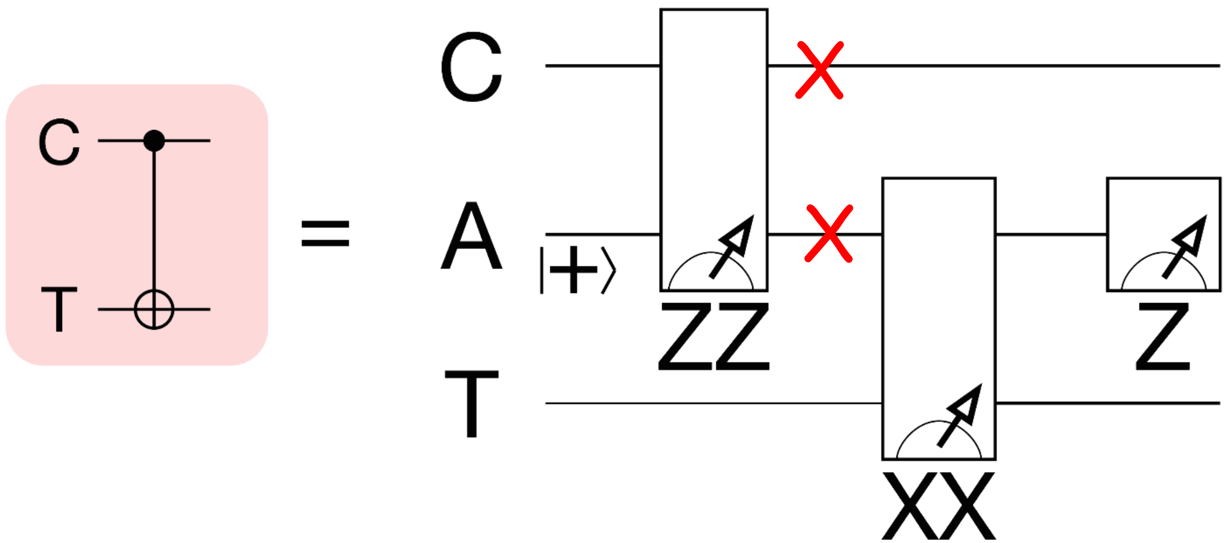
Logical CNOT



CNOT truth table:

in	out
XI	XX
IX	IX
ZI	ZI
IZ	ZZ

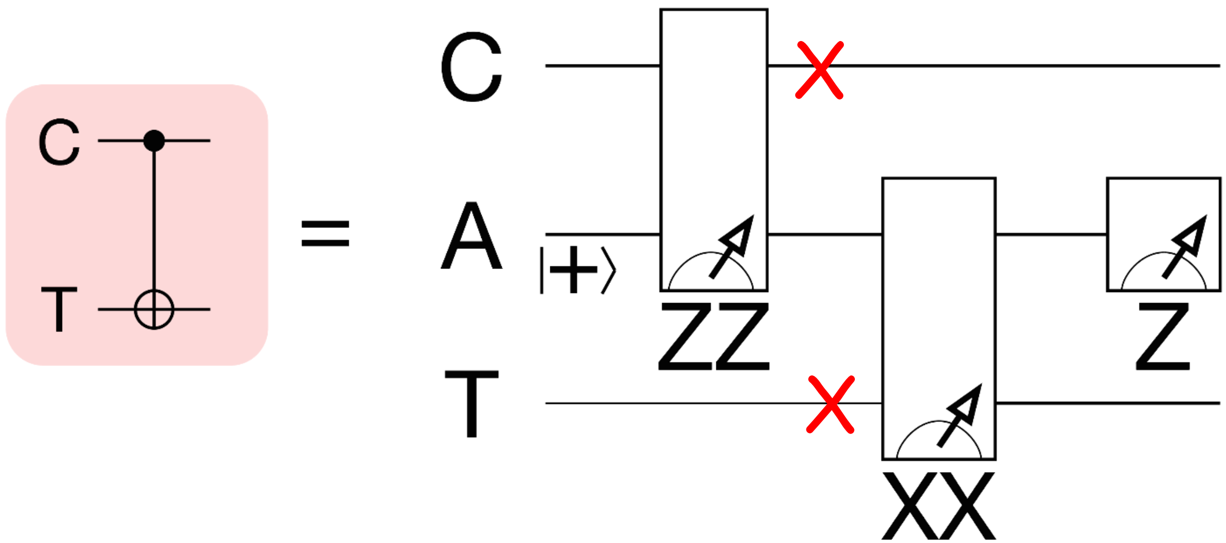
Logical CNOT



CNOT truth table:

in	out
XI	XX
IX	IX
ZI	ZI
IZ	ZZ

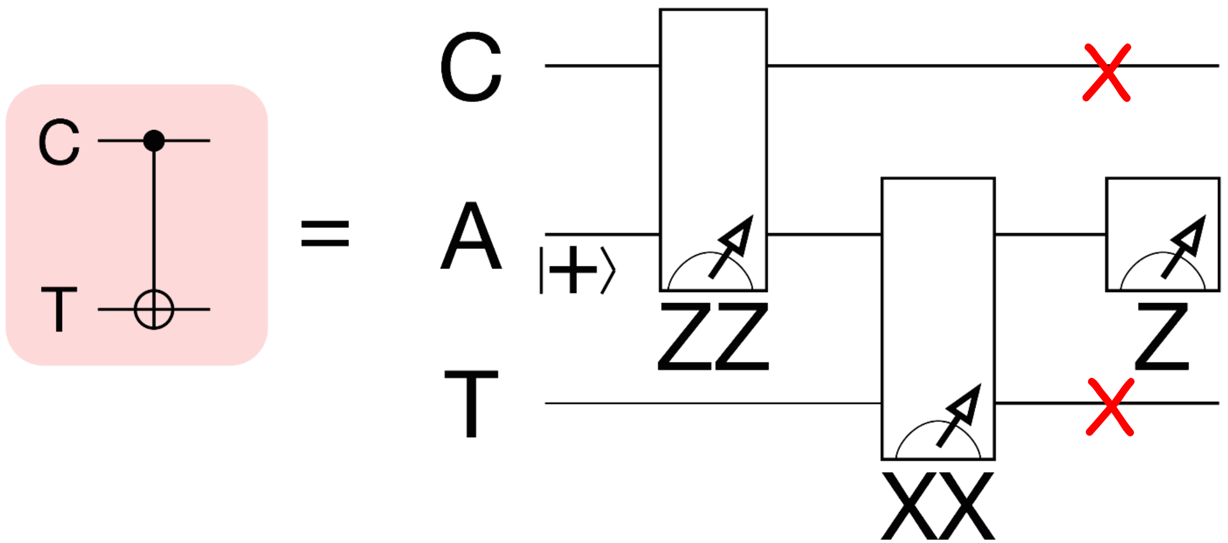
Logical CNOT



CNOT truth table:

	in	out
	XI	XX
	IX	IX
	ZI	ZI
	IZ	ZZ

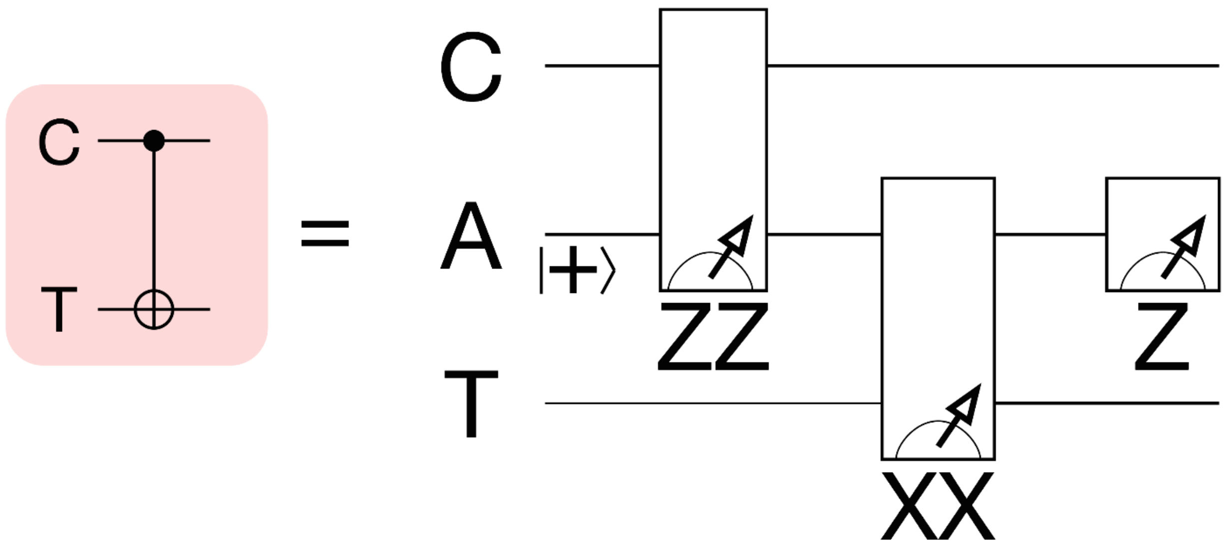
Logical CNOT



CNOT truth table:

in	out
XI	XX
IX	IX
ZI	ZI
IZ	ZZ

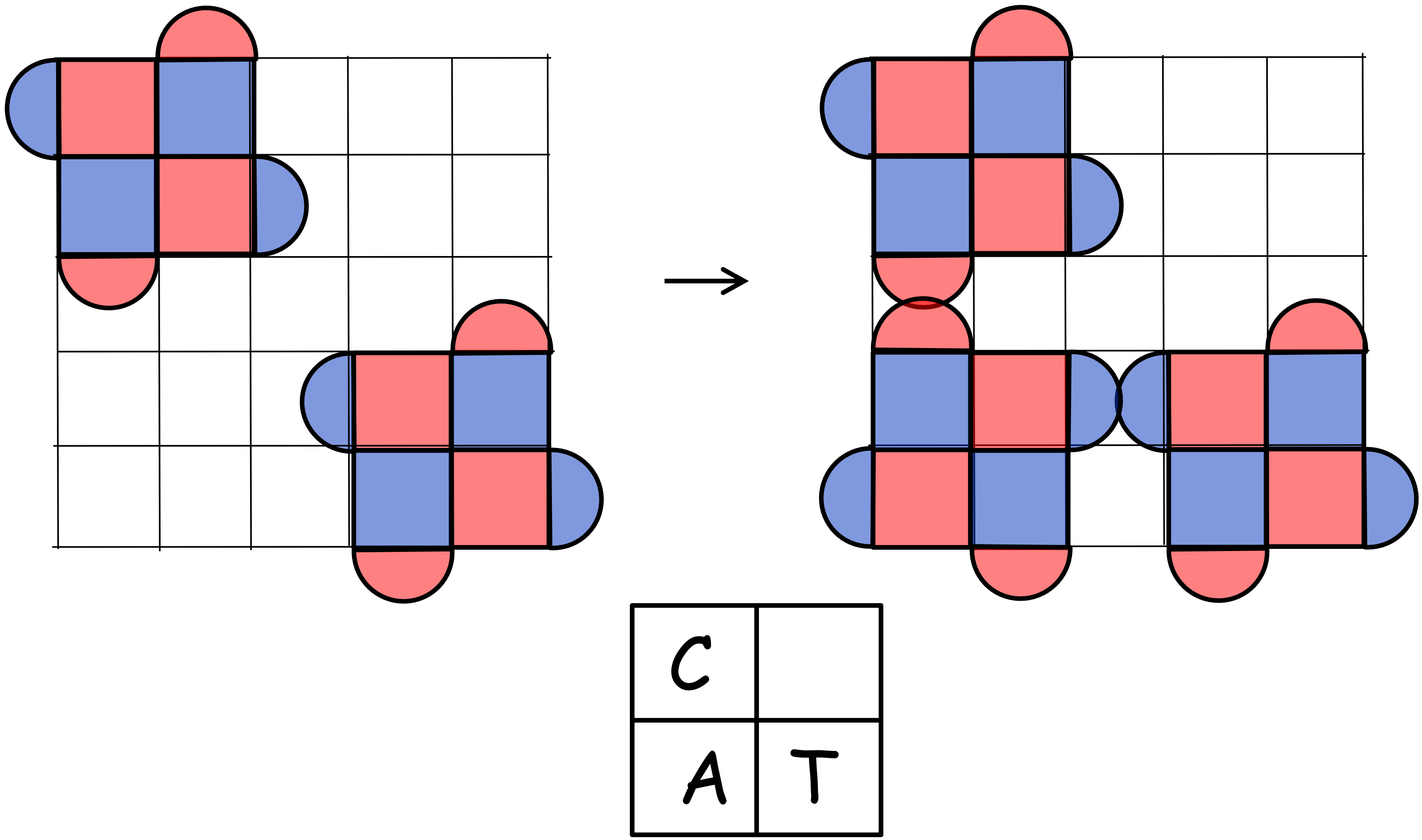
Logical CNOT



It suffices to implement non-destructive logical ZZ and XX measurements

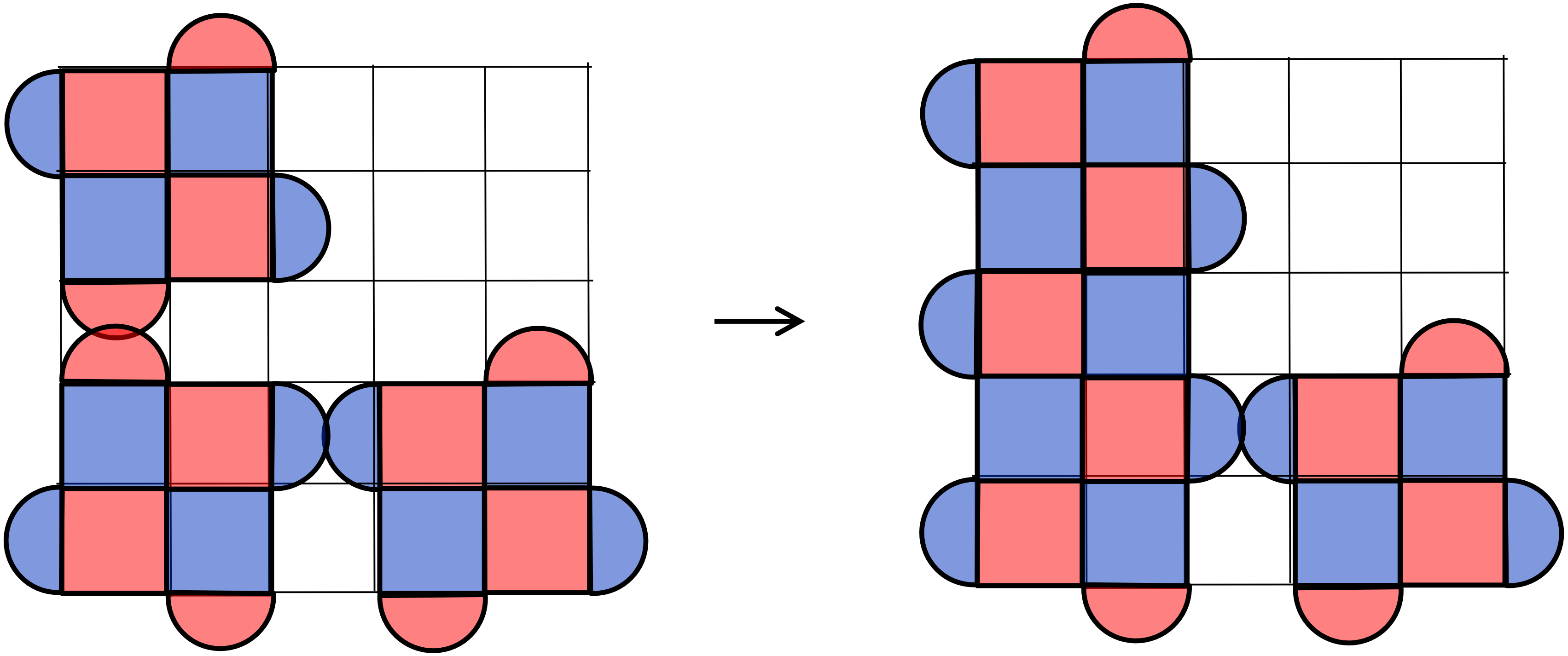
Logical ZZ measurement

Step 1: create one ancillary logical path in logical $|+\rangle$



Logical ZZ measurement

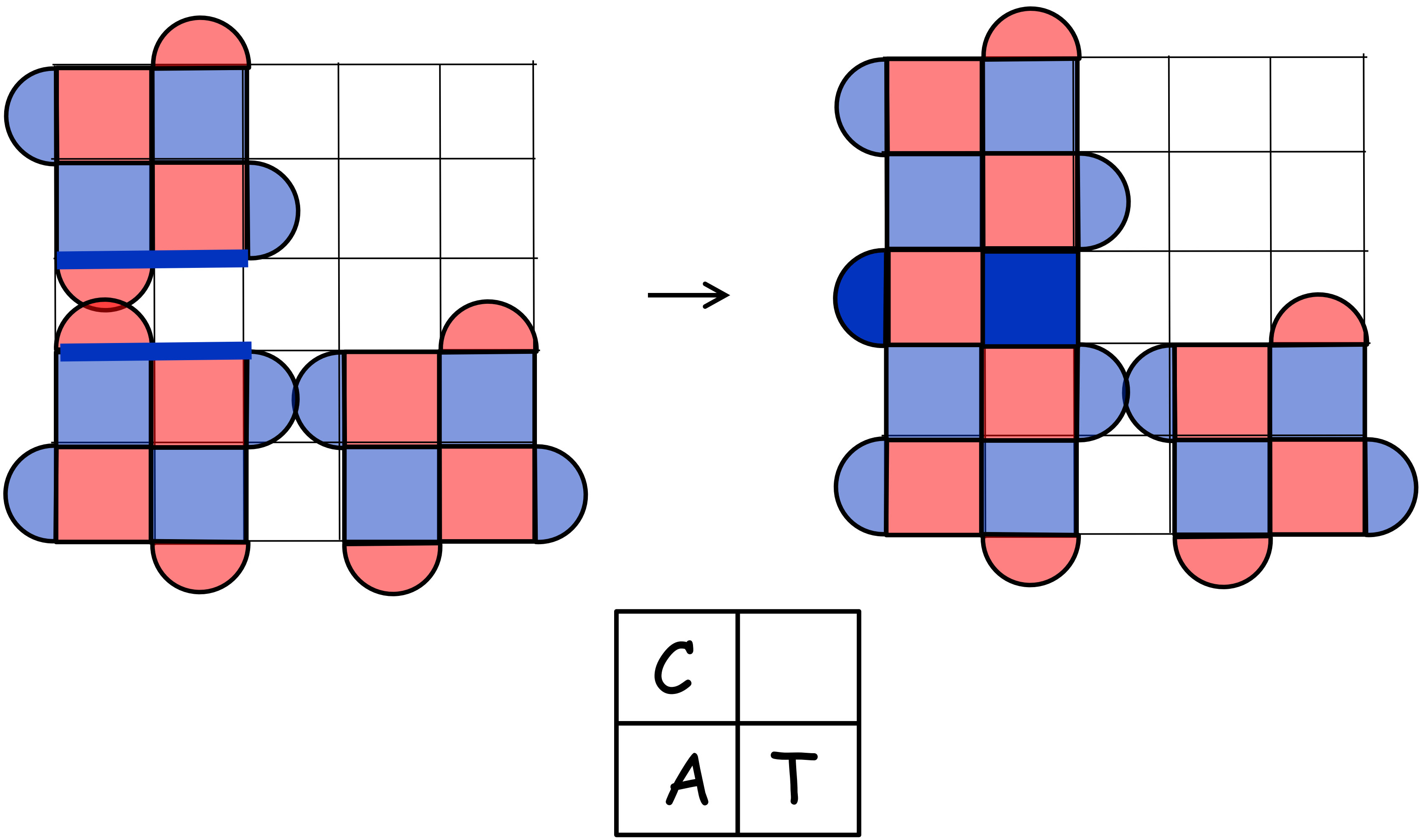
Step 2: merge C and A patches



C	
A	T

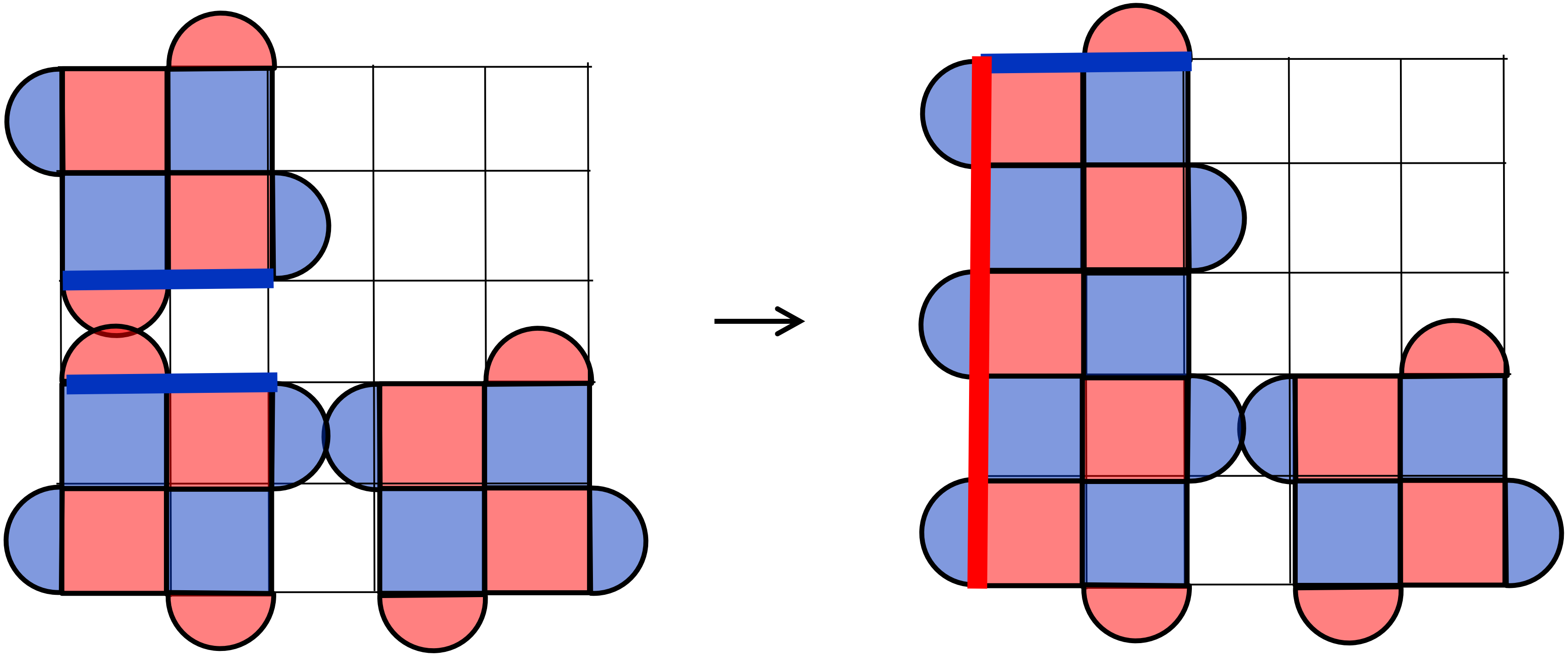
Logical ZZ measurement

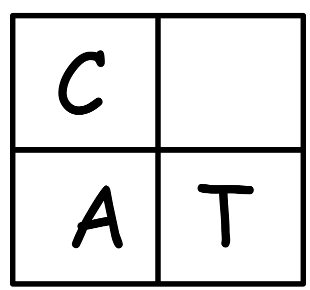
Step 2: Logical ZZ becomes a stabilizer



