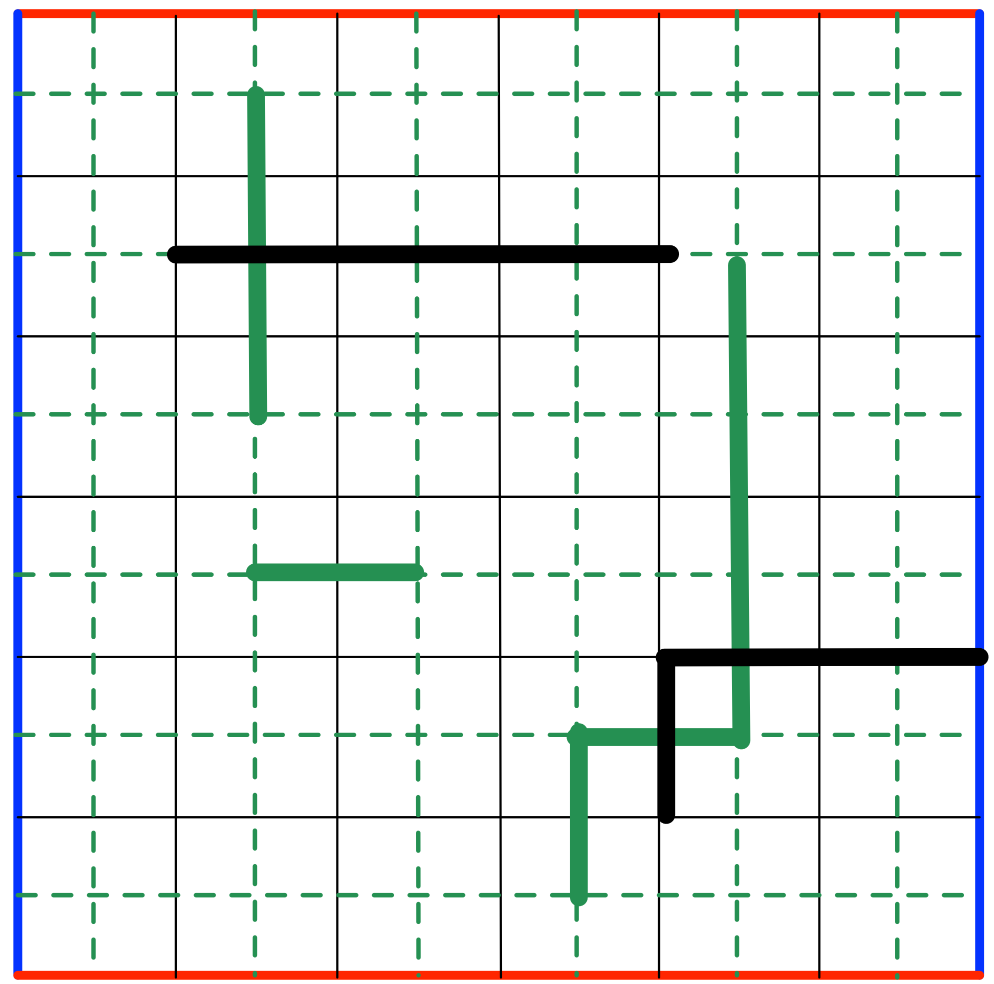
Decoding Homology



A lexicon for the uninitiated

Tutorial Lecture

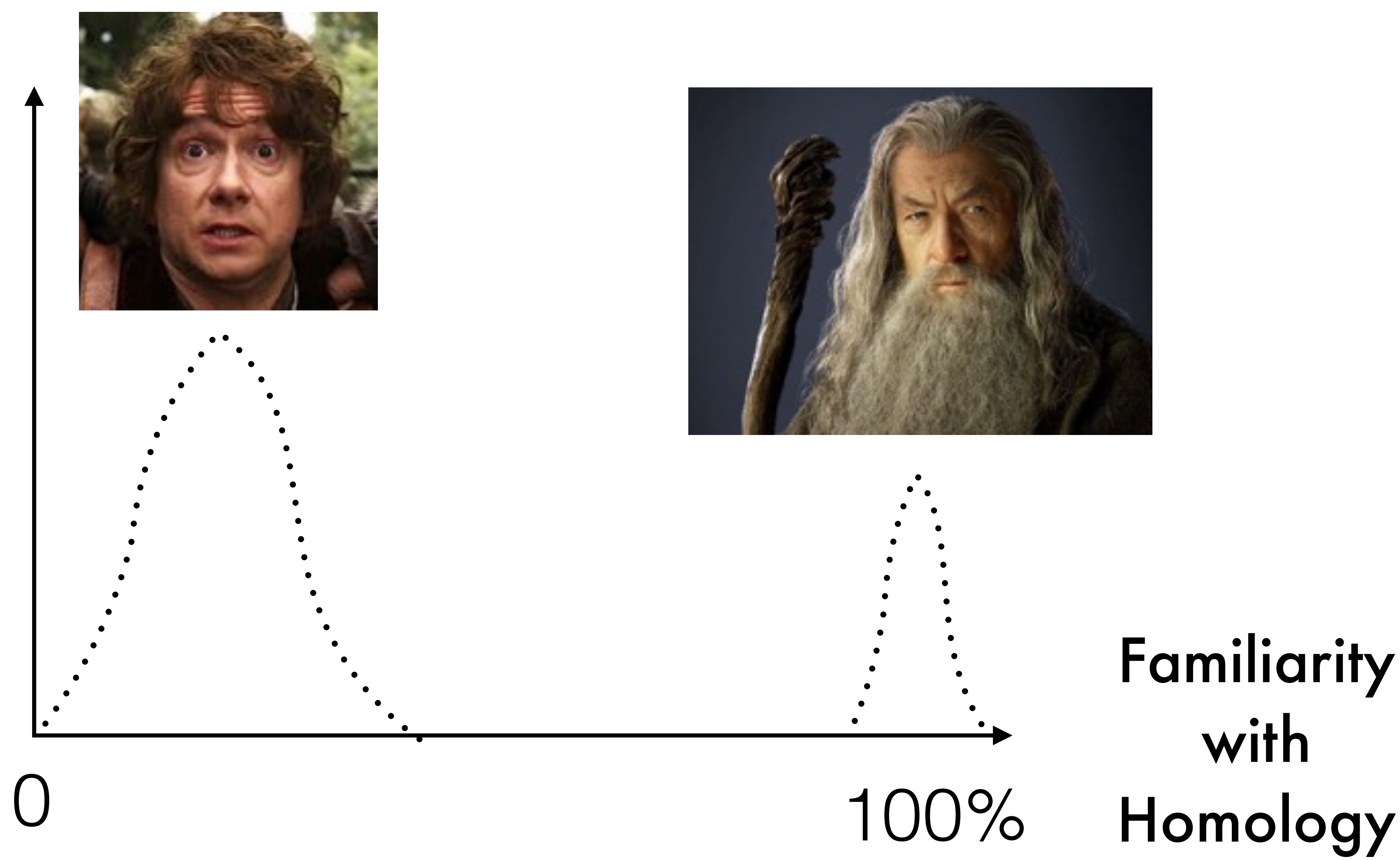
Dan Browne - University College London

What is homology?

- A combination of **topology** and **group theory** providing tools to characterise topological spaces.

The purpose of this talk

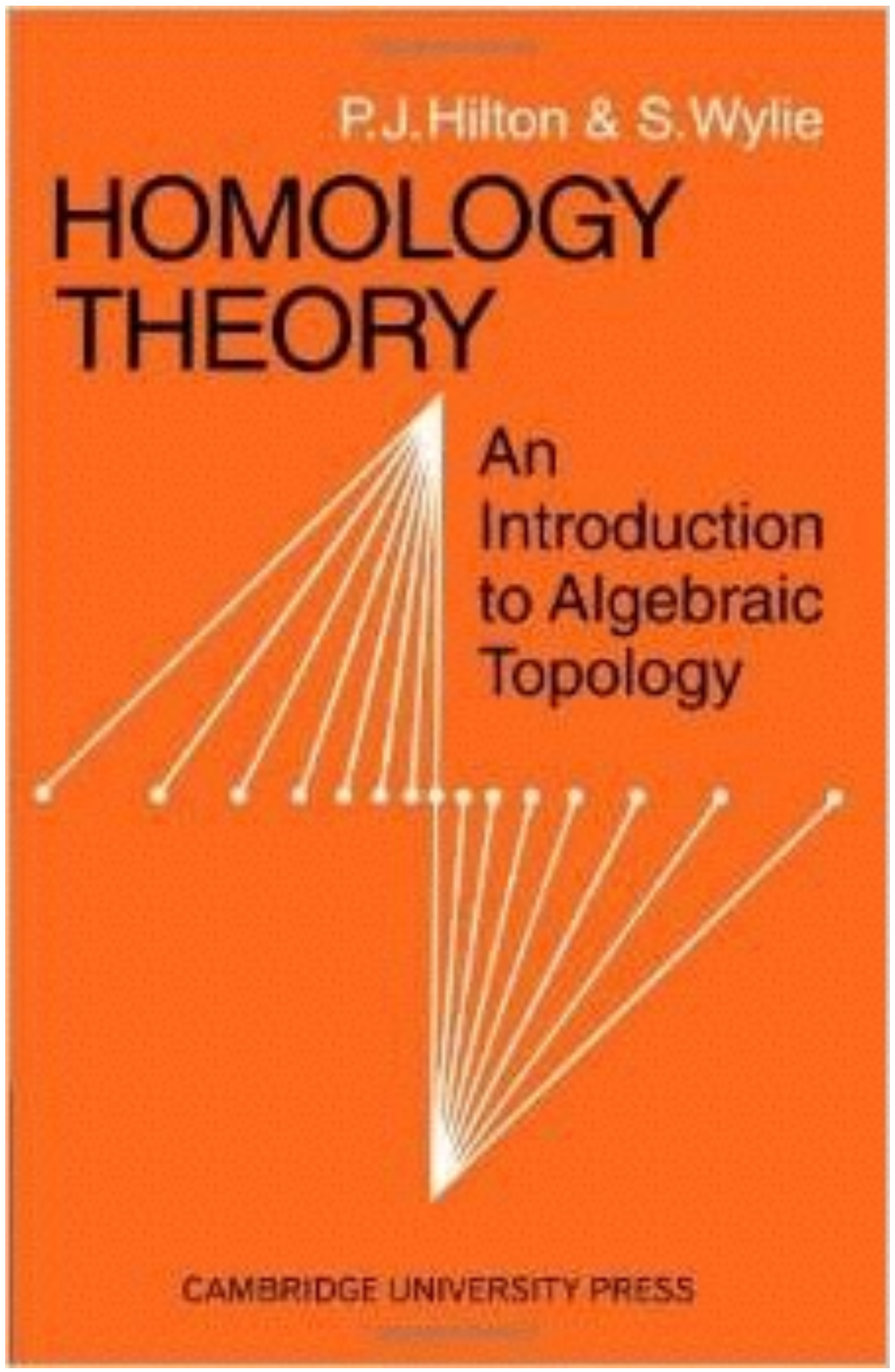
The QEC Community



How mathematicians use (co)homology

- Algebraic topology
- Differential geometry
- Abstract algebra
- E.g. Wiles' proof of Fermat's Last Theorem

How mathematicians learn homology



1 Analytic topology

A *topological space* is a set X in which certain subsets, called *open sets*, are distinguished; the collection of open sets satisfies the axioms:

(O 1) the union of any number of open sets is open;
(O 2) the intersection of any finite number of open sets is open;
(O 3) the whole space and the empty set are open.

To prescribe the open sets is to assign a *topology* to the set X. If $\mathscr{U}$, $\mathscr{V}$ are two topologies on the set X, then $\mathscr{U}$ is *finer* than $\mathscr{V}$ ($\mathscr{V}$ is *coarser* than $\mathscr{U}$) if every set of X which is open in the topology $\mathscr{V}$ is open in the topology $\mathscr{U}$. A set of open sets of X forms a *base* (for the open sets) if every open set of X is a union of sets of the base.

A *closed* subset of the topological space X is the complement of an open set; thus a topology is assigned by prescribing the closed sets and the closed sets must satisfy the axioms:

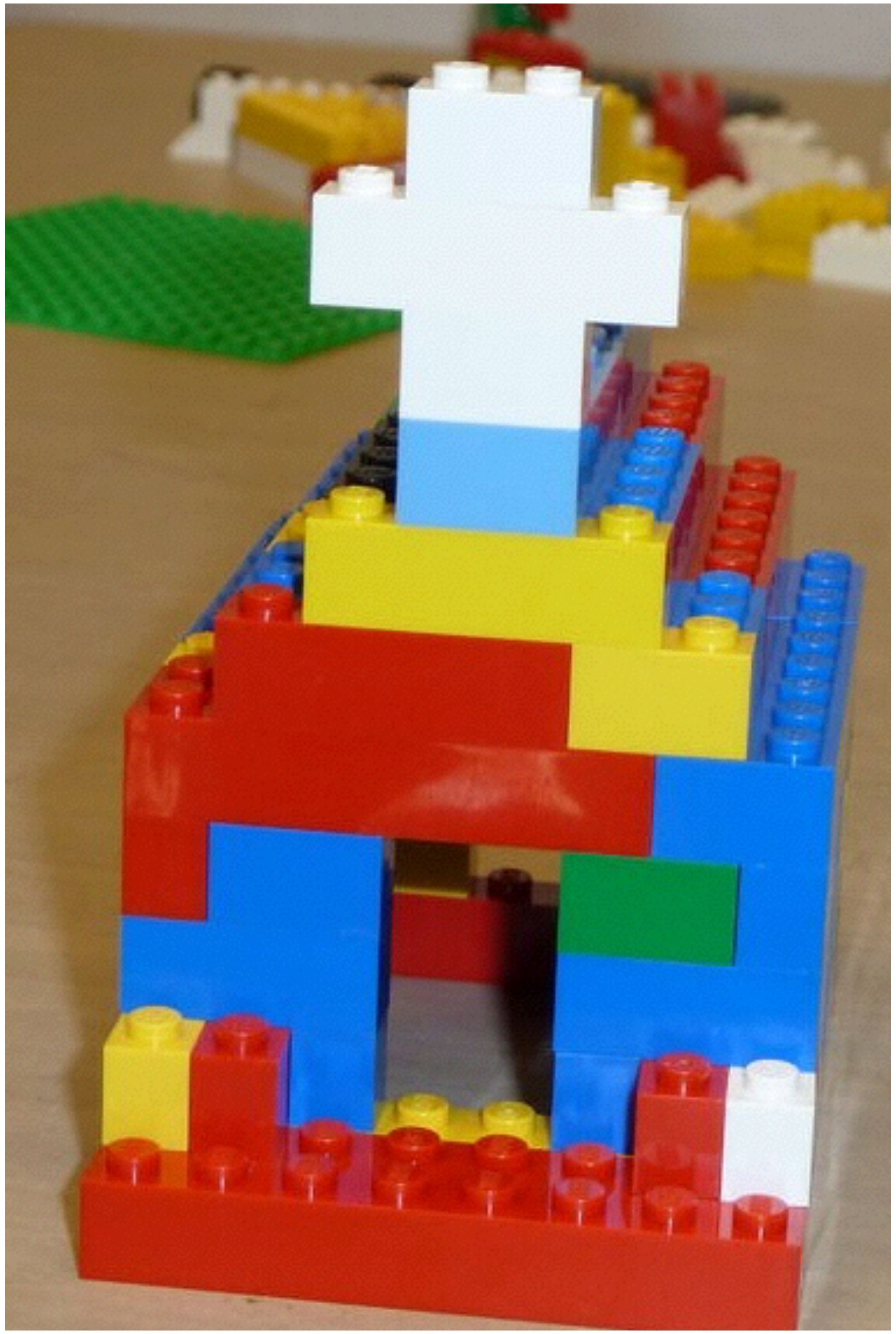
(C 1) the union of any finite number of closed sets is closed;
(C 2) the intersection of any number of closed sets is closed;
(C 3) the whole space and the empty set are closed.

If X_0 is a subset of the topological space X, the *induced topology* in X_0 is that in which the open sets are the intersections with X_0 of

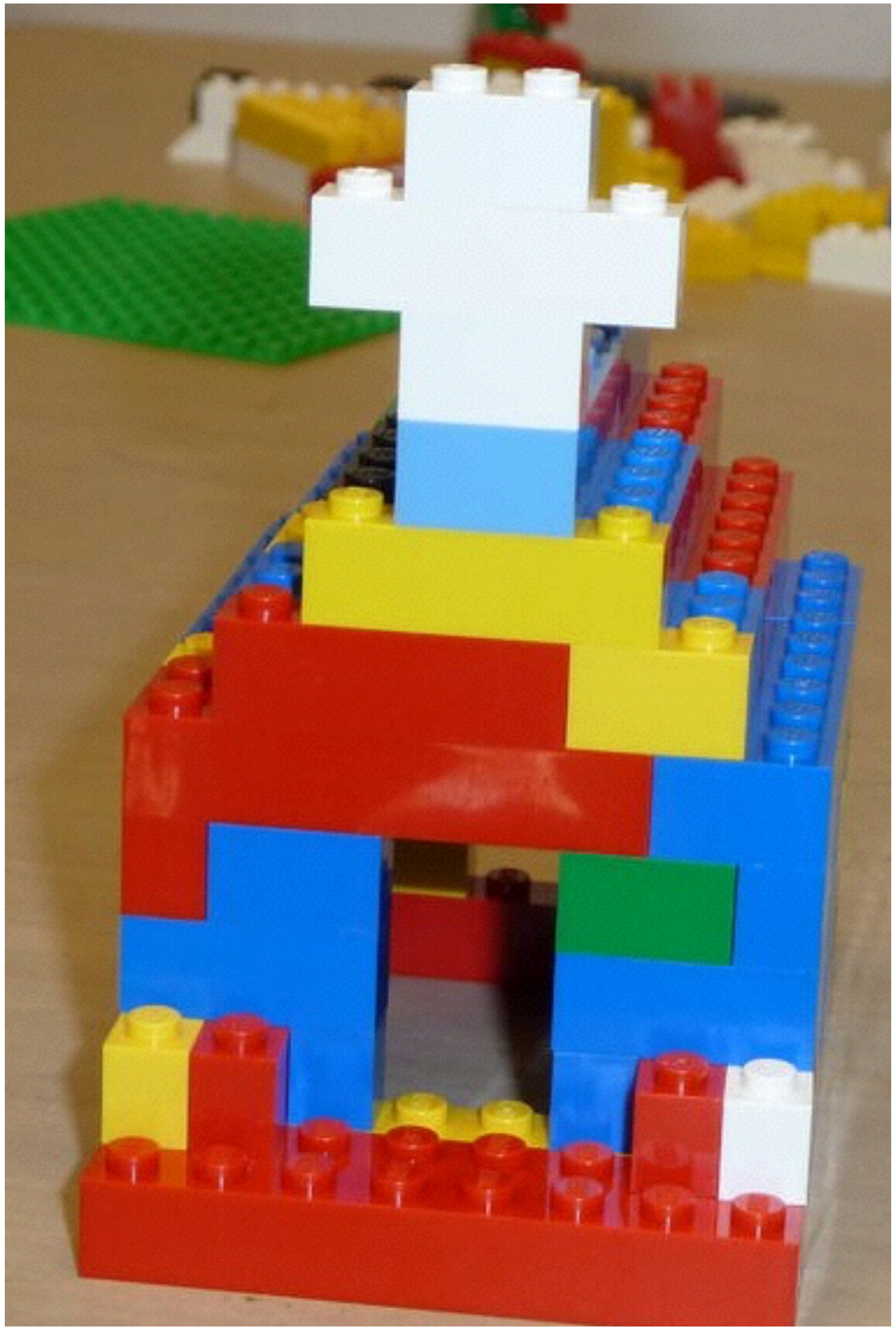
Page 1 of **Hilton and Wylie…**

How mathematicians use (co)homology

How we use homology in QEC



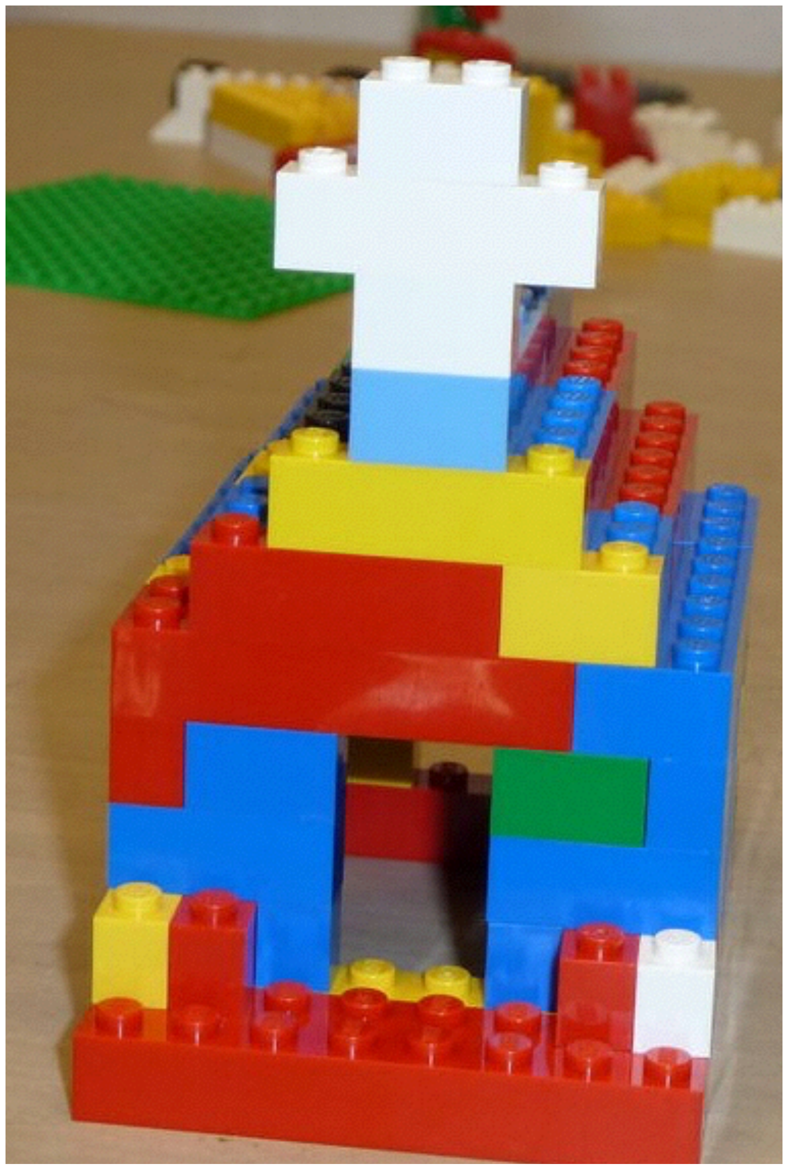
How we use homology in QEC



- The simplest groups
- No infinities
- No infinitessimals
- Qubit codes - particularly simple!

If Homology was taught at school….

