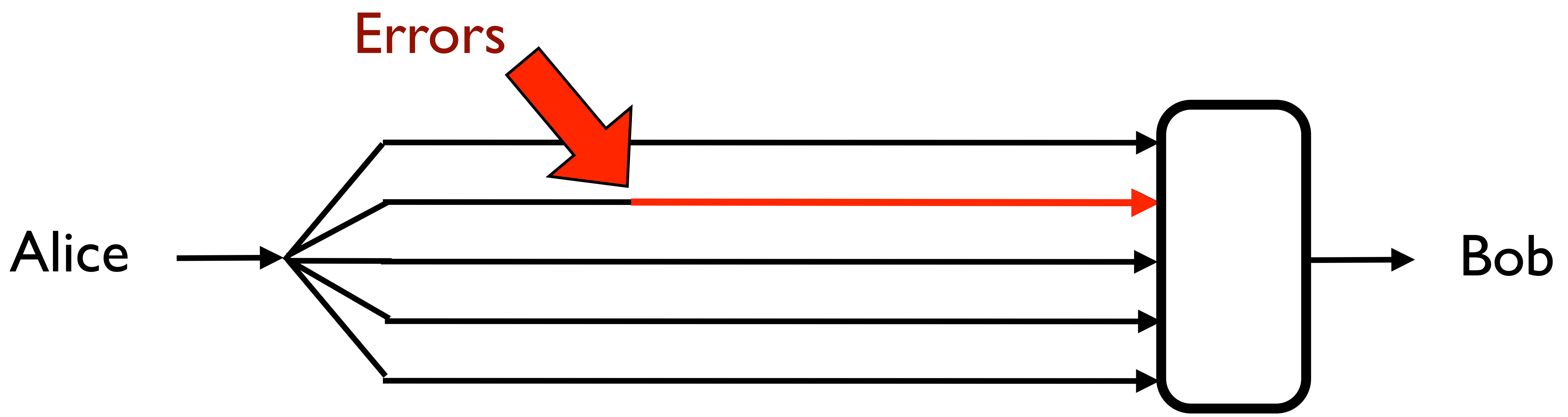
Opportunities and Challenges in Fault-Tolerant Quantum Computation

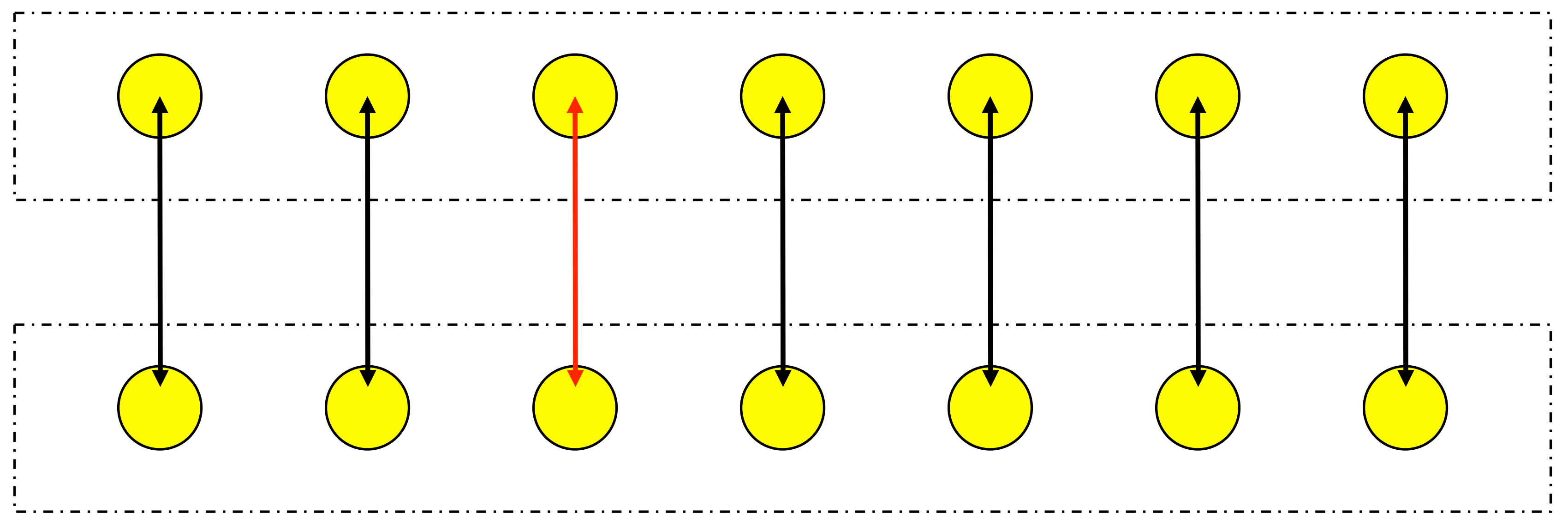
Daniel Gottesman
University of Maryland
Keysight Techologies

Error Correction vs. Fault Tolerance

Error correction: Alice and Bob have perfect quantum computers and errors only occur in transmission.



Fault tolerance: Errors occur during gates and we want to perform computations on qubits encoded in a quantum error-correcting code.



