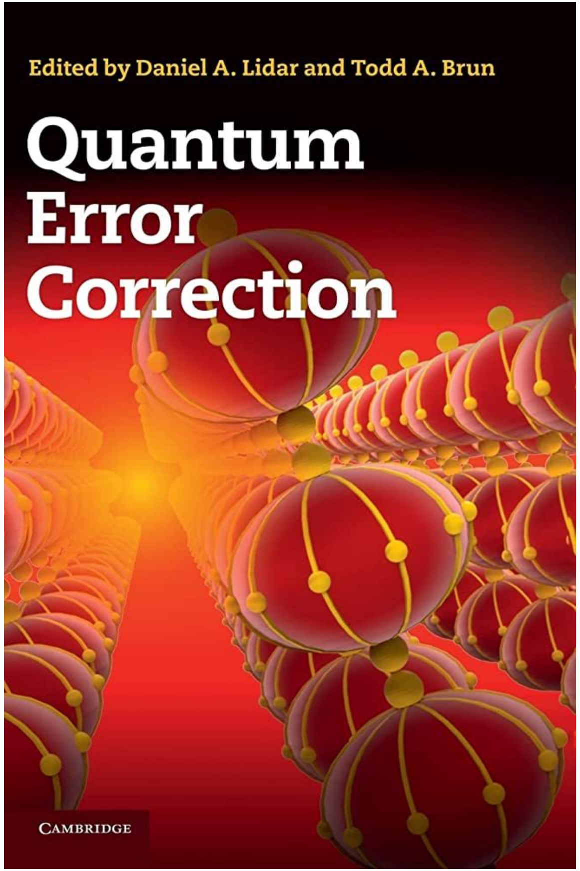
Practical Quantum Computing

Lecture 12
Surface Code, Lattice Surgery,
Decoding, Compiling Circuits

using slides from Austin Fowler and James Wootton

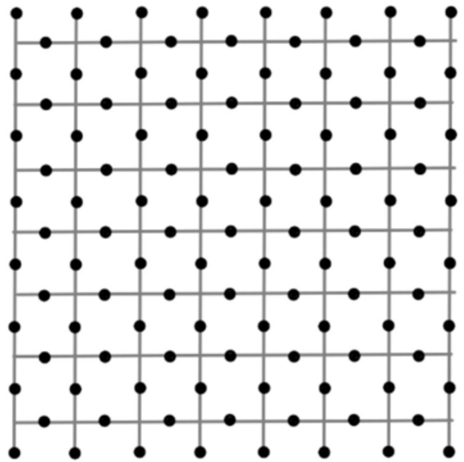
Reading materials

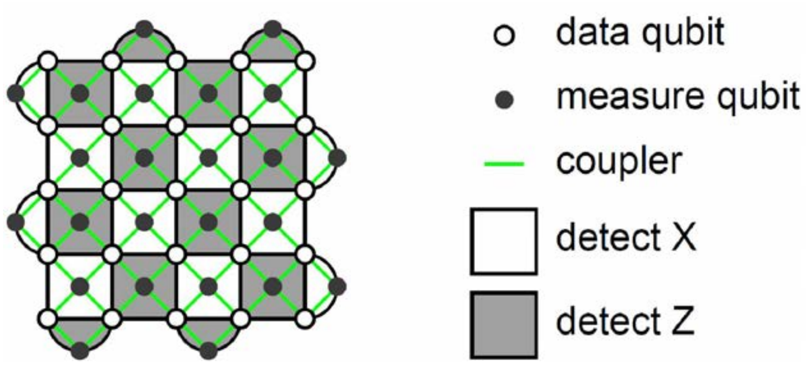
1. https://arxiv.org/abs/1208.0928
2. https://arxiv.org/pdf/1111.4022.pdf
3. https://arxiv.org/pdf/1808.02892.pdf
4. https://arxiv.org/abs/1302.3428
5. https://arxiv.org/abs/2205.09828

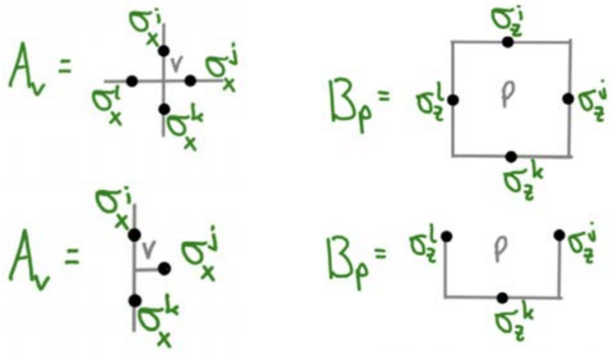


The Surface Code

- 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

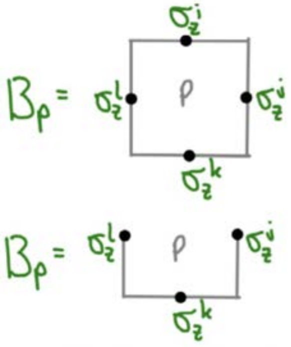


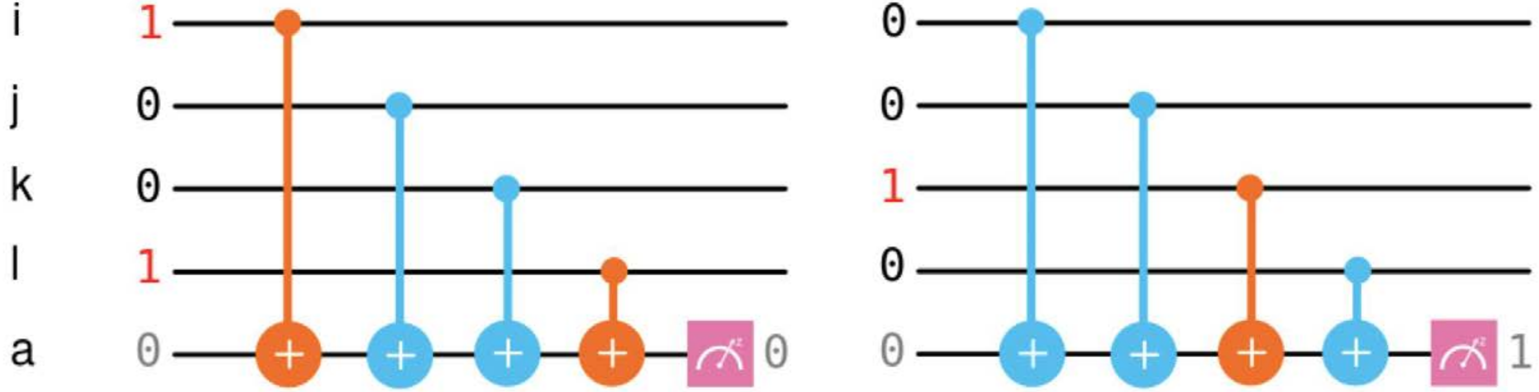




Plaquette Syndrome

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





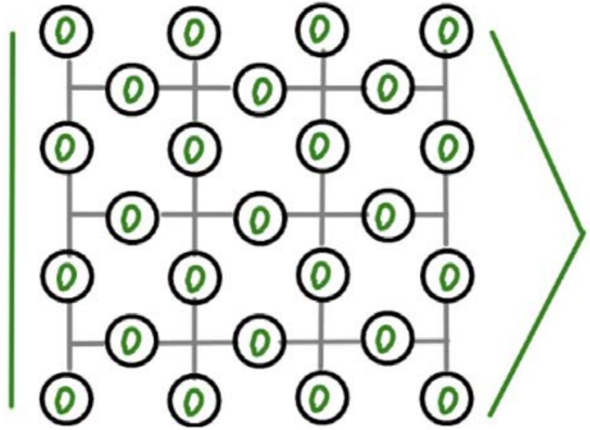
Plaquette Syndrome

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:

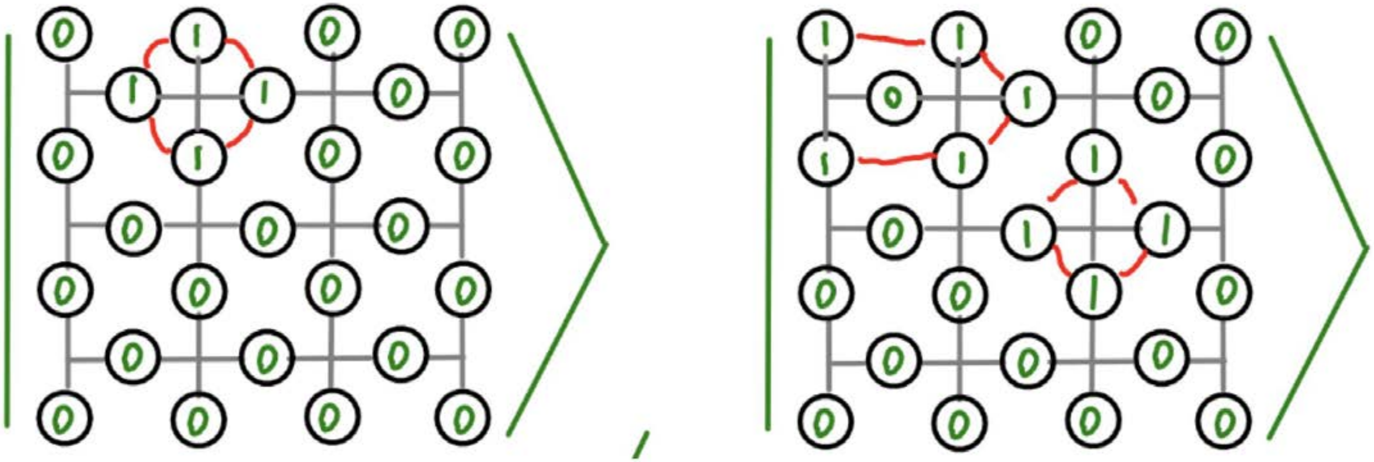
- Even parity on all plaquettes
- How to store a 0 in this?
- How about the state where every code qubit is $|0\rangle$?

$|0\rangle \rightarrow$



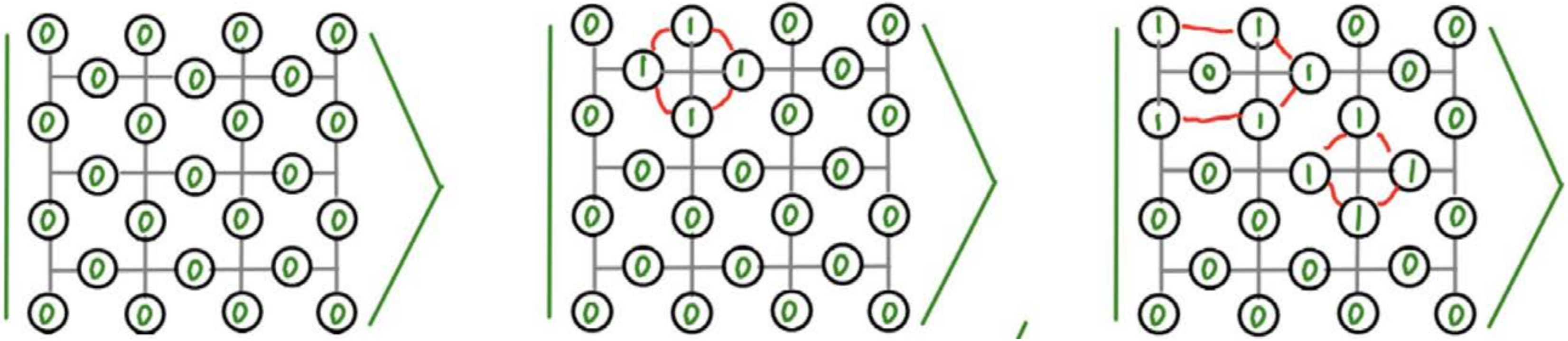
Plaquette Syndrome

- 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

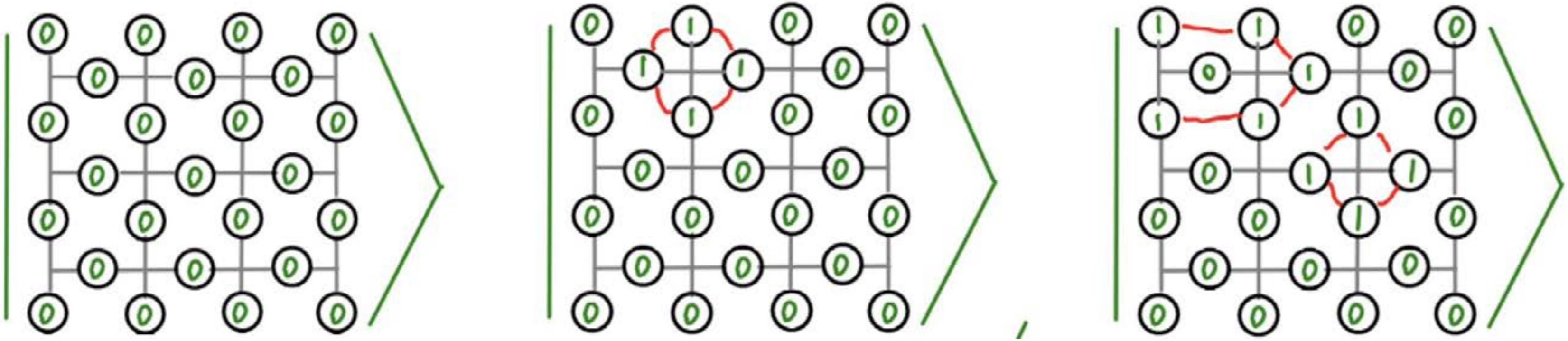
- Given any state for an encoded 0
  - Pick a vertex
  - Apply bit flips around that vertex
- Now you have another valid state for 0
- This generates an exponentially large family



Plaquette Syndrome

- The states in this family can be very different
- But they all share a common feature

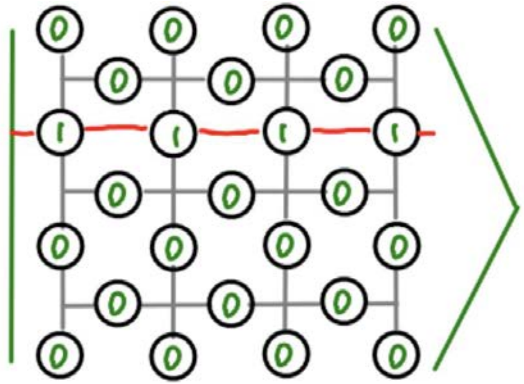
  Any line from top to bottom (passing along edges) has even parity
- This is how we can identify an encoded 0
- And it gives us a clue about how to encode a 1



Plaquette Syndrome

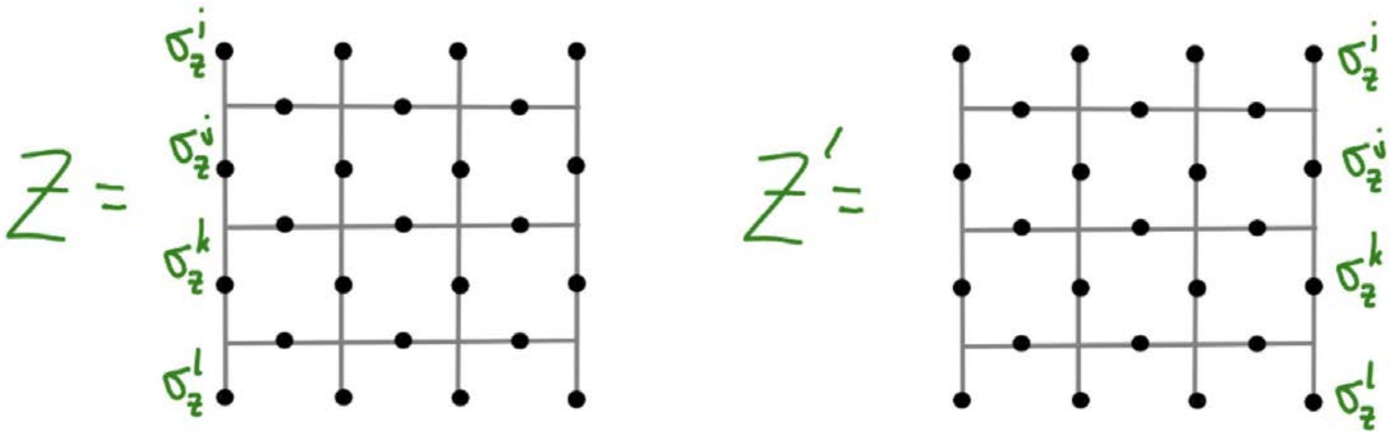
- For our basic encoded 1, we use a bunch of 0s with a line from left to right (passing through plaquettes)
- This also spawns an exponentially large family
- All have *odd* parity for a line from top to bottom
- Unlike the repetition code, distinguishing encoded 0 and 1 requires some effort (which is good!)

$|1\rangle \rightarrow$



Logical X and Z

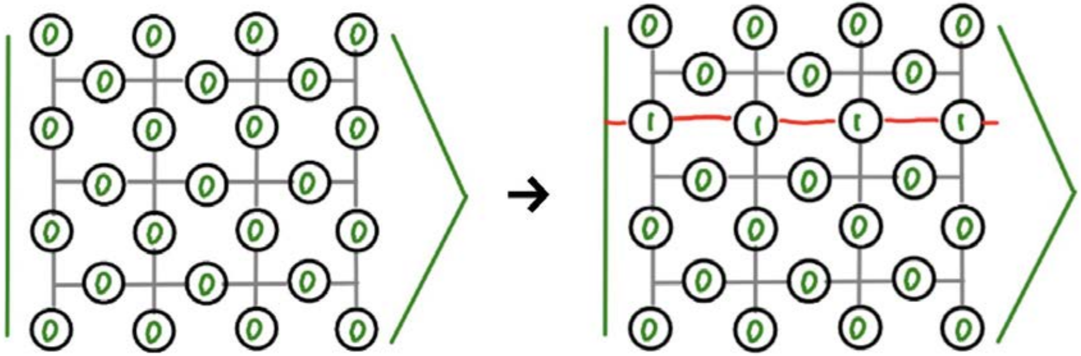
- Distinguishing 0 and 1 corresponds to measuring Z on the physical qubit
- The following observables detect what we need



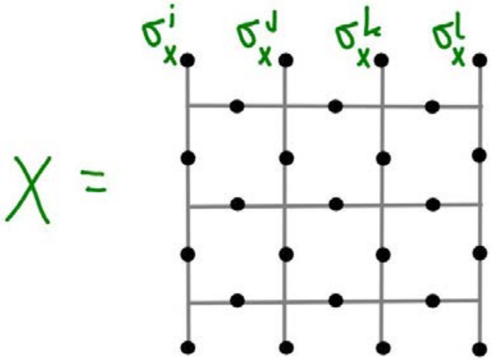
- Or the same on any line from top to bottom
- Uses the edges has a nice advantage: we can think of them as large (unenforced) plaquettes

Logical X and Z

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

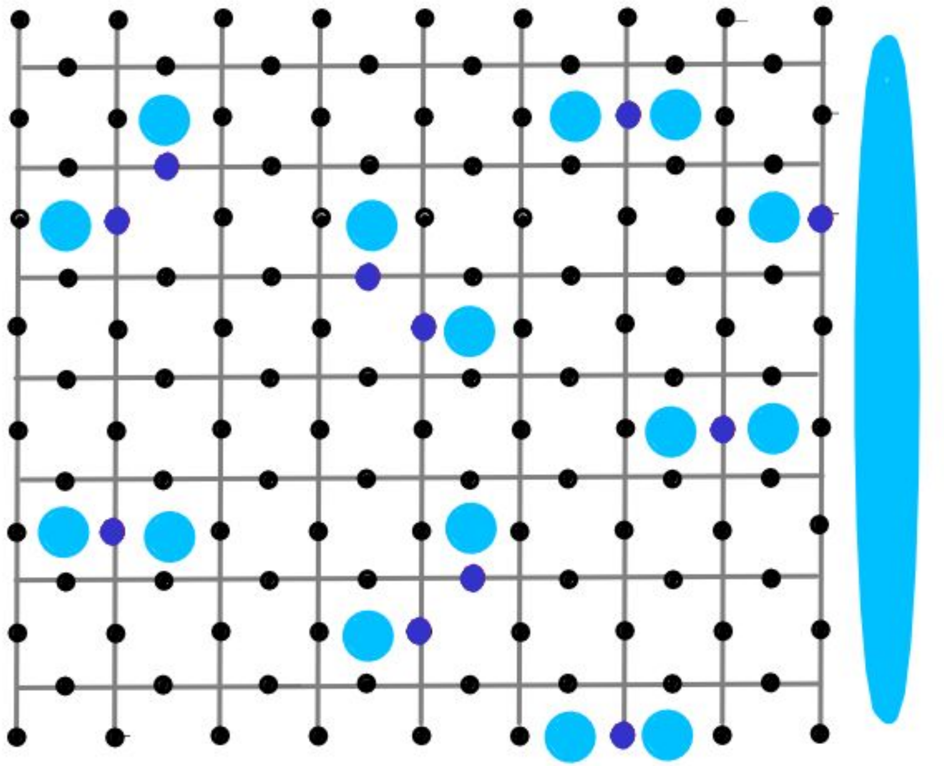


- Such lines of flips act as an X on the logical qubit



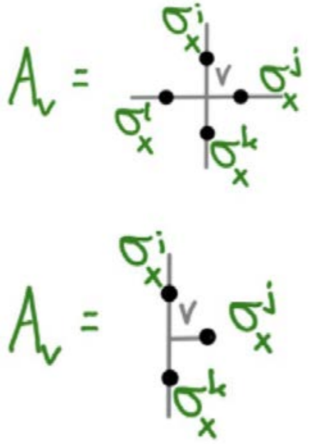
Effects of Errors

- 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

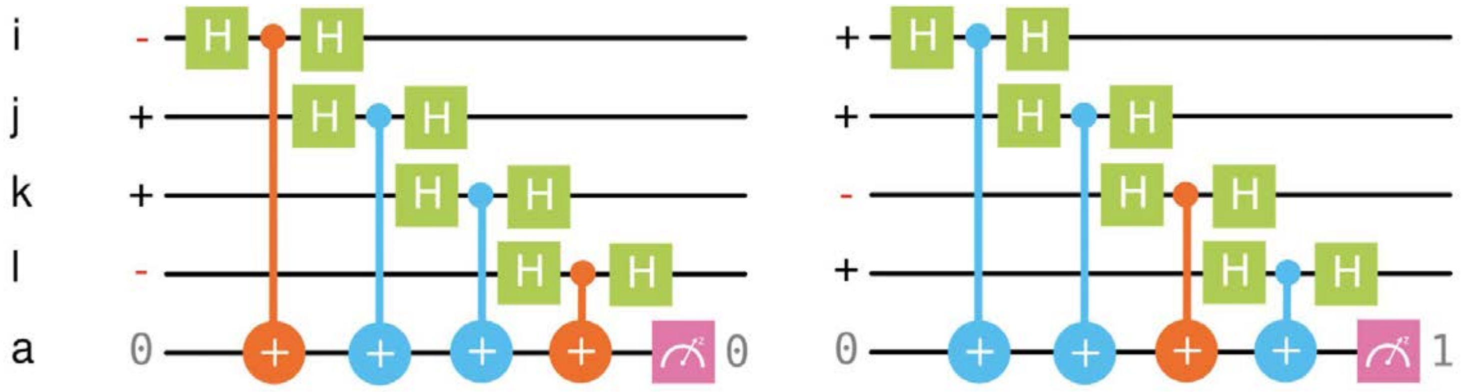


- With the plaquette operators, we can encode and protect a *bit*

Vertex Syndrome

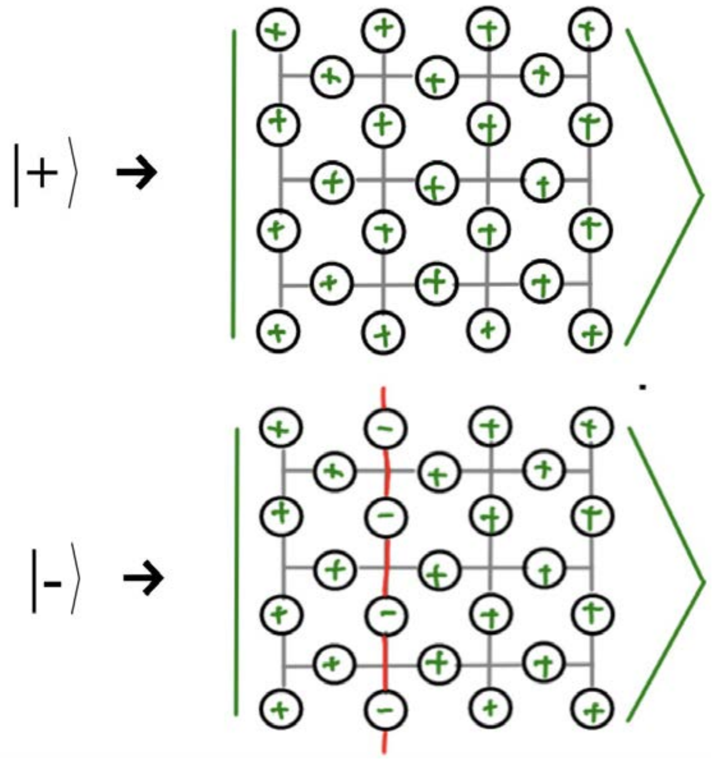


- Now forget the plaquettes and focus on vertices
- These observables can also be measured using CX gates an an ancilla
- In this case they look at the |+⟩ and |-⟩ states, and count the parity of the number of |-⟩s



