Quantum Error Correction

by Stefano, an LMU student

Introduction

Experimental implementation of Quantum Error Correction using two different approach:

- Surface code memories operating below threshold using a superconducting processor

- Programmable quantum processor based on reconfigurable atom arrays

Why we need QEC?

Main problems that need to be overcome:

- Wave function collapse
- No-cloning theorem
- Two types of error instead of one
- Continuous errors and decoherence

Is it possible to overcome all those limitations? Yes, but…

How can we solve those problems ?

Possible solutions:

- Digitisation of errors
- Exploit the error, not the states (to keep quantum information)
- Redundancy (stored the information in more Qubits (enlarged Hilbert space))

Hystorical recap & evolution of QEC

- Peter Shor (repetition code to both detect bit- and phase-flip errors) (1995)
- Gottesman thesis on Stabilizer formalism (1997)
- Kitaev’s code (1997)

Wikipedia

The stabilizer formalism

Errors can be detected by measuring a certain set of operators

Stabilizers

Advantages:

1. Efficient description of many-qubit states
2. Straightforward error detection and decoding
3. Encoded logical operators

Good set of stabilizer S

Good set of code-words
a.k.a. codespace

- Dan Browne, Lecture notes
- Wikipedia

The stabilizer formalism

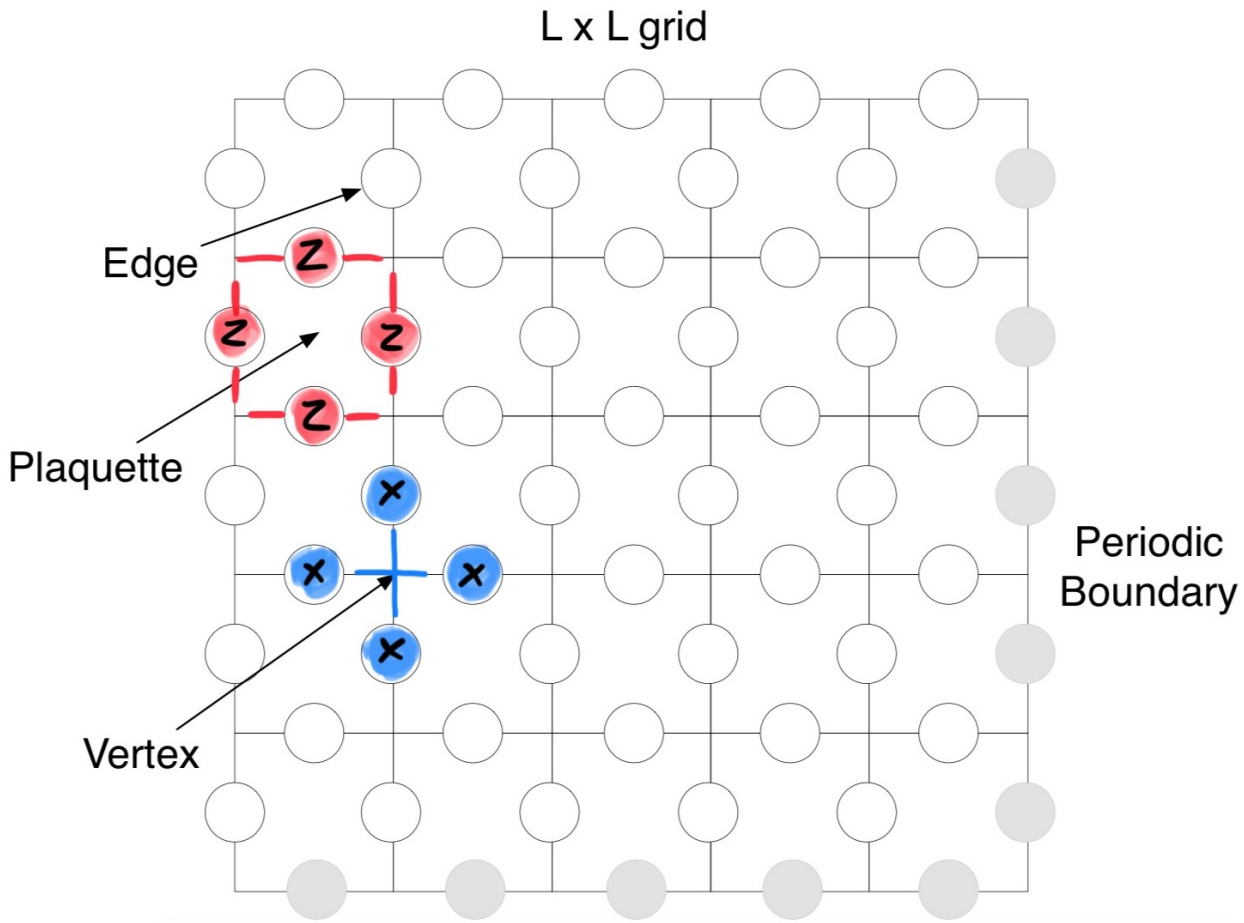
Quick recap of all their properties:

- [n,k,d] stabilizer code defined by m=n-k independent stabilizer generators S_j
- Encoded logical Pauli operators commute with each stabilizer
- The distance d of the code is given by the minimal weight of all encoded logical Pauli operators
- …

- Dan Browne, Lecture notes
- Wikipedia

The Toric Code

- We associate every qubit with every edge of the lattice
- $L \times L$ grid with periodic boundary conditions
- $n = 2L^2$ physical qubit



Stabilizers:

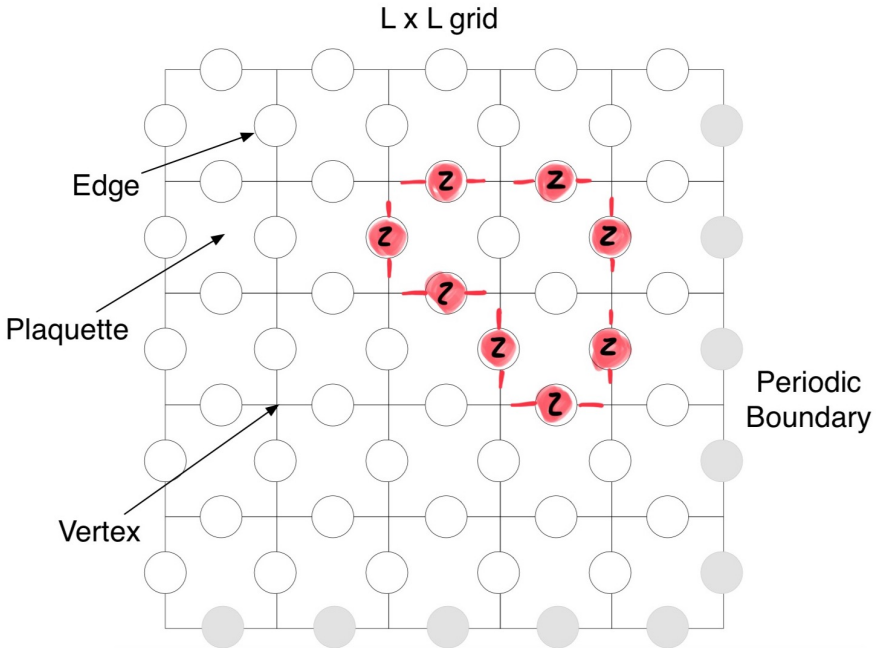
- $A_s = \prod_{j \in s} \sigma_j^x$, star operator
- $B_p = \prod_{j \in p} \sigma_j^z$, plaquette operator

$$H = -\sum_s A_s - \sum_p B_p$$

- Dan Browne, Lecture notes
- Girvin and Yang, Modern condensed….

