Application of Quantum Computation to High Energy Physics

– Quantum Error Correction –

Masazumi Honda

(本多正純)

19th Apr., 2023

Intensive lectures @RESCEU（day 3）

Plan of the intensive lectures

Day 1

- Lecture 1: introduction, basics of quantum computation
- Lecture 2: quantum simulation of spin system
- Hands-on 1: Basics on IBM's qiskit

Day 2

- Lecture 3: quantum field theory (QFT)
- Lecture 4: QFT on quantum computer
- Hands-on 2: Time evolution of spin system

Day 3

- Lecture 5: quantum error correction
- Lecture 6: some advanced topics, future prospects
- Hands-on 3: Constructing ground state of spin system

Plan

1. Basics on quantum error correction

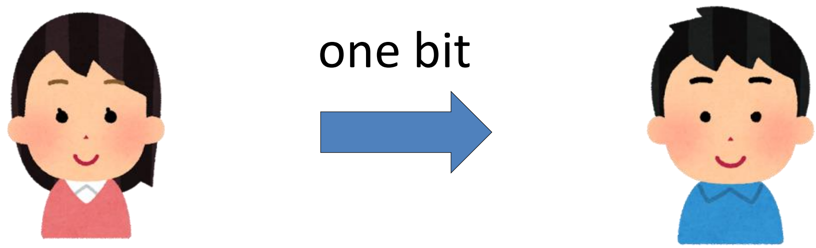
2. Classical linear code

3. A popular quantum code (“CSS code”)

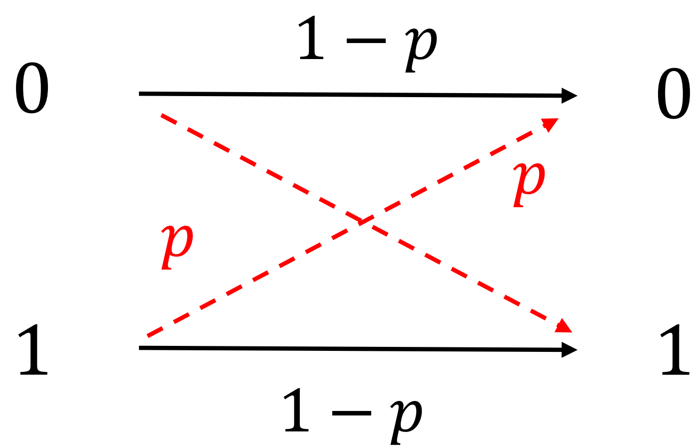
4. Summary

Classical error

Suppose we'd like to send a bit:

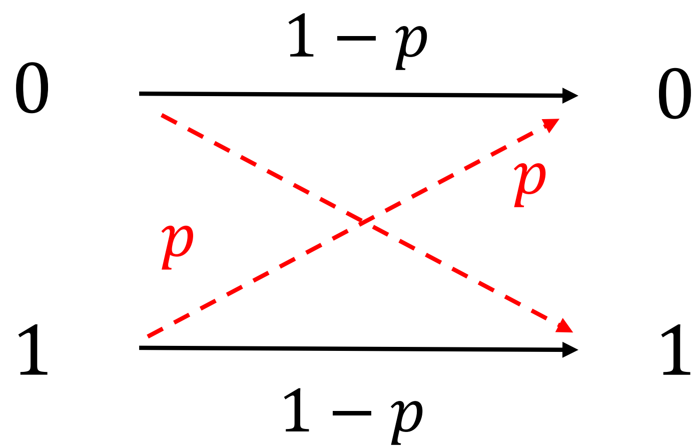


But we have "error" in probability p due to noise:



How can we correct the "error"?

Classical error correction: "Majority voting"



① Duplicate the bit (encoding):

$$0 \rightarrow 000, \qquad\qquad 1 \rightarrow 111$$

② Error detection & correction by "majority voting":

$$001 \rightarrow 000, \quad 011 \rightarrow 111, \quad \text{etc...}$$

③ But it fails if $^{\exists}$multiple "errors":

$$P_{\mathrm{failed}} = 3p^2(1-p) + p^3$$ (improved if $p < 1/2$)

Quantum errors

$$|\psi\rangle \;-\!\!\boxed{\text{Error}}\!\!-\; |\psi'\rangle$$

Differences from classical error:

- Errors are not only bit flips
  —— can be any unitary operators (continuous)
- Measurement destroys states
  —— projected to classical number (or smaller vector)
- No-cloning theorem
  —— impossible to make copies

Nevertheless,

$\exists$ systematic ways to correct errors

Classification of errors

Let's consider single qubit + environment

Error:
$$\begin{cases} |0\rangle \otimes |0\rangle_E \rightarrow |0\rangle \otimes |e_{00}\rangle_E + |1\rangle \otimes |e_{01}\rangle_E \\ |1\rangle \otimes |0\rangle_E \rightarrow |0\rangle \otimes |e_{10}\rangle_E + |1\rangle \otimes |e_{11}\rangle_E \end{cases}$$

For any state $|\psi\rangle \equiv c_0|0\rangle + c_1|1\rangle$,

$$|\psi\rangle \otimes |0\rangle_E \rightarrow c_0(|0\rangle \otimes |e_{00}\rangle_E + |1\rangle \otimes |e_{01}\rangle_E) + c_1(|0\rangle \otimes |e_{10}\rangle_E + |1\rangle \otimes |e_{11}\rangle_E)$$

$$= (c_0|0\rangle + c_1|1\rangle) \otimes \frac{|e_{00}\rangle_E + |e_{11}\rangle_E}{2} + (c_0|0\rangle - c_1|1\rangle) \otimes \frac{|e_{00}\rangle_E - |e_{11}\rangle_E}{2}$$
$$+ (c_0|1\rangle + c_1|0\rangle) \otimes \frac{|e_{01}\rangle_E + |e_{10}\rangle_E}{2} + (c_0|1\rangle - c_1|0\rangle) \otimes \frac{|e_{01}\rangle_E - |e_{10}\rangle_E}{2}$$

$$\equiv \underbrace{|\psi\rangle \otimes |e_I\rangle_E}_{\text{nothing}} + \underbrace{X|\psi\rangle \otimes |e_X\rangle_E}_{\text{bit flip}} + \underbrace{Z|\psi\rangle \otimes |e_Z\rangle_E}_{\text{phase flip}} + \underbrace{Y|\psi\rangle \otimes |e_I\rangle_E}_{\text{both } (Y = iXZ)}$$

Classification of errors (cont'd)

Single qubit case:

$(Y = iXZ)$

2×2 unitary matrix spanned by $\{I, X, Z, Y\}$

2-qubit case:

4×4 unitary matrix spanned by $\{I, X, Z, Y\} \otimes \{I, X, Z, Y\}$

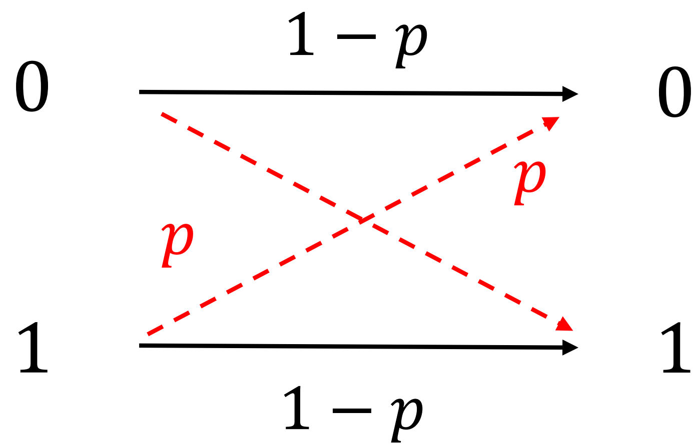
n-qubit case:

$2^n \times 2^n$ unitary matrix spanned by $\{I, X, Z, Y\}^{\otimes n}$

Error = Combination of bit flip & phase flip

Quantum error correction for bit flip

Classical bit flip:



Quantum bit flip:

$$|\psi\rangle \rightarrow \quad X|\psi\rangle$$ w/ probability p

$(c_0|0\rangle + c_1|1\rangle \;\rightarrow\; c_0|1\rangle + c_1|0\rangle)$

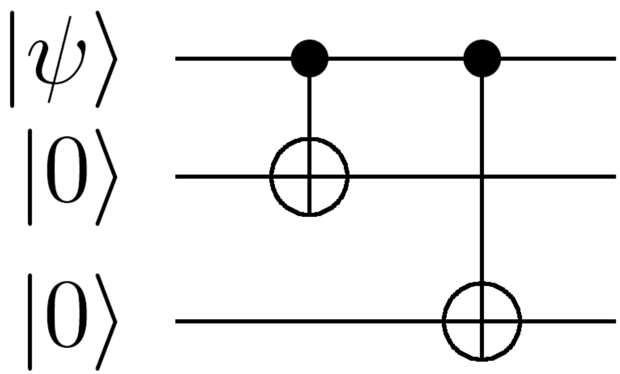
Can we extend the idea of "majority voting"?

