Quantum Error Correction

Zhenyu Cai

UNIVERSITY OF OXFORD

Where is my quantum computer?

Images from Google

Common Causes of Errors

- Imperfect control
- Interaction with the environment

How do we deal with them?

Quantum Error Correction

Classical Repetition Code

- **Without error correction**

Noisy Channel

$$\begin{tikzcd} 0 \arrow[r, "\text{Error with prob. } p"] & 1 \end{tikzcd}$$

- **With Error Correction**

Encode — *Noisy Channel* — *Correction* — *Decode*

$$\begin{tikzcd}
& 0 \arrow[r] & 0 \arrow[r] & 0 \arrow[dr] & \\
0 \arrow[ur] \arrow[r] \arrow[dr] & 0 \arrow[r, "\text{Error}"] & 1 \arrow[r, "\text{Correct}"] & 0 \arrow[r] & 0 \\
& 0 \arrow[r] & 0 \arrow[r] & 0 \arrow[ur] &
\end{tikzcd}$$

Fail when 2 errors occurred, with prob. $\mathcal{O}(p^2)$

Quantum Repetition Code

$$\alpha|\bar{0}\rangle + \beta|\bar{1}\rangle \;\cancel{\longrightarrow}\; \begin{array}{c} \alpha|0\rangle + \beta|1\rangle \\ \alpha|0\rangle + \beta|1\rangle \\ \alpha|0\rangle + \beta|1\rangle \end{array}$$

1 logical qubit

- Quantum no-cloning theorem: cannot create identical copies of a unknown state.

Quantum Repetition Code

$$\alpha|\bar{0}\rangle + \beta|\bar{1}\rangle \xrightarrow{\textit{Encode}} \alpha|000\rangle + \beta|111\rangle \xrightarrow{\textit{Error}} \alpha|010\rangle + \beta|101\rangle$$

1 logical qubit

3 physical qubits

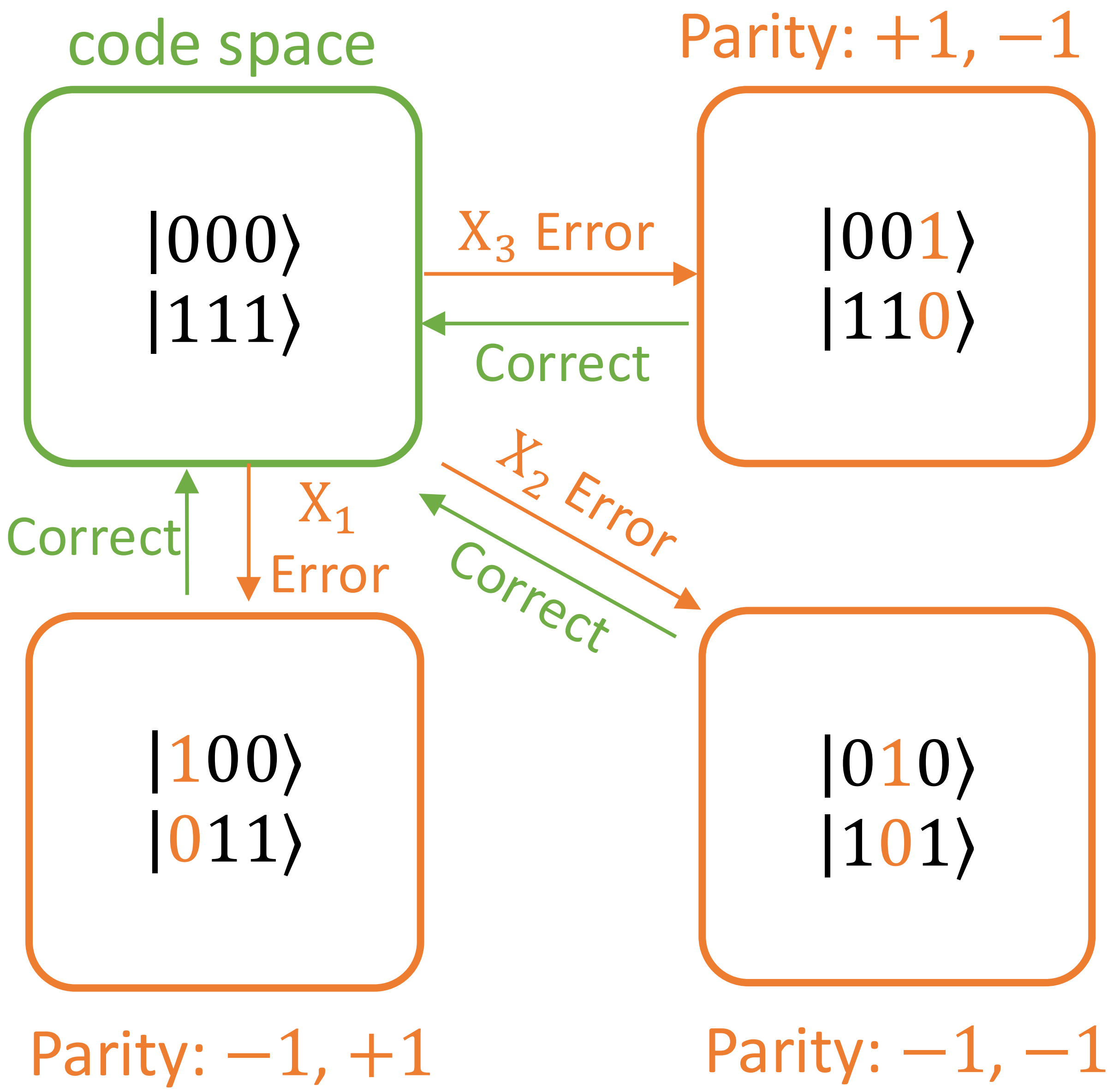
- If we try to detect the errors by measuring the qubits, then the state will collapse into $|010\rangle$ or $|101\rangle$, destroying the superposition.

Measuring the errors, not the state

- Instead of measuring the state, we will perform error checks.
- We measure the parity of the first two qubits, and the parity of the last two qubits. We can then infer the errors using the following table without measuring the state.

State	Error check 1: Parity of 1,2	Error check 2: Parity of 2,3	Flipped qubit
$\alpha\|000\rangle + \beta\|111\rangle$	+1	+1	None
$\alpha\|001\rangle + \beta\|110\rangle$	+1	-1	3
$\alpha\|100\rangle + \beta\|011\rangle$	-1	+1	1
$\alpha\|010\rangle + \beta\|101\rangle$	-1	-1	2

Another way of viewing the code



Stabiliser Formalism

Stabiliser formalism

- Instead of defining a code using the basis states, we start by defining the whole code space using the set of error checks we perform.
- From now on, we will only focus on error checks that are Pauli operators, which we will call stabilisers.

Stabiliser formalism

- The code space $\mathcal{C}$ is defined as the collection of states that are invariant under the action of (or 'stabilised' by) a set of Pauli operators $\mathbb{S}$ called the stabilisers:

$$\mathcal{C} = \{|\psi\rangle \in \mathcal{H} \ \ | \ \ S|\psi\rangle = |\psi\rangle \ \ \forall S \in \mathbb{S}\}$$

- i.e. for any given state, we can check whether it is in the code space by measuring the stabilisers on the state. It is in the code space iff the measuring results are $+1$ for all $S \in \mathbb{S}$.

Properties of stabilisers

- By definition, the set of stabilisers is a subgroup of the N-qubit Pauli group $\mathbb{G}$:
$$\mathbb{S} \subseteq \mathbb{G} = \{\pm 1, \pm i\} \times \{I, X, Y, Z\}^{\otimes N}$$
- In order to have non-empty code space, the stabiliser group must not contain $-I$ (no states are stabilised by $-I$).
- Note that this also implies that the stabiliser group is abelian (since Pauli operators either commute or anti-commute).

Stabiliser generators

- The generators of the stabiliser group $\mathbb{S}$ is denoted as $\widetilde{\mathbb{S}}$.
- If a state is stabilised by all $\tilde{S} \in \widetilde{\mathbb{S}}$, then it is also stabilised by all $S \in \mathbb{S}$. Hence, in order to check whether a state is in the code space, it is sufficient to check whether it is stabilised by all elements in the stabiliser generators $\widetilde{\mathbb{S}}$, no need to check all stabilisers $\mathbb{S}$.
- These stabiliser generators are also called stabiliser checks.
- $|\mathbb{S}| = 2^{|\widetilde{\mathbb{S}}|}$, thus checking only the generators are so much easier!