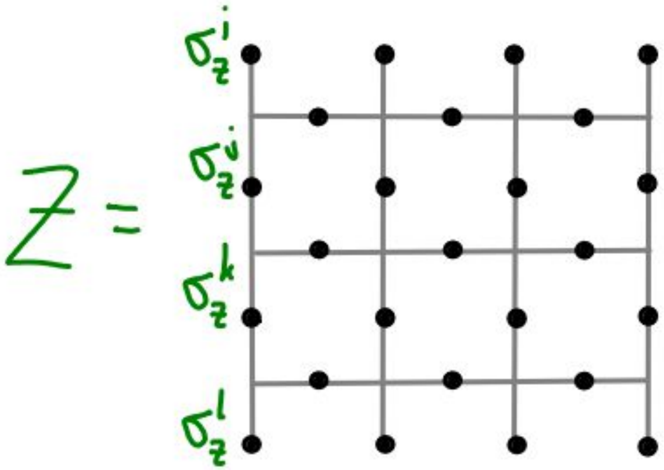
Vertex Syndrome

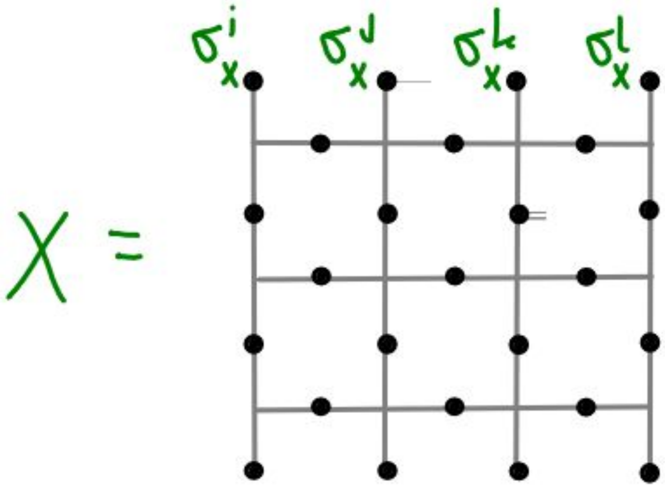
- These operators also allow us to encode and protect a bit value
- In this case, let's use + and - to label the two states
- They are encoded using suitable patterns of $|+\rangle$ and $|-\rangle$ states for the code qubits
- As with the plaquettes, these also correspond to exponentially large families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



Logical X and Z

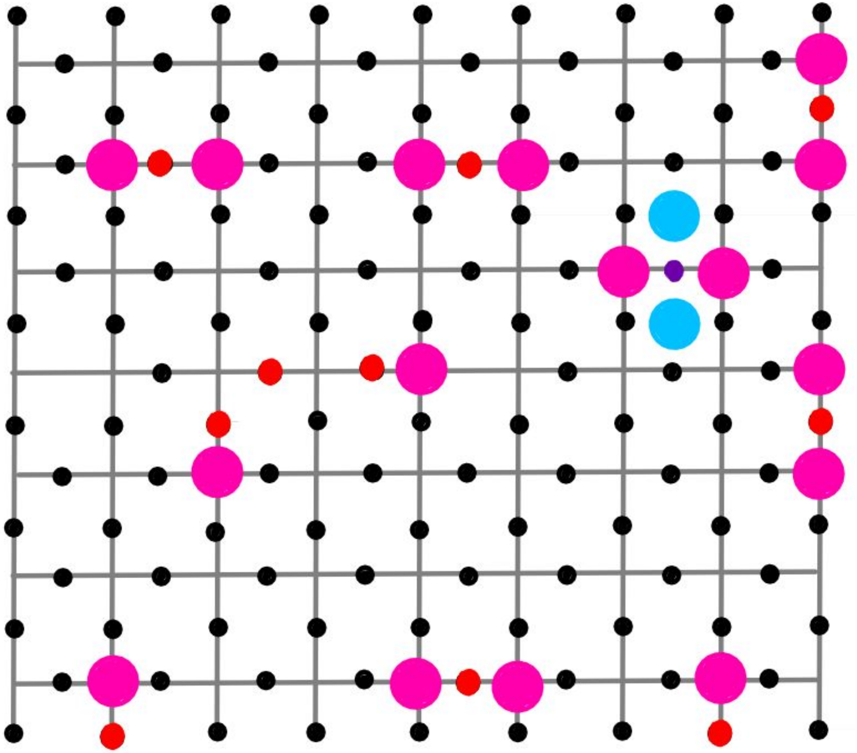
- What is the X operator (distinguish between |+⟩ and |−⟩)?
- What is the Z operator (flip between |+⟩ and |−⟩)?
- Turns out they are exactly the same as before!





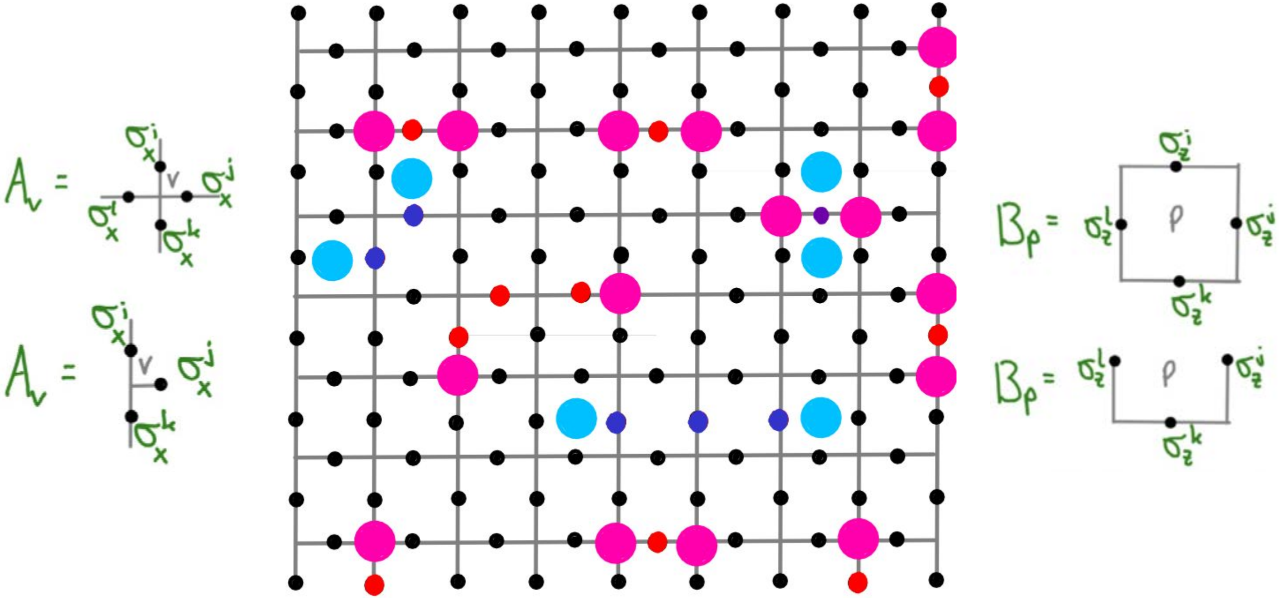
Effects of Errors

- 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*



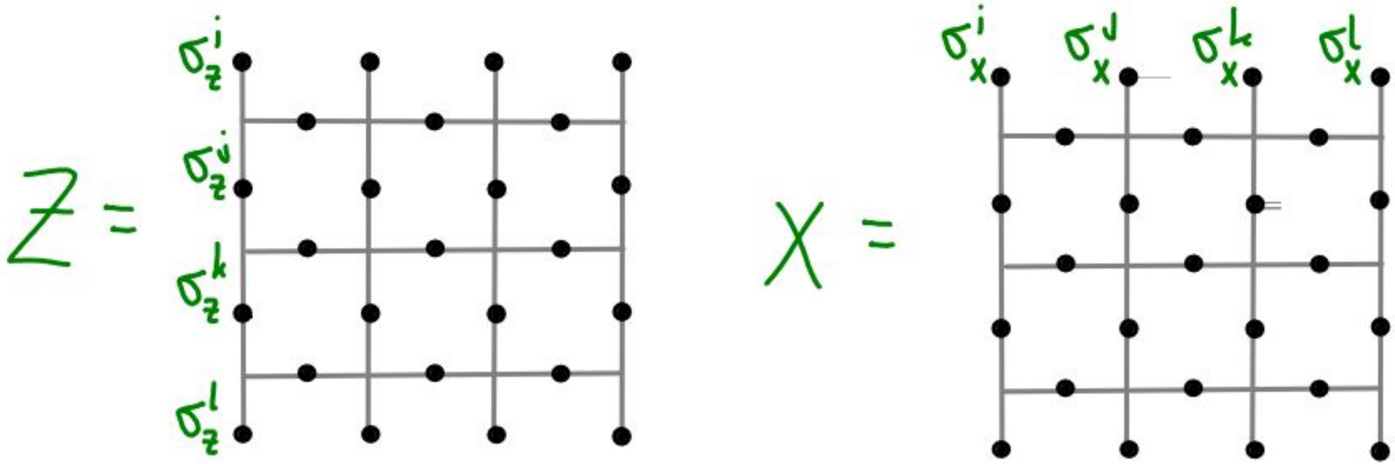
Putting it all Together

- 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

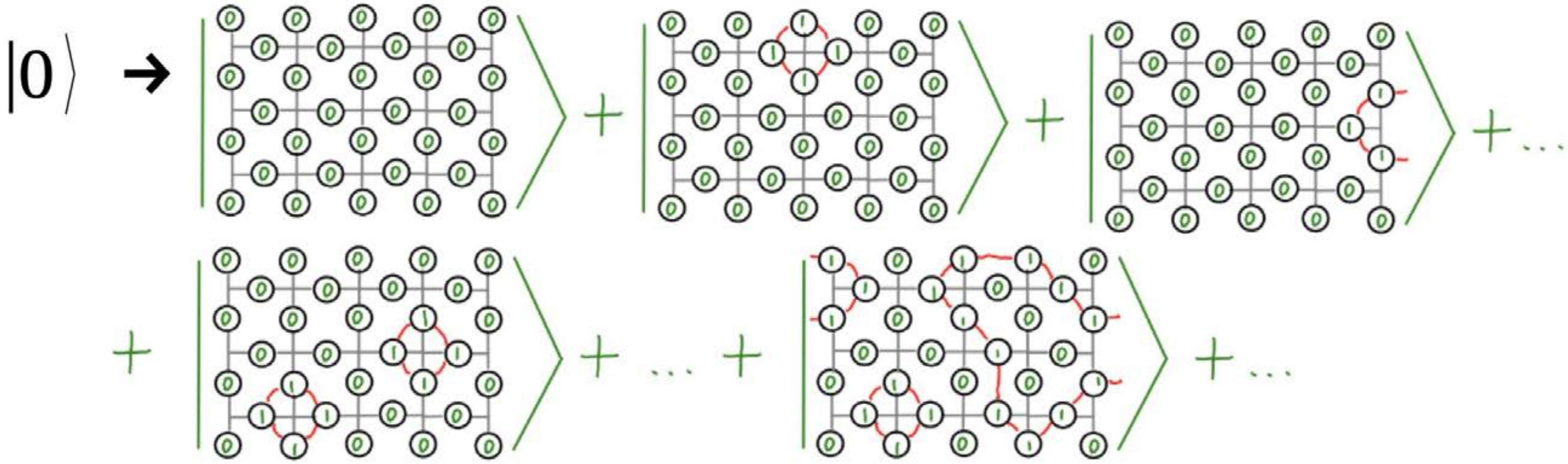
- The Z and X operators on the encoded qubit are exactly the same as before



- 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



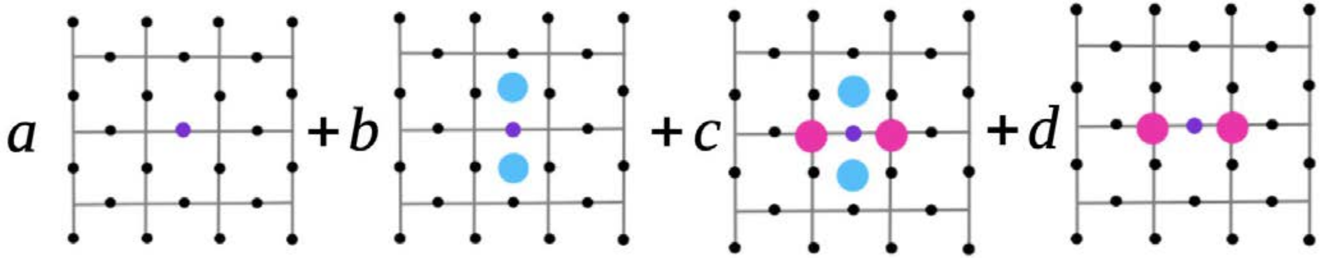
- Though such things can be hard to make, we create and protect them with the syndrome measurements

Putting it all Together

- 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
- Though such things can be hard to make, we create and protect them simply by making the stabilizer measurements



Lattice Surgery

Lattice Surgery

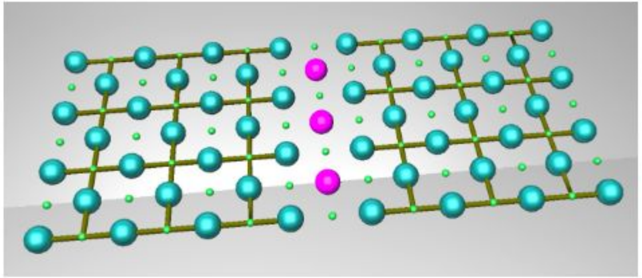


Figure 5. Arrangements of physical qubits for rough lattice merging. Left and right continuous surfaces encode separate logical qubits. The pink qubits form the intermediate qubit line for the merging operation.

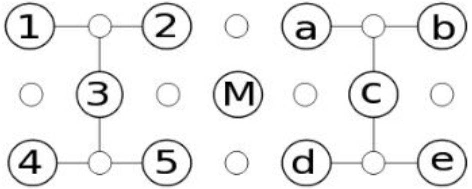


Figure A1. Lattice qubits for merging two rough surfaces of distance 2 into a single surface.

$\alpha\alpha'$ (A.1)

	1	2	3	4	5	M	a	b	c	d	e
	X	X	X								
			X	X	X						
	Z		Z	Z							
		Z	Z		Z						
	Z	Z									
						Z					
							Z	Z			
							Z		Z	Z	
								Z	Z		Z
							X	X	X		
									X	X	X

The stabilizers $X_2 X_M X_a$ are measured across the join to merge the surfaces, with

$\alpha\alpha'$ (A.3)

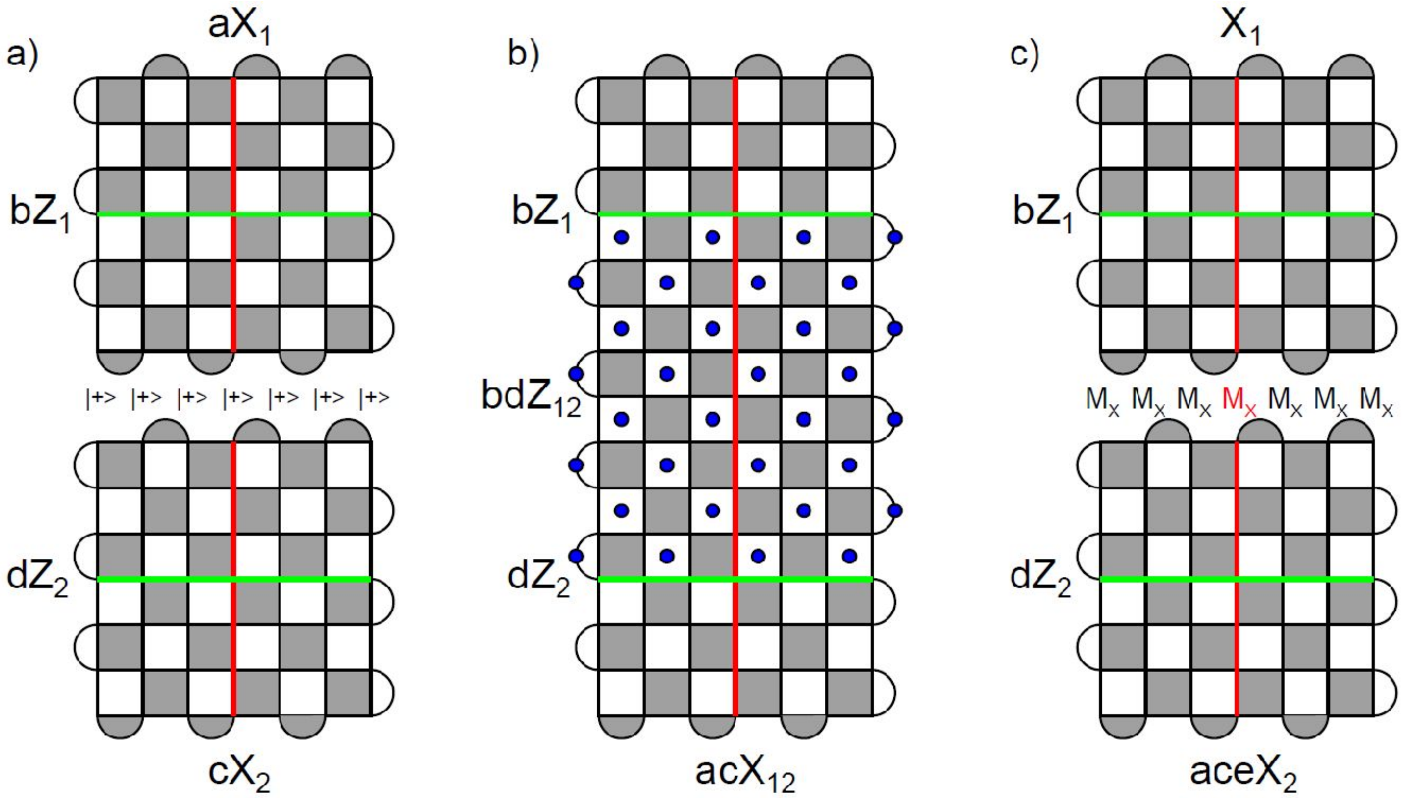
	1	2	3	4	5	M	a	b	c	d	e
$(-1)^m$		X				X	X				
	X	X	X								
			X	X	X						
							X	X	X		
									X	X	X
	Z		Z	Z							
		Z	Z		Z	Z					
						Z	Z	Z			
						Z	Z		Z	Z	
								Z	Z		Z
	Z	Z					Z	Z			

The stabilizer $X_5 X_M X_d$ across the join is now measured, with outcome m', leaving the state as

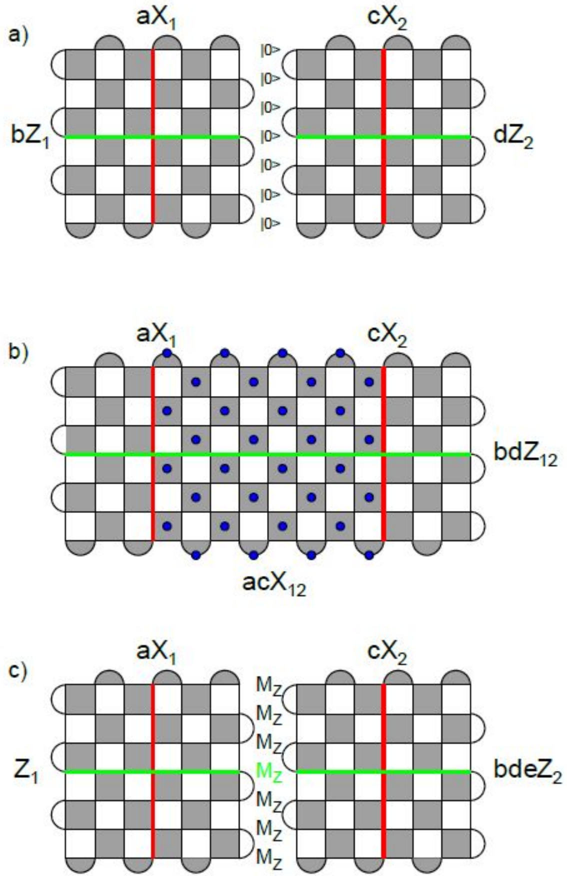
$\alpha\alpha'$ (A.4)

	1	2	3	4	5	M	a	b	c	d	e
$(-1)^m$		X				X	X				
$(-1)^{m'}$					X	X				X	
	X	X	X								
			X	X	X						
							X	X	X		
									X	X	X
	Z		Z	Z							
		Z	Z		Z	Z					
						Z	Z		Z	Z	
								Z	Z		Z
	Z	Z					Z	Z			

arXiv:1808.06709, arXiv:1808.02892



Logical XX measurement



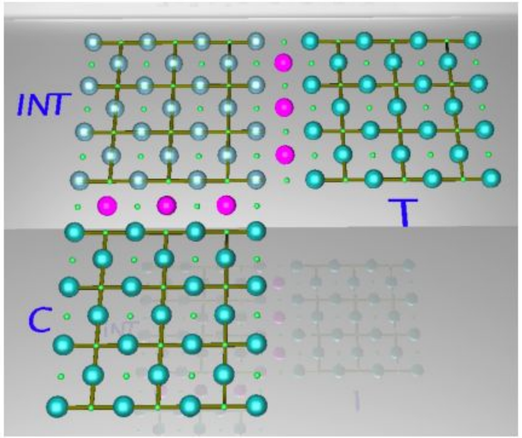


Figure 7. Layout of qubits for a CNOT operation with lattice surgery. Control (C) and target (T) surfaces interact by merging and splitting with the intermediate surface (INT).

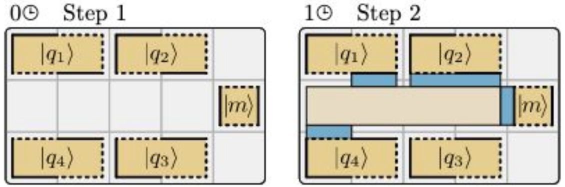


Figure 8: Example of a $Z_{|q_1\rangle} \otimes Y_{|q_2\rangle} \otimes X_{|q_4\rangle} \otimes Z_{|m\rangle}$ measurement to implement a $(Z \otimes Y \otimes \mathbb{1} \otimes X)_{\pi/8}$ gate.

