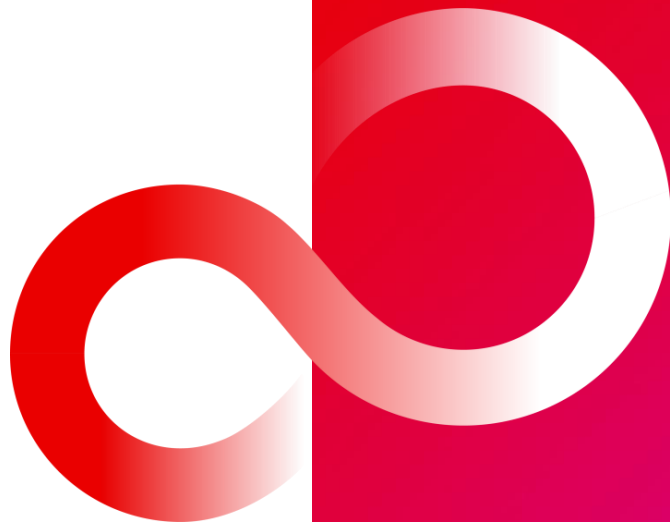


量子エラー訂正と 誤り耐性量子計算

2023/10/13

富士通研究所

赤星友太郎



○名前：赤星友太郎

○経歴

○学部：京都大学工学部・物理工学科

○希薄気体の数値シミュレーション

○大学院（修士～博士）：京都大学基礎物理学研究所・素粒子論グループ

○格子QCDを用いたハドロン間相互作用の研究

○現在：富士通研究所

○量子コンピュータの研究開発（量子エラー訂正）

富士通の量子コンピューティング研究開発戦略



- 量子デバイスから基盤ソフト、アプリまですべての領域に、世界有数の研究機関と取り組む
- ソフトウェア技術に注力する一方、ハードは幅広く可能性を追求
- 量子シミュレータを活用し、エンドユーザーと早期からアプリケーション開拓に取り組む

PRESS RELEASE

2023年10月5日

富士通株式会社

国立研究開発法人理化学研究所

理研と共同で64量子ビットデバイスを開発！

**超伝導量子コンピュータを開発し、量子シミュレータ
と連携可能なプラットフォームを提供**

量子化学計算、量子金融アルゴリズムなどの研究開発を加速

理化学研究所

中村教授



デルフト工科大学

3



広く検討

© 2023 Fujitsu Limited

Emerson教授

技術

藤井教授

回路

方式

- 量子コンピュータにおける**量子エラー訂正**と**誤り耐性量子計算**に関する講義
- エラー訂正の基礎的な話からはじめ、一気に最近の話題まで駆け上がる
- それゆえ、歴史的なことも含め一部スキップする部分があるが、それらは参考文献等で補っていただきたい

- 基礎事項
 - 量子コンピュータ概要
 - 量子コンピュータによる計算（ゲート方式）
- 量子エラー訂正
 - スタビライザー符号
 - 簡単な例：反復符号による量子エラー訂正
 - 近年の主流：表面符号
- 誤り耐性量子計算(FTQC)
 - 概要
 - 論理ゲートの実行方法
 - 近年の主流：Lattice surgeryによる論理ゲート実行
- 最近の話題：Early-FTQCに向けた新量子計算アーキテクチャ



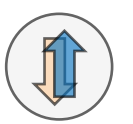
基礎事項

量子コンピュータ概要

量子コンピュータによる計算（ゲート方式）

量子力学の性質を活用し、高速な計算を実現するコンピュータ

量子ビット：量子コンピュータを構成する最小単位（従来型（以下、古典と呼ぶ）のコンピュータにおけるビットに相当）

 = $|0\rangle$  = $|1\rangle$ 古典ビットと異なり、0と1の
重ね合わせ状態も取りうる：  = $\alpha|0\rangle + \beta|1\rangle$

- 重ね合わせによる**並列性**と、量子力学の性質である**干渉効果**により高速化
- 創薬や金融など、古典コンピュータでは困難な計算を高速に実行できると期待されている

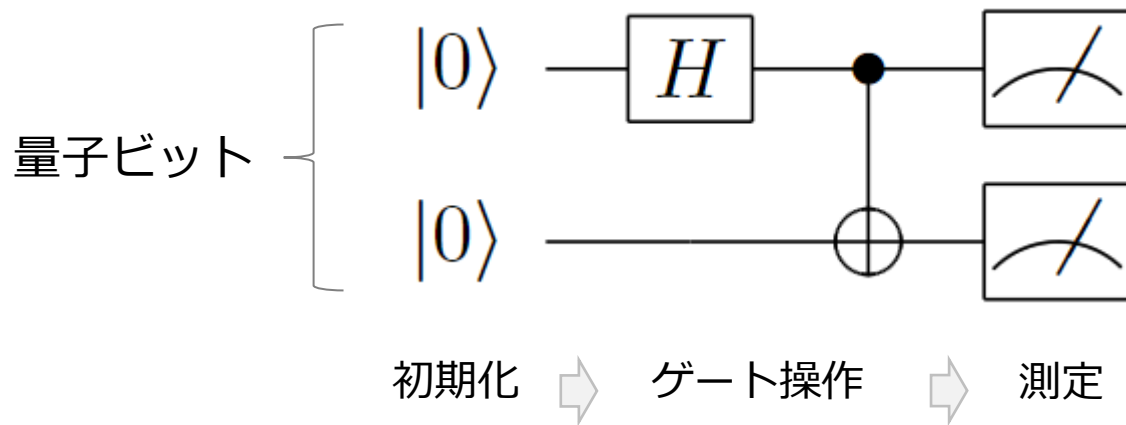


<https://www.fujitsu.com/jp/about/research/technology/quantum/>より

- 量子ビットに量子ゲートを作用させていくことで計算が行われる
- 量子ゲートを作用させる一連の操作は、**量子回路(下図)**によって記述される

2量子ビットの簡単な量子回路の例：

左から右へ計算が進む



○量子ビット (量子状態) の記述

n 量子ビットの量子状態は 2^n 次元ヒルベルト空間のベクトル
量子ゲートは 2^n 次元ヒルベルト空間に作用するユニタリ演算子で記述される

$$\alpha|0\rangle + \beta|1\rangle = \begin{bmatrix} \alpha \\ \beta \end{bmatrix}$$

ブラケット記法

ベクトル記法

$$|0\rangle^{\otimes 3} = |000\rangle$$

多量子ビット状態

○ 代表的な1量子ビットゲート

➤ パウリ演算子

$$X \equiv \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}; \quad Y \equiv \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix}; \quad Z \equiv \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$$

$$\{\sigma_i, \sigma_j\} = 0 \quad (\sigma_1 = X, \sigma_2 = Y, \sigma_3 = Z)$$

$$[\sigma_i, \sigma_j] = 2\epsilon_{ijk}\sigma_k \quad (\epsilon: \text{完全反対称テンソル})$$

$$X|i\rangle = |i \oplus 1\rangle, \quad Z|i\rangle = (-1)^i|i\rangle$$

XOR (mod2和)

○代表的な1量子ビットゲート

➤ アダマールゲート

$$H \equiv \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \quad \begin{aligned} HXH &= Z \\ HZH &= X \end{aligned}$$

➤ Sゲート (phaseゲート)

$$S = \begin{bmatrix} 1 & 0 \\ 0 & i \end{bmatrix} \quad \begin{aligned} SXS^\dagger &= Y \\ SZS^\dagger &= Z \end{aligned}$$

➤ Tゲート

$$T = \begin{bmatrix} 1 & 0 \\ 0 & \exp(i\pi/4) \end{bmatrix} \quad \begin{aligned} TXT^\dagger &= \frac{X+Y}{\sqrt{2}} \\ TZT^\dagger &= Z \end{aligned}$$

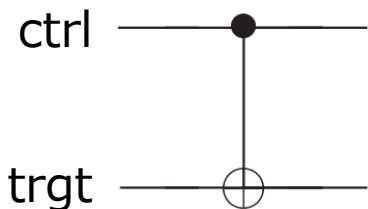
➤ Z軸回転ゲート

$$R_Z(\theta) = e^{-i\theta/2Z} = \begin{bmatrix} e^{-i\theta/2} & 0 \\ 0 & e^{i\theta/2} \end{bmatrix}$$

$\theta = \frac{\pi}{2}$ のものがSゲート, $\theta = \frac{\pi}{4}$ のものがTゲート

○代表的な2量子ビットゲート

➤ CNOTゲート



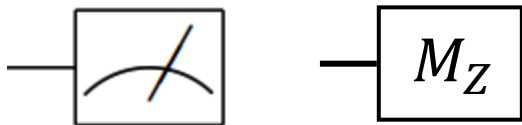
$$U_{CN} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

controlが $|0\rangle$ ならそのまま
controlが $|1\rangle$ ならtargetにX作用

$$U_{CN}|0\rangle_{ctrl}|i\rangle_{trgt} = |0\rangle_{ctrl}|i\rangle_{trgt}$$
$$U_{CN}|1\rangle_{ctrl}|i\rangle_{trgt} = |1\rangle_{ctrl}|i \oplus 1\rangle_{trgt}$$

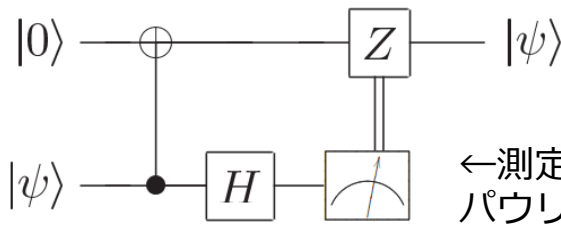
○測定

➤ パウリZ測定



メーター記号 or M_Z で表記

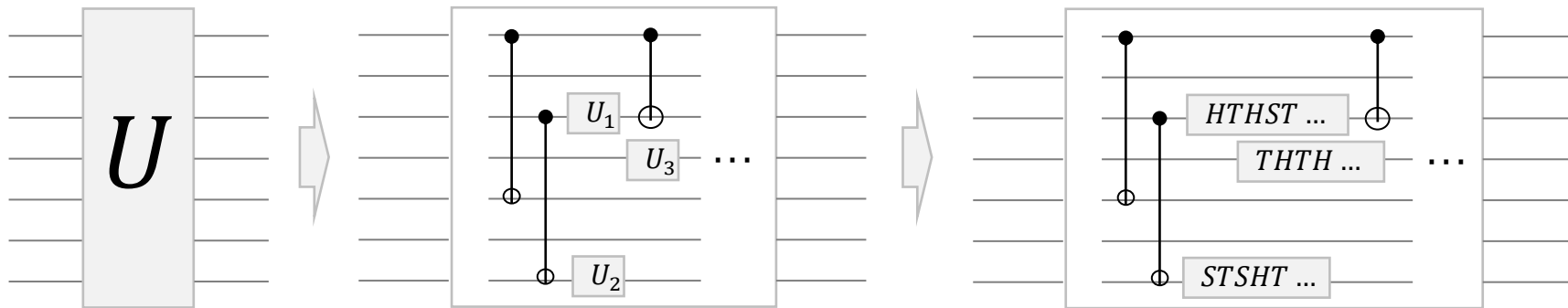
➤ 測定結果に応じたゲート操作(2重線で記述)



←測定値が-1の場合、追加でパウリZを作用

○量子ゲートのユニバーサリティ

- $\{CNOT, H, S, T\}$ が実行できるならば、任意のユニタリ演算子を任意の精度でシミュレートできることが知られている (ユニバーサリティ)



目的のユニタリ U

$CNOT$ と 1量子ビットユニタリ
(U_1, U_2, \dots) で分解できる

1量子ビットユニタリは $\{H, T, S\}$
を用いて任意精度で近似できる
(Solovay-Kitaev)

詳しい議論は : QIQC Chap.4, Appendix

量子エラー訂正

スタビライザー符号

簡単な例：反復符号

近年の主流：表面符号

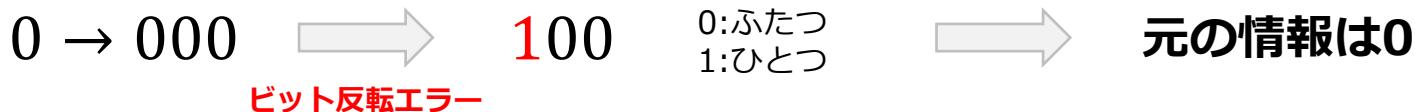
○古典ビットのエラー訂正の（きわめて簡単な）例：

ビット a を複数ビットにコピーし**符号化（冗長化）**することでエラー耐性を持たせる

0 → 000

1 → 111

符号化したビットのうちどれか1つが反転してしまった場合、各ビットの値を確認したうえで、多数決を取れば元のビットの情報を回復できる



各ビットに確率 p で独立にビット反転エラーが発生するとした場合、符号化によりエラー確率は $O(p^2)$ に削減される

○古典との違い：

- 任意の状態を複製して冗長化することができない (**no-cloning定理**)

$$|\psi\rangle \rightarrow \cancel{|\psi\rangle|\psi\rangle|\psi\rangle}$$

- 発生するエラーが連続的**（古典で起こるようなビット反転だけではなく、ユニタリ演算子で記述されるような量子エラーが起こる）
- 不用意に測定すると**量子的な重ね合わせが破壊されてしまう**

古典エラー訂正の枠組みをそのまま適用することはできない
そもそも連続的なエラーを訂正できるのか？

=> できる！ by P. Shor (1995)
これにより、量子エラー訂正の基本的な枠組みが確立

- 量子エラー訂正符号のほとんどは、**スタビライザー符号**と呼ばれるカテゴリに分類される
- そのためにまずは**スタビライザー形式**による量子状態の記述方法を導入する (Gottesman 1997)

- n 量子ビットパウリ群： n 個のパウリ演算子のテンソル積がなす群

$$G_n = \{\pm 1, \pm i\} \times \{I, X, Y, Z\}^{\otimes n}$$

- **スタビライザー群**：パウリ群の可換部分群で、 $-I$ を含まないもの

$-I$ を含まない理由は後述
(スタビライザー状態が自明にならないための条件)

例)

$$S = \{I_1 I_2, X_1 X_2, Z_1 Z_2, -Y_1 Y_2\} \quad (\text{負号は } XZ = -iY \text{ より})$$

$$S = \{III, ZZI, ZIZ, IZZ\} \quad (\text{以下、量子ビットを区別する添え字は基本省略})$$

○ **スタビライザー生成子** : スタビライザー群の独立な元の最大集合例)

$$S = \{II, XX, ZZ, -YY\} \text{ の生成子は } \langle XX, ZZ \rangle$$

$$S = \{III, ZZI, ZIZ, IZZ\} \text{ の生成子は } \langle ZZI, IZZ \rangle$$

※生成子の取り方には任意性がある(独立なものを取ってくればOK)

$$S = \{III, ZZI, ZIZ, IZZ\} \text{ の生成子は } \langle ZZI, IZZ \rangle \\ \langle ZZI, ZIZ \rangle \text{ でもOK}$$

※生成子は n 量子ビット系の場合高々 n 個

- **スタビライザー状態** : スタビライザー群 $S = \{S_i\}$ のすべての元 S_i に対して $S_i|\psi\rangle = +|\psi\rangle$ を満たす状態 $|\psi\rangle$
 - スタビライザー群は可換群なので、その元は同時対角化できる
 - スタビライザー生成子の同時固有状態であれば十分 (n 量子ビット系の場合、 n 個の演算子で状態が指定できる)
 - スタビライザー群に $-I$ が含まれてしまうと、スタビライザー状態は自明になってしまう ($-I|\psi\rangle = +|\psi\rangle$ より)

例)

$$S = \{II, XX, ZZ, -YY\}, \text{ 生成子 } \langle XX, ZZ \rangle$$

$$\text{スタビライザー状態は } \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$$

- 状態の発展は、スタビライザー生成子の変化で追いかける

$$USU^\dagger = S'$$

$$\begin{array}{ccc} \langle XX, ZZ \rangle & \xrightarrow{\text{CNOT}} & \langle IX, ZI \rangle \\ \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle) & & \frac{1}{\sqrt{2}}|0\rangle|+\rangle \end{array} \quad \begin{array}{l} U_{CN}XXU_{CN}^\dagger = IX \\ U_{CN}ZZU_{CN}^\dagger = ZI \end{array}$$

- 測定

- 測定する演算子がスタビライザー生成子と可換：測定値は+ 1
- 測定する演算子がスタビライザー生成子と反可換：反可換な演算子と入れ替え、測定値は ± 1 でランダムに定まる

$$\begin{array}{ccc} \langle XX, ZZ \rangle & \xrightarrow{IZ\text{測定}} & \langle \pm IZ, ZZ \rangle \\ \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle) & & |00\rangle \text{ or } |11\rangle \end{array}$$

- **スタビライザー符号** : スタビライザー群によって定義される部分空間に量子ビットの情報を埋め込む量子エラー訂正符号

例) 生成子 $\langle XXX, ZZI, IZZ \rangle$ で指定されるスタビライザー状態

$$|\psi\rangle = \frac{1}{\sqrt{2}} (|000\rangle + |111\rangle)$$

生成子 XXX を一つ取り除いて、縮退を導入してみる :

XXX の固有値は ± 1 なので、2重縮退 = 1量子ビット分の自由度が入る

生成子 $\langle ZZI, IZZ \rangle$ で指定されるスタビライザー状態は1つの量子ビットとみなすことができる :

$$\begin{array}{ccc} & |\phi\rangle = \alpha|000\rangle + \beta|111\rangle & \\ \text{論理量子ビット} & & \text{物理量子ビット} \end{array}$$

- **論理演算子**：論理量子ビットに作用する演算子
- スタビライザー生成子と可換かつ独立なものを取ってくる

例) 生成子 $\langle ZZI, IZZ \rangle$ で指定されるスタビライザー状態は1つの量子ビットとみなすことができる：

$$|\phi\rangle_L = \alpha|0\rangle_L + \beta|1\rangle_L \quad \begin{array}{l} |0\rangle_L = |000\rangle \\ |1\rangle_L = |111\rangle \end{array}$$

この場合の論理演算子は…

$$\begin{aligned} X_L &= XXX, \\ Z_L &= ZZZ \end{aligned}$$

$$\begin{aligned} X_L|0\rangle_L &= |1\rangle_L, \\ X_L|1\rangle_L &= |0\rangle_L \end{aligned}$$

$$\begin{aligned} Z_L|0\rangle_L &= |0\rangle_L, \\ Z_L|1\rangle_L &= -|1\rangle_L \end{aligned}$$

n 量子ビット系において、 $n - k$ 個の生成子からなるスタビライザー群 $S \in G_n$ で定義されるスタビライザー符号は **k 量子ビット分の自由度 (論理量子ビット) を持つ**

- 元々 n 個の生成子で一意に定まっていたスタビライザー状態から、 k 個の生成子を除いて縮退を導入した結果と考えるとわかりやすい (個人的には)
- 縮退をほどく (論理量子ビットに作用する) 演算子を論理演算子と呼ぶ

別の解釈：

- 生成子の固有値は ± 1 なので、 $n - k$ 個の固有値の組み合わせは 2^{n-k} 通り
- 固有値の組み合わせにより 2^n 次元ヒルベルト空間が 2^{n-k} 分割され、結局スタビライザー状態は 2^k 次元部分ヒルベルト空間の元となる => **論理自由度**

	符号空間	$ZZI = +1$	$ZZI = -1$
$IZZ = +1$		$\{ 000\rangle, 111\rangle\}$	$\{ 100\rangle, 011\rangle\}$
$IZZ = -1$		$\{ 001\rangle, 110\rangle\}$	$\{ 010\rangle, 101\rangle\}$

- 量子エラー訂正符号として機能するためにはスタビライザーをうまくとってくる必要がある
- どうやって取るか? => **古典エラー訂正符号の知見を借りる**
- CSS符号：2つの古典線形符号から構成された量子エラー訂正符号**
 - 2つはそれぞれXエラー、Zエラーの検知・訂正に用いられる

○ $[n, k]$ 古典線形符号

$$H \equiv \begin{bmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \end{bmatrix}$$

$$Hx = 0$$

$$G = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \quad \begin{array}{l} G(0) = (0,0,0) \\ G(1) = (1,1,1) \\ HG = 0 \end{array}$$