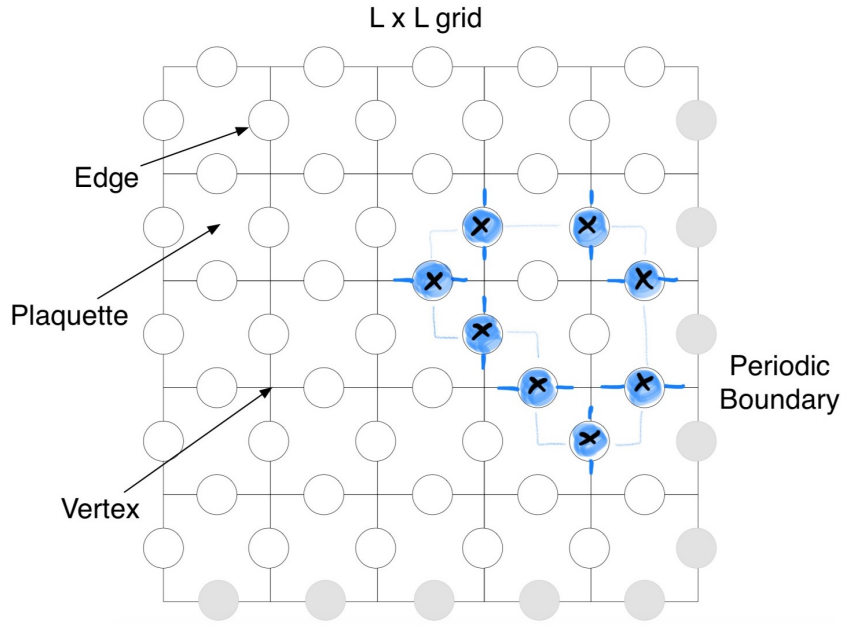
The Toric Code

- Stabilizer commute between each other
- Product of stabilizers is a stabilizer (stabilizer group…)



Stabilizers:

- $A_s = \prod_{j \in s} \sigma_j^x$, star operator
- $B_p = \prod_{j \in p} \sigma_j^z$, plaquette operator



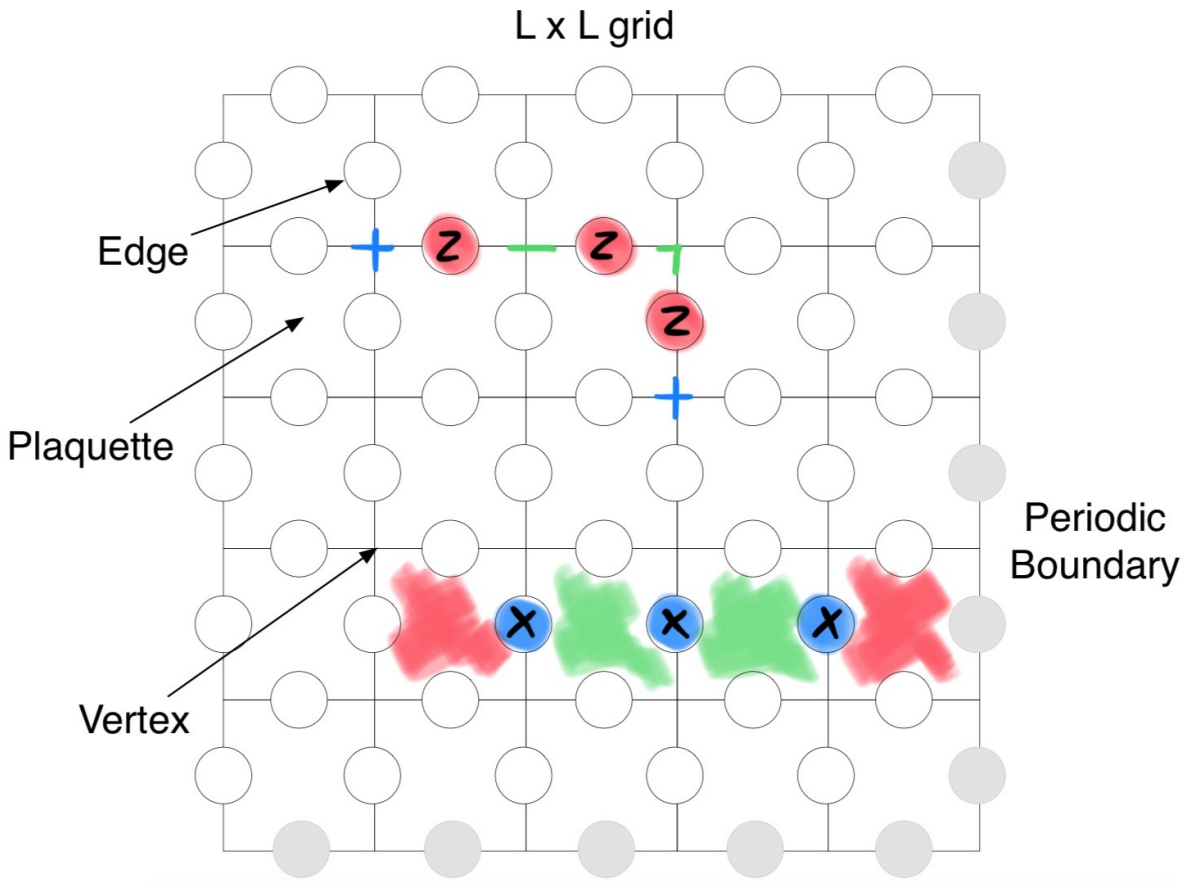
- Dan Browne, Lecture notes
- Girvin and Yang, Modern condensed….

The Toric Code

What are **errors** in this context?

1. Physical errors happen on single **qubit** (bond)
2. They **anti-commute** with “neighbouring” stabilizers
3. The measurement result is called **error syndrome**

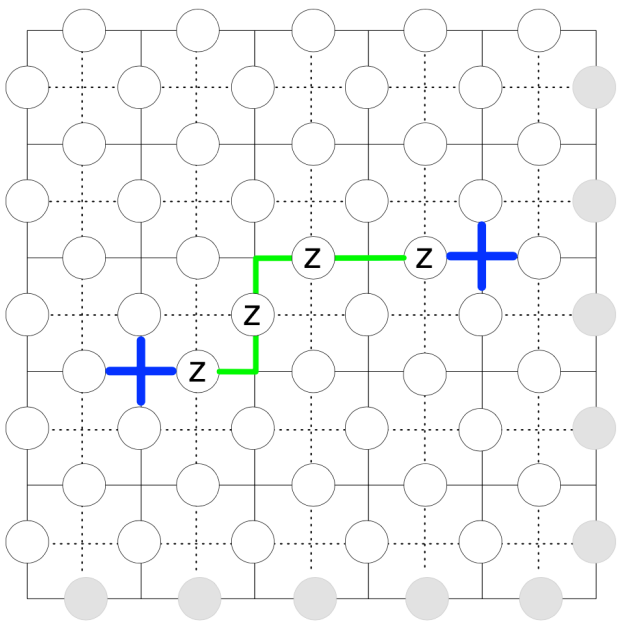
”Given a string of errors on the primal lattice, the only stabilizer generators which anti-commute with the string are the ones at the vertices”

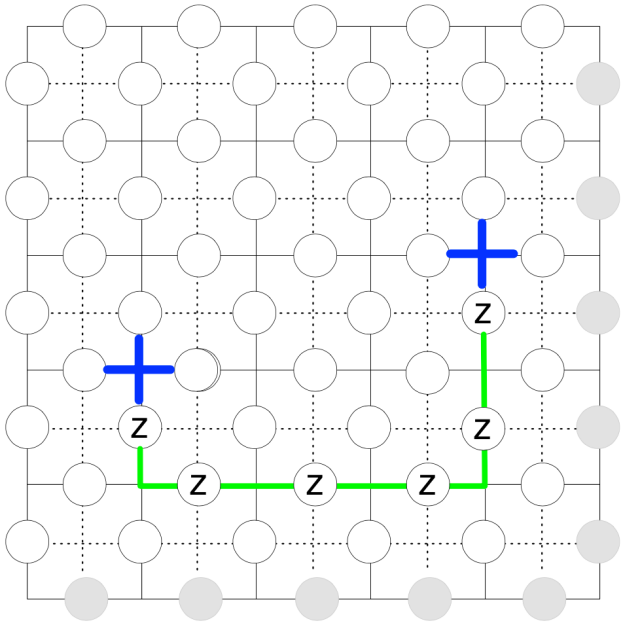


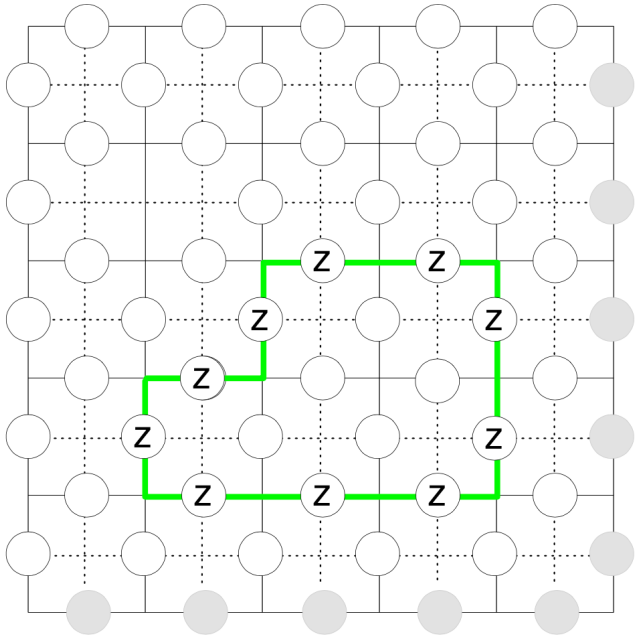
- Dan Browne, Lecture notes
- Girvin and Yang, Modern condensed….

The Toric Code

- Excitations live at the boundary of error paths
- when a path forms a loop, the excitations disappear
- loops always commute with all the stabilizers.
- an operator that commutes with all the stabilizers is a logical operator!





