Surface code と Stabilizer

Surface code と Stabilizer **Agenda**

Part 1　量子エラー訂正技術の飛躍

- 我々は、今どこにいるのか？
- エラー訂正技術の歴史から学ぶ

Part 2　量子回路の振る舞いを計算する

- 量子ゲート X,Z,H
- ゲートから量子回路を構成する
- 二つのqubitを観測する量子回路

Surface code と Stabilizer **Agenda**

Part 3　古典コードから量子コードへ

- Hamming code
- 古典コードから量子コードへ
- Stabilizer Formalism


Part 1
量子エラー訂正技術の飛躍

量子エラー訂正技術の飛躍

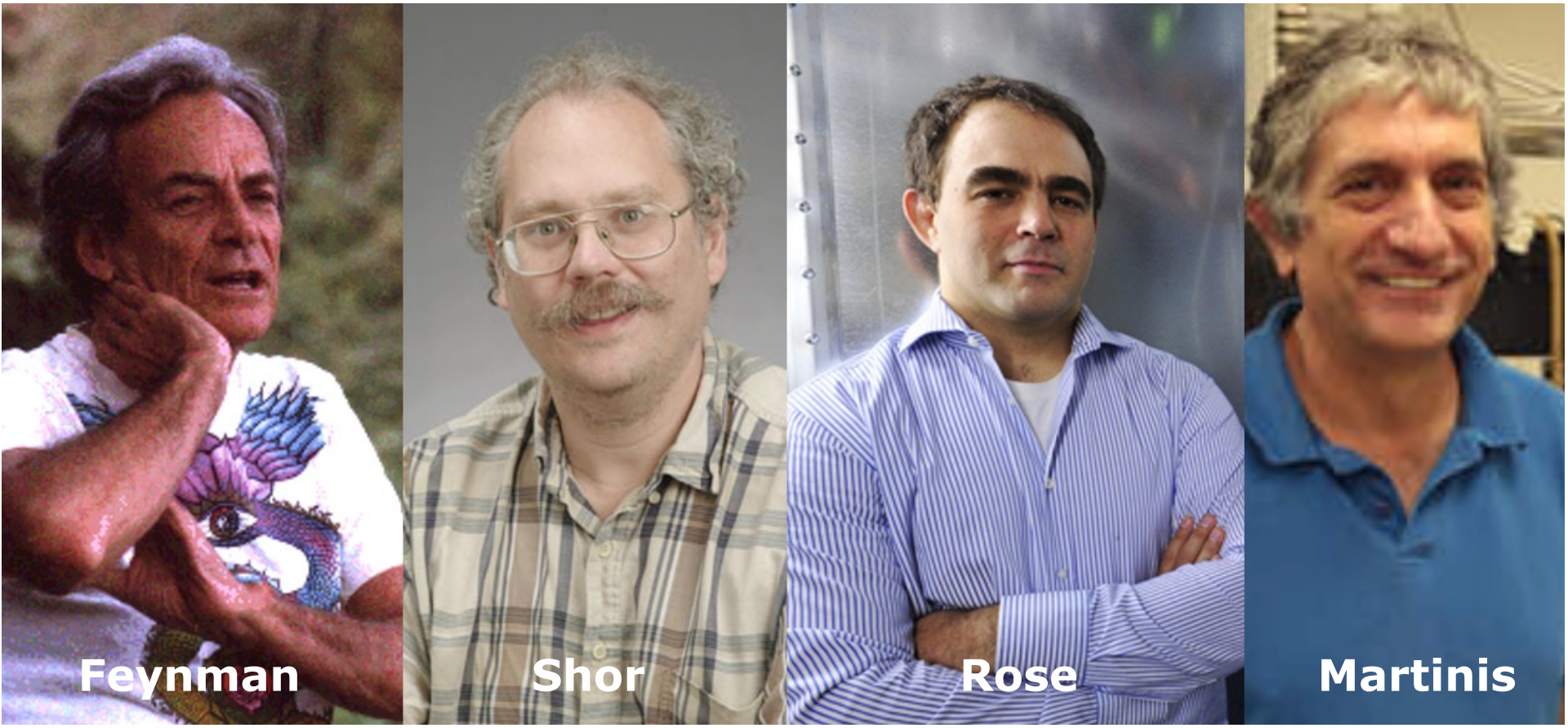
つい先日、MicrosoftによるMajorana粒子を用いたトポロジカル量子デバイスの発表という大きなニュースがあったばかりなのですが、今回のセミナーは、大きな変革期に差し掛かっている量子コンピュータ技術の動向を、量子エラー訂正技術の飛躍にフォーカスして、紹介しようとするものです。

Majorana/Topological のトピックスは含まれていませんが、そうした動きの背景を知る機会になると思います。

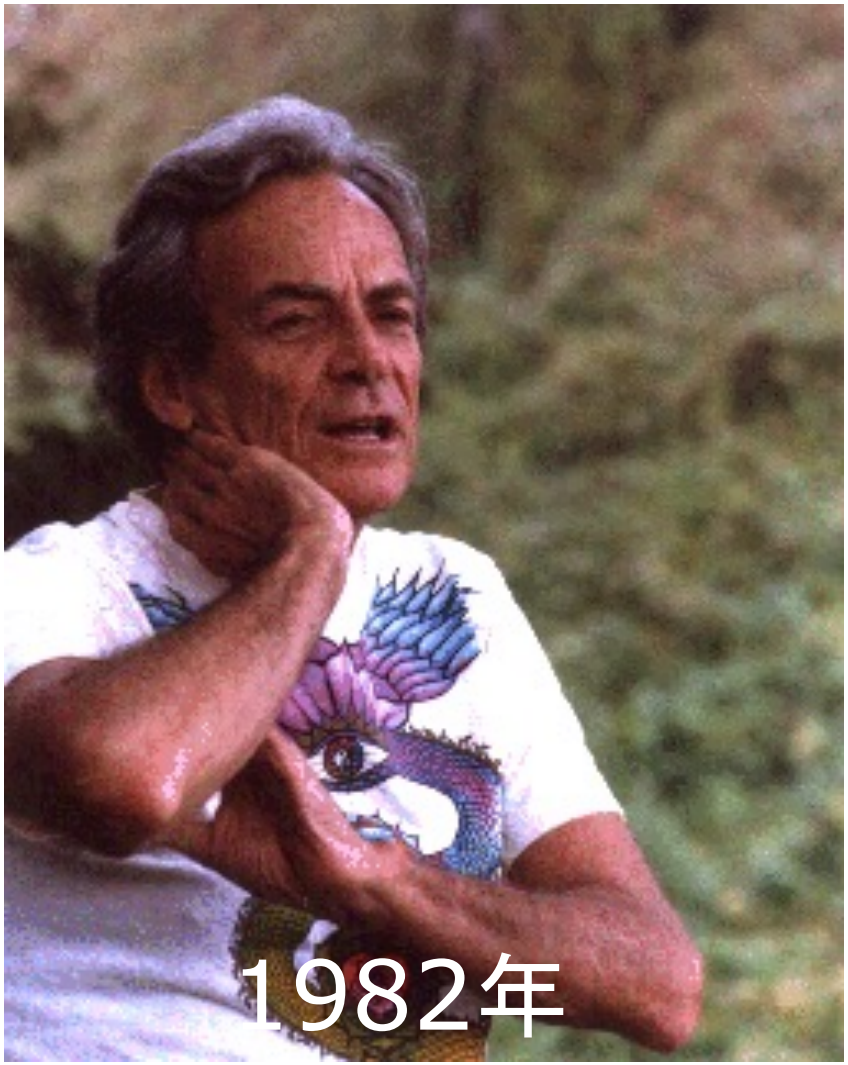
量子コンピュータの現在
我々は、今どこにいるのか？

「量子コンピュータ技術は大きな変革期に差し掛かっている」と述べたのですが、もう少し詳しく我々の現在の立ち位置を確認したいと思います。

量子コンピュータの歴史を振り返ると、時代を大きく画する4つの時期があることがわかります。


Feynman
Shor
Rose
Martinis

第一期：原理的可能性の探求の時代



第一の時期は、1982年にFeynmanが「自然をシミレートするという量子コンピュータ」というの量子コンピュータの初めてのビジョンを打ち出した量子コンピュータの創世記ともいうべき時期です。

第二期：多様で強力な
量子アルゴリズムの発見の時代

第二の時期は、1994年のShorの「素因数分解の量子アルゴリズム」の発見を代表とする様々な量子アルゴリズムの研究が活発に展開された時期です。

この時期に量子コンピュータが理論的にはどういう能力を持つのかという認識は大きく広がりました。その可能性は素晴らしいものに思えました。

ただ、当時そうした量子アルゴリズムを実装する量子コンピュータは現実には存在しませんでした。

ショア

量子コンピュータによる
素因数分解アルゴリズム
の発見

1994年

D-Waveマシンと D-Wave社創設者 Geordie Rose

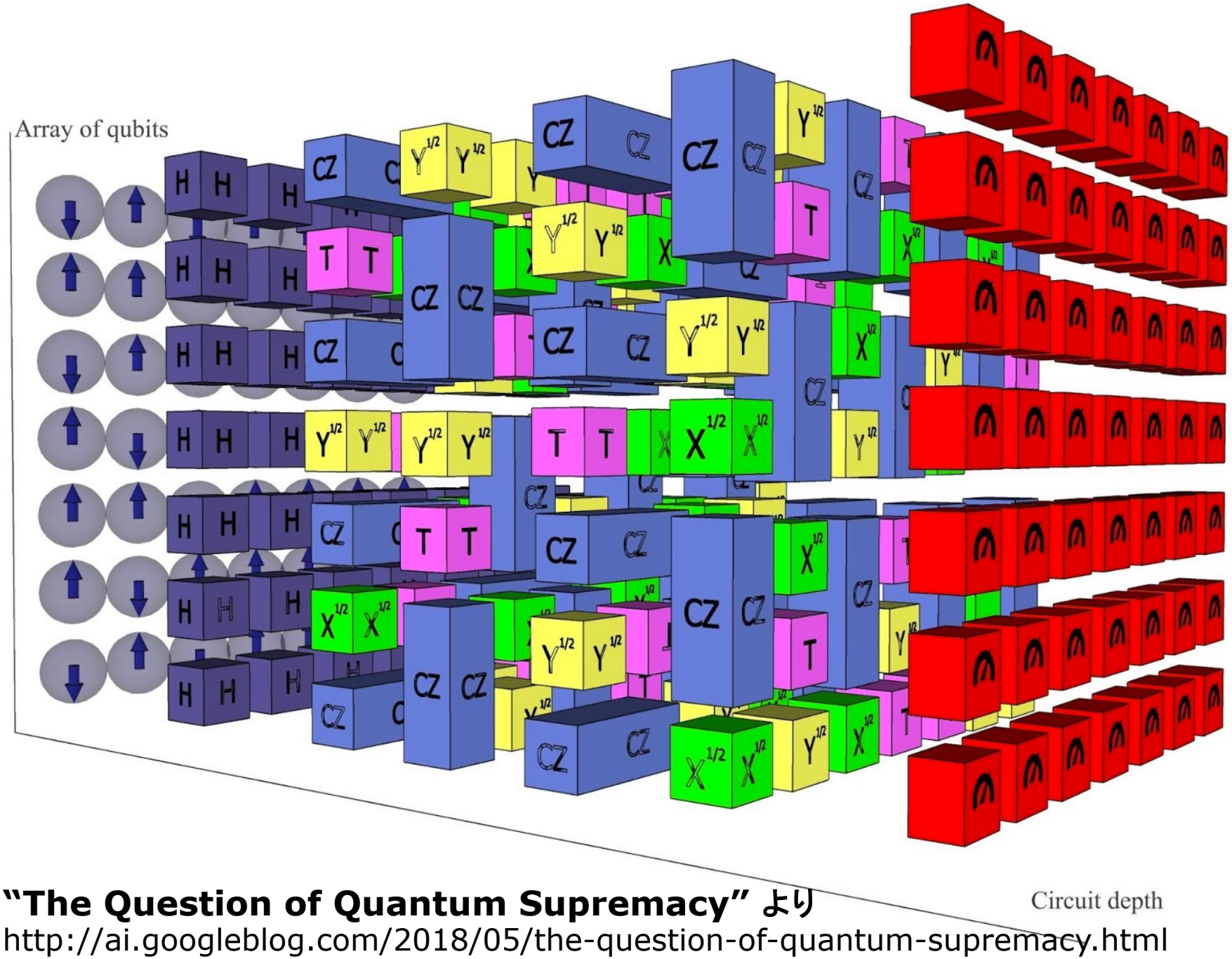
第三期:「量子超越性の実証」の成功

第三の時期は、Shorの画期的な発見から四半世紀が過ぎた2019年に始まります。

Mulitinisに率いられたGoogleのチームが現実に量子コンピュータを制作し、理論の予想通り、量子コンピュータが古典的なコンピュータをはるかに上回る能力を持つことを実際に示したのです。

現実のものとなった量子コンピュータが、実際に計算を行って「量子超越性の実証」に成功したのは画期的なことです。

ただ、その「計算」の実態は、ランダムに構成された量子ゲートの組み合わせのランダムな出力の計測であって、意味のある量子アルゴリズムの実行ではありませんでした。



“The Question of Quantum Supremacy” より
http://ai.googleblog.com/2018/05/the-question-of-quantum-supremacy.html

量子コンピュータを作ることの難しさ

なぜ、21世紀になってようやく実現した量子コンピュータの最初の成功が、第二の時期に認識の上では獲得していた夢のような量子アルゴリズムの実行ではなかったのでしょう？

問題は明確でした。例えば、Shorのアルゴリズムを実行するような規模の量子コンピュータを、技術的に我々は作ることができなかったからです。

量子コンピュータは、外部のノイズに対して非常に脆弱で、小中規模の量子デバイスでも、すぐにエラーを起こして機能できなくなるのです。こうして、量子エラー訂正技術に大きな関心が寄せられるようになりました。

第四期：量子エラー訂正技術のブレークスルー

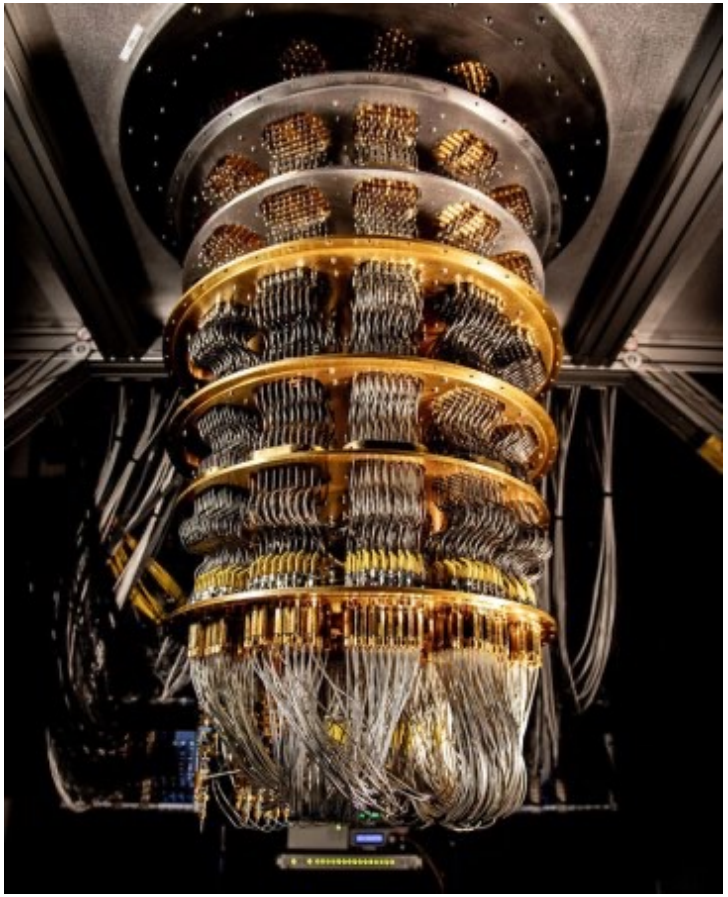
量子コンピュータの歴史の第四の時代は、つい最近始まったばかりです。

2024年、Google,とMicrosoft は、独立に量子エラー訂正技術の画期的なブレークスルーが達成されたとアナウンスします。冒頭で述べたMicrosoftの新量子デバイスも、こうした研究の流れの中に位置づけることができると思います。

一つのqubitレベルでも、量子エラー訂正技術のブレークスルーは、実践的には大きな意味を持ちます。それは、ノイズの下でも安定して稼働する、量子エラーに耐性を持つresilientな量子コンピュータを実現する上での必要不可欠な一歩だからです。

NEWS QUANTUM PHYSICS

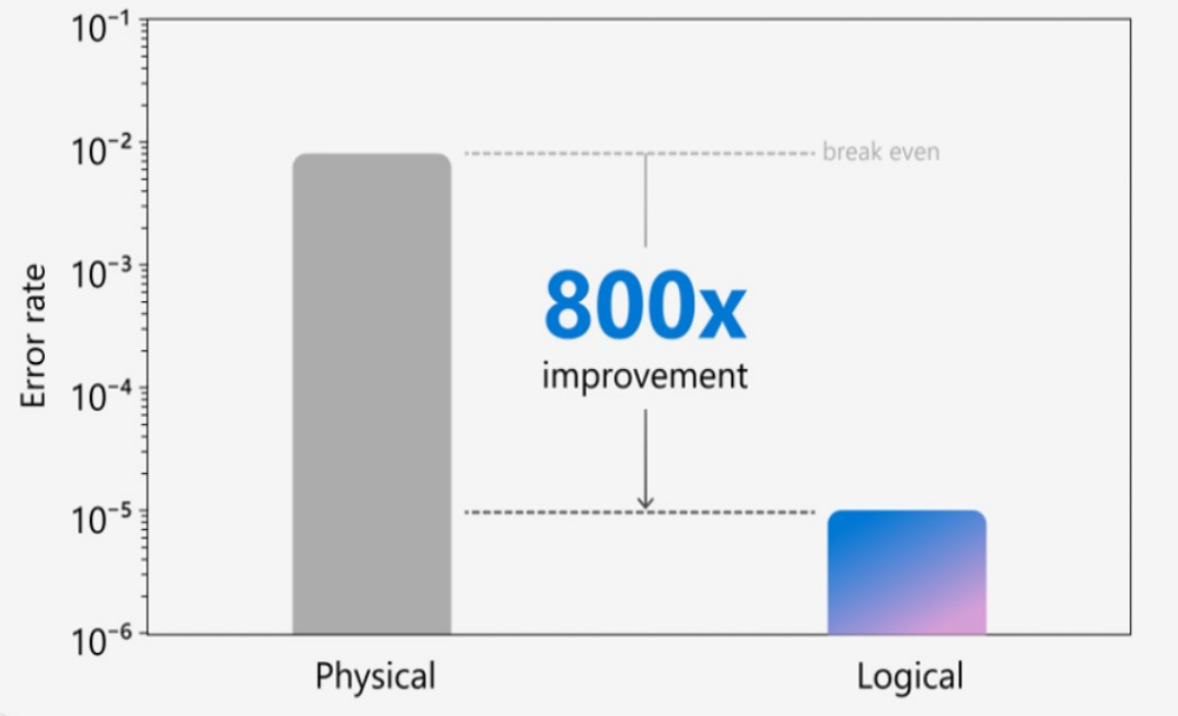
Google's quantum computer reached an error-correcting milestone

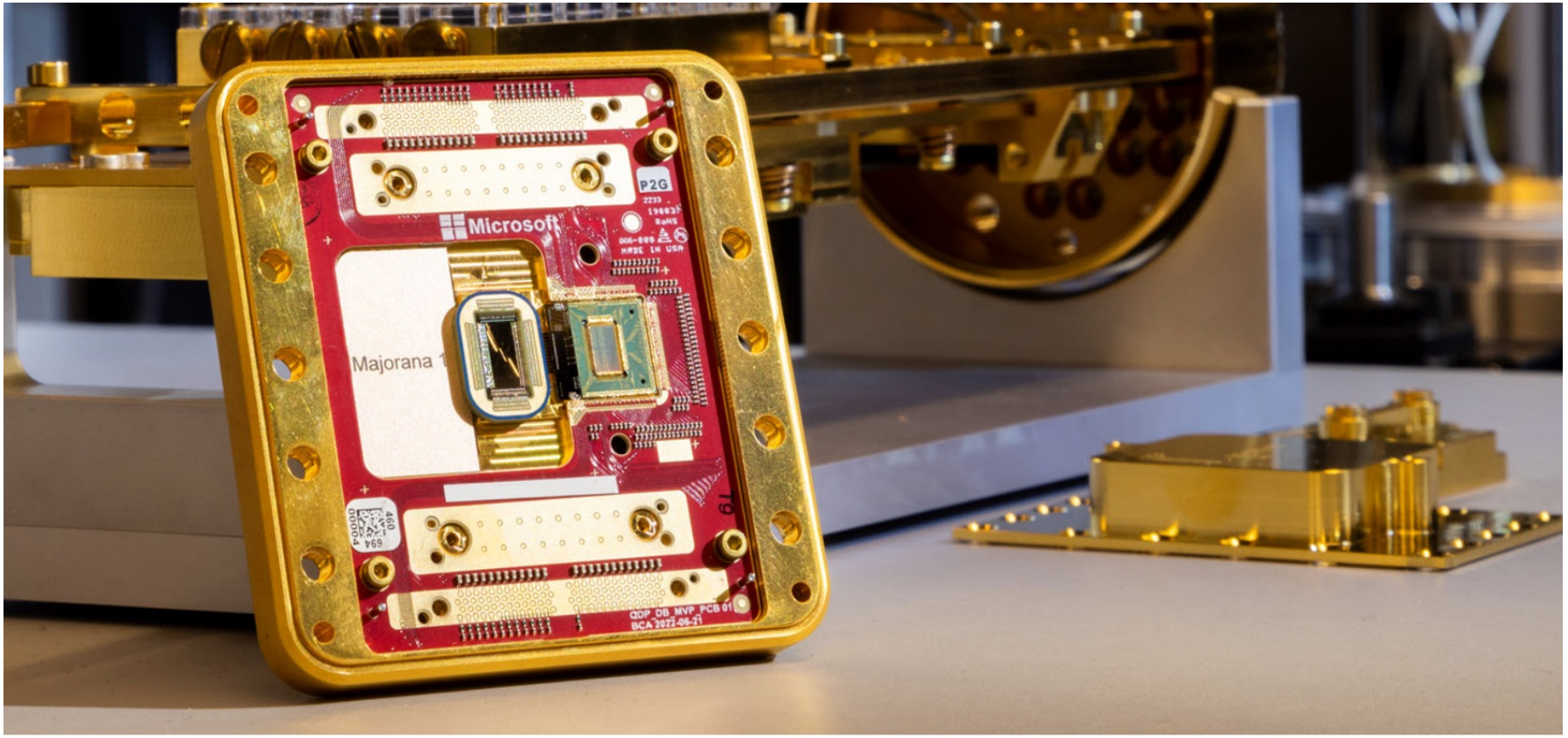


Microsoft And Quantinuum Improve Quantum Error Rates By 800x

Paul Smith-Goodson Contributor
Moor Insights and Strategy Contributor Group

Follow





Microsoft's Majorana 1 chip carves new path for quantum computing

https://news.microsoft.com/source/features/innovation/microsofts-majorana-1-chip-carves-new-path-for-quantum-computing/

新しい時代へ

現在進行中のこの歩みは、様々な量子アルゴリズムを実行可能にする量子コンピュータの規模拡大を可能にする scalabieな量子コンピュータの時代へと我々を導いていくでしょう。

おそらく 10年程度の時間で、scalableな量子コンピュータが実現するだろうと僕は考えています。

量子コンピュータの歴史の中で、私たちは大きな転換点に立っているのです。

Surface code と Stabilizer

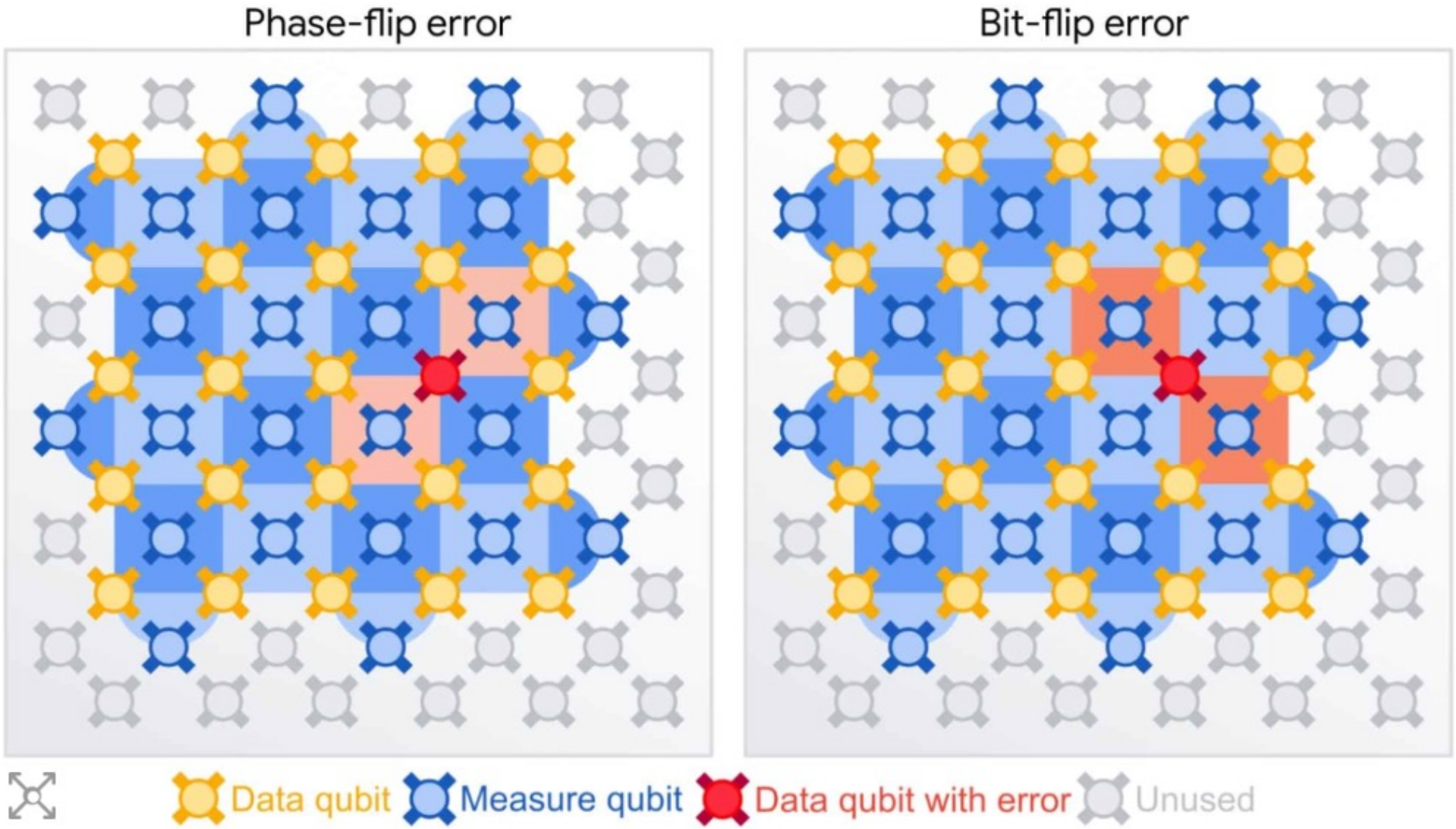
今回のセミナーのテーマの「Surface code と Stabilizer」は、現在の量子コンピュータの進化の原動力である「量子エラー訂正技術」の紹介を目指したものです。

今回のセミナーは、そうした技術の理論的な基礎である Stabilizer にフォーカスしています。

次回のセミナーの「Surface code と Stabilizer 2」では、Surface code にフォーカスしようと思っています。Surface codeは、どのようにscalableな量子コンピュータを構築するのかという重要な展望に、一つの解答を与えるものですj。

また、冒頭で紹介した Microsoftの新技術についても、ぜひ、セミナーをしたいと考えています。ご期待ください。

Surface Code



全画面表示を終了するには esc を押します

Introduction to surface codes

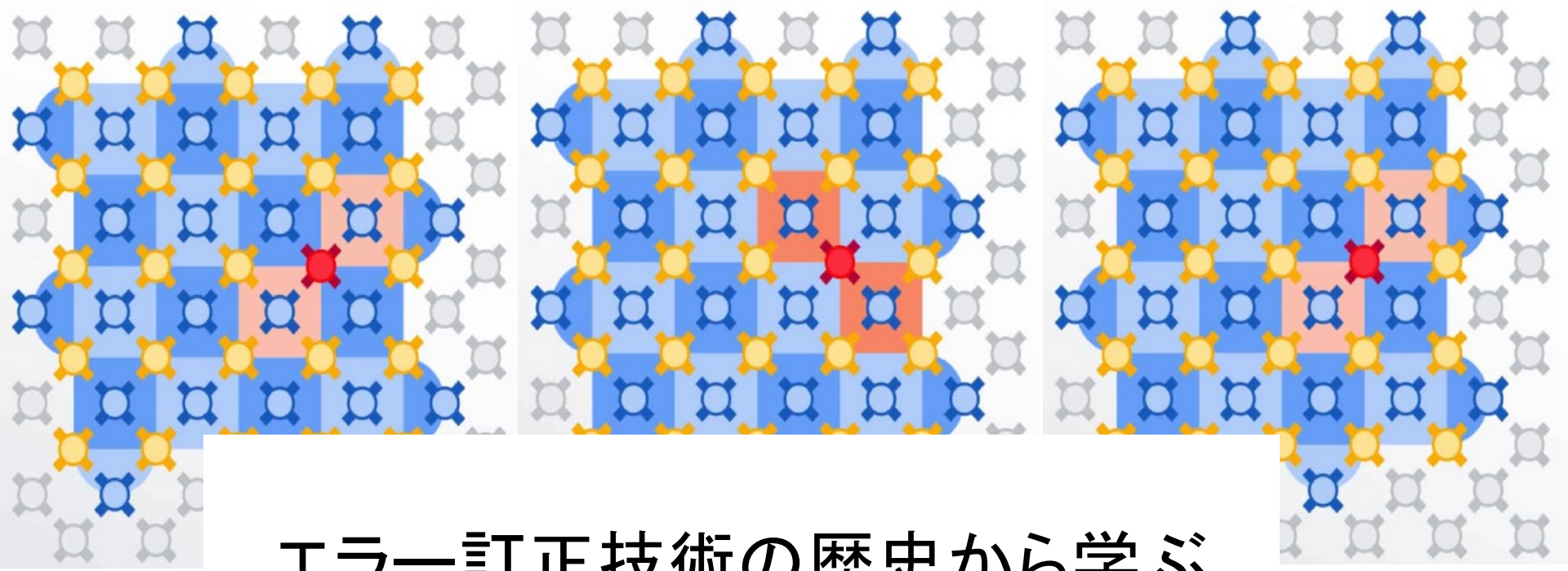
Alexei Kitaev

(Caltech / Google)

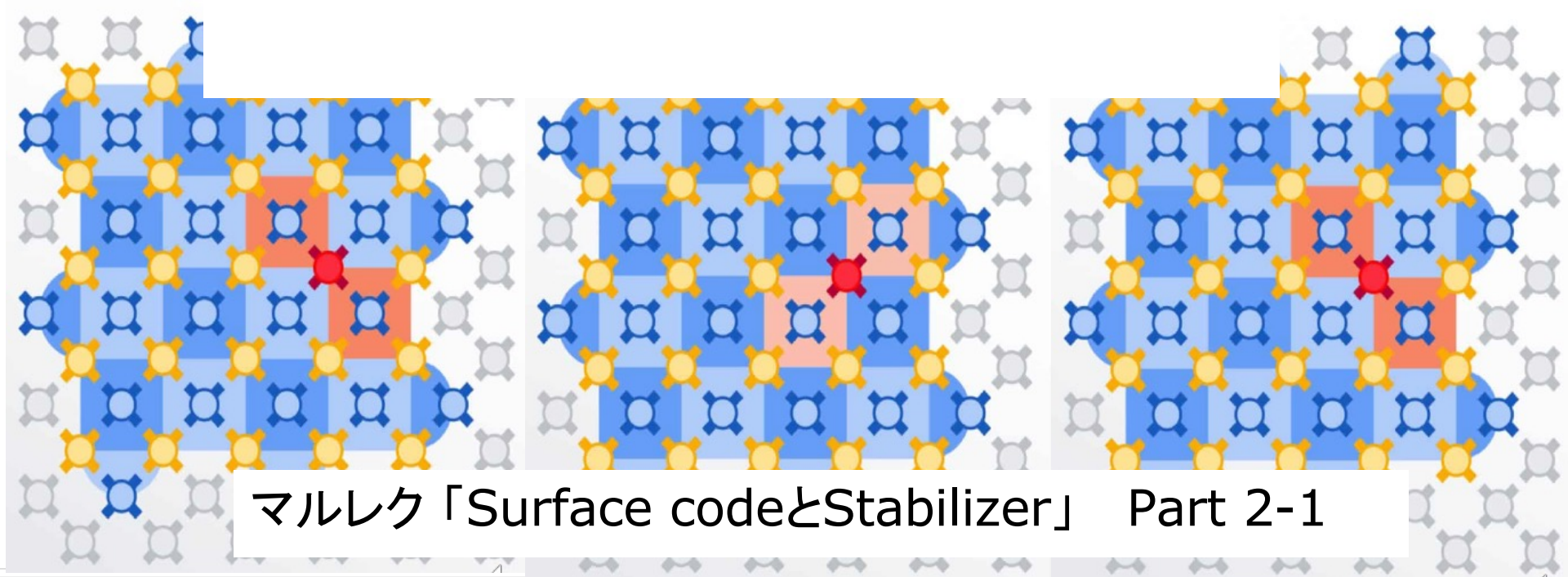
Quantum AI

Google Quantum AI Talks

https://youtu.be/M25fBmF9XR0



エラー訂正技術の歴史から学ぶ



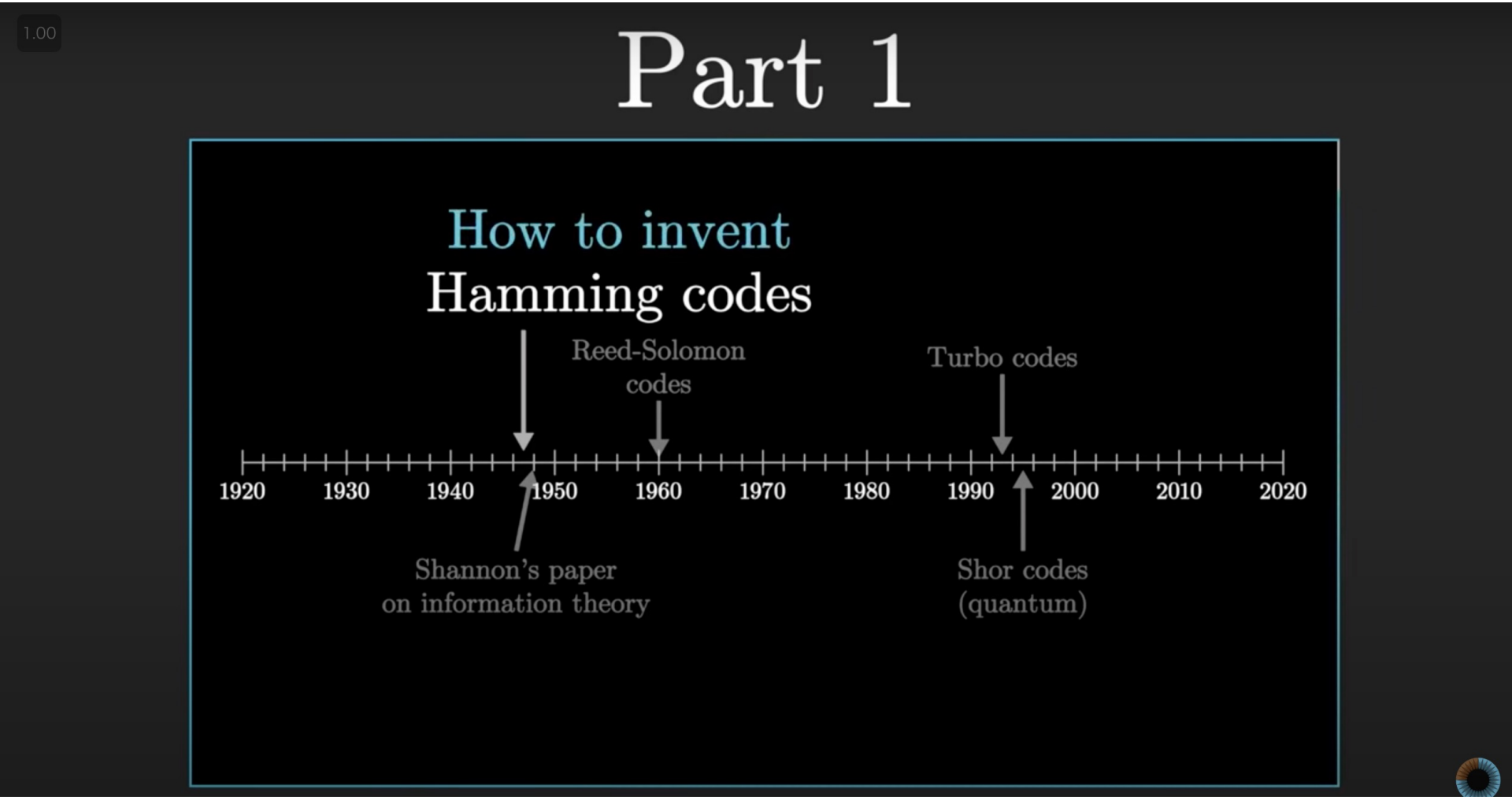
マルレク「Surface codeとStabilizer」　Part 2-1

エラー訂正技術の歴史から学ぶ

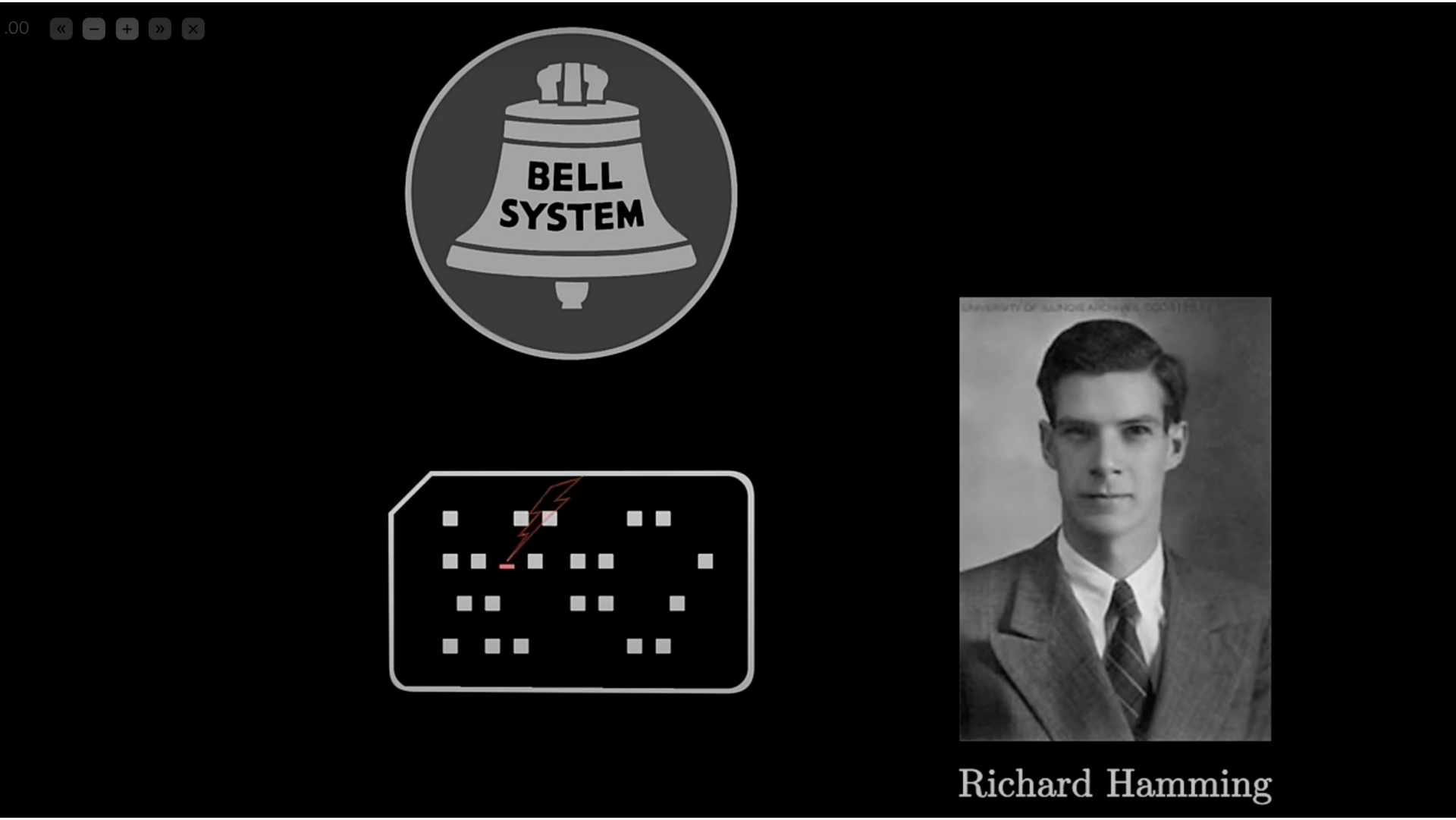
ここでははは。エラー訂正技術の歴史を簡単に振り返ってみようと思います。

今から75年以上前の「電話の時代」、Shannonが「情報理論」を初めて定式化した時代に、エラー訂正技術が生まれました。

当時Bell研にいた、Richard Hammingは、画期的なエラー訂正技術を発見しました。それが、Hamming codeです。Hammingの発見については Part 3 で紹介します。



How to send a self-correcting message
https://www.youtube.com/watch?v=X8jsijhllIA



How to send a self-correcting message

https://www.youtube.com/watch?v=X8jsijhllIA

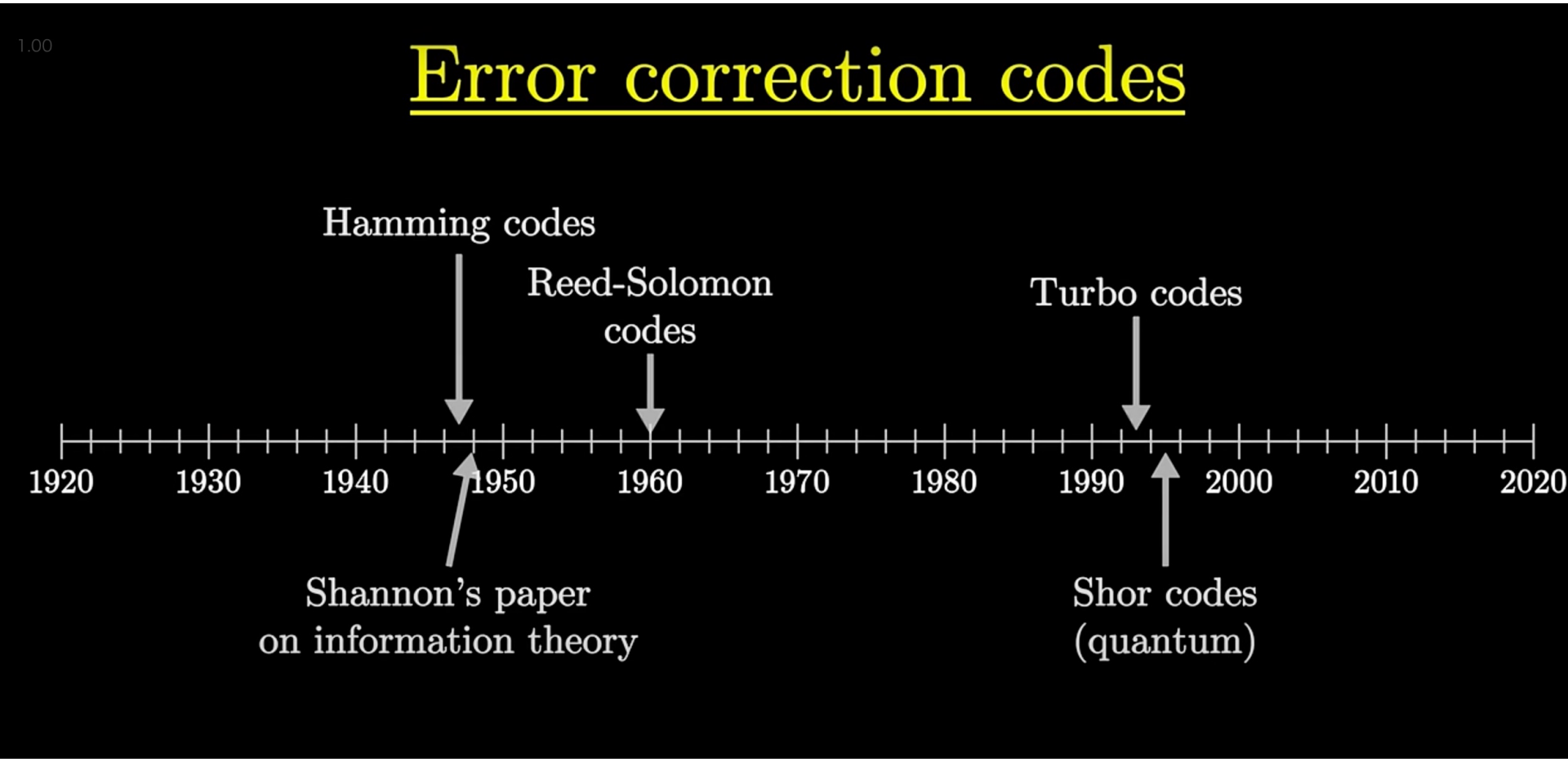
現代のIT技術とエラー訂正技術

現代のIT技術は、エラー訂正技術の技術の革新と深く結びついています。

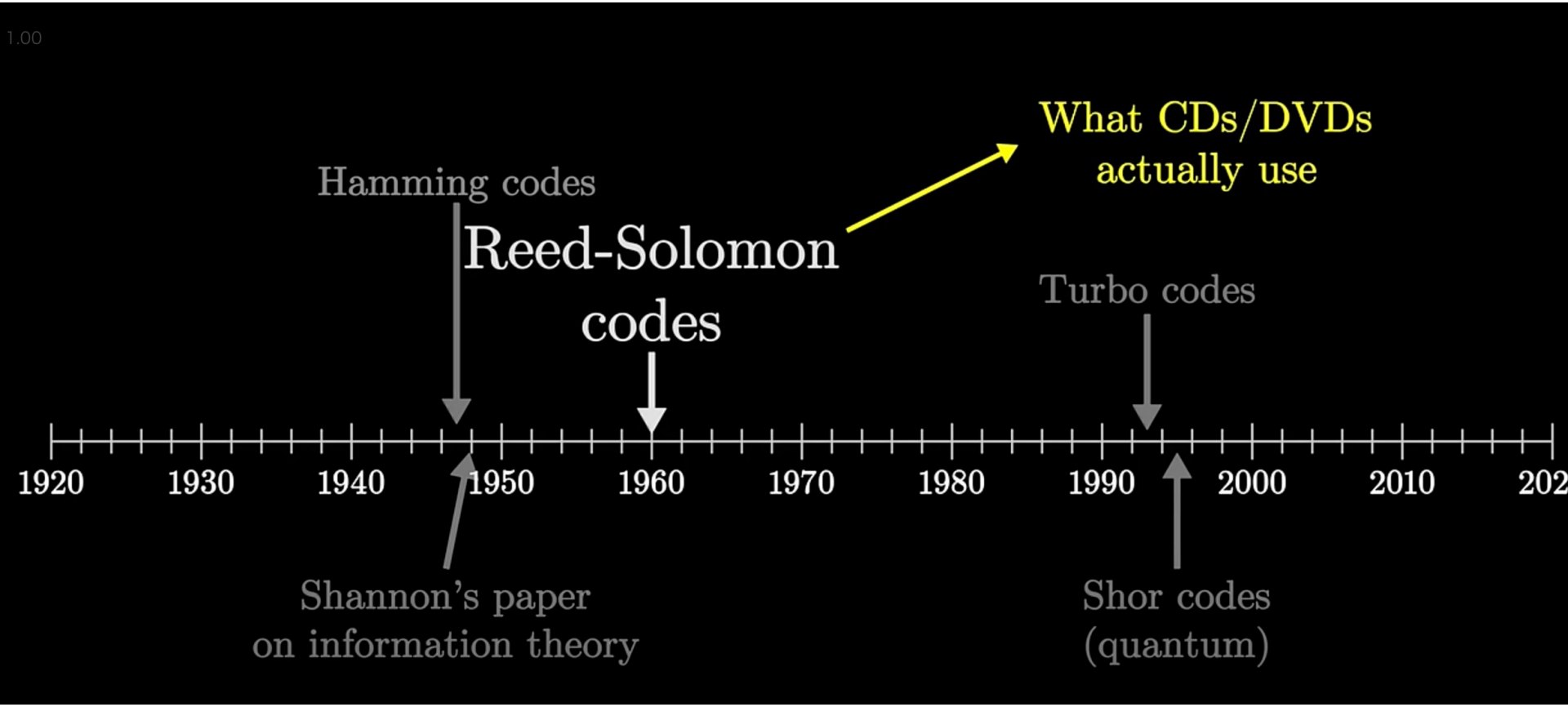
1960年代のReed-Solomon codeなしには、CDやDVDといった身近なストレージ技術の発展はありえなかったし、1990年代のTurbo codeなしには、21世紀初頭のモバイル通信の爆発的な普及は、起こり得なかったと思います。

5G/6GやStarLinkといった通信技術の目覚ましい発展の基礎には、エラー訂正技術の進歩が大きな役割を果たしています。

そうした革新は、現在も続いています。



How to send a self-correcting message
https://www.youtube.com/watch?v=X8jsijhllIA


1.00
What CDs/DVDs
actually use
Hamming codes
Reed-Solomon
codes
Turbo codes
1920
1930
1940
1950
1960
1970
1980
1990
2000
2010
202
Shannon's paper
on information theory
Shor codes
(quantum)

量子エラー訂正技術の登場

“THE PERFECT CODE” と言われたTurbo codeの発見と同じ時期に、量子エラー訂正技術のShor code が登場しているのは興味深いことです。

通信技術とは異なる領域に、エラー訂正技術の活躍の舞台が生まれたのだと思います。

ただ、それをエラー訂正技術の応用の拡大として捉えるのも、またそれを古典論から量子論へのフェーズの変化と考えるのも、少し不十分だと感じています。

情報の理論の深化

それは、単なるエラー訂正技術の問題ではなく、Shannonが創始した、情報の理論の深化の過程が継続的に進行しているのが本質だと思います。

エラー訂正技術の代表的な教科書の一つは、David McKayの”Information Theory, Inference, and Learning Algorithms” だと思います。

タイトルを見てもわかると思いますが、彼の視点は、広くて深いものです。20年前の本ですが、彼の問題意識は、現代にも通用するものです。

Information Theory, Inference, and Learning Algorithms

David J.C. MacKay
mackay@mrao.cam.ac.uk

https://www.inference.org.uk/itprnn/book.pdf#page=2.00

なぜ情報理論と機械学習を統合するのか？ それは、両者が同じコインの表裏として、一体のものだからである。

1960年代には、サイバネティクスという単一の分野に、情報理論家、コンピュータ科学者、神経科学者が集まり、共通の問題を研究していた。

情報理論と機械学習は今でも密接な関係にある。

脳は究極の情報の圧縮システムであり、通信システムである。

また、最先端のアルゴリズムは、データ圧縮とエラー訂正符号の両方の領域において、機械学習と同じツールを使用している。

-- David McKay

生成AIと量子コンピュータの時代の情報の理論へ

生成AIの分野の一部では、「蒸留」の手法が関心を集めています。

ただ、そもそも、情報の圧縮と意味への近似的で確率論的アプローチは、「意味の分散表現」「embedding表現」として、現代の大規模言語モデルの技術的中核をなすものです。

しかもそれは、Shannonの確率論的な情報の理論、「情報＝エントロピー」論の直系の子孫に他なりません。また、その確率論的性質に媒介されて、量子の世界と深くつながります。

生成AIと量子コンピュータの時代は、新しい情報の理論を生み出していくと確信しています。

Part 2

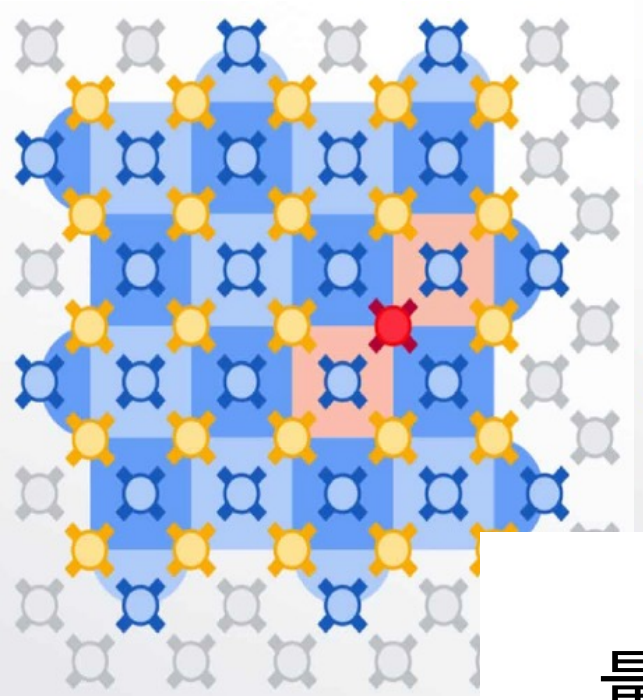
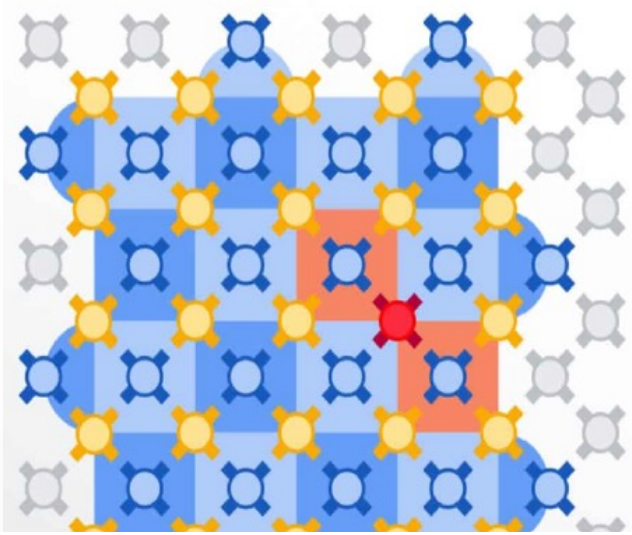
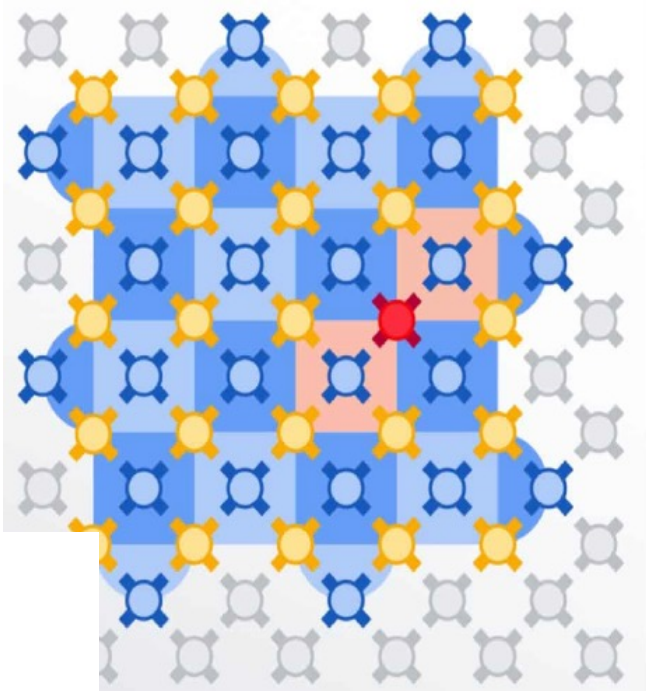
量子回路の振る舞いを計算する

Surface code と Stabilizer **Agenda**

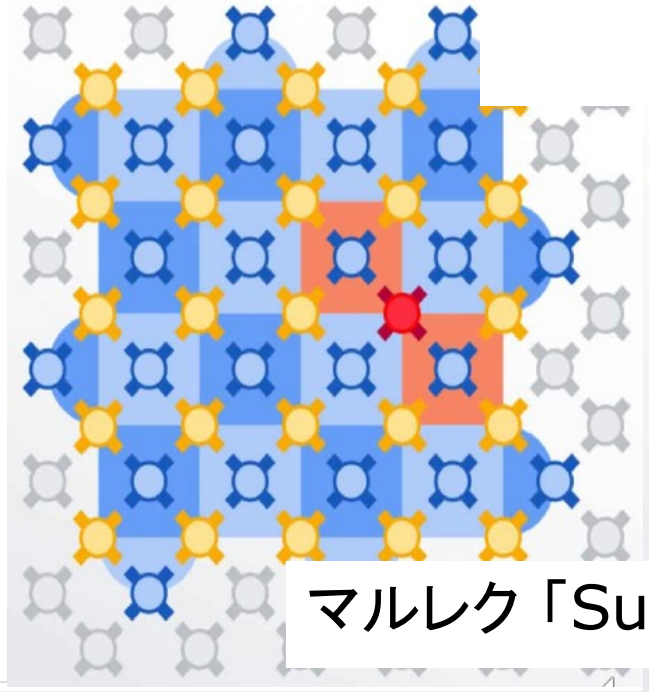
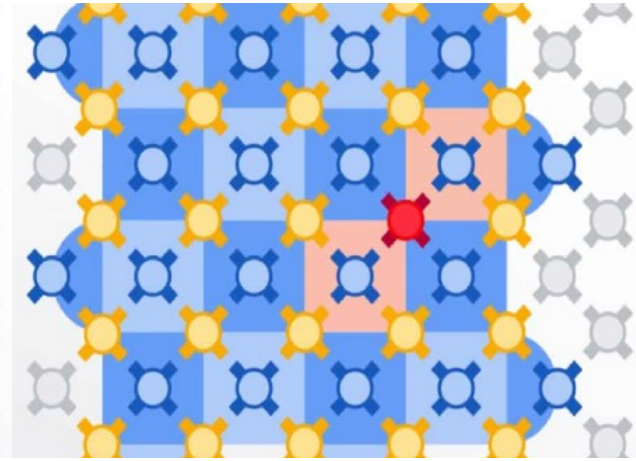
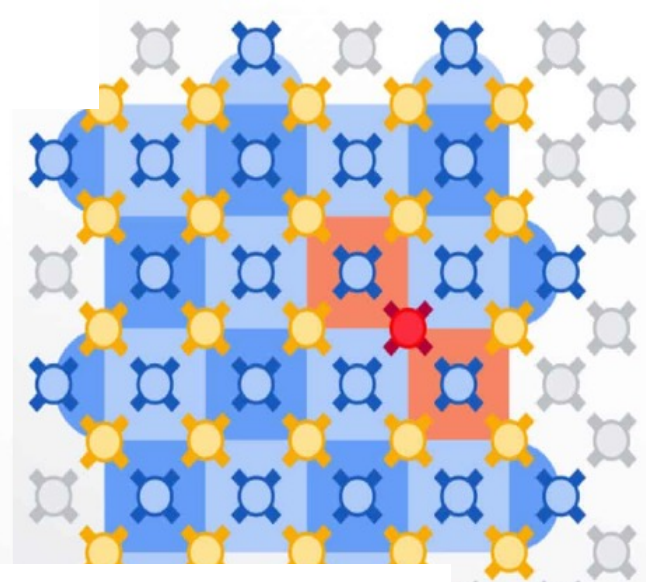
Part 2　量子回路の振る舞いを計算する

- 量子ゲート X,Z,H
- ゲートから量子回路を構成する
- 二つのqubitを観測する量子回路

Part 2「量子回路の振る舞いを計算する」では、現代の量子エラー訂正技術であるsurface codeやstabilizerの理解に向けて、その基礎である量子回路の振る舞いについて基本的な準備をしようと思います。

量子ゲート X,Z,H
基本の確認

マルレク「Surface codeとStabilizer」　Part 1-1

基本的な量子ゲートとその行列表現

基本的な量子ゲートX,Z,Hの行列での表現を見ておきましょう。

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

$$Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$$

$$H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$$

以下では、ゲートの行列表現とゲートとを、区別なく使うことがあります。Xはある時には「行列X」をさし、ある時には「ゲートX」を指します。コンテキストで、区別できると思います。

行列とそのエルミート共役

行列Mの行と列を入れ替え、成分の複素共役をとったものをMのエルミート共役といい、$M^{\dagger}$で表します。(「M ダガー」と読みます)

$$M = M^{\dagger}$$

が成り立つ時、Mをエルミート行列と言います。

$$UU^{\dagger} = U^{\dagger}\mathrm{U} = \mathrm{I}\ (\text{単位行列})$$

が成り立つ時、Uをユニタリ行列と言います。

行列X, Z, Hの基本的性質

この時、次のことがわかります。

$$X^\dagger = X\ ,\ Z^\dagger = Z\ ,\ H^\dagger = H$$

X, Z, Hはエルミート行列です。

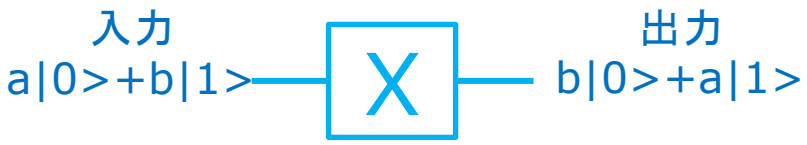
$$X^2 = Z^2 = H^2 = I$$

$$X^\dagger X = X X^\dagger = X^2 = \mathrm{I}$$
$$Z^\dagger Z = Z Z^\dagger = Z2 = \mathrm{I}$$
$$H^\dagger H = H H^\dagger = H^2 = \mathrm{I}$$

X, Z, Hはユニタリ行列です。

X, Z, H の行列と対応する量子ゲートの働き

$$X\begin{pmatrix} a \\ b \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}\begin{pmatrix} a \\ b \end{pmatrix} = \begin{pmatrix} b \\ a \end{pmatrix}$$



量子ゲートXを、ビットを反転させるので、
bit flipperと言います。

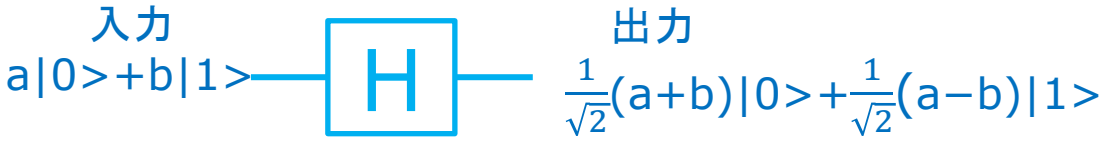
X, Z, H の行列と対応する量子ゲートの働き

$$Z\begin{pmatrix} a \\ b \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}\begin{pmatrix} a \\ b \end{pmatrix} = \begin{pmatrix} a \\ -b \end{pmatrix}$$

量子ゲートZを、phaseを反転させるので、
phase flipperと言います。

X, Z, H の行列と対応する量子ゲートの働き

$$H\begin{pmatrix} a \\ b \end{pmatrix} = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}\begin{pmatrix} a \\ b \end{pmatrix} = \frac{1}{\sqrt{2}}\begin{pmatrix} a+b \\ a-b \end{pmatrix}$$



標準基底とアダマール基底

標準基底 |0>, |1> に対して、

$$|+> = \frac{1}{\sqrt{2}}(|0> + |1>)$$
$$|-> = \frac{1}{\sqrt{2}}(|0> - |1>)$$

で定義される基底をアダマール基底と言います。

$$H|0> = |+>$$
$$H|1> = |->$$

です。

これから、次の式が成り立つことがわかります。

$$X|+> = X(\frac{1}{\sqrt{2}}(|0> + |1>) = |+>$$

$$X|-> = X(\frac{1}{\sqrt{2}}(|0> - |1>) = -|->$$

$$Z|+> = Z(\frac{1}{\sqrt{2}}(|0> + |1>) = |->$$

$$Z|-> = Z(\frac{1}{\sqrt{2}}(|0> - |1>) = |+>$$

Observable X,Zの固有値と固有ベクトル

先の計算

$$X|+> = |+>$$
$$X|-> = -|->$$

から、

Xは固有値 +1と−1を持ち、
固有値 +1の時、固有ベクトルは、|+>
固有値 −1の時、固有ベクトルは、|−>
となることがわかります。

Zについては

$$Z|0> = |0>$$
$$Z|1> = -|1>$$

から、

Zは固有値 +1と−1を持ち、
固有値 +1の時、固有ベクトルは、|0>
固有値 −1の時、固有ベクトルは、|1>
となることがわかります。

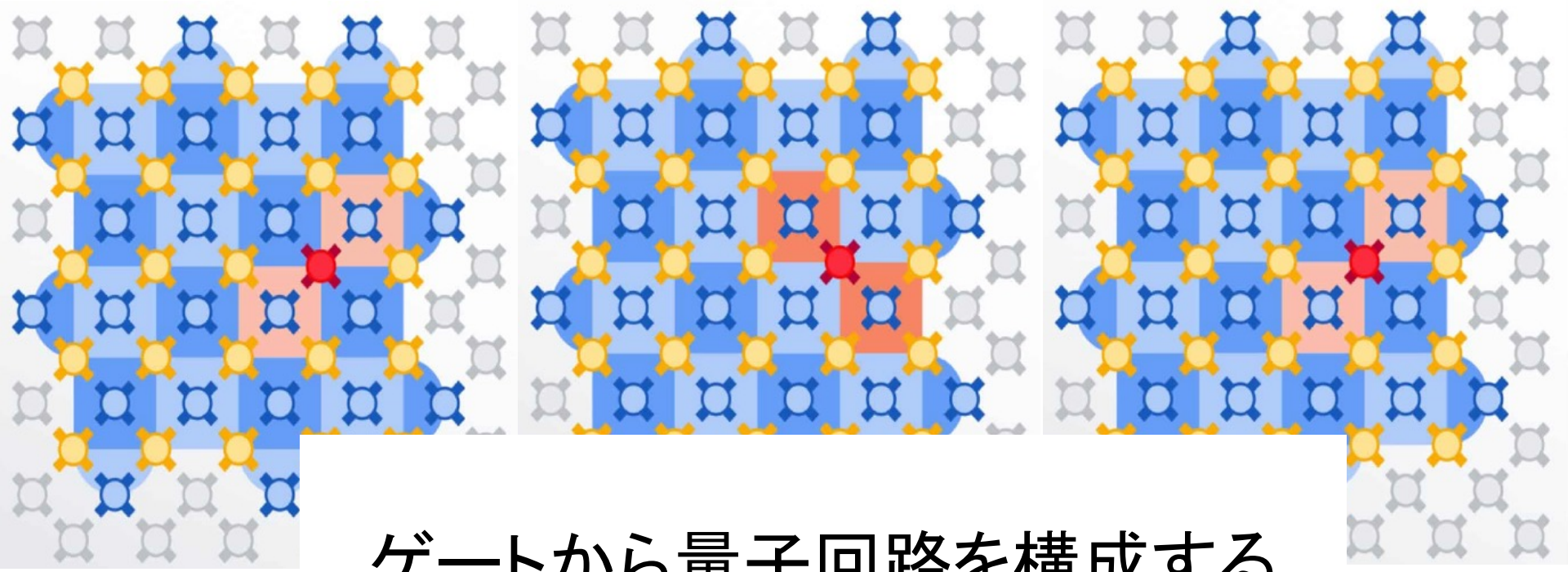
X,Z,Hの関係

簡単な計算で次のことを確かめることができます。

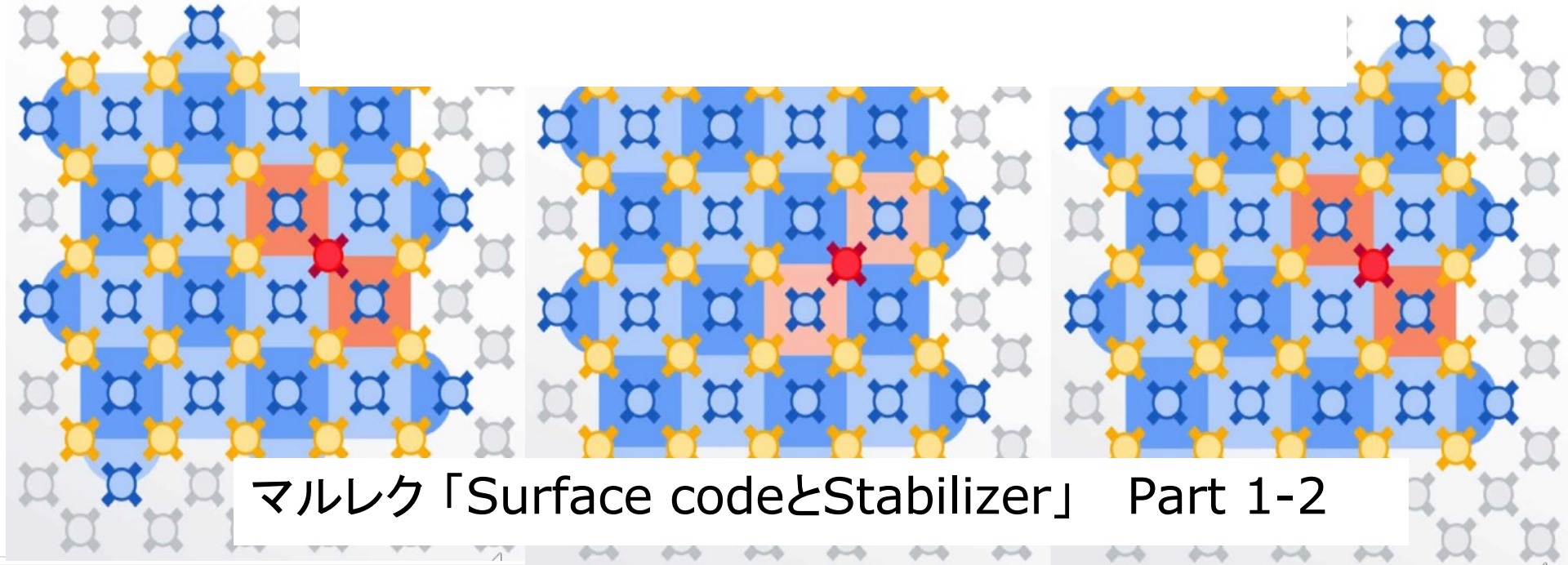
$$XZ = -ZX$$

$$HXH = Z$$

$$HZH = X$$



ゲートから量子回路を構成する



マルレク「Surface codeとStabilizer」　Part 1-2

複数の量子ゲートをつなげて量子回路を作る

複数の量子ゲートをつなげて量子回路を作ることができます。
基本的には、次の二つのパターンでゲートを配置します。

- ゲートを直列に配置する。
- ゲートを並列に配置する。

複数の量子ゲートをつなげて量子回路を作る

1-qubitのゲートを直列につなげて構成された量子回路の働きを理解するのは、比較的容易です。

- 入力に対して、ゲートに対応する行列を順番にかけていけば、出力は計算できる。（行列の積を計算する）
- このとき、構成されるラインをレジスターということがあります。レジスターの値は、ゲートを通過するたびに変化すると考えればいい。

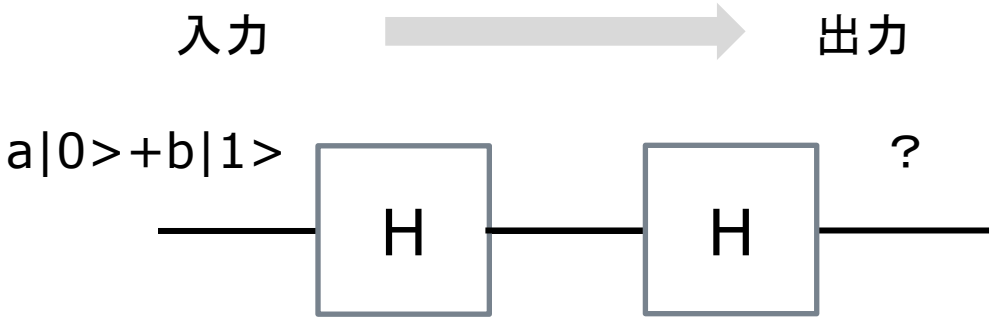
ゲートが並列に配置されている場合は、少し複雑です。

- 基本的には、回路を構成する複数のパラレルなレジスターのテンソル積を計算することになります

ゲートを直列に組み合わせる

ゲートを直列に組み合わせた回路の働きは、直列の回路を
構成するゲートの行列の積を計算することで求めることができます。

二つのHゲートを直列につないだ、次のような回路を考えてみよう。

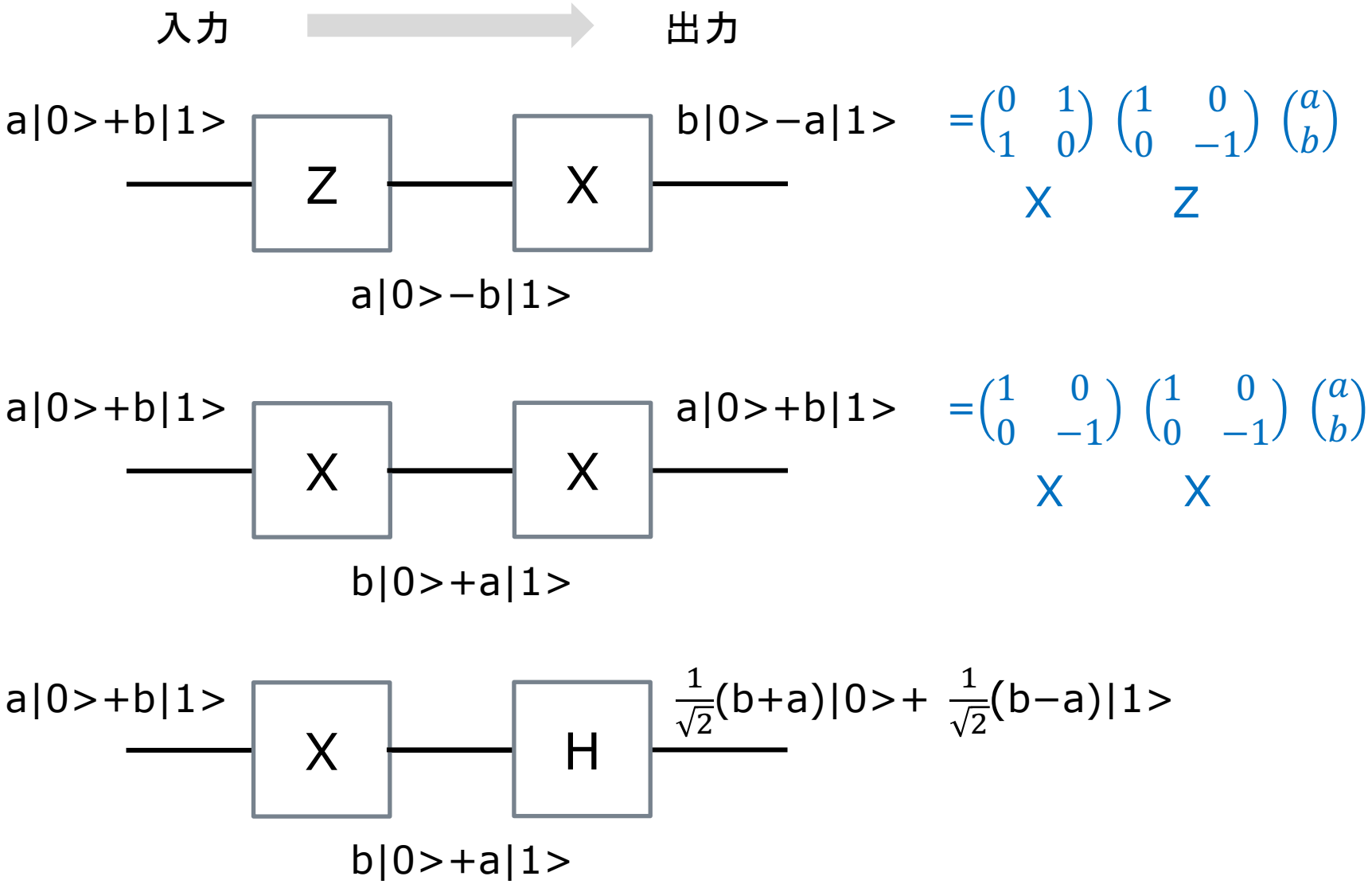


$H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$ なので、行列の積HHを計算する。

$$HH = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

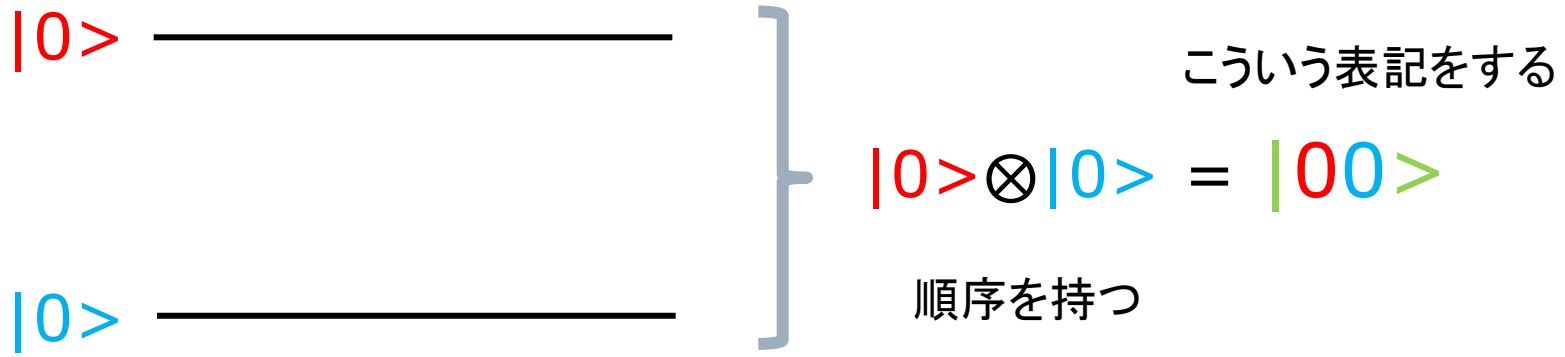
これは単位行列なので、先の回路の出力は、a|0>+b|1>

1-qubitのゲートを、直列に組み合わせる (1)



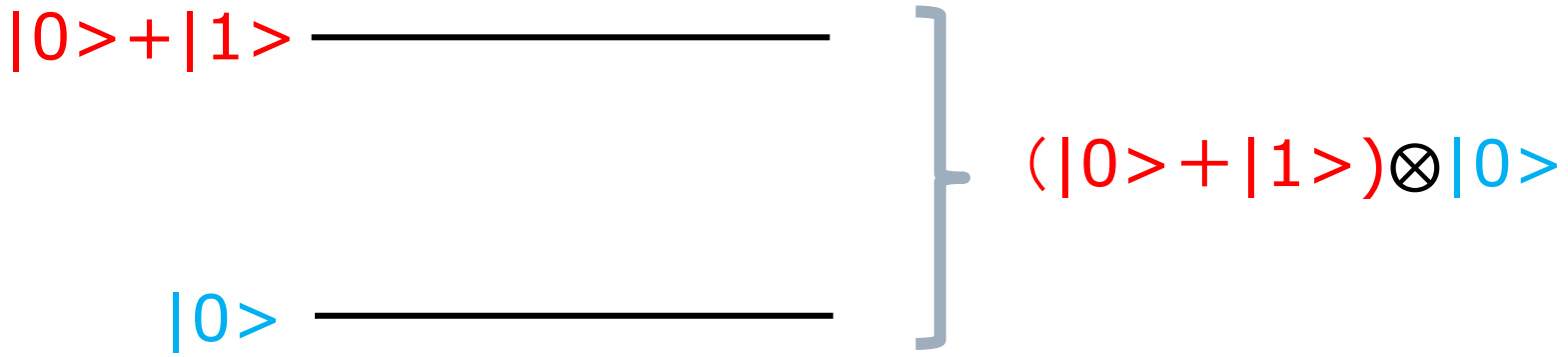
複数のレジスターを並列に組み合わせる

二つのレジスターのテンソル積の例　単純な例 1



複数のレジスターを並列に組み合わせる

二つのレジスターのテンソル積の例　単純な例 2

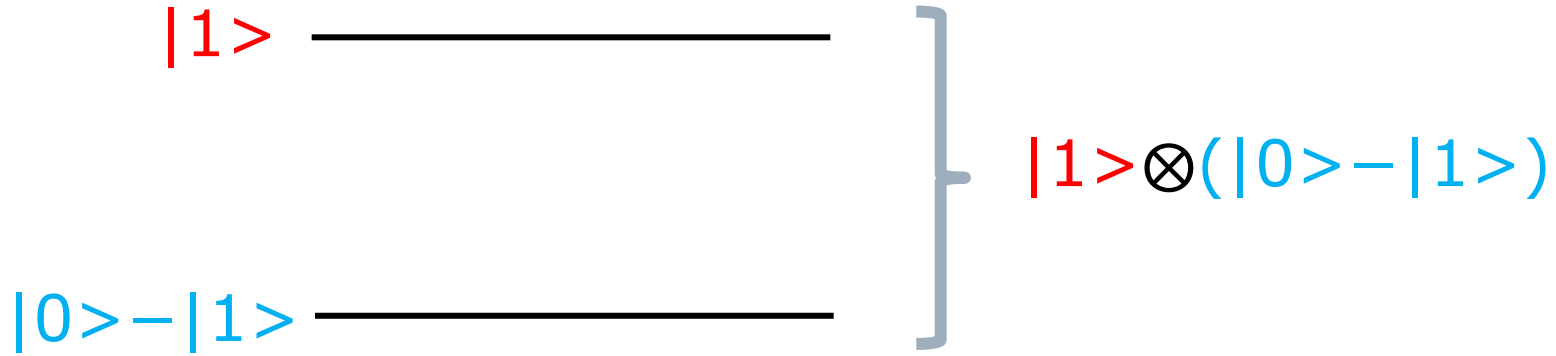


$$(|0\rangle+|1\rangle)\otimes|0\rangle$$
$$= |0\rangle\otimes|0\rangle + |1\rangle\otimes|0\rangle = |00\rangle+|10\rangle$$

$|0\rangle+|1\rangle$という形は本当はおかしい。$\alpha|0\rangle+\beta|1\rangle$の時、$|\alpha|^2+|\beta|^2=1$ という条件があるからだ。本当は、$\frac{1}{\sqrt{2}}|0\rangle+\frac{1}{\sqrt{2}}|1\rangle$なのだが、計算では無視している。

複数のレジスターを並列に組み合わせる

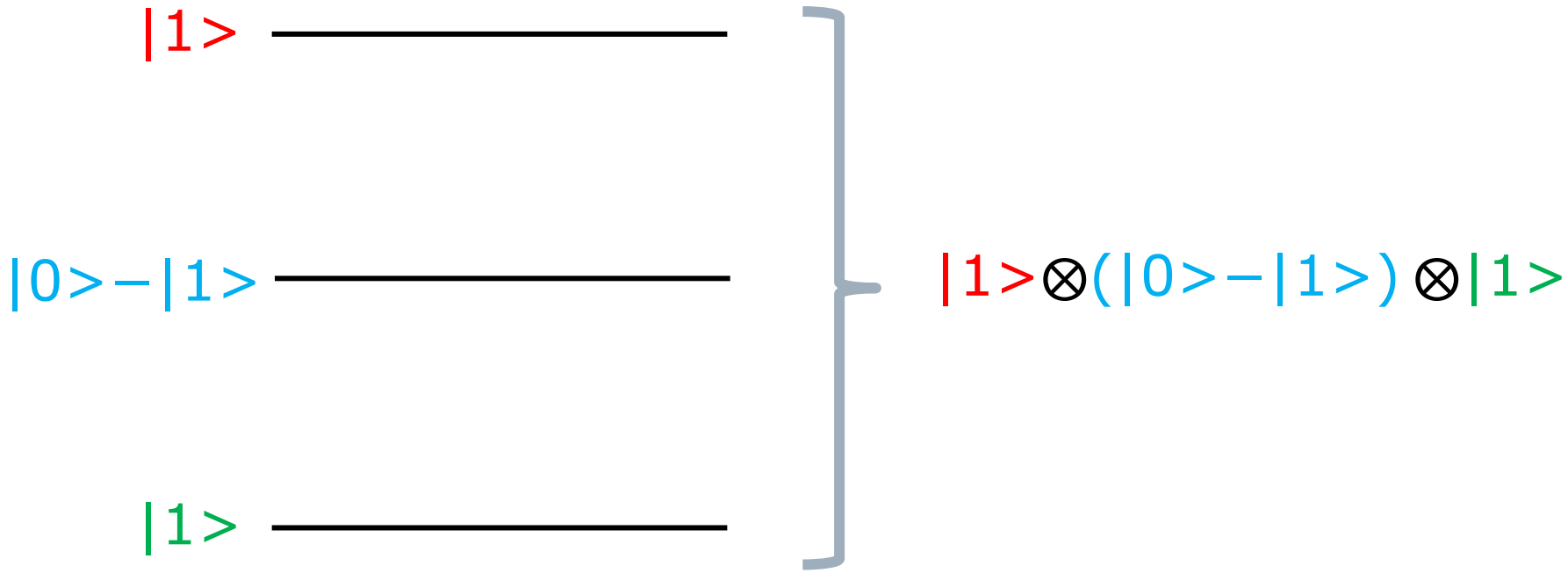
二つのレジスターのテンソル積の例　単純な例 3



$|1\rangle \otimes (|0\rangle - |1\rangle) =$
$|1\rangle \otimes |0\rangle - |1\rangle \otimes |1\rangle = |10\rangle - |11\rangle$

$|0\rangle - |1\rangle$という形は本当はおかしい。$\alpha|0\rangle + \beta|1\rangle$の時、$|\alpha|^2 + |\beta|^2 = 1$ という条件があるからだ。本当は、$\frac{1}{\sqrt{2}}|0\rangle - \frac{1}{\sqrt{2}}|1\rangle$なのだが、計算では無視している。

三つのレジスターのテンソル積の例　単純な例 4



$$
\begin{aligned}
&|1\rangle \otimes (|0\rangle - |1\rangle) \otimes |1\rangle \\
&= (|1\rangle \otimes |0\rangle - |1\rangle \otimes |1\rangle) \otimes |1\rangle \\
&= (|10\rangle - |11\rangle) \otimes |1\rangle = |101\rangle - |111\rangle
\end{aligned}
$$

行列のテンソル積

行列のテンソル積を次のように定義する。(2x2行列で例示)

$$A \otimes B = \begin{pmatrix} A_{11}B & A_{12}B \\ A_{21}B & A_{22}B \end{pmatrix}$$

Bの成分も書くと、

$$A \otimes B = \begin{pmatrix} A_{11}B_{11} & A_{11}B_{12} & A_{12}B_{11} & A_{12}B_{12} \\ A_{11}B_{21} & A_{11}B_{22} & A_{12}B_{21} & A_{12}B_{22} \\ A_{21}B_{11} & A_{21}B_{12} & A_{22}B_{11} & A_{22}B_{12} \\ A_{21}B_{21} & A_{21}B_{22} & A_{22}B_{21} & A_{22}B_{22} \end{pmatrix}$$

行列のテンソル積の例（1）

$A=\begin{pmatrix}1 & -1\\ 0 & 2\end{pmatrix}$, $B=\begin{pmatrix}1 & 2\\ 3 & 4\end{pmatrix}$ としよう。

$$A\otimes B=\begin{pmatrix}\mathbf{1} & \mathbf{-1}\\ \mathbf{0} & \mathbf{2}\end{pmatrix}\otimes\begin{pmatrix}1 & 2\\ 3 & 4\end{pmatrix}=\begin{pmatrix}\mathbf{1}\begin{pmatrix}1 & 2\\ 3 & 4\end{pmatrix} & \mathbf{-1}\begin{pmatrix}1 & 2\\ 3 & 4\end{pmatrix}\\ \mathbf{0}\begin{pmatrix}1 & 2\\ 3 & 4\end{pmatrix} & \mathbf{2}\begin{pmatrix}1 & 2\\ 3 & 4\end{pmatrix}\end{pmatrix}$$

$$=\begin{pmatrix}1 & 2 & -1 & -2\\ 3 & 4 & -3 & -4\\ 0 & 0 & 2 & 4\\ 0 & 0 & 6 & 8\end{pmatrix}$$

行列のテンソル積の例 (2)

$A=\begin{pmatrix}1 & -1\\ 0 & 2\end{pmatrix}$, $B=\begin{pmatrix}1 & 2\\ 3 & 4\end{pmatrix}$ としよう。

$$B\otimes A=\begin{pmatrix}\mathbf{1} & \mathbf{2}\\ \mathbf{3} & \mathbf{4}\end{pmatrix}\otimes\begin{pmatrix}1 & -1\\ 0 & 2\end{pmatrix}=\begin{pmatrix}\mathbf{1}\begin{pmatrix}1 & -1\\ 0 & 2\end{pmatrix} & \mathbf{2}\begin{pmatrix}1 & -1\\ 0 & 2\end{pmatrix}\\ \mathbf{3}\begin{pmatrix}1 & -1\\ 0 & 2\end{pmatrix} & \mathbf{4}\begin{pmatrix}1 & -1\\ 0 & 2\end{pmatrix}\end{pmatrix}$$

$$=\begin{pmatrix}1 & -1 & 2 & -2\\ 0 & 2 & 0 & 4\\ 3 & -3 & 4 & -4\\ 0 & 6 & 0 & 8\end{pmatrix}$$

テンソル積では、
$A\otimes B \neq B\otimes A$
である

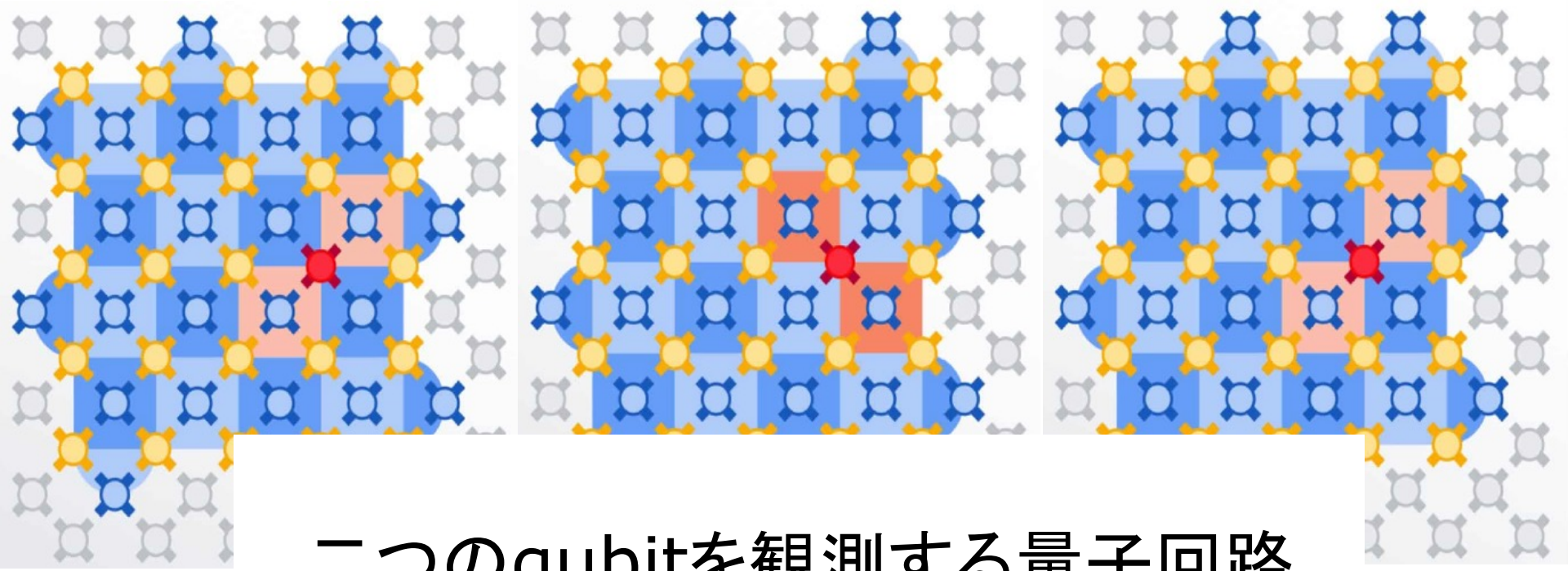
2-qubitの基底

$$|00\rangle = |0\rangle \otimes |0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \otimes \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

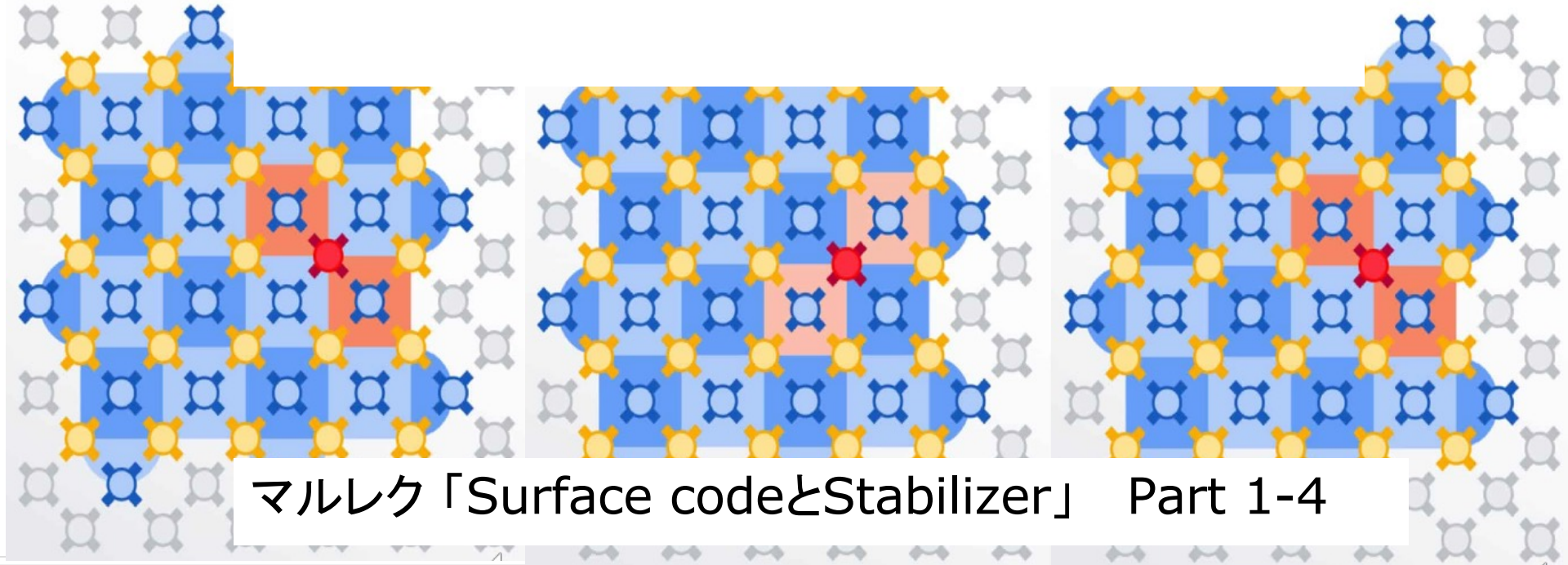
$$|01\rangle = |0\rangle \otimes |1\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \otimes \begin{pmatrix} 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix}$$

$$|10\rangle = |1\rangle \otimes |0\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix} \otimes \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \end{pmatrix}$$

$$|11\rangle = |1\rangle \otimes |1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix} \otimes \begin{pmatrix} 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}$$



二つのqubitを観測する量子回路



マルレク「Surface codeとStabilizer」　Part 1-4

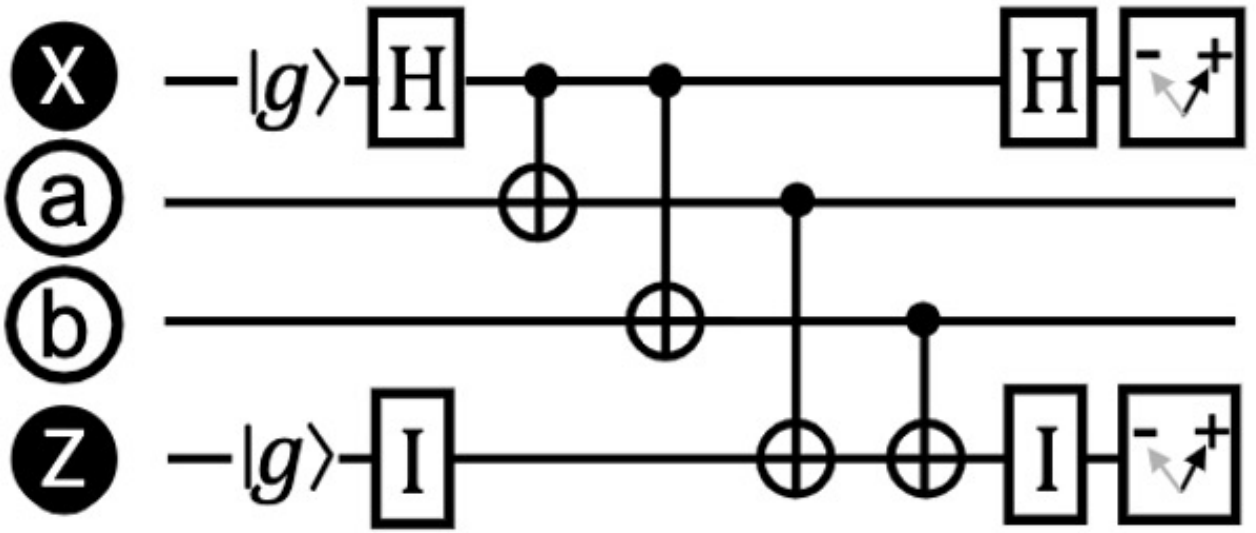
今回扱う量子回路

セミナーに向けたショート・ムービー「エンタングルメントと量子回路」では、二つのライン(レジスタ)を持ち、二つの量子ゲート（CNOTゲートとアダマール・ゲート）からなる量子回路の働きを考えてきました。

その回路は、二つの入力ラインから、|00＞,|01＞,|10＞,|11＞を入力として受け取った時、エンタングル状態（= Bell State）である|Φ+＞,|Ψ+＞,|Φ−＞,|Ψ−＞を、二つのライン上のエンタングルした状態として出力するものでした。

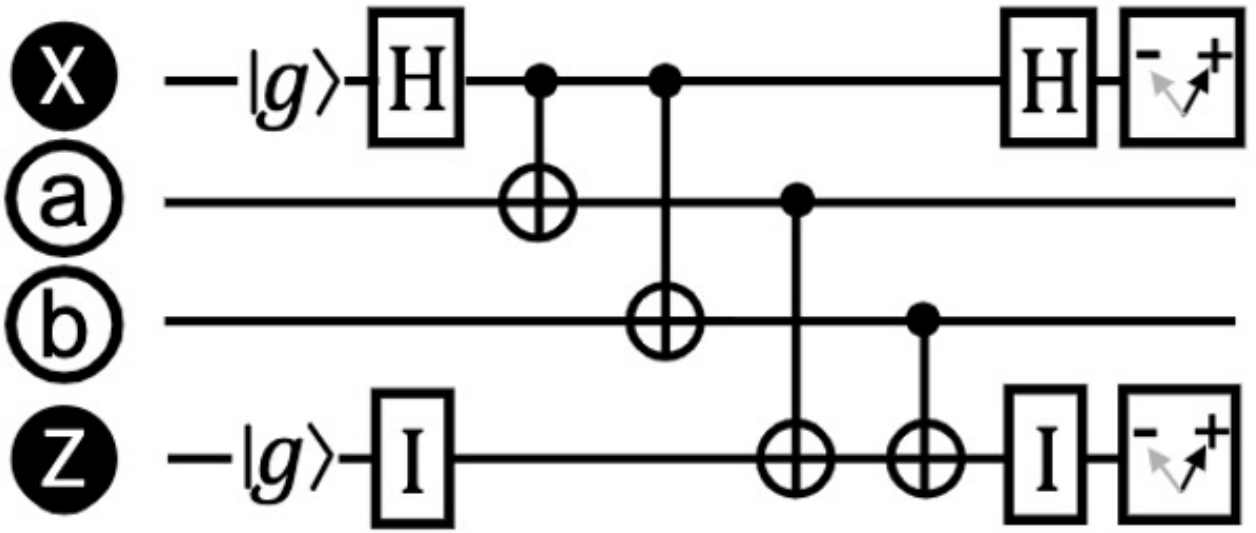
今回のセッションで扱う量子回路は、それより少し複雑です。

次のような形をしています。



この量子回路は、次の論文から借用したものです。

Fowler et al.“Surface codes: Towards practical large-scale quantum computation” https://arxiv.org/abs/1208.0928



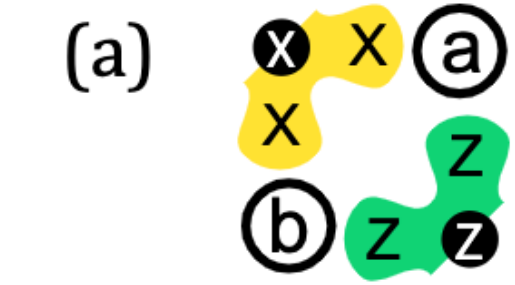
4つのレジスター X,a,b,Z があり、2つのアダマール・ゲート、2つのIゲート、4つのCNOTゲートから構成されています。

これ実は、2つのqubit a, b の状態を観測して、レジスター X, Z に出力するという量子回路なんです。

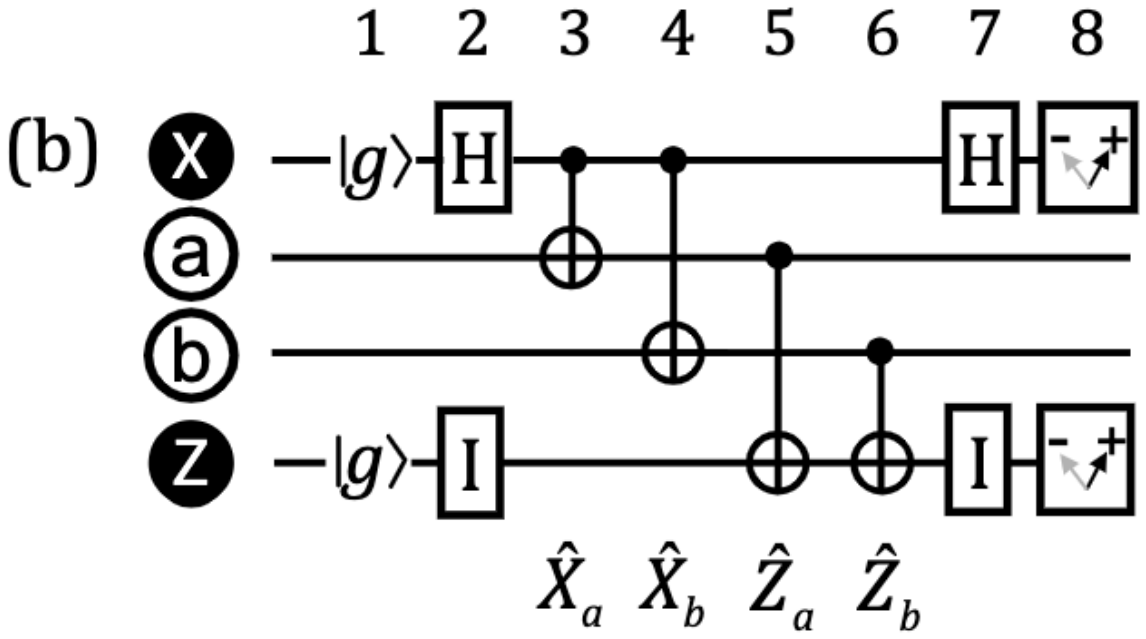
|g＞は、演算子Zの基底状態(ground state) で、|0＞のことと思って構いません。

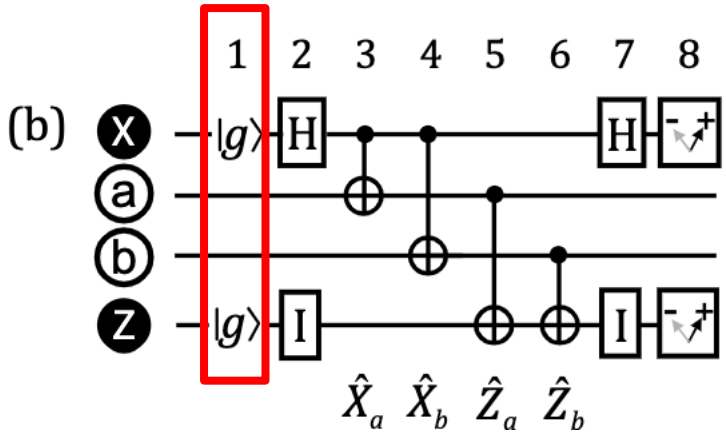
今回扱う量子回路 (b) とその模式図 (a)

a, b を
data qubit
という



X, Z を
measure qubit
という



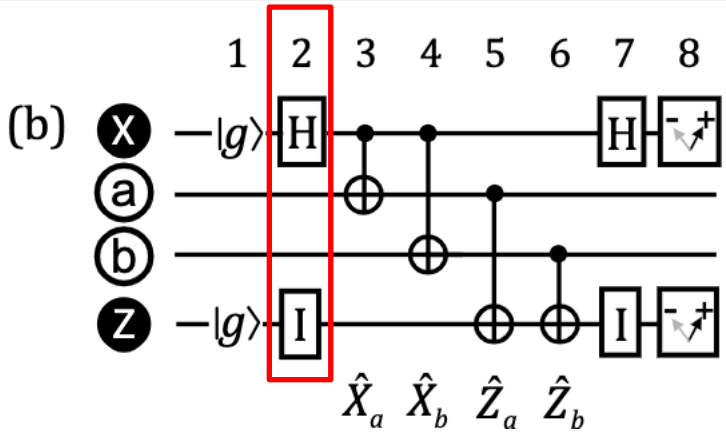


Step 1　　状態 $|\psi_1>$

$$|\psi_1> = |0> \otimes |\psi_a\psi_b> \otimes |0>$$

2-qubit システムの状態$|\psi_a\psi_b>$は一般に
$|\psi_a\psi_b> = A|00> + B|01> + C|10> + D|11>$と表せる

$$|\psi_1> = |0> \otimes (A|00> + B|01> + C|10> + D|11>) \otimes |0>$$
$$|\psi_1> = A|0000> + B|0010> + C|0100> + D|0110>$$



Step 2 状態 $|\psi_2>$

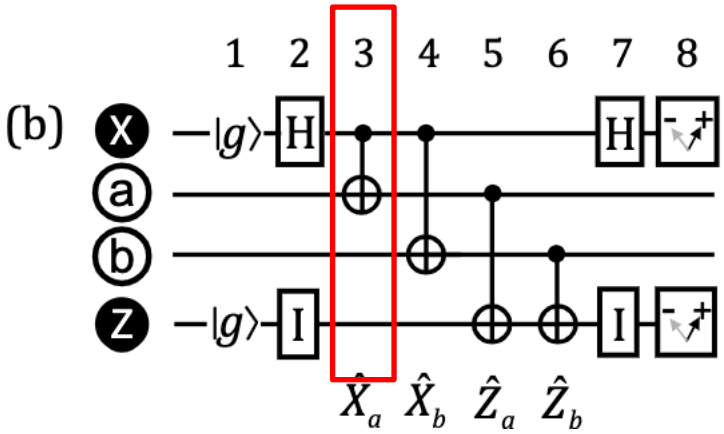
$$|\psi_1> = A|0000> + B|0010> + C|0100> + D|0110>$$

第一レジスターのmeasure-X qubitは、アダマールHゲートの作用を受け、次のように変化する。

$|0> \rightarrow |+> = |0> + |1>$ また、$|1> \rightarrow |-> = |0> - |1>$

第四レジスターの Iゲートは、measure-Z qubitの状態を変えない。

$$\begin{aligned} |\psi_2> &= A|0000> + A|1000> \\ &= B|0010> + B|1010> \\ &= C|0100> + C|1100> \\ &= D|0110> + D|1110> \end{aligned}$$



Step 3　　状態 $|\psi_3>$

Step 3のCNOTは、measure-X qubit(第一レジスタ)をコントロール qubitとし、data qubit a (第二レジスタ) をターゲットqubitとする。

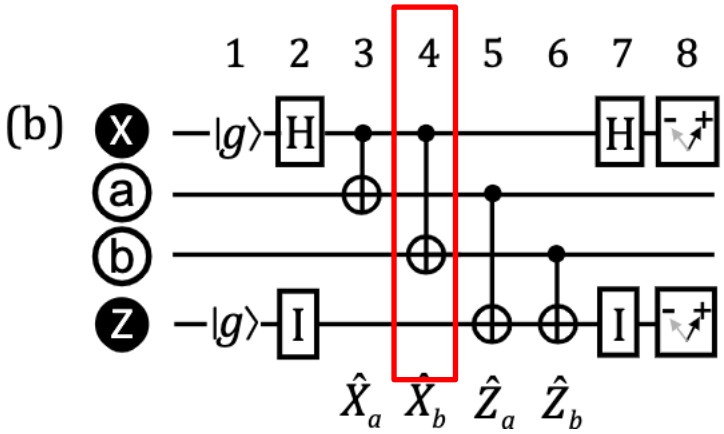
CNOTは、コントロール qubit cとターゲット　t のペア|c t>に対して、次のように作用する。次のように作用する。

|00> → |00>,　|01> → |01>
|10> → |11>, |11> → |10>

$$|\psi_2> =$$
$$A|0000> + A|1000>$$
$$= B|0010> + B|1010>$$
$$= C|0100> + C|1100>$$
$$= D|0110> + D|1110>$$

$$|00> \rightarrow |00>, \quad |01> \rightarrow |01>$$
$$|10> \rightarrow |11>, \quad |11> \rightarrow |10>$$

$$|\psi_3> =$$
$$A|0000> + A|1100>$$
$$+ B|0010> + B|1111>$$
$$+ C|0100> + C|1000>$$
$$+ D|0110> + D|1010>$$



Step 4　　状態 $|\psi_4>$

Step 4の CNOT は、第一レジスタの measure-X qubitをコントロール qubit とし、第三レジスタの data qubit bをターゲット qubit とする。

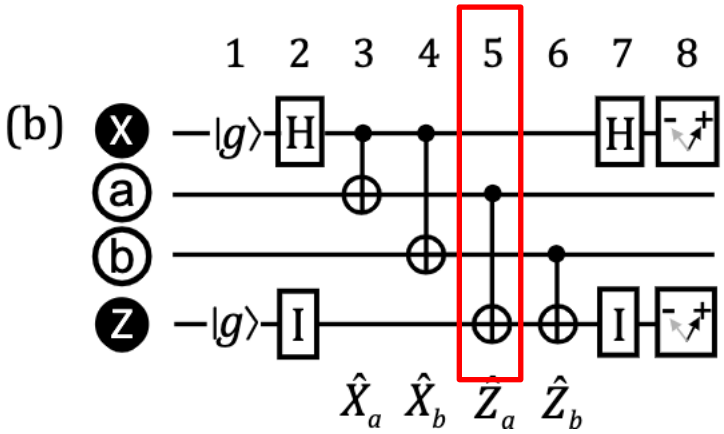
$$|\psi_3> =$$

$$A|0000> + A|1101>$$
$$+ B|0010> + B|1110>$$
$$+ C|0100> + C|1000>$$
$$+ D|0110> + D|1010>$$

$$|00> \rightarrow |00>, \quad |01> \rightarrow |01>$$
$$|10> \rightarrow |11>, \quad |11> \rightarrow |10>$$

$$|\psi_4> =$$

$$A|0000> + A|1110>$$
$$+ B|0010> + B|1101>$$
$$+ C|0100> + C|1010>$$
$$+ D|0110> + D|1001>$$



Step 5

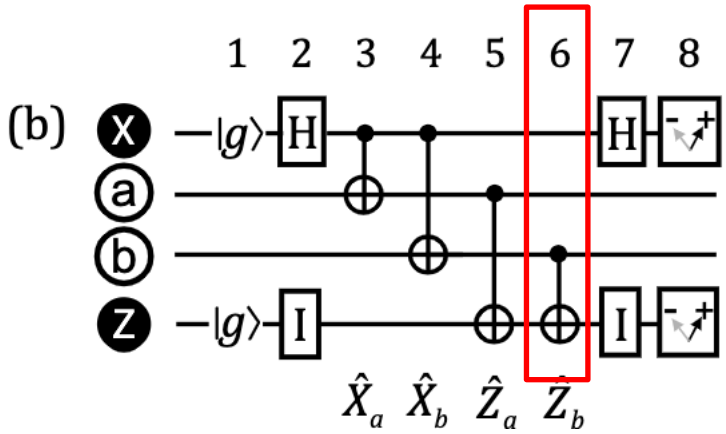
状態 $|\psi_5>$

Step 5の CNOT は、第二レジスタの data qubit a をコントロール qubit とし、第四レジスタの mesure-Z qubit をターゲット qubit とする。

$$|\psi_4> =$$
$$A|0000> + A|1110>$$
$$+ B|0001> + B|1101>$$
$$+ C|0100> + C|1010>$$
$$+ D|0110> + D|1001>$$

$$|00> \rightarrow |00>, \quad |01> \rightarrow |01>$$
$$|10> \rightarrow |11>, \quad |11> \rightarrow |10>$$

$$|\psi_5> =$$
$$A|0000> + A|1111>$$
$$+ B|0010> + B|0010>$$
$$+ C|0101> + C|1010>$$
$$+ D|0111> + D|1000>$$



Step 6

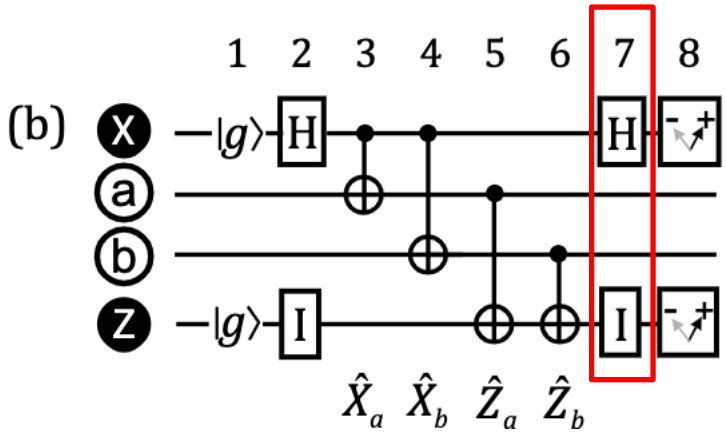
状態 $|\psi_6>$

Step 6の CNOT は、第三レジスタの data qubit b をコントロール qubit とし、第四レジスタの measure-Z qubit をターゲット qubit とする。

$$|\psi_5> =$$
$$A|0000> + A|1111>$$
$$+ B|0010> + B|0010>$$
$$+ C|0101> + C|1010>$$
$$+ D|0111> + D|1000>$$

$$|00> \rightarrow |00>, \quad |01> \rightarrow |01>$$
$$|10> \rightarrow |11>, \; |11> \rightarrow |10>$$

$$|\psi_6> =$$
$$A|0000> + A|1110>$$
$$+ B|0011> + B|1101>$$
$$+ C|0101> + C|1011>$$
$$+ D|0110> + D|1000>$$



Step 7

状態 $|\psi_7>$

measure-X qubitは、アダマールHゲートの作用を受け、次のように変化する。

|0> → |+> = |0> + |1> また、|1> →|−> = |0>−|1>

Iゲートは、measure-Z qubitの状態を変えない。

$$|\psi_6> =$$
$$A|0000> + A|1110>$$
$$+ B|0011> + B|1101>$$
$$+ C|0101> + C|1011>$$
$$+ D|0110> + D|1000>$$

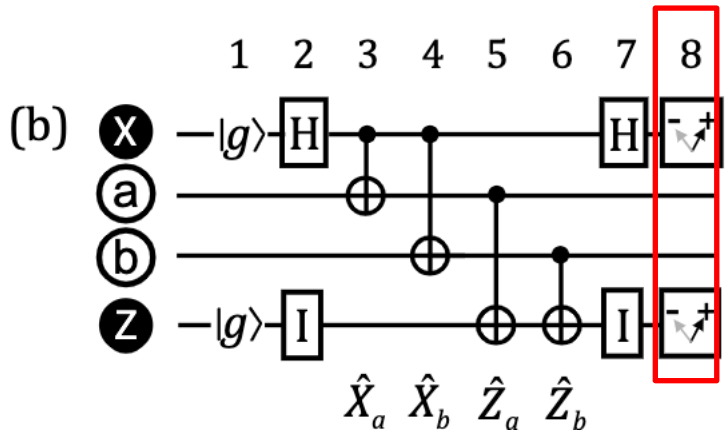
$|+> = |0> + |1>$
$|-> = |0> - |1>$

$H|0> = |+>$
$H|1> = |->$

$$|\psi_7> =$$
$$A|+000> + A|-110>$$
$$+ B|+011> + B|-010>$$
$$+ C|+101> + C|-011>$$
$$+ D|+110> + D|-000>$$

A|0000> +A|1000>
A|0110> −A|1110>
D|0110> +D|1110>
D|0000> −D|1000>

$$= (A + D)|0> \otimes (|00> + |11>) \otimes |0>$$
$$+ (A - D)|1> \otimes (|00> - |11>) \otimes |0>$$
$$+ (B + C)|0> \otimes (|01> + |10>) \otimes |1>$$
$$+ (B - C)|1> \otimes (|01> - |10>) \otimes |1>$$



Step 8

観測

$$|\psi_7> =$$

$$= (A + D)|0> \otimes (|00> + |11>) \otimes |0>$$
$$+ (A - D)|1> \otimes (|00> - |11>) \otimes |0>$$
$$+ (B + C)|0> \otimes (|01> + |10>) \otimes |1>$$
$$+ (B - C)|1> \otimes (|01> - |10>) \otimes |1>$$

第一レジスタ　　第二・第三レジスタ　　第四レジスタ

この状態は、エンタングルしたBell State である！

$X|+> = |+>$、$X|-> = -|->$、
$Z|0> = |0>$、$Z|1> = -|1>$ から、
X, Zは固有値 $+1$と-1を持つことがわかる。

第一レジスタ measure−X qubitの|0>,|1>の観測M_xも
第四レジスタ measure−Z qubitの|0>,|1>の観測M_zも、
固有値 $+1, -1$ を観測値として得る。

この時、システムの固有状態 $|\psi_{ab}\rangle$は、次のようになる。

$\{M_X, M_Z\} =$ $|\psi_{ab}\rangle =$

$\{+1, +1\}$; $|00> + |11>$

$\{-1, +1\}$; $|00> - |11>$

$\{+1, -1\}$; $|01> + |10>$

$\{-1, -1\}$; $|01> - |10>$

Bell State

Phase Flip

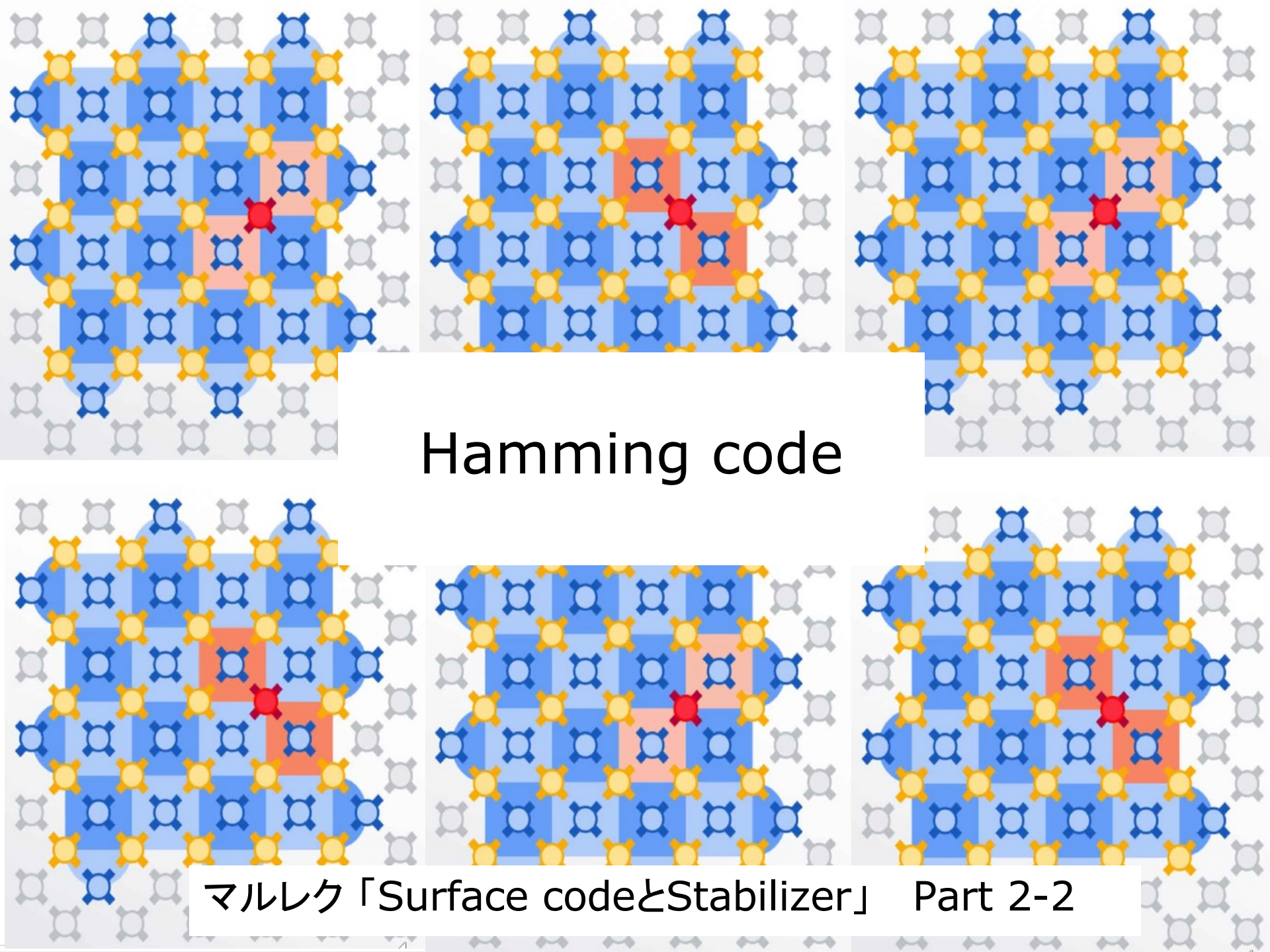
Bit Flip


Part 3
古典コードから量子コードへ
Hamming code と Stabilizer

Surface code と Stabilizer **Agenda**

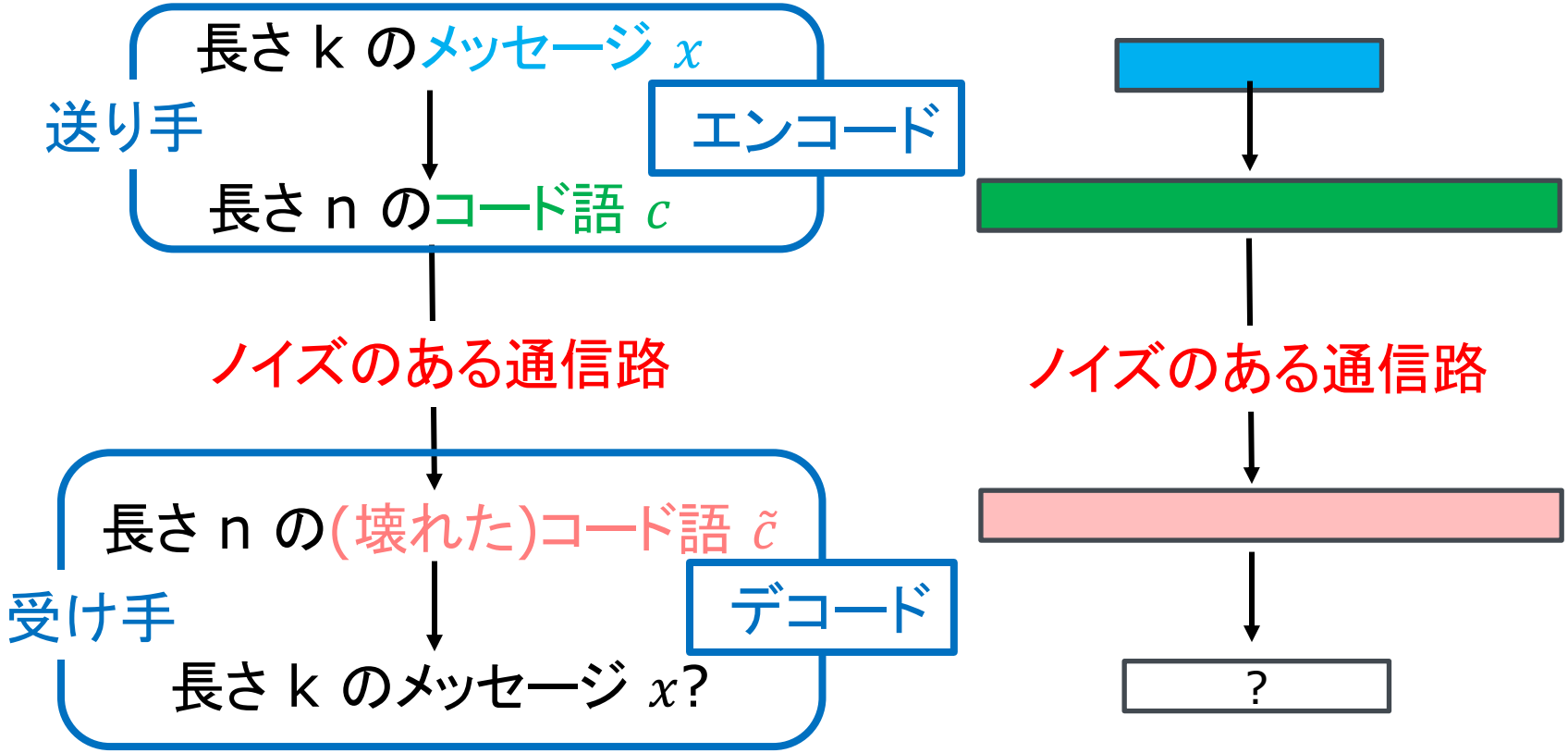
Part 3　古典コードから量子コードへ

- Hamming code
- 古典コードから量子コードへ
- Stabilizer Formalism



マルレク「Surface codeとStabilizer」　Part 2-2

コード理論(古典的)の基本問題



(壊れた)コード語 $\tilde{c}$ から
元のメッセージ x を復元できるか？

定義

- アルファベットΣ 上の長さ n (「ブロック長」ともいう) の「コード」C とは、$C \subseteq \Sigma^n$のことを言う。(CはΣ^n の部分集合である)
- コードCの要素c $(c \in C)$を「コード語」 codewordと呼ぶ。

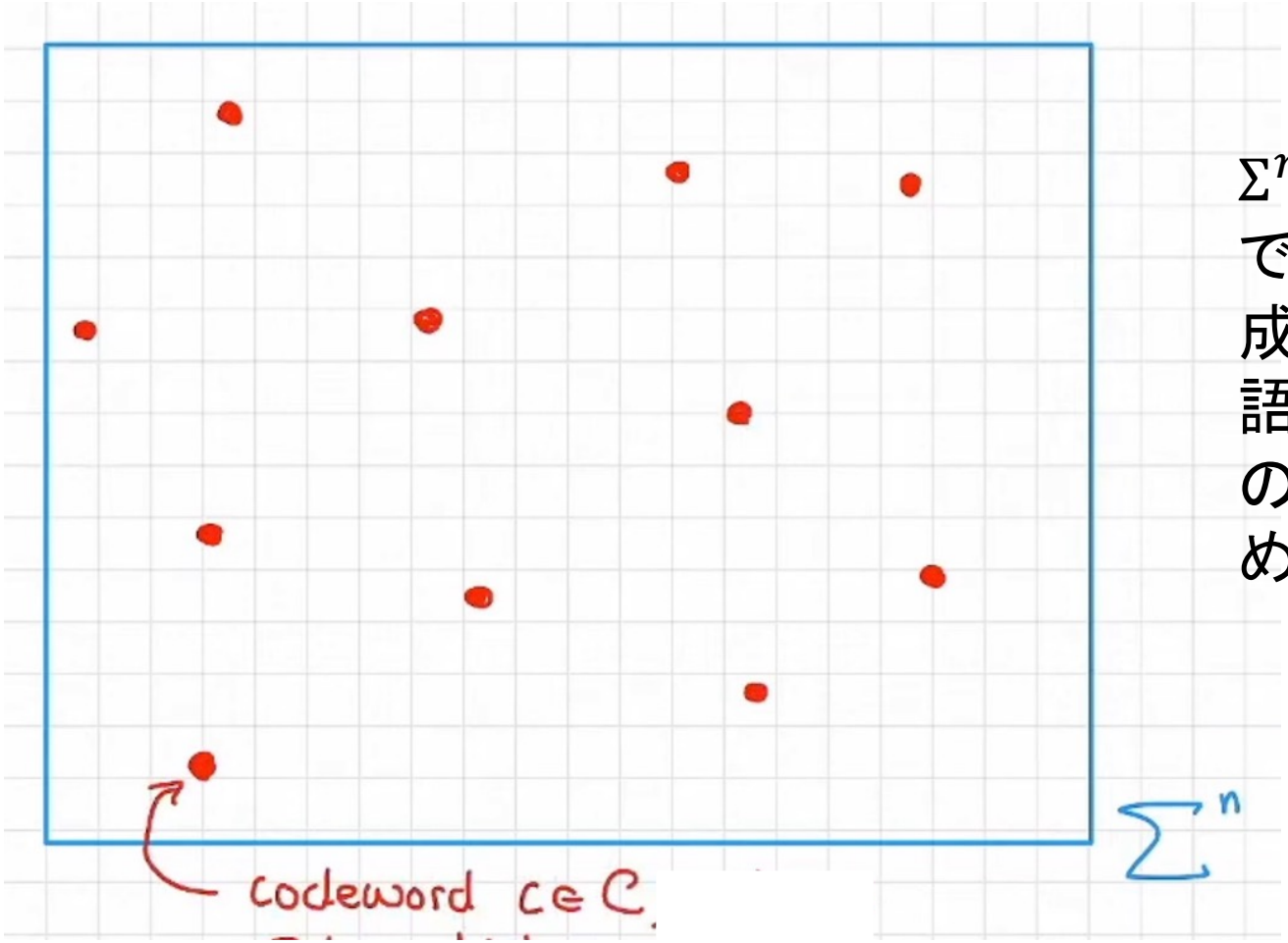
- アルファベットΣ 上の長さ k のメッセージ xを、アルファベットΣ 上の長さ n のコード語 c にエンコードする関数を、ENC と表す。

$$ENC: \Sigma^k \to \Sigma^n$$
$$x \mapsto c$$

である。

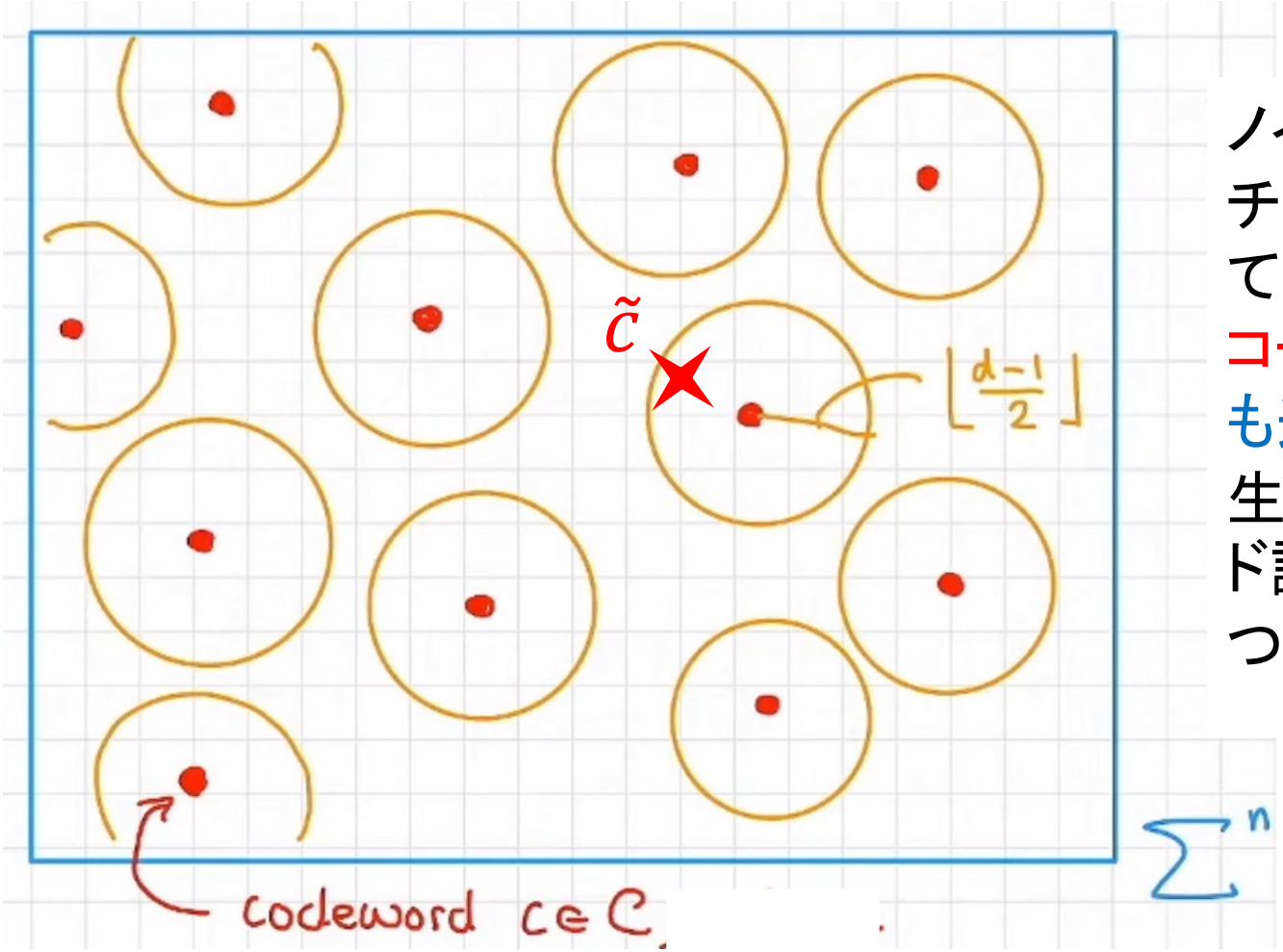
- ENCの値域Cは、$|\Sigma^k|$個の要素からなり、コード語の要素は$|\Sigma^n|$個の要素からなる。$k < n$の時、$C \subset \Sigma^n$である。

$C \subset \Sigma^n$ のイメージ



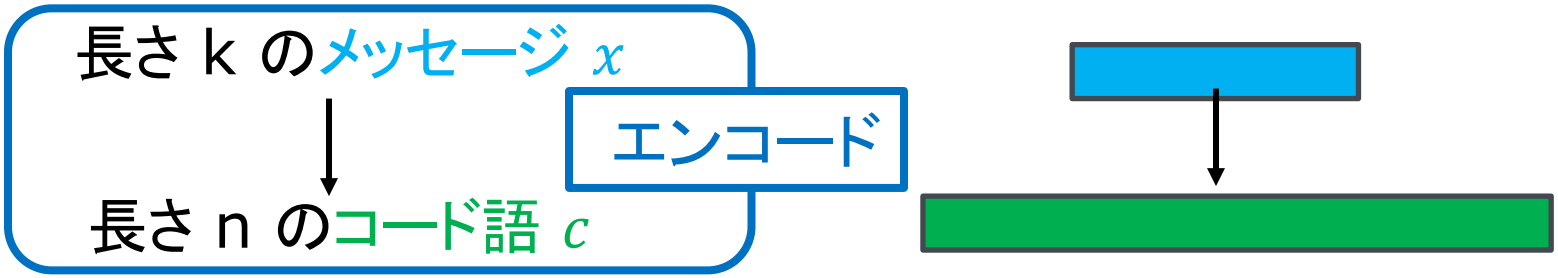
Σ^n の空間の中では、ENCで生成されるコード語 $c \in C$ は、そのごく一部を占めるに過ぎない。

(壊れた)コード語 $\tilde{c}$ から
元のメッセージ x を復元するということ

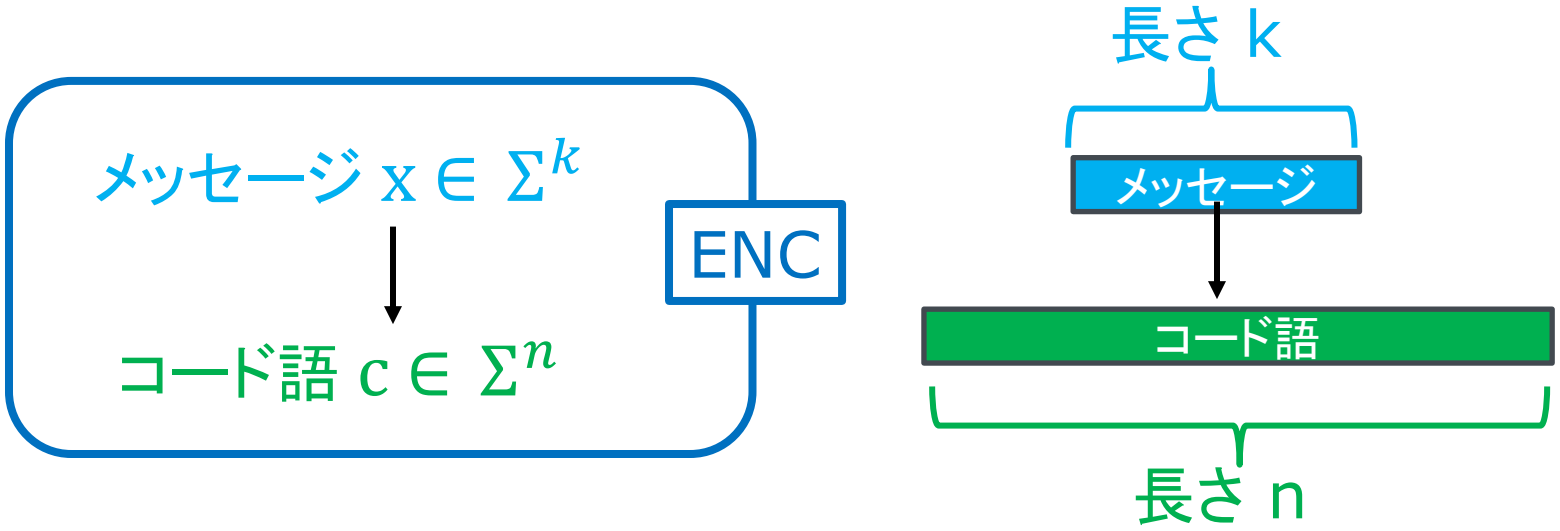


ノイズのあるチャンネルから受け取った、コード語 $\tilde{c}$ に最も近い、ENCで生成されたコード語 $c \in C$ を見つければいい。

エンコーダ ENCの働き



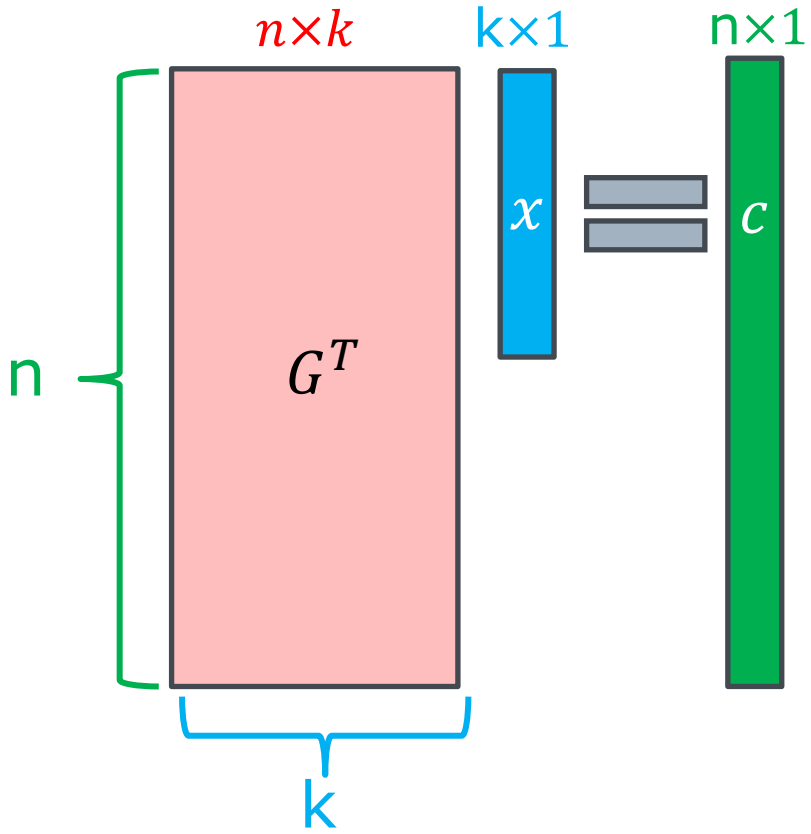
$$ENC: \Sigma^k \to \Sigma^n$$
$$x \mapsto c$$



コードの生成行列

ENCの入力のメッセージxも、その出力のコード語 cも、それぞれk次元とn次元のベクトルと考えることができる。

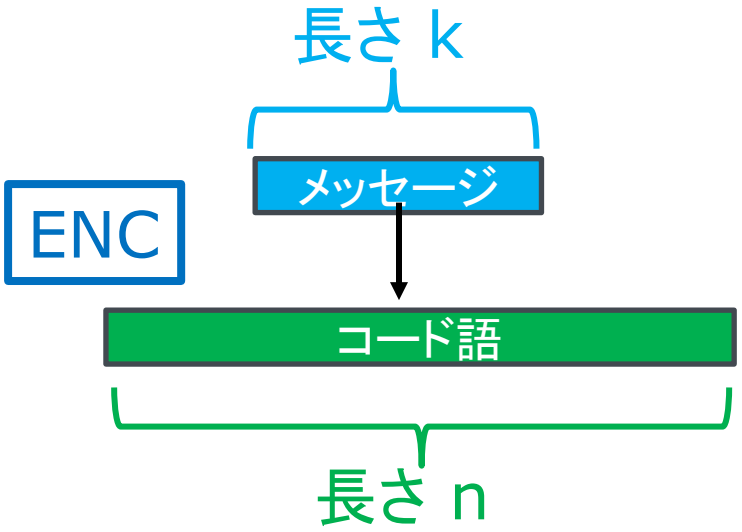
エンコーダENCは、k次元のベクトルをn次元のベクトルに変換する行列G^Tとして表現される。Gをコードの生成行列という。



$G^T x = c$

あるいは

$x^T G = c^T$



エンコーダ ENCの例 1
パリティ・ビットの付加

$ENC: \{0,1\}^3 \rightarrow \{0,1\}^4$ を、次のように定義する。

$$ENC(x_1, x_2, x_3) = (x_1, x_2, x_3, x_1 + x_2 + x_3 \ mod\ 2)$$

例えば、

$$ENC(0,1,1) = (0,1,1,0+1+1) = (0,1,1,2) = (0,1,1,0)$$

この時、

ENC(0,0,0)=(0,0,0,0)
ENC(0,0,1)=(0,0,1,1)
ENC(0,1,0)=(0,1,0,1)
ENC(0,1,1)=(0,1,1,0)
ENC(1,0,0)=(1,0,0,1)
ENC(1,0,1)=(1,0,1,0)
ENC(1,1,0)=(1,1,0,0)
ENC(1,1,1)=(1,1,1,1)

これを C とする

ENC(0,0,0)=(0,0,0,0)
ENC(0,0,1)=(0,0,1,1)
ENC(0,1,0)=(0,1,0,1)
ENC(0,1,1)=(0,1,1,0)
ENC(1,0,0)=(1,0,0,1)
ENC(1,0,1)=(1,0,1,0)
ENC(1,1,0)=(1,1,0,0)
ENC(1,1,1)=(1,1,1,1)

$ENC: \{0,1\}^3 \rightarrow \{0,1\}^4$なら。
$C = Im(ENC)$ となる。

受け取ったコード語 $\tilde{c}$ が、ENC の値域 C に含まれていなければ、それは、何か異常が起きたことを示す。

- 例えば、1-bitが消失して、$\tilde{c} = (0,?,0,1)$を受け取ったら、消失した1-bitは何だと考えることができるか？

ENC(0,1,0)=(0,1,0,1)

- 例えば、 $\tilde{c} = (0,0,0,1)$を受け取ったら、何が言えるか？

ENC(0,0,0)=(0,0,0,0)	(0,0,0,1)
ENC(0,0,1)=(0,0,1,1)	(0,0,0,1)
ENC(0,1,0)=(0,1,0,1)	(0,0,0,1)
ENC(0,1,1)=(0,1,1,0)	(0,1,1,0)
ENC(1,0,0)=(1,0,0,1)	(0,0,0,1)
ENC(1,0,1)=(1,0,1,0)	(1,0,1,0)
ENC(1,1,0)=(1,1,0,0)	(1,1,0,0)
ENC(1,1,1)=(1,1,1,1)	(1,1,1,1)

エンコーダ ENCの例 2
(7, 4) Hamming code

$ENC: \{0,1\}^4 \to \{0,1\}^7$ を、次のように定義する。

$$ENC(r_1, r_2, r_3, r_4) = (\, r_1, r_2, r_3, r_4, r_5, r_6, r_7 \,)$$

ただし、r_5, r_6, r_7 は、次のパリティである。

$$r_5 = r_1 + r_2 + r_3$$
$$r_6 = r_2 + r_3 + r_4$$
$$r_7 = r_1 + r_3 + r_4$$

計算例

$$ENC(0,1,0,1) = (0,1,0,1,0+1+0,1+0+1,0+0+1)$$
$$= (0,1,0,1,1,0,1)$$

$Im(ENC)$

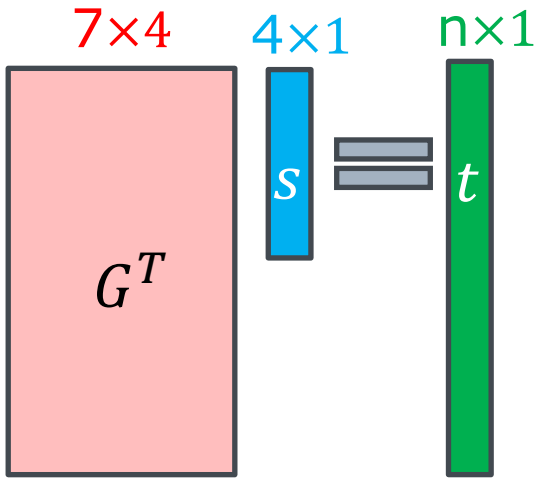
次の表は、4-bit のソースの s から、このエンコーダで生成される 7-bitのターゲット t を示す。 $Im(ENC)$ である。

$$ENC(r_1, r_2, r_3, r_4) = (\, r_1, r_2, r_3, r_4, r_5, r_6, r_7 \,)$$
$$ENC(s) = t$$

s	t	s	t	s	t	s	t
0000	0000000	0100	0100110	1000	1000101	1100	1100011
0001	0001011	0101	0101101	1001	1001110	1101	1101000
0010	0010111	0110	0110001	1010	1010010	1110	1110100
0011	0011100	0111	0111010	1011	1011001	1111	1111111

このコード語 t の集合Cの任意のペアは、少なくとも 3-bit異なっている。

先の表で、$t = G^T s$ とすると、G^T は次のようになる。



Hamming code の生成行列 G^T

$$\mathbf{G}^{\mathsf{T}} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \end{bmatrix} \begin{matrix} r_1 \\ r_2 \\ r_3 \\ r_4 \\ \\ \\ \\ \end{matrix}$$

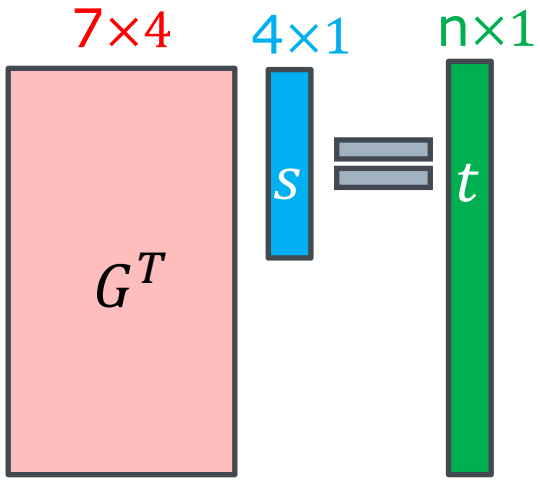
$r_5 = r_1 + r_2 + r_3$

$r_6 = r_2 + r_3 + r_4$

$r_7 = r_1 + r_3 + r_4$

Hamming codeの全てのコード語の集合Cは、生成行列G^Tによって生成される

先の表で、$t = G^T s$ とすると、G^T は次のようになる。



Hamming code
の生成行列 G^T

I_4

P

$$\mathbf{G}^{\mathsf{T}} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \end{bmatrix} \begin{matrix} r_1 \\ r_2 \\ r_3 \\ r_4 \\ \\ \\ \\ \end{matrix}$$

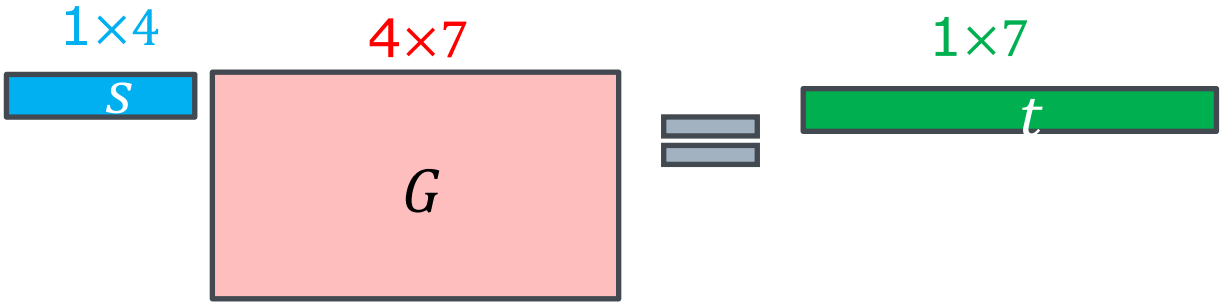
$r_5 = r_1 + r_2 + r_3$

$r_6 = r_2 + r_3 + r_4$

$r_7 = r_1 + r_3 + r_4$

$$G^T = \begin{bmatrix} I_4 \\ P` \end{bmatrix}$$

s,t を行ベクトルとすれば、$t = sG$ となって 、Gは次のようになる。



Hamming code の生成行列 G

$$\mathbf{G} = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 \end{bmatrix}.$$

I_4 P^T

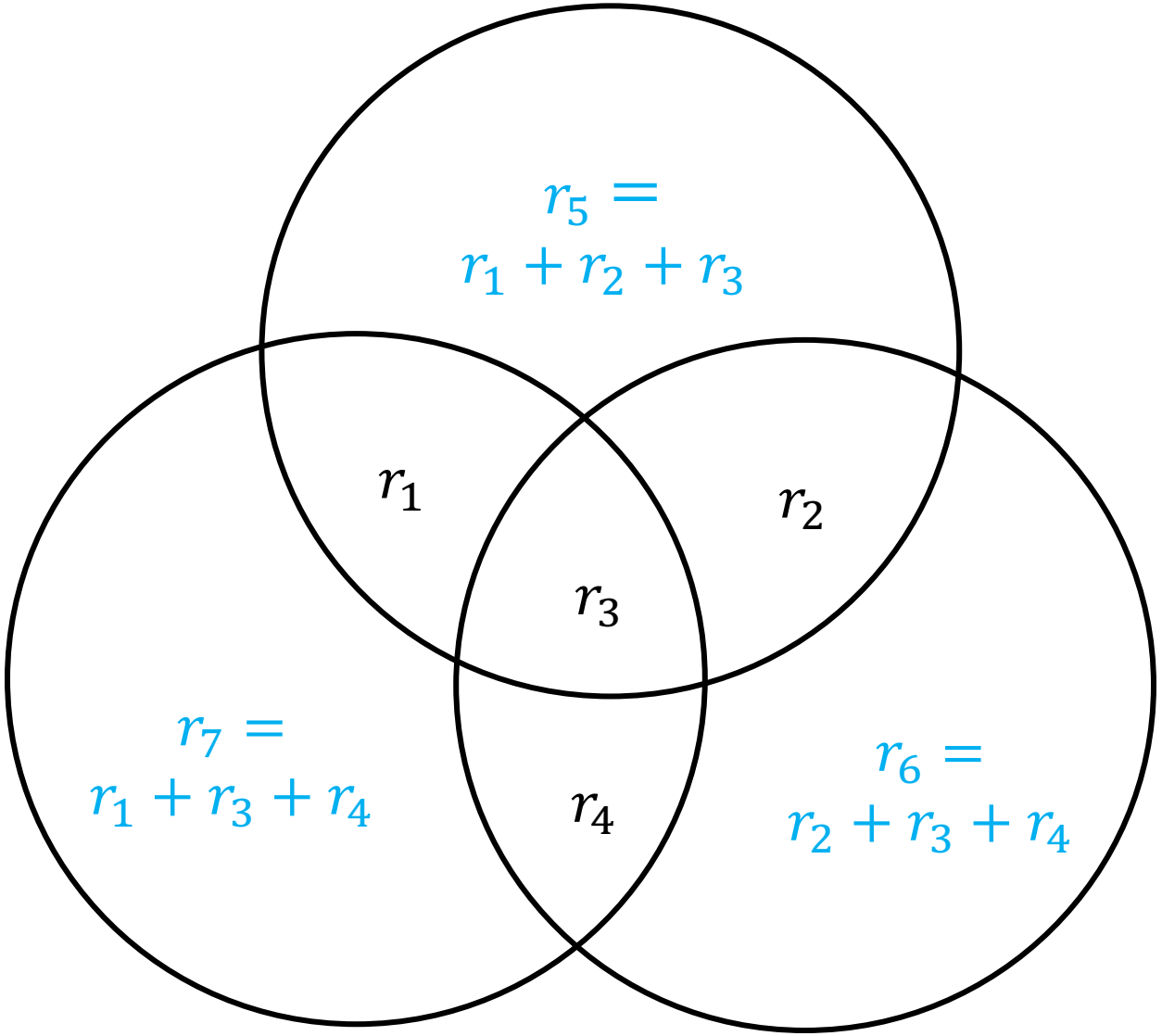
$$G = [I_4 \quad P^T]$$

$r_5 = r_1 + r_2 + r_3$

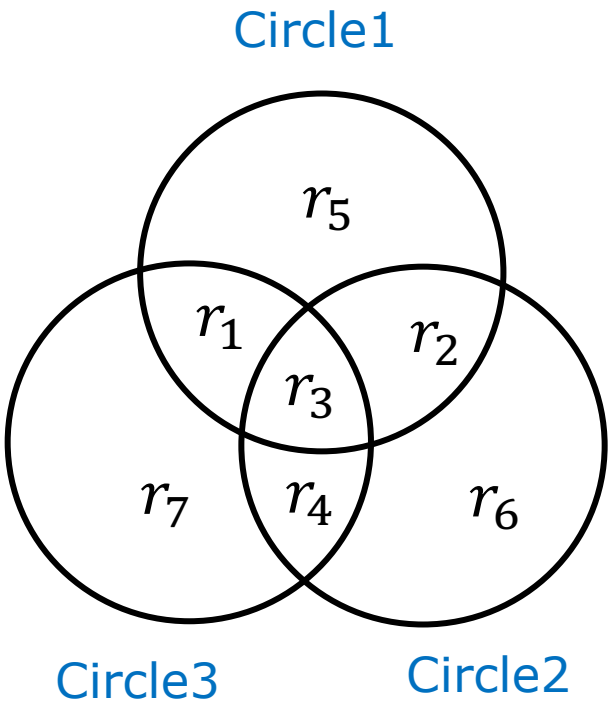
$r_1 \quad r_2 \quad r_3 \quad r_4$ $\qquad r_6 = r_2 + r_3 + r_4$

$r_7 = r_1 + r_3 + r_4$

r_1, r_2, r_3, r_4とr_5, r_6, r_7 の関係は、次のように表現すると分かりやすい。また、パリティのチェックもしやすい。



$\tilde{c}$ のパリティ・チェック

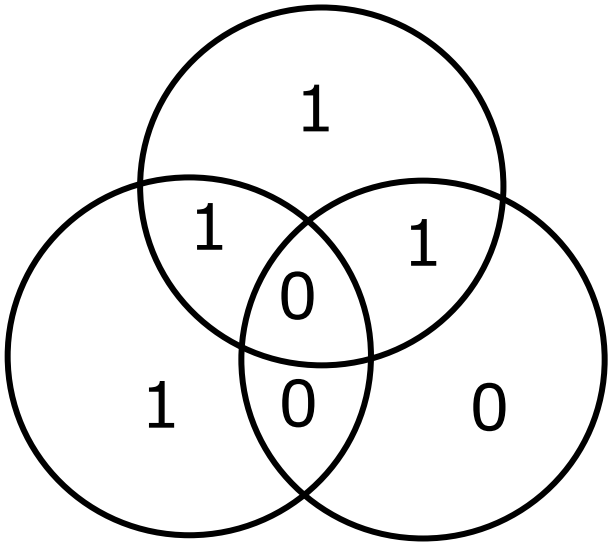


受け取った

$$\tilde{c} = (r_1, r_2, r_3, r_4, r_5, r_6, r_7)$$

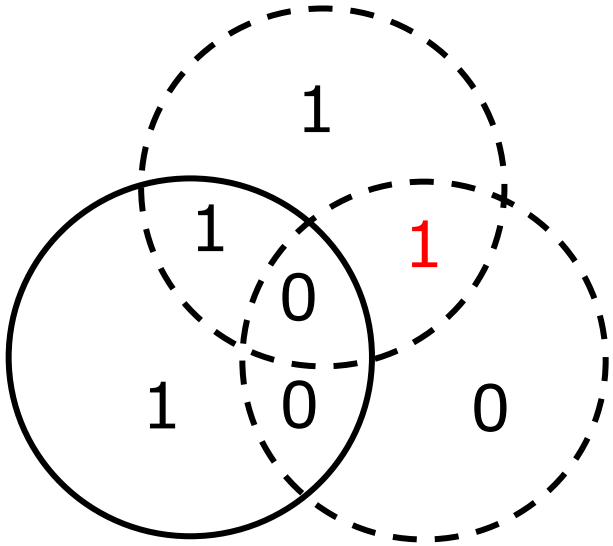
を、この図形に入れて、サークルごとにパリティに矛盾がないかをチェックする。

矛盾があったサークルは、波線で表すことにしよう。

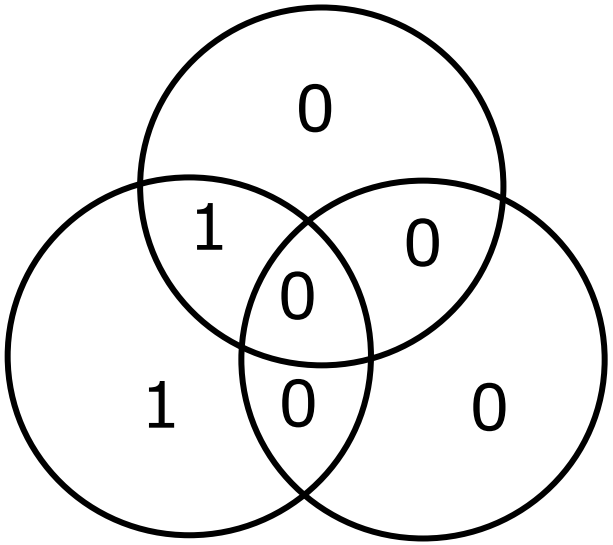


$\tilde{c} = (1,1,0,0,1,0,1)$の場合は、次のようになる。

パリティがあっていない円を波線で表す。

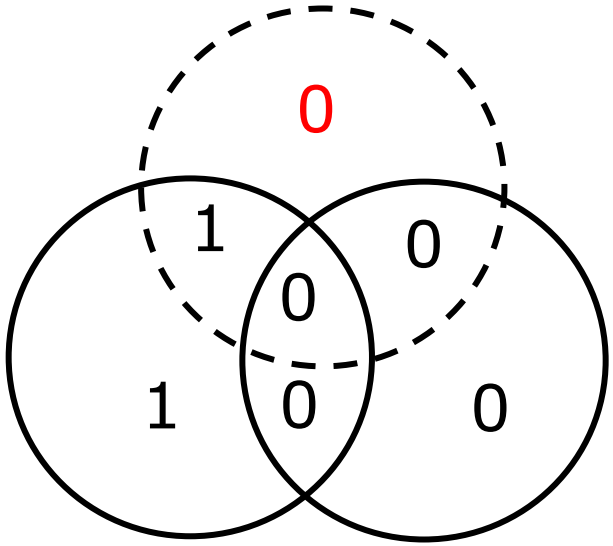


赤字のビットが反転していることがわかる。

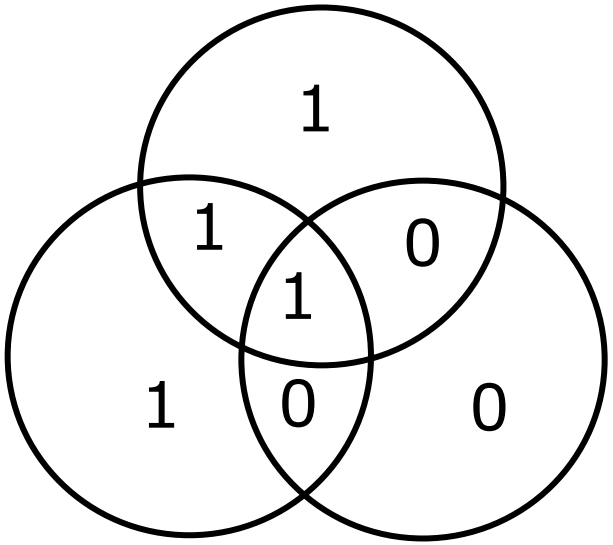


$\tilde{c} = (1,0,0,0,0,0,1)$の場合は、次のようになる。

パリティがあっていない円を波線で表す。

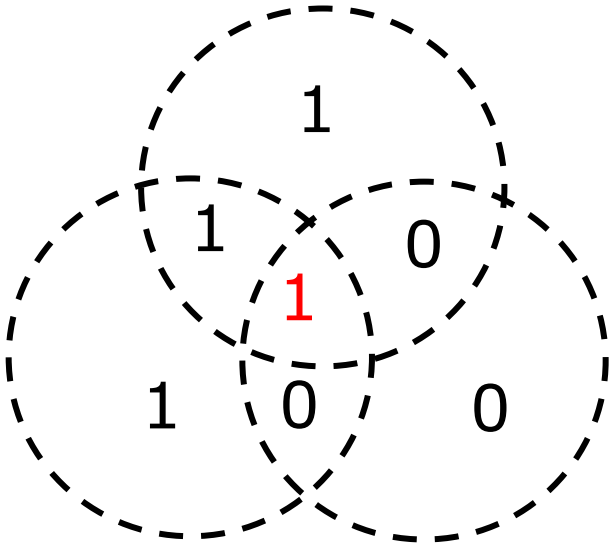


赤字のビットが反転していることがわかる。



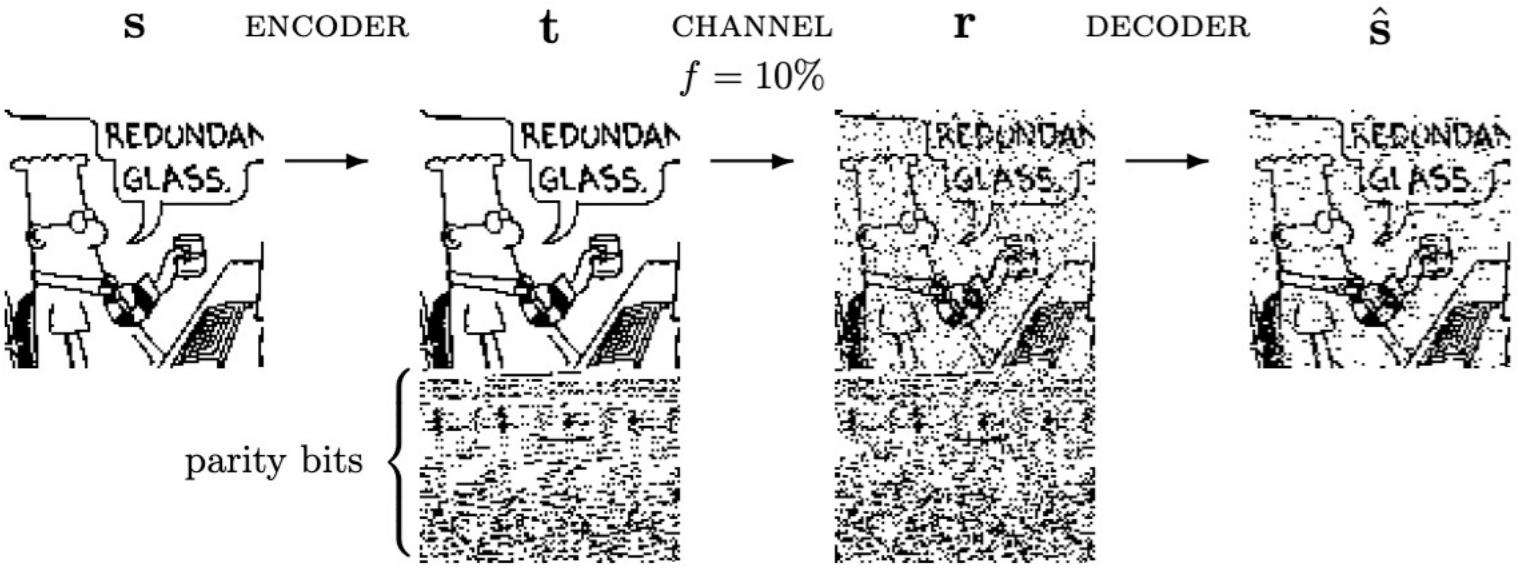
$\tilde{c} = (1,0,1,0,1,0,1)$の場合は、次のようになる。

パリティがあっていない円を波線で表す。

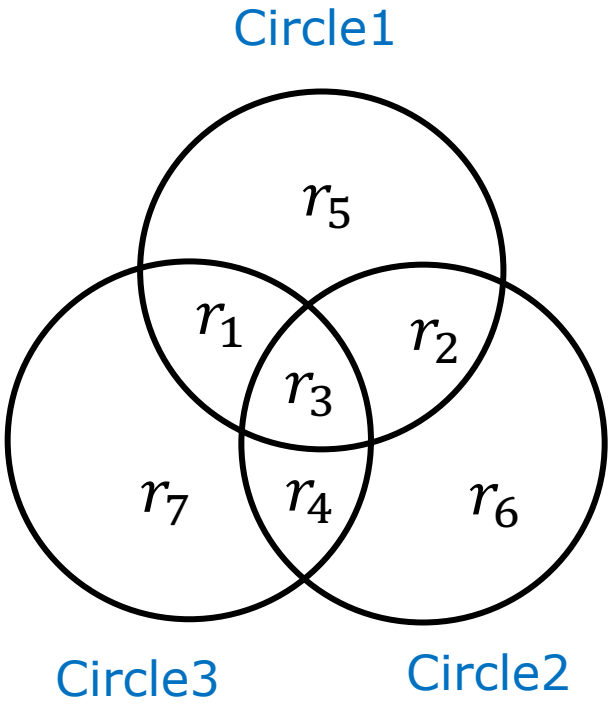


赤字のビットが反転していることがわかる。

Parity bit の利用



$Syndrome$ とbit flip



Circle1, Circle2, Circle3 でパリティが破れている時、その値を1とし、破れていない時、0とする。

z=(Circle1,Circle2,Circle3)を、Syndromeという。

この時、反転が起きたbitは、次の表で与えられる。

Syndrome **z**	000	001	010	011	100	101	110	111
Unflip this bit	*none*	r_7	r_6	r_4	r_5	r_1	r_2	r_3

パリティ・チェク行列 H

Hamming codeが生成する全てのコード語の集合Cの任意の要素yについて、

$$\mathrm{Hy}=0$$

を満たす行列Hをパリティ・チェク行列 と呼ぶ。

コード語の集合Cは、パリティ・チェク行列 HのKernelとして定義される。

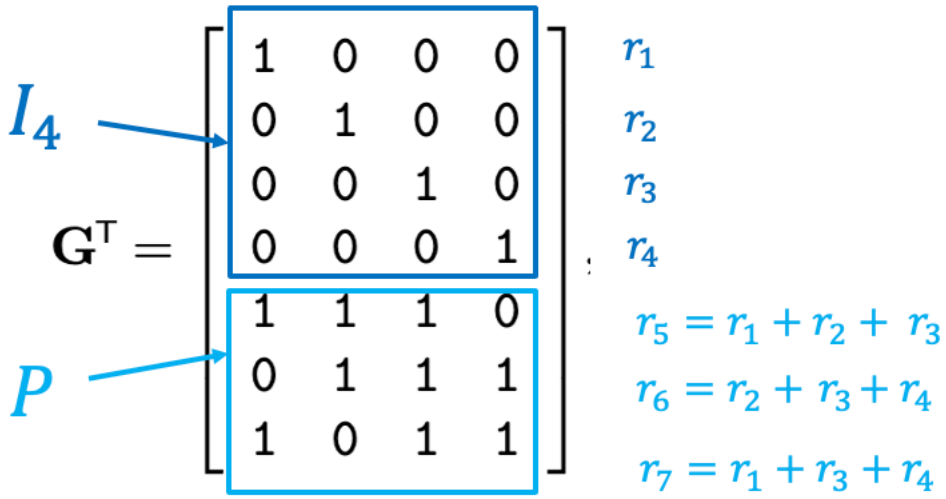
$$C = \{c \in \{0,1\}^7 : Hc = 0\} = Ker(H)$$

先に見たように、Cはまた、生成行列 G のImageとしても定義される。

$$C = \{Gx : x \in \{0,1\}^4\} = Im(G)$$

パリティ・チェク行列 Hと生成行列 G

コード語の集合C の生成行列 G は、次のように $G^T = \begin{bmatrix} I_4 \\ P` \end{bmatrix}$ の形で表現される。



$$G^T = \begin{bmatrix} I_4 \\ P` \end{bmatrix}$$

この時、$H = [\, P \quad I_3 \,]$ は、パリティ・チェック行列となることを示そう。

$H = [\ P \quad I_3\ ]$ と表現されること

$$\mathbf{H} = [\ \mathbf{P} \quad \mathbf{I_3}\ ] = \left[\begin{array}{cccc|ccc} 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 & 1 \end{array}\right]$$

P $\qquad$ I_3

Hがこの形の時、yをコード語とすると、Hy=0となることは、次のようにしてわかる。

$$y = G^T x = \begin{bmatrix} I_4 \\ P \end{bmatrix} x = \begin{bmatrix} x \\ Px \end{bmatrix} \text{ である。}$$

$$Hy = [\ P \quad I_3\ ] \begin{bmatrix} x \\ Px \end{bmatrix} = Px + Px = 2Px = 0 \ \ (mod\ 2)$$

Hはパリティ・チェック行列である。

パリティ･チェク行列 H の性質

チャンネルから得られたコード語 r は、チャンネルに送られた $t = G^T s$ とノイズのベクトル n の和である。

$$r = G^T s + n$$
$$Hr = HG^T s + Hn = 0 + Hn = Hn$$

このベクトル $z = Hr = Hn$ を r のSyndromeと呼ぶ。

Syndromeの計算例

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

r = (1,1,0,0,1,0,1)の場合

Hr^T は、Hの三つの行ベクトルと r の内積から構成される3-次元ベクトルである。その内積は、二つのベクトルが共有する1 の数に等しい。

$$Hr^T = \begin{bmatrix} 3 \\ 1 \\ 2 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix} (mod\ 2)$$

Syndrome **z**	000	001	010	011	100	101	110	111
Unflip this bit	*none*	r_7	r_6	r_4	r_5	r_1	r_2	r_3

Syndromeの計算例

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

r = (1,0,0,0,0,0,1)の場合

Hr^T は、Hの三つの行ベクトルと rの内積から構成される3-次元ベクトルである。その内積は、二つのベクトルが共有する1 の数に等しい。

$$Hr^T = \begin{bmatrix} 1 \\ 0 \\ 2 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \ (mod\ 2)$$

Syndrome **z**	000	001	010	011	100	101	110	111
Unflip this bit	*none*	r_7	r_6	r_4	r_5	r_1	r_2	r_3

Syndromeの計算例

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

r = (1,0,1,0,1,0,1)の場合

Hr^T は、Hの三つの行ベクトルと r の内積から構成される3-次元ベクトルである。その内積は、二つのベクトルが共有する1 の数に等しい。

$$Hr^T = \begin{bmatrix} 3 \\ 1 \\ 2 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix} \ (mod\ 2)$$

Syndrome **z**	000	001	010	011	100	101	110	111
Unflip this bit	*none*	r_7	r_6	r_4	r_5	r_1	r_2	r_3

参考資料

誤り訂正符号の初歩ー古典と量子, 小又志郎

https://www.marulabo.net/docs/komata/

How to send a self-correcting message, 3Blue1Brown

https://www.youtube.com/watch?v=X8jsijhllIA

Algebraic coding theory, Mary Wootters

https://youtube.com/watch?v=vfjN7MmSB6g&list=PLkvhuSoxwjI_UudECvFYArvG0cLbFlzSr

参考資料

All you need to know about classical error correction

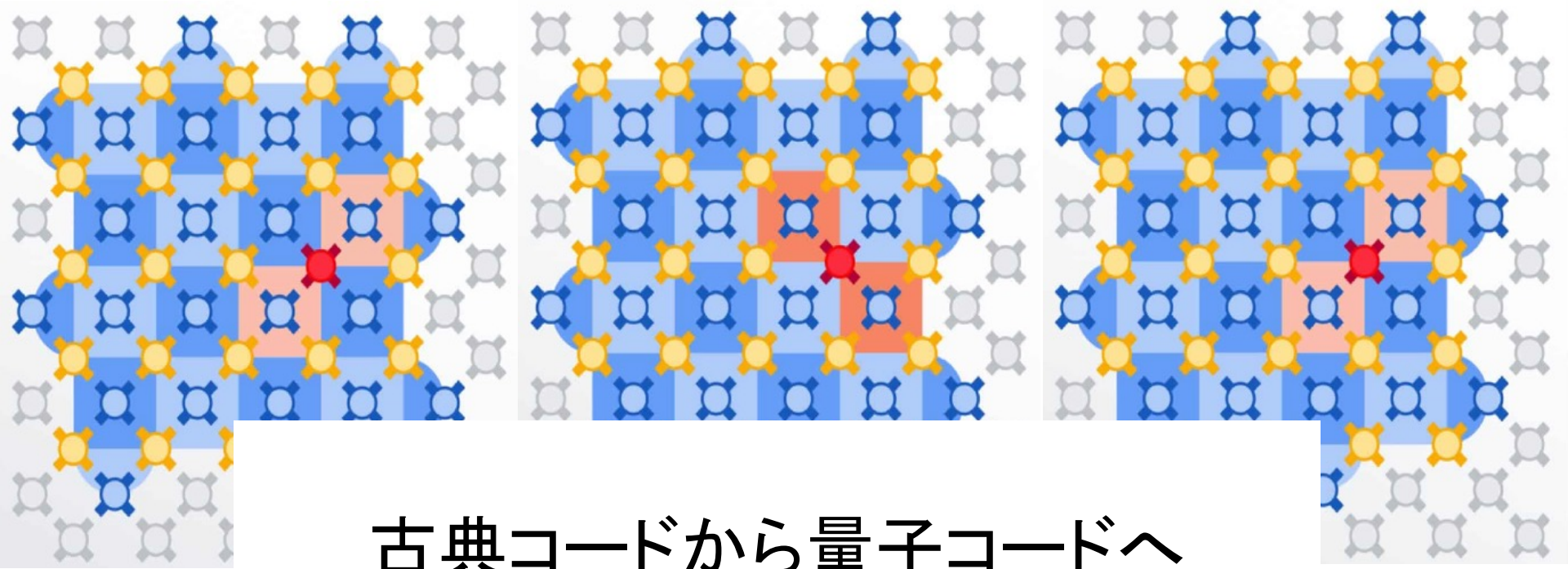
Arthur Pesah

https://arthurpesah.me/blog/2022-05-21-classical-error-correction/

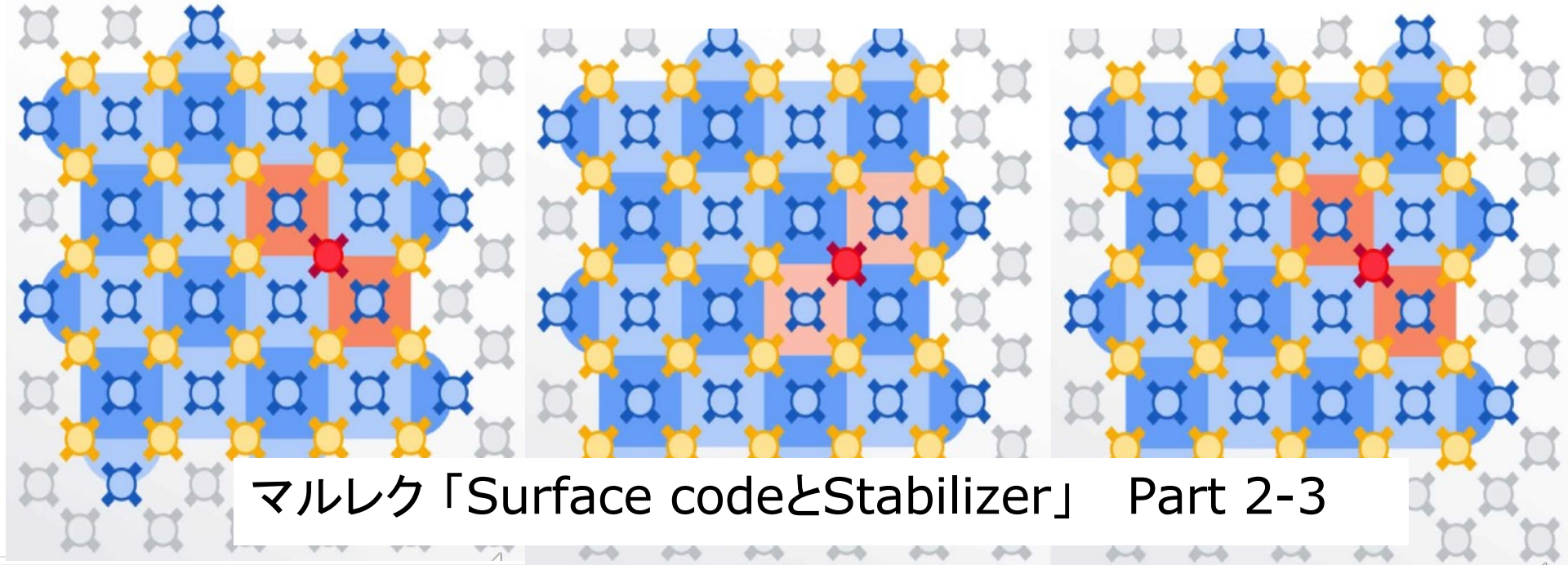
Information Theory, Inference, and Learning Algorithms,

David McKay

https://www.inference.org.uk/itprnn/book.pdf



古典コードから量子コードへ



マルレク「Surface codeとStabilizer」　Part 2-3

古典コードから量子コードへ

このセッションでは、Shannonの情報理論と共に1950年代に登場した古典的なエラー訂正技術が、現代の量子エラー訂正技術に、どのように進化・発展してきたのかを振り返りたいと思います。

19世紀の初め、ラプラスは「もしもある瞬間における全ての物質の力学的状態と力を知ることができれば、不確実なことは何もなくなり、未来も（過去同様に）全て見えているであろう。」と語りました。

100年後の20世紀初頭、量子論と相対論の登場の中で、私たちの世界観は大きく変わりました。

量子論では、物理法則は決定論的にではなく確率論的に定義されることになります。確実な未来の予測は量子の世界では不可能です。

相対論は、我々の認識の「空間と時間」という枠組みが、絶対的なものではなく、情報を伝える光の速度が一定（それは有限です）という条件のもとで、様々に変化することを明らかにしました。

ラプラスは、未来を（決定論的に）予見する能力を「知性」と呼んだのですが、我々人間の「知性」にせよ、AI等の機械の「知性」にせよ、我々は、それらの「知性」の特性を考える必要があるのです。

誰もが知っていて、誰もが知らないこと

現代の常識が半ばそうであるように、相対論は巨視的なマクロな世界を対象とし、量子論は極微のミクロな世界を対象とするというように理論が妥当する領域を二つに分け、現実の運動は、いずれかの理論の「近似」として折衷的に理解するにとしても、基本的な問題は解決されるわけではありません。

エラー訂正技術をテーマとするセミナーで、誰もが問題の所在は知っていて、かつ誰もその答えをまだ知らない問題を繰り返したのには、理由があります。

それは、古典論と量子論の関係を考える上で、古典論的なエラー訂正技術から量子論的エラー訂正技術への発展は、とても示唆的に思えるからです。

両者は、ある意味よく似ている

今回のセッションでは、前回見たHamming codeと、Shor code以降に発見された代表的な量子コードであるSteane codeの類似性にフォーカスしています。

そうした類似性の認識は、以前はCSS constructionとして意識され、現代では、次のセッションで見る、Stabilizer Formalism として定式化されるものです。

量子論の特徴である「不確定性」は、その演算子が一般には交換関係を満たさないことから導かれます。ただ、古典論にもポアソン括弧があるように、量子論を記述する演算子全てが非可換であるわけではありません。

似ているけど違うもの

Stabilizer formalism というのは、量子論の中で可換な演算子の作用する世界を取り出そうという試みに似ています。そうした世界が古典論の世界に似てくるのは、当然かもしれません。

ただ、こうした構成が導くものは、単なる古典論の繰り返しではありません。全く新しい異なる世界が広がります。

古典ビットが量子ビットに変わり、code-wordがcode-spaceに変わるだけでなく、その空間の「状態」そのものが「演算子」として捉え返され、そうした演算子のシークエンスとして、「パリティ・チェック」が再定義されます。それがStabilizerです。

Hamming codeとSteane codeの類似性

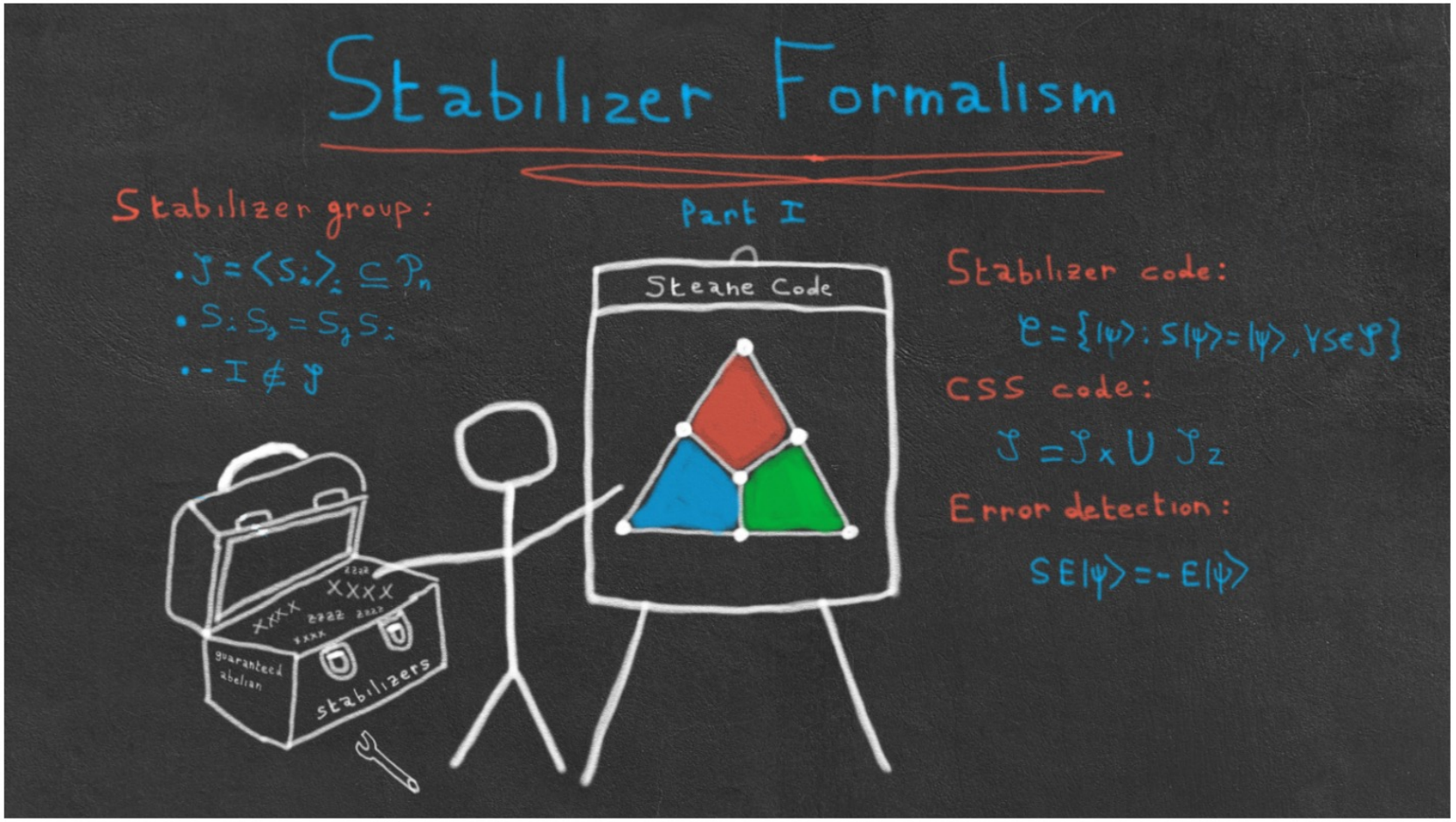
今回は、繰り返しになりますが、Hamming codeとSteane codeの類似性にフォーカスしています。その類似は、理論的というよりイメージ的なものです。

ただ、こうした類似・対応のイメージは、Stabilizer formalismの理解に、役立つものだと思います。

以下の説明では、Steane codeの模式図に、Arthur Pesahのアイコンを借用しています。

The stabilizer trilogy I — Stabilizer codes

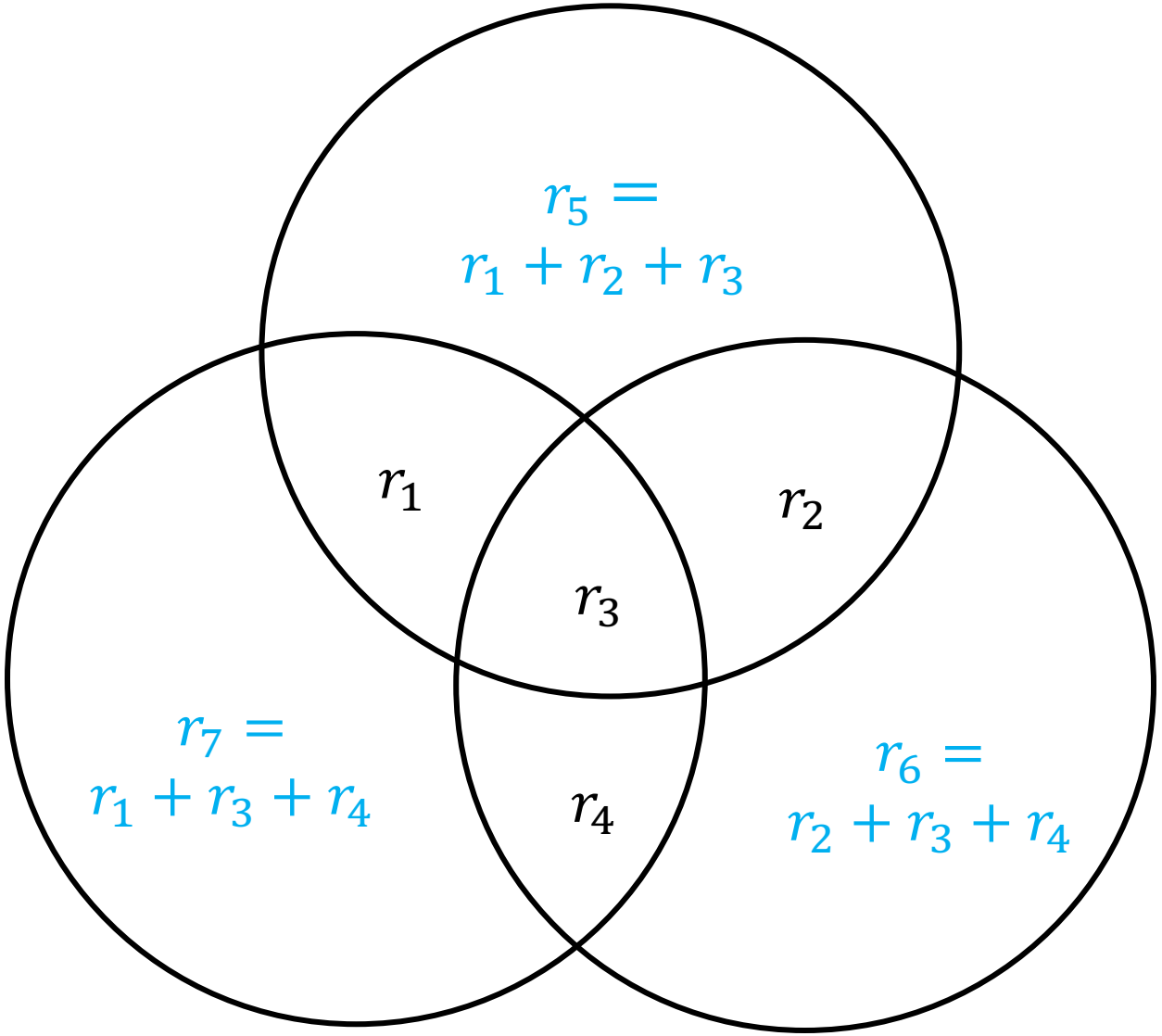
31 Jan 2023 on Quantum Computing



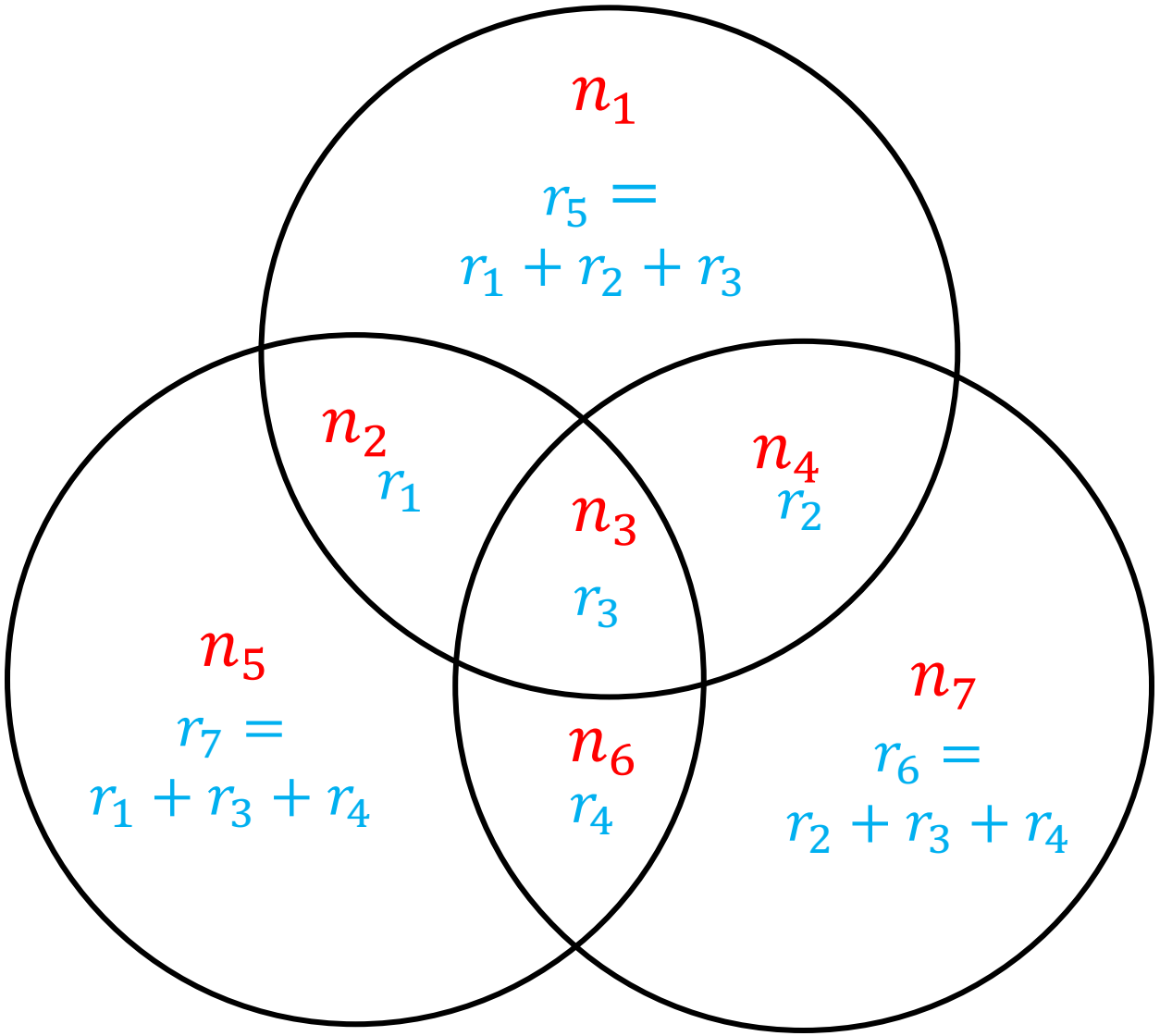
https://arthurpesah.me/blog/2023-01-31-stabilizer-formalism-1/

Hamming codeの模式図

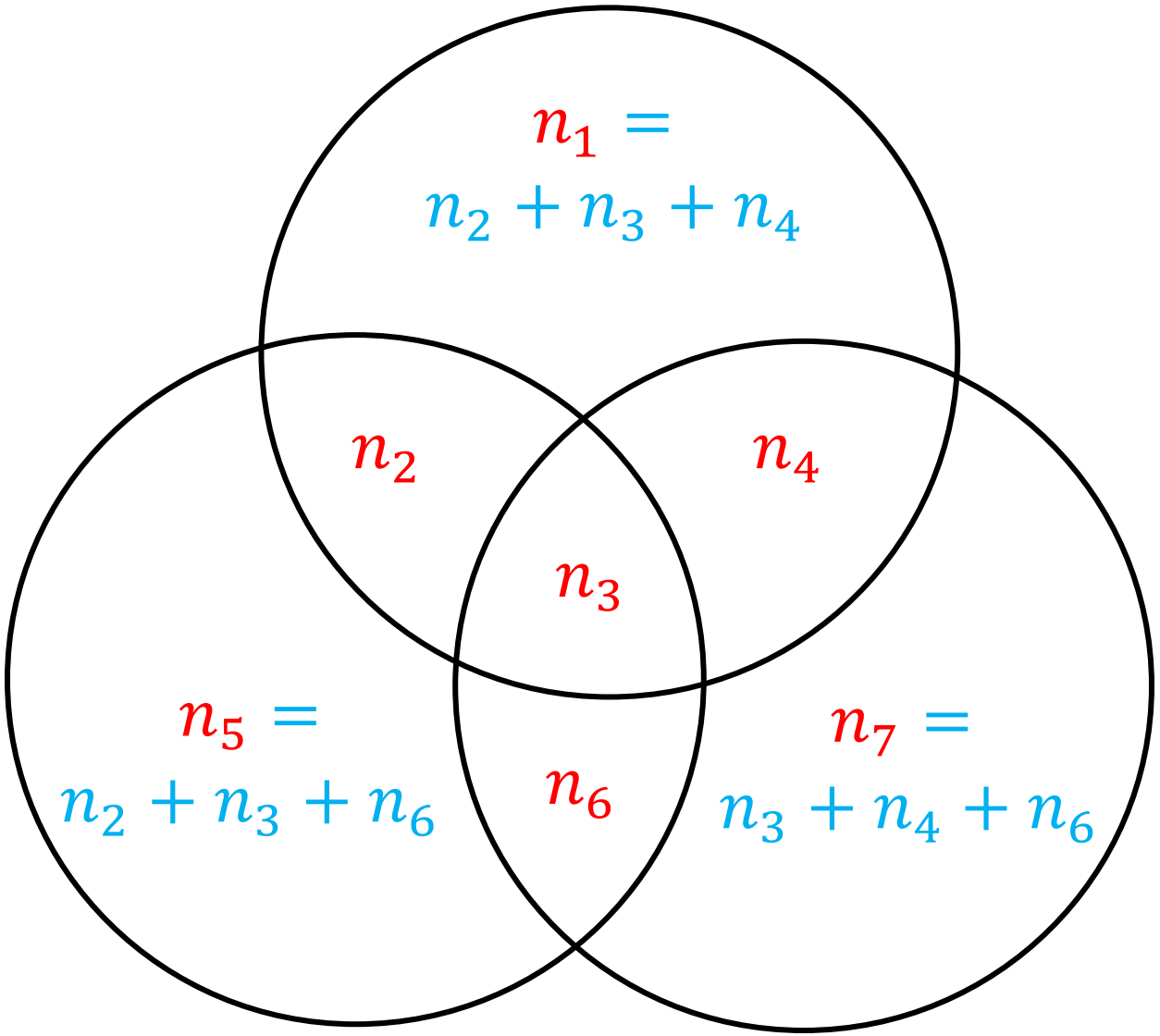
$data\ bit\ \ r_1, r_2, r_3, r_4$とパリティ･ビット r_5, r_6, r_7



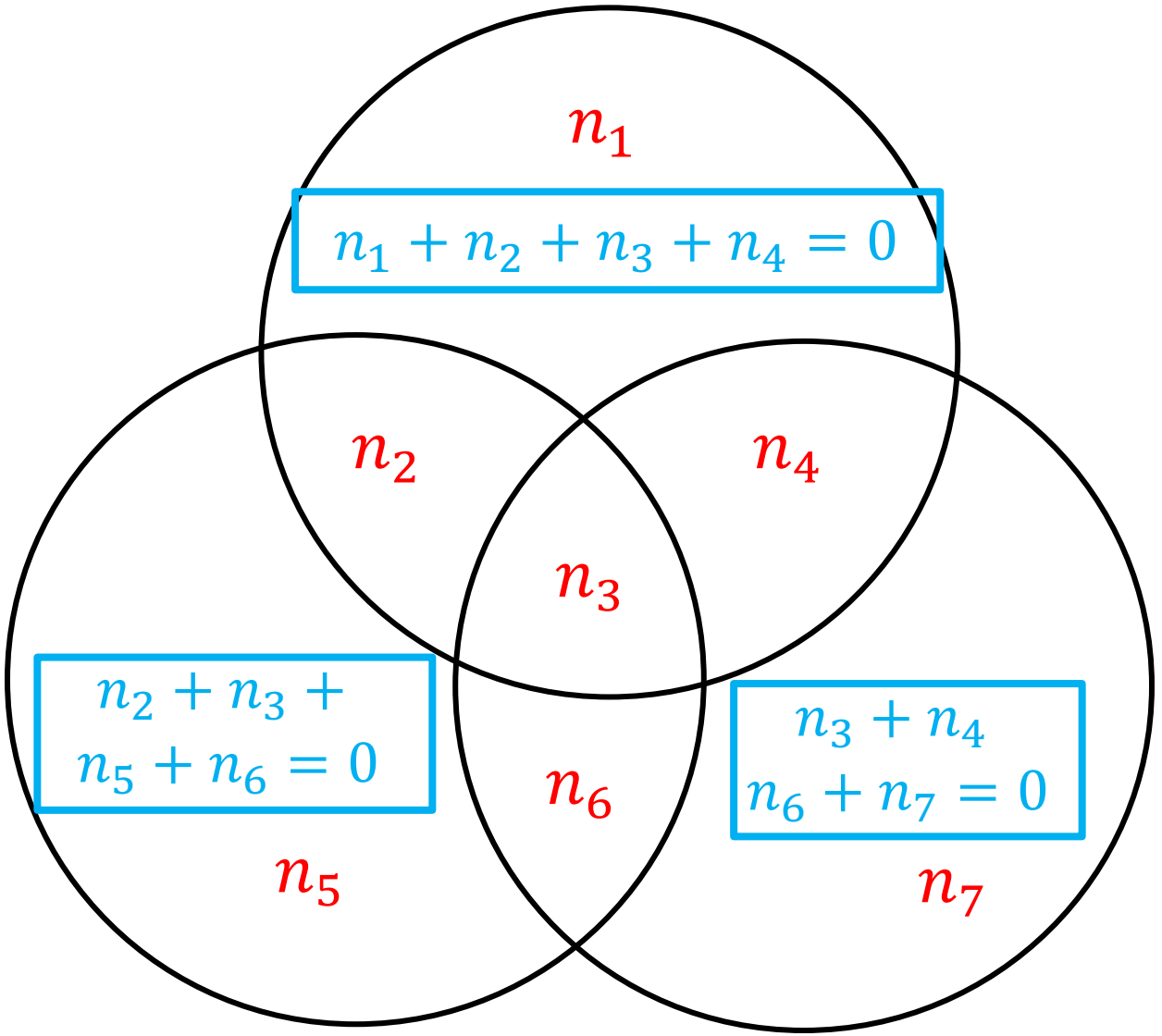
今回は、次のように名前を一般的なものに変える。
Steane codeでは、7のphysicalなqubitが、
一つのlogical なqubitを定義する。



qubit n_2, n_3, n_4, n_6に対して、qubit n_1, n_5, n_7 は、そのparity 情報を持っている

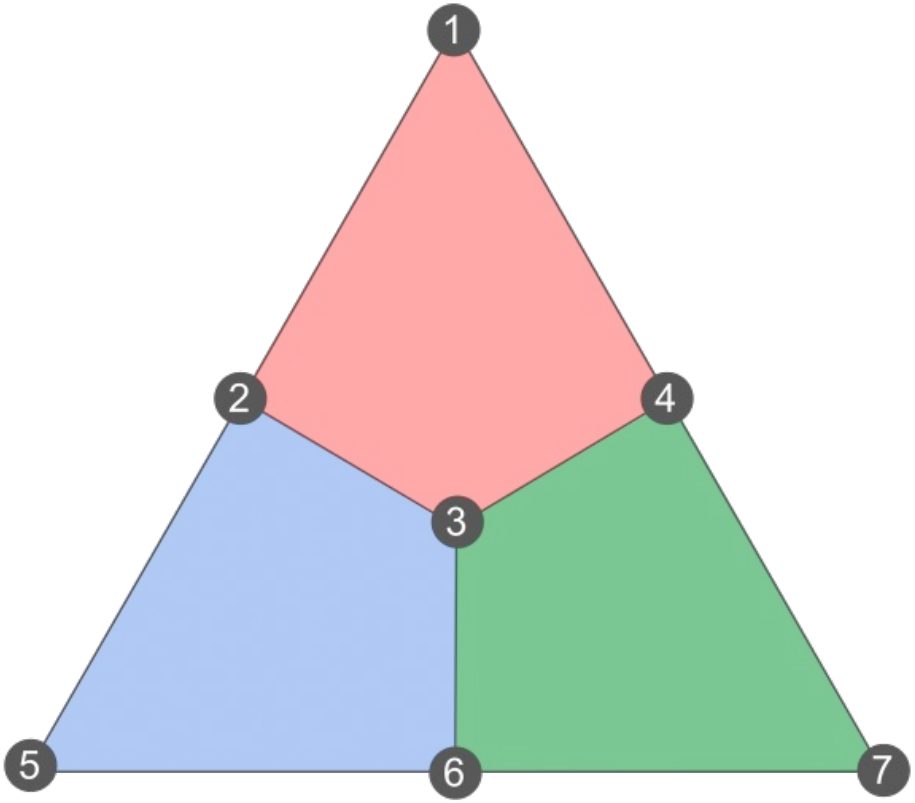


mod 2 では、$n_i = -n_i$であるので、$n_1 = n_2 + n_3 + n_4$ 等の条件は、次のように書ける

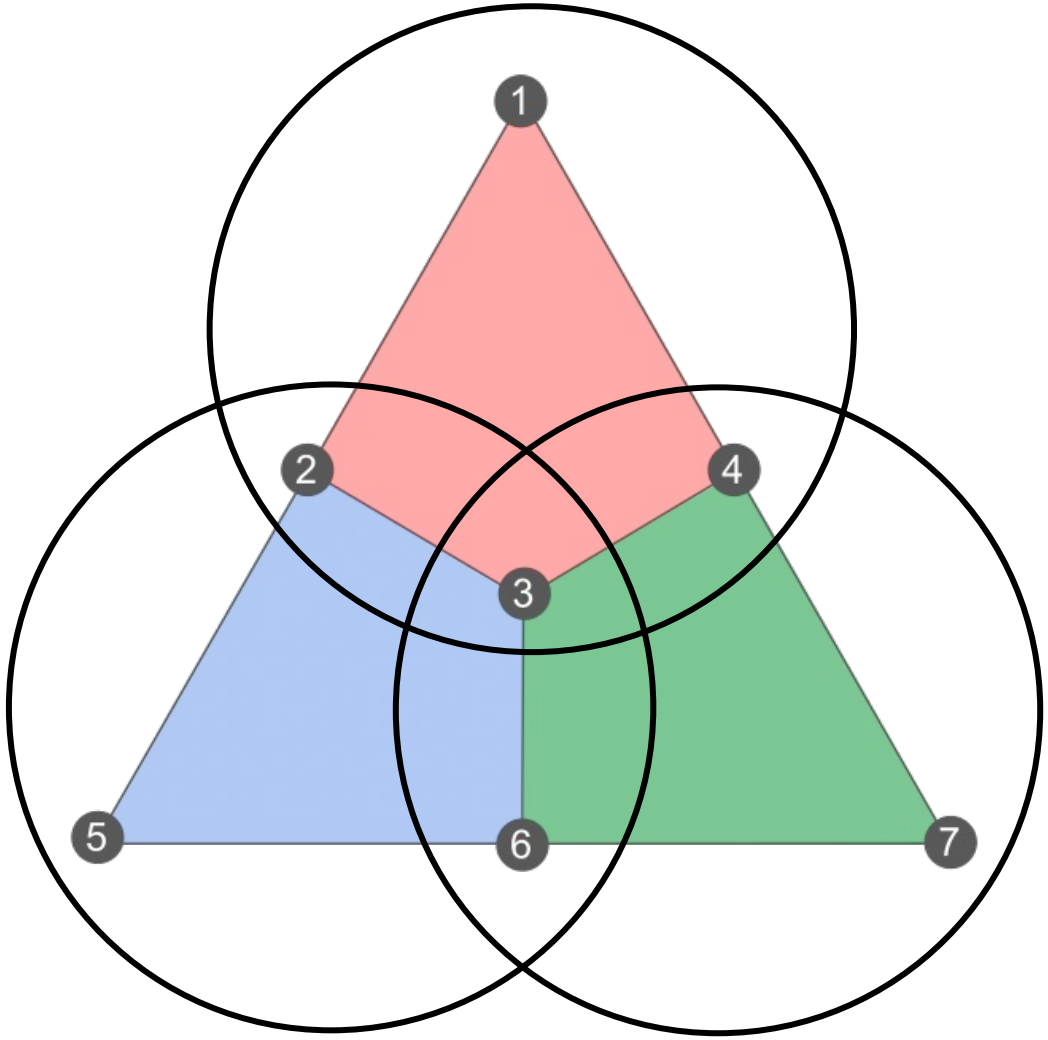


Steane codeの模式図

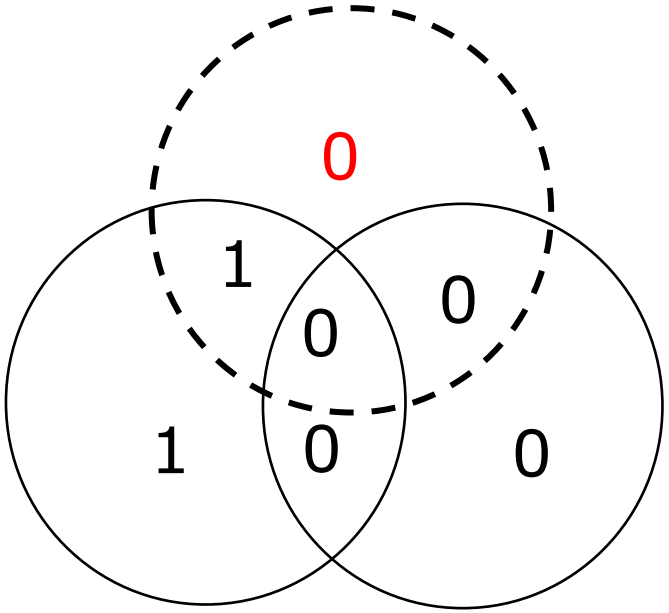
この7つのqubitで1つの論理qubitを表す



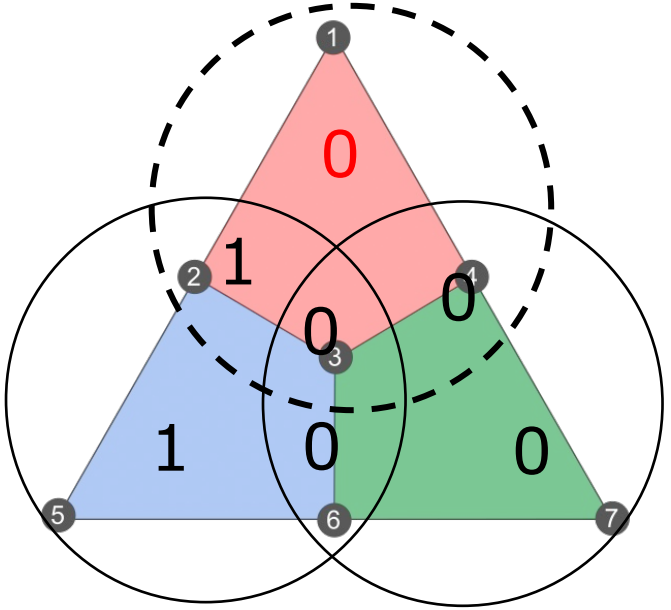
Hamming code とSteane codeの模式図を次のように重ね合わせることができる



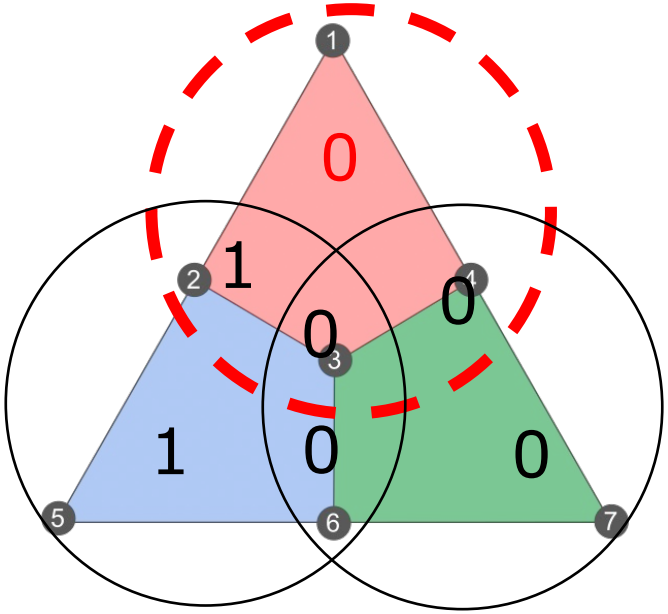
Hamming code



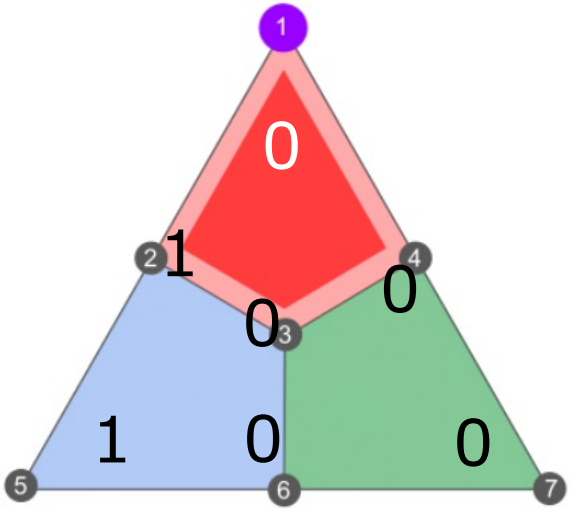
Hamming code
+
Steane code



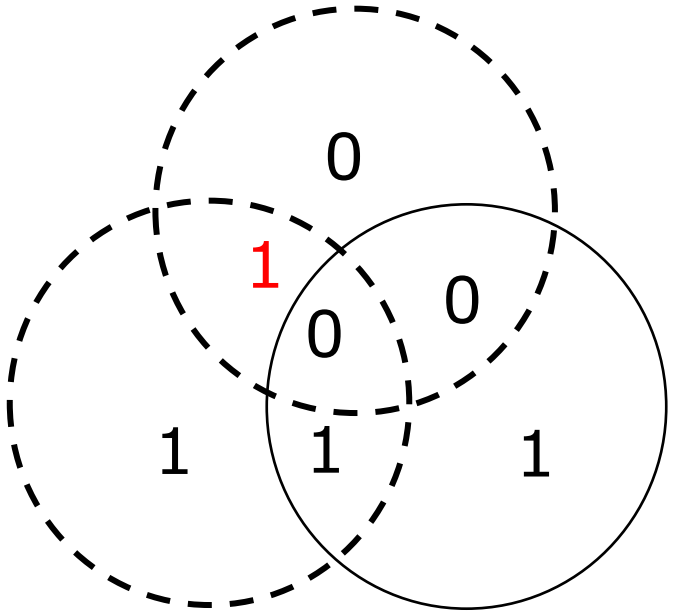
Hamming code
+
Steane code



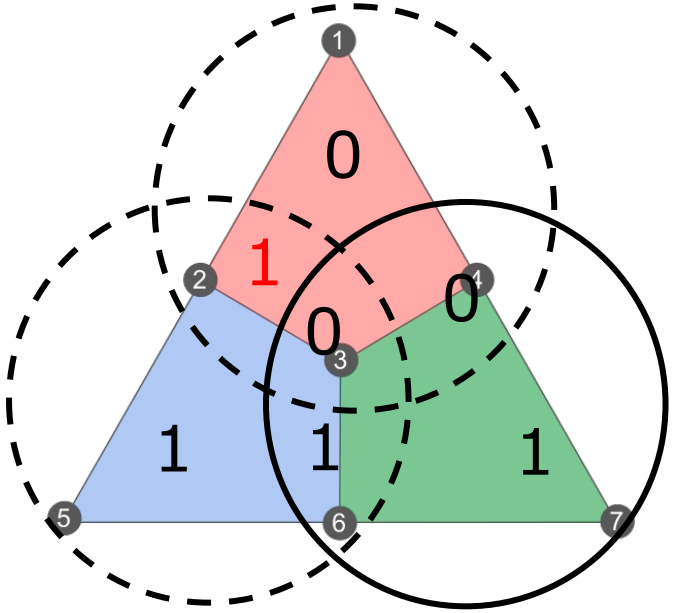
Steane code



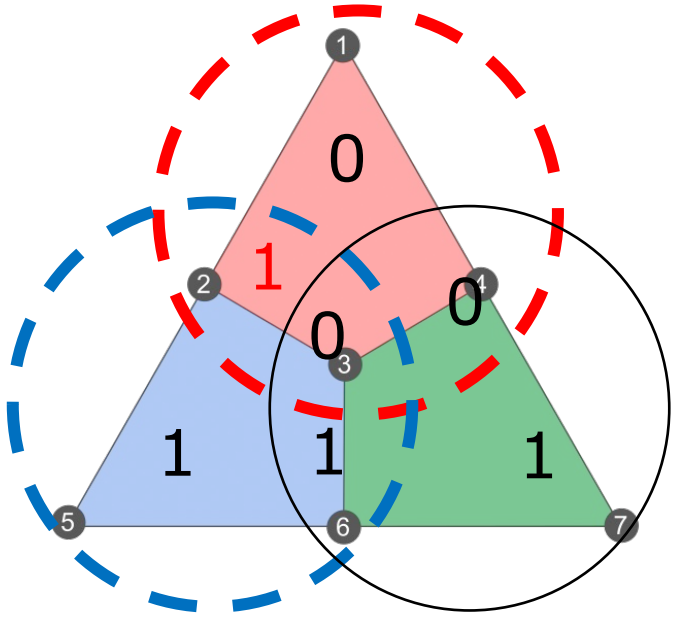
Hamming code



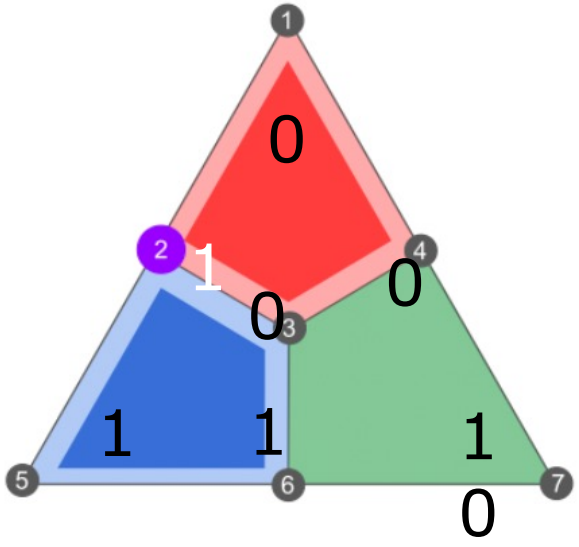
Hamming code
+
Steane code



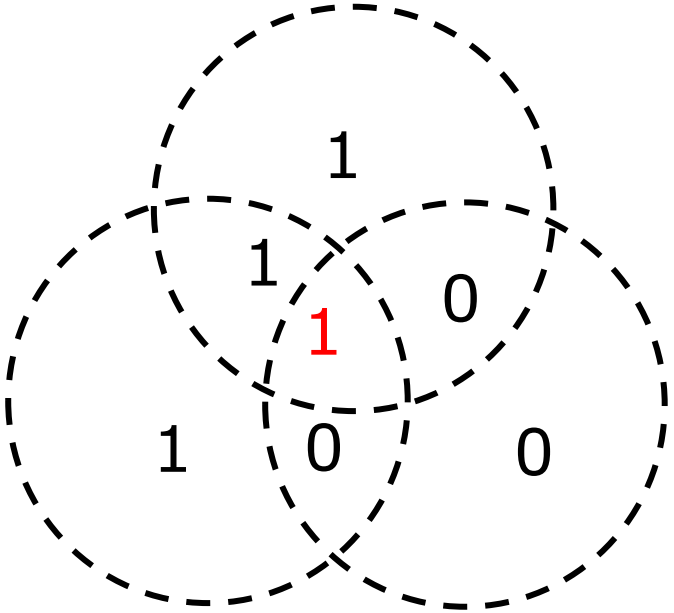
Hamming code
+
Steane code



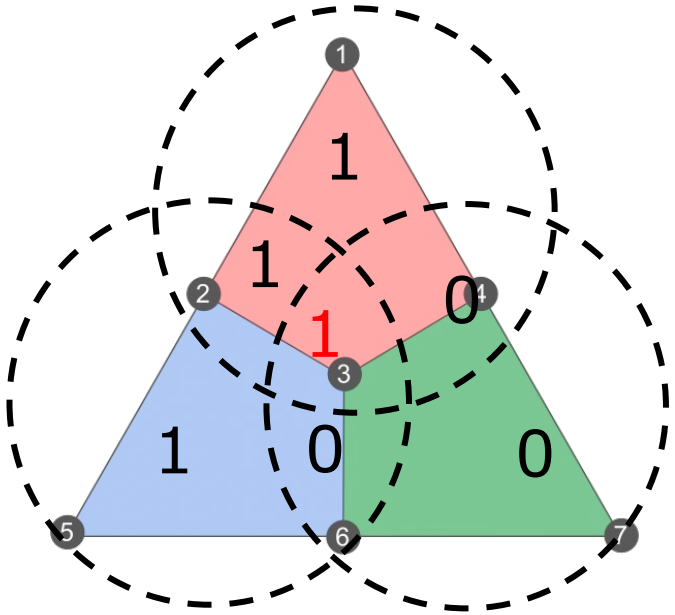
Steane code



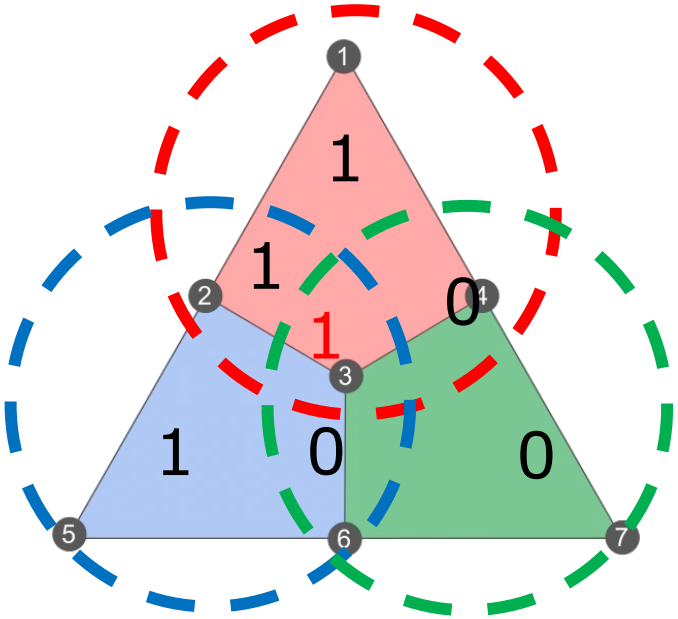
Hamming code



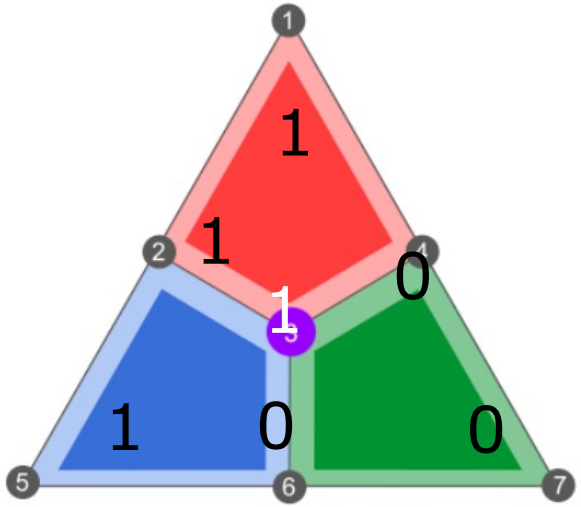
Hamming code
+
Steane code

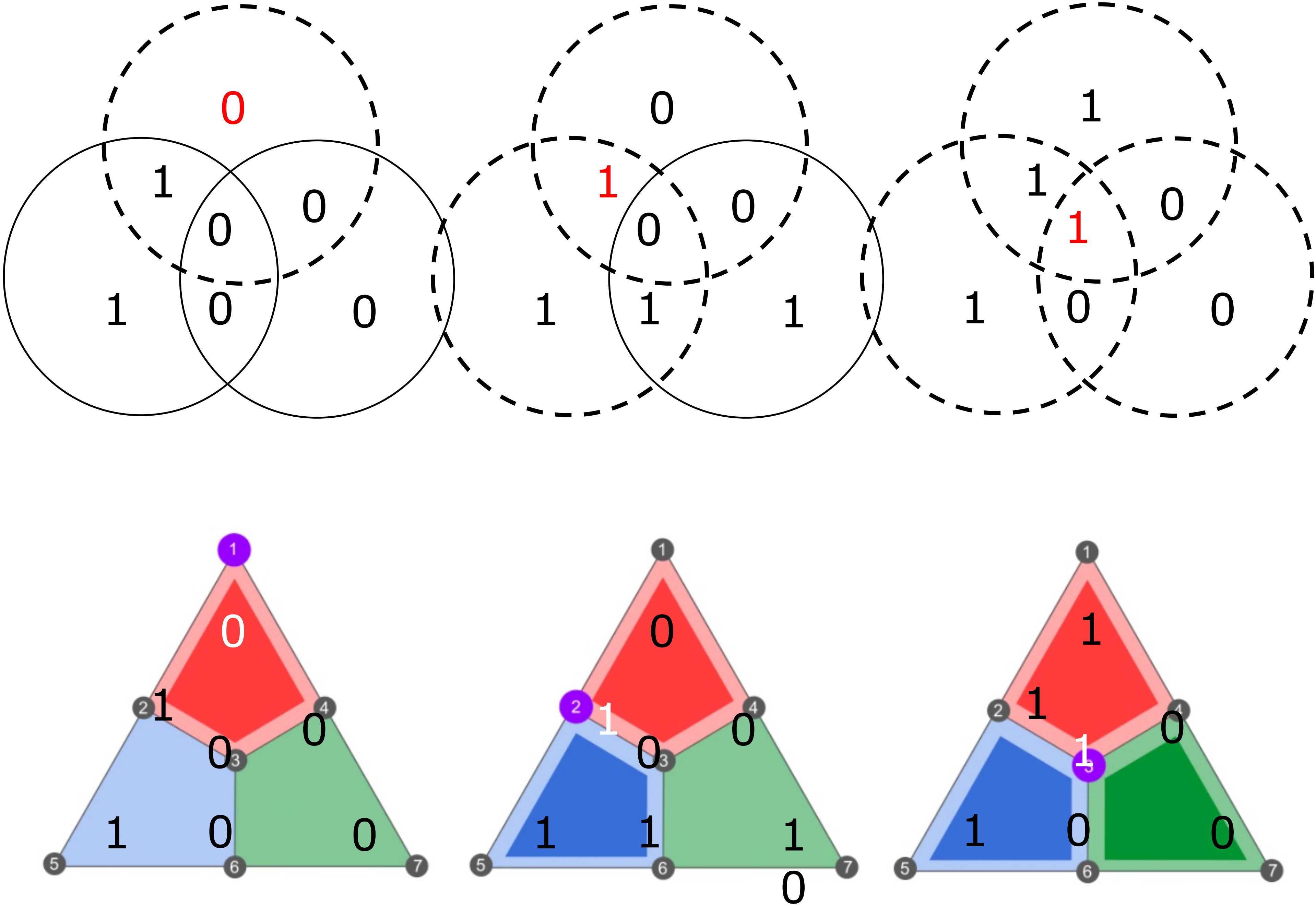


Hamming code
+
Steane code

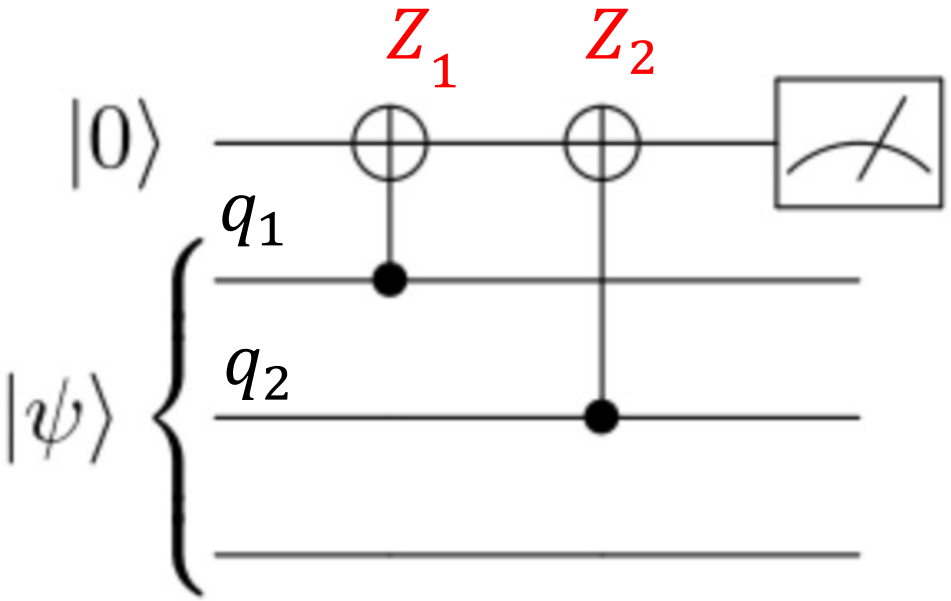


Steane code

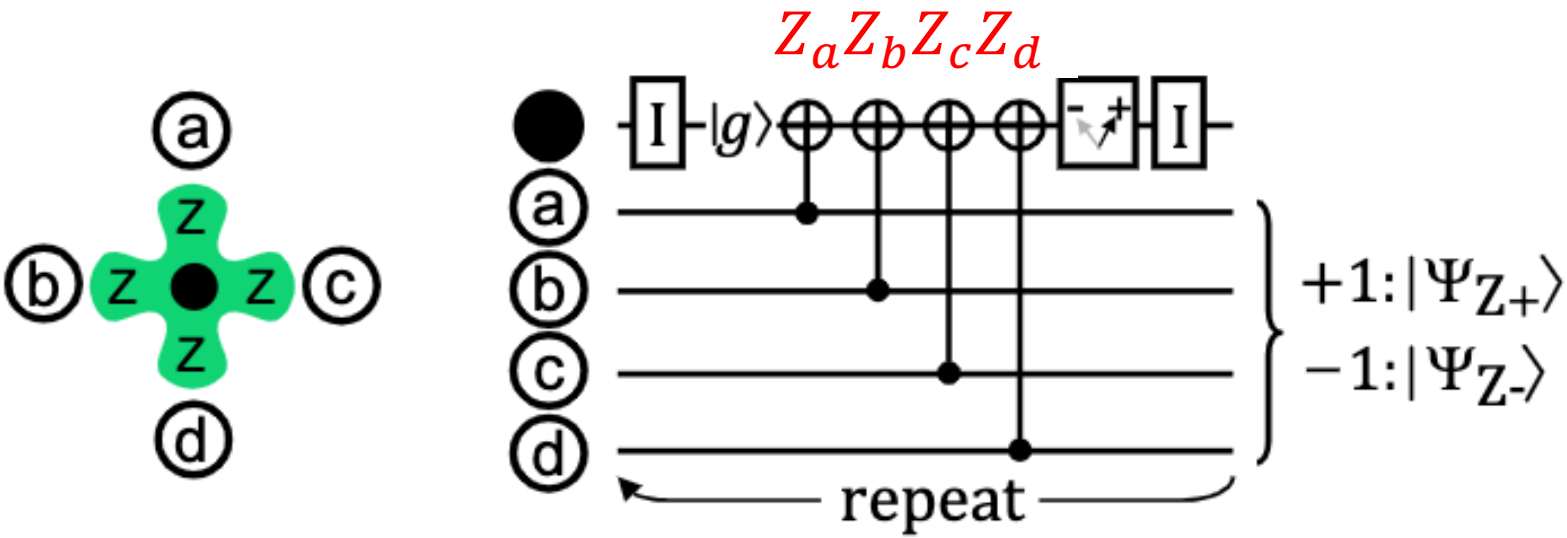


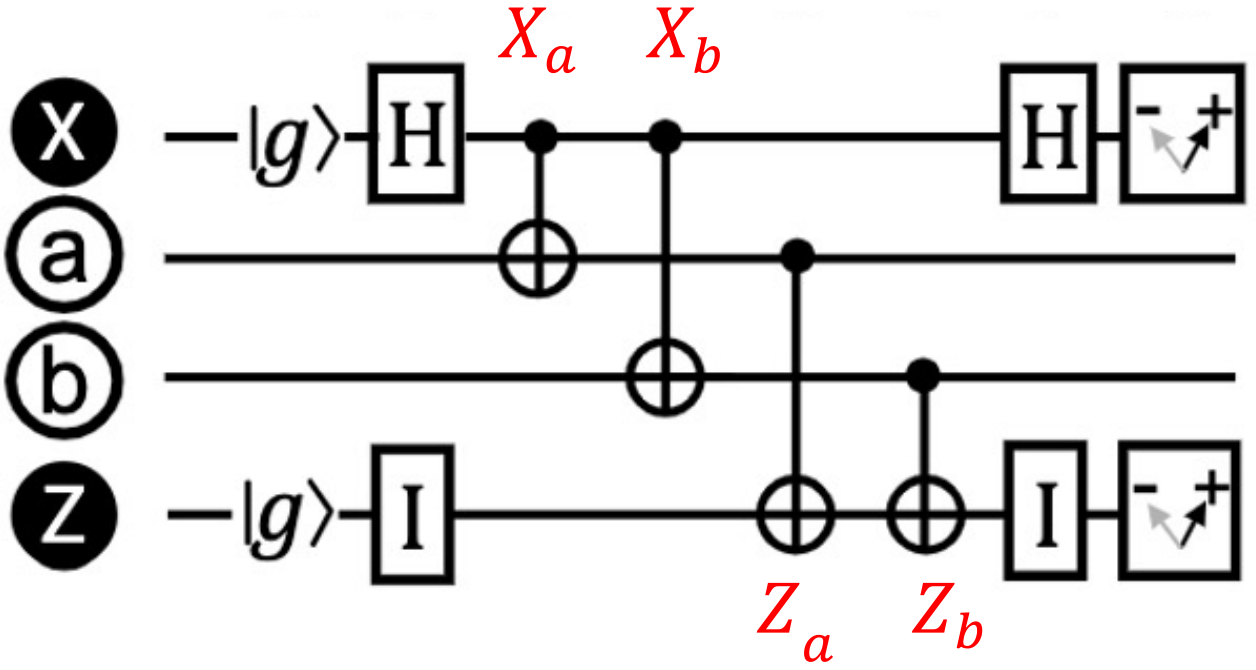


2つのqubit q_1, q_2のparity check をする量子回路　$Z_1 Z_2 |\psi>$



4つのqubit a, b, c, d のparity check をする量子回路　$Z_a Z_b Z_c Z_d$



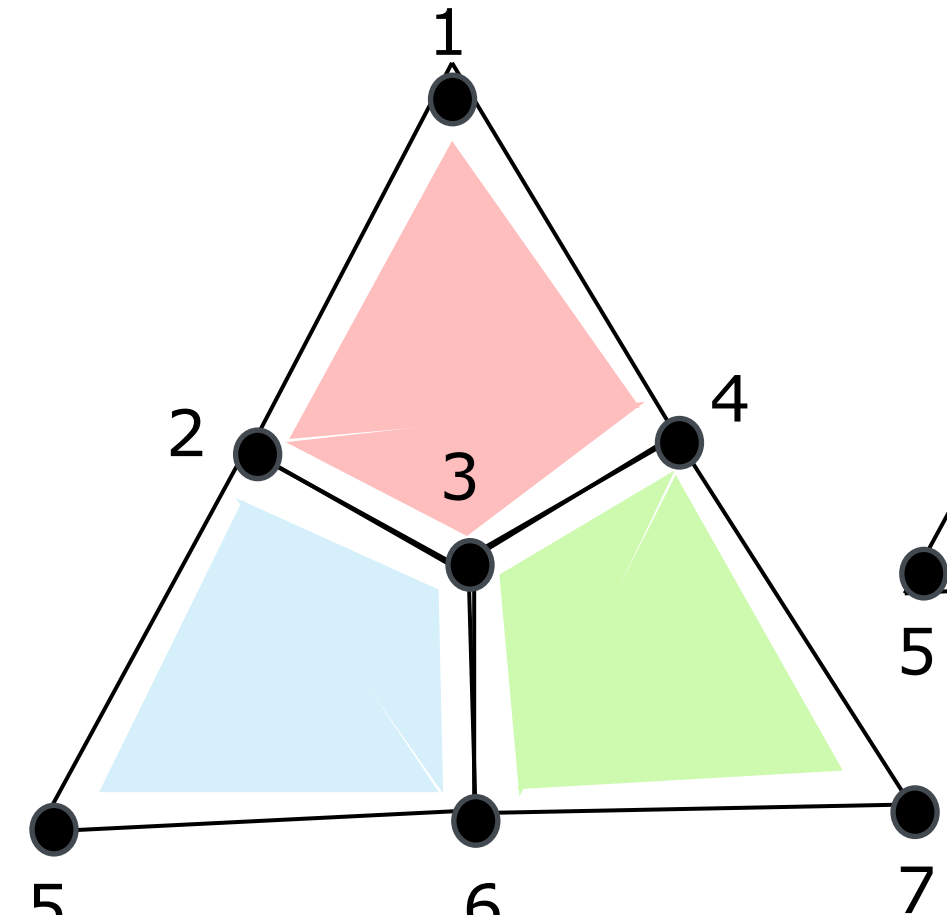


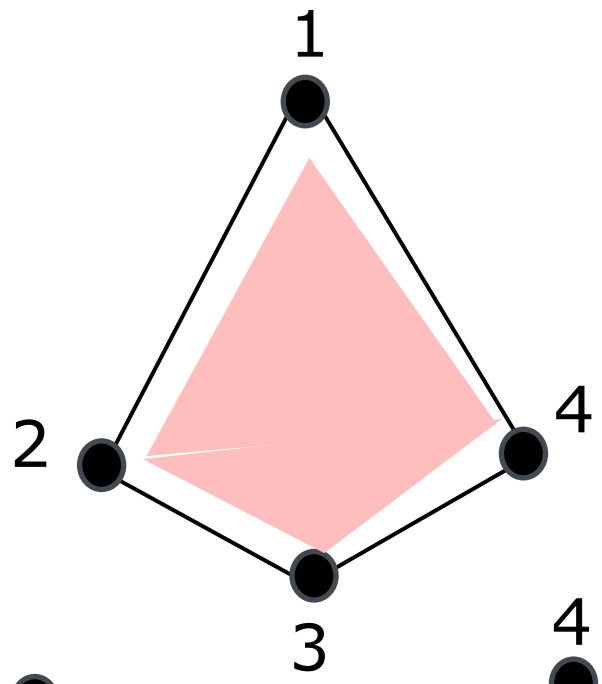
4つのレジスター X,a,b,Z があり、2つのアダマール・ゲート、2つのIゲート、4つのCNOTゲートから構成されています。

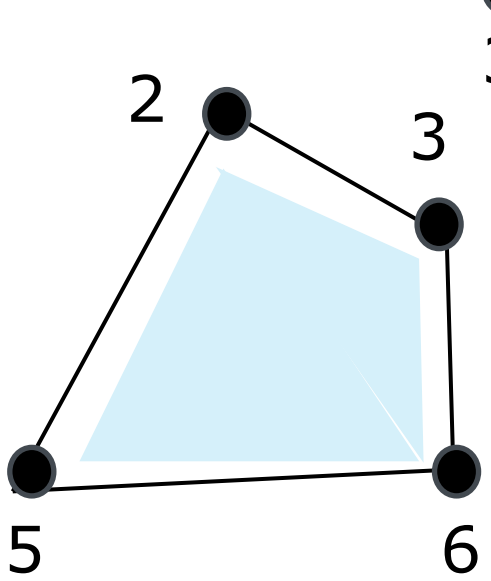
これ実は、2つのqubit a, b の状態を観測して、レジスター X, Z に出力するという量子回路なんです。

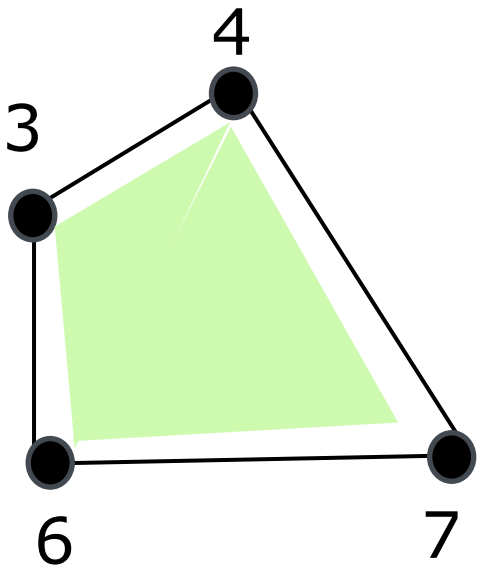
|g＞は、演算子Zの基底状態(ground state) で、|0＞のことと思って構いません。

Steane codeの模式図は
次のように分解できる

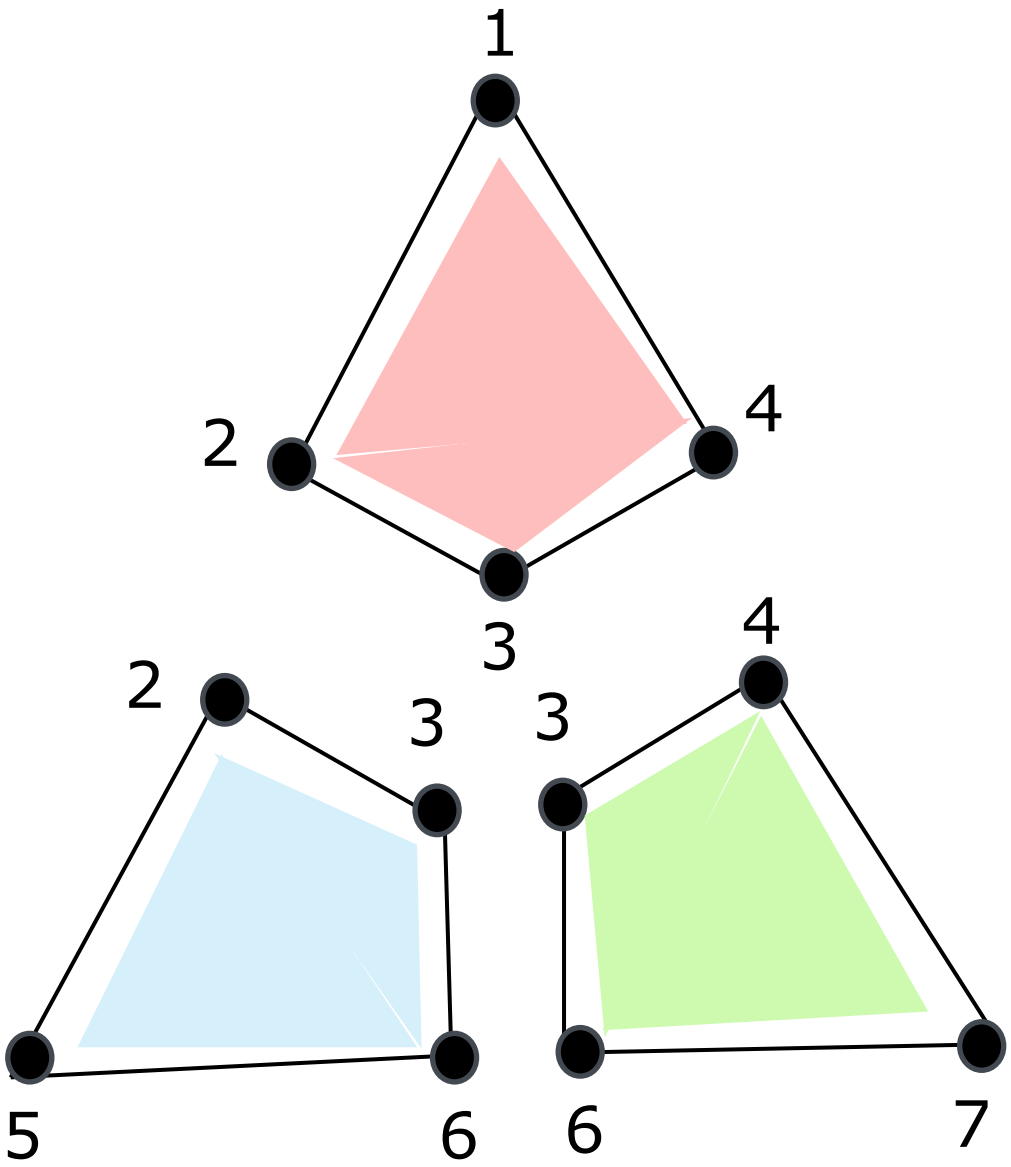


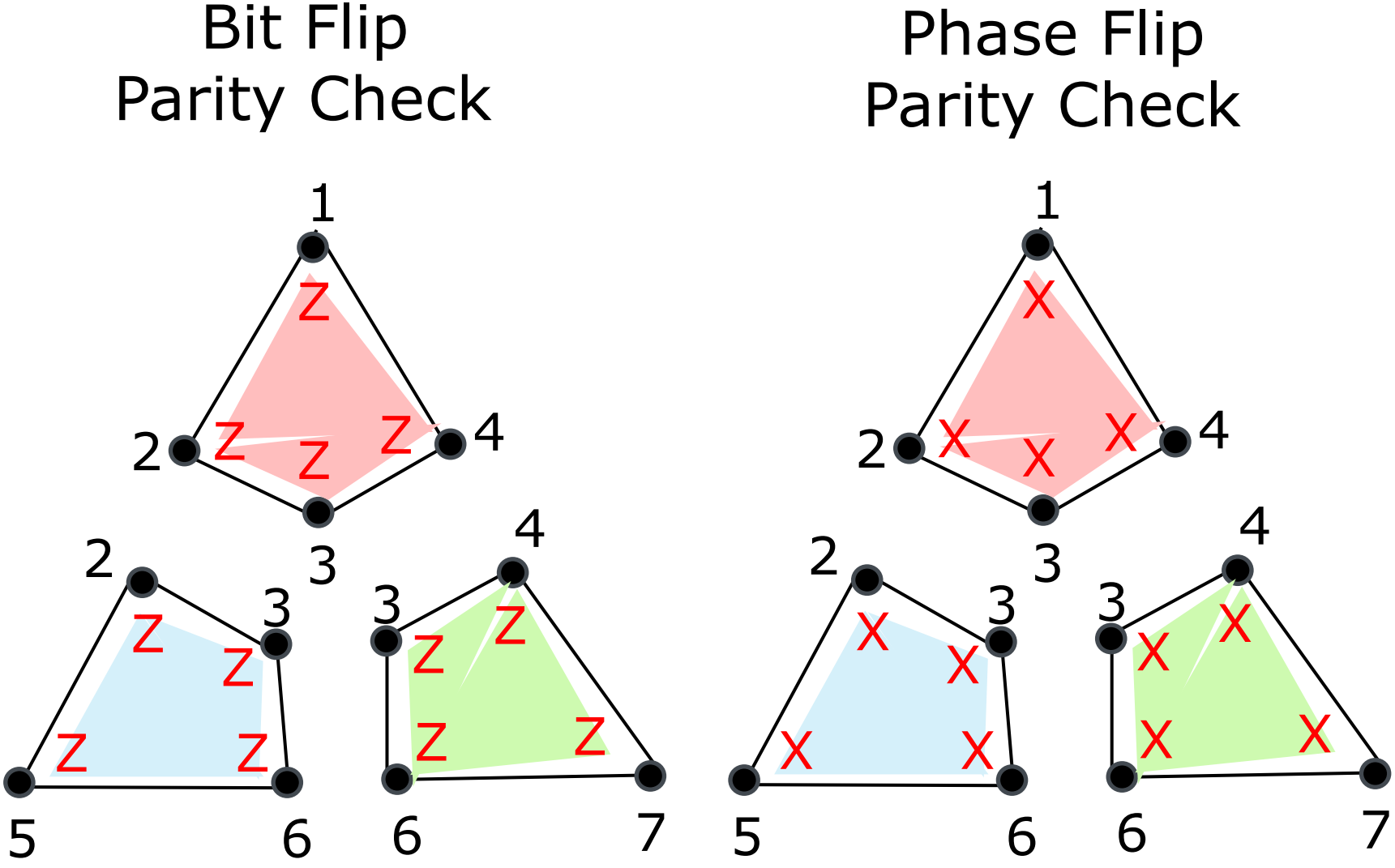




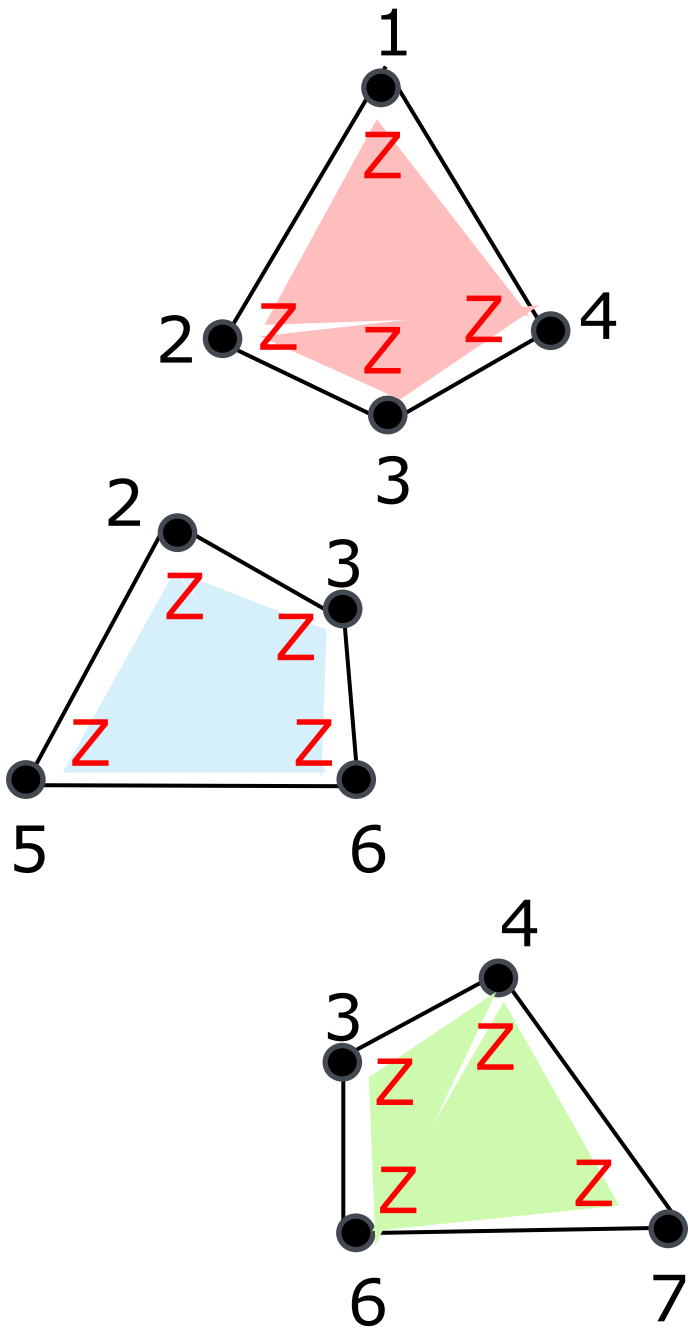


それぞれのパートが
parity checkの
単位となる




Bit Flip
Parity Check
Phase Flip
Parity Check
1
2
3
4
5
6
7
Z
X

Bit Flip
Parity Check



$Z_1 Z_2 Z_3 Z_4 |\psi> = |\psi>$

1, 2, 3, 4

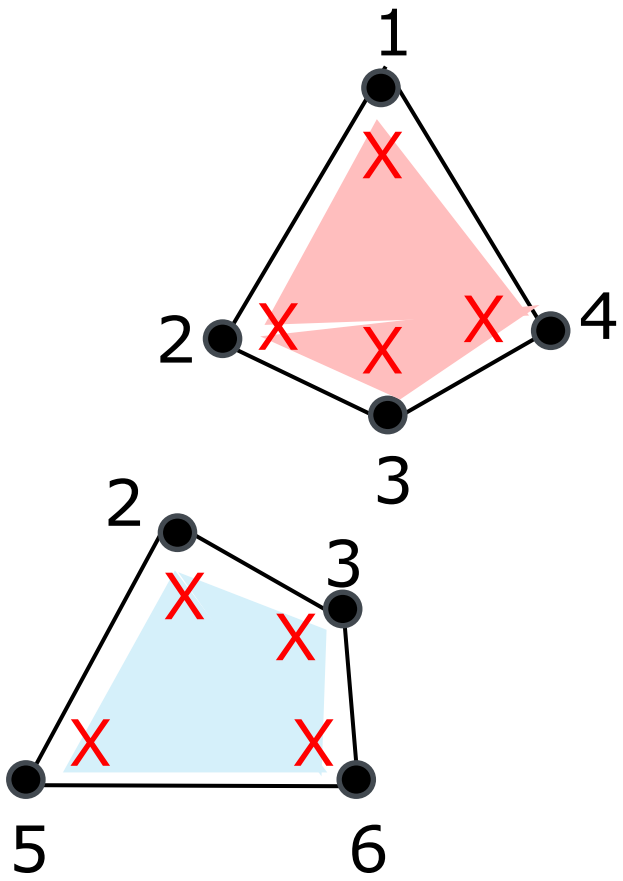
$Z_2 Z_3 Z_5 Z_6 |\psi> = |\psi>$

2, 3, 5, 6

$Z_3 Z_4 Z_6 Z_7 |\psi> = |\psi>$

3, 4, 6, 7

Phase Flip Parity Check

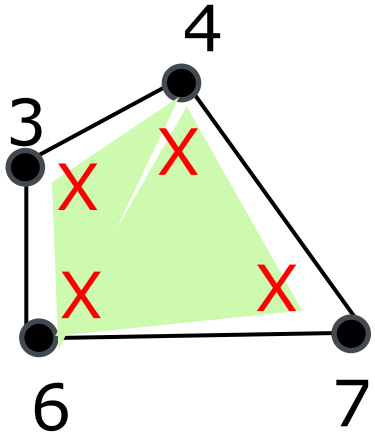


$X_1X_2X_3X_4|\psi> = |\psi>$

1, 2, 3, 4

$X_2X_3X_5X_6|\psi> = |\psi>$

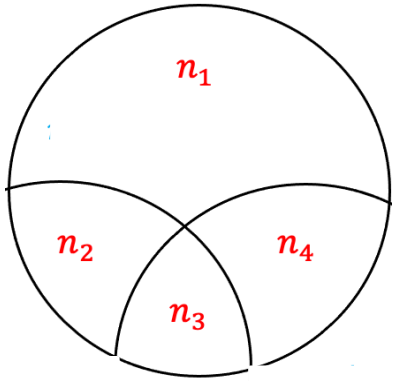
2, 3, 5, 6



$X_3X_4X_6X_7|\psi> = |\psi>$

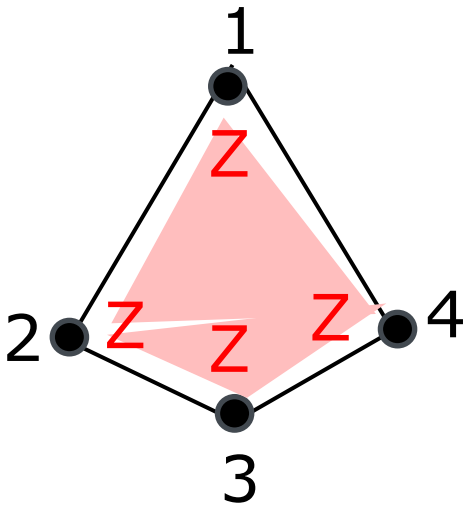
3, 4, 6, 7

No Parity Error



Hamming code

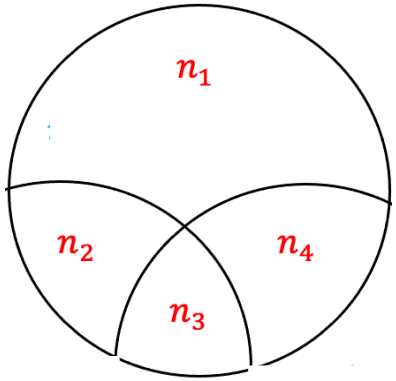
$$n_1 + n_3 + n_3 + n_4 = 0$$



Steane code

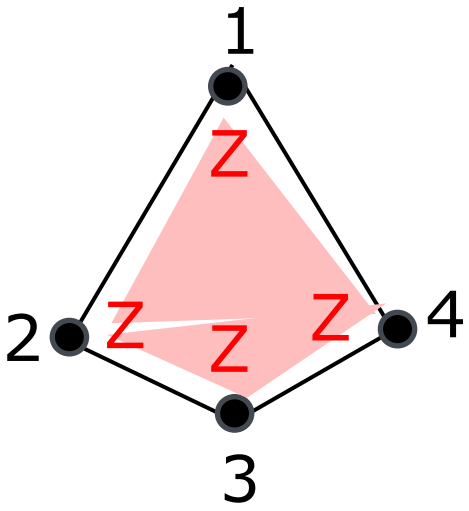
$$Z_1 Z_2 Z_3 Z_4 |\psi> = |\psi>$$

Parity Error



Hamming code

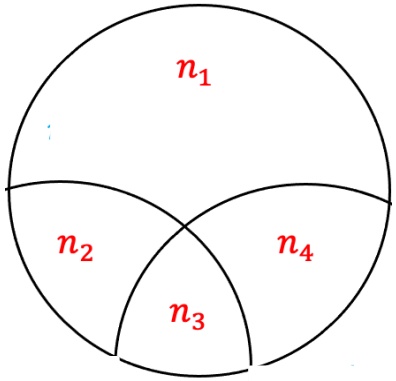
$$n_1 + n_3 + n_3 + n_4 = 1$$



Steane code

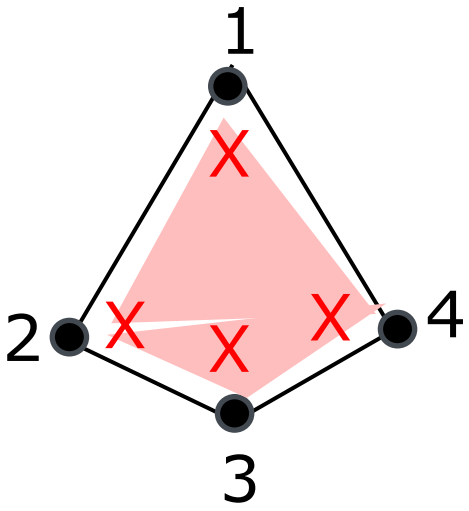
$$Z_1 Z_2 Z_3 Z_4 |\psi> = -|\psi>$$

No Phase Error



Hamming code

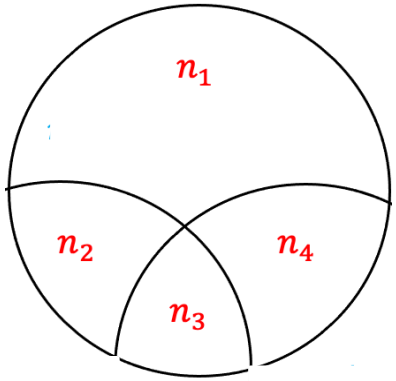
$$n_1 + n_3 + n_3 + n_4 = 0$$



Steane code

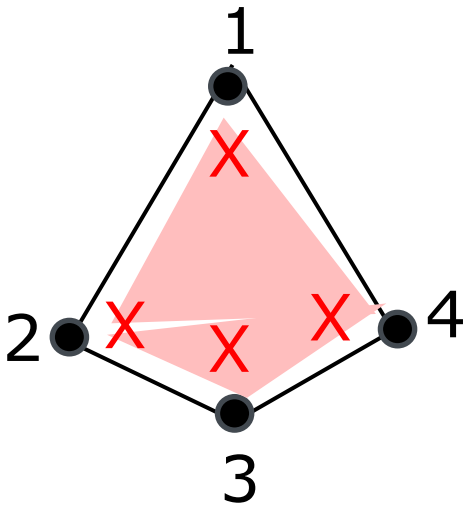
$$X_1 X_2 X_3 X_4 |\psi> = |\psi>$$

Phase Error



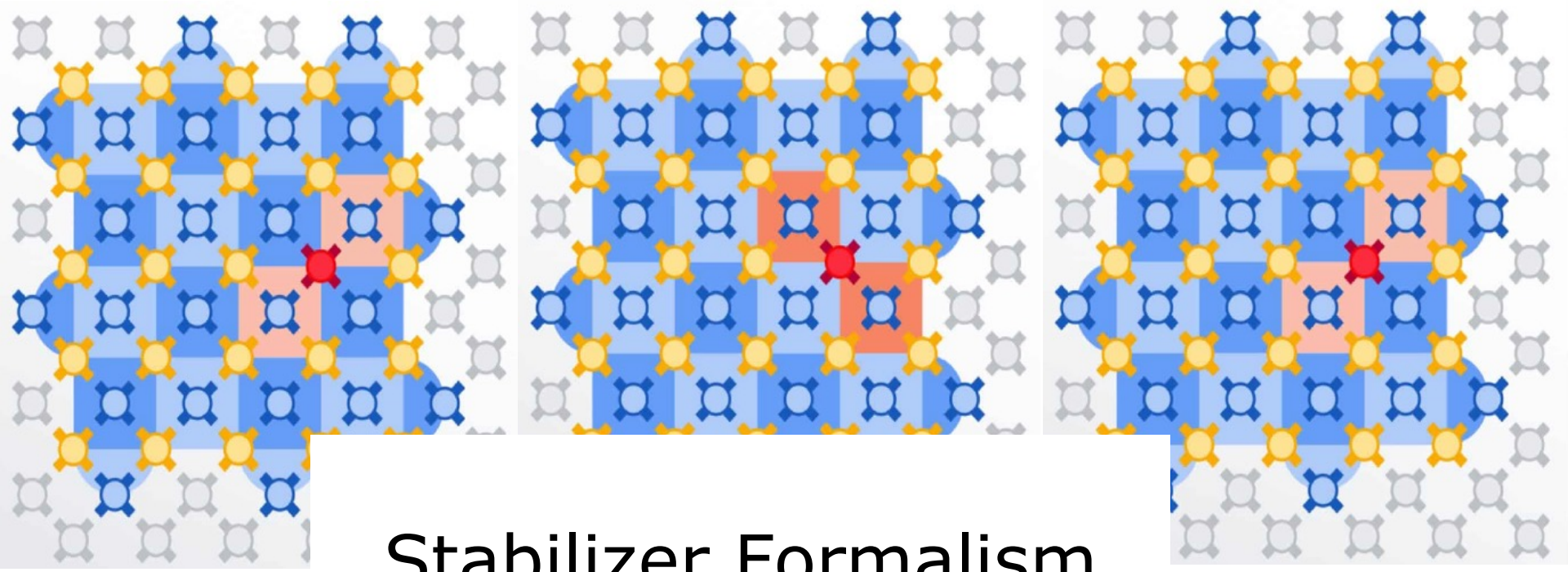
Hamming code

$$n_1 + n_3 + n_3 + n_4 = 1$$

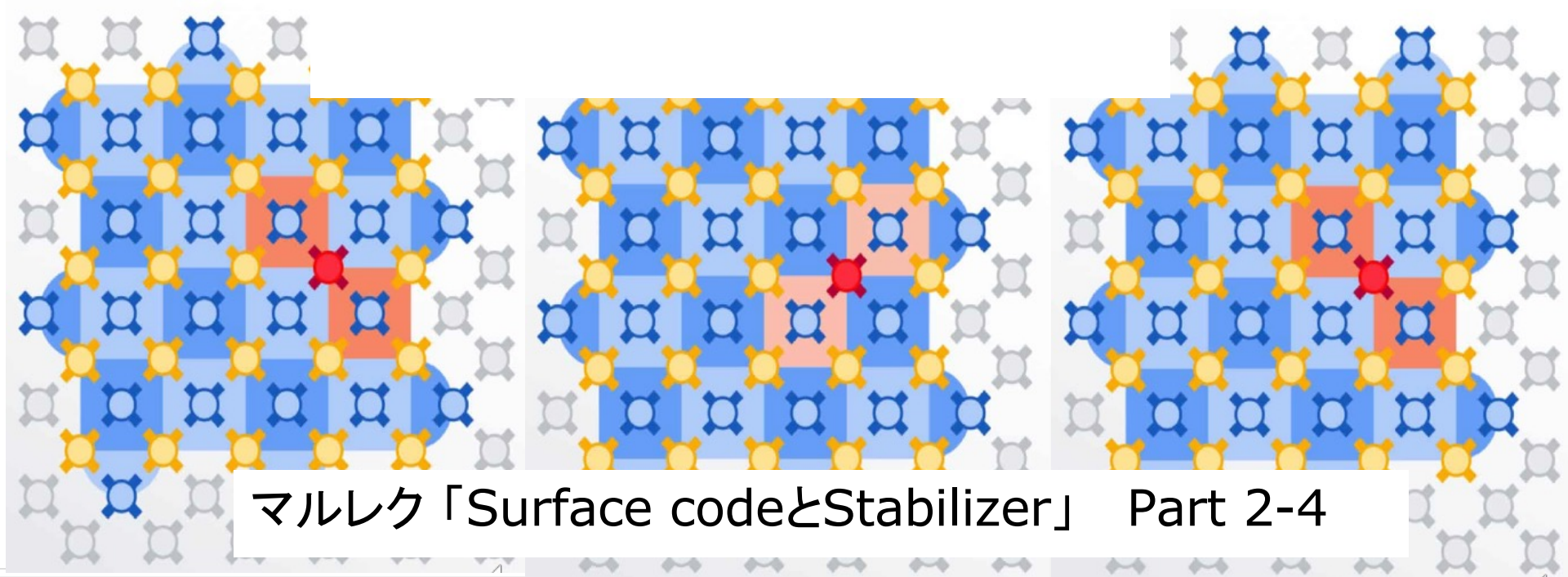


Steane code

$$X_1 X_2 X_3 X_4 |\psi> = -|\psi>$$



Stabilizer Formalism



マルレク「Surface codeとStabilizer」　Part 2-4

Stabilizer Formalism

今回のセッションは、量子エラー訂正技術の中で、Stabilizerという概念の重要性とその構成の数学的な基礎を明確に示したDaniel Gottesmanの “Stabilizer Formalism”の紹介です。

”Stabilizer Codes and Quantum Error Correction”

https://thesis.library.caltech.edu/2900/2/THESIS.pdf

この論文は、1997年のものですが、この分野での最も基本的な論文だと思います。

Pauli 演算子 I, X, Y, Z

二次元のベクトルで表現される一つのqubit $a|0> + b|1>$に作用する次の四つの演算子をPauli 演算子という。

$$I = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, Y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}, Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$$

この時、次の関係が成り立っている。

$$X^2 = Y^2 = Z^2 = I$$
$$XY = iZ \quad YX = -iZ \quad XY = -YX$$
$$YZ = iX \quad ZY = -iX \quad YZ = -ZY$$
$$ZX = iY \quad XZ = -iY \quad ZX = -XZ$$

Pauli 群 $\mathcal{P}_n$

Pauli 群 $\mathcal{P}_n$ は、通常の行列の積を群の演算として定義される、n個の qubitに作用するすべてのPauli演算子の集合のなす群である。

$$\mathcal{P}_n = \{\omega P_1 \otimes \cdots \otimes P_n : P_i \in \{I, X, Y, Z\}, \omega \in \{1, -1, i, -i\}\}$$

phase 因子 $\omega \in \{1, -1, i, -i\}$は、この集合が群の演算で閉じているために必要である。

stabilizer 群 $\mathcal{S}$ とcodespace

stabilizer 群 $\mathcal{S}$ は、Pauli 群 $\mathcal{P}_n$ の可換部分群 $\mathcal{S} \subset \mathcal{P}_n$ で、$-I$ を含まないものをいう。

stabilizer 群 $\mathcal{S}$ の要素は、すべてPauli 群 $\mathcal{P}_n$の要素（ただし $-I$ 以外）からなり、固有値 +1と−1を持つ。

stabilizer 群 $\mathcal{S}$ の要素は、エラー訂正の古典論の文脈では、可能なcodewordの集合に対応するものである。それをcodespaceと呼ぼう。

正しくエンコードされたcodewordに対応するcodespaceの部分集合を次に定義し、stabilizer codeと呼ぶことにする。

stabilizer コード $\mathcal{C}$

stabilizer コード $\mathcal{C}$ を、あるstabilizer 群$\mathcal{S}$の固有値 +1 を持つすべての固有状態の集合として定義する。

$$C = \{|\psi>: S|\psi> = |\psi>, \forall S \in \mathcal{S}\}$$

定義から すべてのstabilizerは可換であり、共通の固有状態を持つ。

$S|\psi> = |\psi>$である時、Sは$|\psi>$を「stabilizeする」という。

トリビアルなstabilizerの例

n=1、すなわち、1-qubitに作用するでstabilizer群を考えてみよう。

Iは、全てをstabilizeする。 $I|\psi>=|\psi>$

$-I$は、なにもstabilizeしない。 $-I|\psi>\neq|\psi>$

(これが-Iを、stabilizer群の定義から除外した理由である)

X は |+> をstabilizeする。 $X|+>=|+>$

–X は |–> をstabilizeする。 $-X|->=|->$

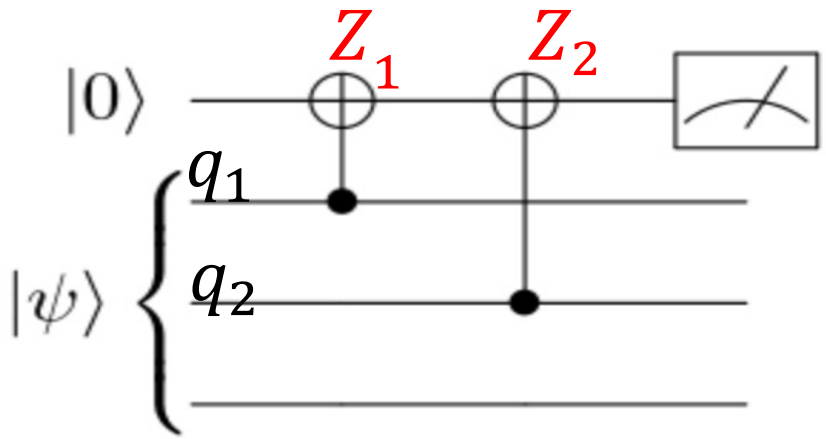
Z は |0> をstabilizeする。 $Z|0>=|0>$

–Z は |1> をstabilizeする。 $-Z|1>=|1>$

Y は |i> をstabilizeする。 $Y|i>=|i>$

–Y は |–i> をstabilizeする。 $-Y|-i>=|-i>$

2-qubitのstabilizerの例



$Z_1 Z_2 |\psi> = |\psi>$ の時

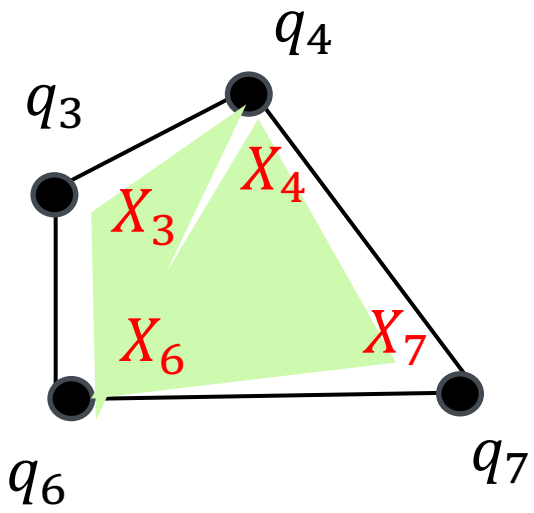
$Z_1 \otimes Z_2$は$|q_1 \otimes q_2>$をstabilizeし、

$|q_1 \otimes q_2> \in C$ (Cは。stabilizer code)

$Z_1 Z_2 |\psi> = -|\psi>$ の時

$|q_1 q_2> \notin C$ (Error syndrome)

4-qubitのstabilizerの例



$X_3X_4X_6X_7\,|\psi> = |\psi>$ の時

$X_3 \otimes X_4 \otimes X_6 \otimes X_7$は$|q_3 \otimes q_4 \otimes q_6 \otimes q_7>$をstabilizeし、

$|q_3 \otimes q_4 \otimes q_6 \otimes q_7 > \in C$ (Cは。stabilizer code)

$X_3X_4X_6X_7\,|\psi> = -|\psi>$ の時

$|q_3q_4q_6q_7> \notin C$ (Error syndrome)

codespaceとstabilizer

先に定義したcodespaceとstabilizer群は、一対一に対応している。この対応が、量子エラー訂正へのstabilizer応用の基礎となる。

重要なことは、古典論での、可能な「状態」のn次元のcode空間の中で、k次元の正しいcodeword の空間を考えるという「状態」ベースのイメージは、量子論では、stabilizer 群の中で、ある状態をstabilizeする「演算子」を考えるという、「演算子」ベースのイメージに変わったことである。

こう書くと、少しわかりにくいかもしれないが、重要なことは、先にstabilizer群の部分集合として定義したstabilizer codeの中で、これらの演算子が果たしている役割は、古典論での「パリティ・チェック」に対応するものである。

stabilizer code と syndrome

stabilizer code がどのように、エラーの検出・訂正に利用されるのかを見ておこう。

それは、単純なことだ。基本的にはすべてのstabilizerを観測すればいいというものである。この観測の結果はsyndromeと呼ばれる。

もしもエラーがなければ、すべてのstabilizerは、+1の固有値を持つ。

もし Pauli エラー Eが起きたとすると、それぞれのstabilizer Sとエラー Eとの関係には二つの可能性がある。演算子 Sとエラー Eが、交換関係を満たす場合と、反交換関係を満たす場合である。

SとEが、交換関係を満たす場合、$SE = ES$ なので、我々は $+1$ を観測することは、次のようにして分かる。

$$SE \mid \psi > = ES \mid \psi > = E \mid \psi >$$

SとEが、反交換関係を満たす場合、$SE = -ES$ なので、我々は -1 を観測する

$$SE \mid \psi > = -ES \mid \psi > = -E \mid \psi >$$

Xエラーを検出するには、ZかYからなる stabilizerが必要である。
$(ZX = -XZ,\ XY = -YX)$

Zエラーを検出するには、XかYからなる stabilizerが必要である。
$(ZX = -XZ,\ XY = -YX)$

stabilizer code と エラー訂正

syndrome を分析することで、エラーを訂正できることがある。

量子コードの特殊性として、そのエラーが正確にどのqubit でおきたかを知る必要がないことが起きる。(Shor code でもそうであった) それについては、別のセッションで述べようと思う。

必要なことは、次のような元の状態をエラーEから回復する訂正演算子 C を見つけることである。すなわち、

$$CE\,|\,\psi\,>\,=\,|\,\psi\,>$$

これは、CEがstabilizer であることを意味する。