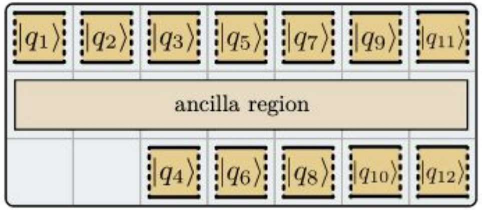
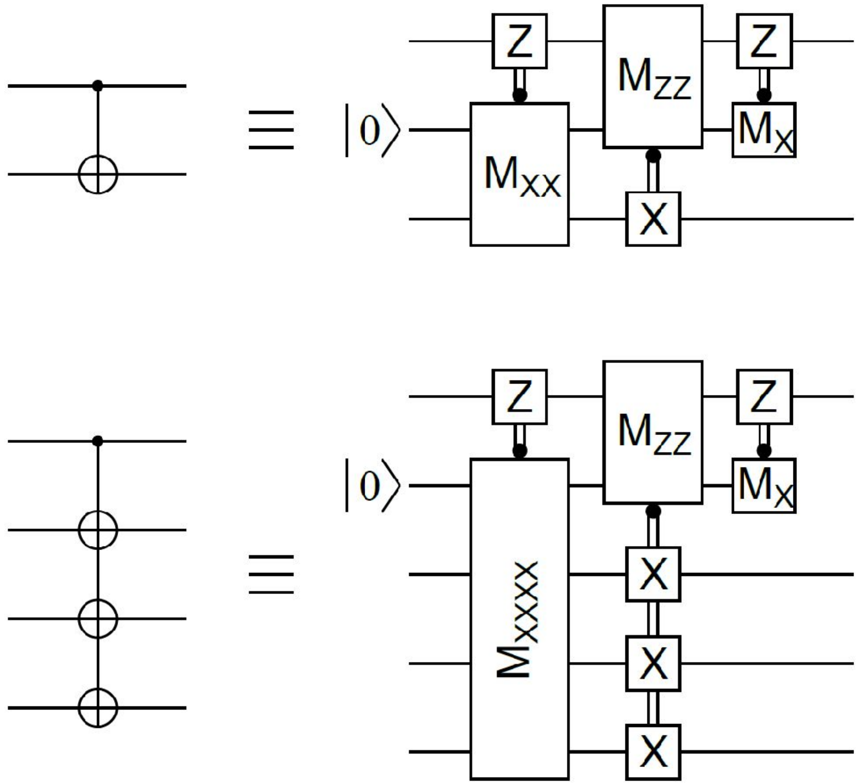


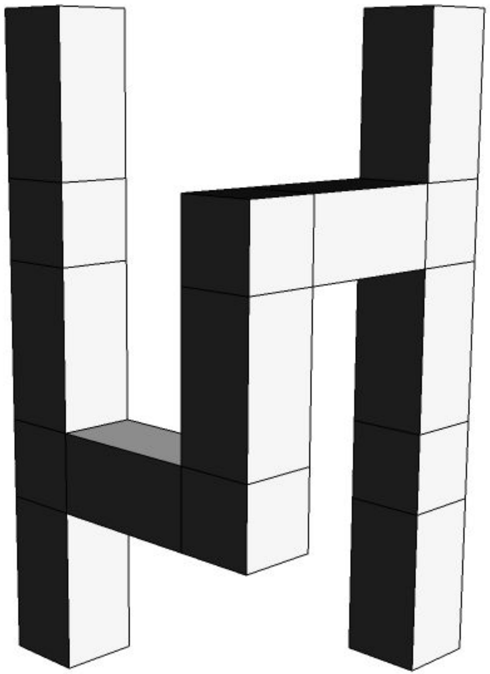
Figure 9: A compact block stores n data qubits in $1.5n + 3$ tiles. The consumption of a magic state can take up to 9⏲.

Logical CNOT

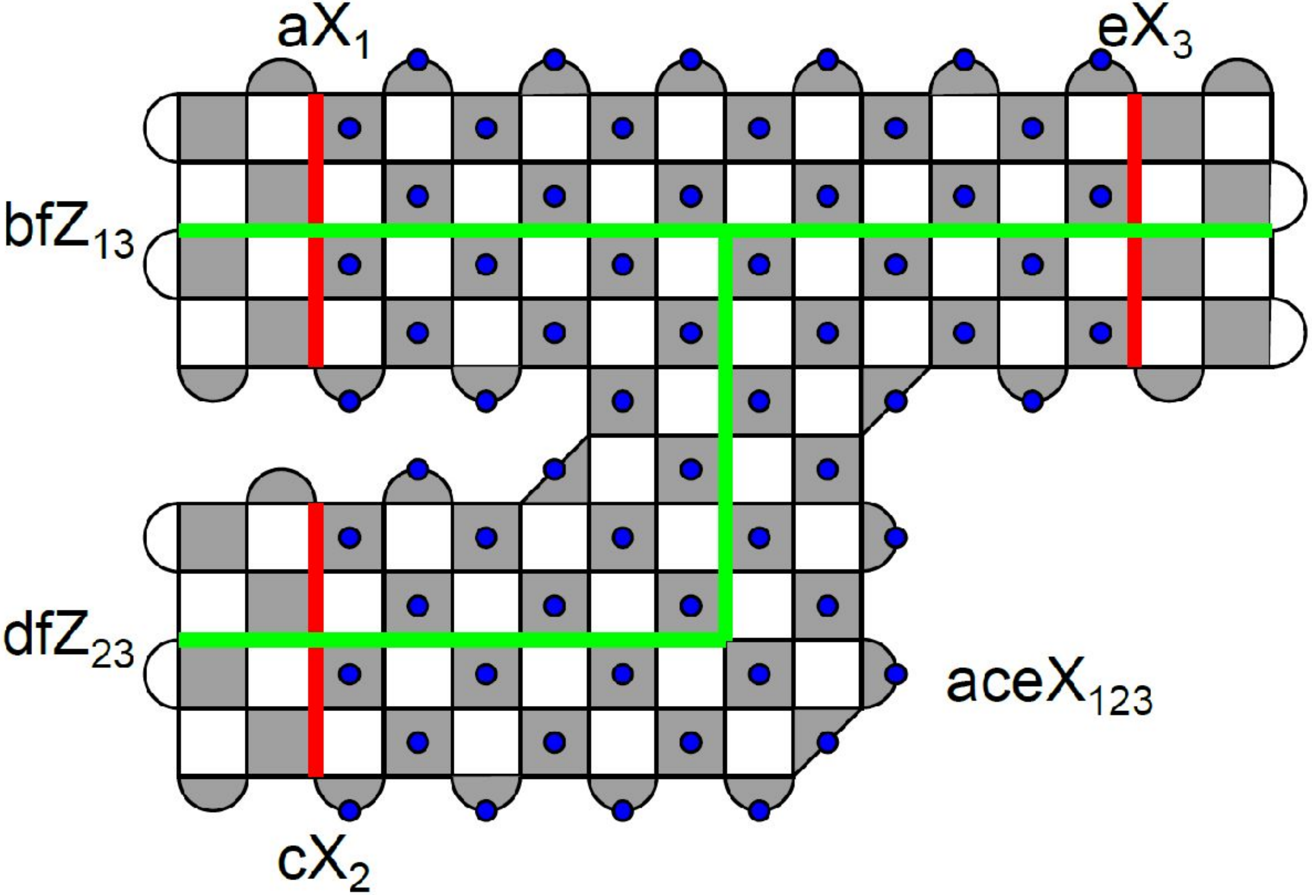
a)

b)



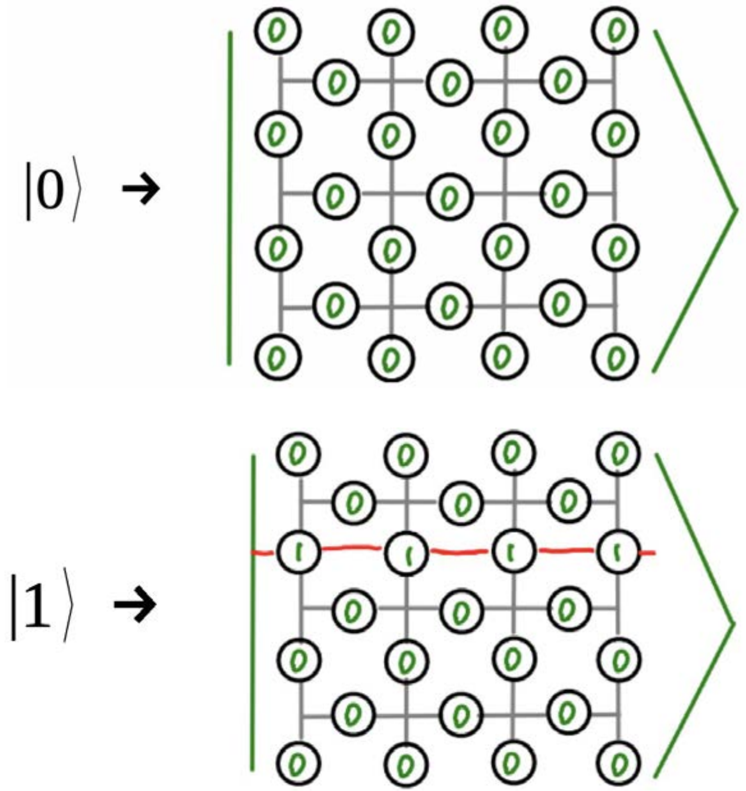


Multi-body logical X measurement



More Logical Gates

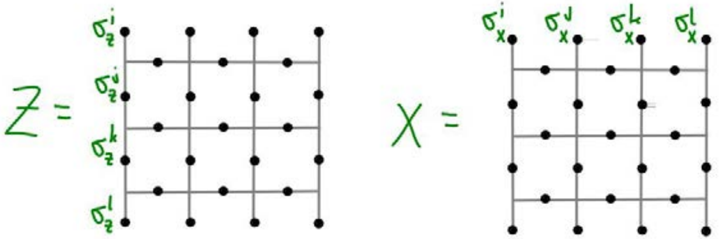
- We've seen how to do logical X and Z
- A logical CX can be done without much trouble
- A logical H requires the lattice to be rotated, but that can be done
- Other logical Clifford gates can be done with some crazy tricks
- But that's all! No other logical operations can be done fault-tolerantly.
-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

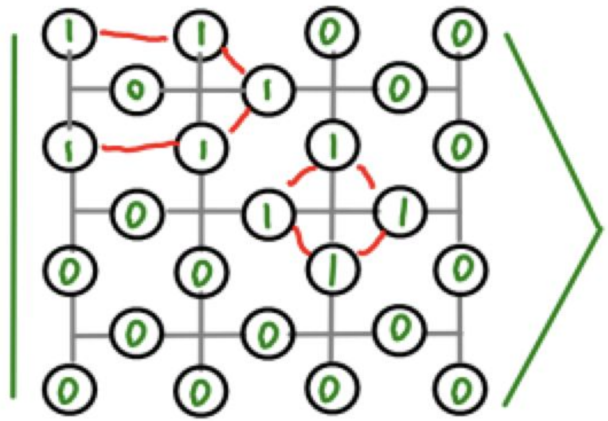


Decoding

Final Readout

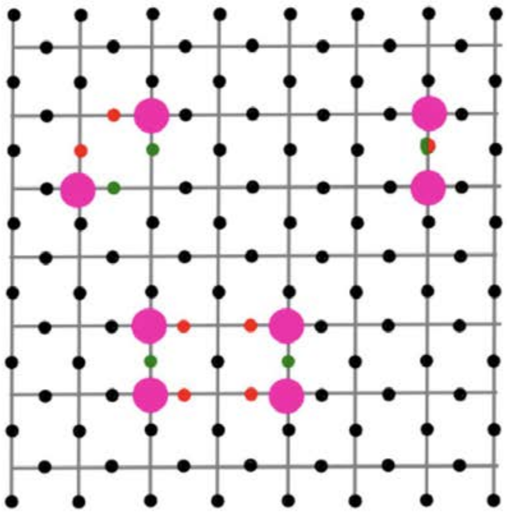
- The logical operators are many-body observables
- So how do we read them out fault-tolerantly
- When you decide on a basis for final measurement, you stop caring about some errors
- You can then measurement in a product basis
- Final readout and final stabilizer measurement can be constructed from the result
- Measurement errors are effectively the same as errors before measurement

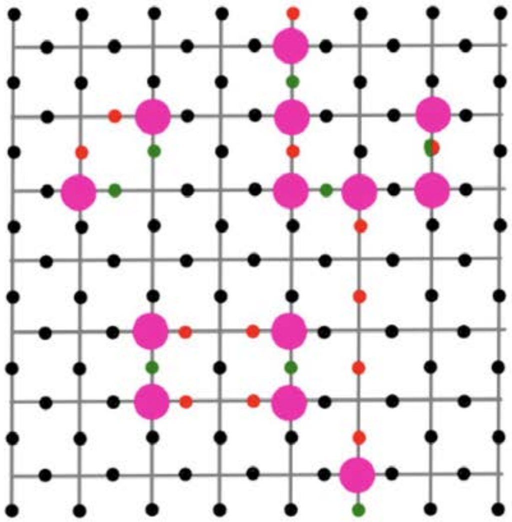




Decoding

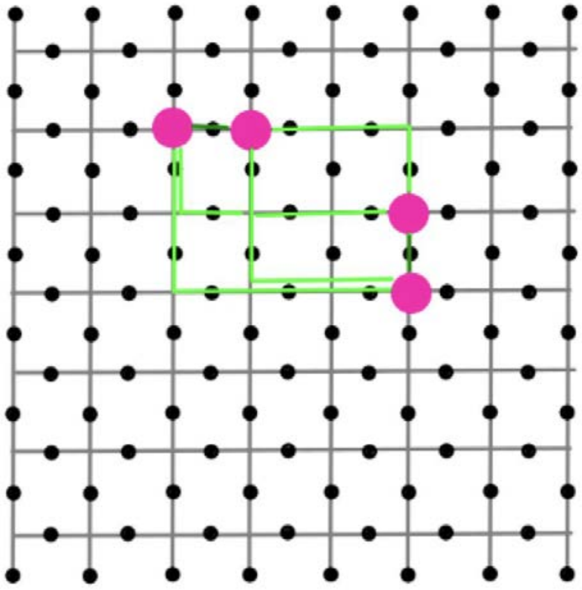
- Given the measurement results, we need to work out what errors happened
- More specifically, the 'equivalence class' of errors
- This is the job of the decoding algorithm



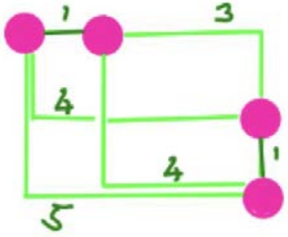


Decoding with MWPM

- A good option is Minimum Weight Perfect Matching
- 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

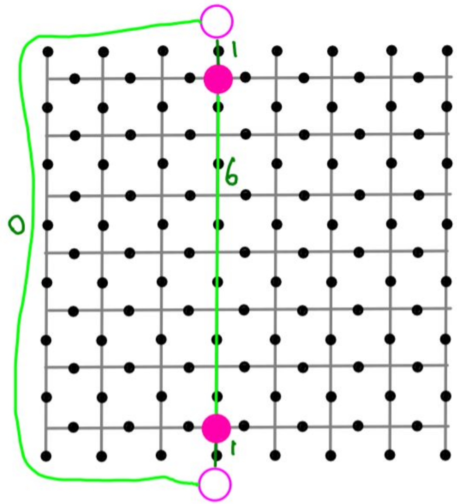


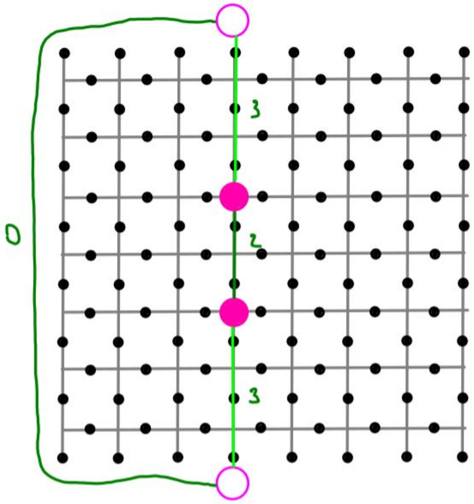
Manhattan distance between vertices



Decoding

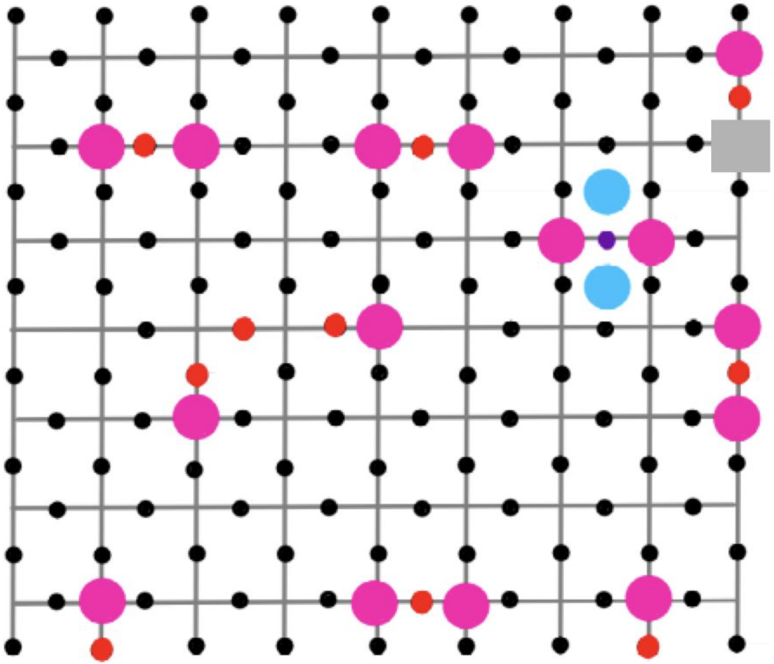
- We need to be careful to account for the effects of the edges
- This is done by introducing extra 'virtual nodes'



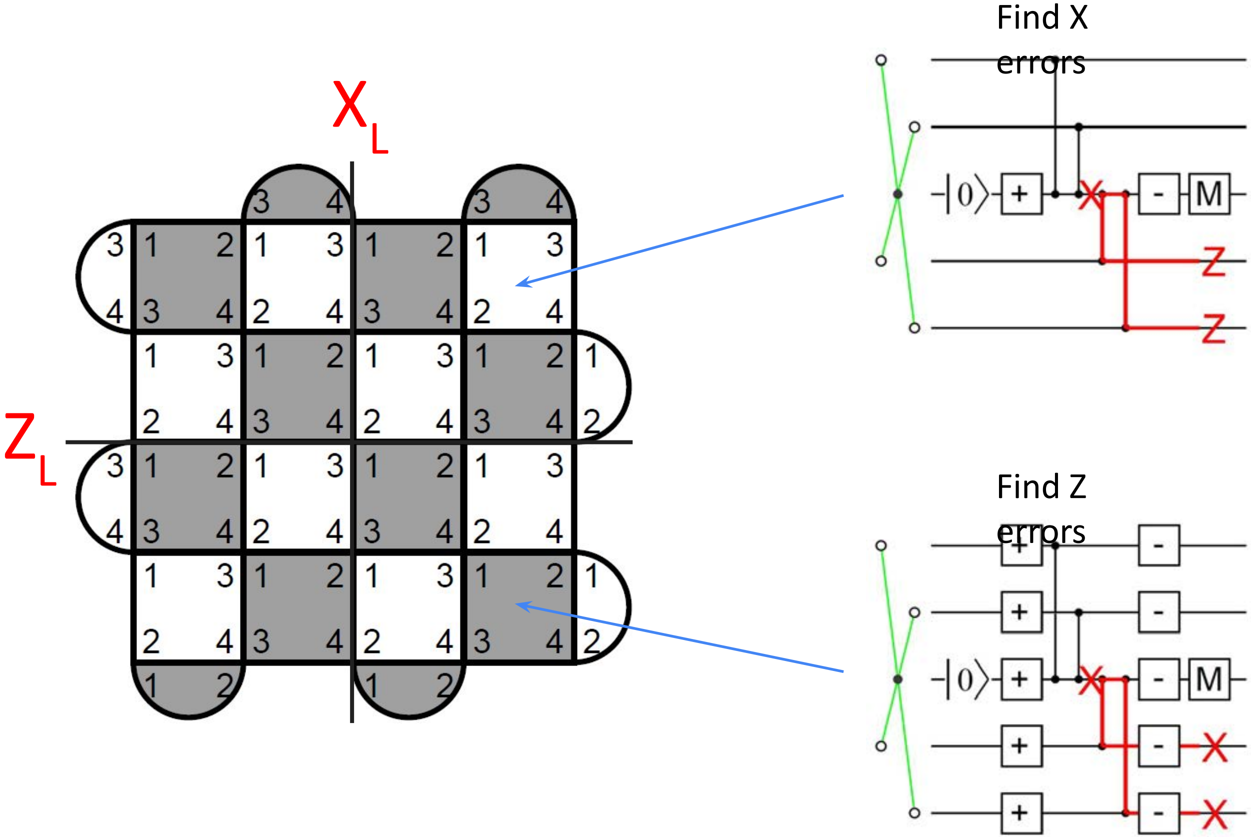


Imperfect Measurements

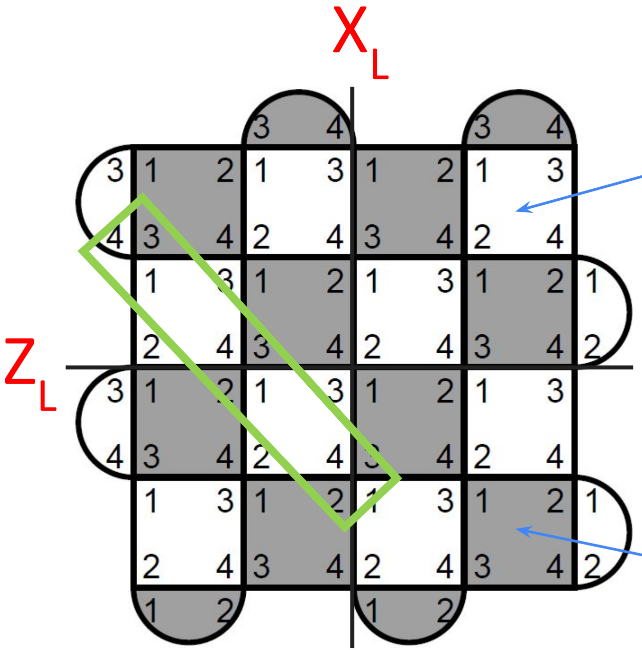
- We have the problem of imperfect measurements
  - The measurements might lie
  - Errors on the additional qubit
  - Errors in the CX gates
- We base the decoding using syndrome changes
- This leads to a 3D MWPM problem (2D space + time)

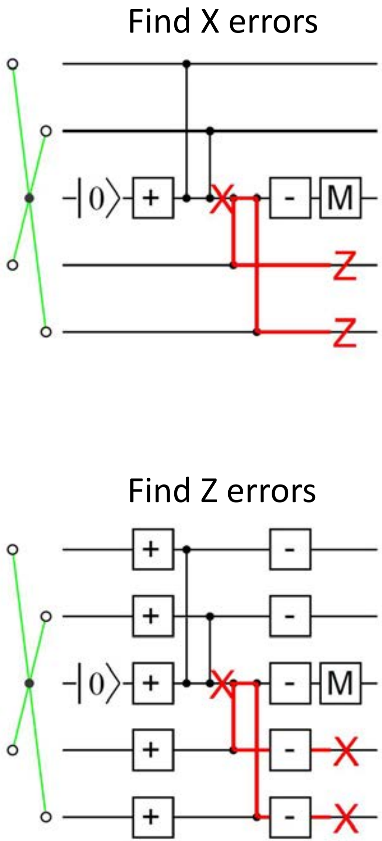


Gate sequence:

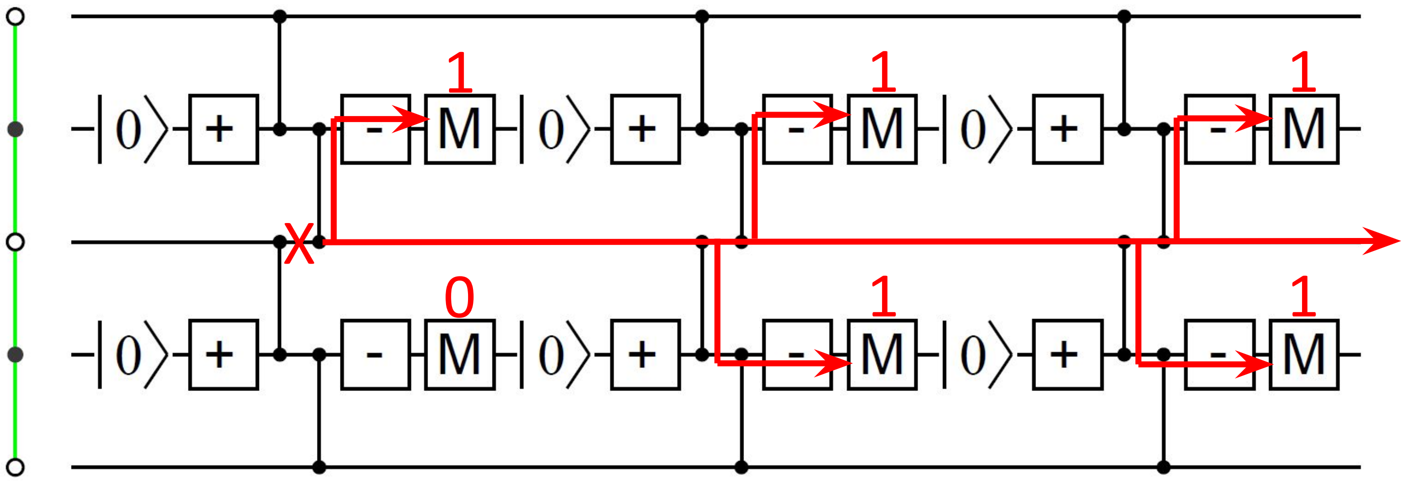


Gate sequence:



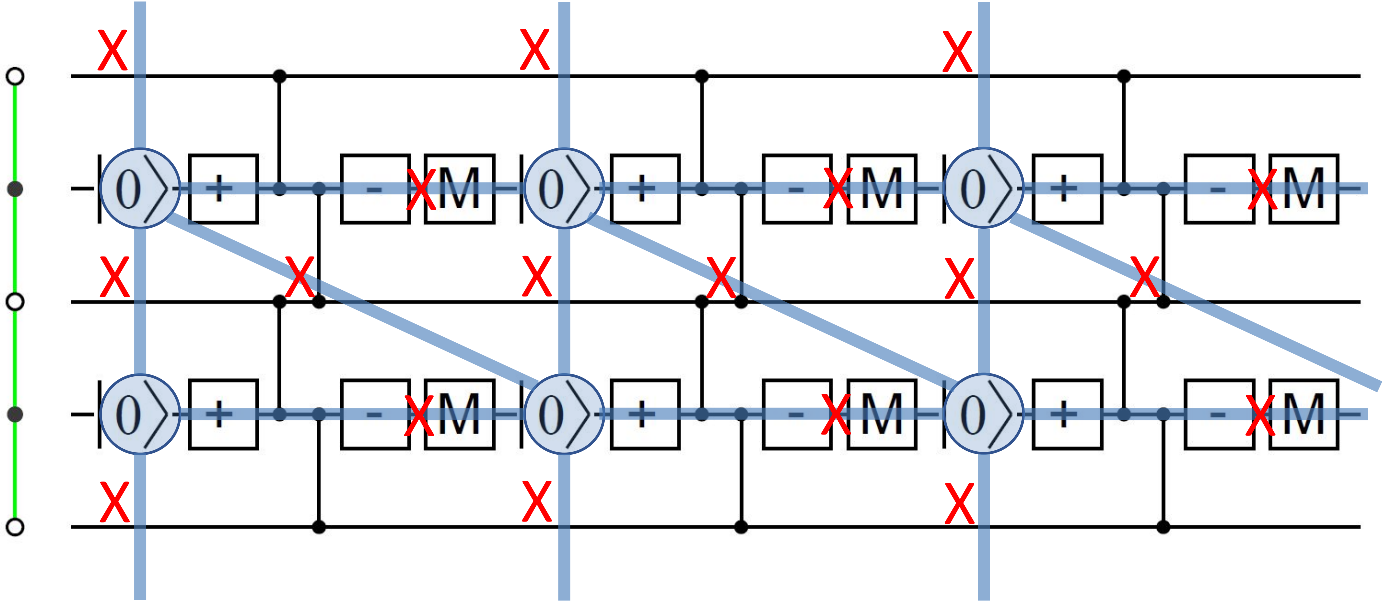


How to do memory:



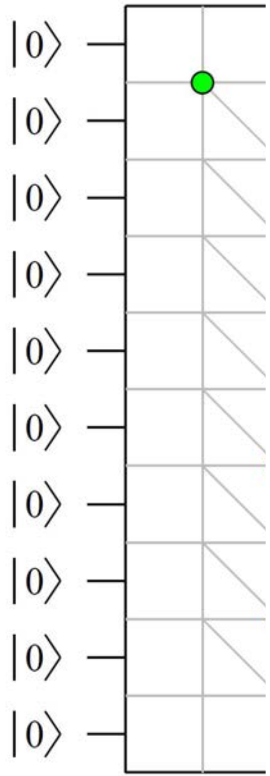
Measurement value change = detection event

How to do memory:



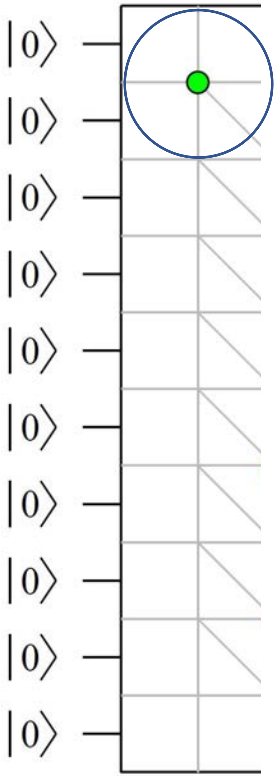
Build graph of all possible detection events

Classical processing



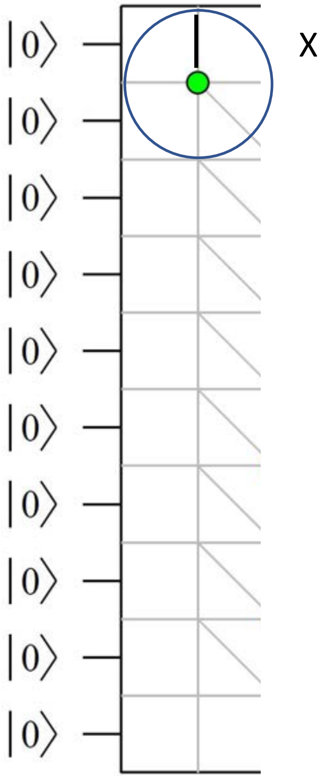
- 10 data qubits
- One detection event

Classical processing



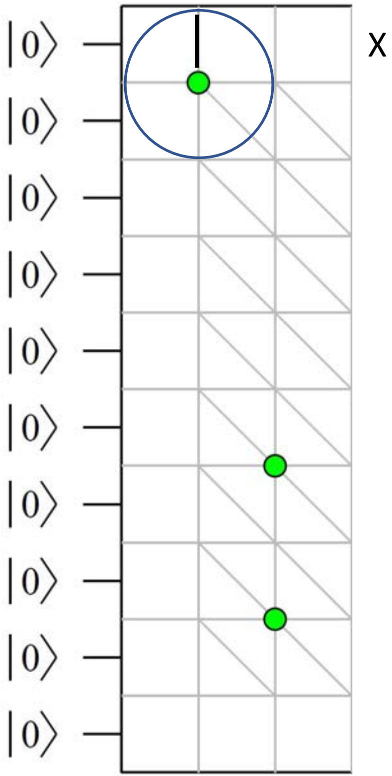
- 10 data qubits
- One detection event
- Explore uniformly, boundary found

Classical processing



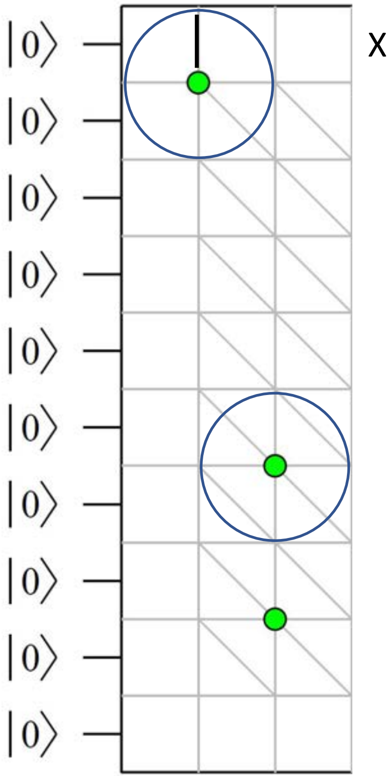
- 10 data qubits
- One detection event
- Explore uniformly, boundary found
- Match detection event to boundary, record belief that X error present

Classical processing



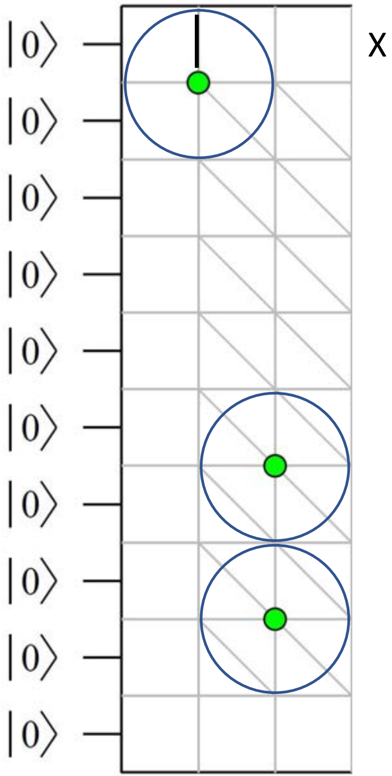
- Two more detection events

Classical processing



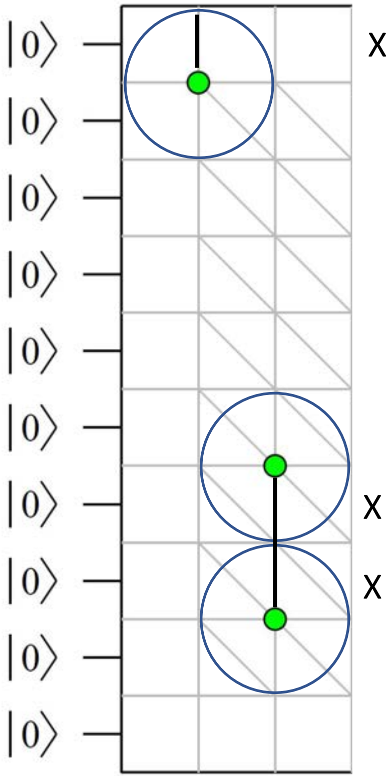
- Two more detection events
- Pick one, explore, current time boundary encountered

Classical processing



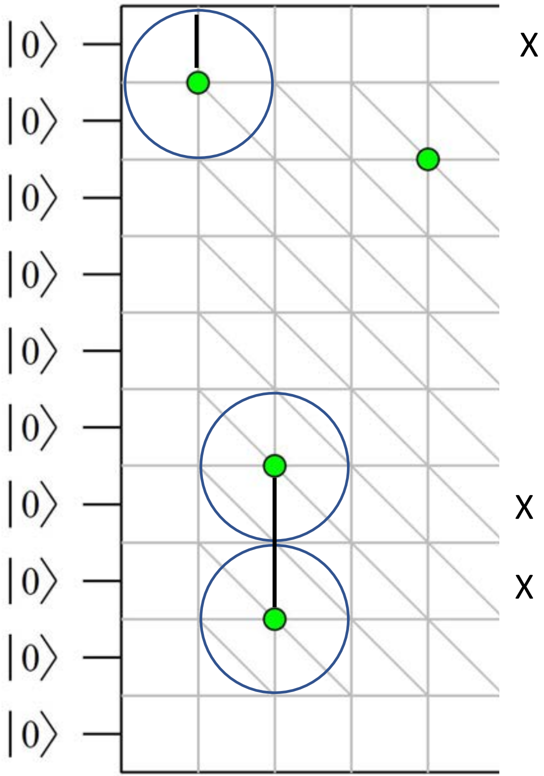
- Two more detection events
- Pick one, explore, current time boundary encountered
- Explore around other, exploratory regions touch

Classical processing



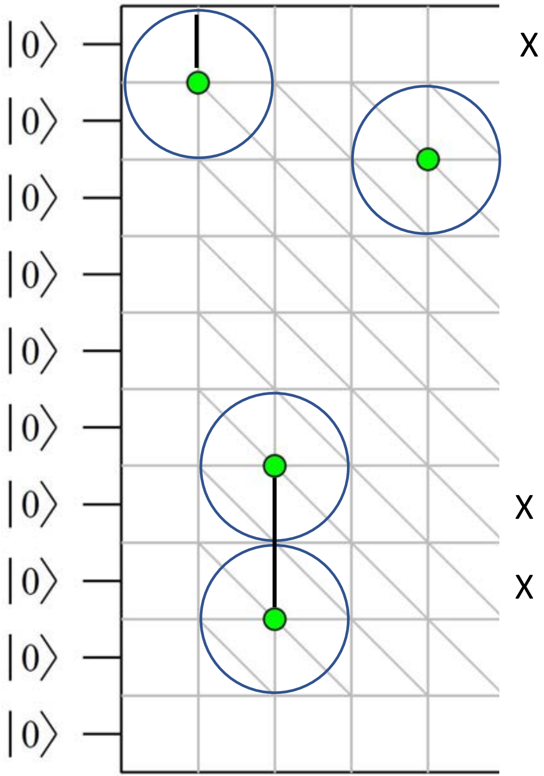
- Two more detection events
- Pick one, explore, current time boundary encountered
- Explore around other, exploratory regions touch
- Match, record belief that two more X errors present

Classical processing



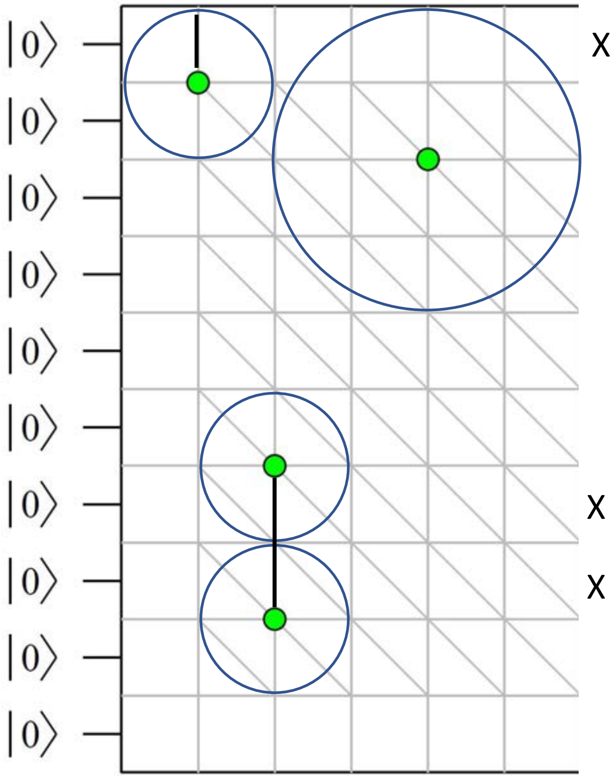
- One more detection event

Classical processing



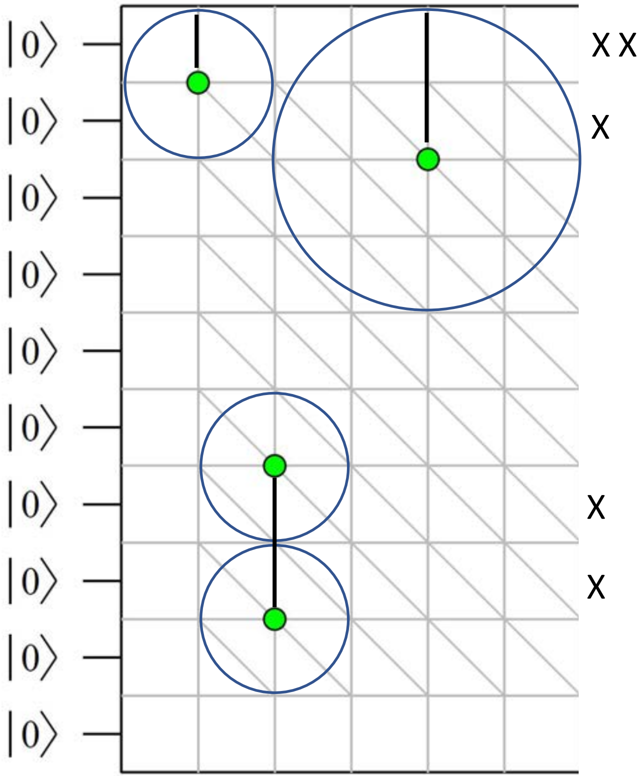
- One more detection event
- Explore, current time boundary encountered, must wait for more data

Classical processing



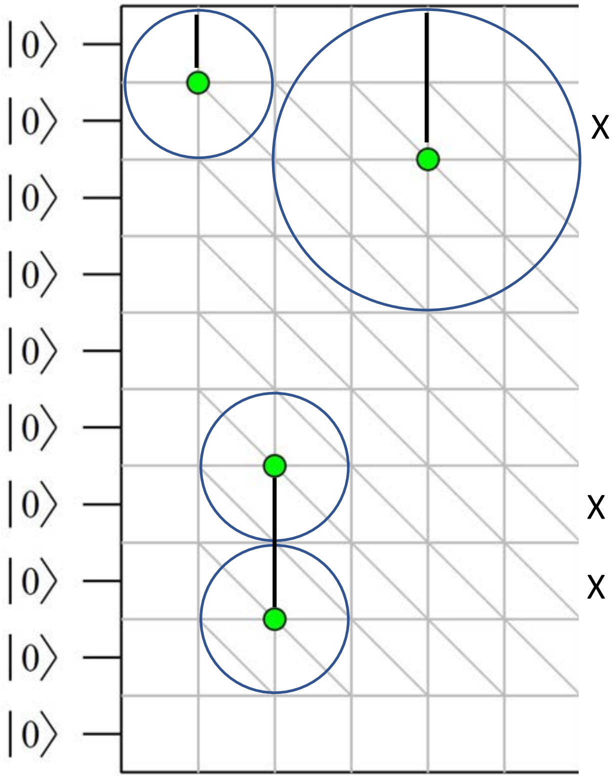
- One more detection event
- Explore, current time boundary encountered, must wait for more data
- Explore further, boundary encountered

Classical processing



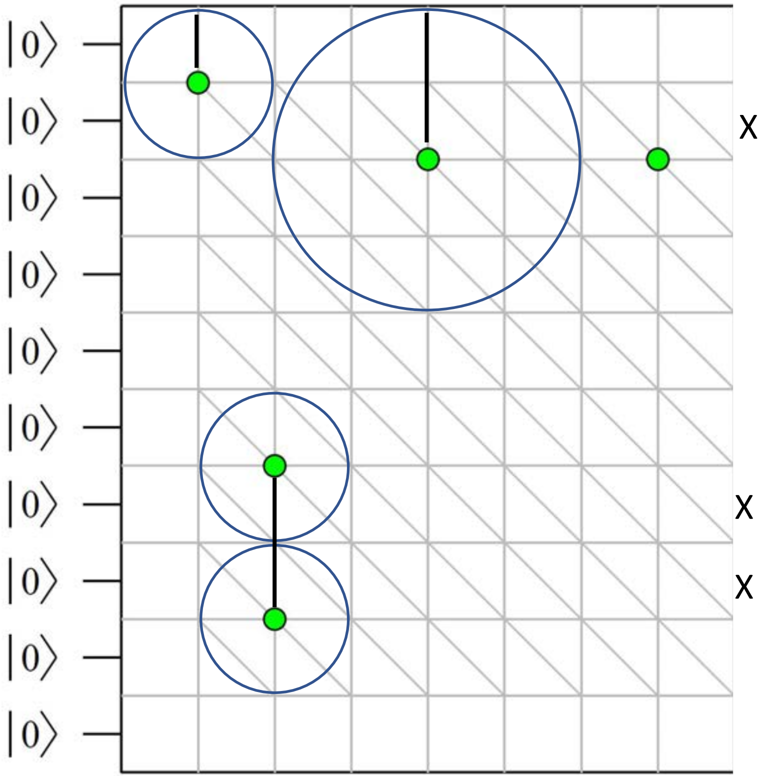
- One more detection event
- Explore, current time boundary encountered, must wait for more data
- Explore further, boundary encountered
- Match, record belief that two more X errors present

Classical processing

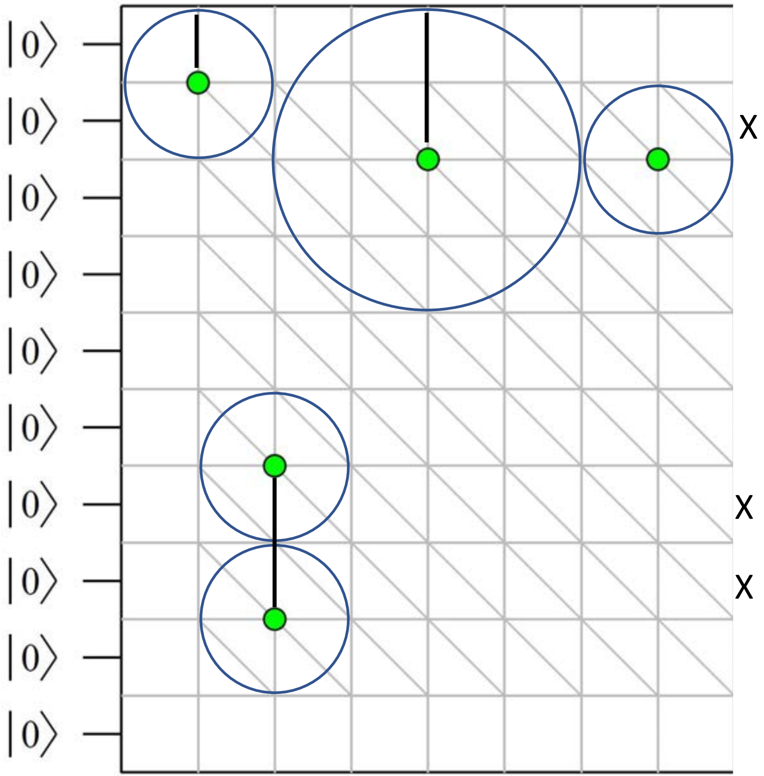


- One more detection event
- Explore, current time boundary encountered, must wait for more data
- Explore further, boundary encountered
- Match, record belief that two more X errors present
- Cancel double error
- **Don’t apply physical corrections**

