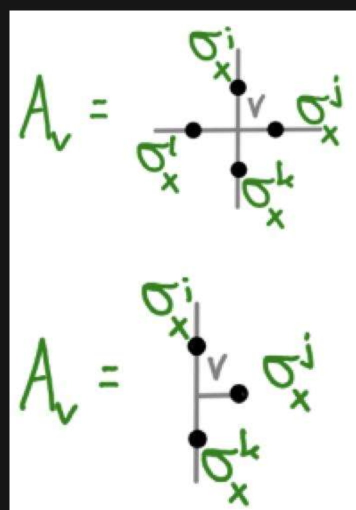
Introduction to the Surface Code

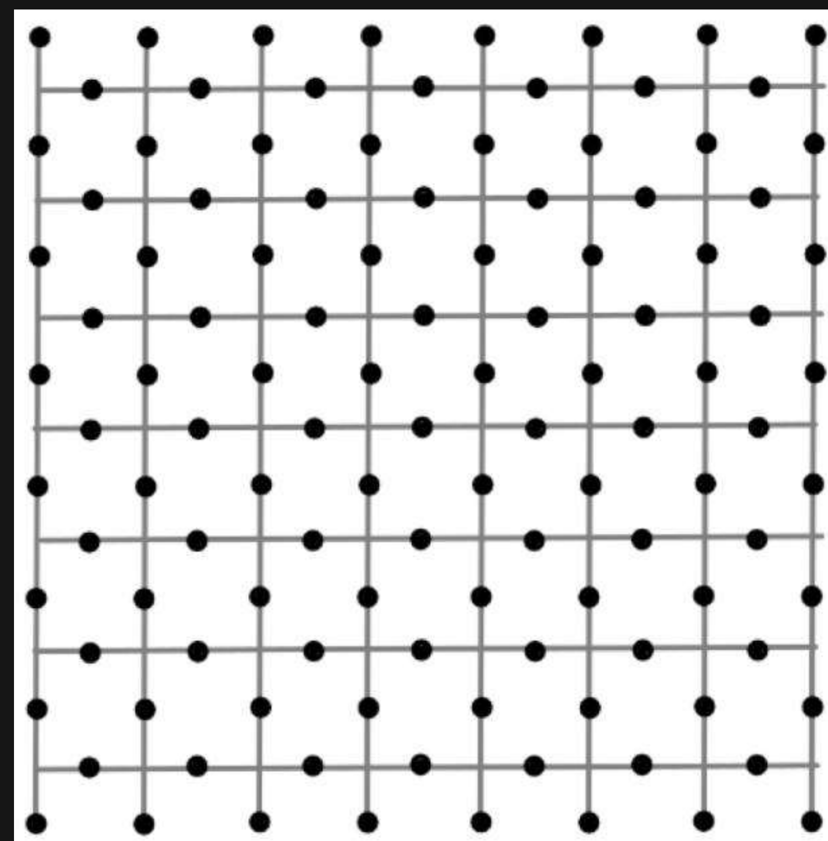
James R. Wootton
IBM Quantum

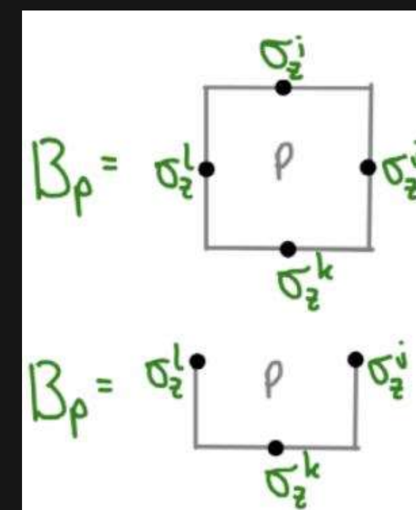
IBM **Quantum**

The Surface Code

- Quantum error correcting codes are defined by the measurements we make
- Let’s move beyond the simple $Z_j Z_{j+1}$ of the repetition code
- In the surface code we use a 2D lattice of code qubits, and define observables for plaquettes and vertices



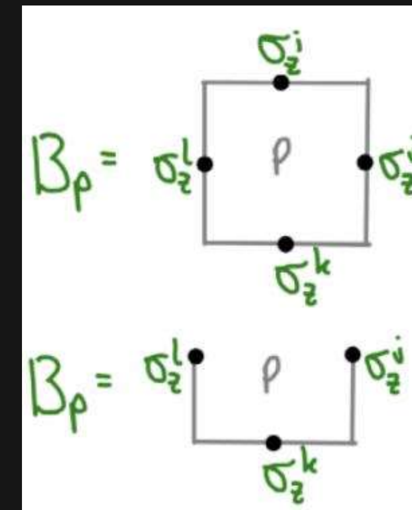


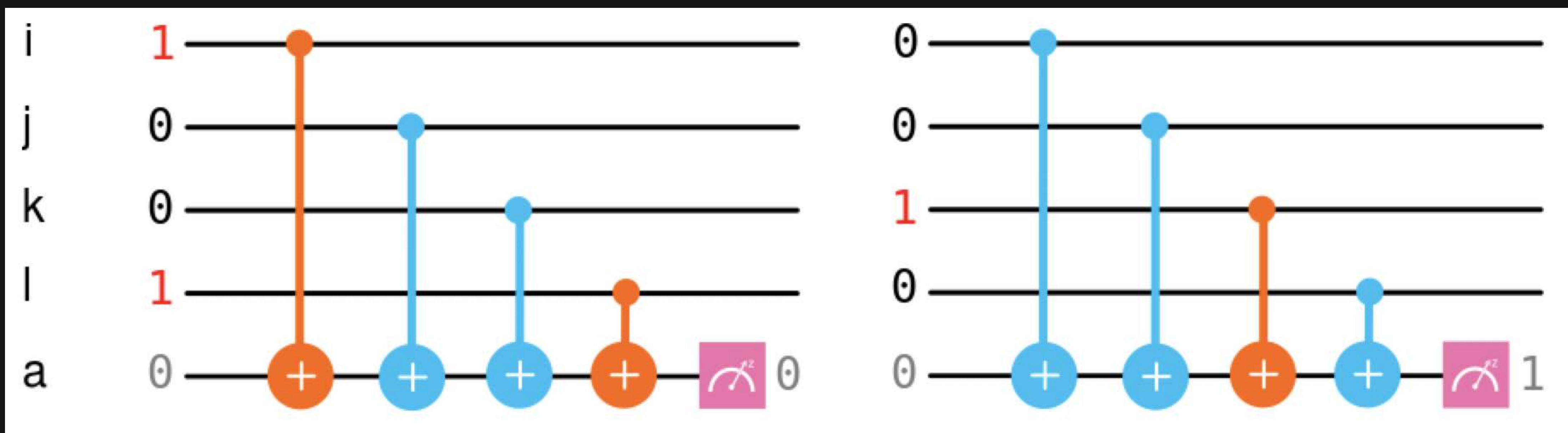


Plaquette Syndrome

IBM **Quantum**

- First let's focus on the plaquette syndrome
- These are similar to the two qubit measurements in the repetition code
- Instead we measure the parity around plaquettes in the lattice
- Can again be done with CX gates and an extra qubit



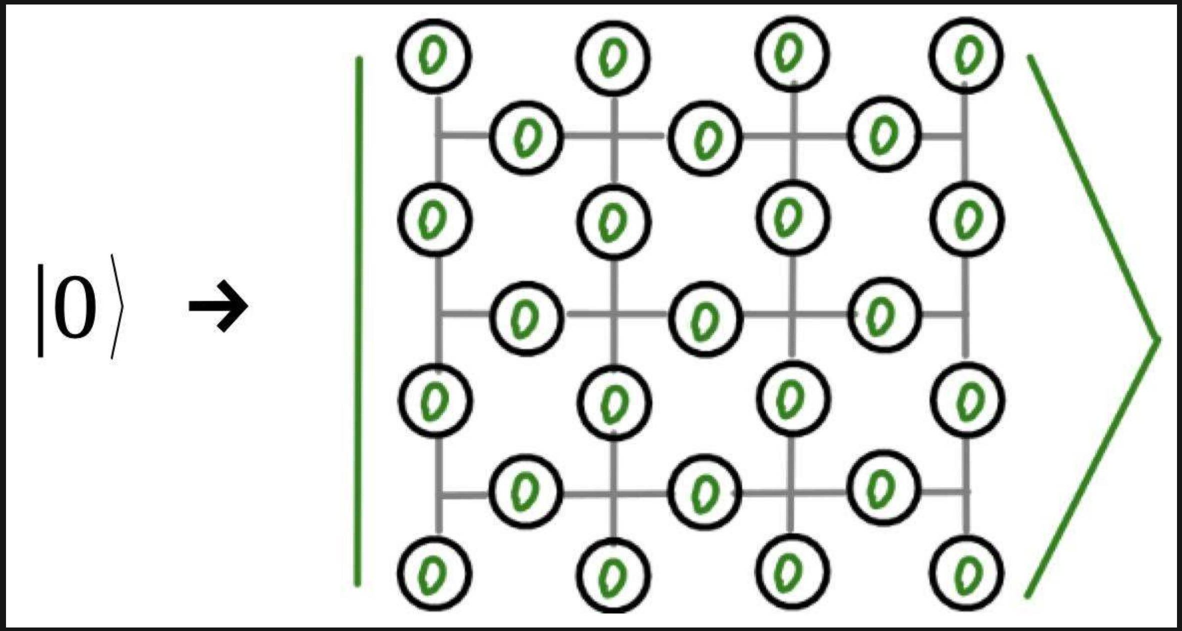


Plaquette Syndrome

- We can define a classical code (storing only a bit) based on the plaquette syndrome alone
- Valid states are those with trivial outcome for all plaquette syndrome measurements:

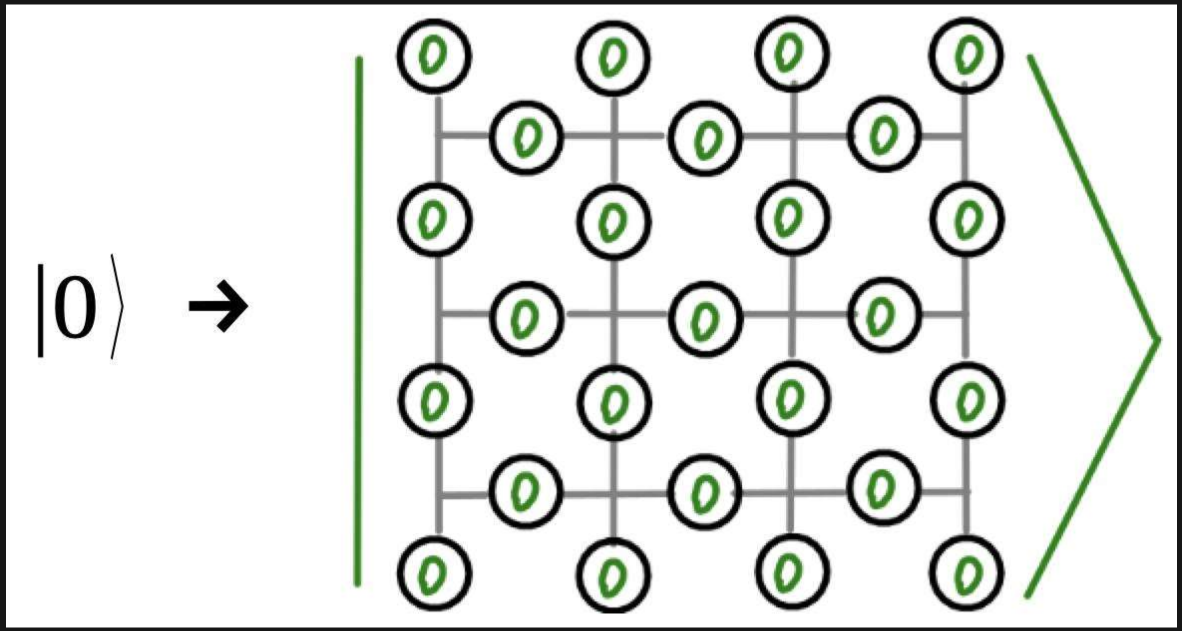
  Even parity on all plaquettes

- How to store a 0 in this?
- How about the state where every code qubit is $|0\rangle$?



