量子計算機アーキテクチャ分野の概観

理化学研究所量子コンピュータ研究センター（RQC） 研究員
東京大学大学院情報理工学系研究科システム情報学専攻 特任助教
上野 洋典

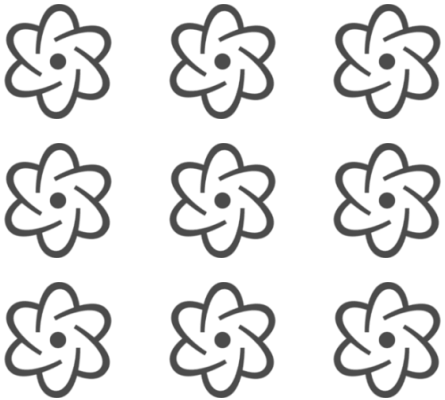
上野 洋典（うえの ようすけ）

- 経歴：
  - 1994年 福岡県北九州市生まれ
  - 2013.4~2017.3: 東大工学部計数工学科システム情報工学コース
  - 2017.4~2022.3: 東大情報理工 システム情報学専攻
    - 博論: 超伝導ディ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
  - 2023.4~2026.3: 理研RQC 基礎科学特別研究員
  - 2024.7~現在:**（兼務）NanoQT Visiting Scientist**
  - 2025.5~現在:**（兼務）東大 情報理工 特任助教（非常勤）**
  - 2026.4~現在:**（本務）理研量子コンピュータ研究センター 研究員（鈴木チーム）**
- 研究対象：量子計算機アーキテクチャ、超伝導ディジタル回路
- 趣味：ラジオ、テニス、演劇
- 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!



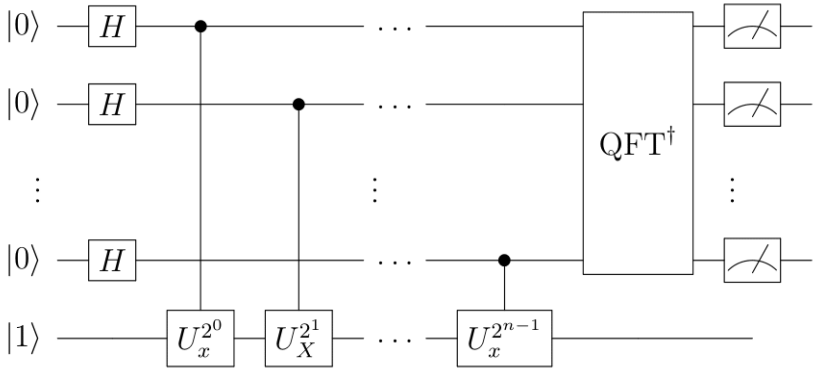
ドイツでの最初と最後のビール

量子ビット/量子アルゴリズムができれば量子計算機は完成か？



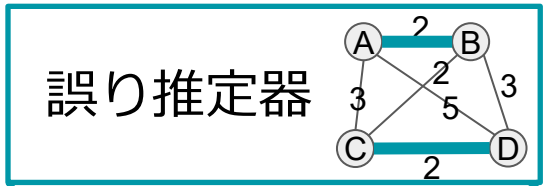
量子ビット

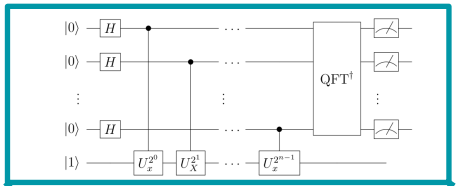
?



量子アルゴリズム

量子計算機システム





量子ビット
（超伝導、原子、光パルスなど）

制御回路
（FPGAやASIC）

OS/ドライバ
（ソフトウェア）

```
entry:
  CX @q1 @q2
  Meas @q1 @r1
  Jump @r1 %br1
```

実行形式
（ソフトウェア）

```
for a in 0..10{
  func(a);
}
```

プログラム
（ソフトウェア）

社会需要

シミュレータ
（ソフトウェア）

キャリブレータ
（ソフトウェア）

参照実装/テスト
（ソフトウェア）

プロファイラ
（ソフトウェア）

コンパイラ/デバッガ
（ソフトウェア）

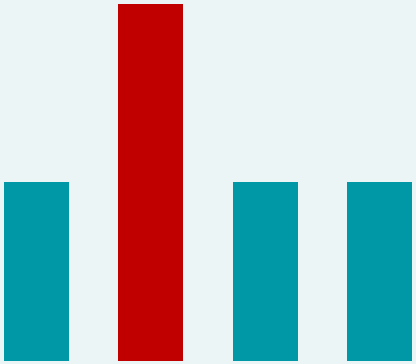
SDK/開発環境
（ソフトウェア）

理研鈴木さんの資料より引用・一部改変

計算機アーキテクチャの役割

① 計算機システム全体を見る

② ボトルネックを見つける



③ 境界を設計する

- アプリ
- アルゴリズム
- プログラミング言語
- アセンブリ/機械語
- **命令セットアーキ**
- **マイクロアーキ**
- ゲート/レジスタ
- デバイス
- 物理

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
- 量子計算機アーキテクチャに関連する研究紹介
  - 超伝導ディジタル回路を用いた量子誤り推定機構（博論、DAC'21、HPCA'22）
  - 誤り耐性量子計算（FTQC）におけるロードストア型アーキテクチャ（HPCA'25）
  - 未出版内容のため公開版では削除
  - 中性原子を対象としたフルスタックFTQCシミュレーションフレームワーク（MICRO'26）
- 量子計算機アーキテクチャ研究の魅力・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
抽象度
低