Step 1/3: encoding ("3-qubit bit flip code")

Quantum bit flip:

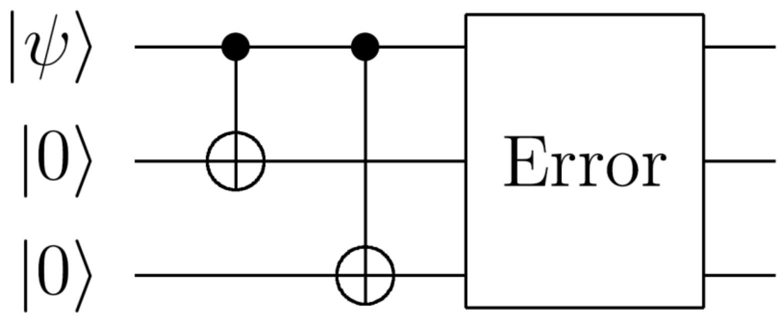
$$|\psi\rangle \rightarrow \quad X|\psi\rangle \qquad \text{w/ probability } p$$

Encoding:



$$|\psi\rangle \equiv c_0|0\rangle + c_1|1\rangle \quad \longrightarrow \quad c_0|000\rangle + c_1|111\rangle$$

Step 2/3: Error detection



Encoding:

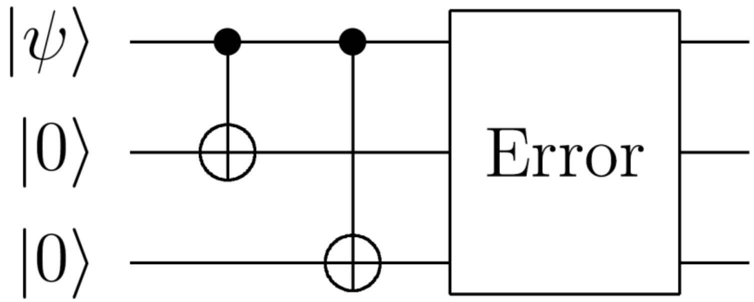
$$|\psi\rangle \rightarrow \ |\psi_E\rangle \equiv c_0|000\rangle + c_1|111\rangle$$

Bit flip error:

$$|\psi_E\rangle \rightarrow \ \ X_{1,2,3}|\psi_E\rangle \qquad \text{w/ probability } p$$

Can we detect the error w/o destroying the state?

Step 2/3: Error detection (cont'd)



Errors can be detected by knowing

"Syndrome measurements"

$$Z_1Z_2 \ \& \ Z_2Z_3$$

No error: $(Z_1Z_2)|\psi_E\rangle = +|\psi_E\rangle$, $(Z_2Z_3)|\psi_E\rangle = +|\psi_E\rangle$

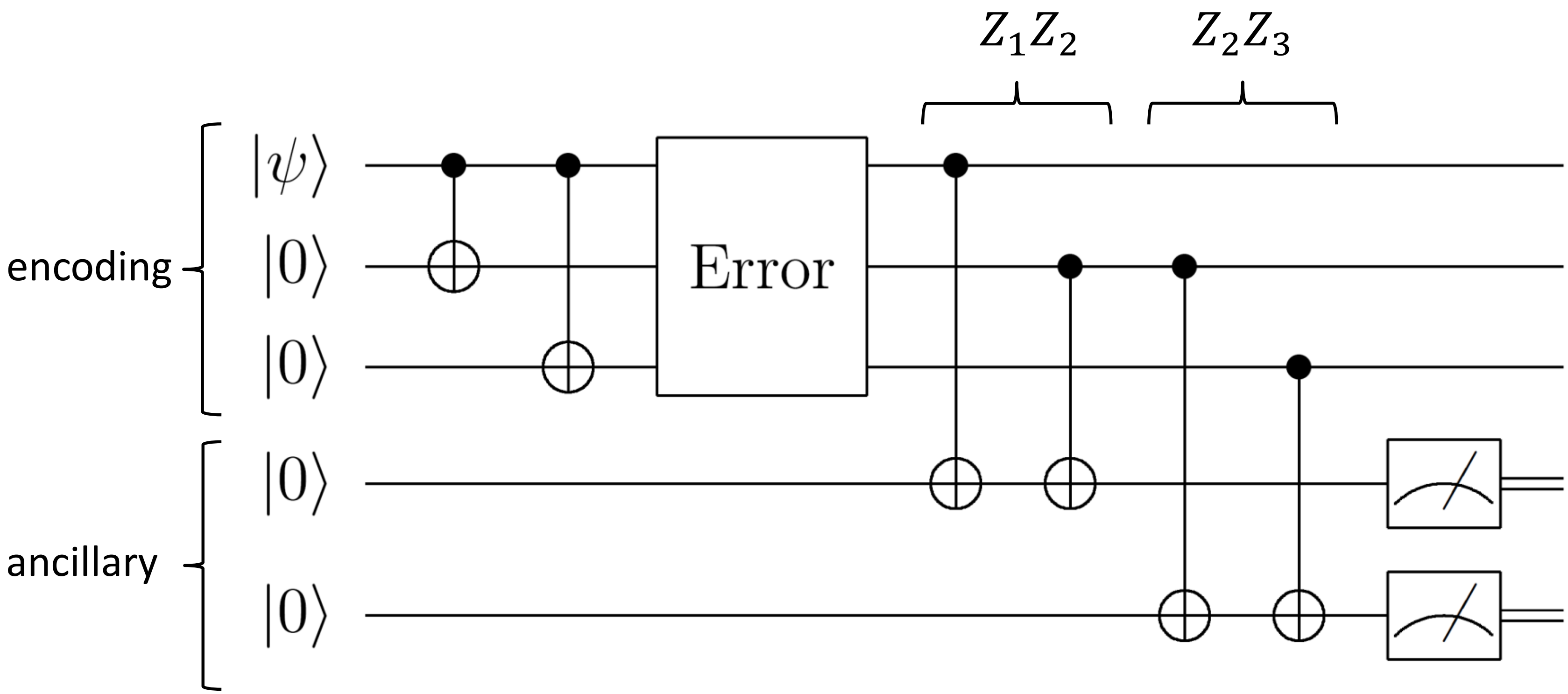
Error on 1st: $(Z_1Z_2)X_1|\psi_E\rangle = -X_1|\psi_E\rangle$, $(Z_2Z_3)X_1|\psi_E\rangle = +X_1|\psi_E\rangle$

Error on 2nd: $(Z_1Z_2)X_2|\psi_E\rangle = -X_2|\psi_E\rangle$, $(Z_2Z_3)X_2|\psi_E\rangle = -X_2|\psi_E\rangle$

Error on 3rd: $(Z_1Z_2)X_3|\psi_E\rangle = +X_3|\psi_E\rangle$, $(Z_2Z_3)X_3|\psi_E\rangle = -X_3|\psi_E\rangle$

But is it possible to know them (w/o destroying the state)?