パリティチェック行列
を与えて定義 $((n-k) \times$
 n 行列)

H のカーネルの元が符号語
 $x = (0,0,0), (1,1,1)$

生成行列 G を与えてもよい
 $(n \times k$ 行列)

- ✓ エラー検知: $y = x + e, Hy = He \neq 0$
- ✓ ある古典線形符号 (H, G) が与えられたとき、その双対符号はパリティチェック行列 G^T , 生成行列 H^T で定義される

○CSS符号の構成

以下を満たす2つの古典線形符号 C_1, C_2 を持つてくる：

- それぞれ $[n, k_1], [n, k_2]$
- $C_2 \subset C_1$
- C_1, C_2^\perp はそれぞれ t 個のエラーを訂正できる

n 量子ビット系を考え、 X, Z スタビライザー生成子を C_1, C_2^\perp のパリティチェック行列を用いて定義する（古典線形符号と同様に t 個の X, Z エラーを訂正できる）

$$H \equiv \begin{bmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \end{bmatrix} \begin{matrix} \Rightarrow S_0 = ZZI \\ \Rightarrow S_1 = IZZ \end{matrix}$$

$$C_2 \subset C_1 \text{ より、 } H(C_2^\perp)H(C_1)^T = 0$$

(X, Z スタビライザーの可換性)

$$\begin{aligned} \because H(C_2^\perp)H(C_1)^T &= [H(C_1)H(C_2^\perp)^T]^T \\ &= [H(C_1)G(C_2)]^T = 0 \end{aligned}$$

○論理自由度

- ここで定義したスタビライザー生成子の数は $n - k_1 + k_2$ 個
- n 量子ビットを用いて符号化
- 残る自由度は $2^{n-(n-k_1+k_2)} = 2^{k_1-k_2}$ ($k_1 - k_2$ 論理自由度を持つ)

○論理演算子

- パリティチェック行列を標準形に変形して読み取れる (詳細は割愛、QCQI chap 10.5.7など参照)

- スタビライザー群：パウリ群の可換部分群かつ $-I$ を含まないもの
- スタビライザー状態：スタビライザー群の固有値 $+1$ 同時固有状態
- スタビライザー符号：スタビライザー状態に論理自由度を埋め込む量子エラー訂正符号

次ページから、簡単なスタビライザー符号（反復符号）を題材に、エラー訂正がどのように実行されるのかを議論していく

- 簡単な例として**3量子ビット反復符号**を考える
- スタビライザー生成子は $\langle Z_0 Z_1 I_2, I_0 Z_1 Z_2 \rangle$ (以下 I は省略)

$$\begin{array}{l} \begin{array}{ccc} 0 & 1 & 2 \\ \downarrow & \downarrow & \downarrow \end{array} \\ |0\rangle_L = |000\rangle & X_L = X_0 X_1 X_2 \\ |1\rangle_L = |111\rangle & Z_L = Z_0 Z_1 Z_2 \end{array}$$

以下、量子エラーとして確率 p でパウリ X 演算子が作用するもの考える：

$$\mathcal{E}(\rho) = (1 - p)\rho + p X\rho X$$

この下でどのようにエラー訂正が行われるか議論していく
(より複雑なエラーに対する耐性を持つ符号は後ほど議論)

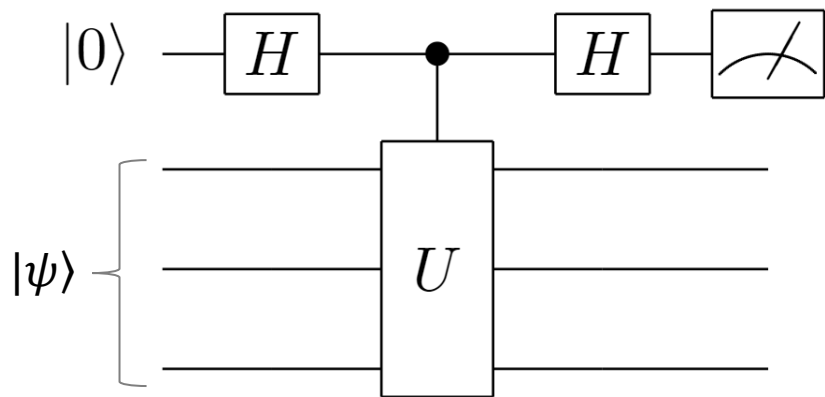
- エラーの検知：スタビライザー演算子の固有値（シンドローム）を測定し、スタビライザー状態からのずれを検知する

	$s_0 = +1$	$s_0 = -1$
$s_1 = +1$	$\{ 000\rangle, 111\rangle\}$ エラーなし	$\{ 100\rangle, 011\rangle\}$ X_0 エラー
$s_1 = -1$	$\{ 001\rangle, 110\rangle\}$ X_2 エラー	$\{ 010\rangle, 101\rangle\}$ X_1 エラー

$s_0 = Z_0 Z_1$ の固有値

$s_1 = Z_1 Z_2$ の固有値

- シンドローム測定はアダマールテスト回路を応用して実行する
- アダマールテスト回路：



測定直前の状態は、

$$\frac{1}{2}|0\rangle(|\psi\rangle + U|\psi\rangle) + \frac{1}{2}|1\rangle(|\psi\rangle - U|\psi\rangle)$$

$|\psi\rangle$ ：スタビライザー状態

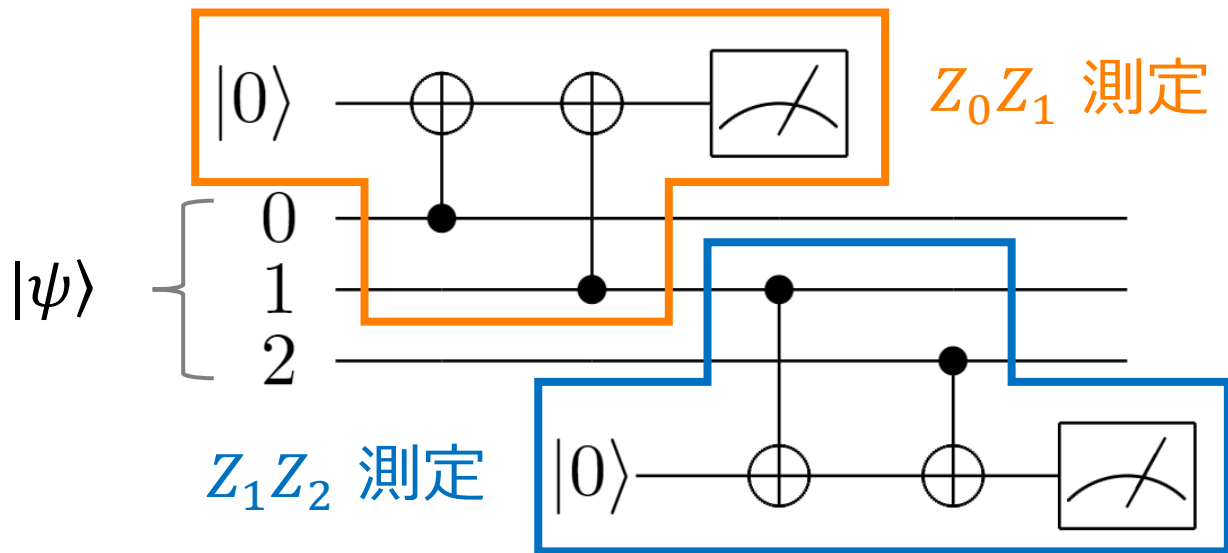
U ：スタビライザー生成子の一つ
とすれば、

エラーの無い場合+1が測定される

エラーを検知すると-1が測定される

簡単な例：反復符号

3量子ビット反復符号の場合：



注) シンドロームを得るために測定する演算子は、スタビライザー符号の定義から**論理量子ビットの状態を壊さない**

スタビライザー演算子：

$$S_0 = Z_0 Z_1,$$
$$S_1 = Z_1 Z_2$$



互いに可換
かつ独立

論理演算子：

$$X_L = X_0 X_1 X_2,$$
$$Z_L = Z_0$$

- ✓ 不用意に測定すると状態を壊してしまう量子の場合でも、状態を壊すことなくエラー検知が可能になっている

簡単な例：反復符号

- エラーの訂正：得られたシンドロームの情報からエラーの位置を特定し、対応するパウリ演算子で相殺することで訂正

$$|0\rangle_L = |000\rangle$$

$s_0 = Z_0 Z_1$ の固有値
 $s_1 = Z_1 Z_2$ の固有値

	$s_0 = +1$	$s_0 = -1$
$s_1 = +1$	$\{ 000\rangle, 111\rangle\}$ エラーなし	$\{ 100\rangle, 011\rangle\}$ X_0 エラー
$s_1 = -1$	$\{ 001\rangle, 110\rangle\}$ X_2 エラー	$\{ 010\rangle, 101\rangle\}$ X_1 エラー

例)

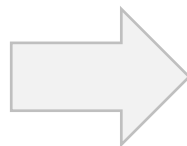
$$|0\rangle_L = |000\rangle$$

$$s_0 = +, s_1 = +$$



$$|100\rangle$$

$$s_0 = -, s_1 = +$$



$$|000\rangle$$

$$s_0 = +, s_1 = +$$

X_0 エラーと判定
 X_0 作用

○エラー箇所之最尤推定と符号距離

注) 実は、シンδροームの値では区別できない別のエラーパターンも存在する：

$$|100\rangle$$

$$s_0 = -, s_1 = +$$

$$|011\rangle$$

$$s_0 = -, s_1 = +$$

基本的に我々が知りえるのは測定で得られるシンδροームの値だけなので、エラー訂正はこれら区別できないエラーパターンのうち、**最も起こりやすいものを採用して行う（エラー箇所之最尤推定）**

発生確率の高い
こちらと解釈

$$|100\rangle$$

$$s_0 = -, s_1 = +$$

$$p(1-p)^2$$

>

$$|011\rangle$$

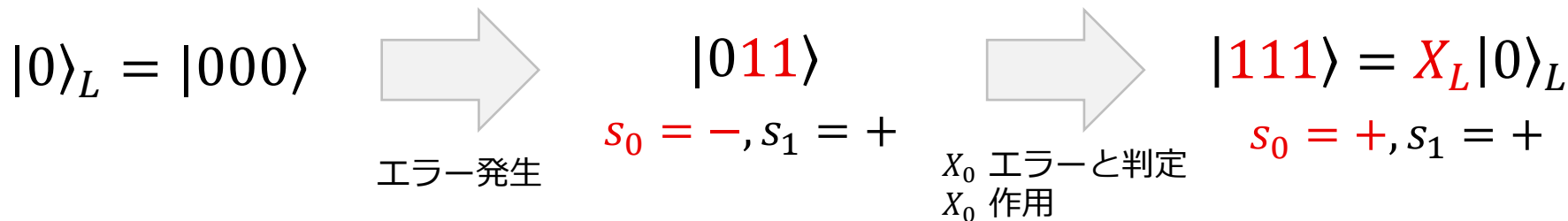
$$s_0 = -, s_1 = +$$

$$p^2(1-p)$$

尤度関数は
エラー発生確率

簡単な例：反復符号

$O(p^2)$ の確率でエラー訂正に失敗してしまう場合も出てくる（論理エラー）



量子エラー訂正符号がどれだけのエラーを許容できるのかを示す値が**符号距離 d**

$d_X = \min wt(X_L)$ $wt(X_L)$: 論理X演算子のウェイト (作用する物理量子ビットの数)

$d_Z = \min wt(Z_L)$ $wt(Z_L)$: 論理Z演算子のウェイト (作用する物理量子ビットの数)

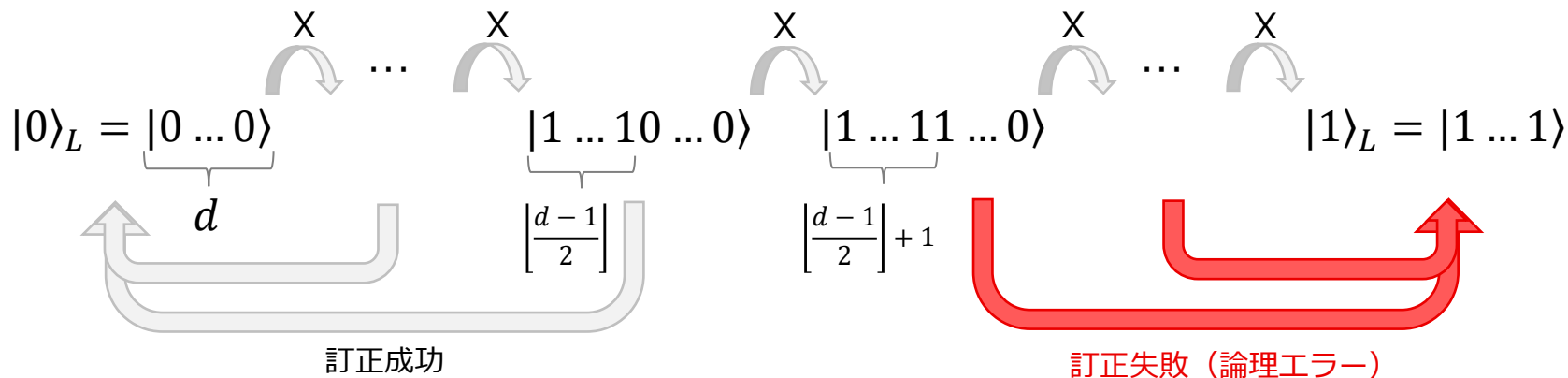
$$d = \min \{d_X, d_Z\}$$

符号距離 d の符号は、 $\lfloor \frac{d-1}{2} \rfloor$ 個のエラーを訂正できる ($\lfloor \cdot \rfloor$: 床関数)

簡単な例：反復符号

3量子ビット反復符号の場合： $d_X = 3, d_Z = 1$

- Xエラーは $\lfloor \frac{3-1}{2} \rfloor = 1$ つまで訂正可能、Zエラーは訂正不可能
- 訂正したいエラーに応じて、適切な d_X, d_Z を持つ符号を選択する
- 許容できるエラーの個数 $\lfloor \frac{d-1}{2} \rfloor$ の直感的な理解：最短距離復号で正しく訂正できる限界



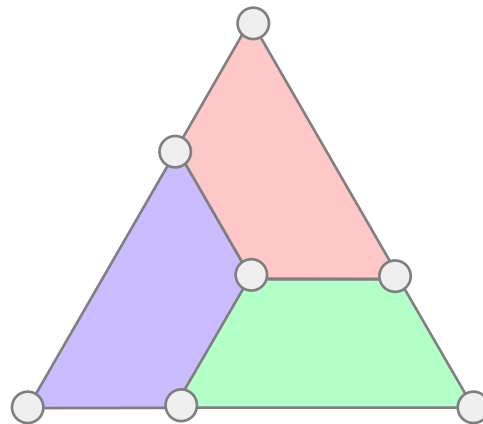
- 非自明かつ比較的簡単な量子エラー訂正符号の例：**Steane符号**
 - 7つの物理量子ビットから構成される符号、符号距離3（任意の1つのパウリエラーを訂正できる）

スタビライザー：

Name	Operator
g_1	$IIIXXXX$
g_2	$IXXIIXX$
g_3	$XIXIXIX$
g_4	$IIIZZZZ$
g_5	$IZZIIZZ$
g_6	$ZIZIZIZ$

論理演算子：

$$X_L = XXXXXXX$$
$$Z_L = ZZZZZZZ$$



いわゆる色符号(color code)と呼ばれる符号の一例

今まで見てきた例では、単純なパウリエラーに対する耐性しか議論してこなかったが、パウリエラーで記述できないアナログなエラーが起こった場合にはどうなるのか？

実は、パウリエラーが訂正できれば、任意の1量子ビットエラーも訂正できることが示せる！（エラーの離散化）

歴史的には、量子コンピュータがそれまで考えられてきた（実現不可能な）アナログコンピュータと異なることを示した重要な性質

簡単な例)

$$|0\rangle_L = |000\rangle \xrightarrow{\text{微小角 } (\delta) \text{ の回転}} e^{-i\delta X_0} |000\rangle \approx |000\rangle - i\delta |100\rangle$$

シンδροーム測定を実行すると、重ね合わさっている異なるシンδροームパターン状態のいずれかに射影される

=> パウリエラーとして訂正できる！（エラーの離散化）

$$|000\rangle - i\delta |100\rangle \begin{cases} \rightarrow |000\rangle & s_0 = +, s_1 = + \\ \rightarrow -i\delta |100\rangle & s_0 = -, s_1 = + \end{cases}$$

- より一般には、密度演算子による定式化によって以下のように記述できる：
- 1量子ビットエラーは密度演算子に作用するCPTP写像として一般に以下のように書ける：

$\sum_k E_k E_k^\dagger \leq 1$ を満たす演算子 (Kraus演算子) の組 $\{E_k\}$ を用いて

$$\mathcal{E}(\rho) = \sum_k E_k \rho E_k^\dagger$$

- 各Kraus演算子 E_k はパウリ行列と単位行列を用いて分解できる：

$$E_k = c_k^1 I + c_k^2 X + c_k^3 Y + c_k^4 Z$$

$$\text{ここで } c_k^i = \text{tr}(E_k P_i), P_i = \{I, X, Y, Z\}$$

- 今、ある n 量子ビットで符号化された状態 $|\psi\rangle$ とエラー訂正操作 R が以下のように与えられたとする：

$$RP_i|\psi\rangle = |\psi\rangle, \quad P_i = X_i, Y_i, Z_i, (i = 0, \dots, n)$$

- エラー訂正操作との合成チャンネルを考えると

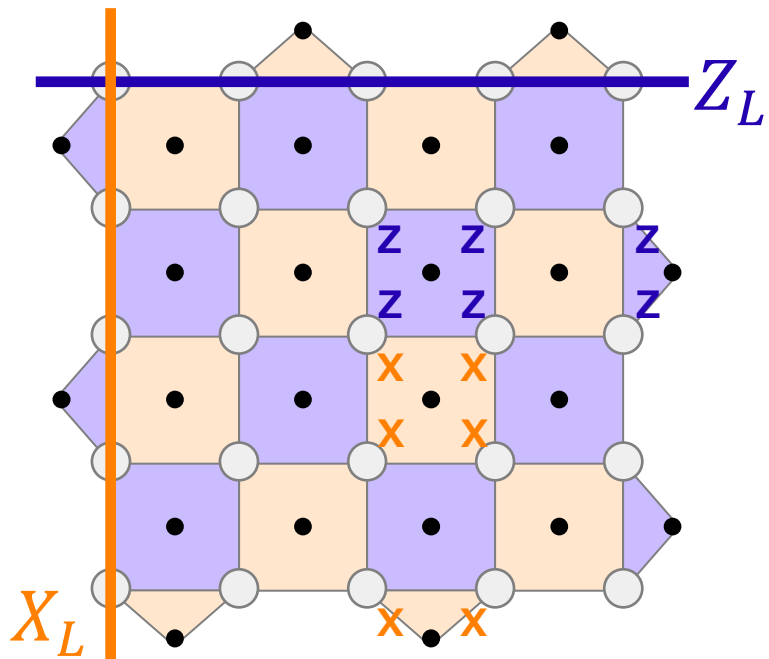
$$\mathcal{R} \circ \mathcal{E}(|\psi\rangle\langle\psi|) = \sum_k RE_k|\psi\rangle\langle\psi|E_k^\dagger R^\dagger \propto |\psi\rangle\langle\psi|$$

$$(\because R(c_0I + c_1X + c_2Y + c_3Z)|\psi\rangle \propto |\psi\rangle)$$

任意の1量子ビットエラーを訂正できる！

- 量子コンピュータの実現に向けて、近年有力視されている量子エラー訂正符号の一つが**表面符号**
- いわゆる**トポロジカル符号の一派**
 - トポロジカル符号…符号ハミルトニアン基底状態（ないし低エネルギー部分空間）がトポロジカル秩序を示すもの
- 性質：
 - ✓スタビライザーが重み4(or2)のパウリ演算子で局所的
 - ✓2次元最近接格子に配置された量子ビットで実現できる（量子コンピュータ実機で実装しやすい）
 - ✓量子エラー訂正の性能が高い

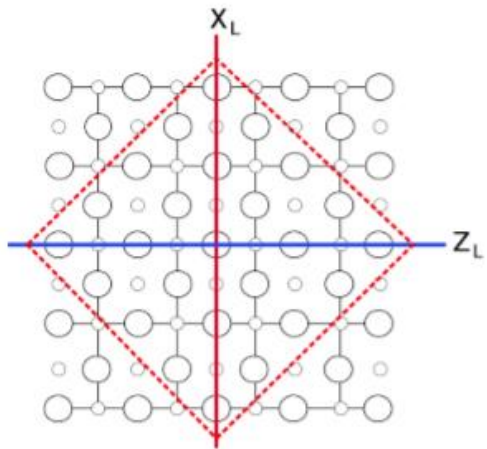
(回転) 表面符号： $d \times d$ 格子にデータ量子ビットを配置し(○)符号化する
スタビライザーは格子面に定義される (Z：青、X：黄色)



- ✓ 1つの $d \times d$ 格子に1つの論理自由度
- ✓ 論理演算子：境界上に定義
青：論理Z演算子、橙：論理X演算子
- ✓ 符号距離：格子の辺上にある量子ビット数 (d)

(補足) Toric codeとの関係

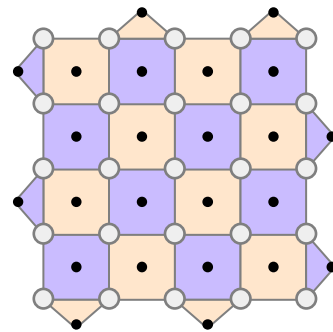
- 代表的なトポロジカル符号であるtoric codeと主な構造は共通
- 違い：
 - トーラスではなく境界のある平面上で定義されている（実機実装が容易）
 - さらに符号の一部を削り、45度回転させている（必要量子ビット数削減）



平面符号(Horsman (2012)より)



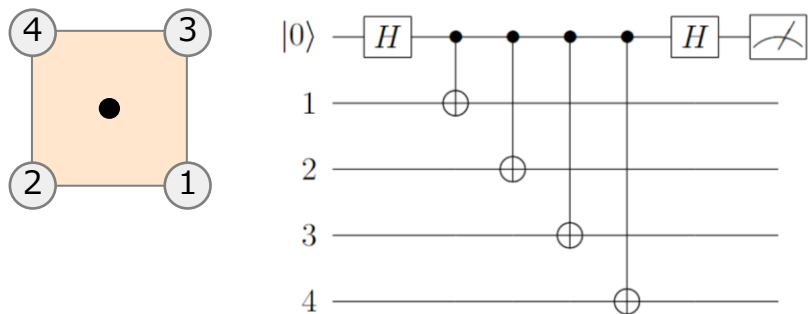
赤点線の外を除外
45度回転



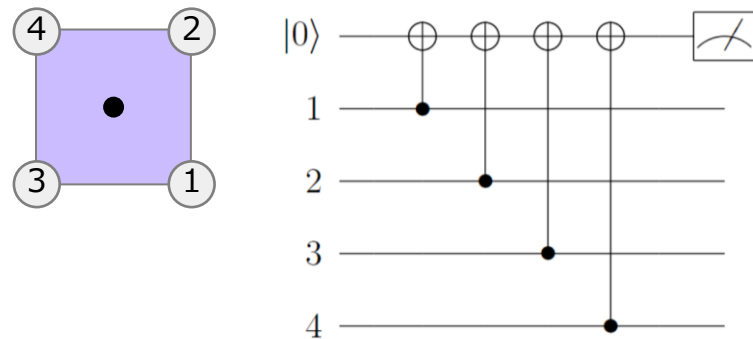
回転表面符号

シンドローム測定は、格子面に配置された測定量子ビットを用いて行う

➤ Xシンドローム測定回路

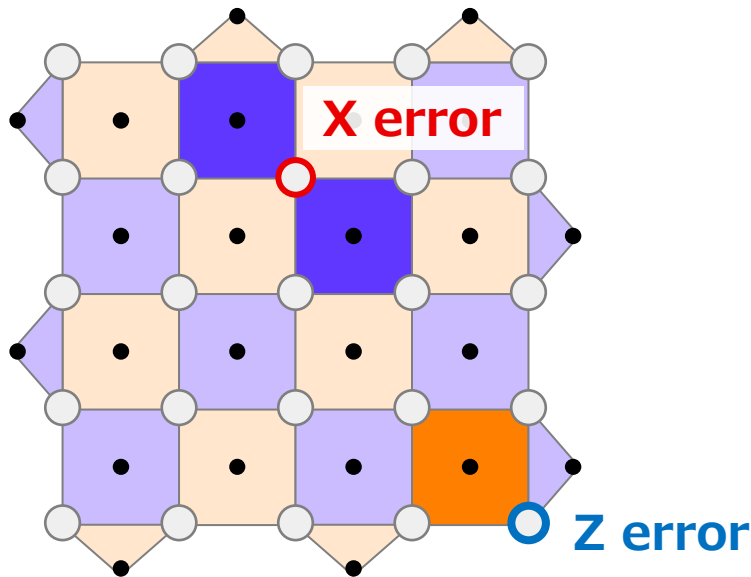


➤ Zシンドローム測定回路



※○内の数字はCNOTを作用させる順番

測定されたシンドロームの値から、エラー発生を検知することが可能



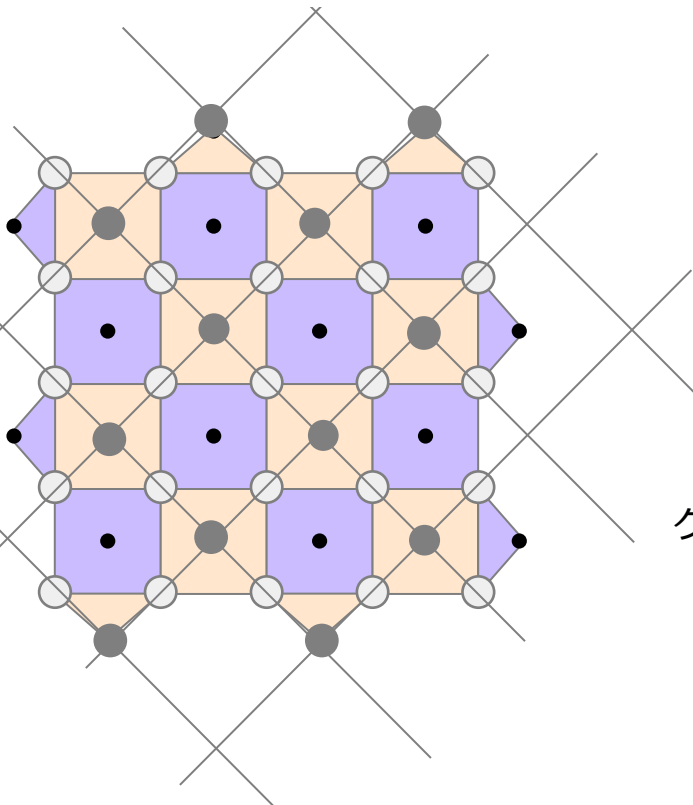
シンドロームは、（境界を除いて）エラー箇所を挟むように反転する

- 一般に、シンドローム値からエラー箇所の推定を行うことは難しい
 - 与えられた符号に対し、いかに効率良く、精度よくエラー推定を行うかを追求する取り組みは、量子エラー訂正符号研究の一つの大きな柱
- 表面符号では、エラー推定問題をグラフのマッチング問題に帰着させる**最小重み完全マッチング (Minimum Weight Perfect Matching, MWPM)** アルゴリズムが知られている

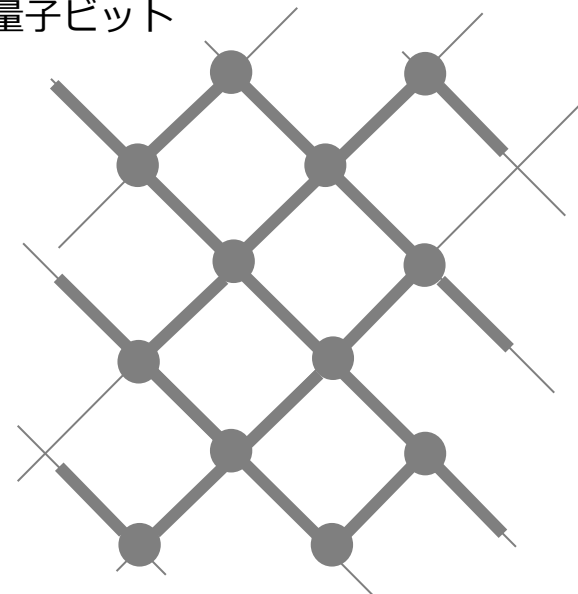
以下、MWPMを用いたZエラー用最尤推定を考える

①デコードグラフの構成：符号の定義に用いている格子の双対格子を取る

頂点：スタビライザー（シンドローム）
辺：量子ビット

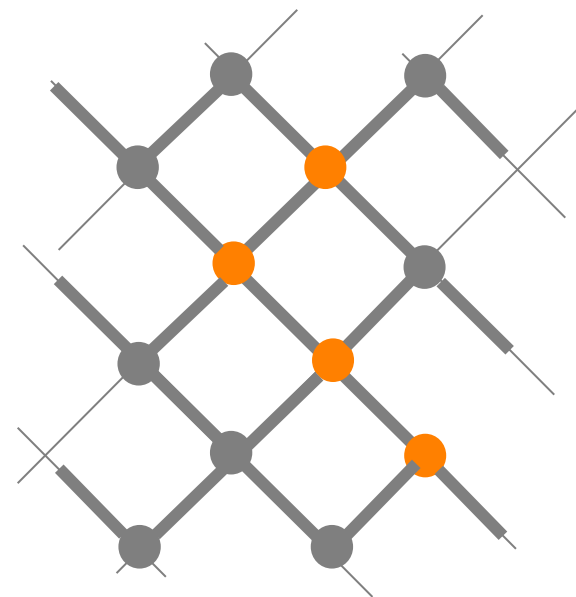
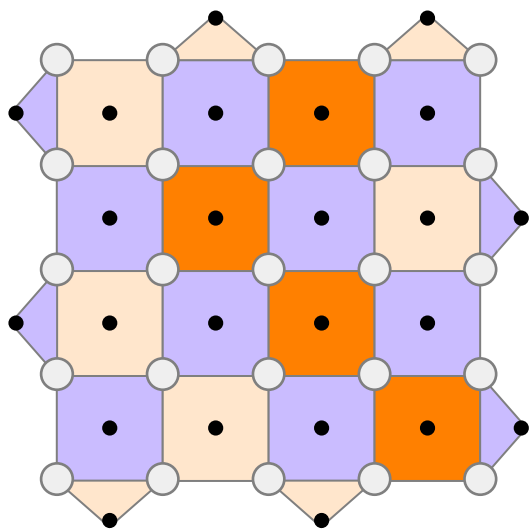


グラフ抽出



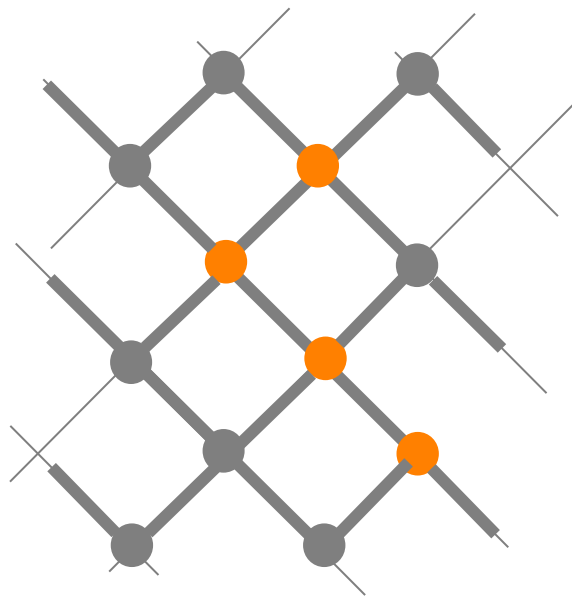
デコードグラフ(Xシンドローム)

②反転したシンδροームに該当するデコードグラフのノードを割り出す

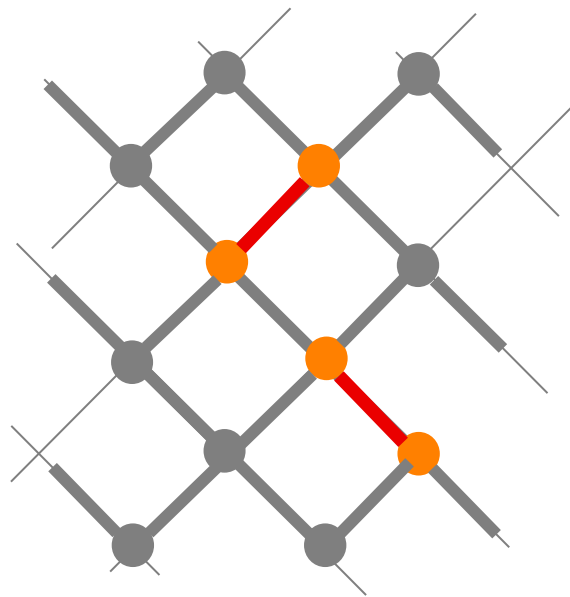


デコードグラフ(Xシンδροーム)

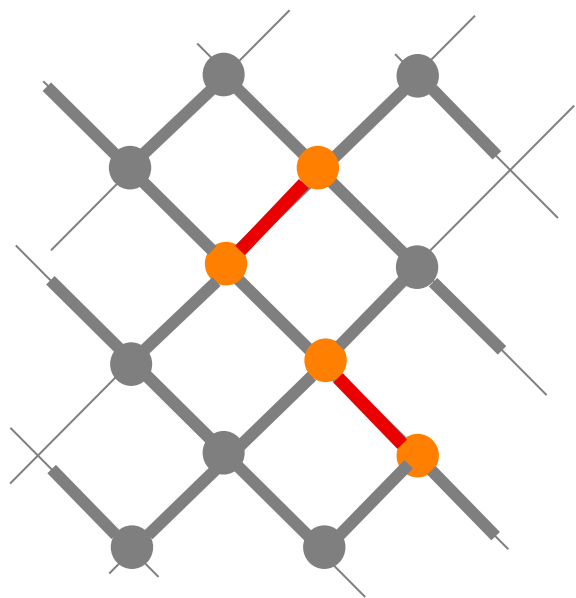
③反転したシンδροームに該当するノード同士を、デコードグラフ上で過不足なく、重複しない最短経路でつなぐ辺を特定する (**MWPM**)



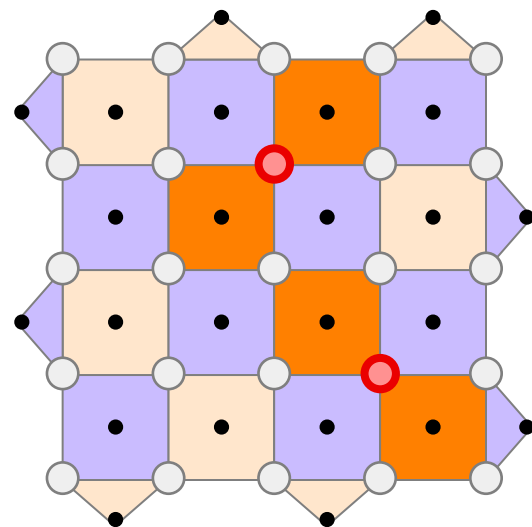
マッチング実行



- ④決定された最短経路に含まれる辺から、該当する量子ビットを特定し、それを推定されたエラー箇所とする



エラー箇所特定



- 最短経路 = 辺の数が最小の経路、すなわちエラーの数が最小
- 各エラーが独立に確率 p で起こるとすると、最も起こりやすいエラーパターンはエラー数最小のもの
- したがって、MWPMによる推定結果は最尤推定結果に他ならない

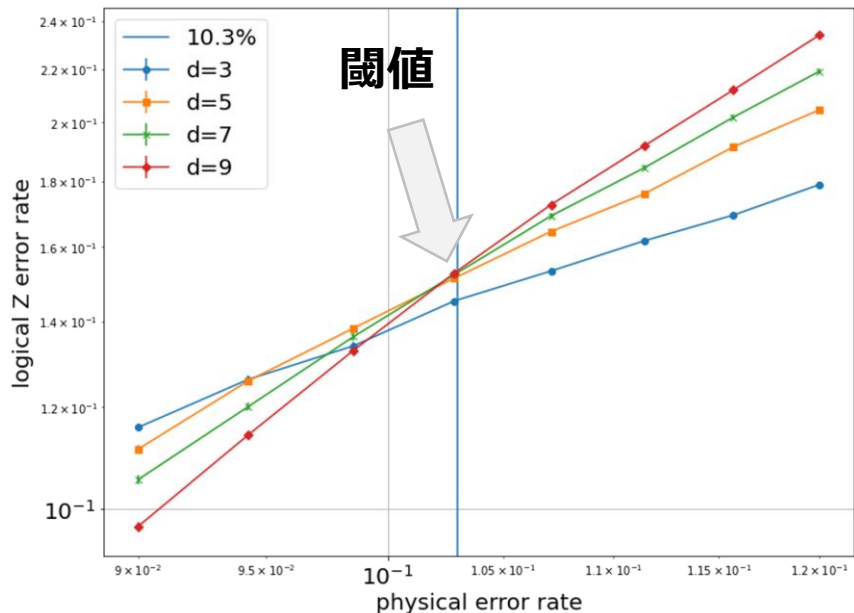
$$p(E) \propto \left(\frac{p}{1-p} \right)^H \quad \begin{array}{l} E: \text{エラー鎖} \\ H: \text{エラー数} \end{array} \quad L(E) = -\log p(E)$$

あるエラーパターン E が発生する確率

MWPMによる推定に対応する（対数）尤度関数

※エラー推定アルゴリズムは他にも様々提案されている（イジングモデルのエネルギー最小化問題にマッピング、機械学習を組み合わせる、etc…）

- 量子エラー訂正符号の研究では、数値計算により符号のエラー訂正性能を評価する

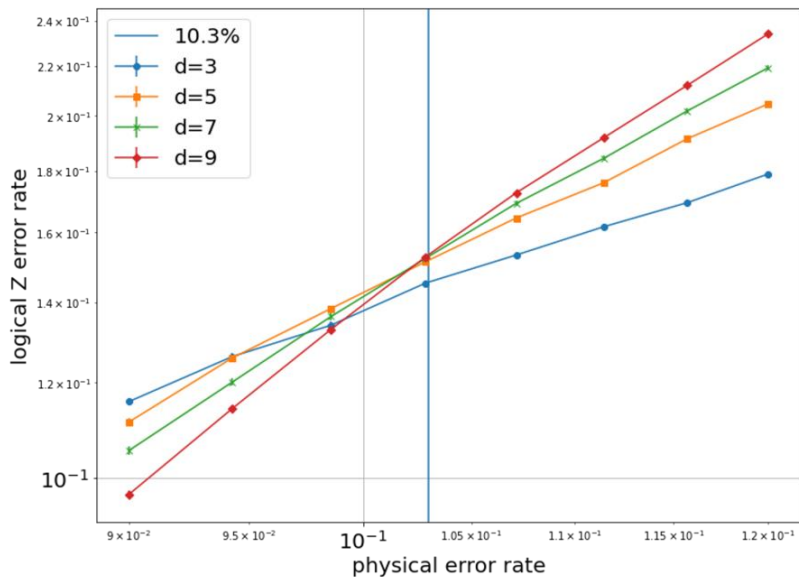


左図の数値計算例

- ✓ 物理量子ビットに独立にX, Zエラーが確率 p で発生すると仮定
- ✓ ランダムにエラーをサンプリングし、各サンプルでエラー訂正を実際に行い、論理エラー確率を計算
- ✓ 色々な p の値で繰り返し、プロット

ある値（閾値）を境に、符号距離を伸ばすほど論理エラーが抑制されるというふるまいが見られる

○ 符号距離 d による論理エラー率のスケールリング



閾値(p_{th})以下では、

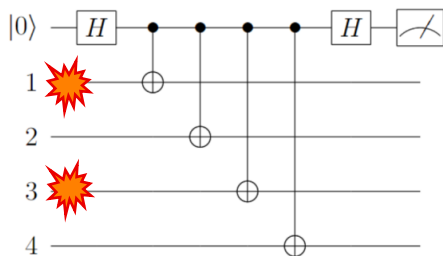
$$P_L \approx C \left(\frac{p}{p_{th}} \right)^{d_e}$$
$$d_e = \frac{d+1}{2}$$

