10. Accurate computaton out of noisy components

(NC 10.5-10.6)

Holy grail:

(c) The threshold theorem
(good enough implies arbitrarily good)

10. Accurate computaton out of noisy components

(NC 10.5-10.6)

Holy grail:

(c) The threshold theorem
(good enough implies arbitrarily good)

Elaborating what is good enough:

(b) Principles of fault-tolerant quantum computaton (don't make a mess)

10. Accurate computaton out of noisy components

(NC 10.5-10.6)

Holy grail:

(c) The threshold theorem
(good enough implies arbitrarily good)

Elaborating what is good enough:

(b) Principles of fault-tolerant quantum computaton (don't make a mess)

How to achieve (b)?

(d)-(f) Fault-tolerant logical Pauli & Clifford gates

(g) Fault-tolerant pi/8 gate, 1-bit teleportation

(h) Fault-tolerant measurements

10. Accurate computaton out of noisy components

(NC 10.5-10.6)

Holy grail:

(c) The threshold theorem
(good enough implies arbitrarily good)

Elaborating what is good enough:

(b) Principles of fault-tolerant quantum computaton (don't make a mess)

How to achieve (b)?

(d)-(f) Fault-tolerant logical Pauli & Clifford gates

(g) Fault-tolerant pi/8 gate, 1-bit teleportation

(h) Fault-tolerant measurements

(i) Overhead and assumptions

Fault-tolerant quantum computation

QECC : assume ancilla preparation, encoding, syndrome measurement, and decoding are perfect

Can QECC still work if these operations are imperfect?

Fault-tolerant quantum computation

QECC : assume ancilla preparation, encoding, syndrome measurement, and decoding are perfect

Can QECC still work if these operations are imperfect?

Recall: quantum circuit for solving classical problems:

(1) Prepare $|0\rangle^{\otimes N}$
(2) Apply a universal set of gates to approx any unitary
(3) Measure in $\{|0\rangle, |1\rangle\}$ basis

Fault-tolerant quantum computation

QECC : assume ancilla preparation, encoding, syndrome measurement, and decoding are perfect

Can QECC still work if these operations are imperfect?

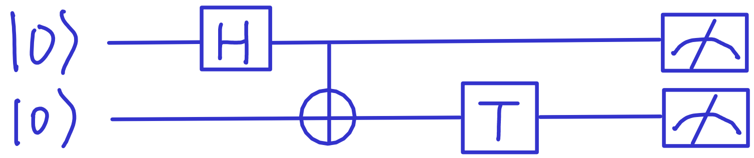
Recall: quantum circuit for solving classical problems:

(1) Prepare $|0\rangle^{\otimes N}$
(2) Apply a universal set of gates to approx any unitary
(3) Measure in $\{|0\rangle, |1\rangle\}$ basis

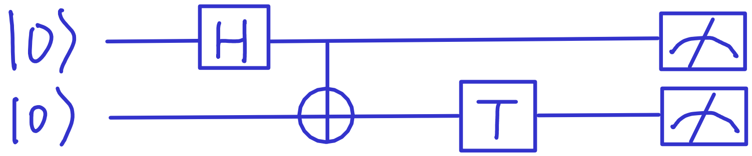
Goal of FTQC:

Given a perfect circuit of the above form, design a new circuit using noisy initial states, gates, & measurement that gives similar final measurement outcomes.

Our approach: concatenation of QECC

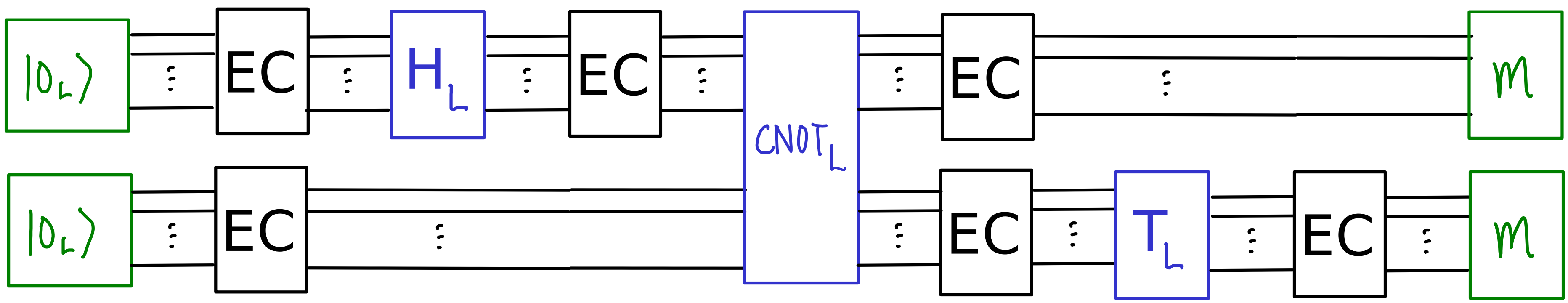


Wanted: circuit C0 (level 0)

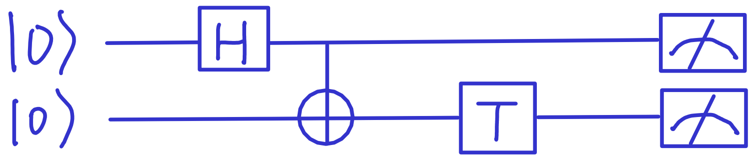


Wanted: circuit C0 (level 0)

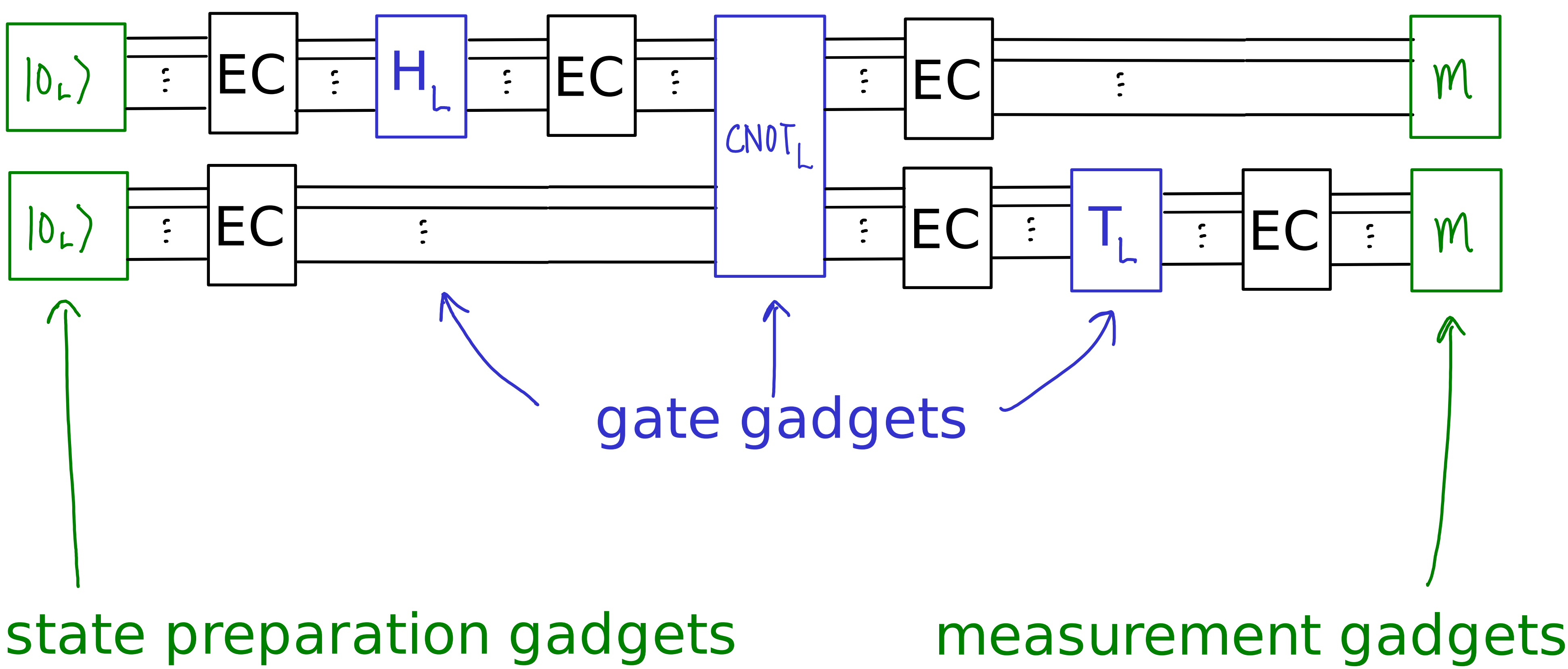
Simulation by
circuit C1 (level 1):

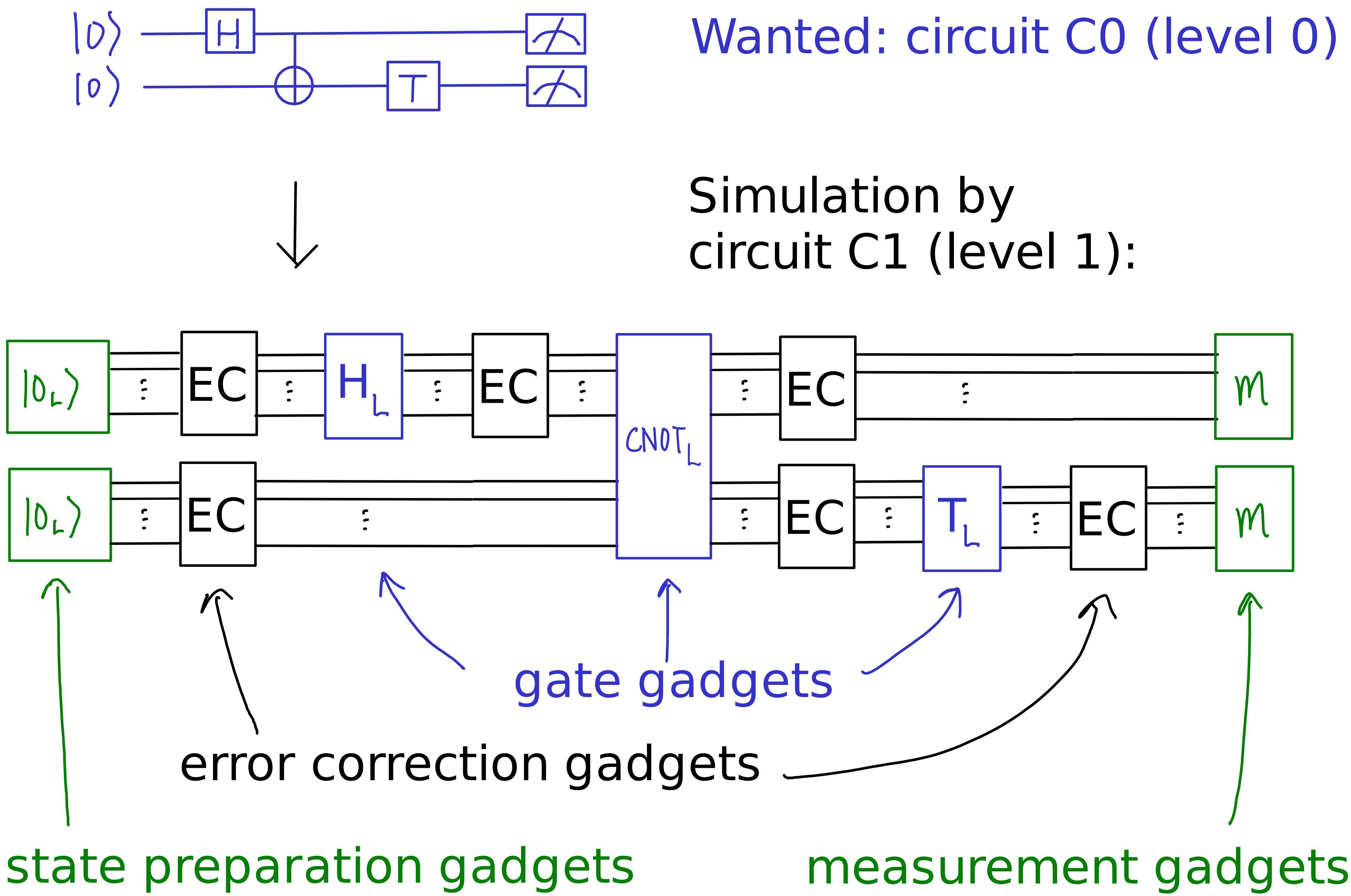


Wanted: circuit C0 (level 0)