Number of logical qubits

- Given N physical qubits, enforcing the constraint of each stabiliser check will freeze out one qubit degree of freedom.
- Hence, given $|\tilde{\mathbb{S}}|$ stabiliser checks, the number of remaining qubit degree of freedom, which is the number of logical qubits encoded, is given by

$$k = N - |\tilde{\mathbb{S}}|$$

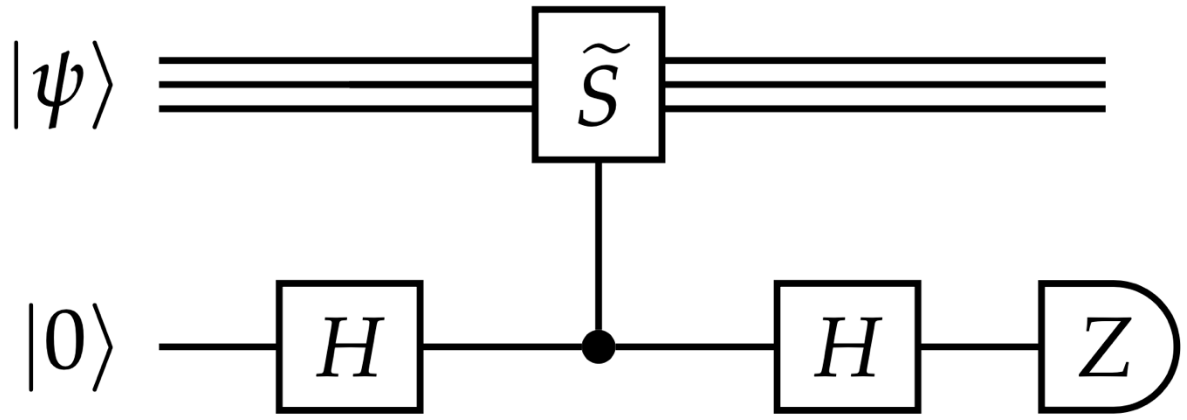
i.e. the dimension of the code space is $2^k = 2^{N-|\tilde{\mathbb{S}}|}$.

Example: 3-qubit bit-flip repetition code

- The set of stabiliser checks is $\widetilde{\mathbb{S}} = \{Z_1 Z_2,\ Z_2 Z_3\}$.
- The full set of stabilisers is $\mathbb{S} = \langle \widetilde{\mathbb{S}} \rangle = \{I,\ Z_1 Z_2,\ Z_2 Z_3,\ Z_1 Z_3\}$
- The number of logical qubits encoded is $\mathrm{k} = \ N - |\widetilde{\mathbb{S}}| = 3 - 2 = 1$
- The code space is simply the common $+1$ eigenspace of these stabiliser checks, which is spanned by $\{|000\rangle, |111\rangle\}$, a space of dimension-2.

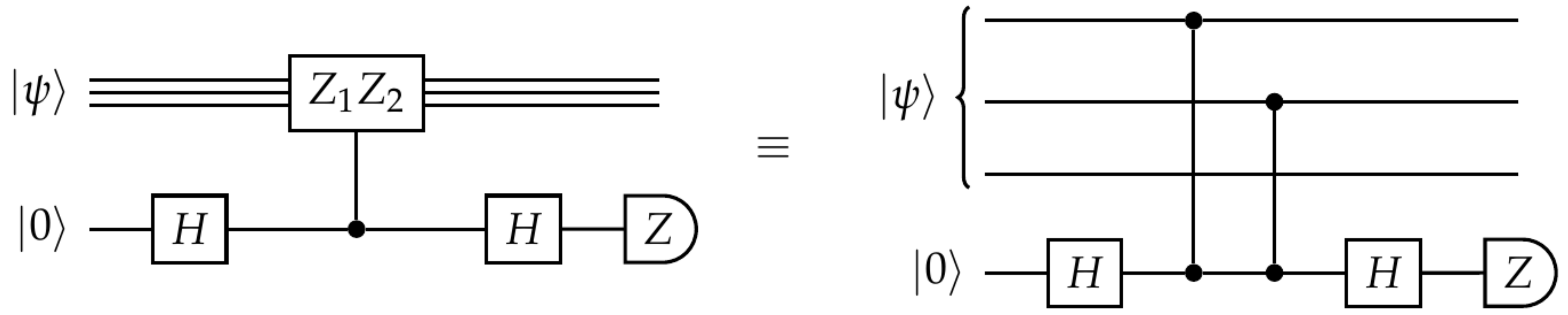
How to perform stabiliser checks

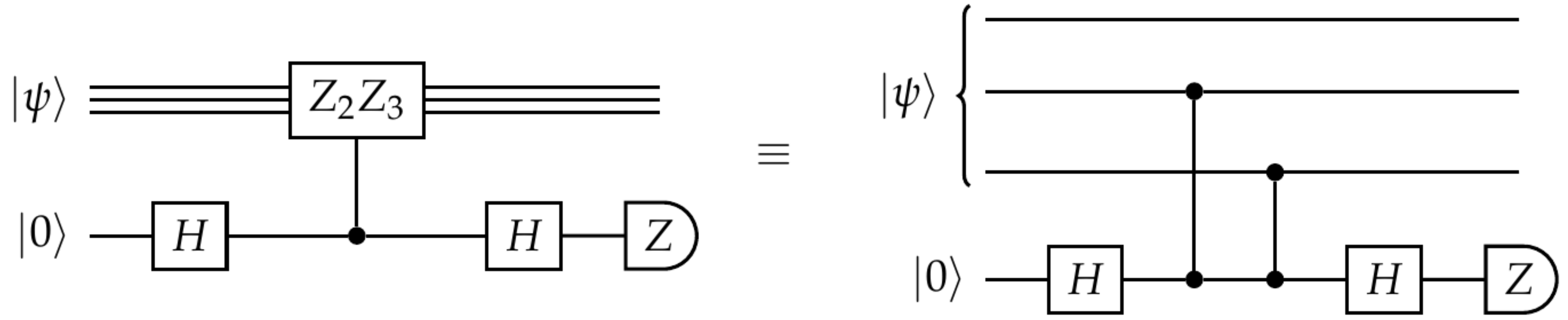
- In order to check whether an incoming state $|\psi\rangle$ is in the code space, we can measure the stabiliser generators $\tilde{S} \in \tilde{\mathbb{S}}$ using the following circuit:



Example: 3-qubit bit-flip repetition code

- Performing stabiliser checks $\widetilde{\mathbb{S}} = \{Z_1Z_2,\ Z_2Z_3\}$.





Error syndrome

- Measuring the ith stabiliser checks $\tilde{S}_i$ will output the measurement result $s_i = \pm 1$.
- The collection of measurement results for all stabiliser checks $\tilde{S}_i \in \widetilde{\mathbb{S}}$ is denoted as $\vec{s} = \{s_1, s_2, \ldots\}$, and is called the error syndrome.
- If the error syndrome is all $+1$: $\vec{s} = \vec{1}$, it means that the incoming state passes all stabiliser checks and thus it is in the code space.

- Note that there are also conventions that uses the label of the eigenstate to denote the measurement result, i.e. using state label 0 instead of the +1 eigenvalue, the state label 1 instead of -1 eigenvalue. In that case, the syndrome entry is given by $m_i = (-1)^{s_i}$ and the all-zero syndrome $\vec{m} = \vec{0}$ denotes passing all stabiliser checks.

Detecting Pauli errors

- Suppose we have a Pauli error E occurs on the code state $|\bar{\psi}\rangle \in \mathcal{C}$, what would be the error syndrome?
- The Pauli error E will commute or anti-commute with the stabiliser generators $\tilde{S}_i \in \widetilde{\mathbb{S}}$:

$$\tilde{S}_i E = \eta(E, \tilde{S}_i) E \tilde{S}_i$$

with the commutator given by

$$\eta(E, \tilde{S}_i) = \begin{cases} +1, & \text{if } E \text{ and } \tilde{S}_i \text{ commute} \\ -1, & \text{if } E \text{ and } \tilde{S}_i \text{ anticommute} \end{cases}$$

Detecting Pauli errors

- Performing the stabiliser check $\tilde{S}_i$ on the erroneous state $E|\bar{\psi}\rangle$ gives:

$$\tilde{S}_i \; E|\bar{\psi}\rangle = \eta(E, \tilde{S}_i) E \tilde{S}_i |\bar{\psi}\rangle = \eta(E, \tilde{S}_i) \; E|\bar{\psi}\rangle$$

- i.e. the measurement output is simply the commutator $\eta(E, \tilde{S}_i)$.
- For a given Pauli error E, the error syndrome that we will obtained by performing the stabiliser checks is simply:

$$\vec{s}(E) = \{\eta(E, \tilde{S}_1), \eta(E, \tilde{S}_2), \eta(E, \tilde{S}_3), \dots\}$$

Minimum-weight decoding

- The process of using the error syndrome to deduce the errors that has occurred (and then choose the right correction operator) is called decoding.
- The weight of a Pauli operator is the number of qubits that it acts non-trivially on. E.g. $Z_1 Z_2$ is weight-2, $X_2 X_4 X_9 X_{10}$ is weight-4.
- Errors with smaller weights are more likely to occur.
- Minimum-weight decoding pick the minimum weight operators that give rise to the syndrome as our guess for the error that occurred. This is also our correction operator since Pauli operators are self-inverse.