エラー訂正と統計力学モデルに対応あり
(random-bond Ising modelなど)
閾値は対応する統計力学モデルにおける
臨界温度に相当

○性能評価で用いられるノイズモデルは3種類

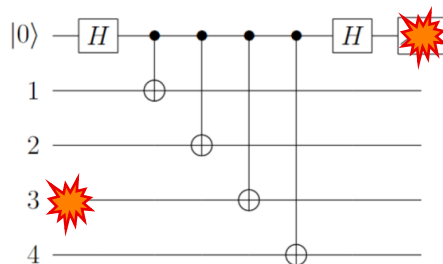
- エラーはすべて独立、確率的にパウリエラーが起こるとしてモデル化

符号容量ノイズ (code capacity)



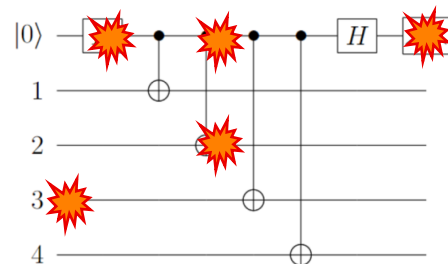
データ量子ビットのみ
エラー発生

現象論的ノイズ (phenomenological)



データ量子ビットおよび
測定にエラー発生

回路レベルノイズ (circuit level)



初期化、ゲート操作、測
定、アイドルングすべて
でエラー発生

- これまでの議論は、測定されたシンドローム値が正しいものとしてそこからエラー箇所推定を行った
- しかし、**実際の量子コンピュータにおける量子ビットの測定はエラーを伴う**
- 測定にもエラーがある場合、**シンドローム測定を複数回行い、それらの差分を用いて3次元的にエラー箇所推定を実行する**

- 差分シンδροーム ($t - 1$ 回目と t 回目のシンδροームの差分)

$$s^{(t)} \oplus s^{(t-1)} = \left(\bigoplus_{i \in \partial s} e_i^{(t-1)} \right) \oplus e_{\text{meas}}^{(t)} \oplus e_{\text{meas}}^{(t-1)}$$

- この差分シンδροームは以下のエラーに反応し反転する

- $t - 1$ 回目に起こったのシンδροーム測定エラー $e_{\text{meas}}^{(t-1)}$

- t 回目に起こったのシンδροーム測定エラー $e_{\text{meas}}^{(t)}$

- $t - 1$ 回目と t 回目の間に発生した物理量子ビットのエラー $\bigoplus_{i \in \partial s} e_i^{(t-1)}$

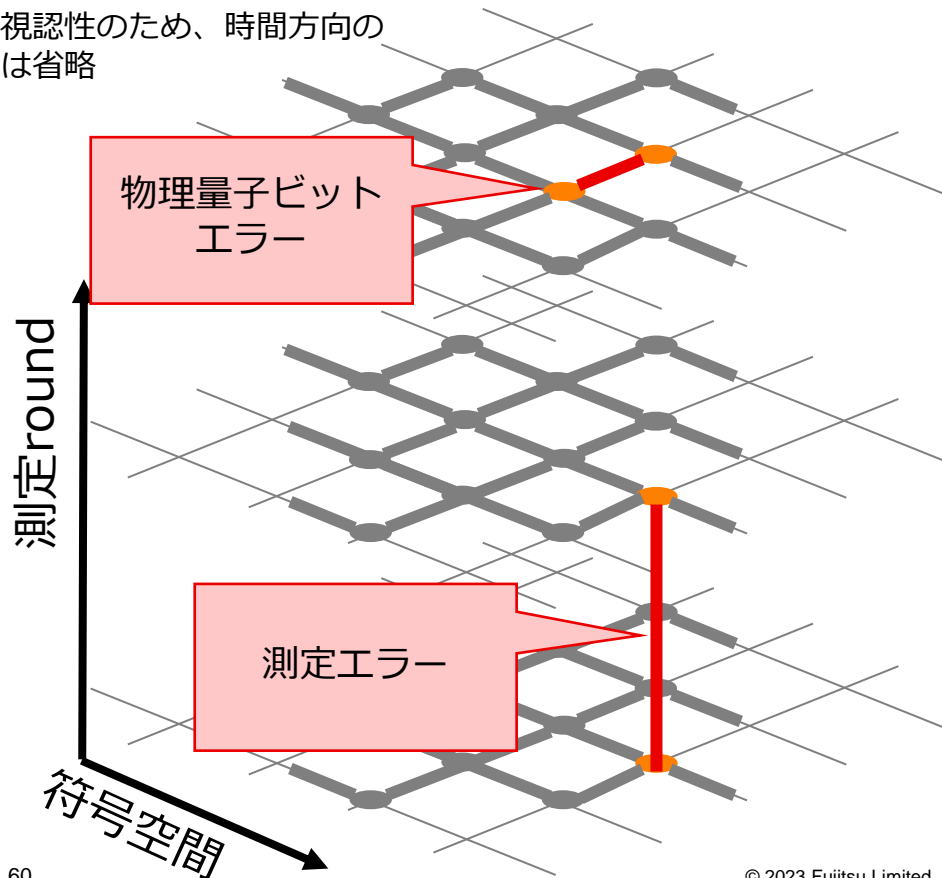
(発展) 測定エラーのある場合のエラー訂正

差分シンδροームを頂点、物理量子ビットおよび測定エラー箇所を辺に持つ3次元的なデコードグラフを構成 (右図)

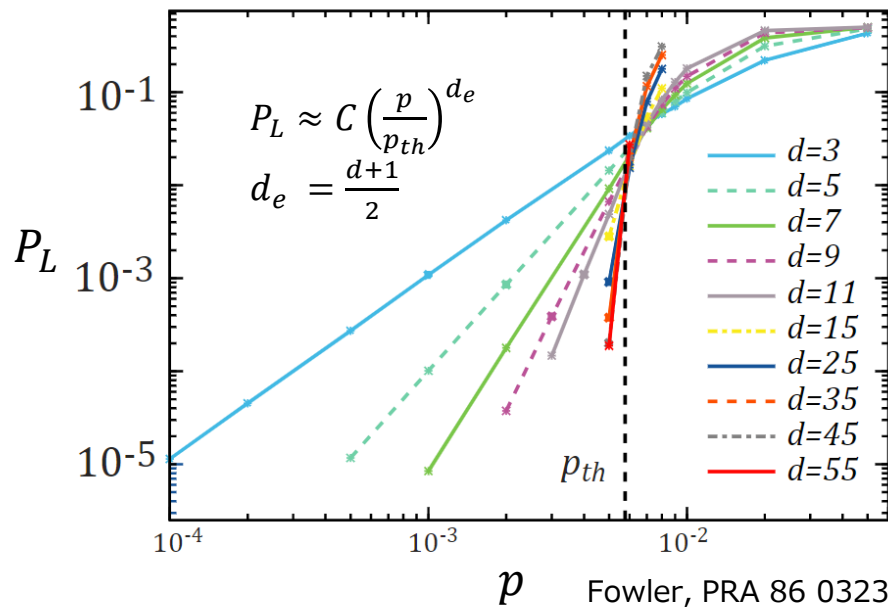
(空間方向は今までのデコードグラフと同様、時間方向は同じ頂点同士をつなぐ)

反転した差分シンδροームをMWPMでマッチングすることで、測定エラーと物理量子ビットエラーの箇所を特定する (右図赤)

※視認性のため、時間方向の辺は省略



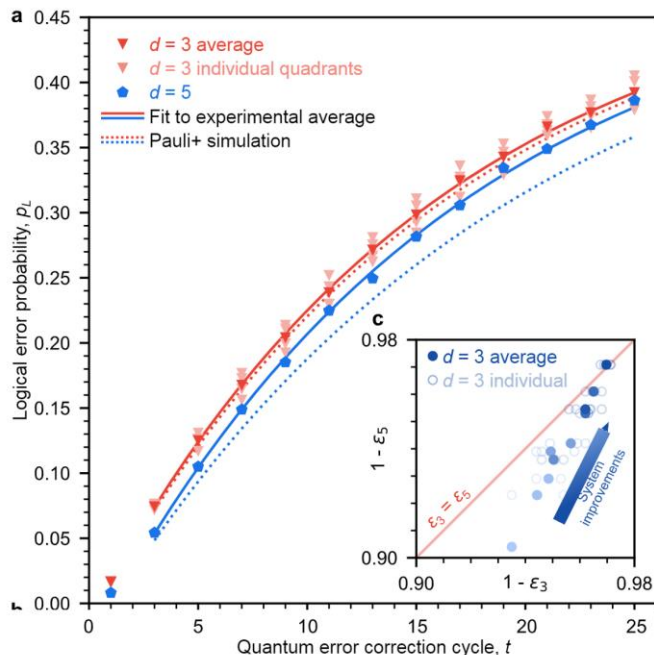
回路レベルノイズモデルでの性能評価例



符号距離が長いほど性能が良くなる
閾値の値：0.5%~0.75%

量子エラー訂正符号の閾値は、後ほど議論するFTQCの閾値定理において重要な役割を果たす

○最近の事例：Googleによる超伝導デバイスを用いた検証(2023)



符号距離3, 5の表面符号を実機実装し、論理エラー確率のふるまいを調べた

若干ではあるが、符号距離を伸ばすことで論理エラー確率が抑制されたと報告

Google Quantum AI, Nature 614 (2023)より

- スタビライザー符号の基礎事項
 - スタビライザー群で指定されるスタビライザー状態を符号語として用いる
- 簡単な例
 - 3量子ビット反復符号：1つのXエラーを訂正可能な符号
 - スタビライザー演算子の固有値測定によるエラー検知
 - エラー箇所の最尤推定と符号距離
 - Steane符号：1量子ビットエラーを訂正可能な符号
 - エラーの離散化：パウリエラーが訂正できればアナログエラーも訂正可能
- 最近の主流：表面符号
 - 諸々の定義
 - エラー箇所の最尤推定アルゴリズム：MWPM法
 - 最近の実験報告（Google）

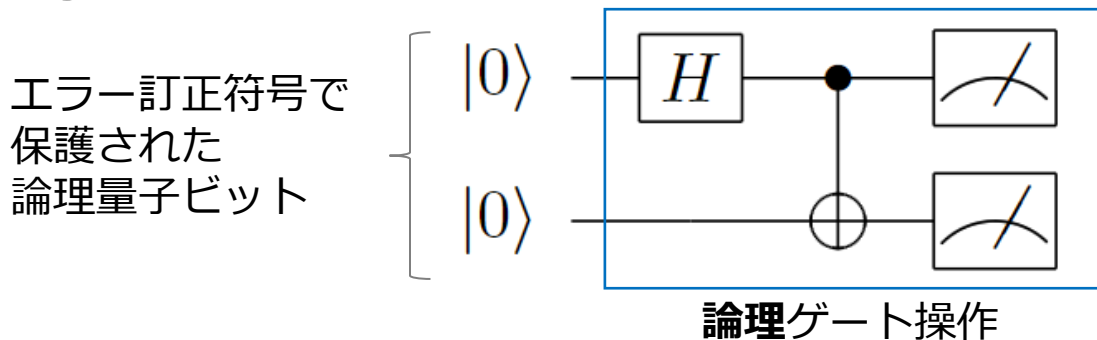
誤り耐性量子計算

概要

論理ゲート操作の実装方法

近年の主流：Lattice surgeryによる論理ゲート実装

- これまでに、エラー耐性を持つ単一の量子ビット（論理量子ビット）をどう構成するか、エラー訂正をどう行うかを議論してきた
- ここからは、論理量子ビットにゲートを作用させ、計算を行うことを考える



量子エラー訂正により論理エラーを抑えながら計算 = 誤り耐性量子計算 (Fault-Tolerant Quantum Computing, FTQC)

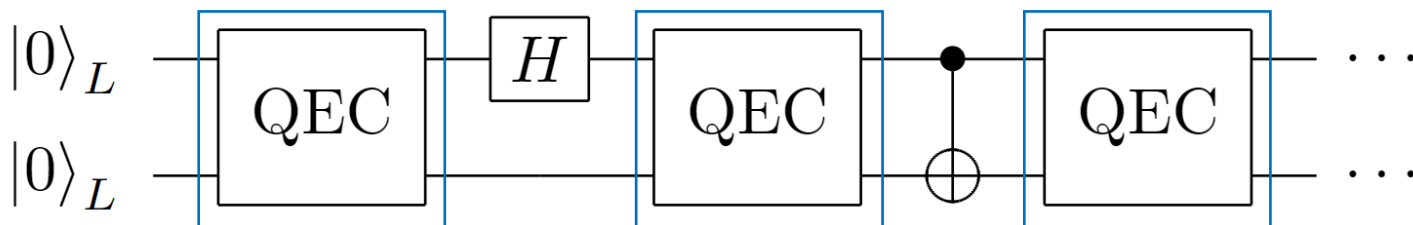
- このようなことが実現可能であることを示す定理として、閾値定理が知られている

閾値定理(ざっくり版)

量子計算機がある閾値より小さいエラー率で動作するとき、任意の量子回路が任意の精度で効率的に実行できる

- ✓ 量子計算において不可避であるエラーも、ある閾値より低く抑えることさえできればもはや問題ではなくなる！
- ✓ 量子コンピュータ実現のための拠り所

- 論理量子ビットの符号化を解除することなくそのまま計算を実行
- 計算の途中でエラー訂正を実施し、エラーの蓄積を抑える



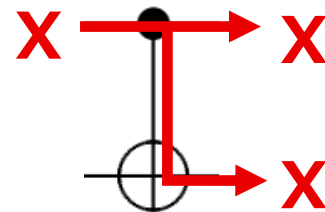
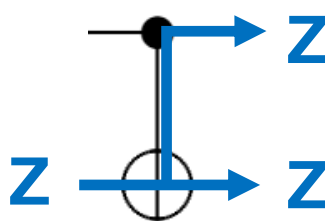
何らかの操作を行った直後にエラー訂正が挿入される

○この時問題になるのが**エラーの伝搬**

例) CNOTゲート

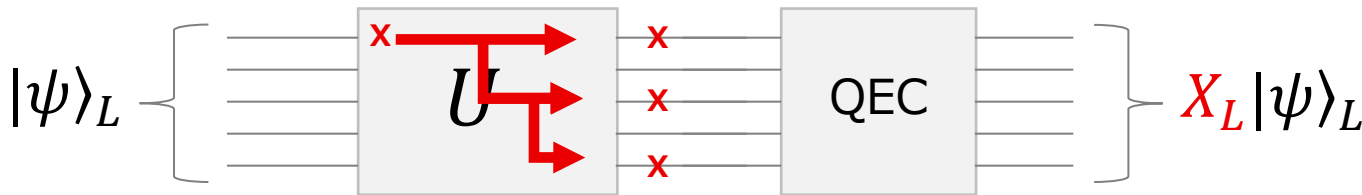
$$U_{CN} Z_t U_{CN}^\dagger = Z_c Z_t$$

$$U_{CN} X_c U_{CN}^\dagger = X_c X_t$$



CNOTの直前に起きた1つのエラーが、CNOTを作用させることで2つに増える

論理量子ゲート内でエラーが伝搬し増幅してしまうと、**本来訂正できるはずのエラーが訂正できなくなってしまう可能性がある**

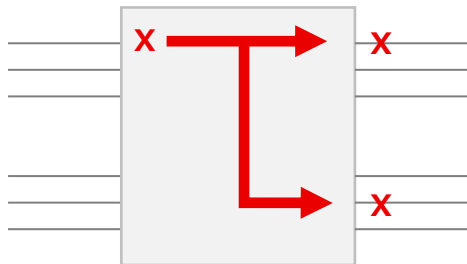


○このようなことが起こらない条件：誤り耐性 (Fault-tolerance)

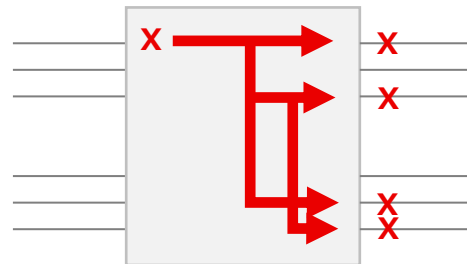
n 個の論理量子ビットに作用する論理演算を考えたとき、この演算が誤り耐性を持つとは、その演算過程の1か所で起こったエラーが高々各 n 論理量子ビットに1つずつしか伝搬しないことを言う

※実用上は、符号のエラー訂正性能に応じて定数個程度（例えば2つとか）の伝搬を許す場合もある

FTな例：



FTではない例：



- 実際にどのようにFault-toleranceが実現されているか、以下Steane符号を具体例にとって議論していく
- (復習) Steane符号
 - 7物理量子ビットからなる量子エラー訂正符号、符号距離3 (1つのエラーを訂正できる)

スタビライザー :

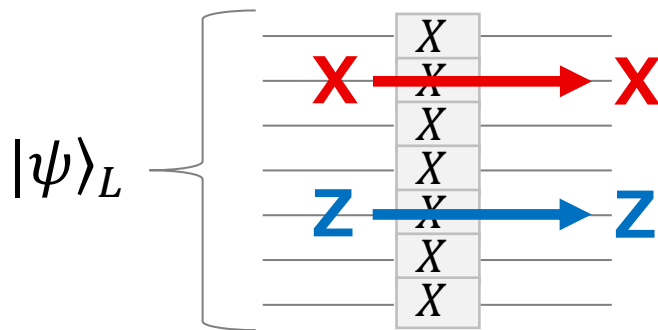
Name	Operator
g_1	$IIIXXXX$
g_2	$IXXIIXX$
g_3	$XIXIXIX$
g_4	$IIIZZZZ$
g_5	$IZZIIZZ$
g_6	$ZIZIZIZ$

論理演算子 :

$$X_L = XXXXXXXX$$
$$Z_L = ZZZZZZZZ$$

例) Steane符号の論理X, Zゲート

論理X, Zゲートは7つの物理量子ビットすべてに物理X、物理Zゲートを作用することで実行できた:



エラー伝搬しない
= Fault-tolerant

直後のエラー訂正で
正しく訂正できる

このように、符号を構成する物理量子ビット毎に物理ゲートを作用させることで実現できる論理ゲートは自動的に誤り耐性を持つ (**transversality**)

例) Steane符号のH, S, CNOTゲート : transversalに実行できる

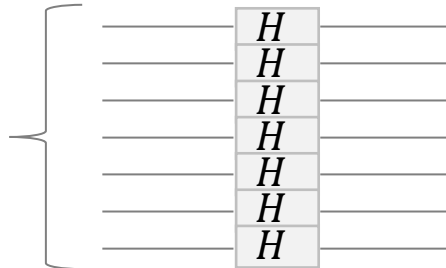
Name	Operator
g_1	<i>IIIXXXX</i>
g_2	<i>IXXIIXX</i>
g_3	<i>XIXIXIX</i>
g_4	<i>IIIZZZZ</i>
g_5	<i>IZZIIZZ</i>
g_6	<i>ZIZIZIZ</i>

$$X_L = XXXXXXXX$$

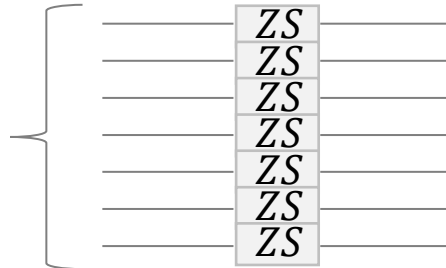
$$Z_L = ZZZZZZZZ$$

(確認は練習問題)