Step 2/3: Error detection (cont'd)



Output on the 4th:

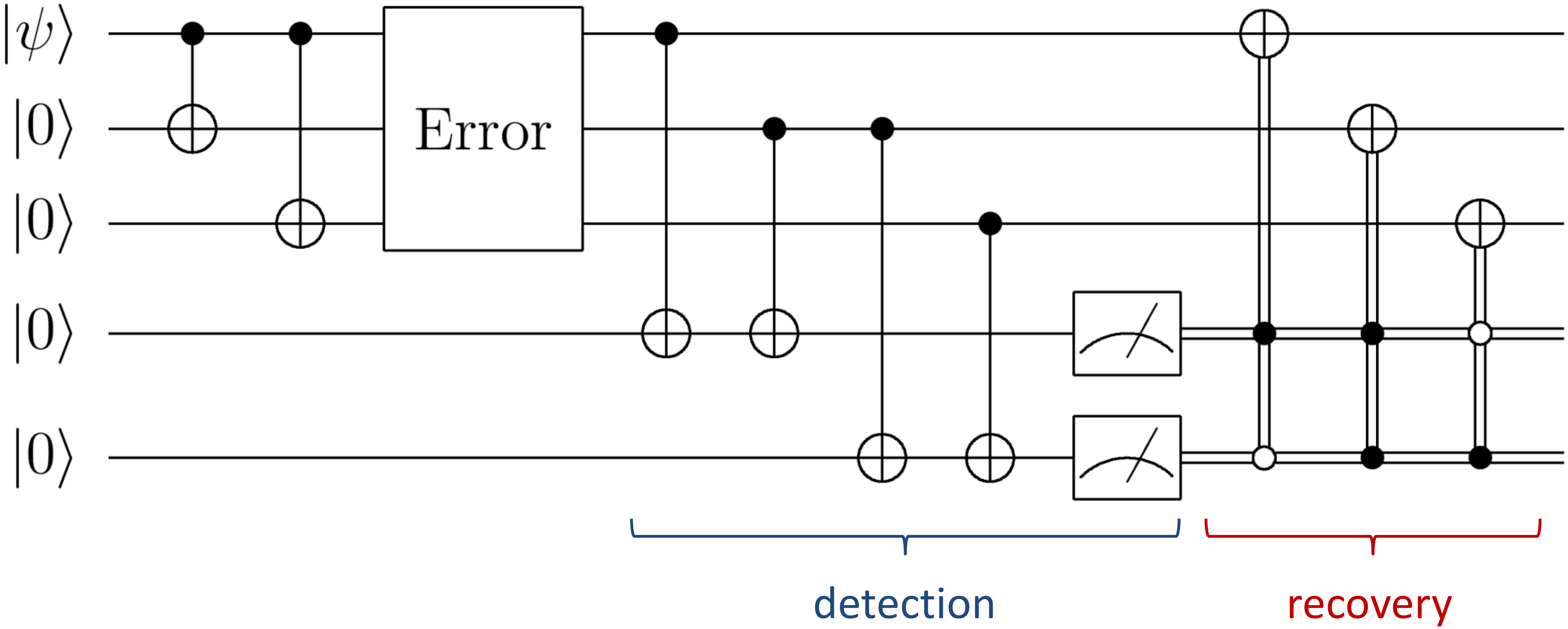
$$0 \text{ if } Z_1Z_2 = +1 \;\; \& \;\; 1 \text{ if } Z_1Z_2 = -1$$

Output on the 5th:

$$0 \text{ if } Z_2Z_3 = +1 \;\; \& \;\; 1 \text{ if } Z_2Z_3 = -1$$

Let's recover the information!!

Step 3/3: Error recovery



As in the classical case, it fails if $^{\exists}$multiple “errors”:

$$P_{\text{failed}} = 3p^2(1-p) + p^3 \qquad \text{(improved if } p < 1/2\text{)}$$

Quantum error correction for phase flip

Phase flip:

no classical analog

$$|\psi\rangle \to \; Z|\psi\rangle \quad \text{w/ probability } p$$

$$(c_0|0\rangle + c_1|1\rangle \;\to\; c_0|0\rangle - c_1|1\rangle)$$

Note:

$$(\,|+\rangle \equiv H|0\rangle,\; |-\rangle \equiv H|1\rangle\,)$$

$$Z|+\rangle = |-\rangle, \qquad Z|-\rangle = |+\rangle$$

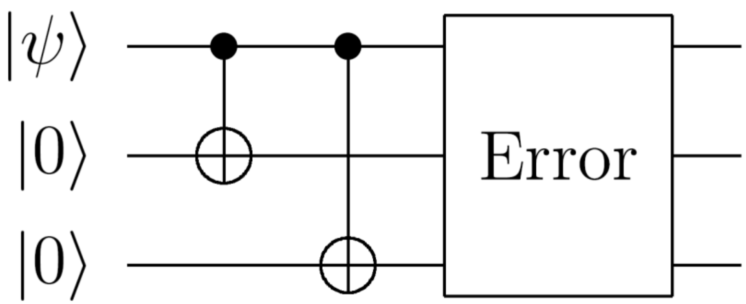
phase flip $=$ bit flip in the basis $|\pm\rangle$

done by a slight modification of the bit flip case

Step 1/3: encoding

Bit flip

($|\psi\rangle \rightarrow X|\psi\rangle$)



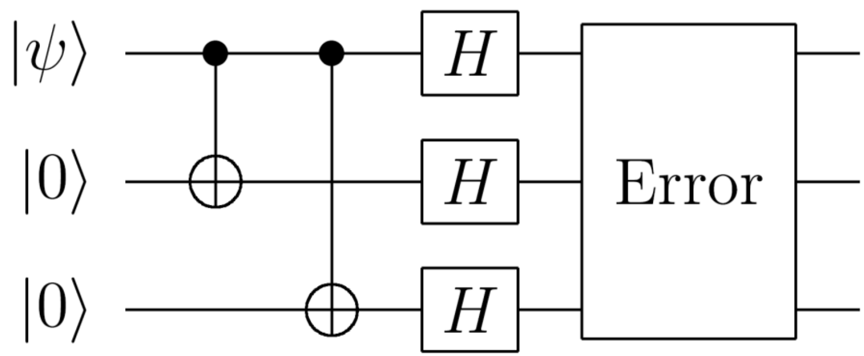
$$|\psi\rangle \equiv c_0|0\rangle + c_1|1\rangle$$

$$\Downarrow$$

$$c_0|000\rangle + c_1|111\rangle$$

Phase flip

($|\psi\rangle \rightarrow Z|\psi\rangle$)