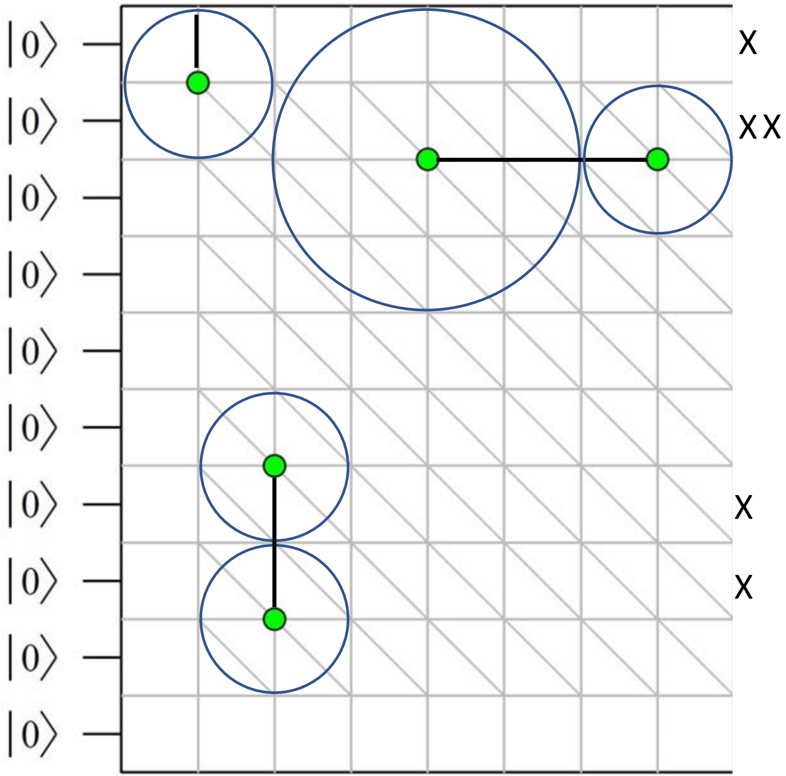
Classical processing



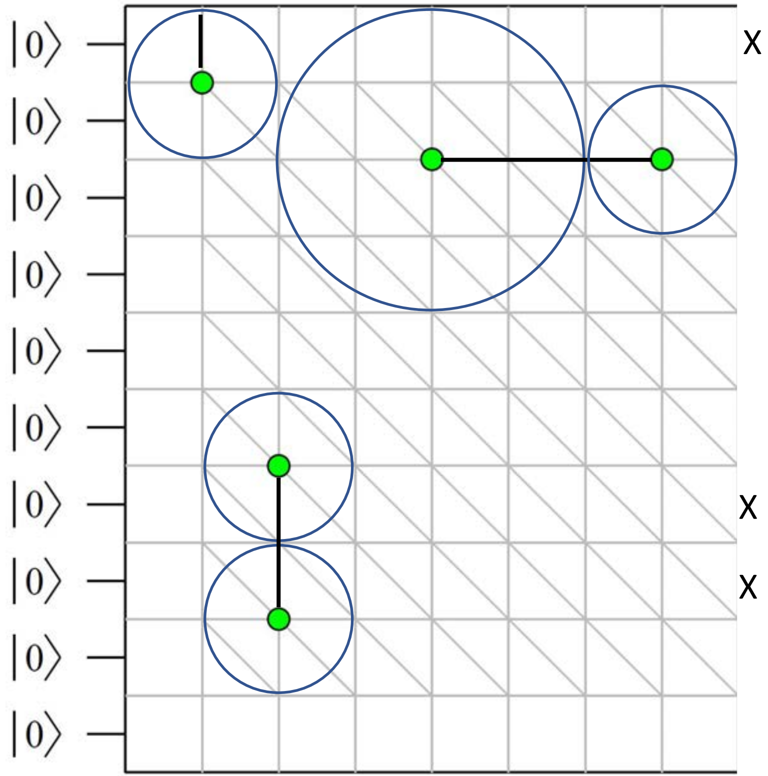
Classical processing



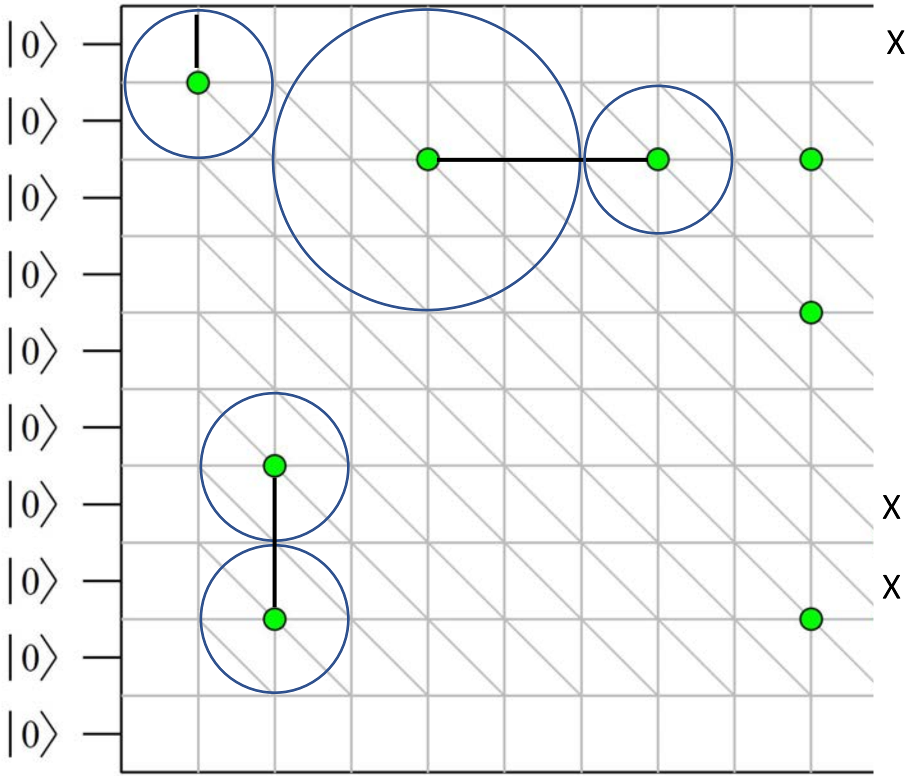
Classical processing



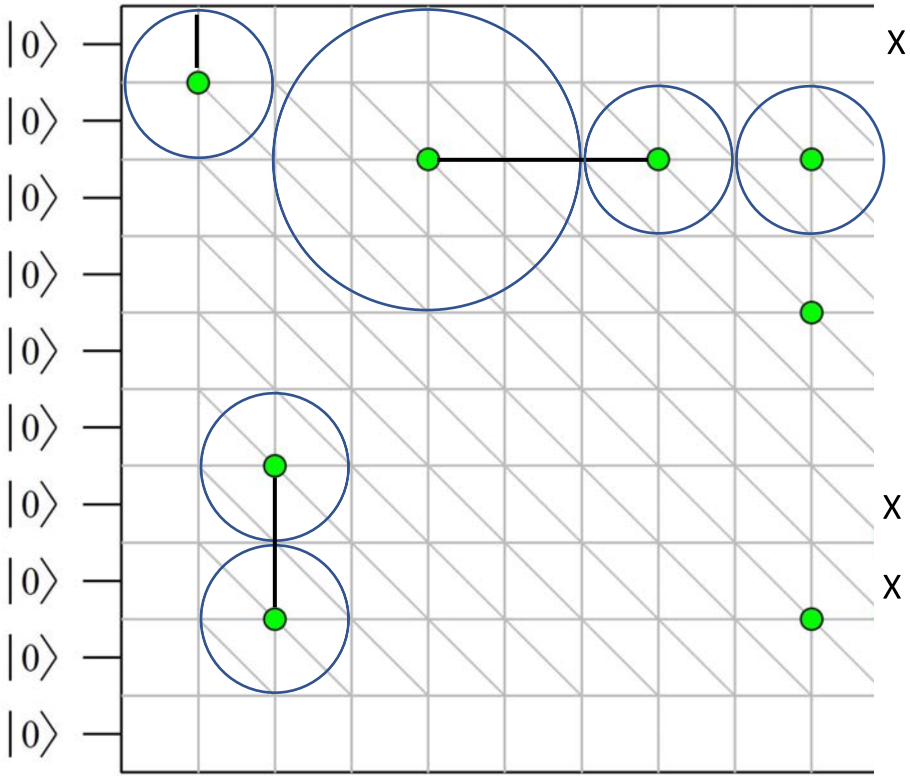
Classical processing



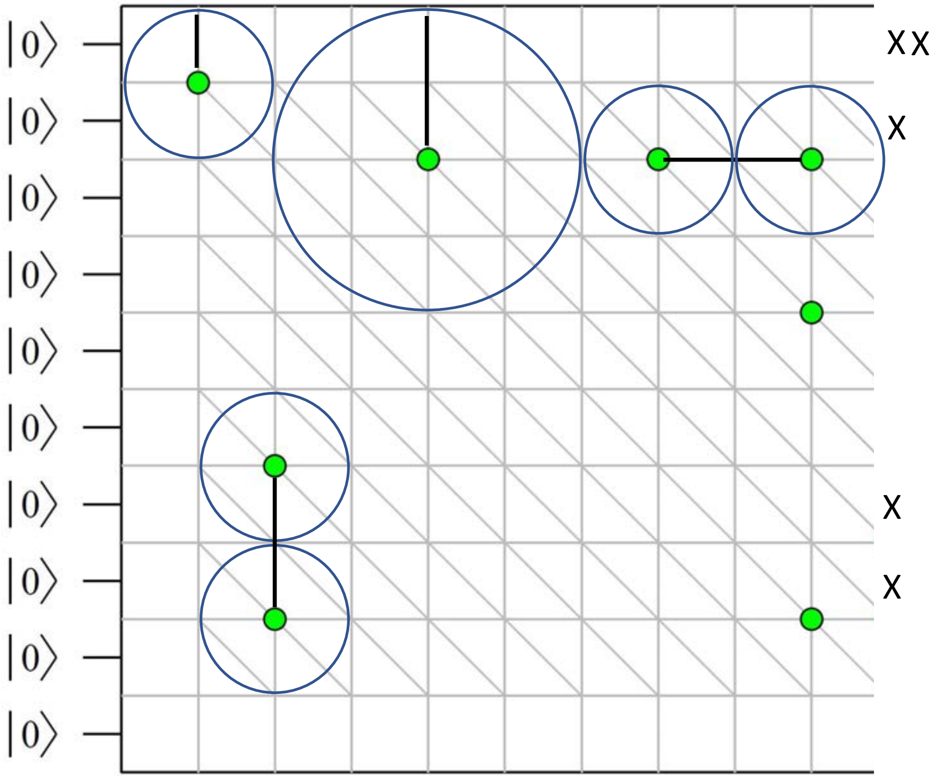
Classical processing



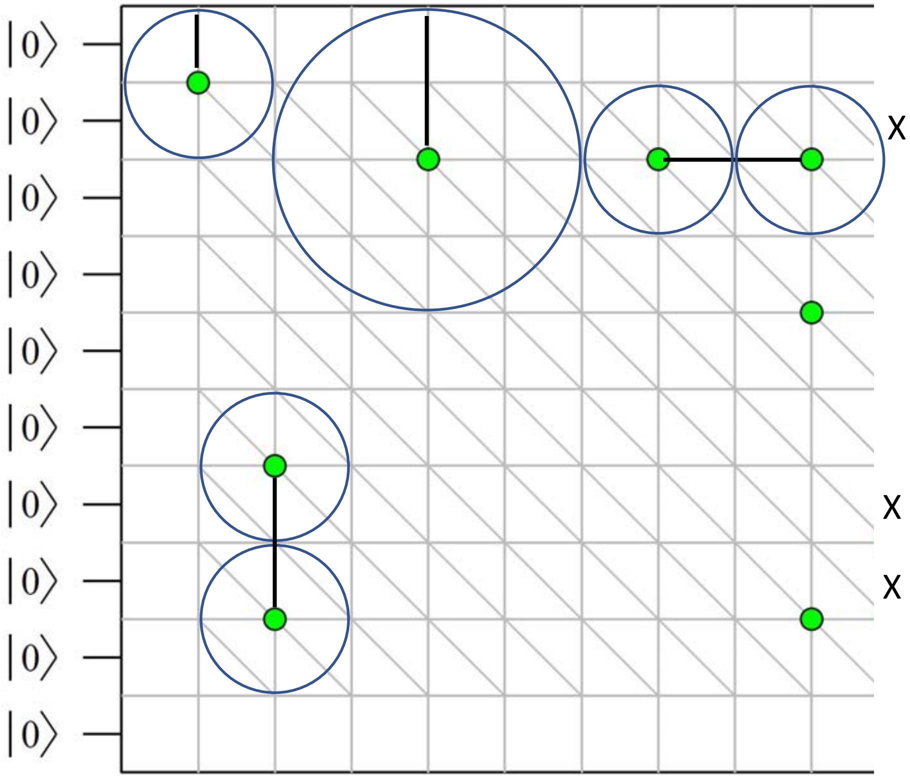
Classical processing



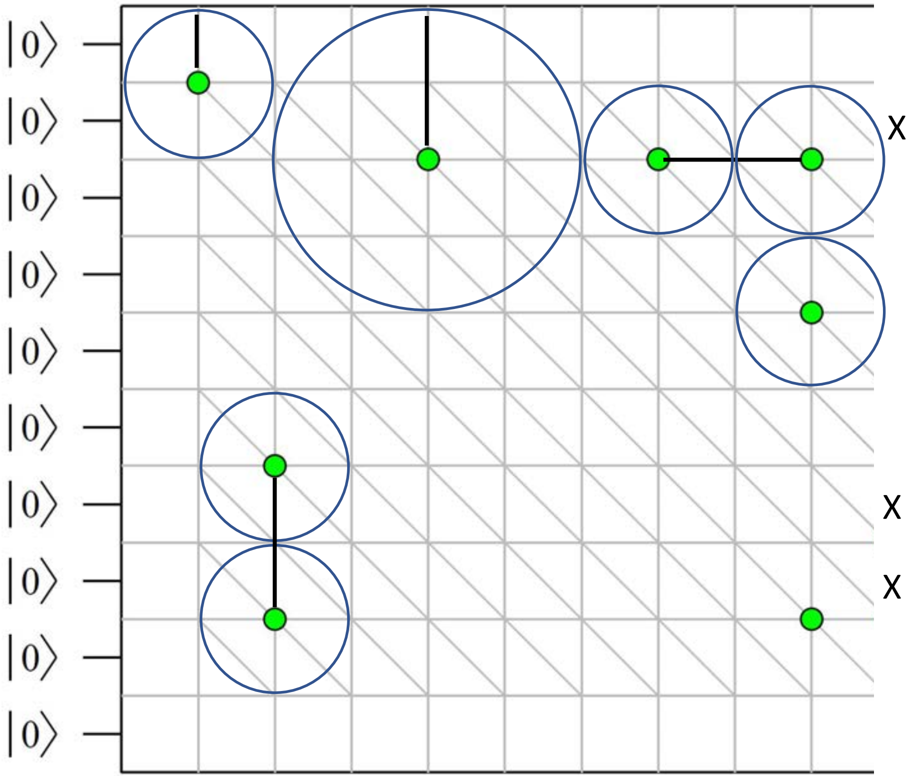
Classical processing



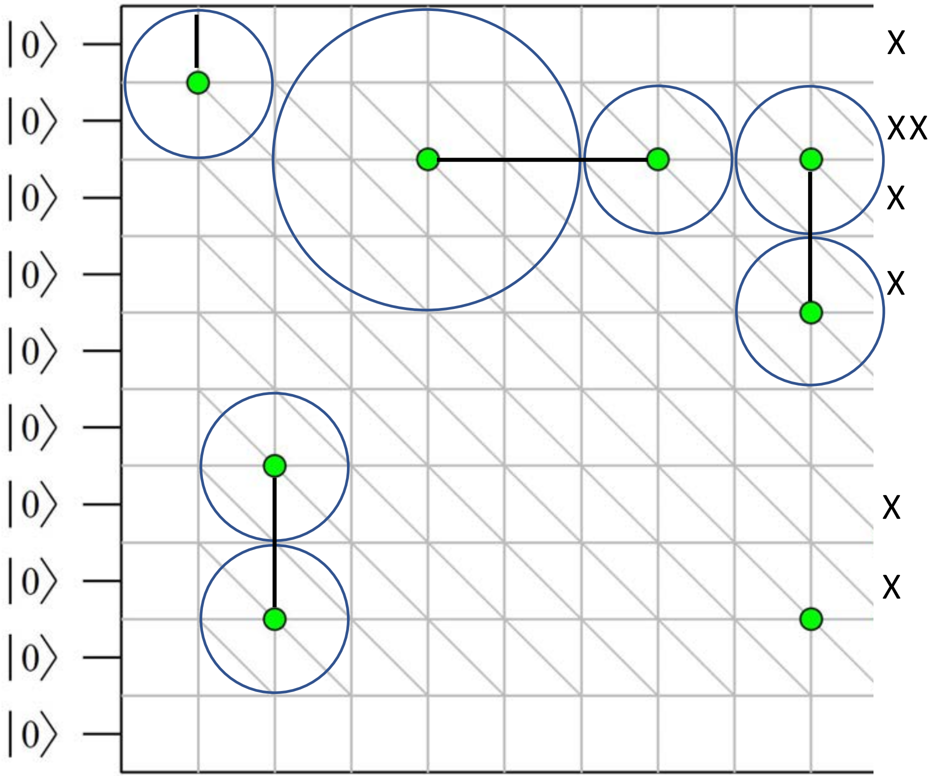
Classical processing



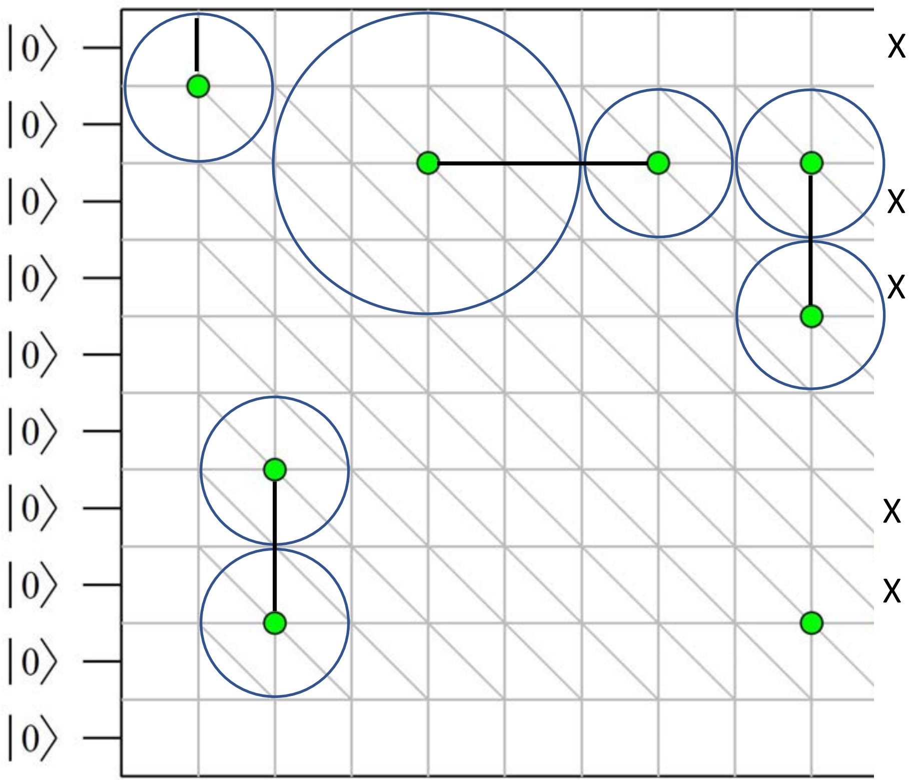
Classical processing



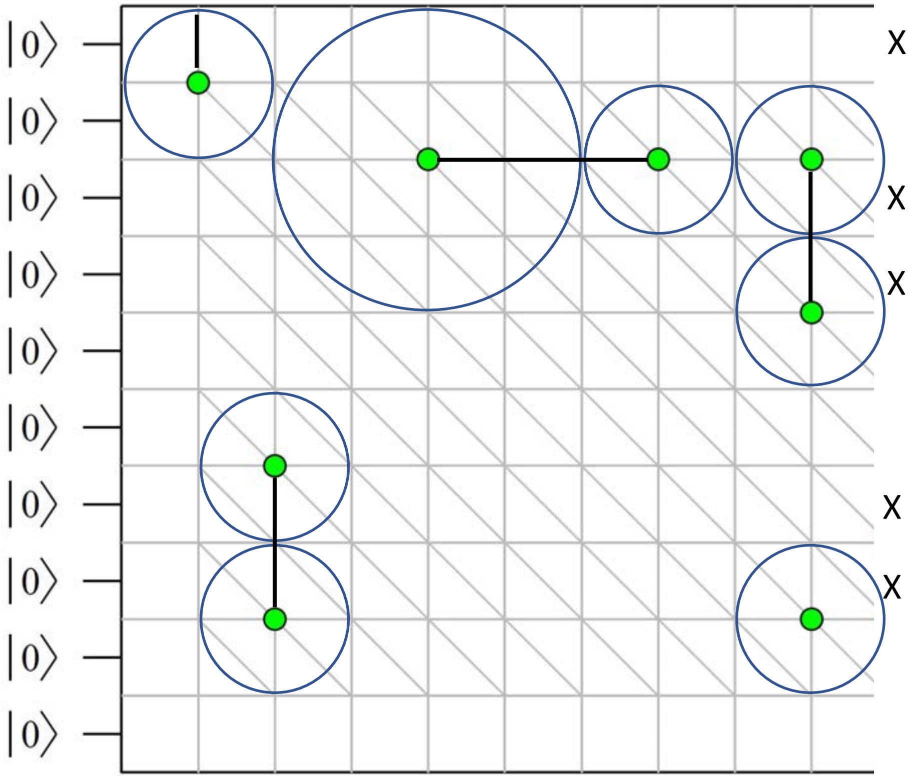
Classical processing



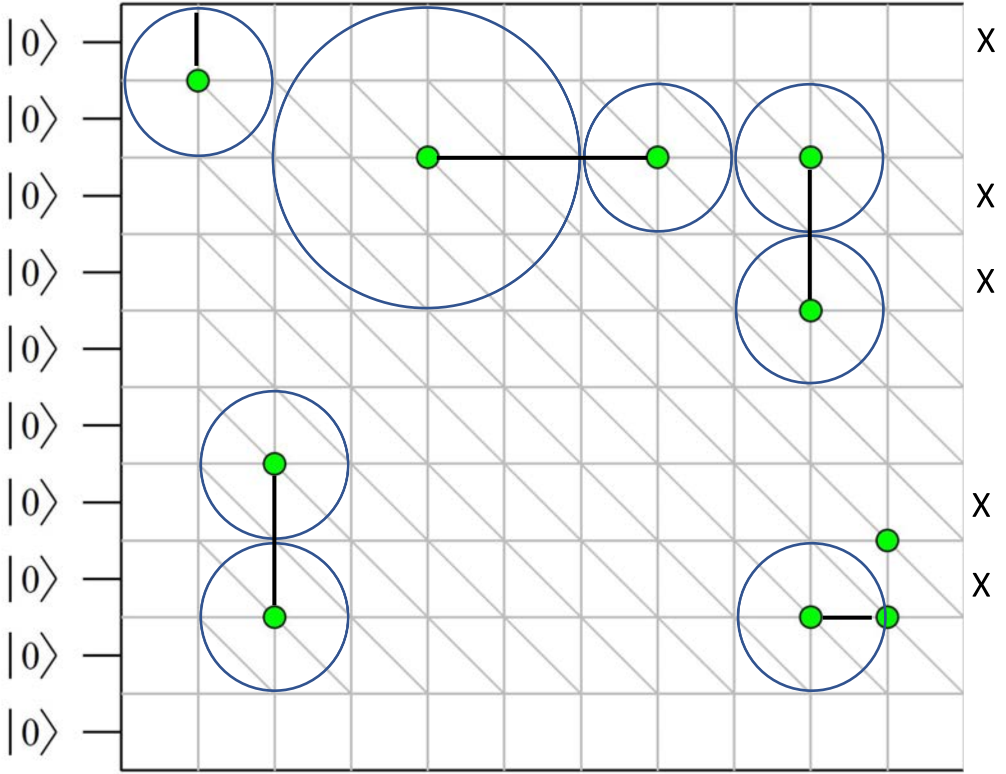