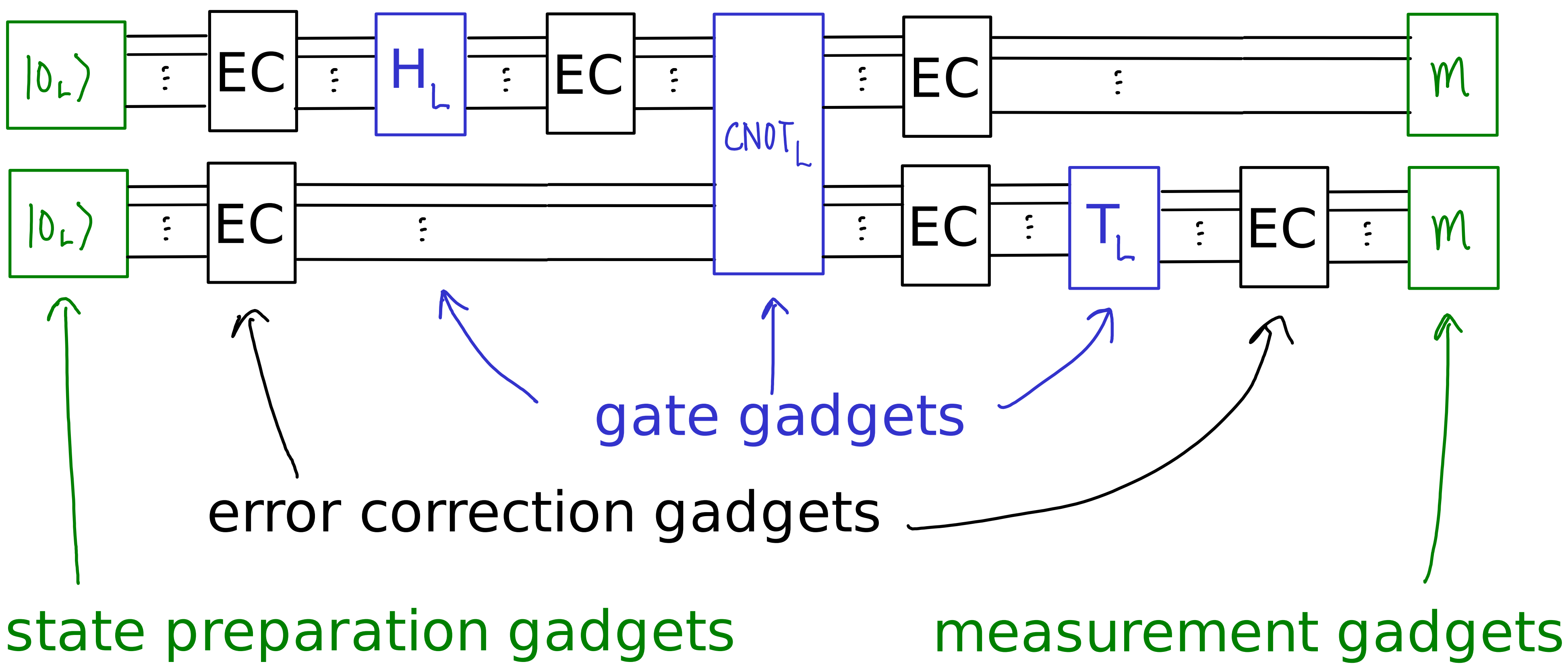
Simulation by circuit C1 (level 1):





Idea: for very low noise, QECC is still useful ...

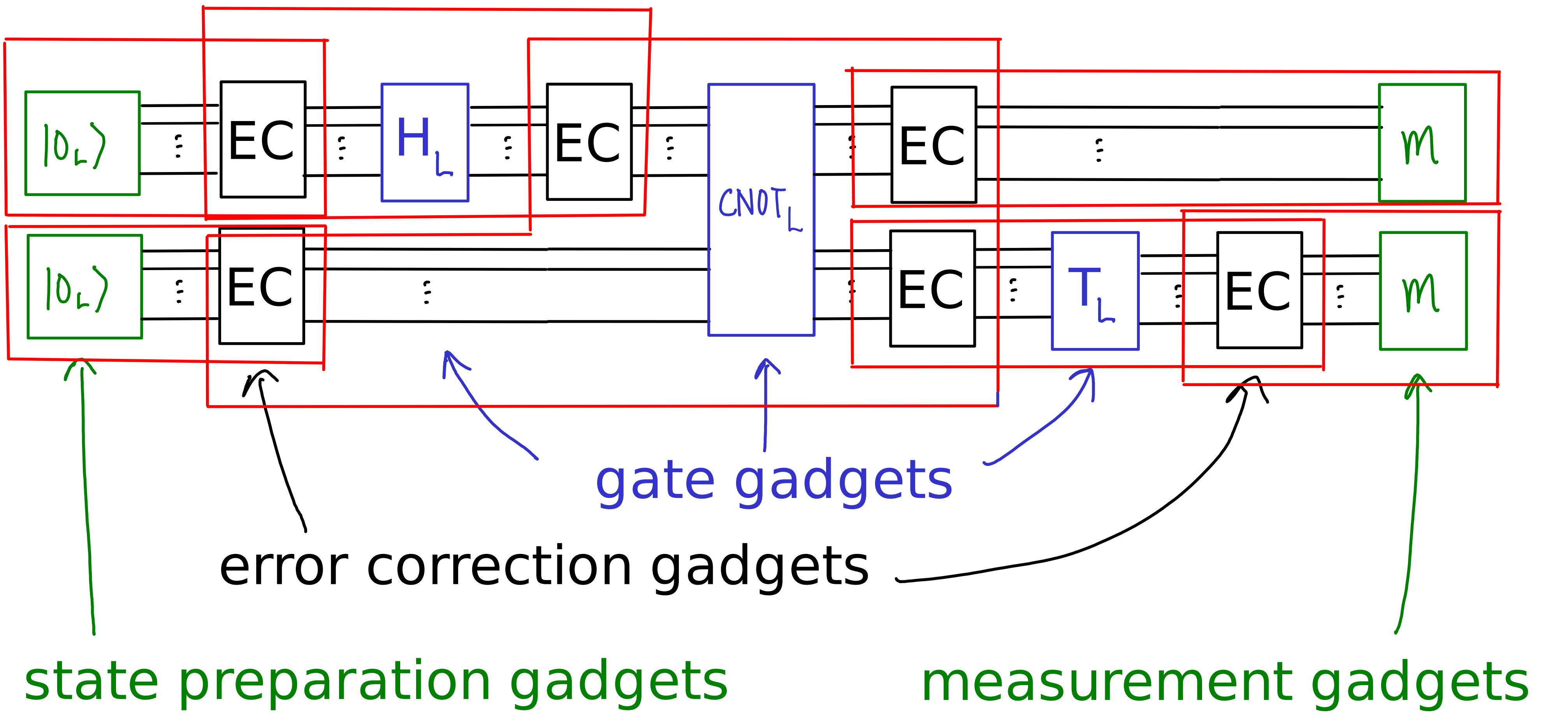
Gadget: circuit of noisy components (state prep, gates, storage, measurements); each fails with some prob.



Simulation by circuit C1 (level 1)

Gadget: circuit of noisy components (state prep, gates, storage, measurements); each fails with some prob.

Goal: each region (rectangle) in C1 simulates a circuit component (state prep, ..., meas) in C0 (level 0) "correctly", unless there are at least 2 faults.



Simulation by circuit C1 (level 1)

Why? physical logical

If each component in C1 (level 1 circuit) fails independently with prob p, then, a region fails to simulate the level 0 component with prob $O(p^2)$.

Why? physical logical

If each component in C1 (level 1 circuit) fails independently with prob p, then, a region fails to simulate the level 0 component with prob $O(p^2)$.

Then, we simulate recursively:

Make a new circuit C2 by replacing each component in C1 with a gadget and with EC gadgets between them.

Why?

physical logical

If each component in C1 (level 1 circuit) fails independently with prob p, then, a region fails to simulate the level 0 component with prob $O(p^2)$.

Then, we simulate recursively:

Make a new circuit C2 by replacing each component in C1 with a gadget and with EC gadgets between them.

Each component in C2 is physical, fails with prob p. Each component of C1 (level-1) is simulated in C2 "correctly" unless 2 or more faults appear in the simulating region. So, simulated C1 (level-1) components is faulty with prob $O(p^2)$.

Why? physical logical

If each component in C1 (level 1 circuit) fails independently with prob p, then, a region fails to simulate the level 0 component with prob $O(p^2)$.

Then, we simulate recursively:

Make a new circuit C2 by replacing each component in C1 with a gadget and with EC gadgets between them.

Each component in C2 is physical, fails with prob p.
Each component of C1 (level-1) is simulated in C2 "correctly" unless 2 or more faults appear in the simulating region. So, simulated C1 (level-1) components is faulty with prob $O(p^2)$.

The simulated C1 in turns simulates C0 (level-0); each C0 component now fails with prob $O((p^2)^2)$.

Concatenation: encode data by QECC, take each register in the QECC and encode again by QECC. (cf 9-bit code)

Concatenation: encode data by QECC, take each register in the QECC and encode again by QECC. (cf 9-bit code)

Will see:

1. High efficiency of error suppression

2. How to design gadgets so that 2 faults are needed to fail a simulation, & why they have constant (circuit) size.

Concatenation: encode data by QECC, take each register in the QECC and encode again by QECC. (cf 9-bit code)

Will see:

1. High efficiency of error suppression

2. How to design gadgets so that 2 faults are needed to fail a simulation, & why they have constant (circuit) size.

The recursive circuit simulation with property 2 above is called a "fault-tolerant" simulation of a circuit.

1. How does the error change with # levels?

2. How does the circuit size change with # levels?

3. How to achieve arbitrarily accurate simulation without arbitrarily accurate physical components?

1. How does the error change with # levels?

Physical error rate is $p_0 = p$

With 1 level, error rate for simulating C0 components

$p_1 \leq \binom{s}{2} p_0^2$ where s = size of the largest region

#ways to have 2 faults

1. How does the error change with # levels?

Physical error rate is $p_0 = p$

With 1 level, error rate for simulating C0 components

$$p_1 \le \binom{s}{2} p_0^2$$

where s = size of the largest region

#ways to have 2 faults

Define the threshold error rate as: $p^* = \binom{s}{2}^{-1}$

$p_0 = p < p^* \Rightarrow p_1 < p_0$

1. How does the error change with # levels?

Physical error rate is $p_0 = p$

With 1 level, error rate for simulating C0 components

$p_1 \leq \binom{s}{2} p_0^2$ where s = size of the largest region

#ways to have 2 faults

Define the threshold error rate as: $p^* = \binom{s}{2}^{-1}$

$p_0 = p < p^* \Rightarrow p_1 < p_0$

Define $r_0 = \frac{p_0}{p^*}$, $r_1 = \frac{p_1}{p^*}$ $\left(p_1 = p^* r_1 \right)$

1. How does the error change with # levels?

Physical error rate is $p_0 = p$

With 1 level, error rate for simulating C0 components

$p_1 \leq \binom{s}{2} p_0^2$ where s = size of the largest region

#ways to have 2 faults

Define the threshold error rate as: $p^* = \binom{s}{2}^{-1}$

$p_0 = p < p^* \Rightarrow p_1 < p_0$

Define $r_0 = \frac{p_0}{p^*}$, $r_1 = \frac{p_1}{p^*}$ $(p_1 = p^* r_1)$

Note: $r_1 = \frac{p_1}{p^*} \leq \frac{\binom{s}{2} p_0^2}{p^*} = \left(\frac{p_0}{p^*}\right)^2 = r_0^2$.

1. How does the error change with # levels?

Physical error rate is $p_0 = p$

With 2 levels, error rate for C2 to simulate C1 components

$$p_1 \leq \binom{s}{2} p_0^2$$

where s = size of the largest region

#ways to have 2 faults

1. How does the error change with # levels?

Physical error rate is $p_0 = p$

With 2 levels, error rate for C2 to simulate C1 components

$$p_1 \leq \binom{s}{2} p_0^2$$

where s = size of the largest region

#ways to have 2 faults

Those C1 components in turns simulate C0 components with error rate

$$p_2 \leq \binom{s}{2} p_1^2 ,$$

1. How does the error change with # levels?

Physical error rate is $p_0 = p$

With 2 levels, error rate for C2 to simulate C1 components

$$p_1 \le \binom{s}{2} p_0^2$$ where s = size of the largest region

#ways to have 2 faults

Those C1 components in turns simulate C0 components with error rate

$$p_2 \le \binom{s}{2} p_1^2 ,$$

$$p_2 = p^* r_2 , \quad r_2 = \frac{p_2}{p^*} \le \frac{\binom{s}{2} p_1^2}{p^*} = \left(\frac{p_1}{p^*}\right)^2 = r_1^2$$

1. How does the error change with # levels?

Physical error rate is $p_0 = p$

With 2 levels, error rate for C2 to simulate C1 components

$$p_1 \leq \binom{s}{2} p_0^2$$ where s = size of the largest region

#ways to have 2 faults

Those C1 components in turns simulate C0 components with error rate

$$p_2 \leq \binom{s}{2} p_1^2 ,$$

$$p_2 = p^* r_2 , \quad r_2 = \frac{p_2}{p^*} \leq \frac{\binom{s}{2} p_1^2}{p^*} = \left(\frac{p_1}{p^*}\right)^2 = r_1^2 \leq (r_0^2)^2$$

1. How does the error change with # levels?

Physical error rate is $p_0 = p$

With k levels: error rate is p_k, $r_k = \frac{p_k}{p^*} \leq r_{k-1}^2$

Solving the recursion for the ratios:

$$r_k \leq r_{k-1}^2 \leq (r_{k-2}^2)^2 \leq r_{k-3}^{2^3} \leq \cdots \quad r_0^{2^k}$$

1. How does the error change with # levels?

Physical error rate is $p_0 = p$

With k levels: error rate is p_k, $r_k = \frac{p_k}{p^*} \leq r_{k-1}^2$

Solving the recursion for the ratios:

$$r_k \leq r_{k-1}^2 \leq (r_{k-2}^2)^2 \leq r_{k-3}^{2^3} \leq \cdots \quad r_0^{2^k}$$

The error decreases doubly exponentially with k!

$$p_k = r_k p^* \leq r_0^{2^k} p^* = \left(\frac{p}{p^*}\right)^{2^k} p^* .$$

2. How big is a level-k circuit?

For an original circuit C0 of size N (width * depth, roughly # components), an overall error of simulating C0 to be $\leq \varepsilon$, set component-wise error rate $< \varepsilon/N$.

How many levels are needed?

2. How big is a level-k circuit?

For an original circuit C0 of size N (width * depth, roughly # components), an overall error of simulating C0 to be $\leq \varepsilon$, set component-wise error rate $< \varepsilon/N$.

How many levels are needed? Want $r_k p^* = p_k \leq \varepsilon/N$

Suffices for: $r_k p^* \leq r_0^{2^k} p^* \leq \varepsilon/N$

or: $r_0^{2^k} \approx \varepsilon/N$

2. How big is a level-k circuit?

For an original circuit C0 of size N (width * depth, roughly # components), an overall error of simulating C0 to be $\leq \varepsilon$, set component-wise error rate $< \varepsilon/N$.

How many levels are needed? Want $r_k p^* = p_k \leq \varepsilon/N$

Suffices for: $r_k p^* \leq r_0^{2^k} p^* \leq \varepsilon/N$

or: $r_0^{2^k} \approx \varepsilon/N$

$$2^k \log r_0 \approx \log \varepsilon/N$$

log

$$2^k \approx \frac{\log N/\varepsilon}{\log 1/r_0}$$

log

$$k \approx \log\left(\frac{\log N/\varepsilon}{\log 1/r_0}\right)$$

Size of level k circuit

$\leq N S^k$ (singly exponential in k)

Size of level k circuit

$\leq N S^k$ (singly exponential in k)

$$\approx N \left(\frac{\log N/\varepsilon}{\log 1/r_0} \right)^{\log S} \qquad \left(\text{from } 2^k \approx \frac{\log N/\varepsilon}{\log 1/r_0} \right)$$

Size of level k circuit

$\leq N S^k$ (singly exponential in k)

$$\approx N \left(\frac{\log N/\varepsilon}{\log 1/r_0} \right)^{\log S} \qquad \left(\text{from } 2^k \approx \frac{\log N/\varepsilon}{\log 1/r_0} \right)$$

$$\text{Overhead} := \frac{\text{size } C_k}{\text{size } C_0} = \left(\frac{\log N/\varepsilon}{\log 1/r_0} \right)^{\log S}$$

$$= \text{polylog}\left(N, 1/\varepsilon, 1/r_0 \right). \qquad (\text{deg} = \log S)$$

Size of level k circuit

$$\lesssim N S^{k} \quad \text{(singly exponential in k)}$$

$$\approx N \left(\frac{\log N/\varepsilon}{\log 1/r_0} \right)^{\log S} \qquad \left(\text{from } 2^k \approx \frac{\log N/\varepsilon}{\log 1/r_0} \right)$$

$$\text{Overhead} := \frac{\text{size } C_k}{\text{size } C_0} = \left(\frac{\log N/\varepsilon}{\log 1/r_0} \right)^{\log S}$$

$$= \text{polylog}\left(N, 1/\varepsilon, 1/r_0 \right). \qquad (\deg = \log S)$$

Good news (complexity): polynomial speed up by QC (such as Grover's algorithm) is roughly preserved.

Not so good news (practical): overhead is large.

For example: encoding 1 qubit in 7,

- N = 10000 (small circuit)
- e = 1% (reasonable accuracy)
- S = size of largest region, dominated by EC gadget

Not so good news (practical): overhead is large.

For example: encoding 1 qubit in 7,

N = 10000 (small circuit)

e = 1% (reasonable accuracy)

S = size of largest region, dominated by EC gadget

Each syndrome requires measuring 6 generators, repeating 3 times, each generator is weight 3, measurement circuit size roughly 6. A region simulating CNOT has 4 EC gadets, so,

Not so good news (practical): overhead is large.

For example: encoding 1 qubit in 7,

N = 10000 (small circuit)

e = 1% (reasonable accuracy)

S = size of largest region, dominated by EC gadget

Each syndrome requires measuring 6 generators, repeating 3 times, each generator is weight 3, measurement circuit size roughly 6. A region simulating CNOT has 4 EC gadets, so,

$$S \approx 4 \times 3 \times 36 \approx 432$$

$$p^* \approx \binom{S}{2}^{-1} \approx (93096)^{-1} \approx 10^{-5}.$$

Not so good news (practical): overhead is large.

For example: encoding 1 qubit in 7,
  N = 10000 (small circuit)
  e = 1% (reasonable accuracy)
  S = size of largest region, dominated by EC gadget

Each syndrome requires measuring 6 generators, repeating 3 times, each generator is weight 3, measurement circuit size roughly 6. A region simulating CNOT has 4 EC gadets, so,

$$S \approx 4 \times 3 \times 36 \approx 432$$ actually larger

$$p^* \approx \binom{S}{2}^{-1} \approx (93096)^{-1} \approx 10^{-5}.$$

Overhead: $$\left(\frac{\log N/\varepsilon}{\log 1/r_0}\right)^{\log S} \approx \left(\frac{20}{\log 1/r_0}\right)^{8.8} \approx \begin{cases} 3 \times 10^{11} & \text{if } r_0 = \frac{1}{2} \\ 7 \times 10^{6} & \text{if } r_0 = 1/10 \end{cases}$$

Our estimate is crude, better analysis allows fewer repetition etc, s can be about 100, best $p^* \sim 10^{-3}$ (need more assumptions).

Our estimate is crude, better analysis allows fewer repetition etc, s can be about 100, best p* ~ 10^{-3} (need more assumptions).

Meanwhile, a not so gentle reminder:
Present expts: 20-160 qubits, error: 0.1-5%
(These numbers did not even exist around year 2000, when the threshold result was proved, progress!!)

We should be optimistic, but only very cautiously.

2019 writting
Use other screenshots for this part of the discussion.
The qubit count and error have not changed too much!

The threshold theorem: if p < p*, then, we can simulate any circuit (C0) of size N with arbitrarily small error ε by a circuit (Ck) of size $N \left(\frac{\log N/\varepsilon}{\log 1/r_0} \right)^{\log S}$,

where S = size of largest gadget, r0 = p/p*, p* = $\binom{S}{2}^{-1}$.

The threshold theorem: if p < p*, then, we can simulate any circuit (C0) of size N with arbitrarily small error ε by a circuit (Ck) of size $N\left(\frac{\log N/\varepsilon}{\log 1/r_0}\right)^{\log S}$,

where S = size of largest gadget, r0 = p/p*, p* = $\binom{S}{2}^{-1}$.

Assumptions:

- each component err independently
  (relaxed by Aliferis, Gottesman, Preskill, Bombin etc)
- p does not change with system size
- parallelism scales with system size
- perfect classical computation (for EC)
- the gadgets exist

The threshold theorem: if p < p*, then, we can simulate any circuit (C0) of size N with arbitrarily small error ε by a circuit (Ck) of size $N\left(\frac{\log N/\varepsilon}{\log 1/r_0}\right)^{\log S}$,

where S = size of largest gadget, r0 = p/p*, p* = $\binom{S}{2}^{-1}$.

Assumptions:

- each component err independently
  (relaxed by Aliferis, Gottesman, Preskill, Bombin etc)
- p does not change with system size
- parallelism scales with system size
- perfect classical computation (for EC)
- the gadgets exist

Goal: design these gadgets, with S as small as possible.

W25: start topic 10 here ... return to first half of this set of slides later.

Fault-tolerant quantum computation

QECC : assume ancilla preparation, encoding, syndrome measurement, and decoding are perfect

Can QECC still work if these operations are imperfect?

To quantum-compute more & more reliably, do we need higher & higher precision in our elementary operations?

Fault-tolerant quantum computation

QECC : assume ancilla preparation, encoding, syndrome measurement, and decoding are perfect

Can QECC still work if these operations are imperfect?

To quantum-compute more & more reliably, do we need higher & higher precision in our elementary operations?

The surprise -- under moderate assumptions on the noise and on the architecture, 1-in-million to 1% noise in elementary operations is good enough!

Fault-tolerant quantum computation

QECC : assume ancilla preparation, encoding, syndrome measurement, and decoding are perfect

Can QECC still work if these operations are imperfect?

To quantum-compute more & more reliably, do we need higher & higher precision in our elementary operations?

The surprise -- under moderate assumptions on the noise and on the architecture, 1-in-million to 1% noise in elementary operations is good enough!

Intuition: if noise is low enough, QECC "kind of work" so, we take additional care to not spread errors ...

10. Accurate computaton out of noisy components

(NC 10.5-10.6)

Holy grail:

(c) The threshold theorem
(good enough implies arbitrarily good)

Elaborating what is good enough:

(b) Principles of fault-tolerant quantum computaton (don't make a mess)

How to achieve (b)?

(d)-(f) Fault-tolerant logical Pauli & Clifford gates

(g) Fault-tolerant pi/8 gate, 1-bit teleportation

(h) Fault-tolerant measurements

(i) Overhead and assumptions

Principles for Fault Tolerant QC:

(1) Use a discrete universal set of gates at each level

A slight deviation from an intended gate should lead to an illegitimate gate if we were to detect the deviation.

e.g., if the smallest angle of rotation is pi/8, a rotation of pi/8 + pi/200 is "erroneous".

Principles for Fault Tolerant QC:

(1) Use a discrete universal set of gates at each level

A slight deviation from an intended gate should lead to an illegitimate gate if we were to detect the deviation.

e.g., if the smallest angle of rotation is pi/8, a rotation of pi/8 + pi/200 is "erroneous".

We use the universal set of gates {H,T,CNOT}.

We add $R = T^2$ to the set for reasons that'll be clear later.

Principles for Fault Tolerant QC:

(2) Never leave quantum data unprotected.

Cannot correct error on unencoded quantum data. Also, an error during encoding of an unknown quantum state into a QECC can introduce a logical error that can never be caught.

Principles for Fault Tolerant QC:

(2) Never leave quantum data unprotected.

Cannot correct error on unencoded quantum data. Also, an error during encoding of an unknown quantum state into a QECC can introduce a logical error that can never be caught.

Solution:

known!!

Start the circuit with $|0_L\rangle^{\otimes N}$, apply a universal set of encoded gates, measure $|0_L\rangle$, $|1_L\rangle$ directly.

Principles for Fault Tolerant QC:

(2) Never leave quantum data unprotected.

Cannot correct error on unencoded quantum data. Also, an error during encoding of an unknown quantum state into a QECC can introduce a logical error that can never be caught.

Solution:

known!!

Start the circuit with $|0_L\rangle^{\otimes N}$, apply a universal set of encoded gates, measure $|0_L\rangle, |1_L\rangle$ directly.

NB: threshold theorem only considers circuits with no unknown quantum input / output. The simulation takes the descriptipn of a quantum circuit C0, and outputs measurement outcomes. Size of C0 is arbitrary, but it has to be known before the simulation.)

Principles for Fault Tolerant QC:

(3) Encoded operations should not spread errors.

(4) EC between gadgets for state preparation and and encoded operation.

(5) Syndrome measurements themselves should not spread errors.

We will elaborate on (3)-(5) the rest of this segment of the course.

Encoded operations for stabilizer codes:

Recall that a logical operation of a stabilizer code is a unitary U that permutes the stabilizer group elements by conjugaton.

Equivalently, U conjugates every generator into a product of the generators.

Encoded operations for stabilizer codes:

Recall that a logical operation of a stabilizer code is a unitary U that permutes the stabilizer group elements by conjugaton.

Equivalently, U conjugates every generator into a product of the generators.

Prescription: given a list of generators, look for Pauli matrices that commute with all the generators, and pick an anticommuting pair to be the logical X, Z.

Encoded operations for stabilizer codes:

Recall that a logical operation of a stabilizer code is a unitary U that permutes the stabilizer group elements by conjugaton.

Equivalently, U conjugates every generator into a product of the generators.

Prescription: given a list of generators, look for Pauli matrices that commute with all the generators, and pick an anticommuting pair to be the logical X, Z.

How do we find logical H or CNOT ?

Note that the logical X, Z should be chosen so that the logical H and CNOT are easy to implement.

Before discussing encoded operations, we first see characterizations of H and CNOT (unencoded).

Lemma: If a unitary U on 2-dim satisfies the conditions

$$U X U^\dagger = Z \,, \qquad U Z U^\dagger = X$$

then, U = H (the Hadamard gate).

Lemma: If a unitary U on 2-dim satisfies the conditions

$$UXU^\dagger = Z, \quad UZU^\dagger = X$$

then, U = H (the Hadamard gate).

If a unitary U on 4-dim satisfies the conditions

$$U XI U^\dagger = XX, \quad U ZI U^\dagger = ZI$$

$$U IX U^\dagger = IX, \quad U IZ U^\dagger = ZZ$$

(fill in omitted product signs)

then, U = CNOT.

Lemma: If a unitary U on 2-dim satisfies the conditions

$$UXU^\dagger = Z\,, \qquad UZU^\dagger = X$$

then, U = H (the Hadamard gate).

If a unitary U on 4-dim satisfies the conditions

$$UXIU^\dagger = XX\,, \qquad UZIU^\dagger = ZI$$
$$UIXU^\dagger = IX\,, \qquad UIZU^\dagger = ZZ$$

(fill in omitted product signs)

then, U = CNOT.

Both results are up to an overall phase of U.

Lemma: If a unitary U on 2-dim satisfies the conditions

$$U X U^\dagger = Z\,, \qquad U Z U^\dagger = X$$

then, U = H (the Hadamard gate).

If a unitary U on 4-dim satisfies the conditions

$$U XI U^\dagger = XX\,, \qquad U ZI U^\dagger = ZI$$
$$U IX U^\dagger = IX\,, \qquad U IZ U^\dagger = ZZ$$

(fill in omitted product signs)

then, U = CNOT.

Both results are up to an overall phase of U.

Lemma says "the commutation relations specify U".

The converse holds, that H and CNOT conjugate the Pauli matrices as described above. (Proof: A2 / Ex)

Lemma: If a unitary U on 2-dim satisfies the conditions

$$U X U^\dagger = Z \,, \quad U Z U^\dagger = X$$

then, U = H (the Hadamard gate).

If a unitary U on 4-dim satisfies the conditions

$$U XI U^\dagger = XX \,, \quad U ZI U^\dagger = ZI$$
$$U IX U^\dagger = IX \,, \quad U IZ U^\dagger = ZZ$$

(fill in omitted product signs)

then, U = CNOT.

Both results are up to an overall phase of U.

Lemma says "the commutation relations specify U".

The converse holds, that H and CNOT conjugate the Pauli matrices as described above. (Proof: A2 / Ex)

These identities also states how H and CNOT propagate Pauli errors (more later).

Proof of lemma:

Suppose $UXU^\dagger = Z$, $UZU^\dagger = X$.

We can find $U|0\rangle$, $U|1\rangle$ up to an overall phase each:

Proof of lemma:

Suppose $UXU^\dagger = Z$, $UZU^\dagger = X$.

We can find $U|0\rangle$, $U|1\rangle$ up to an overall phase each:

$$\left.\begin{aligned} Z &= \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} = |0\rangle\langle 0| - |1\rangle\langle 1| \\ I &= \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = |0\rangle\langle 0| + |1\rangle\langle 1| \end{aligned}\right\} \therefore \begin{aligned} |0\rangle\langle 0| &= \tfrac{1}{2}(I+Z) \\ |1\rangle\langle 1| &= \tfrac{1}{2}(I-Z) \end{aligned}$$

Similarly, $|+\rangle\langle +| = \frac{1}{2}(I+X)$, $|-\rangle\langle -| = \frac{1}{2}(I-X)$.

Proof of lemma:

Suppose $UXU^\dagger = Z$, $UZU^\dagger = X$.

We can find $U|0\rangle$, $U|1\rangle$ up to an overall phase each:

$$\left.\begin{array}{l} Z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} = |0\rangle\langle 0| - |1\rangle\langle 1| \\ \\ I = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = |0\rangle\langle 0| + |1\rangle\langle 1| \end{array}\right\} \therefore \begin{array}{l} |0\rangle\langle 0| = \frac{1}{2}(I+Z) \\ |1\rangle\langle 1| = \frac{1}{2}(I-Z) \end{array}$$

Similarly, $|+\rangle\langle +| = \frac{1}{2}(I+X)$, $|-\rangle\langle -| = \frac{1}{2}(I-X)$.

$$U|0\rangle\langle 0|U^\dagger = U\,\tfrac{1}{2}(I+Z)\,U^\dagger$$

$$= \tfrac{1}{2}(I+UZU^\dagger) = \tfrac{1}{2}(I+X) = |+\rangle\langle +|,$$

where hypothesis is used

Proof of lemma:

Suppose $UXU^\dagger = Z$, $UZU^\dagger = X$.

We can find $U|0\rangle$, $U|1\rangle$ up to an overall phase each:

$$\left.\begin{aligned} Z &= \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} = |0\rangle\langle 0| - |1\rangle\langle 1| \\ I &= \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = |0\rangle\langle 0| + |1\rangle\langle 1| \end{aligned}\right\} \therefore \begin{aligned} |0\rangle\langle 0| &= \tfrac{1}{2}(I+Z) \\ |1\rangle\langle 1| &= \tfrac{1}{2}(I-Z) \end{aligned}$$

Similarly, $|+\rangle\langle +| = \frac{1}{2}(I+X)$, $|-\rangle\langle -| = \frac{1}{2}(I-X)$.

$$U|0\rangle\langle 0|U^\dagger = U\tfrac{1}{2}(I+Z)U^\dagger$$
$$= \tfrac{1}{2}(I+UZU^\dagger) = \tfrac{1}{2}(I+X) = |+\rangle\langle +|,$$

$$U|1\rangle\langle 1|U^\dagger = U\tfrac{1}{2}(I-Z)U^\dagger$$
$$= \tfrac{1}{2}(I-UZU^\dagger) = \tfrac{1}{2}(I-X) = |-\rangle\langle -|.$$

Proof of lemma:

Suppose $UXU^\dagger = Z$, $UZU^\dagger = X$.

We can find $U|0\rangle$, $U|1\rangle$ up to an overall phase each:

$$\left.\begin{aligned} Z &= \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} = |0\rangle\langle 0| - |1\rangle\langle 1| \\ I &= \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = |0\rangle\langle 0| + |1\rangle\langle 1| \end{aligned}\right\} \quad \therefore \begin{aligned} |0\rangle\langle 0| &= \tfrac{1}{2}(I+Z) \\ |1\rangle\langle 1| &= \tfrac{1}{2}(I-Z) \end{aligned}$$

Similarly, $|+\rangle\langle +| = \frac{1}{2}(I+X)$, $|-\rangle\langle -| = \frac{1}{2}(I-X)$.

$$U|0\rangle\langle 0|U^\dagger = U\,\tfrac{1}{2}(I+Z)\,U^\dagger$$
$$= \tfrac{1}{2}(I+UZU^\dagger) = \tfrac{1}{2}(I+X) = |+\rangle\langle +|,$$

$$U|1\rangle\langle 1|U^\dagger = U\,\tfrac{1}{2}(I-Z)\,U^\dagger$$
$$= \tfrac{1}{2}(I-UZU^\dagger) = \tfrac{1}{2}(I-X) = |-\rangle\langle -|.$$

$$\therefore\ U|0\rangle = a|+\rangle,\quad U|1\rangle = b|-\rangle,\qquad |a| = |b| = 1.$$

To relate a and b, use 2nd commutation relation:

$$U |+\rangle\langle +| U^\dagger = U \tfrac{1}{2}(I + X) U^\dagger = \tfrac{1}{2}(I + Z) = |0\rangle\langle 0|$$

$$\therefore\ U|+\rangle \propto |0\rangle$$

To relate a and b, use 2nd commutation relation:

$$U\,|+\rangle\langle+|\,U^\dagger = U\,\tfrac{1}{2}(I+X)\,U^\dagger = \tfrac{1}{2}(I+Z) = |0\rangle\langle 0|$$

$$\therefore\ U|+\rangle \propto |0\rangle$$

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

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

To relate a and b, use 2nd commutation relation:

$$U|+\rangle\langle+|U^\dagger = U\tfrac{1}{2}(I+X)U^\dagger = \tfrac{1}{2}(I+Z) = |0\rangle\langle 0|$$

$$\therefore U|+\rangle \propto |0\rangle$$

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

$$\therefore U|+\rangle = \frac{1}{\sqrt{2}}(U|0\rangle + U|1\rangle)$$

$$= \underbrace{\frac{1}{\sqrt{2}}(a|+\rangle + b|-\rangle)}_{\propto |0\rangle \text{ iff } a=b.}$$ from earlier

To relate a and b, use 2nd commutation relation:

$$U|+\rangle\langle+|U^\dagger = U\tfrac{1}{2}(I+X)U^\dagger = \tfrac{1}{2}(I+Z) = |0\rangle\langle 0|$$

$$\therefore U|+\rangle \propto |0\rangle$$

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

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

$$= \underbrace{\frac{1}{\sqrt{2}}\left(a|+\rangle + b|-\rangle\right)}_{\propto |0\rangle \text{ iff } a=b.}$$ from earlier

$$\therefore U|0\rangle = a|+\rangle, \quad U|1\rangle = a|-\rangle$$

$\therefore U = H$ up to an overall phase a.

The proof is similar for U = CNOT, left as exercise with answer below.

Given: $U XI U^\dagger = XX, \quad U ZI U^\dagger = ZI$

$$U IX U^\dagger = IX, \quad U IZ U^\dagger = ZZ$$

$$\begin{aligned}
\text{So } U|00\rangle\langle 00|U^\dagger &= \frac{1}{4} U(I+Z)\otimes(I+Z)U^\dagger \\
&= \frac{1}{4} U(II + ZI + IZ + ZZ)U^\dagger \\
&= \frac{1}{4}(II + ZI + ZZ + IZ) = |00\rangle\langle 00|
\end{aligned}$$

$$\begin{aligned}
U|01\rangle\langle 01|U^\dagger &= \frac{1}{4} U(I+Z)\otimes(I-Z)U^\dagger \\
&= \frac{1}{4} U(II + ZI - IZ - ZZ)U^\dagger \\
&= \frac{1}{4}(II + ZI - ZZ - IZ) = |01\rangle\langle 01|
\end{aligned}$$

$$U|10\rangle\langle 10|U^\dagger = \frac{1}{4} U(I-Z)\otimes(I+Z)U^\dagger$$

$$= \frac{1}{4} U(II - ZI + (IZ - ZZ))U^\dagger$$

$$= \frac{1}{4}(II - ZI + ZZ - IZ)$$

$$= \frac{1}{4}(I-Z)\otimes(I-Z) = |11\rangle\langle 11|$$

$$U|11\rangle\langle 11|U^\dagger = \frac{1}{4} U(I-Z)\otimes(I-Z)U^\dagger$$

$$= \frac{1}{4} U(II - ZI - (IZ + ZZ))U^\dagger$$

$$= \frac{1}{4}(II - ZI - ZZ + IZ)$$

$$= \frac{1}{4}(I-Z)\otimes(I+Z) = |1\rangle\langle 1| \otimes |0\rangle\langle 0| = |10\rangle\langle 10|$$

So $U|00\rangle = a|00\rangle$, $\quad U|01\rangle = b|01\rangle$

$U|10\rangle = c|11\rangle$, $\quad U|11\rangle = d|10\rangle$

We now show that a,b,c,d are equal. Of many correct methods, an easy way connects 2 variable for each constraint. e.g.,

From the commutation relations on ZI, IZ:

$$U|+\rangle|0\rangle = (a|00\rangle + c|11\rangle)/\sqrt{2}$$

Meanwhile:

$$\begin{aligned} U|+\rangle|0\rangle\langle +|\langle 0|U^\dagger &= U\, |+\rangle\langle +| \otimes |0\rangle\langle 0|\, U^\dagger \\ &= \tfrac{1}{4} U\,(I+X)(I+Z)\,U^\dagger \\ &= \tfrac{1}{4} U\,(II + XI + IZ + XZ)\,U^\dagger \\ &= \tfrac{1}{4}\,(II + XX + ZZ - YY) \\ &= \tfrac{1}{\sqrt{2}}(|00\rangle + |11\rangle)\,\tfrac{1}{\sqrt{2}}(\langle 00| + \langle 11|) \end{aligned}$$

$$\begin{aligned} & UXZU^\dagger \\ &= U(XI)(IZ)U^\dagger \\ &= UXIU^\dagger \cdot UIZU^\dagger \\ &= XX \cdot ZZ \\ &= -YY \quad (Y = iXZ) \end{aligned}$$

So, a = c.

$$U|0\rangle|+\rangle = (a|00\rangle + b|01\rangle)/\sqrt{2}$$

Meanwhile:

$$U|0\rangle|+\rangle\langle 0|\langle +|U^\dagger = U\,(|0\rangle\langle 0| \otimes |+\rangle\langle +|)\,U^\dagger$$

$$= \frac{1}{4} U\,(I+Z)(I+X)\,U^\dagger$$

$$= \frac{1}{4} U\,(II + ZI + IX + ZX)\,U^\dagger$$

$$= \frac{1}{4}\,(II + ZI + IX + ZX) = |0\rangle|+\rangle\langle 0|\langle +|$$

So, a = b.

With a=b=c, superpose all computational basis input:

$$U|+\rangle|+\rangle = U\frac{1}{2}(|00\rangle+|01\rangle+|10\rangle+|11\rangle) = \frac{1}{2}(a|00\rangle + b|01\rangle + c|11\rangle + d|10\rangle)$$

$$= \frac{1}{2}(a|00\rangle + a|01\rangle + a|11\rangle + d|10\rangle)$$

$$U|++\rangle\langle ++|U^\dagger = \frac{1}{4}U(I+X)(I+X)U^\dagger = \frac{1}{4}U\,(II + IX + XI + XX)\,U^\dagger$$

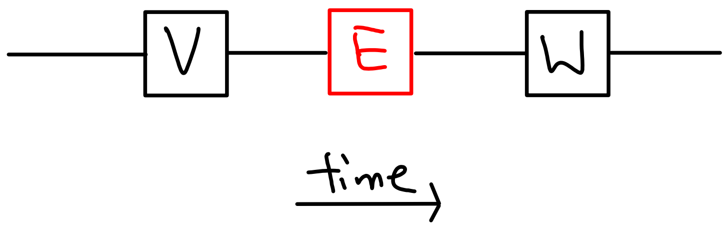
$$= \frac{1}{4}\,(II + IX + XX + XI) = |++\rangle\langle ++|$$

So, a = d. So, U = a * CNOT.

$\square$

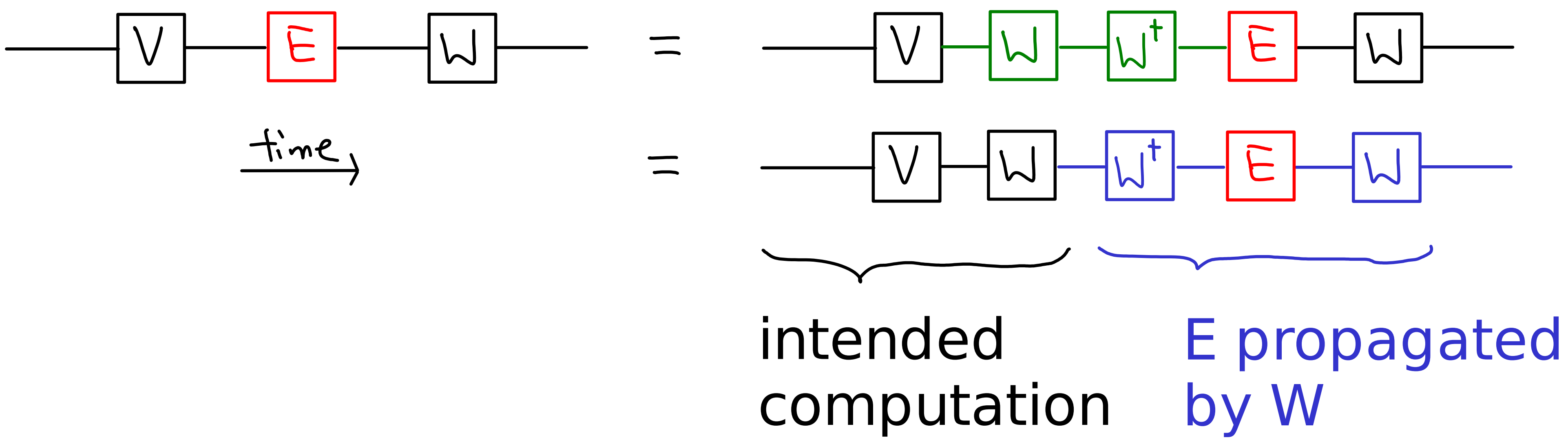
How H and CNOT propagate Pauli errors:

Suppose a Pauli error E occurs right before a single-qubit gate W:



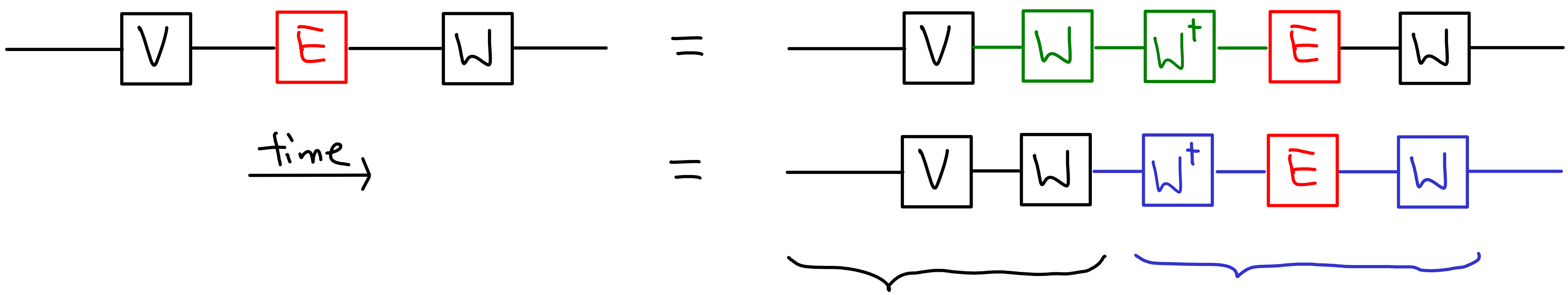
How H and CNOT propagate Pauli errors:

Suppose a Pauli error E occurs right before a single-qubit gate W:



How H and CNOT propagate Pauli errors:

Suppose a Pauli error E occurs right before a single-qubit gate W:



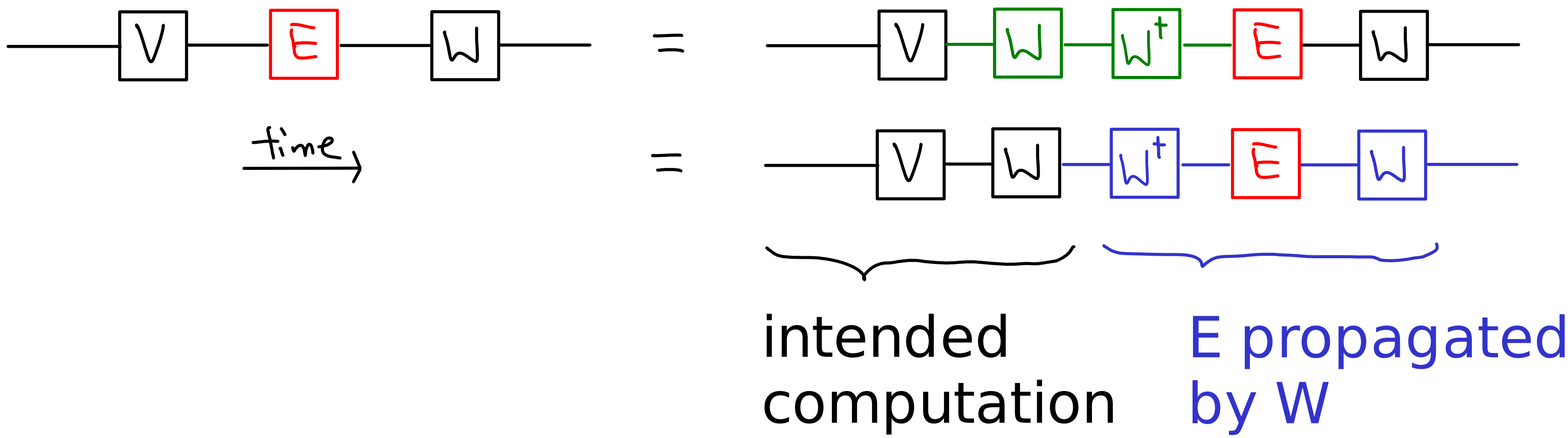
intended computation

E propagated by W

So, we can "propagate" the error to the right (later in time), E becomes $WEW^\dagger$.

How H and CNOT propagate Pauli errors:

Suppose a Pauli error E occurs right before a single-qubit gate W:

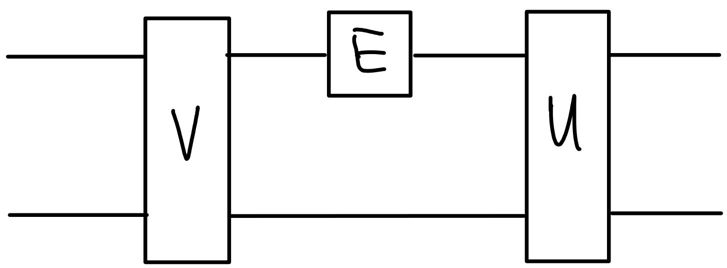


So, we can "propagate" the error to the right (later in time), E becomes $WEW^\dagger$.

Note a single qubit gate propagates a Pauli error (E) to another single-qubit error ($WEW^\dagger$) on the same qubit.

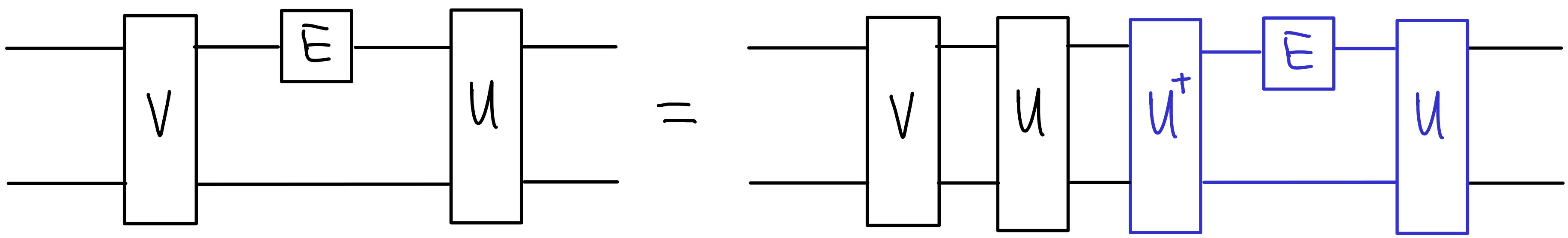
Trickier for U = CNOT:

$U XI U^\dagger = XX$, $U IX U^\dagger = IX$, $U ZI U^\dagger = ZI$, $U IZ U^\dagger = ZZ$



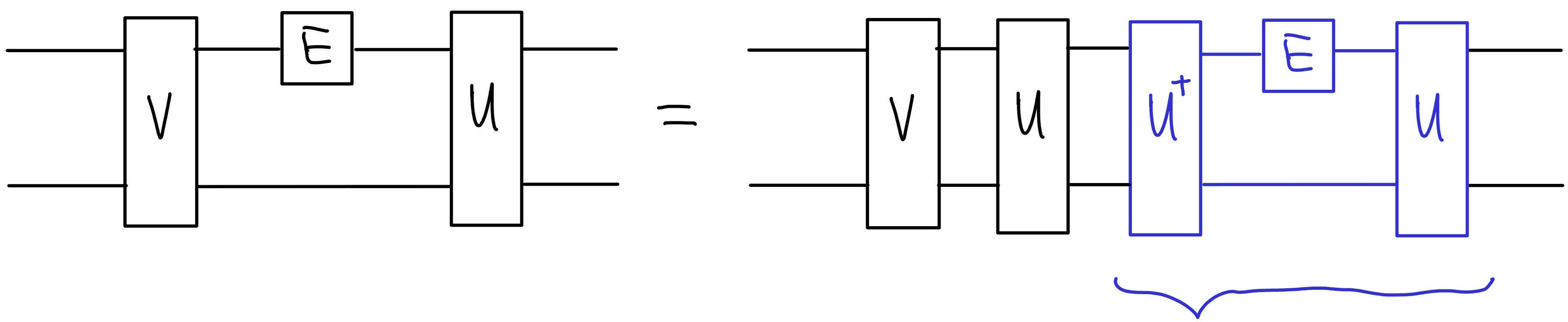
Trickier for U = CNOT:

$$U XI U^\dagger = XX\ ,\quad U IX U^\dagger = IX\ ,\quad U ZI U^\dagger = ZI\ ,\quad U IZ U^\dagger = ZZ$$



Trickier for U = CNOT:

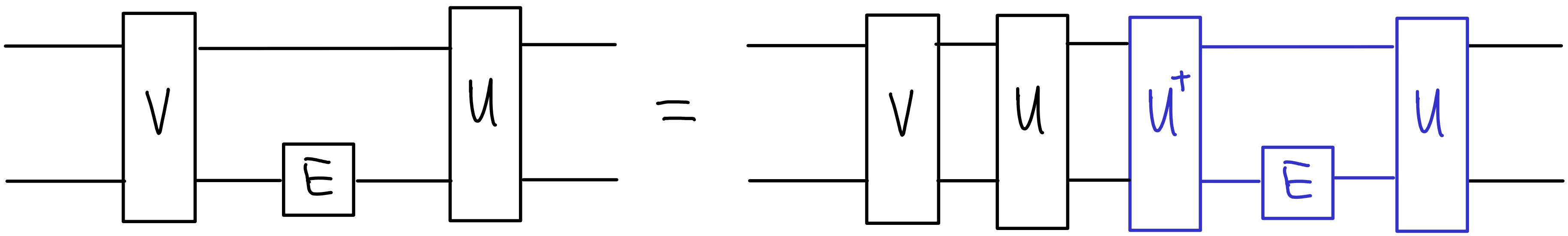
$U\,XI\,U^\dagger = XX\,,\quad U\,IX\,U^\dagger = IX\,,\quad U\,ZI\,U^\dagger = ZI\,,\quad U\,IZ\,U^\dagger = ZZ$



If E = X:
XI is propagated to XX,
1 error is turned into 2.
We call this "forward"
propagation (along the
direction of the CNOT).
This happens classically
too.