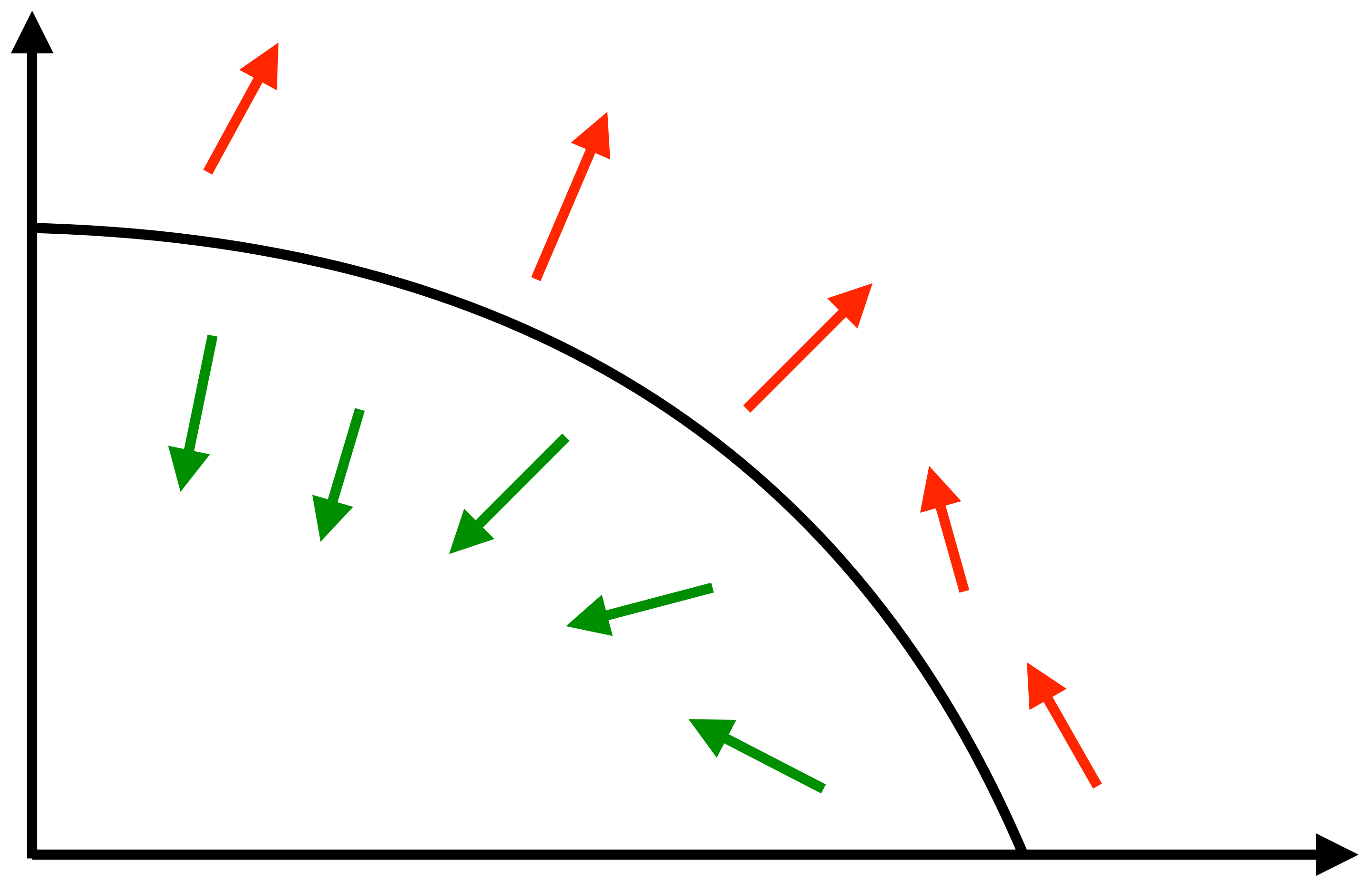
We want to avoid error propagation that causes errors to spread beyond our control.

[S96]

Threshold theorem

Threshold theorem: If the error rate per gate and time step is below some threshold value p_t, then reliable quantum computation is possible with overhead which is a polynomial in the log of the size of the computation.

The threshold depends on choice of fault-tolerant protocol, but also it is not a single number: There are many parameters in an error model because different gates having different errors and there are many possible types of error in each gate.



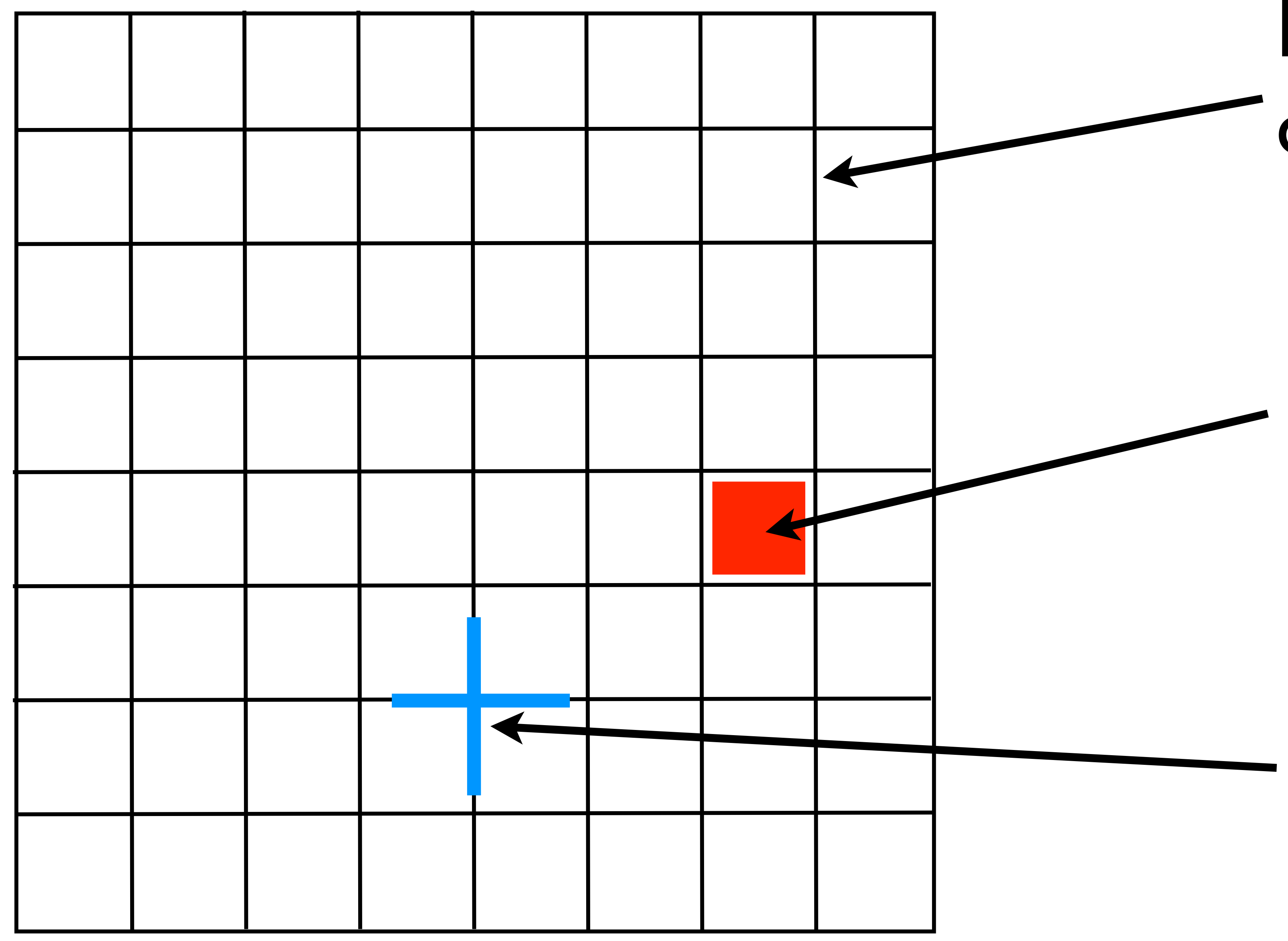
Instead, the threshold is actually a surface in a high-dimensional space.