$$|\psi\rangle \equiv c_0|0\rangle + c_1|1\rangle$$

$$\Downarrow$$

$$c_0|+++\rangle + c_1|---\rangle$$

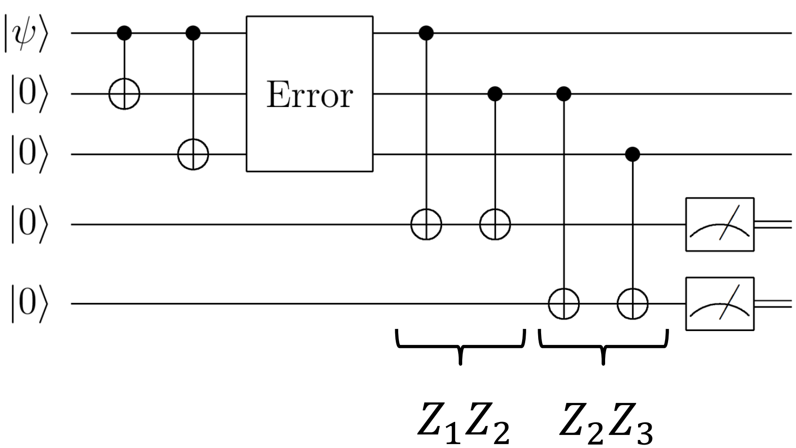
Step 2/3: error detection

Bit flip

$(\, |\psi\rangle \rightarrow X|\psi\rangle \,)$

done by knowing

$$Z_1Z_2 \ \& \ Z_2Z_3$$

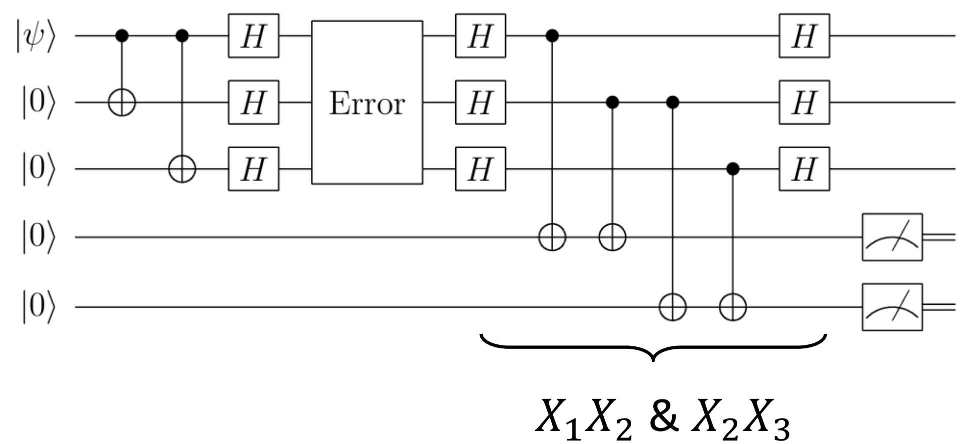


Phase flip

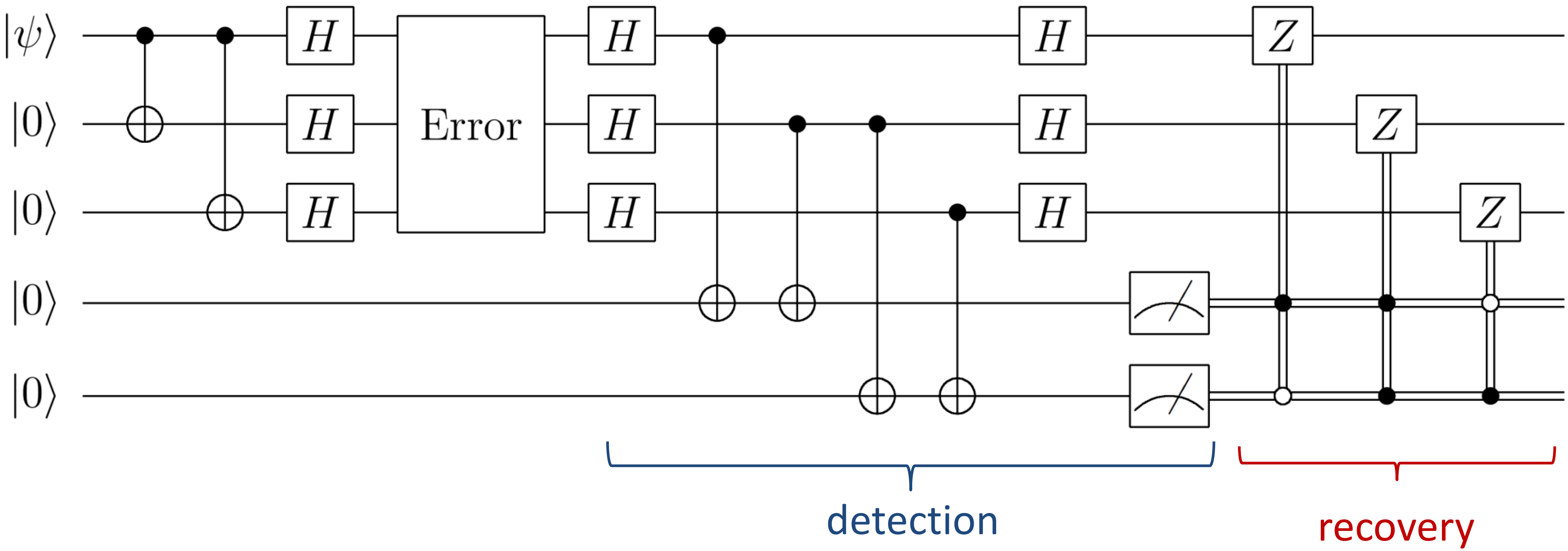
$(\, |\psi\rangle \rightarrow Z|\psi\rangle \,)$

done by knowing

$$X_1X_2 \ \& \ X_2X_3$$



Step 3/3: Error recovery



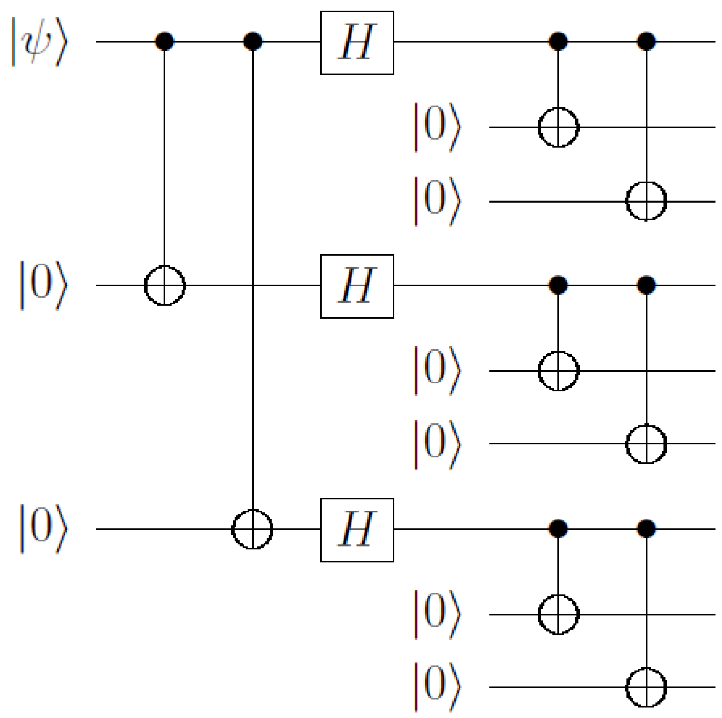
Similarly, it fails if $\exists$ multiple "errors":

$$P_{\text{failed}} = 3p^2(1-p) + p^3 \qquad \text{(improved if } p < 1/2\text{)}$$

Correction against arbitrary error on single qubit

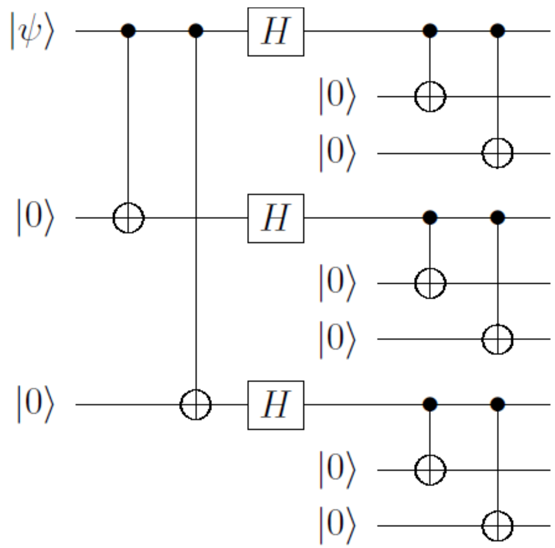
"Shor code" = a combination of the codes against bit flip & phase flip

Encoding:



$$|\psi\rangle \equiv c_0|0\rangle + c_1|1\rangle \to c_0 \frac{(|000\rangle+|111\rangle)^{\otimes 3}}{2\sqrt{2}} + c_1 \frac{(|000\rangle-|111\rangle)^{\otimes 3}}{2\sqrt{2}}$$

Error detection & recovery in Shor code



Ex.1) Bit flip on qubit 1

- detection by $Z_1Z_2 = -1$, $Z_2Z_3 = +1$
- recovery by applying X_1

Ex.2) Phase flip on qubit 1 (the same for qubit 2 & 3 cases)

- detection by $X_1X_2X_3X_4X_5X_6 = -1,\ X_4X_5X_6X_7X_8X_9 = +1$
- recovery by applying Z_1, Z_2 or Z_3