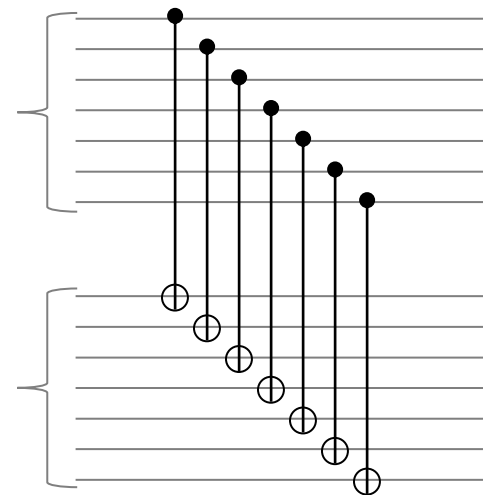
➤ 論理Hゲート



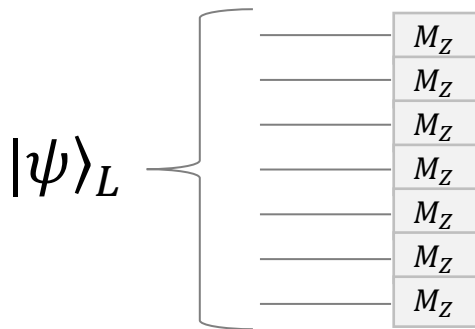
➤ 論理Sゲート



➤ 論理CNOTゲート



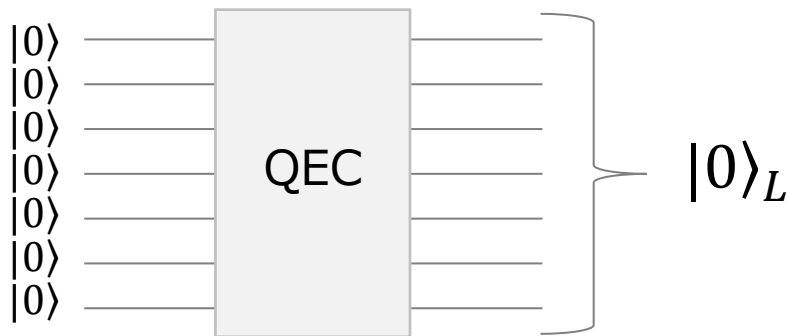
- 論理量子ビットのZ基底による測定も、transversalな方法でfault-tolerantに実行できる



符号を構成する物理量子ビットをZ測定
⇒ 測定プロセスでエラー伝搬は起こらない
⇒ Fault-tolerant

- ✓ 各測定値の積を取ることで、Zシンドローム値や論理Zの測定値を得られる（これらは測定基底と可換なため）
- ✓ 測定エラーは測定直前のXエラー+正しい測定とみなせるため、上記のように得られたZシンドローム値を用いてエラー訂正を行える

- 論理量子ビットの初期化は、transversalな初期化+スタビライザー測定による符号空間への射影によって実現可能



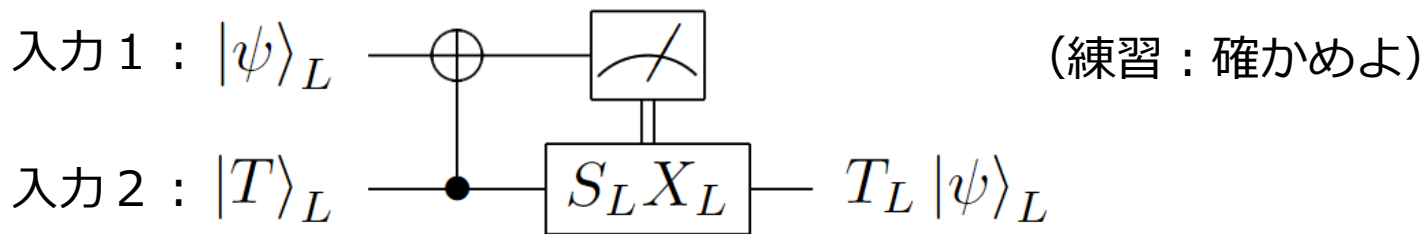
- ✓ スタビライザー測定はエラー訂正プロトコルの一部
- ✓ $|0\rangle$ 初期化が済み次第エラー訂正プロトコルを実行し、初期化時のエラーを取り除く

- ある特定の量子ゲートの組（ユニバーサルゲートセット）を組み合わせることで任意の量子計算が可能であることは既に見た
- これらがすべてtransversalに実行できれば、誤り耐性を容易に満たすことが出来るが…？

実は、ユニバーサルゲートセットすべてをtransversalに実行できる量子エラー訂正符号は存在しないことが知られている（Eastin-Knillの定理）

例) Steane符号では、ユニバーサルゲートのうち $\{S, CNOT, H\}$ は transversal に実行できるが、 T は実行できない

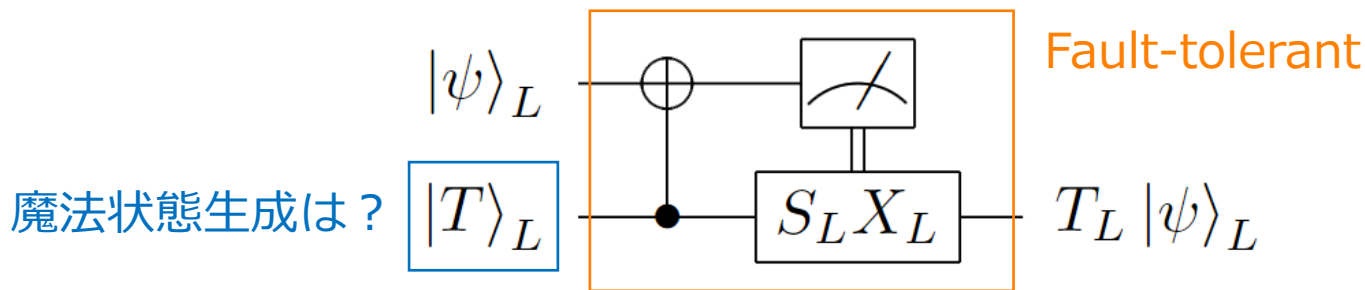
- transversal に構成できない論理ゲートは、**ゲートテレポーテーション**を介して間接的に実行するのが定石



- 実行に必要な状態 $|T\rangle$ は魔法状態と呼ばれる :

$$|T\rangle_L = \frac{1}{\sqrt{2}} (|0\rangle_L + e^{i\frac{\pi}{4}} |1\rangle_L)$$

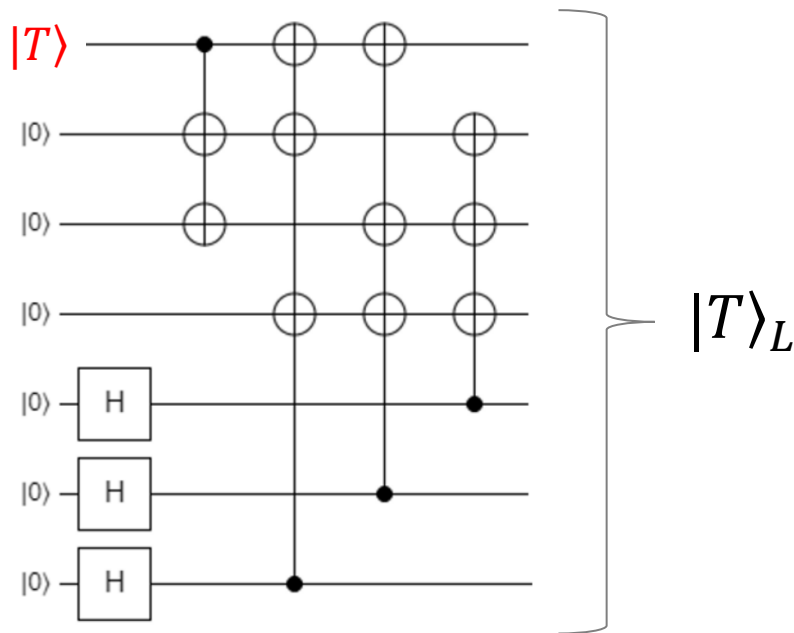
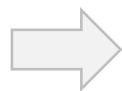
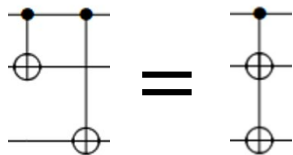
- ゲートテレポーテーション回路で実行されるCNOT, X, S, 測定はすべてFault-tolerant



- Tゲート実行に必要な魔法状態の生成がFault-tolerantになれば、全体としてFault-tolerantになる
- 実はここが厄介：魔法状態注入 + 魔法状態蒸留という手続きによってFault-toleranceを達成するのが定石

○ Steane符号の例：以下の回路で符号化された $|T\rangle_L$ を作る

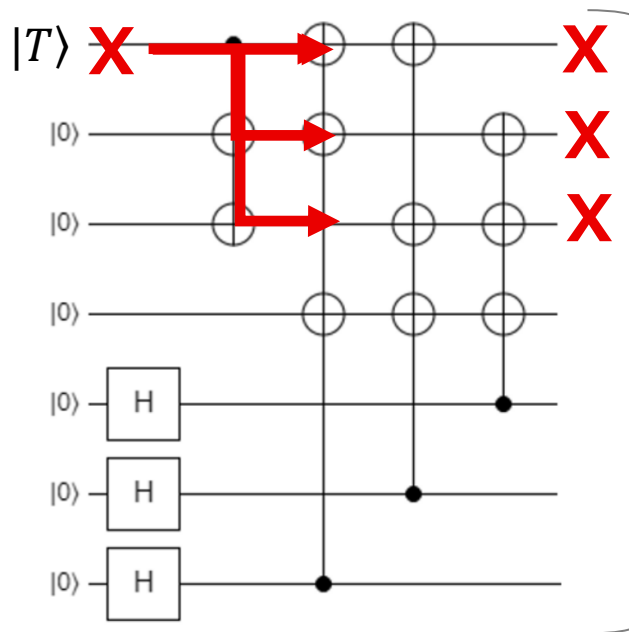
予め物理量子ビットで $|T\rangle$ を作って入力



ただし、この魔法状態注入操作はFault-tolerantではない：

Name	Operator
g_1	<i>IIIXXXX</i>
g_2	<i>IXXIIXX</i>
g_3	<i>XIXIXIX</i>
g_4	<i>IIIZZZZ</i>
g_5	<i>IZZIIZZ</i>
g_6	<i>ZIZIZIZ</i>

$$X_L = XXXXXXXX \\ = XXXIIII(\text{up to } g_1)$$



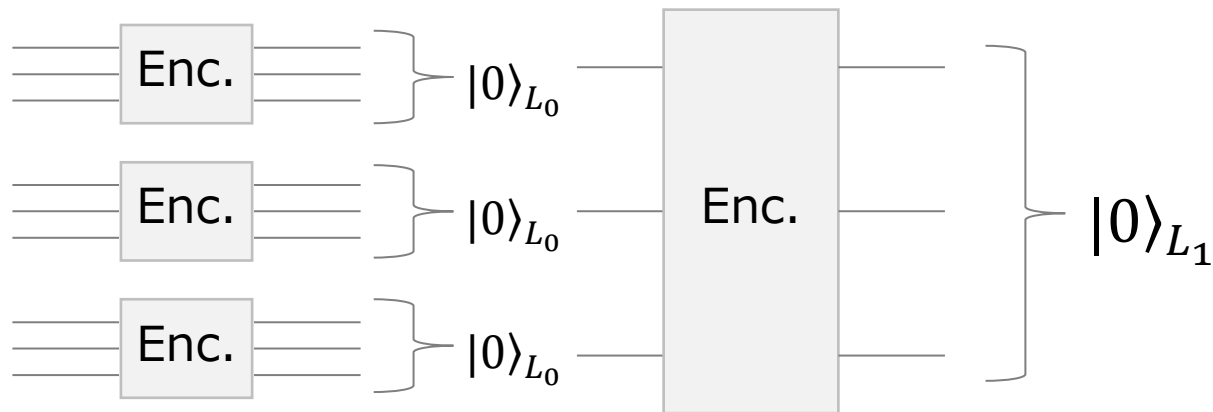
$$X_L |T\rangle_L$$

1か所のエラーが伝搬し、
論理エラーを引き起こす

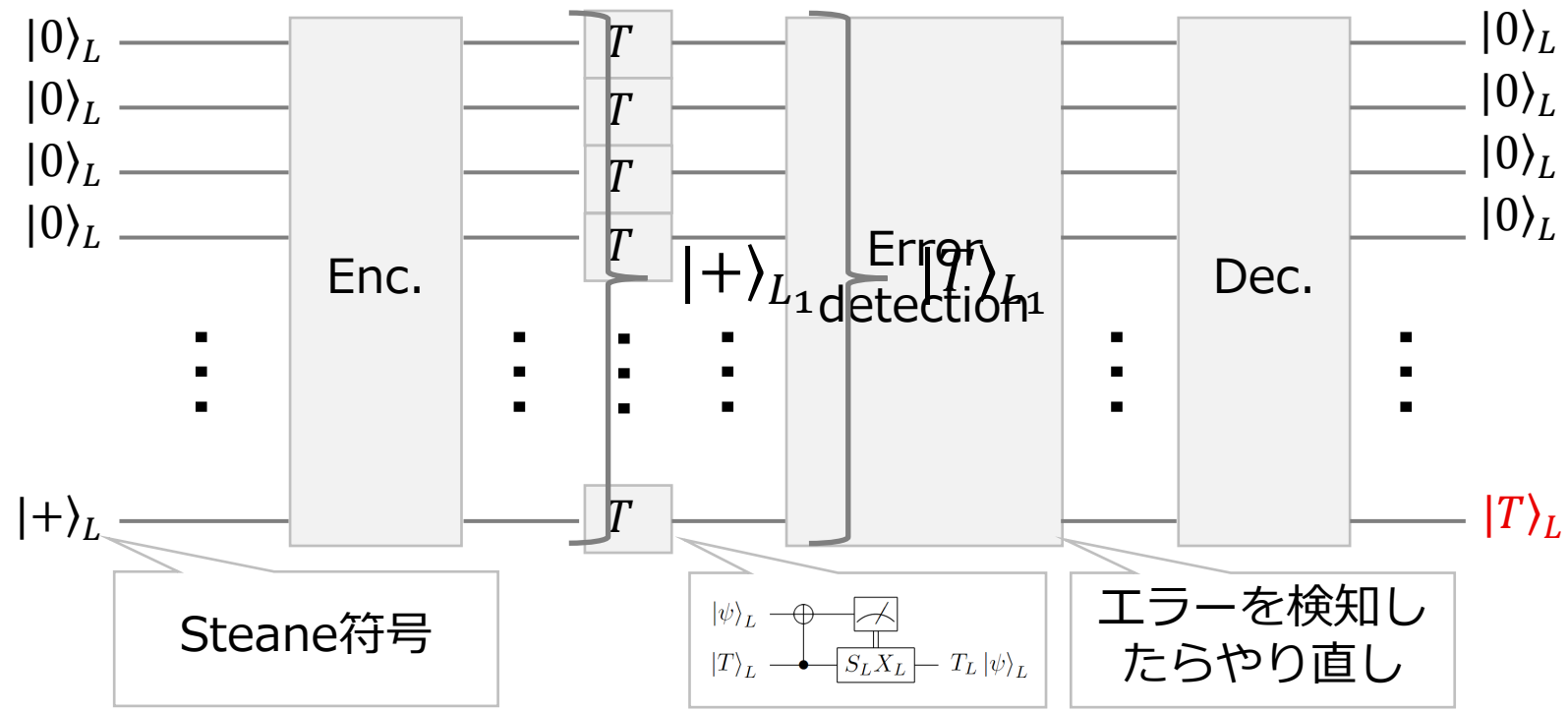
- 魔法状態注入だけでは、魔法状態は $O(p)$ で論理エラーを含む
- ここからエラーを取り除き、よりきれいな魔法状態を作り出すプロセスを**魔法状態蒸留(magic state distillation)**と呼ぶ

基本的な戦略：transversalな論理Tゲートが実行可能な符号と接続し、エラー検知を行ってエラーを減らす

- 符号の接続(concatenation)…論理量子ビットを複数用意し、それらをさらに符号化すること



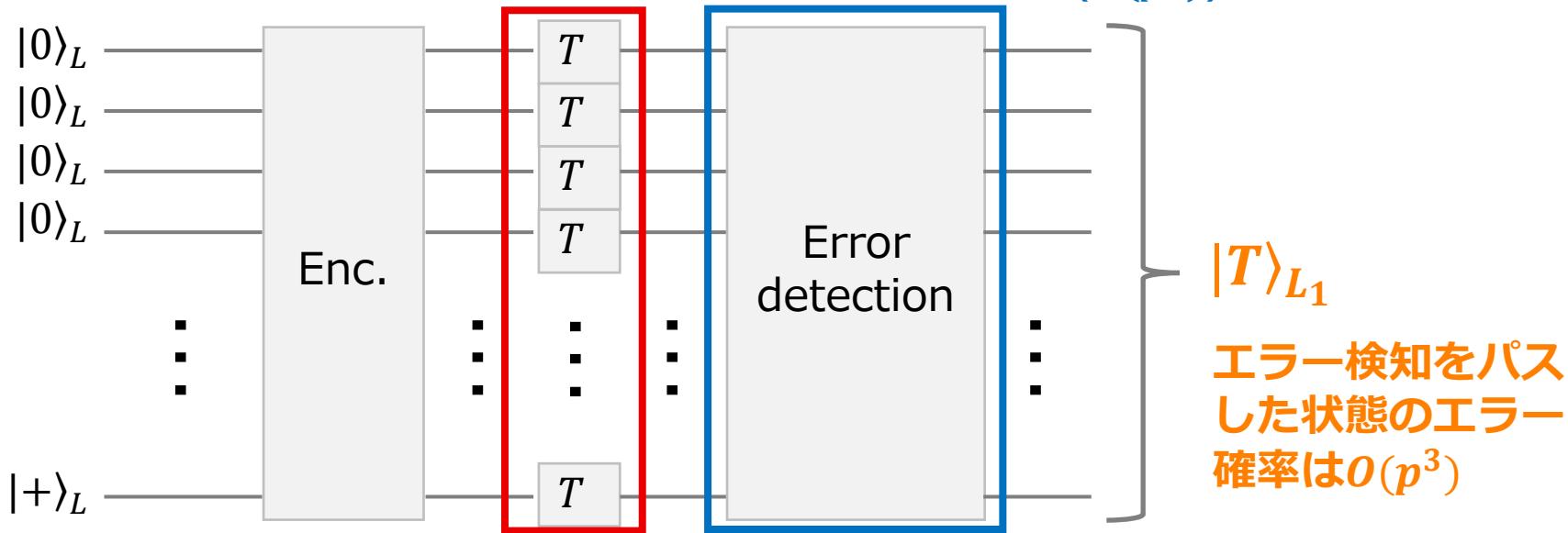
処理の流れ



エラー削減のメカニズム

接続した符号は符号距離3

2か所までのエラー($O(p^2)$)は検知可能



それぞれのTゲートは $O(p)$ のエラー確率
(先述の魔法状態注入で作った魔法状態を利用)

- 接続には、典型的に[[**15,1,3**]]Reed-Muller符号が用いられる
 - 15個の量子ビットから構成される符号、符号距離 3
 - 論理Tゲートがtransversalに実行できる
- transversal論理Tゲート実行のため、**15個**の魔法状態を消費する
- 蒸留された魔法状態は**1個**得られる
 - 前スライドで言及した通り、transversal論理Tゲートの各要素が $O(p)$ のエラーを含むため、エラー検知をパスした状態のエラー率は $O(p^3)$ まで改善