The Full Process of QEC

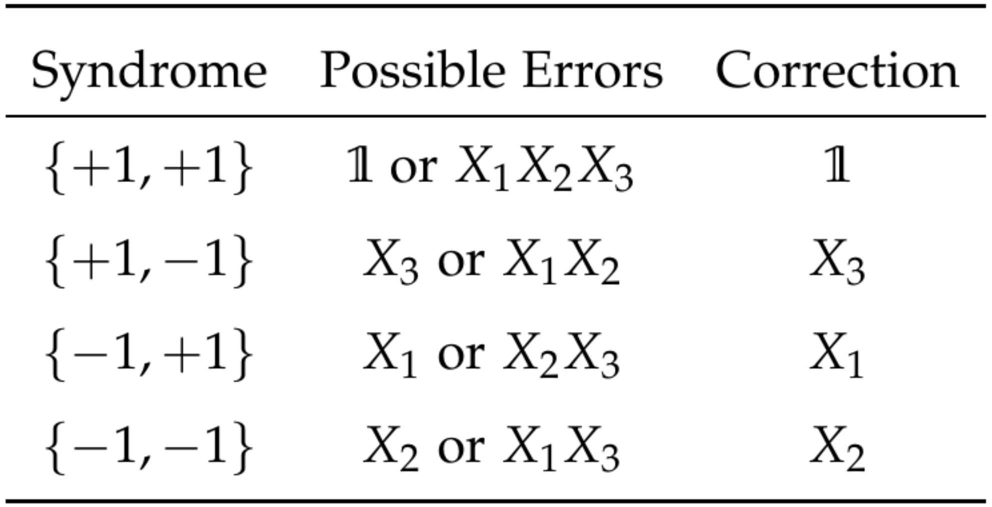
$$\begin{tikzcd}
\begin{array}{c}\text{Erroneous State}\\ E\left|\bar{\psi}\right\rangle\end{array} \arrow[rr, "\begin{array}{c}\text{Measuring stb checks}\\ \widetilde{\mathbb{S}} = \{\tilde{S}_1, \tilde{S}_2 \cdots\}\end{array}"] & & \begin{array}{c}\text{Error Syndrome}\\ \vec{s}(E) = \{\eta\left(E, \tilde{S}_1\right), \eta\left(E, \tilde{S}_2\right), \cdots\}\end{array} \arrow[d, "\begin{array}{c}\text{Finding minimum}\\ \text{weight operator } R \text{ with}\\ \text{the given syndrome:}\\ \vec{s}(R) = \vec{s}(E)\end{array}"] \\
& & \text{Recover operator } R \arrow[llu, "\text{Apply correction}"']
\end{tikzcd}$$

Since R and E anti-commute with the same stabiliser checks, RE will commute with all stabiliser checks and thus $RE\left|\bar{\psi}\right\rangle$ is back into the code space

Example: 3-qubit bit-flip repetition code

- Stabiliser checks $\widetilde{\mathbb{S}} = \{Z_1 Z_2,\ Z_2 Z_3\}$
- E.g. syndrome for the error X_1:

$$\vec{s}(X_1) = \{\eta(X_1, Z_1 Z_2), \eta(X_1, Z_2 Z_3) = \{-1, +1\}$$



Syndrome	Possible Errors	Correction
$\{+1,+1\}$	$\mathbb{1}$ or $X_1X_2X_3$	$\mathbb{1}$
$\{+1,-1\}$	X_3 or X_1X_2	X_3
$\{-1,+1\}$	X_1 or X_2X_3	X_1
$\{-1,-1\}$	X_2 or X_1X_3	X_2

- Wrong correction is applied if two bit flips happens together, thus the code fail with probability $\mathcal{O}(p^2)$

Beyond Bit-flip Errors

Phase errors

- Bit-flip repetition code:

  $\widetilde{\mathbb{S}} = \{Z_1Z_2,\ Z_2Z_3\}$ → detect X errors.

- Exchange the role of X and Z → phase-flip repetition code

  $\widetilde{\mathbb{S}} = \{X_1X_2,\ X_2X_3\}$→ detect Z errors.

$$\begin{tikzcd}
\alpha|\bar{+}\rangle + \beta|\bar{-}\rangle \arrow[r, "\textit{Encode}"] & \alpha|+++\rangle + \beta|---\rangle \arrow[r, "\textit{Z Error}"] & \alpha|+-+\rangle + \beta|-+-\rangle \arrow[l, bend left, "\textit{Correct}"]
\end{tikzcd}$$

Shor 9-qubit Code

Bit-flip code

$$|\widetilde{0}\rangle \mapsto |000\rangle$$
$$|\widetilde{1}\rangle \mapsto |111\rangle$$

Phase-flip code

$$|\overline{+}\rangle \mapsto |+++\rangle$$
$$|\overline{-}\rangle \mapsto |---\rangle$$

Shor's 9-qubit code: concatenating bit-flip and phase-flip code

$$|\overline{\widetilde{+}}\rangle \xrightarrow{Eq.~(8)} |\widetilde{+}\widetilde{+}\widetilde{+}\rangle = \frac{1}{2\sqrt{2}}\Big(|\widetilde{0}\rangle + |\widetilde{1}\rangle\Big)^{\otimes 3} \xrightarrow{Eq.~(7)} \frac{1}{2\sqrt{2}}(|000\rangle + |111\rangle)^{\otimes 3}$$
$$|\overline{\widetilde{-}}\rangle \xrightarrow{Eq.~(8)} |\widetilde{-}\widetilde{-}\widetilde{-}\rangle = \frac{1}{2\sqrt{2}}\Big(|\widetilde{0}\rangle - |\widetilde{1}\rangle\Big)^{\otimes 3} \xrightarrow{Eq.~(7)} \frac{1}{2\sqrt{2}}(|000\rangle - |111\rangle)^{\otimes 3}$$

Shor's 9-qubit code

$$\begin{tikzcd}
 & {|000\rangle + |111\rangle} & & {|000\rangle - |111\rangle} \\
 & \otimes & & \otimes \\
{|\overline{\widetilde{+}}\rangle} \arrow[r, "Encode"] & {|000\rangle + |111\rangle} \arrow[rr, shift left, "{Z_1,\, Z_2 \text{ or } Z_3}"] & & {|000\rangle + |111\rangle} \arrow[ll, shift left, "Correct"] \\
 & \otimes & & \otimes \\
 & {|000\rangle + |111\rangle} & & {|000\rangle + |111\rangle}
\end{tikzcd}$$

Shor's 9-qubit code

$$\begin{tikzcd}
 & |000\rangle + |111\rangle & & |010\rangle + |101\rangle \\
 & \otimes & & \otimes \\
|\bar{\tilde{+}}\rangle \arrow[r, "\textit{Encode}"] & |000\rangle + |111\rangle \arrow[rr, shift left, "X_2"] & & |000\rangle + |111\rangle \arrow[ll, shift left, "\textit{Correct}"] \\
 & \otimes & & \otimes \\
 & |000\rangle + |111\rangle & & |000\rangle + |111\rangle
\end{tikzcd}$$

Shor's 9-qubit code

- The set of stabiliser checks for the Shor's code is

$$\widetilde{\mathbb{S}} = \{Z_1Z_2 \quad Z_2Z_3 \quad Z_4Z_5 \quad Z_5Z_6 \quad Z_7Z_8 \quad Z_8Z_9 \\ X_1X_2X_3X_4X_5X_6 \quad X_4X_5X_6X_7X_8X_9\}$$

- There are more Z checks than X checks, thus the code can detect and correct more X errors than Z errors.