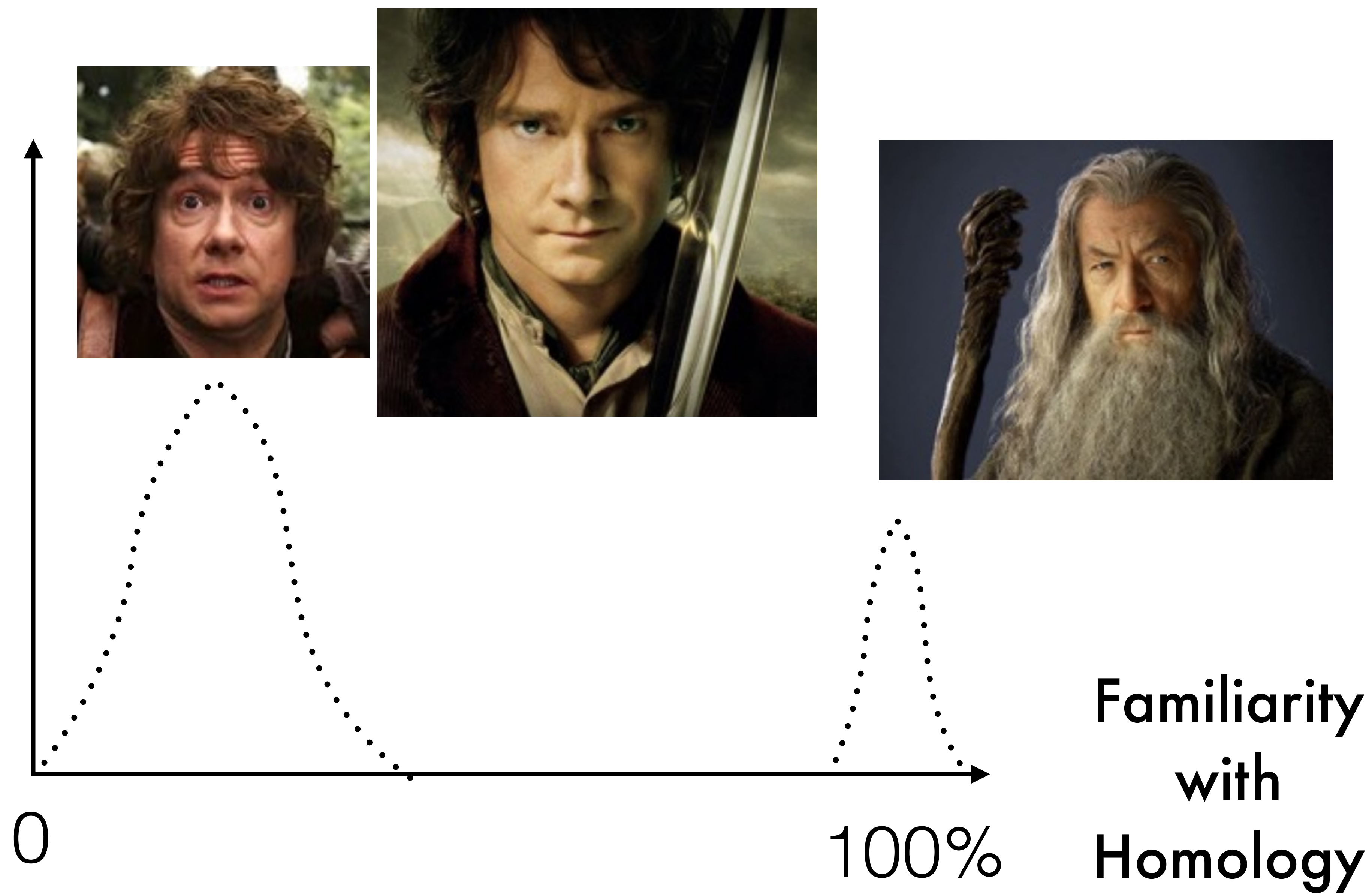
Why we use homology in QEC



- **Homology** captures **all** features of **Kitaev surface codes.**
- **Toric**, **planar**, **3D**, **4D codes:** (almost) identical definitions in homology terms.
- Homology = how these codes “work”
- Powerful basis for generalisation
- Convenient terminology - **if you know it!**

This lecture

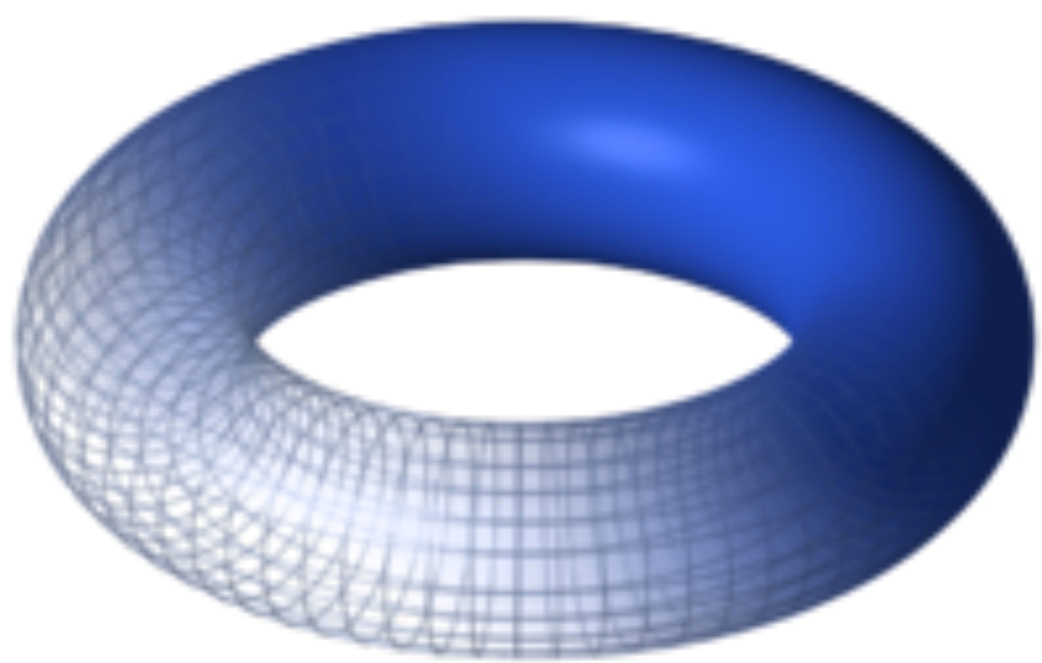
The QEC Community



This lecture

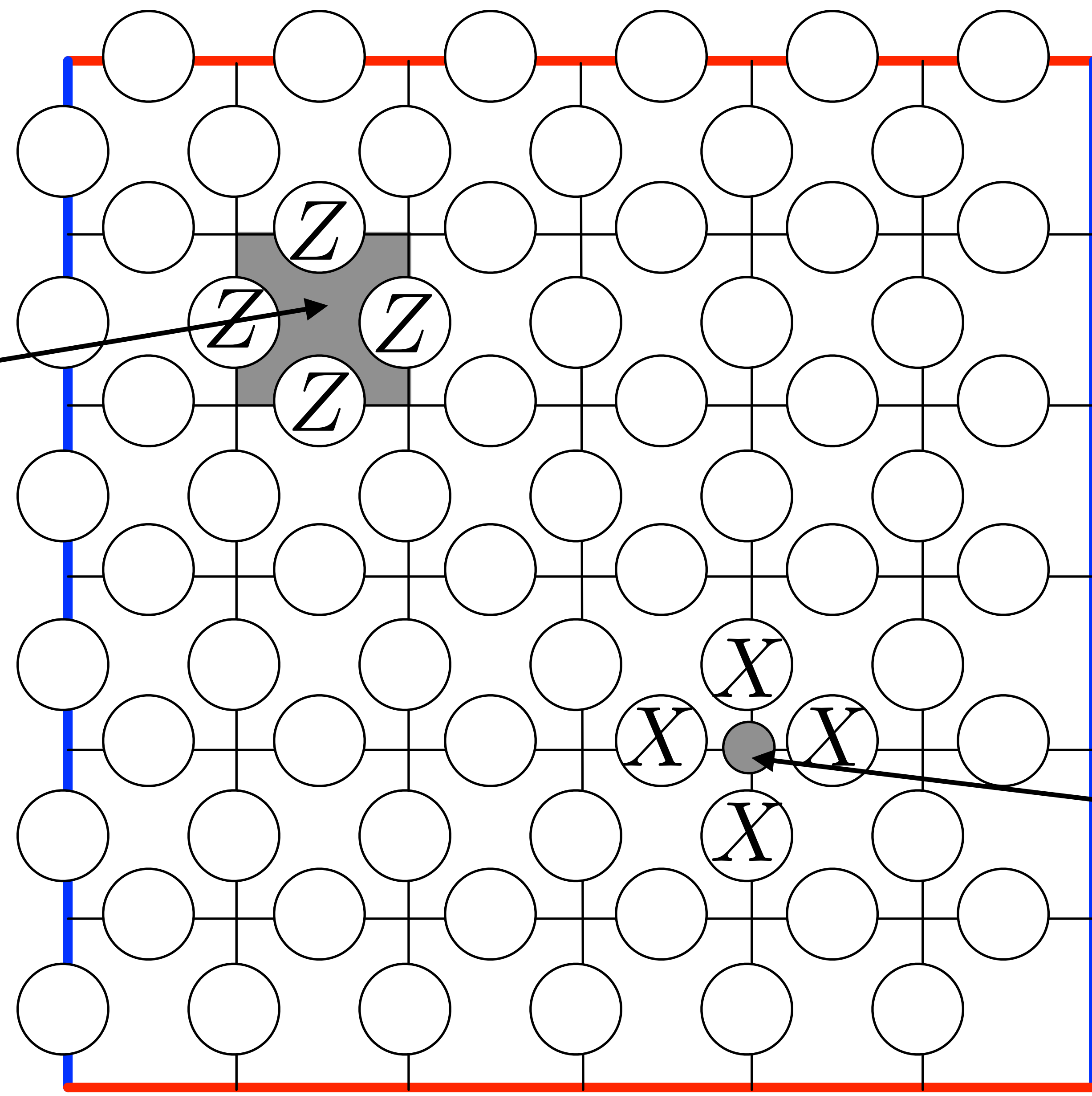
An introduction to the key **concepts** and **terminology** of **homology**.

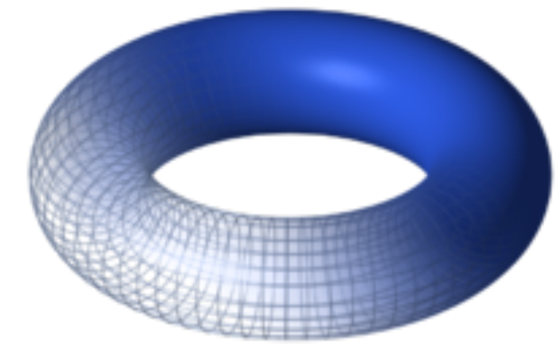
Illustrated with **concrete examples** from the **toric code.**



The **Toric code**

- Encodes **2 qubits** with distance **L** on an **L x L** toric lattice.
- Stabilizer generators associated with each **plaquette** and **vertex**.





Periodic boundaries
Like colours identified

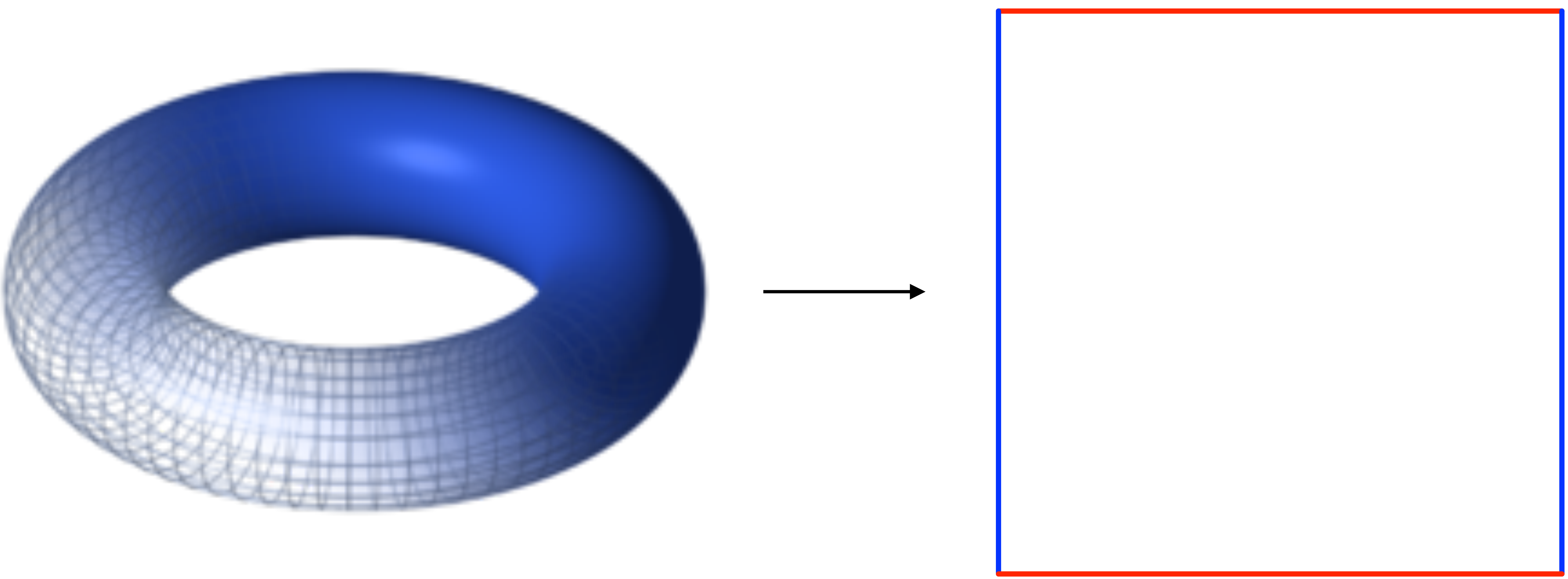
- **A.Y. Kitaev, *Fault-tolerant quantum computation by anyons***, Annals Phys. 303 (2003) 2-30

What is homology?

- A combination of **topology** and **group theory** providing tools to characterise topological spaces.

Topology - Cellulation

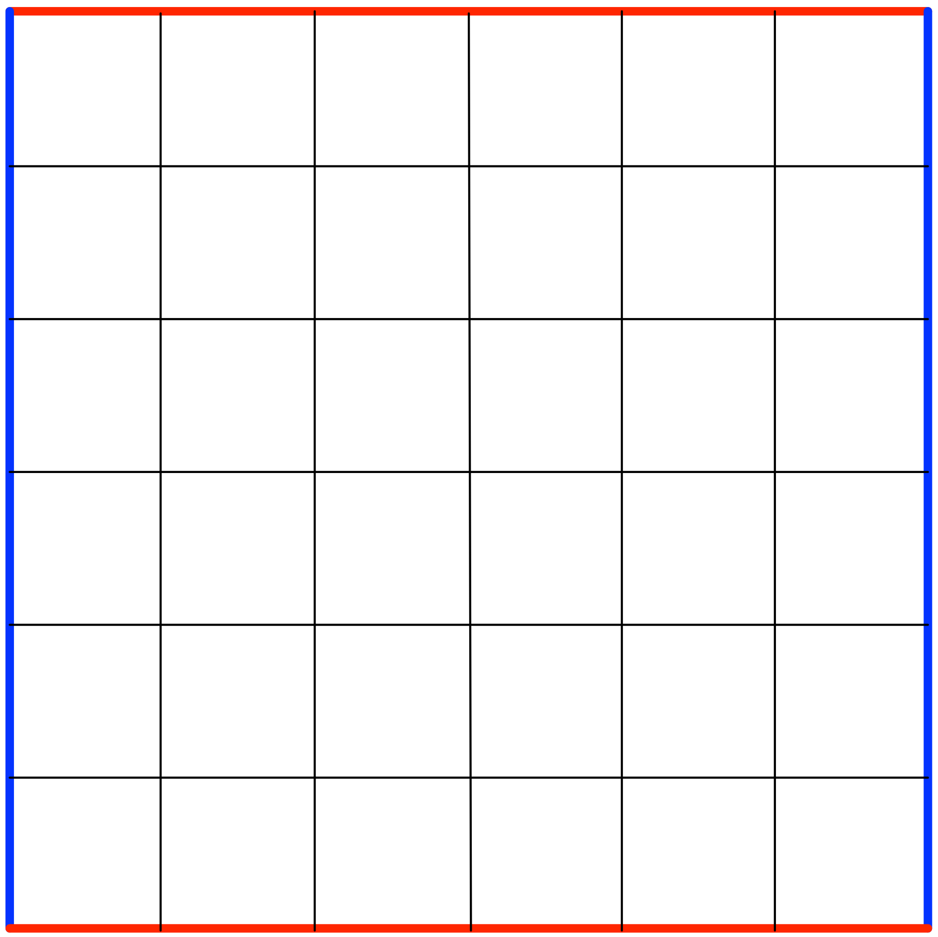
- A division of a **d-dimensional space** into a **tiling** of **d-dimensional objects**.



E.g. the **torus**

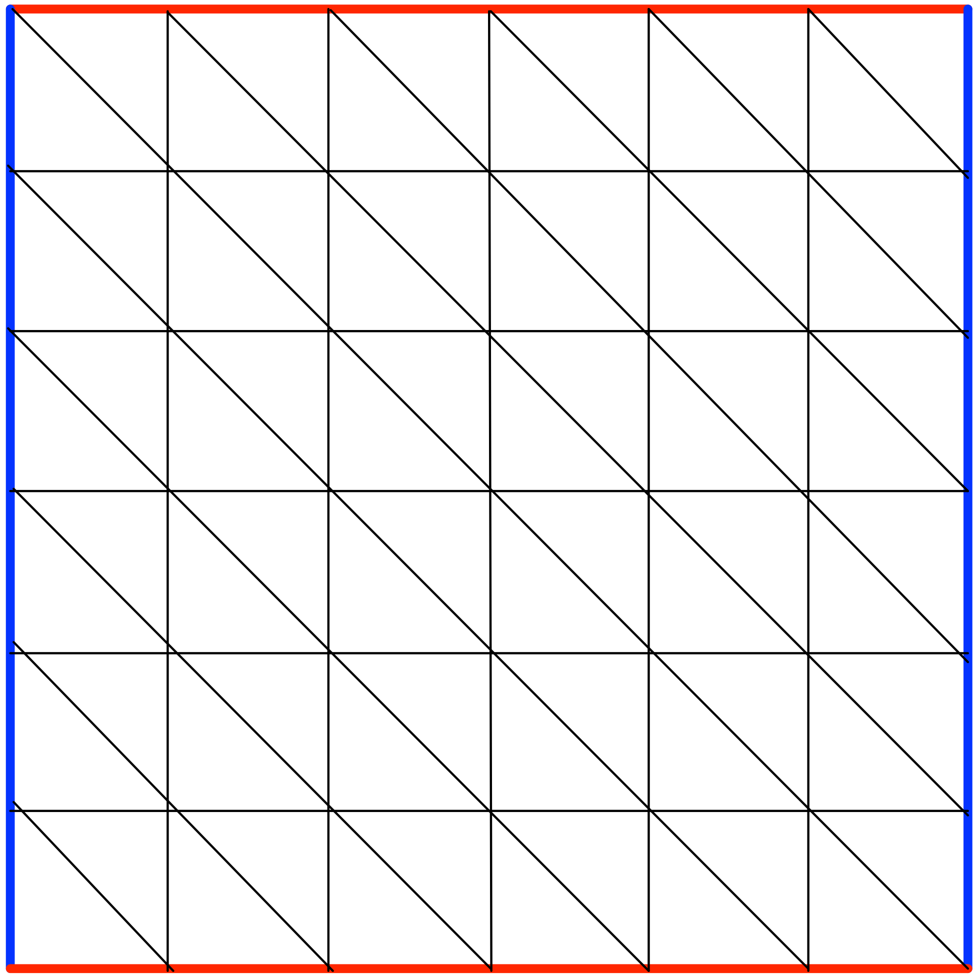
Topology - Cellulation

- A division of a **d-dimensional space** into a **tiling** of **d-dimensional objects**.



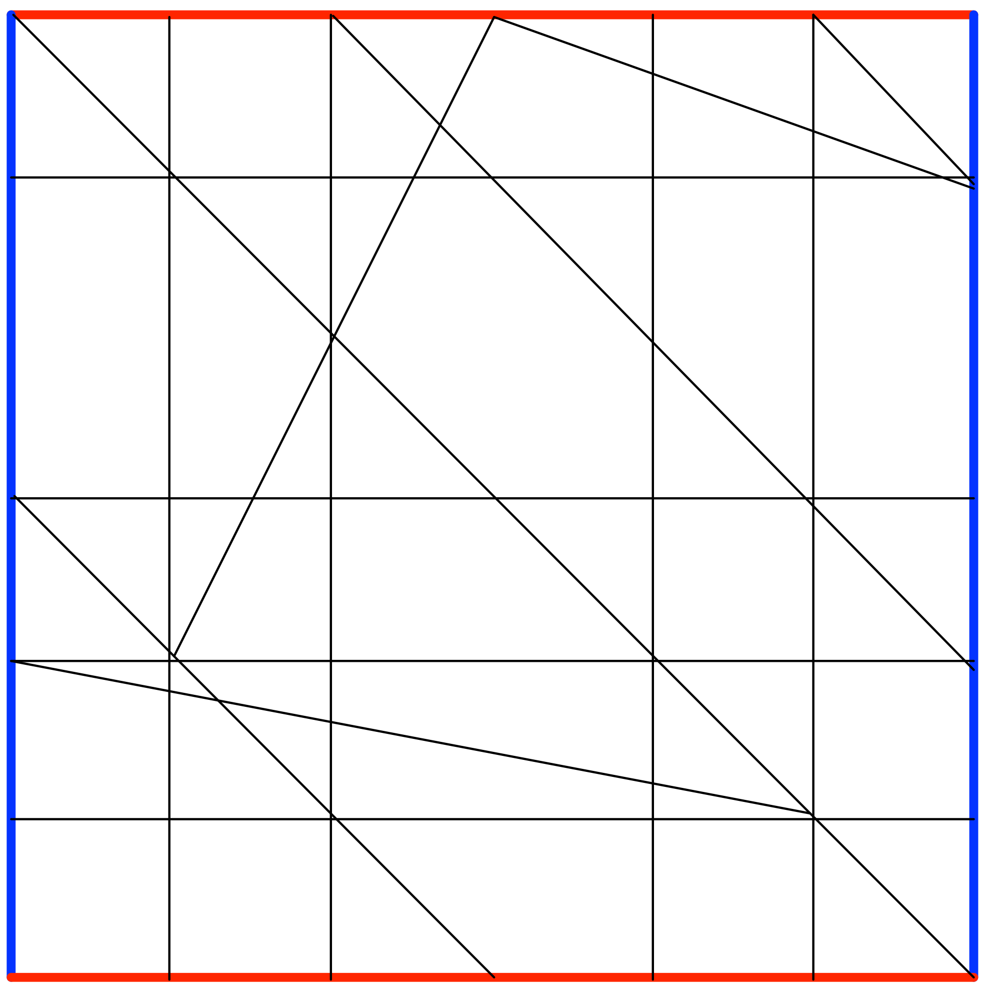
Topology - Cellulation

- A division of a **d-dimensional space** into a **tiling** of **d-dimensional objects**.



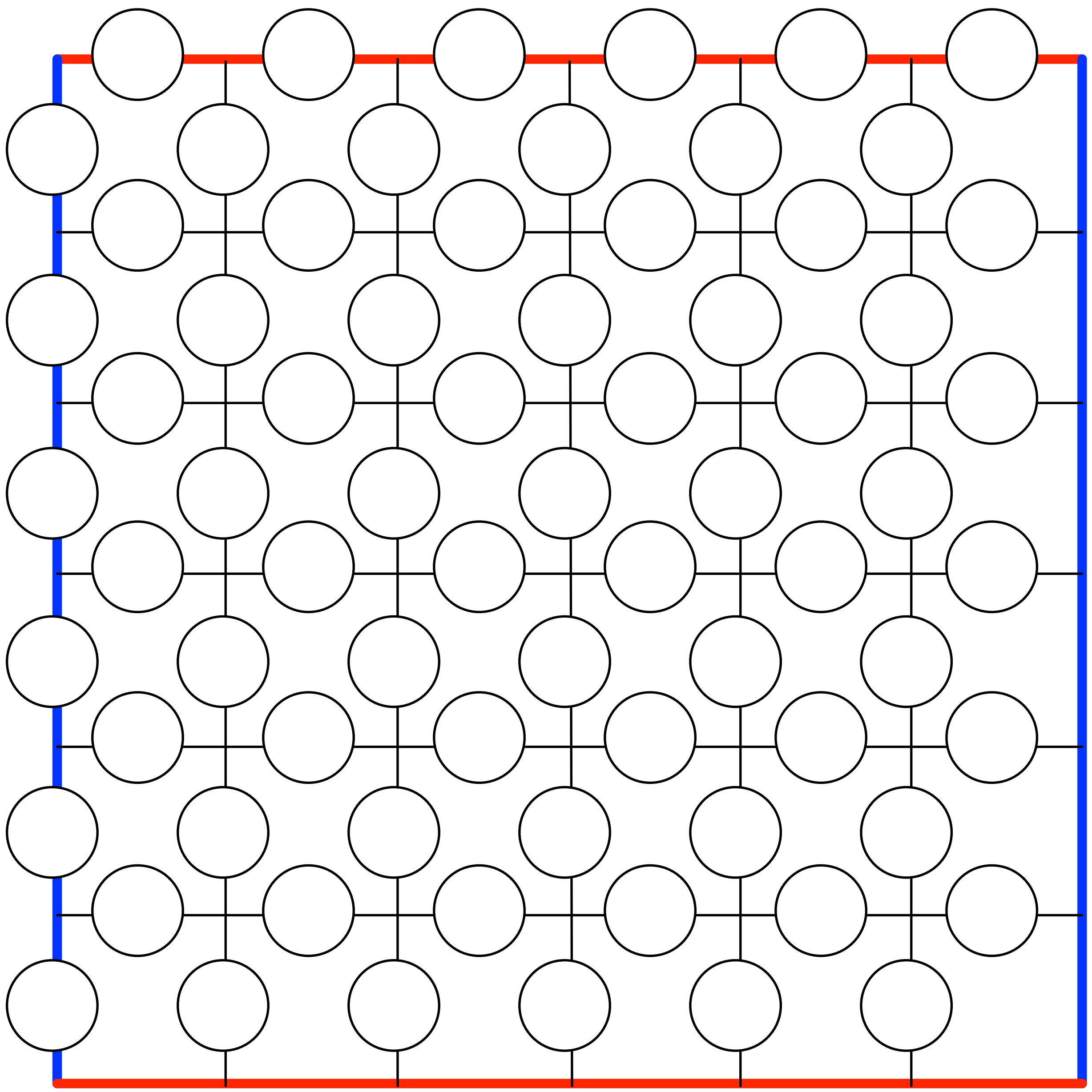
Topology - Cellulation

- A division of a **d-dimensional space** into a **tiling** of **d-dimensional objects**.



Cellulation in the **Toric code**

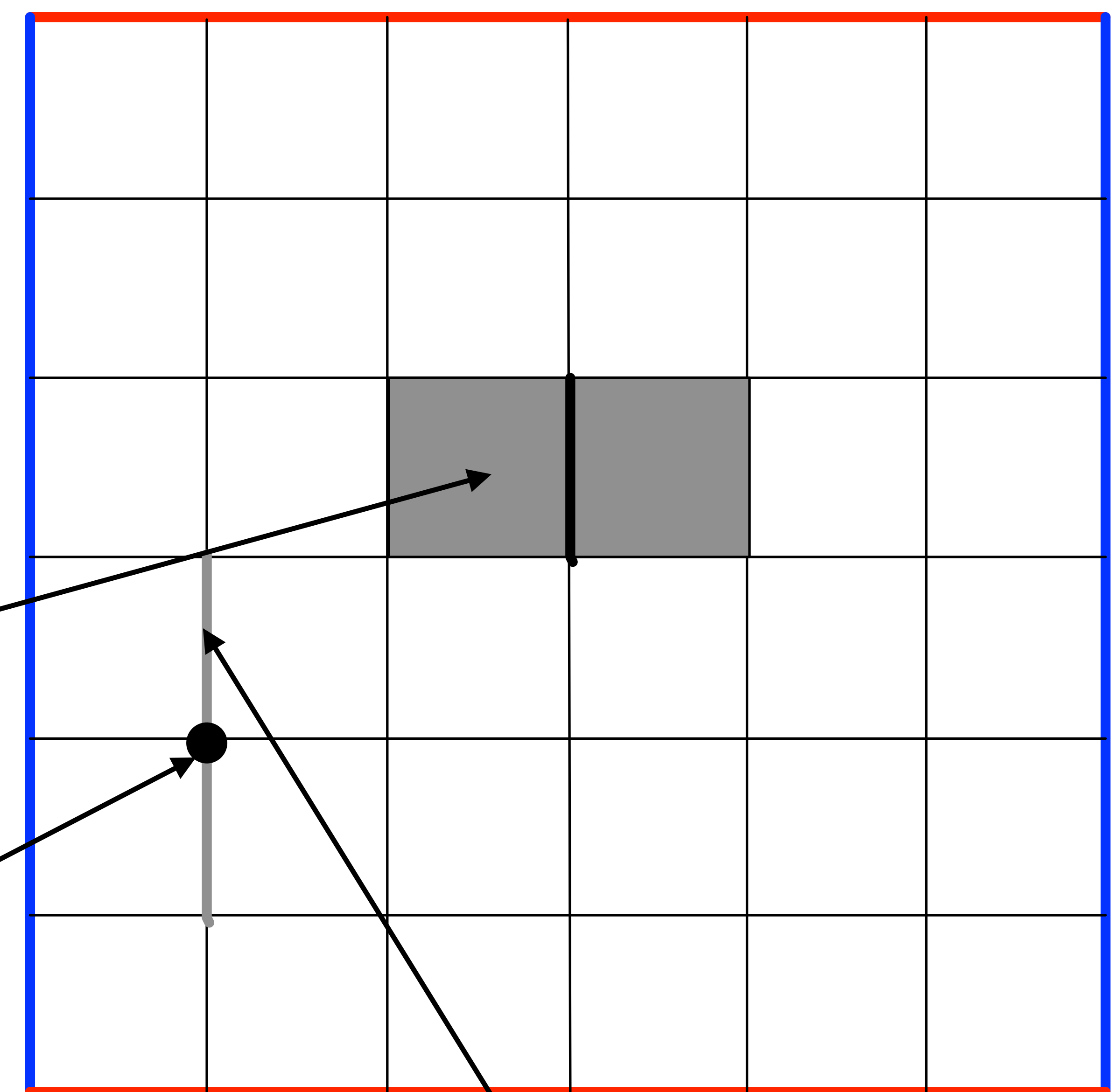
- **Toric code:** Qubits associated with **edges** of a cellulation of the torus



Topology - Cellulation

- Where two **n-dim.** objects **meet** an **(n-1)-dim**. object is defined.

- Terminology: **n-cells.**



What is homology?

- A combination of topology and **group theory** providing tools to characterise topological spaces.

Z_2 - the simplest group

The group of a single bit

$$x \in \{0, 1\}$$

$$0 \to 1 \qquad 1 \to 0$$

$$x \to x \oplus 1$$

- The group: $\mathbb{Z}_2$

- Elements: 0, 1
- Group composition: **addition modulo 2**

$$0 + 0 = 0$$
$$0 + 1 = 1$$
$$1 + 1 = 0$$

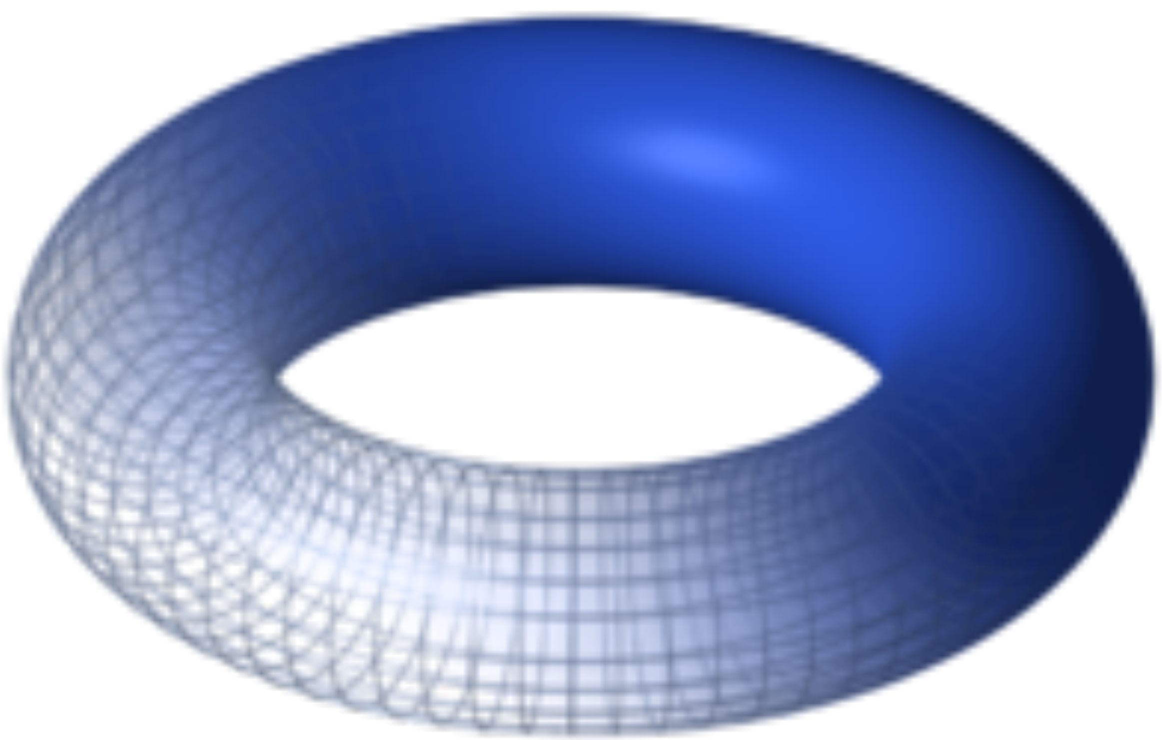
An **Abelian** group.

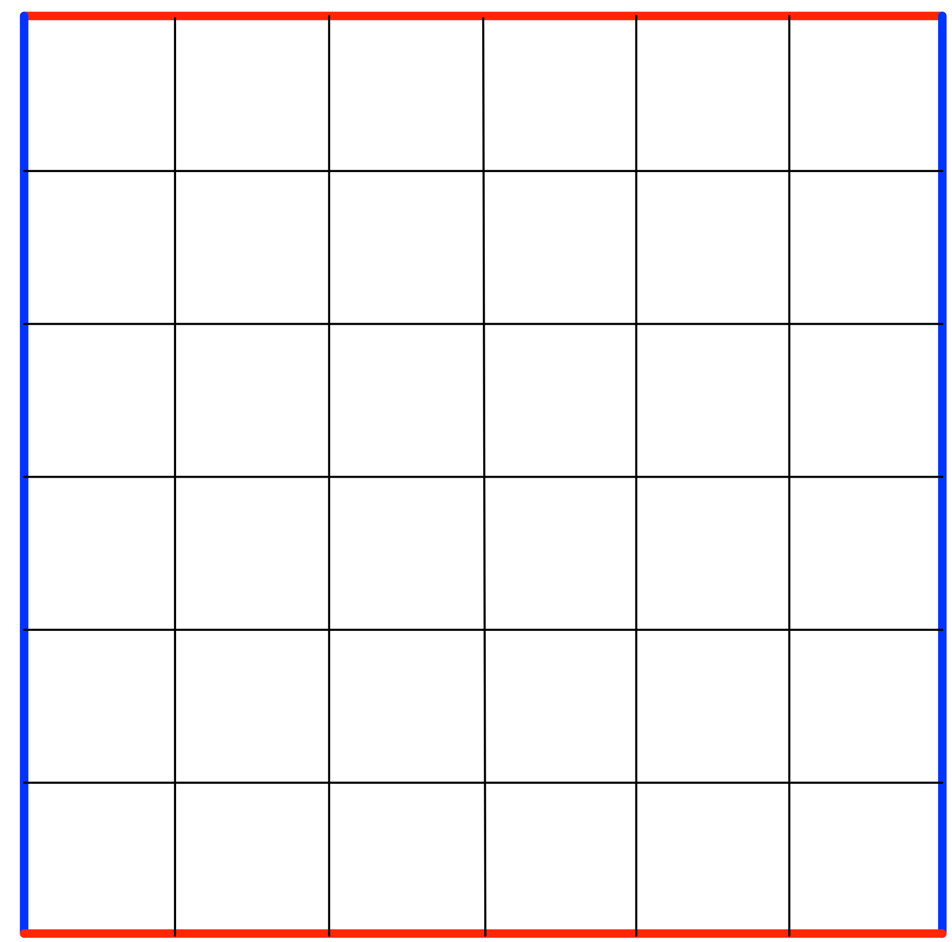
Every element is **self-inverse**.

Chains

Chains

- *Starting points:*
    - a **cellulation** of a topological surface (or space)





- a **group:**

$$\mathbb{Z}_2$$

Chains

- *Definition:* ***n-chain***
  - An assignment of an **element** of the **group** (here $\mathbf{Z_2}$) to every **n-cell** in the cellulation.

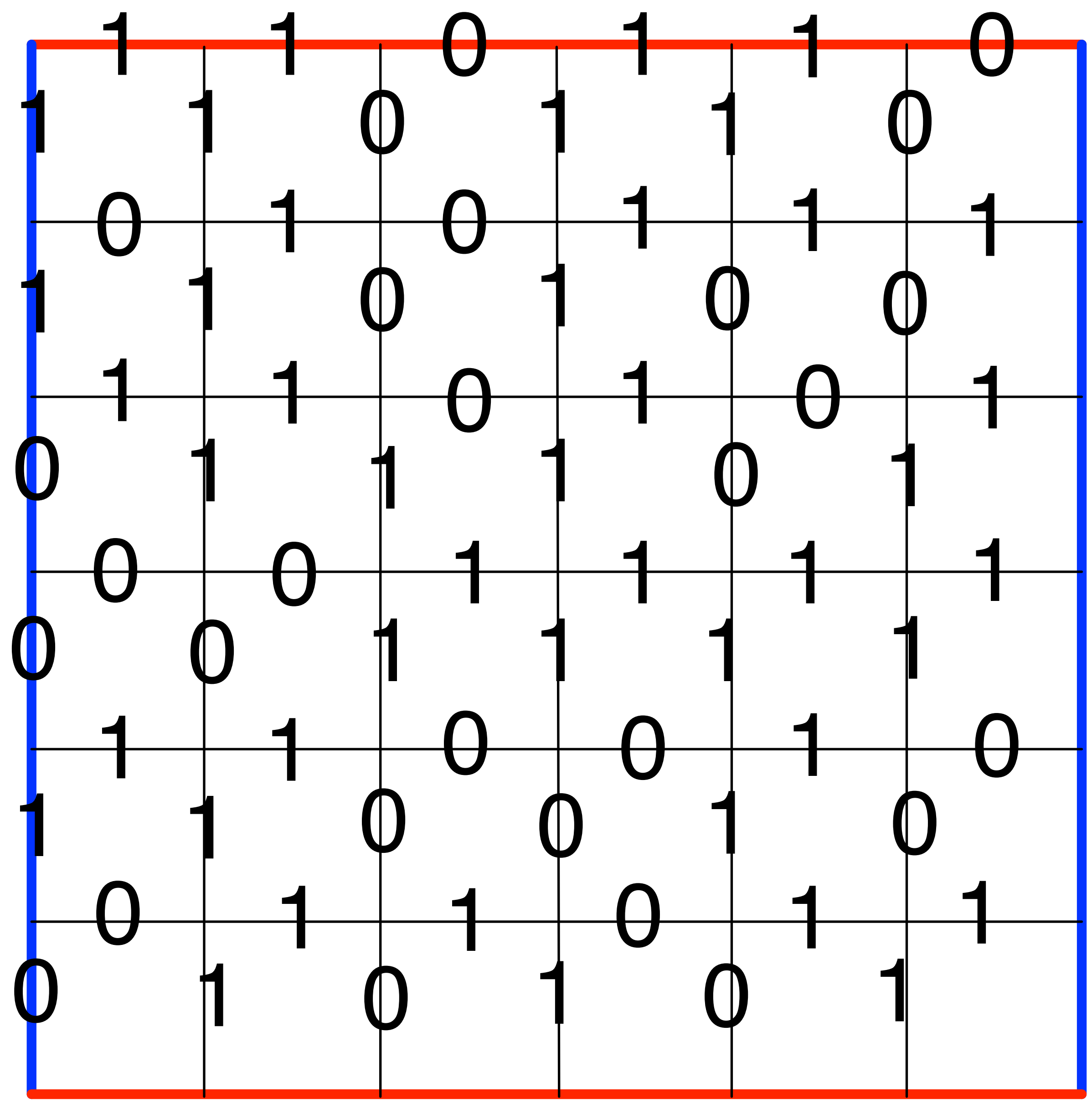
- *Example:* 2-chain

1	1	0	1	1	0
1	1	0	1	1	1
1	1	1	1	0	1
0	0	1	1	1	1
1	1	0	0	1	0
0	1	1	1	1	1

Chains

- *Definition:* ***n-chain***
  - An assignment of an **element** of the **group** (here $\mathbf{Z_2}$) to every **n-cell** in the cellulation.

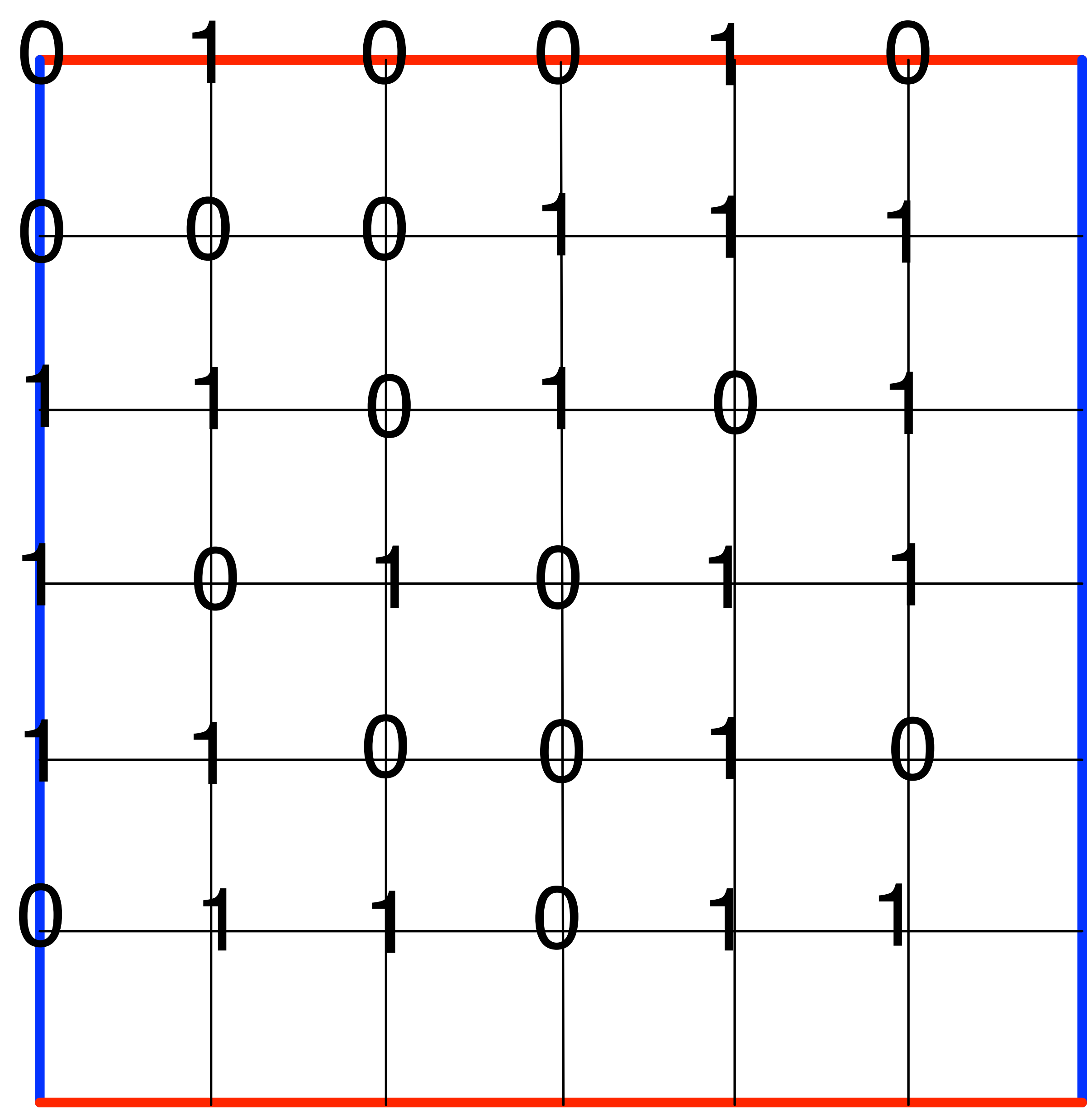
- *Example:* 1-chain



Chains

- *Definition:* ***n-chain***
  - An assignment of an **element** of the **group** (here $\mathbf{Z_2}$) to every **n-cell** in the cellulation.

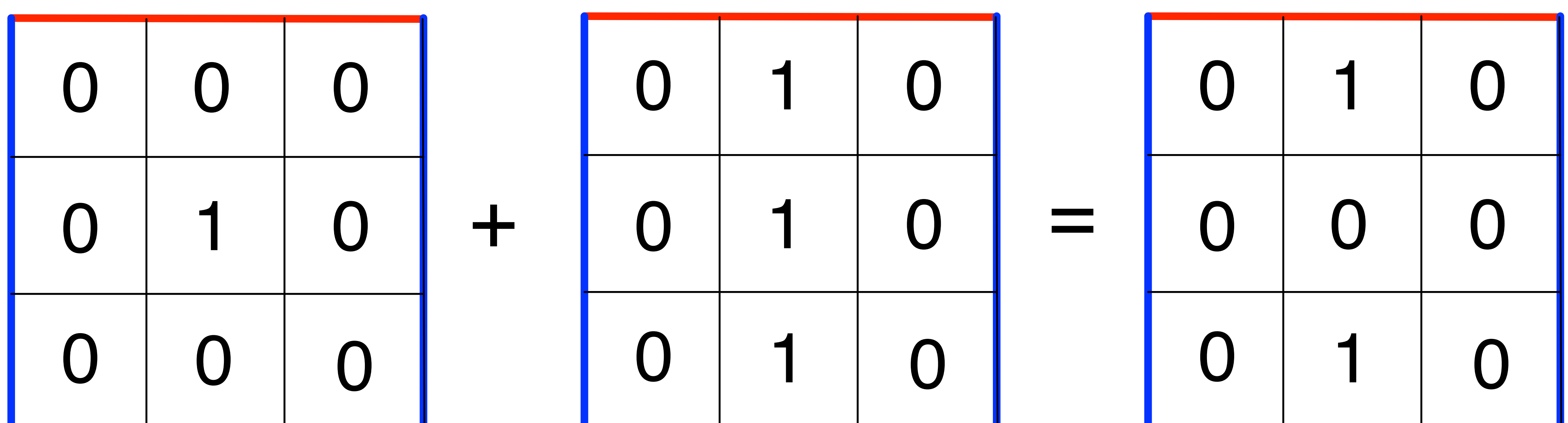
- *Example:* 0-chain



Chains

- *Definition:* ***n-chain***
  - An assignment of an **element** of the **group** (here $\mathbf{Z_2}$) to every **n-cell** in the cellulation.

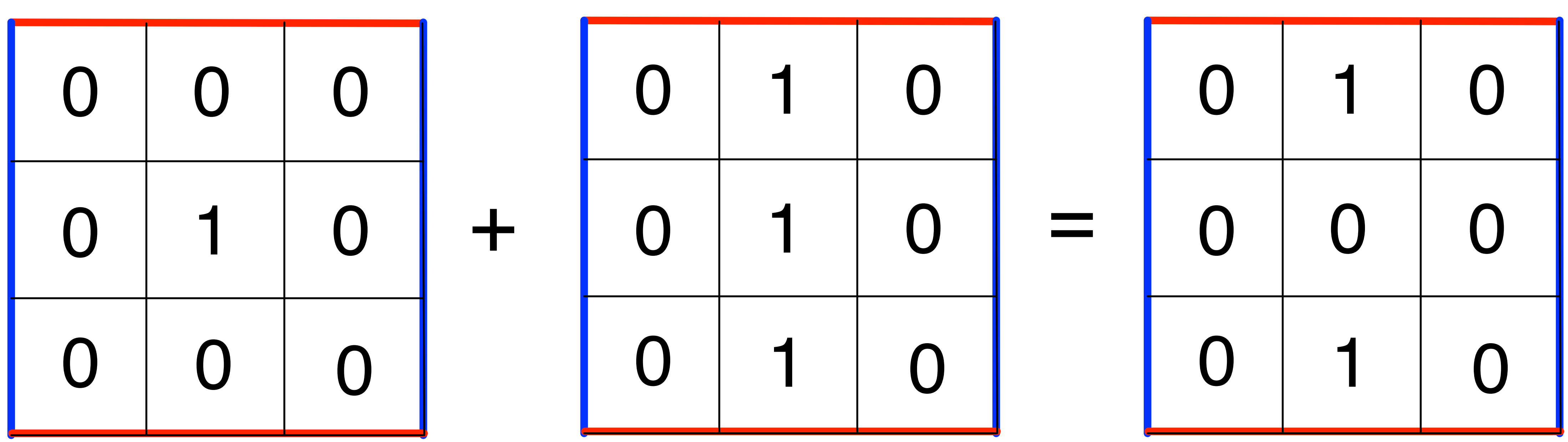
- Each set of n-chains forms a **group.**
- Group **composition:**cell-wise (bitwise) **addition** mod 2.
- Group **generators:** associated with each n-cell.



Chains

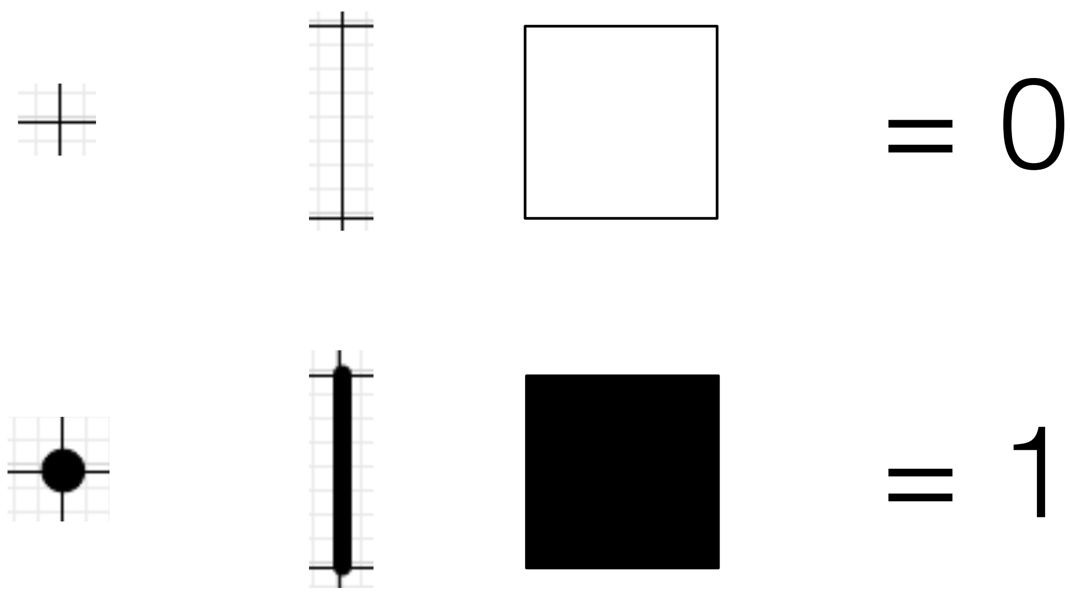
- *Definition:* ***n-chain***
  - An assignment of an **element** of the **group** (here $\mathbf{Z_2}$) to every **n-cell** in the cellulation.

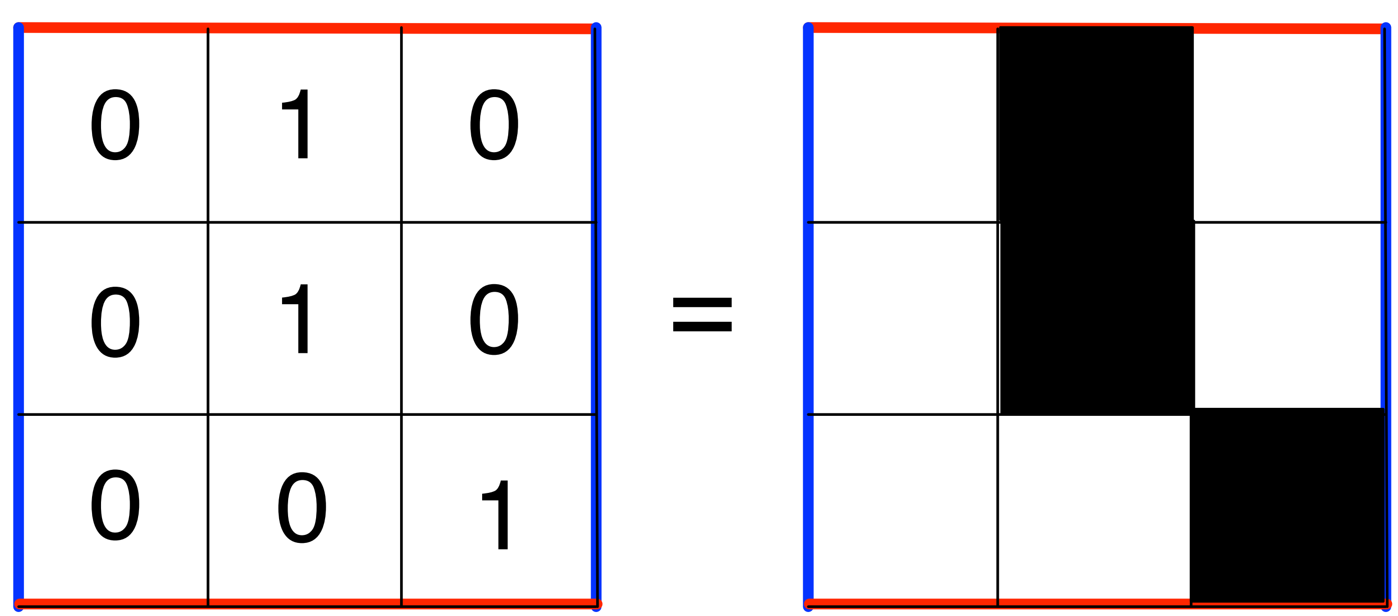
- Each set of n-chains forms a **vector space** over Z_2**.**
- **Vector addition:** cell-wise (bitwise) **addition** mod 2.
- Space **basis** vectors**:** associated with each n-cell.



Chains

- Useful alternative notation - **shading** (1's mark out a subset)

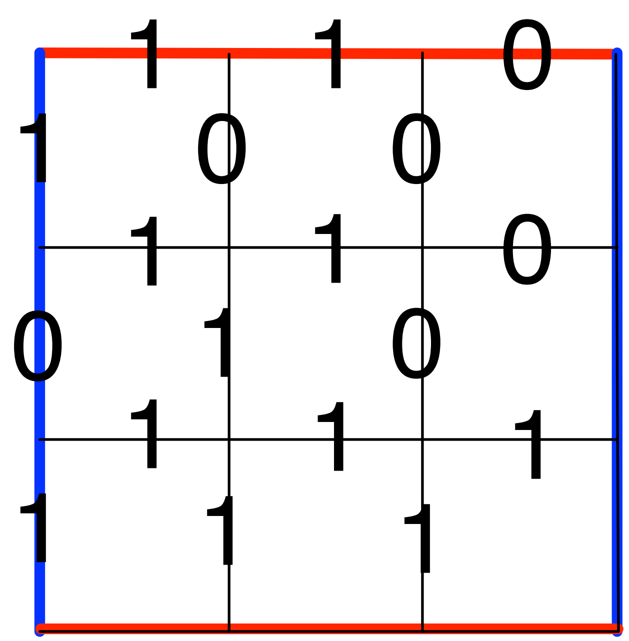




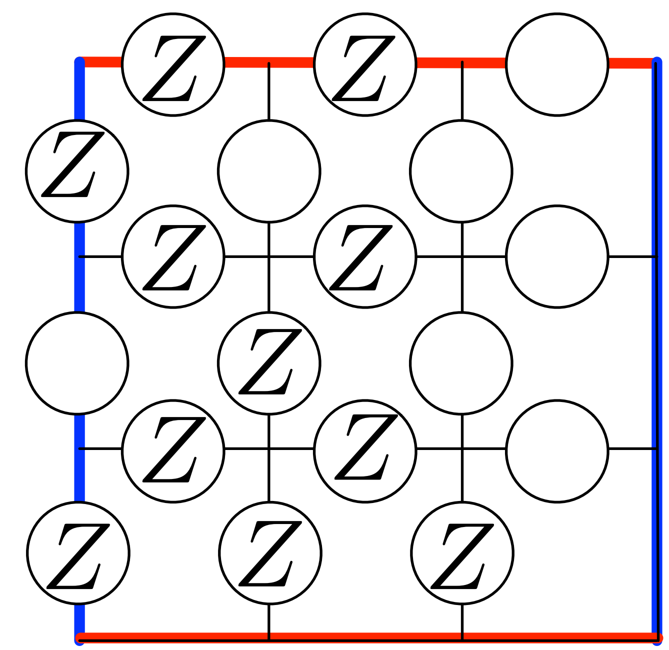
Chains in the **Toric code**

- 1-chains: 0s and 1s assigned to **edges**
  = 0s and 1s assigned to **qubits**.
- 1-chain represents ***errors***, ***stabilizer***, ***corrections*** for tensors of *same-type* Pauli operators.

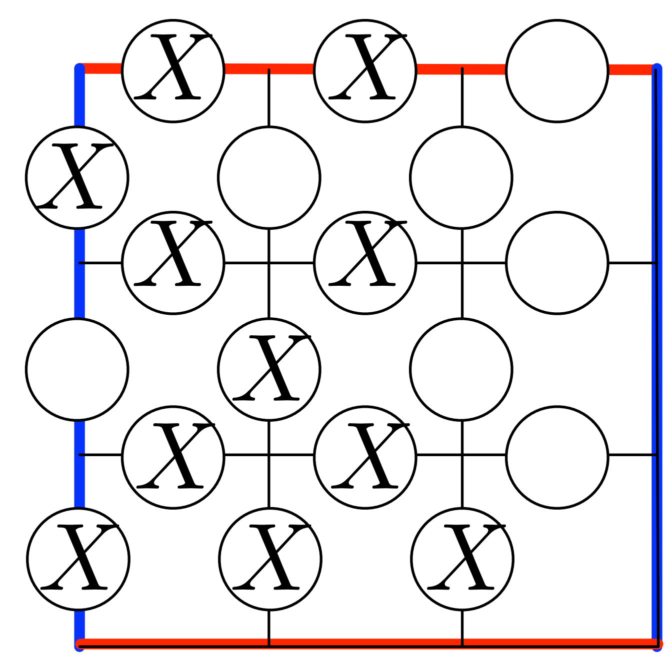
E.g. **c** =



Z(c) =



X(c) =



NB *Chain* **group structure** =
operator **group structure**

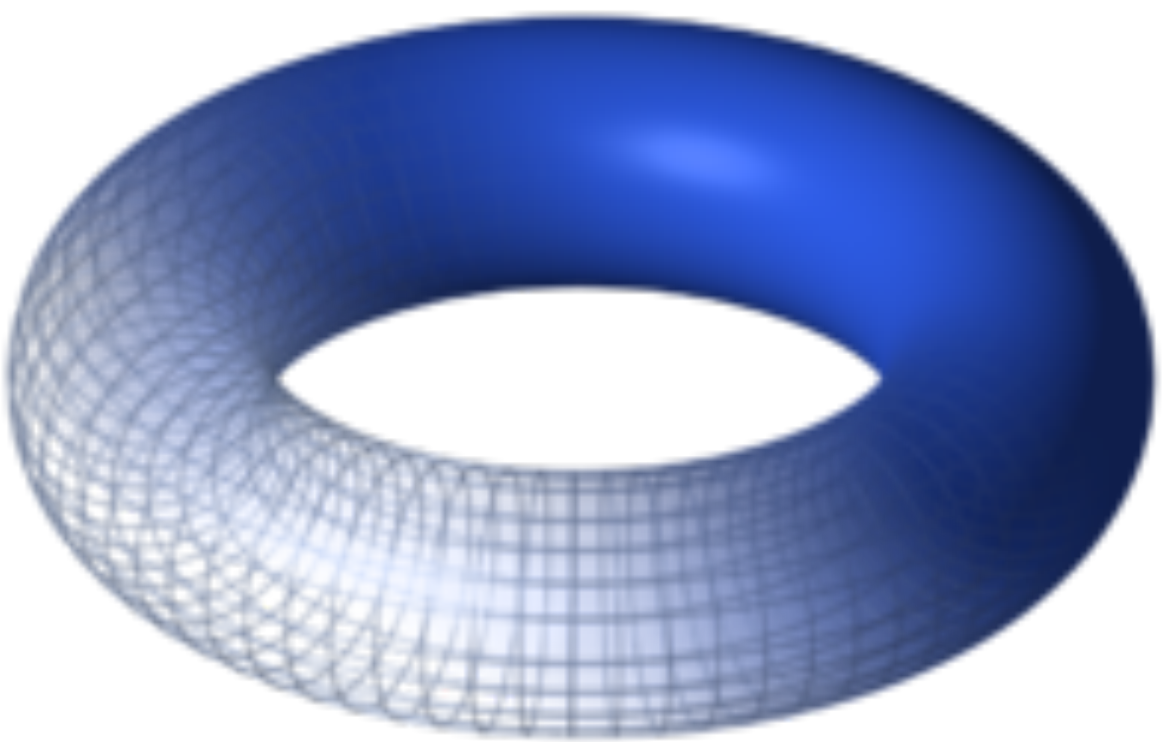
Chains

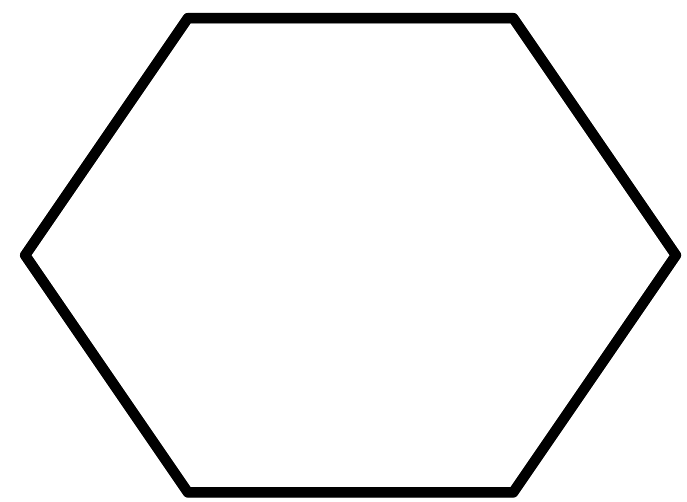
- **Warning:** "Chain" is a "**false friend**"
  - **Not** (usually) 1-dimensional or **string-like**
  - Confusingly, the 1-chain group does contain string-like elements!

Boundary

Boundary

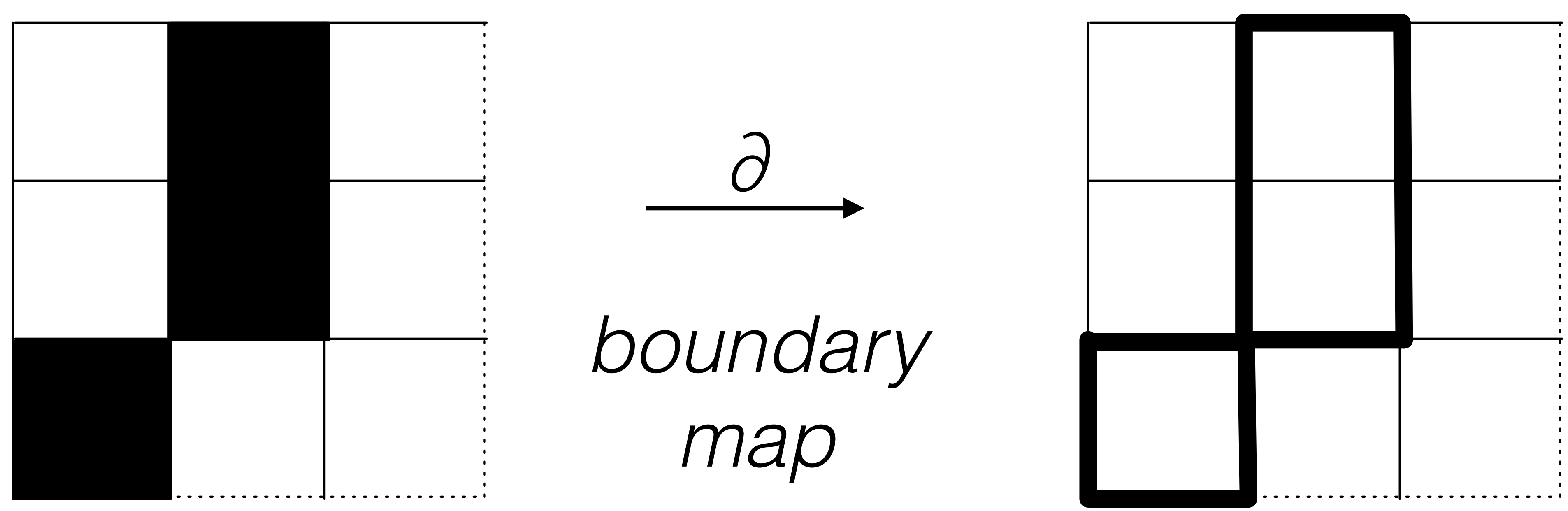
- Intuitively, **n-dim** objects have an **(n-1)-dim**. boundary / surface / edge.





Boundary

- In Z_2 homology, using our “shading” notation, the boundary map is **intuitive**:

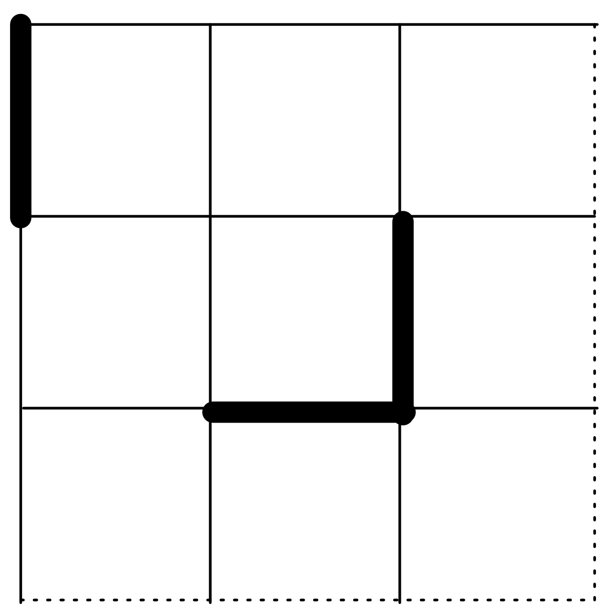


2-chain

1-chain

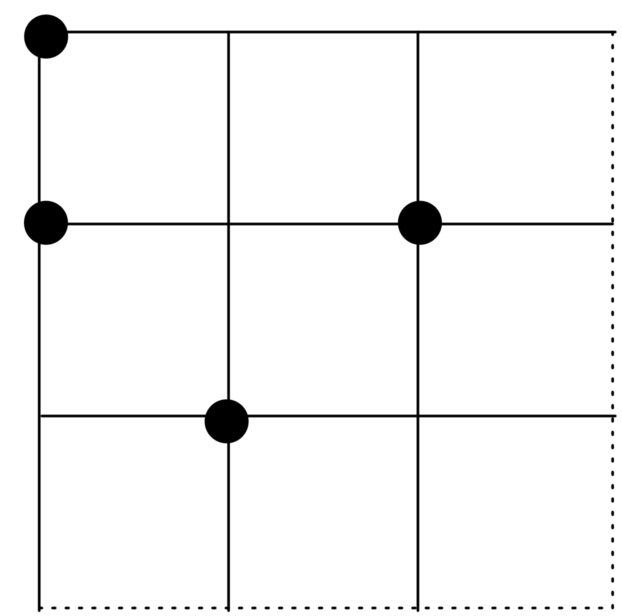
Boundary

- In Z_2 homology, using our "shading" notation, the boundary map is **intuitive**:



$\xrightarrow{\partial}$

boundary map



1-chain

0-chain

Boundary

- Formally the **boundary map** ∂ is a **group homomorphism** (= linear map) from **n**-chains to **(n-1)**-chains.

$$\begin{array}{ccccc} \text{2-chain group} & \xrightarrow{\partial} & \text{1-chain group} & \xrightarrow{\partial} & \text{0-chain group} \end{array}$$

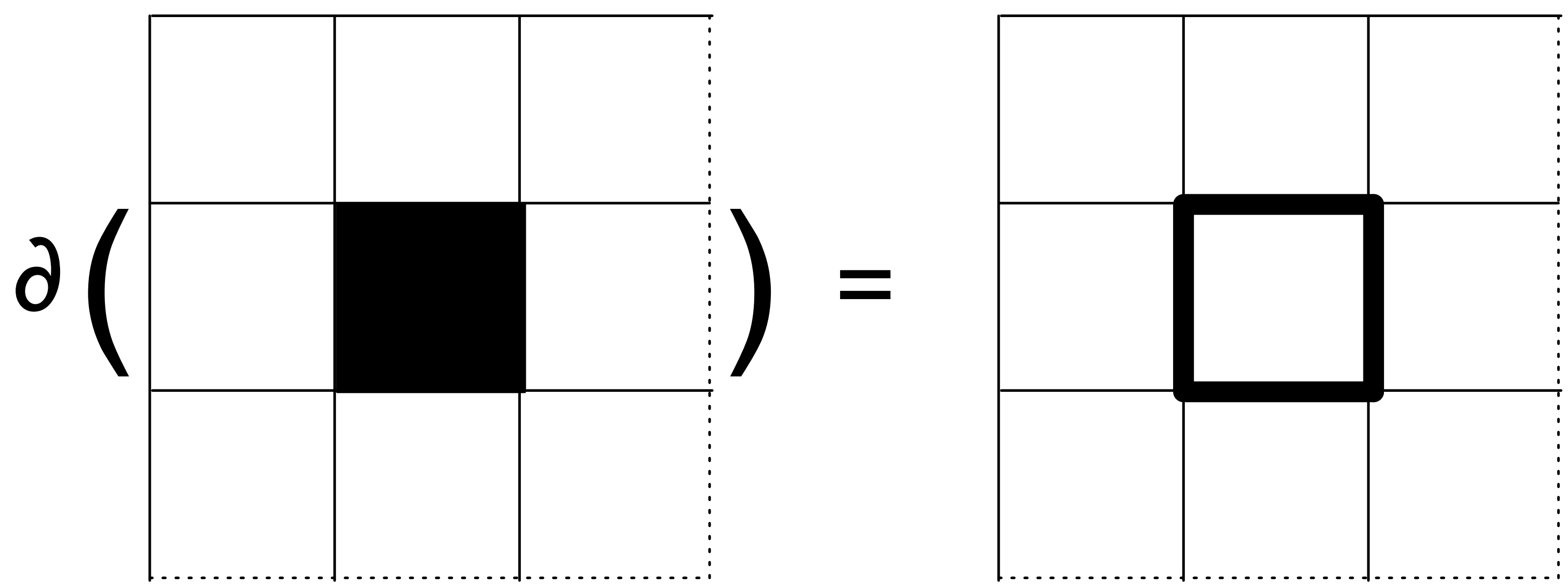
Defined on generators (single cells) and extended to arbitrary chains via:

$$\partial(a + b) = \partial(a) + \partial(b)$$

Boundary

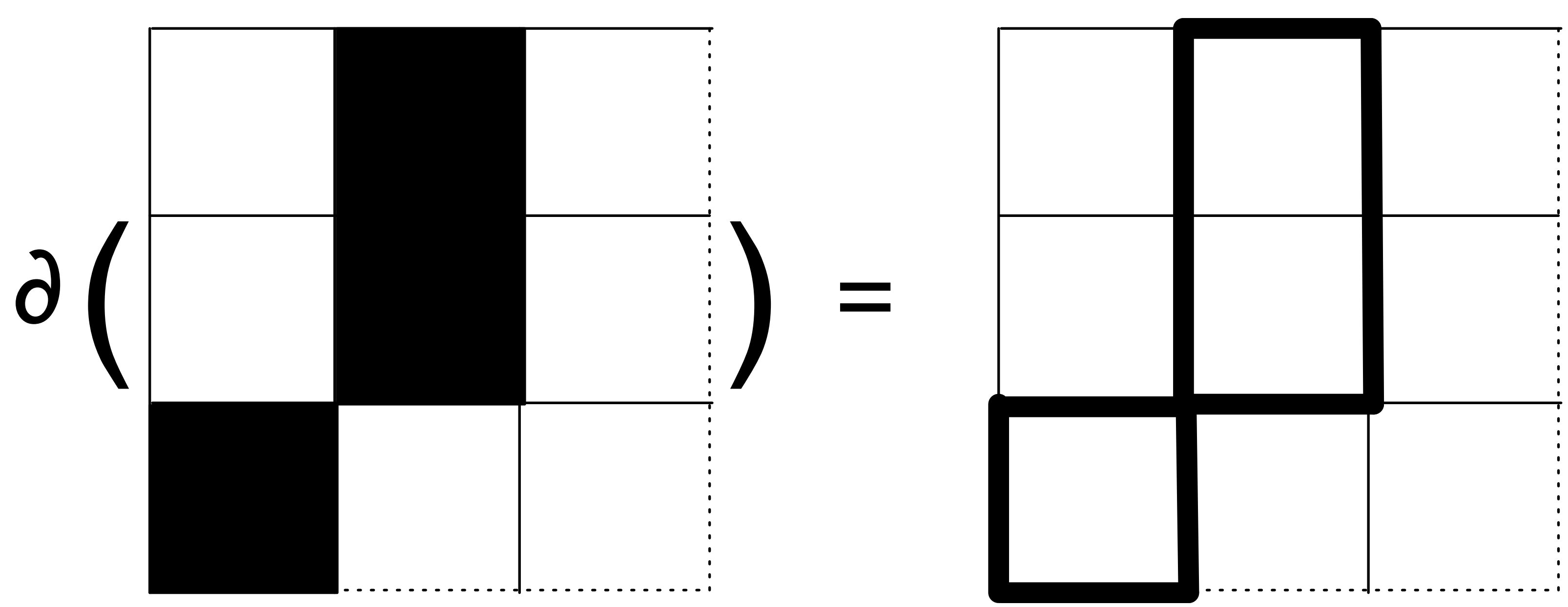
- Example - if we define a 2-cell's boundary map:

2-cell



$$\partial(a+b) = \partial(a) + \partial(b)$$

implies



Boundary

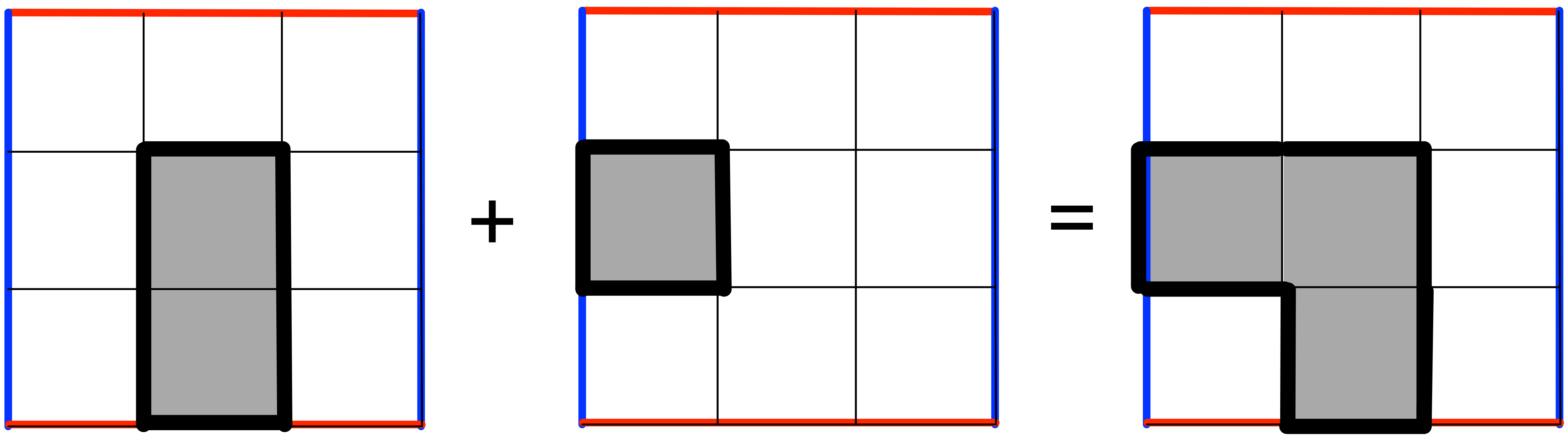
- **Terminology**: This structure of chain groups and boundary maps is called a **chain complex**.

E.g.

$$\underbrace{\text{2-chain group} \xrightarrow{\partial} \text{1-chain group} \xrightarrow{\partial} \text{0-chain group}}_{\textbf{chain complex}}$$

Boundary group

- The set of n-chains which are **boundaries** of (n+1)-chains form a **group -** a subgroup of the n-chain group**.**

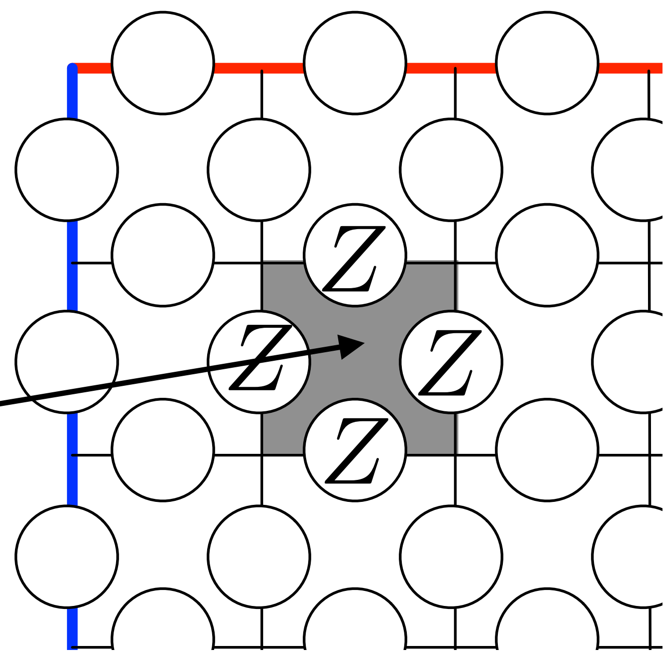


- We call this the **n-boundary group B_n**.

Boundary in the **Toric code**

- The **subgroup** of the **stabilizer** generated by the **plaquette** operators is in one-to-one correspondence with the **1-boundary group**.

Plaquette generator



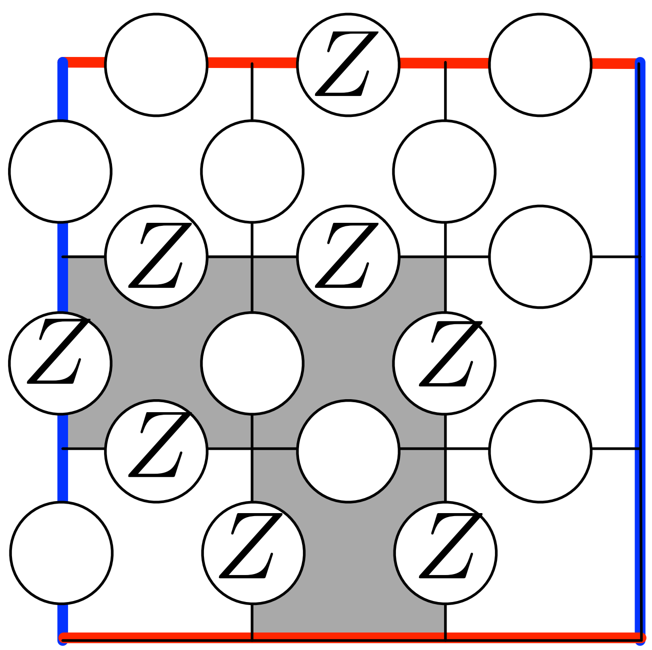
Plaquette operator: **Z(** ∂(p) **)**

Defined by **boundary** of the 2-cell (plaquette) p.

Generates the entire boundary group!

Boundary in the **Toric code**

- The **subgroup** of the **stabilizer** generated by the **plaquette** operators is in one-to-one correspondence with the **1-boundary group**.



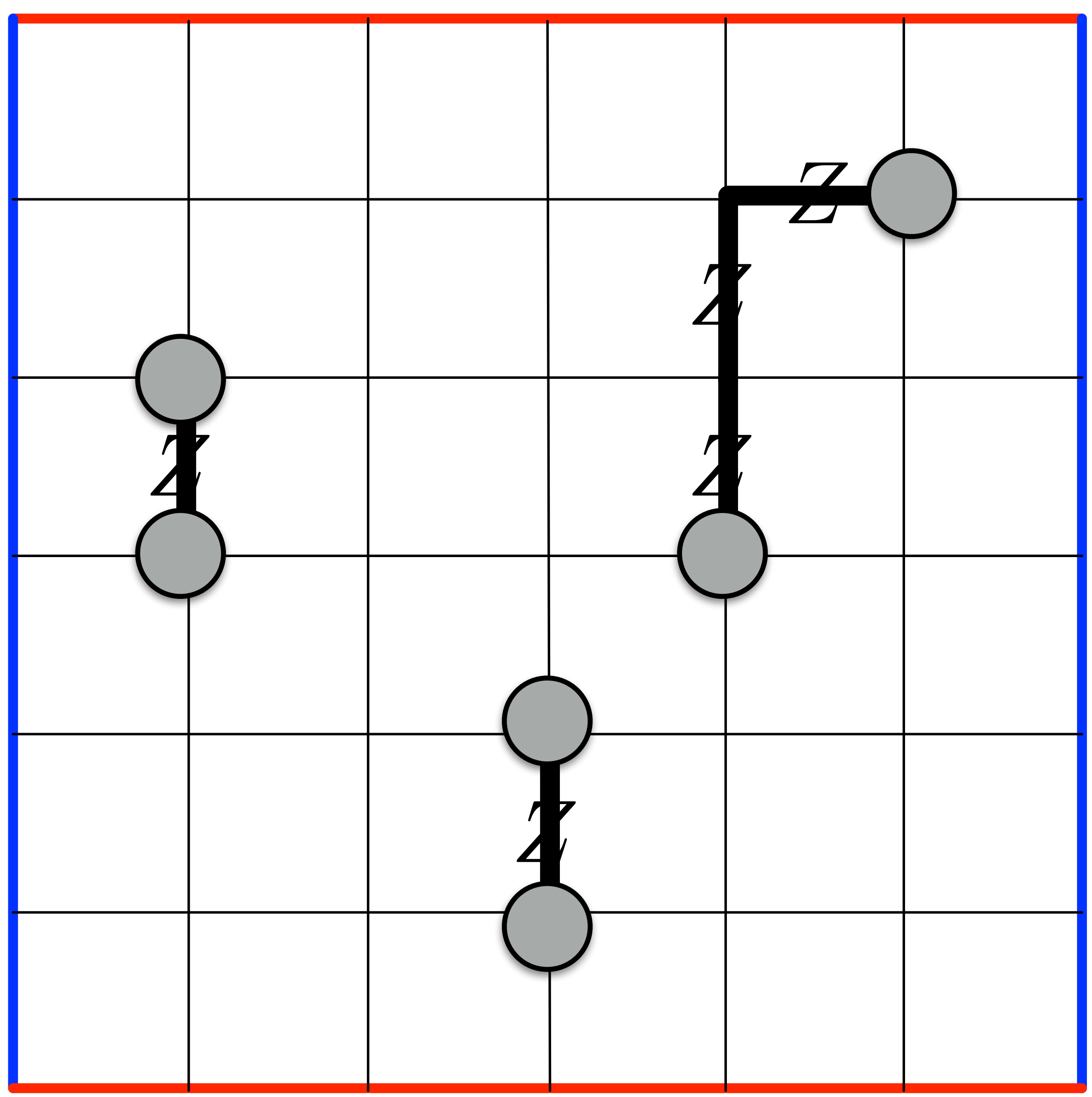
Plaquette operator: **Z(** ∂(p) **)**

Defined by **boundary** of the 2-cell (plaquette) p.

Generates the entire boundary group!

Boundary in the **Toric code**

- **Z-errors** are **detected** by **vertex** operator measurements.
- Can represent a set of **Z-errors** by a **1-chain**.
- The **syndrome** (vertex outcomes) corresponds precisely to its **boundary**.



vertex syndrome

=

∂ (Z-error 1-chain)

Cycles

Cycles

- The **null chain - 0**
  - Every chain group has an **identity** operator
  - This is the element with 0 at every cell

0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0

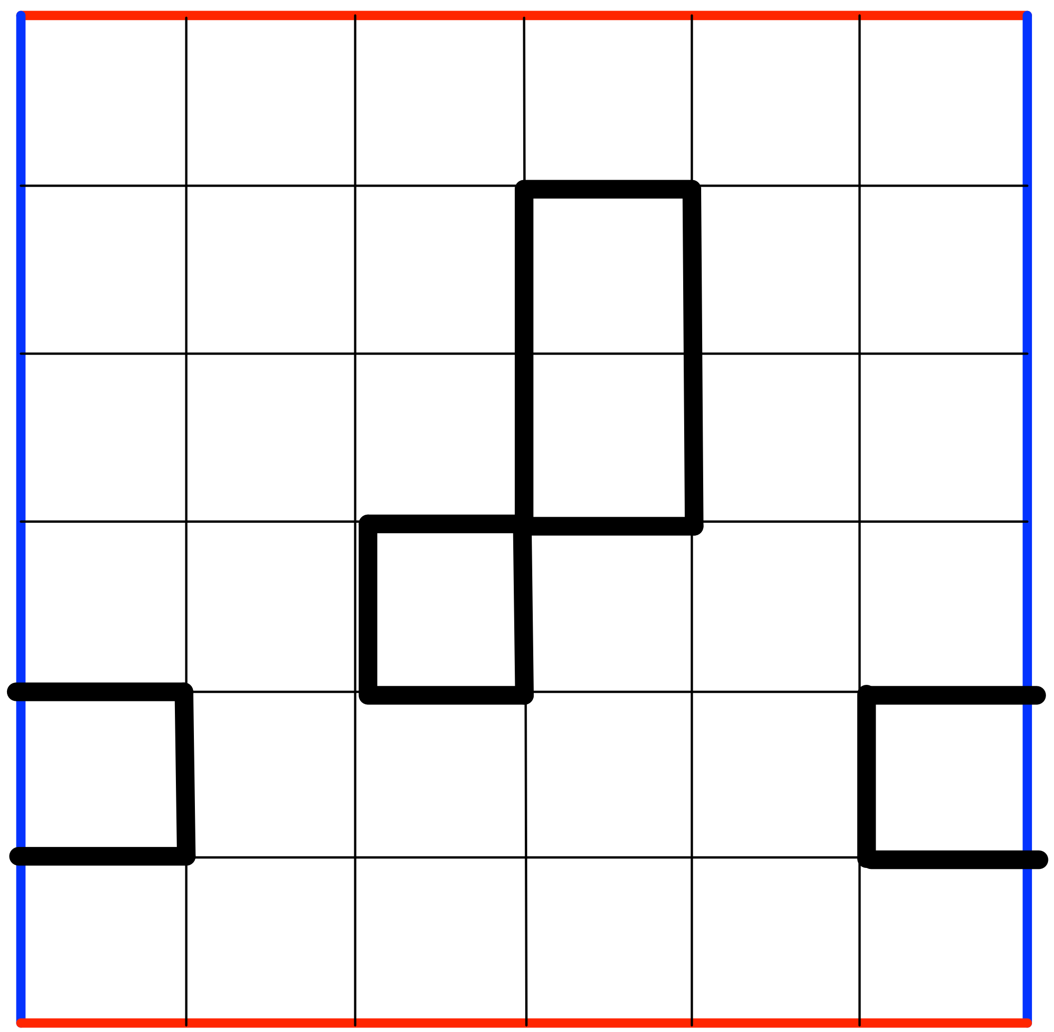
= **0**

Cycles

- *Definition:* A **cycle** is a chain whose **boundary** is the **null-chain**.

$$\partial(a) = 0$$

1-cycle

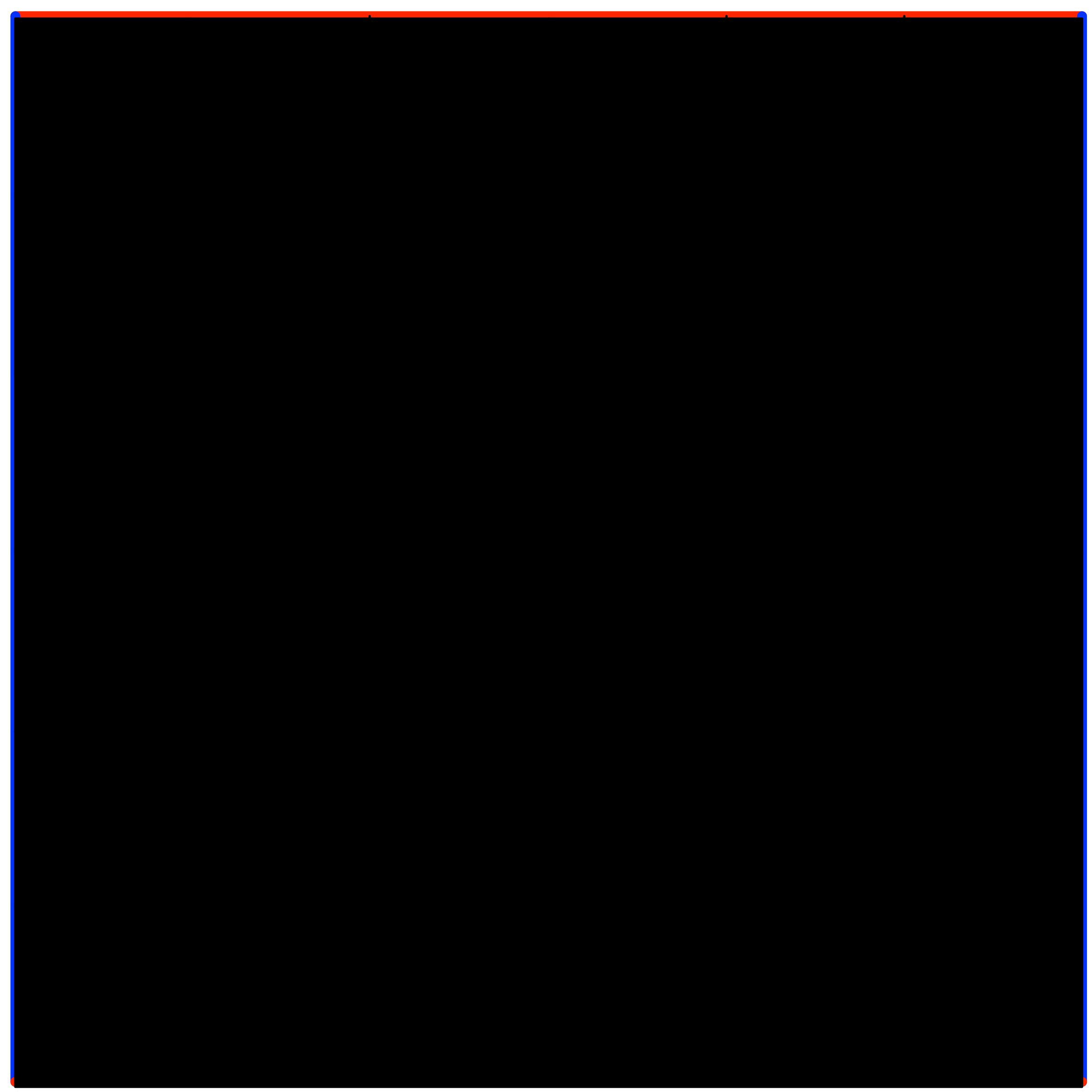


Cycles

- *Definition:* A **cycle** is a chain whose **boundary** is the **null-chain**.

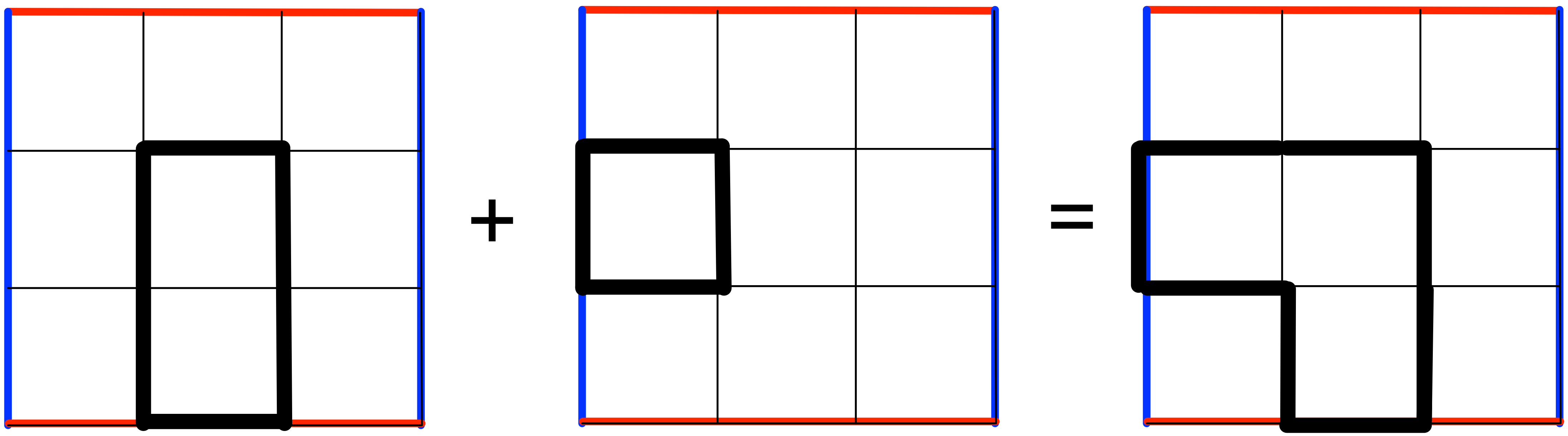
$$\partial(a) = 0$$

2-cycle



Cycle group

- Each set of n-**cycles** forms a **group**.



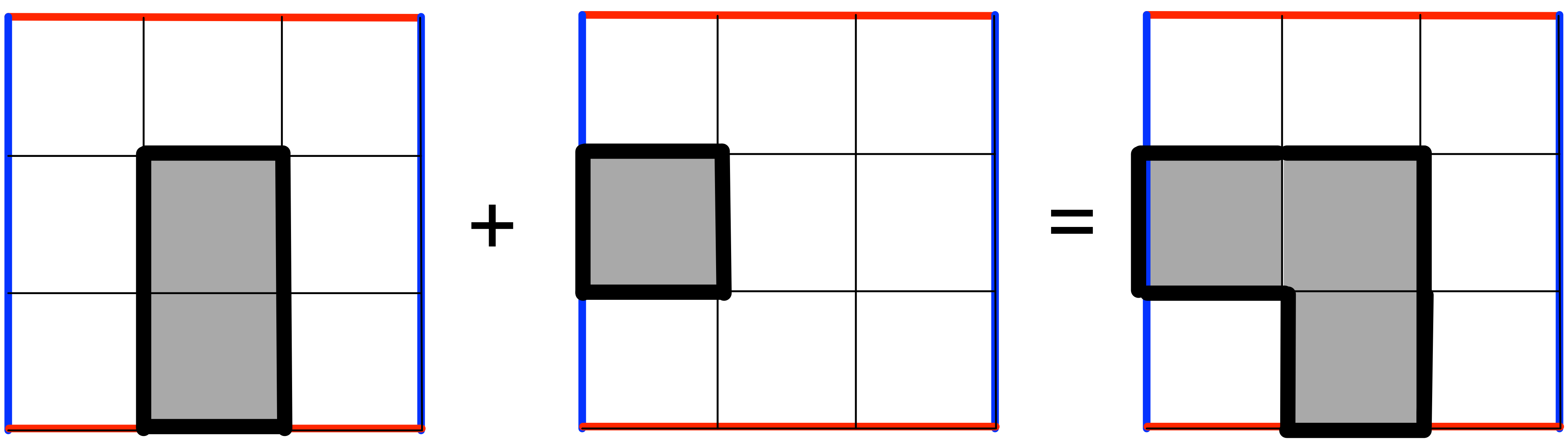
- We call this the **n-cycle group C_n**.

Cycle group

- This looks familiar.

Boundary group

- The set of n-**boundaries** form a **group.**



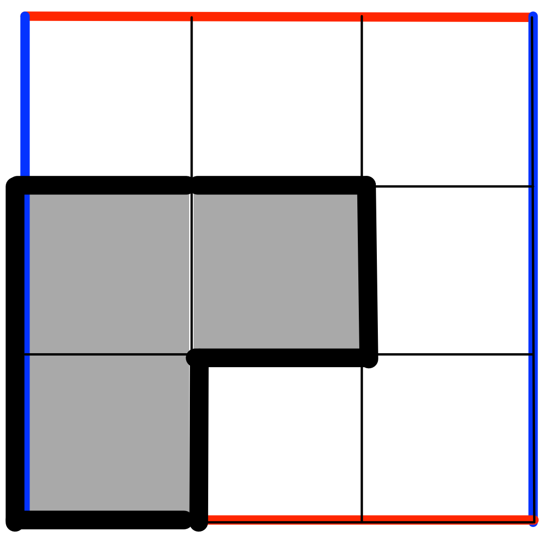
- We call this the **n-boundary group B_n**.

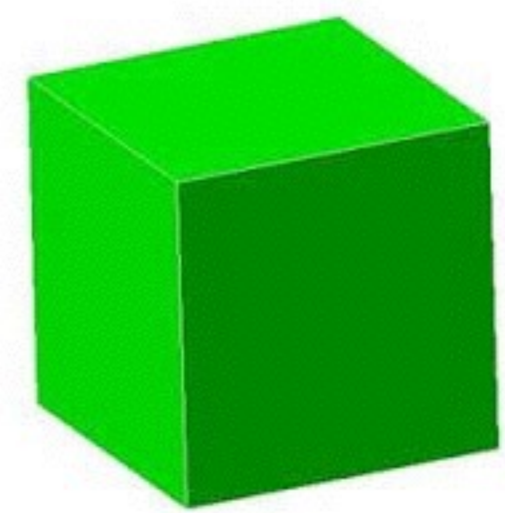
The central observations of homology

- Every **boundary** is a **cycle**.
- But **not every cycle** is a **boundary**.

Every boundary is a cycle

- In geometric homology, this is an observation, since a boundary, by definition, must be "closed".





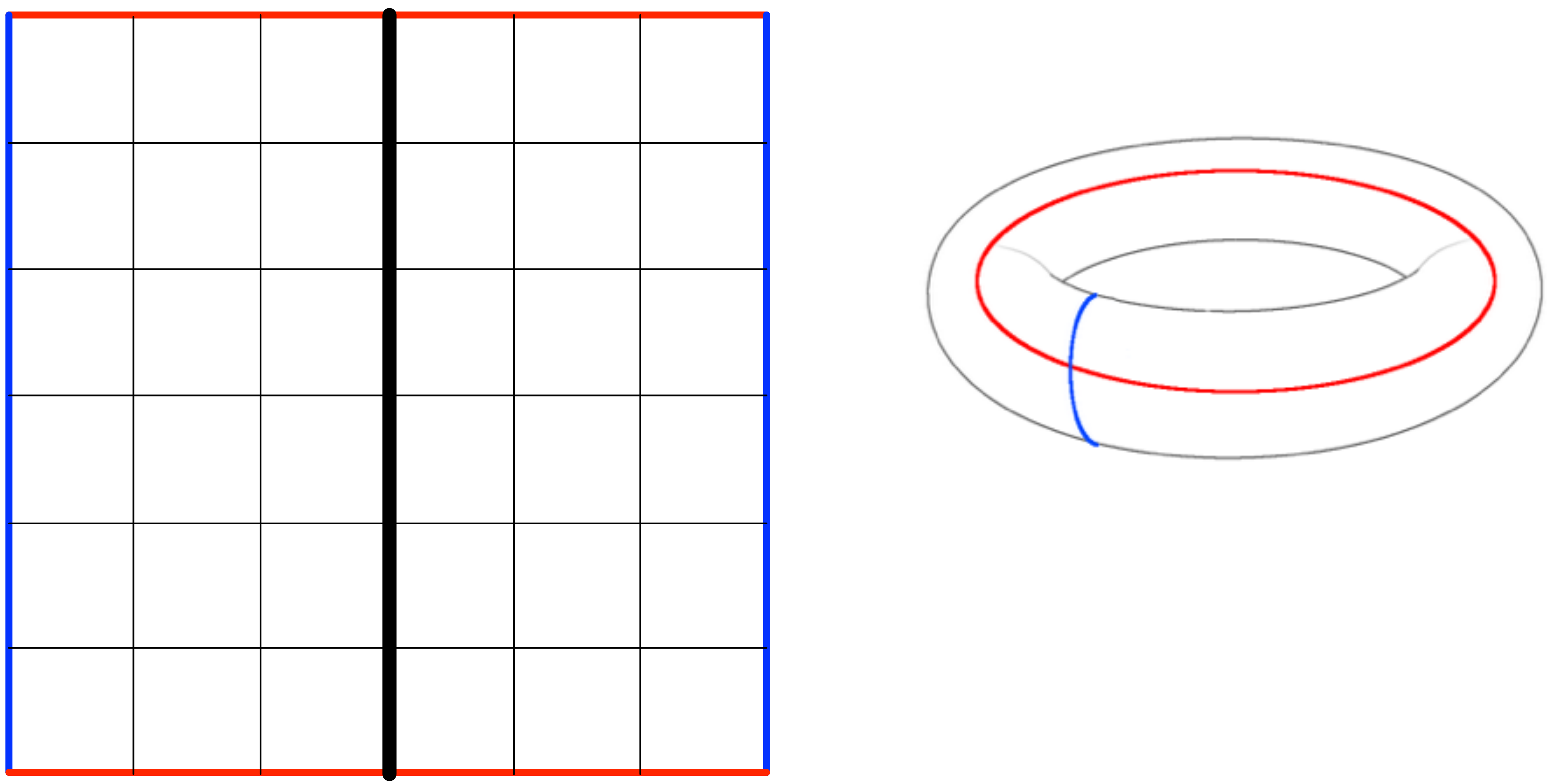
- In abstract homology, this becomes a **defining feature** of any boundary map ∂.

$$\partial^2 = 0$$

starting point for abstract homology

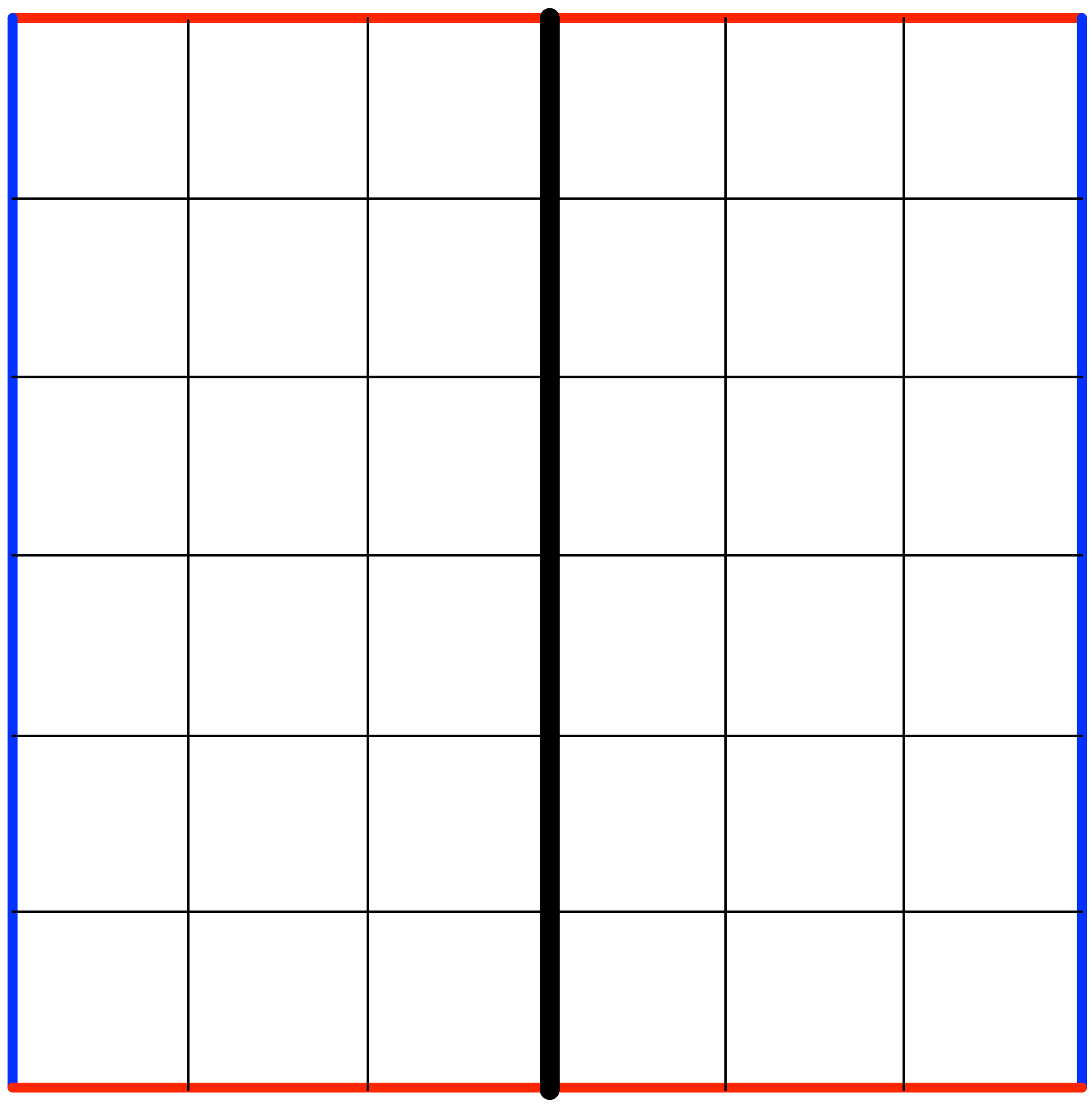
Not every cycle is a boundary

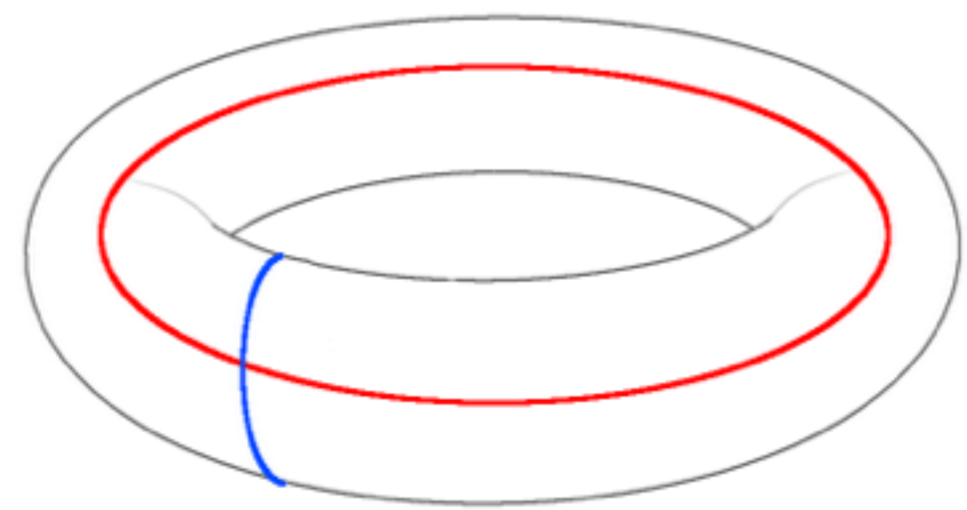
- Consider the following 1-chain on a torus:



Not every cycle is a boundary

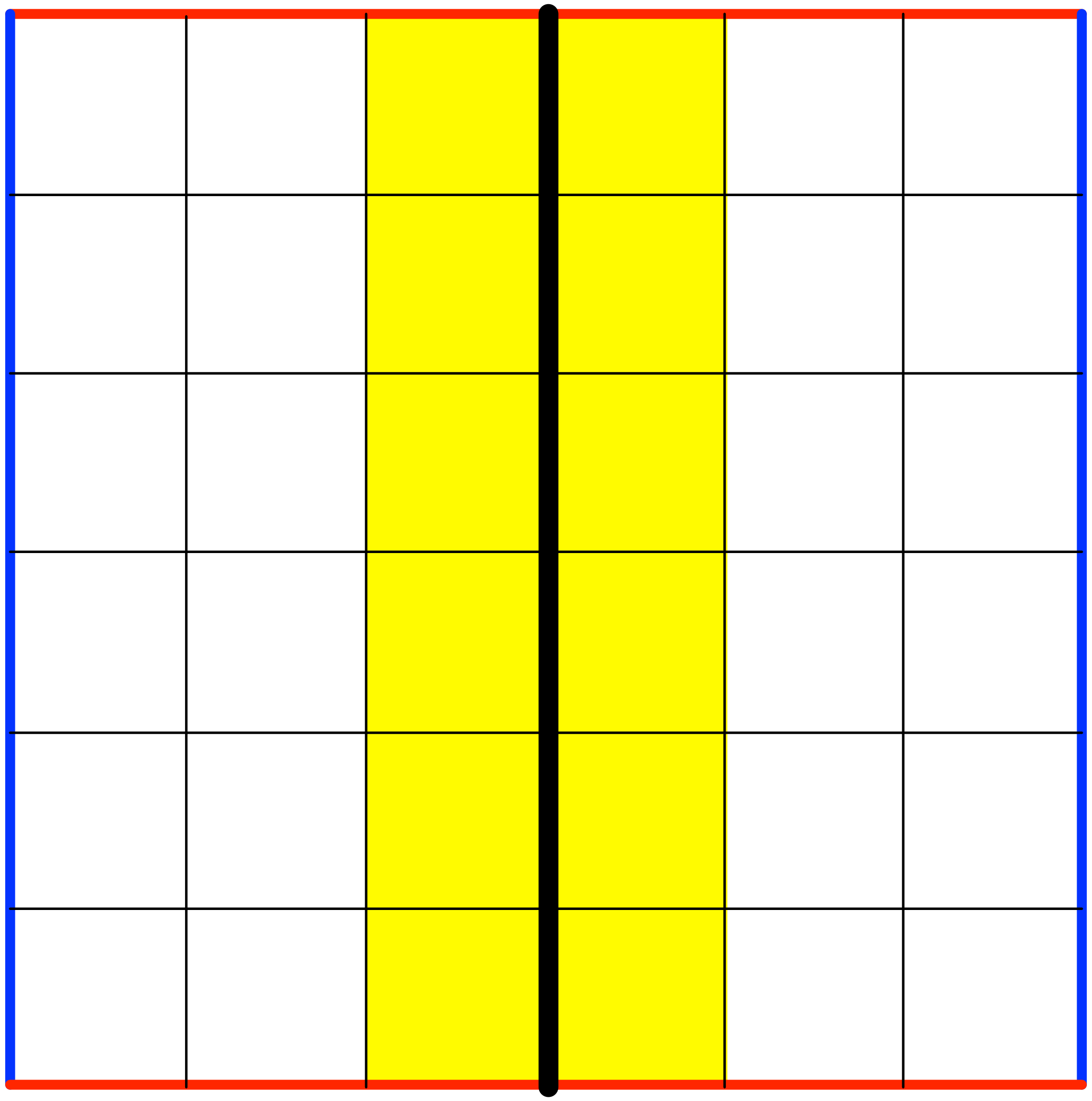
- It has **null boundary** (no ends), and hence is a **cycle**.





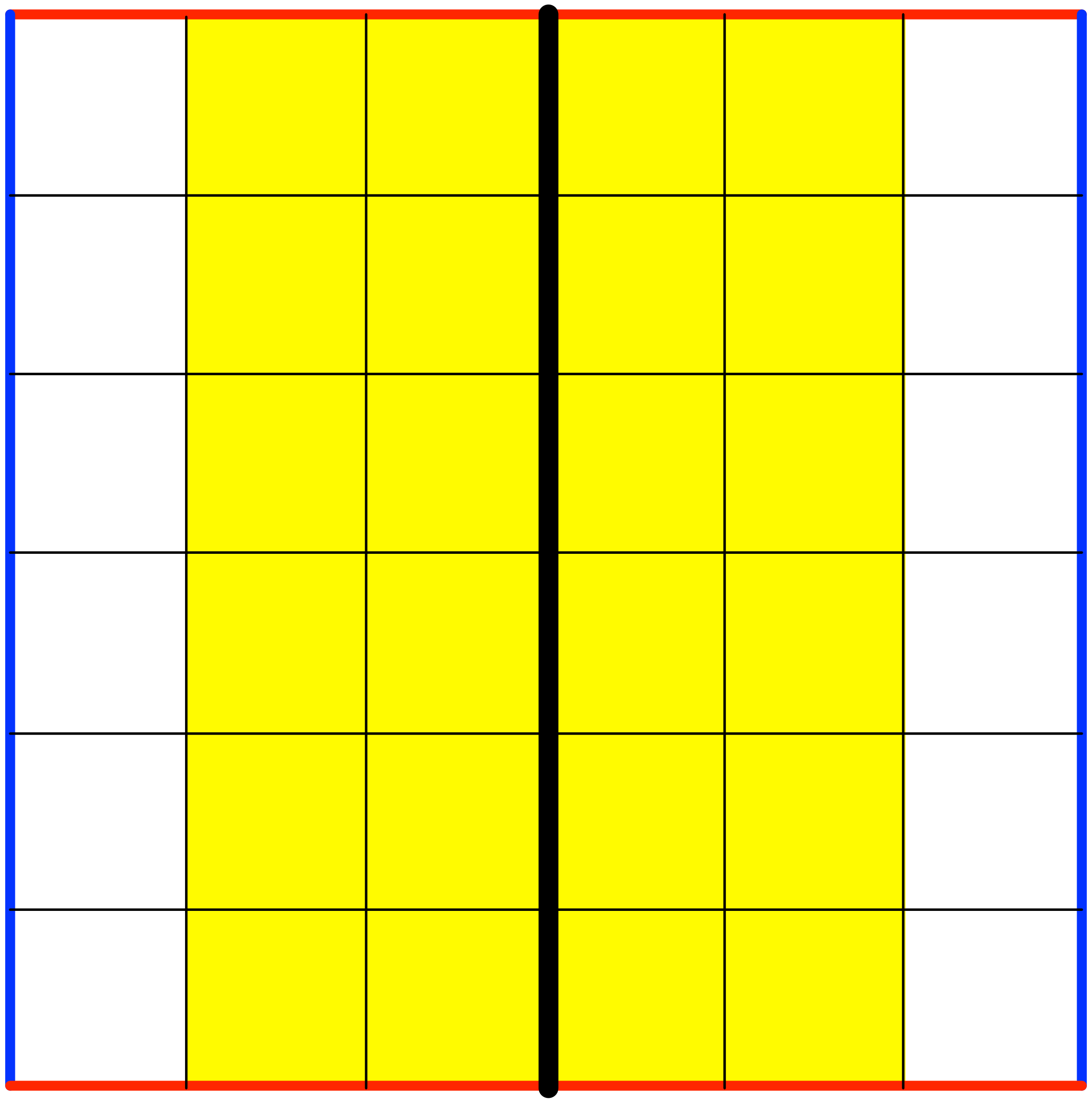
Not every cycle is a boundary

- But if we try and use it to enclose a finite area…



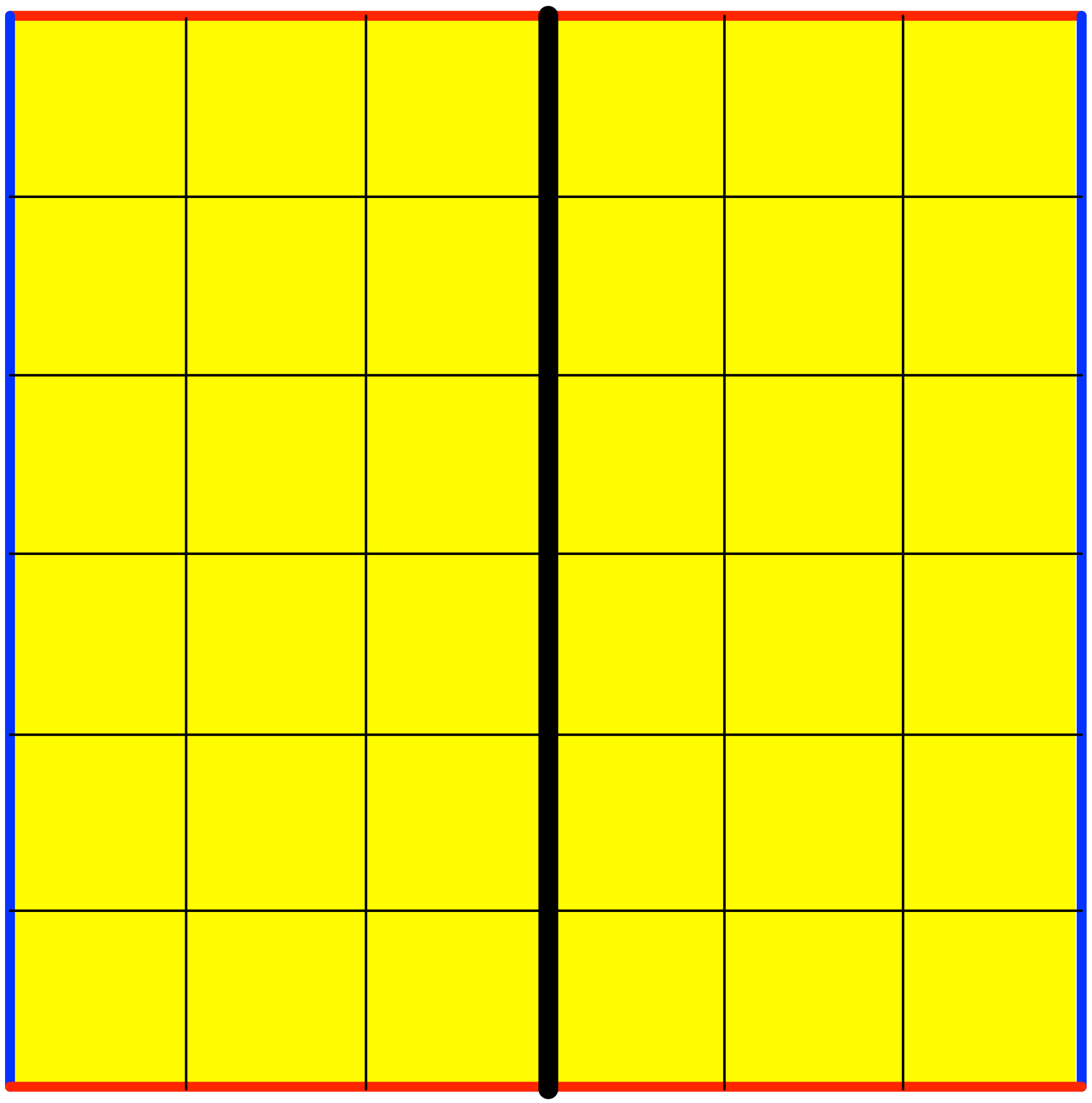
Not every cycle is a boundary

- But if we try and use it to enclose a finite area…



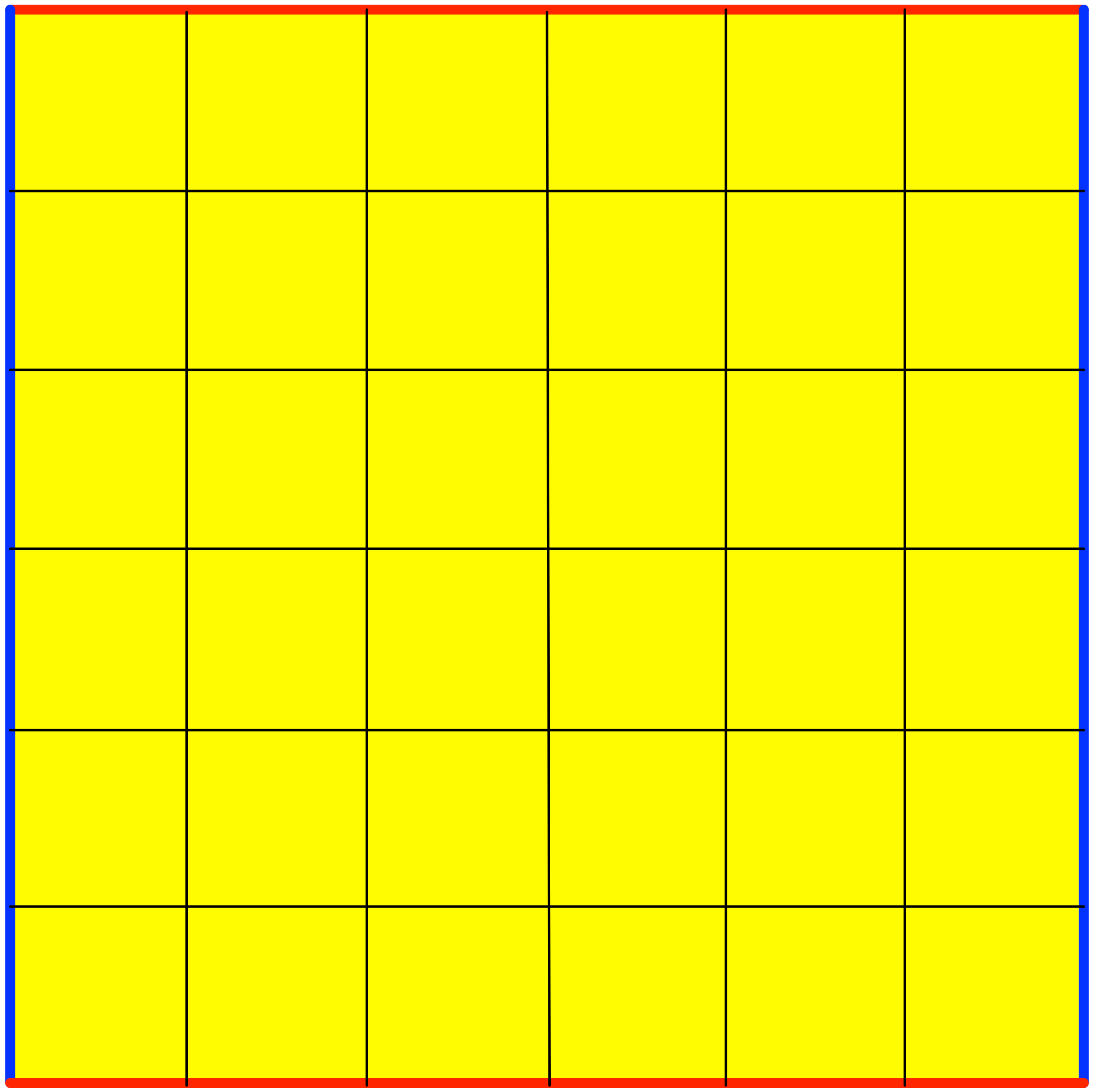
Not every cycle is a boundary

- ..we cover the **whole torus**….



Not every cycle is a boundary

- ..which is a 2-chain with **no** boundary.



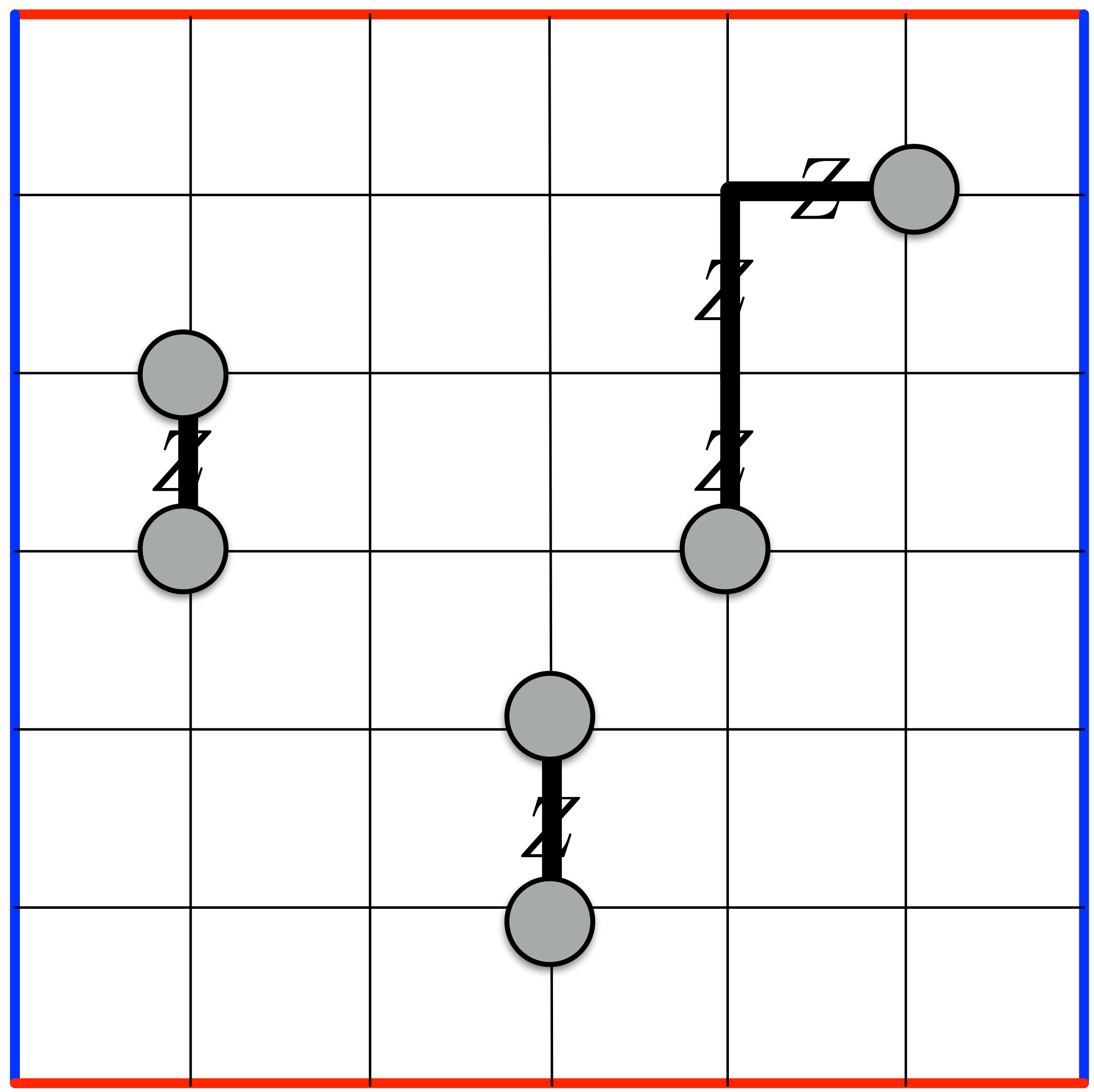
Cycles in the **Toric code**

- Recall that:

vertex syndrome

=

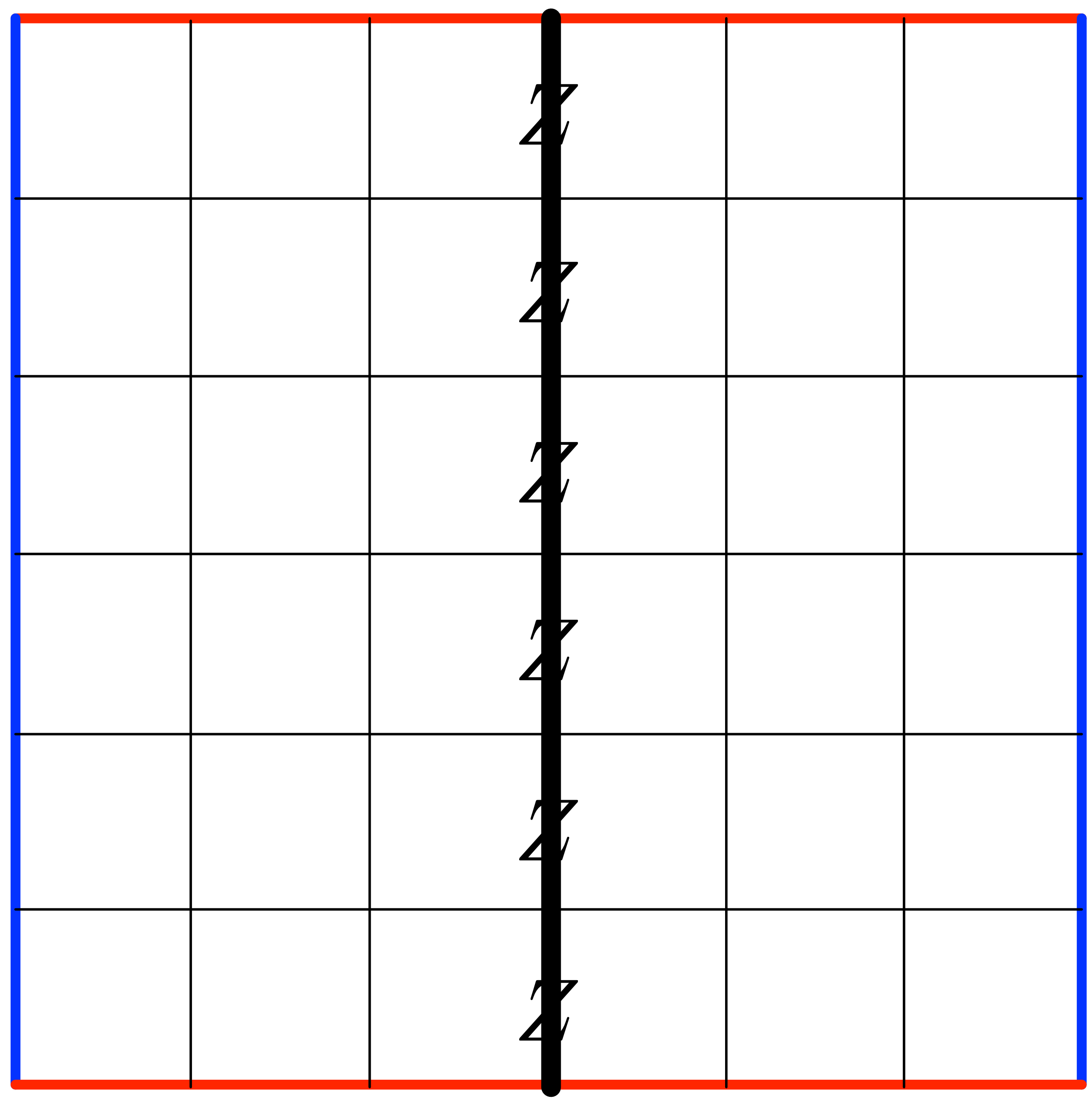
∂ (Z-error 1-chain)



Cycles in the **Toric code**

- Thus if **c** is a **1-cycle**, the operator **Z(c) commutes** with **all** vertex operators, and hence the entire stabilizer.
- Thus **1-cycles** represent **logical operators** on the toric code.

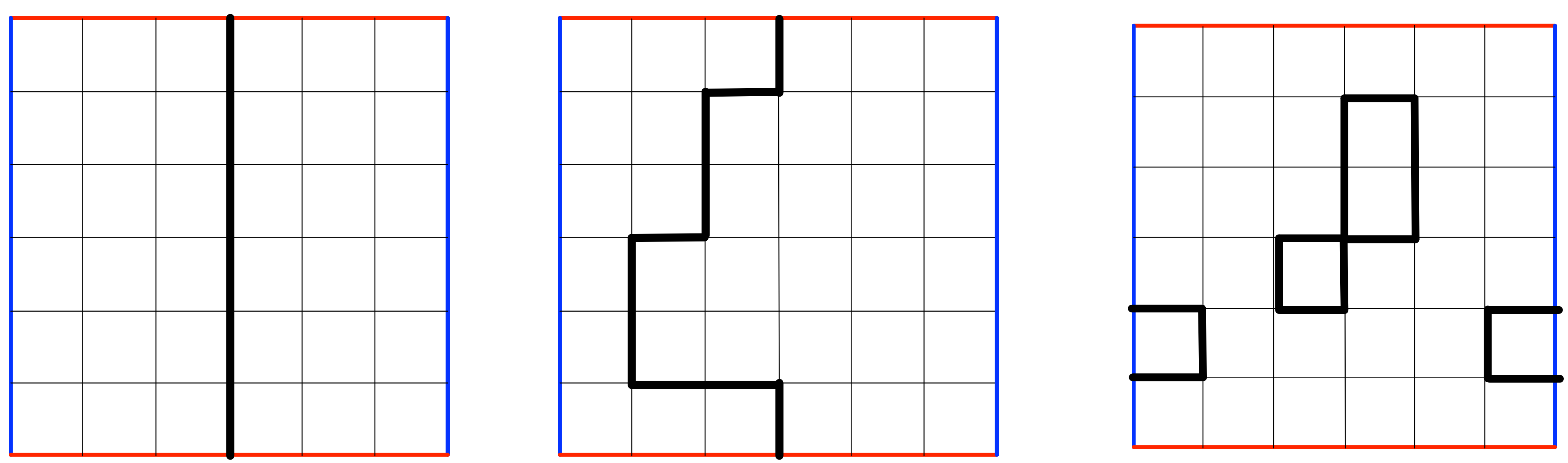
E.g.



Homological equivalence

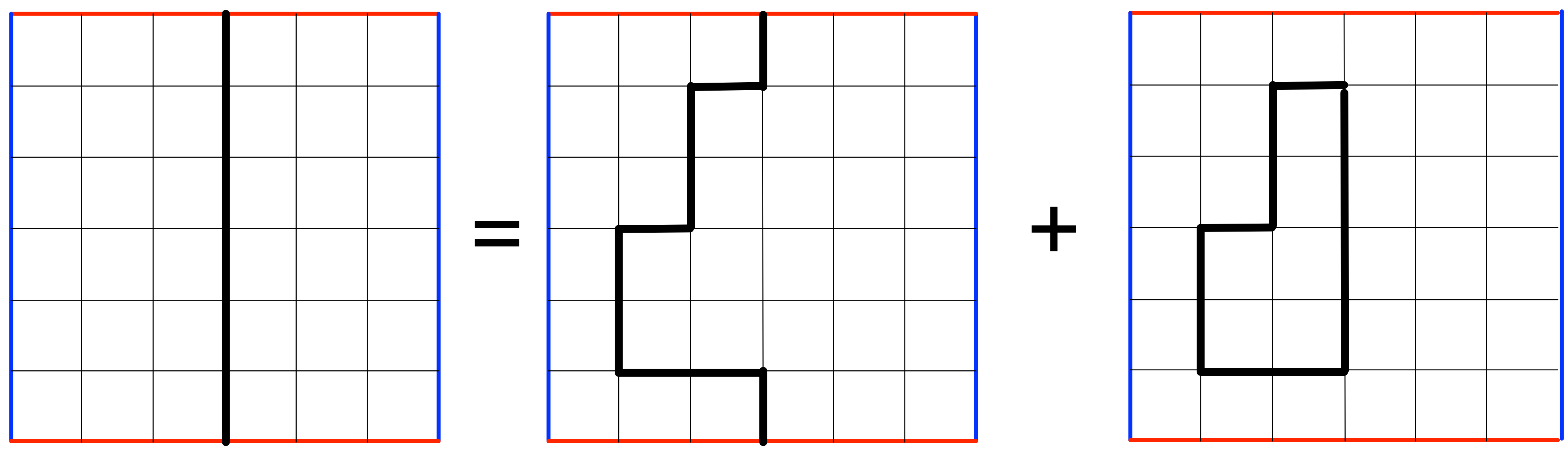
Homological equivalence

- Some cycles are boundaries, some not.
- This is one notion of **equivalence**.
- **Homological equivalence** is **stronger** (and more **useful**).



Homological equivalence

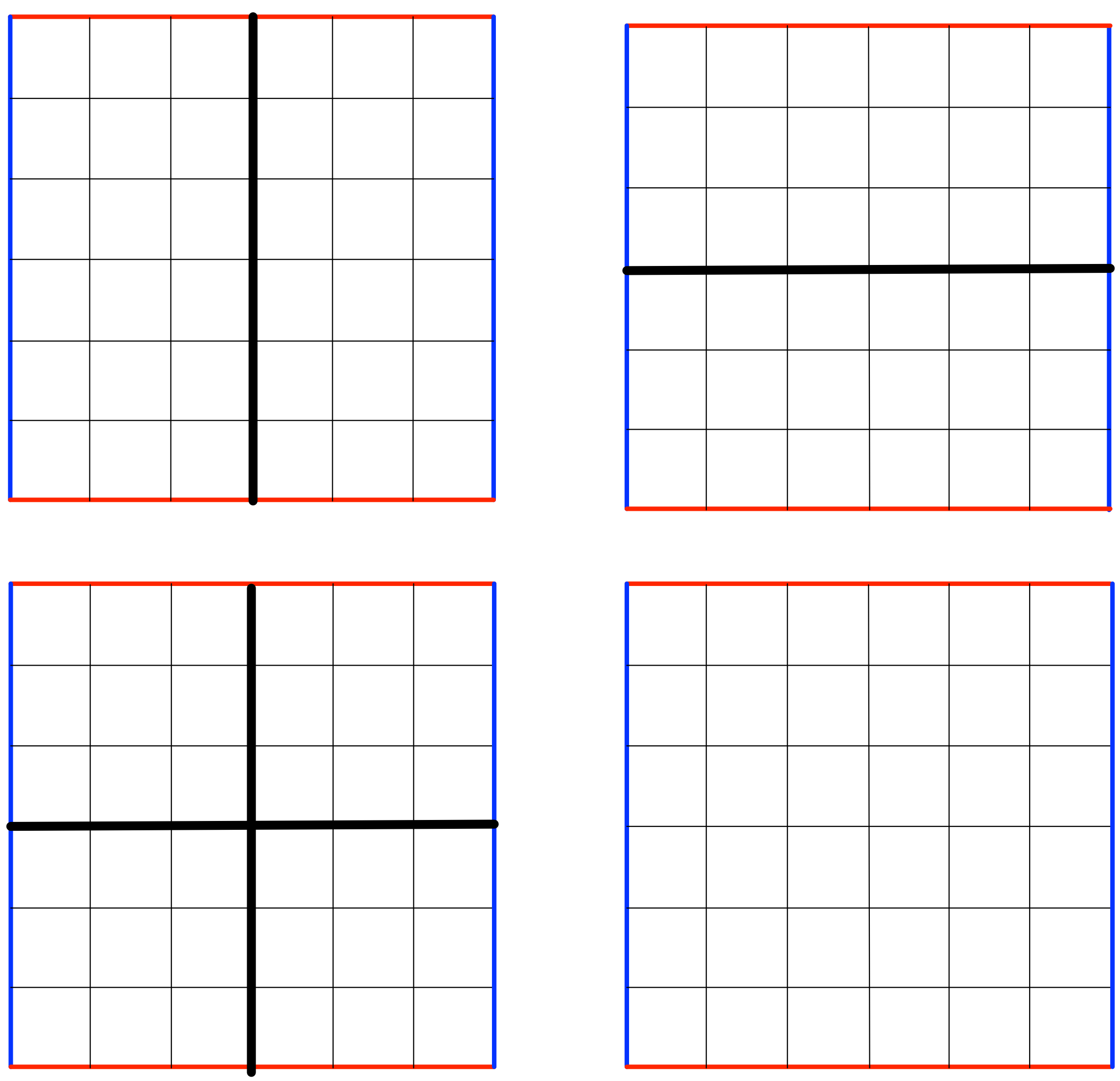
- *Definition:* Two chains **c** and **d** are **homologically equivalent** if **c** = **d + e,** where **e** is a **boundary.**

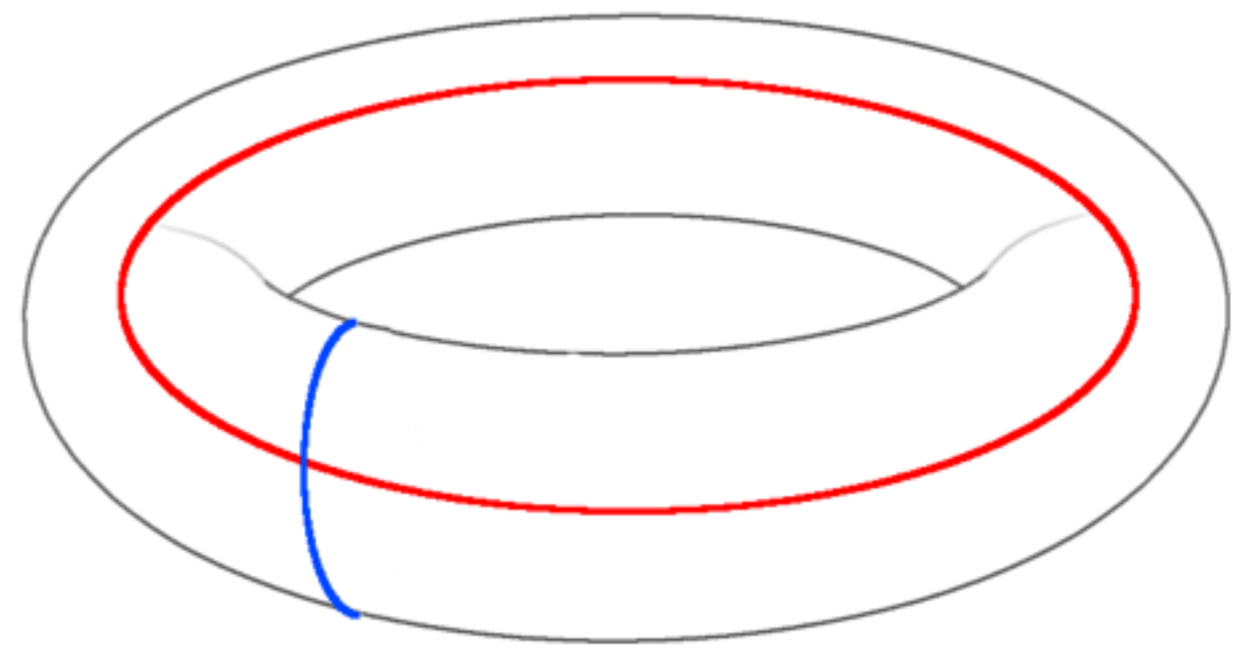


- *I.e.* ***homologically equivalent*** *chains are* ***equal*** *up to the* ***addition of a boundary.***

Homological equivalence

- A very natural notion of equivalence in homological terms
- On the torus, 4 equivalence classes:





These classes form a **group** isomorphic to $\mathbf{Z_2 x Z_2}$ (2 bit group)

Homology group

- *Definition:* The ***n***-th **homology group** is the quotient group

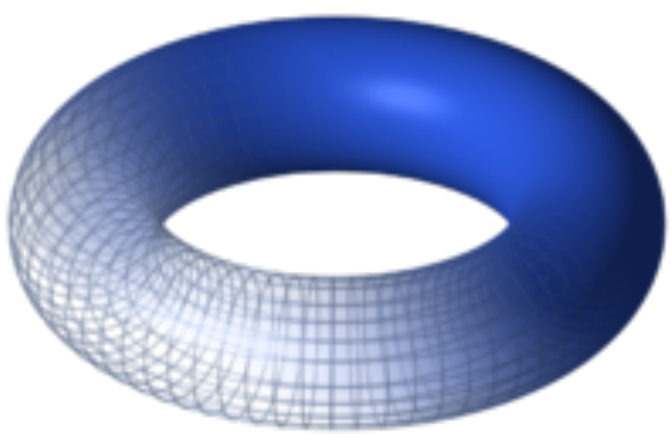
$$\frac{C_n}{B_n}$$

the homological equivalence classes of n-cycles.

- Homology groups capture **topological properties** of a surface.
- They are **independent** of the **cellulation** used.

Homology group

- E.g. The first homology group counts “handles” in a surface.



$\mathbb{Z}_2 \times \mathbb{Z}_2$

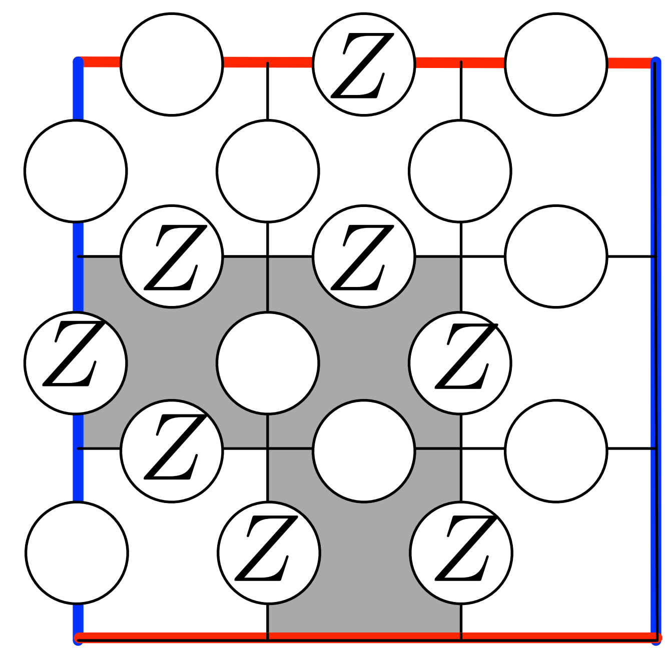
$(\mathbb{Z}_2 \times \mathbb{Z}_2)^2$

$(\mathbb{Z}_2 \times \mathbb{Z}_2)^3$

Homological equivalence in the **Toric Code**

- Homological equivalence = equivalence up to (addition of) a **boundary**.

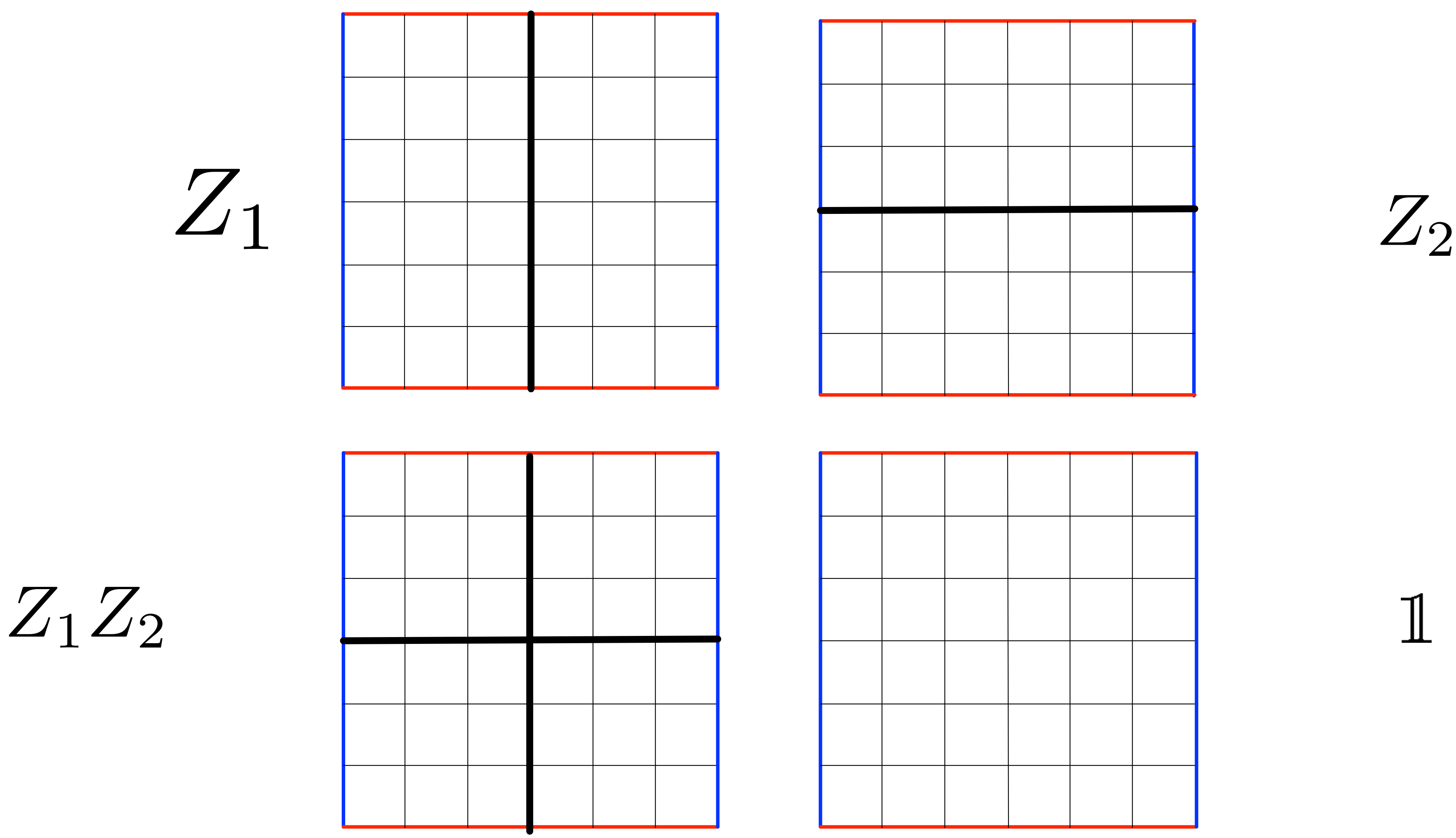
- The **1-boundary group** corresp. to the **Z-subgroup** of the **stabilizer**.



- **Homological equivalence** of 1-chains
  = **equivalence** under **Z-stabilizer** multiplication
  = **equivalence** on **code-space** (for Z-only Pauli operators.)

Homology Groups in the **Toric Code**

- **1st homology group** defines inequivalent **logical Z** operators



Homology in the toric code

- We have now covered the key concepts of **$\mathbb{Z}_2$ homology**.
  - ✤chains
  - ✤boundaries
  - ✤cycles
  - ✤homological equivalence
  - ✤homology groups

- Each plays an **important role** in the toric code.
- Properties of Z-stabilizers and Z-errors are **fully described**.

Homology in the toric code

How can we **complete the picture** and fully include **X-errors**?

Cohomology

Cohomology

- **Cohomology** is to **homology** as **bras** are to **kets**.

$\langle\phi|$: $|\psi\rangle \to \langle\phi|\psi\rangle \in \mathbb{C}$

linear functional

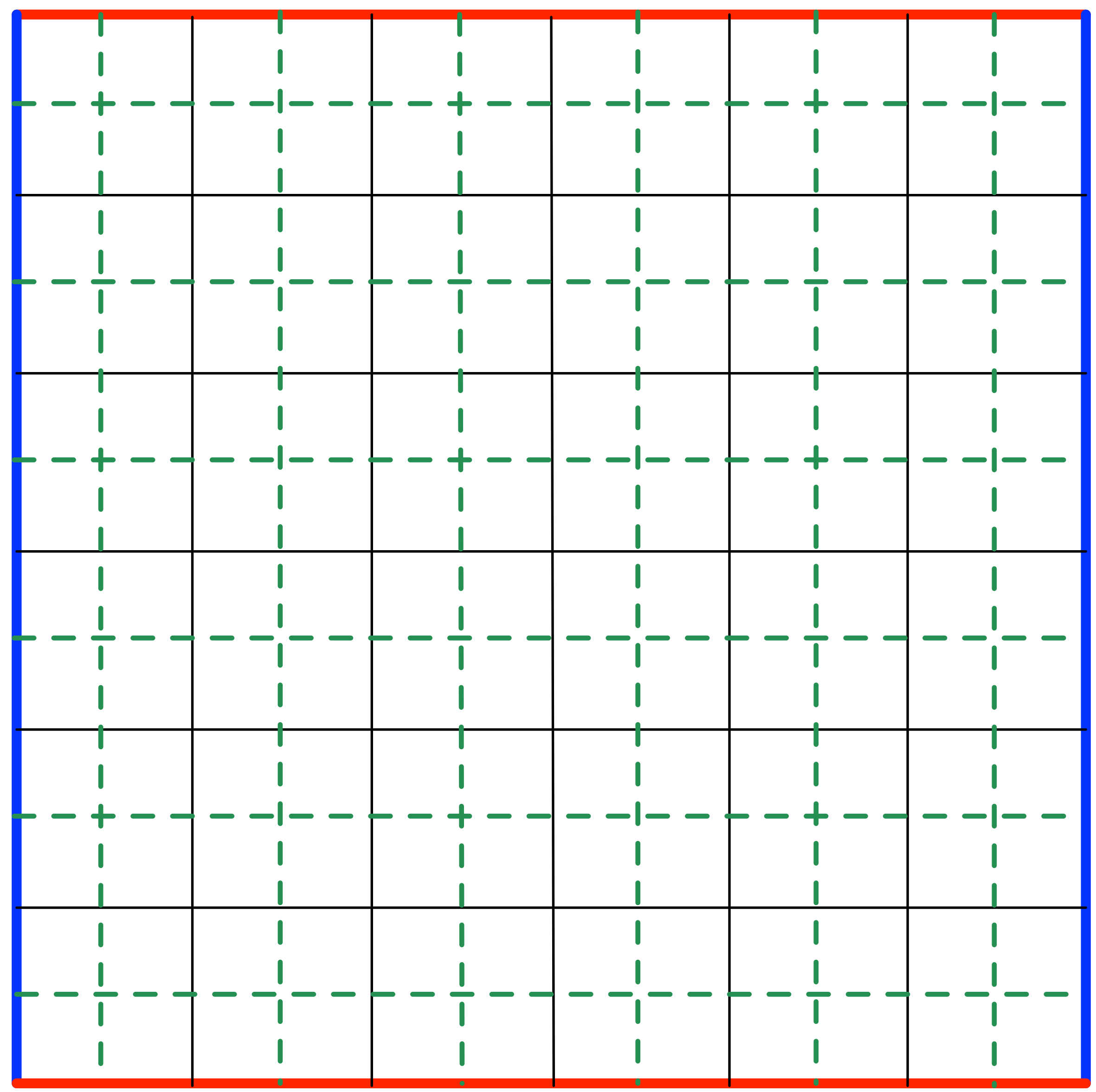
$\text{co-}n\text{-chain}:$ $n\text{-chain} \to \langle \text{co-}n\text{-chain}, n\text{-chain}\rangle \in \mathbb{Z}_2$

This **dual** construction provides:

- co-chains *(c.f. "bras" to chains "kets")*
- co-boundaries *(c.f. "dagger" of operators)*
- co-cycles
- co-homological equivalence
- co-homology groups

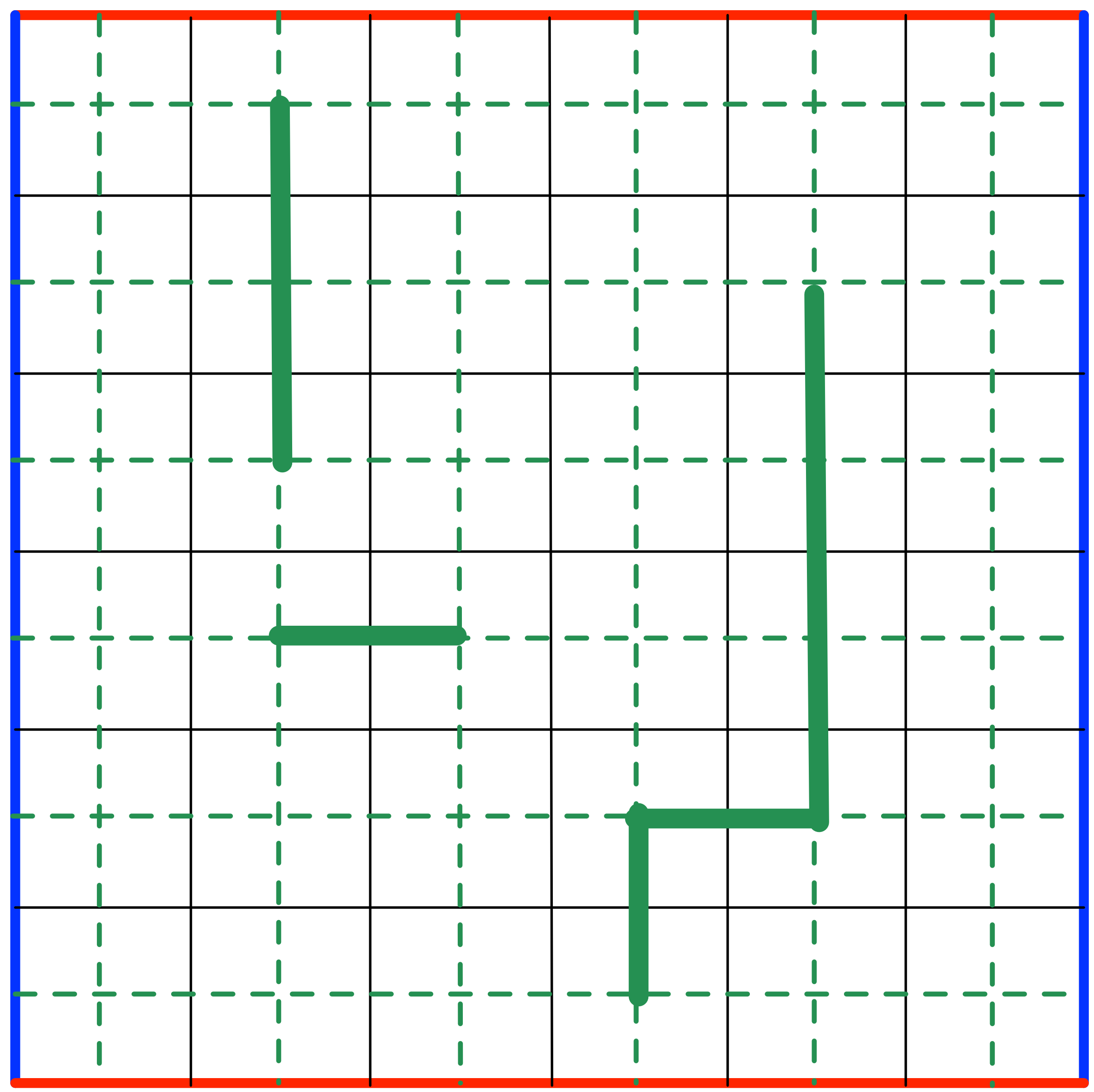
Cohomology

- In cellular homology, **co-homology** can be represented on the **dual lattice.**



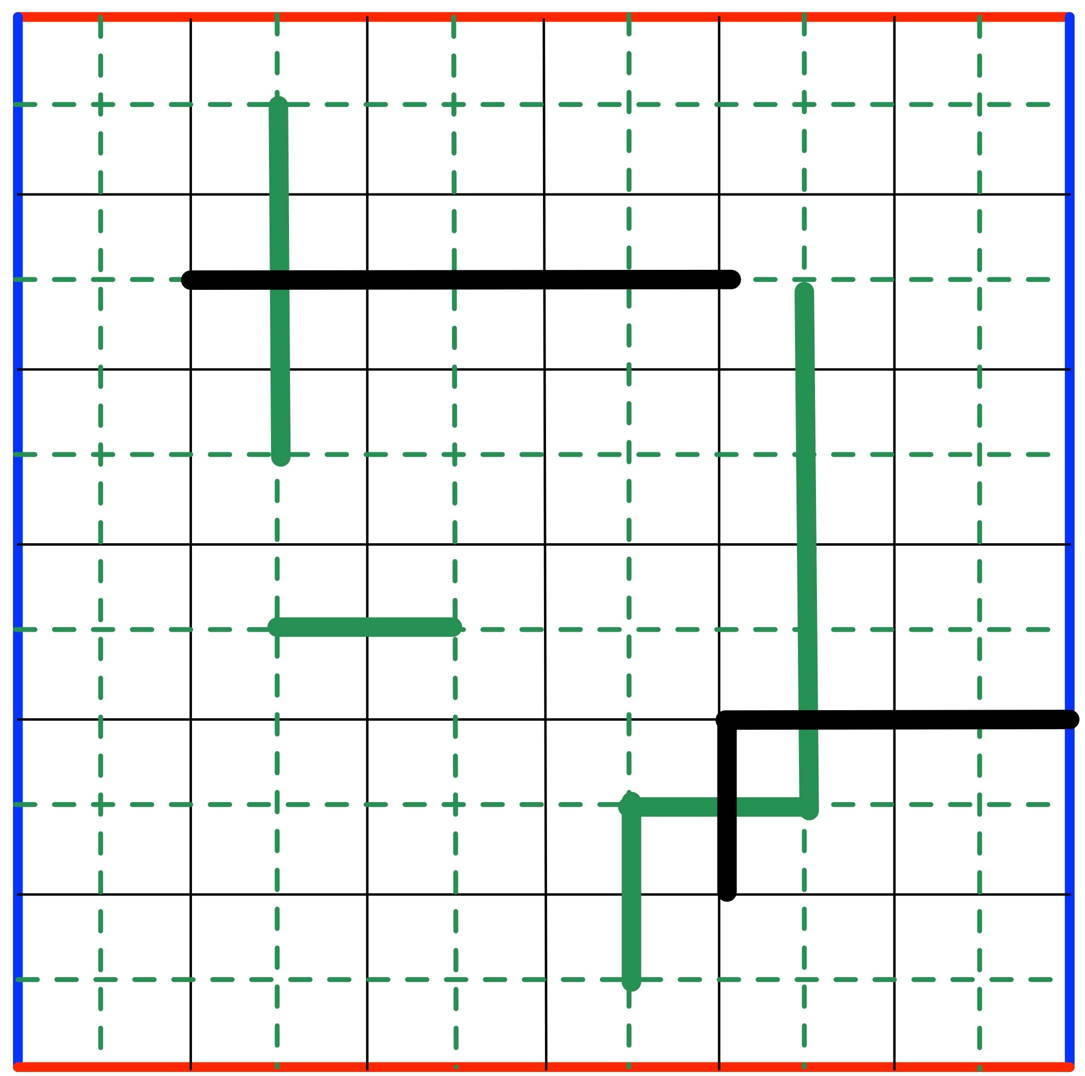
Cohomology

- E.g. in 2D, **1-cochains** are assignments of Z_2 to **edges** on the **dual lattice…**



Cohomology

- with a **“scalar product”** with **1-chains** defined:
  < , > = number of **crossings** with 1-chain, modulo 2



- E.g. here < *co-1-chain*, *chain* > = 1

Cohomology in the **Toric code**

- The roles played by
    - ✤chains
    - ✤boundaries
    - ✤cycles
    - ✤homological equivalence
    - ✤homology groups
- for Z-stabilizers and Z-errors….

Cohomology in the **Toric code**

- are played by

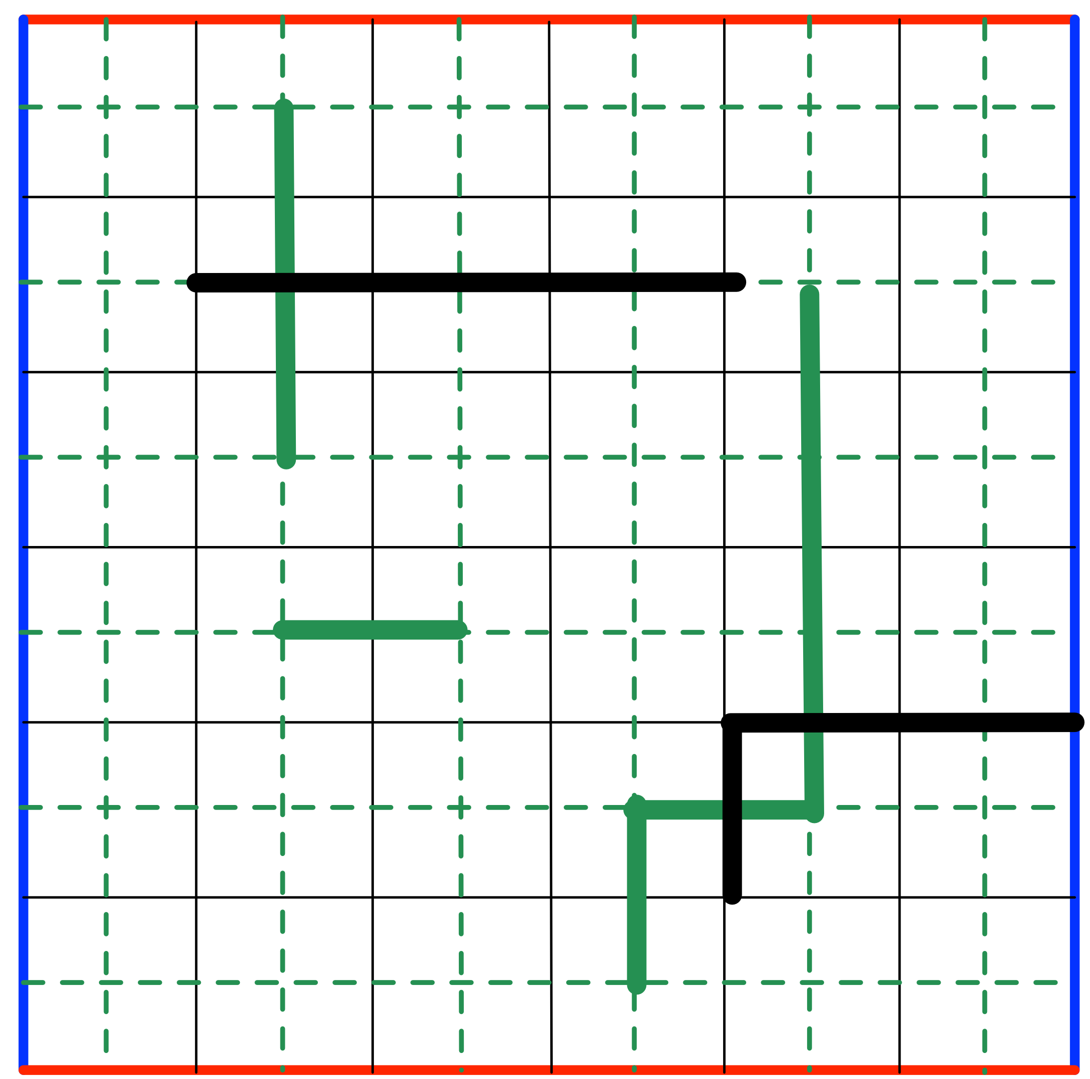
  - co-chains
  - co-boundaries
  - co-cycles
  - co-homological equivalence
  - co-homology groups

- for X-stabilizers and X-errors….

Cohomology in the **Toric code**

- **Z operators** identified with **1-chains**.
- **X operators** identified with **1-cochains**

Operator **commutation** is fully described by the **scalar product** between chain and cochain.



$$Z[a]X[b] = (-1)^{\langle b,a \rangle} X[b]Z[a]$$

Stabilizer code **commutation rules** encoded **homologically**!

Homological codes

Homological codes

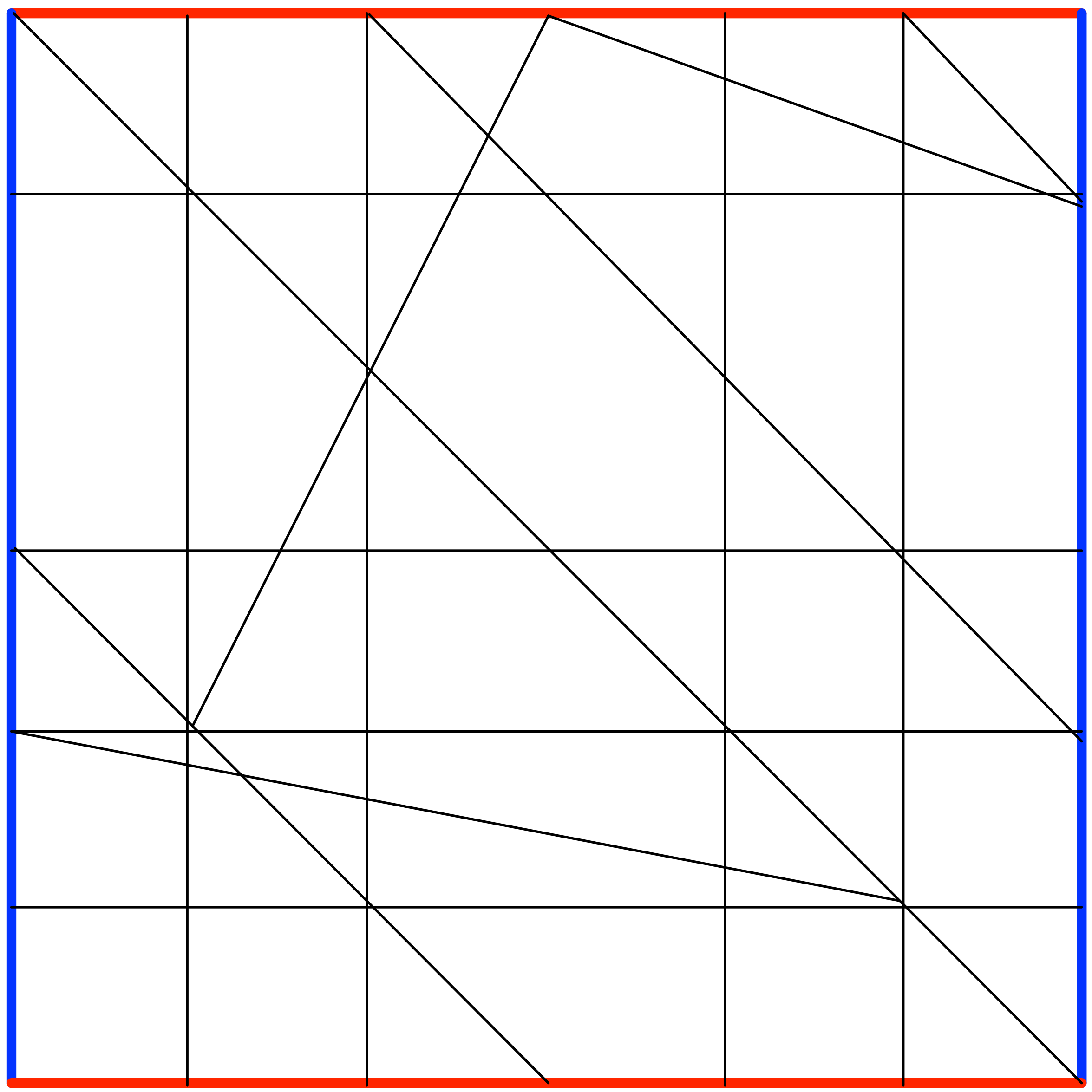
- **Every feature** of the **toric code** can be described **homologically**.
- Homology can be applied to a **wide variety** of **topological spaces.**

Homological codes

- **Surface code on generalised torii**

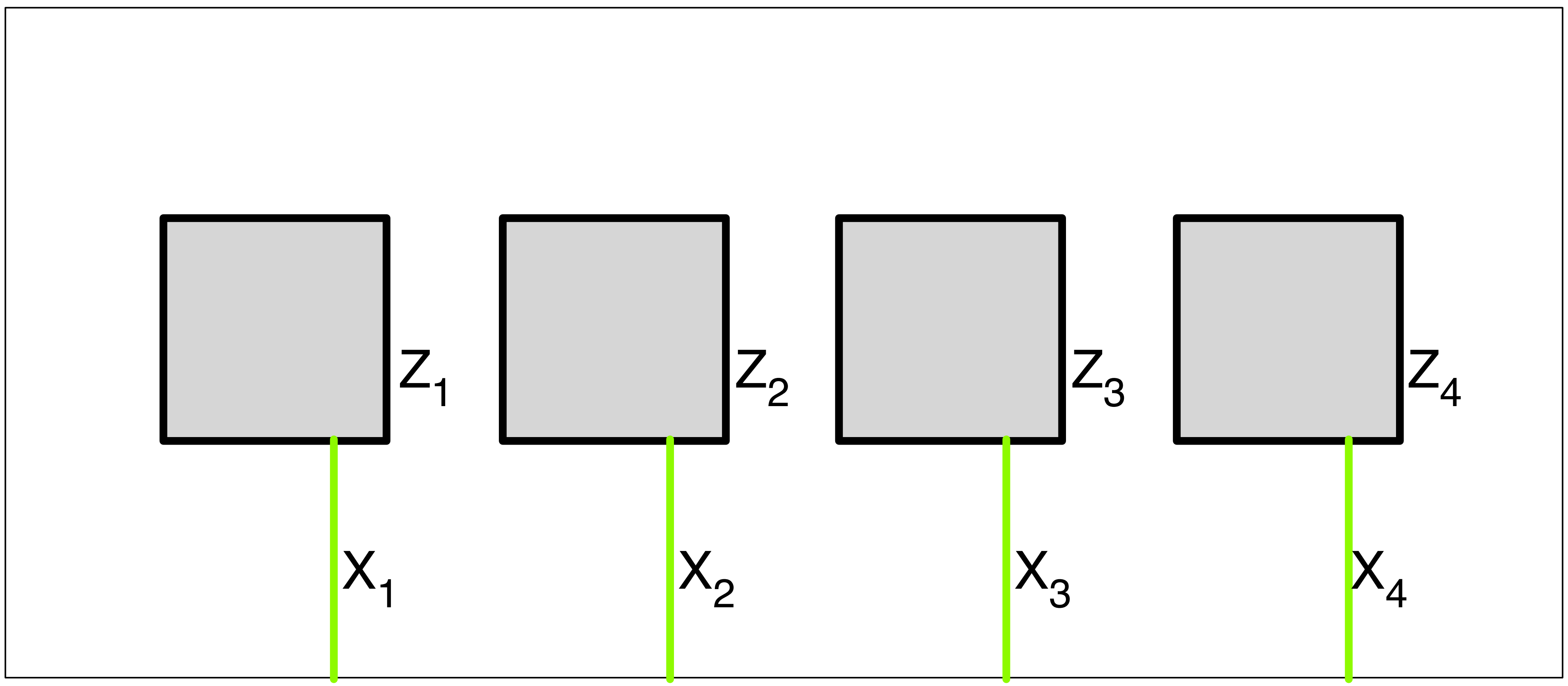
Homological codes

- **Toric code on a non-square cellulation**



Homological codes

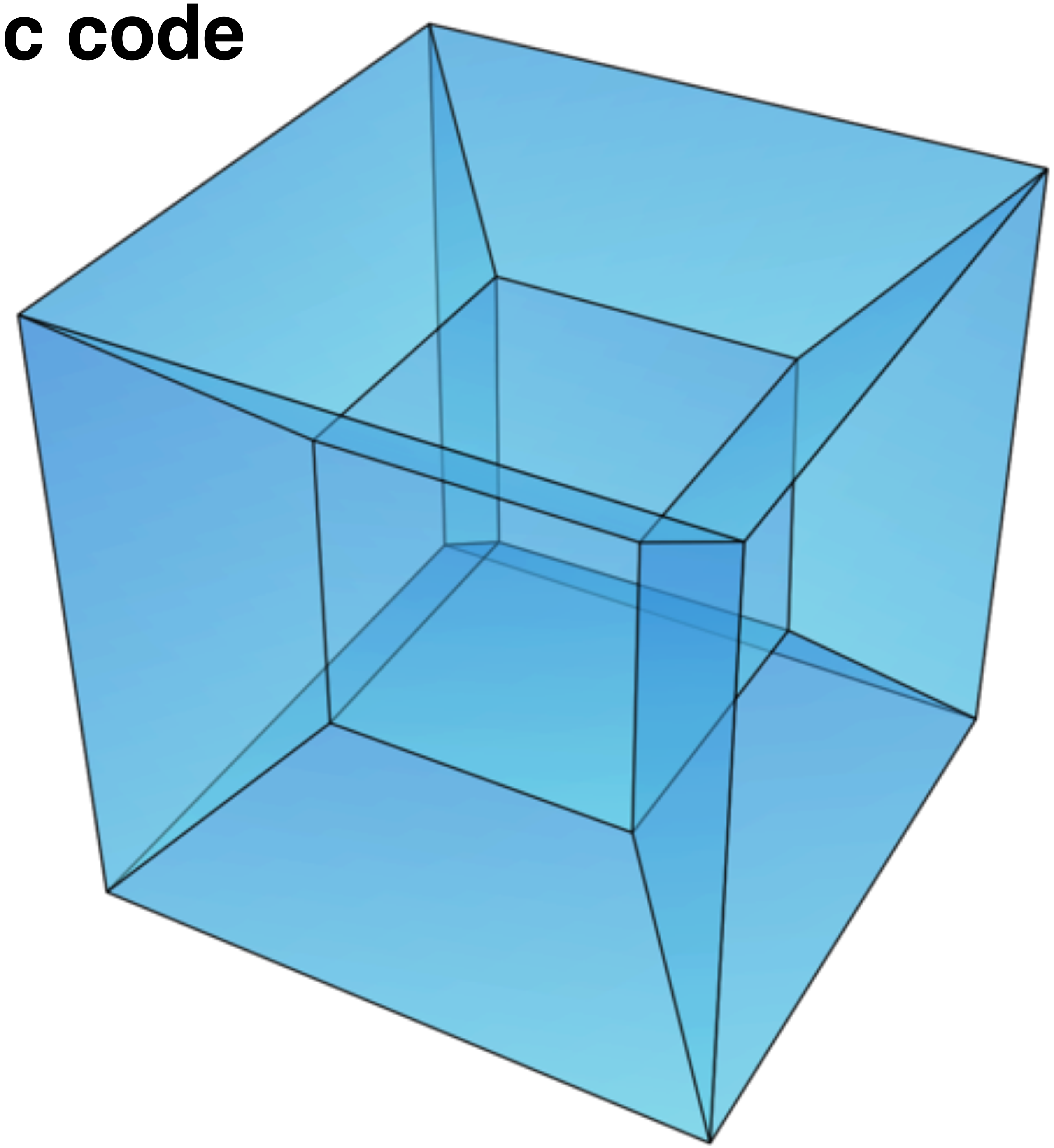
- **Planar code with boundaries**



- **S.B. Bravyi, A.Y. Kitaev, *Quantum Codes on a Lattice with Boundary***, Quantum Computers and Computing, 2001, 2 (1), pp. 43-48.
- **M.H. Freedman, D.A. Meyer**, ***Projective Plane and Planar Quantum Codes***, Foundations of Computational Mathematics, July 2001, Volume 1, Issue 3, pp 325-332

Homological codes

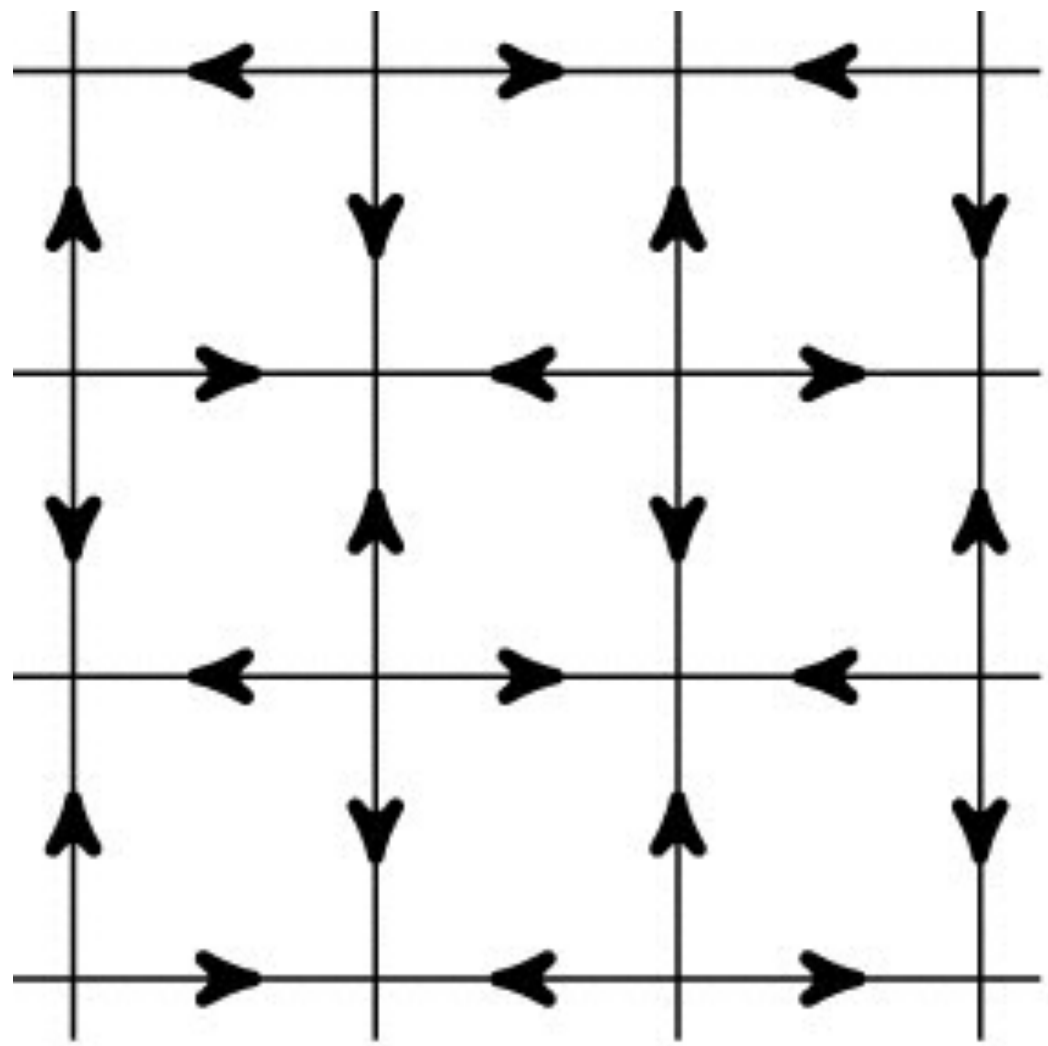
- **4-D toric code**



- **E. Dennis, A.Y. Kitaev, A. Landahl and J. Preskill, *Topological quantum memory,*** J. Math. Phys. 43, 4452 (2002)

Homological codes

- $\mathbf{Z_d}$ homology - **qudit** topological codes

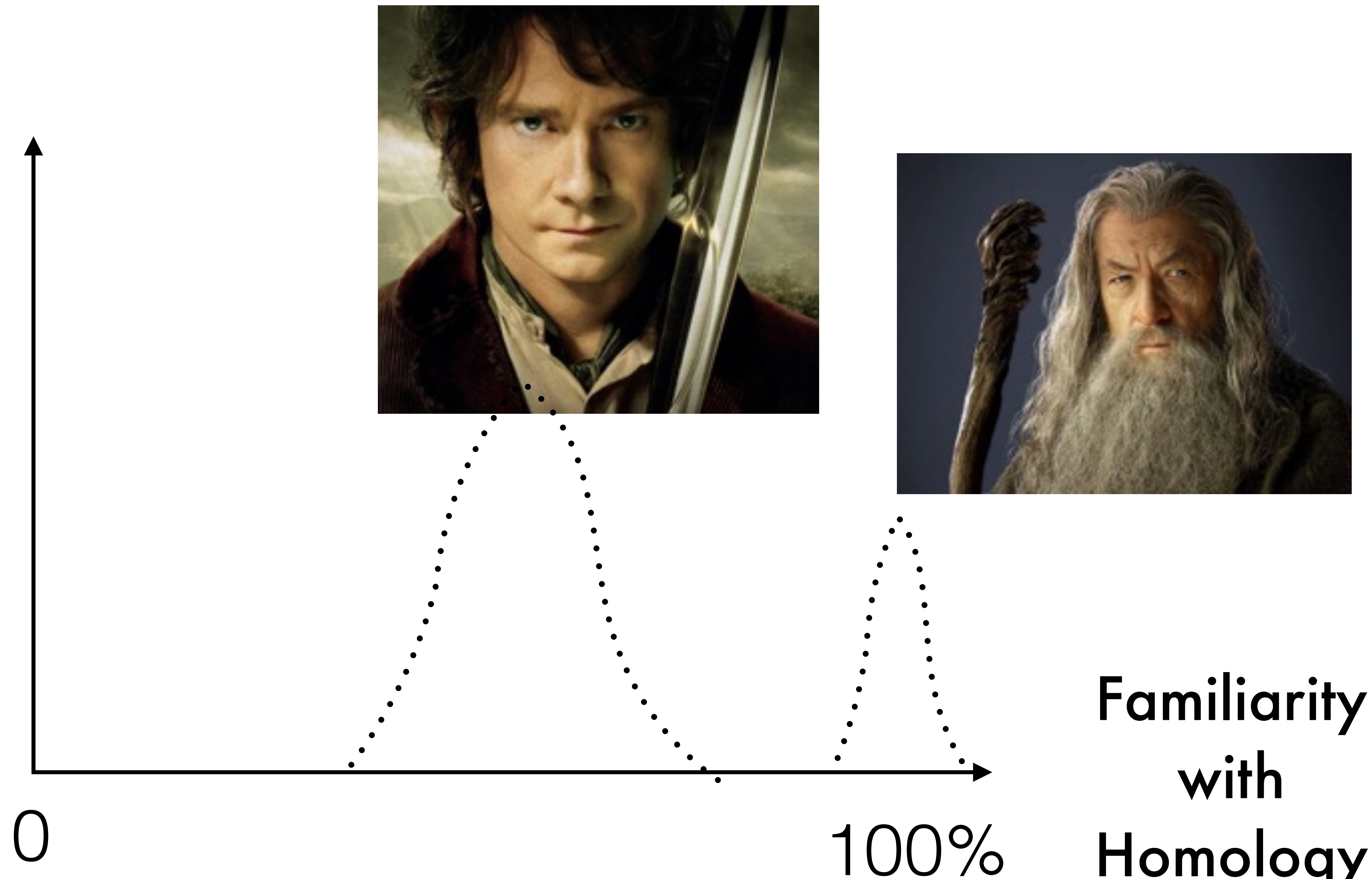


Summary

Cellular homology makes Kitaev's surface code **simple** to describe and **infinitely generalisable**.

The surface code provides a **simple illustration** of the key ideas of **homology** and **cohomology**.

Thank you



Further reading:

Lecture notes on Topological Codes and Homology,

Dan Browne, **http://bit.do/topo1** ***(draft version - please give feedback!)***