Plan

1. Basics on quantum error correction

2. Classical linear code

3. A popular quantum code (“CSS code”)

4. Summary

Classical linear code

(= a compact way to specify classical code)

$[n, k]$ code:

k bits $\xRightarrow{\text{encoding}}$ n bits $(k \leq n)$

It is specified by a $n \times k$ "generator matrix" G:

k-dim. vector $\xRightarrow{G}$ n-dim. vector

$||$ $||$

k bits n bits

Examples of classical linear code

- [3,1] code (the simplest majority voting)

  encoding: $0 \rightarrow 000,\ 1 \rightarrow 111$

  generator: $G = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} \quad \Longrightarrow \quad G(x = 0,1) = \begin{pmatrix} x \\ x \\ x \end{pmatrix}$

- [6,2] code

Examples of classical linear code

- [3,1] code (the simplest majority voting)

encoding: $0 \to 000,\ 1 \to 111$

generator: $G = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} \Rightarrow G(x = 0,1) = \begin{pmatrix} x \\ x \\ x \end{pmatrix}$

- [6,2] code

encoding: $00 \to 000000,\ 01 \to 000111,$ etc…

generator: $G = \begin{pmatrix} 1 & 0 \\ 1 & 0 \\ 1 & 0 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \end{pmatrix} \Rightarrow G\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} x \\ x \\ x \\ y \\ y \\ y \end{pmatrix}$

An equivalent definition by "parity check matrix"

Generator G ($n \times k$ matrix)

consists of k linearly independent n-dim. vectors :

$$G = (\vec{g}_1, \vec{g}_2, \cdots, \vec{g}_k)$$

Parity check matrix H ($n - k \times n$ matrix)

consists of $\mathrm{n} - k$ linearly indep. n-dim. vectors orthogonal to $\vec{g}$'s :

$$H = \begin{pmatrix} \vec{h}_1^T \\ \vec{h}_2^T \\ \vdots \\ \vec{h}_{n-k}^T \end{pmatrix} \quad \text{w/ } \overrightarrow{h_i} \cdot \overrightarrow{g_j} = 0 \pmod 2$$

Equivalently,

$$HG = \mathbf{0}_{(\boldsymbol{n}-\boldsymbol{k})\times \boldsymbol{k}} \pmod 2$$

Parity check matrix & error detection

Suppose we encode k-bits into n-bits:

$$x \text{ (}k\text{-bits)} \xRightarrow{\text{encoding}} y = Gx \text{ (n-bits)}$$

Then, we have

$$Hy = HGx = 0$$

But if we have an error: $y \to y' \equiv y + e$,

$$Hy' = He \neq 0$$ *Error is detected by H!*

Parity check = Syndrome measurement

Ex.) parity check matrix for [3,1] code

$G: n \times k$ matrix, $H: (n-k) \times n$ matrix, $HG = \mathbf{0}_{(n-k)\times k}$

Generator:

$$G = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} \quad \Rightarrow \quad G(x = 0{,}1) = \begin{pmatrix} x \\ x \\ x \end{pmatrix}$$

(an example of)

Parity check matrix:

$$H = \begin{pmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \end{pmatrix} \qquad (HG = \mathbf{0}_{2\times 1})$$

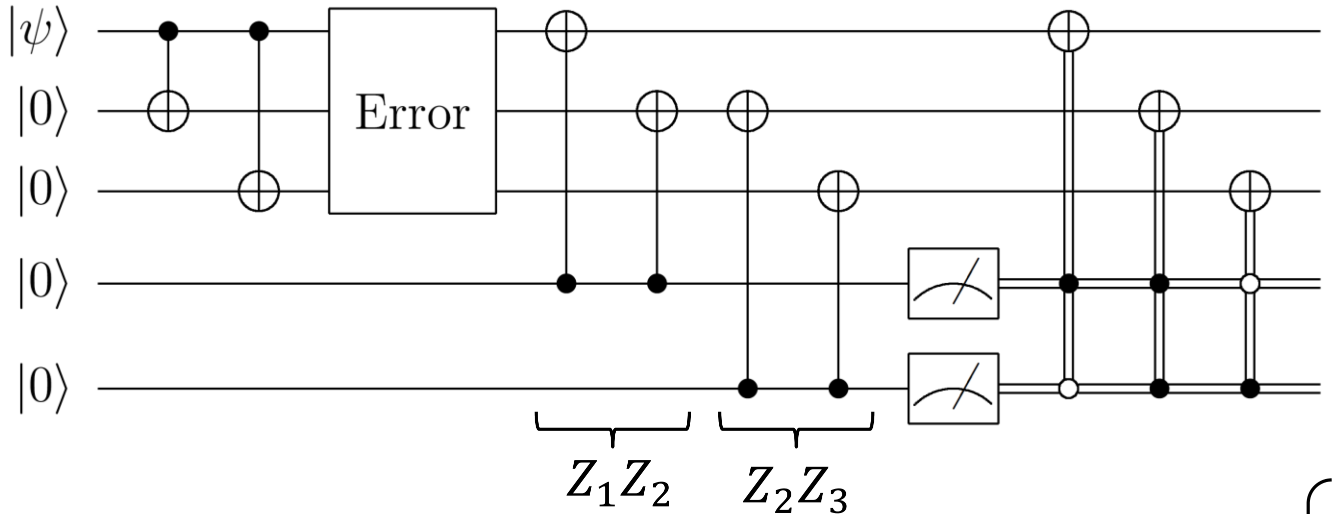
Error detection (for bit flip):

$$H\begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} = H\begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix} \qquad H\begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \qquad H\begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \end{pmatrix} \qquad \text{etc...}$$

😄 ☹️ ☹️

Correspondence to 3-qubit bit flip code

3-qubit bit flip code:



Parity check in [3,1] code:

$$H = \begin{pmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \end{pmatrix}$$

$$H\begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} = H\begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix} \quad \begin{matrix} Z_1Z_2 = +1 \\ Z_2Z_3 = +1 \end{matrix}$$

$$H\begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} = H\begin{pmatrix} 0 \\ 1 \\ 1 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \quad \begin{matrix} Z_1Z_2 = -1 \\ Z_2Z_3 = +1 \end{matrix}$$

$$H\begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} = H\begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \end{pmatrix} \quad \begin{matrix} Z_1Z_2 = -1 \\ Z_2Z_3 = -1 \end{matrix}$$

$$H\begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} = H\begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \end{pmatrix} \quad \begin{matrix} Z_1Z_2 = +1 \\ Z_2Z_3 = -1 \end{matrix}$$

How many errors can be corrected?

In [3,1] code, $(0 \to 000, 1 \to 111)$

"majority voting" can correct only 1-bit errors

$(001 \to 000, 011 \to 111,$ etc...)

Similarly,

$[2t+1, 1]$ code can correct only errors on up to t-bits

(ex. [1,5] code: $00001 \to 00000, 00011 \to 00000,$ etc...)

How about more general $[n, k]$ code?