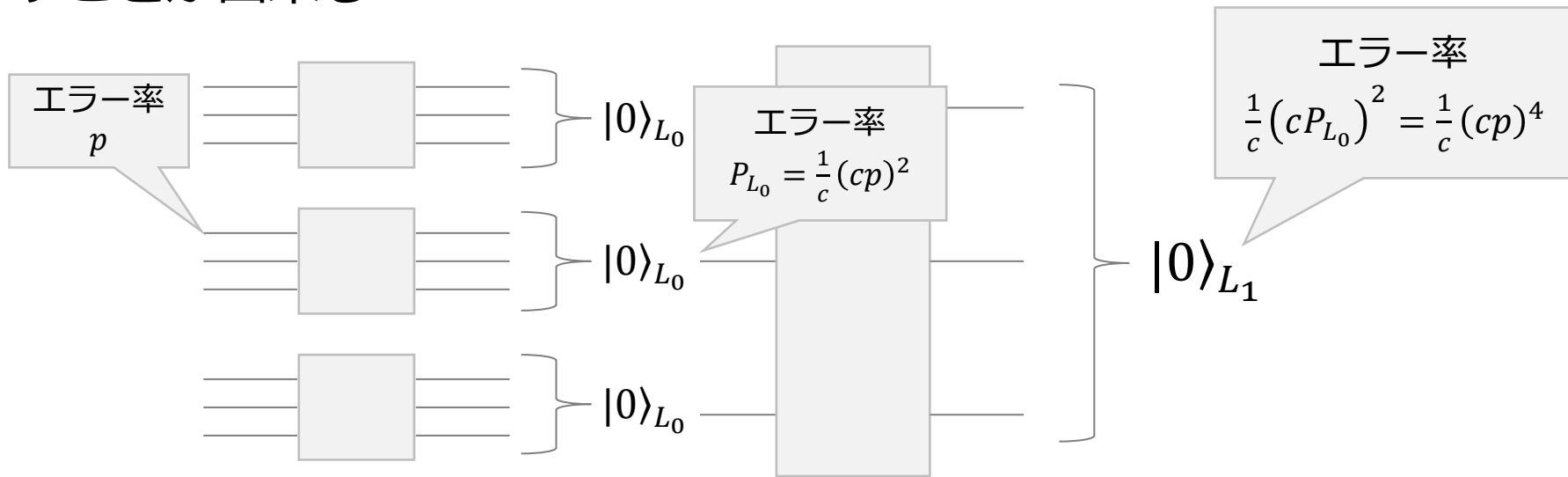
15-to-1 プロトコルと呼ばれる

- Steane符号を例にとり、論理演算の実行方法を議論してきた
 - $\{CNOT, H, S\}$ はtransversalに、 T は魔法状態蒸留を用いることで、すべてFault-tolerantに実行できる
 - 測定や計算基底への初期化もFault-tolerantに実行できる

Steane符号は1量子ビットを訂正できる符号であるため、結果としてすべての論理演算において論理エラー確率を $O(p^2)$ に抑制できる

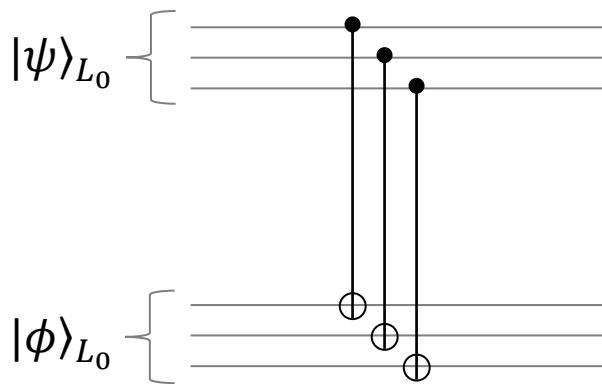
次に、ここから閾値定理（かっちり版）を導いてみる

- 先述の**接続(concatenation)**により、論理エラー率をさらに減らすことができる

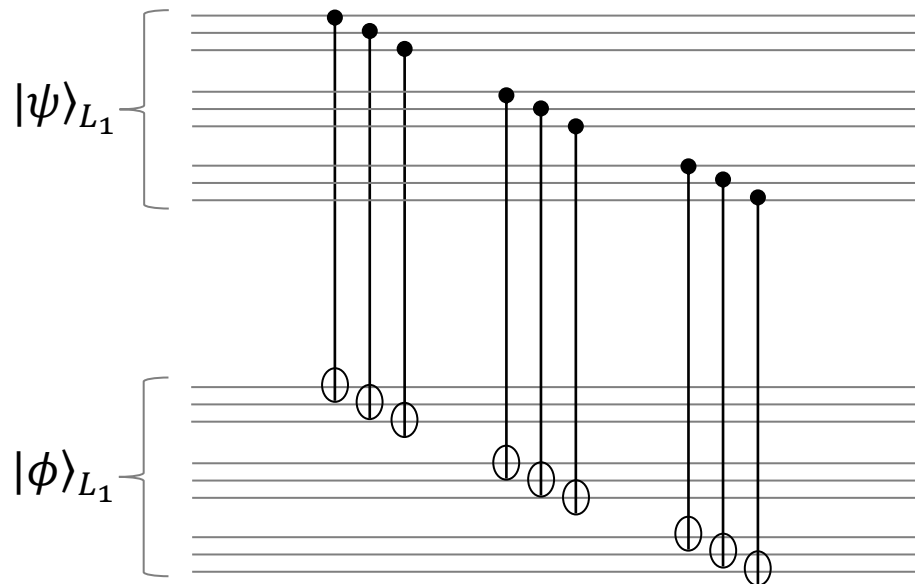


k 回接続すると、エラー率を $p \rightarrow \frac{1}{c}(cp)^{2^k}$ に抑制できる

- 符号化された状態への誤り耐性演算に必要な物理ゲートの数の最大数を D とすると、 k 回接続した場合には D^k 個の物理ゲートが必要



$$k = 1, D = 3$$



$$k = 2, D^2 = 9$$

いま、 $p(n)$ 個のゲートを含む量子回路の実行を考える (n :問題サイズ、 $p(n)$: n の多項式)

この回路をエラー確率 ϵ で実行するためには、各ゲートのエラー確率が $\epsilon/p(n)$ であればよく、 k 回の接続符号を用いると

$$\frac{(cp)^{2^k}}{c} \leq \frac{\epsilon}{p(n)}$$

を満たす k を取ってくればよい (ただし $p < 1/c = p_{th}$)

このとき、必要な物理ゲート数は $d^k \cdot p(n)$ だが、

$$d^k = 2^{k \log_2 d} \leq \left(\frac{\log_2 \left(\frac{p(n)}{c\epsilon} \right)}{\log_2 \left(\frac{1}{cp} \right)} \right)^{\log_2 d} = O \left(\text{poly} \left(\log_2 \left(\frac{p(n)}{\epsilon} \right) \right) \right)$$

$$\frac{(cp)^{2^k}}{c} \leq \frac{\epsilon}{p(n)} \text{ より } 2^k \leq \frac{\log_2 \left(\frac{p(n)}{c\epsilon} \right)}{\log_2 \left(\frac{1}{cp} \right)} \text{ を用いた}$$

より、回路全体は高々 $O(p(n) \text{poly}(\log(p(n)/\epsilon)))$ 個の物理ゲートで実行できる

以上の議論をまとめると次の閾値定理を得る：

閾値定理

量子計算機がある閾値 p_{th} より小さいエラー率で動作する
とする

このとき、 $p(n)$ 個のゲートを含む量子回路は高々
 $O(p(n)poly(\log(p(n)/\epsilon)))$ 個の物理ゲートを用いてエラー
確率 ϵ で実行することができる

現状最も有力な表面符号では閾値 $p_{th} \sim 0.75\%$

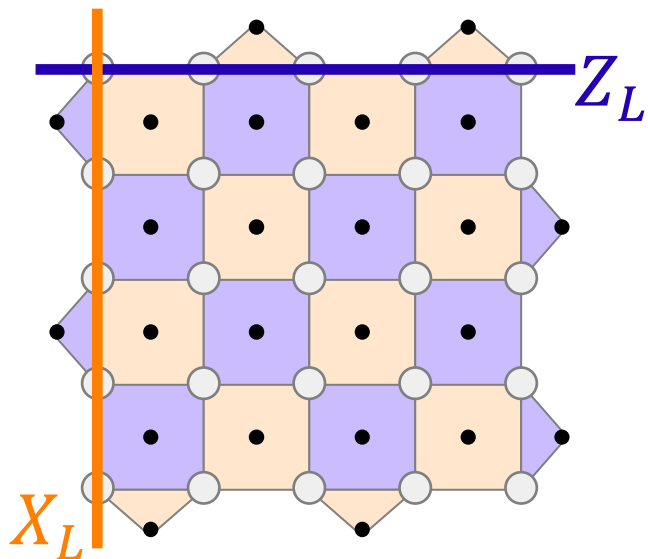
※もちろん、実際のエンジニアリングでは様々な困難がある

- 物理エラー率を閾値以下に下げること
- 量子ビット数
- エラー訂正の速度

etc…

- 既に述べたように、近年では表面符号と呼ばれる量子エラー訂正符号が注目されている
 - 実装のしやすさと閾値の高さ（ $\sim 0.75\%$ ）
- それゆえ、表面符号を用いたFTQCが広く研究されてきている
- ここからは、表面符号を用いて論理ゲートを効率良く実現するためのテクニックである、**Lattice surgery**を紹介する

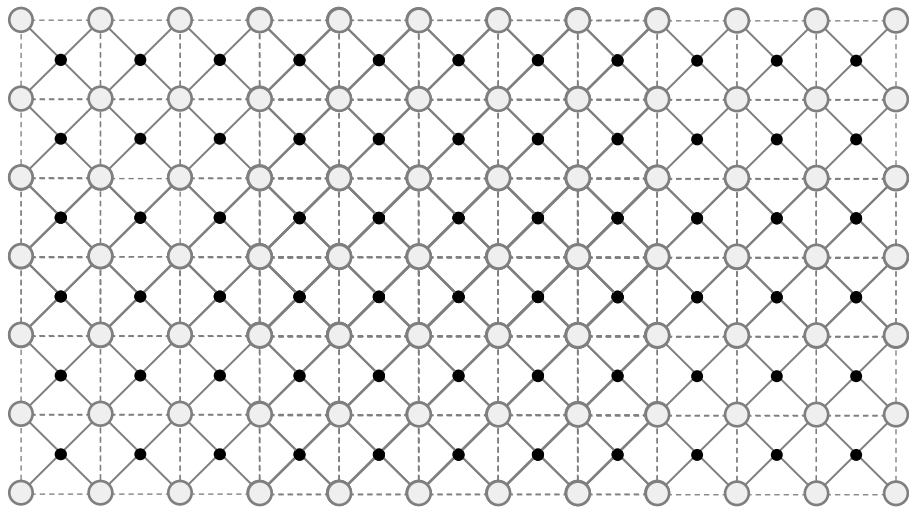
準備：表面符号における論理ゲートはどのように構成されるか？



- ✓ transversalに実行できるのは**論理H(向きが90度回転する), CNOTゲート**
- ✓ それ以外の論理S, Tゲートは魔法状態を用いて間接実行する
- ✓ 初期化・測定などはSteane符号の時と同様に実行可能

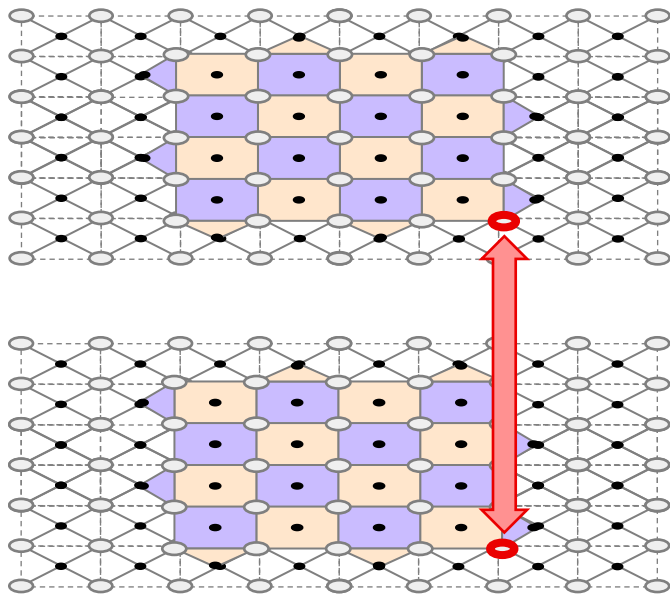
- 原理的にtransversalに実行できる論理ゲートも、量子コンピュータ実機で実行するのが難しい場合がある

例) 超伝導量子ビット



超伝導量子ビットは1枚のチップ平面上に量子ビットが並べられており（左図：○、●）、最近接した量子ビット同士でのみCNOTゲートを作作用できる（左図：実線でつながっているもの同士）

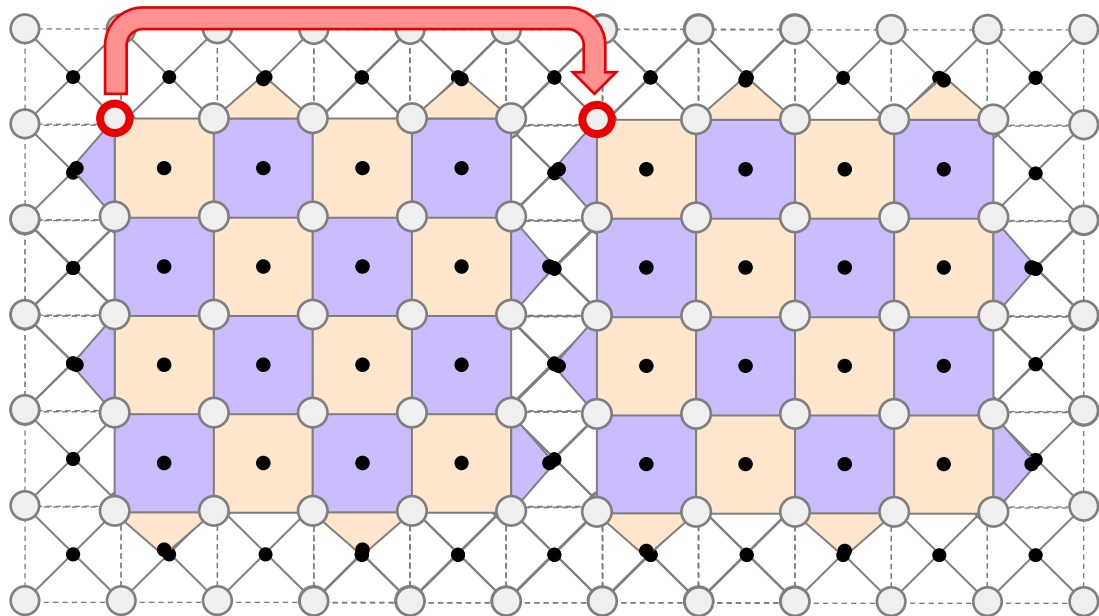
○超伝導実機で無理やりtransversal CNOT？



量子ビットを3次的に配置して
(平面チップを層状に重ねる)
上下を接続すればtransversal CNOT
ができるのでは？

**実機の構造上非現実的
(配線スペースやエラーの増加など)**

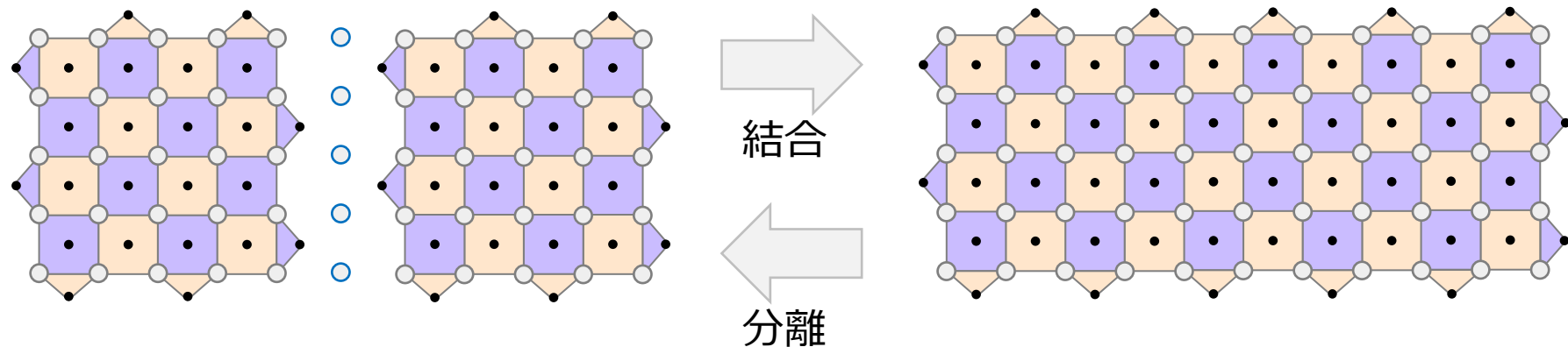
超伝導実機で無理やりtransversal CNOT ?



SWAPなどを大量に入れる必要あり、実行時間遅い&ノイズが多く厳しい

- このように実機制約がある場合に、表面符号で効率よくCNOTゲートを実行する方法はないか？
- このような背景で生み出されたのがLattice surgery
 - Lattice surgeryでは、平面上で隣接した符号間の局所的な操作を組み合わせることでCNOT演算を実行できる（余計にSWAP操作などを入れる必要なし）
 - 実機実装が現実的であることから、近年のFTQCアーキテクチャの基礎となっている

- 基本的な操作は、符号の結合（merge）と分離（split）

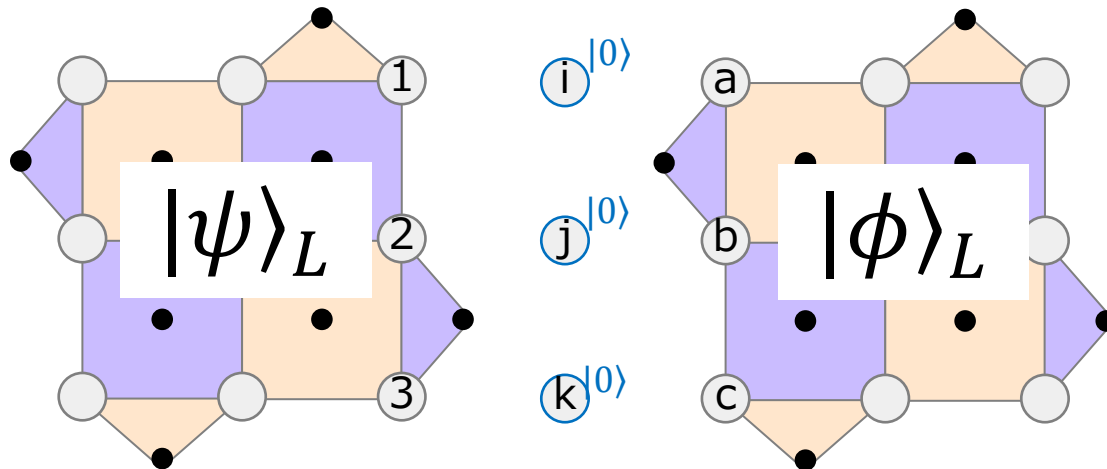


結合：2つの符号間のスタビライザー演算子を測定することで1つにする

分離：1つの符号の一部の物理量子ビットを測定することで2つに分ける

**結合と分離を連続で行うことで、境界の種類に応じて論理 $Z \otimes Z$ ($X \otimes X$)測定
を実行できる**

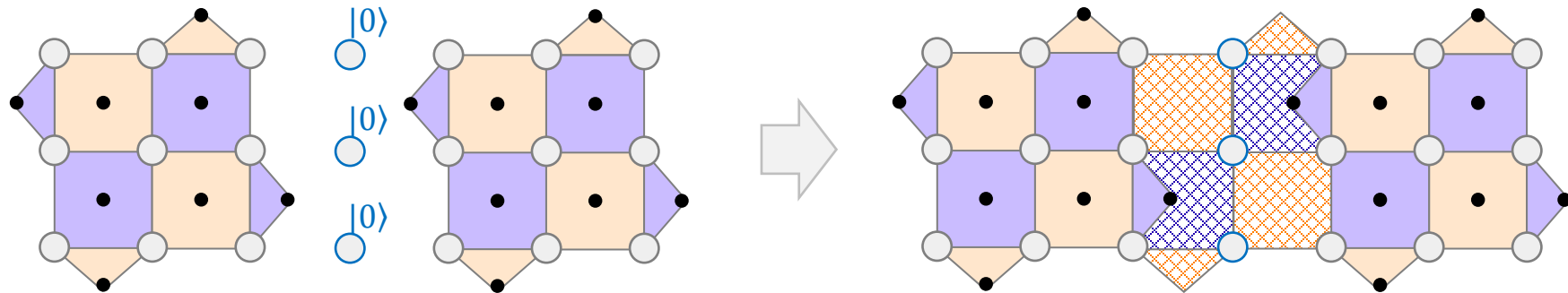
- 以下、簡単な例として $d = 3$ 表面符号の結合&分離を考える



操作実行前の状態は $|\psi\rangle_L |\phi\rangle_L |0\rangle_i |0\rangle_j |0\rangle_k$

結合&分離によって、最終的に $(1 + (-1)^M X_L \otimes X_L) |\psi\rangle_L |\phi\rangle_L$ が得られることを示す

○merge実行



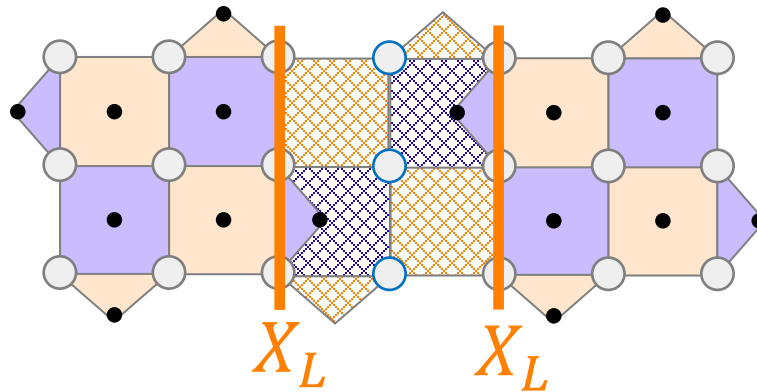
網掛け部分が新たに測定されるスタビライザー演算子

- ✓ Zスタビライザーは元々のZスタビライザーと $|0\rangle$ 初期化により測定前から固有値 + 1 確定
- ✓ Xスタビライザーは測定した段階で固有値 ± 1 どちらかになる

merge実行後の状態は $P_{X_{ia}} P_{X_{3k}} P_{X_{jkbc}} P_{X_{12ij}} |\psi\rangle_L |\phi\rangle_L |0\rangle_i |0\rangle_j |0\rangle_k$

$$P_{X_{i\dots j}} = \frac{1}{2} (1 + (-1)^{m_{i\dots j}} X_{i\dots j}) \text{ (射影演算子)}$$

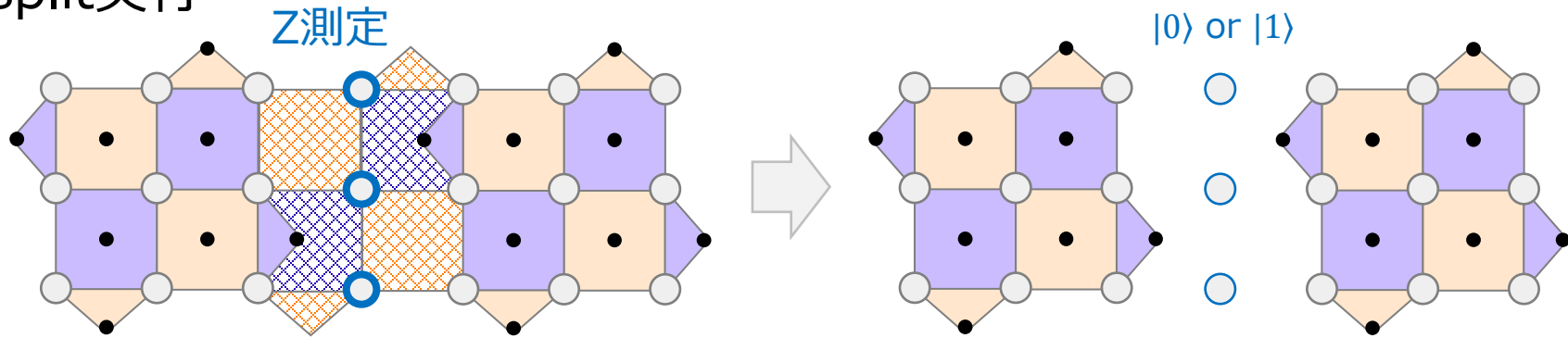
merge実行で測定されるXスタビライザーの積は、**2つの符号化された状態の論理X演算子の積**になっていることに注意



merge時のXスタビライザー測定により、間接的に論理 $X \otimes X$ 測定が実行されている
(測定値は各Xスタビライザー測定値の積)

このままでは1つの符号になってしまっているので、分離して再び2つの符号に戻す

○ split実行



merge実行前に $|0\rangle$ に初期化していた量子ビットを再びZ基底で測定する

merge実行後の状態 $P_{X_{ia}} P_{X_{3k}} P_{X_{jkbc}} P_{X_{12ij}} |\psi\rangle_L |\phi\rangle_L |0\rangle_i |0\rangle_j |0\rangle_k$
 を展開して、最終的にどうなっているかを見ていく

$$\begin{aligned}
 & P_{X_{ia}} P_{X_{3k}} P_{X_{jkbc}} P_{X_{12ij}} |\psi\rangle_L |\phi\rangle_L |0\rangle_i |0\rangle_j |0\rangle_k \\
 &= (|\psi\rangle_L |\phi\rangle_L + (-1)^M X_L \otimes X_L |\psi\rangle_L |\phi\rangle_L) |0\rangle_i |0\rangle_j |0\rangle_k \\
 &+ (-1)^{m_{ia}} X_a (|\psi\rangle_L |\phi\rangle_L + (-1)^M X_L \otimes X_L |\psi\rangle_L |\phi\rangle_L) |1\rangle_i |0\rangle_j |0\rangle_k \\
 &+ (-1)^{m_{ia}+m_{12ij}} X_{12} X_a (|\psi\rangle_L |\phi\rangle_L + (-1)^M X_L \otimes X_L |\psi\rangle_L |\phi\rangle_L) |0\rangle_i |1\rangle_j |0\rangle_k \\
 &+ (-1)^{m_{3k}} X_3 (|\psi\rangle_L |\phi\rangle_L + (-1)^M X_L \otimes X_L |\psi\rangle_L |\phi\rangle_L) |0\rangle_i |0\rangle_j |1\rangle_k \\
 &+ (-1)^{m_{12ij}} X_{12} (|\psi\rangle_L |\phi\rangle_L + (-1)^M X_L \otimes X_L |\psi\rangle_L |\phi\rangle_L) |1\rangle_i |1\rangle_j |0\rangle_k \\
 &+ (-1)^{m_{ia}+m_{3k}} X_3 X_a (|\psi\rangle_L |\phi\rangle_L + (-1)^M X_L \otimes X_L |\psi\rangle_L |\phi\rangle_L) |1\rangle_i |0\rangle_j |1\rangle_k \\
 &+ (-1)^{m_{ijbc}} X_{bc} (|\psi\rangle_L |\phi\rangle_L + (-1)^M X_L \otimes X_L |\psi\rangle_L |\phi\rangle_L) |0\rangle_i |1\rangle_j |1\rangle_k \\
 &+ (-1)^{m_{ia}+m_{jkbc}} X_{abc} (|\psi\rangle_L |\phi\rangle_L + (-1)^M X_L \otimes X_L |\psi\rangle_L |\phi\rangle_L) |1\rangle_i |1\rangle_j |1\rangle_k
 \end{aligned}$$

splitに伴うZ測定で
いずれかが選ばれる

- 測定結果が $(+1, +1, +1)$ だった場合：

$$\text{測定後の状態は } (|\psi\rangle_L|\phi\rangle_L + (-1)^M X_L \otimes X_L |\psi\rangle_L|\phi\rangle_L) |0\rangle_i |0\rangle_j |0\rangle_k$$

論理 $X \otimes X$ 測定後の状態になっている

- 測定結果が $(-1, +1, +1)$ だった場合：

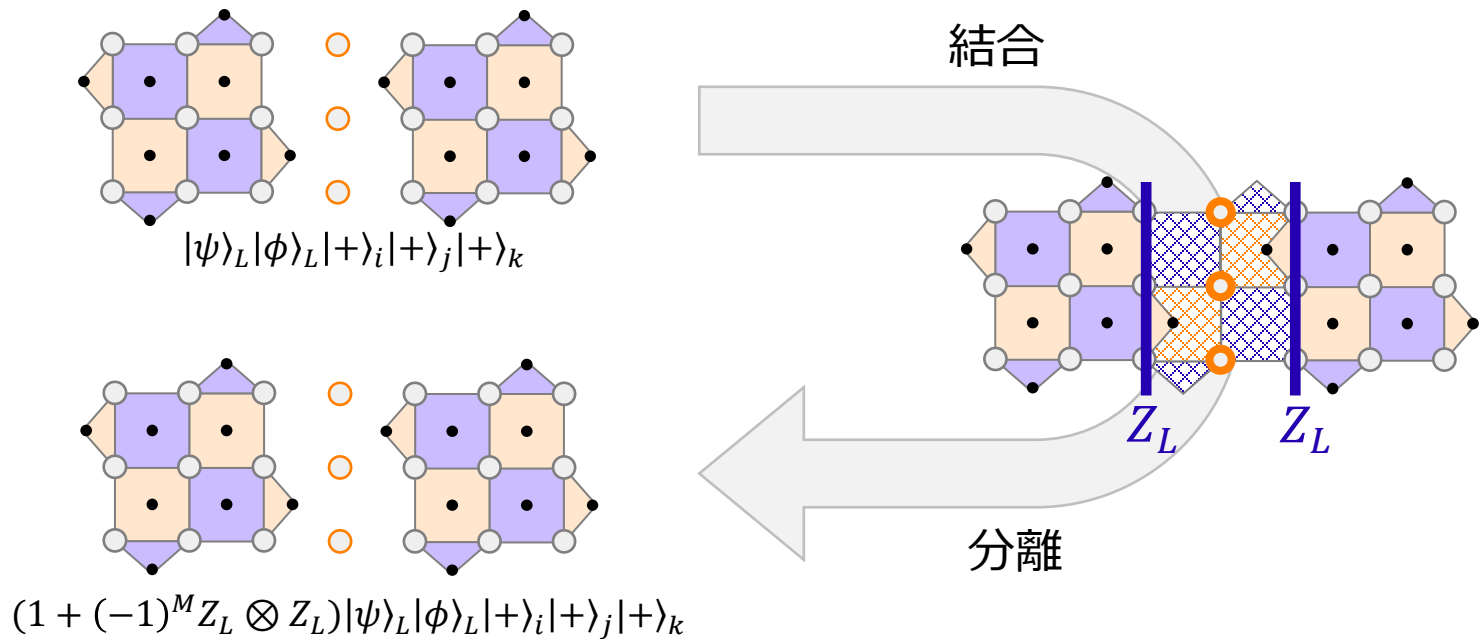
$$\text{測定後の状態は } (-1)^{m_{ia}} X_a (|\psi\rangle_L|\phi\rangle_L + (-1)^M X_L \otimes X_L |\psi\rangle_L|\phi\rangle_L) |1\rangle_i |0\rangle_j |0\rangle_k$$

X_{ia} を作用して打ち消すことで論理 $X \otimes X$ 測定後の状態になる

(ほかの測定結果の場合も同様)

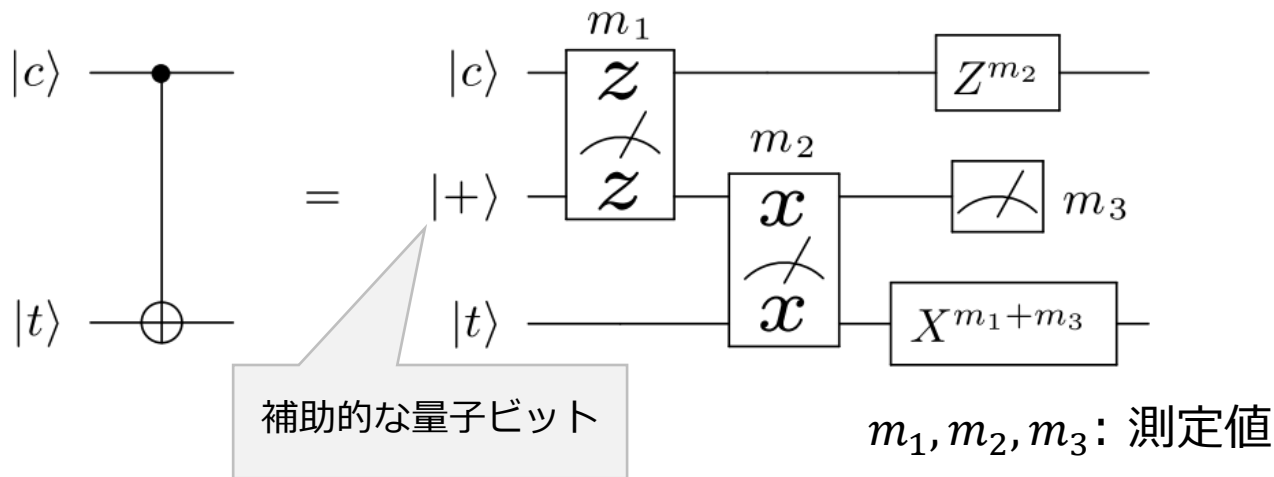
測定結果に応じて適切なパウリ演算子を作用することで、いずれの場合にも論理 $X \otimes X$ 測定が達成される！

- 論理Z ⊗ Z測定も、論理Z演算子が定義されている境界同士で結合 & 分離を行えばまったく同様に実行可能：



- 議論してきた論理 XX 、 ZZ 測定を組み合わせることで、論理CNOT演算を実行できる！

準備：CNOTは以下の等価回路で実行できる



(確認) 測定値がすべて+1だった場合を考える

$$|c\rangle|+\rangle|t\rangle = (\alpha|0\rangle + \beta|1\rangle)|+\rangle|t\rangle \quad (|c\rangle = \alpha|0\rangle + \beta|1\rangle)$$

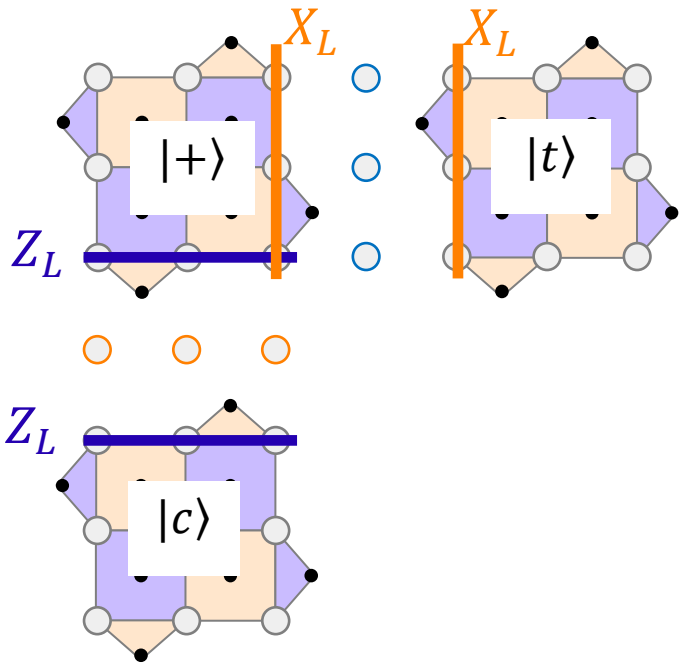
$$\rightarrow (\alpha|00\rangle + \beta|11\rangle)|t\rangle \quad (\text{論理}Z_cZ_+ \text{測定})$$

$$\rightarrow (\alpha|00\rangle + \beta|11\rangle)|t\rangle + (\alpha|01\rangle + \beta|10\rangle)X_t|t\rangle \quad (\text{論理}X_tX_+ \text{測定})$$

$$\rightarrow \alpha|0\rangle|0\rangle|t\rangle + \beta|1\rangle|0\rangle X_t|t\rangle \quad (\text{論理}Z_+ \text{測定})$$

$$\rightarrow \alpha|0\rangle|t\rangle + \beta|1\rangle X_t|t\rangle \quad (\text{アンシラ無視})$$

実機での配置例：

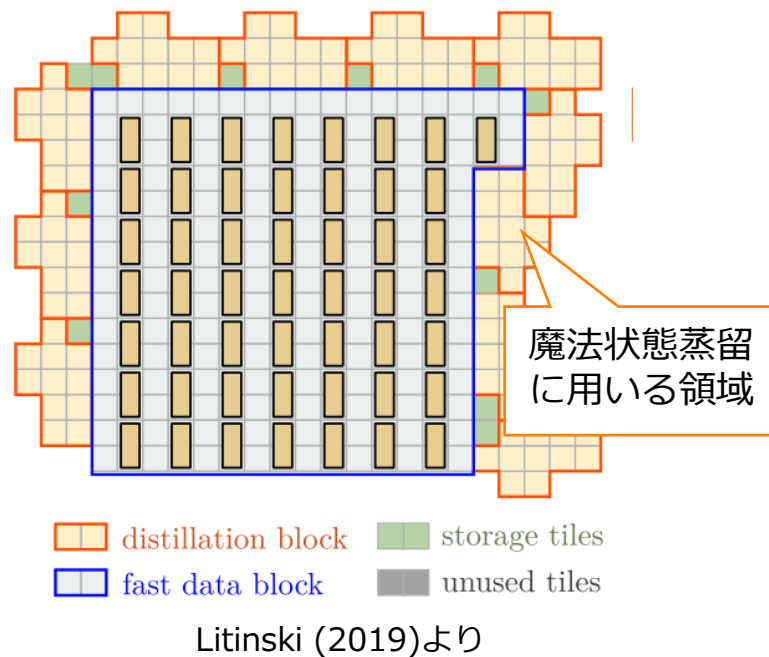
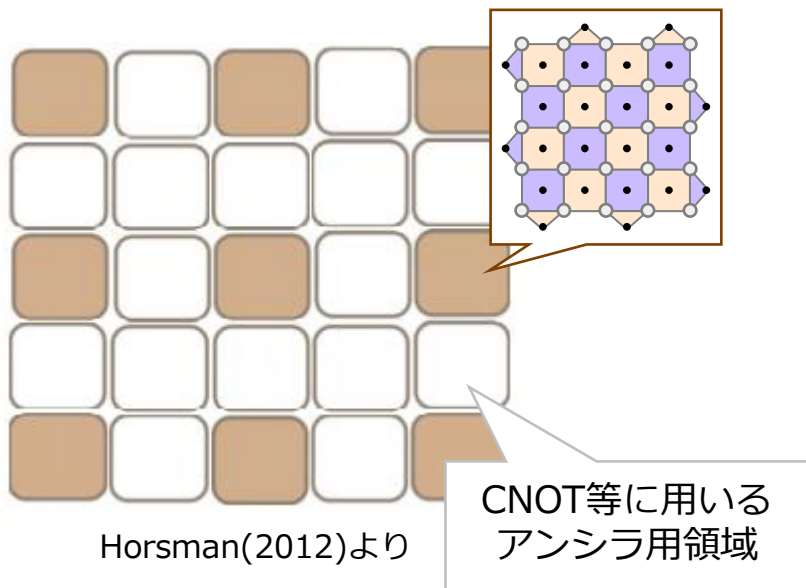


操作手順：

1. $|+\rangle$ に初期化されたアンシラを準備
2. アンシラとctrl量子ビットでZZ測定
3. アンシラとtrgt量子ビットでXX測定
4. アンシラをZ測定
5. 各測定結果に応じてパウリ演算子を作用

測定されるスタビライザーはすべて局所的なので、実機制約下でも問題なく実行できる（アンシラは必要）

Lattice surgeryでFTQCを実行するための論理量子ビット配置例：



○ 閾値定理

- 量子コンピュータがある閾値以下のエラー率で動作するとき、高々多項式オーバーヘッドで任意精度の量子計算が実行できる
- 現状知られている閾値は**0.75%程度（表面符号）**

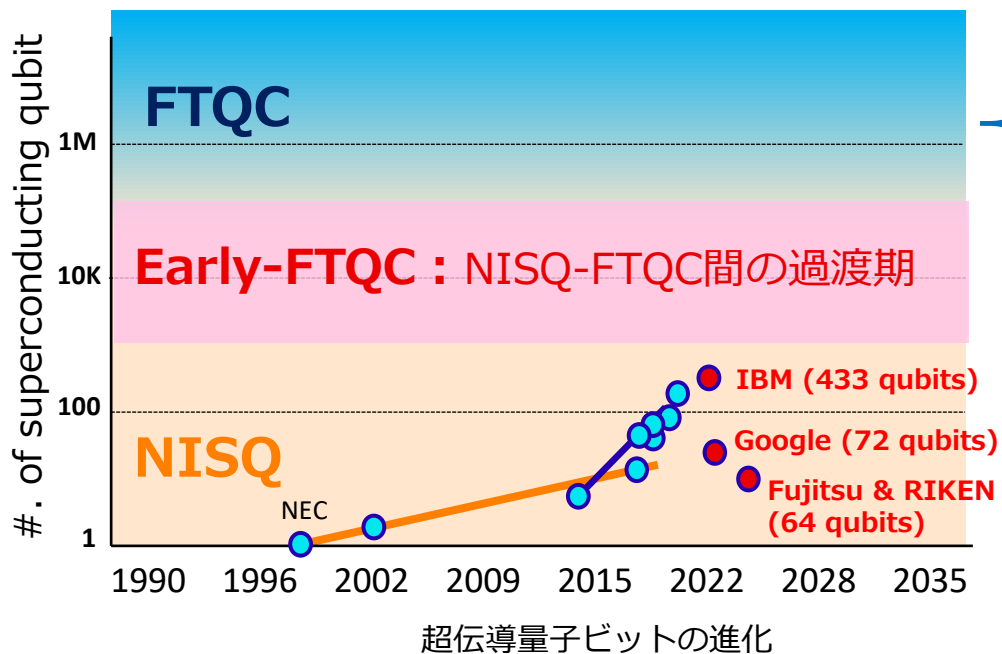
○ 近年の主流： **Lattice surgery**による論理ゲート実行

- モチベーション：実機制約がある場合でも、表面符号で効率良く論理ゲートが実行できないか？
- **結合&分離により、論理 XX , ZZ 測定を実行できる**
- それらを組み合わせることで論理CNOT演算を実現
- Lattice surgeryに基づく論理量子ビット配置例

Early-FTQC向け新量子計算アーキテクチャ

Y. Akahoshi et al., arXiv:2303.13181 に基づく

100量子ビットを超えるノイズあり中規模量子コンピュータ(NISQ)は利用可能になりつつあるが、ノイズの影響により複雑な計算は困難



2048ビット素因数分解
 2×10^7 qubits (C. Gidney et al. (2019))

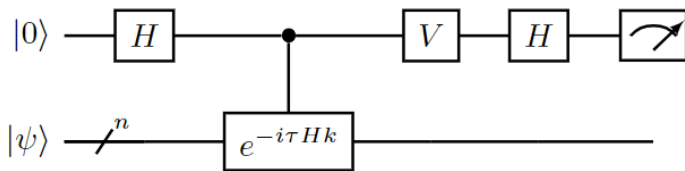
量子化学計算
 4×10^6 qubits (J. Lee et al. (2021))

物性計算
 $O(10^5)$ qubits (N. Yoshioka et al. (2022))

Early-FTQCにおいて、量子計算の有用性をデモンストレーションしたい…

- このような背景の下、近年Early-FTQCを念頭に置いた研究が増え
てきている

➤ 統計的位相推定



アンシラ1つで実行可能

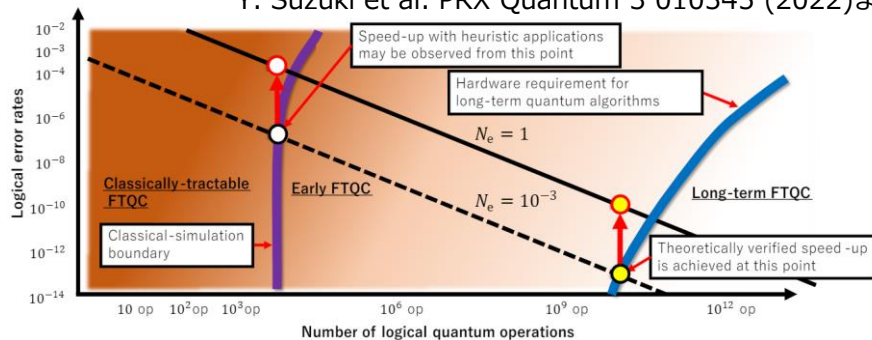
測定値の平均：
$$gk = \langle \psi | e^{-i\tau Hk} | \psi \rangle$$

$$= \sum_i p_i e^{-i\tau \lambda_i k}$$

古典処理によりエネルギー推定

➤ FTQCリソース削減

Y. Suzuki et al. PRX Quantum 3 010345 (2022)より



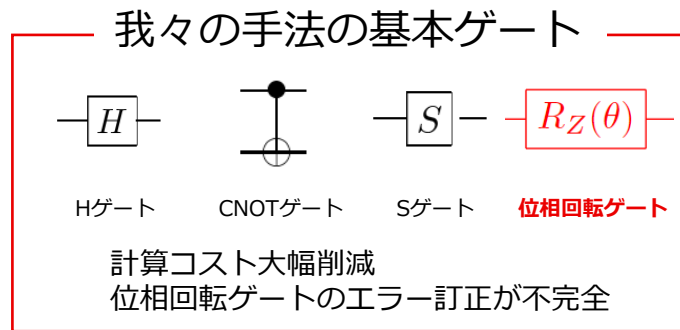
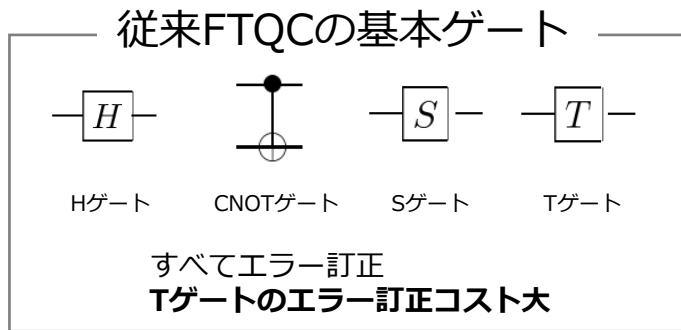
NISQで用いられる**量子エラー緩和**を
組み合わせることで、FTQCに必要な
リソースを削減する

- Early-FTQCを念頭に置いた量子計算アーキテクチャ研究はあるものの、あくまでFTQCをどこまで妥協できるか、といったもの

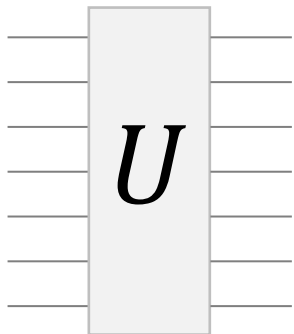
そもそもEarly-FTQCに特化した量子計算アーキテクチャを設計できないだろうか？

既存アプローチよりも高い性能を発揮することが出来るか？

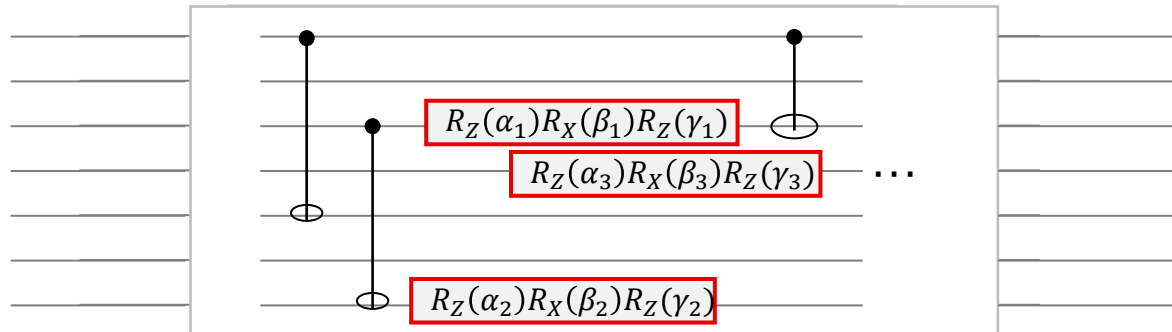
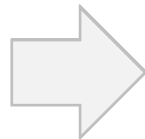
- ✓ **基本ゲートを見直す**ことで、物理量子ビット数と実行時間を削減
 - Tゲートではなく、任意角度の回転ゲートを直接実行する
 - Solovay-Kitaev (~100T) 不要、魔法状態蒸留スキップ
- ✓ 副作用として位相回転ゲートの量子エラー訂正が不完全になるが、**位相回転ゲートの実行方法を工夫し可能な限り精度を高める**



ユニバーサルティ？



目的のユニタリU



- ✓ CNOTと1量子ビットユニタリ(U_1, U_2, \dots)で分解できる
- ✓ 1量子ビットユニタリは任意回転ゲート3つで分解 (オイラー角)
- ✓ 1量子ビットユニタリは{H, T, S}を用いて近似

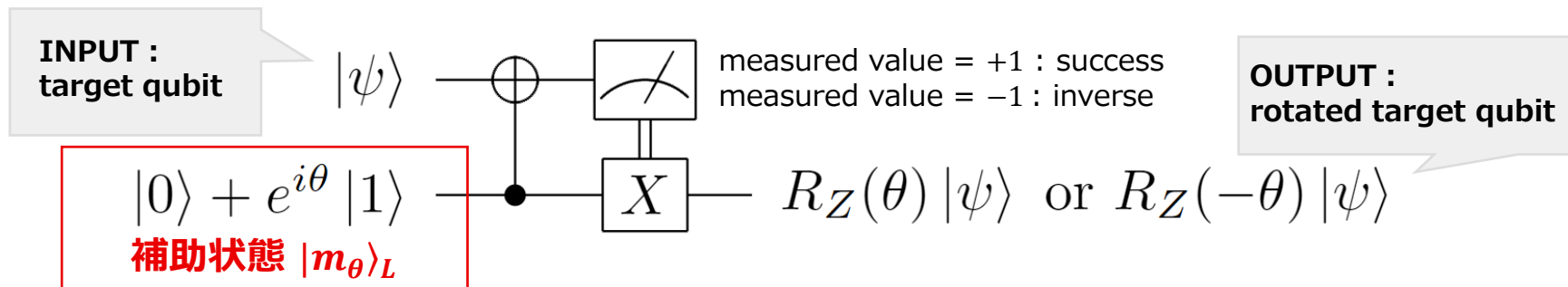
	Clifford*ゲート	Non-Clifford**ゲート	リソース要求
NISQ	ノイズあり	ノイズあり	小（符号化無し）
FTQC	ノイズなし	ノイズなし	大（符号化、魔法状態蒸留）
本研究	ノイズなし	ノイズあり	中（符号化のみ）

*Cliffordゲート：パウリ群を不変に保つゲート（CNOT,H,S）

**non-Cliffordゲート：Cliffordゲートに含まれないゲート

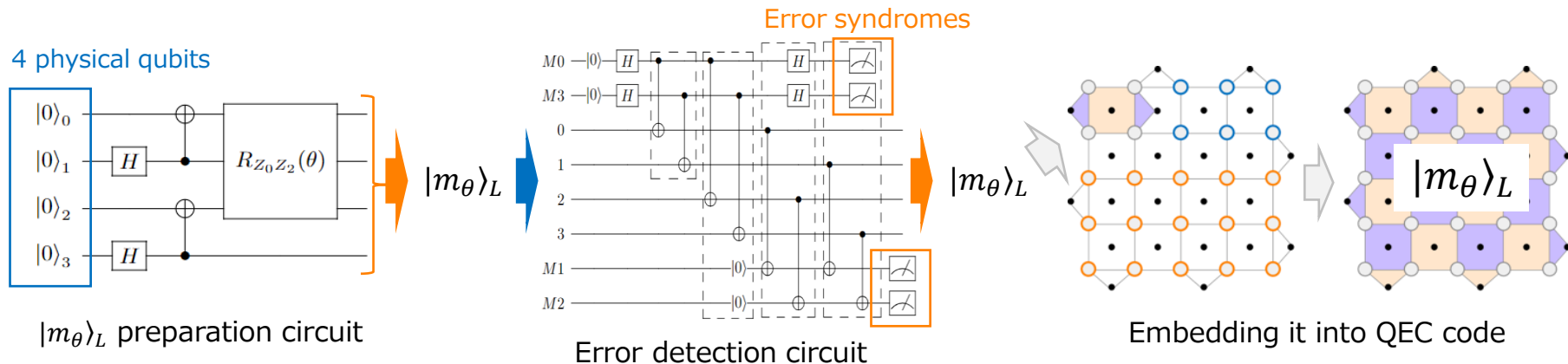
任意回転ゲート(non-Clifford)をどれだけ高精度に実行できるかが性能のカギ

ゲートテレポーション回路で実行：



- ✓ 測定結果によって出力される状態が変化（五分五分）
- ✓ 逆回転になった場合、 2θ 回転角で回転しなおす
- ✓ これを正しい出力が得られるまで繰り返す（Repeat-Until-Success, RUS）
- ✓ 補助状態生成でエラーが発生する
 - 魔法状態蒸留は実行できないので、なるべくエラーが生じないような生成プロセスを工夫する必要がある

エラーを減らす工夫：[[4,1,1,2]]エラー検知符号に符号化し、ポストセレクションと組み合わせることでエラーを削減

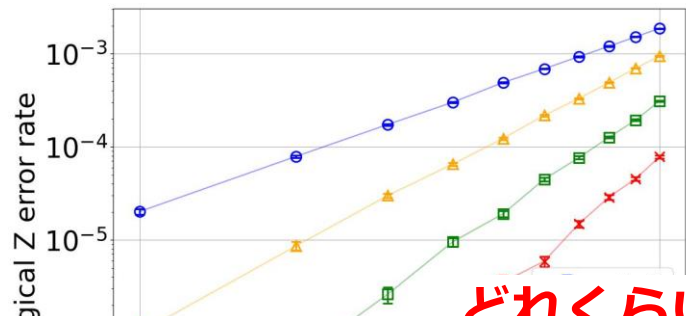


回路レベルノイズ模型（あらゆる操作でエラーが確率 p で発生）の下で、生成される

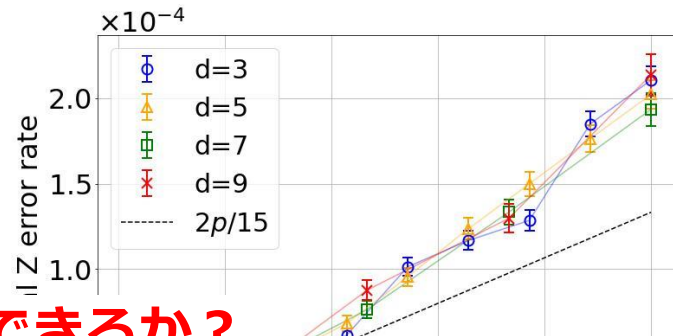
補助状態の論理エラー確率は $p_L = \frac{2p}{15} + O(p^2)$

魔法状態生成に関する先行研究と比較して高い精度 (e.g. $46p/15$ in Y. Li New J. Phys. 17, 023037 (2015))

➤ 量子エラー訂正シミュレーション



➤ 補助状態生成シミュレーション



どれくらいの計算ができるか？

10^4 物理量子ビット (エラー確率 $p = 10^{-4}$) のデバイスを想定した場合、
我々のアーキテクチャを用いると、**64論理量子ビット**に対して

Cliffordゲート : 1.72×10^7 回

任意回転ゲート : 3.75×10^4 回

実行可能($d = 7$)

フィット関数

積むことが可能

✓ $O(p^4)$ is negligible at $p \sim 10^{-4}$

- FTQCアーキテクチャ(D. Litinski, Quantum 3, 128 (2019))との比較
- 同じデバイス(10^4 物理量子ビット、 $p = 10^{-4}$)を想定

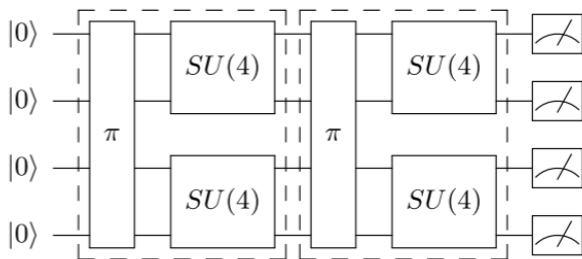
TABLE II. Comparison between the STAR architecture and existing FTQC architecture [30] on the early-FTQC device that consists of 10^4 physical qubits with a square grid connectivity and an error rate $p = 10^{-4}$.

Arch.	Num. of logical qubits	Non-Clifford gate execution time [clock]
STAR Compact ($d = 7$)	64	18
FTQC Fast ($d = 7$)	0	46
FTQC Intermediate ($d = 7$)	32	230
FTQC Compact ($d = 7$)	51	414

compact/intermediate/fast: 論理量子ビットの配置方法の違い
(使用する量子ビット数はcompact < intermediate < fast)

FTQCと比較し、より多くの論理量子ビットを生成でき、ゲート操作も高速

○NISQにおける性能指標：量子ボリューム



ベンチマーク回路 (4 qubits、depth 2)

n 量子ビット、深さ n のベンチマーク回路 (左図) がreasonableに実行できたとき、量子ボリュームは 2^n

- ✓ 同じデバイス(10^4 物理量子ビット、 $p = 10^{-4}$)を想定すると、NISQでは $QV = 2^{37}$ (A. W. Cross et al., Phys. Rev. A 100, 032328 (2019)における評価式を用いて算出)
- ✓ 我々のアプローチでは、 $QV = 2^{64}$

量子多体系シミュレーションとは相性が良い：

$$H = \sum_i c_i P_i \quad e^{-iHt} = e^{-it \sum_i c_i P_i} \approx \left(\prod_i e^{-i \frac{t}{N} c_i P_i} \right)^N$$

Trotter分解

パウリのテンソル積

それぞれ任意回転ゲートで実行

例) 32-site 2D Hubbard modelでおよそ100 Trotterステップくらい踏める
(先述のリソース見積もりに基づく試算)

- 来るべきEarly-FTQCに向けて、量子計算の有用性を示すための研究が進められている
- 本研究では、Early-FTQCに特化した量子計算アーキテクチャを提案した
 - Cliffordゲートは量子エラー訂正、non-Cliffordゲート（任意回転ゲート）はノイズを含む、部分的な誤り耐性量子計算
 - 数値計算に基づくリソース見積もりによれば、1万量子ビット程度のデバイスで既存アプローチよりも高い性能を発揮できる可能性がある
- 今後
 - 手法のさらなる改善
 - 有用なアルゴリズム探索

全体のまとめ

- 本講義では、量子エラー訂正および誤り耐性量子計算の基礎と、最近の主流や研究について、具体例を交えながら紹介した
- 量子エラー訂正
 - スタビライザー符号から表面符号まで紹介
- 誤り耐性量子計算
 - Steane符号を例に、誤り耐性のある論理演算の構成方法を議論
 - 近年の主流であるLattice surgeryも紹介
- Early-FTQC向け量子計算アーキテクチャ
 - 比較的近い将来実現する実機で有効な量子計算方法の提案

- 量子計算の基礎事項
 - M. Nielsen, I. Chuang “Quantum computation and quantum information” (QCQI)
- 量子エラー訂正・FTQC
 - P. Shor “Scheme for reducing decoherence in quantum computer memory”. Physical Review A. 52 (1995)
 - D. Gottesman “Stabilizer Codes and Quantum Error Correction”, PhD thesis, <https://arxiv.org/abs/quant-ph/9705052> (1997)
 - A. Calderbank, P. Shor “Good Quantum Error-Correcting Codes Exist” Physical Review A 54 1098 (1996)
 - A. Fowler et al. “Surface codes: Towards practical large-scale quantum computation”, Physical Review A 86 032324 (2012)
 - C. Horsman et al. “Surface code quantum computing by lattice surgery”, New Journal of Physics, 14 123011 (2012)

○量子エラー訂正・FTQC

- Google Quantum AI, “Suppressing quantum errors by scaling a surface code logical qubit”, Nature 614 (2023)
- D. Litinski “A game of surface code”, Quantum 3, 128 (2019)
- C. Gidney, M. Eker, “How to factor 2048 bit RSA integers in 8 hours using 20 million noisy qubits”, Quantum 5, 433 (2021)
- J. Lee et al. “Even More Efficient Quantum Computations of Chemistry Through Tensor Hypercontraction”, PRX Quantum 2, 030305 (2021)
- N. Yoshioka et al. “Hunting for quantum-classical crossover in condensed matter problems”, arXiv:2210.14109 (2022)
- Y. Akahoshi et al. “Partially Fault-tolerant Quantum Computing Architecture with Error-corrected Clifford Gates and Space-time Efficient Analog Rotations”, arXiv:2303.13181 (2023)

Thank you