[KLZ96, AB96, K97, AGP05]

Surface Codes

Surface codes are today's standard approach to fault tolerance for large quantum computers.

A surface code is based on a graph covering a surface. The codewords are +1 eigenstates of operator constraints based on the graph:



Each edge of the graph has a qubit associated with it.

Each face (plaquette) has a constraint

$X \otimes X \otimes X \otimes X$

Each vertex has a constraint too

$Z \otimes Z \otimes Z \otimes Z$

Logical qubits are associated with topological non-trivial loops in the surface -- e.g., a surface code on the torus has 2 logical qubits.

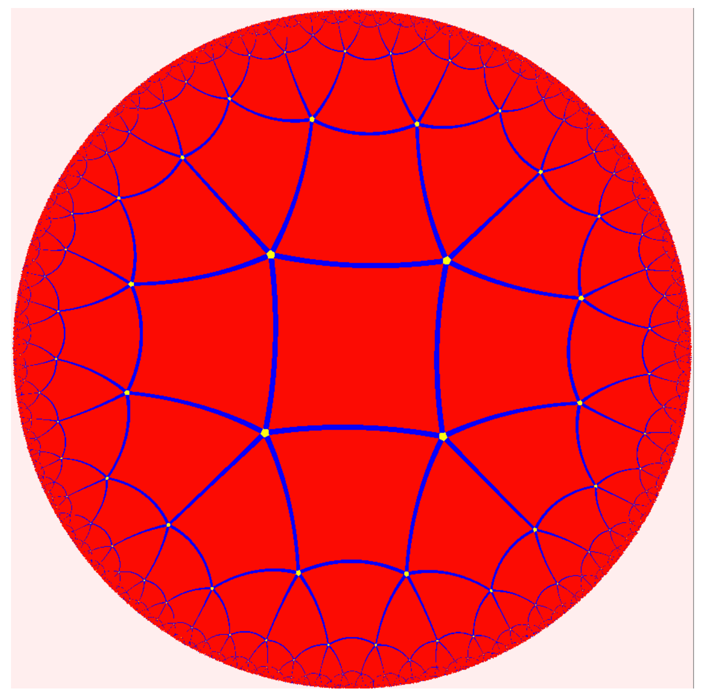
[K97, DKLP01, RH06]

Low-Density Parity Check Codes

LDPC codes are quantum error-correcting codes with two additional properties:

- Each constraint involves only a constant number of qubits
- Each qubit is in only a constant number of constraints

Surface codes are an example of LDPC codes.



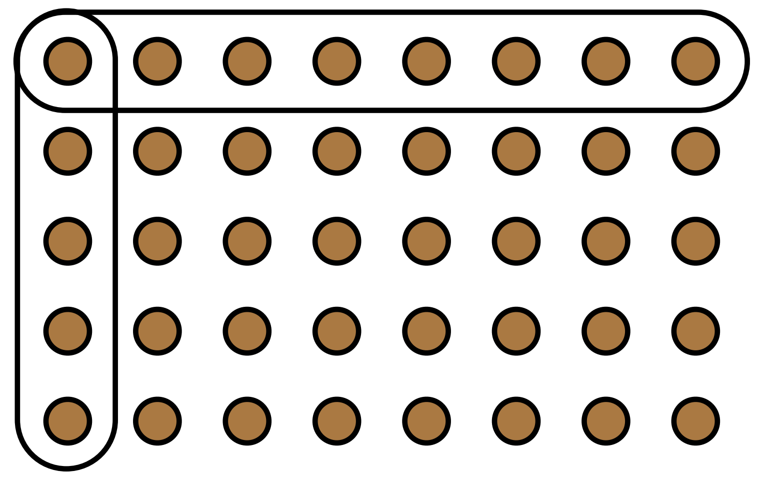
The advantage of LDPC codes for fault tolerance is that to perform error correction, one only needs to touch a small number of qubits. There are fault-tolerant methods of error correction for many other codes, but they generally involve lots of extra overhead. An LDPC code does not need that.

