

量子計算機アーキテクチャ分野の研究動向2025

資料: 上野のresearchmapで公開

理研量子コンピュータ研究センター 基礎科学特別研究員

東京大学大学院情報理工学系研究科 特任助教

上野 洋典

上野 洋典（うえの ようすけ）

- 所属：理研RQC 超伝導量子計算システム研究ユニット
基礎科学特別研究員（ポスドク）
- 経歴：
 - 2013.4~2017.3: 東大工学部計数工学科
 - 2017.4~2022.3: 東大情報理工 システム情報学専攻
 - 教員：近藤正章先生（慶應）、中村宏先生（いずれも**計算機アーキテクチャ**研究者）
 - 博論：超伝導デジタル回路を用いたオンライン量子誤り訂正
 - 2022.5~2023.2: ミュンヘン工科大学 訪問研究員
 - 2023.4~現在: (本務) 理研RQC 基礎科学特別研究員
 - 2025.5~現在: (兼務) 東大 中村・高瀬研 特任助教（非常勤）
- 研究対象、興味：
 - 計算機アーキテクチャ、超伝導デジタル（SFQ）回路、誤り耐性量子計算
- 物理の方々に囲まれつつ計算機アーキテクチャやってます！



ドイツでの
最初と最後のビール

今日の内容

- 量子計算機アーキテクチャ分野の研究動向
 - 計算機アーキテクチャの研究対象・隣接分野
 - 計算機アーキテクチャ分野における量子計算機関連の研究動向
- 量子誤り訂正とそれを支える計算機アーキテクチャ
 - 表面符号における誤り推定
 - 超伝導デジタル回路を用いた誤り推定機構（DAC2021/HPCA2022）
 - 表面符号＋格子手術を用いた誤り耐性量子計算
 - ロードストア型FTQCアーキテクチャ（HPCA2025）
- まとめ

計算機アーキテクチャ

What is **Computer architecture**?

- “*Computer Architecture* is the science and art of selecting and interconnecting hardware components to create computers that meet **functional**, **performance** and **cost** goals.”

- WWW Computer Architecture Page

建築（アーキテクチャ）とのアナロジー



計算機アーキテクチャ分野の研究対象

高
↓
対象度
↓
低

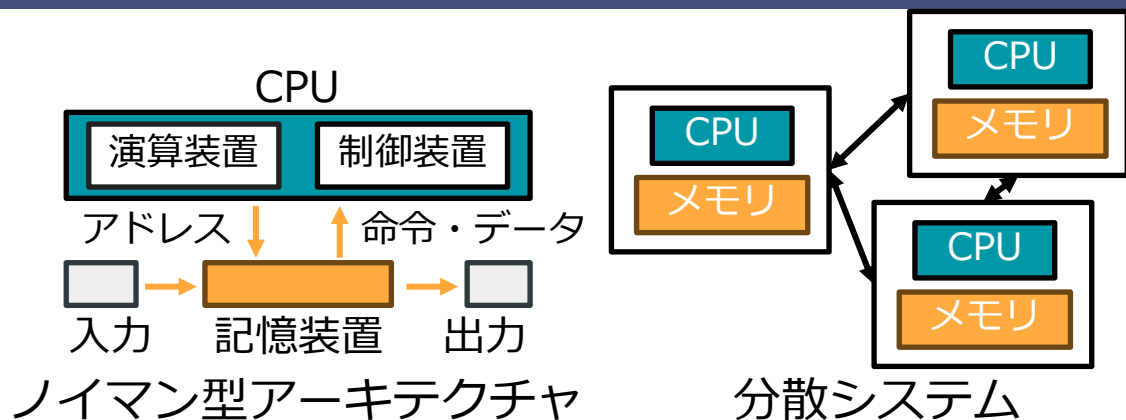
計算機全体の構成
計算機を取り巻く環境

システムアーキテクチャ
ノイマン型アーキテクチャ,
キャッシュ, 分散システム

命令セットアーキテクチャ
(ISA)
RISC, CISC, VLIW

計算機の中身

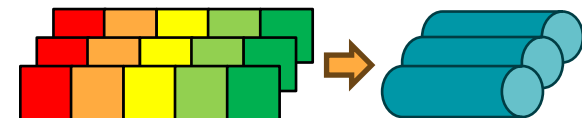
マイクロアーキテクチャ
CPU, パイプライン,
投機的実行



x86 (CISC) vs. RISC-V (RISC)

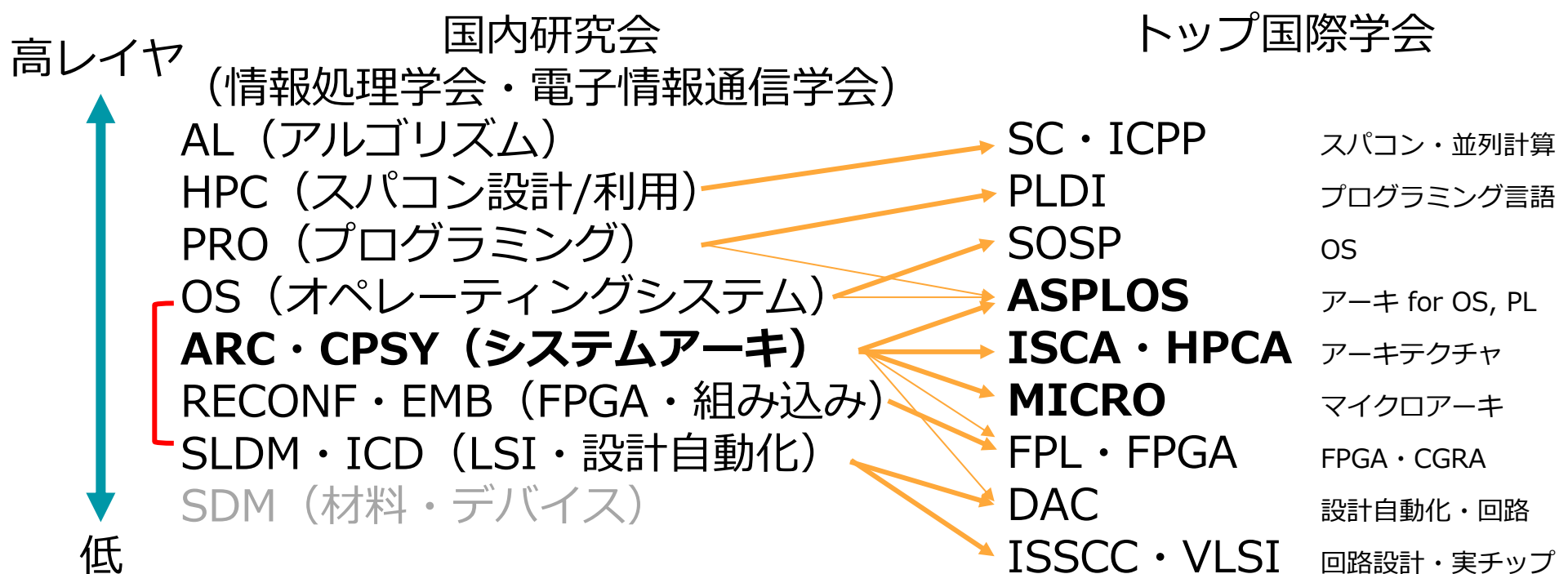


パイプライン



ベクトル演算器

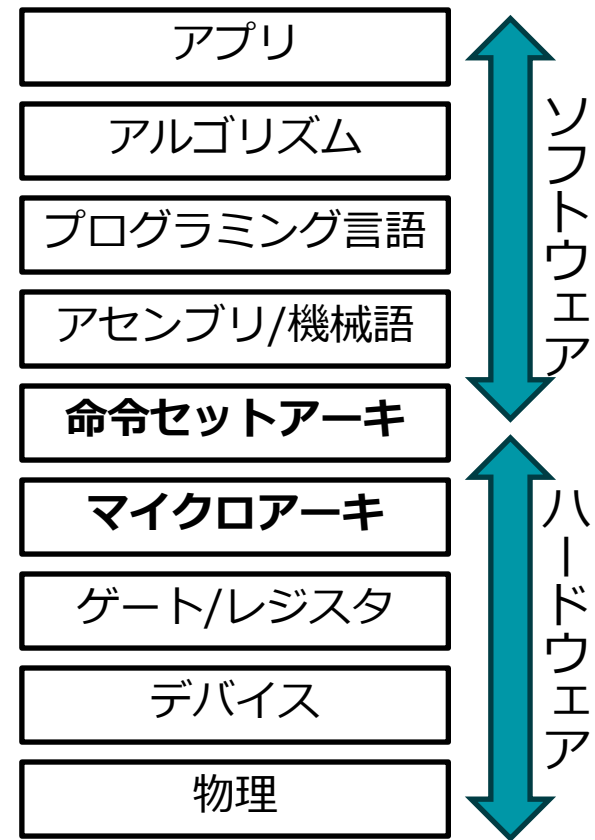
計算機アーキテクチャの隣接分野



上は計算機と不可分なソフトウェアまで
下はビットまで

計算機アーキテクチャの役割

- 抽象化のレベルを適切に定める
 - プログラマが何を意識して何を意識しないか
- 各レイヤの変化に応じてコストを最小化しつつ計算機的设计を更新
- 各コンポーネントを統合して計算機全体としての性能見積もり、フィードバック
 - どこに何を押し付けるか決める
- **計算能力の継続的な向上の展望**を示す



計算機アーキテクチャ研究の特徴

- 現実志向：今ある計算機をどのように「うまく」使うか
 - 「うまい」 = 速い、消費電力が小さい、安全 etc.
 - 観察 -> 問題の特定 -> 解決策の提示
- 未来志向：将来の計算機がどのようなになるか、
どんな問題が生じるか、どのように解決できるか
 - 未来の計算機の姿を妥当に予想 -> 問題の特定 -> 解決策の提示
 - 将来生じうる問題を先回りして解決しておく

今日の内容

- 量子計算機アーキテクチャ分野の研究動向
 - 計算機アーキテクチャの研究対象・隣接分野
 - 計算機アーキテクチャ分野における量子計算機関連の研究動向
- 量子誤り訂正とそれを支える計算機アーキテクチャ
 - 表面符号における誤り推定
 - 超伝導デジタル回路を用いた誤り推定機構（DAC2021/HPCA2022）
 - 表面符号＋格子手術を用いた誤り耐性量子計算
 - ロードストア型FTQCアーキテクチャ（HPCA2025）
- まとめ

量子計算機アーキテクチャ分野

システムアーキテクチャ

命令セットアーキテクチャ (ISA)

マイクロアーキテクチャ

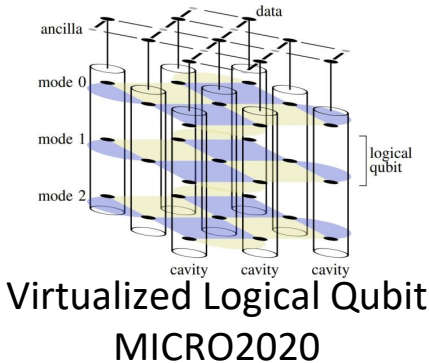
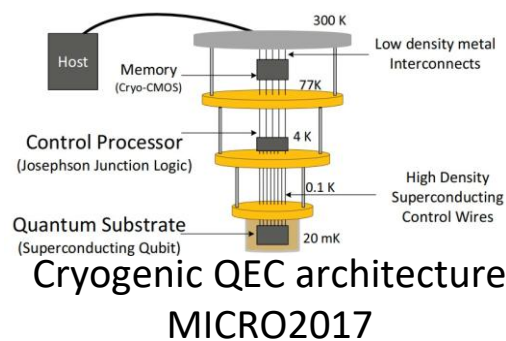
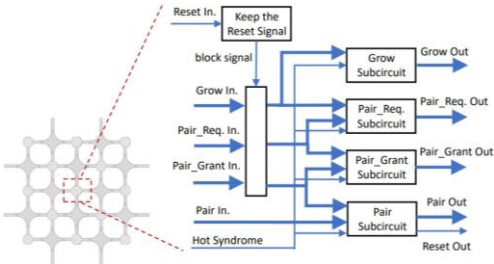


Table 1. Overview of eQASM instructions. The operator :: concatenates the two bit strings.

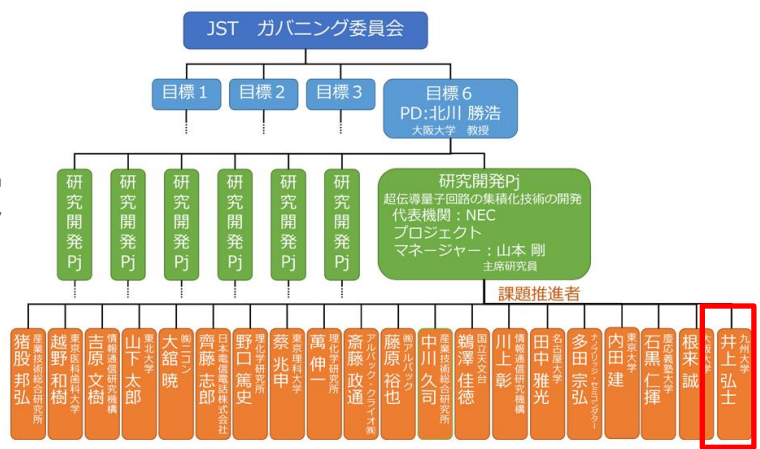
Type	Syntax	Description
Control	CMP R _s , R _t	CoMPare GPR R _s and R _t and store the result into the comparison flags.
	BR <Comp. Flag>, Offset	(BRanch) Jump to PC + Offset if the specified comparison flag is '1'.
	FBR <Comp. Flag>, Rd	(Fetch Branch Register) Fetch the specified comparison flag into GPR Rd.
	LDI Rd, Imm	(Load Immediate) Rd = sign_ext(Imm[19:0], 32).
Data Transfer	LDUI Rd, Imm, R _s	(Load Unsigned Immediate) Rd = Imm[14:0]:R[16:0].
	LD Rd, R _t (Imm)	(Load from memory) Load data from memory address R _t + Imm into GPR Rd.
	ST R _s , R _t (Imm)	(Store to memory) Store the value of GPR R _s in memory address R _t + Imm.
	FMR Rd, Q1	(Fetch Measurement Result) Fetch the result of the last measurement instruction on qubit i into GPR Rd.
Logical	AND/OR/XOR Rd, R _s , R _t	Logical and, or, exclusive or, not.
Arithmetic	ADD/SUB Rd, R _s , R _t	Addition and subtraction.
Waiting	QWAIT Imm	(Quantum WAIT Immediate/Register) Specify a timing point by waiting for the number of cycles indicated by the immediate value Imm or the value of GPR R _s .
Target Specify	SMIS Sd, <Qubit List>	(Set Mask Immediate for Single/Two-qubit operations) Update the single- (two-)qubit operation target register Sd (Td).
Q. Bundle	[P1:] Q.Op [: Q.Op]*	Applying operations on qubits after waiting for a small number of cycles indicated by P1.



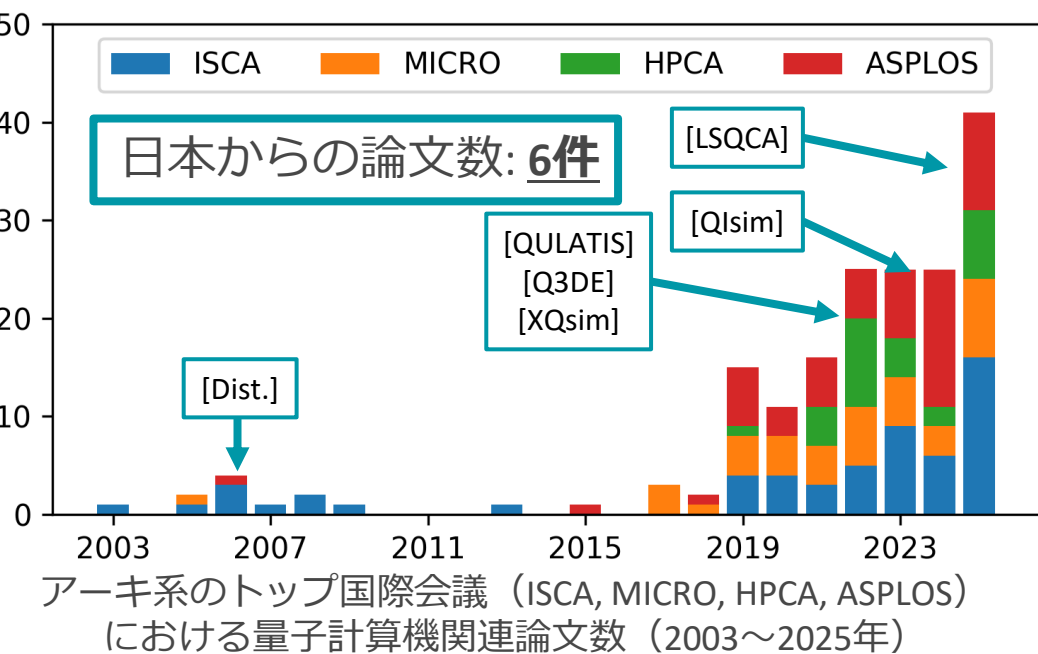
K. Bertels et al., “eQASM: An Executable Quantum Instruction Set Architecture,” in HPCA2019.
C. Duckering et al., “Virtualized Logical Qubits: A 2.5 D Architecture for Error-Corrected Quantum Computing,” in MICRO2020.
S. Tannu et al., “Taming the Instruction Bandwidth of Quantum Computers via Hardware-Managed Error Correction,” in MICRO2017.
A. Holmes et al., “NISQ+: boosting quantum computing power by approximating quantum error correction,” in ISCA2020.

量子計算機開発におけるアーキテクチャの知見

- 理論では考えられていないコンポーネントを詰める
 - 例: 誤り訂正符号のエラー推定が多項式時間の古典処理でできる
 - 誰がどこで解くのか、計算機全体の構成はどうなるのか
- 各要素技術を統合して考えて、性能見積もり、開発へのフィードバック
 - 例: 超伝導ムーンショットの井上先生チーム
- 経済的な制約
 - 例: $p = 10^{-5}$ の量子ビット+軽量誤り推定 vs. $p = 10^{-3}$ の量子ビット+高精度誤り推定



国内外の量子計算機アーキテクチャ研究動向

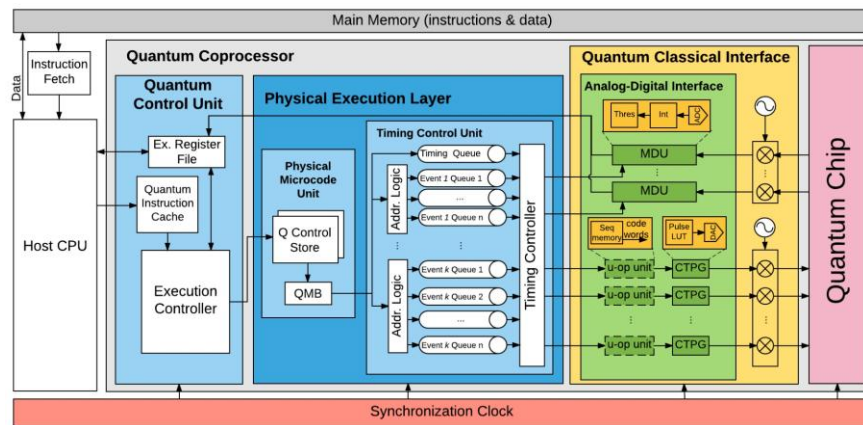


年	量子関連論文割合
2003~2018	0~1%程度
2019	5.4% (15/276)
2020	3.6% (11/308)
2021	5.0% (16/323)
2022	7.7% (25/325)
2023	6.1% (25/412)
2024	5.3% (25/475)
2025	7.6% (41/539)

トップ国際会議における量子計算機関連論文数

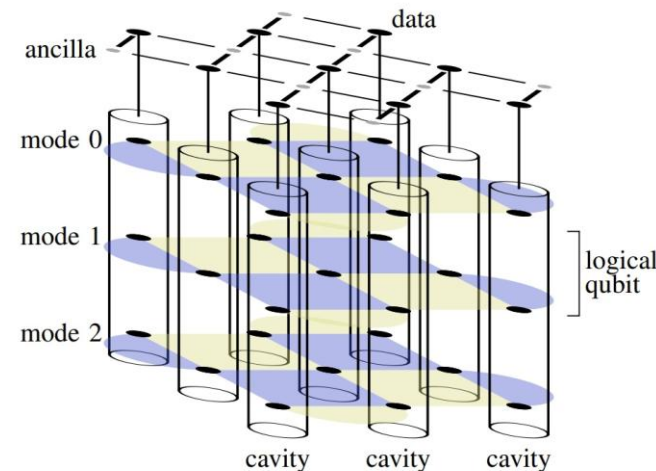
[Dist.] R. Van Meter, W. Munro, K. Nemoto, K. Itoh, "Distributed Arithmetic on a Quantum Multicomputer", **ISCA2006**.
[QULATIS] Y. Ueno, M. Kondo, M. Tanaka, Y. Suzuki, Y. Tabuchi, "QULATIS: A Quantum Error Correction Methodology toward Lattice Surgery", **HPCA2022**.
[Q3DE] Y. Suzuki, ..., K. Inoue, T. Tanimoto, "Q3DE: A fault-tolerant quantum computer architecture for multi-bit burst errors by cosmic rays", **MICRO2022**.
[XQsim] I. Byun, ..., T. Tanimoto, M. Tanaka, K. Inoue, J. Kim, "XQsim: modeling cross-technology control processors for 10+K qubit quantum computers", **ISCA2022**.
[Qlsim] D. Min, ..., M. Tanaka, K. Inoue, J. Kim, "Qlsim: Architecting 10+K Qubit QC Interfaces Toward Quantum Supremacy", **ISCA2023**.
[LSQCA] T. Kobori, Y. Suzuki, Y. Ueno, T. Tanimoto, S. Todo, Y. Tokunaga, "LSQCA: Resource-Efficient Load/Store Architecture for Limited-Scale Fault-Tolerant Quantum Computing", **HPCA2025**.

量子計算機アーキテクチャ研究のインパクト



Quantum Microarchitecture (Delft U.)

MICRO'17 Best paper



Virtualized Logical Qubits (Chicago U.)

MICRO'20 Best paper runner-up

- 計算機アーキテクチャ分野でも量子計算機研究は評価される
- 物理（量子情報）＋アーキテクチャの知見でインパクト大

X. Fu et al., "An Experimental Microarchitecture for a Superconducting Quantum Processor", MICRO 2017.

C. Duckering et al., "Virtualized Logical Qubits: A 2.5D Architecture for Error-Corrected Quantum Computing", MICRO 2020.

国内外の量子計算機アーキテクチャ研究動向

主要な研究グループ	論文数（割合）	First quantum paper in top conferences
University of Chicago (Fred Chong (@UCSB until 2015))	37本 (21.0%)	ISCA2003
Georgia Tech. (Moinuddin Qureshi, Swamit Tannu (UW-Madison))	15+8本 (13.1%)	MICRO2017
UC San Diego (Yufei Ding)	20本 (11.3%)	ASPLOS2019
Princeton University (Margaret Martonosi)	14本 (7.9%) (内Chicagoと共同6)	ISCA2007

- 50%の論文が上位4グループから出ている

各グループの初期の量子関連論文

Chicago University

- [初期] M. Oskin, F. Chong, I. Chuang, J. Kubiatowicz,
Building Quantum Wires: The Long and the Short of it, ISCA2003. (arXiv2001.06598)
- [最近] A. Litteken, L. Seifert, J. Chadwick, N. Nottingham, F. Chong, J. Baker,
Qompress: Efficient Compilation for Ququarts Exploiting Partial and Mixed Radix Operations for
Communication Reduction, ASPLOS2023.

Georgia Tech.

- [初期] P. Das, C. Pattison, S. Manne, D. Carmean, K. Svore, M. Qureshi, N. Delfosse,
AFS: Accurate, Fast, and Scalable Error-Decoding for Fault-Tolerant Quantum Computers, HPCA2022.
(arXiv2001.06598)
- [最近] S. Vittal, P. Das, M. Qureshi, Astrea: Accurate Quantum Error-Decoding via Practical Minimum-
Weight Perfect-Matching, ISCA2023.

最初は「**アーキわかる（興味ある）物理研究者**」と併走
その過程で「**物理わかるアーキ研究者**」を育成

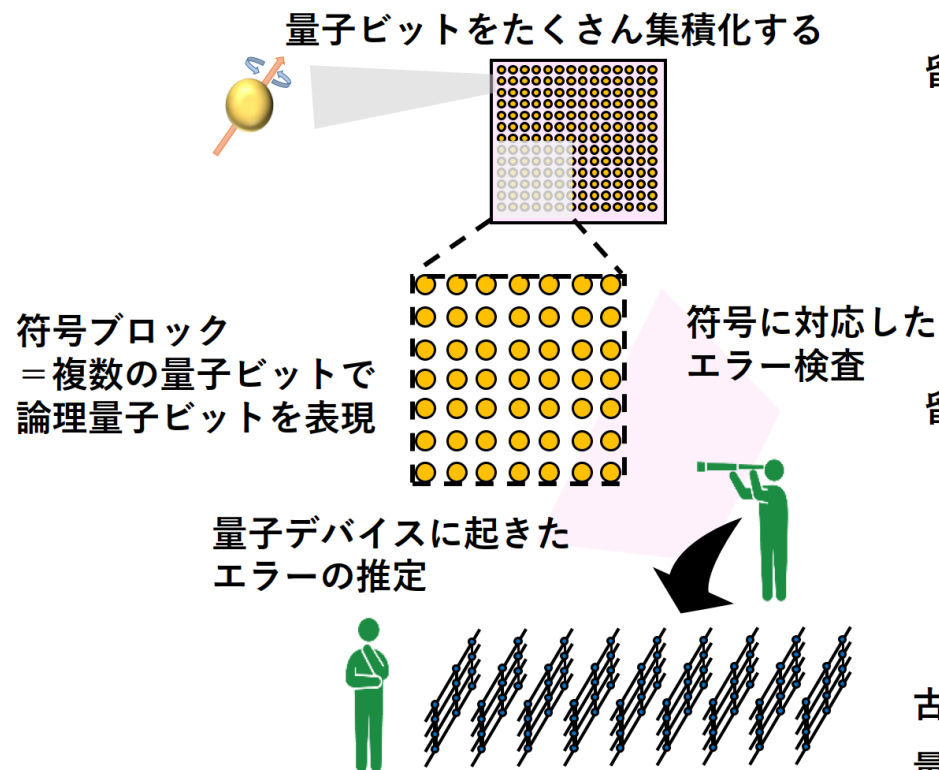
量子計算機アーキテクチャ研究動向まとめ

- 計算機アーキテクチャ
＝ 計算機の中身、全体の構成、取り巻く環境
 - 主な役割：各要素技術の統合、レイヤ分け、計算機としての展望を示す
- 計算機アーキテクチャ分野でも量子計算機はインパクト大
 - 物理（量子情報）系＋計算機系研究者の共同研究が重要
- 海外では量子計算機アーキテクチャの研究が盛ん、国内はまだこれから
- （個人的には）未来志向＝FTQCアーキにフォーカスしたい

今日の内容

- 量子計算機アーキテクチャ分野の研究動向
 - 計算機アーキテクチャの研究対象・隣接分野
 - 計算機アーキテクチャ分野における量子計算機関連の研究動向
- 量子誤り訂正とそれを支える計算機アーキテクチャ
 - 表面符号における誤り推定
 - 超伝導デジタル回路を用いた誤り推定機構（DAC2021/HPCA2022）
 - 表面符号＋格子手術を用いた誤り耐性量子計算
 - ロードストア型FTQCアーキテクチャ（HPCA2025）
- まとめ

解決策：量子誤り訂正符号により、誤りが生じても高い確率で訂正できるようにする



留意点 1 論理量子ビットの情報を見ずに
エラーの情報のみを得る符号でないといけない

➡ 通常の計算機で使えるような
効率的な符号が使えない

留意点 2 量子ビットはエラー率が非常に高いため、
とても大きな符号を使う必要がある

➡ 1つの符号を作るために
大量の量子ビットが必要

古典メモリの誤り訂正： 必要サイズは1.1倍程度に

量子メモリの誤り訂正： 必要サイズは数十倍以上に！

作成したいもの

制御装置



演算命令

測定値

エラーの無い
量子デバイス



しかし、エラーのない量子デバイスは
現実には作れない

実際に作成するもの (FTQC)

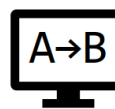
制御装置



演算命令

測定値

翻訳器

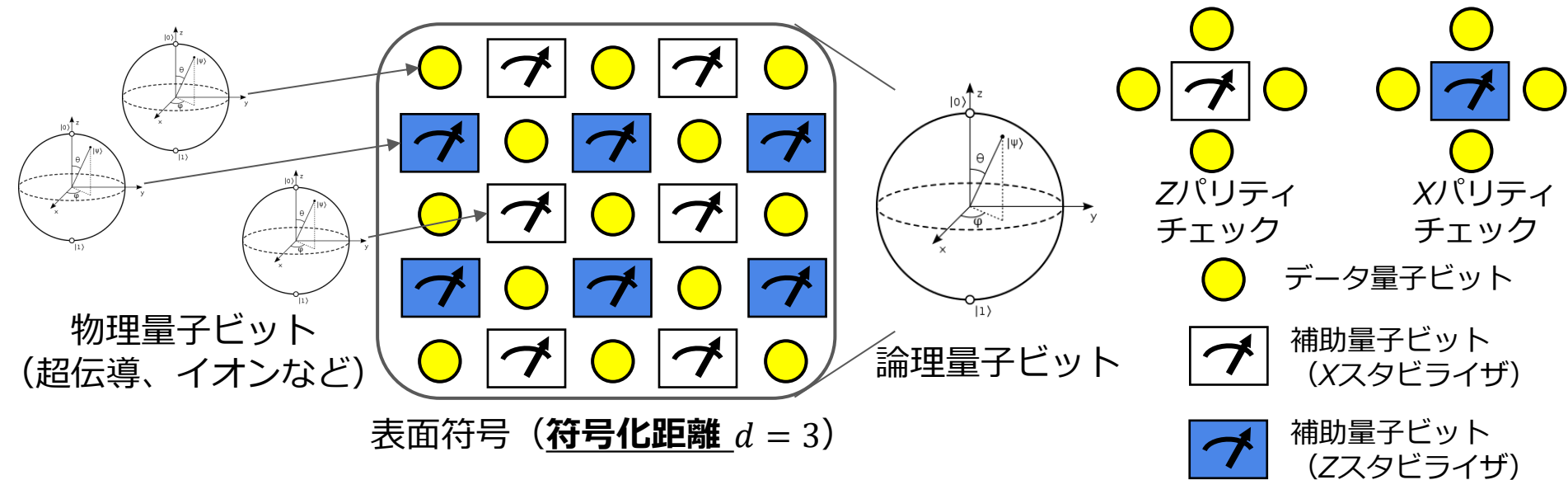


エラーのある
量子デバイス



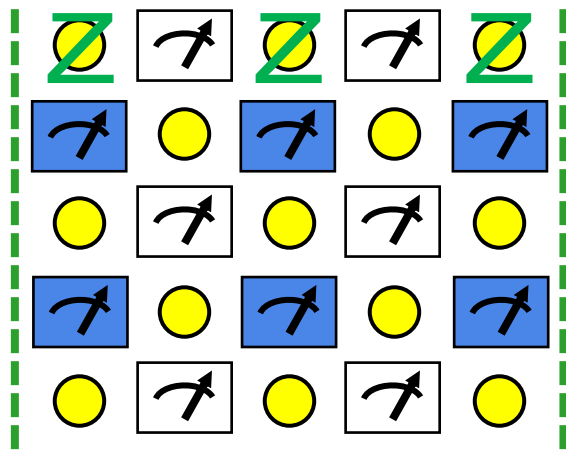
間に論理命令を物理命令に翻訳するためのインターフェイスを挟むことで、
計算機モデルとしてノイズレスな量子デバイスをシミュレートする

量子誤り訂正符号：表面符号

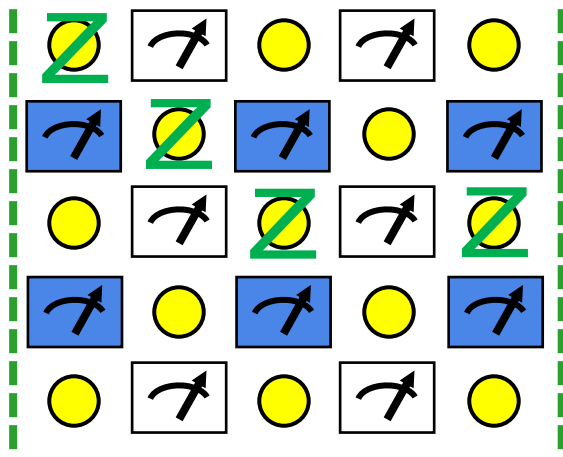


- 規則的に並んだ複数の量子ビットで論理量子ビットを表現
 - 論理ビットの状態を表すデータ量子ビットと観測用の補助量子ビット
- 補助量子ビットの観測値: XとZの2種類のエラーのうち、対応するもののパリティチェック

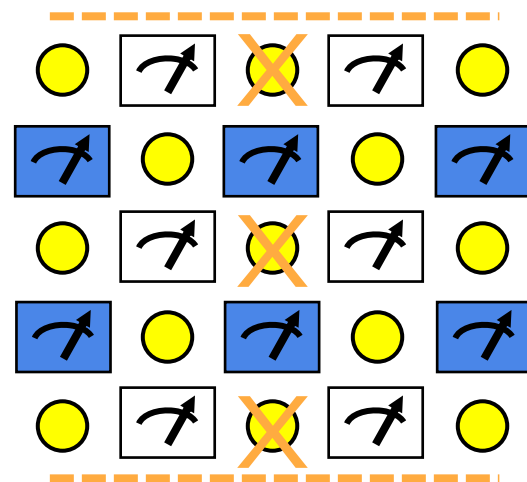
表面符号における論理Pauli演算



論理Z演算の一例



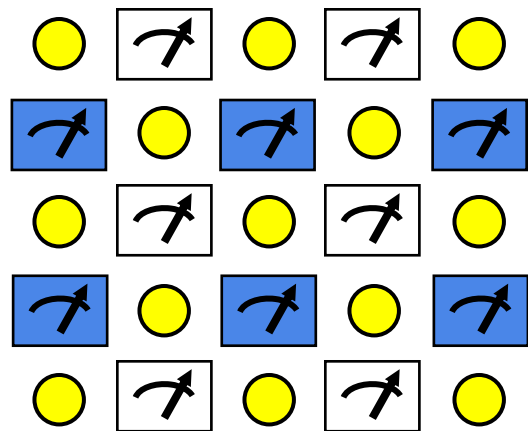
論理Z演算の別の例



論理X演算の一例

- 論理Z演算：左右の境界をつなぐようにデータ量子ビットにZ演算を作用
- 論理X演算：上下の境界をつなぐようにデータ量子ビットにX演算を作用
- 表面符号の左右、上下の境界は意味が異なる

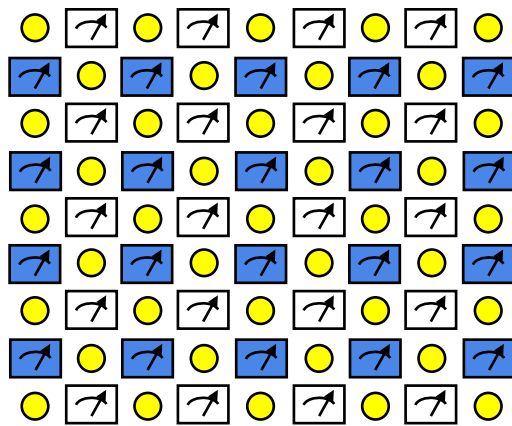
表面符号の符号距離



符号距離 $d = 3$

データビット : $3^2 + 2^2 = 13$

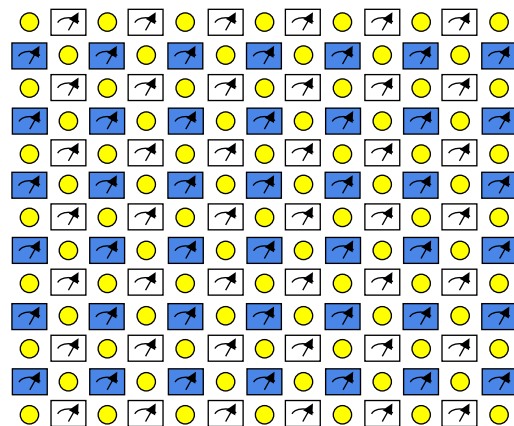
補助ビット : $2 \times 3 \times (3 - 1) = 12$



符号距離 $d = 5$

データビット : 41

補助ビット : 40



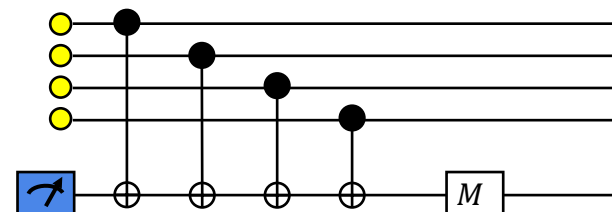
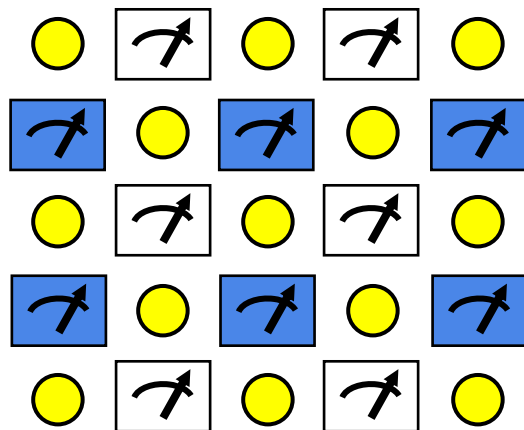
符号距離 $d = 7$

データビット : 85

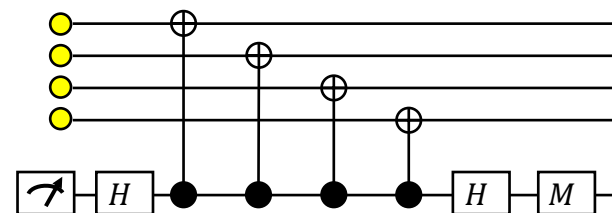
補助ビット : 84

- 符号距離 d : 論理操作をするのに触る必要があるデータ量子ビットの数
- 表面符号の構成に必要な量子ビット数は $O(d^2)$
- 1ビットエラー率 p の場合、望まない論理操作が起きる確率は $O(p^d)$
- 多項式的なオーバーヘッドで指数的な論理エラー率の低減

表面符号におけるエラーパリティチェック

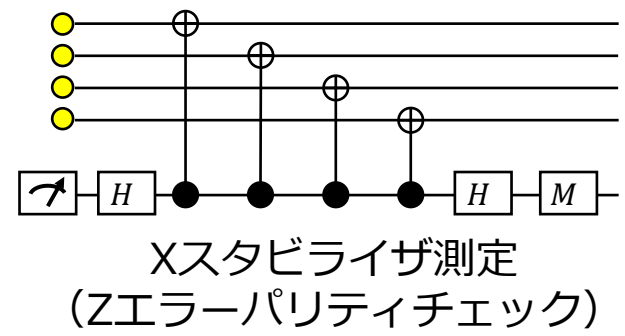
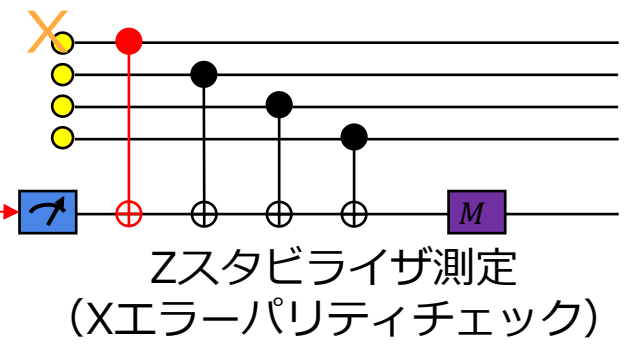
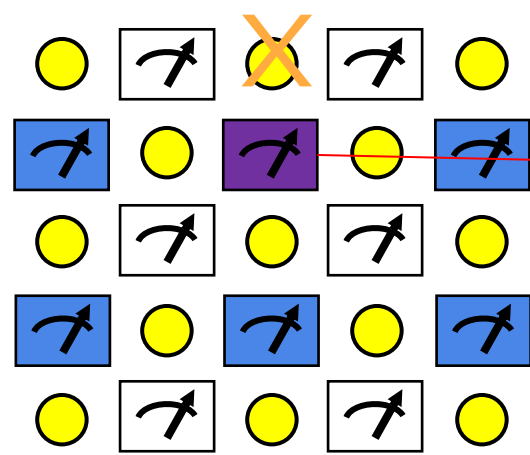


Zスタビライザ測定
(Xエラーパリティチェック)

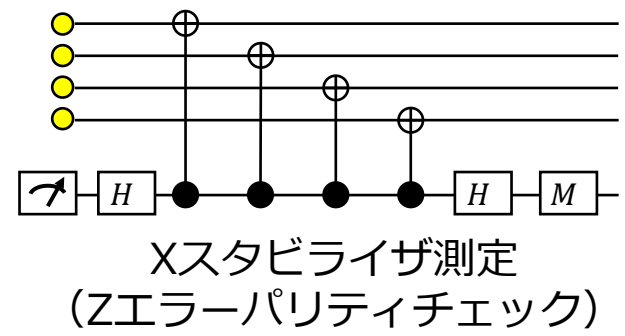
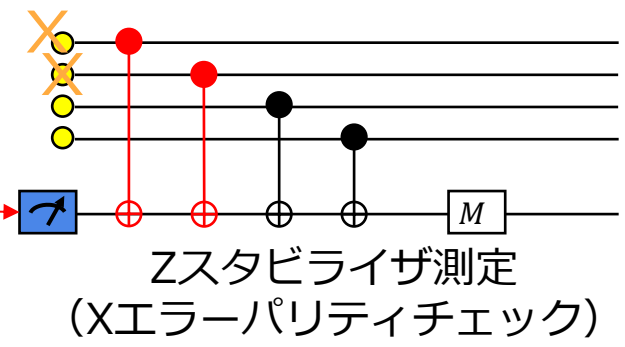
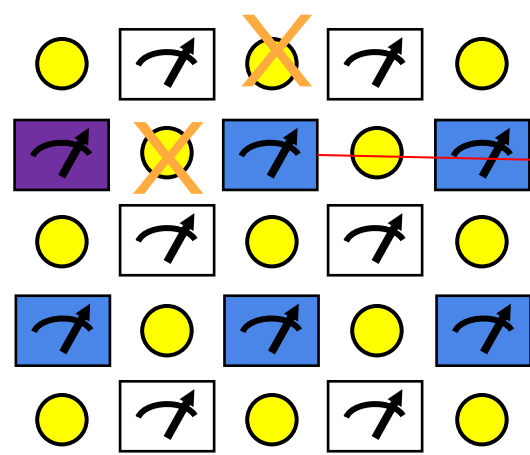


Xスタビライザ測定
(Zエラーパリティチェック)

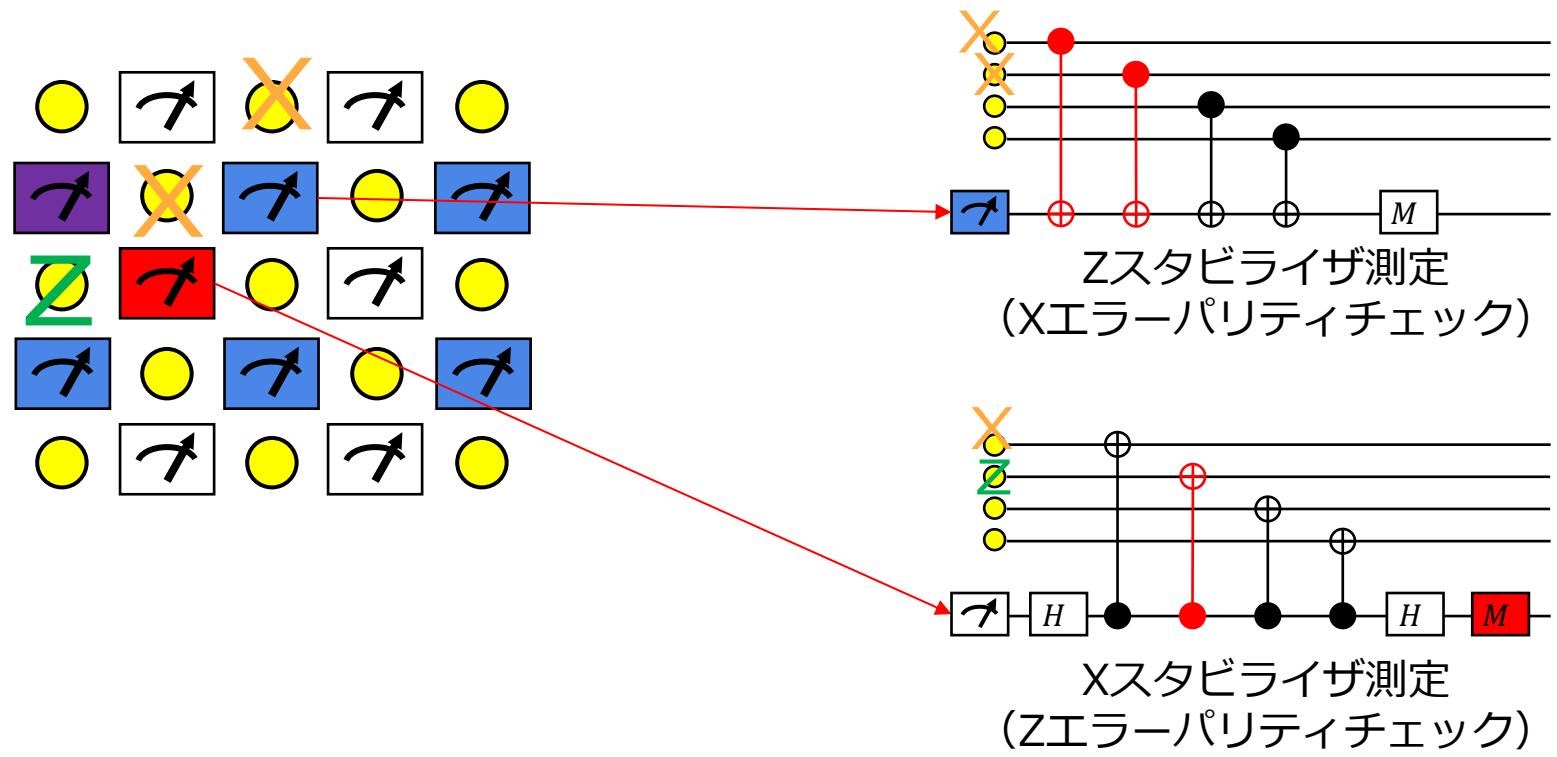
表面符号におけるエラーパリティチェック



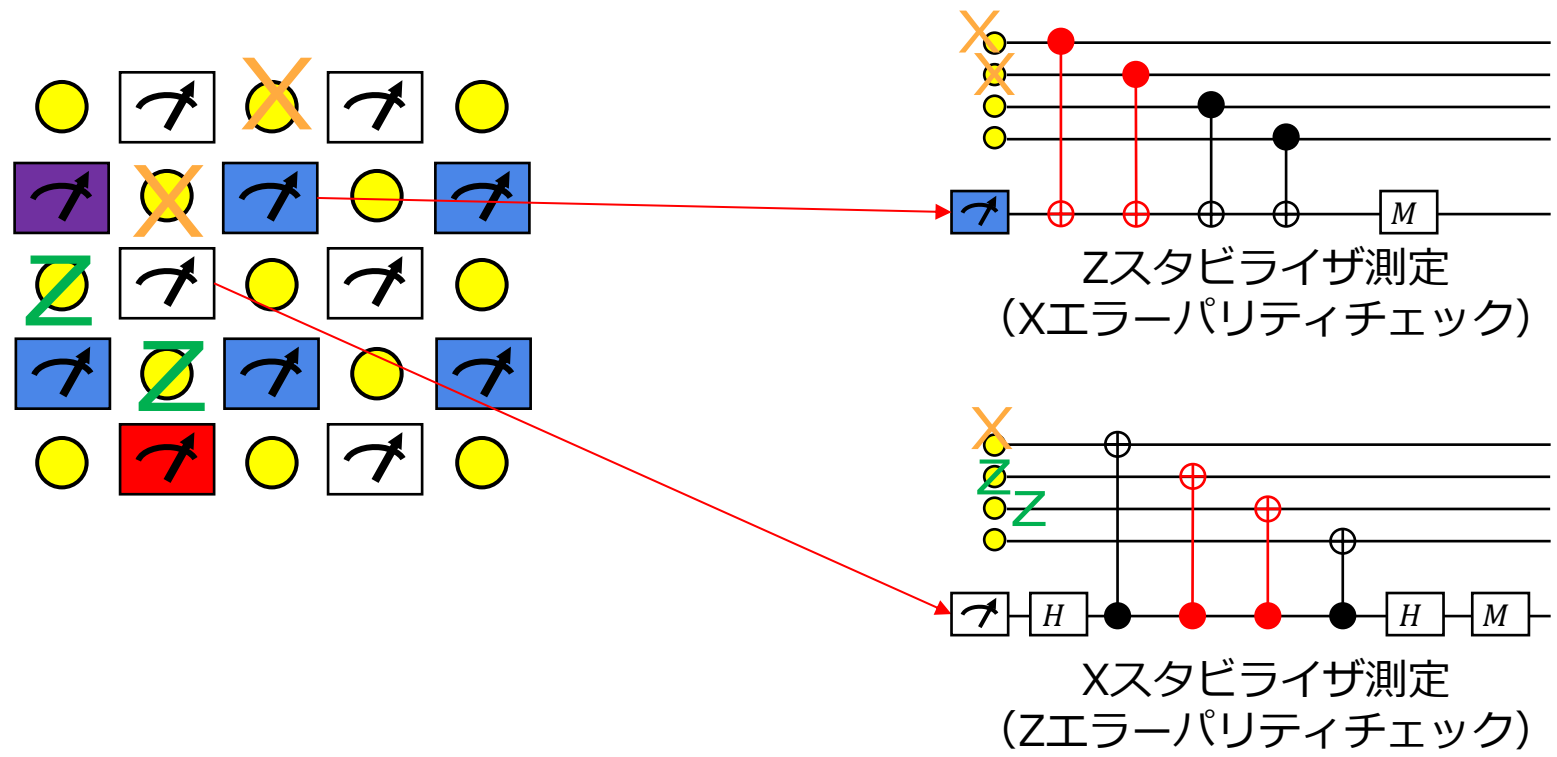
表面符号におけるエラーパリティチェック



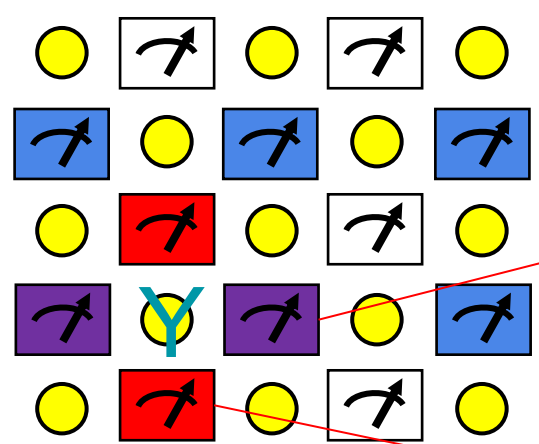
表面符号におけるエラーパリティチェック



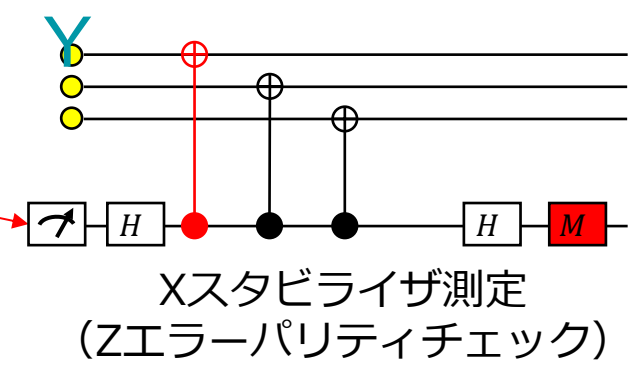
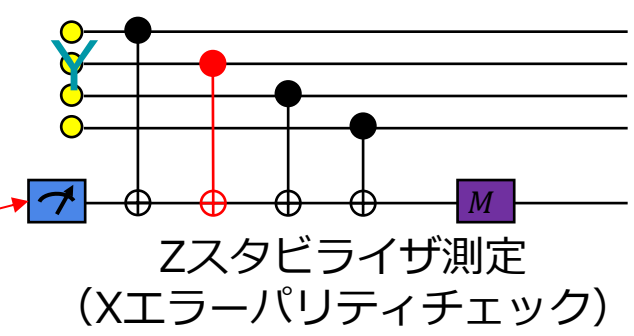
表面符号におけるエラーパリティチェック



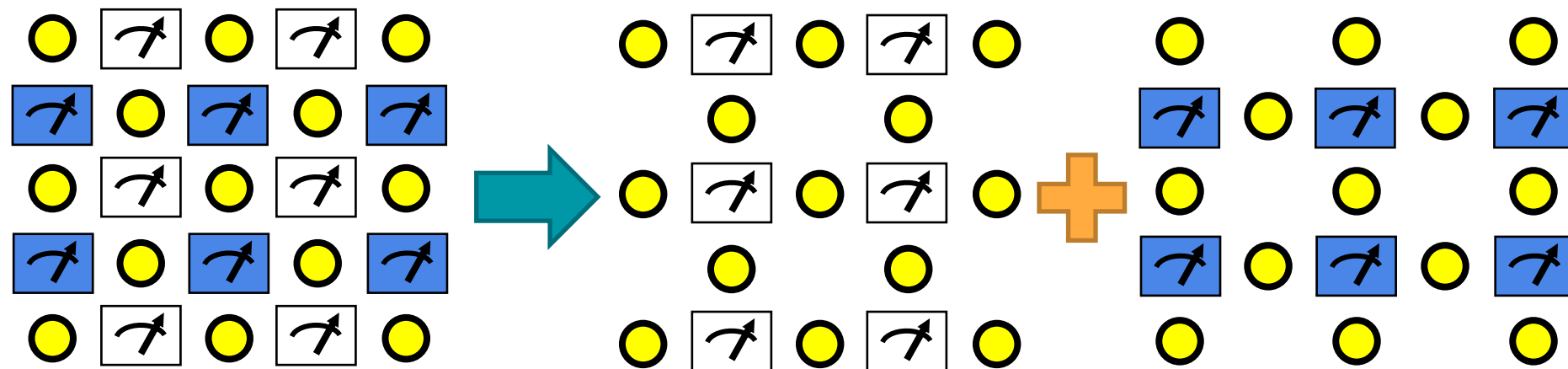
表面符号におけるエラーパリティチェック



$Y = iXZ$ なので
 X と Z が同時に生じているとみなせる

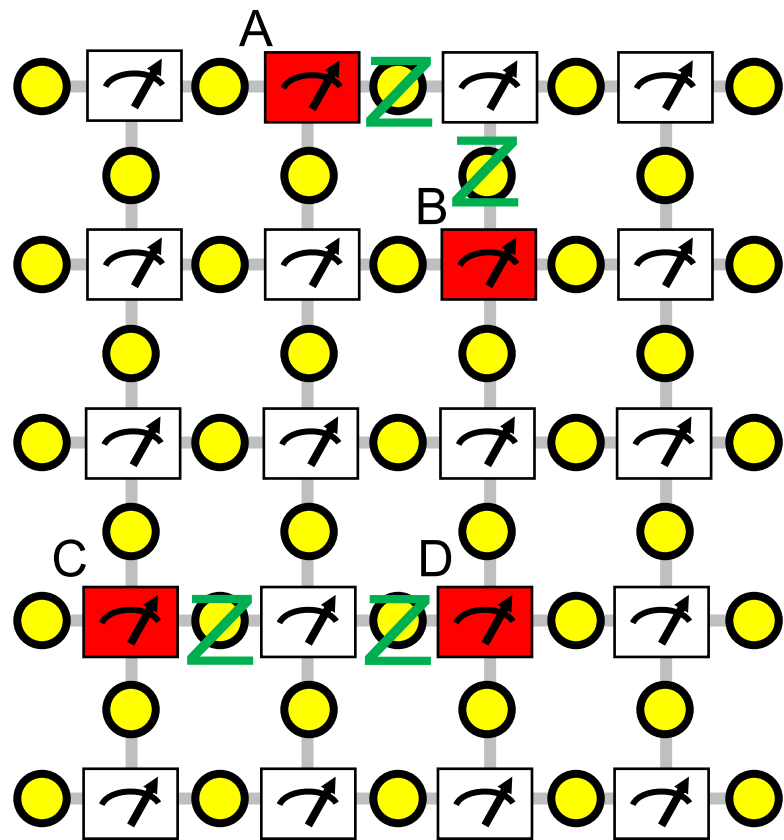


表面符号におけるエラー推定（復号）



- XとZのそれぞれのエラーを独立に推定できる
 - マッチング問題に帰着する場合はXとZの格子を独立にエラー推定
- XとZに相関がある場合、その情報を使えない

表面符号におけるエラー推定（復号）

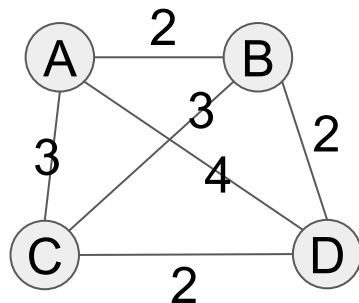


仮定

- XとZのエラーは独立に推定できる
- なるべく短いエラー鎖が生じる



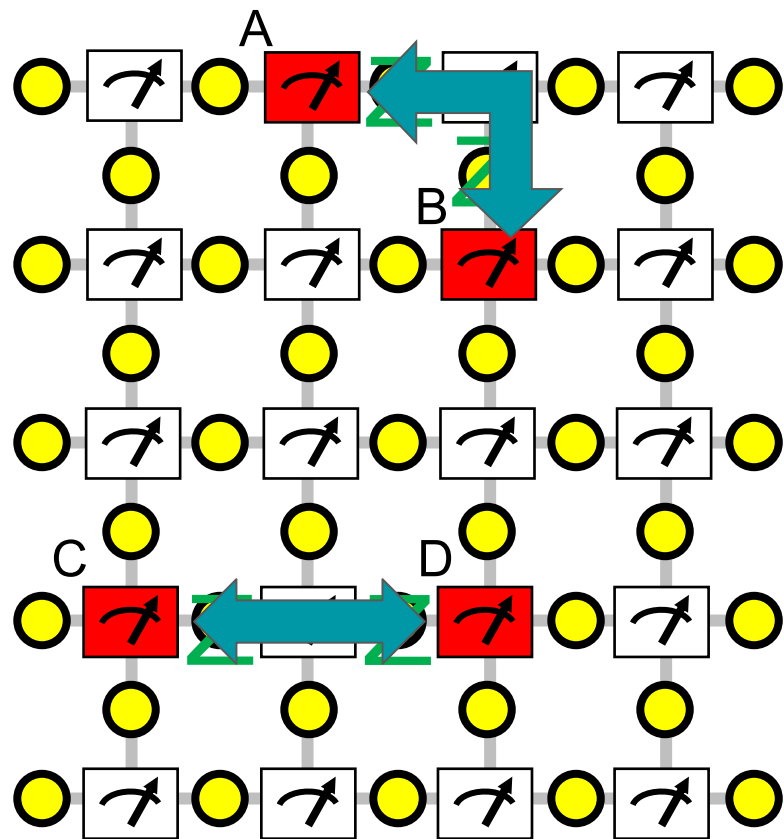
Minimum Weight Perfect Matching (MWPM)



V : Hot syndromes
 W_e : Manhattan distance

Exact solution: **Blossom algorithm ($O(n^3)$)**

表面符号におけるエラー推定（復号）

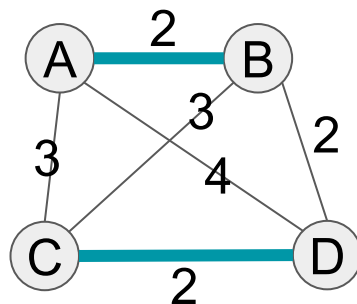


仮定

- XとZのエラーは独立に推定できる
- なるべく短いエラー鎖が生じる



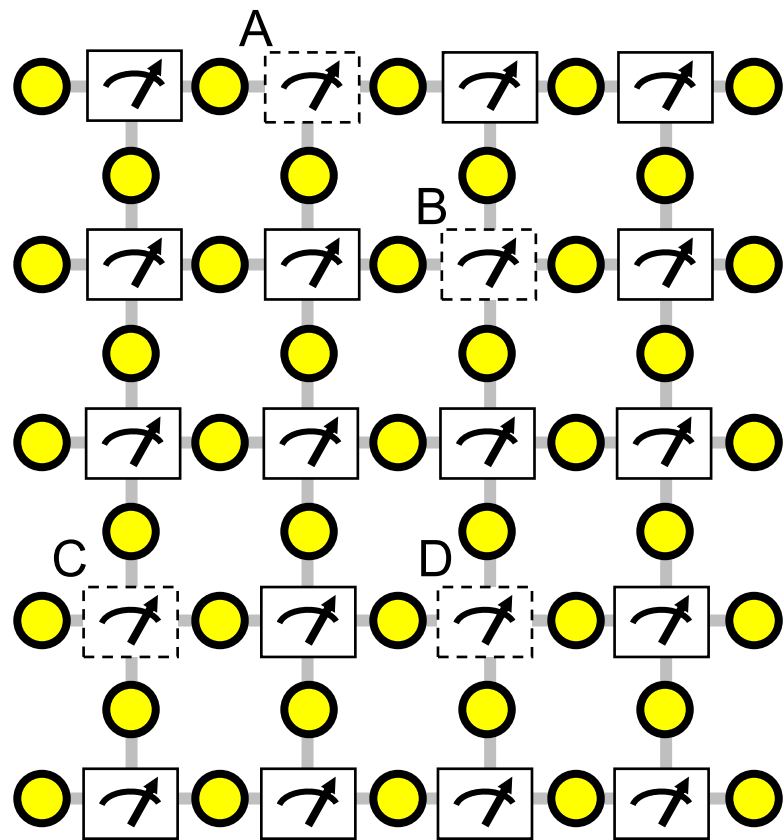
Minimum Weight Perfect Matching (MWPM)



V : Hot syndromes
 W_e : Manhattan distance

Exact solution: **Blossom algorithm ($O(n^3)$)**

表面符号におけるエラー推定（復号）

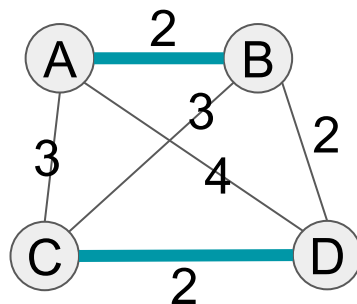


仮定

- XとZのエラーは独立に推定できる
- なるべく短いエラー鎖が生じる



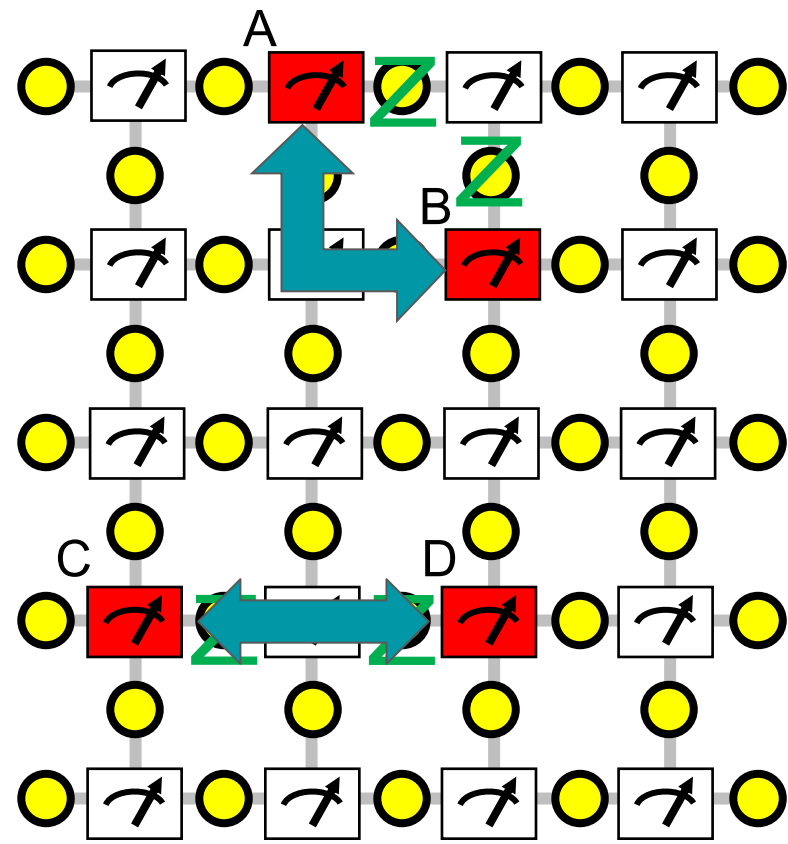
Minimum Weight Perfect Matching
(MWPM)



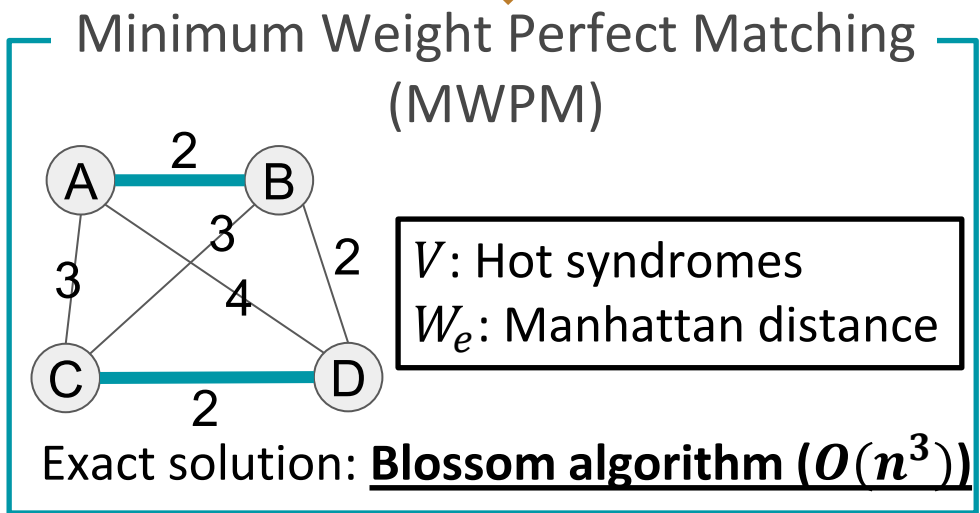
V : Hot syndromes
 W_e : Manhattan distance

Exact solution: **Blossom algorithm ($O(n^3)$)**

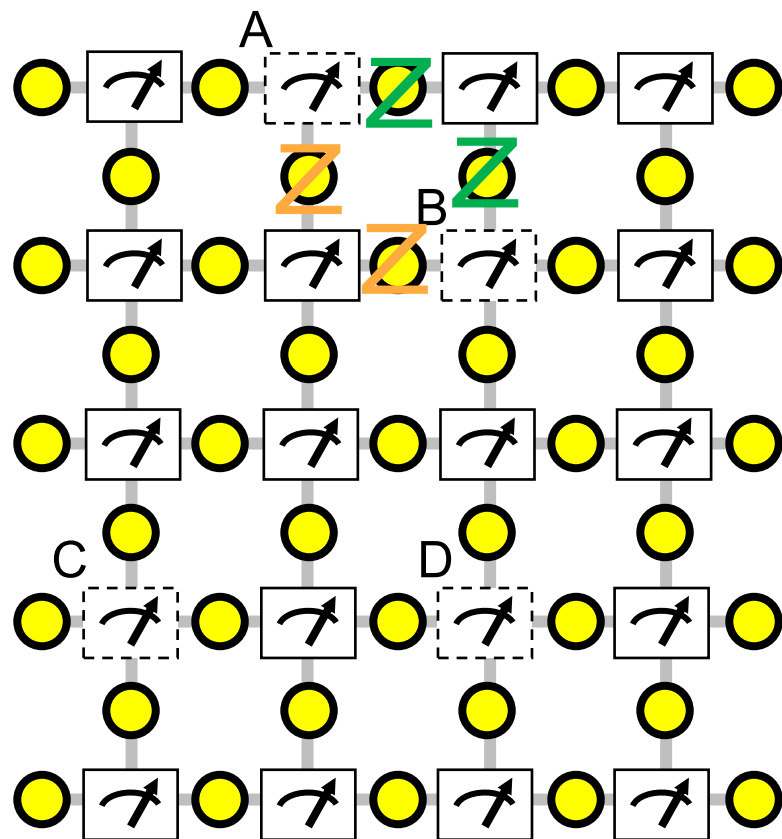
異なるつなぎ方をした場合



- 仮定
- XとZのエラーは独立に推定できる
 - なるべく短いエラー鎖が生じる

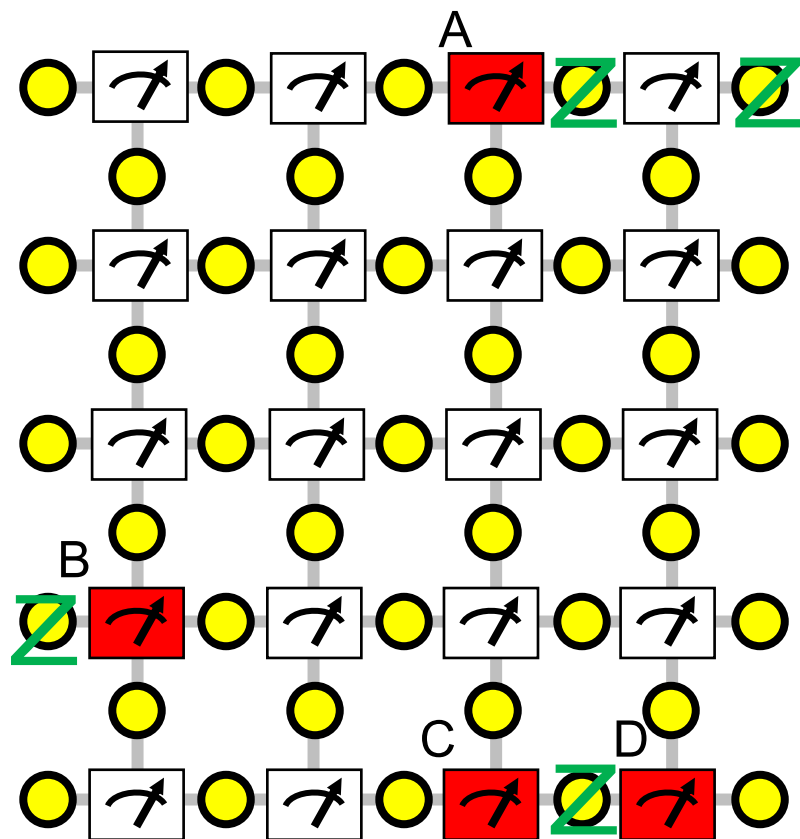


異なるつなぎ方をした場合



- エラーと訂正の合計が closed loop を作る場合も推定成功
 - (Topologically) trivial chain と呼ばれる
 - 論理 I 操作になる
 - = 論理状態に影響を与えない
 - = 論理エラーにならない
- 訂正の成否は奇パリティの ペア決めのみに依存 してつなぎ方に依らない

境界につながるエラー

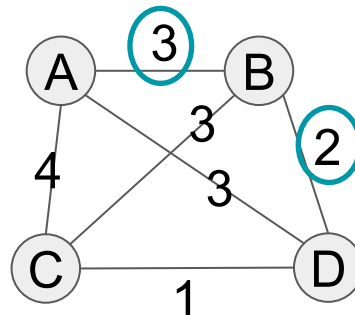


- グラフのエッジ重みを工夫することで境界につながるエラーもマッチング問題により推定できる

- 「ノード間のマンハッタン距離」と「それぞれのノードの最寄りの境界からの距離の和」のうち小さい方を重みとする

マンハッタン距離 : 5

境界との距離の和 : $1+2=3$



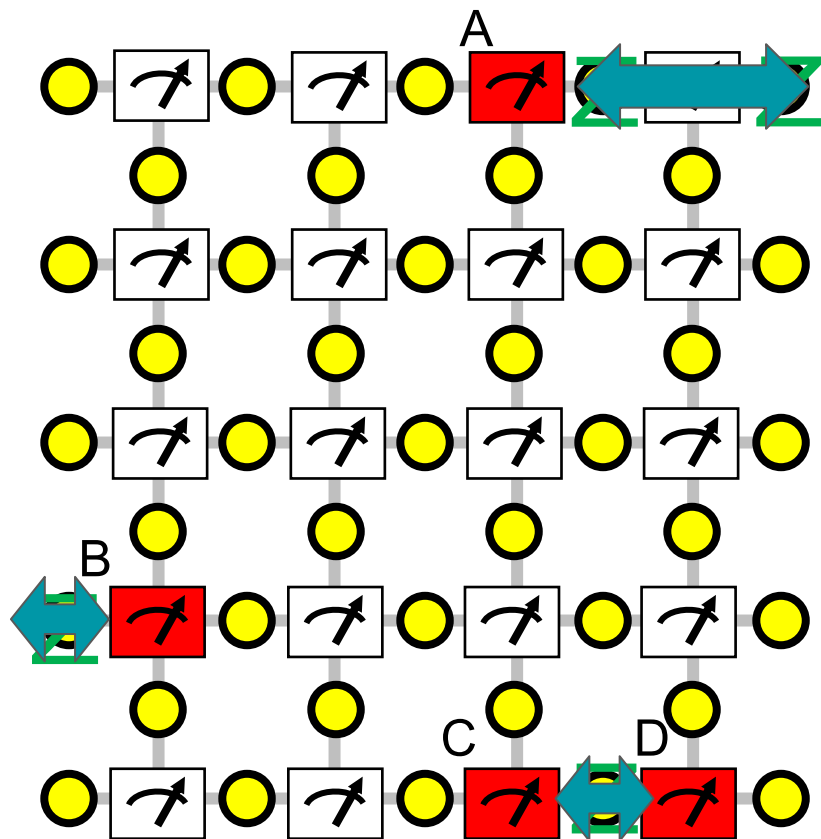
マンハッタン距離 : 4

境界との距離の和 : $1+1=2$

マンハッタン距離 : 1

境界との距離の和 : $1+2=3$

境界につながるエラー

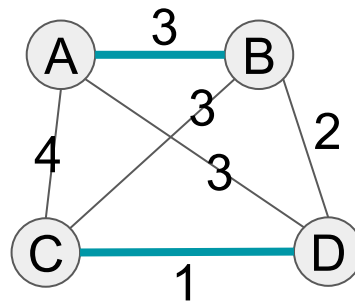


- グラフのエッジ重みを工夫することで境界につながるエラーもマッチング問題により推定できる

- 「ノード間のマンハッタン距離」と「それぞれのノードの最寄りの境界からの距離の和」のうち小さい方を重みとする

マンハッタン距離 : 5

境界との距離の和 : $1+2=3$



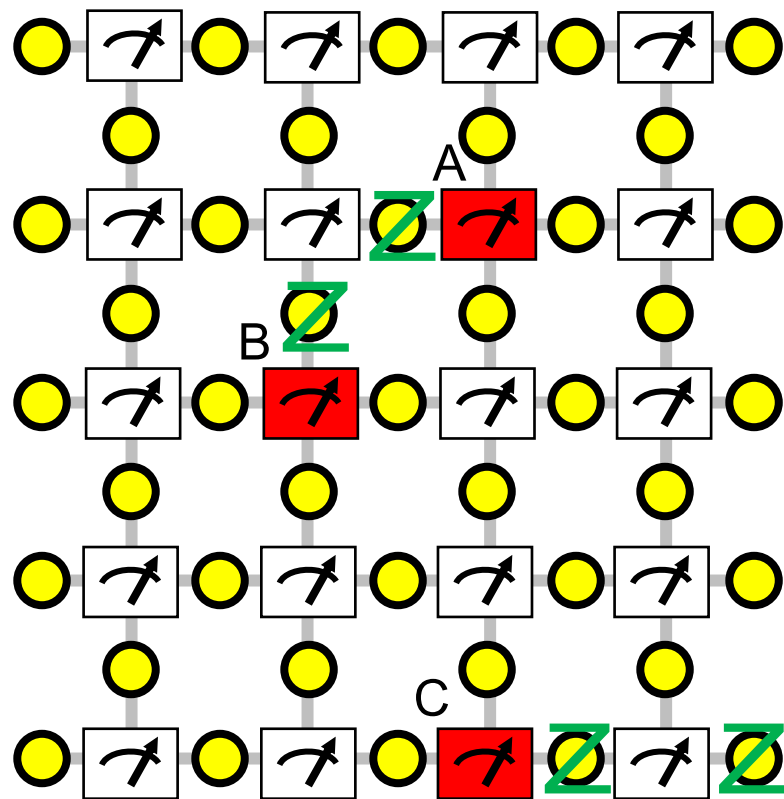
マンハッタン距離 : 4

境界との距離の和 : $1+1=2$

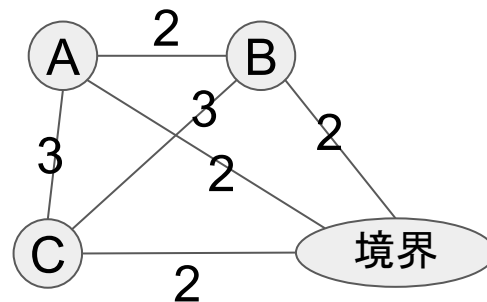
マンハッタン距離 : 1

境界との距離の和 : $1+2=3$

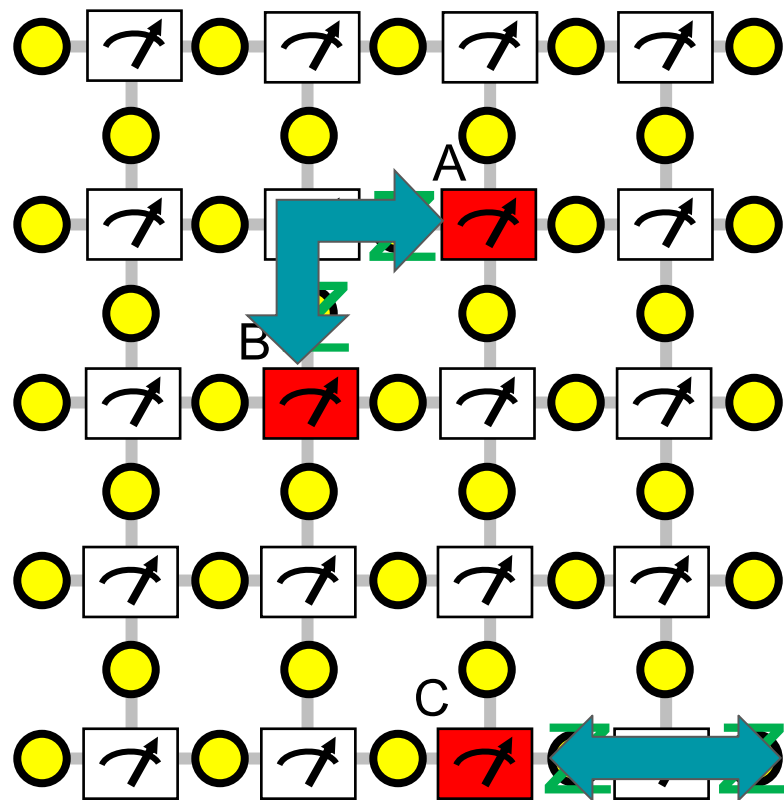
頂点が奇数の場合



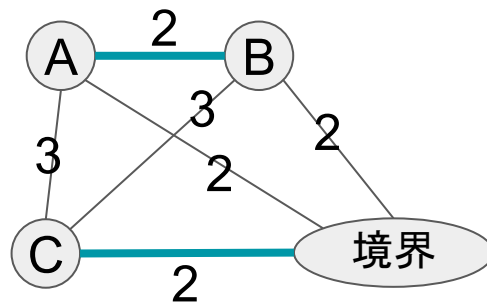
- 奇数ノードの完全グラフには完全マッチングが存在しない
- 「境界を表す仮想的なノード」を追加して偶数ノードの完全グラフをつくる
 - 境界ノードとのエッジの重みは最寄りの境界への距離



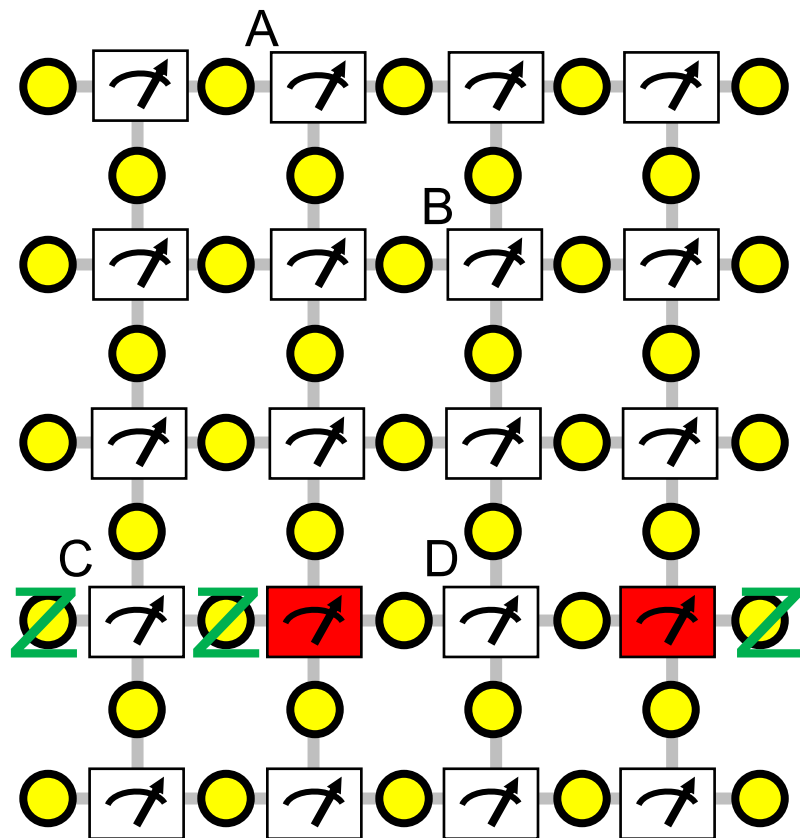
頂点が奇数の場合



- 奇数ノードの完全グラフには完全マッチングが存在しない
- 「境界を表す仮想的なノード」を追加して偶数ノードの完全グラフをつくる
 - 境界ノードとのエッジの重みは最寄りの境界への距離

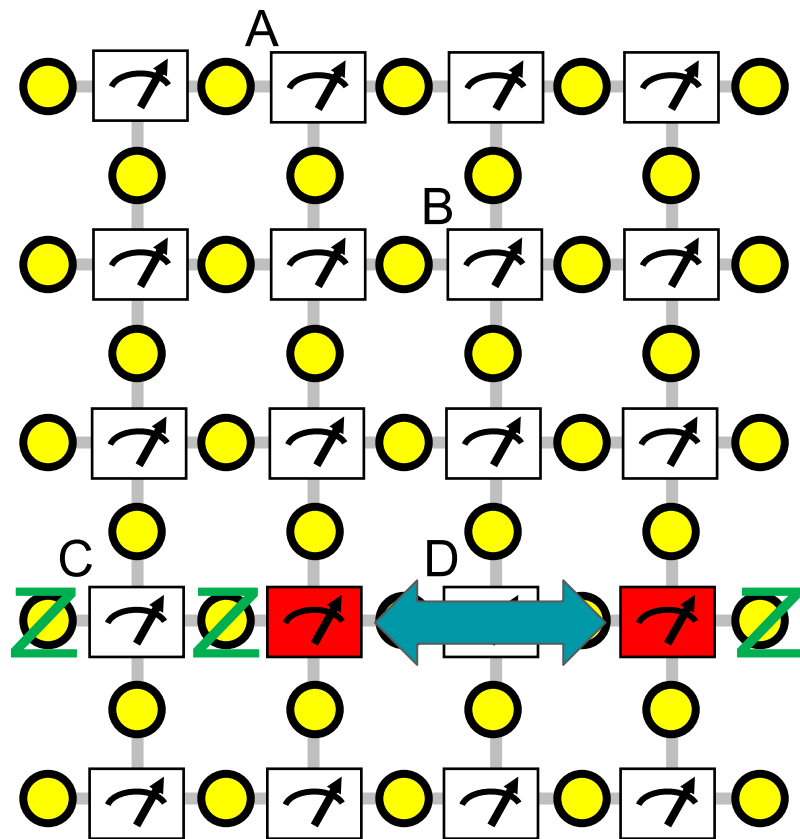


論理エラーが生じる場合



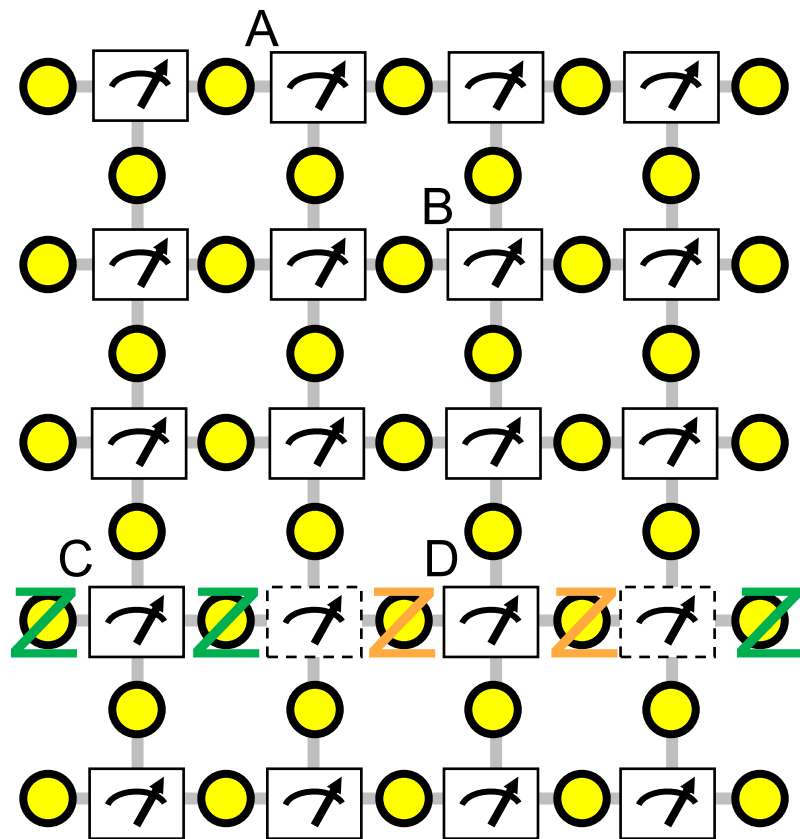
- エラーと訂正の合計がある境界から別の境界へとつながるchainを作る
 - Non-trivial chain
 - 論理Z操作になる
 - = 望まない論理ビットの状態変化
 - = 論理エラー
- 最低で符号距離 d の半分以上のエラーで推定に失敗する

論理エラーが生じる場合



- エラーと訂正の合計がある境界から別の境界へとつながるchainを作る
 - Non-trivial chain
 - 論理Z操作になる
 - = 望まない論理ビットの状態変化
 - = 論理エラー
- 最低で符号距離 d の半分以上のエラーで推定に失敗する

論理エラーが生じる場合



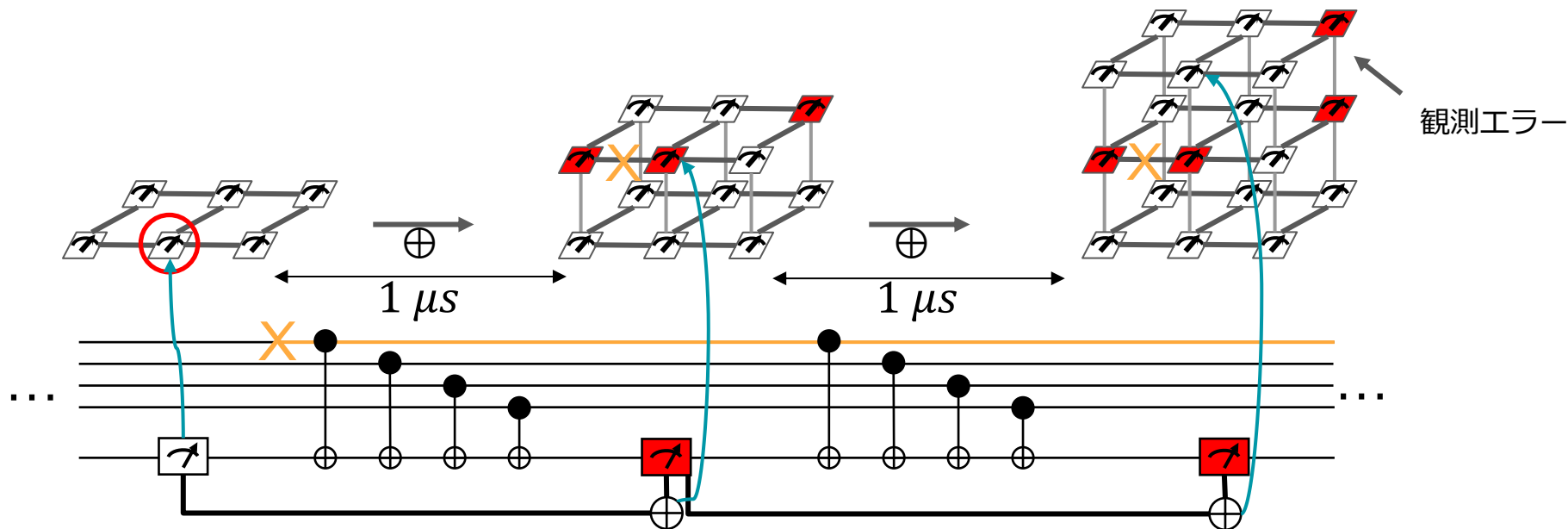
- エラーと訂正の合計がある境界から別の境界へとつながるchainを作る
 - Non-trivial chain
 - 論理Z操作になる
 - = 望まない論理ビットの状態変化
 - = 論理エラー
- 最低で符号距離 d の半分以上のエラーで推定に失敗する

補助量子ビットの観測エラーへの対応

1回目の観測

2回目の観測

3回目の観測



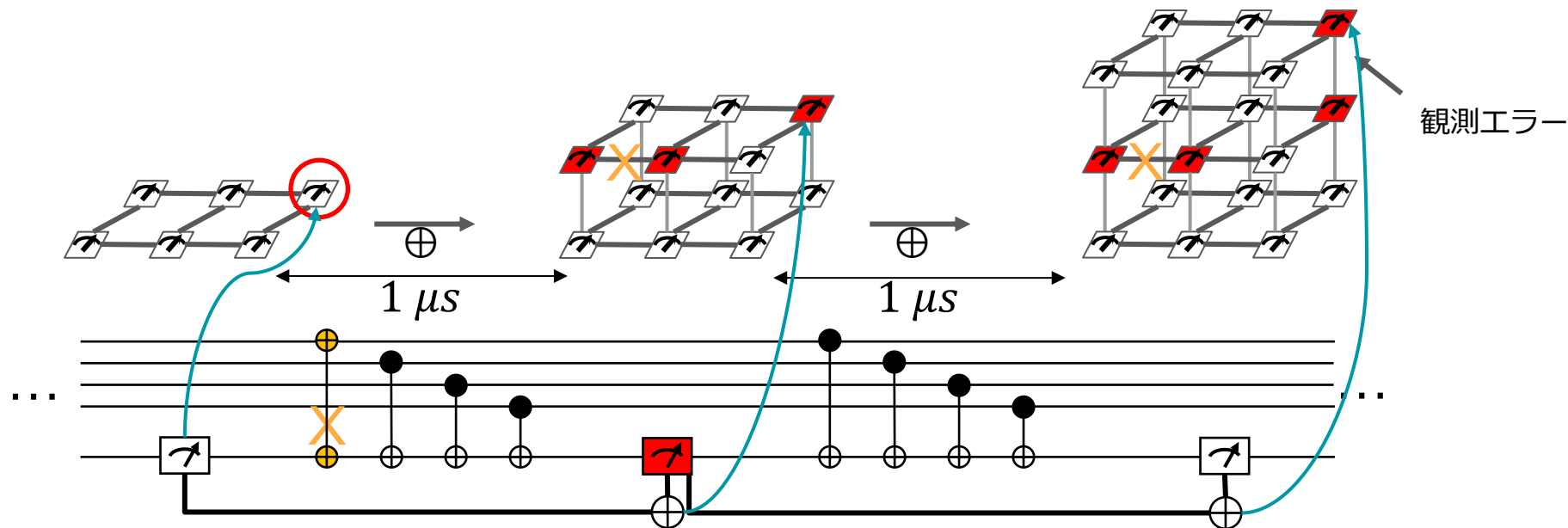
- データ量子ビットのエラーは訂正するまで残る、観測エラーは一時的
- 複数回観測をおこない観測値を垂直にスタックする
 - 前時刻の観測値とXORをとることで、観測値の変化した時点を表す
- エラー推定 = 3次元格子上でグラフマッチング問題を解く

補助量子ビットの観測エラーへの対応

1回目の観測

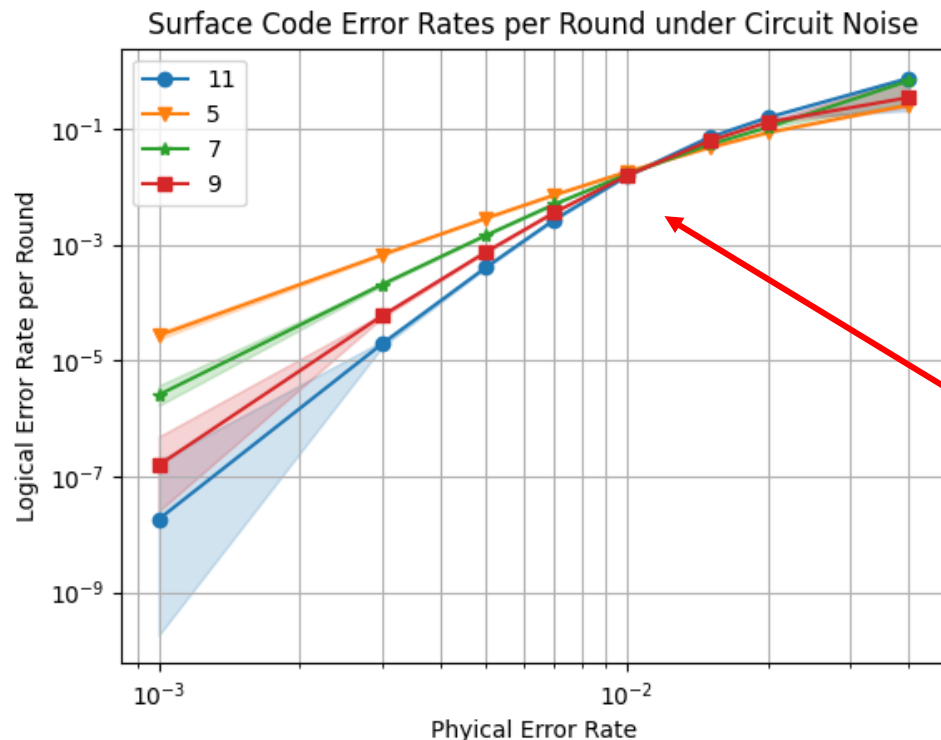
2回目の観測

3回目の観測



- データ量子ビットのエラーは訂正するまで残る、観測エラーは一時的
- 複数回観測をおこない観測値を垂直にスタックする
 - 前時刻の観測値とXORをとることで、観測値の変化した時点を表す
- エラー推定 = 3次元格子上でグラフマッチング問題を解く

表面符号の性能



しきい値
1%程度

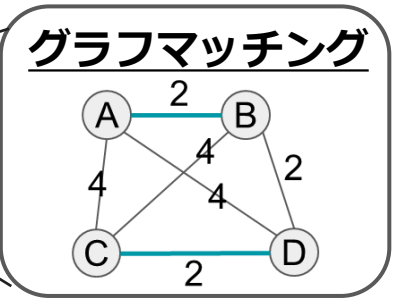
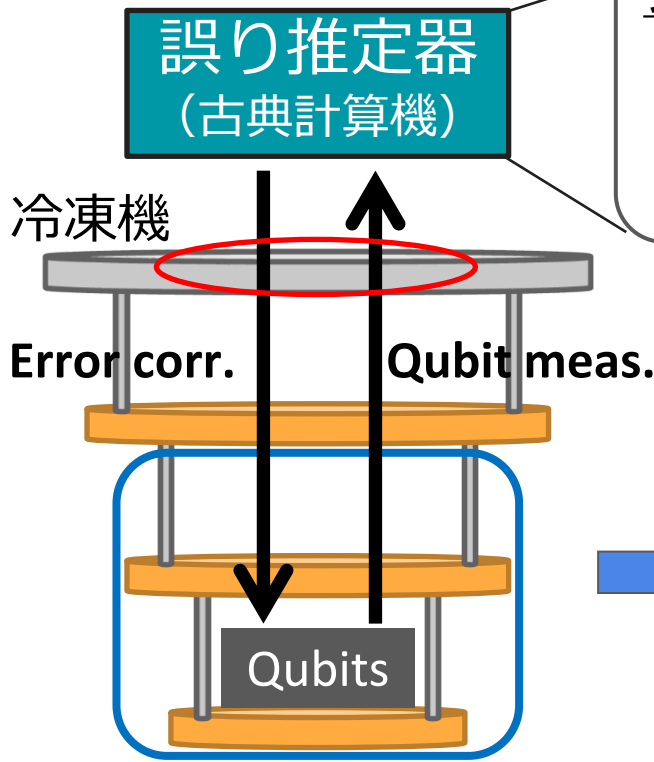
- しきい値: 物理エラー率がこの値以下であれば、符号距離 d を増やせば指数的に論理エラー率を下げられる

今日の内容

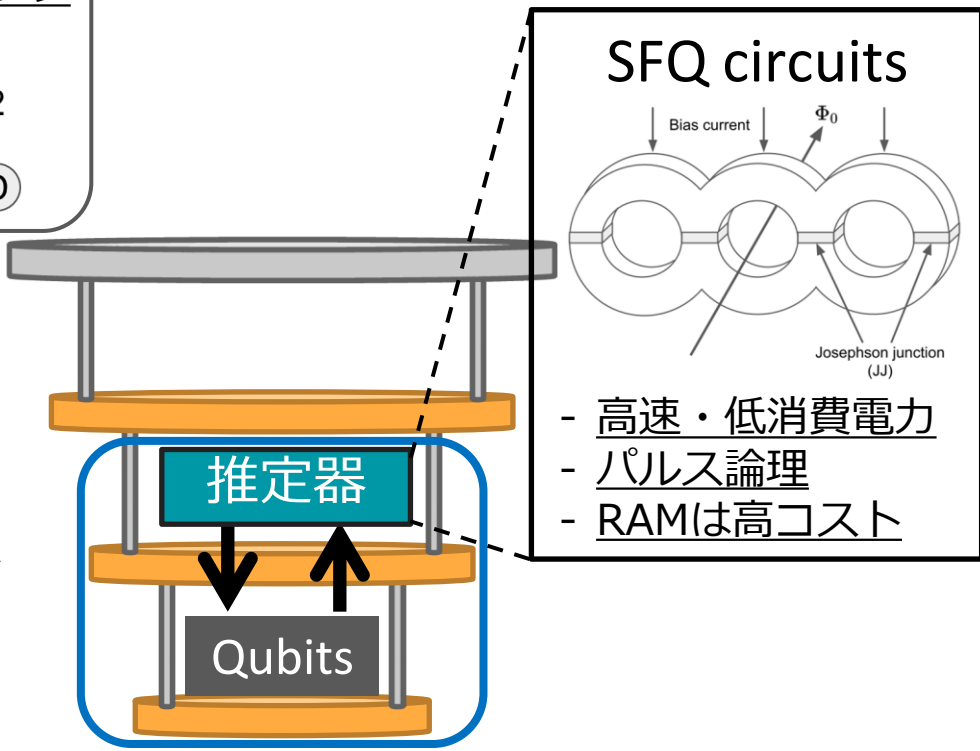
- 量子計算機アーキテクチャ分野の研究動向
 - 計算機アーキテクチャの研究対象・隣接分野
 - 計算機アーキテクチャ分野における量子計算機関連の研究動向
- 量子誤り訂正とそれを支える計算機アーキテクチャ
 - 表面符号における誤り推定
 - 超伝導デジタル回路を用いた誤り推定機構（DAC2021/HPCA2022）
 - 表面符号＋格子手術を用いた誤り耐性量子計算
 - ロードストア型FTQCアーキテクチャ（HPCA2025）
- まとめ

超伝導量子ビット+極低温環境で動作する誤り推定器

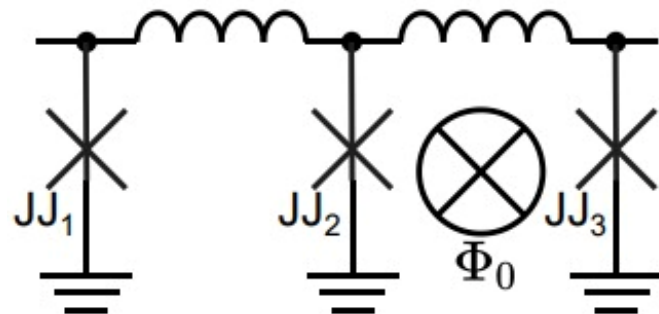
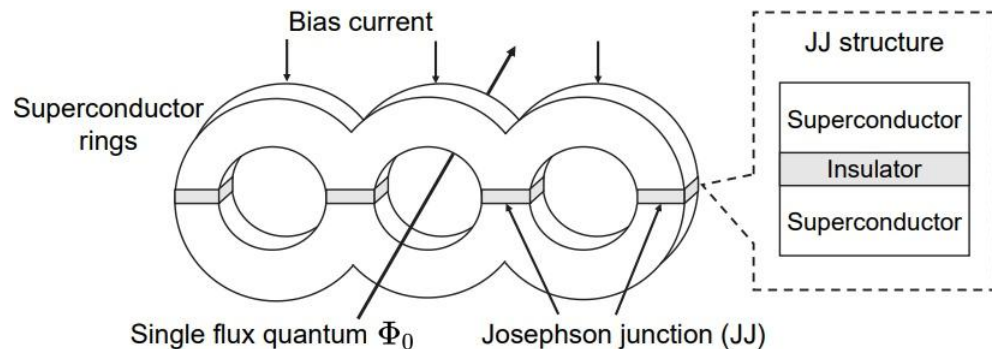
従来アーキテクチャ



提案アーキテクチャ

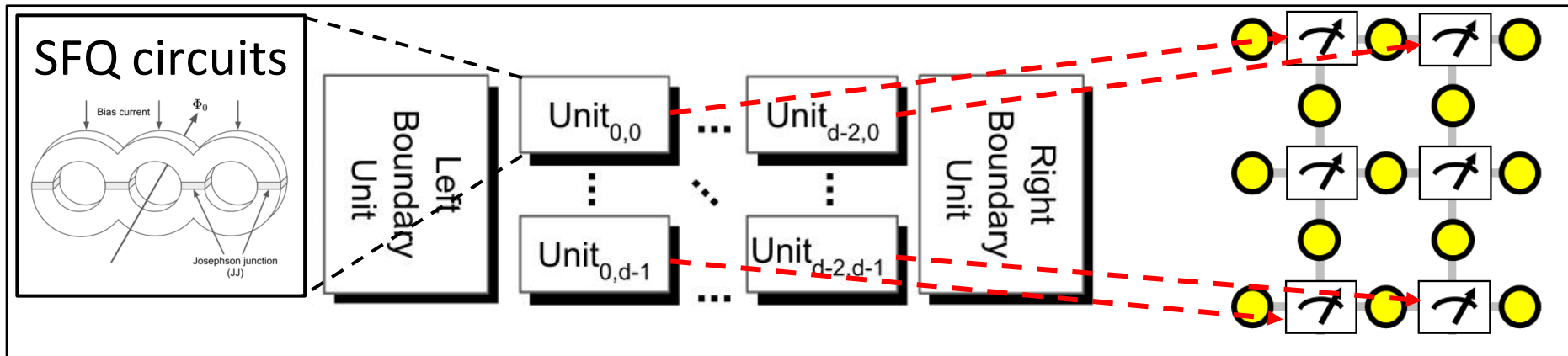


超伝導古典回路によるエラー推定



- 単一磁束量子 (SFQ: Single Flux Quantum) 回路
- 超伝導リング内の磁束量子の有無で0 or 1を表現
- 4K程度の極低温環境でのみ動作
- CMOSに比べて高速・低消費電力
- 大規模なメモリの構築は難しい
 - Blossom AlgorithmをSFQで実行するのは現実的でない

提案手法: QECCOOL

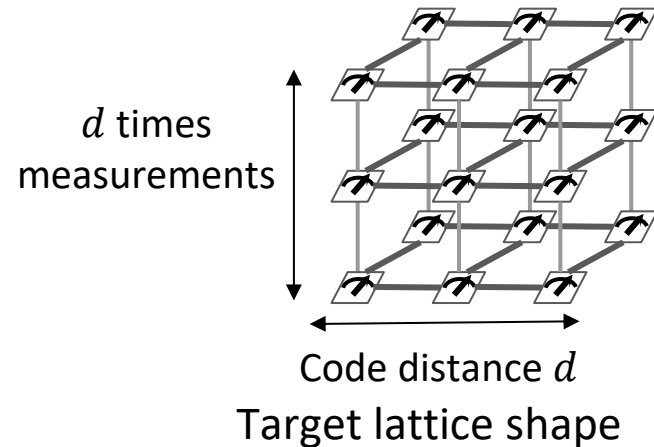
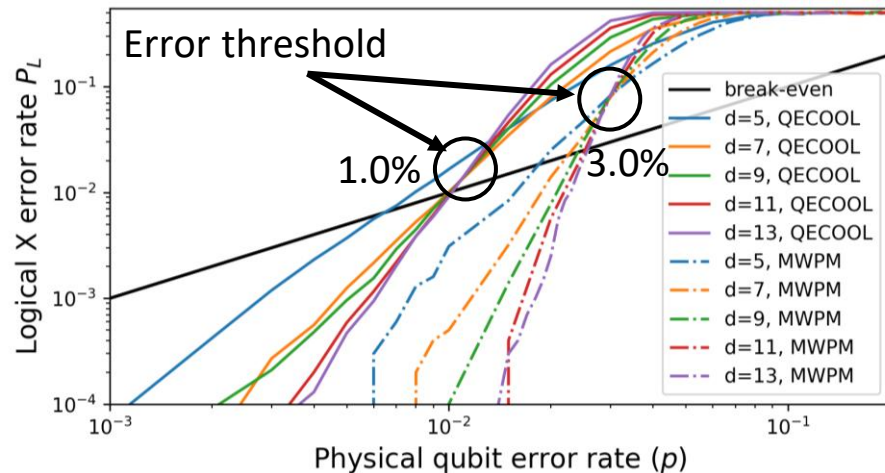


QECCOOLのアーキテクチャ

- Quantum Error COrrrection by On-Line decoding algorithm
- 大規模なRAMを必要としない**分散型のアーキテクチャ**
 - 補助量子ビットに1対1に対応する**Unit**を導入
 - Unit同士の3種類の信号伝播によりマッチング問題を解く

Y. Ueno, M. Kondo, M. Tanaka, Y. Suzuki, Y. Tabuchi, "QECCOOL: On-Line Quantum Error Correction with a Superconducting Decoder for Surface Code", 58th IEEE/ACM Design Automation Conference. (DAC 2021)

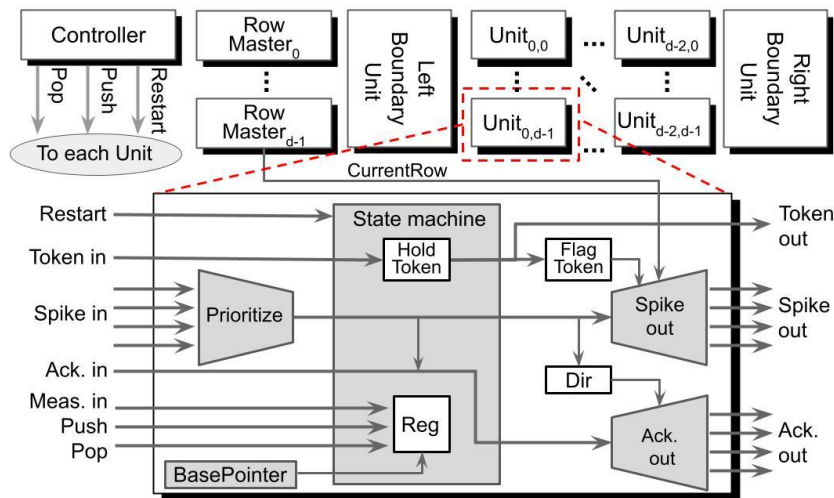
QECoolの誤り推定性能



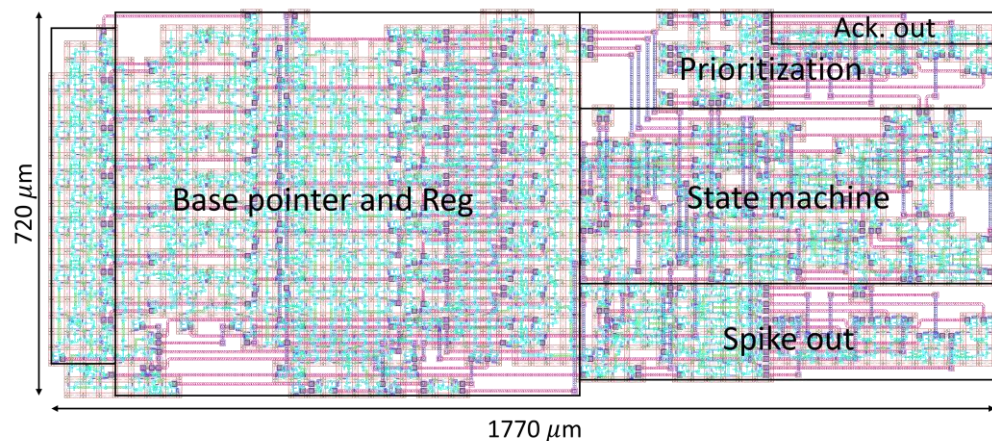
Experimental condition

- Measurement process is performed **once every $1 \mu\text{s}$**
- Each QECool Unit has a **7-bit** buffer to store syndrome values
- If buffer entry size is greater than $K = 3$, QECool is performed; otherwise, each Unit waits for measurement process
- MWPM operates with batch-QEC manner
- しきい値: QECool $p = 0.01$, MWPM $p = 0.03$

QECool誤り推定器のSFQ回路による実装



Architecture overview of QECool



SFQ design layout of QECool Unit

JJs: 3177

Area: 1.274 mm²

Latency: 215 ps

Power cons.: **2.78 μW**

Decoder power consumption per one logical qubits

Suppose $d = 9$,

$$9 \times 8 \times 2 \times 2.78_{[\mu W]} = \boxed{400 \mu W}$$

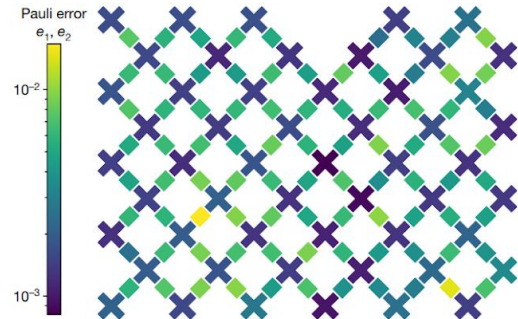
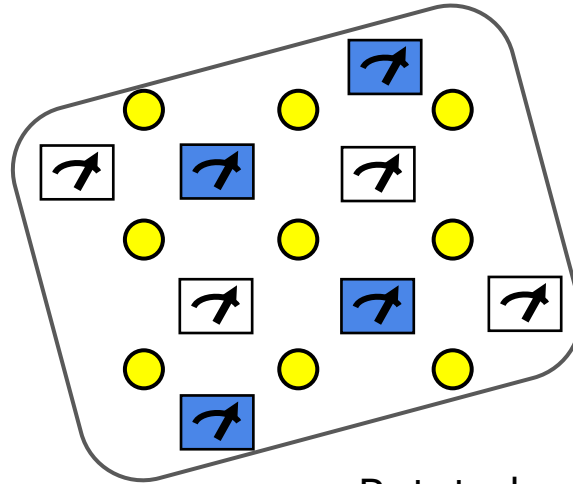
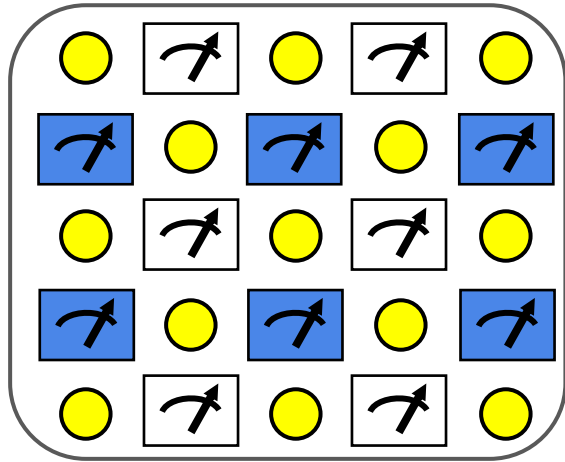
QECoolまとめ

- 極低温環境での誤り推定はスケーラブルな超伝導FTQCの実現のために必須
- SFQ回路で実装したQECool誤り推定器はレイテンシ制約を満たしつつ極低温環境で動作する
- 1論理ビット ($d = 9$) あたりの誤り推定器の消費電力は $400 \mu\text{W}$
 - 4K環境の許容消費電力を1Wとすると、2500論理ビットの誤り推定を冷凍機内で行える

今日の内容

- 量子計算機アーキテクチャ分野の研究動向
 - 計算機アーキテクチャの研究対象・隣接分野
 - 計算機アーキテクチャ分野における量子計算機関連の研究動向
- 量子誤り訂正とそれを支える計算機アーキテクチャ
 - 表面符号における誤り推定
 - 超伝導デジタル回路を用いた誤り推定機構（DAC2021/HPCA2022）
 - 表面符号＋格子手術を用いた誤り耐性量子計算
 - ロードストア型FTQCアーキテクチャ（HPCA2025）
- まとめ

補足: Rotated surface code



Cf. Google Sycamoreの配置

通常の表面符号 ($d = 3$)

データ量子ビット: $d^2 + (d - 1)^2$

補助量子ビット: $d(d - 1) \times 2$

符号パラメータ: $[d^2 + (d - 1)^2, 1, d]$

Rotated surface code ($d = 3$)

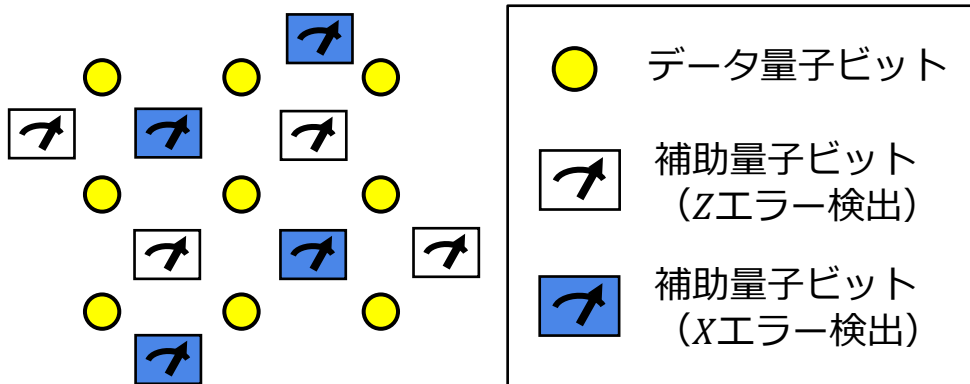
データ量子ビット: d^2

補助量子ビット: $(\frac{(d-1)^2}{2} + d - 1) \times 2$

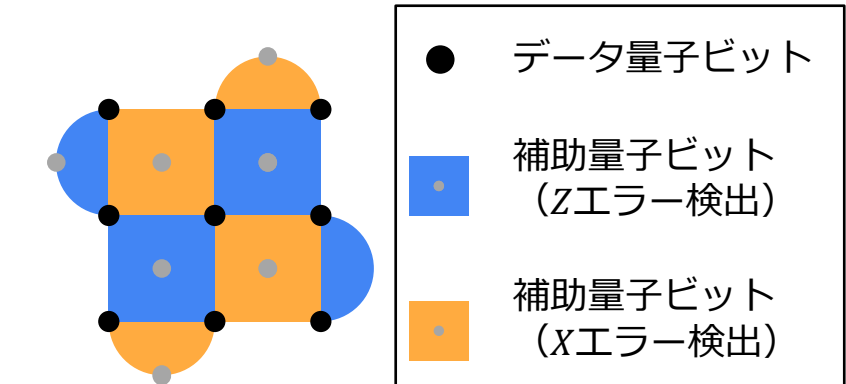
符号パラメータ $[d^2, 1, d]$

- 表面符号を45度回転させると符号距離を保ったまま物理量子ビット数を削減できる

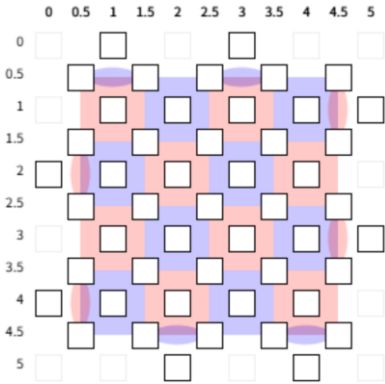
補足: Rotated surface codeの表記



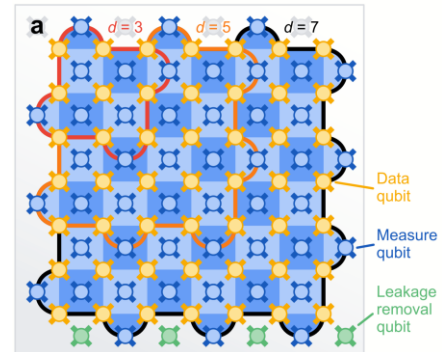
本資料前半での表記



本資料後半での表記

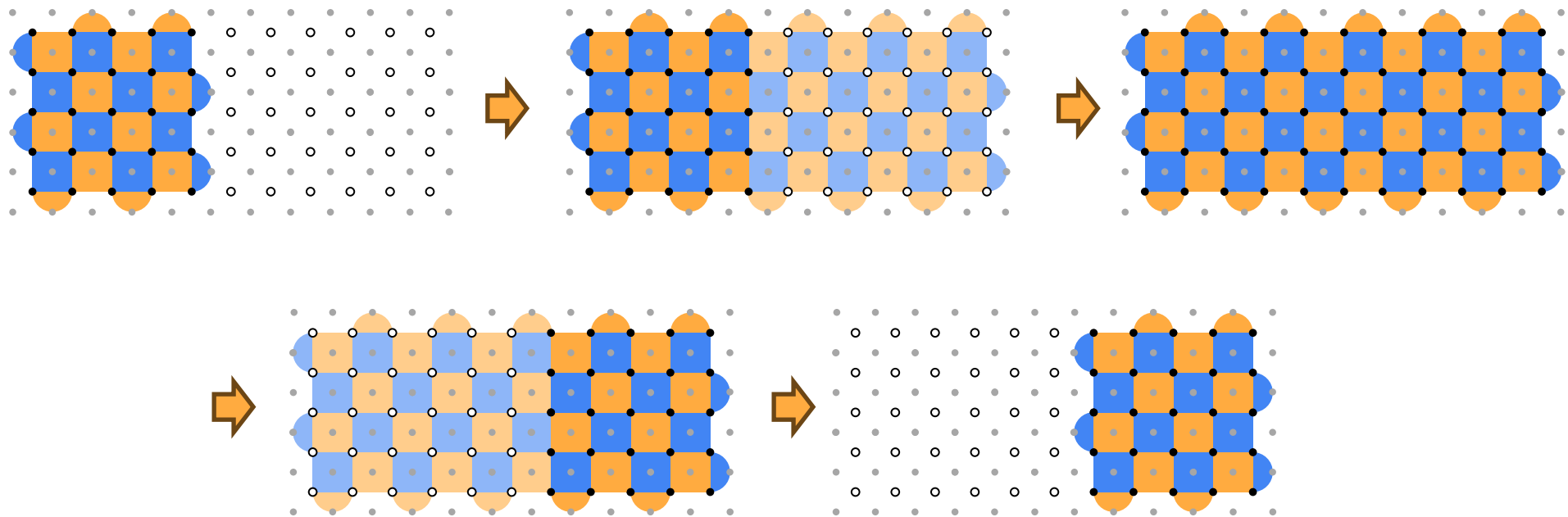


Crumble (後述) での表記 ($d = 5$)

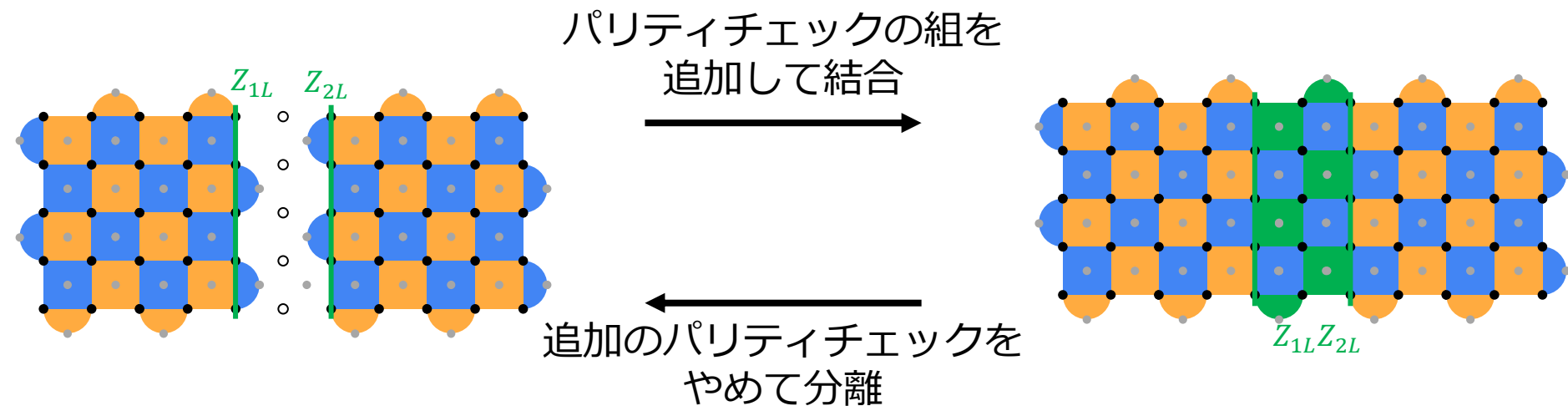


Google論文での表記
(From arXiv2408.13687)

符号の変形

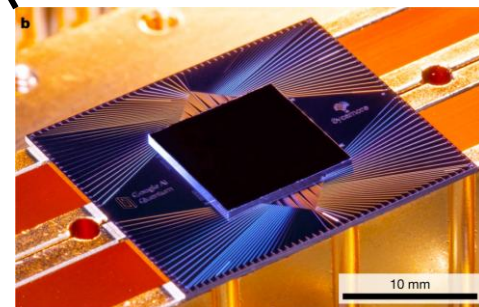
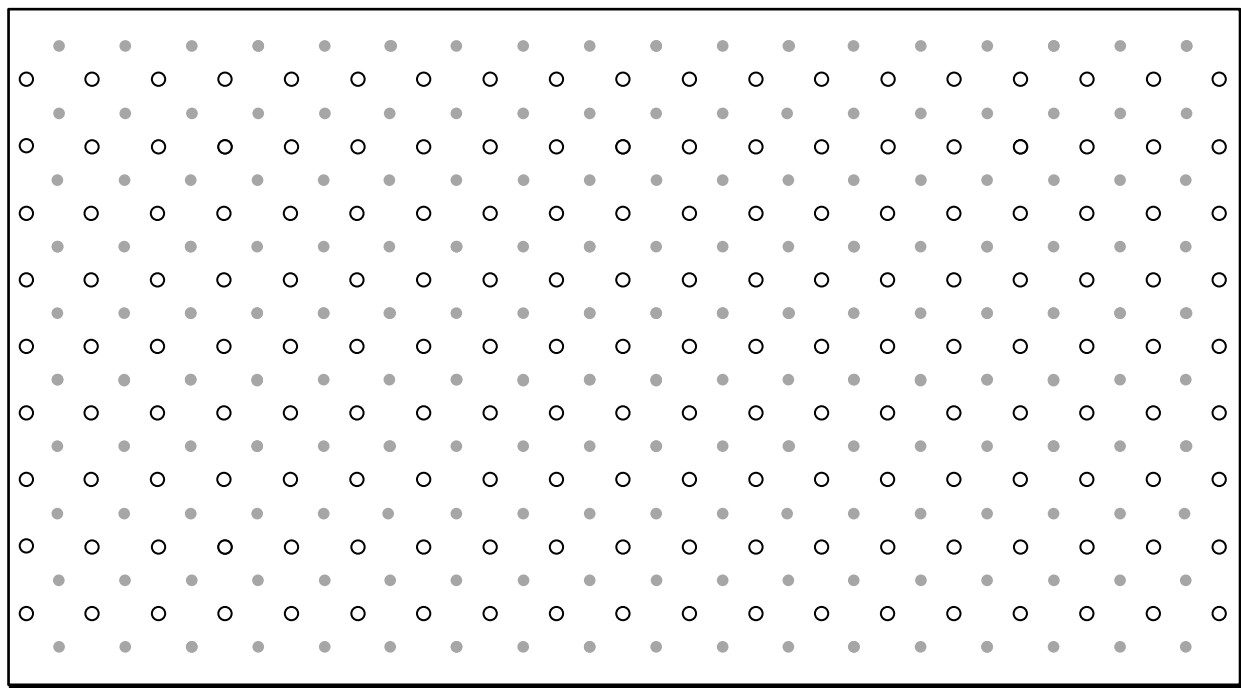


- パリティチェックの組を変えることで符号を変形
 - 論理状態を保ったまま変形 -> 拡大、縮小、**移動**、回転
 - 論理状態を変えながら変形 -> 論理S, CNOTゲート



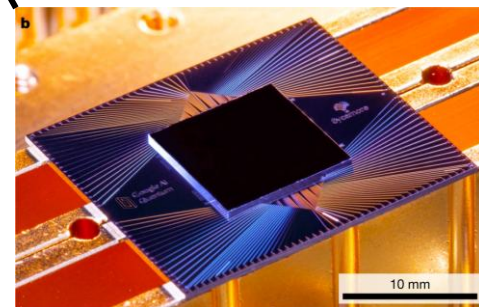
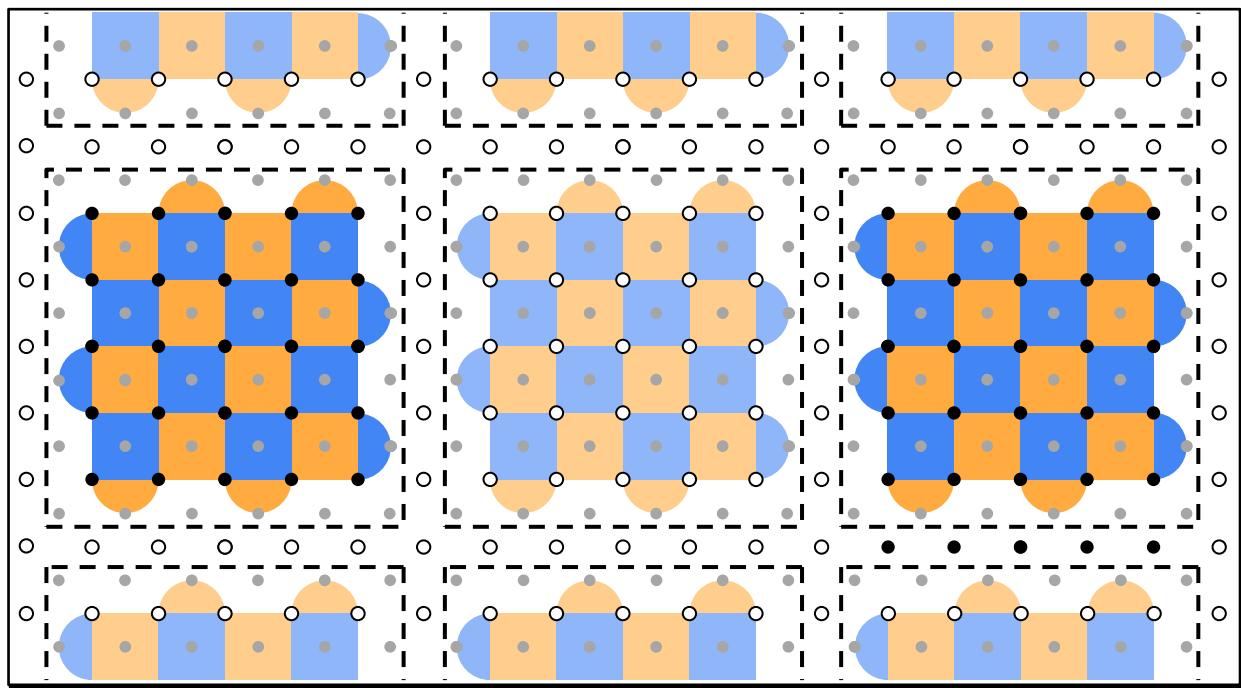
- 論理ビットの間のパリティチェックの結果が2論理ビットのZZ測定になる

実際の量子ビットでの実装



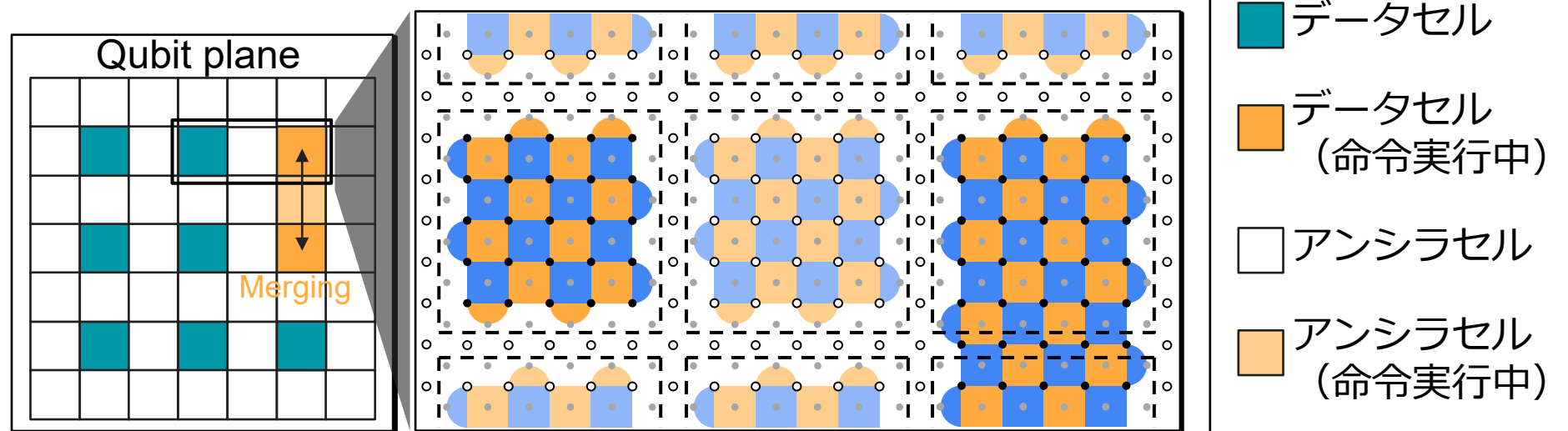
- 表面符号で計算を実行できるように区画を分ける
- 一部に論理状態を確保して一部は計算用に空けておく

実際の量子ビットでの実装



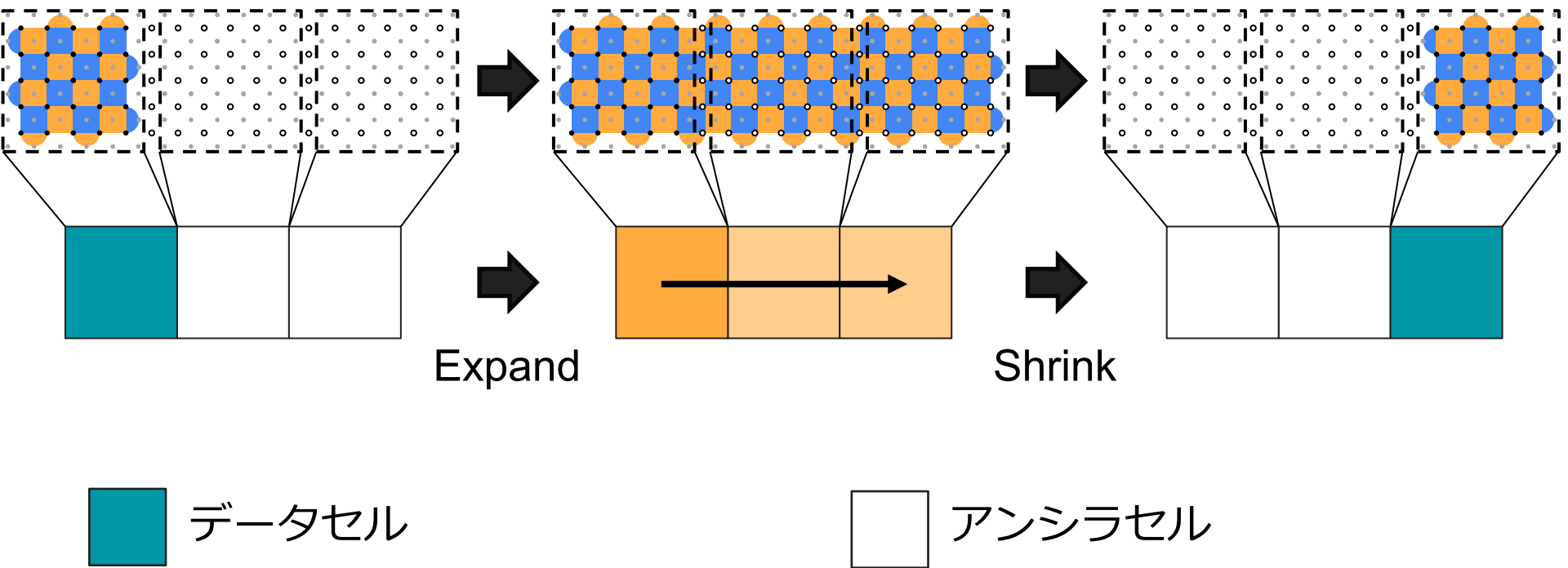
- 表面符号で計算を実行できるように区画を分ける
- 一部に論理状態を確保して一部は計算用に空けておく

表面符号による論理演算を実行するQubit plane

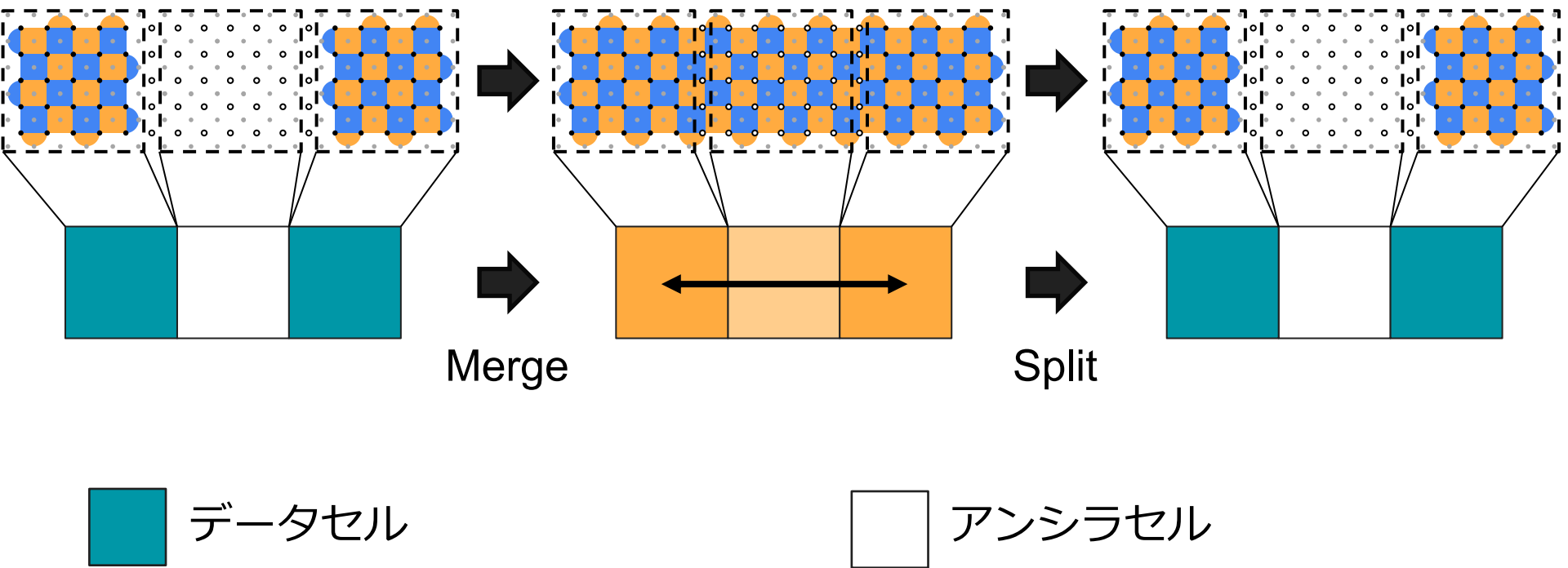


- 1つ1つのセルが符号距離 d の表面符号を構成できる物理量子ビットを持つ
- **データセル**に計算用の論理状態を確保
- 空きスペース (**アンシラセル**) を使って
計算 (= データセルの**拡大・縮小**、**結合・分離**) を行う

例：論理ビットの移動

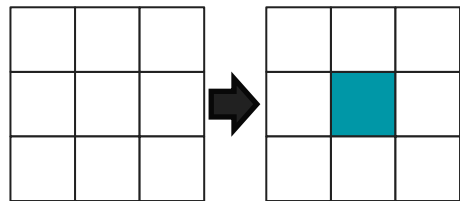


例：格子手術

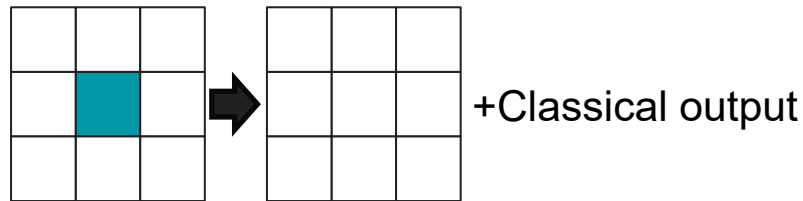


格子手術命令セットの一例

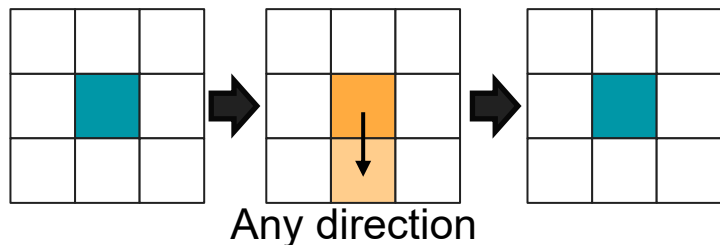
INIT_Z, INIT_X (0 code beat)



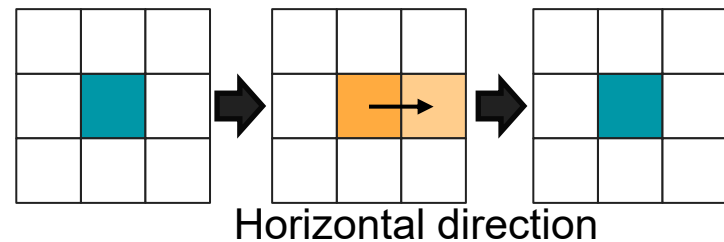
MEAS_X, MEAS_Z (0 code beat)



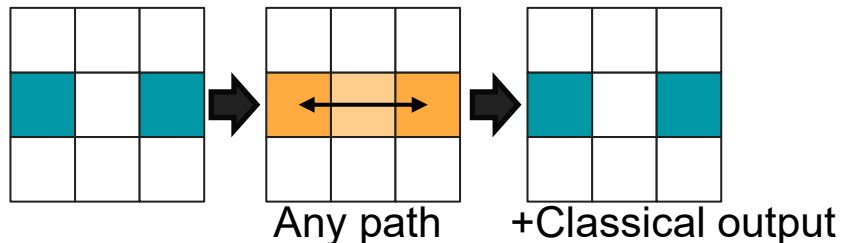
OP_H (3 code beats)



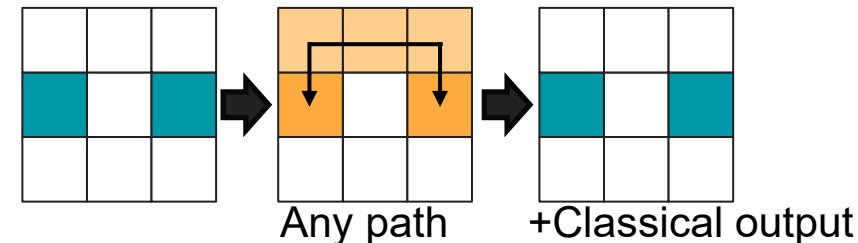
OP_S (2 code beats)



MEAS_ZZ (1 code beat)



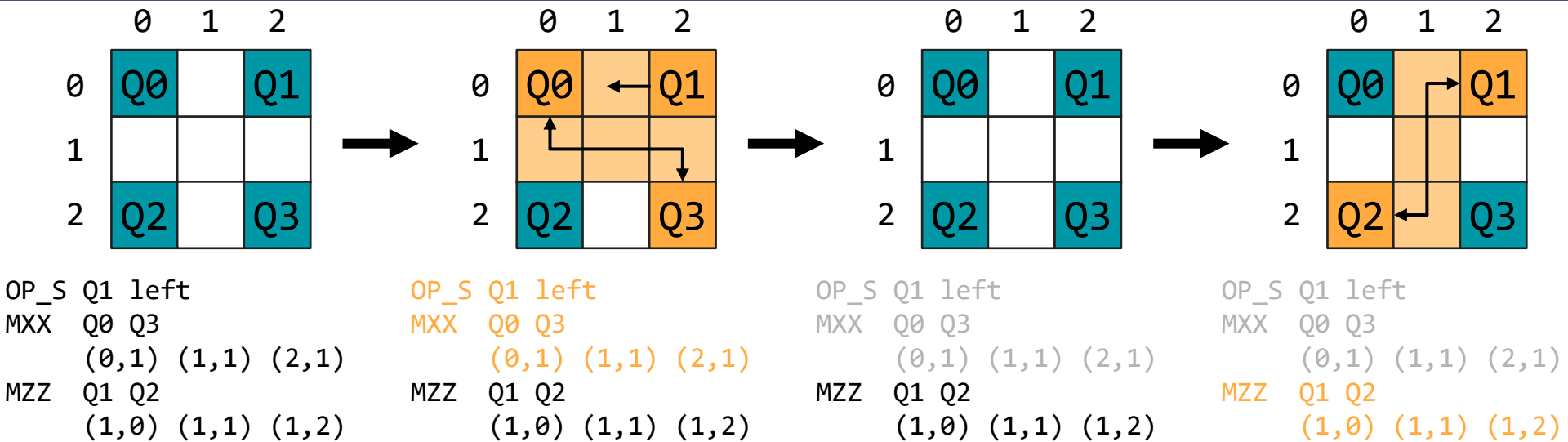
MEAS_XX (1 code beat)



今日の内容

- 量子計算機アーキテクチャ分野の研究動向
 - 計算機アーキテクチャの研究対象・隣接分野
 - 計算機アーキテクチャ分野における量子計算機関連の研究動向
- 量子誤り訂正とそれを支える計算機アーキテクチャ
 - 表面符号における誤り推定
 - 超伝導デジタル回路を用いた誤り推定機構（DAC2021/HPCA2022）
 - 表面符号＋格子手術を用いた誤り耐性量子計算
 - ロードストア型FTQCアーキテクチャ（HPCA2025）
- まとめ

格子手術のコンパイルと問題点

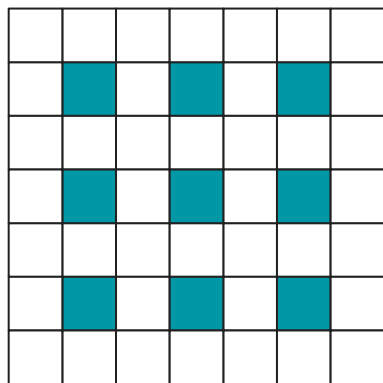


従来の格子手術コンパイルの問題点1

- 各命令がハードウェア情報に依存しすぎておりポータビリティが低い
 - チップの変形やデータ位置の変更によりコンパイルがやり直しになる

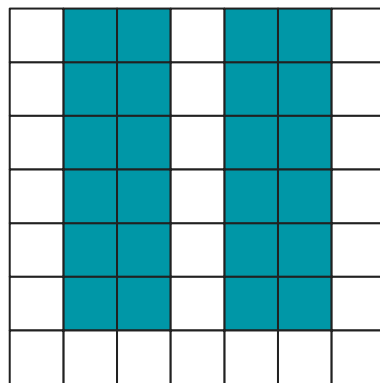
データセル配置と実行時間

- データセル配置密度と実行時間の間のトレードオフがある



Low: $1/4$

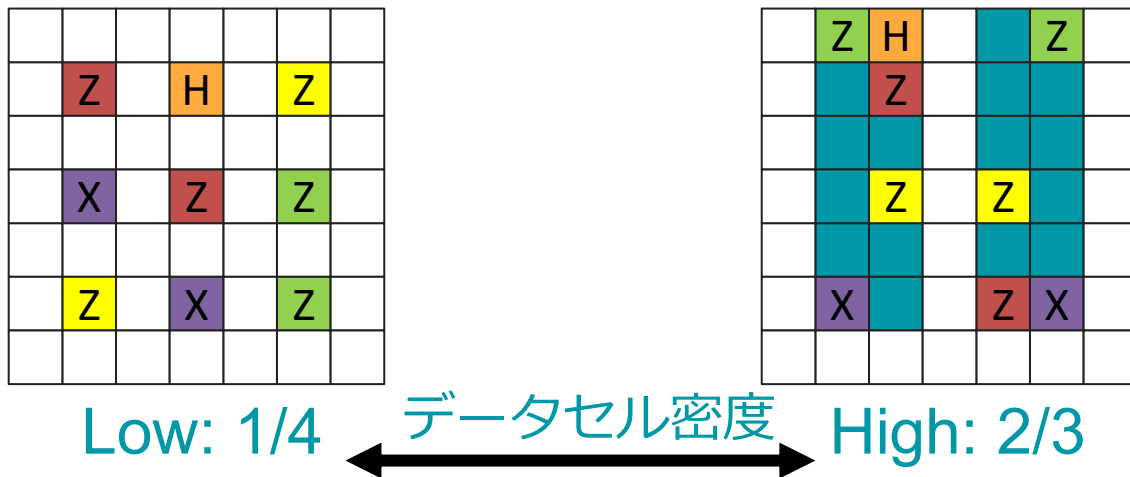
データセル密度



High: $2/3$

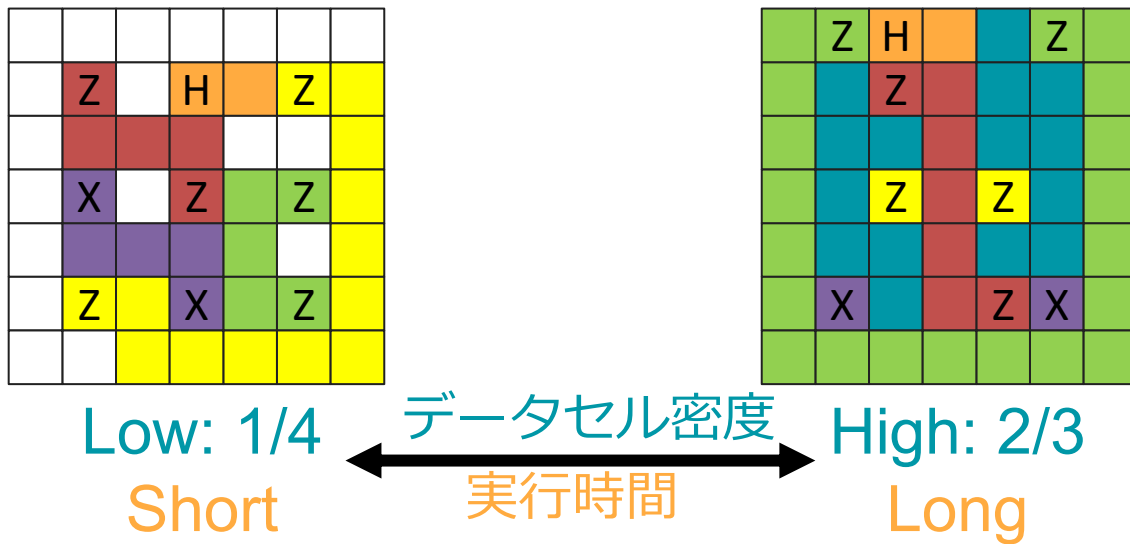
データセル配置と実行時間

- データセル配置密度と実行時間の間のトレードオフがある

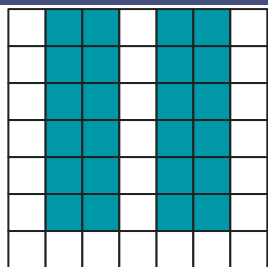


データセル配置と実行時間

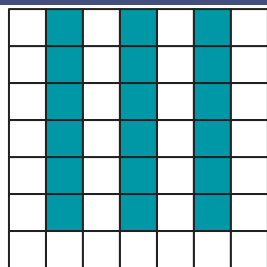
- データセル配置密度と実行時間の間のトレードオフがある



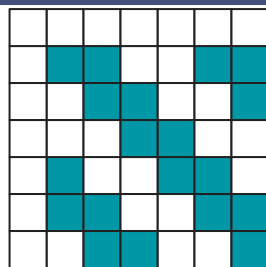
従来のデータセル配置



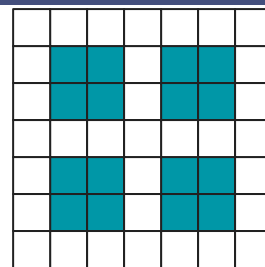
2/3 [1]



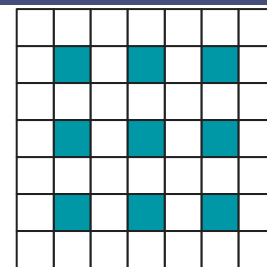
1/2 [2]



1/2 [3]



4/9 [4]



1/4 [5]

← データセル配置密度

任意のデータセルがアンシラセルに隣接しており、常に任意のデータセルに対する操作が可能



従来の格子手術コンパイルの問題点2

- 空間の利用効率が低い
 - 領域の多くが演算用のアンシラセルで、利用されない時間が多い

[1] J. Lee et al., PRX Quantum 2, 030305 (2021). [2] M. Beverland et al., arXiv:2211.07629 (2022). [3] Y. Ueno et al., arXiv:2411.17519 (2024).

[4] C. Chamberland and E. T. Campbell, PRX Quantum 3, 010331 (2022). [5] M. Beverland et al., PRX Quantum 3, 020342 (2022).

従来のデータセル配置



「すべてのデータセルにいつでもアクセスできる」ことは
必須なのか？

2/3 [1]

1/2 [2]

1/2 [3]

4/9 [4]

1/4 [5]

← データセル配置密度

任意のデータセルがアンシラセルに隣接しており、
常に任意のデータセルに対する操作が可能



従来の格子手術コンパイルの問題点2

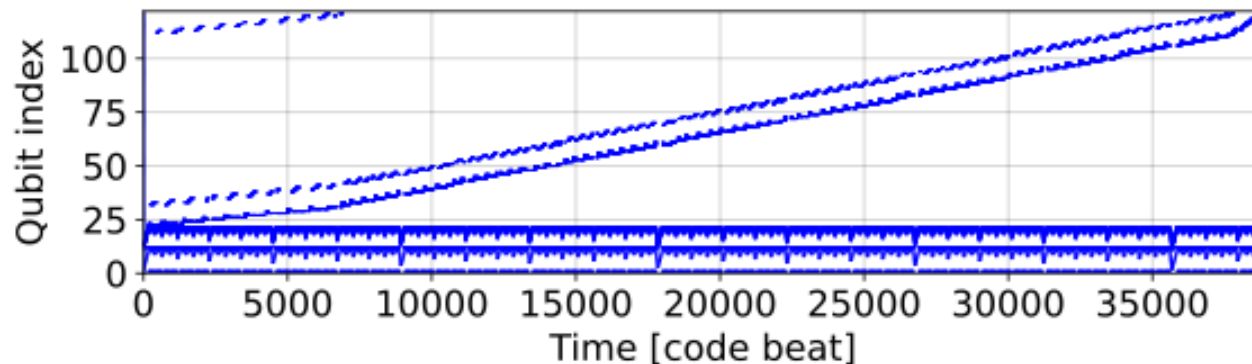
- 空間の利用効率が低い
 - 領域の多くが演算用のアンシラセルで、利用されない時間が多い

[1] J. Lee et al., PRX Quantum 2, 030305 (2021). [2] M. Beverland et al., arXiv:2211.07629 (2022). [3] Y. Ueno et al., arXiv:2411.17519 (2024).

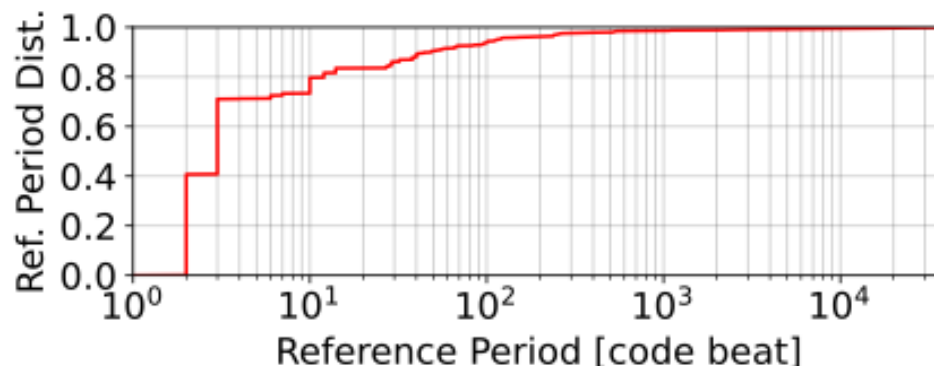
[4] C. Chamberland and E. T. Campbell, PRX Quantum 3, 010331 (2022). [5] M. Beverland et al., PRX Quantum 3, 020342 (2022).

QPEにおけるデータセルへのアクセスパターン

Reference time stamp for SELECT subroutine of quantum phase estimation

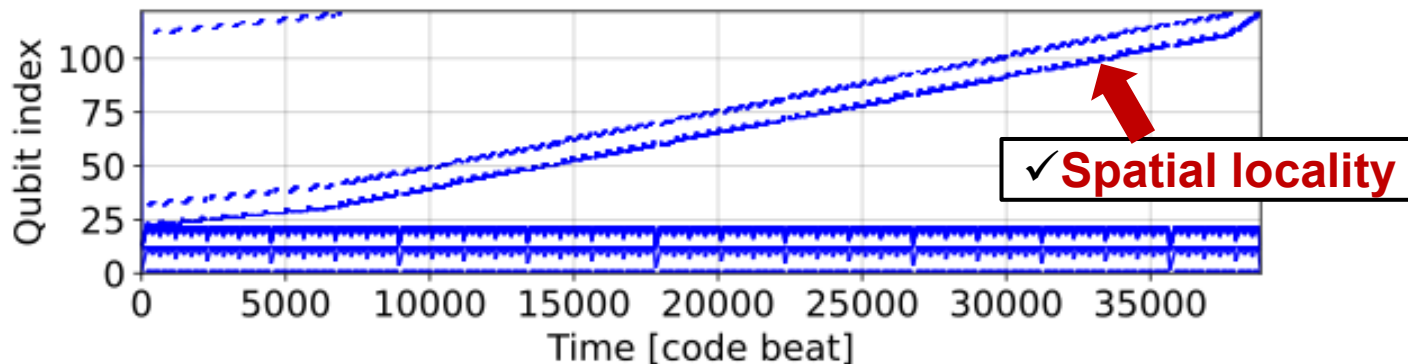


Cumulative distribution of average reference period for SELECT circuit

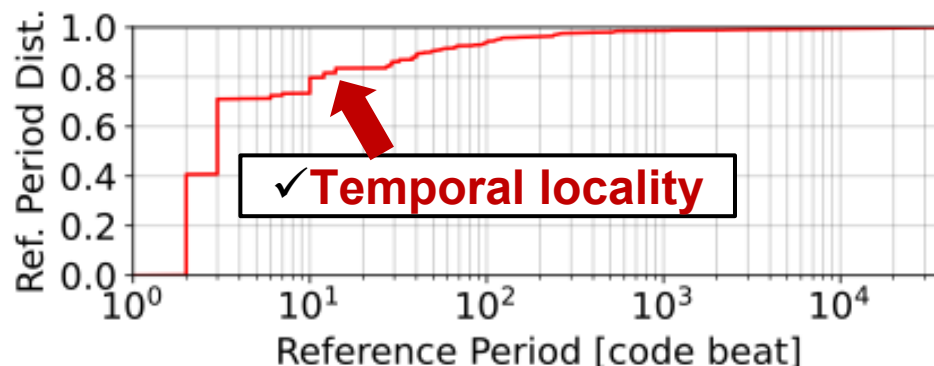


QPEにおけるデータセルへのアクセスパターン

Reference time stamp for SELECT subroutine of quantum phase estimation

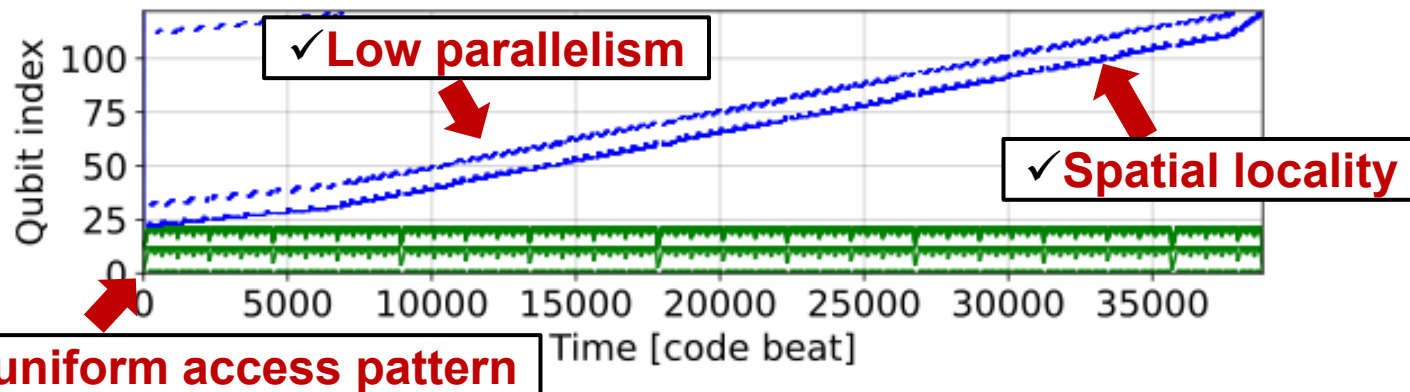


Cumulative distribution of average reference period for SELECT circuit

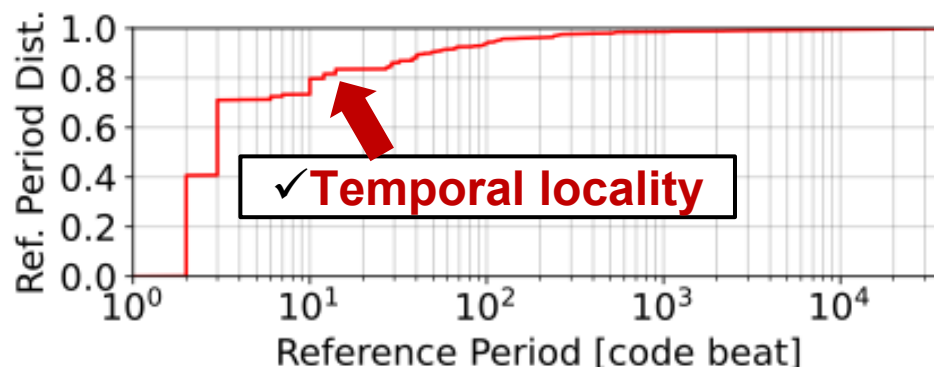


QPEにおけるデータセルへのアクセスパターン

Reference time stamp for SELECT subroutine of quantum phase estimation

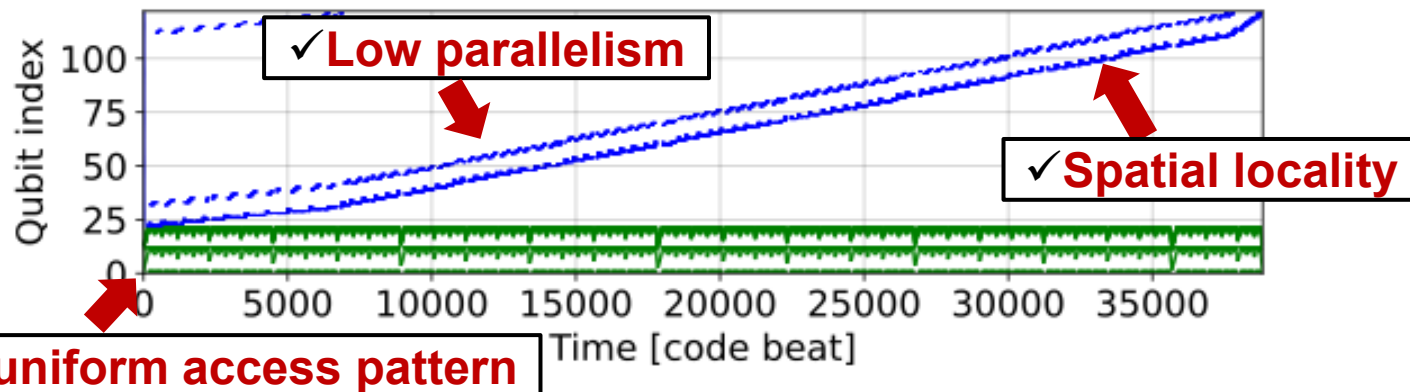


Cumulative distribution of average reference period for SELECT circuit

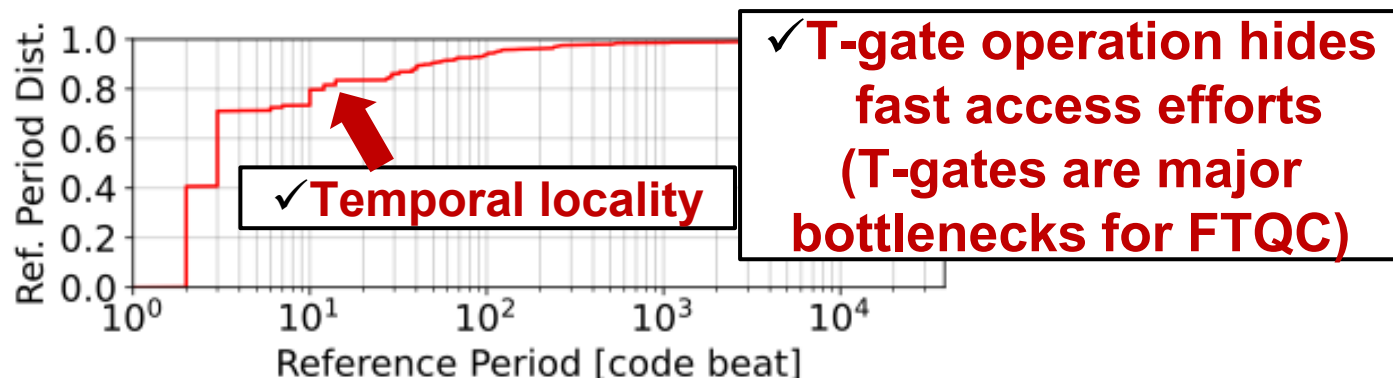


QPEにおけるデータセルへのアクセスパターン

Reference time stamp for SELECT subroutine of quantum phase estimation

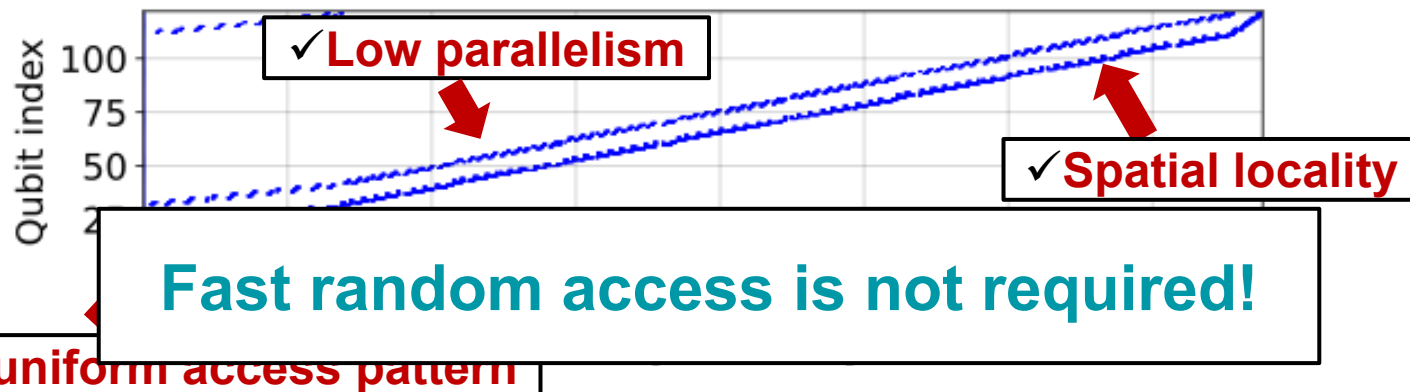


Cumulative distribution of average reference period for SELECT circuit

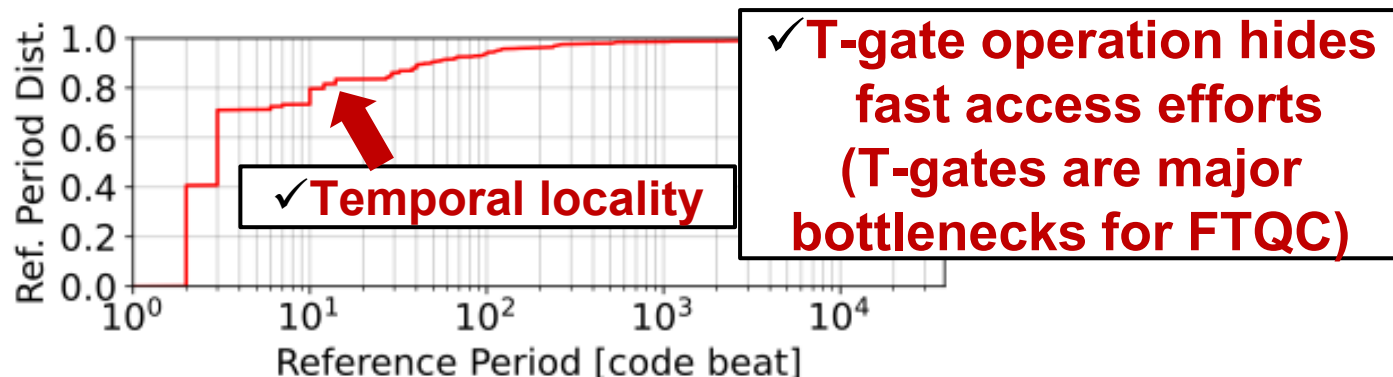


QPEにおけるデータセルへのアクセスパターン

Reference time stamp for SELECT subroutine of quantum phase estimation



Cumulative distribution of average reference period for SELECT circuit



QPEにおけるデータセルへのアクセスパターン

Reference time stamp for SELECT subroutine of quantum phase estimation

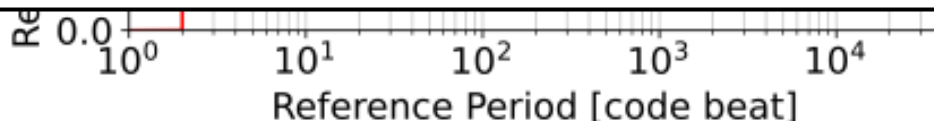


Our goal

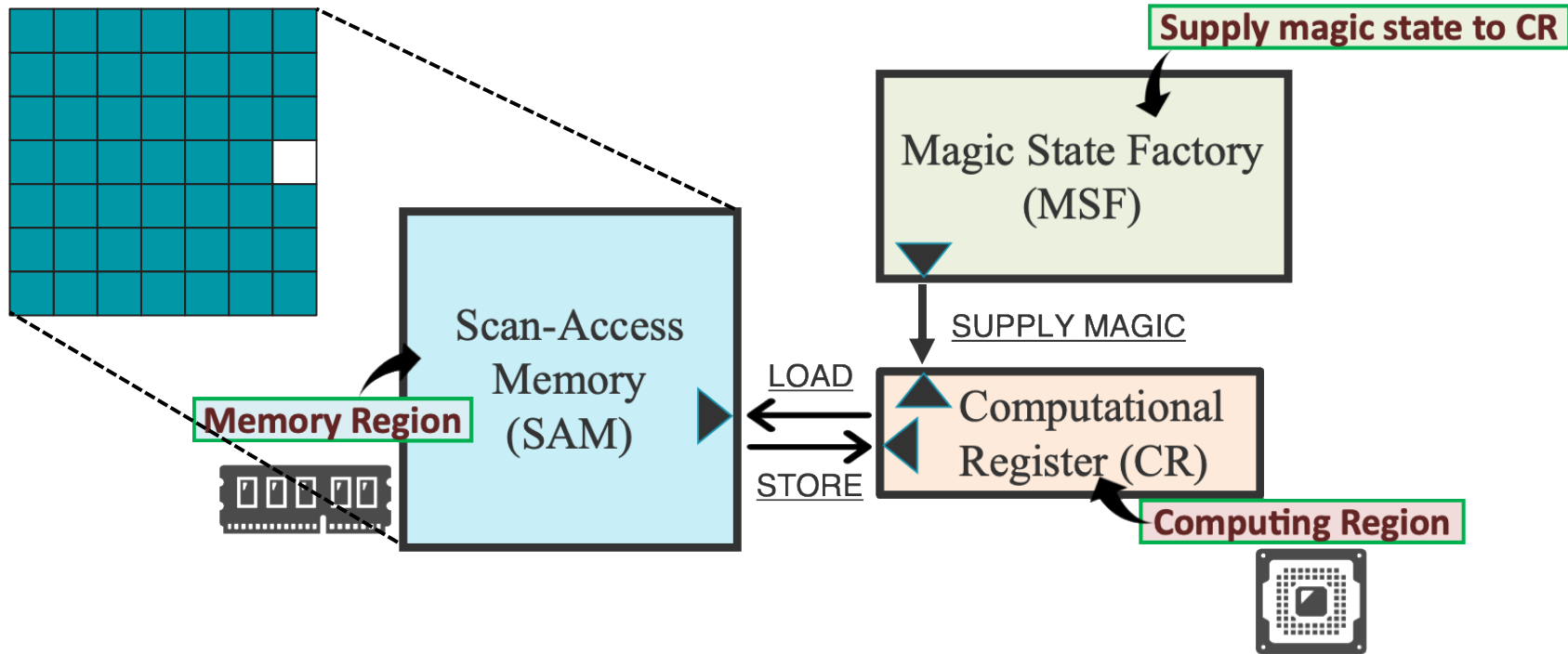
Higher memory density with a small time overhead

Our approach

- ✓ **Divide qubit plane into register and memory region**
- ✓ **Utilize access locality and optimize for biased access pattern**
- ✓ **Conceal the overhead by T-gate operations and bottleneck operation**



ロードストア型誤り耐性量子計算機アーキテクチャ



- 高密度にデータセルを配置するメモリ領域を構成
- データセルの移動を活用して計算領域に読み出し

ISA of our load/store architecture for FTQC

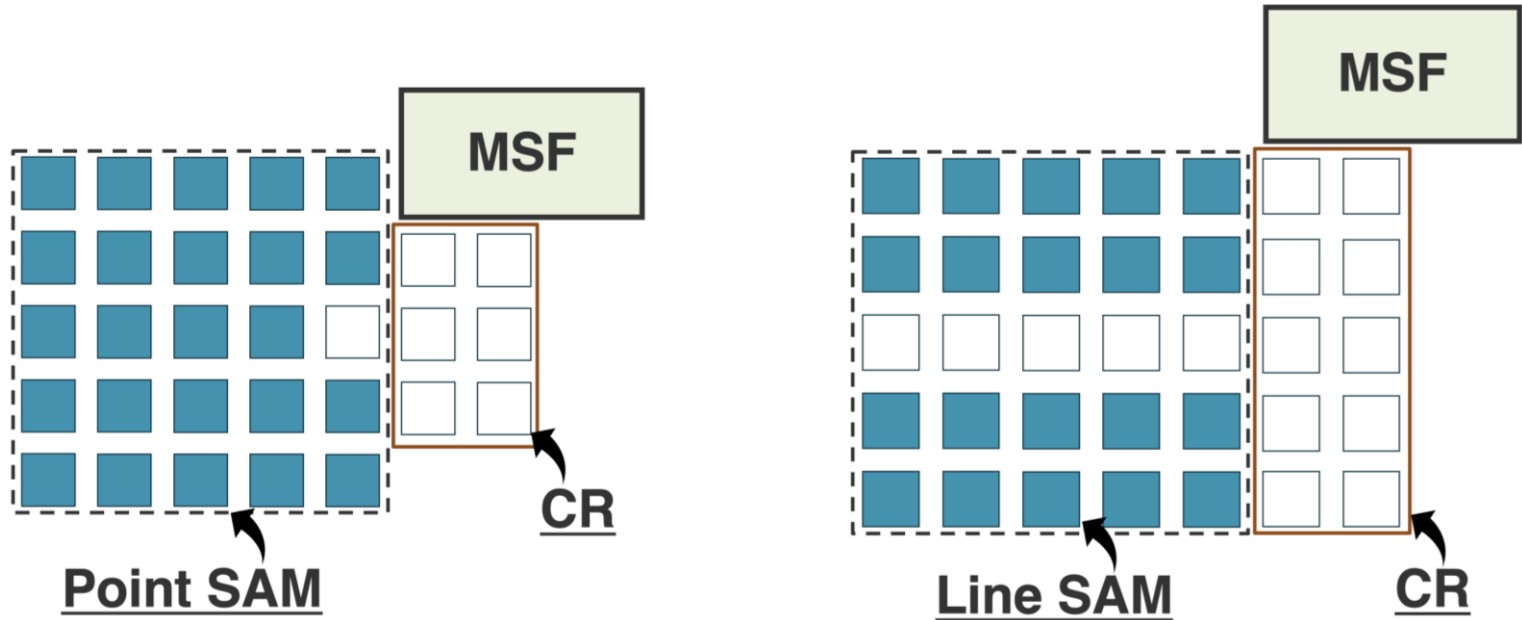
Load/
Store

Type	Syntax	Latency	Description
Memory	LD M C	variable	(Load) Load logical qubit from SAM to CR
	ST C M	variable	(Store) Store logical qubit from CR to SAM
Preparation	PZ.C C	0 beat	(Zero-Init) Initialize a logical qubit to $ 0\rangle$ state
	PP.C C	0 beat	(Plus-Init) Initialize a logical qubit to $ +\rangle$ state
	PM C	variable	(Magic-init) Move magic state from MSF to CR
Unitary	HD.C C	3 beat	(Hadamard) Hadamard gate on a logical qubit
	PH.C C	2 beat	(Phase) Phase gate on a logical qubit
Measurement	MX.C C V	0 beat	(Pauli-X Meas) Pauli-X measurement on a logical qubit and store outcome
	MZ.C C V	0 beat	(Pauli-Z Meas) Pauli-Z measurement on a logical qubit and store outcome
	MXX.C C1 C2 V	1 beat	(Pauli-XX Meas) Pauli-XX measurement on logical qubits and store outcome
	MZZ.C C1 C2 V	1 beat	(Pauli-ZZ Meas) Pauli-ZZ measurement on logical qubits and store outcome
Control	SK V	variable	(Skip) Skip next instruction if a provided value is zero
In-Memory Preparation	PZ.M M	0 beat	(Zero-Init) Initialize a logical qubit to $ 0\rangle$ state
	PP.M M	0 beat	(Plus-Init) Initialize a logical qubit to $ +\rangle$ state
In-Memory Unitary	HD.M M	variable	(Hadamard) Hadamard gate on a logical qubit
	PH.M M	variable	(Phase) Phase gate on a logical qubit
In-Memory Measurement	MX.M M V	0 beat	(Pauli-X Meas) Pauli-X measurement on a logical qubit and store outcome
	MZ.M M V	0 beat	(Pauli-Z Meas) Pauli-Z measurement on a logical qubit and store outcome
	MXX.M C M V	variable	(Pauli-XX Meas) Pauli-XX measurement on logical qubits and store outcome
	MZZ.M C M V	variable	(Pauli-ZZ Meas) Pauli-ZZ measurement on logical qubits and store outcome
Optimized Unitary	CX M1 M2	variable	(CNOT) CNOT gate on logical qubits

- ✓ Load/Store instructions are independent of memory region implementation (e.g., QEC code, allocation)
- ✓ Enable of abstraction, easily programable, and high program portability

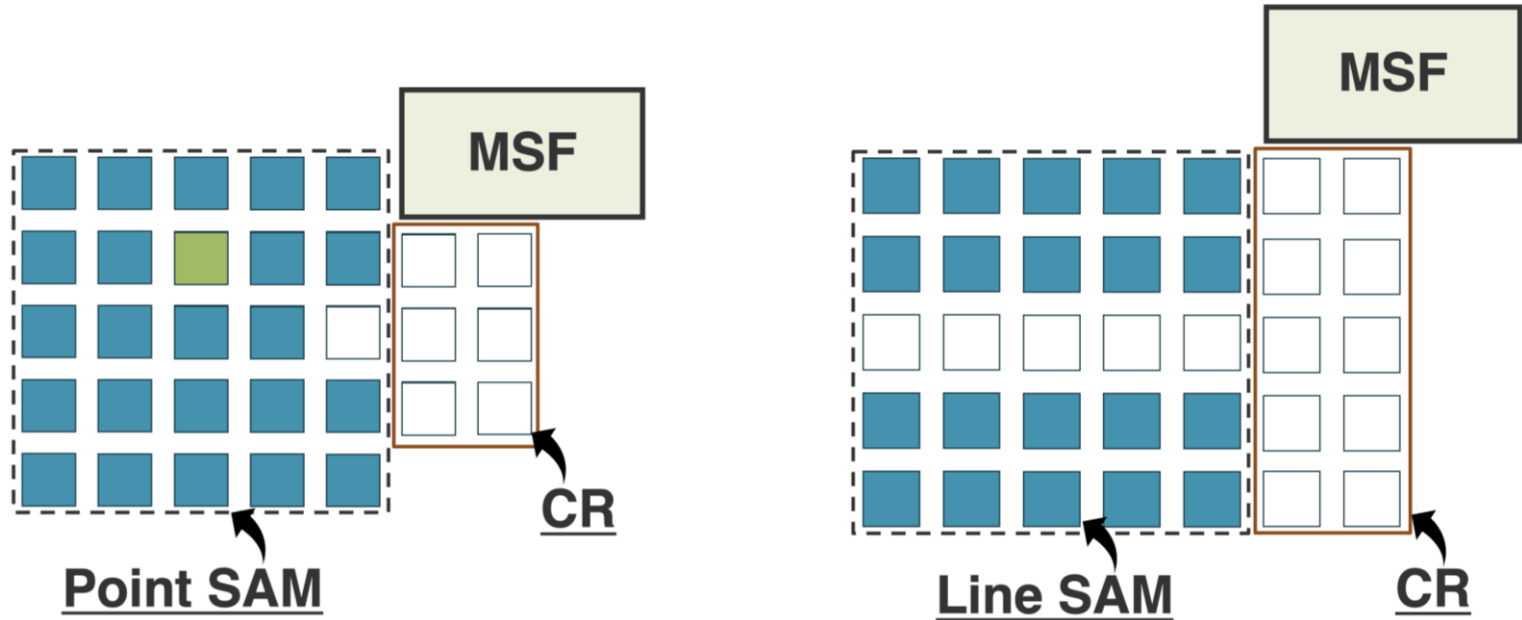
ロードストア型誤り耐性量子計算機アーキテクチャ

- Scan-Access Memory



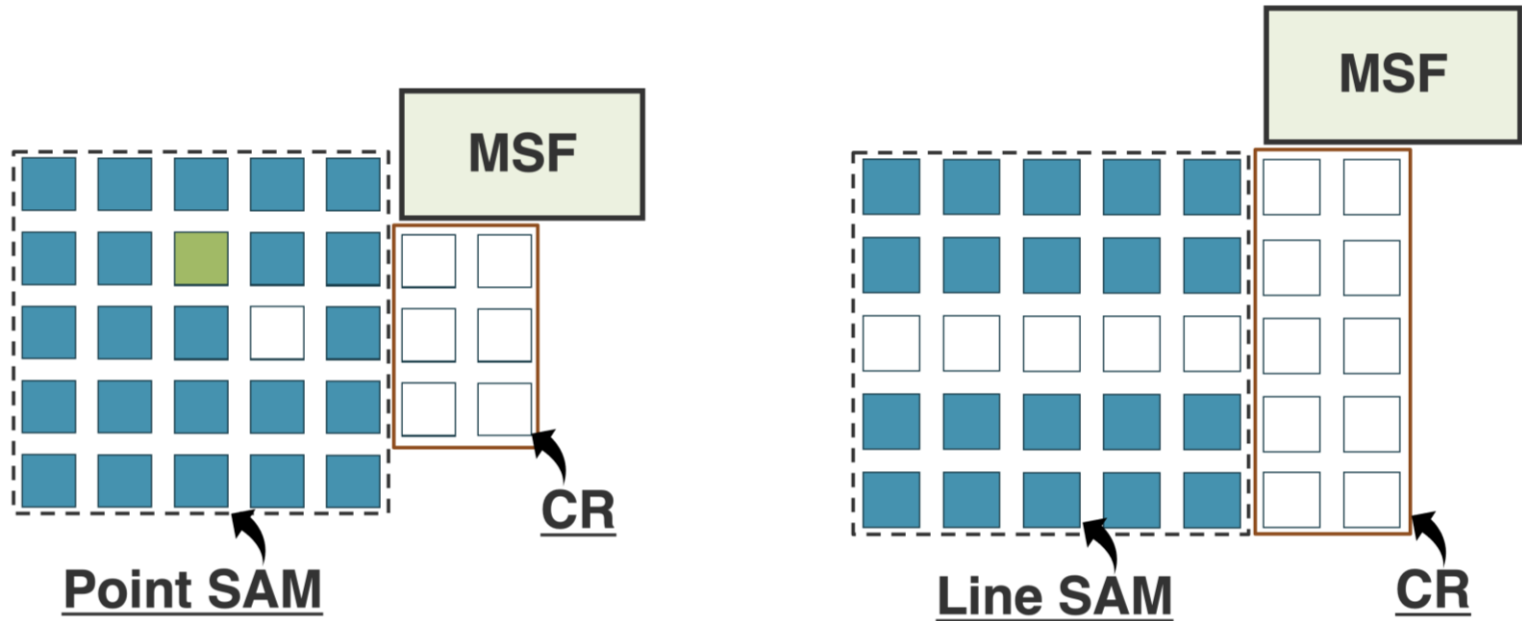
ロードストア型誤り耐性量子計算機アーキテクチャ

- Scan-Access Memory



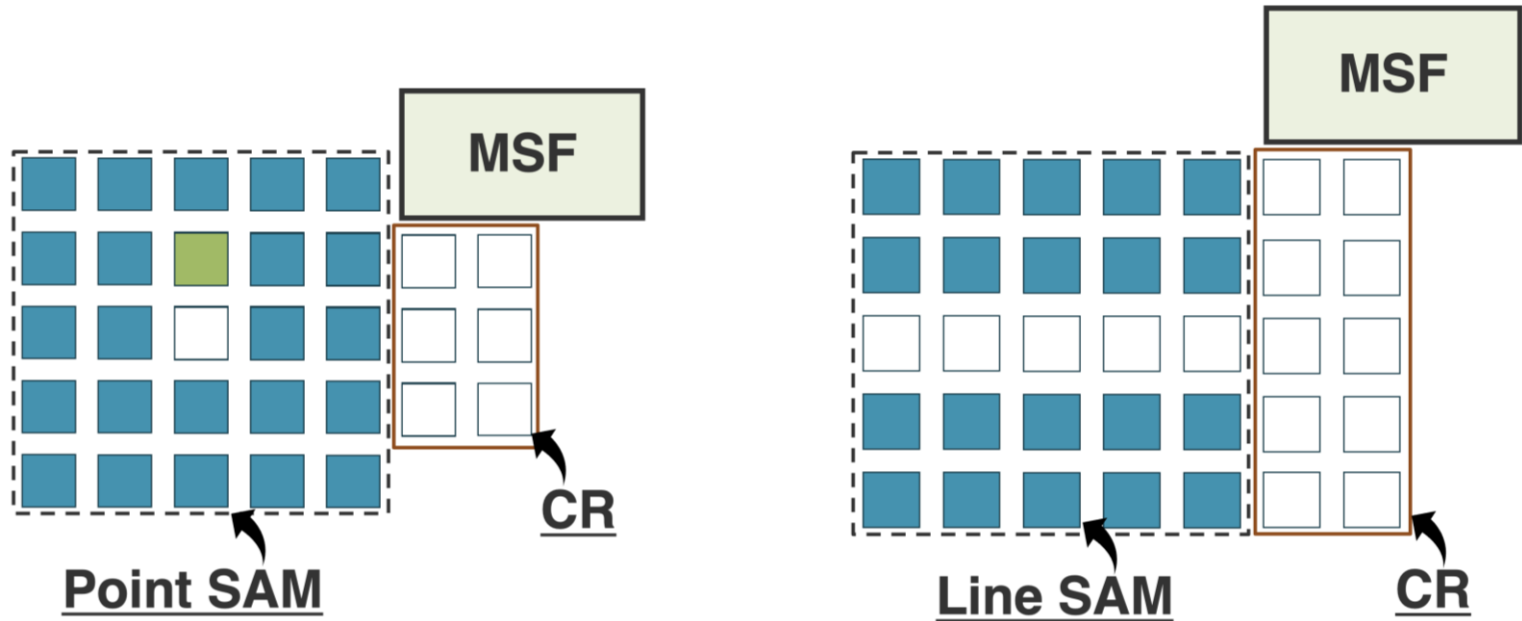
ロードストア型誤り耐性量子計算機アーキテクチャ

- Scan-Access Memory



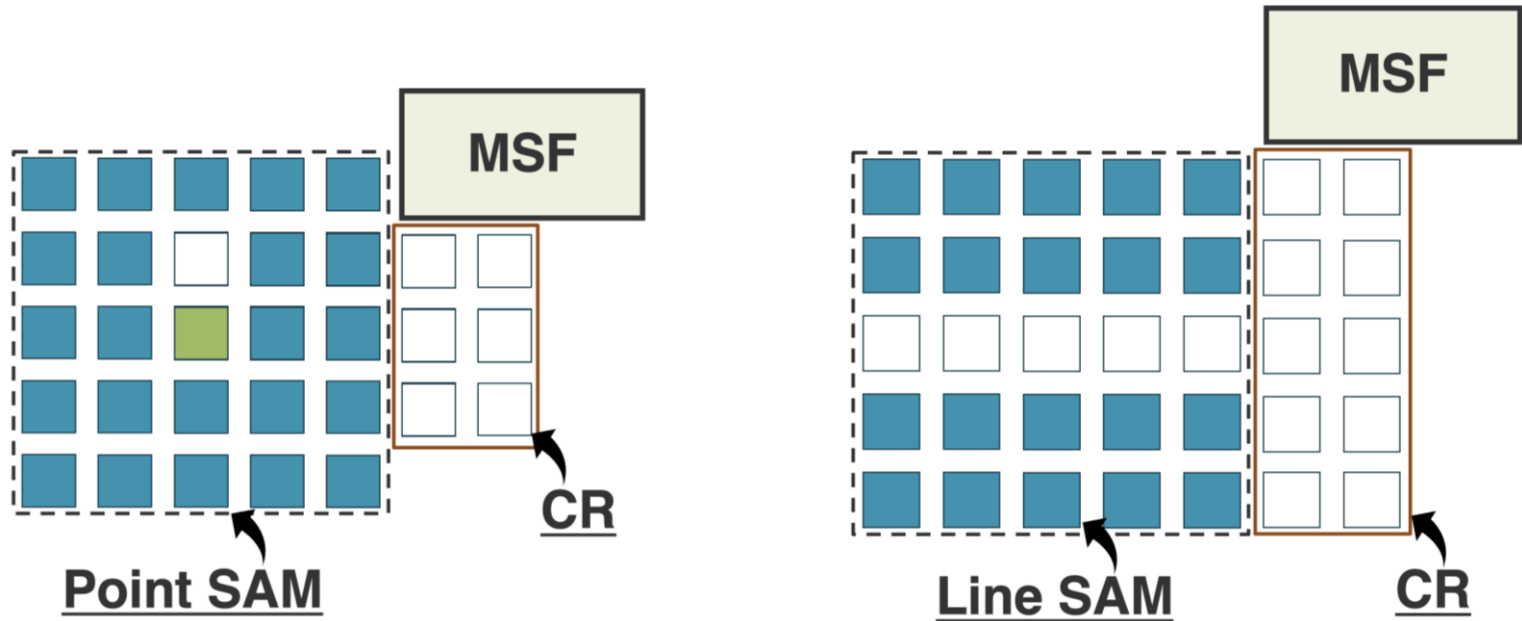
ロードストア型誤り耐性量子計算機アーキテクチャ

- Scan-Access Memory



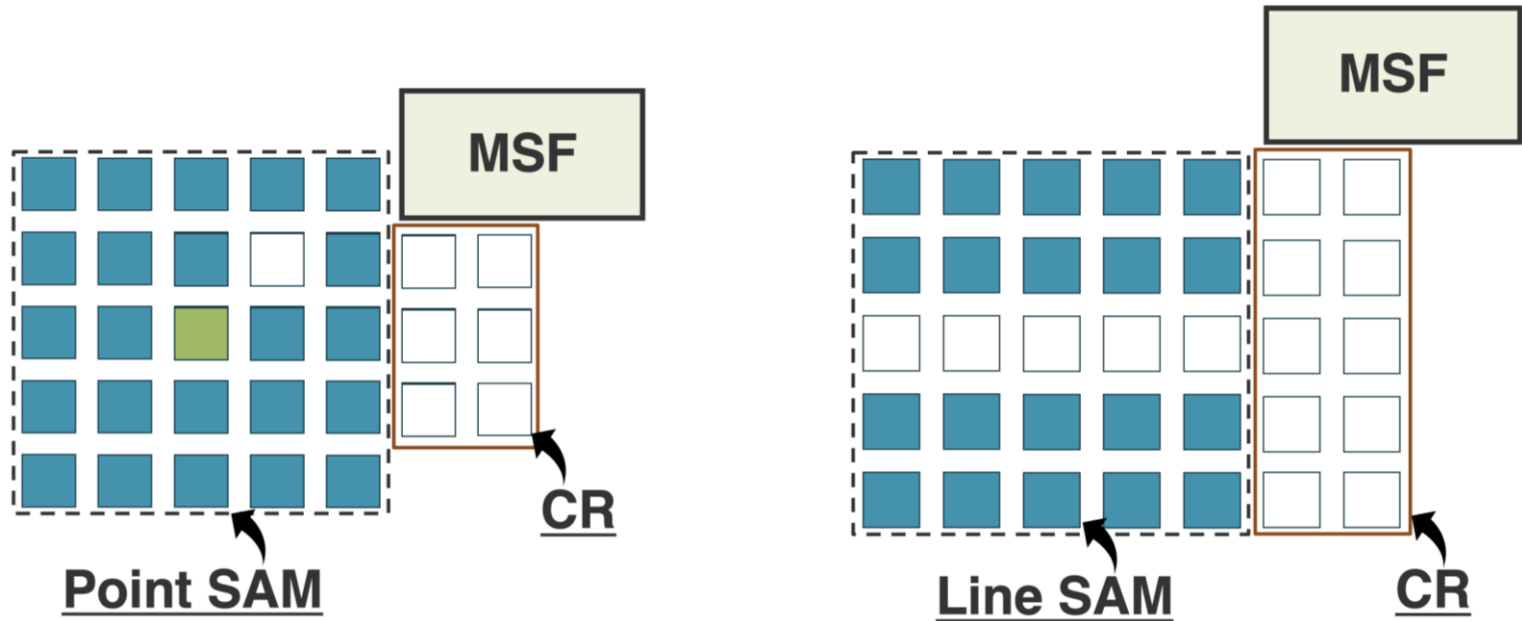
ロードストア型誤り耐性量子計算機アーキテクチャ

- Scan-Access Memory



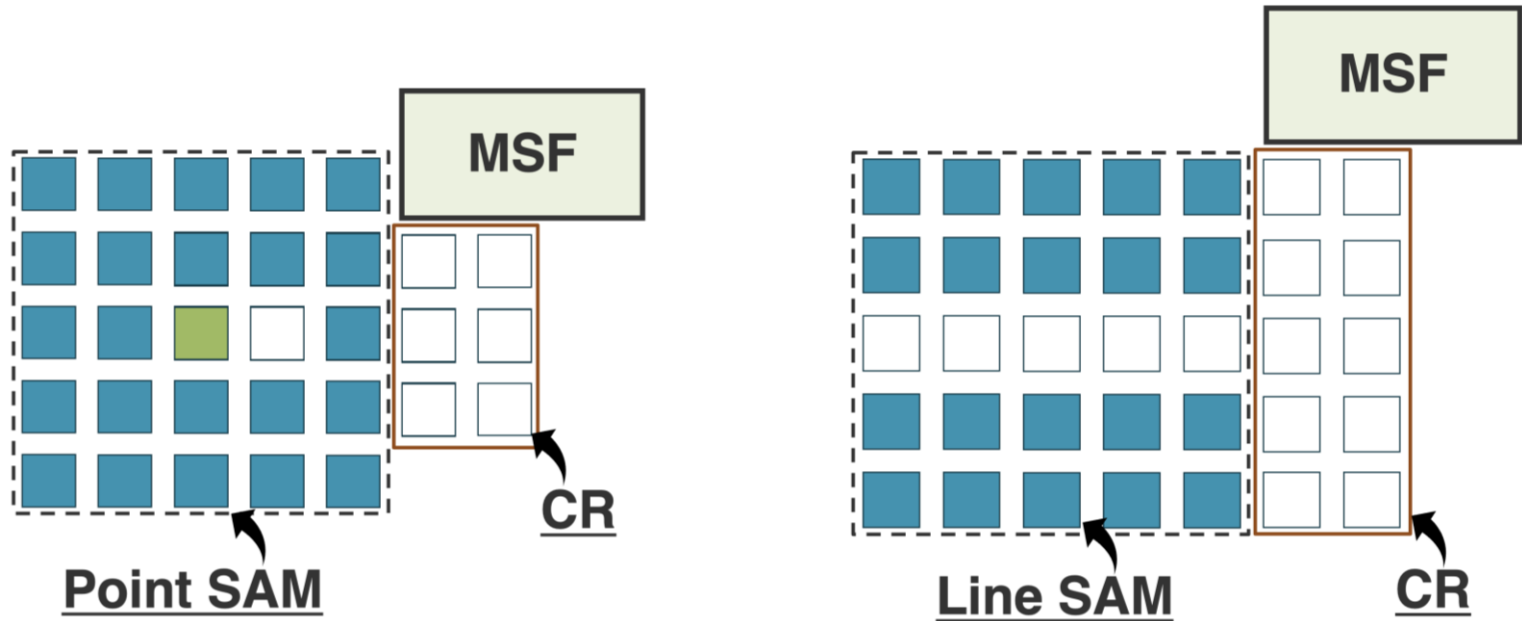
ロードストア型誤り耐性量子計算機アーキテクチャ

- Scan-Access Memory



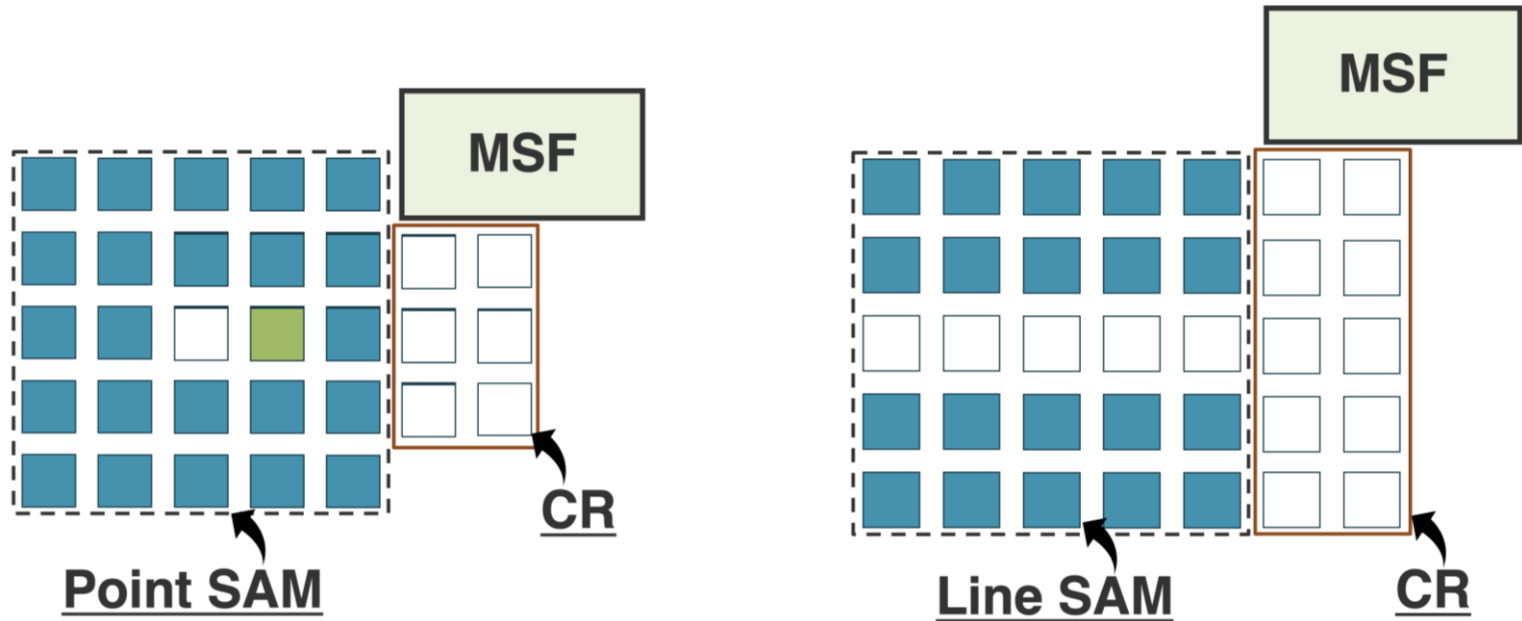
ロードストア型誤り耐性量子計算機アーキテクチャ

- Scan-Access Memory



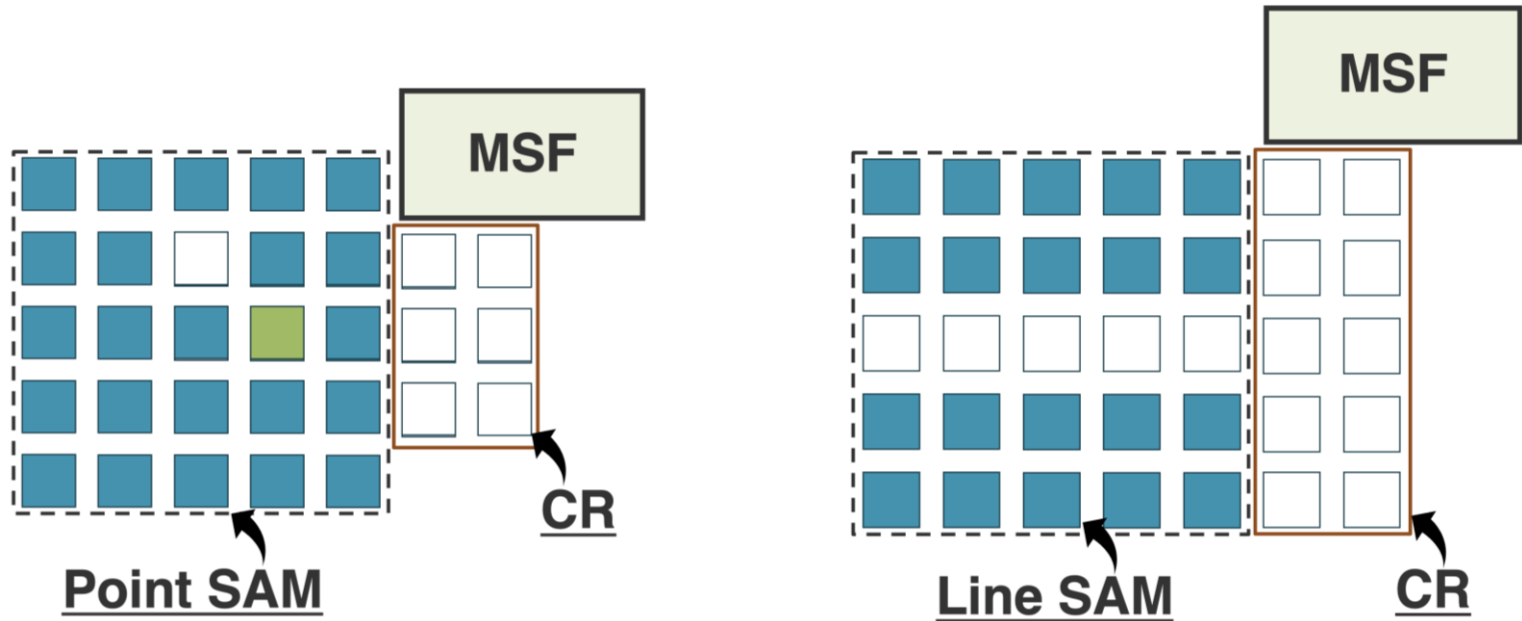
ロードストア型誤り耐性量子計算機アーキテクチャ

- Scan-Access Memory



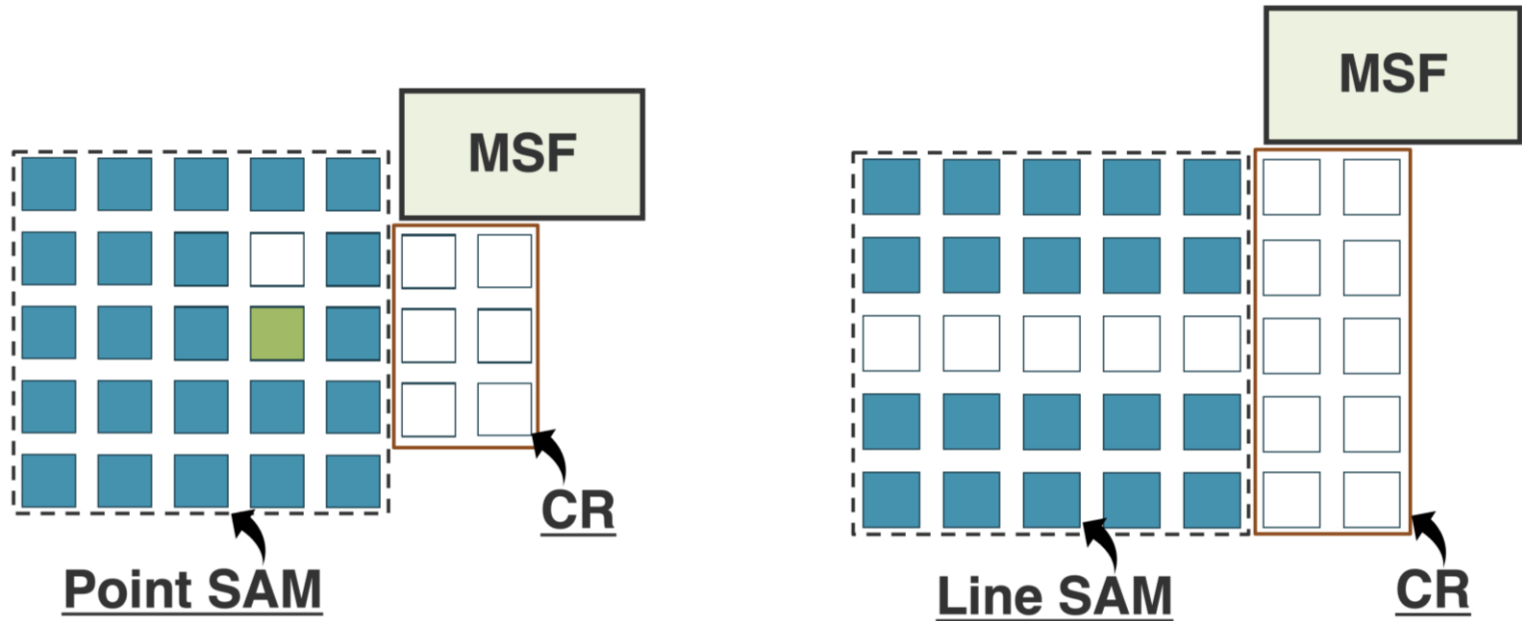
ロードストア型誤り耐性量子計算機アーキテクチャ

- Scan-Access Memory



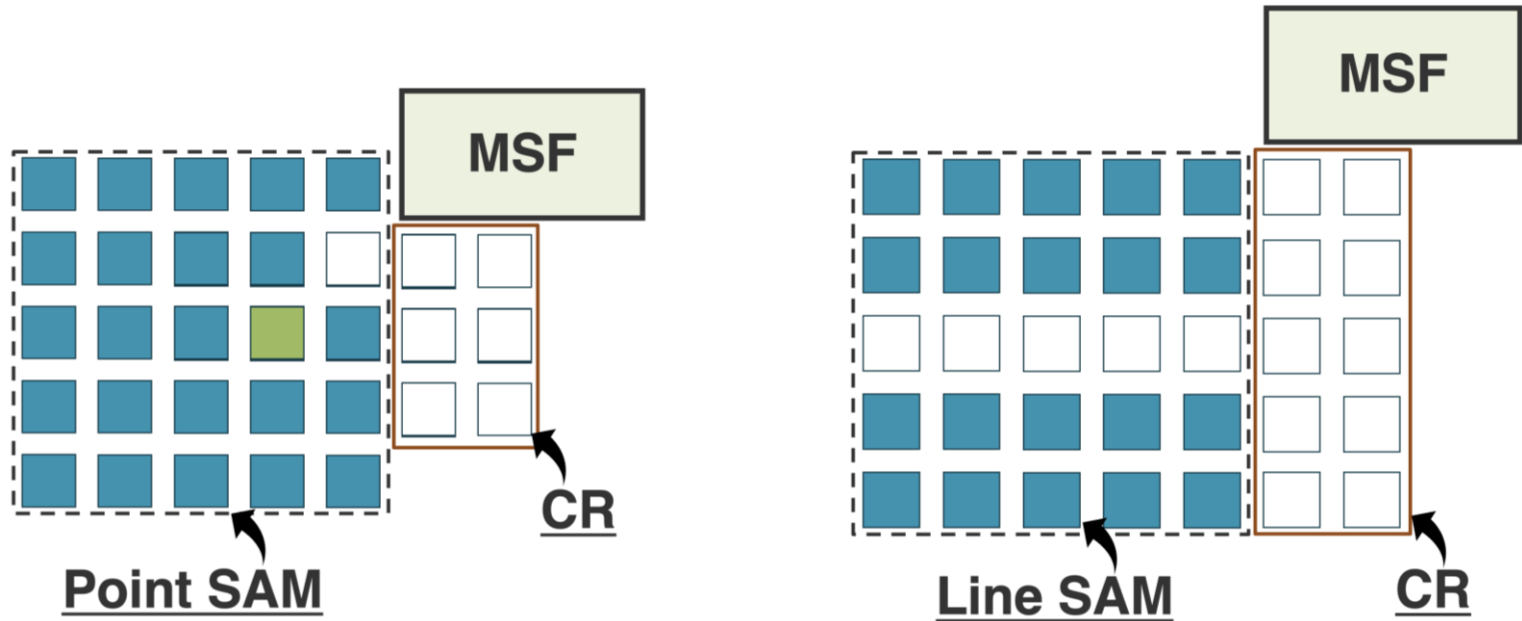
ロードストア型誤り耐性量子計算機アーキテクチャ

- Scan-Access Memory



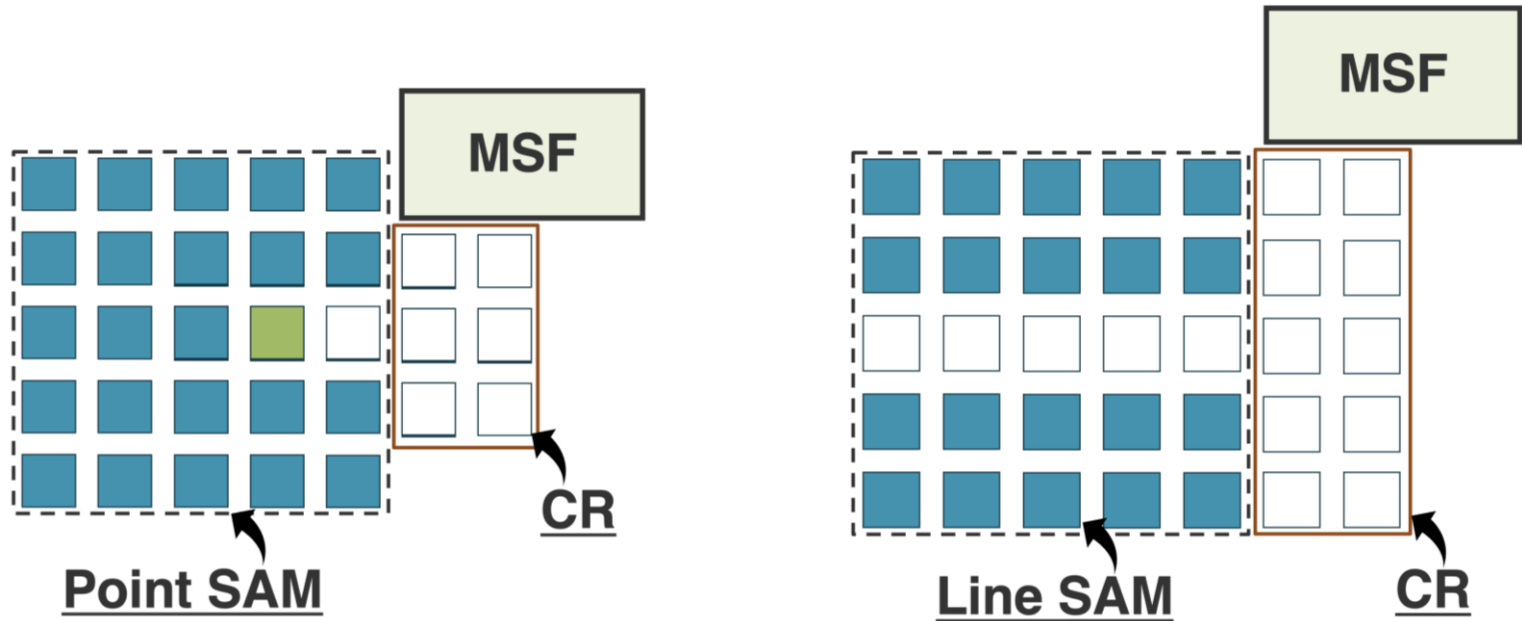
ロードストア型誤り耐性量子計算機アーキテクチャ

- Scan-Access Memory



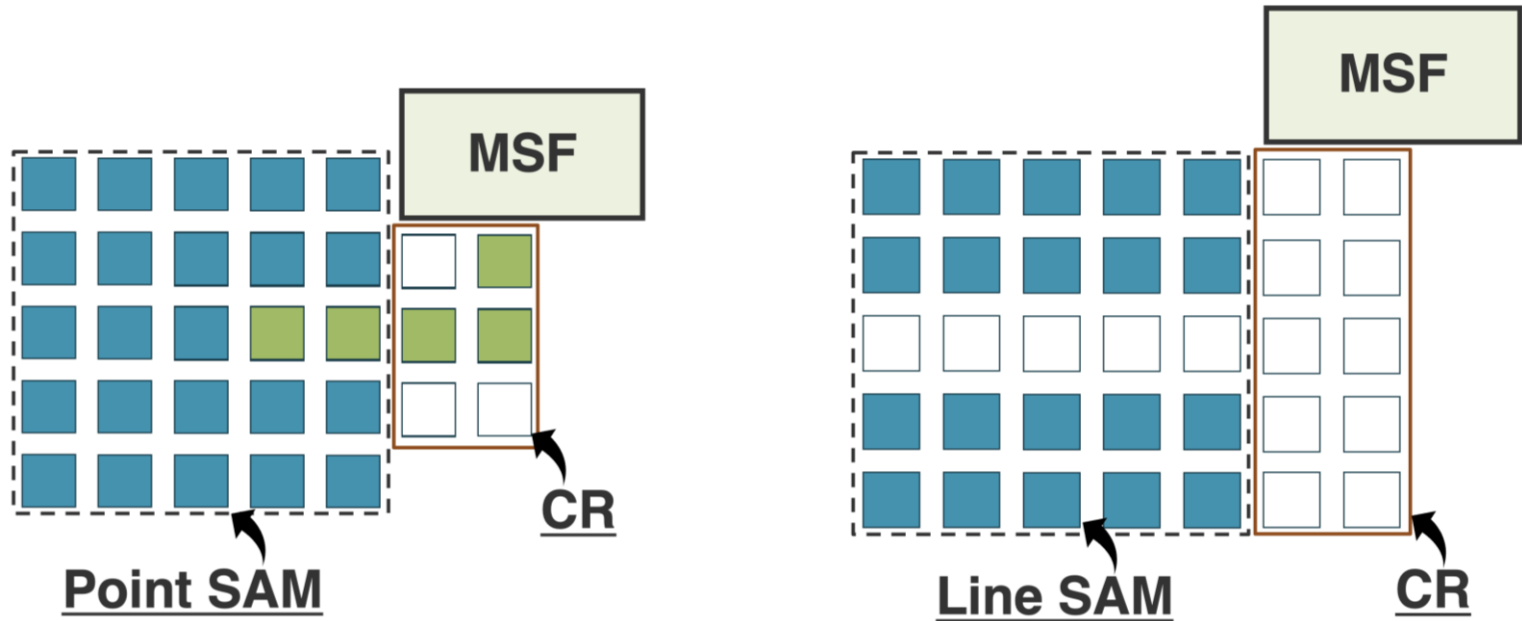
ロードストア型誤り耐性量子計算機アーキテクチャ

- Scan-Access Memory



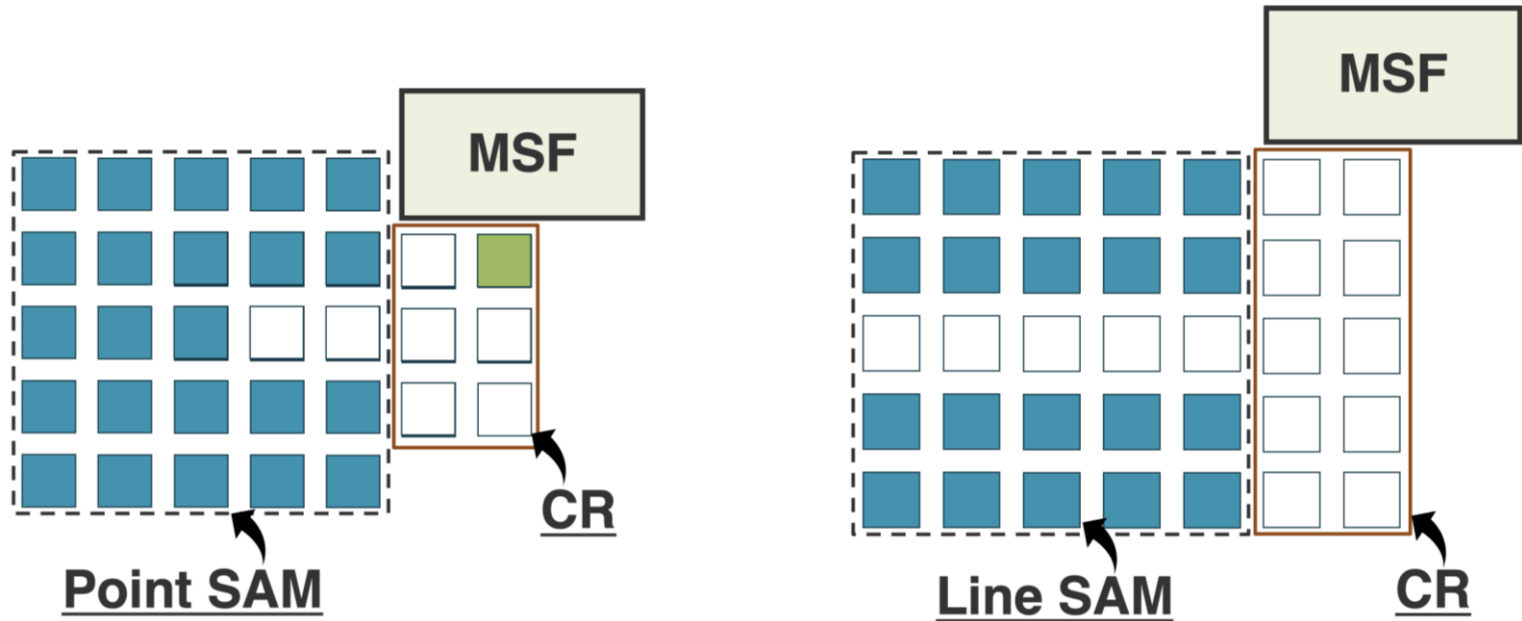
ロードストア型誤り耐性量子計算機アーキテクチャ

- Scan-Access Memory



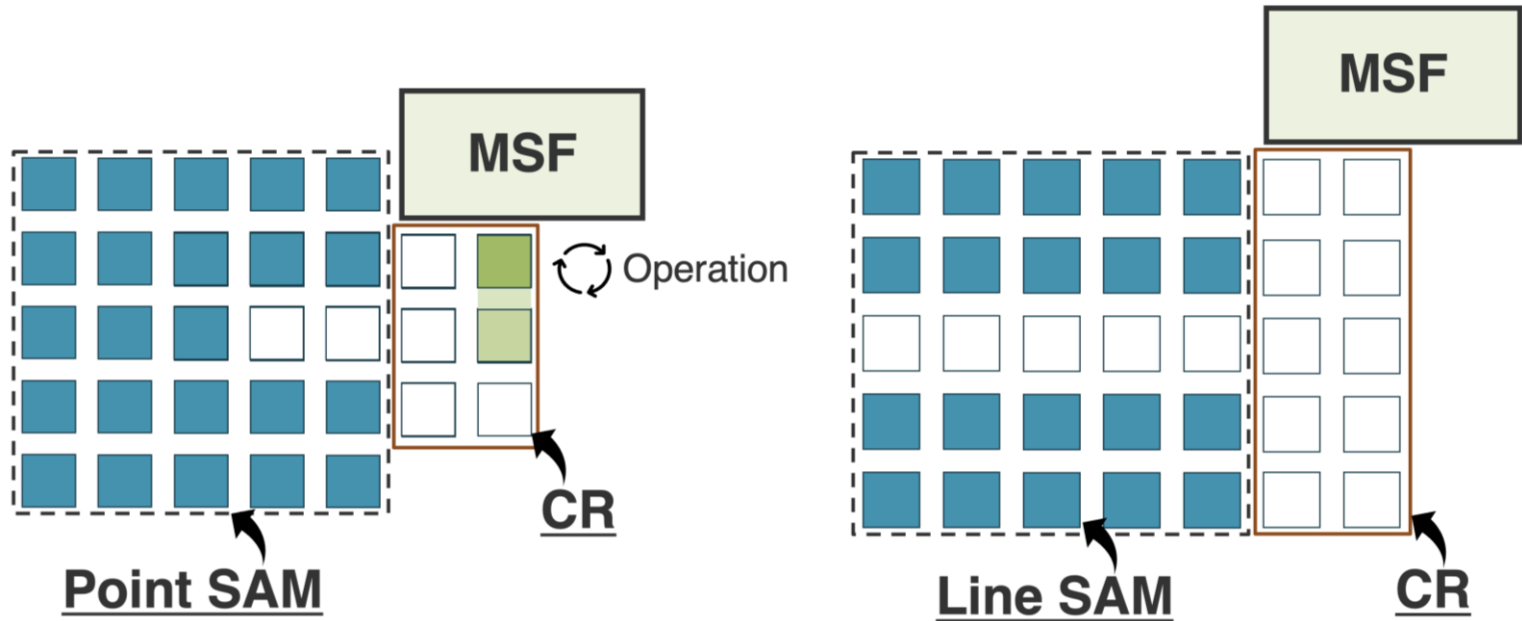
ロードストア型誤り耐性量子計算機アーキテクチャ

- Scan-Access Memory



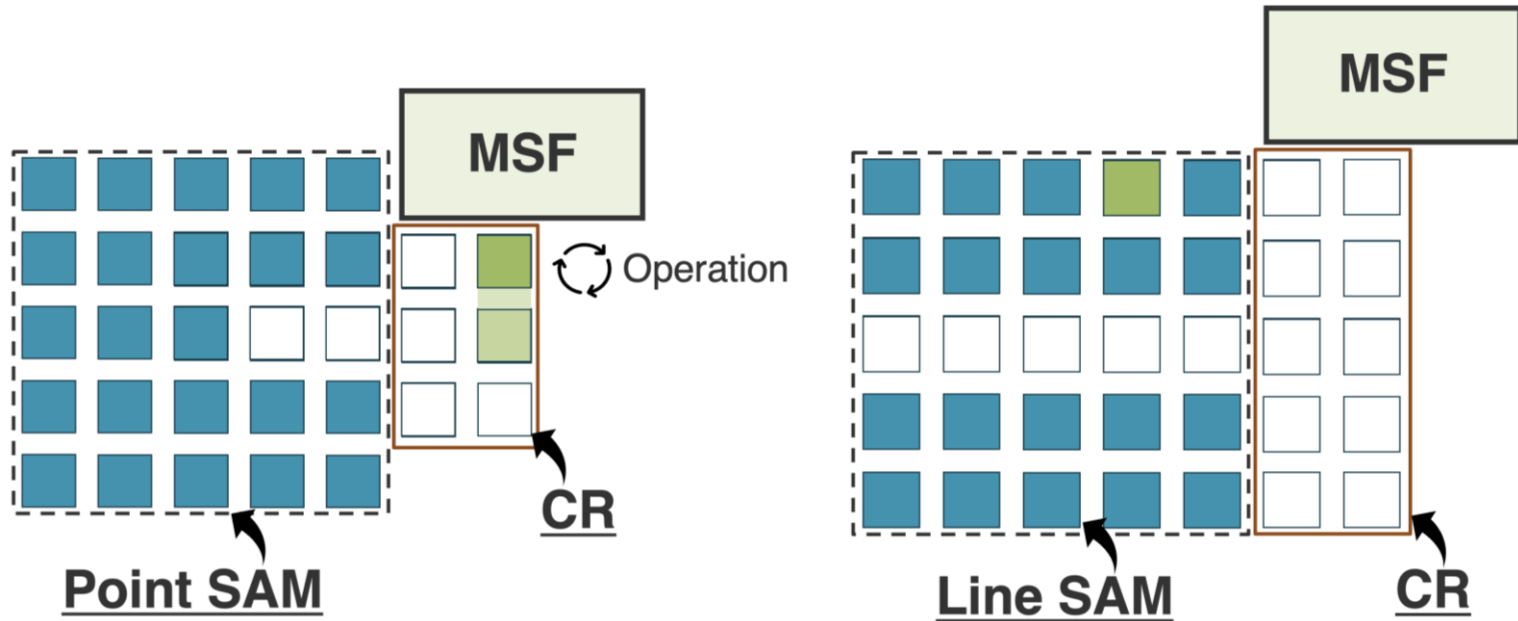
ロードストア型誤り耐性量子計算機アーキテクチャ

- Scan-Access Memory



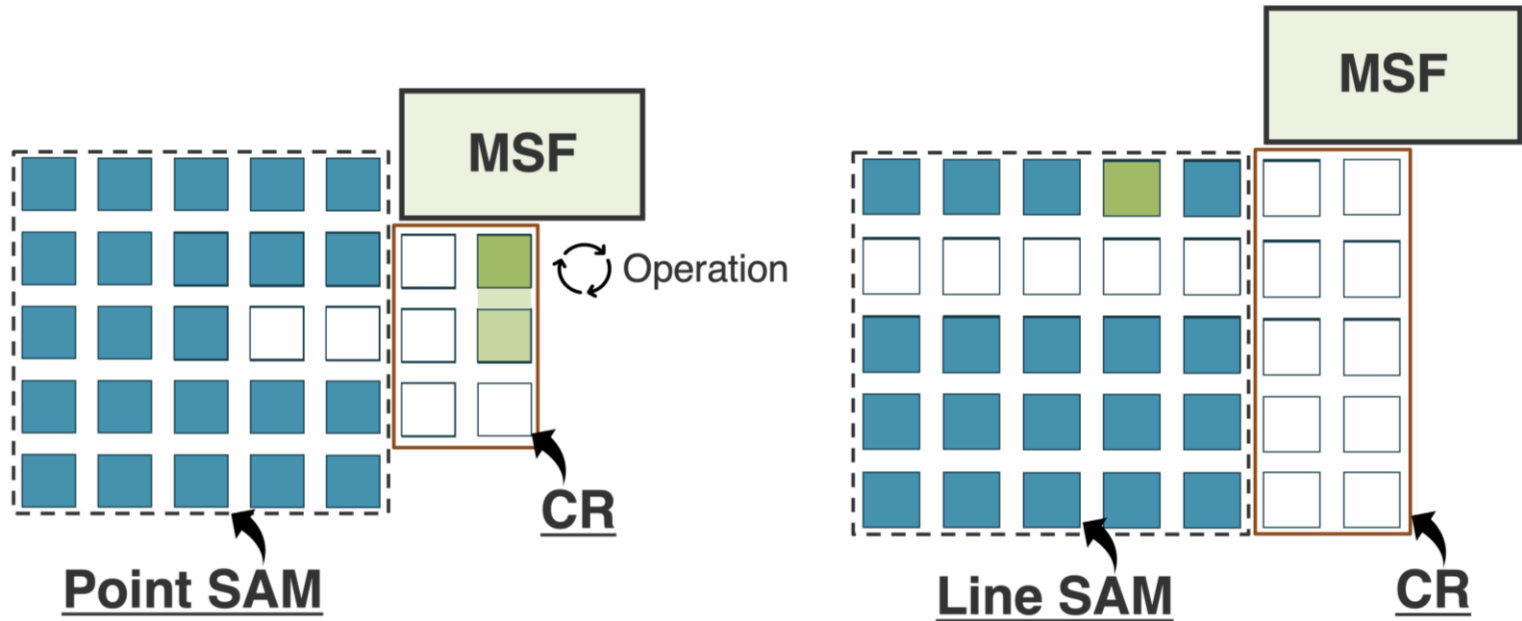
ロードストア型誤り耐性量子計算機アーキテクチャ

- Scan-Access Memory



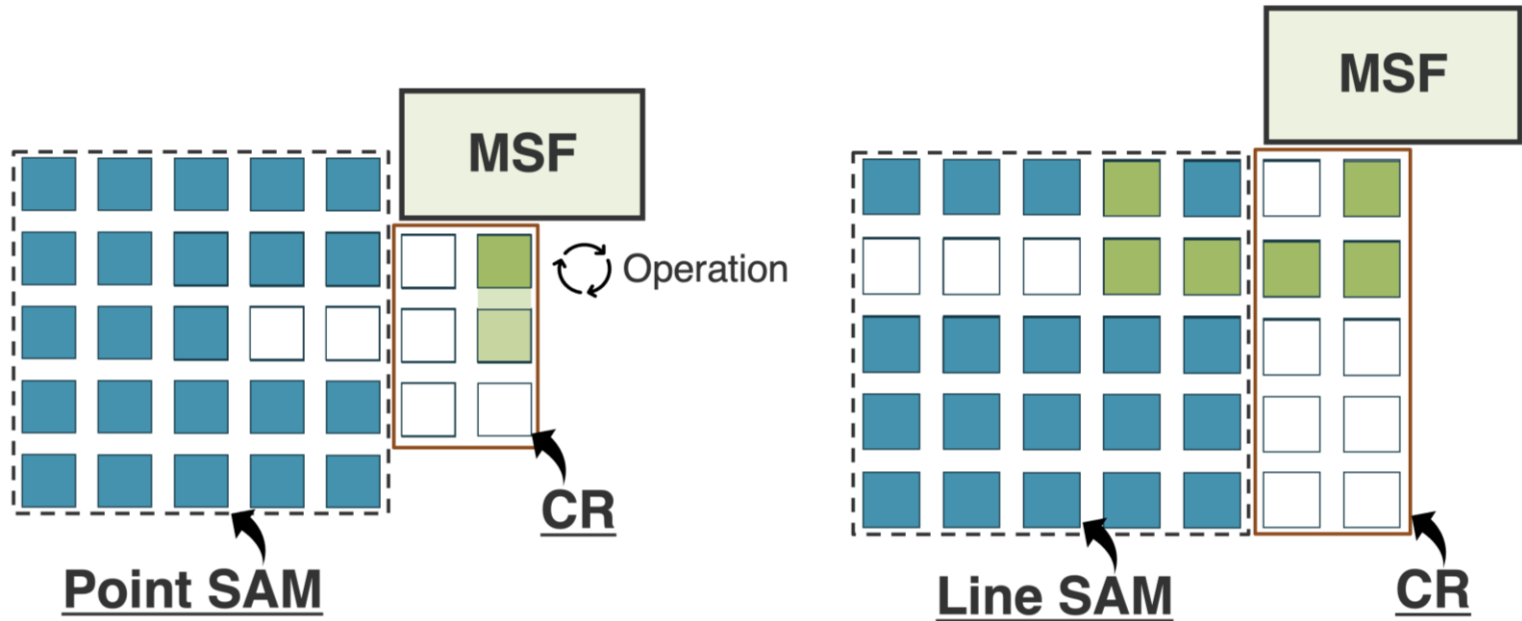
ロードストア型誤り耐性量子計算機アーキテクチャ

- Scan-Access Memory



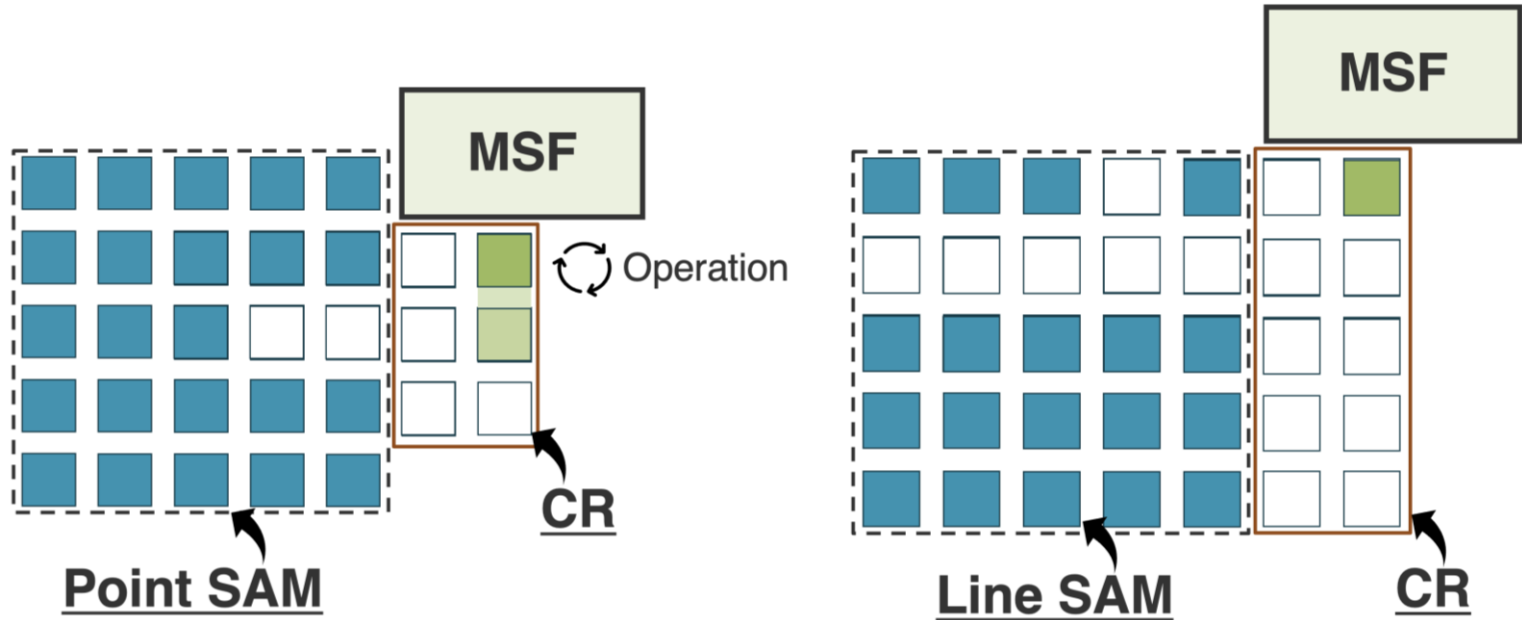
ロードストア型誤り耐性量子計算機アーキテクチャ

- Scan-Access Memory



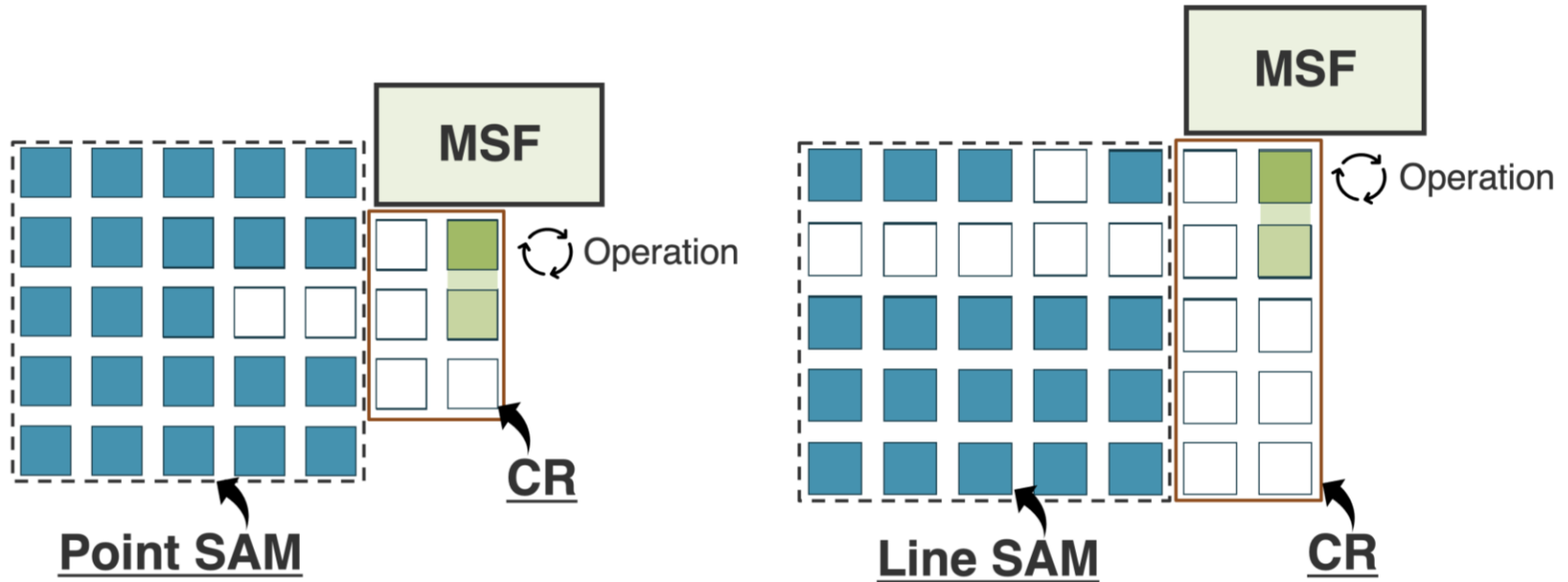
ロードストア型誤り耐性量子計算機アーキテクチャ

- Scan-Access Memory



ロードストア型誤り耐性量子計算機アーキテクチャ

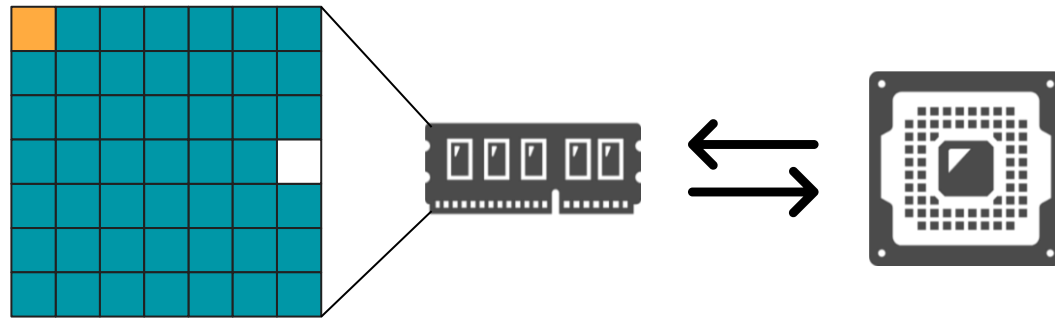
- Scan-Access Memory



ロードストア型FTQCを採用するコスト

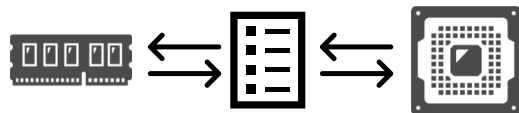
(-) Decreasing performance due to long memory access latency

Example of worst case

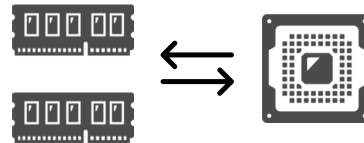


✓ Classical computer uses many techniques for improving memory access latency

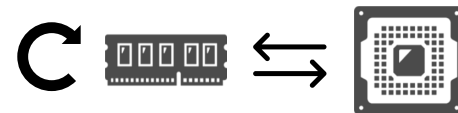
Cache / Scratchpad memory



Multi-channel memory



In-memory computing

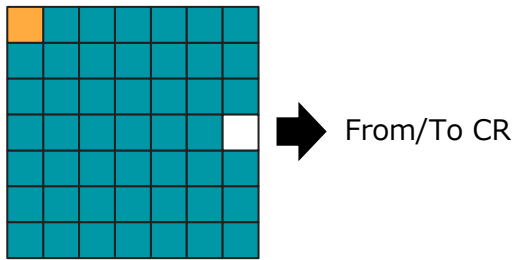
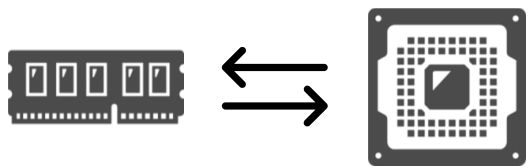


ロードストア型を採用するコスト

ロードストア型の欠点はメモリ-レジスタ間通信による計算の低速化

プログラムによっては通信の遅さがボトルネックに

最悪ケースでは最もアクセスの遅いセルが
繰り返し呼び出される形になる



通常の計算機では典型的プログラムが持つ偏りを活用し様々なテクニックでこれらの問題を解決

キャッシュ / 専用メモリ

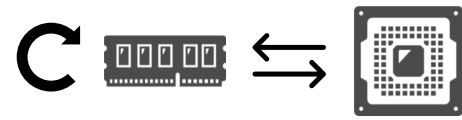
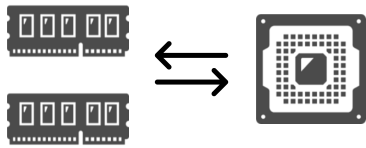
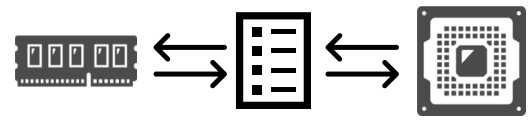
頻繁にアクセスするものや
アクセスが予測されるものを
中規模で高速なメモリに保存する

マルチチャンネルメモリ

複数のメモリバンクに分割し、
並列して読み出しを行うことで
実効的な読み出し速度を上げる

インメモリ計算

メモリ内部に軽量の計算機構を入れ
一部の計算をメモリ内部で行う



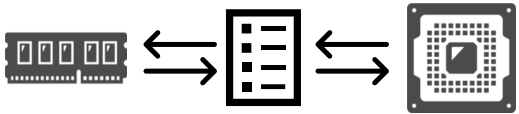
キャッシュなどの技術はメモリアクセスが典型的な特性を持っていることを仮定している 量子も同様の性質を持つか？

ロードストアの影響を軽減する最適化

最適化機構の提案

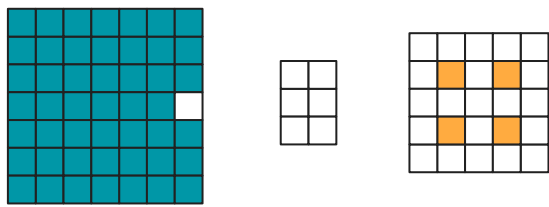
指針：キャッシュ/専用メモリ

頻繁にアクセスするものや
アクセスが予測されるものを
中規模で高速なメモリに保存する



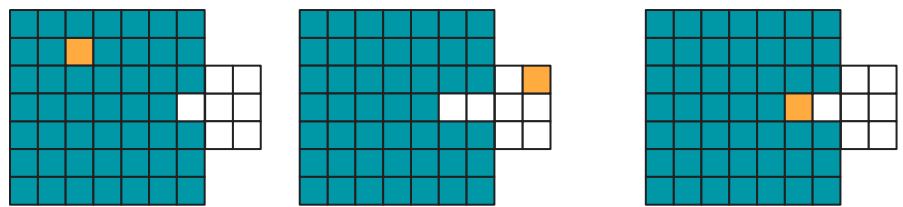
最適化2：Hybrid Floorplan

レジスタのサイズを拡張し、静的解析に基づいて
頻繁にアクセスされる少数のデータを専用領域に配置する

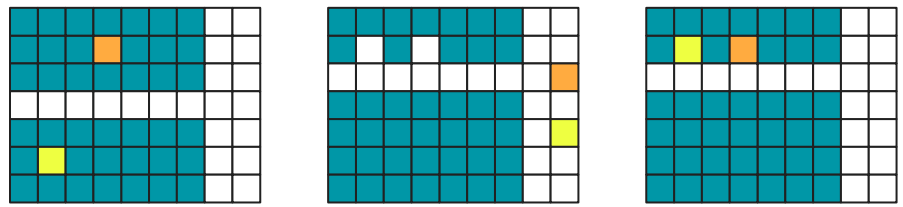


最適化1：Locality-aware store

利用したデータは元の場所ではなくアクセスの早い場所に戻す
これにより、プログラムの知識を用いずに再参照を早くする



同時に利用したデータは同時にアクセスしやすい場所に戻す
これにより、空間的な局所性を活用し典型的ロードを早くする

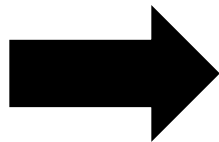
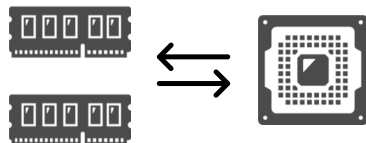


ロードストアの影響を軽減する最適化

最適化機構の提案

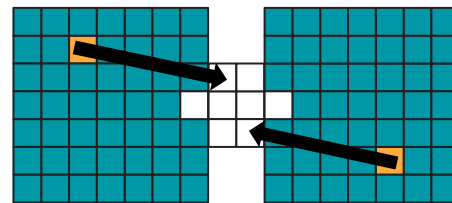
指針 2 : マルチチャンネルメモリ

複数のメモリバンクに分割し、
並列して読み出しを行うことで
実効的な読み出し速度を上げる



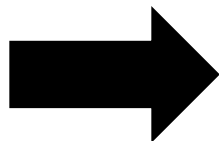
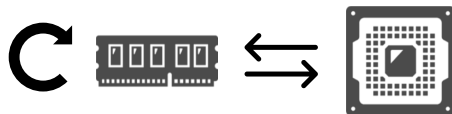
最適化 3 : Multi-bank SAM

データを複数のSAMに分割して配置し、
2並列でのLOAD/STOREを可能にする



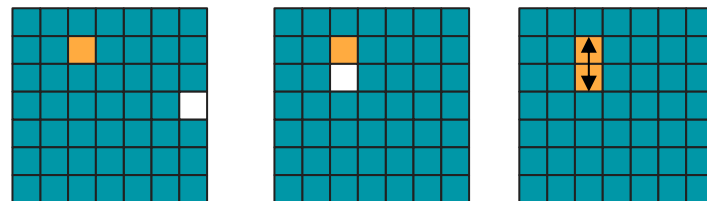
指針 3 : インメモリ計算

メモリ内部に軽量の計算機構を入れ
一部の計算をメモリ内部で行う



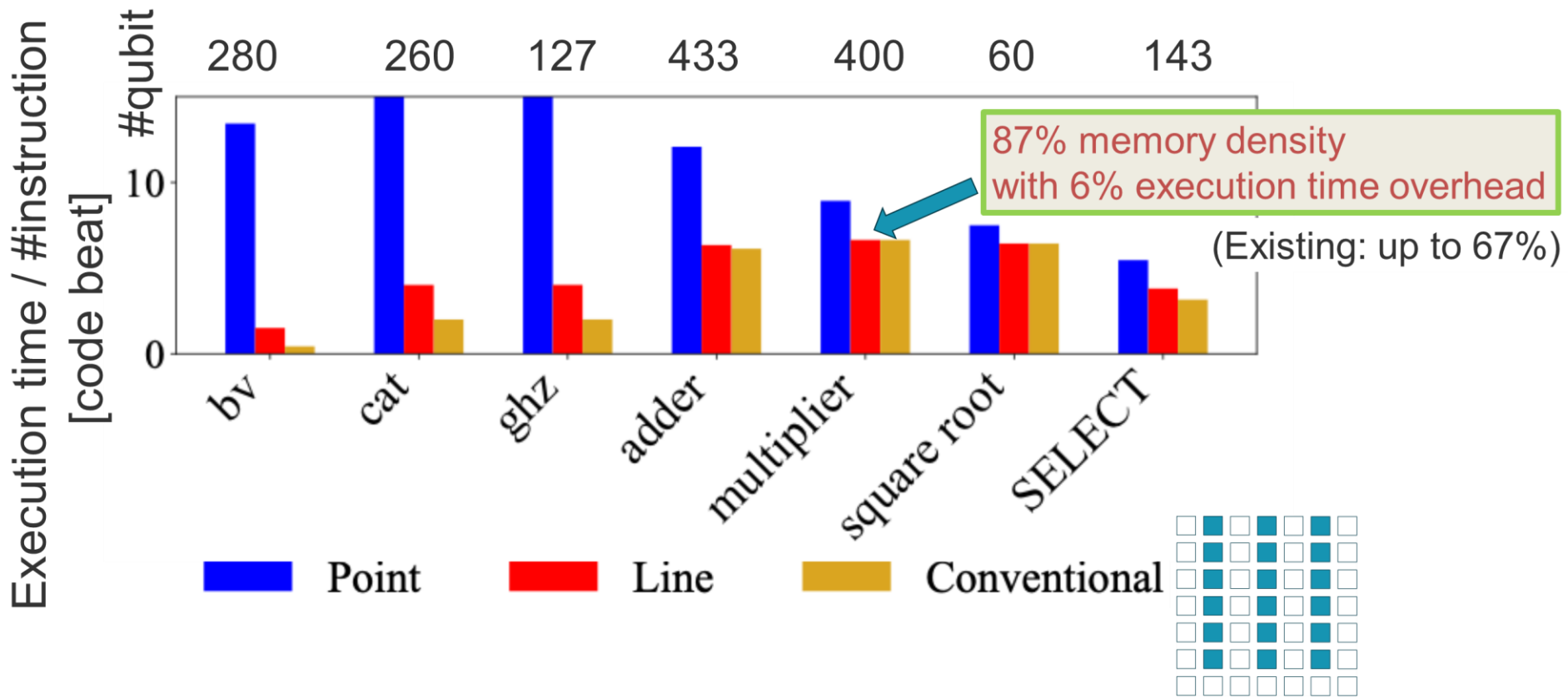
最適化 4 : In-memory operations

移動用の追加セルで完結できる論理操作は、
CRに読みださずにメモリ内部で完結させる

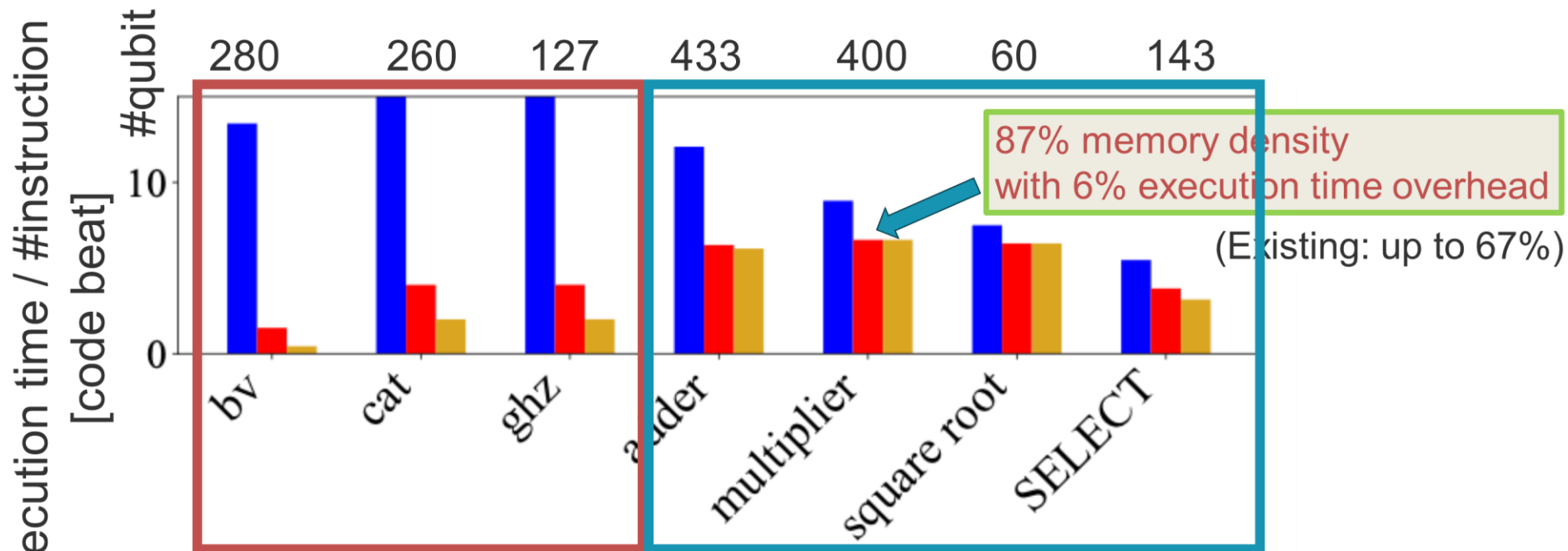


論理HゲートやSゲートをその場で実行

Basic evaluation for benchmark program



Basic evaluation for benchmark program



Large time overhead

- Less T-gates
- Not bottleneck in practice

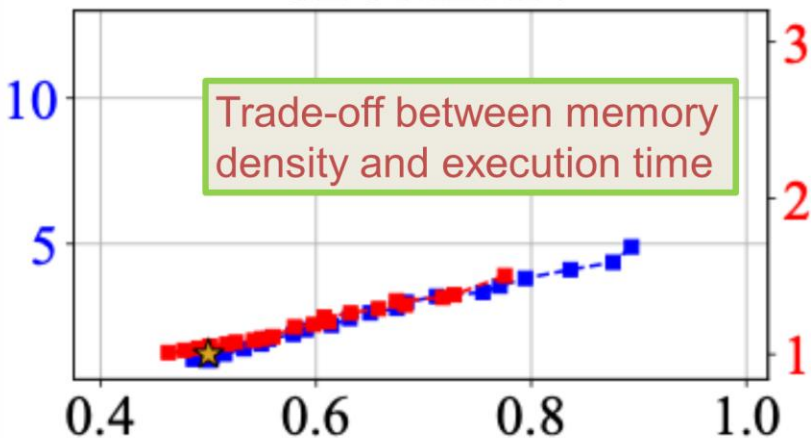
Small time overhead

- Low parallelism and many T-gates
- Bottleneck for many practical applications

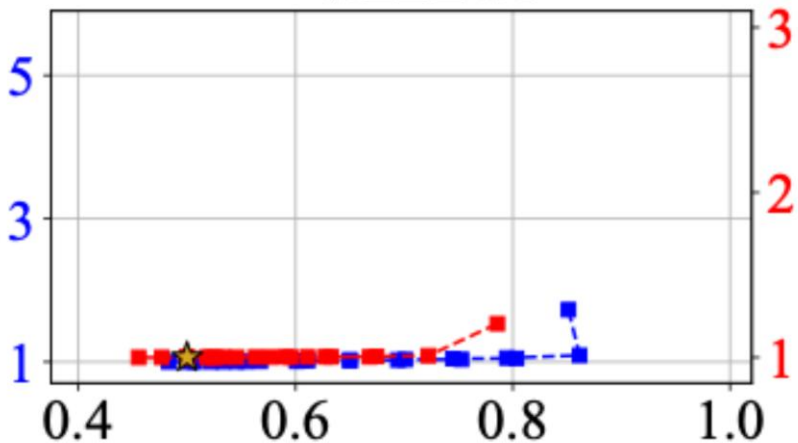
Hybrid floorplan for benchmark program

Execution time overhead

GEOMEAN



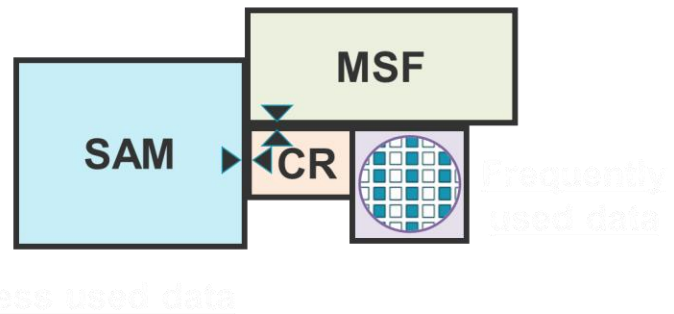
SELECT



Memory density

---●--- Point ---●--- Line ★ Conventional

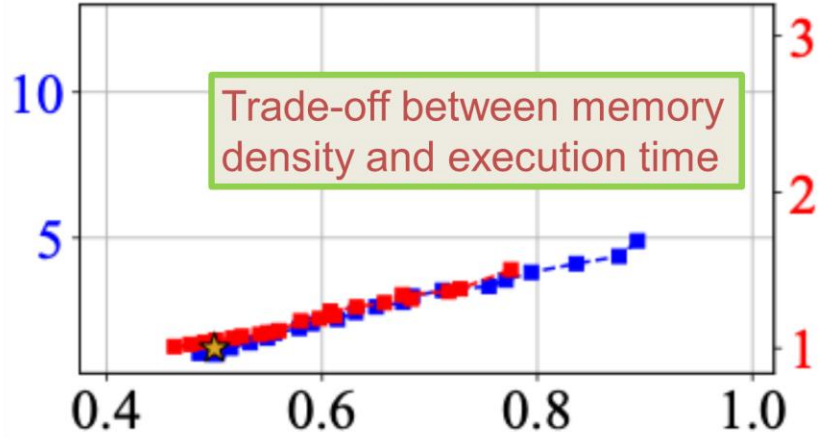
Each Points: how large the conventional floorplan in Hybrid floorplan (Left is larger)



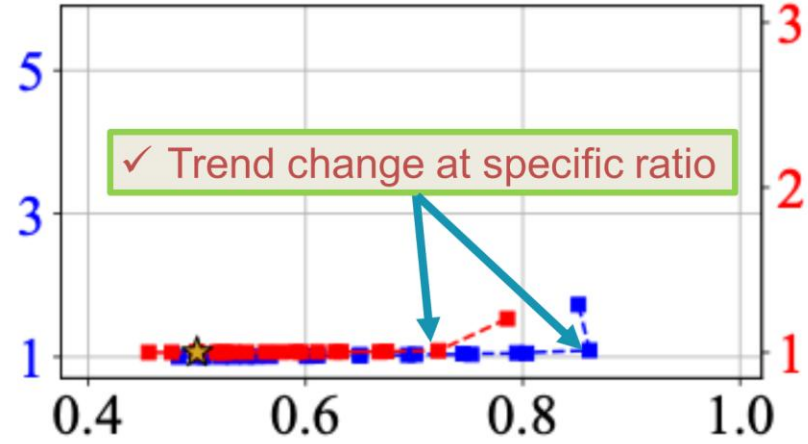
Hybrid floorplan for benchmark program

Execution time overhead

GEOMEAN



SELECT

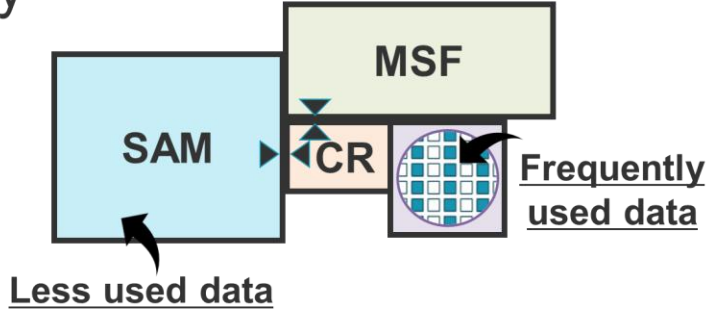


Memory density

---●--- Point -.-.-.- Line ☆ Conventional

Each Points: how large the conventional floorplan in Hybrid floorplan (Left is larger)

Utilize non-uniform access

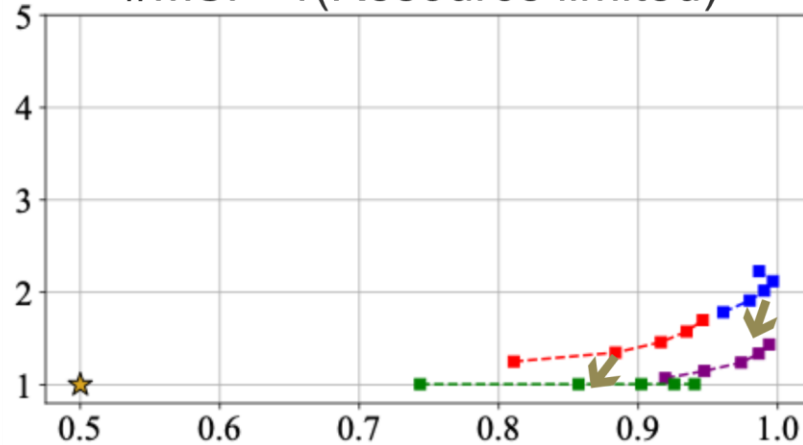


Optimized hybrid floorplan evaluation for SELECT

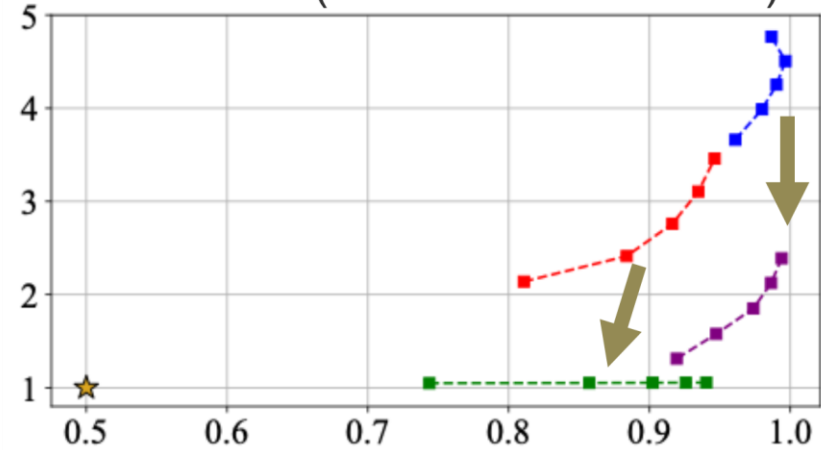
Execution time overhead

Each point: different #Logical qubits=467, 1711, 3753, 6595, 10235 (Right is larger)

#MSF=1(Resource limited)



#MSF=4(Resource abundant)



Memory density

--■-- Point

--■-- Line

--■-- Hybrid Point

--■-- Hybrid Line

★ Conventional

✓ Hybrid floorplan can reduce execution time by slightly decreasing memory density

Hybrid floorplan architecture

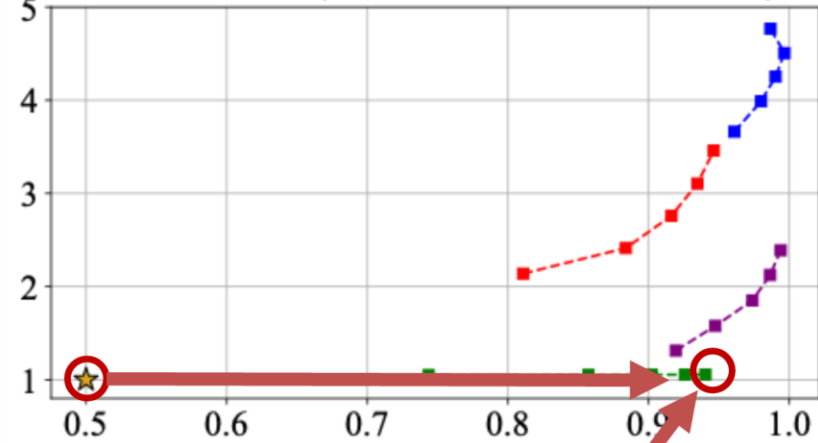
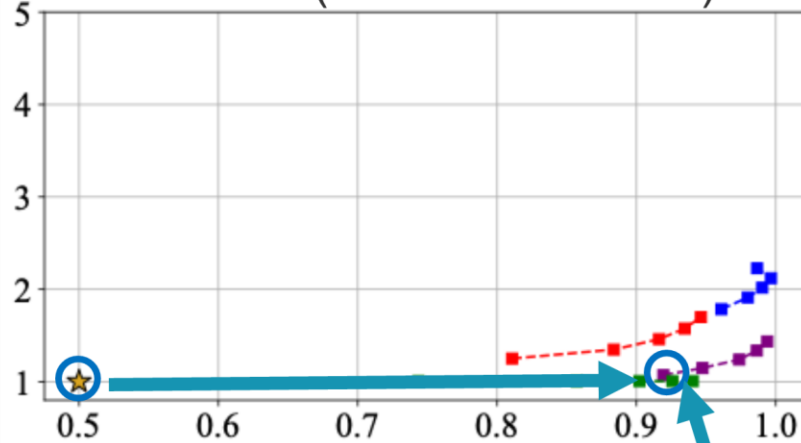
Optimized hybrid floorplan evaluation for SELECT

Execution time overhead

Each point: different #Logical qubits=467, 1711, 3753, 6595, 10235 (Right is larger)

#MSF=1(Resource limited)

#MSF=4(Resource abundant)



Memory density

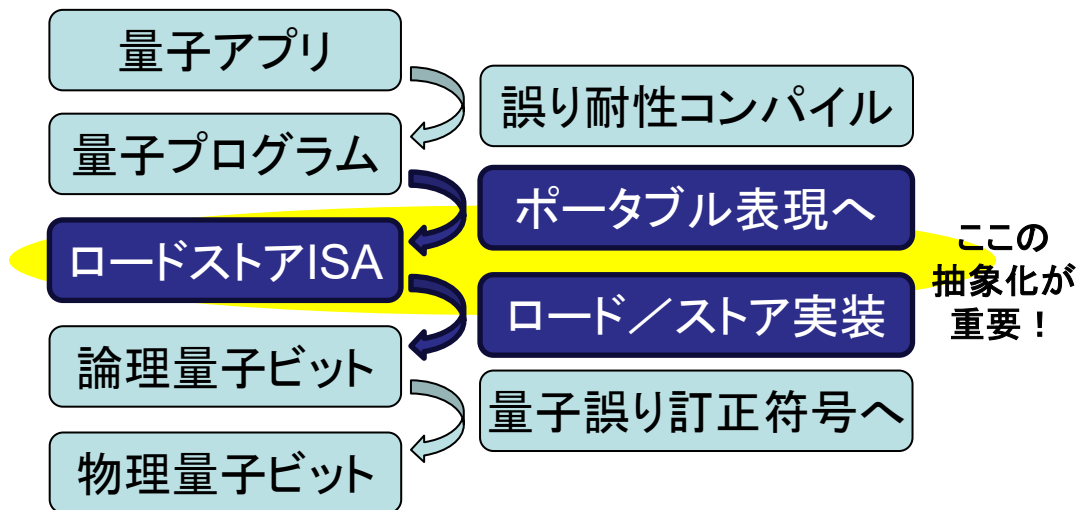
--- Point --- Line --- Hybrid Point --- Hybrid Line ★ Conventional

factory=1, instance size=21 → 92% memory density with 7% execution time overhead by Hybrid point-SAM architecture

factory=4, instance size=101 → 94% memory density with 6% execution time overhead by Hybrid line-SAM architecture

ロードストア型FTQCまとめ

- 高密度メモリ領域＋小さな演算領域の導入で空間効率を向上
 - メモリアクセスレイテンシ軽減手法による時間オーバヘッドの低減
- ロードストア型ISAによる抽象化でプログラムのポータビリティを向上



今日の内容

- 量子計算機アーキテクチャ分野の研究動向
 - 計算機アーキテクチャの研究対象・隣接分野
 - 計算機アーキテクチャ分野における量子計算機関連の研究動向
- 量子誤り訂正とそれを支える計算機アーキテクチャ
 - 表面符号における誤り推定
 - 超伝導デジタル回路を用いた誤り推定機構（DAC2021/HPCA2022）
 - 表面符号＋格子手術を用いた誤り耐性量子計算
 - ロードストア型FTQCアーキテクチャ（HPCA2025）
- まとめ

まとめ

- 計算機アーキテクチャの興味
＝ 計算機の中身、全体の構成、取り巻く環境
- 計算機アーキテクチャの役割
 - 各要素技術の統合、レイヤ分け
 - 理論の要求とハードウェア性能を鑑みてベターな選択肢を探す
 - 計算機としての展望を示す
 - あるべき計算機の姿を想像し、将来生じうる問題を先回りして解決しておく
- 計算機アーキテクチャ分野でも量子計算機はホットなトピック
 - その割には人口が少ないので常に需要あり