Classical processing



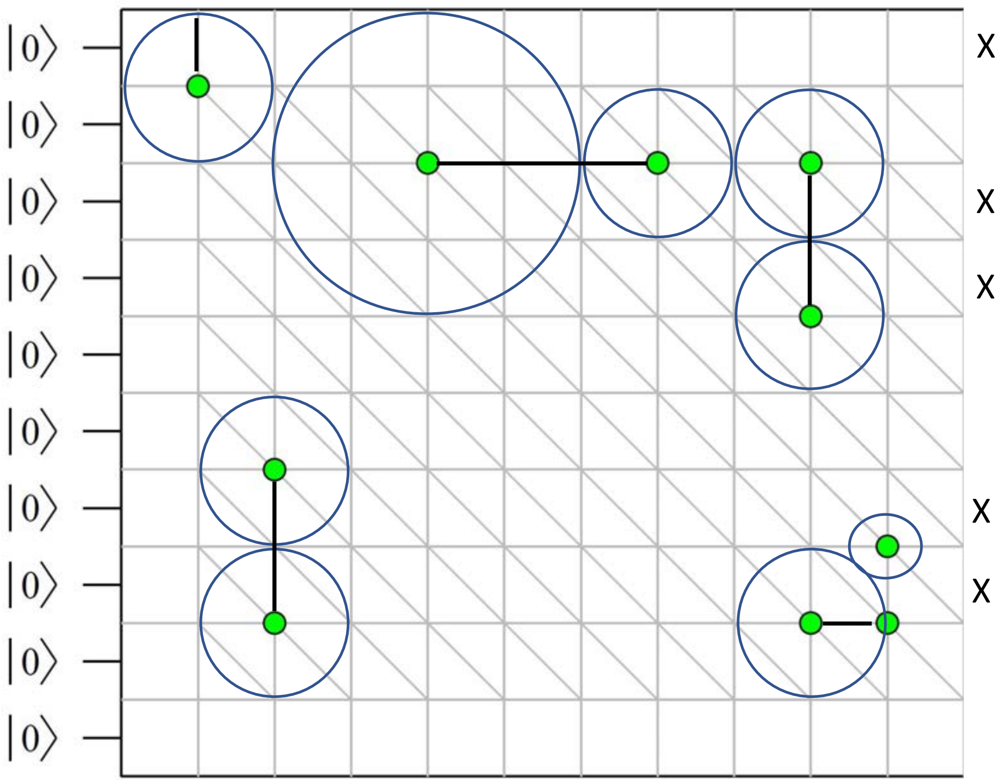
Classical processing



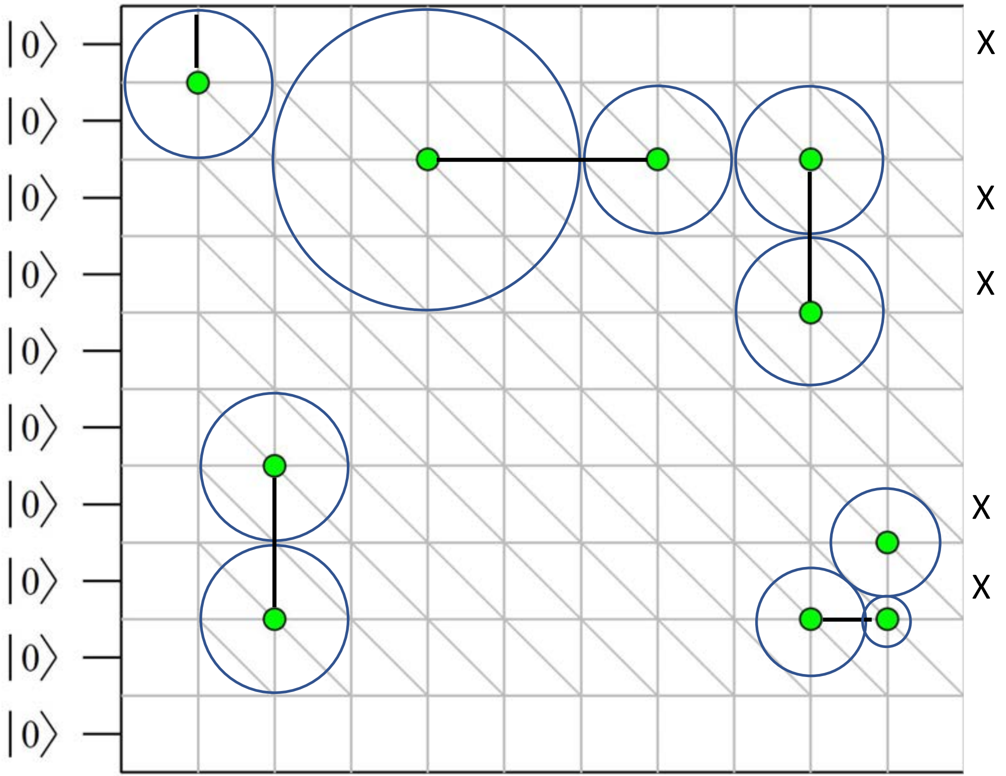
Classical processing



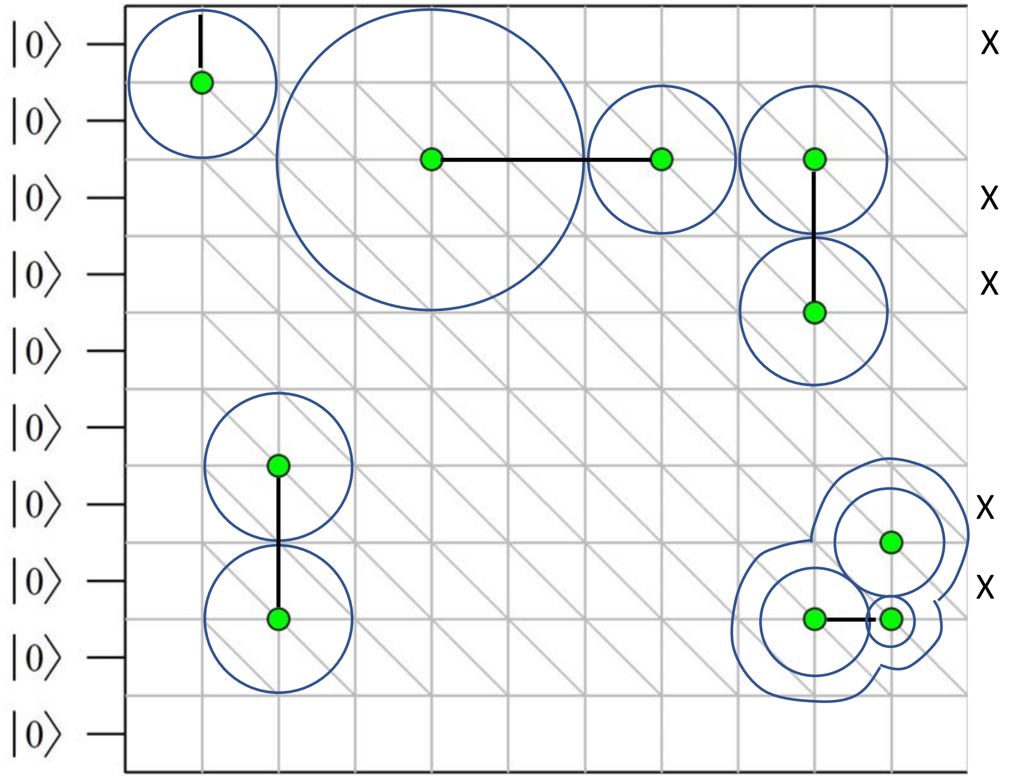
Classical processing



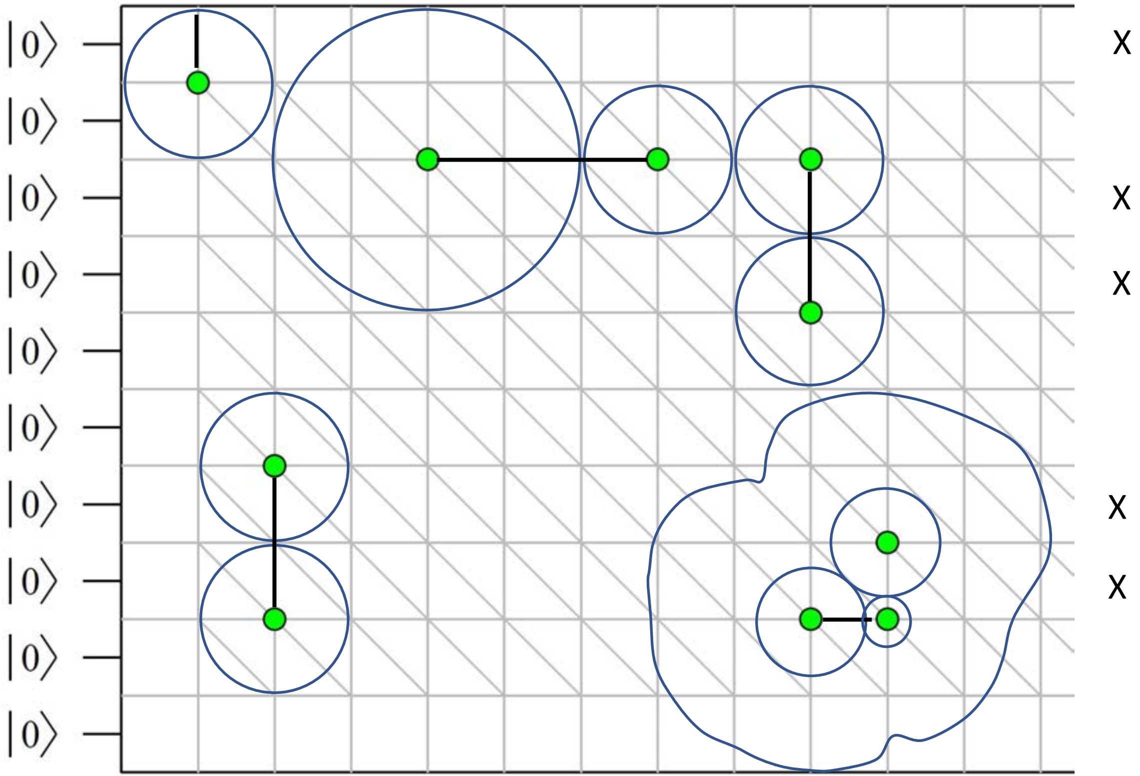
Classical processing



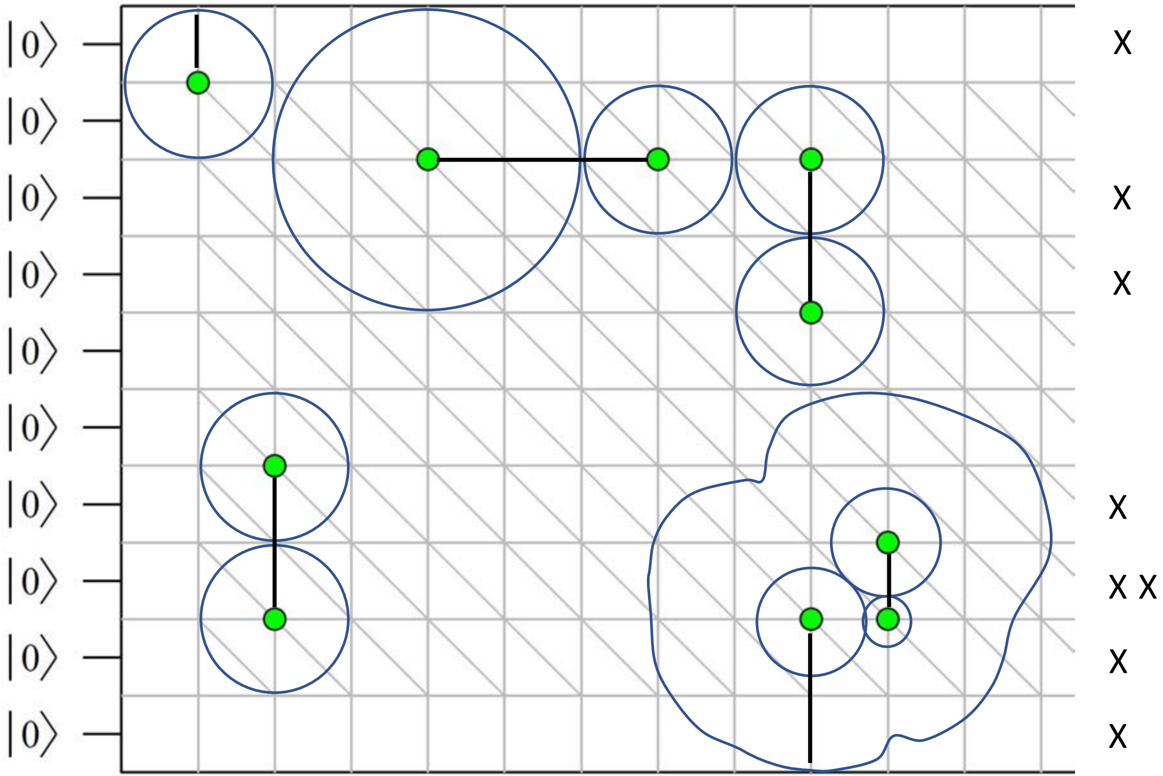
Classical processing



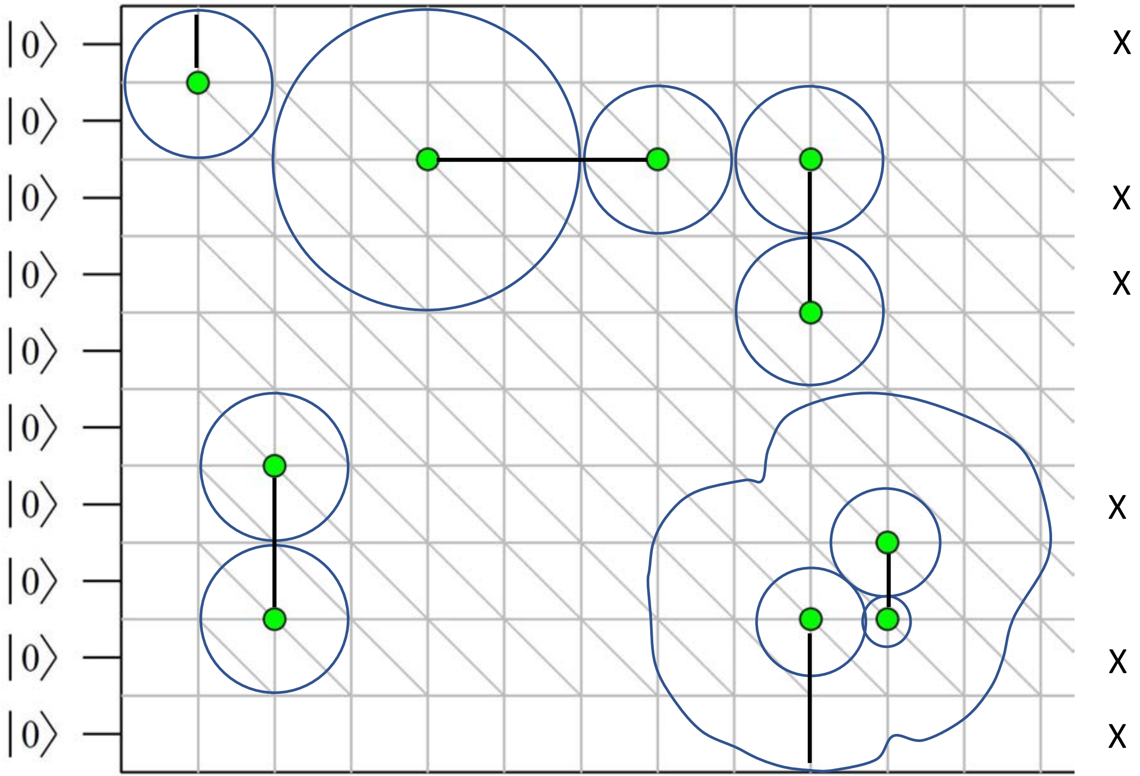
Classical processing



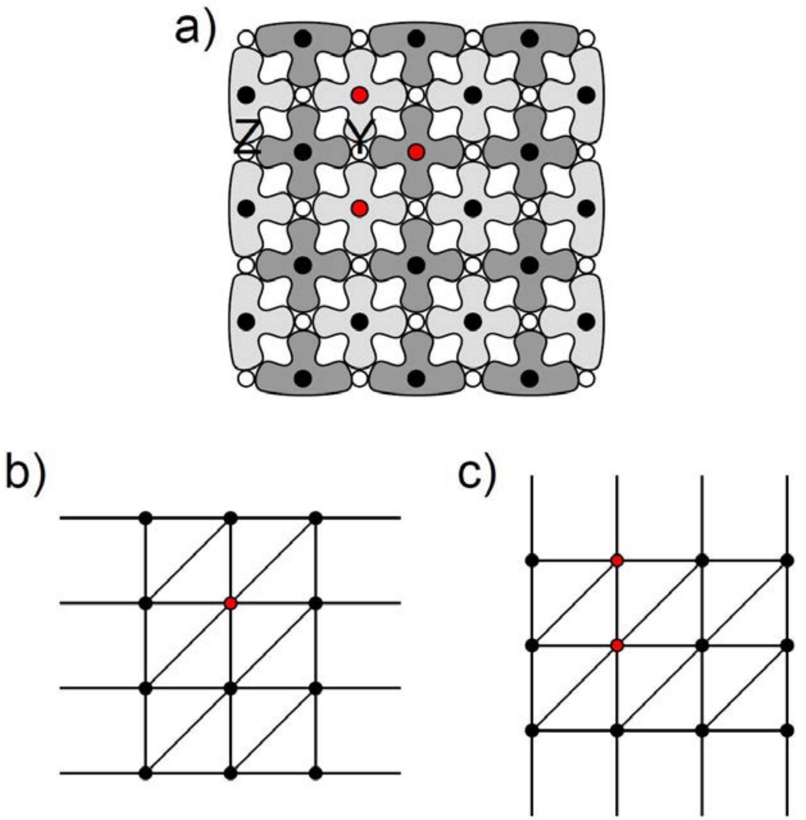
Classical processing



Classical processing

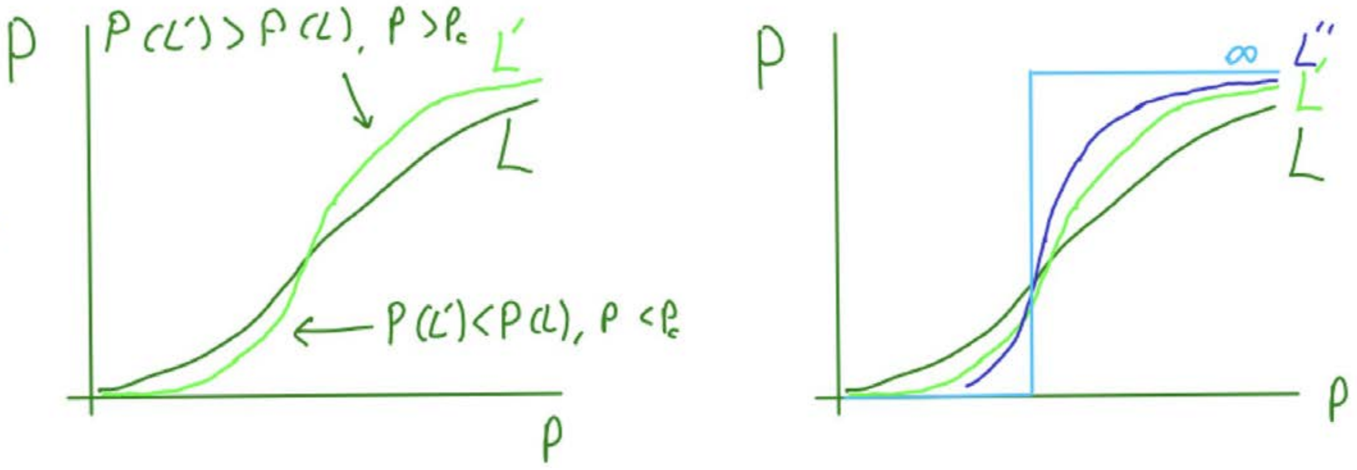


Correlated errors

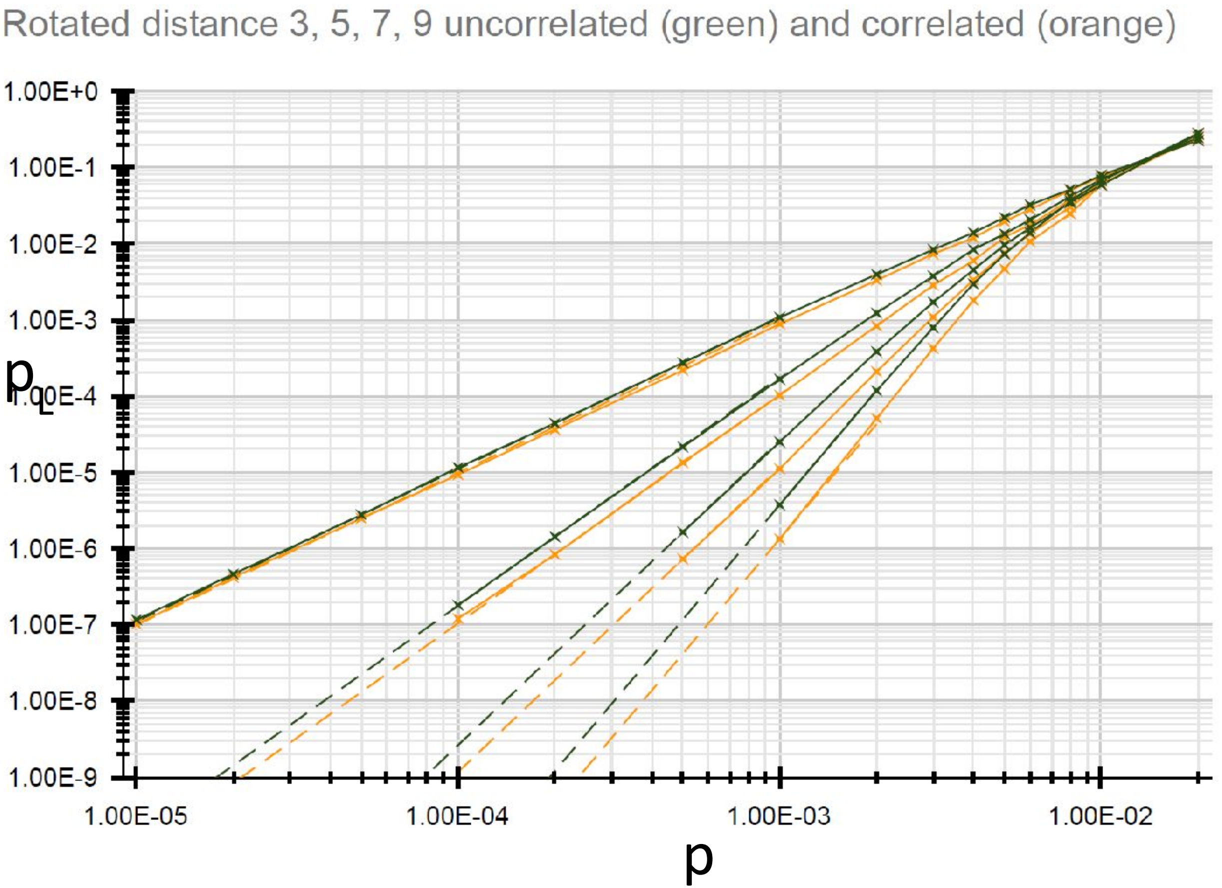


Threshold

- 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)