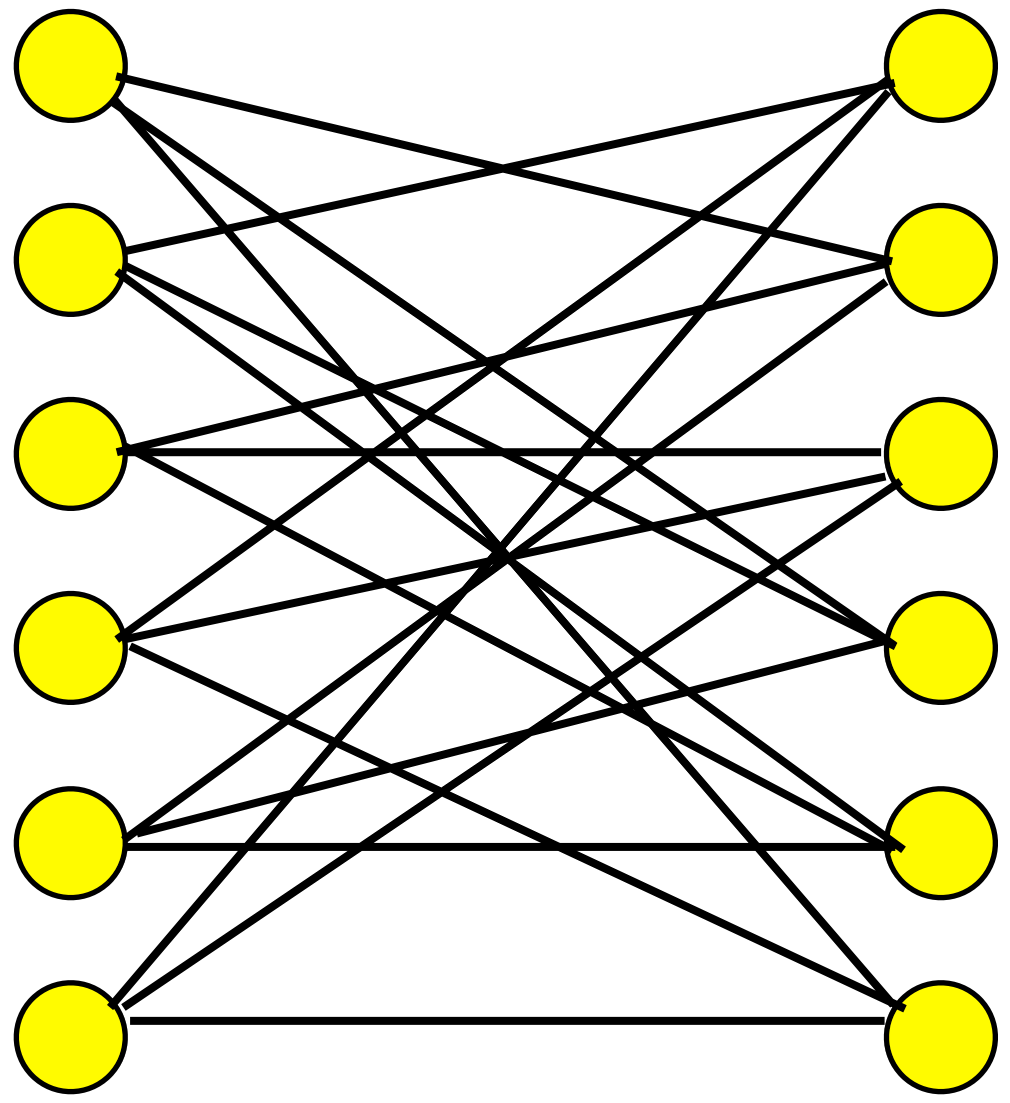
In order to get low overhead, we should work with LDPC codes which encode qubits at a high rate. High-rate LDPC codes allow a threshold for fault tolerance with a constant qubit overhead as the computation gets larger.

[G13]

Better LDPC Codes?

There are families of LDPC codes that have constant rate (ratio logical qubits/physical qubits) and correct typical errors, such as hypergraph product codes.





In the past two years, we have finally resolved a long-standing open question with the discovery of good LDPC codes, i.e., ones with a constant rate which can correct worst-case errors on a constant fraction of qubits.

It remains unclear if these new codes will lead to better fault-tolerant protocols or not.

[TZ09, LTZ15, PK21, LZ22]

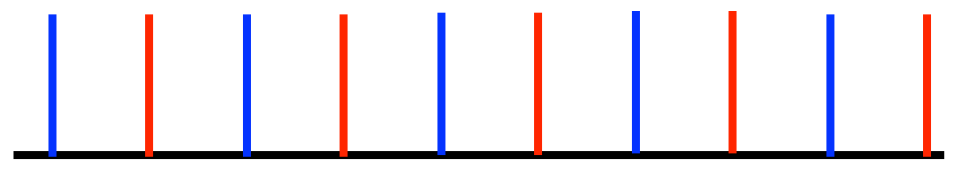
LDPC Codes vs. Surface Codes

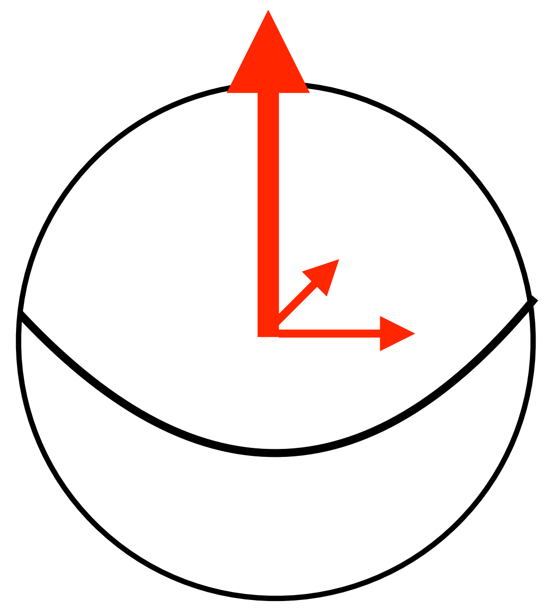
- Threshold: Surface codes have a threshold of about 0.7%, whereas LDPC codes have been shown to work with threshold of about 0.3%.
- Overhead: Surface codes require an overhead of hundreds or thousands of physical qubits per logical qubit, whereas LDPC codes could reduce this by an order of magnitude or more.
- Decoding: There are many different polynomial-time decoders known for surface codes, but in practice they may not be quite fast enough. Some polynomial-time decoders are known for LDPC codes as well, but this area needs active development. Some decoders may be faster than for surface codes.
- Connectivity: Surface codes can easily be arranged in 2D, whereas LDPC codes cannot.
- Logical Gate Constructions: Logical gates can be done in a variety of ways for surface codes, for instance through lattice surgery. We have some mediocre ways of doing gates in LDPC codes.
- Logical Parallelism: Surface code gates allow parallelism compatible with the geometric constraints. Existing LDPC gate constructions require sequential logical circuits.

Overhead: [TDB21,CKBB21] Decoding: [GK18, FGL18, GGKL20] Gates: [D13, KP19, CKBB21, QWV22]

Hardware-Specific Fault Tolerance

Most of our existing fault tolerant protocols are fairly generic and can work with a wide variety of different hardware systems. As we move to building actual fault tolerant systems, we will want to tailor the protocols to the properties of the specific systems.