Logical ZZ measurement

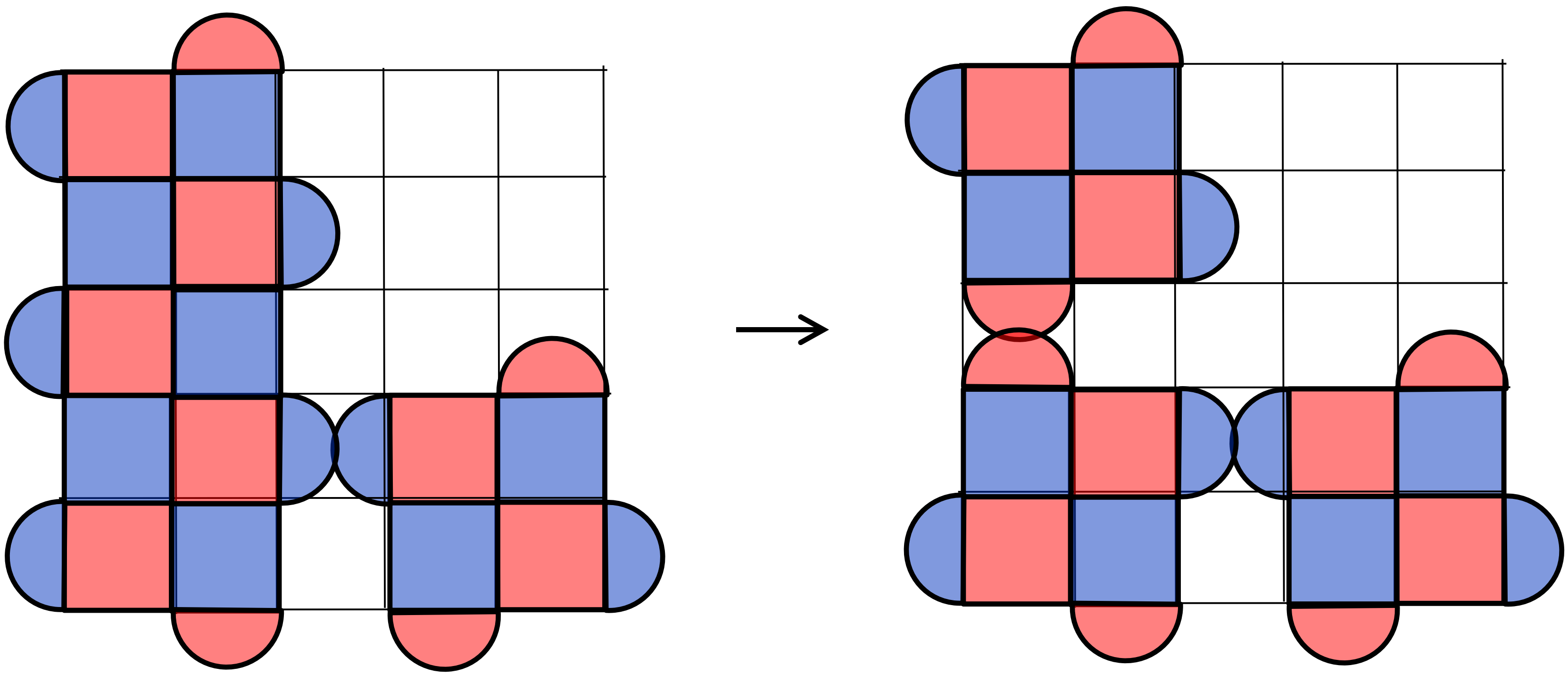
Step 2: the merged patch has one logical qubit





Logical ZZ measurement

Step 3: disconnect C and A patches



C	
A	T

Lattice surgery: summary

Z-Prep	X-Prep
Z-Meas	X-Meas
CNOT	H

Requires only measurements of local stabilizers supported on faces, edges, and sites of the 2D grid.

Fault-tolerant ancilla injection:
Lodyga et al, arxiv:1404.2495

Open problems:
Phase-shift gate S by code deformation
Maximum likelihood decoding
Resource optimization

Outline