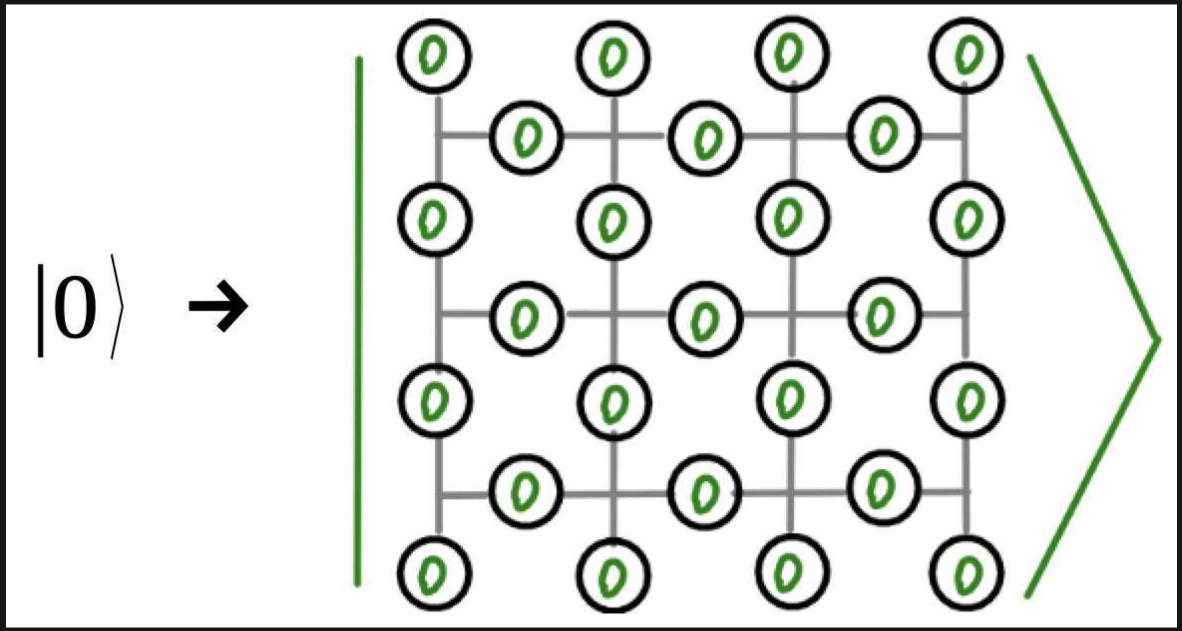
Plaquette Syndrome

- We can define a classical code (storing only a bit) based on the plaquette syndrome alone
- Valid states are those with trivial outcome for all plaquette syndrome measurements:
  Even parity on all plaquettes
- How to store a 0 in this?
- How about the state where every code qubit is $|0\rangle$?



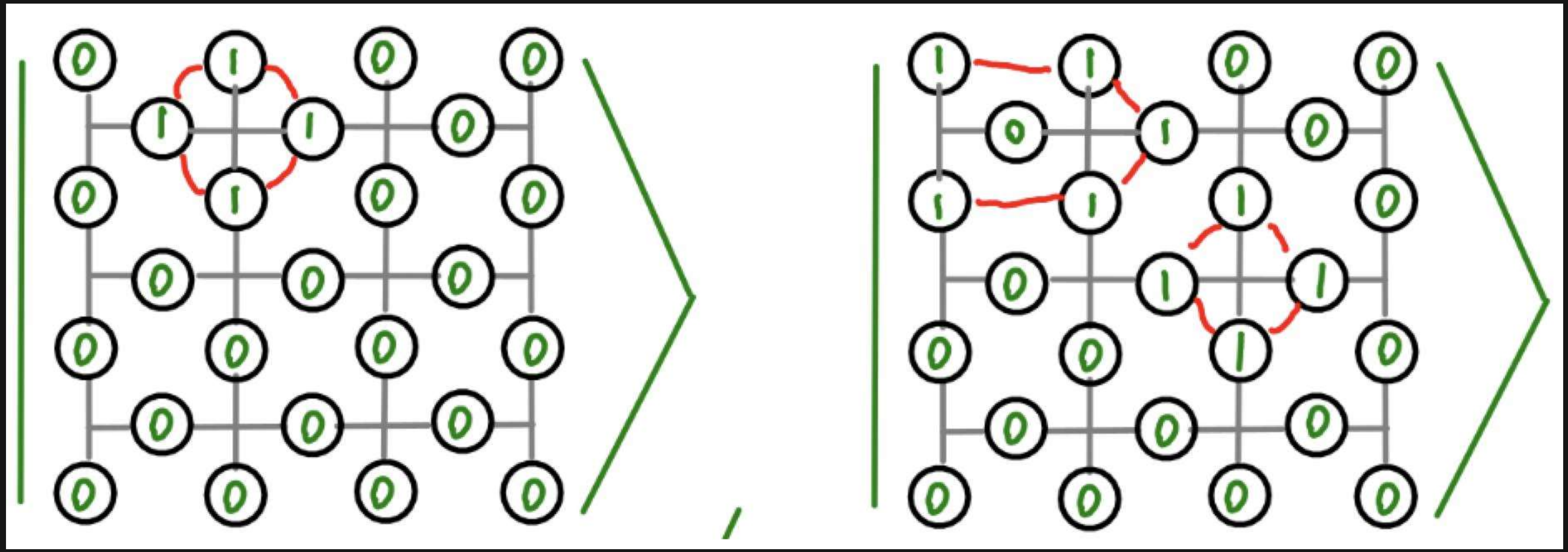
Plaquette Syndrome

- We can define a classical code (storing only a bit) based on the plaquette syndrome alone
- Valid states are those with trivial outcome for all plaquette syndrome measurements:
  Even parity on all plaquettes
- How to store a 0 in this?
- How about the state where every code qubit is $|0\rangle$?



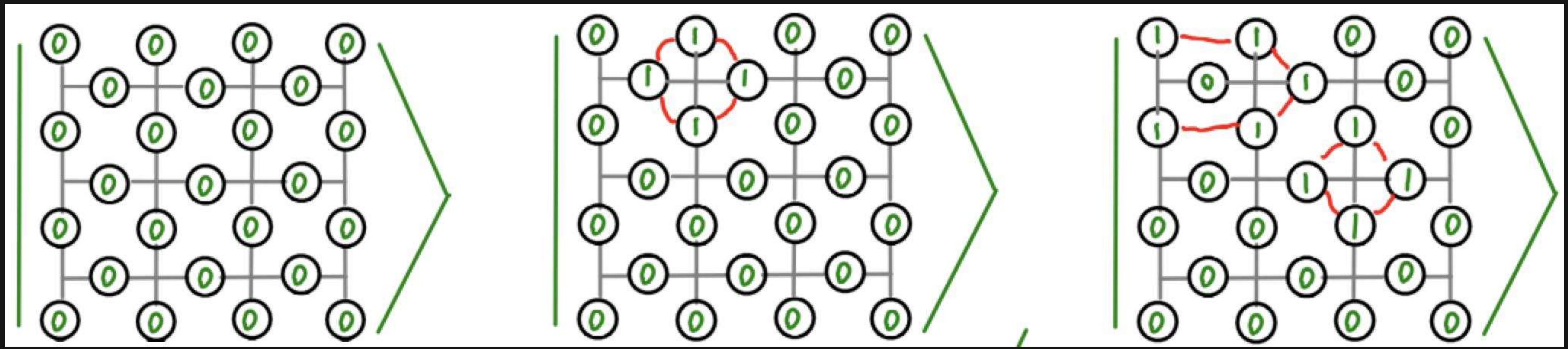
Plaquette Syndrome

- There are ‘nearby’ states that also have even parity on all plaquettes
- These can’t be a different encoded state: they are only a few bit flips away from our encoded 0 state
- We’ll treat them as alternative ways to store a 0



Plaquette Syndrome

- Given any state for an encoded 0
  - Pick a vertex
  - Apply bit flips around that vertex
- Now you have another valid state for 0
- This generates an exponentially large family



Plaquette Syndrome

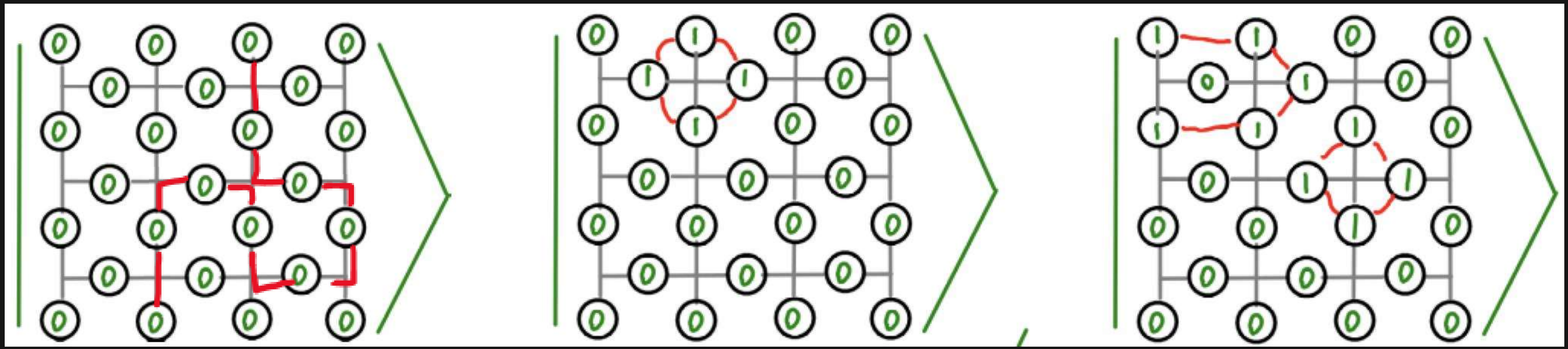
– The states in this family can be very different

– But they all share a common feature

  • Any line from top to bottom (passing along edges) has even parity

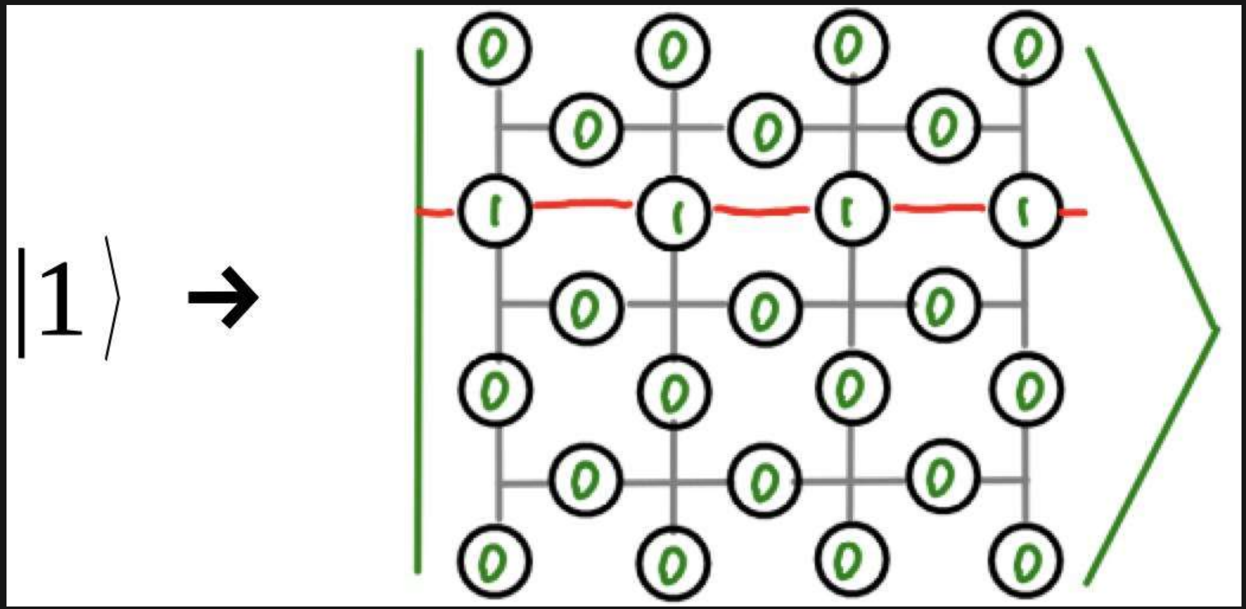
– This is how we can identify an encoded 0