Shor's 9-qubit code

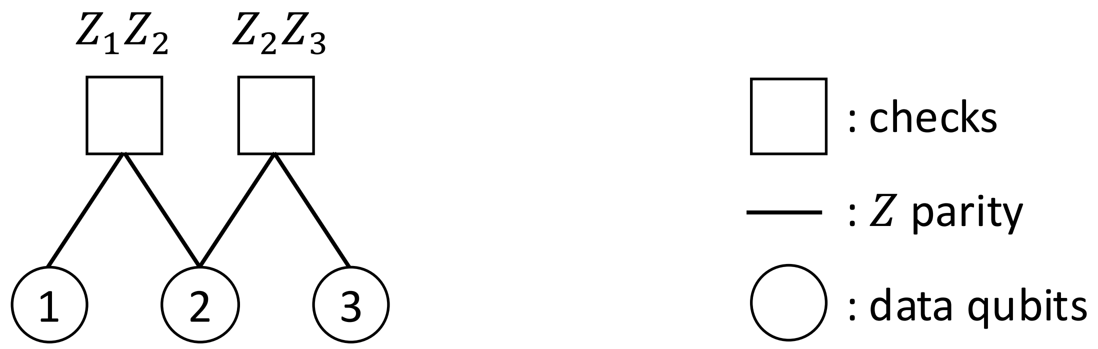
$$\begin{tikzcd}
 & {|000\rangle + |111\rangle} & {|010\rangle + |101\rangle} \\
 & \otimes & \otimes \\
{|\overline{\widetilde{\mp}}\rangle} \arrow[r, "Encode"] & {|000\rangle + |111\rangle} \arrow[r, shift left, "X_2, X_4\ and\ X_9"] & {|100\rangle + |011\rangle} \arrow[l, shift left, "Correct"] \\
 & \otimes & \otimes \\
 & {|000\rangle + |111\rangle} & {|001\rangle + |110\rangle}
\end{tikzcd}$$

Constructing QEC Codes

Tanner Graph

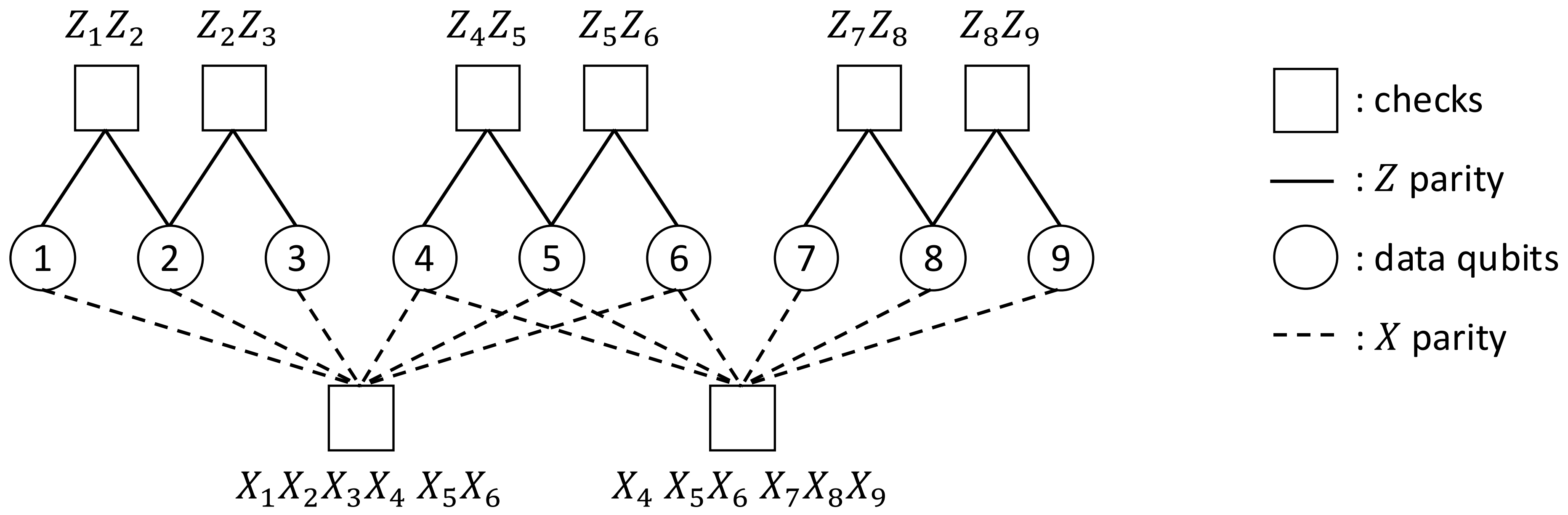
- A Tanner graph is a bipartite graph with the two disjoint subsets of vertices representing the data qubits and the checks (stabiliser generators), respectively.

- A check is connected to a data qubit if and only if it is checking the parity of that data qubit.



CSS Code

- Codes with stb checks that are purely X or purely Z.



- All X checks need to commute with all Z checks

  ➔ In the Tanner graph, any pair of X and Z checks must intersect at an even number of data qubits.

Logical Operations

Logical operations

- Besides protecting the logical information contained in a logical qubit, we would also want to perform logical gates on it in order to carry out computations.
- Can we simply decode, operate and then encode?

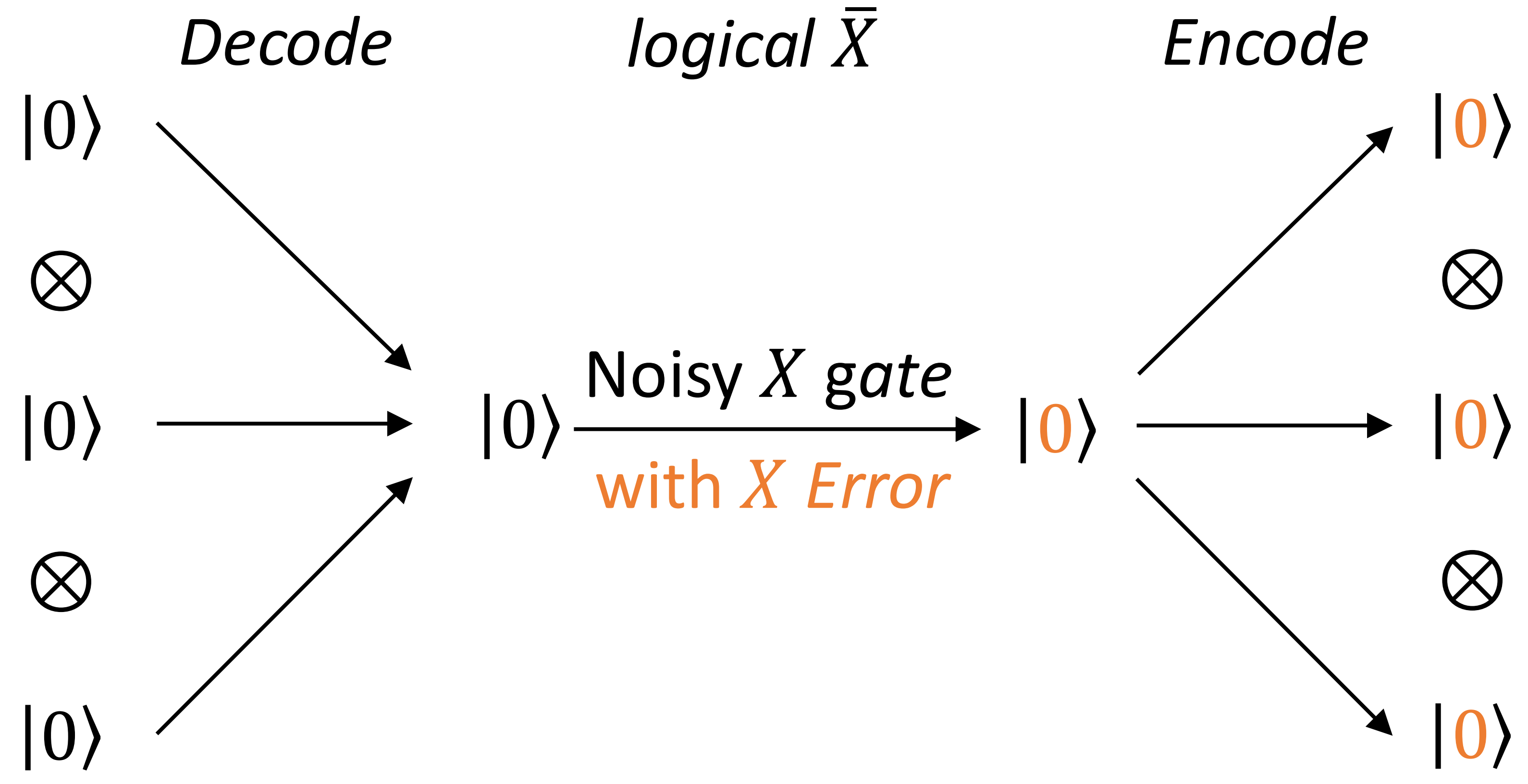
Logical operations

- We want to perform logical $\bar{X}$ gates on the logical qubit $|\bar{0}\rangle = |000\rangle$.

$$\begin{tikzcd}
 & \textit{Decode} & \textit{logical } \bar{X} & \textit{Encode} & \\
|0\rangle \arrow[ddr] & & & & |1\rangle \\
\otimes & & & & \otimes \\
|0\rangle \arrow[r] & |0\rangle \arrow[r, "X\ gate"] & |1\rangle \arrow[uur] \arrow[r] \arrow[ddr] & & |1\rangle \\
\otimes & & & & \otimes \\
|0\rangle \arrow[uur] & & & & |1\rangle
\end{tikzcd}$$

Logical operations

- We want to perform logical $\bar{X}$ gates on the logical qubit $|\bar{0}\rangle = |000\rangle$.