- Stabilizer codes
- The decoding problem and code distance
- Fault-tolerant code deformation
- Example 1: Shor's 4-qubit code
- Example 2: lattice surgery
- Maximum likelihood decoding

Max-Likelihood Decoder: pick the most likely equivalence class of errors consistent with the observed syndrome

$$L^* = \arg\max_L \sum_S \Pr(D \cdot L \cdot S)$$

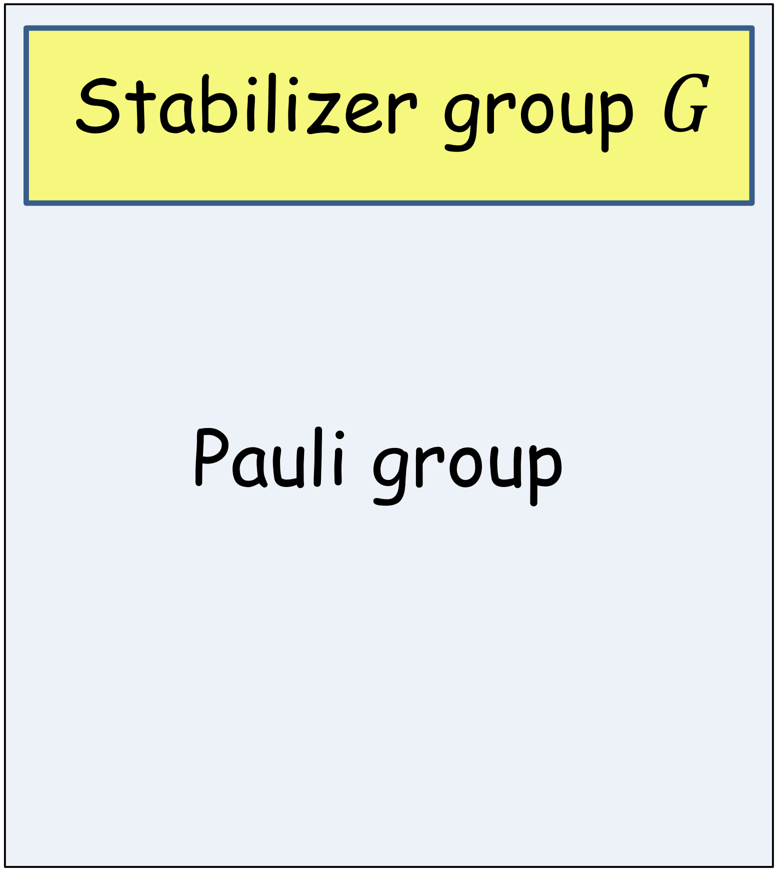
Recovery: $R(D) = D \cdot L^*$

Optimal decoder for a given error model.

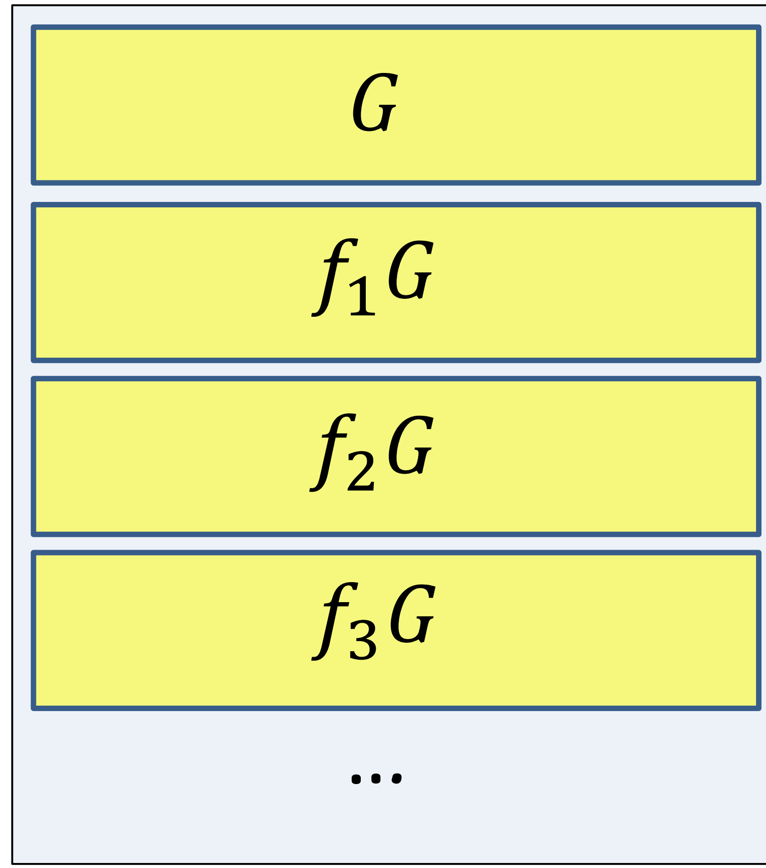
Computing the ML recovery is #P-hard problem