$\to$ concept of "distance" is useful (next slide)

How many errors can be corrected? (cont'd)

Hamming distance:

$d(x, y) \equiv$ (# of different components between x & y)

Hamming weight:

$$\mathrm{wt}(x) \equiv d(x, 0)$$

In particular,

$$\mathrm{d(x, y)} = \mathrm{wt}(x + y)$$

Distance of a code C:

C: "$[n, k, d]$ code"

$$\mathrm{d(C)} \equiv \min_{\mathrm{y_1, y_2} \in C,\, y_1 \neq y_2} d(y_1, y_2) = \min_{y \in C, y \neq 0} \mathrm{wt}(y)$$

How many errors can be corrected? (cont'd)

Hamming distance:

$$d(x, y) \equiv \text{(\# of different components between } x \text{ \& } y\text{)}$$

Hamming weight:

$$\mathrm{wt}(x) \equiv d(x, 0)$$

In particular,

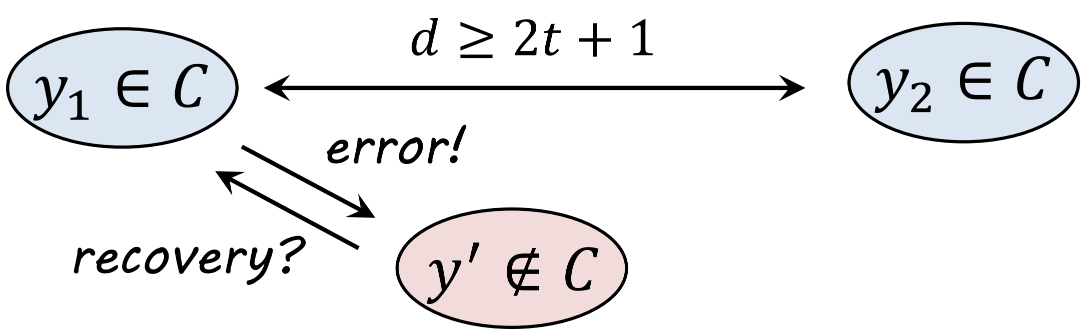
$$\mathrm{d(x, y)} = \mathrm{wt}(x + y)$$

Distance of a code C:

C: "$[n, k, d]$ code"

$$\mathrm{d(C)} \equiv \min_{y_1, y_2 \in C,\, y_1 \neq y_2} d(y_1, y_2) = \min_{y \in C, y \neq 0} \mathrm{wt}(y)$$

If $d(C) = 2t + 1$, then we can correct errors on up to t-bits



Examples

$$\left(\begin{array}{c} d(x,y) \equiv (\text{\# of different components between } x \;\&\; y),\;\; \mathrm{wt}(x) \equiv d(x,0) \\ \mathrm{d(C)} \equiv \min\limits_{y_1,y_2 \in C,\, y_1 \neq y_2} d(y_1,y_2) = \min\limits_{y \in C, y \neq 0} \mathrm{wt}(y) \end{array} \right)$$

- [3, 1] code:

$$G(x = 0,1) = \begin{bmatrix} x \\ x \\ x \end{bmatrix} \quad \Rightarrow \quad \mathrm{d(C)} = 3$$

can correct 1 error

- [6, 2] code:

Examples

$$\left(\begin{array}{c} d(x,y) \equiv (\text{\# of different components between } x \text{ \& } y),\ \ \mathrm{wt}(x) \equiv d(x,0) \\ \mathrm{d(C)} \equiv \min\limits_{y_1,y_2 \in C,\ y_1 \neq y_2} d(y_1,y_2) = \min\limits_{y \in C, y \neq 0} \mathrm{wt}(y) \end{array} \right)$$

- [3, 1] code:

$$G(x = 0{,}1) = \begin{pmatrix} x \\ x \\ x \end{pmatrix} \Rightarrow \mathrm{d(C)} = 3$$

can correct 1 error

- [6, 2] code:

$$G\begin{pmatrix} x_1 = 0,1 \\ x_2 = 0,1 \end{pmatrix} = \begin{pmatrix} x_1 \\ x_1 \\ x_1 \\ x_2 \\ x_2 \\ x_2 \end{pmatrix} \Rightarrow \mathrm{d(C)} = 3$$

Plan

1. Basics on quantum error correction

2. Classical linear code

3. A popular quantum code (“CSS code”)

4. Summary

Calderbank-Shor-Steane (CSS) code

Ingredients = 2 classical linear codes

- C_1: $[n, k_1]$ code w/ parity check mat. H_1
- $C_2 \subset C_1$: $[n, k_2]$ code w/ parity check mat. H_2 $(k_2 < k_1)$

CSS code = $[n, k_1 - k_2]$ quantum code

specify a codeword by equivalent class C_1/C_2:

$$|\bar{v}\rangle \equiv \frac{1}{\sqrt{2^{k_2}}} \sum_{w \in C_2} |v + w\rangle \qquad (v \in C_1)$$

$2^{k_2-k_1}$ $\quad$ 2^{k_1} $\quad$ 2^{k_2}

Error correction in CSS code (bit flip)

bit flip error

$$|\bar{v}\rangle = \frac{1}{\sqrt{2^{k_2}}}\sum_{w\in C_2}|v+w\rangle \quad \Longrightarrow \quad |\bar{v}'\rangle \equiv \frac{1}{\sqrt{2^{k_2}}}\sum_{w\in C_2}|v+w+e_1\rangle$$

1. Introduce ancillary qubits:

$$|\bar{v}\rangle \otimes |0\rangle_A \quad \Longrightarrow \quad |\bar{v}'\rangle \otimes |0\rangle_A$$

2. Make unitary trans. s.t. $|v\rangle \otimes |0\rangle_A \rightarrow |v\rangle \otimes |H_1 v\rangle_A$:

$$\Longrightarrow \frac{1}{\sqrt{2^{k_2}}}\sum_{w\in C_2}|v+w+e_1\rangle \otimes |H_1(v+w+e_1)\rangle_A = |\bar{v}'\rangle \otimes |H_1 e_1\rangle_A$$

3. Measure $|H_1 e_1\rangle_A$ to identify where errors occur

4. Recover the errors by acting X on appropriate places

Error correction in CSS code (phase flip)

phase flip error

$$|\bar{v}\rangle = \frac{1}{\sqrt{2^{k_2}}}\sum_{w\in C_2}|v+w\rangle \quad \Rightarrow \quad |\bar{v}'\rangle \equiv \frac{1}{\sqrt{2^{k_2}}}\sum_{w\in C_2}(-1)^{(v+w)\cdot e_2}|v+w\rangle$$

1. Basis change by acting H on all the qubits:

(Note $H|x\rangle = \frac{1}{\sqrt{2}}\sum_{y=0,1}(-1)^{xy}|y\rangle$)

$$\Rightarrow \quad \frac{1}{\sqrt{2^{k_2}}}\sum_{u}\sum_{w\in C_2}(-1)^{(v+w)\cdot(e_2+u)}|u\rangle \quad = \frac{1}{\sqrt{2^{k_2}}}\sum_{u'}\sum_{w\in C_2}(-1)^{(v+w)\cdot u'}|u'+e_2\rangle$$

bit flip in this basis!

2. Correct the “bit flip error” as in the bit flip case:

$$\Rightarrow \quad \frac{1}{\sqrt{2^{k_2}}}\sum_{u'}\sum_{w\in C_2}(-1)^{(v+w)\cdot u'}|u'\rangle$$

3. Go back to the original basis by acting H on all the qubits:

$$\Rightarrow \quad \frac{1}{\sqrt{2^{k_2}}}\sum_{w\in C_2}|v+w\rangle = |\bar{v}\rangle \quad ☑$$

The most popular case: "Steane (7-qubit) code"

Generic CSS = $[n, k_1 - k_2]$ quantum code:

- C_1: $[n, k_1]$ code w/ generator G_1 & parity check mat. H_1
- $C_2 \subset C_1$: $[n, k_2]$ code w/ generator G_2 & parity check mat. H_2

$$\Rightarrow \quad |\bar{v}\rangle \equiv \frac{1}{\sqrt{2^{k_2}}} \sum_{w \in C_2} |v + w\rangle$$

Steane code = $[7, 1]$ quantum code :

$(n = 7, k_1 = 4, k_2 = 3)$

- C_1: $[7, 4]$ code w/ generator G_1 & parity check mat. H_1
- $C_2 \subset C_1$: $[7, 3]$ code w/ generator $G_2 = H_1^T$ & parity check mat. $H_2 = \mathrm{G}_1^{\mathrm{T}}$

$$G_1 = \begin{pmatrix} 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 \end{pmatrix} = H_2^T, \qquad H_1 = \begin{pmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{pmatrix} = G_2^T$$