– And it gives us a clue about how to encode a 1



Plaquette Syndrome

– For our basic encoded 1, we use a bunch of 0s with a line from left to right (passing through plaquettes)



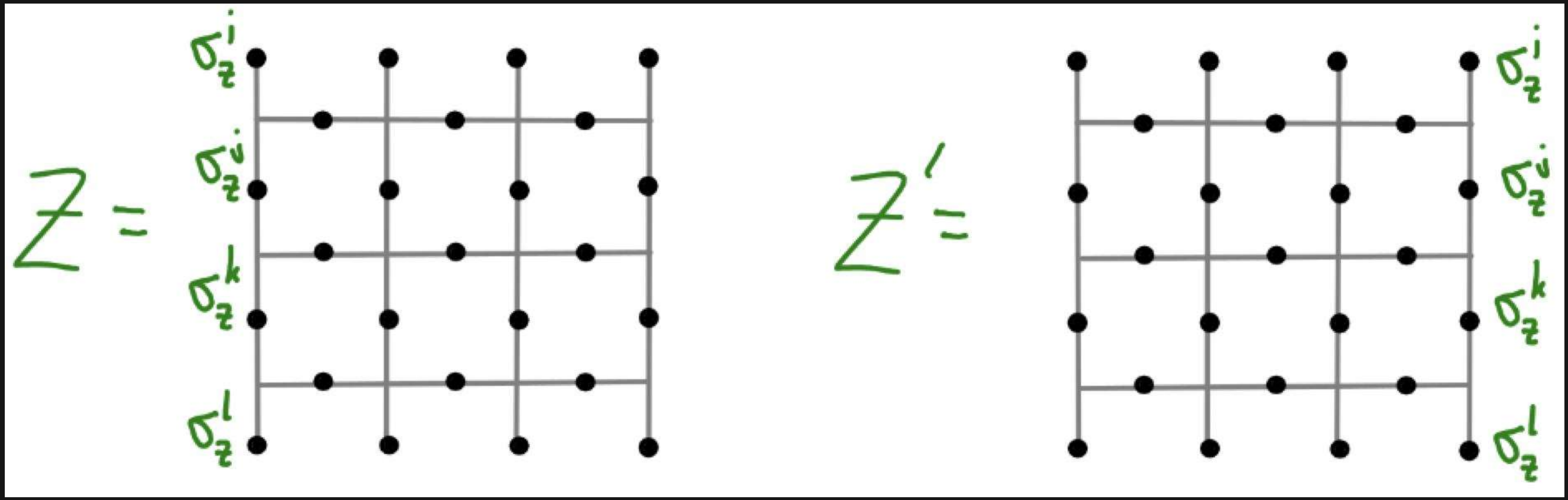
– This also spawns an exponentially large family

– All have *odd* parity for a line from top to bottom

– Unlike the repetition code, distinguishing encoded 0 and 1 requires some effort (which is good!)

Logical X and Z

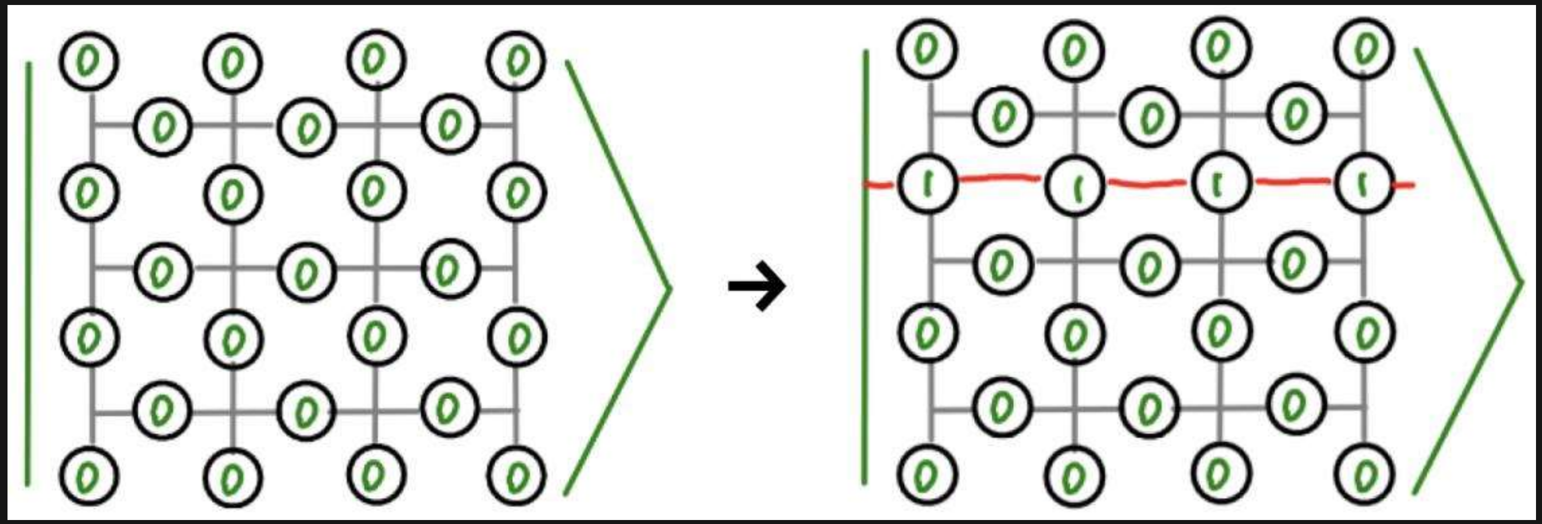
- Distinguishing 0 and 1 corresponds to measuring Z on the physical qubit
- The following observables detect what we need



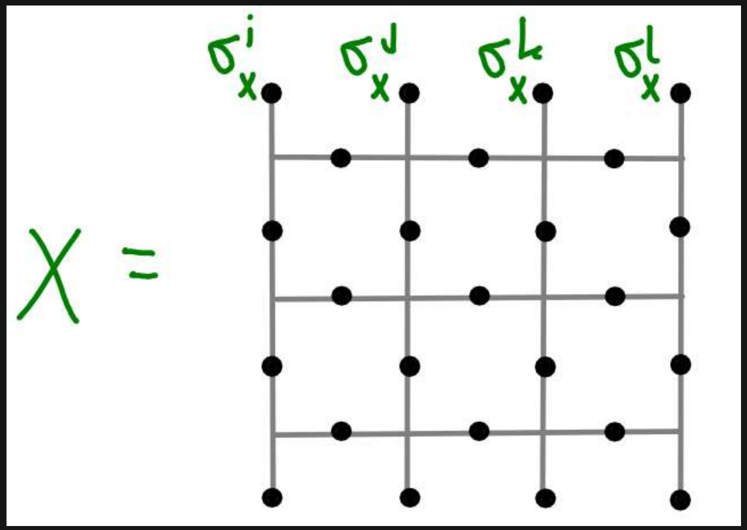
- Or the same on any line from top to bottom
- Uses the edges has a nice advantage: we can think of them as large (unenforced) plaquettes

Logical X and Z

- To flip between 0 and 1, we can flip a line of qubits

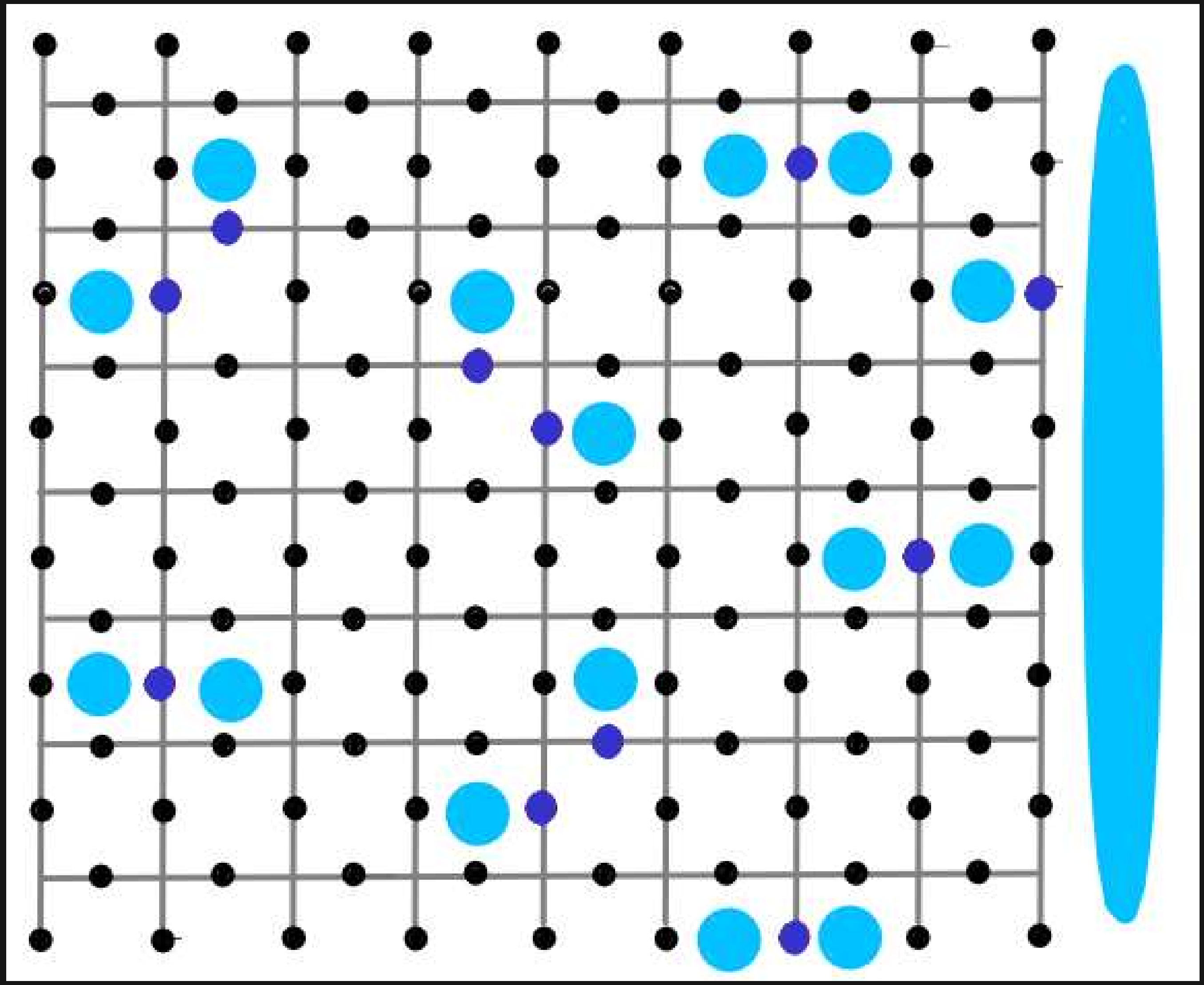


- Such lines of flips act as an X on the logical qubit



Effects of Errors

- Applying an X to any code qubit changes the parity of its two plaquettes
- An isolated X creates a pair of defects
- Further Xs can be move a defect, or annihilate pairs of them
- A logical X requires many errors to stretch across the lattice

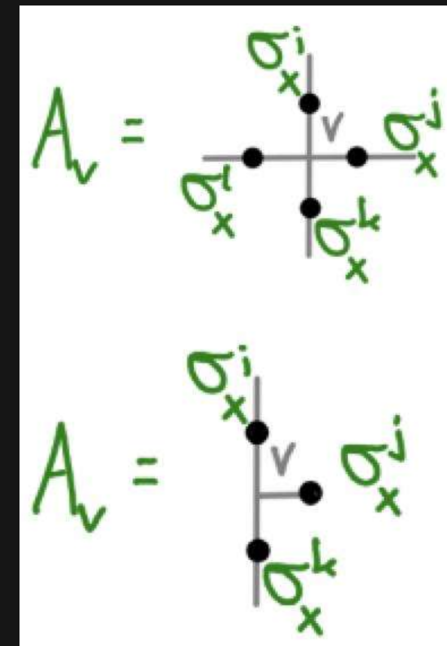


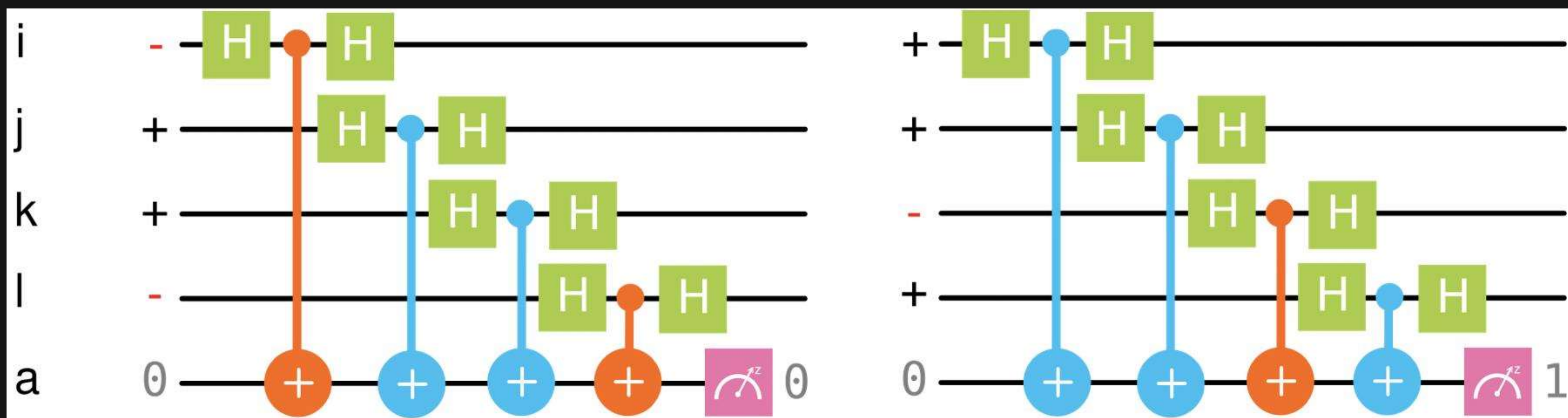
- With the plaquette operators, we can encode and protect a *bit*

Vertex Syndrome

IBM **Quantum**

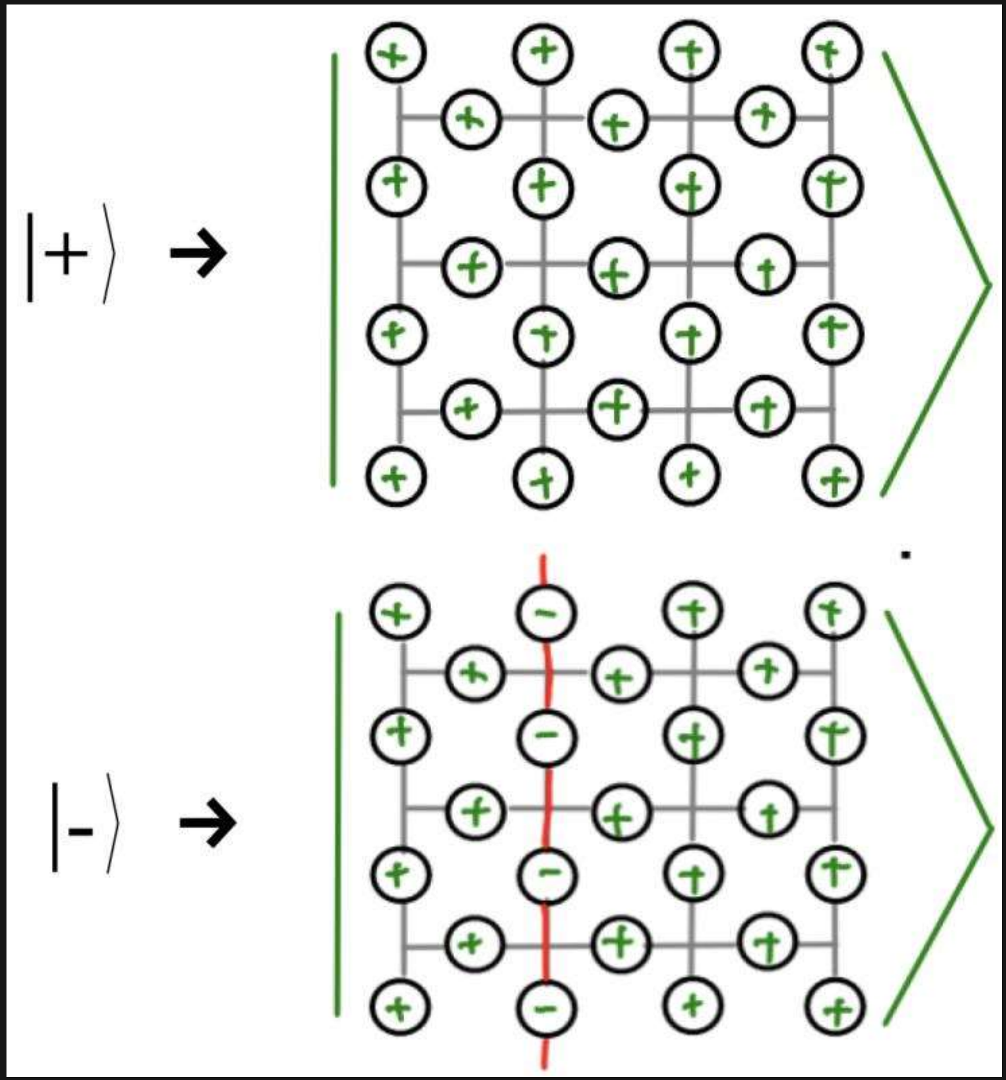
- Now forget the plaquettes and focus on vertices
- These observables can also be measured using CX gates an an ancilla
- In this case they look at the $|+\rangle$ and $|-\rangle$ states, and count the parity of the number of $|-\rangle$s





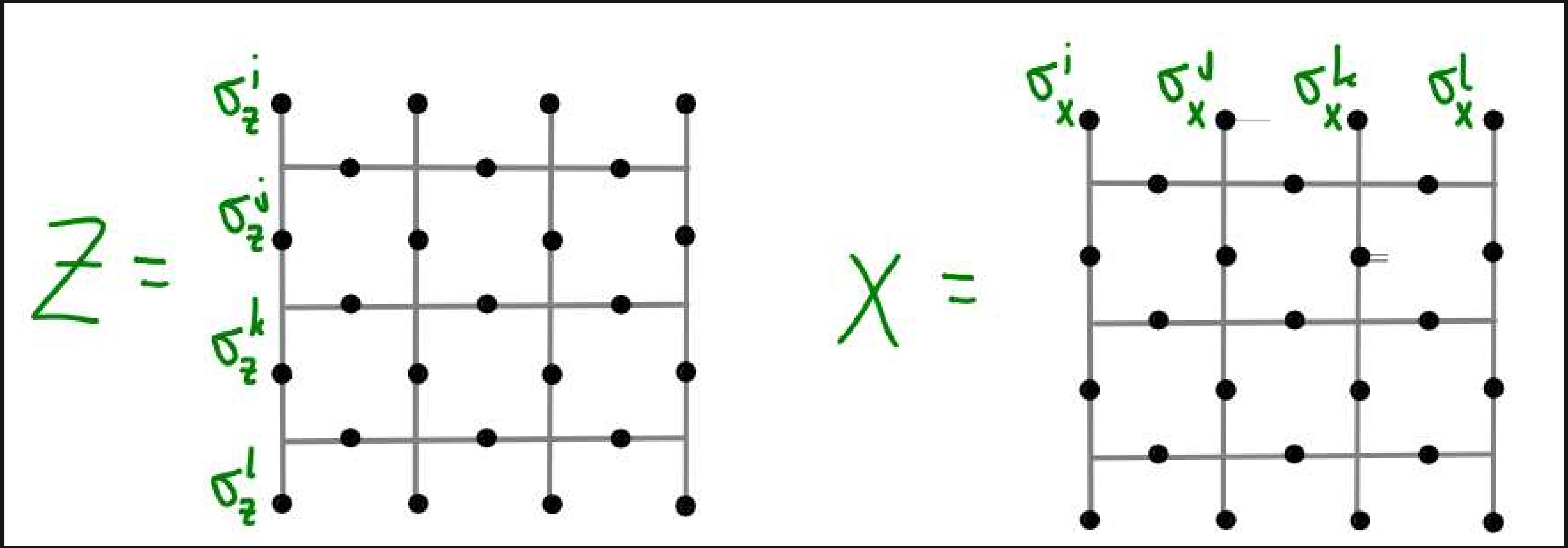
Vertex Syndrome

- These operators also allow is to encode and protect a bit value
- In this case, let's use + and - to label the two states
- They are encoded using suitable patterns of $|+\rangle$ and $|-\rangle$ states for the code qubits
- As with the plaquettes, these also correspond to exponentially large families of states



Logical X and Z

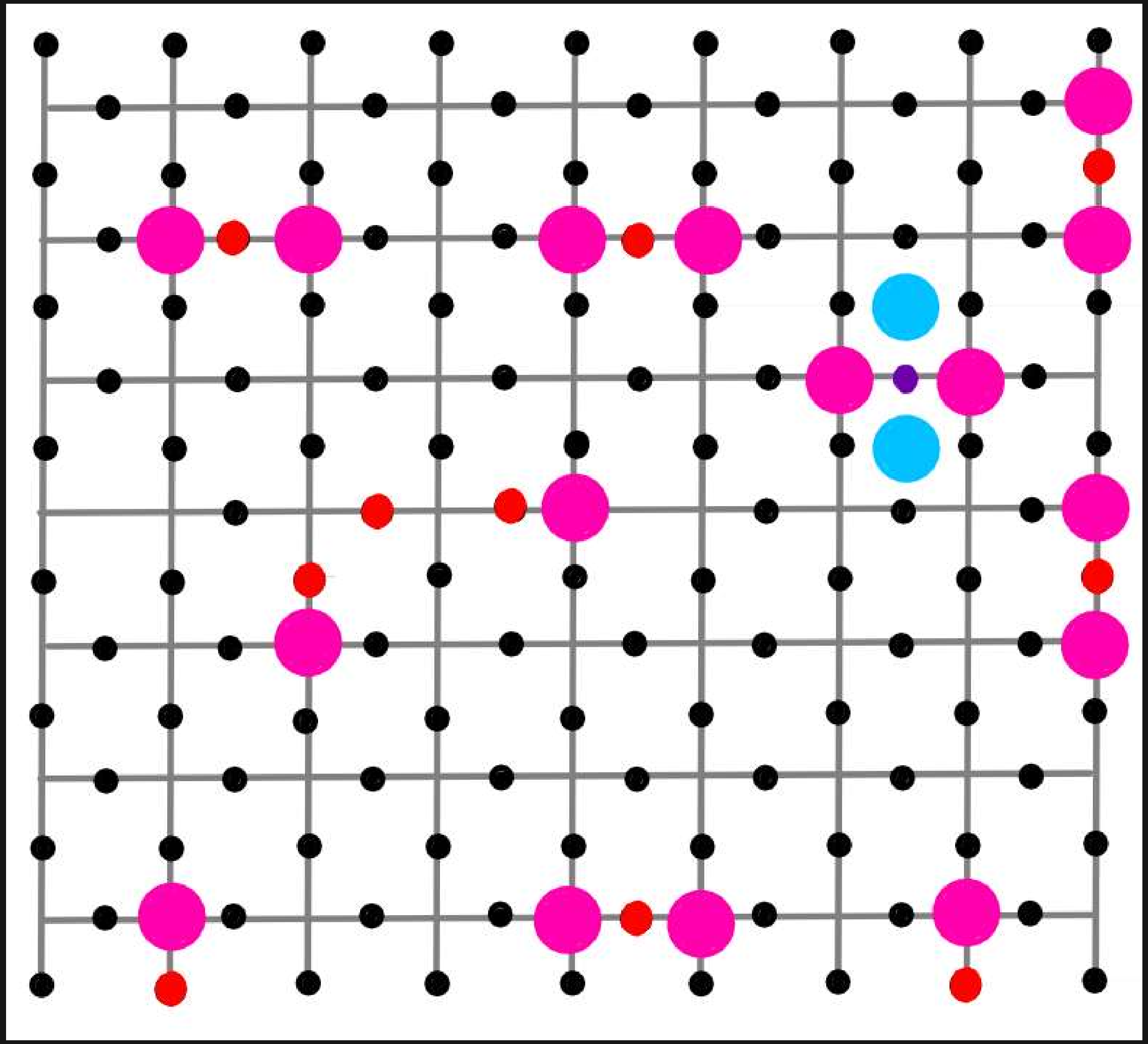
- What is the X operator (distinguish between $|+\rangle$ and $|-\rangle$)?
- What is the Z operator (flip between $|+\rangle$ and $|-\rangle$)?
- Turns out they are exactly the same as before!



Effects of Errors

IBM **Quantum**

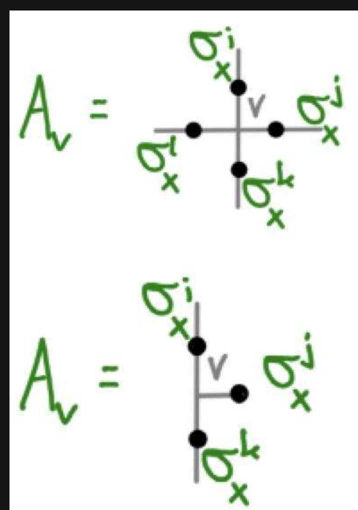
- Applying a Z to any code qubit changes the X parity of its two vertices
- An isolated Z creates a pair of defects
- Further Zs can be move a defect, or annihilate pairs of them
- A logical Z requires many errors to stretch across the lattice

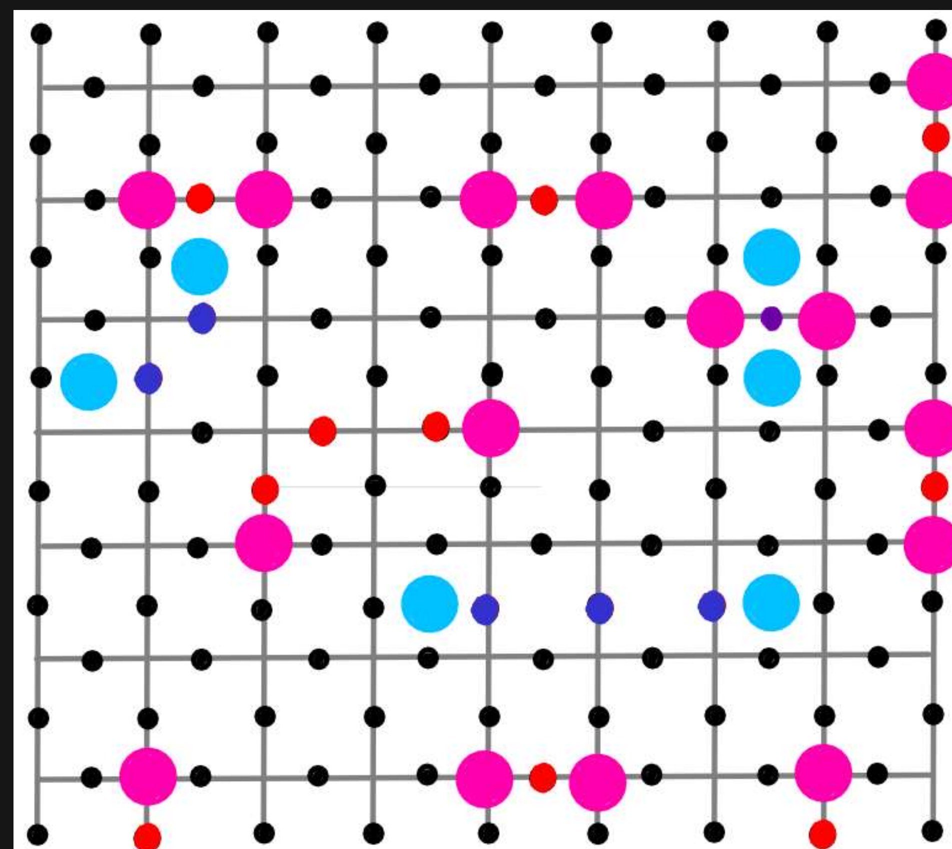


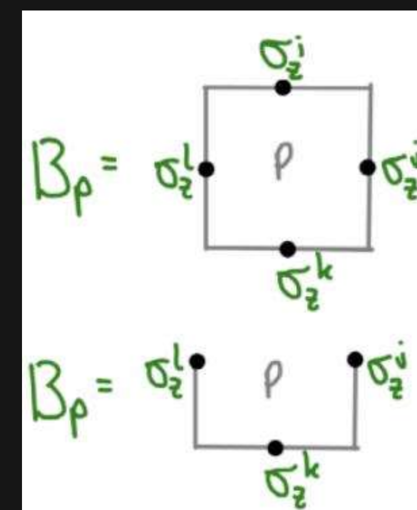
- With the vertex operators, we can encode and protect a *bit*

Putting it all Together

- The plaquette and vertex operators commute
- This allows us to detect both X and Z errors
- Since Y~XZ, we can detect Y errors too

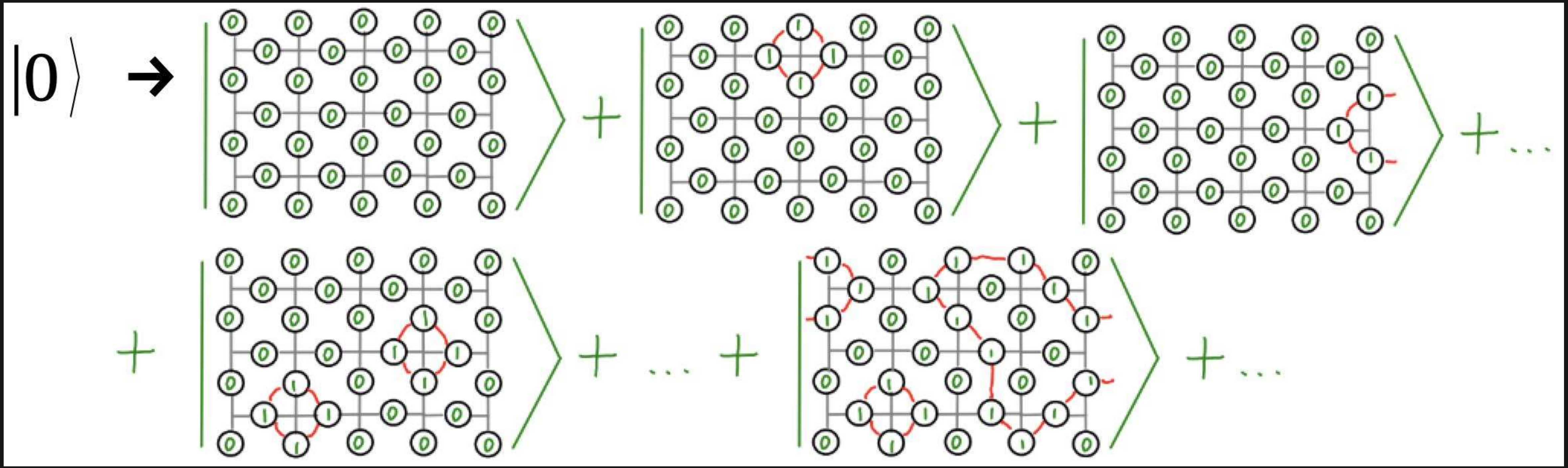






Putting it all Together

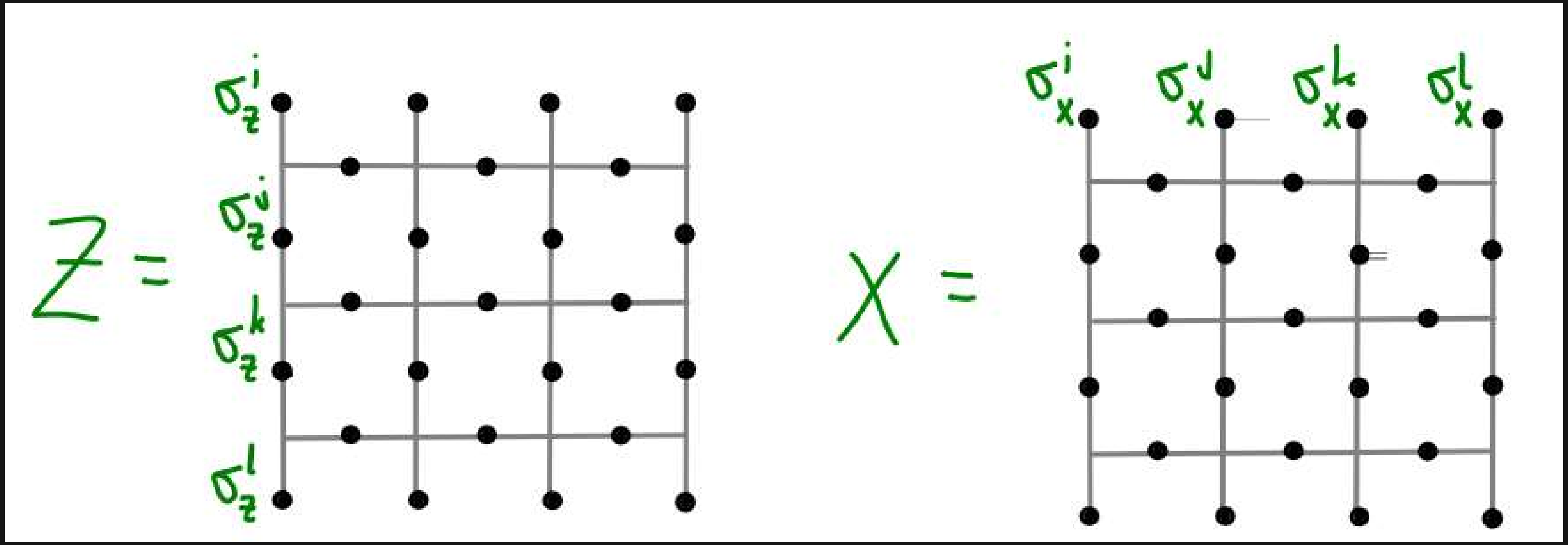
- Encoded states now unique: superposition of all previous solutions
- For example, the encoded 0



- Satisfies $A_v|\psi\rangle = |\psi\rangle$ and $B_v|\psi\rangle = |\psi\rangle$, so will give the 0 outcome for all stabilizer measurements

Putting it all Together

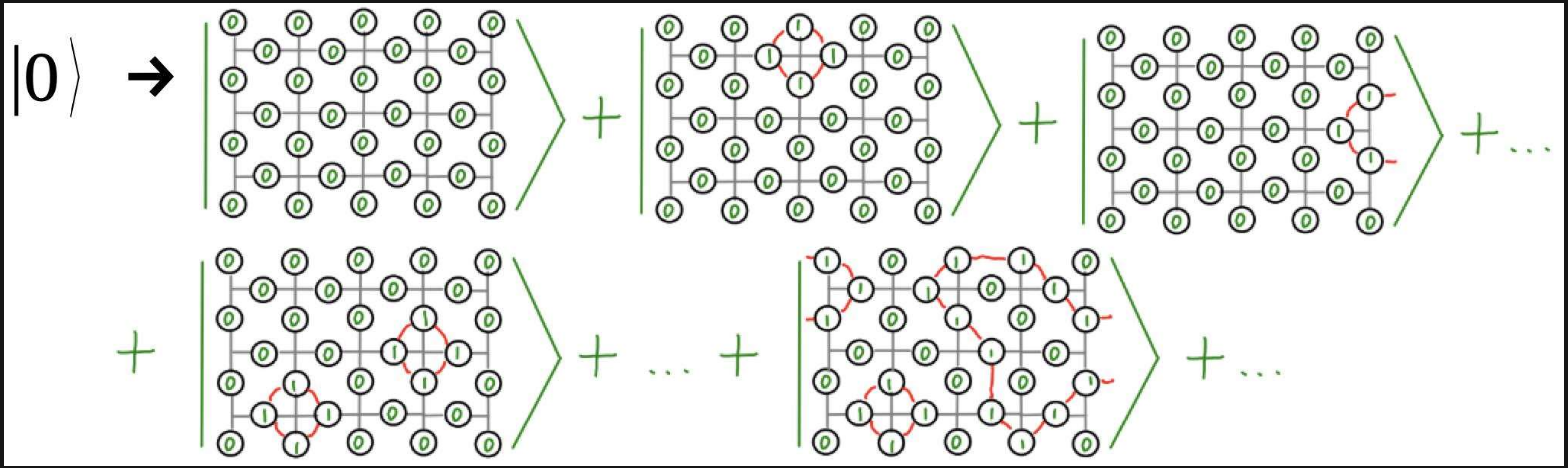
- The Z and X operators on the encoded qubit are exactly the same as before



- These, and the Hadamard, can be performed fault-tolerantly

Putting it all Together

- The states we need are highly entangled quantum states
- They are examples of topologically ordered states



- Though such things can be hard to make, we create and protect them with the syndrome measurements

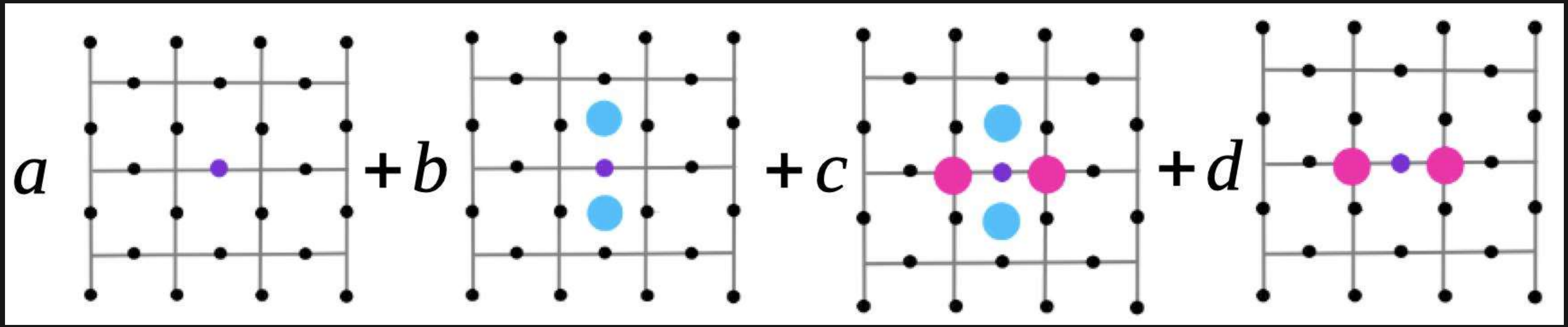
Putting it all Together

IBM **Quantum**

- We are not just protected against X and Z, but all local errors
- As mentioned earlier, Y~XZ
- Everything else can be expressed

$$E = a\,I + b\,X + c\,Y + d\,Z$$

- This creates a superposition of different types of error on the surface code



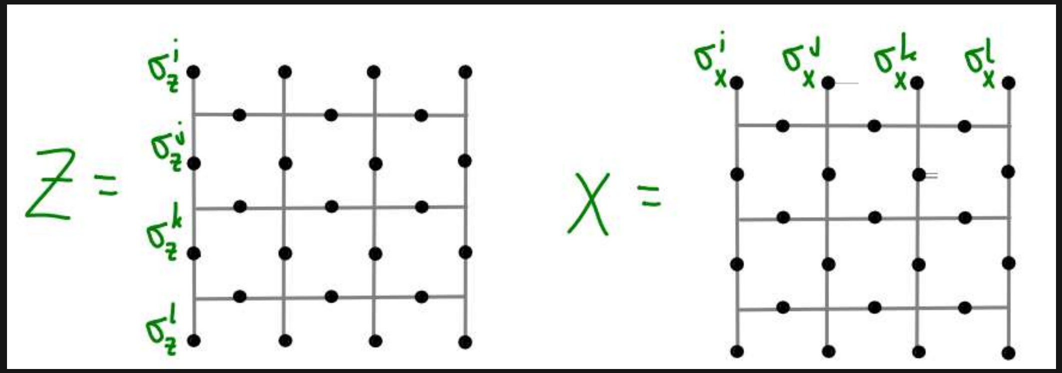
- Measuring the stabilizers collapses this to a simple X, Y or Z

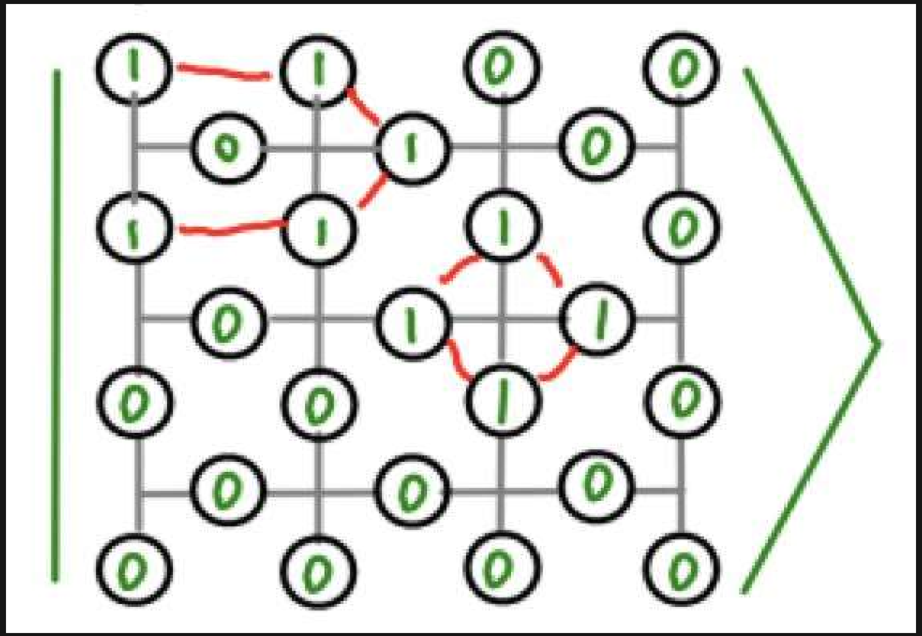
Final Readout

- The logical operators are many-body observables
- So how do we read them out fault-tolerantly

- When you decide on a basis for final measurement, you stop caring about some errors
- You can then measurement in a product basis
- Final readout and final stabilizer measurement can be constructed from the result

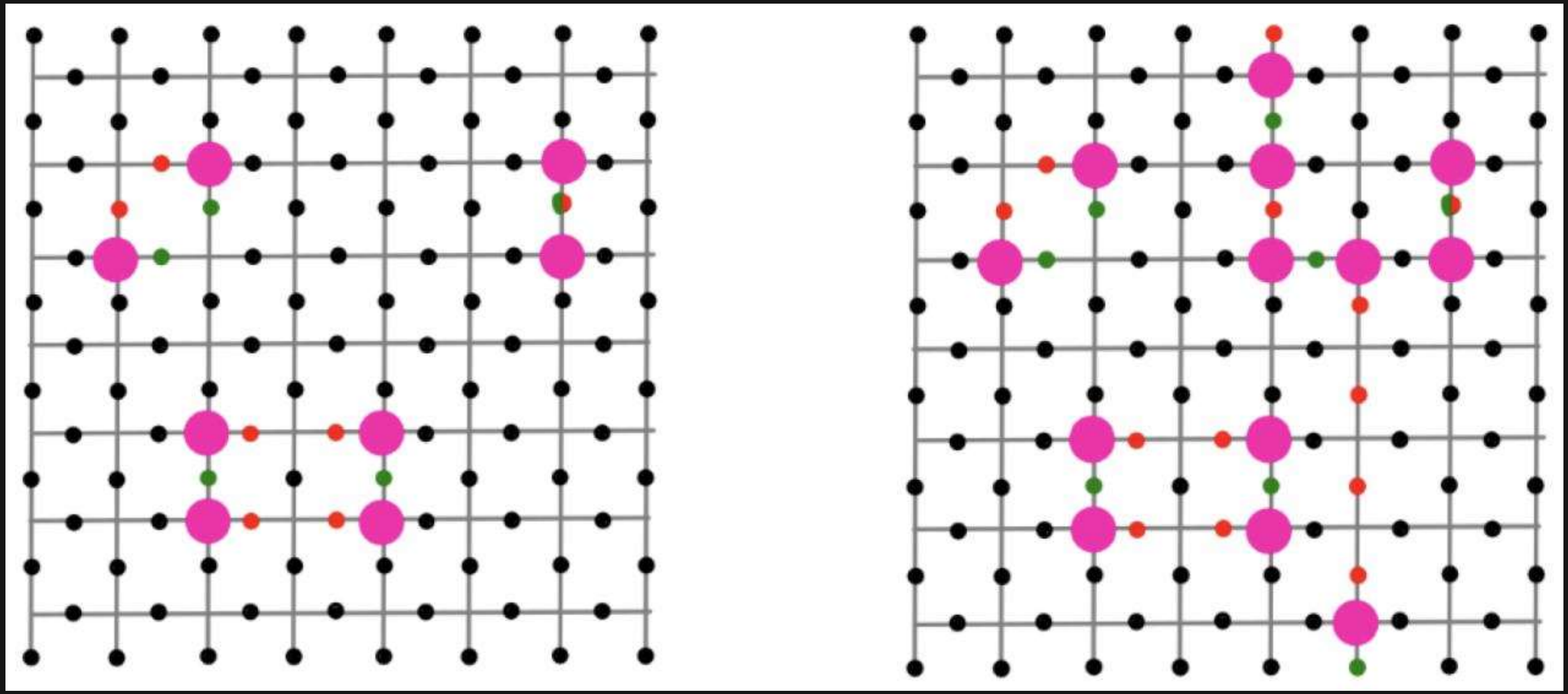
- Measurement errors are effectively the same as errors before measurement





Decoding

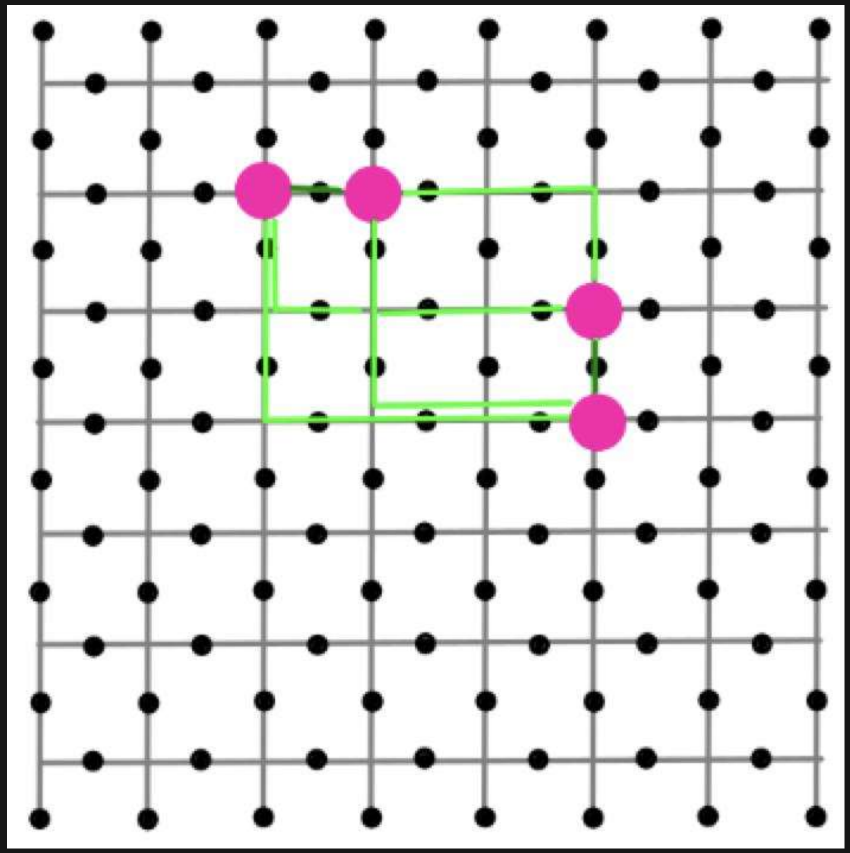
- Given the measurement results, we need to work out what errors happened
- More specifically, the ‘equivalence class’ of errors
- This is the job of the decoding algorithm

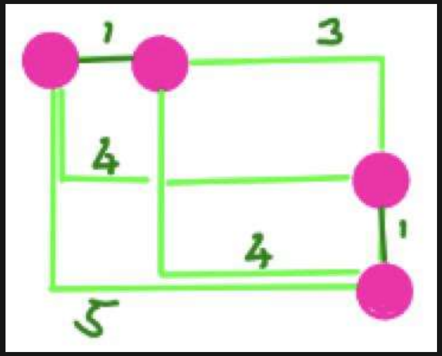


Decoding with MWPM

- A good option is MWPM
- Just as we considered for the repetition code

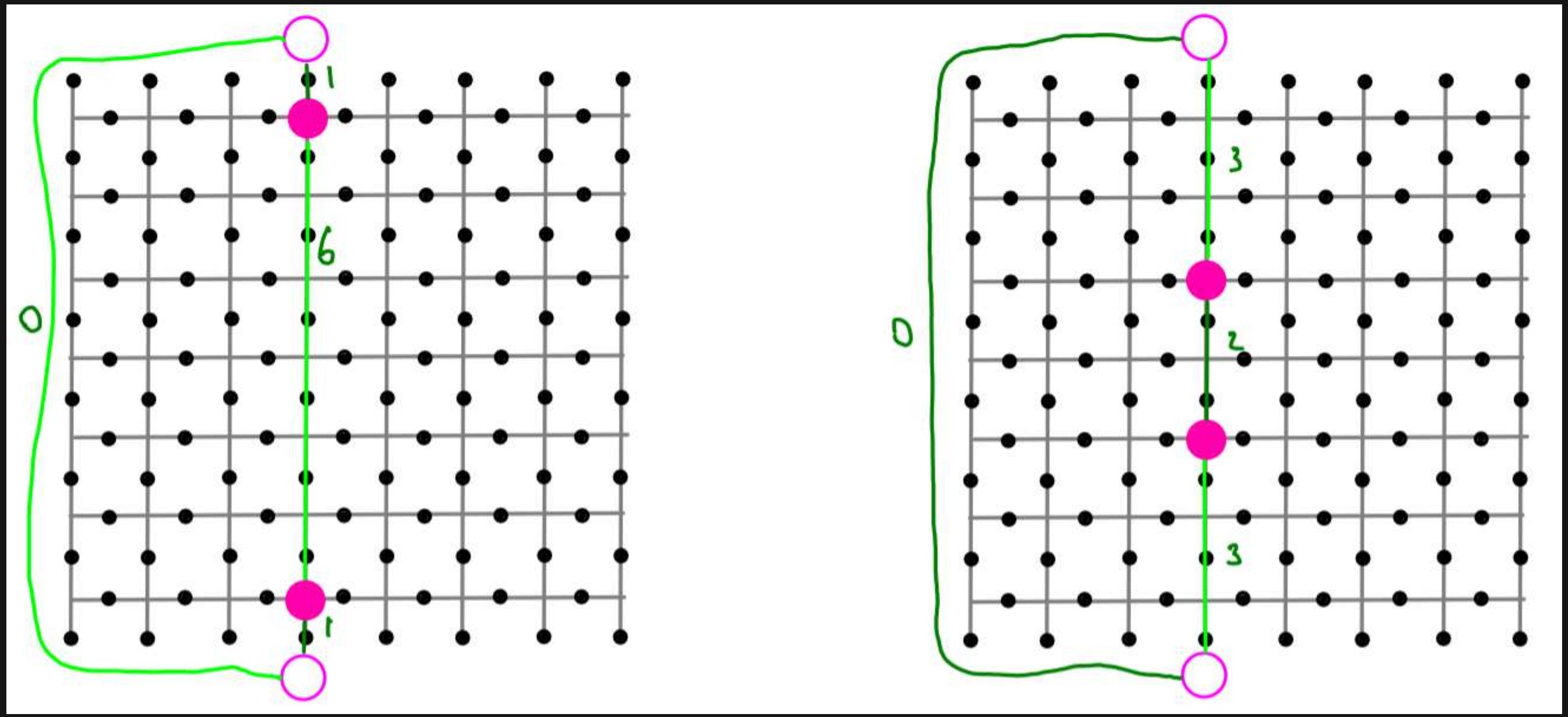
- Again we start with the simple and unrealistic case: errors only between measurement
- Each round can be decoded separately, corresponding to MWPM on a 2D graph
- Decoding for X and Z errors can also be done independently





Decoding

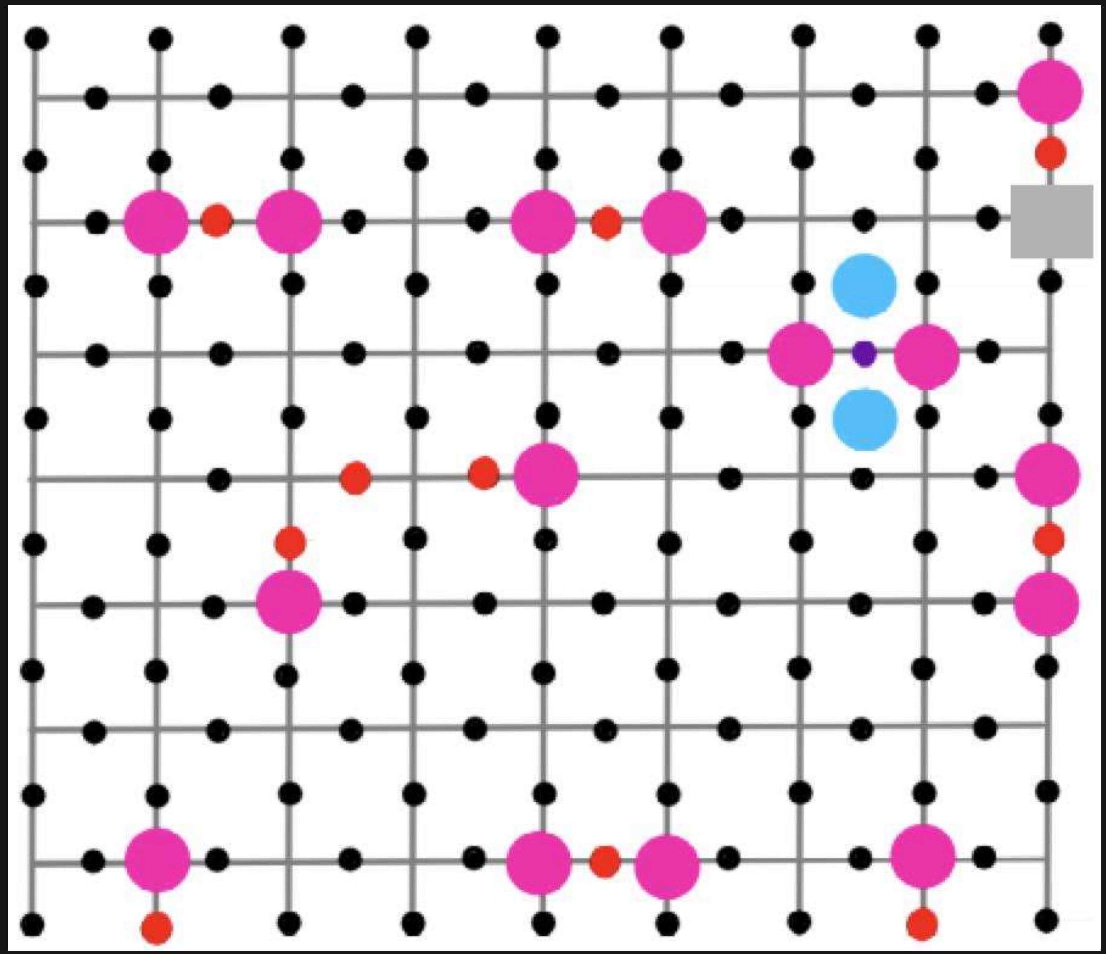
- We need to be careful to account for the effects of the edges
- This is done by introducing extra ‘virtual nodes’
  (also required for the repetition code, but we ignored it earlier)



Imperfect Measurements

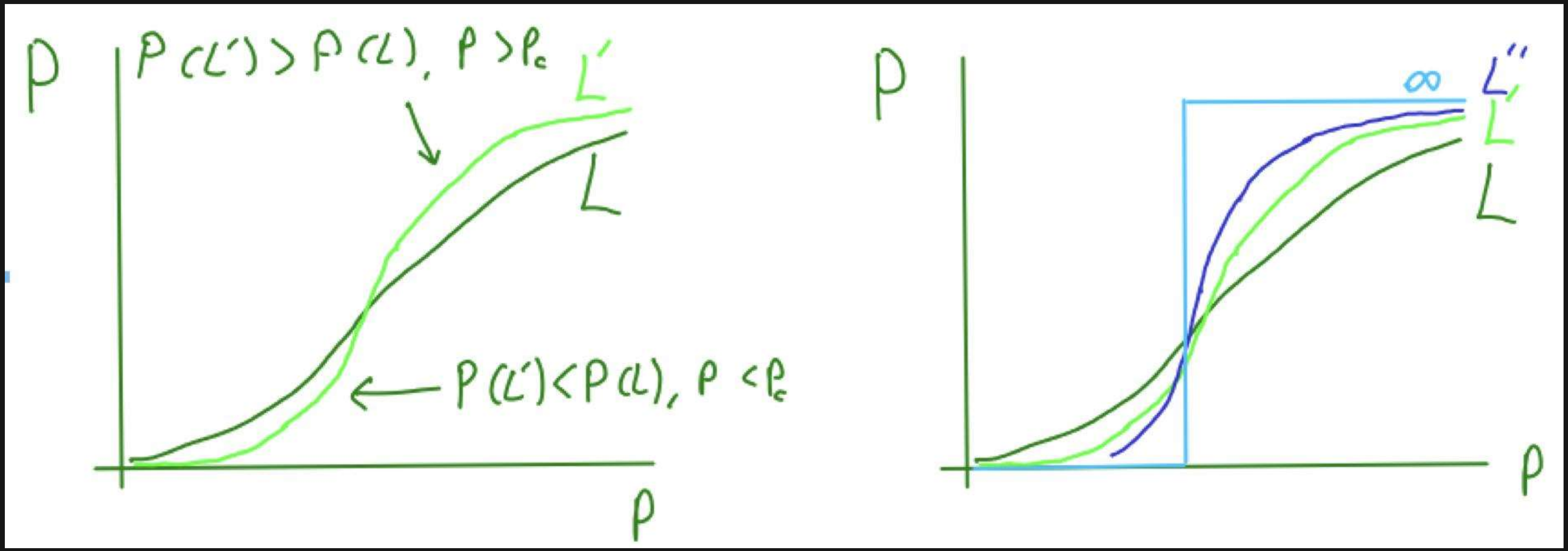
IBM **Quantum**

- Again, we have the problem of imperfect measurements
  - The measurements might lie
  - Errors on the additional qubit
  - Errors in the CX gates
- We base the decoding using syndrome changes
- This leads to a 3D MWPM problem (2D space + time)



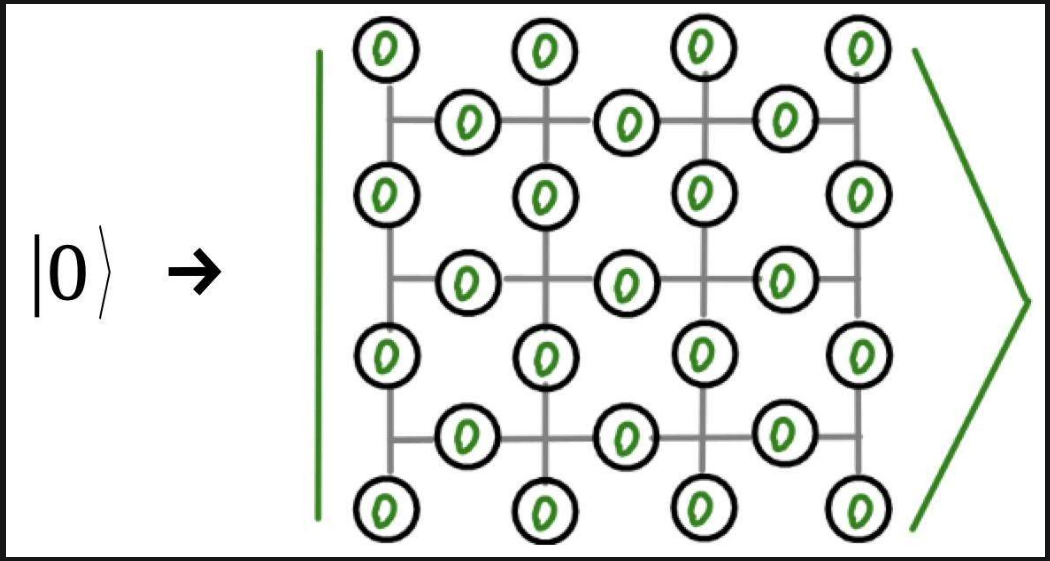
Threshold

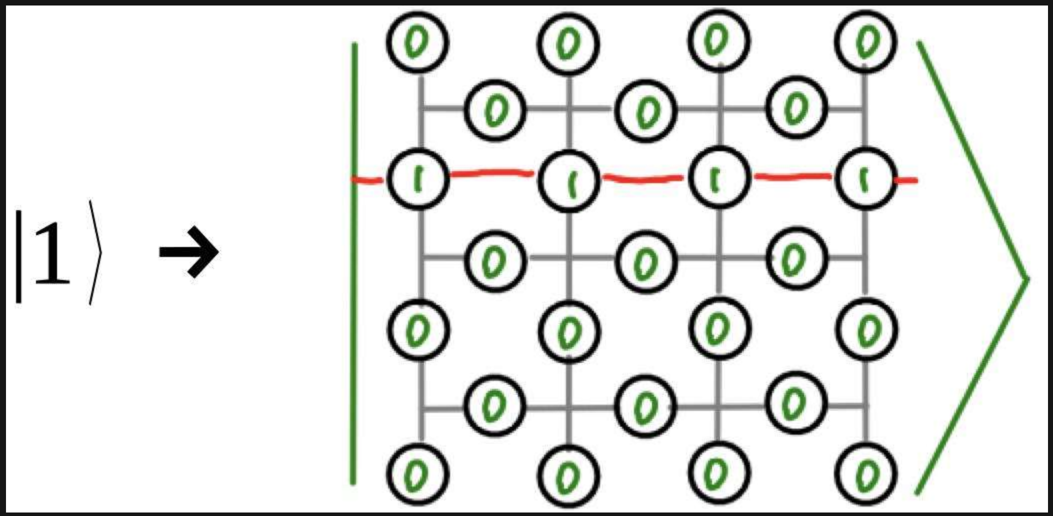
- Correcting according to the right class removes the effects of errors
- Correcting according to the wrong class causes an operation on the encoded qubit (without our knowing)
- What is the probability of such an error, *P*, given the probability on the qubits of the code, *p*?
- We find a phase transition as *L* is increased (for an *LxL* grid)



More Logical Gates

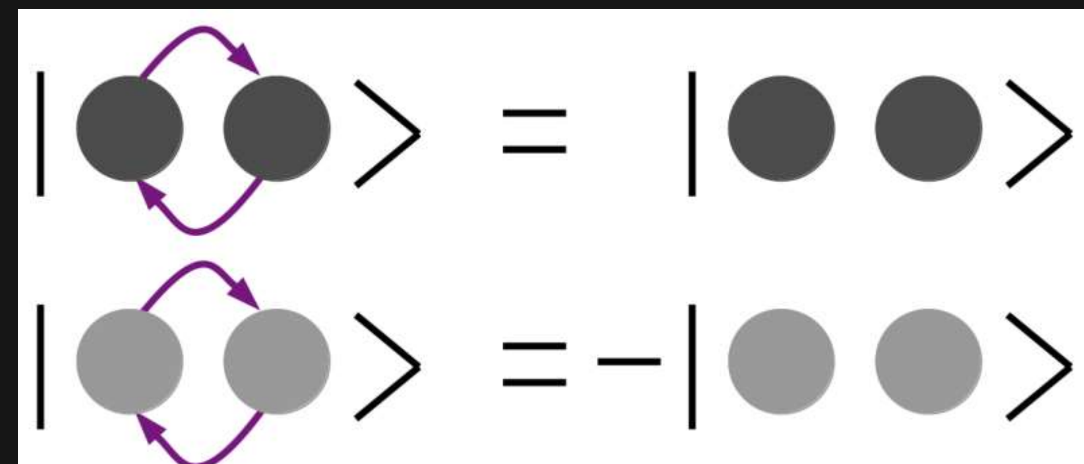
- We've seen how to do logical X and Z
- A logical CX can be done without much trouble
- A logical H requires the lattice to be rotated, but that can be done
- Other logical Clifford gates can be done with some crazy tricks
- But that's all! No other logical operations can be done fault-tolerantly.
- A solution is *magic state distillation*, using the logical gates we have to clean up the one we don't
- But that's beyond today's lecture…



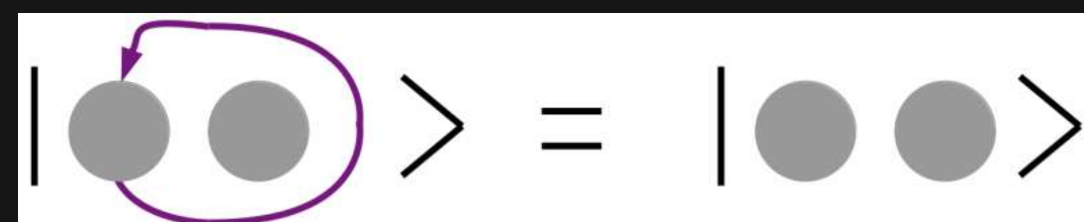


Anyons in the Surface Code

- There are many variants on how qubits can be encoded and manipulated in the surface code
- They all depend on the unique topological nature
- The ‘defects’ created by errors in the surface code behave like particles

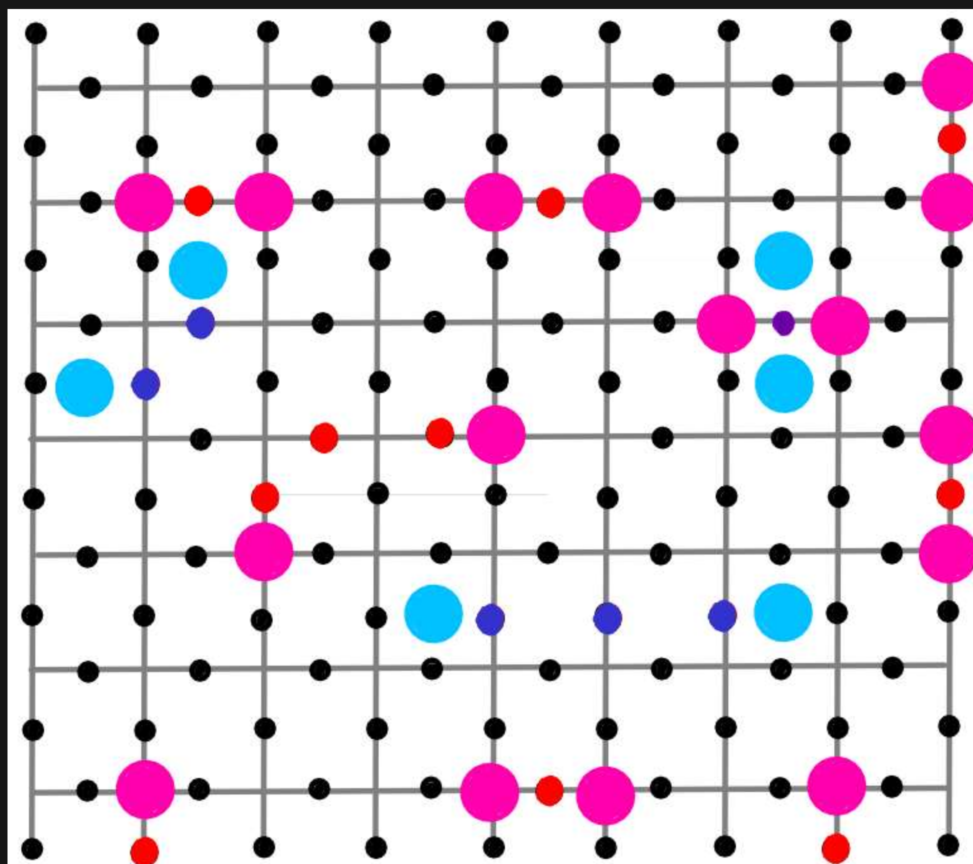


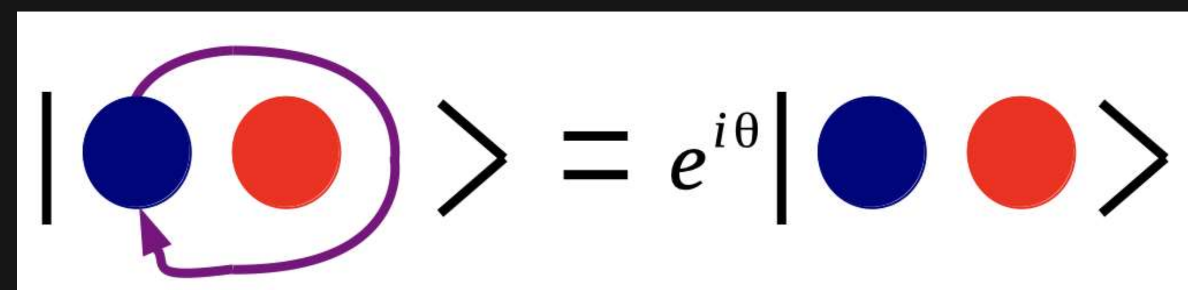
- All particles in our universe are either bosons or fermions
- This due to topological restrictions in a 3D universe



Anyons in the Surface Code

- The surface code is only a 2D ‘universe’, so doesn’t have these restrictions
- How do these particles behave?





Anyons in the Surface Code

- Braiding a particle corresponds to applying a stabilizer
- Their eigenstates defines the braiding phase
- Neither bosons nor fermions, but anyons!