

Last time we discussed the toric code. This is a CSS code, where the qubits are associated with the edges of an  $L \times L$  square lattice on a 2D torus (i.e., with periodic boundary conditions). The Z-type generators of the code stabilizer are weight four, with support on the four edges making up an elementary square ("plaquette") of the lattice, and the X-type generators are weight four with support on the four edges that meet at a site. There are two encoded qubits, and the logical Pauli operators  $Z_1, Z_2$  are weight  $L$  with support on a cycle winding around the torus in the vertical and horizontal direction respectively. The logical Pauli operators  $X_1, X_2$  are weight  $L$  with support on a cycle of the dual lattice winding around the torus in the horizontal and vertical direction respectively. The code has length  $n=2L^2$ , distance  $L$ , and  $k=2$  encoded qubits.

The code is highly degenerate. A Z-type operator can be viewed as a 1-chain on the lattice, and it commutes with the stabilizer if the 1-chain is a cycle (has a trivial boundary). The operator lies in the stabilizer if the cycle is homologically trivial (is the boundary of a 2-chain); otherwise it is a nontrivial operation acting on the code space. Similarly, an X-type operator can be viewed as a 1-chain on the dual lattice; it commutes with the stabilizer if the 1-chain is a cycle, and is contained in the stabilizer if the cycle is homologically trivial.

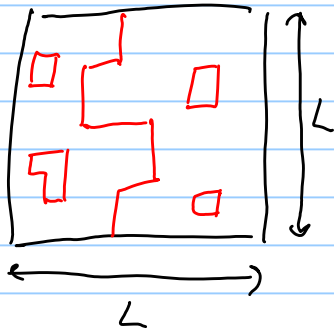
Now consider recovery from Z-type errors (recovery from X-type errors can be described similarly, with lattice and dual lattice interchanged). Suppose errors occur on a chain  $E$ .

Measuring the error syndrome (all the site operators  $\{X_s = \bigotimes_{e \in S} X_e\}$ ) reveals the boundary  $S = \partial E$  of the error 1-chain (the 0-chain with support on the sites where there are an odd number of errors on the 4 edges that meet at the site). Boundary sites have  $X_s = -1$  and other sites have  $X_s = +1$ .

To attempt recovery from error, we apply  $Z$  to a chain  $R$  with the same boundary:  $\partial R = S$ . Therefore  $E+R$  is a cycle:  $\partial(E+R) = 0$ . This means that after recovery the syndrome is trivial — we are back in the code space. However, recovery is successful only if  $E+R$  is a trivial cycle; otherwise, there has been an encoded error.

Consider an i.i.d. error model, where  $Z$  errors occur independently, with probability  $\epsilon$ , at each edge. Suppose we choose  $R$  to be the most likely error chain that can account for the observed syndrome — this is the minimum weight chain with boundary  $S$ . There is a classical

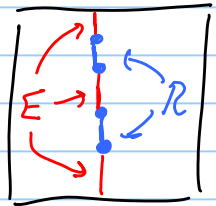
algorithm that computes the minimum weight  $l$ -chain with specified boundary in polynomial time. We wish to estimate the probability that  $E+R$  is homologically nontrivial.



If  $E+R$  is nontrivial, then it must contain a closed loop with length  $l \geq L$ .

Let's find an upper bound on the prob that  $E+R$  contains any long loops (whether trivial or not).

For any particular closed loop of length  $l$  contained in  $E+R$ ,  $E$  must contain at least  $l/2$  of the edges in  $E$ . Otherwise, we could reduce the weight of  $R$  by choosing  $R$  to contain the edges contained in  $E$ , rather than the complement of that set. Furthermore, there are fewer than  $2^l$  ways of choosing which edges in the loop are contained in  $E$  and which are contained in  $R$ . Therefore, for any specified closed loop  $C$  on the lattice of length  $l$ ,

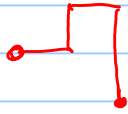


$$\text{Prob}(C \subseteq E+R) \leq 2^l \epsilon^{l/2}$$

To bound the probability of a recovery failure, we sum over all closed loops with length  $l \geq L$

$$\text{Prob of failure} \leq \sum_{l \geq L} (\# \text{ of loops of length } l) 2^l \epsilon^{l/2}$$

A loop of length  $l$  is a special case of a connected path of length  $l$ . A (directed) path can begin at any one of  $L^2$  lattice sites.



The first step of the path can be in any one of 4 directions, and each subsequent step can be in any one of 3 directions.

Therefore,

$$\# \text{ of paths of length } l \leq L^2 4 3^{l-1} = \frac{4}{3} L^2 3^l,$$

and we conclude that

$$\text{Prob of failure} \leq \frac{4}{3} L^2 \sum_{l=L}^{2L^2} 3^l 2^l \epsilon^{l/2}$$

Note that there are no loops with length greater than  $2L^2$  because there are only  $2L^2$  edges on the lattice

$$\Rightarrow \text{Prob of failure} \leq \frac{8}{3} L^4 (36\epsilon)^{L/2} \quad (\text{assuming } 36\epsilon \leq 1)$$

- there are fewer than  $2L^2$  terms in the sum, and each term is no larger than  $(36\epsilon)^{L/2}$ .

$$\text{We have Prob of failure} \leq \text{poly}(L) \left(\epsilon/\epsilon_0\right)^{L/2}$$

where  $\epsilon_0 = 1/36 \hat{=} .0278$ . If  $\epsilon < 2.7\%$ , the prob. of failure becomes exponentially small as the lattice size  $L$  increases.

As with the concatenated codes discussed last time, we find that the distance  $L$  of the toric code is too pessimistic an indicator of the code's performance. It is possible for  $L/2$  errors to cause a recovery failure if the locations of the errors are strategically chosen, but such error patterns are highly atypical. For randomly distributed errors, recovery succeeds with high probability even if the errors occur at a constant rate up to 2.7% (so that the total number of errors in the code block is  $O(L^2)$ , far greater than the code distance). Numerical simulations show that recovery succeeds for errors occurring at an even higher rate, up to about 10.6%.

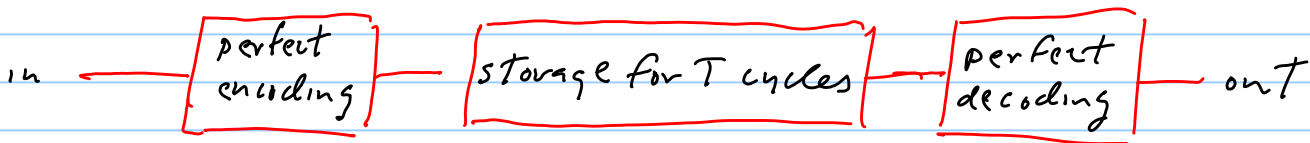
## Fault-tolerant error recovery

Up until now we have discussed protecting quantum information using quantum error-correcting codes, where we have assumed that the recovery procedure can be performed perfectly. Next we want to consider how to use quantum error-correcting codes to protect a quantum computer from noise. There are two key issues that we need to address:

- 1) Aside from just *protecting* quantum information, we will need to *process* it. So we must figure out how to perform nontrivial quantum gates without leaving the code space, and so without losing the protection afforded by the code.
- 2) We will need to do the error recovery and the information processing using the imperfect gates that are achievable in a realistic quantum computer.

We will discuss issue (2) first, and postpone issue (1) for later. Specifically, if we use a stabilizer code to protect quantum information, we will need to measure the stabilizer generators (check operators) of the code. In principle this can be done by executing a quantum circuit (including qubit measurements and preparations). But the gates and measurements themselves will be prone to error (including the "identity gate" -- even qubits that are not being processed may suffer "storage errors"). Will the error recovery work if the recovery procedure itself is noisy?

For now, let's not worry about processing the encoded information; we'll just try to use a QECC to operate a reliable quantum memory. Furthermore, for now, let's suppose that we have an ideal encoder that we can use once to store quantum information in the memory, and a decoder that can perform an idealized error-correction and decoding step once when we are ready to retrieve a quantum state from the memory. But in between we are to protect the information in the memory by repeatedly performing cycles of error recovery using noisy gates. If we store the information for all together  $T$  error correction cycles, with what fidelity will we be able to retrieve the quantum information from the memory at the end?



How should we describe the noise in the gates? Though we could consider more general noise models, let's use this simple model: each gate (or preparation or measurement) in a quantum circuit is either ideal with probability  $1 - \epsilon$ , or faulty with probability  $\epsilon$ . If the gate is faulty, it is replaced by an arbitrary TPCP map. With this model we can hope to use successfully a QECC that corrects  $t$  errors. The rough idea is that if the fault rate  $\epsilon$  is small, the number of errors occurring during a cycle of error correction will only rarely exceed the number that the code can protect against.

Let's try to be more precise. What properties should an error correction "gadget" have to protect a quantum state successfully. Two fundamental properties are needed, which I will call "Property 0" and "Property 1". Before explaining the properties, we introduce some terminology.

We will say that a quantum circuit (possibly including state preparation steps and measurement steps) is " $r$ -good" if the circuit contains no more than  $r$  faults.

We will say that the quantum state of a code block is " $s$ -deviated" from the code space if the state can be obtained by acting on a codeword (a state in the code space) with an error whose weight is at most  $s$ . That is, the "error" can be expanded in terms of Pauli operators of weight up to  $s$ .

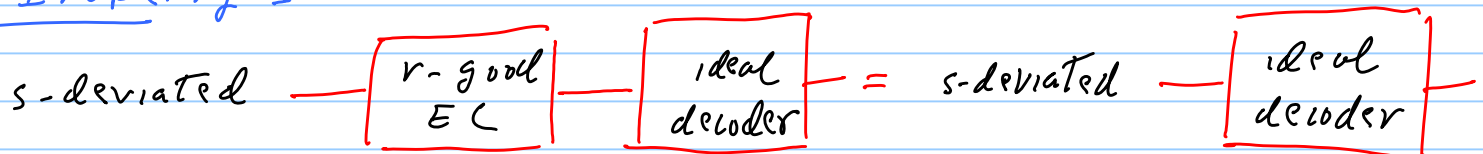
If  $|c\rangle$  is codeword, we say the state of the block is  $s$ -deviated if the state has a purification of the form

$$\sum_a E_a | \psi \rangle \otimes | a \rangle_E \quad \text{where } E_a \text{ is a Pauli op., and}$$

where the states  $\{ | a \rangle_E \}$  of the environment are not necessarily mutually orthogonal or normalized, and  $| a \rangle_E \neq 0$  only for  $\text{wt}(E_a) \leq s$

Then a property we would like our error correction gadgets to have is:

### Property 1

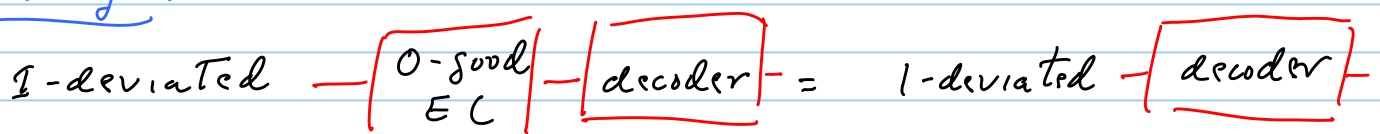


$$\text{where } r + s \leq t$$

If the code corrects  $t$  errors, and the incoming state is  $s$ -deviated from  $| \psi \rangle \in \mathcal{H}_{\text{code}}$  where  $s \leq t$ , then the ideal decoder will recover  $| \psi \rangle$  (and decode it). Property 1 says that the output of the  $r$ -good EC is still correctable to  $| \psi \rangle$ , provided  $r + s \leq t$ . This means that the  $r$  faults introduce no more than  $r$  additional errors in the block.

Consider for example  $t=1$ . Then property 1 encompasses two sub-properties, depending on whether  $r$  or  $s$  is zero.

### Property 1a



This says that the EC gadget with no faults corrects one error successfully. (Actually, it says that when combined with property 0 below, which says output from 0-good EC is a codeword.)

## Property 16

$$0\text{-deviated} \text{ --- } \boxed{\begin{array}{c} 1\text{-good} \\ \text{EC} \end{array}} \text{ --- } \boxed{\text{decoder}} = 0\text{-deviated} \text{ --- } \boxed{\text{decoder}}$$

The fault in the EC introduces at most 1 error in the block.

## Property 0

$$\text{anything} \text{ --- } \boxed{\begin{array}{c} r\text{-good} \\ \text{EC} \end{array}} \text{ --- } r\text{-deviated} \quad \text{where } r \leq t$$

For the case  $t=1$ , the subproperties are

Prop 0a: output from 0-good EC is a codeword

Prop 0b: output from 1-good EC is 1-deviated

(For the "perfect" 5-qubit code, all states are 1-deviated so property 0b is automatic, for any procedure.)

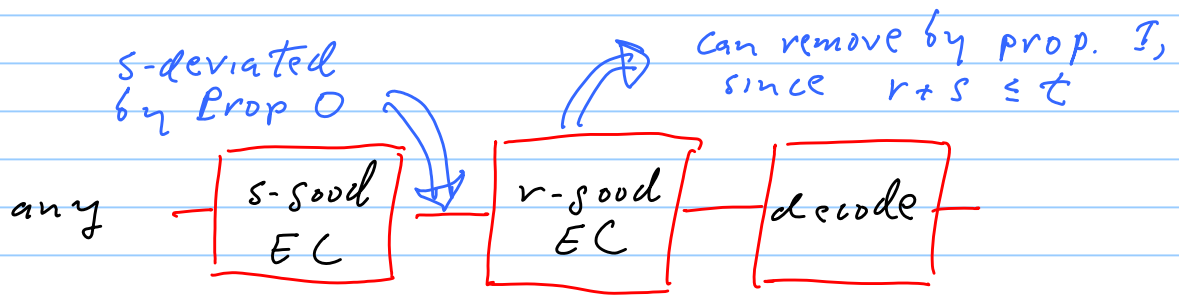
Consider now two consecutive EC gadgets, which we call an "extended EC". The extended EC is " $t$ -good" if the "leading" EC is  $s$ -good and the "trailing" EC is  $r$ -good, where  $r+s \leq t$ .

Claim: A  $t$ -good extended EC obeys

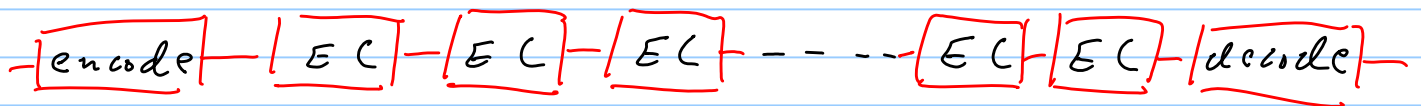
$$\text{anything} \text{ --- } \boxed{\text{EC}_1} \text{ --- } \boxed{\text{EC}_2} \text{ --- } \boxed{\text{decode}} = \text{anything} \text{ --- } \boxed{\text{EC}_1} \text{ --- } \boxed{\text{decode}}$$

(The output from the decoder is unchanged if we remove the second EC.)

The claim follows from properties 0 and 1.



Now -- suppose we consider  $T$  consecutive EC gadgets, in between an ideal encoder and an ideal decoder



Suppose that every extended EC is  $t$ -good. Then we can invoke the claim repeatedly, removing ECs one at a time, starting at the back, until we have the equivalent circuit



that recovers the encoded state with perfect fidelity. Our quantum memory preserves the quantum state perfectly after  $T$  steps, unless at least one extended EC has at least  $t+1$  faults. We can bound the probability of a memory failure:

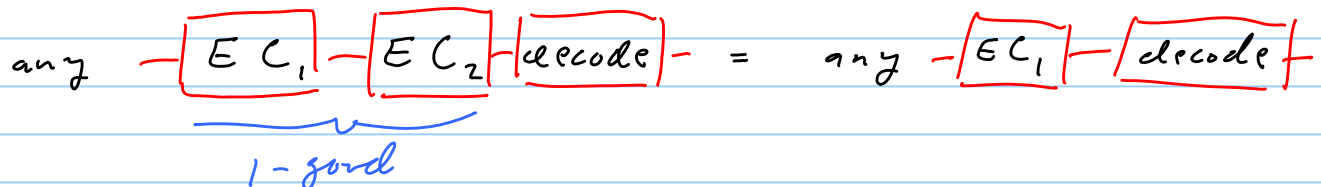
$$\text{Failure Probability} \leq (T-1) \binom{A}{t+1} \epsilon^{t+1}$$

if there are  $A$  gates contained in an extended EC (pair of ECs). For  $\epsilon$  small, we can store quantum information reliably for a time

$$T = O(\epsilon^{-(t+1)})$$

compared to a time  $O(\epsilon^{-1})$  if we did not protect the quantum state using a QECC.

In the  $t=1$  case:



- If  $\text{EC}_1$  has one fault, then its output is 1-deviated, and the following perfect  $\text{EC}_2$  corrects the error.
- If  $\text{EC}_1$  has no faults, its output is a codeword and the one error introduced by  $\text{EC}_2$  is correctable.

The point (in the  $t=1$  case) is that if each extended EC has at most 1 fault, then each time a fault causes an error, the error is corrected before a second fault can cause an encoded error.

Now .. how can we build an error correction gadget that obeys the properties 0 and 1. To be concrete, consider the  $t=1$  case -- a code that corrects one error. In particular, consider the property 1b: we are to ensure that, that if the incoming state is a codeword, then a single fault in the EC will cause only a single (hence correctable) error in the code block.

If we are not careful, this will not be true. A single faulty gate in the EC might have two effects.

- an error in the code block.
- a nontrivial and incorrect syndrome.

Then, guided by the incorrect syndrome, we will flip the wrong qubit in a misguided effort to correct the error; this will introduce a second error in the block, overwhelming the codes error-correction capability.

A general approach to avoid being misled by incorrect syndrome information is to measure the syndrome multiple times. For the  $t=1$  case, it suffices to measure the complete error syndrome twice.

- if the syndrome is trivial (indicates no error) the first time, then we need not repeat the syndrome measurement, nor do we take any action to recover from error.

- if the syndrome is nontrivial, we repeat the syndrome measurement. If we obtain the same error syndrome twice in a row, then we trust the syndrome and apply the indicated recovery step. But if we obtain a different syndrome when we measure a second time, then we do not trust the syndrome and we do not attempt to recover.

Let's check that this procedure really has the required properties, starting with property 1.

First, suppose that the full EC (including the repetition of the syndrome measurement) is 0-good (no faults). Then if the incoming block is 0-deviated, the syndrome will be trivial, and if it is 1-deviated, the block will be projected onto a state with a definite Pauli error that can be correctly inferred from the syndrome. The second syndrome measurement (also with no faults) is guaranteed to give the same result, and recovery succeeds.

Next suppose that the EC is 1-good, and that the incoming state is a codeword. Then if the first syndrome measurement has no faults, the trivial syndrome is measured correctly, and no action is taken to recover. If the first syndrome measurement has a fault, then the syndrome measurement is repeated and the second syndrome measurement has no fault. *If the syndrome measurement procedure has the property that a single fault introduces only a single error in the block*, then the second syndrome measurement will identify this error correctly, and it will be corrected successfully.

Now let's check property 0, in the case where the incoming state is arbitrary. If the EC has no faults, we will obtain a valid syndrome, and we can correct the state to the codespace successfully. But what if the EC has one fault? Again, if the first syndrome measurement has a fault and yields a nontrivial syndrome, then the second one has no fault and yields a valid syndrome. Therefore we can correct to the codespace successfully.

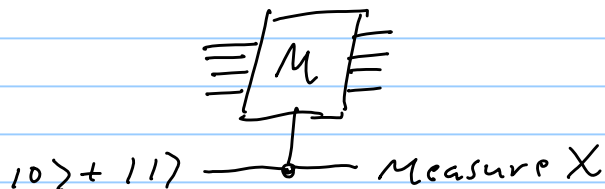
It may be that the first syndrome measurement has a fault, and yields a trivial syndrome that is incorrect. In that case the syndrome measurement will not be repeated. *We need to be sure to design our syndrome measurement procedure so that in this case the actual state of the block is only 1-deviated.*

But suppose the first syndrome measurement has no fault, and yields a (correct) nontrivial syndrome. Then the second syndrome measurement might have a fault. *We need to be sure to design our syndrome measurement procedure so that, although a fault in the second syndrome measurement might cause both an error in the block and an invalid syndrome, that nevertheless after recovery the state of the block is only 1-deviated.*

=====

Now, note that we assumed above, in the case where the incoming state is a codeword, that a single fault in the syndrome measurement results in just one error in the block. But how do we make sure this is true? If we are not careful, even gates in the EC correction that do not have faults might *propagate* errors from one qubit to another.

A general procedure for measuring whether the eigenvalue of operator  $M$  acting on the code block is  $+1$  or  $-1$  is to use the circuit



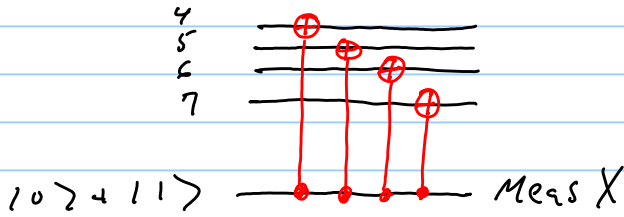
$$\begin{aligned} X=1 &\Rightarrow M = +I \\ X=-1 &\Rightarrow M = -I \end{aligned}$$

If  $M$  is a weight- $m$  Pauli operator, then  $M$  controlled by the ancilla qubit can be expressed as a circuit of 2-qubit gates.

For example, in the case of the  $[[7, 1, 3]]$  code, all stabilizer generators have weight 4

For example  $M_{X3} = I I I X X X X$ , and

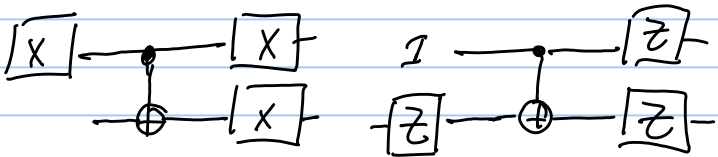
the circuit contains  $\Lambda(X)$  (CNOT) gates acting on qubits 4, 5, 6, 7 in the block



This circuit measures a bit of the syndrome if it has no faults, but it is not fault tolerant.

A single fault in the circuit can cause two errors in the block.

Let's consider how Pauli errors are propagated by CNOT gates. Claim



that is - acting by conjugation, a CNOT maps Pauli operators according to:

$$X I \rightarrow X X$$

$$Z I \rightarrow Z I$$

$$I X \rightarrow X X$$

$$I Z \rightarrow Z Z$$

control → target

That is:  $(X I) \circ (\text{CNOT}) = (\text{CNOT}) \circ (X X)$

$$(I X) \circ (\text{CNOT}) = (\text{CNOT}) \circ (I X)$$

$$(Z I) \circ (\text{CNOT}) = (\text{CNOT}) \circ (Z I)$$

$$(I Z) \circ (\text{CNOT}) = (\text{CNOT}) \circ (I Z)$$

(order of operations from left to right)

To check this, note that if  $UMU^{-1} = N$  (or  $UM = NU$ ) then if  $M|\psi\rangle = \lambda|\psi\rangle$ , then

$$NU|\psi\rangle = UM|\psi\rangle = \lambda U|\psi\rangle. \quad \text{That is, } U$$

maps eigenstate of  $M$  to eigenstate of  $N$  with same eigenvalue.

For the case of a CNOT:

$$(|0\rangle \pm |1\rangle) |\psi\rangle \mapsto |0\rangle |\psi\rangle \pm |1\rangle X|\psi\rangle$$

(eigenstate of  $XI$  mapped to eigenstate of  $XX$  with same eigenvalue.

$$(a|0\rangle + b|1\rangle) |0\rangle \rightarrow a|00\rangle + b|11\rangle$$

$$(a|0\rangle + b|1\rangle) |1\rangle \rightarrow a|01\rangle + b|10\rangle$$

Eigenstate of  $IZ$  mapped to eigenstate of  $ZZ$  with same eigenvalue

Furthermore

$$|0\rangle |\psi\rangle \rightarrow |0\rangle |\psi\rangle$$

$$|1\rangle |\psi\rangle \rightarrow |1\rangle X|\psi\rangle$$

Eigenvalue of  $ZI$  preserved

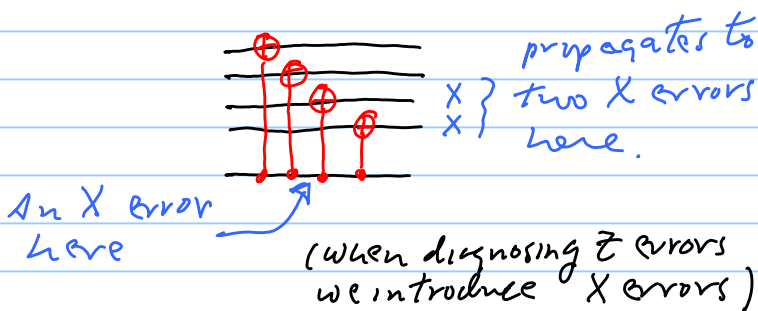
And  $\Delta(X)$  preserves eigenvalue of  $IX$

$$(a|0\rangle + b|1\rangle) |+\rangle \rightarrow (a|0\rangle + b|1\rangle) |+\rangle$$

$$(a|0\rangle + b|1\rangle) |-\rangle \rightarrow (a|0\rangle - b|1\rangle) |-\rangle$$

We say the CNOT propagates  $X$  "forward" (from control to target) and  $Z$  "backward" (from target to control).

The  $[[7, 1, 3]]$  code is a  $d=3$  CSS code - an encoded error can arise from two  $X$  errors or two  $Z$  errors



The same problem arises when we measure the  $Z$ -type check operators. A  $Z$  error introduced by a single fault can propagate to two  $Z$  errors in the block

The flaw in our procedure is that a single "ancilla" qubit interacts with multiple qubits in the code block

To fix the flaw we can either change the code or change the circuit.

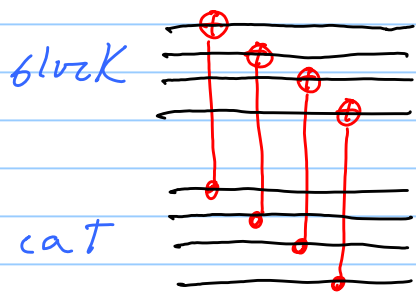
We may define the "spread" of the circuit as the maximum number of qubits in the code block that can be affected by a single fault. For a CSS code that corrects one error, we want the "X spread" and the "Z spread" to both be 1.

We can reduce the spread to 1 if we encode the ancilla. For example, we can use the repetition code, replacing

$$|0\rangle + |1\rangle \longrightarrow |0\rangle + |1\rangle = |0000\rangle + |1111\rangle$$

(a "cat state")

Now each qubit of the ancilla touches just one qubit of the code block



If there are no faults, then

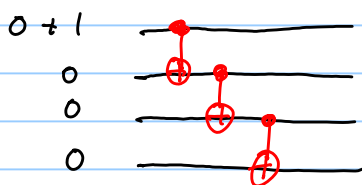
$$|0\rangle + |1\rangle \longrightarrow |0\rangle + M|1\rangle$$

We measure in the basis  $|0\rangle \pm |1\rangle$  by measuring the encoded  $\bar{X} = XXXX$ ,

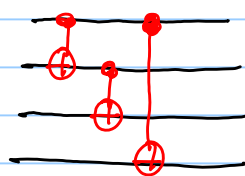
which can be done by measuring each qubit of the ancilla in the X basis, and then computing the parity of the measurement outcomes.

But, we are not done yet with designing a fault-tolerant procedure --- we need to worry about the *preparation* of the ancilla cat state. Z errors in the ancilla state can cause the syndrome to be incorrect, a problem we can address by repeating the syndrome measurement. More serious are X errors in the ancilla, because these can propagate to the code block. Therefore, when we encode the cat state, we don't want to allow a single fault during the encoding circuit to result in two X errors in the state.

A circuit that encodes the cat state is:

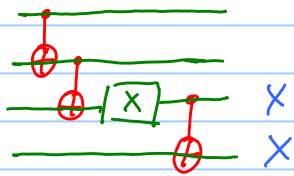


or



(The latter has lower depth, since the last two gates can be done in one step.)

But this circuit is not fault tolerant

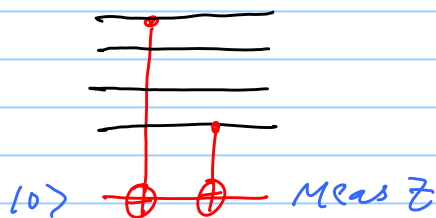


A single fault generates a weight-2 X error so the output state is

$$|0011\rangle + |1100\rangle$$

which results in a wt 2 error if used to measure an X-type check operator.

We should test the cat state before use, by introducing another ancilla qubit:



If the state fails the test, we discard it and try again.

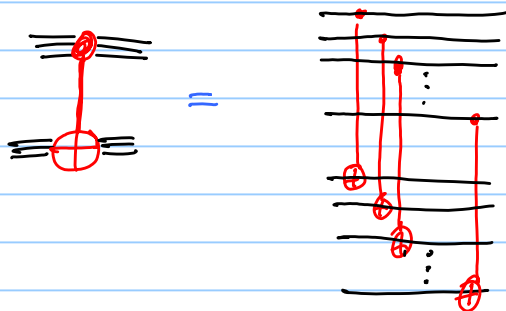
(For the lower depth circuit, we would compare 3rd and 4th qubits instead.)

We now have a complete design for an EC gadget for the  $[[7,1,3]]$  code; it is a 5-step procedure with spread 1, satisfying properties 0 and 1.

- 1) prepared ancilla
- 2) verify ancilla
- 3) measure syndrome
- 4) repeat if necessary
- 5) recover

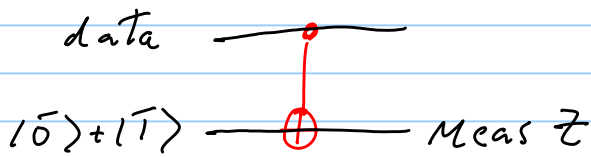
Here is another design of a fault-tolerant EC gadget for the  $[[7,1,3]]$  code. This time, instead of encoding the ancilla using the repetition code, the ancilla is encoded using the same  $[[7,1,3]]$  code that corrects the data.

This gadget uses a general property of CSS codes which we will derive in the next lecture: an encoded CNOT can be executed *transversally*. This means that the circuit for the encoded CNOT is



We perform 7 CNOT gates in parallel to realize the logical CNOT acting on two code blocks

The syndrome meas. procedure for X-type errors is:

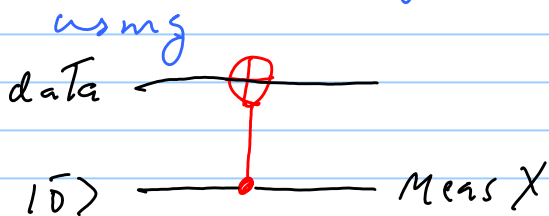


Here the wires are code blocks.

Ideally, the target block is an X eigenstate, so the

CNOT gate acts trivially on the code space. But the CNOT gate propagates X errors in the data to the target ancilla, where the errors can be detected by measuring the Z-type check operators. Furthermore, these Z-type check operators can be measured (destructively) by measuring Z for each of the seven qubits in the ancilla block, and applying the classical Hamming parity checks to the meas. outcomes.

Similarly, we measure syndrome for Z errors



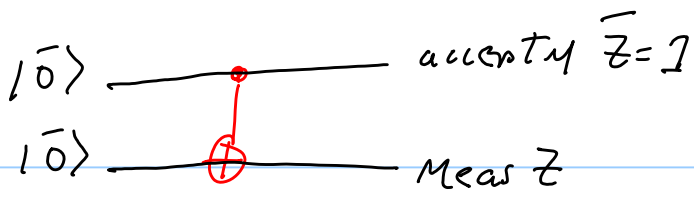
Again, the CNOT gate acts trivially on code space.

Z errors propagate from data to ancilla and are detected

by performing X measurement on ancilla qubits and applying Hamming parity check to outcomes.

This procedure has the advantage that it is highly parallelized --- for both the X and Z syndrome, the interaction between data block and ancilla block occurs in a single time step. A further advantage is that repetition of the syndrome measurement is not necessary. A single faulty CNOT gate might cause an X error in the data and an X error in the ancilla, or a Z error in the data and a Z error in the ancilla. But, the error occurs in the same "position" in both the data block and the ancilla block. For example, if a single fault introduces an error in the first qubit of the data block and the first qubit of the ancilla block, we can't obtain an incorrect syndrome that instructs us to flip another qubit in the data block at a position other than the first position. Rather if the syndrome indicates there is an error, the indicated error will be in the first position, and no weight-two error can occur.

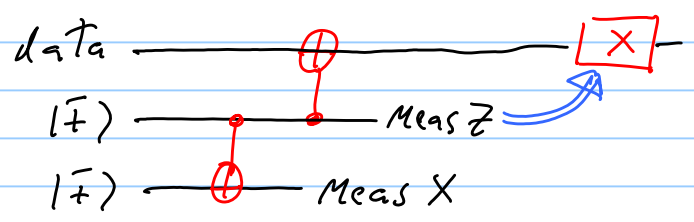
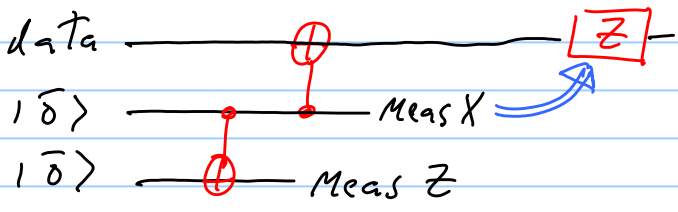
Of course, we need to encode and test the ancilla. A single fault in the encoding circuit might cause a high-weight error, so when we try to prepare the encoded  $|0\rangle$  we get the encoded  $|1\rangle$  instead (or a state one-deviated from the encoded  $|1\rangle$ ). We test the encoded ancilla by using yet another encoded ancilla, and reject the ancilla if it fails the test.



To measure  $\bar{z}$  of the block, we measure  $z$  for each qubit, use Hamming parity checks to correct to the code space, then apply another parity check to extract the value of the encoded  $\bar{z}$  (we get the right answer if block is 1-deviated from a  $\bar{z}$  eigenstate).

If there is only one fault in the preparation and verification of the two ancilla states, then if there is a fault in the preparation of one block (which might have a high-weight error) the other block has no errors. Therefore, if the ancilla being tested has an encoded error, the error will be detected and the ancilla will be rejected. The procedure checks only for X errors, not Z errors, but only the X error can propagate to the data when an accepted ancilla interacts with the data.

The full fault-tolerant procedures for Z and X error correction are:



Two other noteworthy points:

- It is not actually necessary to apply the Z or X recovery operations. An efficient classical computation suffices to propagate the Z and X errors found in each syndrome measurement forward through the circuit, so we can interpret the measurement of a logical Z or logical X at the end of the circuit.
- We can make the syndrome measurement procedure deterministic (i.e. avoid throwing away ancillas that fail the test) by preparing three ancilla blocks and consuming two of them in two tests of the third block. If both tests indicate that the ancilla is an encoded  $|1\rangle$  rather than an encoded  $|0\rangle$ , then we either perform a logical X to correct the ancilla, or we record that the ancilla is actually a  $|1\rangle$  and propagate the logical error through the circuit by an efficient classical computation.