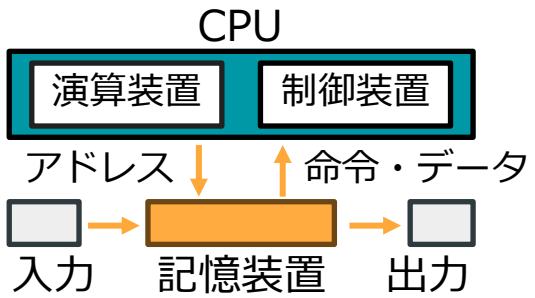
計算機全体の構成
計算機を取り巻く環境

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

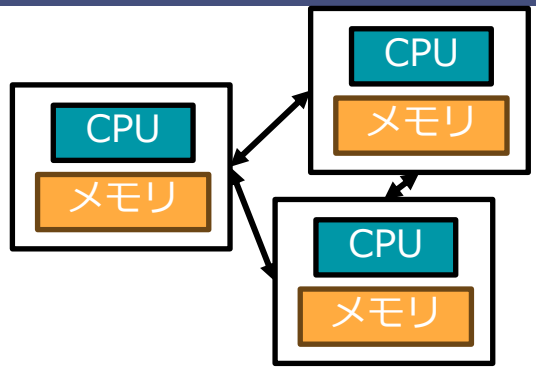
命令セットアーキテクチャ（ISA）
RISC, CISC, VLIW

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

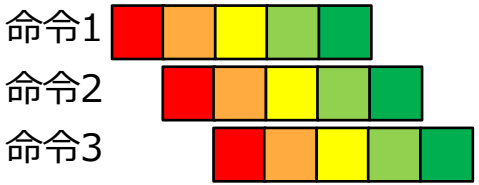


ノイマン型アーキテクチャ

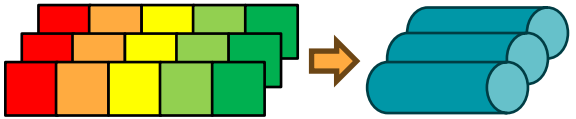


分散システム

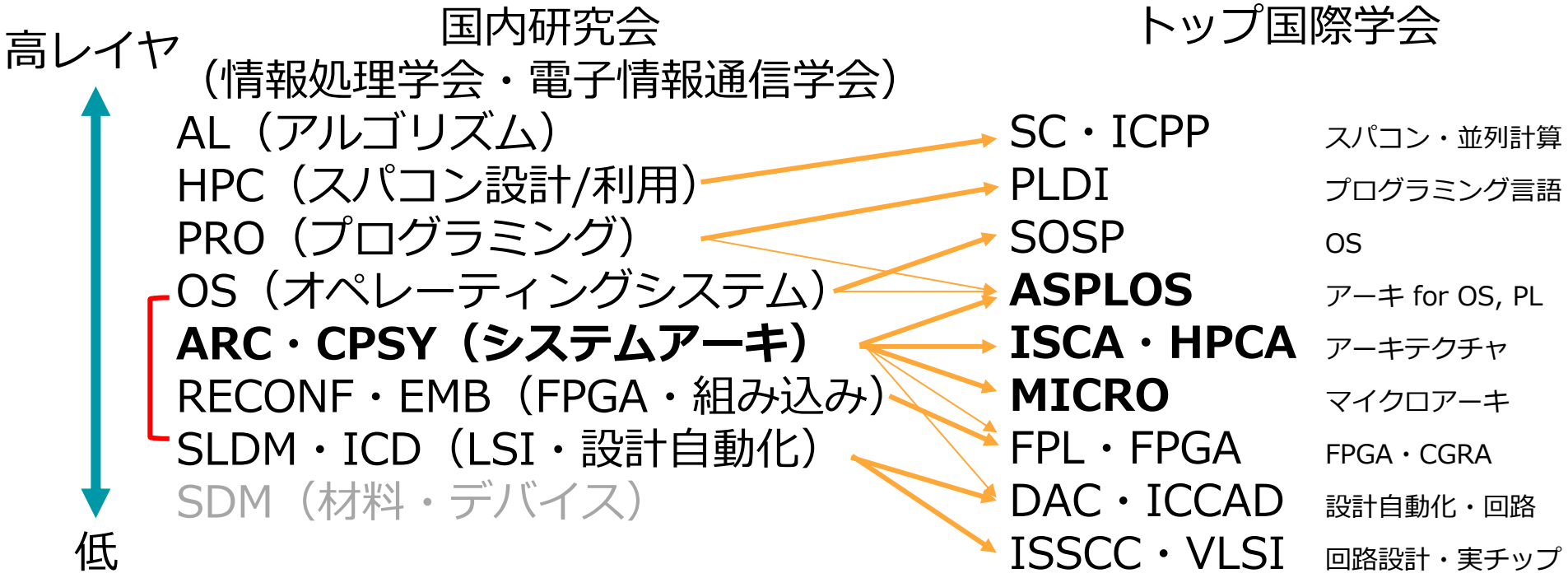
x86 (CISC) vs. RISC-V (RISC)



パイプライン


ベクトル演算器

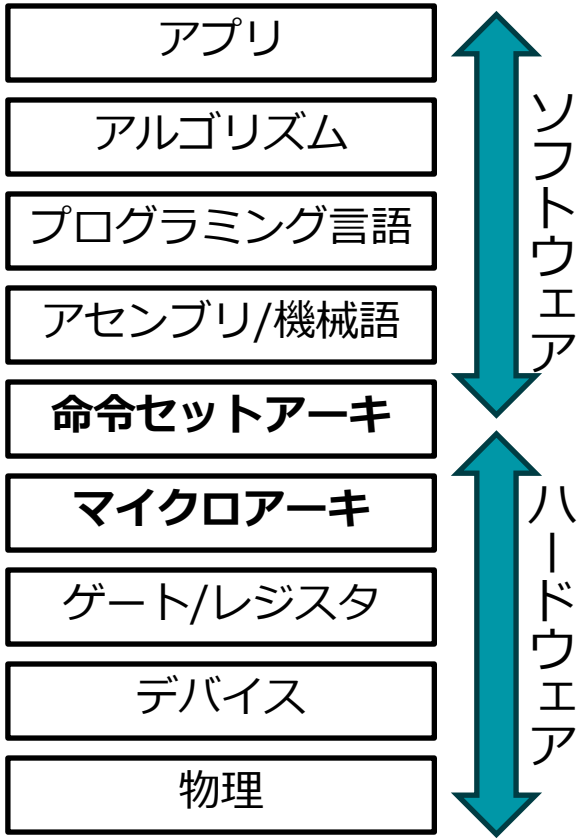
計算機アーキテクチャの隣接分野



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の設計を更新
- 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