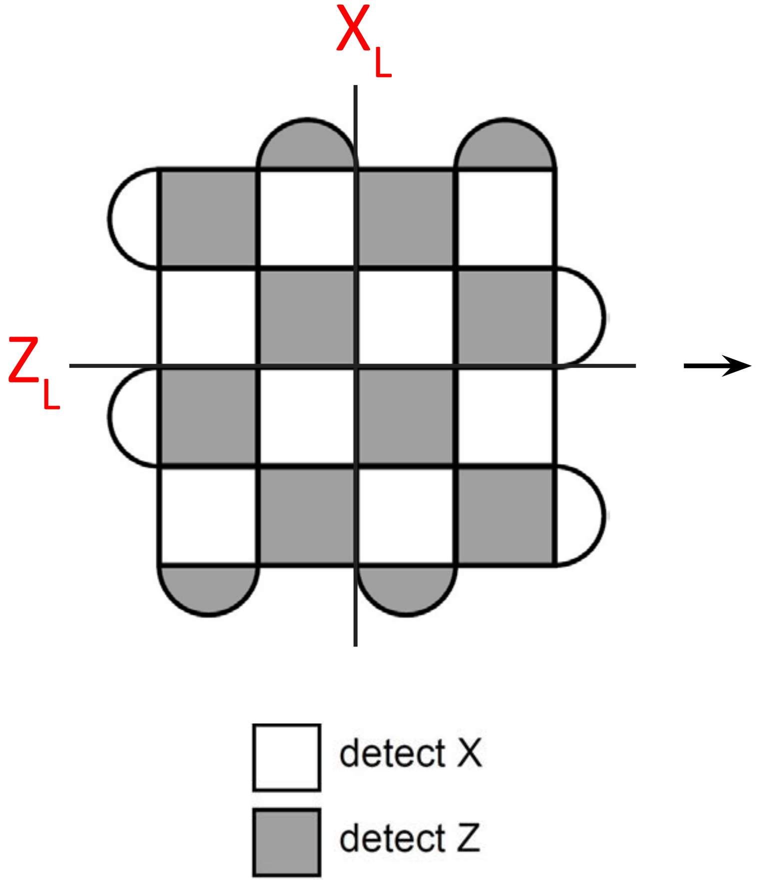
Simulated performance

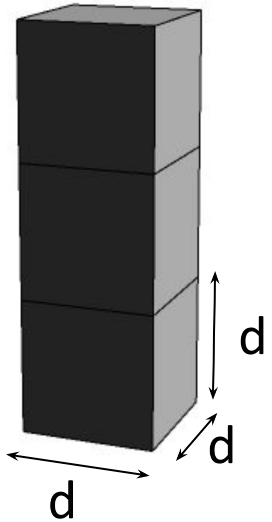


- $p_L = 0.1(100p)^{(d+1)/2}$
- O(1) parallel algorithm
- Low latency

Logical Gates and Experiments

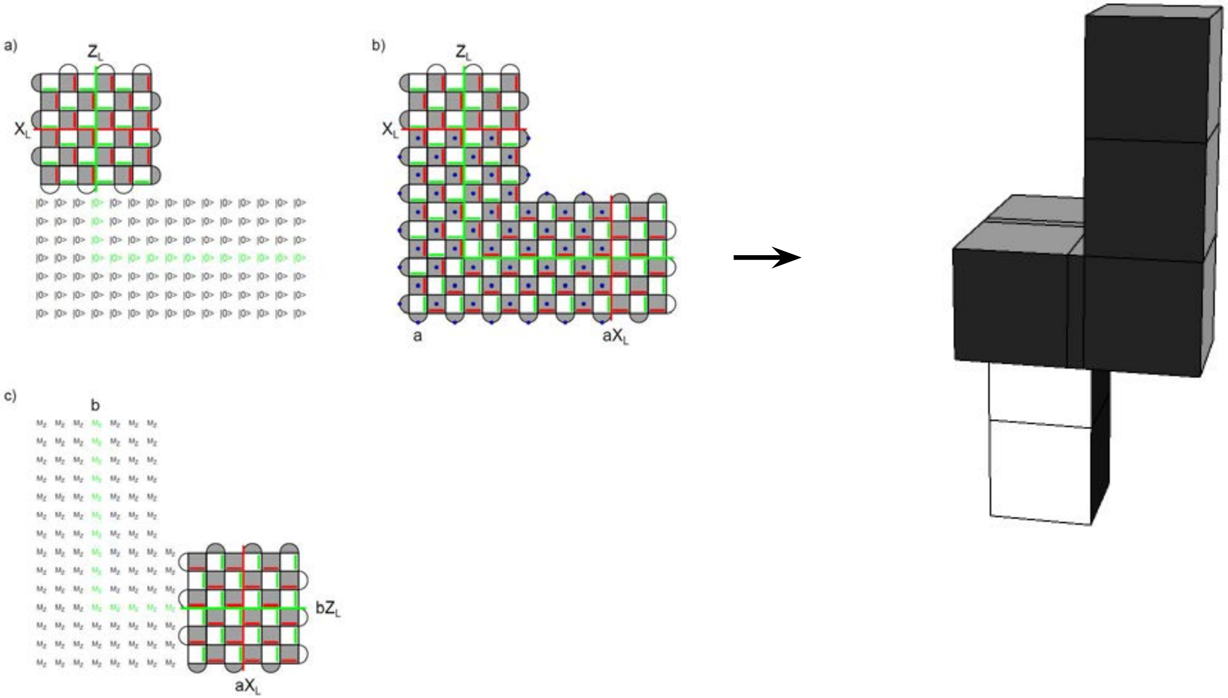
Logical identity



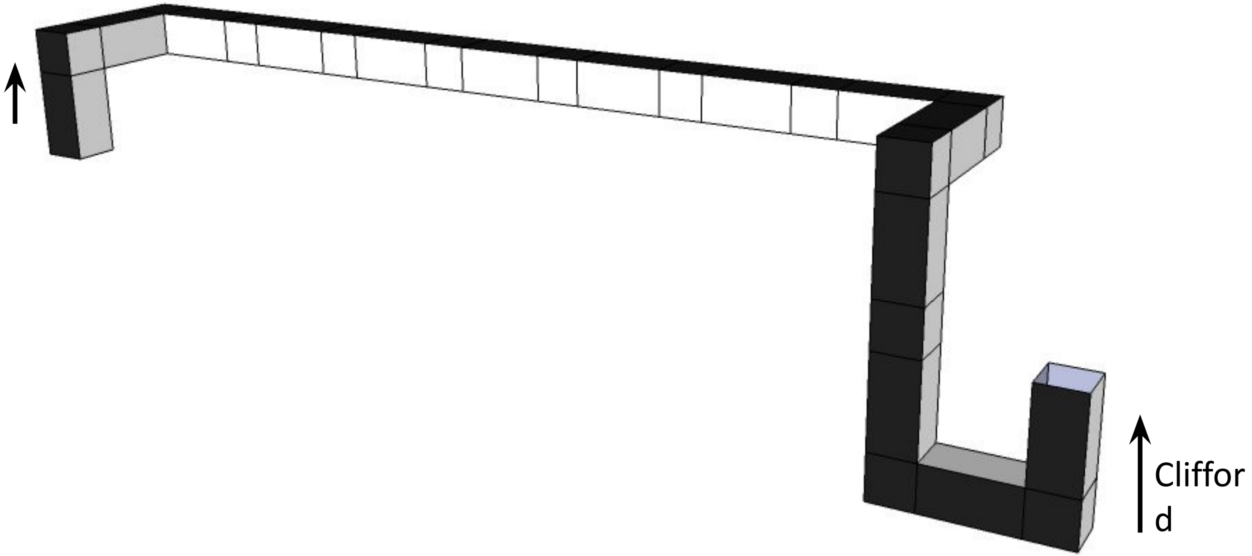


$$p_L = 0.1(100p)^{(d+1)/2}$$

Logical move

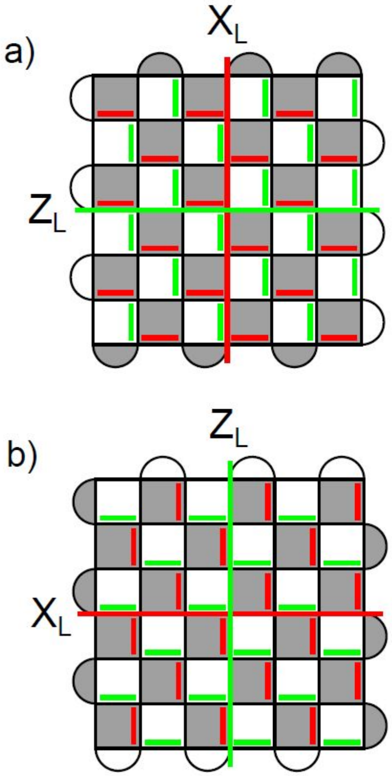


Logical move

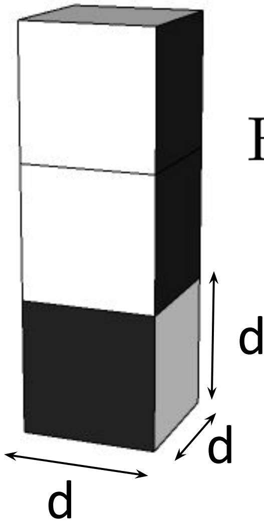


Can move anywhere, even back in time, up to Pauli operators.

Logical Hadamard



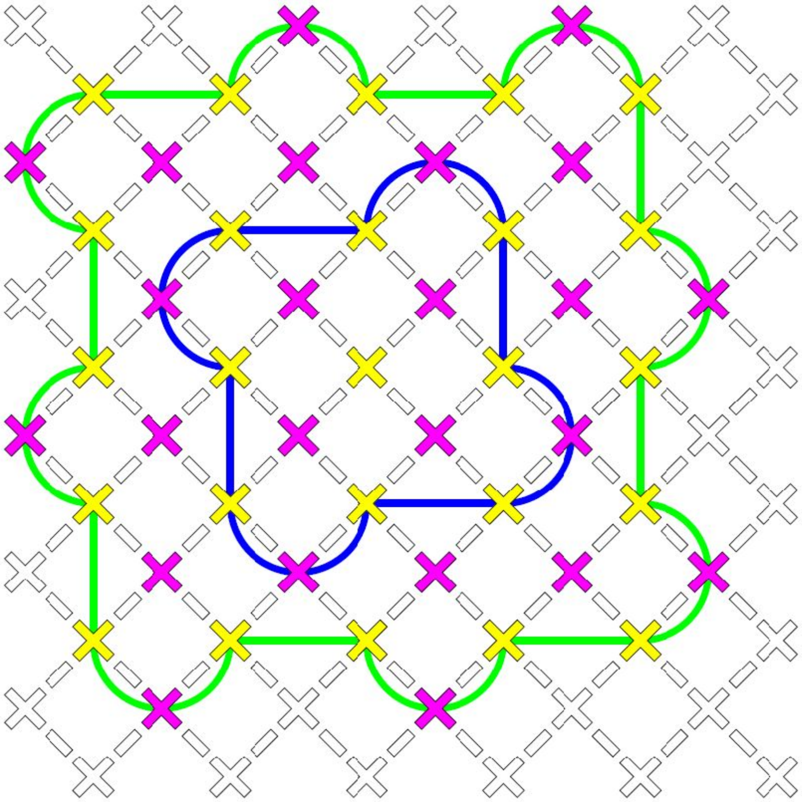
→



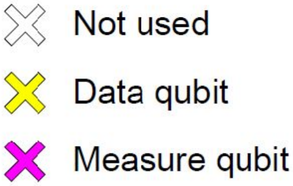
$$H = \frac{1}{\sqrt{2}}\begin{bmatrix}1 & 1\\ 1 & -1\end{bmatrix}$$

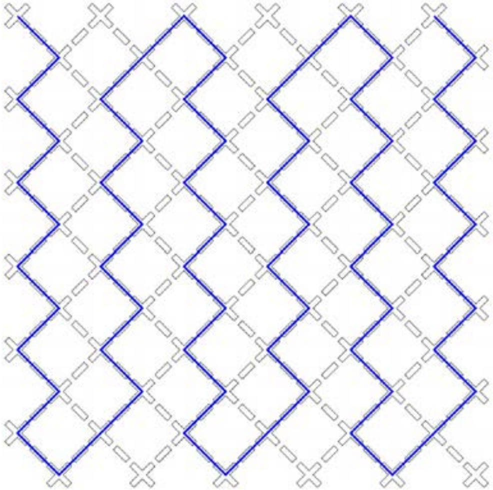
$$p_L = 0.1(100p)^{(d+1)/2}$$

Device and experiment

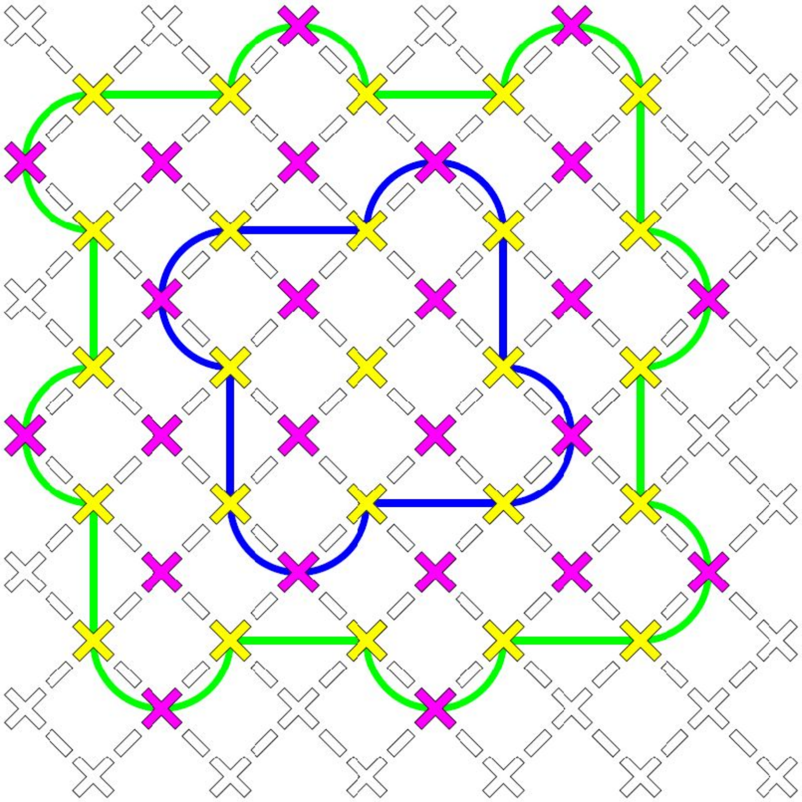


- Show d=5 better than d=3
- Continuous running
- Real-time decoding
- d=34 extreme exponential suppression (d=11 arXiv:2102.06132)





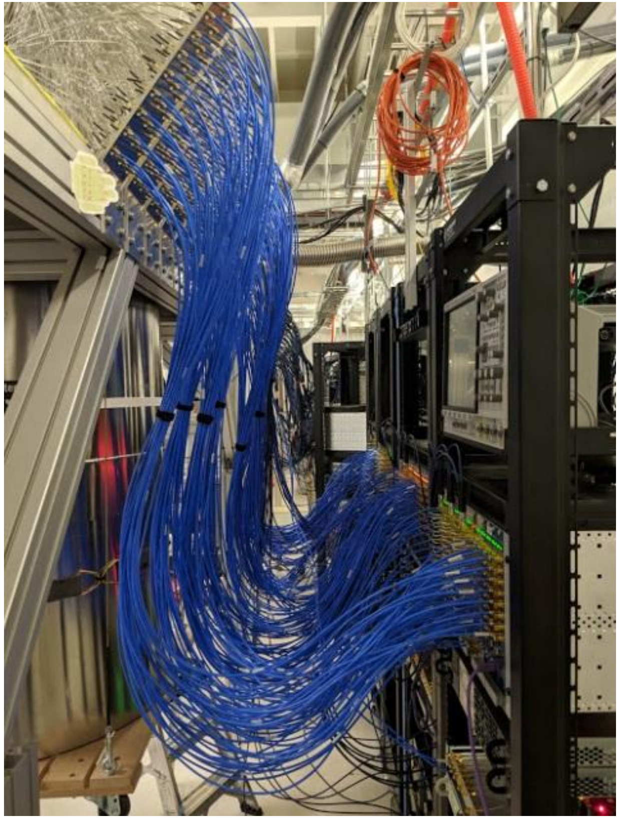
Challenges



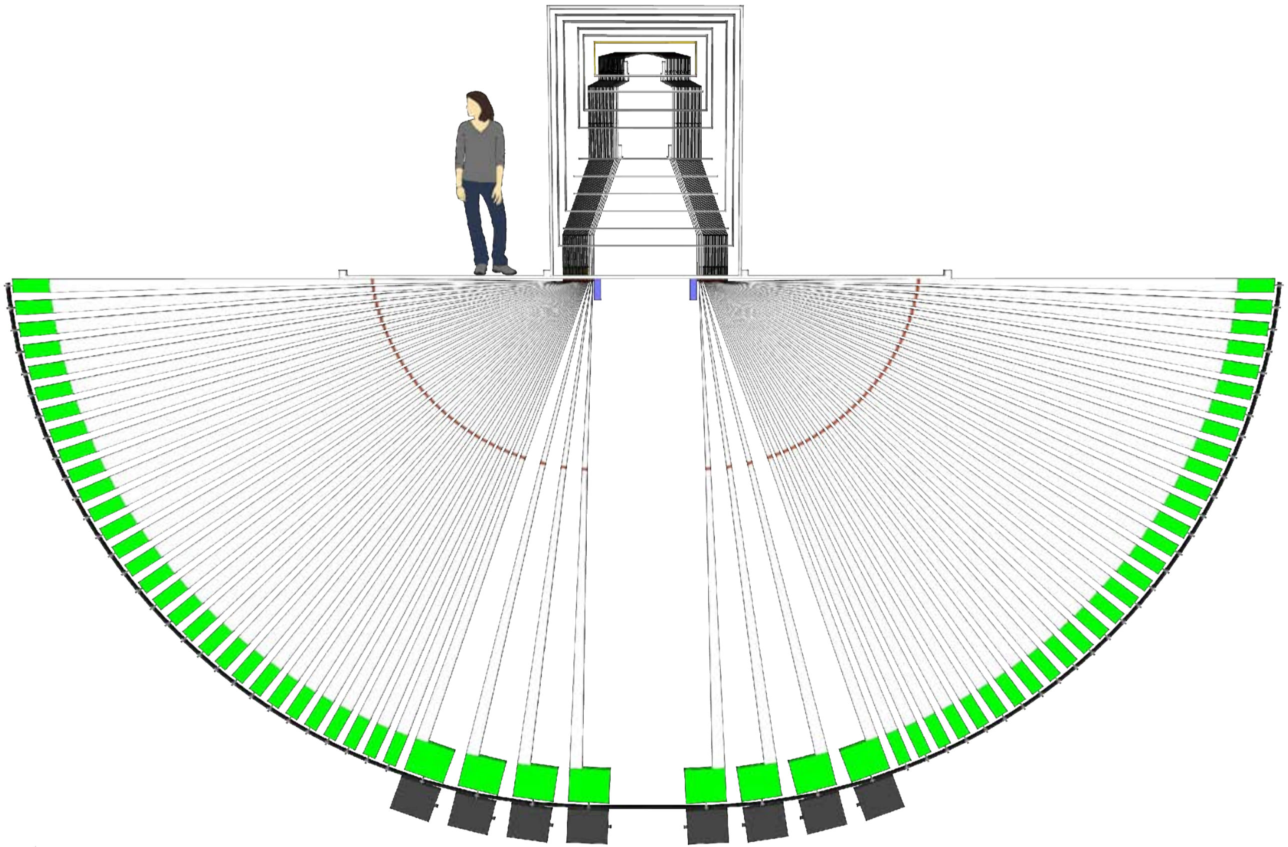
Not used
Data qubit
Measure qubit

- Yield < 100%
- Coherence ~20us
- Readout and reset ~1us (arXiv:2102.06131)
- Calibration 0.5% (arXiv:1907.02510)
- Cosmic rays (arXiv:2104.05219)
- Decoding (arXiv:1202.5602)
- Target 1.5 to 10x suppression

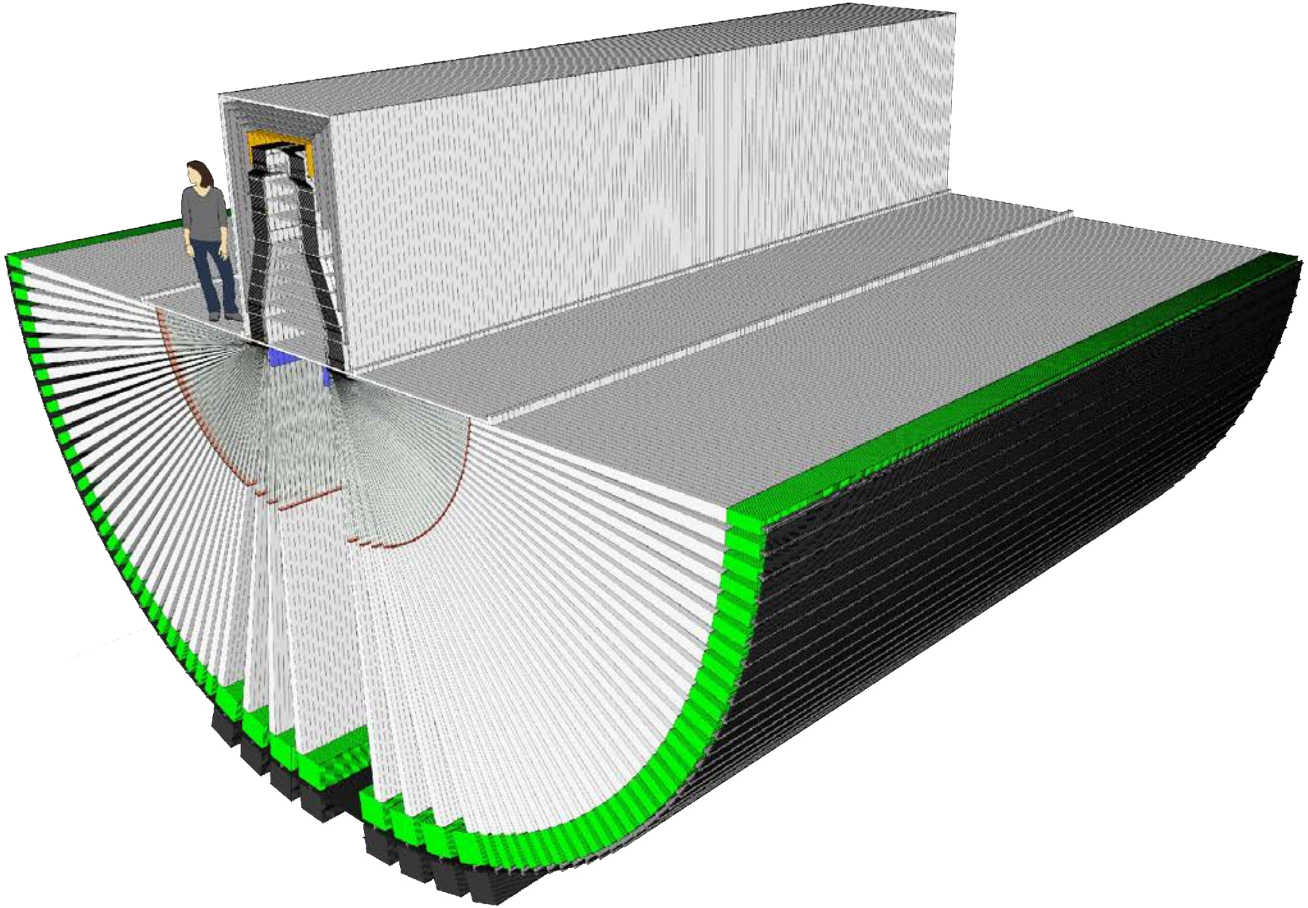
Scalable?



10k scalable qubits



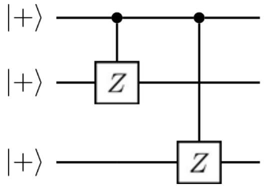
1M qubits



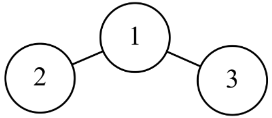
Compiling

Graph states

$$H|0\rangle = |+\rangle$$



=



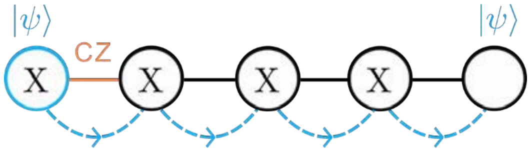
$$= \begin{matrix} X & Z & Z \\ Z & X & I \\ Z & I & X \end{matrix}$$

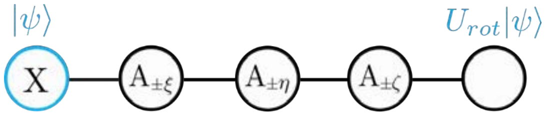
- Graph state with nodes i and edge neighbourhood n_i

$$s_i = X_i \prod_{j \in n_i} Z_i$$

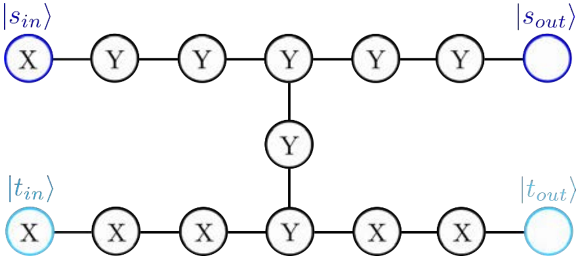
- Every stabilizer state can be converted to a graph state using only
local Clifford operations

Teleportation based QC





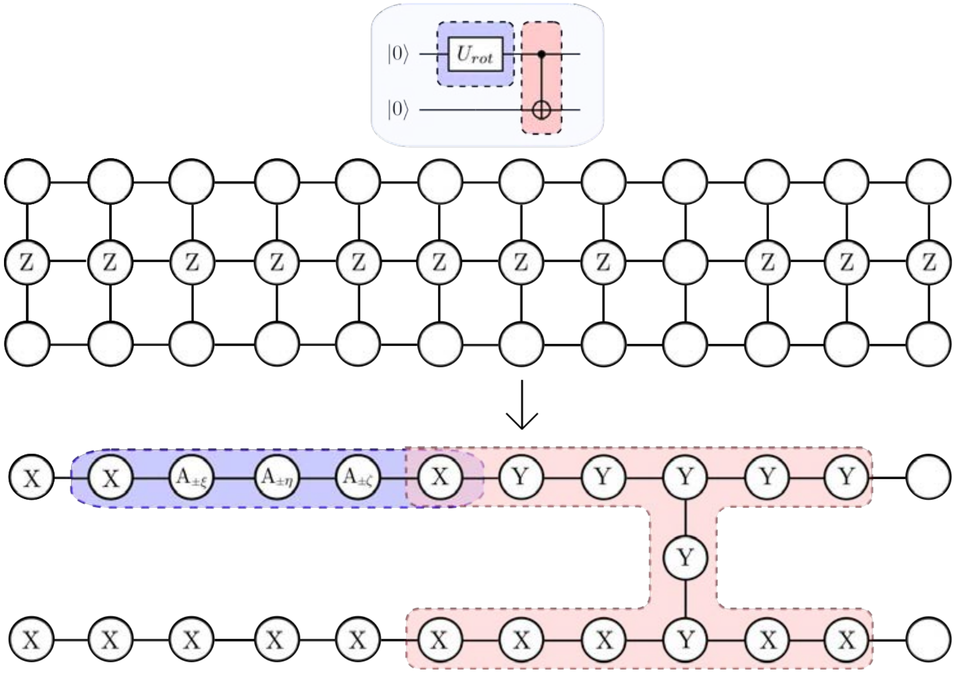
$$U_{rot} = e^{-i\frac{\zeta}{2}Z} e^{-i\frac{\eta}{2}X} e^{-i\frac{\xi}{2}Z}$$



$$|s_{out}\rangle|t_{out}\rangle = CNOT|s_{in}\rangle|t_{in}\rangle$$

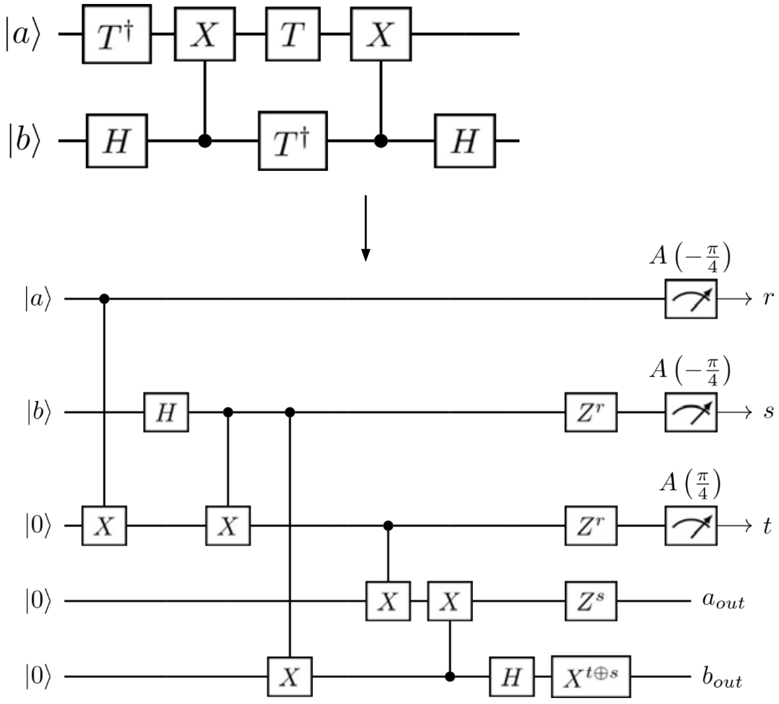
Traditional circuit compilation

- **Shortcomings of traditional approach**
- Performs Clifford operations on a stabilizer state which can be simulated efficiently on a classical device
- No room for optimisation other than at the circuit level.

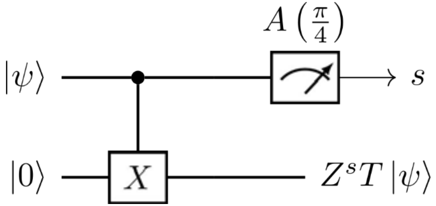


Example of Algorithm specific compilation

Input circuit



Teleportation widget

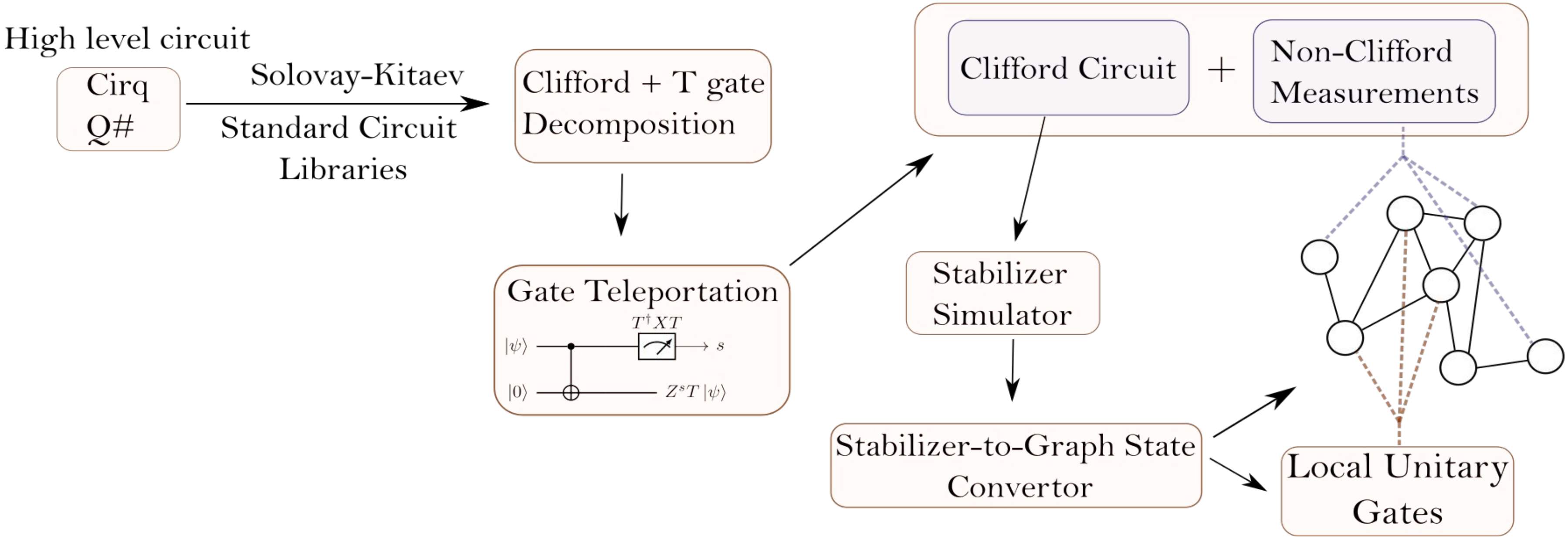


H

$$|ab\rangle = |00\rangle + |10\rangle$$

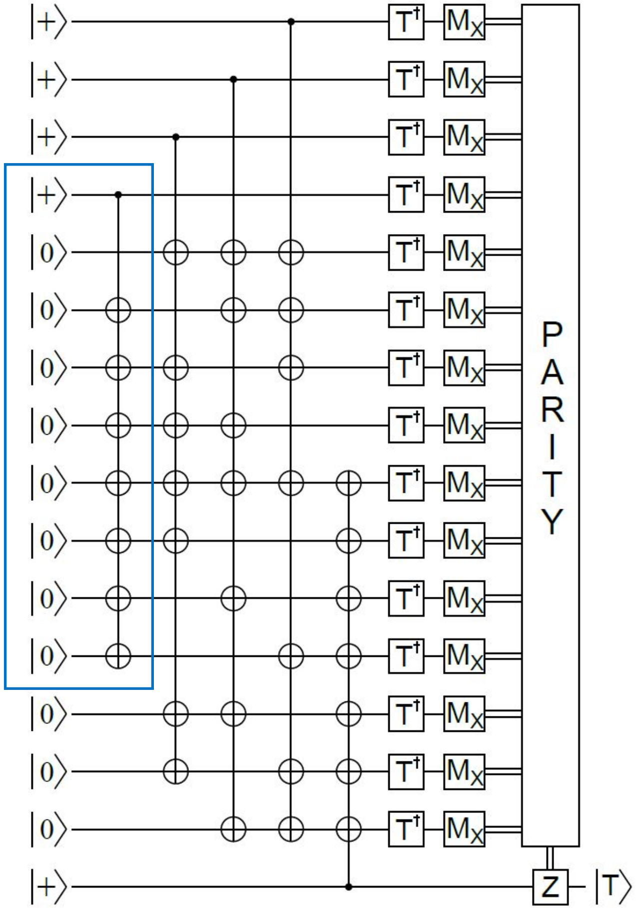
H

Algorithm specific graph compiler



Jabalizer – A Julia based graph compiler https://github.com/QSI-BAQS/Jabalizer.jl

State distillation



$$|\psi\rangle = \frac{1}{\sqrt{2}}\left(|00\ldots0\rangle + |11\ldots1\rangle\right)$$

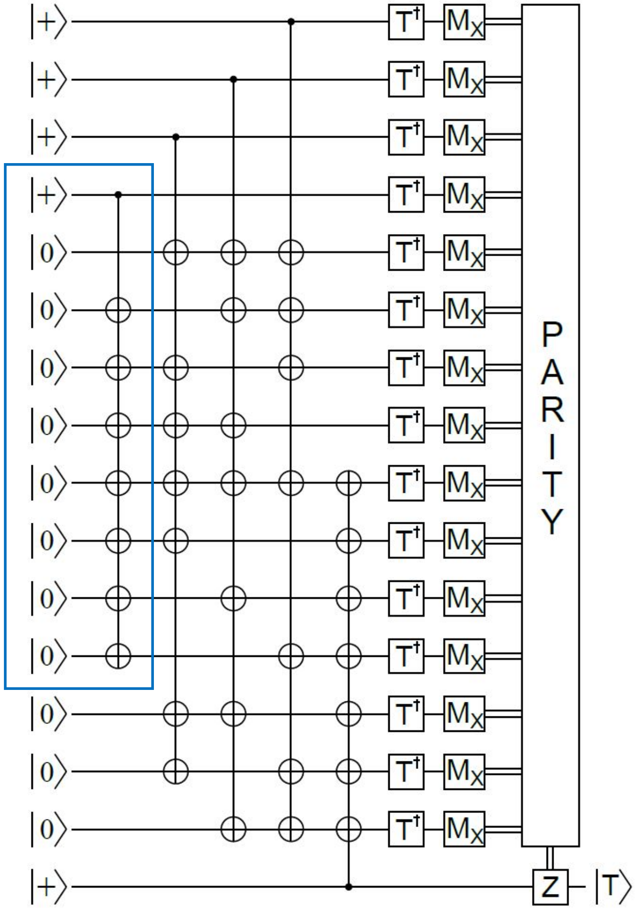
$$XX\ldots X|\psi\rangle = |\psi\rangle$$

$$|\phi\rangle = \frac{1}{\sqrt{2}}\left(|00\ldots0\rangle - |11\ldots1\rangle\right)$$

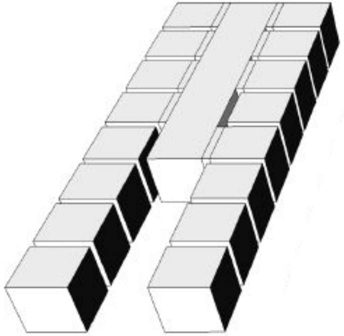
$$XX\ldots X|\phi\rangle = -|\phi\rangle$$

$$|00\ldots0\rangle = \frac{1}{\sqrt{2}}\left(|\psi\rangle + |\phi\rangle\right)$$

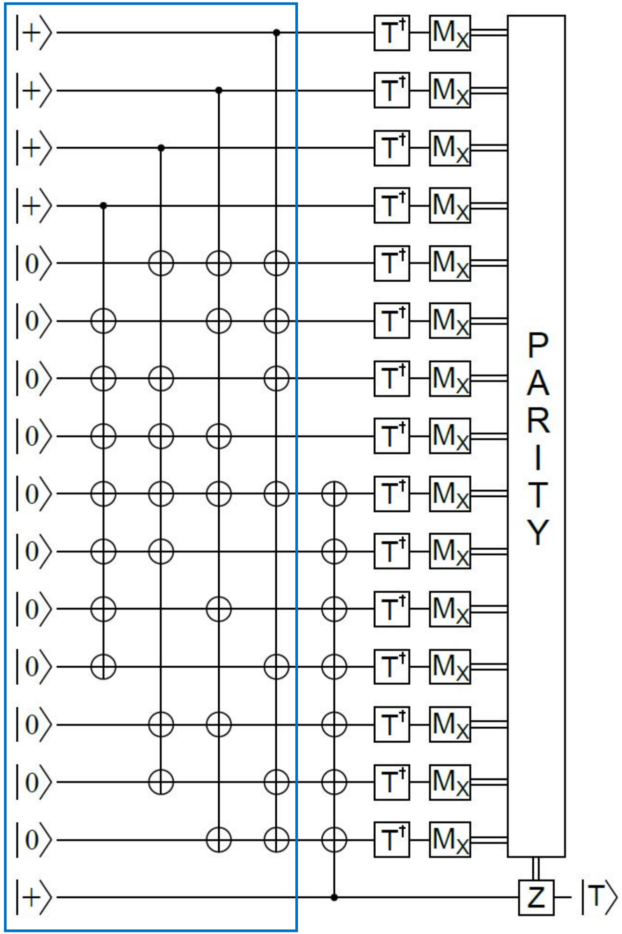
State distillation



Measure multi-body X operator!



State distillation



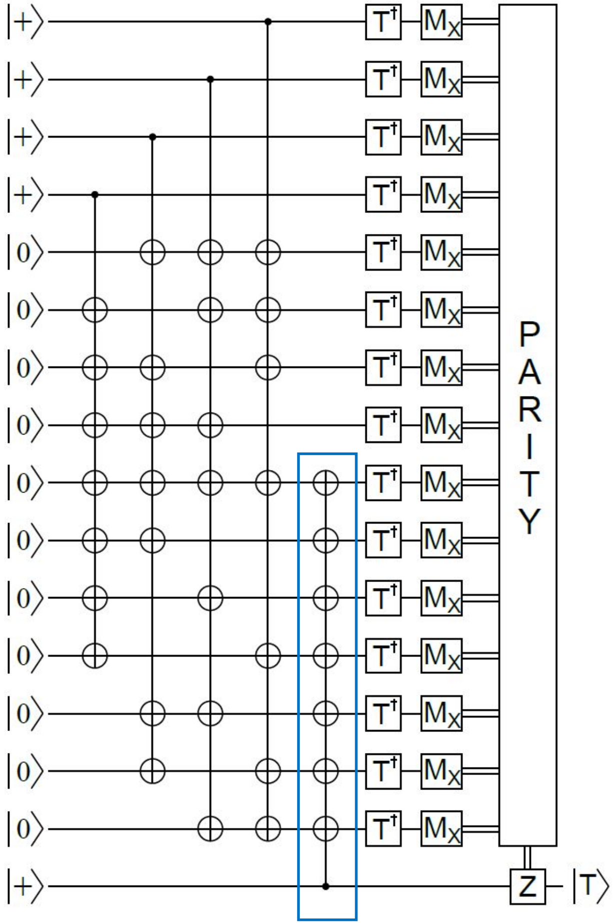
$$\frac{1}{\sqrt{2}}|0_L\rangle(|0\rangle + |1\rangle)$$

$$\frac{1}{\sqrt{2}}\left(|0_L 0\rangle + |1_L 1\rangle\right)$$

$$\frac{1}{\sqrt{2}}\left(|0_L 0\rangle + e^{i\pi/4}|1_L 1\rangle\right)$$

$$\frac{1}{\sqrt{2}}\left(|0\rangle + (-1)^{M_X} e^{i\pi/4}|1\rangle\right)$$

State distillation



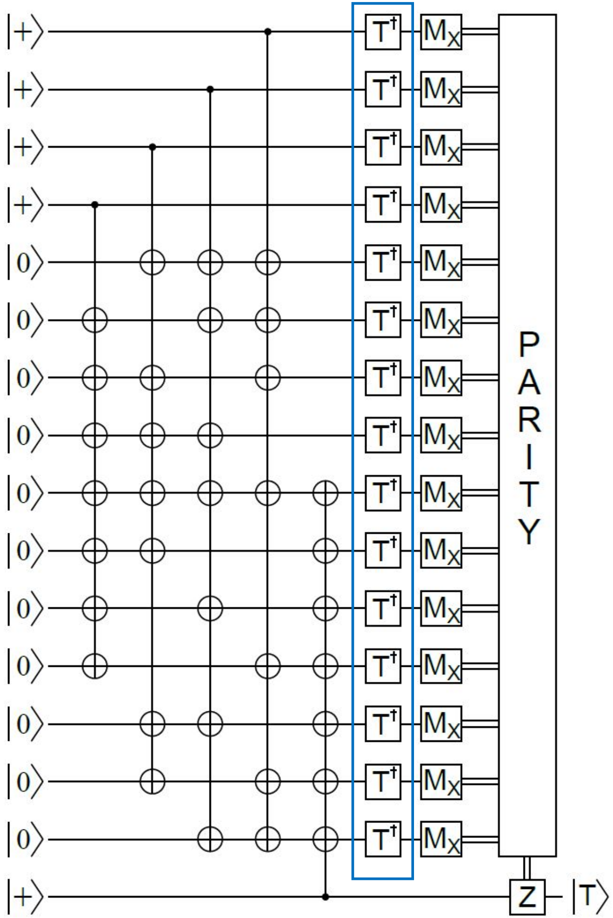
$$\frac{1}{\sqrt{2}}|0_L\rangle(|0\rangle + |1\rangle)$$

$$\frac{1}{\sqrt{2}}\left(|0_L 0\rangle + |1_L 1\rangle\right)$$

$$\frac{1}{\sqrt{2}}\left(|0_L 0\rangle + e^{i\pi/4}|1_L 1\rangle\right)$$

$$\frac{1}{\sqrt{2}}\left(|0\rangle + (-1)^{M_X} e^{i\pi/4}|1\rangle\right)$$

State distillation



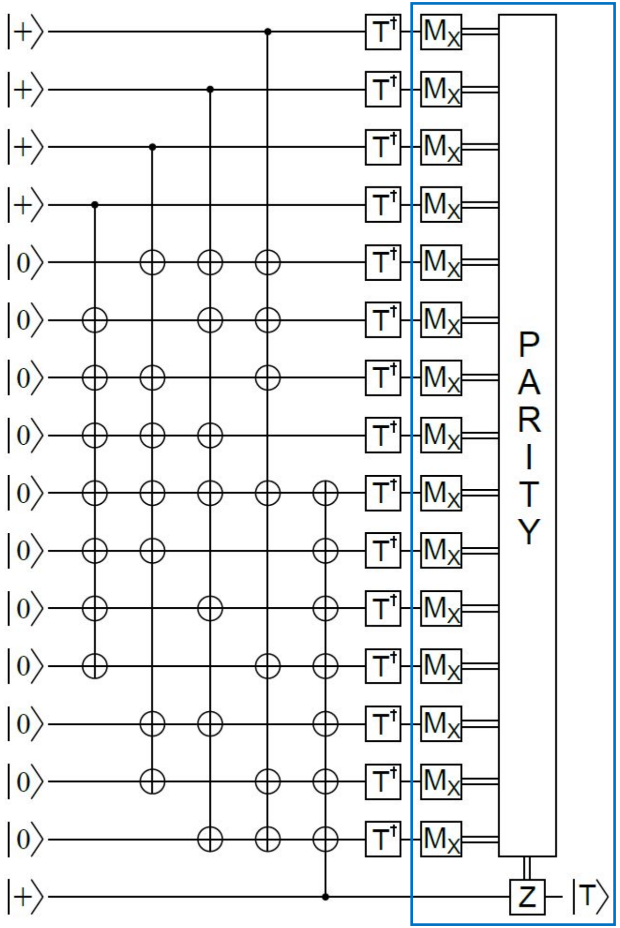
$$\frac{1}{\sqrt{2}}|0_L\rangle(|0\rangle + |1\rangle)$$

$$\frac{1}{\sqrt{2}}\left(|0_L 0\rangle + |1_L 1\rangle\right)$$

$$\frac{1}{\sqrt{2}}\left(|0_L 0\rangle + e^{i\pi/4}|1_L 1\rangle\right)$$

$$\frac{1}{\sqrt{2}}\left(|0\rangle + (-1)^{M_X} e^{i\pi/4}|1\rangle\right)$$

State distillation



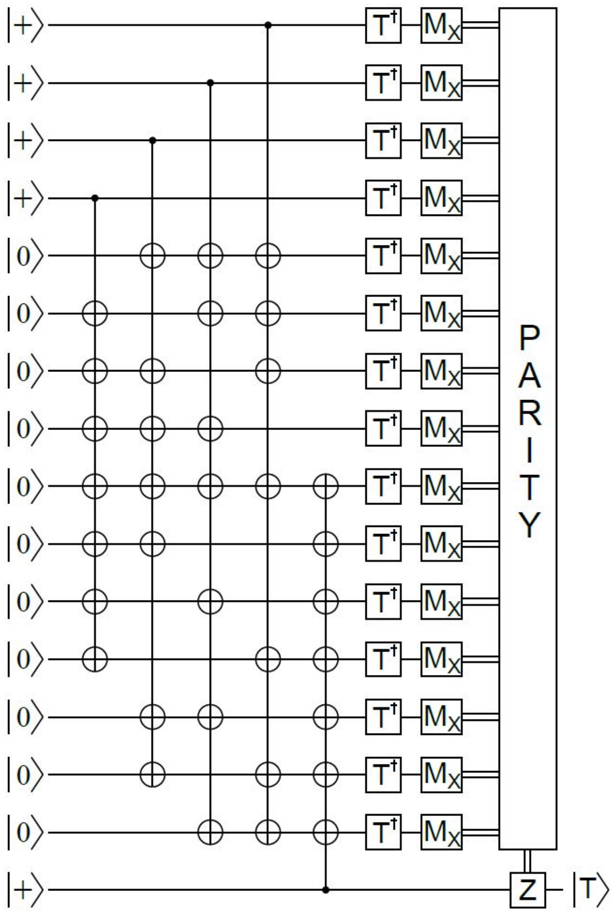
$$\frac{1}{\sqrt{2}}|0_L\rangle(|0\rangle + |1\rangle)$$

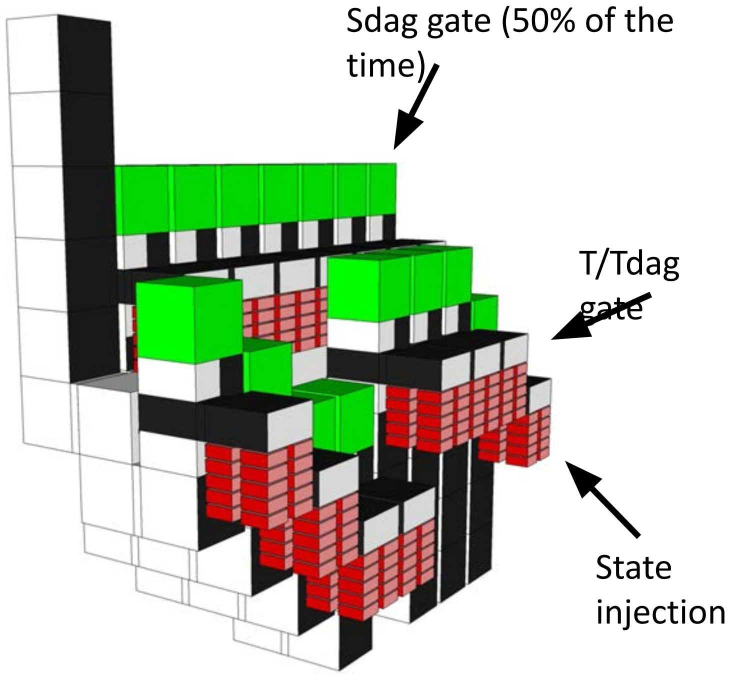
$$\frac{1}{\sqrt{2}}\left(|0_L 0\rangle + |1_L 1\rangle\right)$$

$$\frac{1}{\sqrt{2}}\left(|0_L 0\rangle + e^{i\pi/4}|1_L 1\rangle\right)$$

$$\frac{1}{\sqrt{2}}\left(|0\rangle + (-1)^{M_X} e^{i\pi/4}|1\rangle\right)$$

State distillation





Good news so far:

- 2D nearest neighbor
- Modest fidelity requirements
- Compact universal computation
- Arbitrary range interactions
- Multi-body measurements
- Flexible code strength