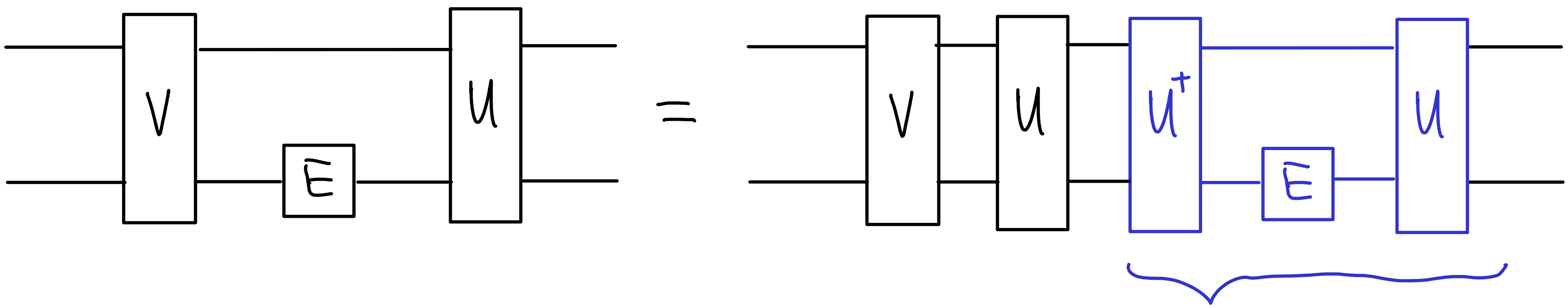
Trickier for U = CNOT:

$$U\,XI\,U^\dagger = XX\,,\quad U\,IX\,U^\dagger = IX\,,\quad U\,ZI\,U^\dagger = ZI\,,\quad U\,IZ\,U^\dagger = ZZ$$



Trickier for U = CNOT:

$$U XI U^\dagger = XX, \quad U IX U^\dagger = IX, \quad U ZI U^\dagger = ZI, \quad U IZ U^\dagger = ZZ$$

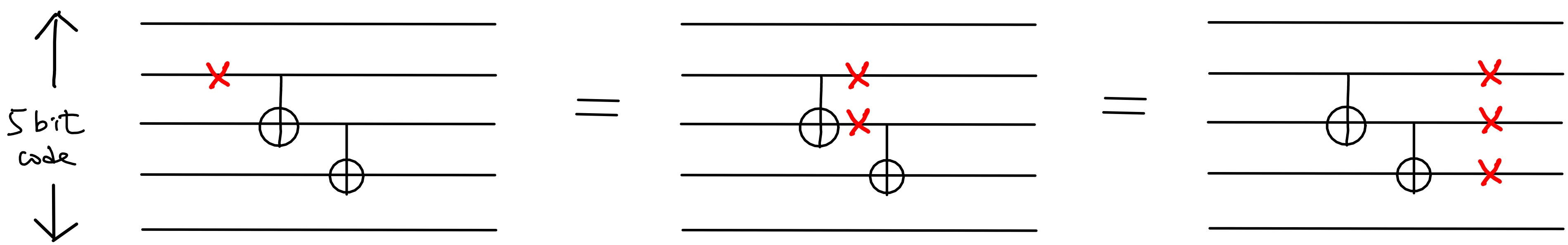


If E = Z:
IZ is propagated to ZZ
1 error turned into 2.
This is called "back action"
or "backward" propagation
of error (against the direction
of CNOT). That the error can
reach the controlled qubit is
uniquely quantum !

Suppose we start with a QECC that corrects any 1-qubit error.

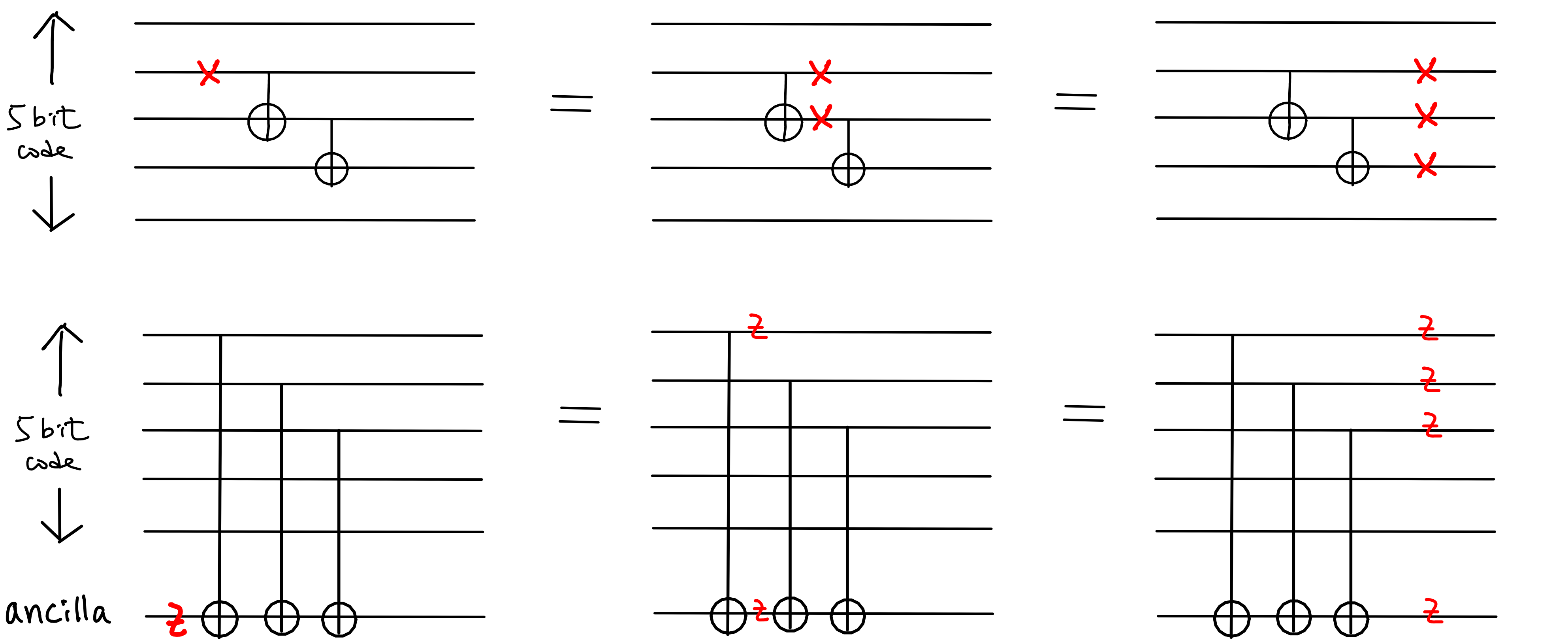
Suppose we start with a QECC that corrects any 1-qubit error.

If a circuit propagate 1 error to multiple errors within the code block (before the next error correction step), the new errors may not be correctible, resulting in a logical error. e.g.,

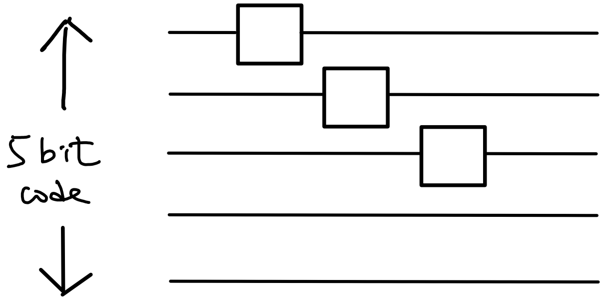


Suppose we start with a QECC that corrects any 1-qubit error.

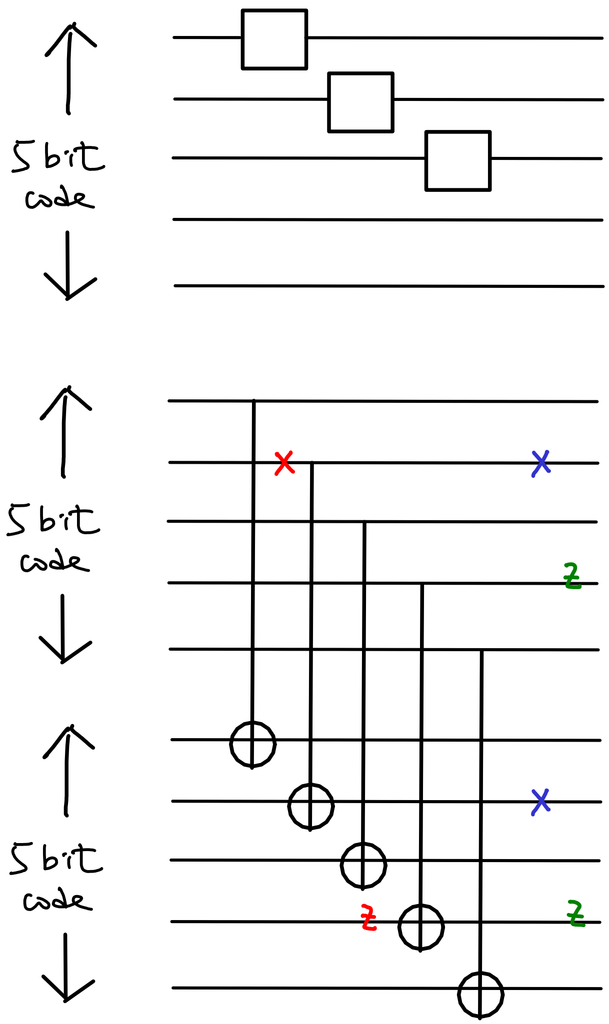
If a circuit propagate 1 error to multiple errors within the code block (before the next error correction step), the new errors may not be correctible, resulting in a logical error. e.g.,



In contrast, the following circuits do not propagate 1 error to multiple errors within the same code-block:

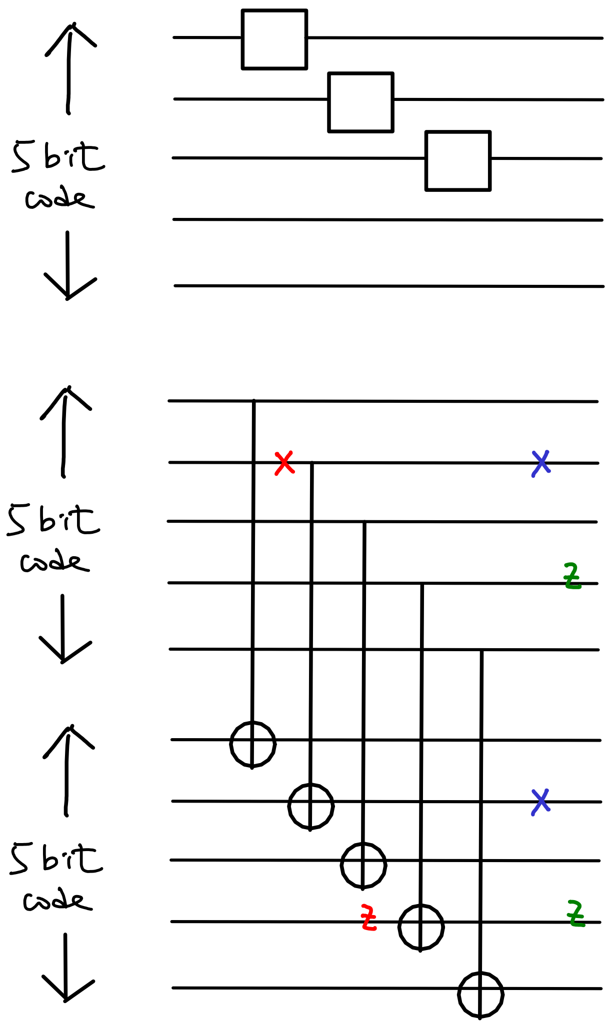


In contrast, the following circuits do not propagate 1 error to multiple errors within the same code-block:



e.g., 1 red X spreads to 2 blue X's, one in each code block.