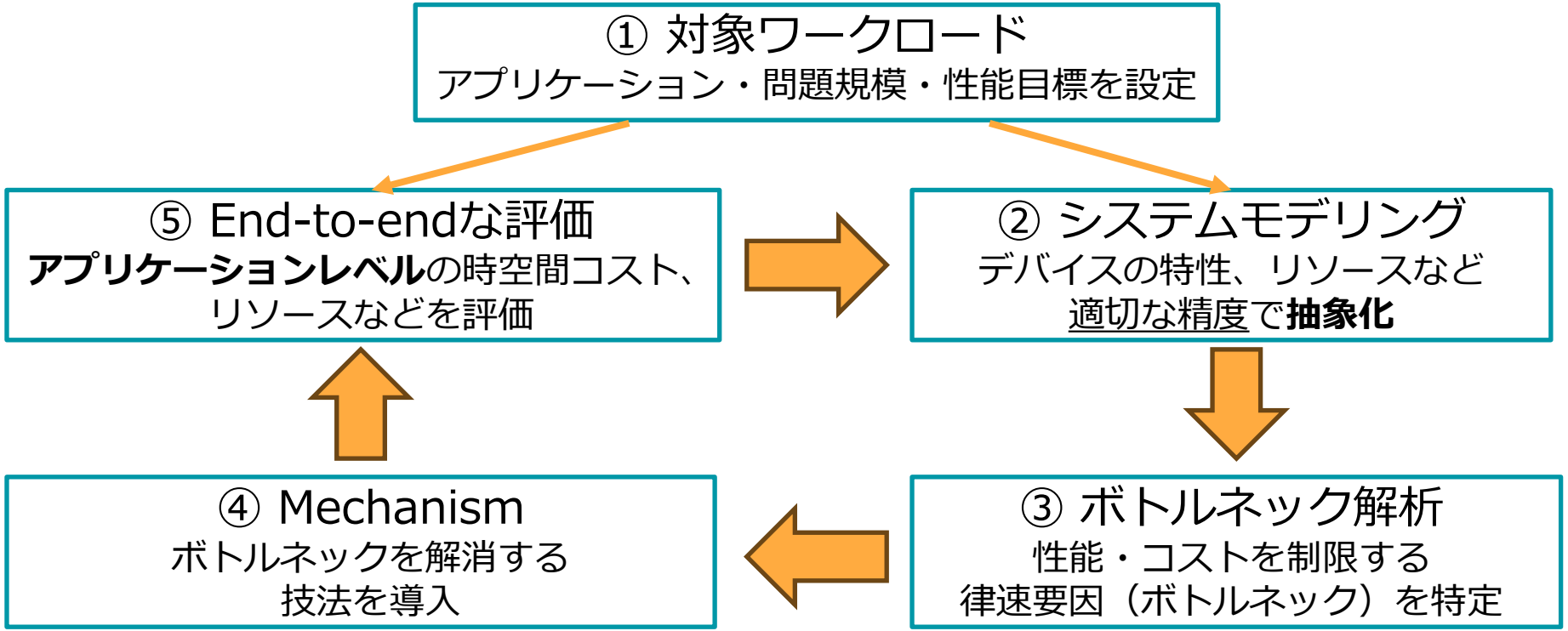
計算機アーキテクチャ研究の特徴

- 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

- 問題解決のアプローチ：**適切な**抽象化・モデル化

計算機アーキテクチャ研究の型



「モデル化→ボトルネック解析→ボトルネックの解消→評価」を反復する

今日の内容

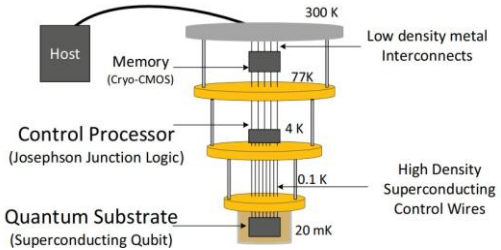
- 量子計算機アーキテクチャ分野の研究動向
  - 計算機アーキテクチャの研究対象・隣接分野
  - 計算機アーキテクチャ分野における量子計算機関連の研究動向
- 量子計算機アーキテクチャに関連する研究紹介
  - 超伝導ディジタル回路を用いた量子誤り推定機構（博論、DAC'21、HPCA'22）
  - 誤り耐性量子計算（FTQC）におけるロードストア型アーキテクチャ（HPCA'25）
  - 未出版内容のため公開版では削除
  - 中性原子を対象としたフルスタックFTQCシミュレーションフレームワーク（MICRO'26）
- 量子計算機アーキテクチャ研究の魅力・まとめ

量子計算機アーキテクチャ分野

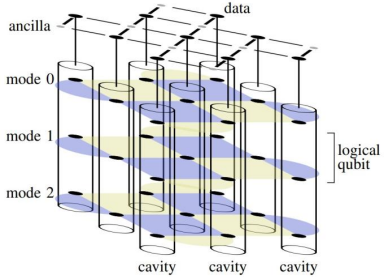
システムアーキテクチャ

命令セットアーキテクチャ（ISA）

マイクロアーキテクチャ



Cryogenic QEC architecture
MICRO2017

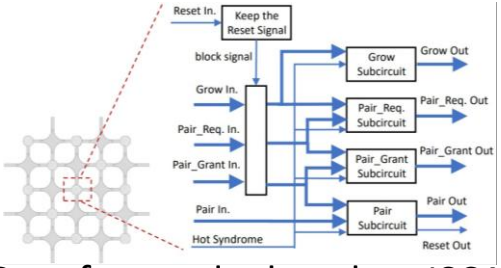


Virtualized Logical Qubit
MICRO2020

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

Type	Syntax	Description
Control	`CMP Rs, Rt`	(**CoMPare** GPR `Rs` and `Rt` and store the result into the comparison flags.
	`BR <Comp. Flag>, Offset`	(**BRanch**) Jump to `PC + Offset` if the specified comparison flag is '1'.
Data Transfer	`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).
	`LDUI Rd, Imm, Rs`	(**LoaD Unsigned Immediate**) Rd = Imm[14..0]::Rs[16..0].
	`LD Rd, Rt(Imm)`	(**LoaD** from memory) Load data from memory address `Rt + Imm` into GPR `Rd`.
	`ST Rs, Rt(Imm)`	(**STore** to memory) Store the value of GPR `Rs` in memory address `Rt + Imm`.
	`FMR Rd, Qi`	(**Fetch Measurement Result**) Fetch the result of the last measurement instruction on qubit `i` into GPR `Rd`.
Logical	`AND/OR/XOR Rd, Rs, Rt` `NOT Rd, Rt`	Logical and, or, exclusive or, not.
Arithmetic	`ADD/SUB Rd, Rs, Rt`	Addition and subtraction.
Waiting	`QWAIT Imm` `QWAITR Rs`	(**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 `Rs`.
Target Specify	`SMIS Sd, <Qubit List>` `SMIT Td, <Qubit Pair List>`	(**Set Mask Immediate** for Single-/Two-qubit operations) Update the single- (two-)qubit operation target register `Sd` (`Td`).
Q. Bundle	`[PI,] Q_Op [| Q_Op]*`	Applying operations on qubits after waiting for a small number of cycles indicated by `PI`.

eQASM, HPCA2019

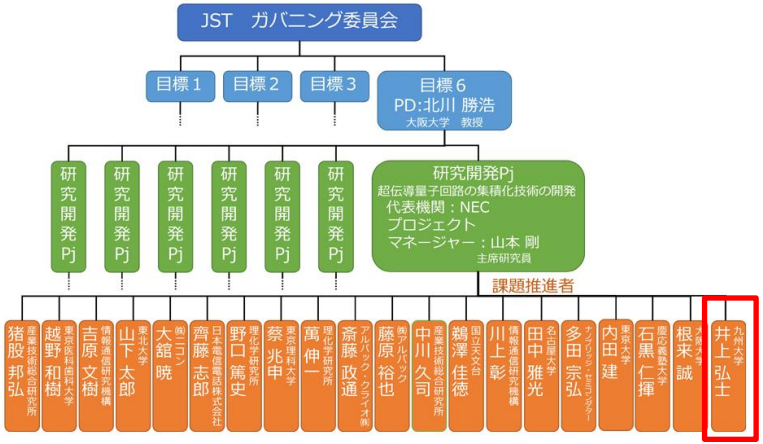


SFQ surface code decoder , ISCA2020

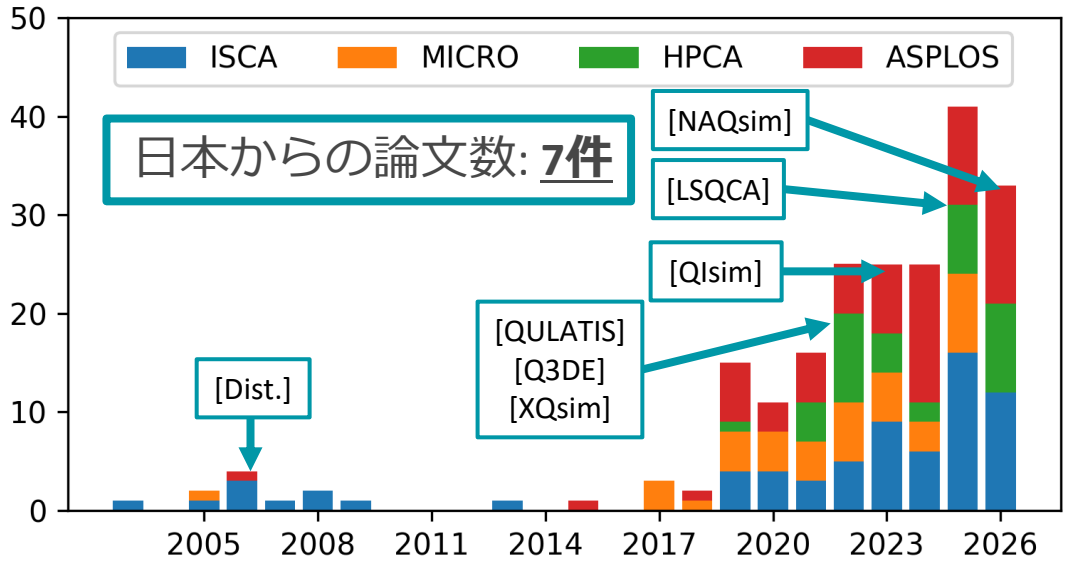
K. Bertels et al., “eQASM: An Executable Quantum Instruction Set Architecture,” in HPCA2019.
C. Duckering et al., “Virtualized Logical Qubits: A 2.5 D Architecture for Error-Corrected Quantum Computing,” in MICRO2020.
S. Tannu et al., “Taming the Instruction Bandwidth of Quantum Computers via Hardware-Managed 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



アーキ系のトップ国際会議（ISCA, MICRO, HPCA, ASPLOS）における量子計算機関連論文数（2003～2026年）

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)
2026	7.6% (33/434) MICRO除く

[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**.
[QIsim] D. Min, …, M. Tanaka, K. Inoue, J. Kim, “QIsim: 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.**
[NAQsim] **Y. Ueno**, S. Sunami, T. Hinokuma, Y. Suzuki, A. Goban, H. Yamasaki, T. Tanimoto, I. Byun “NAQsim: Full-Stack Architecture Simulation Framework for Fast and Space-Efficient Neutral Atom Quantum Computing”, **MICRO2026**.

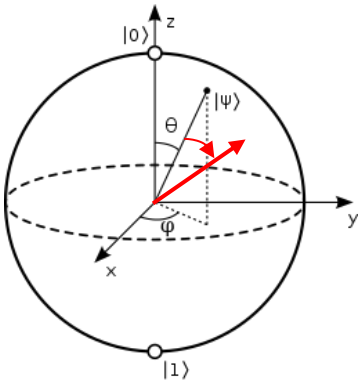
量子計算機アーキテクチャ研究動向まとめ

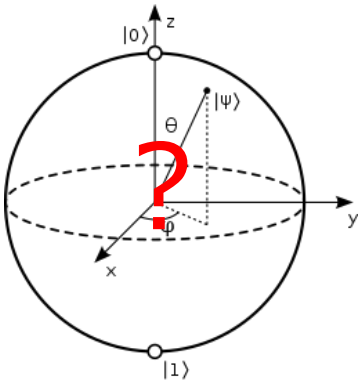
- 計算機アーキテクチャ
  ＝計算機の中身、全体の構成、取り巻く環境
  - 主な役割：各要素技術の統合、レイヤ分け、計算機としての展望を示す
- 計算機アーキテクチャ分野でも量子計算機はインパクト大
  - 物理（量子情報）系＋計算機系研究者の共同研究が重要
- 海外では量子計算機アーキテクチャの研究が盛ん、
  国内はまだこれから

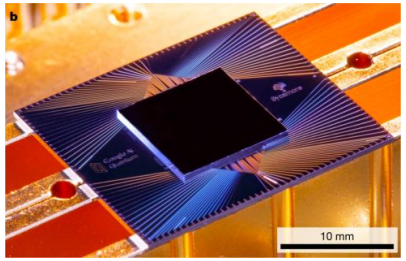
今日の内容

- 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
- 量子計算機アーキテクチャに関連する研究紹介
  - 超伝導ディジタル回路を用いた量子誤り推定機構（博論、DAC'21、HPCA'22）
  - 誤り耐性量子計算（FTQC）におけるロードストア型アーキテクチャ（HPCA'25）
  - 未出版内容のため公開版では削除
  - 中性原子を対象としたフルスタックFTQCシミュレーションフレームワーク（MICRO'26）
- 量子計算機アーキテクチャ研究の魅力・まとめ

量子誤り訂正符号に求められる条件







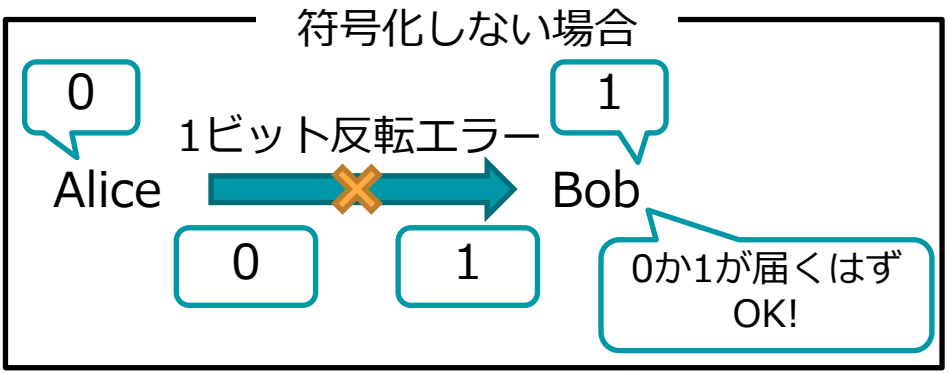
[3] より

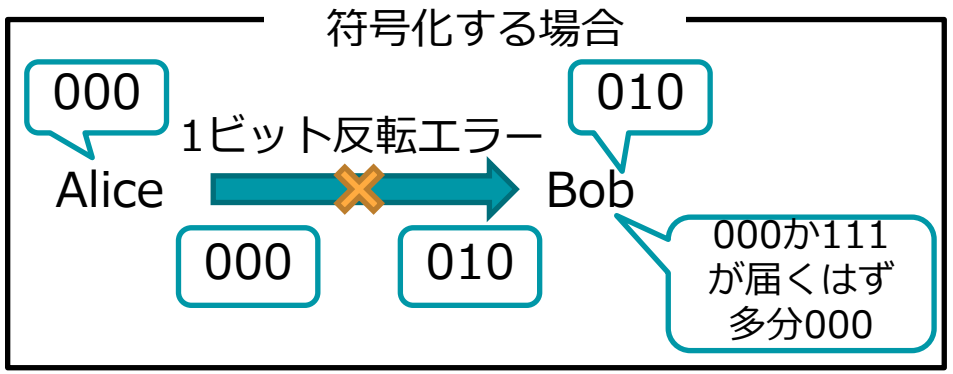
- 量子ビット数: 100~1000
- エラーレート: 0.1~1%

- 連続的な情報に生じたエラーを訂正できる
- 直接状態を観測することなく訂正できる
- 実験的に達成できる誤り率を超えるしきい値を持つ

[3] Frank Arute, Kunal Arya, Ryan Babbush, et al. Quantum supremacy using a programmable superconducting processor. Nature 574, 505–510 (2019).

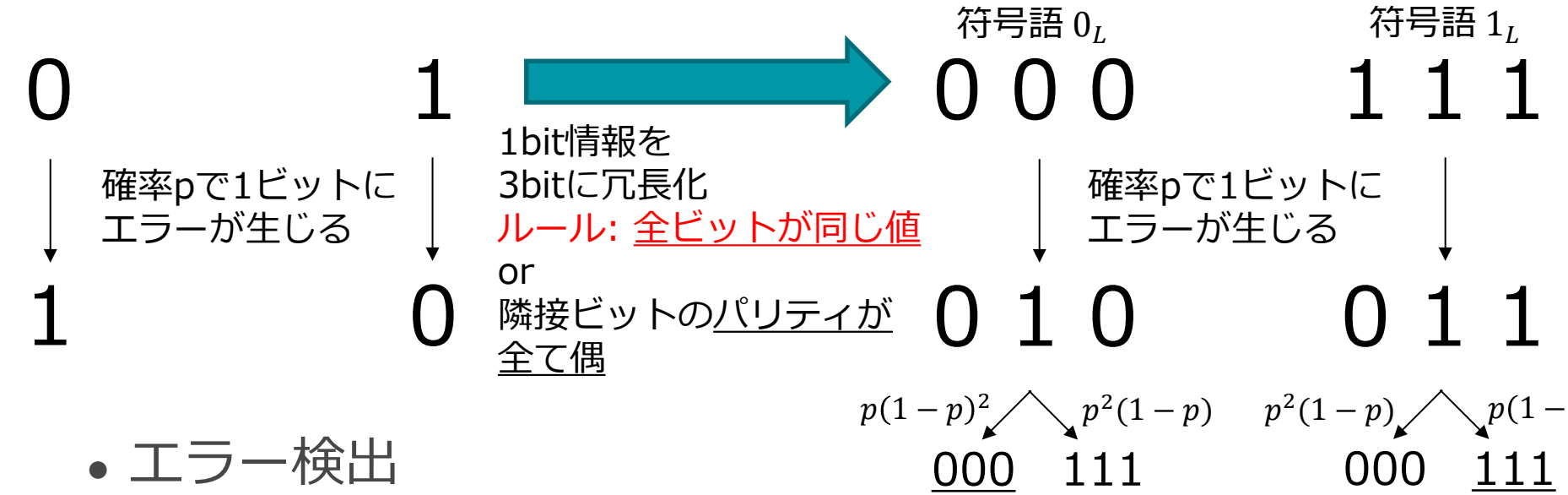
誤り訂正符号の役割





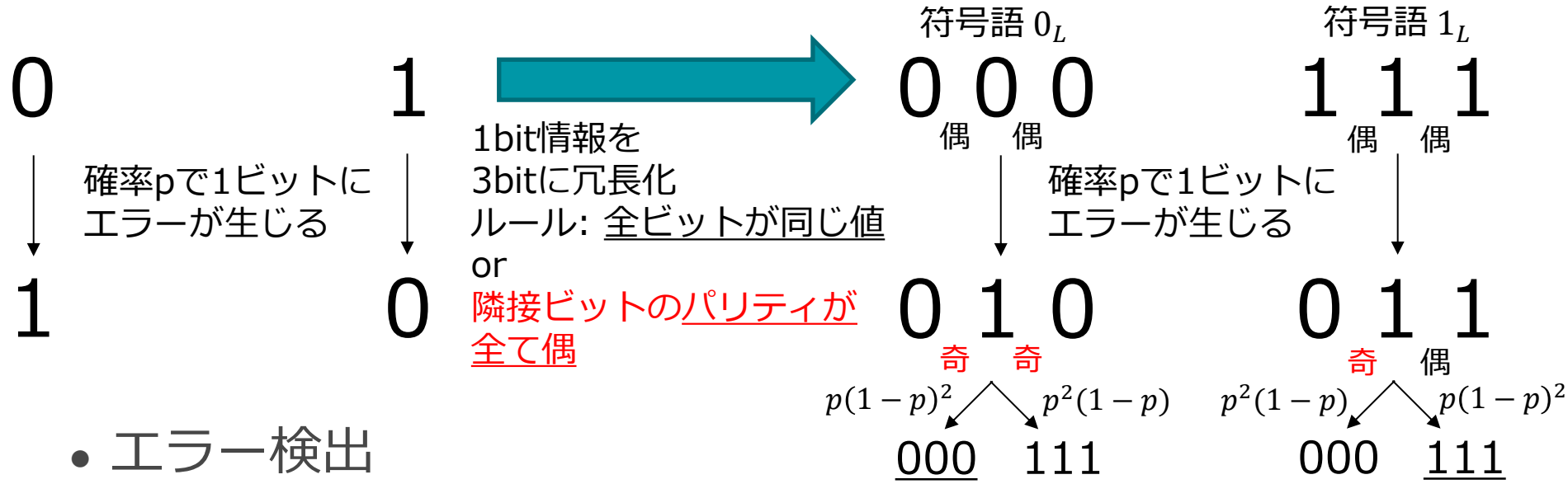
- 誤り訂正符号: 元データを冗長に表現して
  誤りの検出・訂正を可能にする符号
- 符号パラメータ$[n, k, d]$
  - n: 符号長（どの程度冗長に表現するか）
  - k: 情報数（元データのビット数）
  - d: 符号距離（論理操作を行う際に触る必要のある最低ビット数）
    - $\lfloor (d-1)/2 \rfloor$ビットまでは訂正可能、$d-1$個までは検出可能

誤り訂正符号 古典の場合



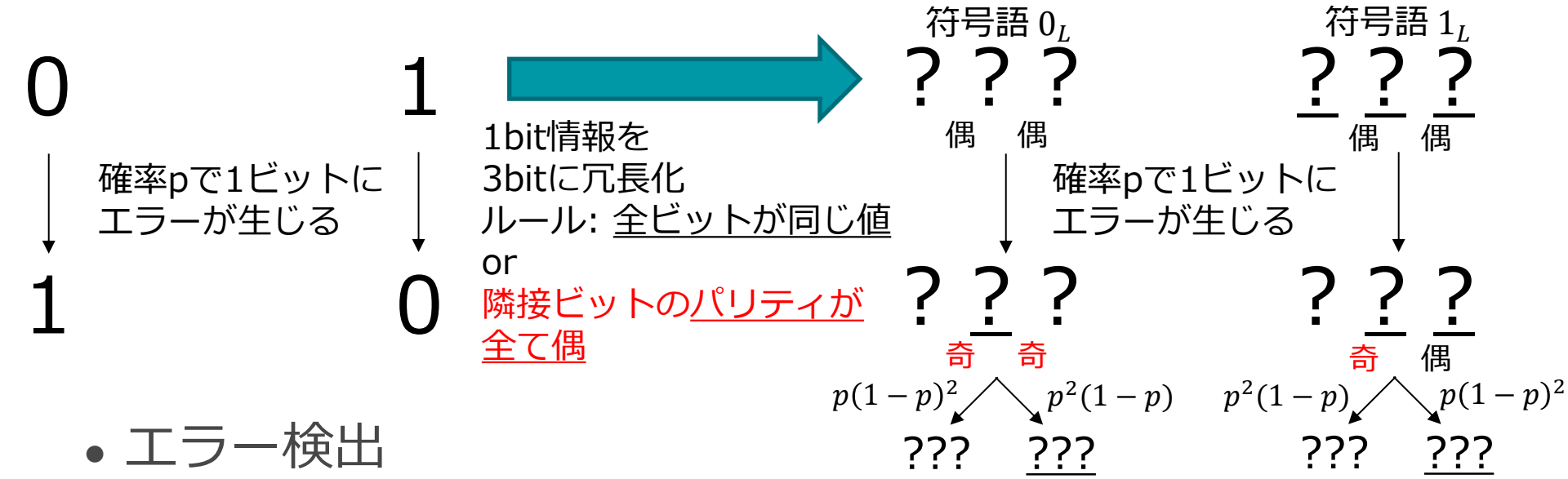
- エラー検出
  - 符号を構成する各ビットをチェックしてすべて一致しているかどうか
  - 隣接ビットのパリティがすべて偶であるか <- 量子でも可能なアプローチ
- エラー訂正（復号）
  - パリティ検査の結果から最も近い（確率の高い）符号語を特定

誤り訂正符号 古典の場合



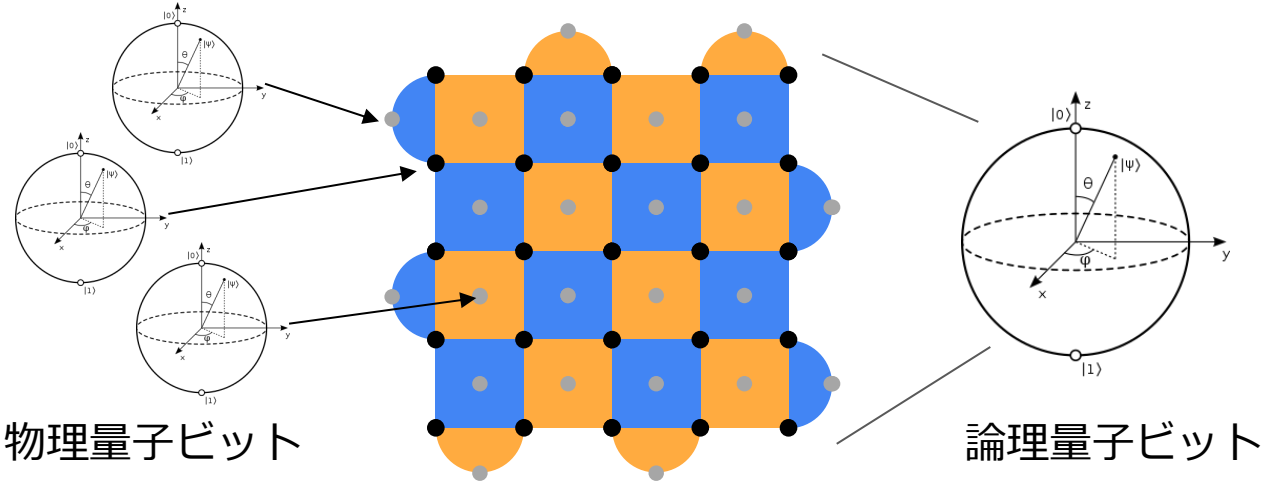
- エラー検出
  - 符号を構成する各ビットをチェックしてすべて一致しているかどうか
  - 隣接ビットのパリティがすべて偶であるか <- 量子でも可能なアプローチ
- エラー訂正（復号）
  - パリティ検査の結果から最も近い（確率の高い）符号語を特定

誤り訂正符号 古典の場合



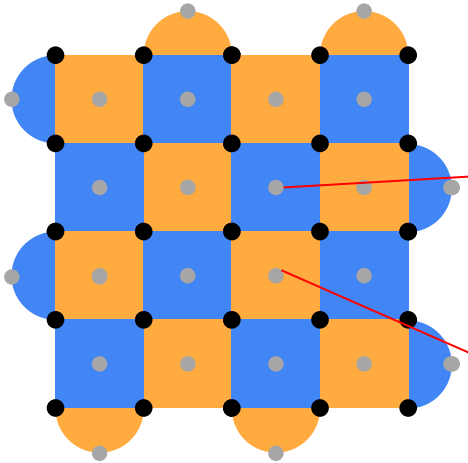
- エラー検出
  - 符号を構成する各ビットをチェックしてすべて一致しているかどうか
  - 隣接ビットのパリティがすべて偶であるか <- 量子でも可能なアプローチ
- エラー訂正（復号）
  - パリティ検査の結果から最も近い（確率の高い）符号語を特定

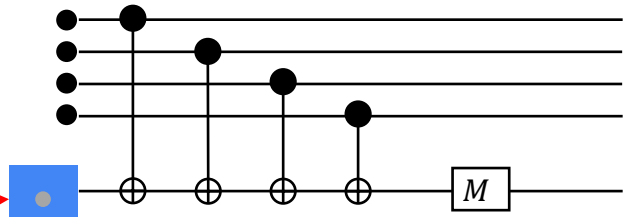
量子誤り訂正符号: 表面符号



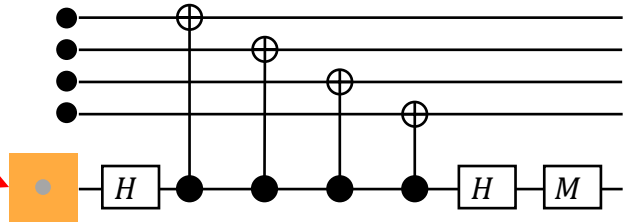
- 規則的に並んだ複数の量子ビットで論理量子ビットを表現
  - 論理ビットの状態を表す**データ量子ビット**と観測用の**補助量子ビット**
- 補助量子ビットの観測値: XとZの2種類のエラーのうち、対応するもののパリティチェック