Codewords in Steane code

$$\begin{cases} C_1: & [7,4] \text{ code w/ generator } G_1 \\ C_2 \subset C_1: & [7,3] \text{ code w/ generator } G_2 \end{cases} \Longrightarrow |\bar{v}\rangle \equiv \frac{1}{\sqrt{8}} \sum_{w \in C_2} |v+w\rangle$$

$$G_1 = \begin{pmatrix} 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 \end{pmatrix} \equiv (w_1, w_2, w_3, u), \qquad G_2 = \begin{pmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 0 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \\ 1 & 1 & 1 \end{pmatrix} = (w_1, w_2, w_3)$$

$$C_2 = \{0, w_1, w_2, w_3, w_1 + w_2, w_1 + w_3, w_2 + w_3, w_1 + w_2 + w_3\}$$

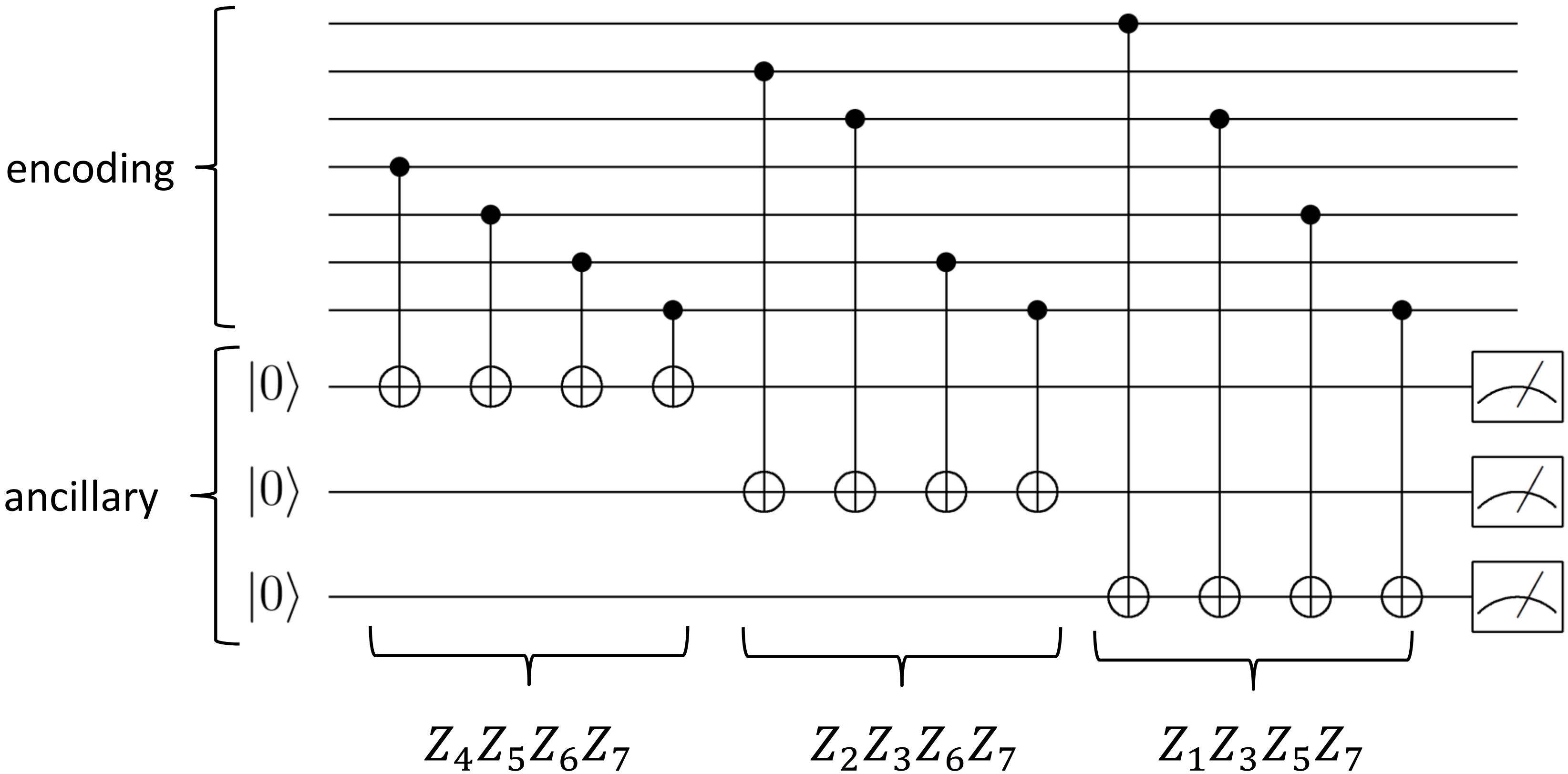
$$C_1 = \{C_2\text{'s elements}, u + C_2\text{'s elements}\}$$

$$\Downarrow$$

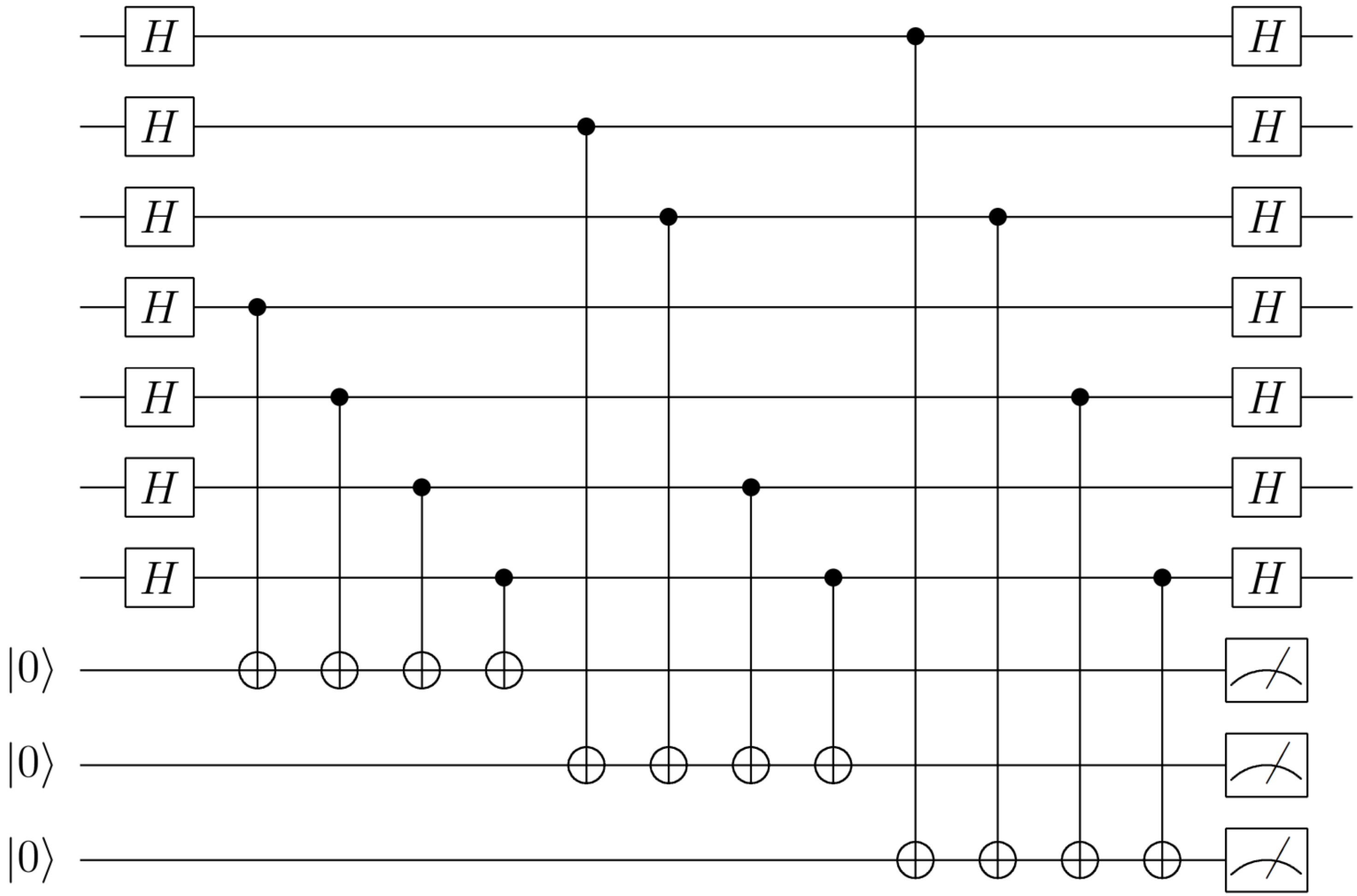
$$|\bar{0}\rangle \equiv \frac{1}{\sqrt{8}} \sum_{w \in C_2} |0000000 + w\rangle, \quad |\bar{1}\rangle \equiv \frac{1}{\sqrt{8}} \sum_{w \in C_2} |1111111 + w\rangle$$