Iyer and Poulin (2013)

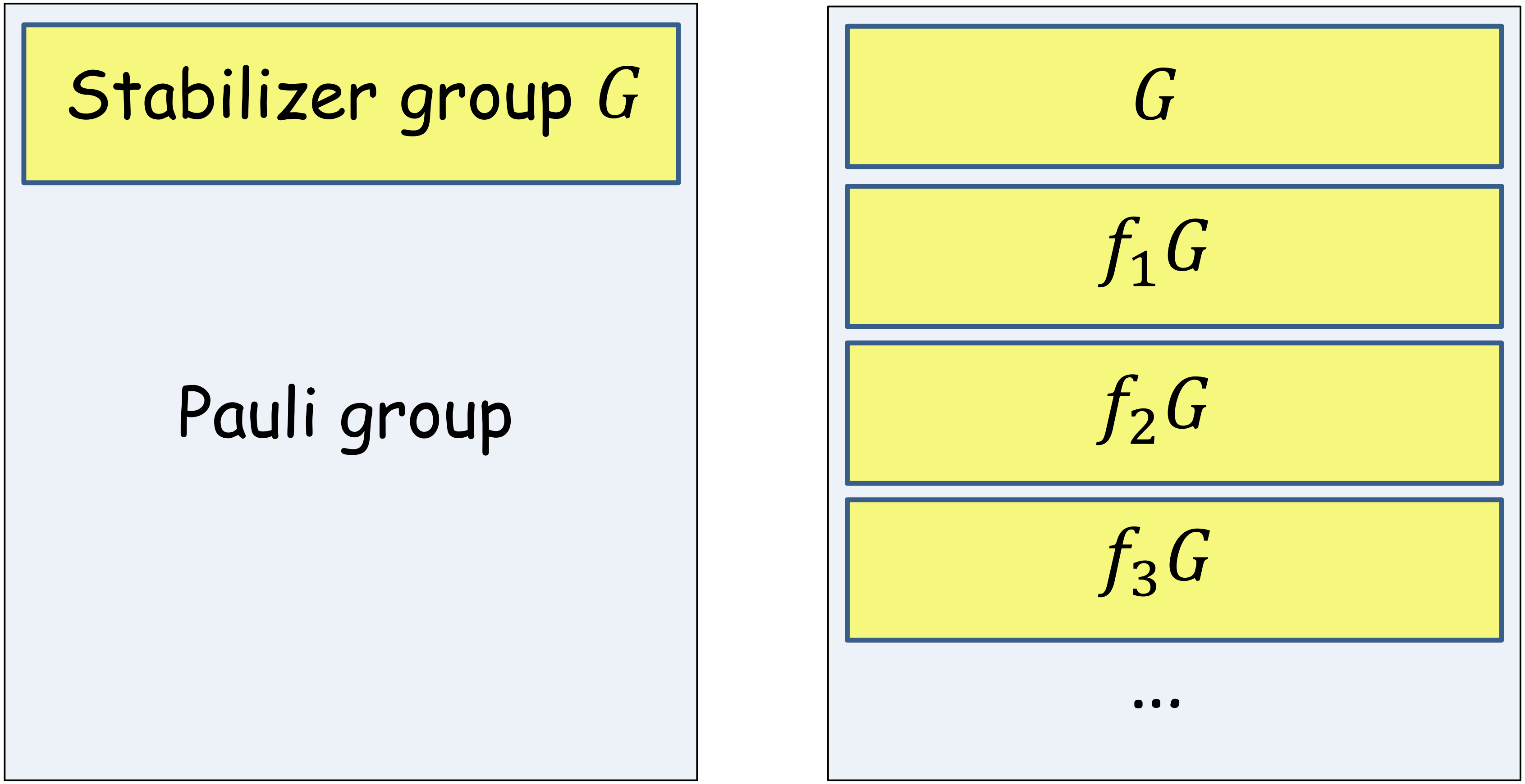
Some terminology:



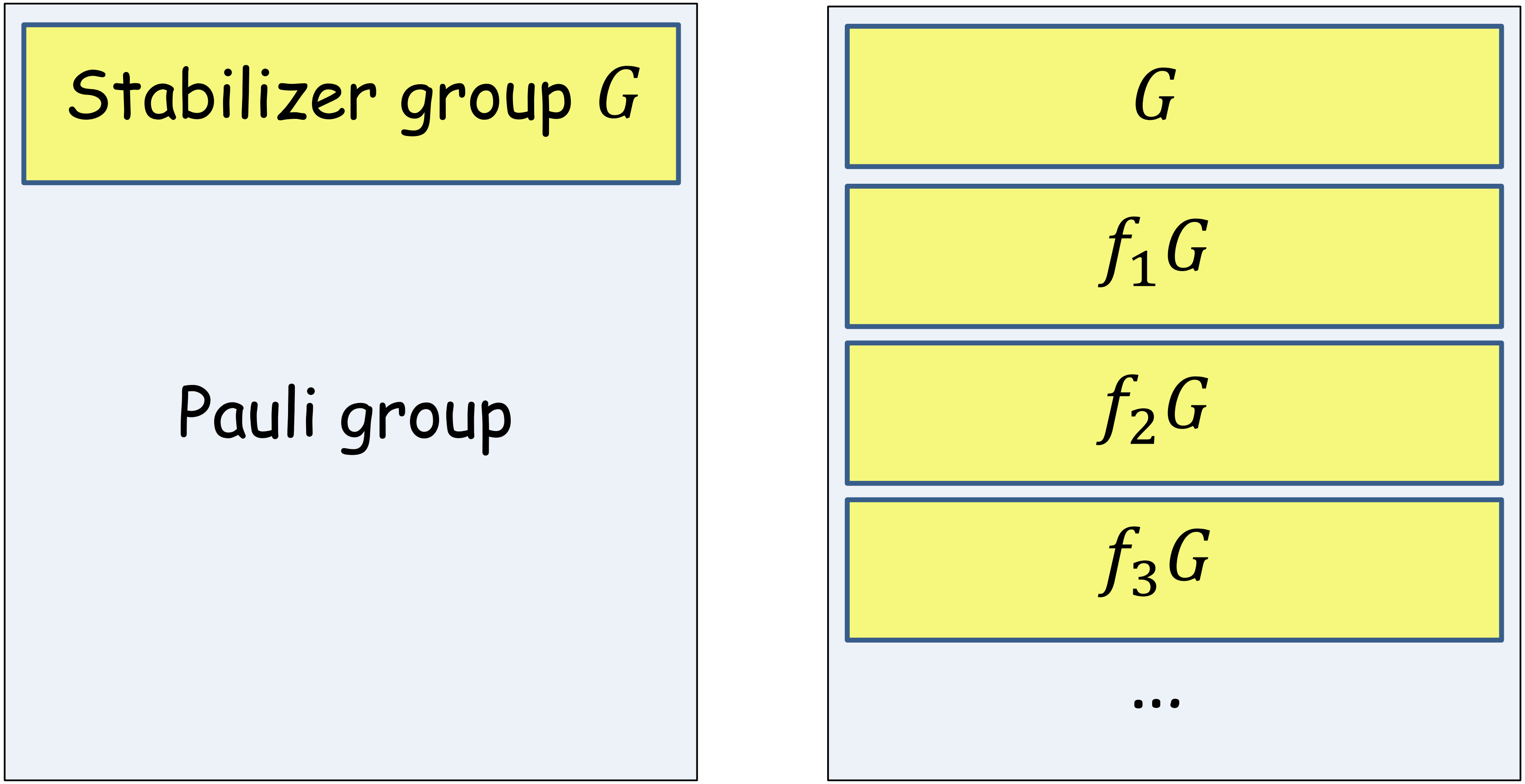
$\{I, X, Y, Z\}^{\otimes n}$



cosets of the stabilizer group



Errors in the same coset have the same action on the codespace



Coset probability: $$\Pr(fG) = \sum_{g \in G} \Pr(fg)$$

The four cosets consistent with the syndrome s :

I-coset	X-coset
$f(s)G$	$f(s)\bar{X}G$

Y-coset	Z-coset
$f(s)\bar{Y}G$	$f(s)\bar{Z}G$

We fixed some canonical error $f(s)$ consistent with s

$\bar{X}, \bar{Y}, \bar{Z}$ are the logical operators

The four cosets consistent with the syndrome s :

I-coset	X-coset
$f(s)G$	$f(s)\bar{X}G$

Y-coset	Z-coset
$f(s)\bar{Y}G$	$f(s)\bar{Z}G$

Coset probability:

$$\Pr(fG) = \sum_{g \in G} \Pr(fg)$$

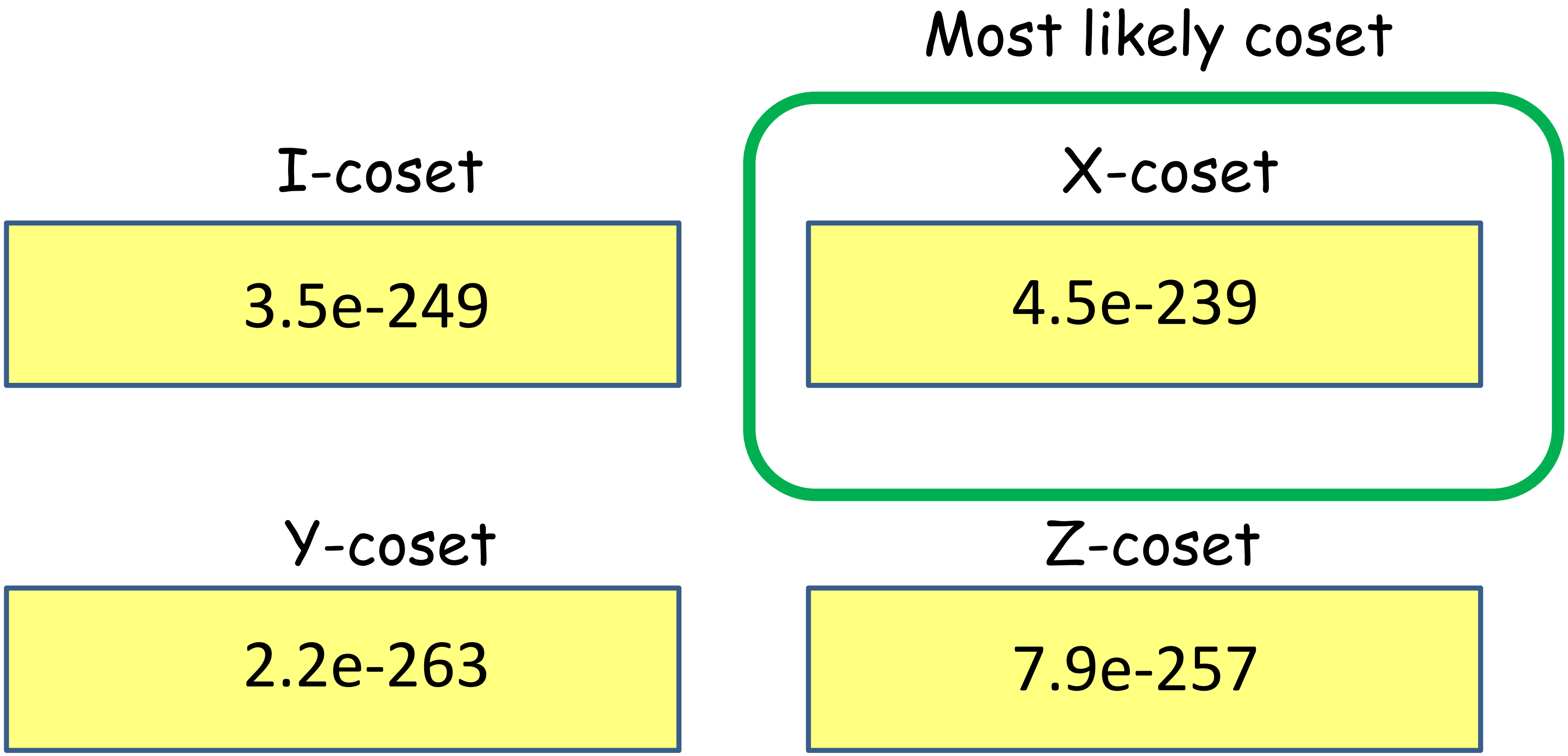
The four cosets consistent with the syndrome s :

I-coset	X-coset
3.5e-249	4.5e-239

Y-coset	Z-coset
2.2e-263	7.9e-257

Real example for d=25, ϵ=10%

Coset probability: $$\Pr(fG) = \sum_{g \in G} \Pr(fg)$$



All errors in the same coset have the same action on the codespace

The optimal decoding strategy is to pick the most likely coset.

Approximate algorithm for MLD:

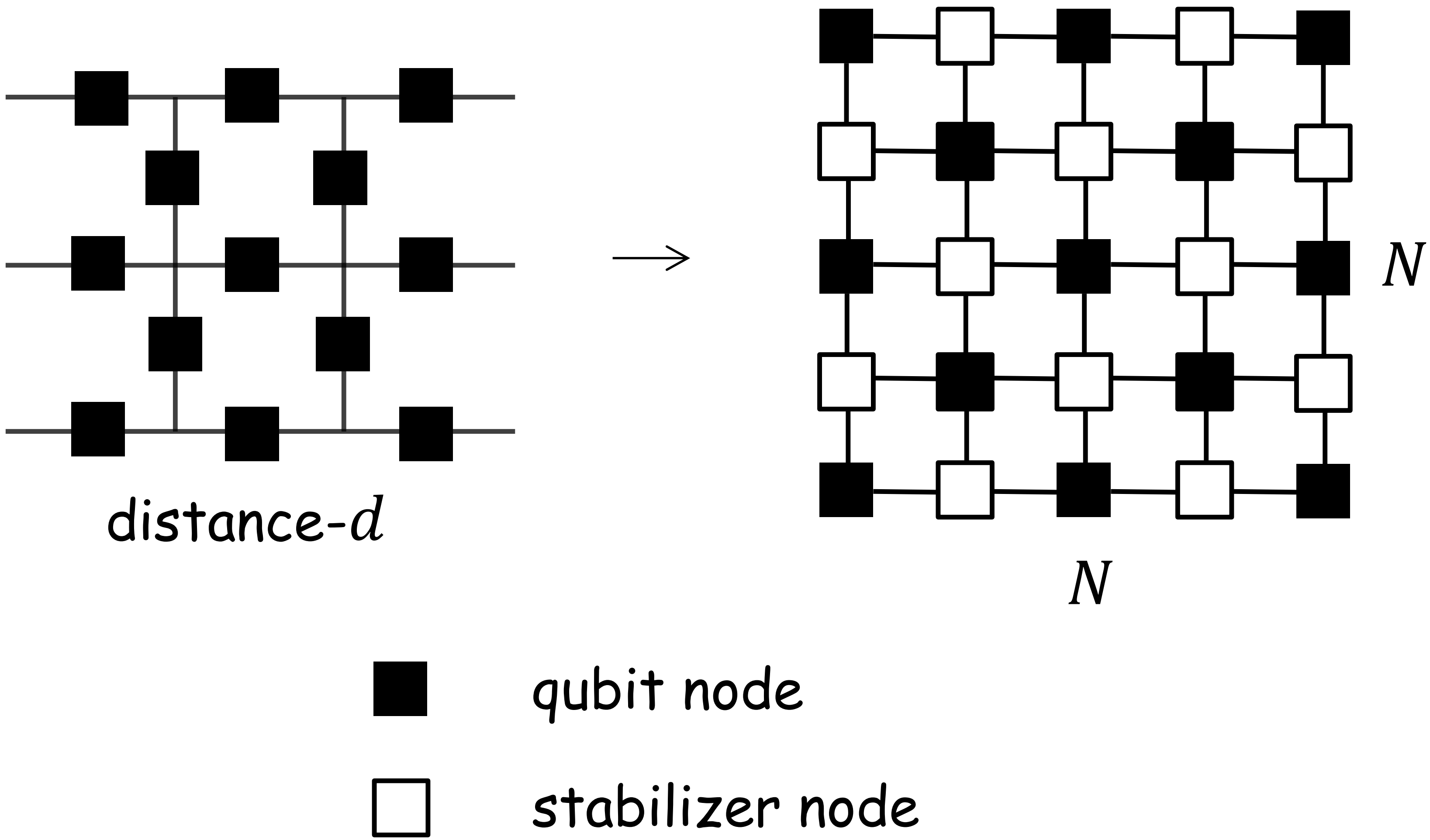
Step 1: express the coset probability as a contraction of a tensor network on a 2D grid.

Step 2: contract the network column by column using matrix product states

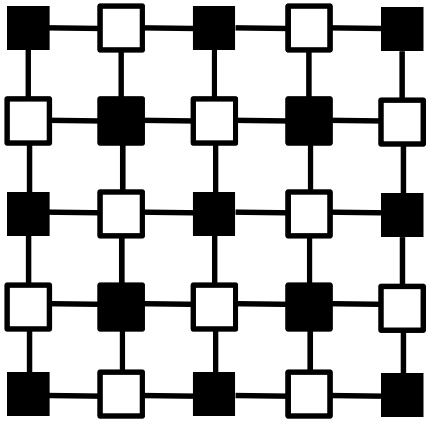
Illustrative example: the trivial coset

$$\Pr(G) = \sum_{g \in G} \Pr(g)$$

Extended surface code lattice

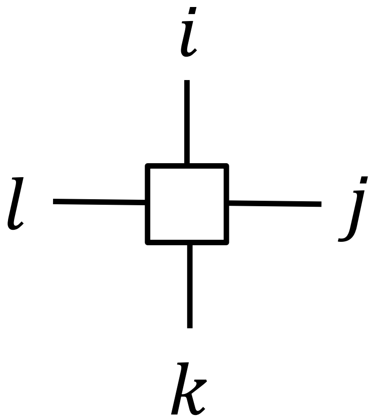


$$N = 2d - 1$$



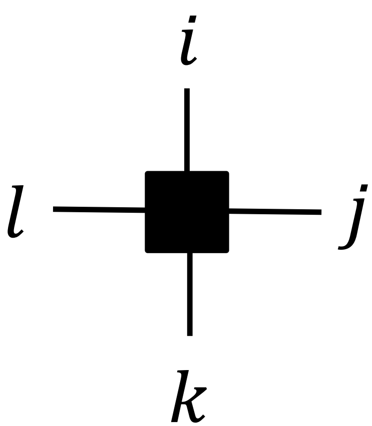
Nodes = tensors

Edges = tensor indexes (0 or 1)



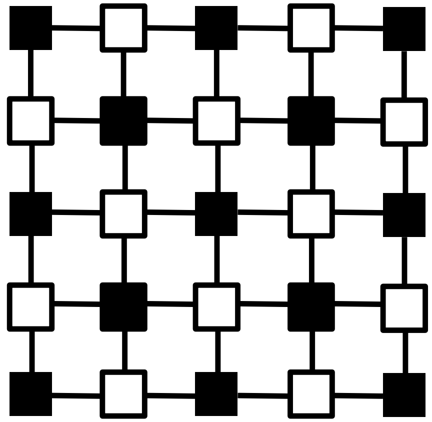
$$T_{i,j,k,l} = \begin{cases} 1 & \text{if} \quad i = j = k = l \\ 0 & \text{otherwise} \end{cases}$$

depends only on the code



$$T_{i,j,k,l} = \begin{cases} 1 - \epsilon & \text{if} \quad i \oplus k = j \oplus l = 0 \\ \epsilon/3 & \text{otherwise} \end{cases}$$

depends on the coset



Nodes = tensors

Edges = tensor indexes (0 or 1)

Contraction value of a tensor network :

$$c = \sum_{\gamma} \prod_{nodes} T(\gamma)$$

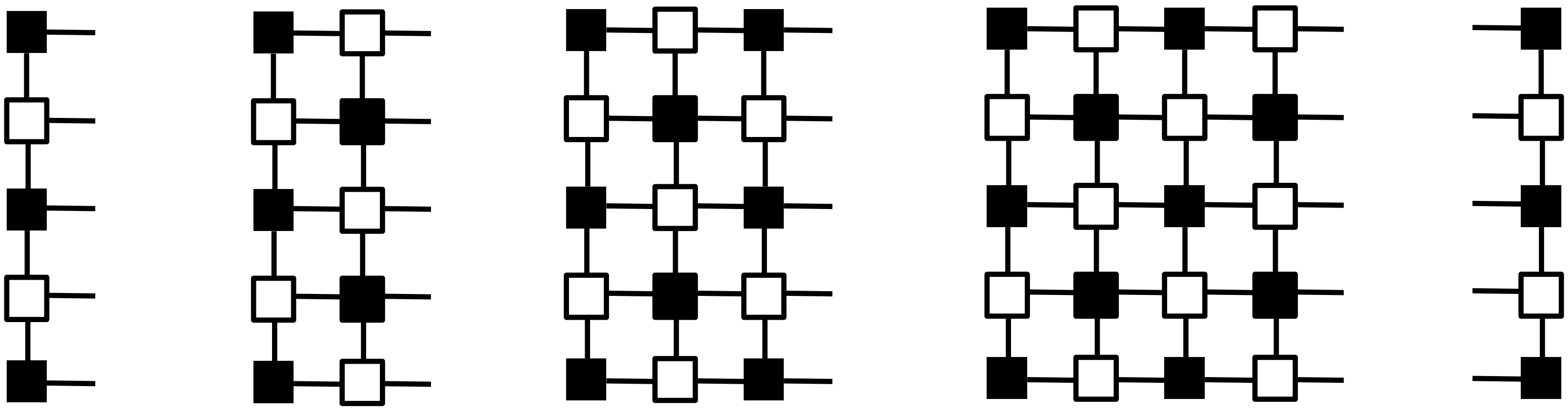
$\gamma =$ edge labeling by 0 and 1

$$\Pr(G) = c$$

Approximate contraction of 2D tensor networks

Murg, Verstraete, Cirac PRA 75, 033605 (2007)

Think of the contraction as a sequence of N-qubit states:



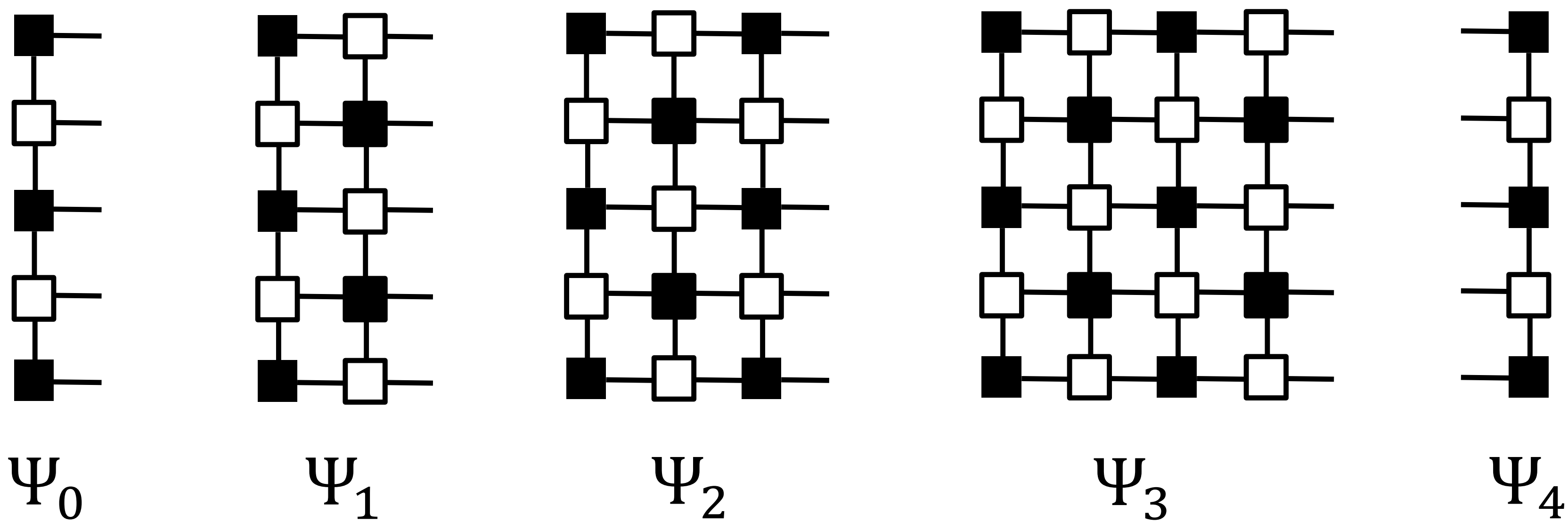
Ψ_0 Ψ_1 Ψ_2 Ψ_3 Ψ_4

$$\Pr(G) = \langle \Psi_3 | \Psi_4 \rangle$$

Approximate contraction of 2D tensor networks

Murg, Verstraete, Cirac PRA 75, 033605 (2007)

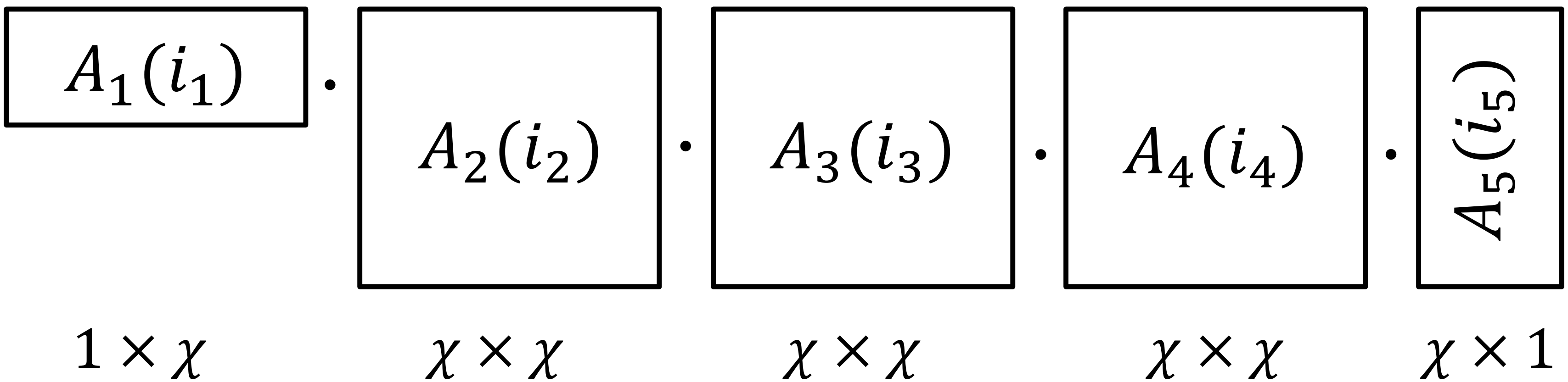
Think of the contraction as a sequence of N-qubit states:



Let's hope that the time evolution is weakly-entangling. Approximate Ψ's by matrix product states with a small bond dimension.

Matrix Product States (MPS)

$$\langle i_1 i_2 i_3 i_4 i_5 | \Psi \rangle =$$



χ - bond dimension

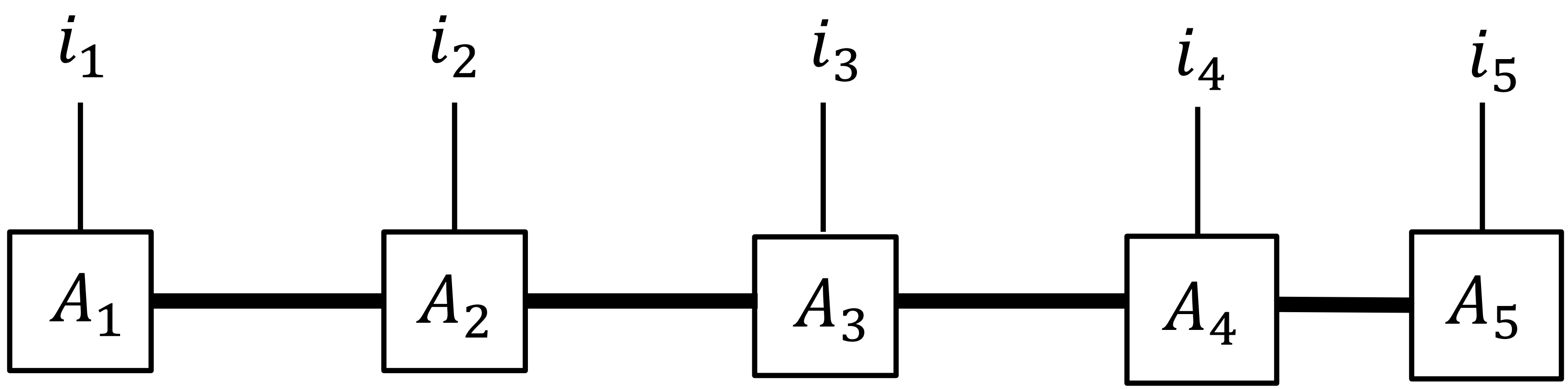
MPS admits a concise description as a list of matrices ($N\chi^2$ real parameters)

Matrix Product States (MPS)

$$\langle i_1 i_2 i_3 i_4 i_5 | \Psi \rangle =$$

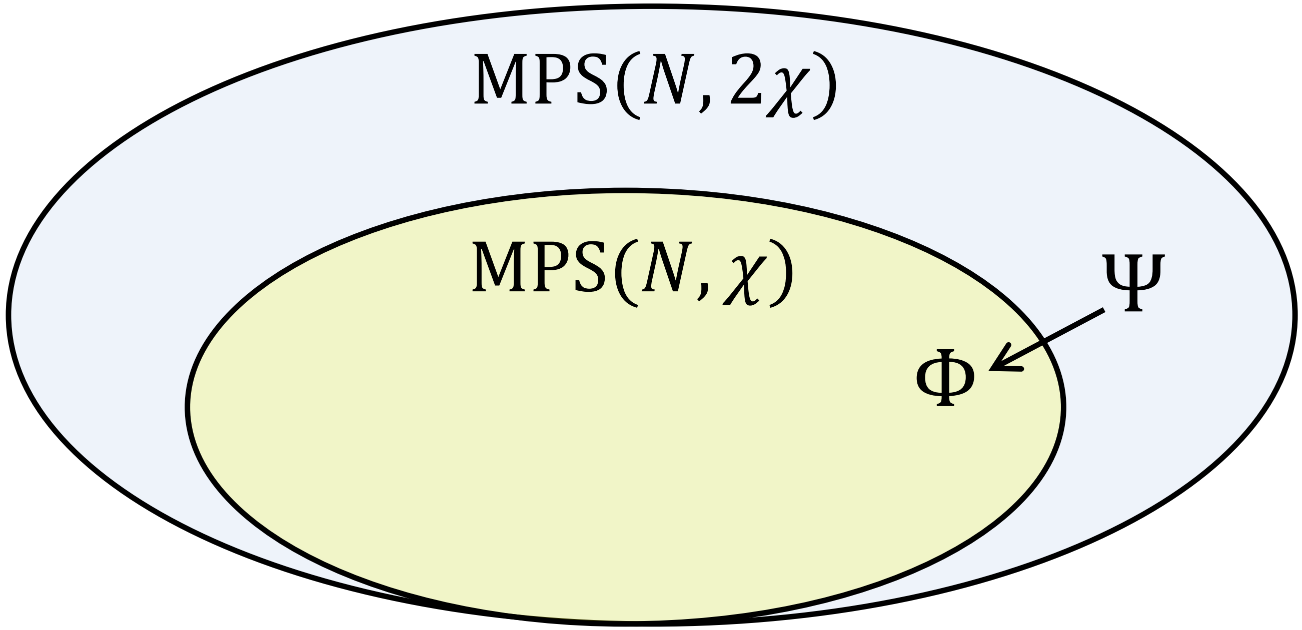
$$A_1(i_1) \cdot A_2(i_2) \cdot A_3(i_3) \cdot A_4(i_4) \cdot A_5(i_5)$$

$=$



Fact 1: Suppose $\Psi, \Phi \in \mathrm{MPS}(N, \chi)$. Then the inner product $\langle\Psi|\Phi\rangle$ can be computed in time $O(N\chi^3)$

MPS compression



Efficient compression algorithm:
Schollwock, Ann. Phys. 326, 96 (2011)

Fact 2: MPS with a bond dimension 2χ can be approximated by an MPS with a bond dimension χ in time

$$N \cdot \mathrm{svd}(2\chi) + N \cdot \mathrm{qr}(2\chi) = O(N\chi^3)$$

How large bond dimension do we need ?