- Use the full Hilbert space: e.g., bosonic codes such as the GKP code.
- Use more information about the error models: e.g., fault tolerance for dominant phase errors.
- Deal with more general types of errors: e.g., cross-talk and leakage.
- Adapt codes to specific features of the hardware: e.g., adapt to connectivity of the hardware.

Bosonic codes: [GKP00, AND+17, FNM+19, CET+20]

Dephasing codes: [AP08, PSG+19, GM19, ATB+20]

New Approaches to Fault Tolerance

We still have plenty of room for improvement of fault tolerant protocols. New techniques and new ways of thinking about fault tolerance would be helpful.

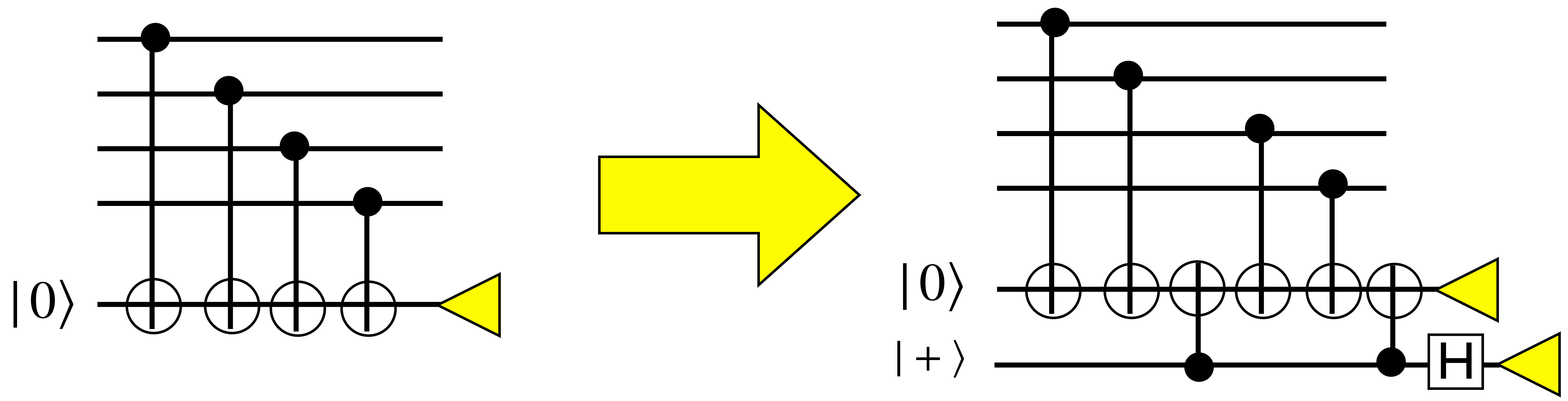
One such new approach would be to think about fault-tolerant protocols in a picture treating space and time on a more equal footing. Consider the following fault-tolerant tools:

- Flag fault tolerance
- Code deformation
- Floquet codes
- Nickerson-Bombin states

Together they point the way to this idea.

Flag Qubits

Traditional approaches to fault tolerance insist that we should arrange our circuits so one faulty gate can only cause one error in a block of the code.



Flag fault tolerance relaxes that constraint, instead allowing gates which cause error propagation within a code block but adding extra checks so that we can identify the location which originally caused the error. If we know where the error began, we know what kind of propagation it has undergone and we can correct it even though it is on multiple qubits.

Lesson: Error propagation is OK if we know the faulty gate.

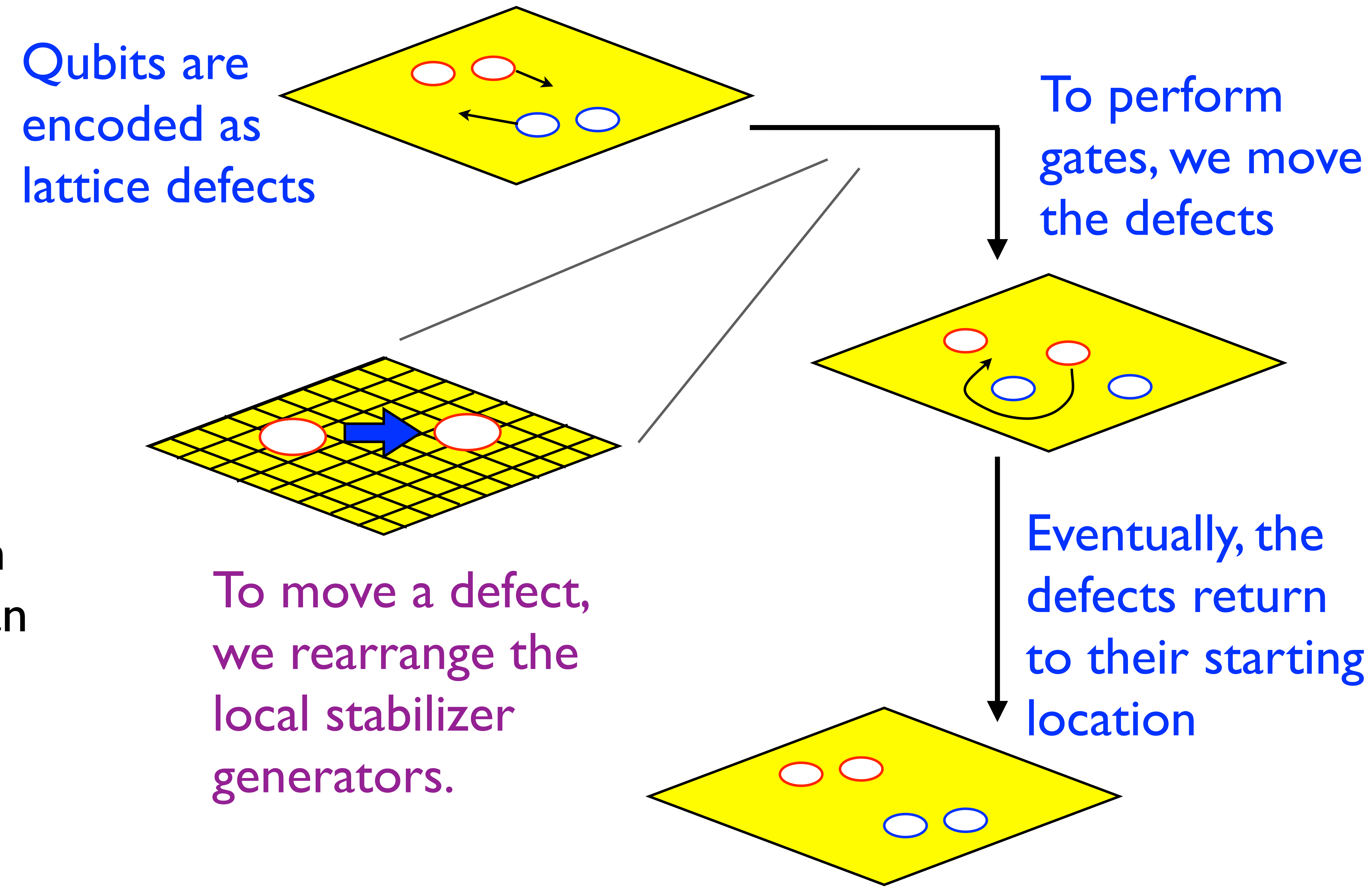
[CR17, SYK+22]

Code Deformation

What does it mean to do gates via code deformation?

We deform through a series of codes, eventually returning to the original code. By doing a topologically non-trivial path, we can do an encoded gate.

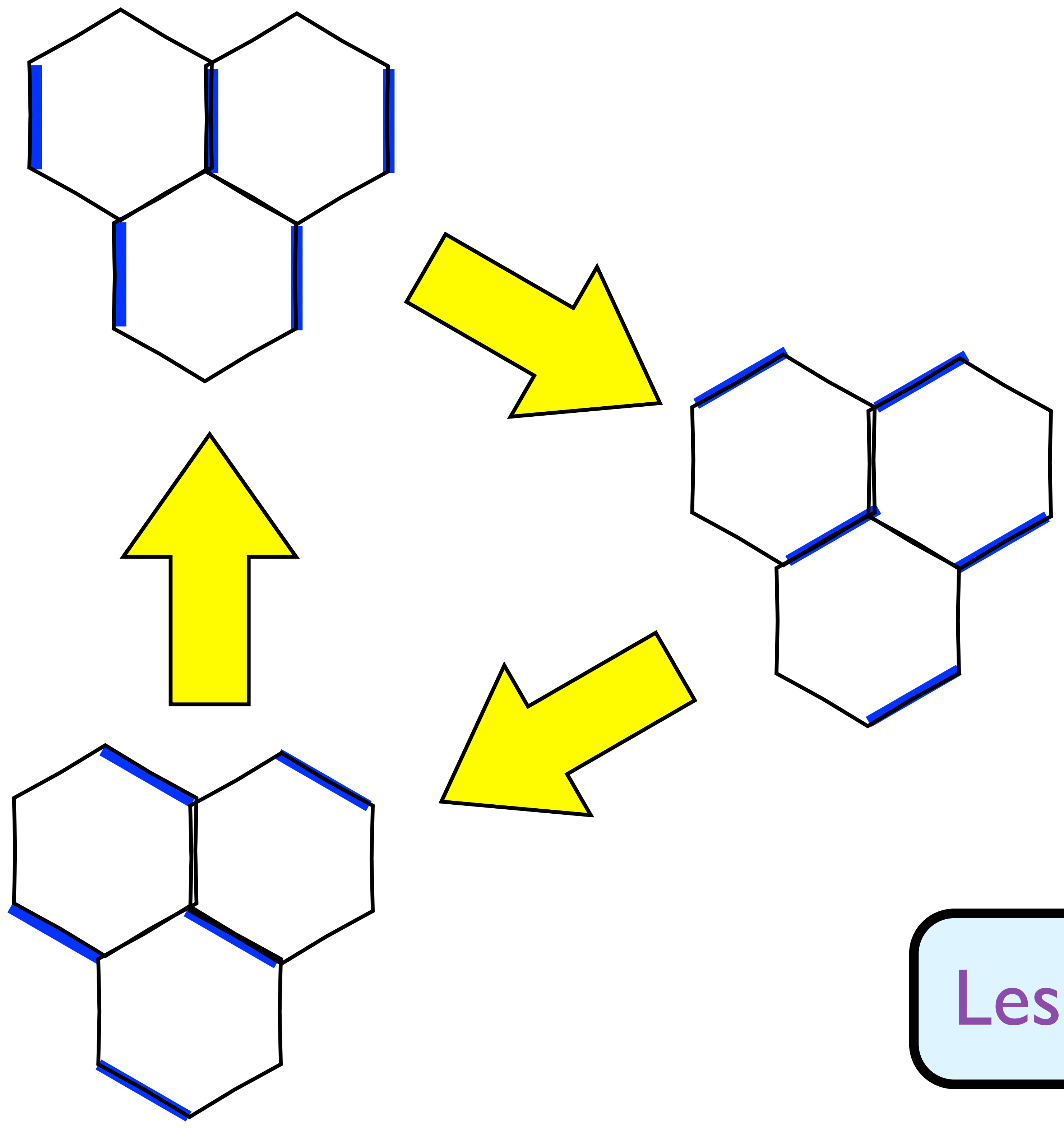
This example shows how to do it for surface codes, but in fact, any fault tolerant gate can be thought of as a code deformation.


Qubits are encoded as lattice defects
To perform gates, we move the defects
To move a defect, we rearrange the local stabilizer generators.
Eventually, the defects return to their starting location

[RH06, GZ13]

Floquet Codes

If we are doing gates by shifting between a sequence of codes, why do we insist that one of them is the "right" code and the others are just temporary stopping points?