Decode
logical $\bar{X}$
Encode
$|0\rangle \otimes |0\rangle \otimes |0\rangle$
$|0\rangle$
Noisy X *gate*
with X *Error*
$|0\rangle$
$|0\rangle \otimes |0\rangle \otimes |0\rangle$

Logical operations

- We can instead directly operating on the encoded state

logical $\bar{X}$

$$\begin{tikzcd}
{|0\rangle} \arrow[r, "X\ gate"] & {|1\rangle} \\
\otimes & \otimes \\
{|0\rangle} \arrow[r, "X\ gate"] & {|1\rangle} \\
\otimes & \otimes \\
{|0\rangle} \arrow[r, "X\ gate"] & {|1\rangle}
\end{tikzcd}$$

Logical operations

- We can instead directly operating on the encoded state

logical $\bar{X}$

$$\begin{tikzcd}
{|0\rangle} \arrow[r, "X\ gate"] & {|1\rangle} & {|1\rangle} \\
\otimes & \otimes & \otimes \\
{|0\rangle} \arrow[r, "X\ gate", "\text{with } X\ Error"'] & {|0\rangle} \arrow[r, "\text{correct}"] & {|1\rangle} \\
\otimes & \otimes & \otimes \\
{|0\rangle} \arrow[r, "X\ gate"] & {|1\rangle} & {|1\rangle}
\end{tikzcd}$$

Logical operation in stabiliser formalism

- But how do we find the set of logical operations for a given stabiliser code?
- By definition, all stabilisers will act trivially on the code states

$$S|\bar{\psi}\rangle = |\bar{\psi}\rangle \qquad \forall S \in \mathbb{S}, |\bar{\psi}\rangle \in \mathcal{C}$$

Hence, the stabilisers are just logical identity:

$$\mathbb{S} = \bar{I}$$

Logical operation in stabiliser formalism

- Logical Pauli operations must commute with logical identity.
- Hence the set of logical Pauli operations $\overline{\mathbb{G}}$ is simply the subgroup of Pauli group $\mathbb{G}$ that commutes with the stabilisers $\mathbb{S}$.

$$\overline{\mathbb{G}} = \{G \in \mathbb{G} \mid \mathbb{S}G = G\mathbb{S}\}$$

which is simply the normaliser of the stabiliser subgroup of the Pauli group.

- But what logical action does each element in $\overline{\mathbb{G}}$ correspond to?

Identifying Logical Pauli Operators

- Look at their actions on known logical basis state.
- Identify the set of logical X and Z operator for different logical qubits, such that they all satisfy the correct commutation relationship.
- i.e. logical X and Z on the same logical qubit anti-commute, and all other pairings commute.
- All other logical Pauli operators can be generated using logical X and Z.

A second look at the QEC process

- An error E occur on the code state $|\bar{\psi}\rangle$ leading to syndrome $\vec{s}(E)$
- Choose the minimum weight recovery operator R with the same syndrome $\vec{s}(R) = \vec{s}(E)$
- The resultant state $RE|\bar{\psi}\rangle$ is back into the code space → RE is a logical operator
- If RE is a stabiliser → $|\bar{\psi}\rangle$ unchanged → successful quantum error correction.
- If RE is a non-stabiliser logical operator → logical error on $|\bar{\psi}\rangle$ → failed quantum error correction.

Distance

- The weight of the smallest logical errors (non-stabiliser logical operations) is called the distance of the code.
- A code that encodes n physical qubits into k logical qubits with distance d is called an $[[n, k, d]]$ code.
- A code with distance d can correct any errors up to weight $\lfloor (d-1)/2 \rfloor$

Correctable Errors

- The error E has the weight $\mathrm{wt}(E) \leq \lfloor (d-1)/2 \rfloor$.
- Using minimum-weight decoding, the weight of recovery operator R is smaller or equal to the weight of E, which implies $\mathrm{wt}(R) \leq \lfloor (d-1)/2 \rfloor$.
- Hence, the weight of RE need to satisfy:

$$\mathrm{wt}(RE) \leq \mathrm{wt}(R) + \mathrm{wt}(E) \leq d - 1 < d$$

- The smallest weight of the logical error is d → RE is not a logical error, it is a stabiliser → successful quantum error correction

Logical Operators for CSS Codes

- Codes with stb checks that are purely X or purely Z.

- Search for its logical X by looking at tensor products of physical X that commute with all Z checks while not being an X stabiliser.
- Similarly for logical Z.

- The smaller weight between the logical X and Z give us the code distance.

Bit-flip Repetition Code

- CSS code
- Stabiliser checks: $\widetilde{\mathbb{S}} = \{Z_1Z_2,\ Z_2Z_3\}$.
- Code state:$|\bar{0}\rangle = |000\rangle$, $|\bar{1}\rangle = |111\rangle$
- Shortest X string that commute with $\widetilde{\mathbb{S}}$ and not in $\mathbb{S}$: $\bar{X} = X_1X_2X_3$
- Shortest Z string that commute with $\widetilde{\mathbb{S}}$ and not in $\mathbb{S}$: $\bar{Z} = Z_1$
- Code distance: $d = 1$
- $\lfloor (d-1)/2 \rfloor = 0$ since it cannot correct any Z errors.

Shor’s 9-qubit code

- CSS code
- Stabiliser checks: $\widetilde{\mathbb{S}} = \{Z_1Z_2 \quad Z_2Z_3 \quad Z_4Z_5 \quad Z_5Z_6 \quad Z_7Z_8 \quad Z_8Z_9$
  $X_1X_2X_3X_4X_5X_6 \quad X_4X_5X_6X_7X_8X_9\}$
- Code state: $|\mp\rangle \propto (|000\rangle + |111\rangle)^{\otimes 3}$, $|\overline{-}\rangle = (|000\rangle - |111\rangle)^{\otimes 3}$
- Shortest X string that commute with $\widetilde{\mathbb{S}}$ and not in $\mathbb{S}$: $\bar{X} = X_1X_2X_3$
- Shortest Z string that commute with $\widetilde{\mathbb{S}}$ and not in $\mathbb{S}$: $\bar{Z} = Z_1Z_4Z_7$
- Code distance: $d = 3$
- Can correct all errors up to weight $\lfloor (d-1)/2 \rfloor = 1$

General Errors

Syndrome Subspaces

- The projector for the $+1$ eigenspace for the Pauli operator G is:
$$\Pi_+ = (I + G)/2$$
- It is easy to check by applying it on the ±1 eigenstates of G, denoted as $|G_\pm\rangle$:
$$\Pi_+|G_+\rangle = \frac{|G_+\rangle + |G_+\rangle}{2} = |G_+\rangle$$
$$\Pi_+|G_-\rangle = \frac{|G_-\rangle - |G_-\rangle}{2} = 0$$
- Similarly, The projector for the -1 eigenspace of G is:
$$\Pi_- = (I - G)/2$$

Syndrome Subspaces

- Measuring the ith stabiliser check $\tilde{S}_i$ with measurement result $s_i = \pm 1$ will project the incoming state into $s_i = \pm 1$ eigenspace define by the projector
$$\Pi_{s_i} = (I + s_i \tilde{S}_i)/2$$
- After measuring all stabiliser checks $\tilde{S}_i \in \widetilde{\mathbb{S}}$ and obtain the error syndrome $\vec{s}$, the incoming state is projected into the $\vec{s}$-syndrome subspace defined by the projector:

$$\Pi_{\vec{s}} = \prod_{i=1}^{|\widetilde{\mathbb{S}}|} (I + s_i \tilde{S}_i)/2$$

Syndrome Subspaces

- Bit-flip repetition code

$\frac{I+Z_1Z_2}{2}\frac{I+Z_2Z_3}{2}$	(+1, +1)-syndrome no errors $\|000\rangle$ $\|111\rangle$	(+1, -1)-syndrome X_3 errors $\|001\rangle$ $\|110\rangle$	$\frac{I+Z_1Z_2}{2}\frac{I-Z_2Z_3}{2}$
$\frac{I-Z_1Z_2}{2}\frac{I+Z_2Z_3}{2}$	(-1, +1)-syndrome X_1 errors $\|100\rangle$ $\|011\rangle$	(-1, -1)-syndrome X_2 errors $\|010\rangle$ $\|101\rangle$	$\frac{I-Z_1Z_2}{2}\frac{I-Z_2Z_3}{2}$

Linear Combination of Errors

- Error: $E = \alpha_0 I + \alpha_1 X_1 + \alpha_2 X_2 + \alpha_3 X_3$
- Erroneous state: $E|000\rangle = \alpha_0|000\rangle + \alpha_1|100\rangle + \alpha_2|010\rangle + \alpha_3|001\rangle$
- By measuring error syndrome, we will collapse the state above into one of the syndrome subspace with probability $|\alpha_i|^2$.
- This effectively means that we have collapse E into one of the correctable errors.

(+1, +1)-syndrome
no errors

$|000\rangle$
$|111\rangle$

(+1, -1)-syndrome
X_3 errors

$|001\rangle$
$|110\rangle$

(-1, +1)-syndrome
X_1 errors

$|100\rangle$
$|011\rangle$

(-1, -1)-syndrome
X_2 errors

$|010\rangle$
$|101\rangle$

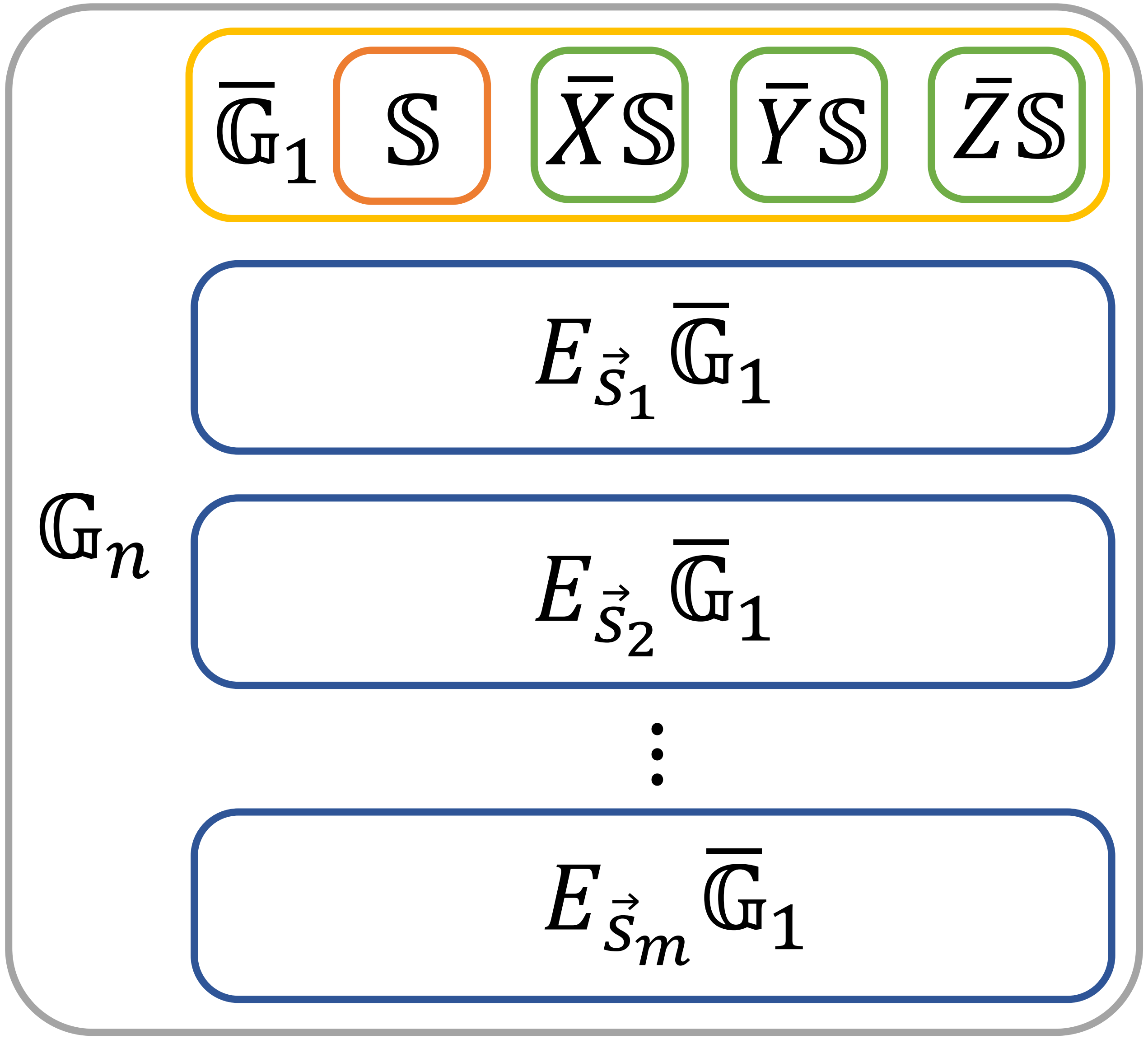
Linear Combination of Errors

- In general, if the incoming error is a linear sum of correctable errors, then the act of syndrome measurement will collapse the incoming error into one of the correctable error components.
- If a QEC code can correct a given set of errors, then it can also correct any linear combinations of these errors.

General errors

- If we can correct X and Z errors, then we can also correct Y errors. In this way, we can correct any linear combination of I, X, Y, Z, which is any single-qubit errors.
- This also extend to multi-qubit: if we can correct X and Z errors up to a certain weight, we can correct any errors up to that weight.

All Pauli operations on 1 logical qubits



$\mathbb{G}_n$
$\overline{\mathbb{G}}_1$
$\mathbb{S}$
$\bar{X}\mathbb{S}$
$\bar{Y}\mathbb{S}$
$\bar{Z}\mathbb{S}$
$E_{\vec{s}_1}\overline{\mathbb{G}}_1$
$E_{\vec{s}_2}\overline{\mathbb{G}}_1$
$\vdots$
$E_{\vec{s}_m}\overline{\mathbb{G}}_1$

$$m = 2^{|\tilde{\mathbb{S}}|}$$
$$= 2^{n-k}$$

Fault tolerance

Every Component is Noisy

- We have only focus on errors spontaneously to qubits, and assuming all the operations we perform are error free.
- In practice, all of the following steps consist of noisy physical components:
  - the preparation (encoding) of the code state,
  - the logical gates,
  - the measurement of the code state,
  - and even the stabiliser checks we perform

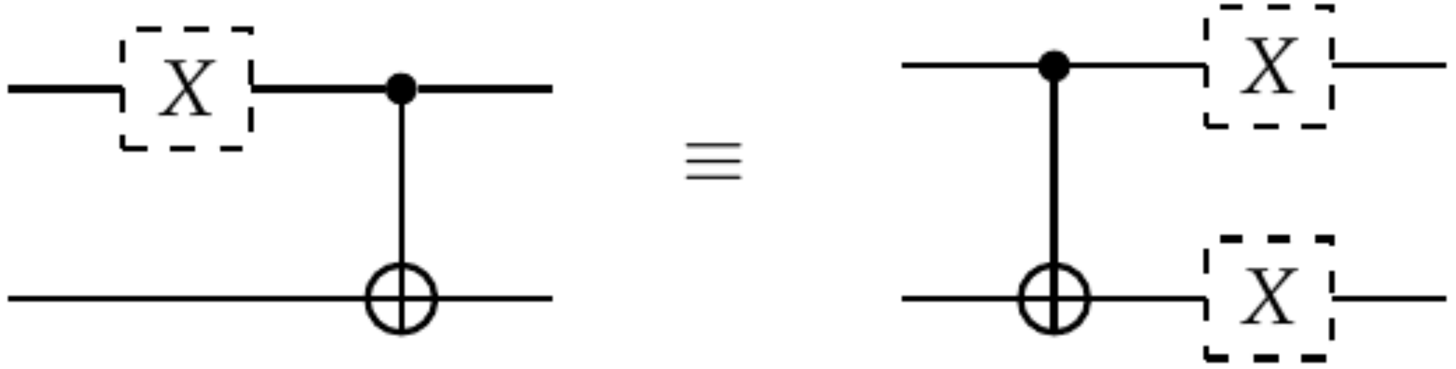
Fault tolerance

- Fault tolerance: the ability to carry out computation up to the desired accuracy and precision using these noisy components.

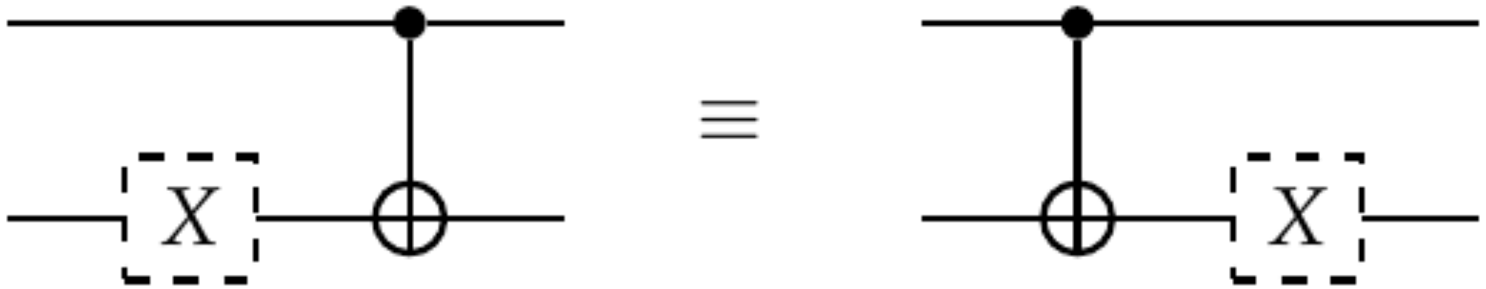
- A key principle for constructing fault-tolerant operations is to prevent errors from propagating in an uncontrollable manner

Error Propagation

$$\text{CNOT}(X \otimes I)\text{CNOT} = (X \otimes X)$$

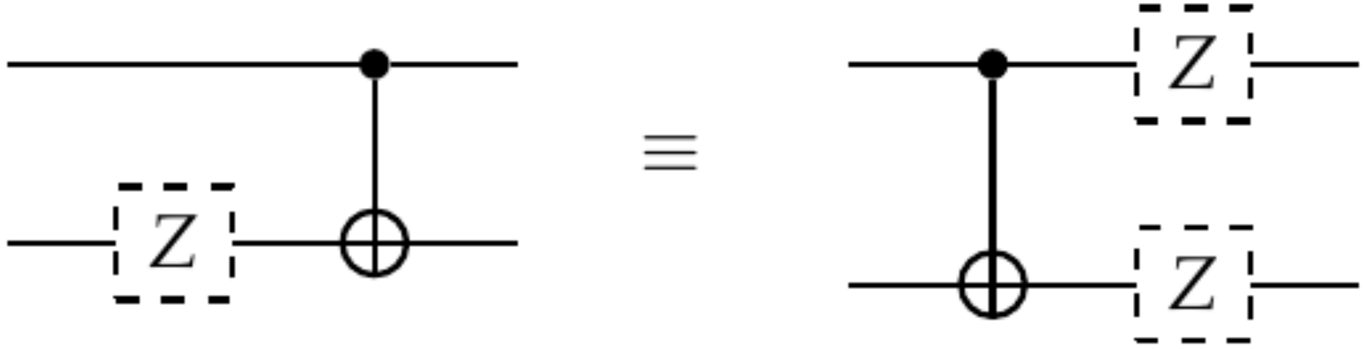


$$\text{CNOT}(I \otimes X)\text{CNOT} = (I \otimes X)$$

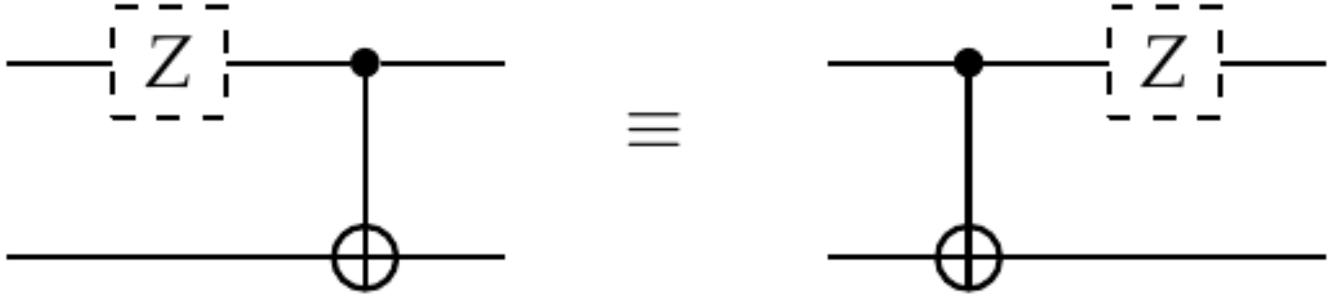


Error Propagation

$$\mathrm{CNOT}(I \otimes Z)\mathrm{CNOT} = (Z \otimes Z)$$



$$\mathrm{CNOT}(Z \otimes I)\mathrm{CNOT} = (Z \otimes I)$$



Transversal Logical Operations

- Transversal logical operations: implementing a logical gate by applying a single layer of physical gates (i.e. physical gates implemented in parallel) which does not contain entangling gates acting on two or more qubits within the same code block.
- No error propagation in the same code block $\rightarrow$ fault tolerance.

Transversal Pauli Gate

- Bit-flip repetition code: transversal logical $\bar{X}$

logical $\bar{X}$

$$\begin{tikzcd}
{|0\rangle} \arrow[r, "X\ gate"] & {|1\rangle} \\
\otimes & \otimes \\
{|0\rangle} \arrow[r, "X\ gate"] & {|1\rangle} \\
\otimes & \otimes \\
{|0\rangle} \arrow[r, "X\ gate"] & {|1\rangle}
\end{tikzcd}$$

Transversal Pauli Gate

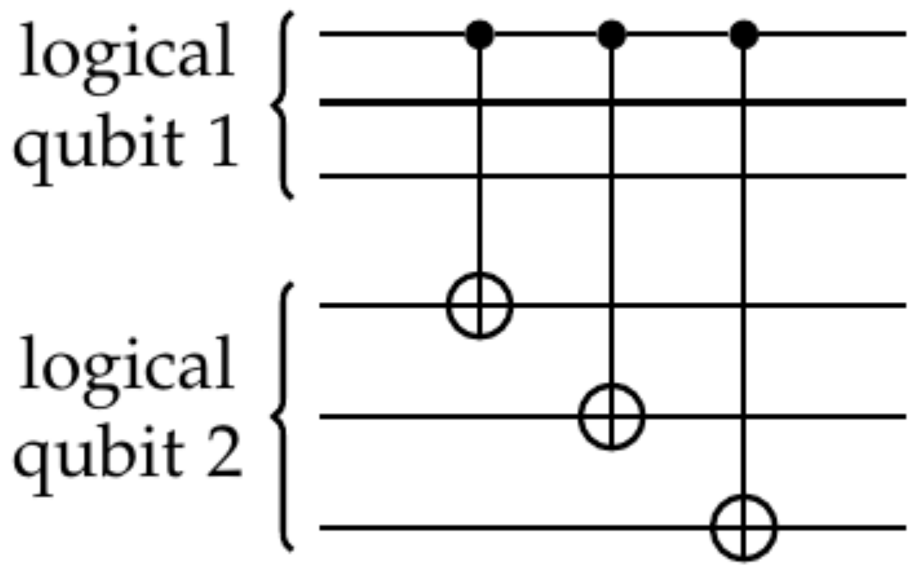
- Bit-flip repetition code: transversal logical $\bar{X}$

logical $\bar{X}$

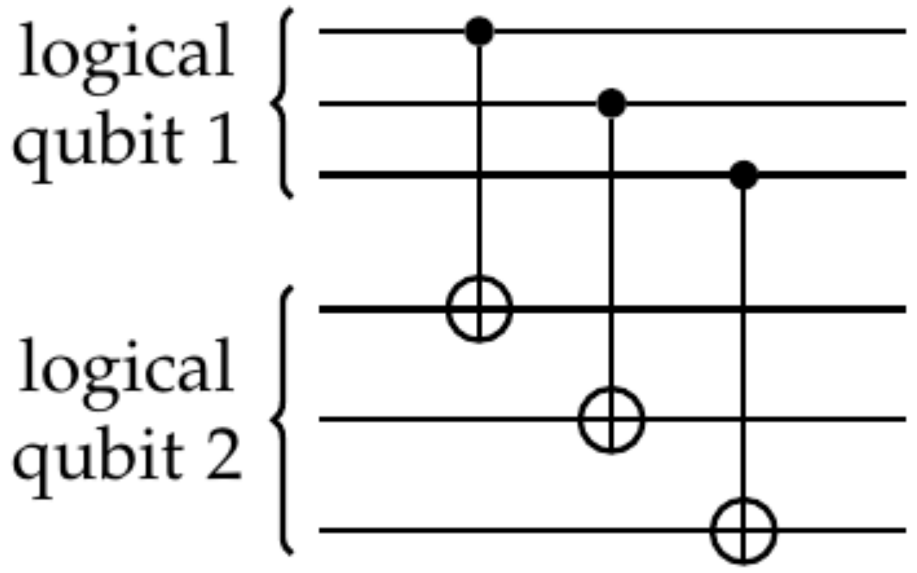
$$\begin{tikzcd}
{|0\rangle} \arrow[r, "X\ gate"] & {|1\rangle} & {|1\rangle} \\
\otimes & \otimes & \otimes \\
{|0\rangle} \arrow[r, "X\ gate", "\text{with } X\ Error"'] & {|0\rangle} \arrow[r, "\text{correct}"] & {|1\rangle} \\
\otimes & \otimes & \otimes \\
{|0\rangle} \arrow[r, "X\ gate"] & {|1\rangle} & {|1\rangle}
\end{tikzcd}$$

Logical CNOTs

- Bit-flip repetition code: logical CNOTs

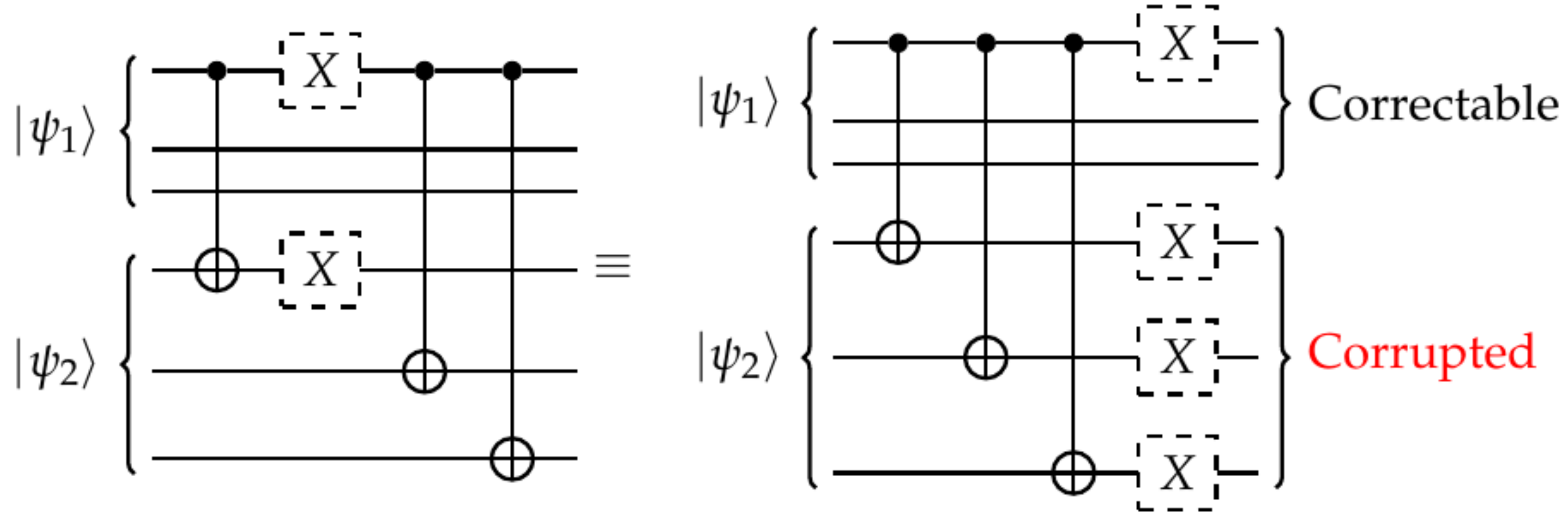


Non-transversal



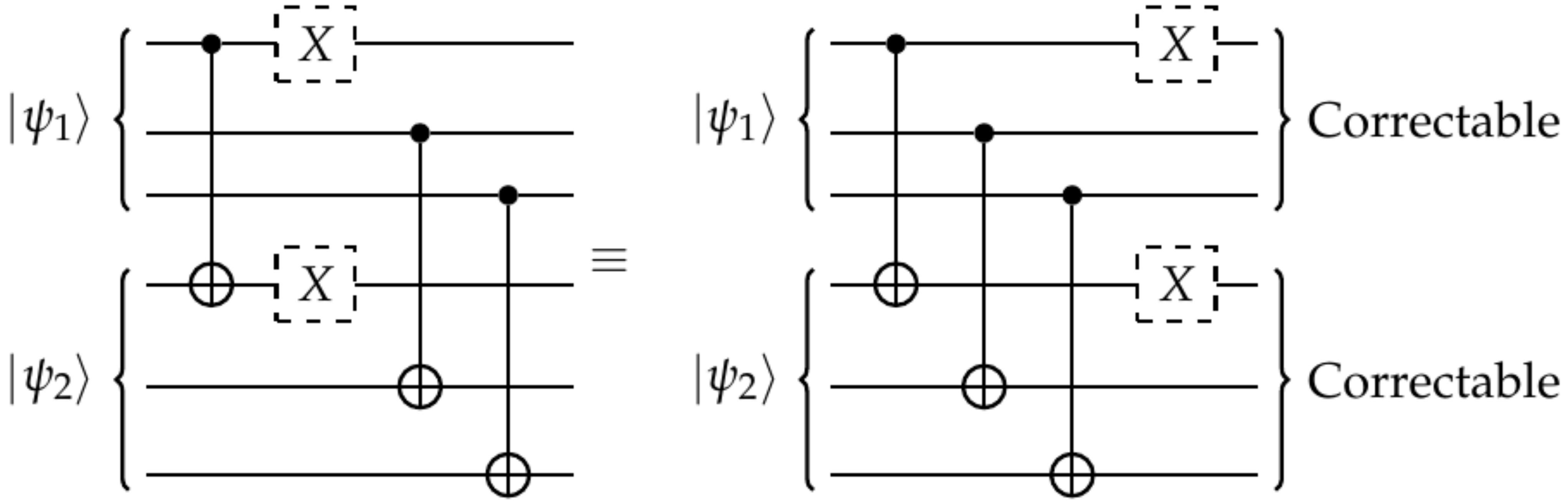
Transversal

Non-transversal CNOTs



- When physical CNOT fail with probability p, non-transversal logical CNOT fail with probability $\mathcal{O}(p)$.

Transversal CNOTs



- When physical CNOT fail with probability p, transversal logical CNOT fail with probability $\mathcal{O}(p^2)$.

Other Transversal Operations

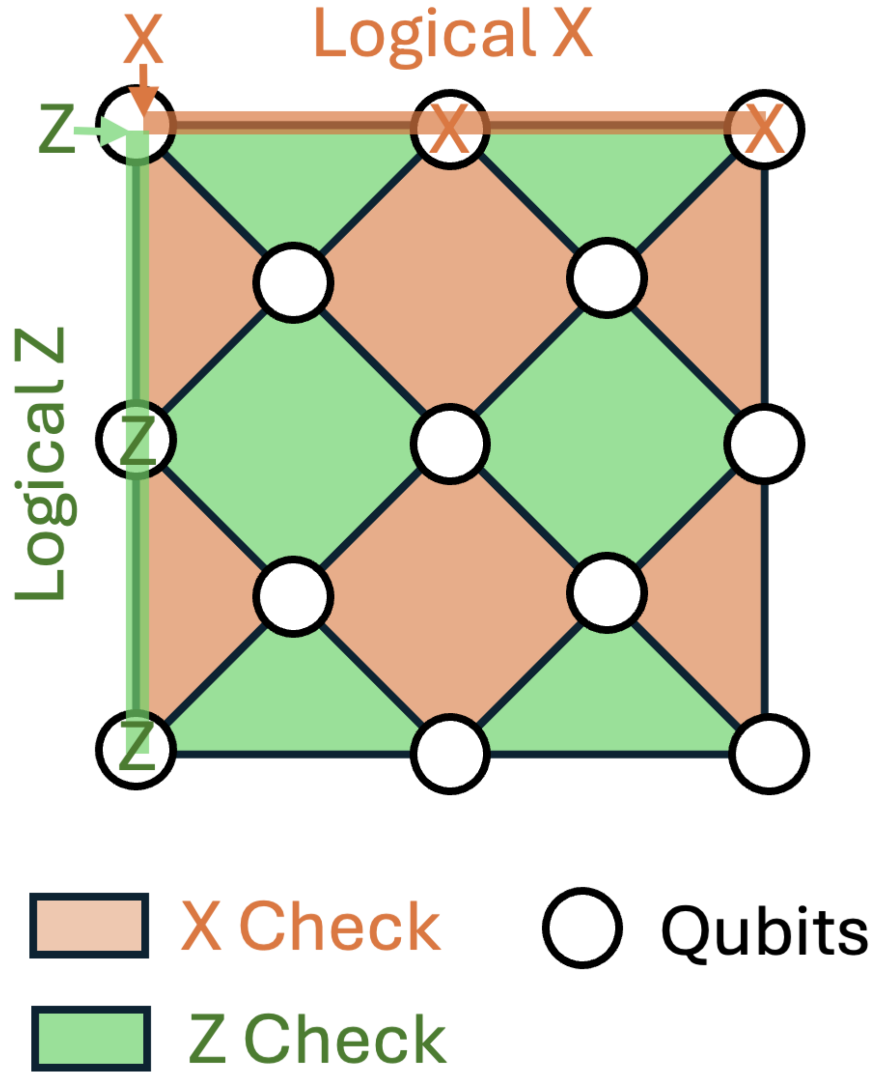
- Logical Pauli: Multi-qubit physical Pauli operators, which can be implemented using one layer of single-qubit Pauli in parallel.
- Logical Clifford: CNOTs are transversal for all CSS codes. Some CSS code like the $[[7,1,3]]$ Steane Code has transversal H and transversal $S = e^{-i\frac{\pi}{4}Z}$.
- $T = e^{-i\frac{\pi}{8}Z}$. Code with transversal Clifford don’t have transversal T (like the Steane Code), vice versa.
- Eastin-Knilll theorem: no QEC codes can implement all logical gates transversally.

Surface Code

Topological Code

- Topological Code
  - Local stabiliser checks for the underlying qubit layout, enabling efficient implementation.
  - Non-local logical operations for robustness against local errors.
- Prominent example: surface codes, which can achieve different distance.

Surface Code


X
Logical X
Z
X
X
Logical Z
Z
Z
X Check
Z Check
Qubits