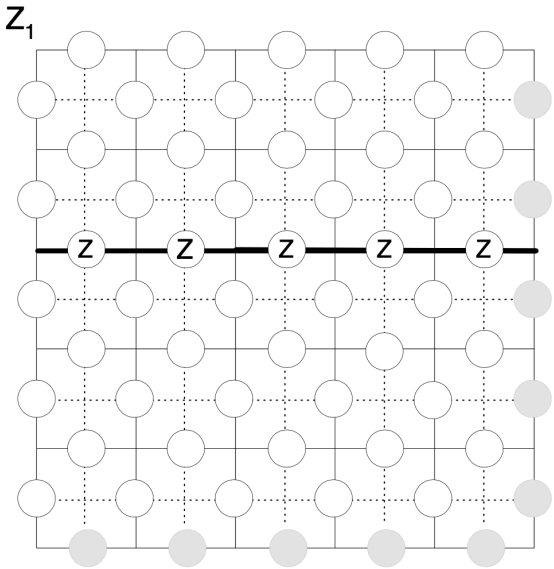


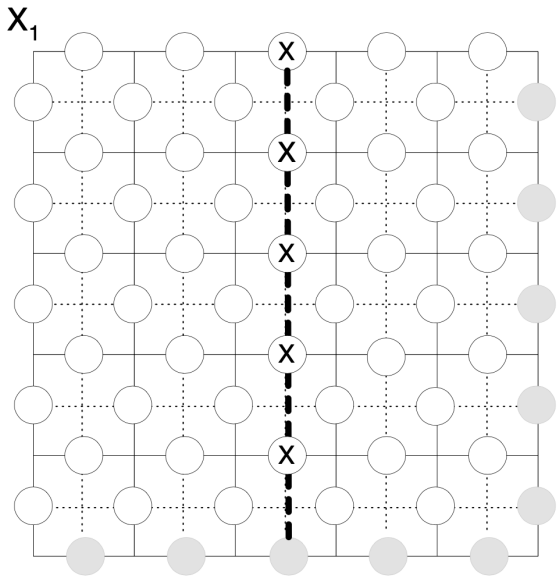
- Dan Browne, Lecture notes
- Girvin and Yang, Modern condensed….

The Toric Code

What are logical operators here?

A logical operator can either be trivial (i.e. a stabilizer), or non-trivial (X,Y,Z operator acting on one of the k encoded logical qubits)

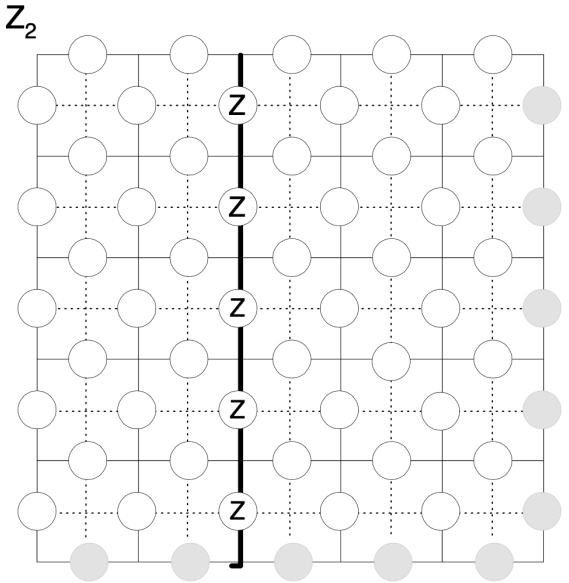


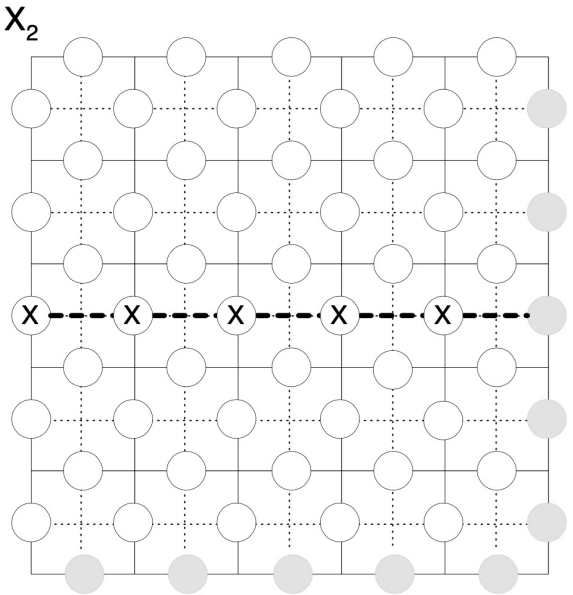


Different types of logical operator depend on different type of loop configurations!!

Notion of equivalence: two loops configuration are said to be equivalent if there exist a stabilizer such that:

$$l_1 = S \times l_2$$





Equivalence class label for X errors: $(k_1, k_2) = \mathbb{Z}_2 \times \mathbb{Z}_2$
Equivalence class label for Z errors: $(k_1, k_2) = \mathbb{Z}_2 \times \mathbb{Z}_2$

- Dan Browne, Lecture notes
- Girvin and Yang, Modern condensed….

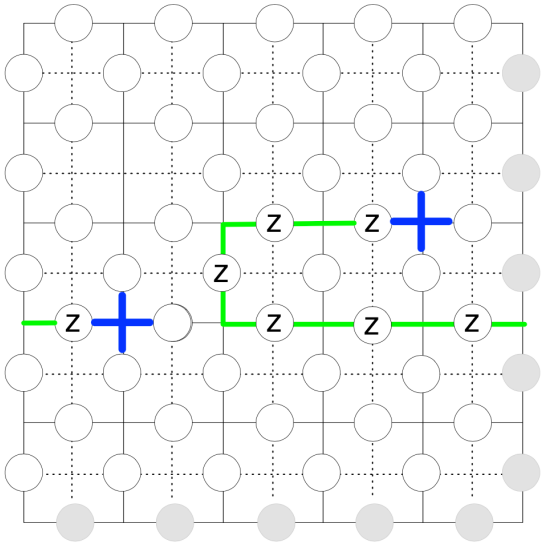
The Toric Code

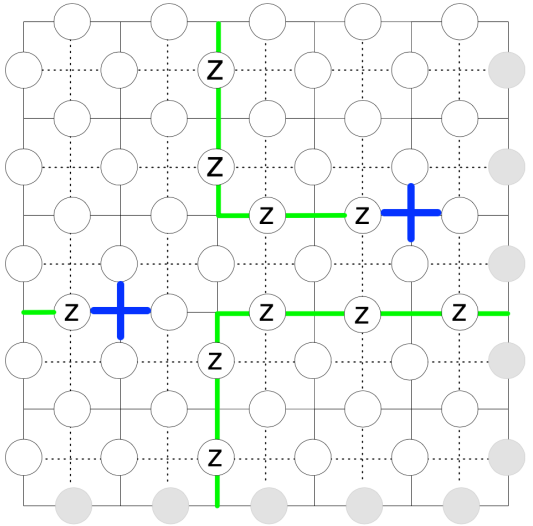
The same syndrome can be caused by different string errors

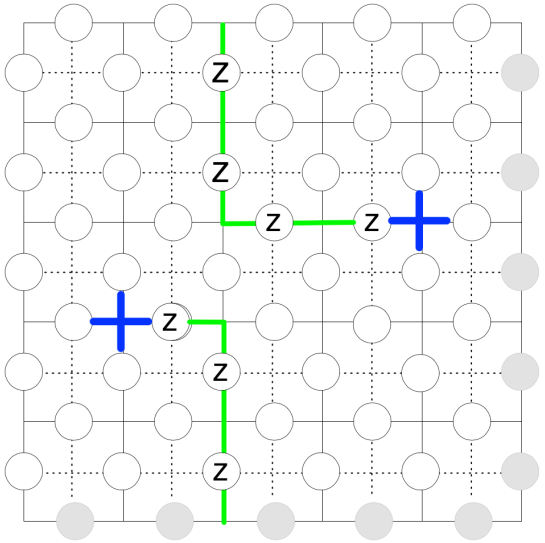
The role of DECODING is to find the string excitation that actually match with the error string!

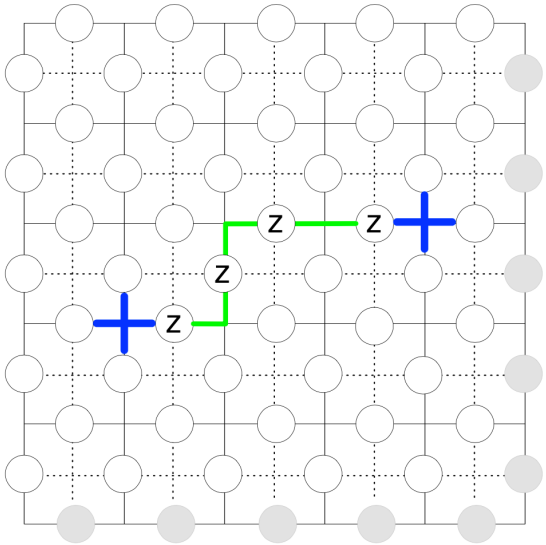
A wrong correcting string result in a logical error!

Different types of decoding algorithm exist with their pro and contro…







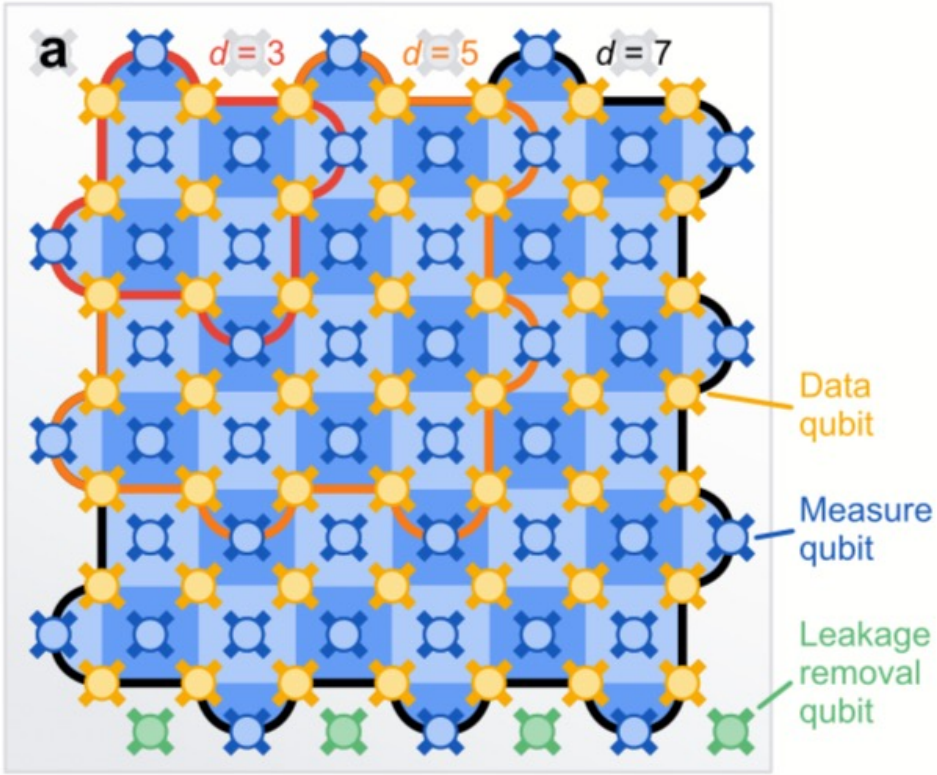


- Dan Browne, Lecture notes
- Girvin and Yang, Modern condensed….

Implementing a quantum memory

Context: realization of surface code operating below threshold on two superconducting processors. The d-7 code is realized on a 105-qubit processor while the 5-d code on a 72-qubit processor.

105-qb processor: square grid of transmon qubits with improved operational fidelities compared to previous work.

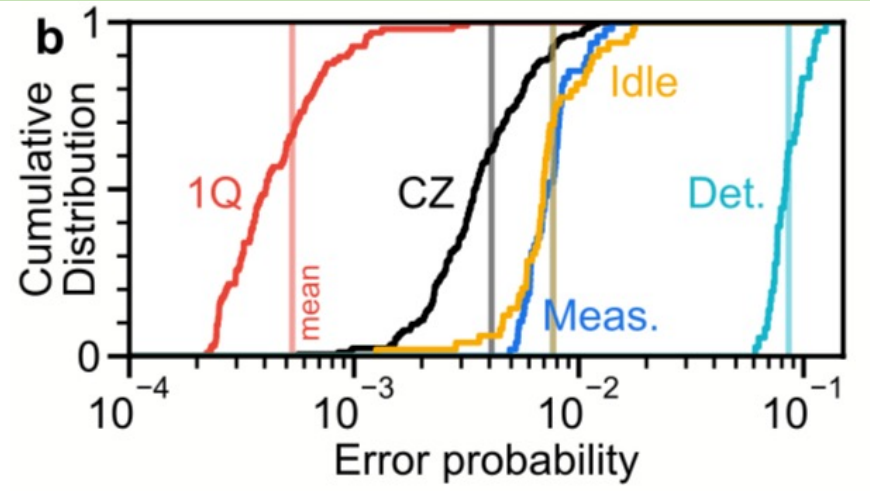


- d-7 and d-5 surface code memories operating below threshold
- $\Lambda = \frac{\varepsilon_d}{\varepsilon_{d+2}} = 2.14 \pm 0.02$ error suppression factor with distance

Implementing a quantum memory

"If the physical operations are below a critical noise threshold, the logical error should be suppressed exponentially as we increase the number of physical qubits per logical qubits"

Cumulative distributions of error probabilities:



$$\varepsilon_d \propto \left(\frac{p}{p_{thr}}\right)^{\frac{d+1}{2}}$$

d= code distance

$2d^2 - 1$= number of physical qubits per logical qubit

p= physical error rate

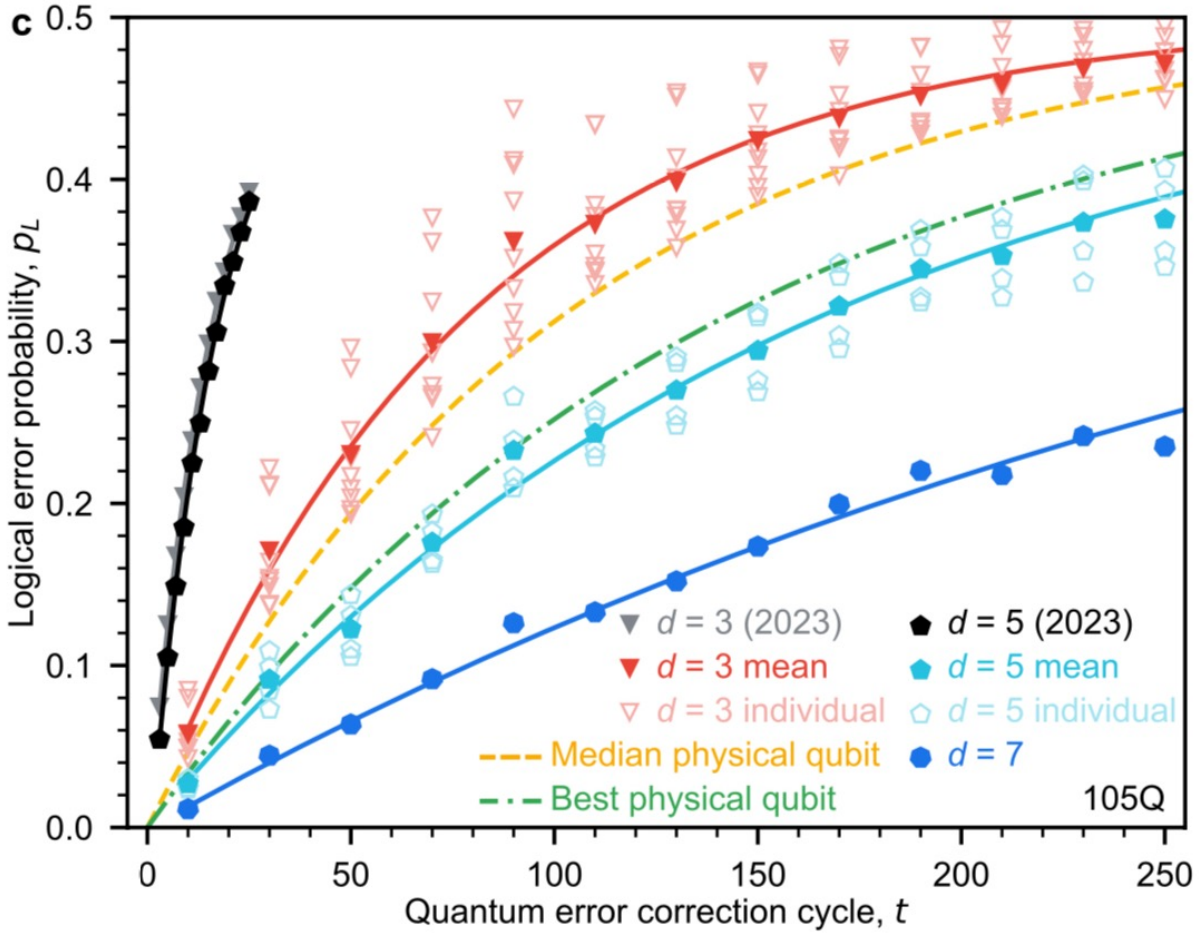
ε_d =logical error rate

p_{thr}= threshold error rate

The threshold depend on the particular type of circuit-code!!

Google Quantum AI, 2024

Implementing a quantum memory



1. Prepare data qubits in an initial logical state
2. Repeat a variable number of error correction cycles
3. Send the error syndromes to an external decoder
4. Run data qubits leakeage removal (DQLR) to ensure leakage to higher states is short-lived
5. Measure the final logical qubit and check if the decoder's corrected logial measurement outcomes agrees with the initial state.

Google Quantum AI, 2024

Implementing a quantum memory

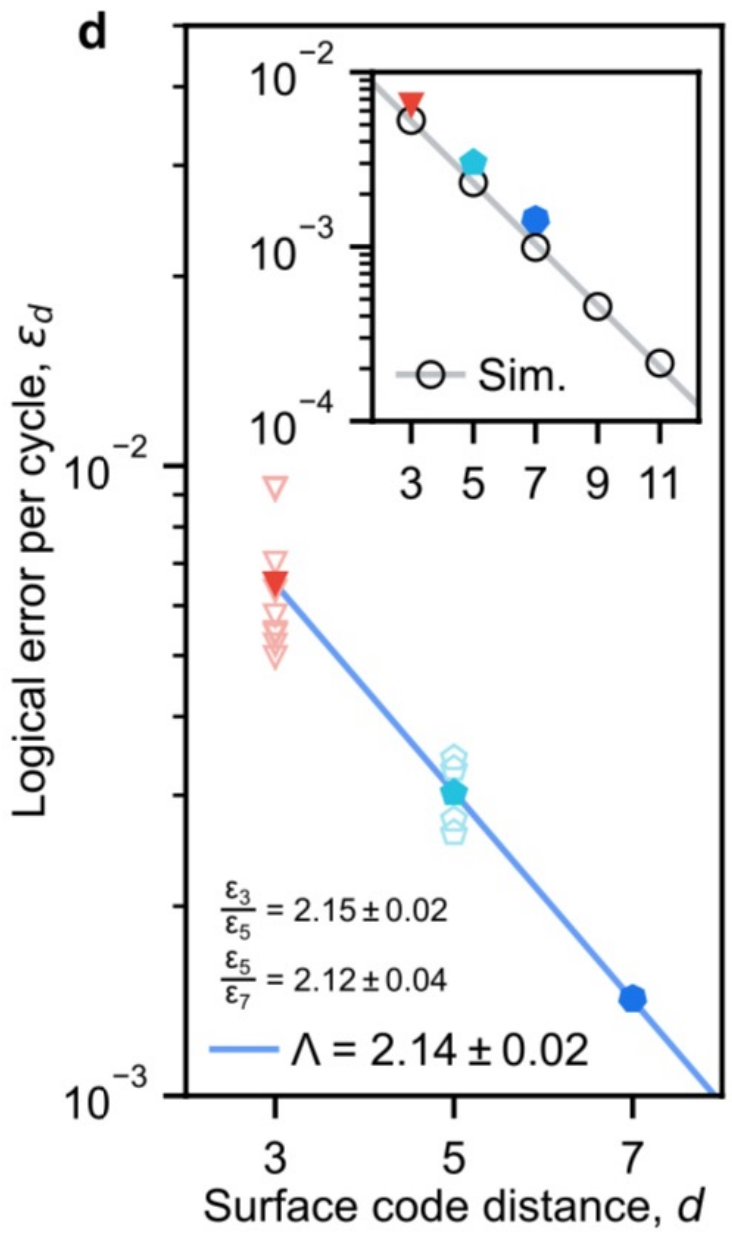
From the cumulative logical error probability, they extract the logical error rate

$$p_L = \frac{1}{2}\left(1 - (1 - 2\,\varepsilon_d)^t\right)$$

Solving for eps_d we get:

$$\varepsilon_d = \frac{1}{2}\left(1 - (1 - 2\,p_L)^{\frac{1}{t}}\right)$$

$\Lambda = \frac{\varepsilon_d}{\varepsilon_{d+2}} = 2.14 \pm 0.02$ error suppression factor with distance



Programmable Quantum Processor

Programmable quantum processor based on encoded logical qubits operating with 280 physical qubits. Logical-level control and zoned architecture in reconfigurable neutral-atom arrays.

Properties:

- High two-qubit fidelityies
- Arbitrary connectivity
- Single-qubit rotations
- Mid-circuit readout

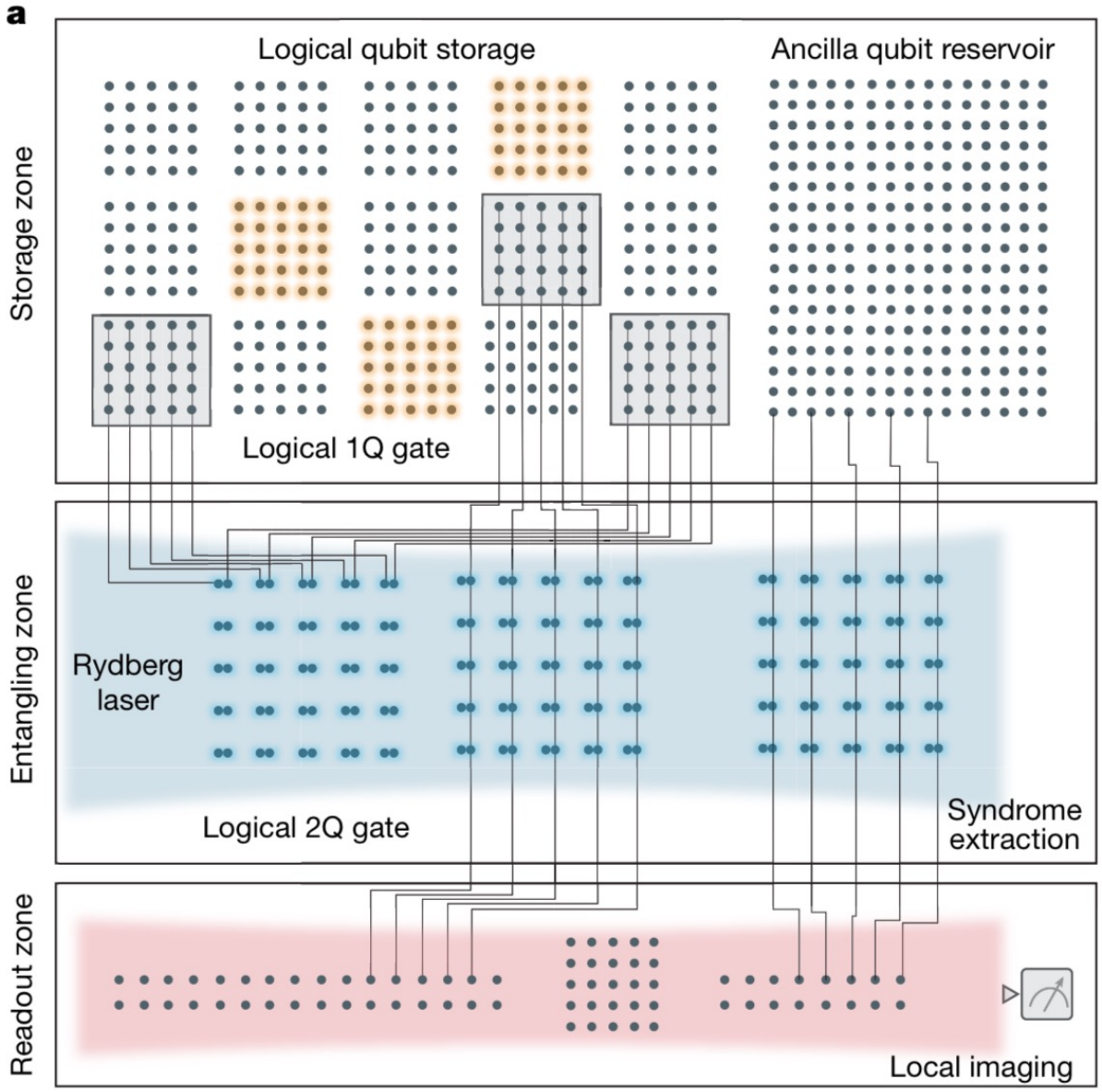
Programmable Quantum Processor

Properties:
- 280 physical qubits
- High two-qubit fidelityies
- Arbitrary connectivity
- Single-qubit rotations
- Mid-circuit readout

Storage zone:
- dense qubit storage
- free from entangling-gate errors
- Long coherence times

Entangling zone:
- parallel logical qubit encoding
- stabilizer measurements
- logical gate operations.

Readout zone
- mid-circuit readout



Bluvstein, 2023

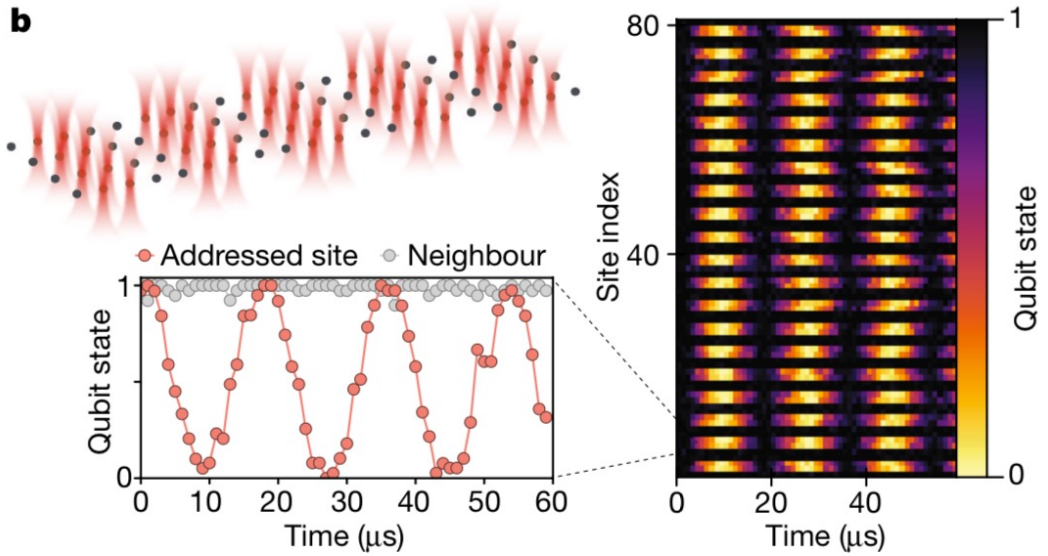
Programmable Quantum Processor

How can we control these neutral atoms?

- Tweezer array (using SLM)
- Laser tuned at a precise resonant transition

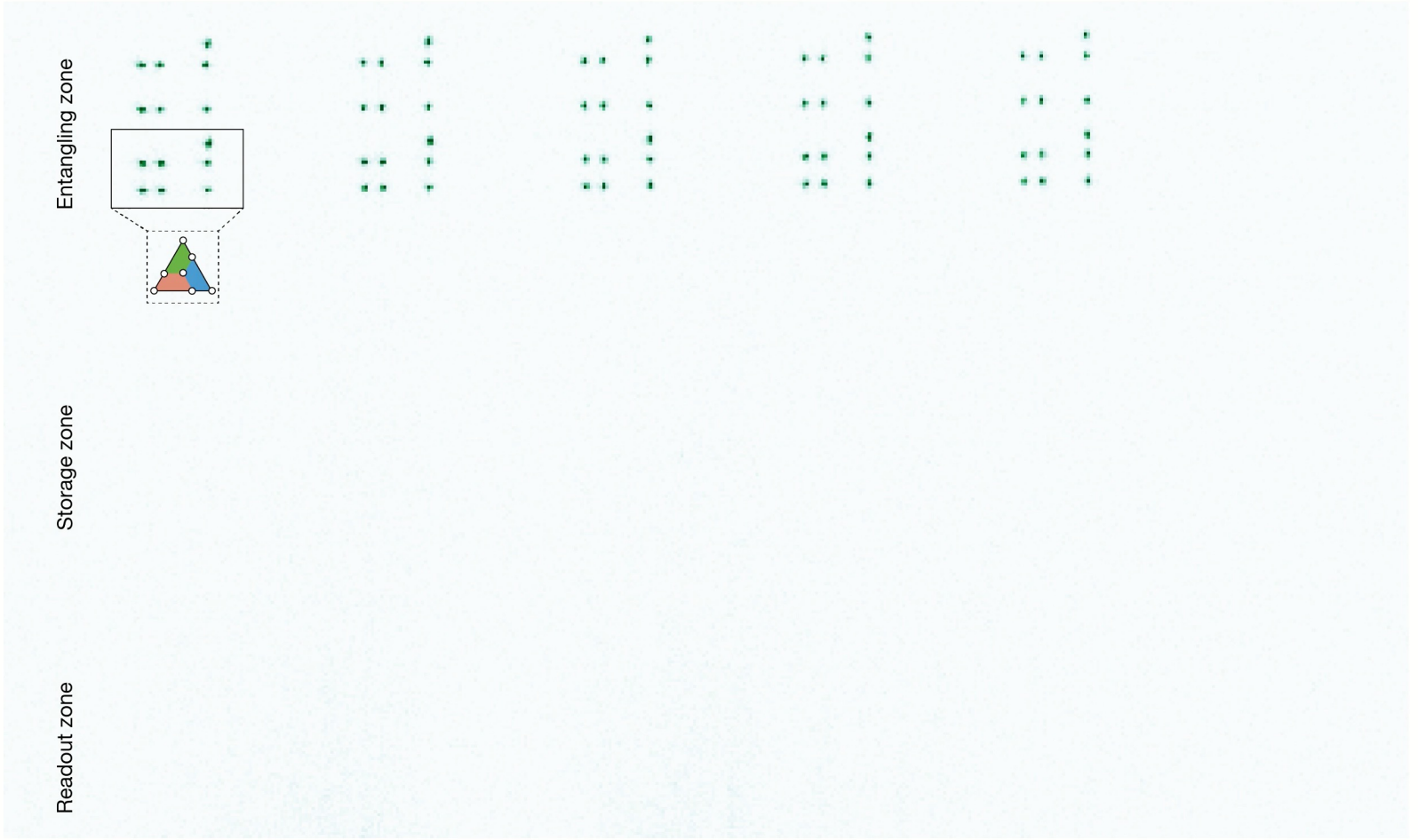
Single qubit Raman excitation through a 2D AOD;
Parallel grid illumination delivers the same instruction to multiple atomic qubits.

Operations act on physical qubit independently, i.e. it is transversal, i.e. physical errors cannot spread to other qubit



Bluvstein, 2023

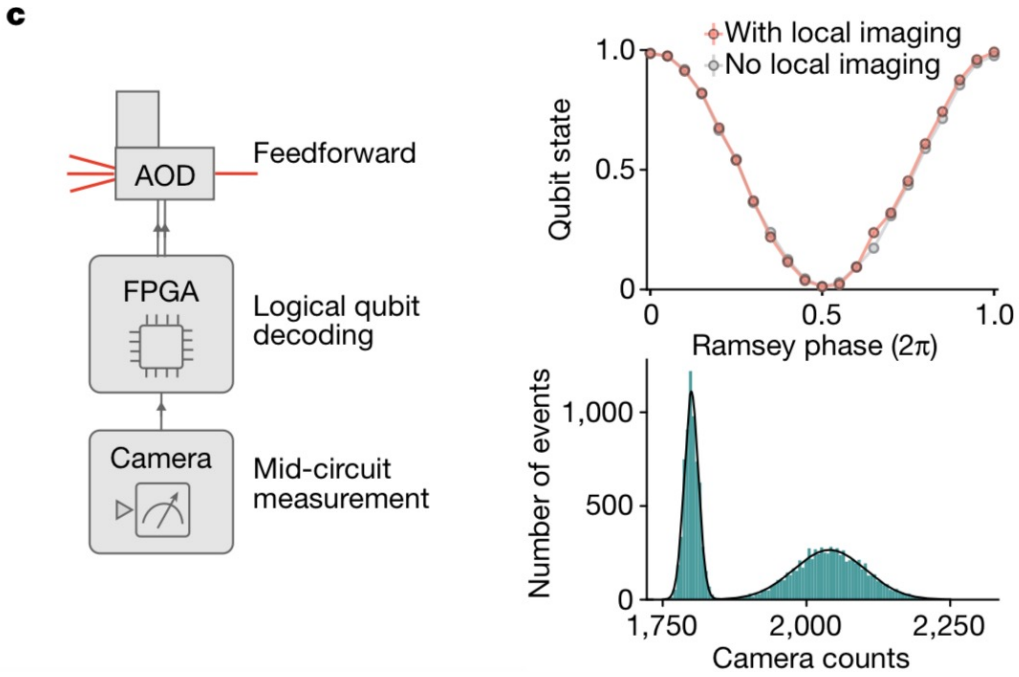
Programmable Quantum Processor



Programmable Quantum Processor

Mid-circuit readout does not affect qubit in the storage zone…

Mid-circuit readout is enabled by moving the atom 100 µm to a readout zone and illuminating with a focused imaging beam, resulting in a high-fidelity imaging, as well as negligible decoherence on stored qubits.



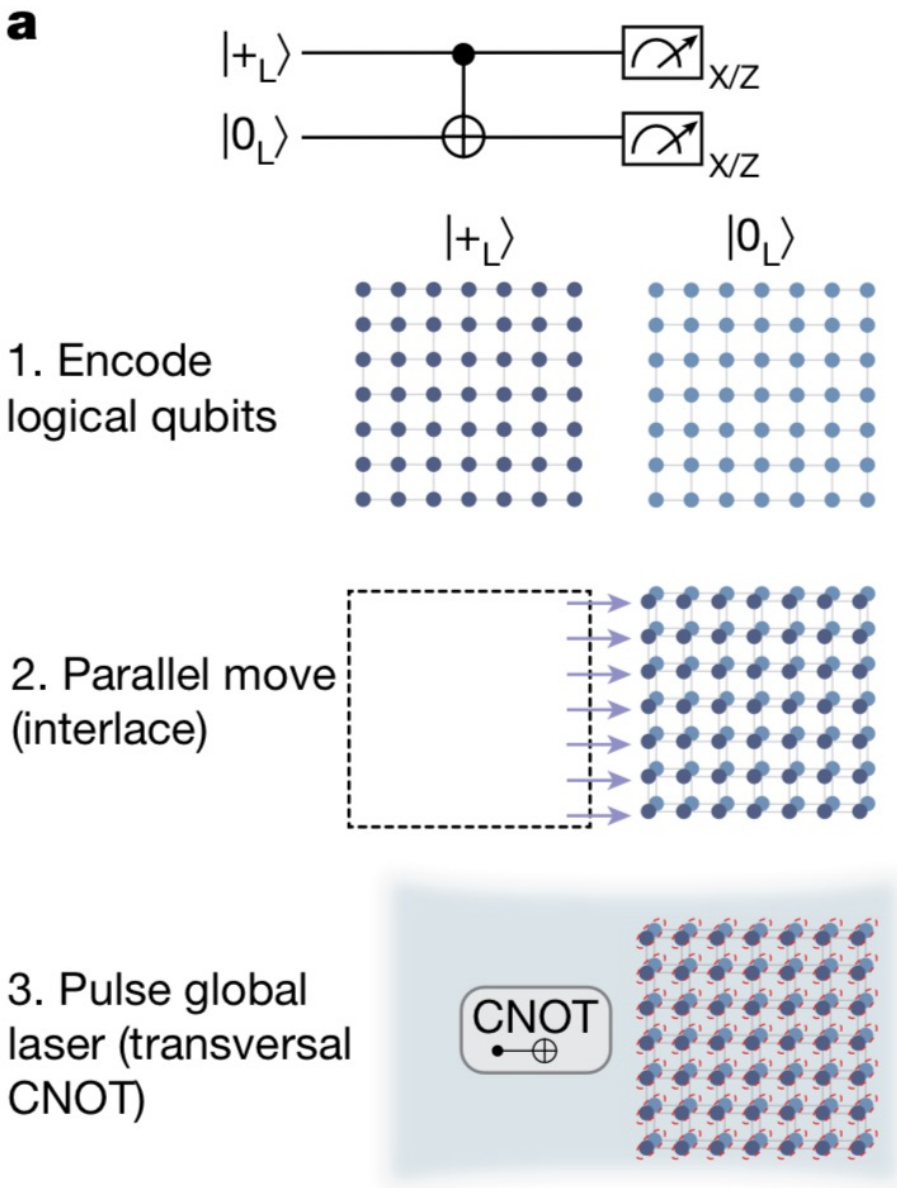
Bluvstein, 2023

Programmable Quantum Processor

Fault-tolerant implementation of a C-NOT gate under threshold for a d-7 surface logical qubit.

While for superconducting qubits, it was nice to reduce idling errors of a code, for neutral atoms qubits can be idly stored for long times with low errors, and the central challenges is to improve entangling operations with code distance.

The transversalilty property implies that the gate is inherently fault-tolerant, meaning errors cant spread within the code block (i.e. no logical fault).



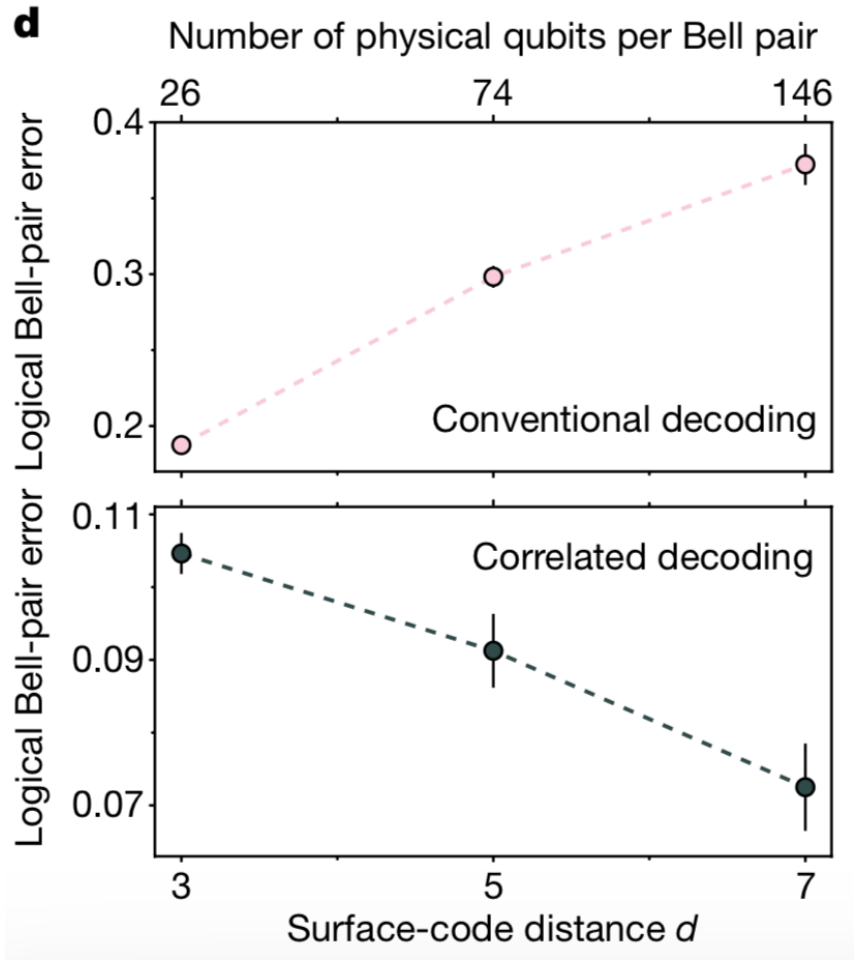
Programmable Quantum Processor

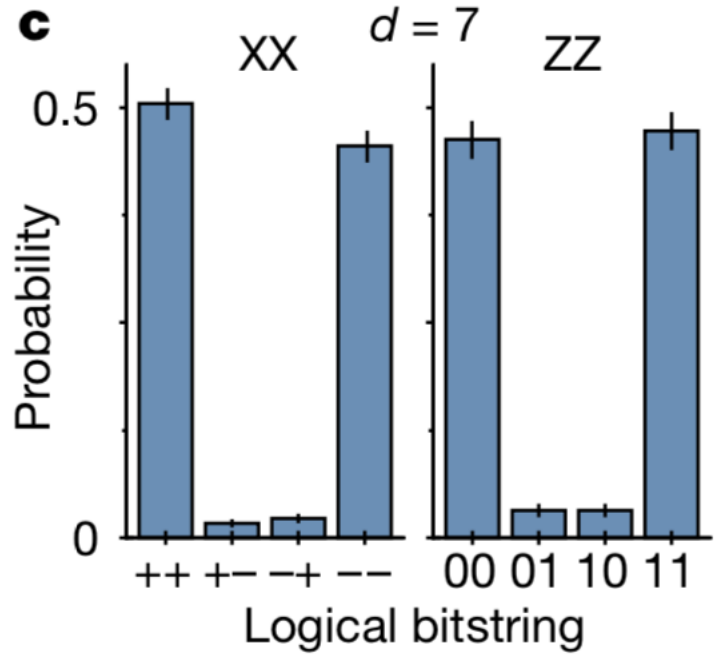
$[[d^2, 1, d]]$

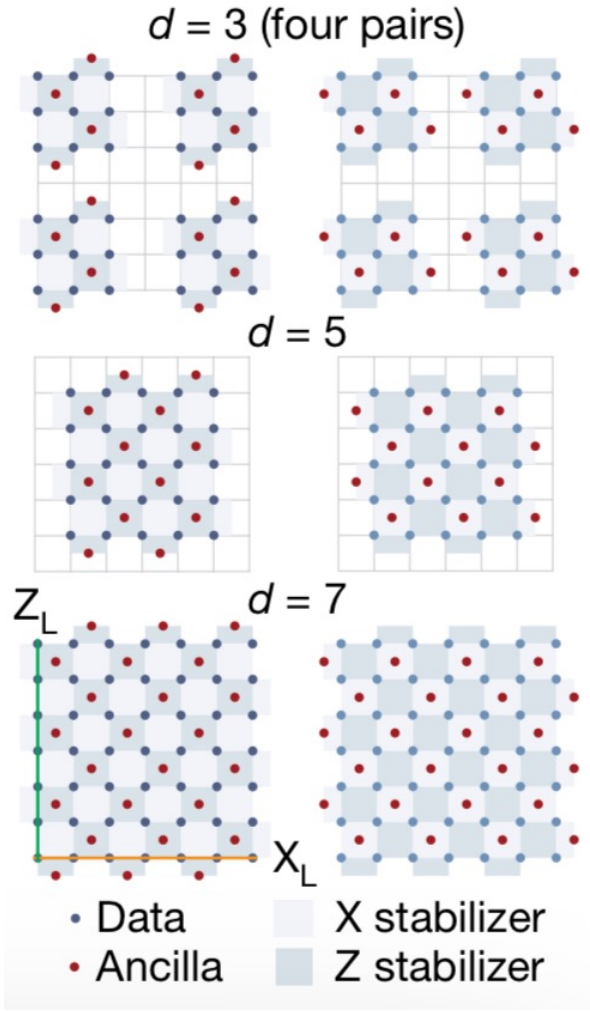
d^2 = number of physical qubits

$d^2 - 1$ =number of ind. Stabilizer

Fault-tolerant implementation of a C-NOT gate under threshold for a surface logical qubit.

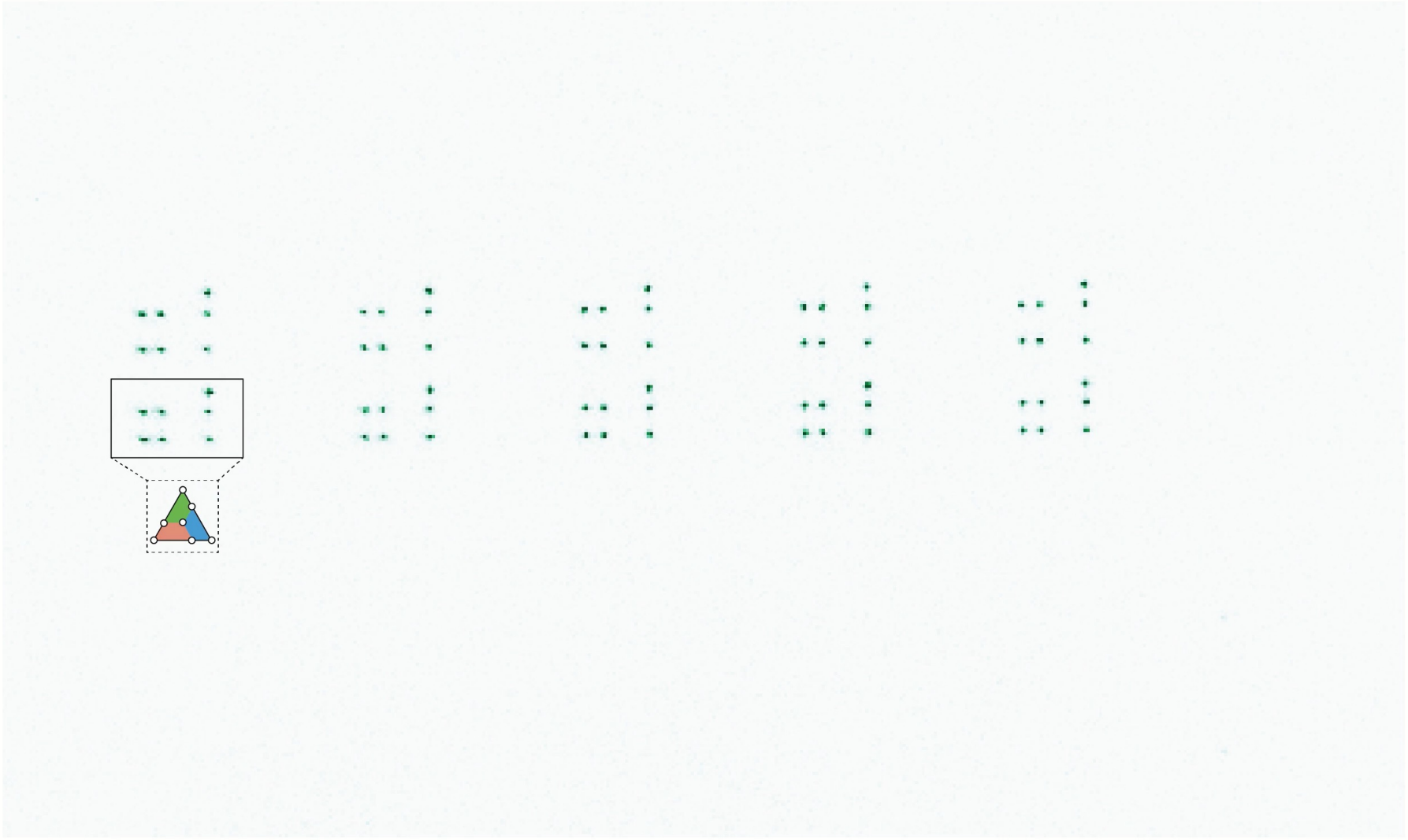






Bluvstein, 2023

Programmable Quantum Processor



Summary or Comparison…

	Superconducting circuit	Neutral atom
T1 coherence time	15-30 µs	7.6 s
T2 coherence time	30 µs	1500 ms
Two-qubit gate duration	12 ns	0.25 ms
Number of operations	≈2500	≈6000

Thank you !

References:

- Chater 7, Preskill
- General Wikipedia page
- Dan Browne, Lecture notes from websites
- Girvin and Yang, Theoretical condensed matter physics
- Google Quantum AI, 2024
- Bulvstein et Al., 2023
- Guest post, 3 October 2023, thequantuminsider.com

The stabilizer formalism

QEC codes in terms of Pauli operators rather than state vector ket (code-words)

Advantages:

1. Efficient description of many-qubit states
2. Straightforward error detection and decoding
3. Encoded logical operators

- Dan Browne, Lecture notes
- Wikipedia

The stabilizer formalism

Pauli group P_n → Stabilizer Subgroup S → Stabilizer code (codespace) C

Put here the three definitions from the three vectorspace…

[Do not show this slide, only for questions later…]

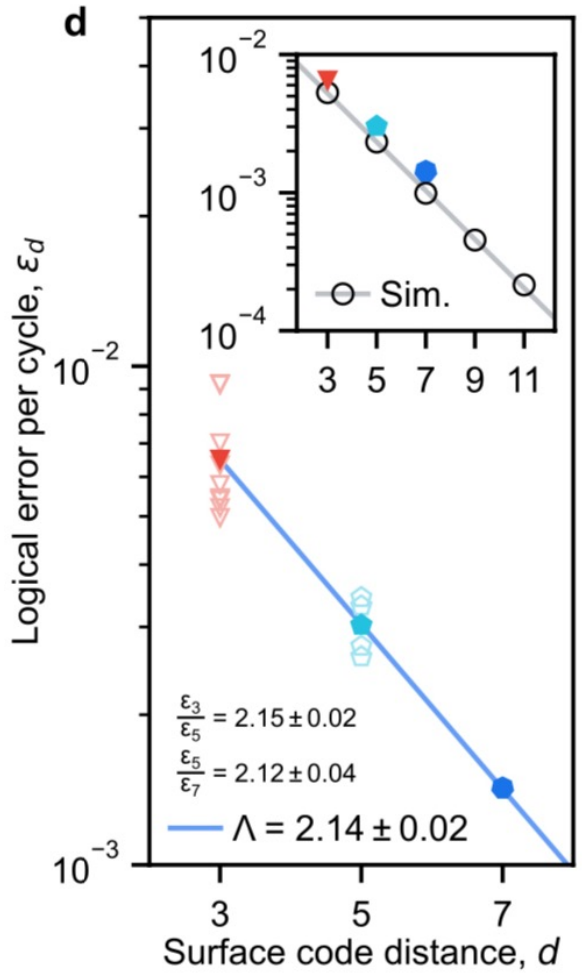
Implementing a quantum memory

Consider a binomial problem with logical error probability eps_d at each step. The cumulative error probability p_L after t cycles is the probability of an odd number of logical errors, giving the expression for p_L

$$p_L = \frac{1}{2}\left(1 - (1 - 2\,\varepsilon_d)^t\right)$$

Solving for eps_d we get:

$$\varepsilon_d = \frac{1}{2}\left(1 - (1 - 2\,p_L)^{\frac{1}{t}}\right)$$



$\Lambda = \frac{\varepsilon_d}{\varepsilon_{d+2}} = 2.14 \pm 0.02$ error suppression factor with distance

Logical error per cycle is: …put explanation here…

Surface code distance is:…put explanation here…