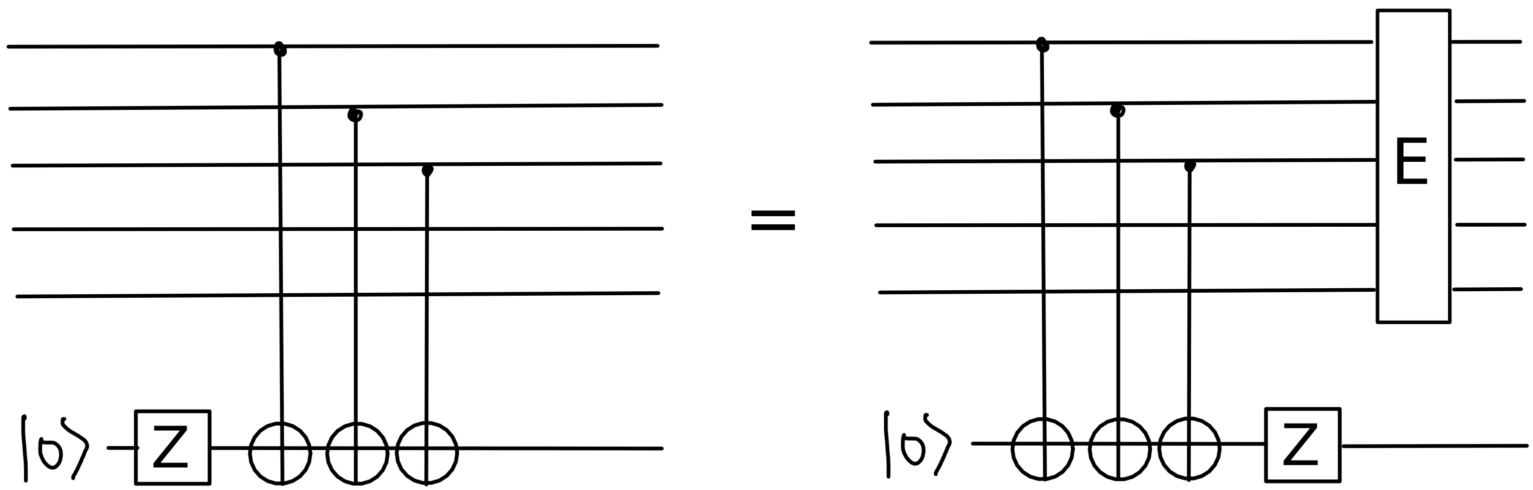
In contrast, the following circuits do not propagate 1 error to multiple errors within the same code-block:



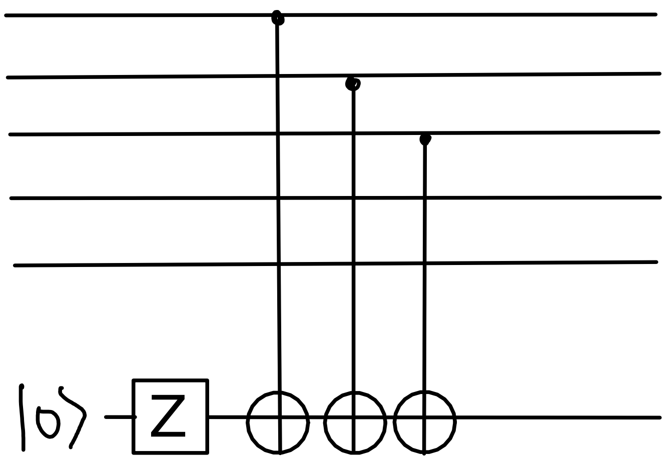
e.g., 1 red X spreads to 2 blue X's, one in each code block.

e.g., 1 red Z spreads to 2 green Z's, one in each code block.

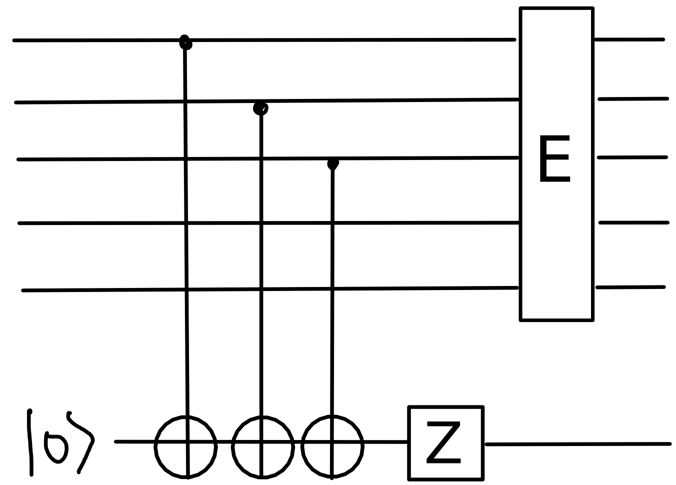
Question: How many Z errors are in E?



Question: How many Z errors are in E?

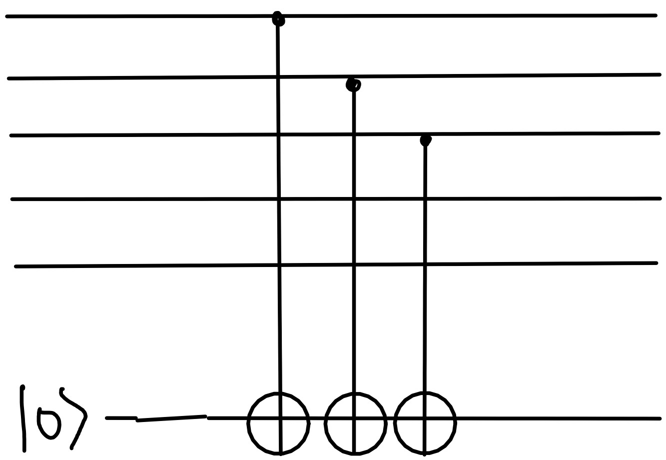


=



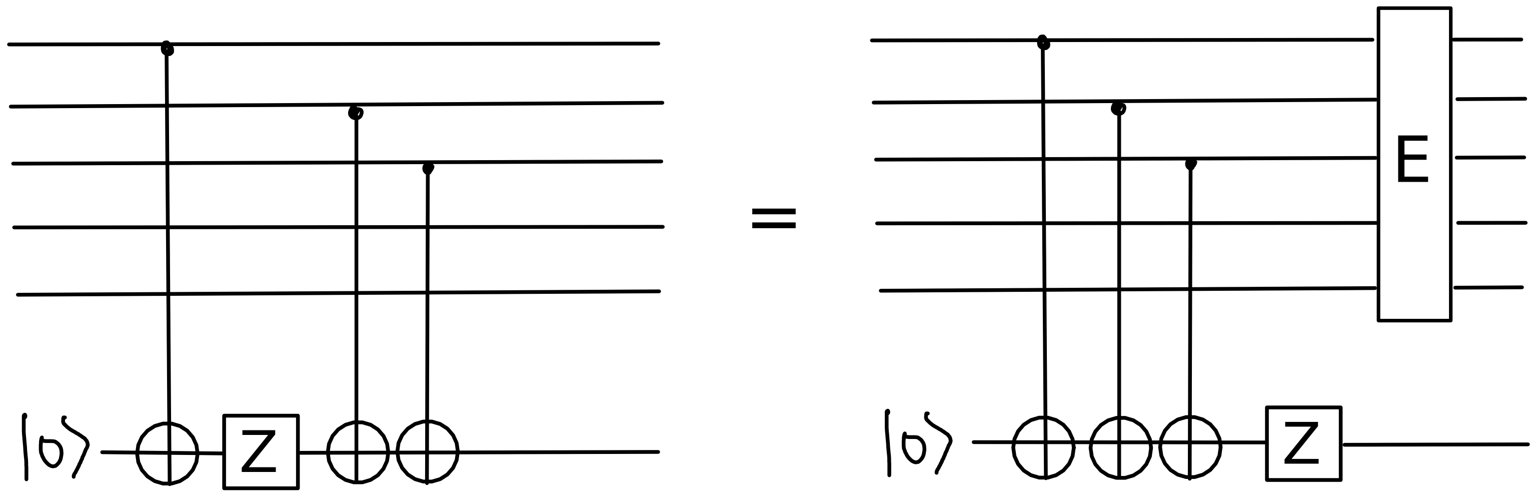
Answer: 3, E = ZZZII

||



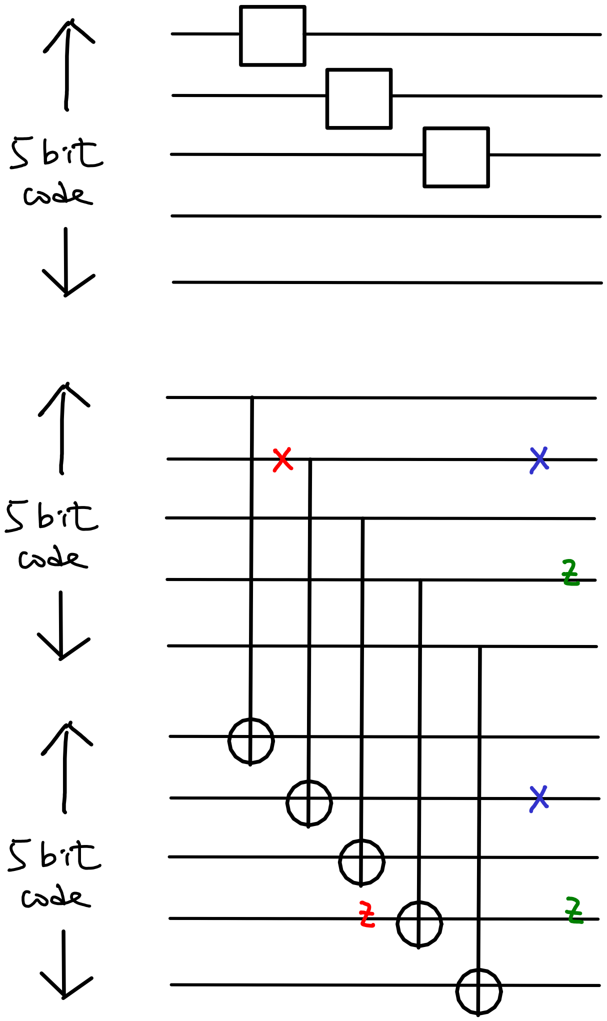
But really no error ...

A better question:



E=IZZII here!

In contrast, the following circuits do not propagate 1 error to multiple errors within the same code-block:



e.g., 1 red X spreads to 2 blue X's, one in each code block.

e.g., 1 red Z spreads to 2 green Z's, one in each code block.

These circuits are called "transversal".

Goal: find transversal encoded operations.