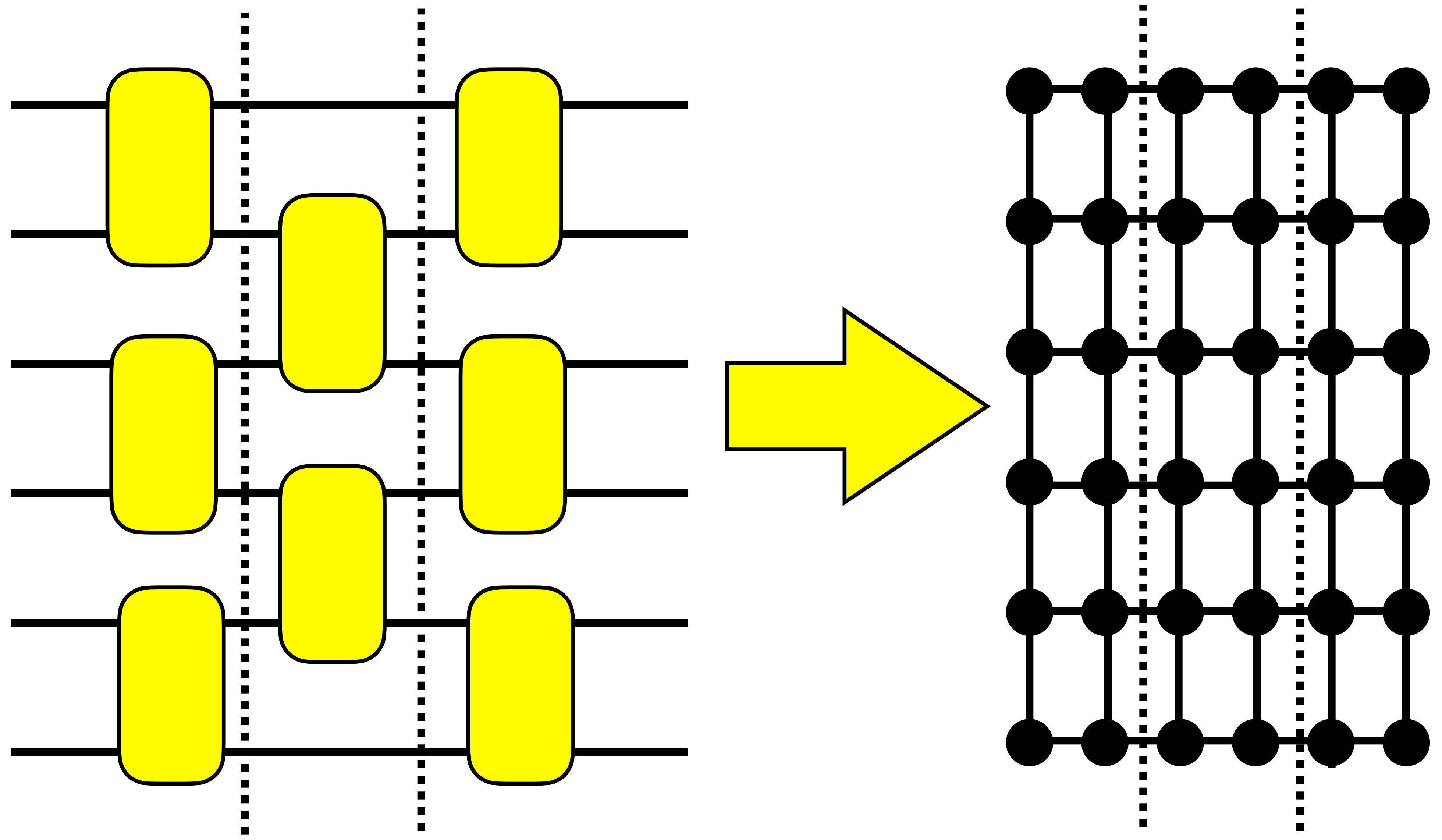
[HH21]

Floquet codes dispense with the idea of a single home code and instead cycle through a sequence of codes.

For instance, the honeycomb code implements something equivalent to a surface code by measuring sequences of two-qubit operators to learn error syndromes and shift between codes.

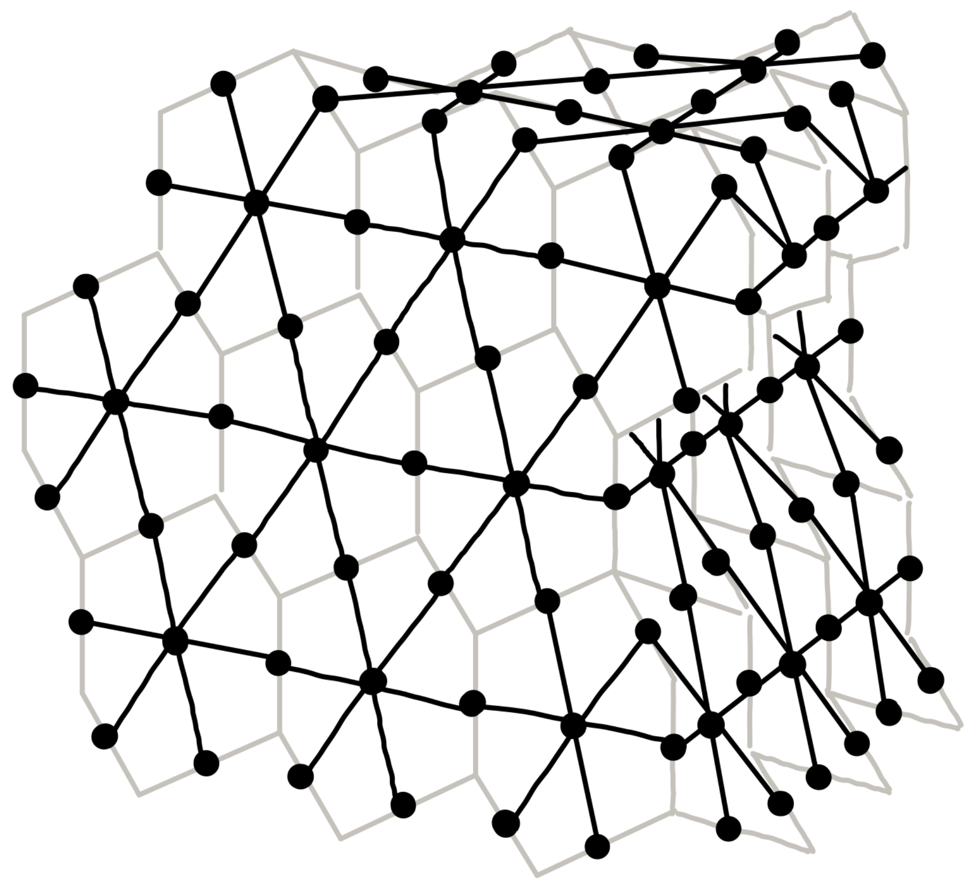
Lesson: The code can change with time.

Nickerson-Bombin States



In measurement-based computation, we start with a large entangled state (a cluster state) and make a sequence of single-qubit measurements. We can convert any quantum circuit into an appropriate measurement pattern, with the measurements arranged into layers, one for each time step in the circuit.

But in the measurement-based model, there is no requirement to have such layers, and Nickerson and Bombin found states and measurement patterns with improved fault tolerance that do not have a natural breakdown into time steps.



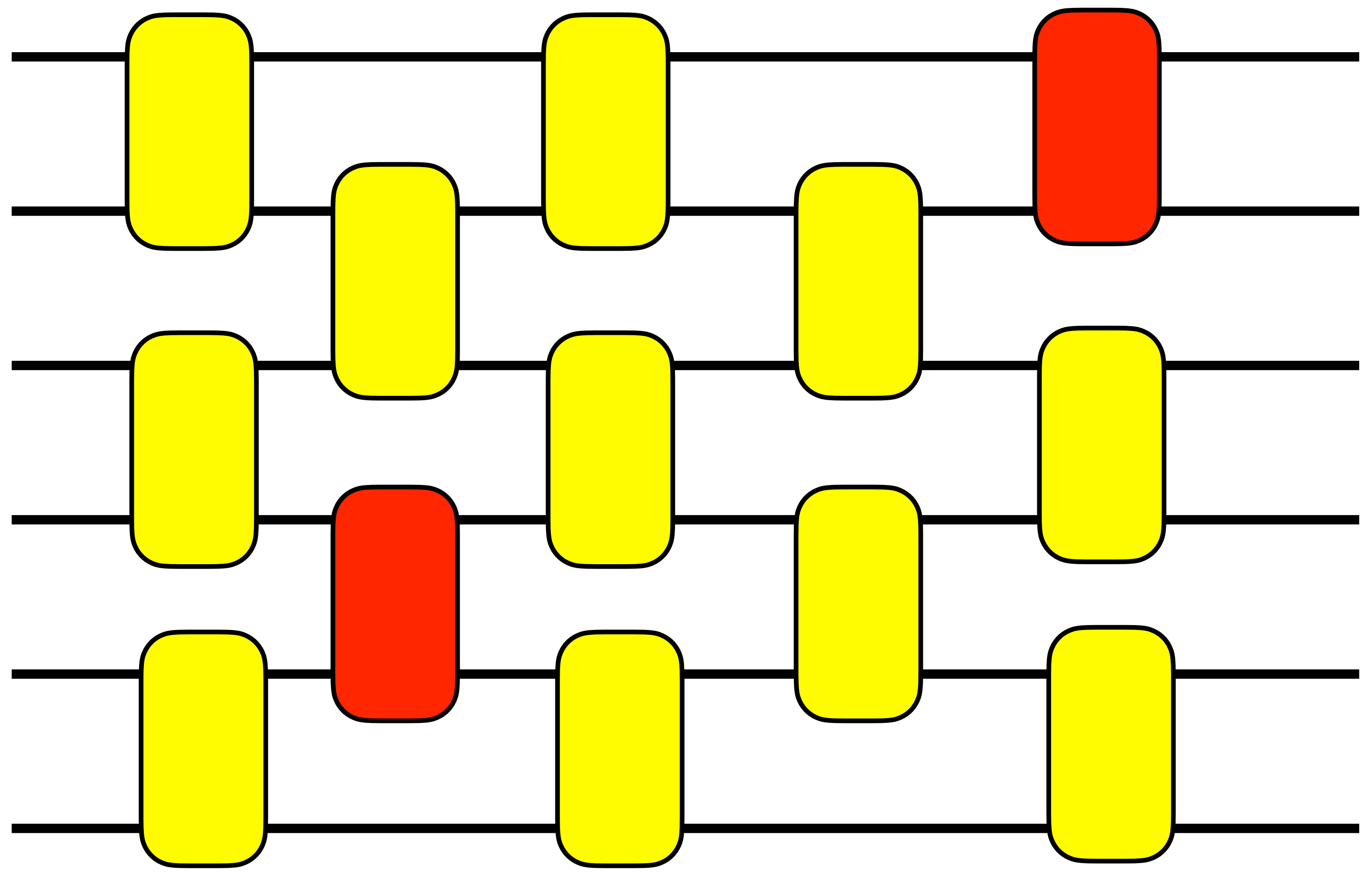
(from arXiv:1810.09621)

Lesson: Look at space and time together.

[NB18]

Putting it Together

Let us take these lessons to heart. When we do, what is left? What do we really want out of a fault-tolerant circuit?



We'd like to identify which gates had errors and what those errors are.

Can we do that? Suppose we are using n qubits to encode the information and have m additional qubits. We do a circuit containing T gates, each of which could have a possible errors.

If we measure all ancillas, we have up to m bits of information about the errors.

If the error rate is p, we expect pT faulty gates. To find the exact error, we need $T[h(p) + p\log_2 a]$ bits of information, where $h(p) = -p\log_2 p - (1-p)\log_2 p$

But if we have a circuit of depth d, we have T = d(n+m), so we require constant d. It is not clear if it possible to make fault tolerant circuits with constant depth.

Putting It Together II

We can therefore think of a fault-tolerant protocol as a space-time code: Its goal is to find the locations in space and time that have faults.

- But how is fault tolerance possible with non-constant depth circuits? The design of most fault-tolerant circuits ensures that many different faults will have the same effect on the data. They are degenerate space-time codes.
- What is the quantum code? If we stop at any time slice, the data must be encoded in a quantum error-correcting code (or we wouldn't be able to correct faults occurring then). But the code changes with time.
- How do we do logical gates if the code keeps changing? As we continually update the code, we also update what we consider to be the logical basis states. The relative relationship between the current logical state and the logical basis states may change over time, implying logical gates have been performed.
- How do we design fault-tolerant protocols in this picture? Beats me.
- Is this space-time code also a regular code on a larger space? I don't know.
- …? I don't know that either.