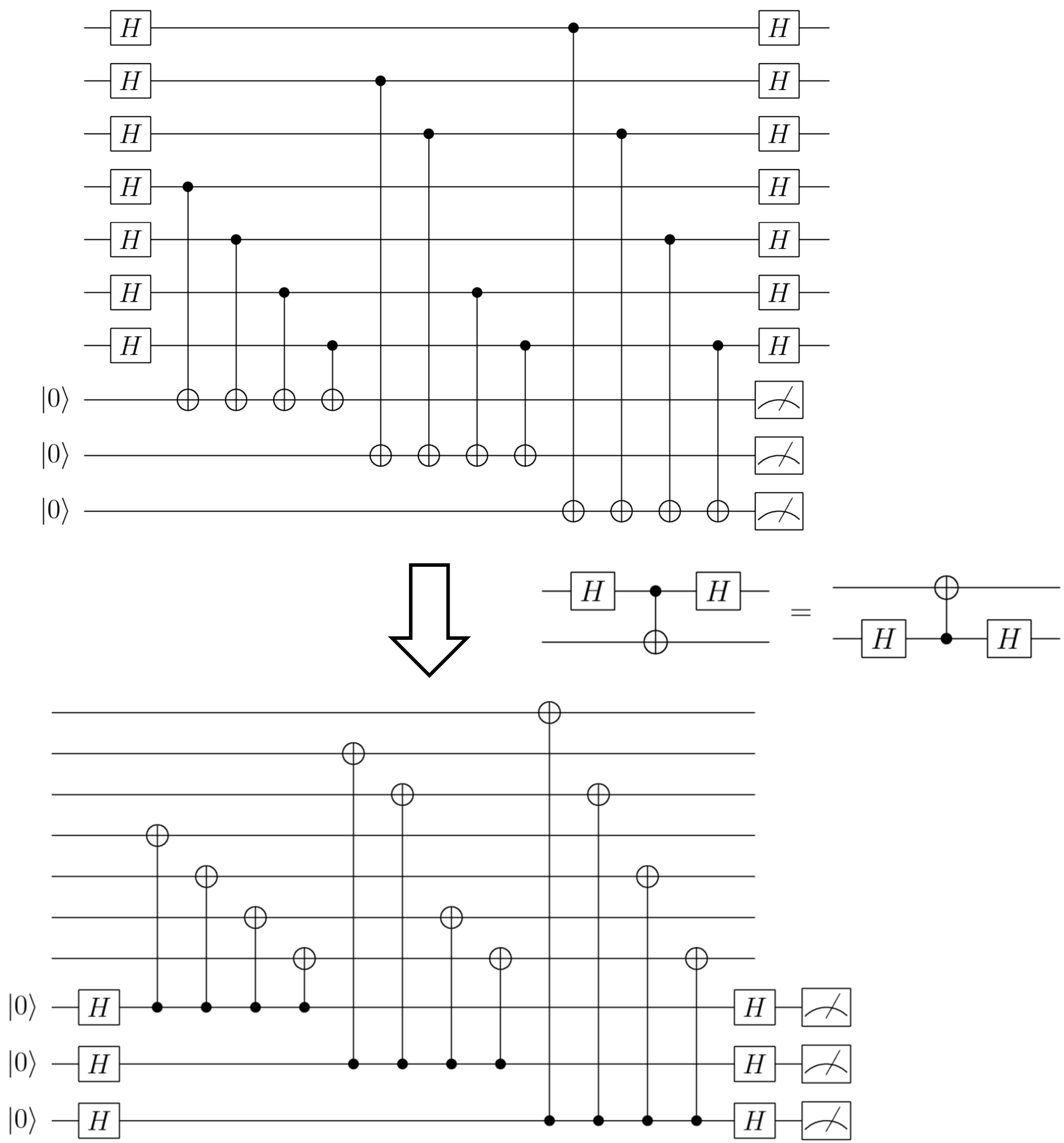
Bit flip error detection in Steane code

$$H_1 = \begin{pmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{pmatrix}$$



Phase flip error detection in Steane code





Prime factorization beyond supercomputer?

- Steane code: (error probability ϵ) $\Longrightarrow$ $\mathcal{O}(\epsilon^2)$
- 2-level Steane code: (err. prob. ϵ) $\Longrightarrow$ $\mathcal{O}(\epsilon^4)$

  =replacing each qubit in Steane code by Steane code
- L-level Steane code: (err. prob. ϵ) $\Longrightarrow$ $\mathcal{O}(\epsilon^{2L})$

Suppose

130-digit (=432-bit) prime factorization problem

which takes a few months by (slightly earlier) supercomputer

Prime factorization beyond supercomputer?

- Steane code: (error probability ϵ) $\Longrightarrow$ $\mathcal{O}(\epsilon^2)$
- 2-level Steane code: (err. prob. ϵ) $\Longrightarrow$ $\mathcal{O}(\epsilon^4)$

  =replacing each qubit in Steane code by Steane code
- L-level Steane code: (err. prob. ϵ) $\Longrightarrow$ $\mathcal{O}(\epsilon^{2L})$

Suppose

130-digit (=432-bit) prime factorization problem

which takes a few months by (slightly earlier) supercomputer

- Shor's algorithm requires 5×432 qubits
- need 3-level Steane code to have small errors (expected)

$\Longrightarrow$ $5 \times 432 \times 7^3 + (\text{ancilla}) \sim 1000000$ qubits!

Summary

Summary

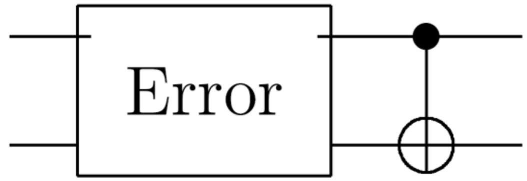
- Real quantum computer has errors
- Error correction is important to get reliable results
- Quantum errors are not only bit flips
- Generic error can be understood as a combination of bit flips & phase flips
- Quantum error correction likely requires a huge number of qubits

(Note: this lecture doesn't include recent progress of quantum correction.
The latest prospect may be quantitatively different.)

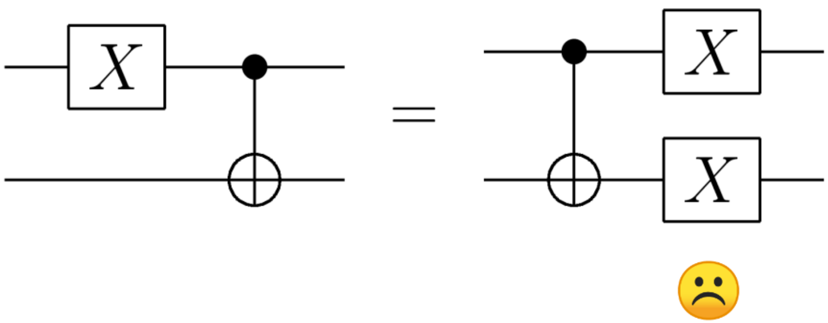
Here is the end of lecture 5!

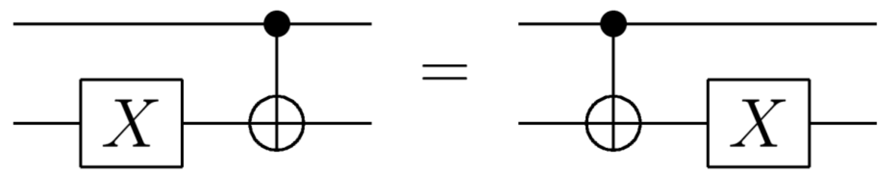
Appendix

Propagation of errors in implementing CX



Bit flip error:





Phase flip error: