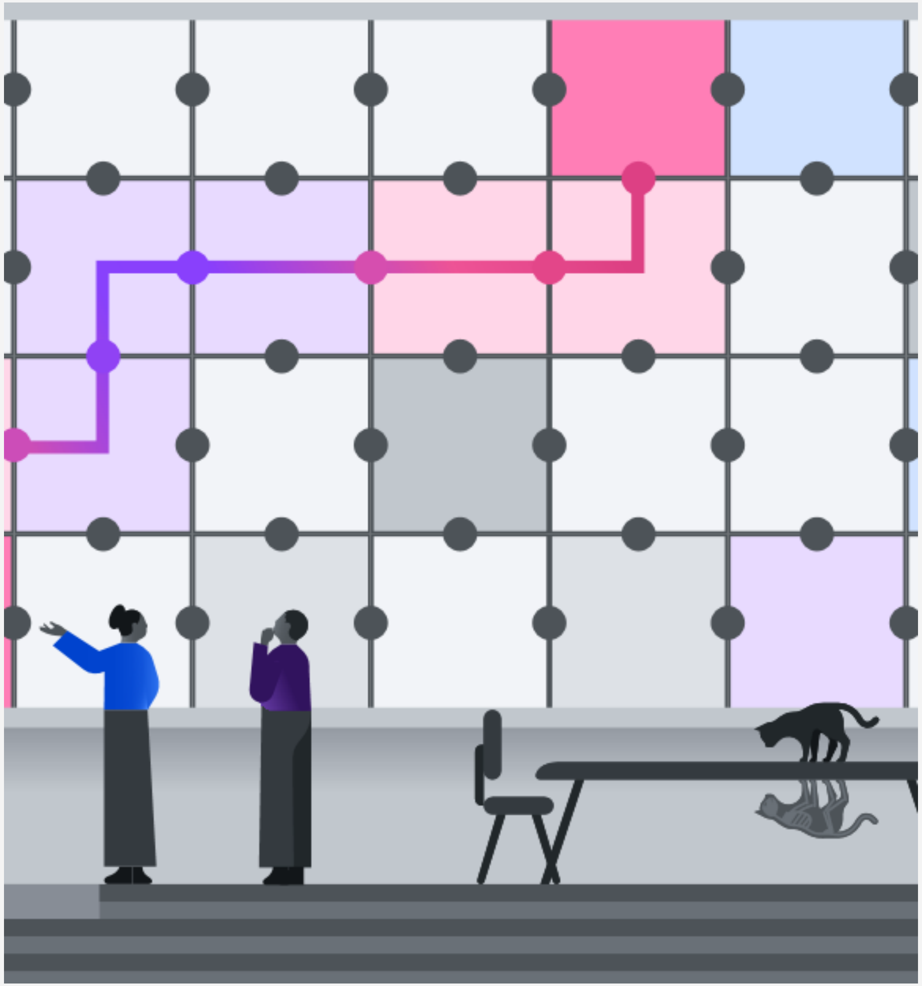
# Understanding quantum information and computation

By John Watrous



Lesson 15

## Quantum code constructions

# Classical linear codes

Let $\Sigma = \{0, 1\}$ denote the binary alphabet.

A *classical linear code* is a non-empty set of binary strings $\mathcal{C} \subseteq \Sigma^n$ with this property:

$$u, v \in \mathcal{C} \;\Rightarrow\; u \oplus v \in \mathcal{C}$$

Example: 3-bit repetition code

The 3-bit repetition code $\{000, 111\}$ is a classical linear code.

Example: $[7, 4, 3]$-Hamming code

The $[7, 4, 3]$-Hamming code is the classical linear code containing these strings:

$$\begin{array}{cccc} 0000000 & 1100001 & 1010010 & 0110011 \\ 0110100 & 1010101 & 1100110 & 0000111 \\ 1111000 & 0011001 & 0101010 & 1001011 \\ 1001100 & 0101101 & 0011110 & 1111111 \end{array}$$

# Classical linear codes

Let $\Sigma = \{0, 1\}$ denote the binary alphabet.

A *classical linear code* is a non-empty set of binary strings $\mathcal{C} \subseteq \Sigma^n$ with this property:

$$u, v \in \mathcal{C} \;\Rightarrow\; u \oplus v \in \mathcal{C}$$

Two natural ways to describe a classical linear code:

1. *Generators:* a minimal list of strings $u_1, \ldots, u_m \in \Sigma^n$ such that

$$\mathcal{C} = \{\alpha_1 u_1 \oplus \cdots \oplus \alpha_m u_m \,:\, \alpha_1, \ldots, \alpha_m \in \{0, 1\}\}$$

2. *Parity checks:* a minimal list of strings $v_1, \ldots, v_r \in \Sigma^n$ such that

$$\mathcal{C} = \{u \in \Sigma^n \,:\, u \cdot v_1 = \cdots = u \cdot v_r = 0\}$$

(where $u \cdot v$ is the binary dot product of $u$ and $v$).

# Classical linear codes

1. *Generators:* a minimal list of strings $u_1, \ldots, u_m \in \Sigma^n$ such that

$$\mathcal{C} = \{\alpha_1 u_1 \oplus \cdots \oplus \alpha_m u_m : \alpha_1, \ldots, \alpha_m \in \{0,1\}\}$$

2. *Parity checks:* a minimal list of strings $v_1, \ldots, v_r \in \Sigma^n$ such that

$$\mathcal{C} = \{u \in \Sigma^n : u \cdot v_1 = \cdots = u \cdot v_r = 0\}$$

**Example: 3-bit repetition code**

The 3-bit repetition code $\{000, 111\}$ is a classical linear code.

1. Generator: 111
2. Parity checks: 110, 011

# Classical linear codes

1. *Generators:* a minimal list of strings $u_1, \ldots, u_m \in \Sigma^n$ such that

$$\mathcal{C} = \{\alpha_1 u_1 \oplus \cdots \oplus \alpha_m u_m \,:\, \alpha_1, \ldots, \alpha_m \in \{0,1\}\}$$

2. *Parity checks:* a minimal list of strings $v_1, \ldots, v_r \in \Sigma^n$ such that

$$\mathcal{C} = \{u \in \Sigma^n \,:\, u \cdot v_1 = \cdots = u \cdot v_r = 0\}$$

Example: $[7,4,3]$-Hamming code

The $[7,4,3]$-Hamming code is the classical linear code containing these strings:

| | | | |
|---|---|---|---|
| 0000000 | 1100001 | 1010010 | 0110011 |
| 0110100 | 1010101 | 1100110 | 0000111 |
| 1111000 | 0011001 | 0101010 | 1001011 |
| 1001100 | 0101101 | 0011110 | 1111111 |

1. Generators: 0110100, 1010010, 1100001, 1111000
2. Parity checks: 1111000, 1100110, 1010101

# Classical linear codes

1. *Generators:* a minimal list of strings $u_1, \ldots, u_m \in \Sigma^n$ such that

$$\mathcal{C} = \{\alpha_1 u_1 \oplus \cdots \oplus \alpha_m u_m \,:\, \alpha_1, \ldots, \alpha_m \in \{0,1\}\}$$

2. *Parity checks:* a minimal list of strings $v_1, \ldots, v_r \in \Sigma^n$ such that

$$\mathcal{C} = \{u \in \Sigma^n \,:\, u \cdot v_1 = \cdots = u \cdot v_r = 0\}$$

Note: parity checks are equivalent to *stabilizer generators* containing only $Z$ and $\mathbb{1}$ Pauli matrices.

2

**Example: 3-bit repetition code**

The 3-bit repetition code $\{000, 111\}$ is a classical linear code.

1. Generator: 111
2. Parity checks: 110, 011

Equivalently, the strings in this code are standard basis states for the stabilizer code with stabilizer generators $Z\,Z\,\mathbb{1}$ and $\mathbb{1}\,Z\,Z$.

# CSS codes

Stabilizer generators containing only $Z$ and $\mathbb{1}$ Pauli matrices are equivalent to parity checks.

**Example: 3-bit repetition code**

The 3-bit repetition code $\{000, 111\}$ is a classical linear code.

Parity checks: 110, 011
Stabilizer generators: $Z\,Z\,\mathbb{1}$, $\mathbb{1}\,Z\,Z$

**Example: $[7,4,3]$-Hamming code**

The $[7,4,3]$-Hamming code is the classical linear code containing these strings:

| | | | |
|---|---|---|---|
| 0000000 | 1100001 | 1010010 | 0110011 |
| 0110100 | 1010101 | 1100110 | 0000111 |
| 1111000 | 0011001 | 0101010 | 1001011 |
| 1001100 | 0101101 | 0011110 | 1111111 |

Parity checks: 1111000, 1100110, 1010101
Stabilizer generators: $Z\,Z\,Z\,Z\,\mathbb{1}\,\mathbb{1}\,\mathbb{1}$, $Z\,Z\,\mathbb{1}\,\mathbb{1}\,Z\,Z\,\mathbb{1}$, $Z\,\mathbb{1}\,Z\,\mathbb{1}\,Z\,\mathbb{1}\,Z$

# CSS codes

Stabilizer generators containing only $Z$ and $\mathbb{1}$ Pauli matrices are equivalent to parity checks. These are called *$Z$ stabilizer generators.*

Stabilizer generators containing only $X$ and $\mathbb{1}$ Pauli matrices are also equivalent to parity checks — for the plus/minus basis $\{|+\rangle, |-\rangle\}$.

Example: $[7, 4, 3]$-Hamming code

$$\begin{array}{cccc} 0000000 & 1100001 & 1010010 & 0110011 \\ 0110100 & 1010101 & 1100110 & 0000111 \\ 1111000 & 0011001 & 0101010 & 1001011 \\ 1001100 & 0101101 & 0011110 & 1111111 \end{array}$$

Parity checks: 1111000, 1100110, 1010101

The stabilizer generators $X\,X\,X\,X\,\mathbb{1}\,\mathbb{1}\,\mathbb{1}$, $X\,X\,\mathbb{1}\,\mathbb{1}\,X\,X\,\mathbb{1}$, $X\,\mathbb{1}\,X\,\mathbb{1}\,X\,\mathbb{1}\,X$ define a stabilizer code that includes these states:

$$\begin{array}{cccc} |{+}{+}{+}{+}{+}{+}{+}\rangle & |{-}{-}{+}{+}{+}{+}{-}\rangle & |{-}{+}{-}{+}{+}{-}{+}\rangle & |{+}{-}{-}{+}{+}{-}{-}\rangle \\ |{+}{-}{-}{+}{-}{+}{+}\rangle & |{-}{+}{-}{+}{-}{+}{-}\rangle & |{-}{-}{+}{+}{-}{-}{+}\rangle & |{+}{+}{+}{+}{-}{-}{-}\rangle \\ |{-}{-}{-}{-}{+}{+}{+}\rangle & |{+}{+}{-}{-}{+}{+}{-}\rangle & |{+}{-}{+}{-}{+}{-}{+}\rangle & |{-}{+}{+}{-}{+}{-}{-}\rangle \\ |{-}{+}{+}{-}{-}{+}{+}\rangle & |{+}{-}{+}{-}{-}{+}{-}\rangle & |{+}{+}{-}{-}{-}{-}{+}\rangle & |{-}{-}{-}{-}{-}{-}{-}\rangle \end{array}$$

# CSS codes

Stabilizer generators containing only $Z$ and $\mathbb{1}$ Pauli matrices are equivalent to parity checks. These are called *$Z$ stabilizer generators.*

Stabilizer generators containing only $X$ and $\mathbb{1}$ Pauli matrices are also equivalent to parity checks — for the plus/minus basis $\{|+\rangle, |-\rangle\}$. These are called *$X$ stabilizer generators.*

### Definition: CSS codes

Stabilizer codes that can be expressed using only $Z$ stabilizer generators and $X$ stabilizer generators are called *CSS codes.*

1

### Example: e-bit stabilizer code

$$\begin{array}{c} Z\,Z \\ X\,X \end{array}$$

The code space is the one-dimensional space spanned by

$$|\phi^+\rangle = \frac{|0\rangle|0\rangle + |1\rangle|1\rangle}{\sqrt{2}} = \frac{|+\rangle|+\rangle + |-\rangle|-\rangle}{\sqrt{2}}$$

# CSS codes

Stabilizer generators containing only $Z$ and $\mathbb{1}$ Pauli matrices are equivalent to parity checks. These are called *$Z$ stabilizer generators.*

Stabilizer generators containing only $X$ and $\mathbb{1}$ Pauli matrices are also equivalent to parity checks — for the plus/minus basis $\{|+\rangle, |-\rangle\}$. These are called *$X$ stabilizer generators.*

**Definition: CSS codes**

Stabilizer codes that can be expressed using only $Z$ stabilizer generators and $X$ stabilizer generators are called *CSS codes.*

**Example: 7-qubit Steane code**

$$
\begin{array}{ccccccc}
Z & Z & Z & Z & \mathbb{1} & \mathbb{1} & \mathbb{1}\\
Z & Z & \mathbb{1} & \mathbb{1} & Z & Z & \mathbb{1}\\
Z & \mathbb{1} & Z & \mathbb{1} & Z & \mathbb{1} & Z\\
X & X & X & X & \mathbb{1} & \mathbb{1} & \mathbb{1}\\
X & X & \mathbb{1} & \mathbb{1} & X & X & \mathbb{1}\\
X & \mathbb{1} & X & \mathbb{1} & X & \mathbb{1} & X
\end{array}
$$

**Example: 9-qubit Shor code**

$$
\begin{array}{ccccccccc}
Z & Z & \mathbb{1} & \mathbb{1} & \mathbb{1} & \mathbb{1} & \mathbb{1} & \mathbb{1} & \mathbb{1}\\
\mathbb{1} & Z & Z & \mathbb{1} & \mathbb{1} & \mathbb{1} & \mathbb{1} & \mathbb{1} & \mathbb{1}\\
\mathbb{1} & \mathbb{1} & \mathbb{1} & Z & Z & \mathbb{1} & \mathbb{1} & \mathbb{1} & \mathbb{1}\\
\mathbb{1} & \mathbb{1} & \mathbb{1} & \mathbb{1} & Z & Z & \mathbb{1} & \mathbb{1} & \mathbb{1}\\
\mathbb{1} & \mathbb{1} & \mathbb{1} & \mathbb{1} & \mathbb{1} & \mathbb{1} & Z & Z & \mathbb{1}\\
\mathbb{1} & \mathbb{1} & \mathbb{1} & \mathbb{1} & \mathbb{1} & \mathbb{1} & \mathbb{1} & Z & Z\\
X & X & X & X & X & X & \mathbb{1} & \mathbb{1} & \mathbb{1}\\
\mathbb{1} & \mathbb{1} & \mathbb{1} & X & X & X & X & X & X
\end{array}
$$

# Error detection and correction

Consider a CSS code.

- The $Z$ stabilizer generators detect $X$ errors but are oblivious to $Z$ errors (and corrections).
- The $X$ stabilizer generators detect $Z$ errors but are oblivious to $X$ errors (and corrections).

Suppose the following:

- The $Z$ stabilizer generators allow for the correction of up to $j$ $X$ errors.
- The $X$ stabilizer generators allow for the correction of up to $k$ $Z$ errors.

Then the CSS code allows for the correction of *any error* on up to $\min\{j, k\}$ qubits — we can simply detect and correct $X$ errors and $Z$ errors on this many qubits separately.

7-qubit Steane code

$$
\begin{array}{ccccccc}
Z & Z & Z & Z & \mathbb{1} & \mathbb{1} & \mathbb{1} \\
Z & Z & \mathbb{1} & \mathbb{1} & Z & Z & \mathbb{1} \\
Z & \mathbb{1} & Z & \mathbb{1} & Z & \mathbb{1} & Z \\
X & X & X & X & \mathbb{1} & \mathbb{1} & \mathbb{1} \\
X & X & \mathbb{1} & \mathbb{1} & X & X & \mathbb{1} \\
X & \mathbb{1} & X & \mathbb{1} & X & \mathbb{1} & X
\end{array}
$$

# Code spaces of CSS codes

Consider a CSS code on $n$ qubits.

Let $z_1, \ldots, z_s \in \Sigma^n$ be parity checks corresponding to the $Z$ stabilizer generators.

$$\mathcal{C}_Z = \{u \in \Sigma^n : u \cdot z_1 = \cdots = u \cdot z_s = 0\}$$
$$\mathcal{D}_Z = \{\alpha_1 z_1 \oplus \cdots \oplus \alpha_s z_s : \alpha_1, \ldots, \alpha_s \in \{0,1\}\}$$

Let $x_1, \ldots, x_t \in \Sigma^n$ be parity checks corresponding to the $X$ stabilizer generators.

$$\mathcal{C}_X = \{u \in \Sigma^n : u \cdot x_1 = \cdots = u \cdot x_t = 0\}$$
$$\mathcal{D}_X = \{\alpha_1 x_1 \oplus \cdots \oplus \alpha_t x_t : \alpha_1, \ldots, \alpha_t \in \{0,1\}\}$$

# Code spaces of CSS codes

Let $z_1, \ldots, z_s \in \Sigma^n$ be parity checks corresponding to the $Z$ stabilizer generators.

$$\mathcal{C}_Z = \{u \in \Sigma^n : u \cdot z_1 = \cdots = u \cdot z_s = 0\}$$
$$\mathcal{D}_Z = \{\alpha_1 z_1 \oplus \cdots \oplus \alpha_s z_s : \alpha_1, \ldots, \alpha_s \in \{0,1\}\}$$

Let $x_1, \ldots, x_t \in \Sigma^n$ be parity checks corresponding to the $X$ stabilizer generators.

$$\mathcal{C}_X = \{u \in \Sigma^n : u \cdot x_1 = \cdots = u \cdot x_t = 0\}$$
$$\mathcal{D}_X = \{\alpha_1 x_1 \oplus \cdots \oplus \alpha_t x_t : \alpha_1, \ldots, \alpha_t \in \{0,1\}\}$$

The code space of the CSS code is spanned by vectors of either of these forms:

$$|u \oplus \mathcal{D}_X\rangle = \frac{1}{\sqrt{2^t}} \sum_{v \in \mathcal{D}_X} |u \oplus v\rangle \qquad (\text{for } u \in \mathcal{C}_Z)$$
$$H^{\otimes n}|u \oplus \mathcal{D}_Z\rangle = \frac{1}{\sqrt{2^s}} \sum_{v \in \mathcal{D}_Z} H^{\otimes n}|u \oplus v\rangle \qquad (\text{for } u \in \mathcal{C}_X)$$

# Code spaces of CSS codes

The code space of the CSS code is spanned by vectors of either of these forms:

$$|u \oplus \mathcal{D}_X\rangle = \frac{1}{\sqrt{2^t}} \sum_{v \in \mathcal{D}_X} |u \oplus v\rangle \qquad (\text{for } u \in \mathcal{C}_Z)$$

$$H^{\otimes n}|u \oplus \mathcal{D}_Z\rangle = \frac{1}{\sqrt{2^s}} \sum_{v \in \mathcal{D}_Z} H^{\otimes n}|u \oplus v\rangle \qquad (\text{for } u \in \mathcal{C}_X)$$

Example: 7-qubit Steane code

$$\begin{matrix} Z & Z & Z & Z & \mathbb{1} & \mathbb{1} & \mathbb{1} \\ Z & Z & \mathbb{1} & \mathbb{1} & Z & Z & \mathbb{1} \\ Z & \mathbb{1} & Z & \mathbb{1} & Z & \mathbb{1} & Z \\ X & X & X & X & \mathbb{1} & \mathbb{1} & \mathbb{1} \\ X & X & \mathbb{1} & \mathbb{1} & X & X & \mathbb{1} \\ X & \mathbb{1} & X & \mathbb{1} & X & \mathbb{1} & X \end{matrix}$$

We could encode $|0\rangle$ and $|1\rangle$ as follows:

$$\begin{aligned} |0\rangle \mapsto\ & |0000000\rangle + |0011110\rangle + |0101101\rangle \\ & + |0110011\rangle + |1001011\rangle + |1010101\rangle \\ & + |1100110\rangle + |1111000\rangle \end{aligned}$$

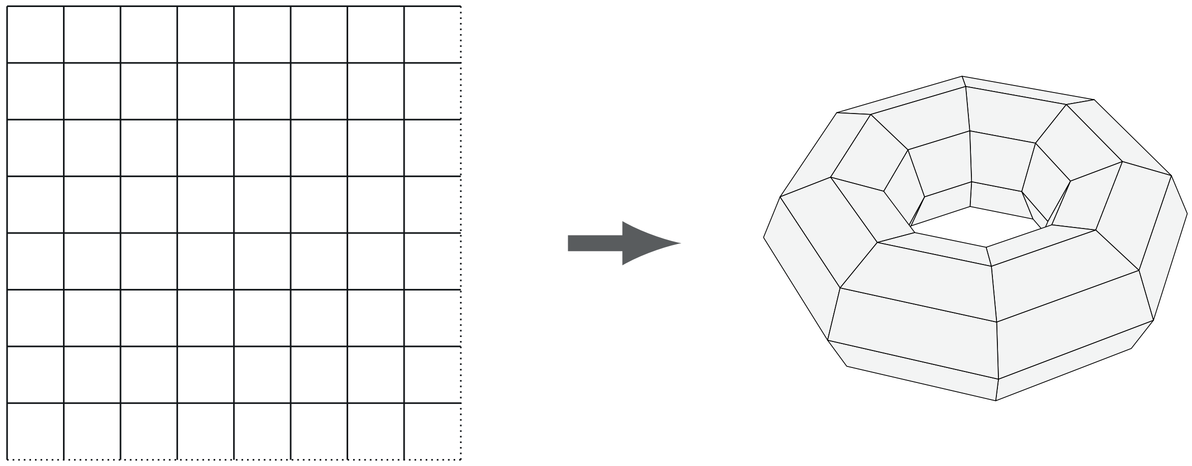
$$\begin{aligned} |1\rangle \mapsto\ & |0000111\rangle + |0011001\rangle + |0101010\rangle \\ & + |0110100\rangle + |1001100\rangle + |1010010\rangle \\ & + |1100001\rangle + |1111111\rangle \end{aligned}$$

# Toric code

The *toric code* is an example of a quantum error correcting code (actually a family of codes) with a few key properties.

- Low weight stabilizer generators
- Geometric locality
- Large distance

Let $L \geq 2$ be a positive integer and consider an $L \times L$ lattice with periodic boundaries.



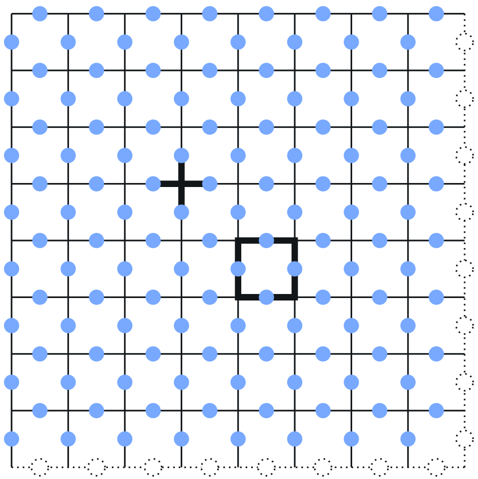
$L = 8$

# Toric code

The *toric code* is an example of a quantum error correcting code (actually a family of codes) with a few key properties.

- Low weight stabilizer generators
- Geometric locality
- Large distance

Let $L \geq 2$ be a positive integer and consider an $L \times L$ lattice with periodic boundaries.

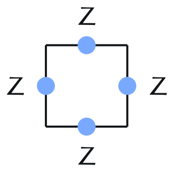


$L = 8$

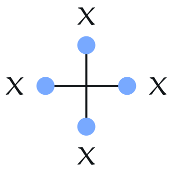
Qubits are placed on the *edges* of the lattice

$\Rightarrow n = 2L^2$ qubits

There are two types of stabilizer generators:




$Z$ stabilizer generators (tiles)



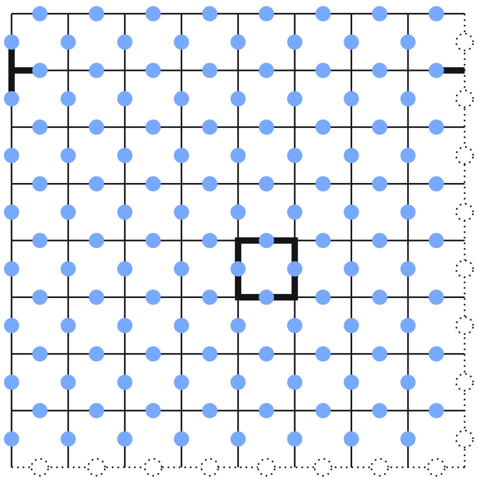

$X$ stabilizer generators (vertices)

# Toric code

The *toric code* is an example of a quantum error correcting code (actually a family of codes) with a few key properties.

- Low weight stabilizer generators
- Geometric locality
- Large distance

Let $L \geq 2$ be a positive integer and consider an $L \times L$ lattice with periodic boundaries.

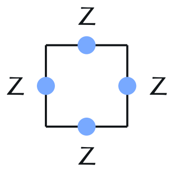


$L = 8$

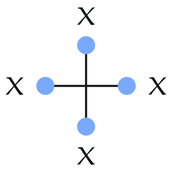
Qubits are placed on the *edges* of the lattice

$\Rightarrow n = 2L^2$ qubits

There are two types of stabilizer generators:




$Z$ stabilizer generators (tiles)



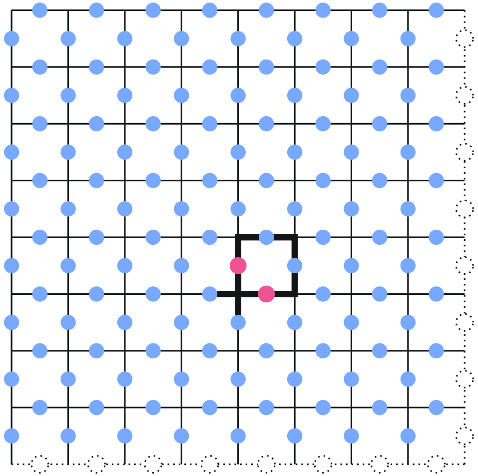

$X$ stabilizer generators (vertices)

# Toric code

The *toric code* is an example of a quantum error correcting code (actually a family of codes) with a few key properties.

- Low weight stabilizer generators
- Geometric locality
- Large distance

Let $L \geq 2$ be a positive integer and consider an $L \times L$ lattice with periodic boundaries.

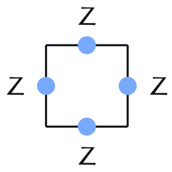


$L = 8$

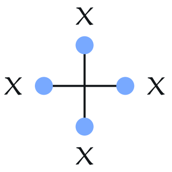
Qubits are placed on the *edges* of the lattice

$\Rightarrow n = 2L^2$ qubits

There are two types of stabilizer generators:

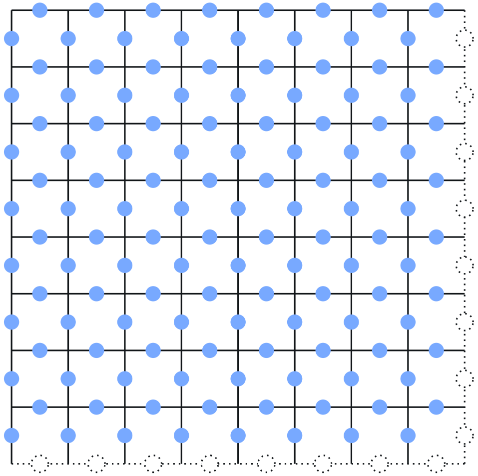



$Z$ stabilizer generators (tiles)



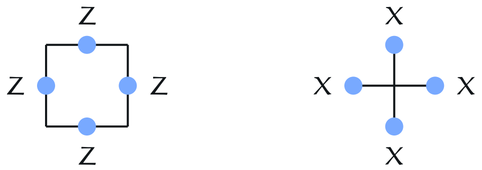

$X$ stabilizer generators (vertices)

# Toric code



Qubits are placed on the *edges* of the lattice
$\Rightarrow n = 2L^2$ qubits

There are two types of stabilizer generators:




$Z$ stabilizer generators (tiles)

$X$ stabilizer generators (vertices)

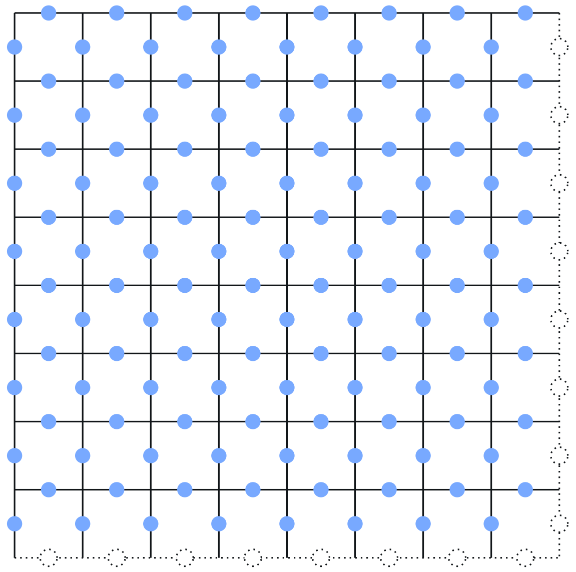
The product of all of the $Z$ stabilizer generators is the identity — but removing any one leaves an independent set. Similar for the $X$ stabilizer generators.

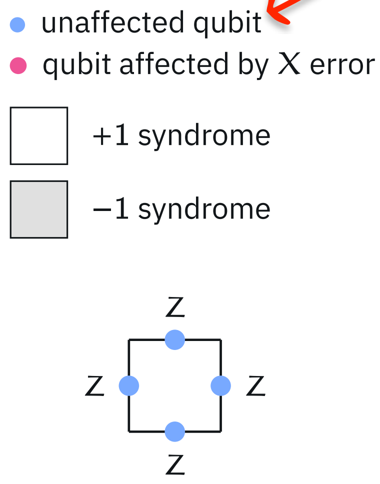
This leaves $L^2 - 1$ stabilizer generators of each of the two types.

The toric code (for this choice of $L$) therefore encodes $2L^2 - 2(L^2 - 1) = 2$ logical qubits into $2L^2$ physical qubits.

# Errors and syndromes

The toric code is a CSS code, which allows us to consider $X$ errors and $Z$ errors separately. Let us focus on $X$ *errors* — $Z$ errors work similarly by symmetry.



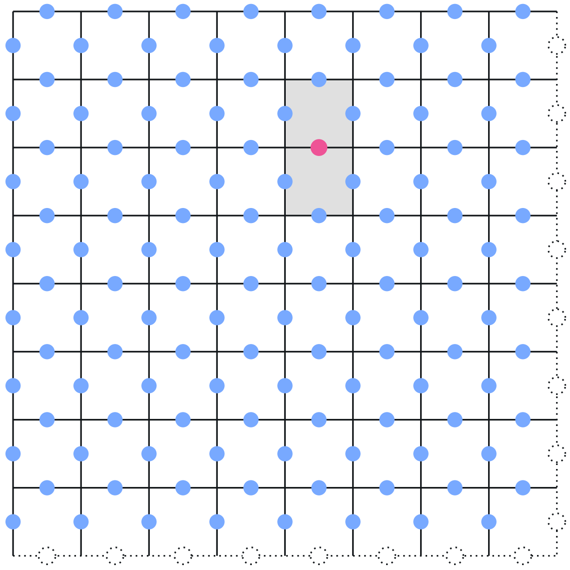



$Z$ stabilizer generator

1 X Error

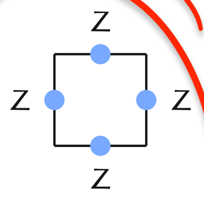
# Errors and syndromes

The toric code is a CSS code, which allows us to consider $X$ errors and $Z$ errors separately. Let us focus on $X$ *errors* — $Z$ errors work similarly by symmetry.



- unaffected qubit
- qubit affected by $X$ error
- +1 syndrome
- −1 syndrome

No error
Even Error

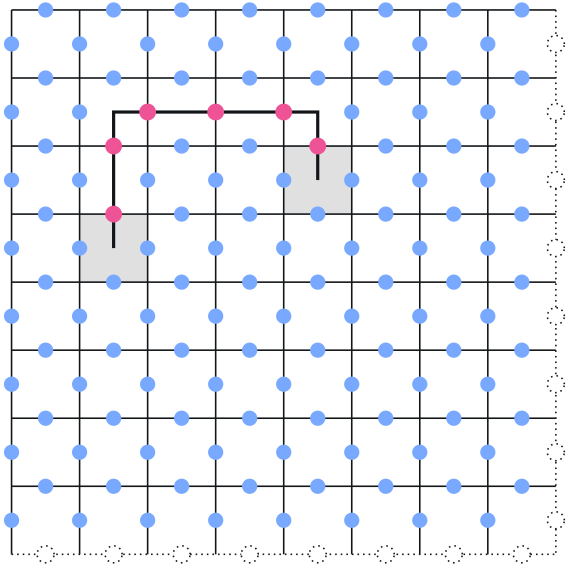



$Z$ stabilizer generator

1+X Error

# Errors and syndromes

The toric code is a CSS code, which allows us to consider $X$ errors and $Z$ errors separately. Let us focus on *$X$ errors* — $Z$ errors work similarly by symmetry.



- unaffected qubit
- qubit affected by $X$ error

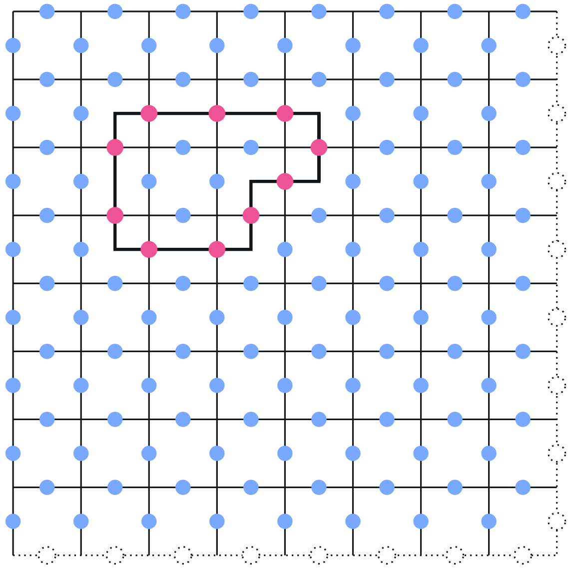
+1 syndrome

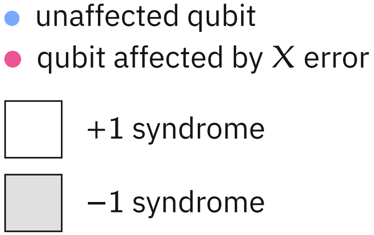
−1 syndrome

Chains of adjacent $X$ errors cause −1 syndrome outcomes at the *endpoints.*

# Errors and syndromes

The toric code is a CSS code, which allows us to consider $X$ errors and $Z$ errors separately. Let us focus on *$X$ errors* — $Z$ errors work similarly by symmetry.



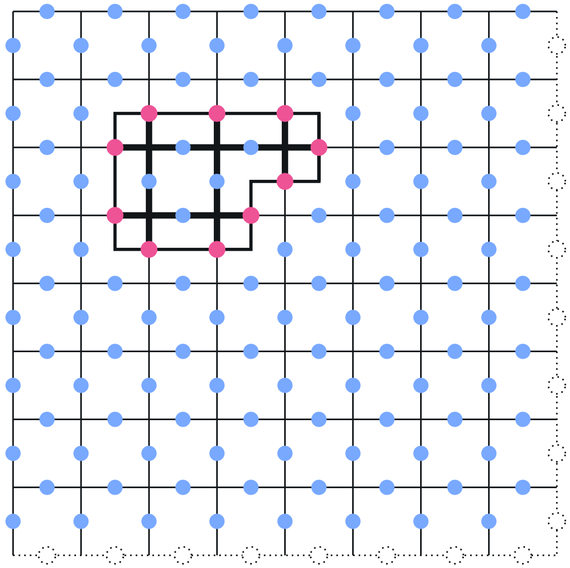



Chains of adjacent $X$ errors cause $-1$ syndrome outcomes at the *endpoints.*

*Closed loops* of adjacent $X$ errors are undetected by the code.

# Errors and syndromes

The toric code is a CSS code, which allows us to consider $X$ errors and $Z$ errors separately. Let us focus on *$X$ errors* — $Z$ errors work similarly by symmetry.



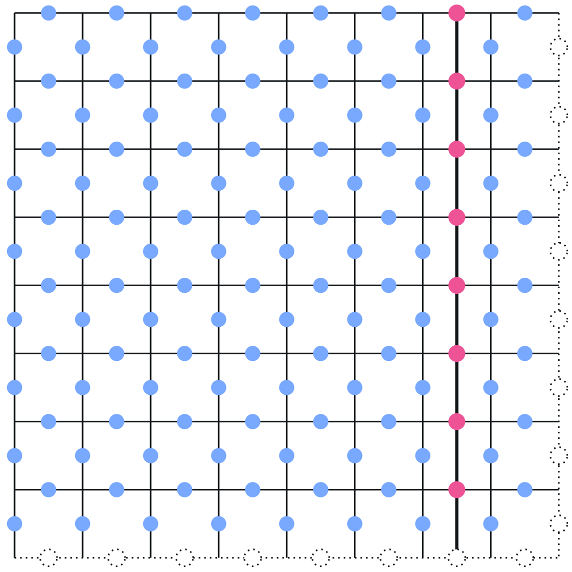
Chains of adjacent $X$ errors cause $-1$ syndrome outcomes at the *endpoints.*

*Closed loops* of adjacent $X$ errors are undetected by the code.

- Loops crossing every line an even number of times are products of $X$ stabilizer generators.

# Errors and syndromes

The toric code is a CSS code, which allows us to consider $X$ errors and $Z$ errors separately. Let us focus on *$X$ errors* — $Z$ errors work similarly by symmetry.



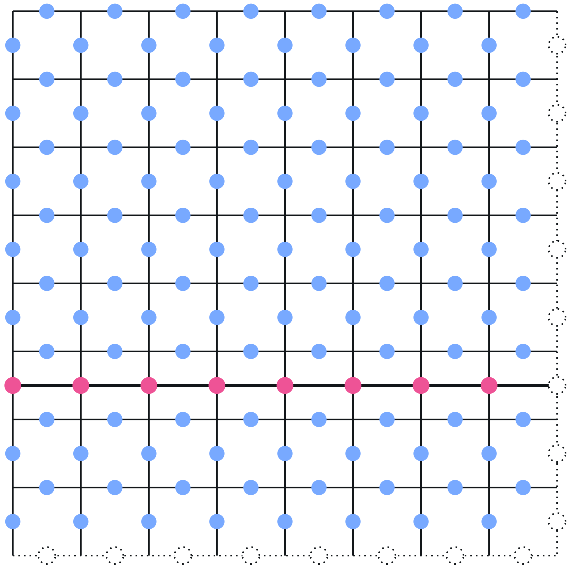
Chains of adjacent $X$ errors cause $-1$ syndrome outcomes at the *endpoints.*

*Closed loops* of adjacent $X$ errors are undetected by the code.

- Loops crossing every line an even number of times are products of $X$ stabilizer generators.
- Loops crossing any line an odd number of times are nontrivial errors that are *undetected* by the code.

# Errors and syndromes

The toric code is a CSS code, which allows us to consider $X$ errors and $Z$ errors separately. Let us focus on *$X$ errors* — $Z$ errors work similarly by symmetry.



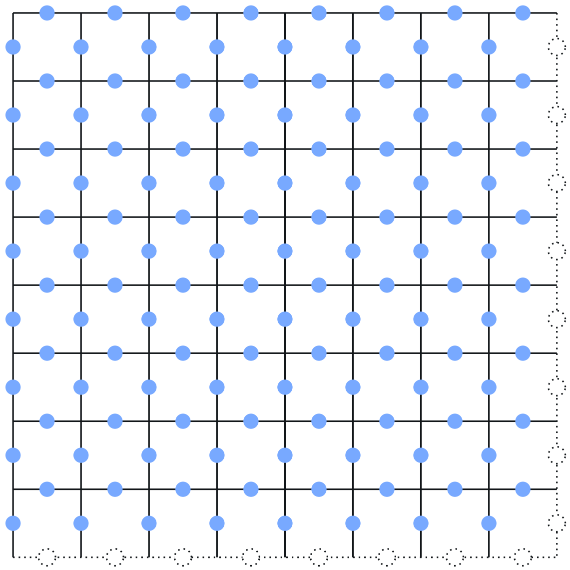
Chains of adjacent $X$ errors cause $-1$ syndrome outcomes at the *endpoints.*

*Closed loops* of adjacent $X$ errors are undetected by the code.

- Loops crossing every line an even number of times are products of $X$ stabilizer generators.
- Loops crossing any line an odd number of times are nontrivial errors that are *undetected* by the code.

# Errors and syndromes

The toric code is a CSS code, which allows us to consider $X$ errors and $Z$ errors separately. Let us focus on *$X$ errors* — $Z$ errors work similarly by symmetry.



Chains of adjacent $X$ errors cause $-1$ syndrome outcomes at the *endpoints.*

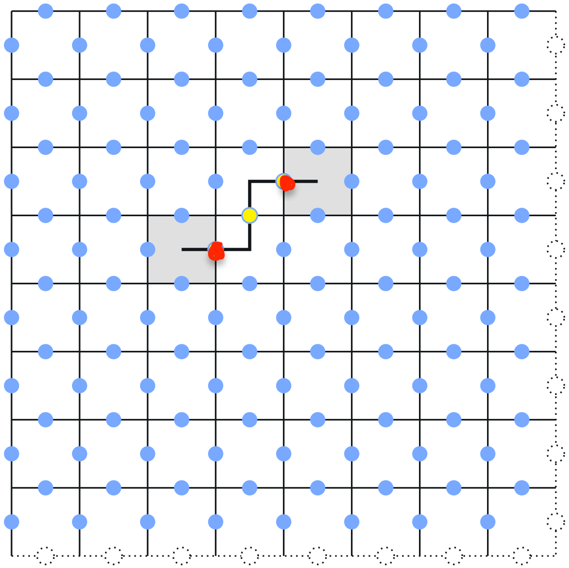
*Closed loops* of adjacent $X$ errors are undetected by the code.

- Loops crossing every line an even number of times are products of $X$ stabilizer generators.
- Loops crossing any line an odd number of times are nontrivial errors that are *undetected* by the code.

The minimum weight of a nontrivial, undetectable error is $L$.

The toric code is therefore a $[[2L^2, 2, L]]$ stabilizer code.

# Correcting errors



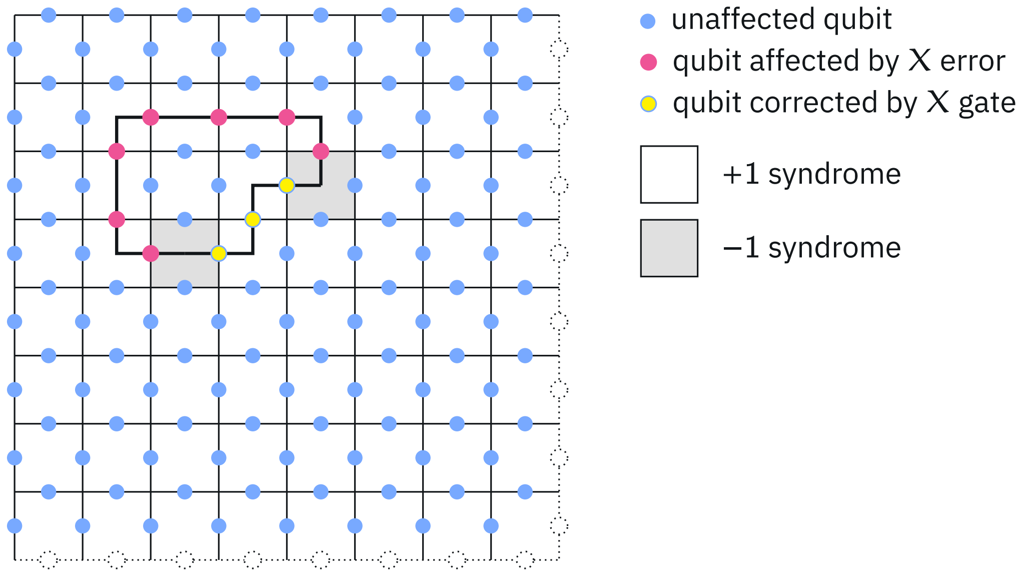
- unaffected qubit
- qubit affected by $X$ error
- qubit corrected by $X$ gate

+1 syndrome

−1 syndrome → endpoints

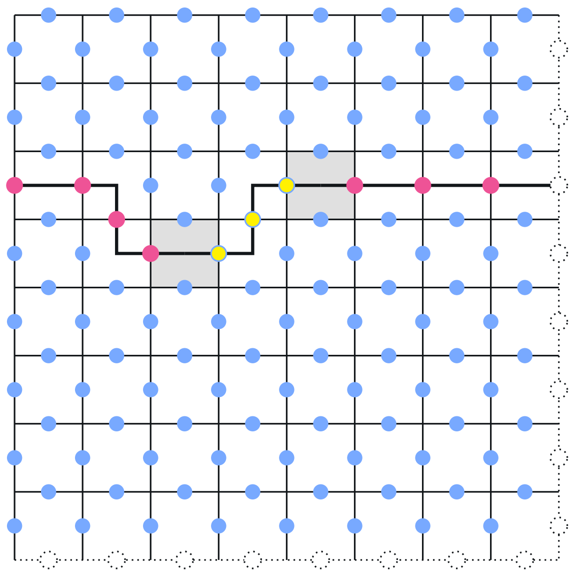
We can attempt to correct errors by pairing together −1 syndrome measurements with *shortest paths* of corrections.

# Correcting errors




We can attempt to correct errors by pairing together $-1$ syndrome measurements with *shortest paths* of corrections.

# Correcting errors



unaffected qubit

qubit affected by $X$ error

qubit corrected by $X$ gate

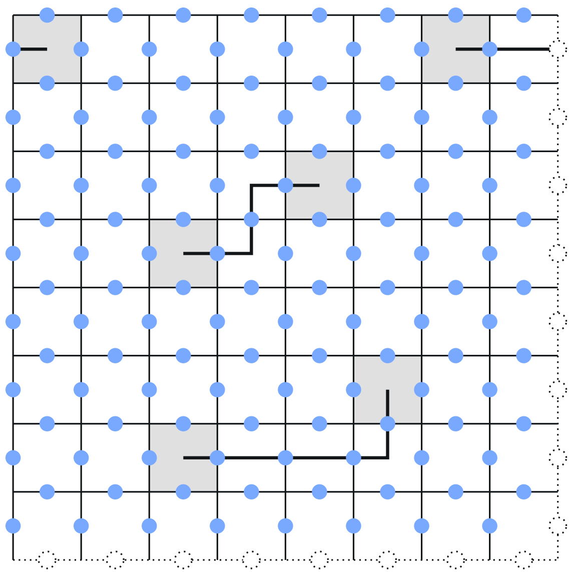
$+1$ syndrome

$-1$ syndrome

This strategy corrects low-weight errors, but may not work for high-weight errors.

We can attempt to correct errors by pairing together $-1$ syndrome measurements with *shortest paths* of corrections.

# Correcting errors



- unaffected qubit
- qubit affected by $X$ error
- qubit corrected by $X$ gate

+1 syndrome

−1 syndrome

This strategy corrects low-weight errors, but may not work for high-weight errors.

Lowest-weight pairings can be found by efficient classical algorithms.

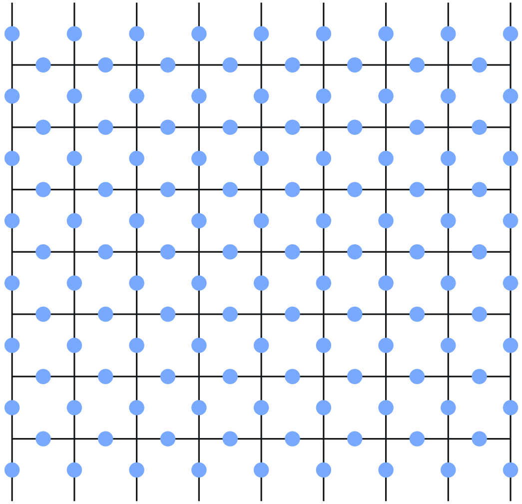
We can attempt to correct errors by pairing together $-1$ syndrome measurements with *shortest paths* of corrections.

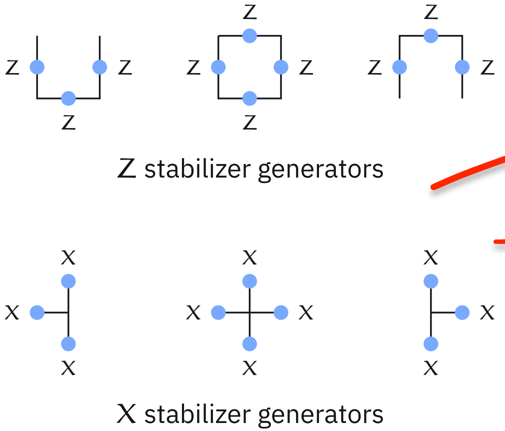
Depending on the noise model, lowest-weight pairings may not correct the most likely errors — but the method works well for simple noise models.

# Other code families

## Surface codes

The toric code doesn't actually require periodic boundaries — it can be defined on a *two-dimensional surface* instead.

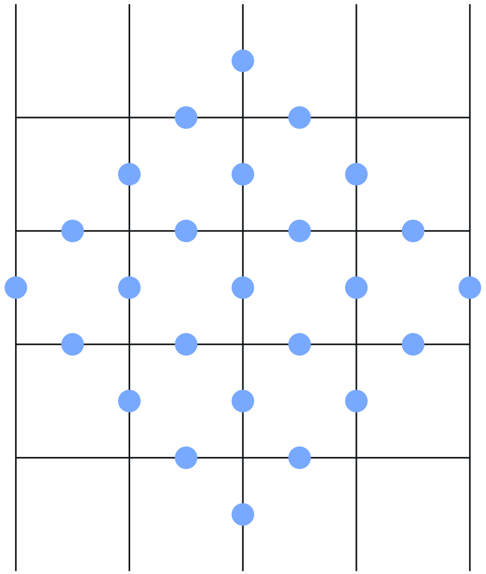




# Other code families

## Surface codes

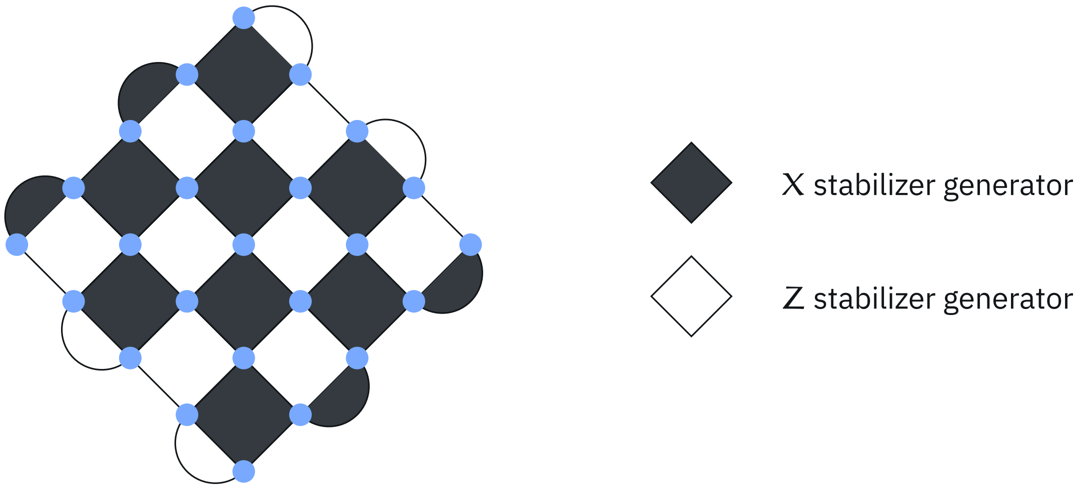
The toric code doesn't actually require periodic boundaries — it can be defined on a *two-dimensional surface* instead.



# Other code families

## Surface codes

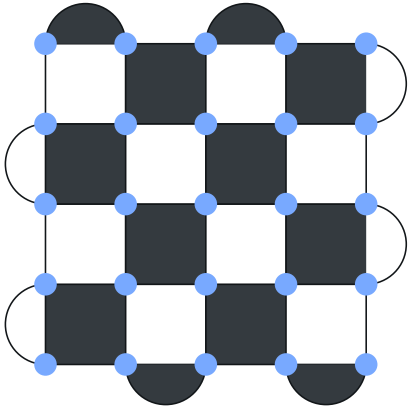
The toric code doesn't actually require periodic boundaries — it can be defined on a *two-dimensional surface* instead.

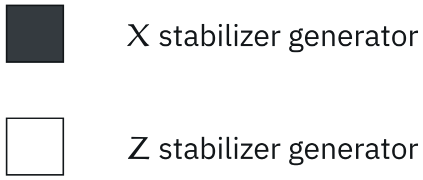


# Other code families

## Surface codes

The toric code doesn't actually require periodic boundaries — it can be defined on a *two-dimensional surface* instead.



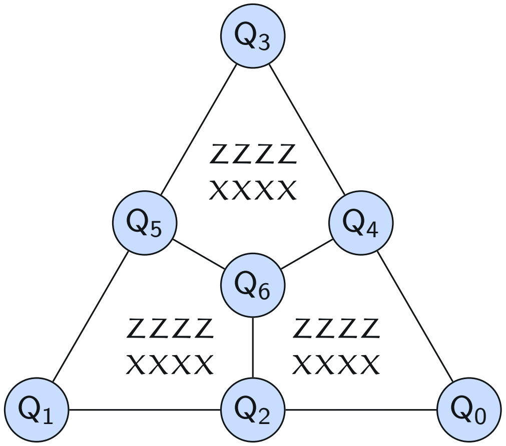


# Other code families

## Color codes

Consider the 7-qubit Steane code for qubits $(Q_6, Q_5, Q_4, Q_3, Q_2, Q_1, Q_0)$.

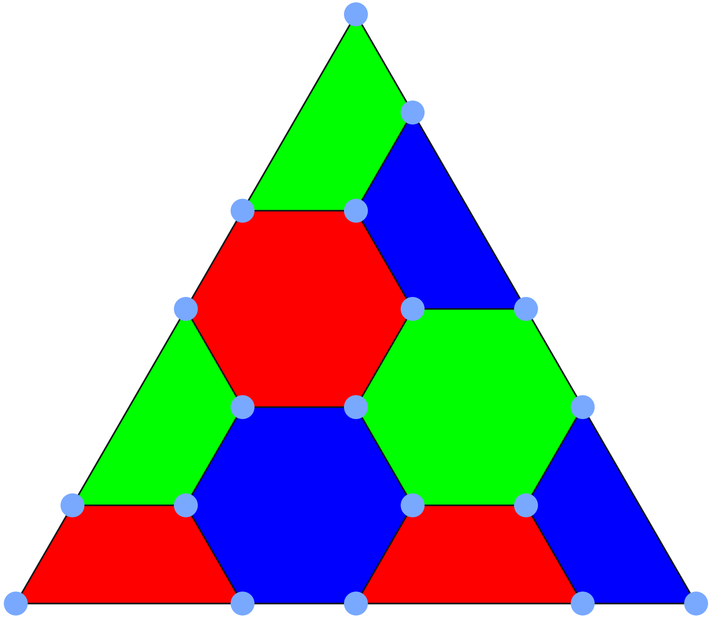
$$\begin{array}{ccccccc}
Z & Z & Z & Z & \mathbb{1} & \mathbb{1} & \mathbb{1} \\
Z & Z & \mathbb{1} & \mathbb{1} & Z & Z & \mathbb{1} \\
Z & \mathbb{1} & Z & \mathbb{1} & Z & \mathbb{1} & Z \\
X & X & X & X & \mathbb{1} & \mathbb{1} & \mathbb{1} \\
X & X & \mathbb{1} & \mathbb{1} & X & X & \mathbb{1} \\
X & \mathbb{1} & X & \mathbb{1} & X & \mathbb{1} & X
\end{array}$$




Color codes generalize this basic pattern to other graphs and lattices.

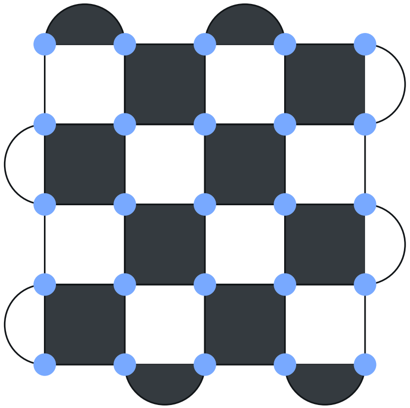
# Other code families

Color codes

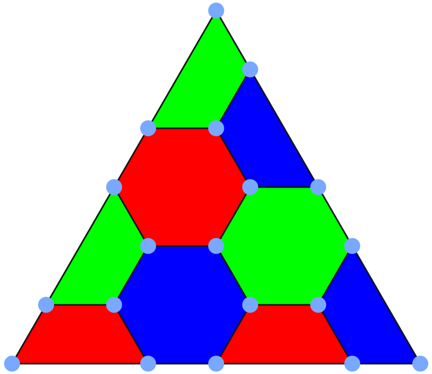


# Other code families

Surface codes



Color codes



Many other constructions for quantum error correcting codes are known.

Example: Gross code

The *gross code* is a recently discovered $[[144, 12, 12]]$ stabilizer code.

It requires an additional 144 qubits for performing syndrome measurements and has a biplanar embedding.