表面符号におけるエラーパリティチェック



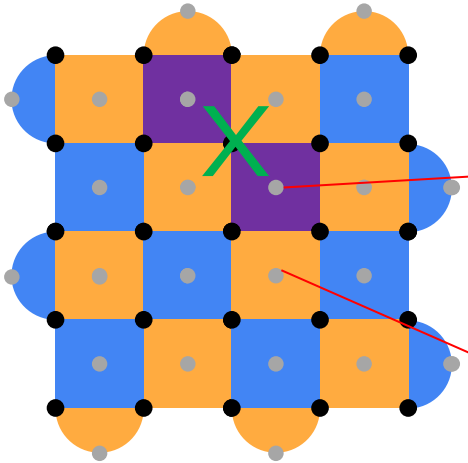


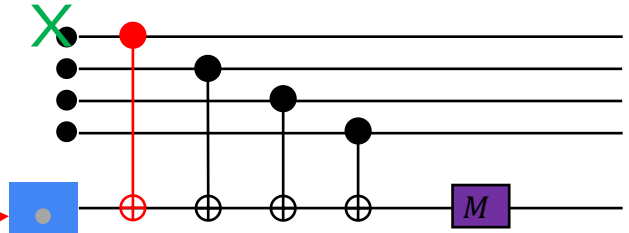
Z-stabilizer measurement
(X error parity check)



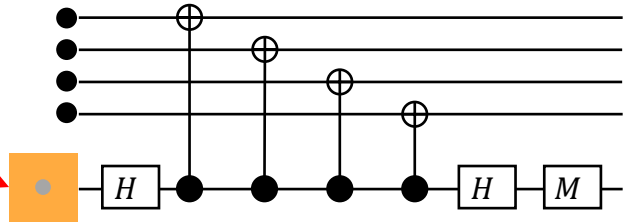
X-stabilizer measurement
(Z error parity check)

表面符号におけるエラーパリティチェック



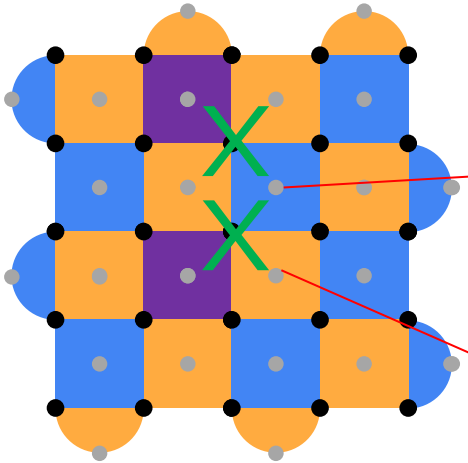


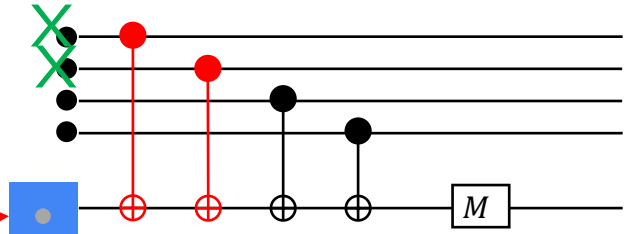
Z-stabilizer measurement
(X error parity check)



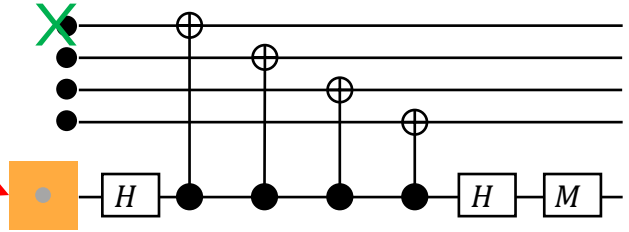
X-stabilizer measurement
(Z error parity check)

表面符号におけるエラーパリティチェック



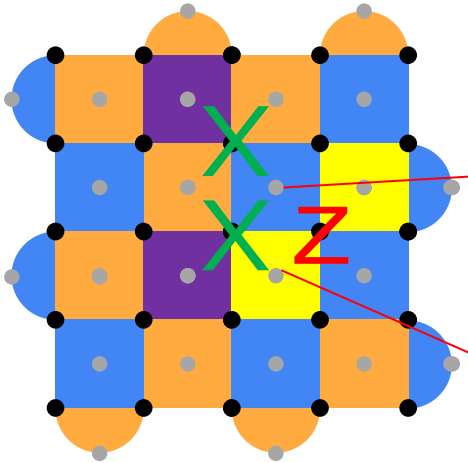


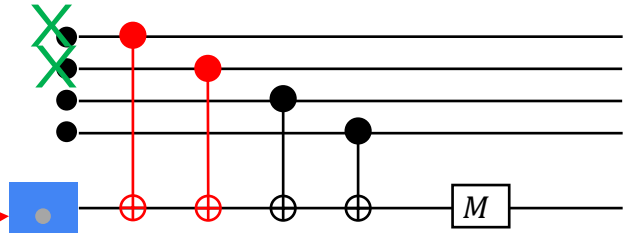
Z-stabilizer measurement
(X error parity check)



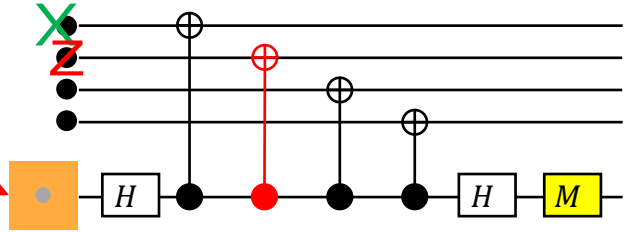
X-stabilizer measurement
(Z error parity check)

表面符号におけるエラーパリティチェック



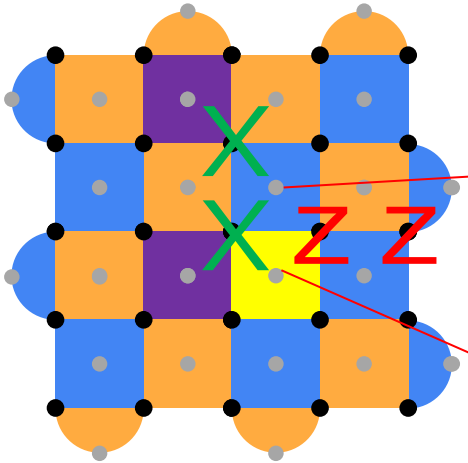


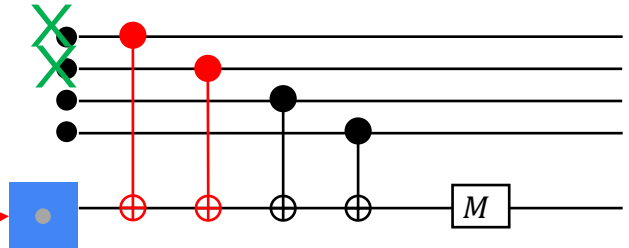
Z-stabilizer measurement
(X error parity check)



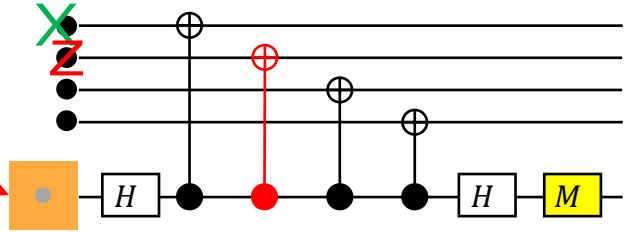
X-stabilizer measurement
(Z error parity check)

表面符号におけるエラーパリティチェック



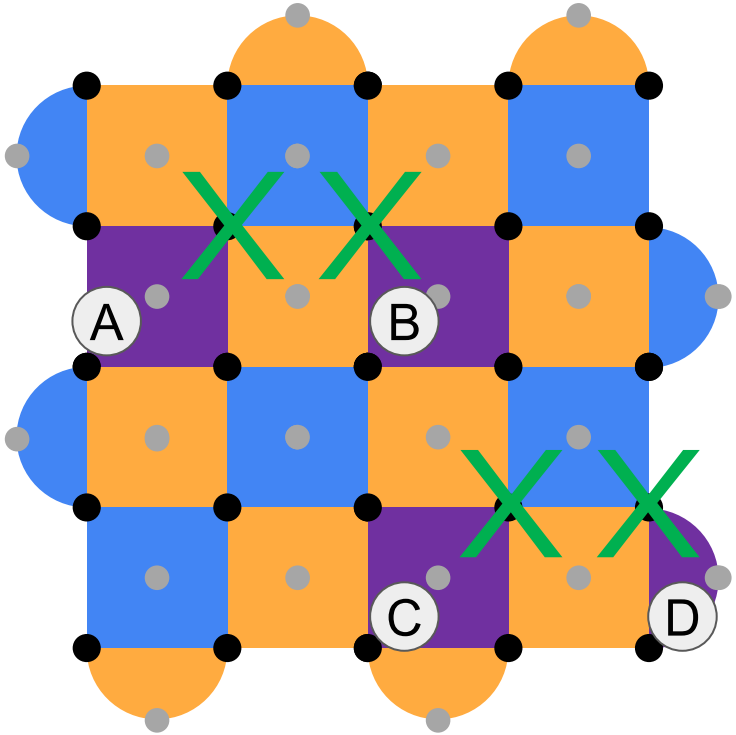


Z-stabilizer measurement
(X error parity check)



X-stabilizer measurement
(Z error parity check)

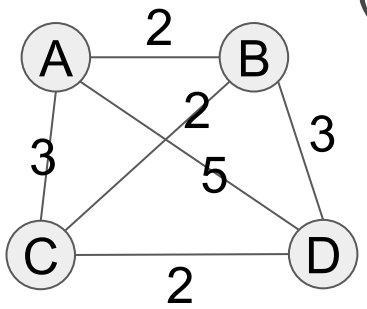
表面符号におけるエラー推定（復号）



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

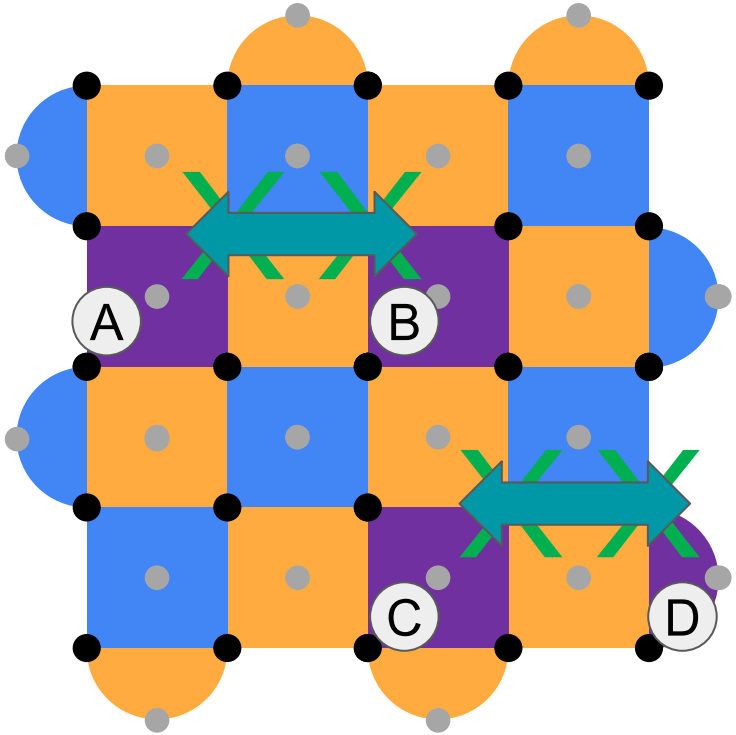
Minimum Weight Perfect Matching (MWPM)



V: Hot syndromes
W_e: Manhattan distance

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

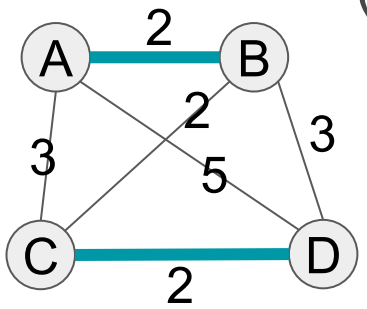
表面符号におけるエラー推定（復号）



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

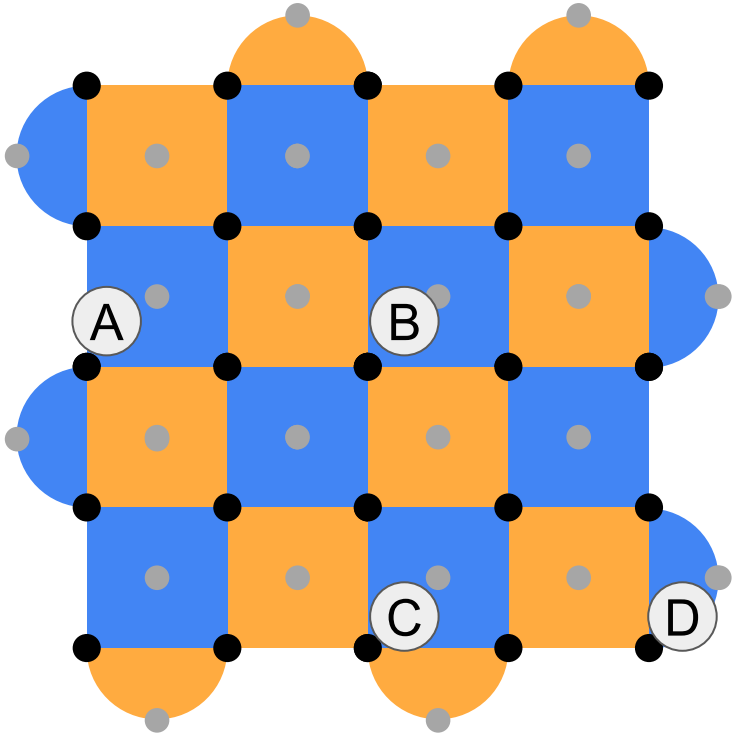
Minimum Weight Perfect Matching (MWPM)



V: Hot syndromes
W_e: Manhattan distance

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

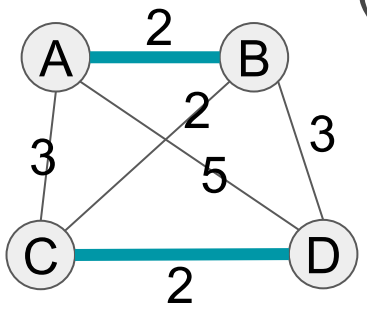
表面符号におけるエラー推定（復号）



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

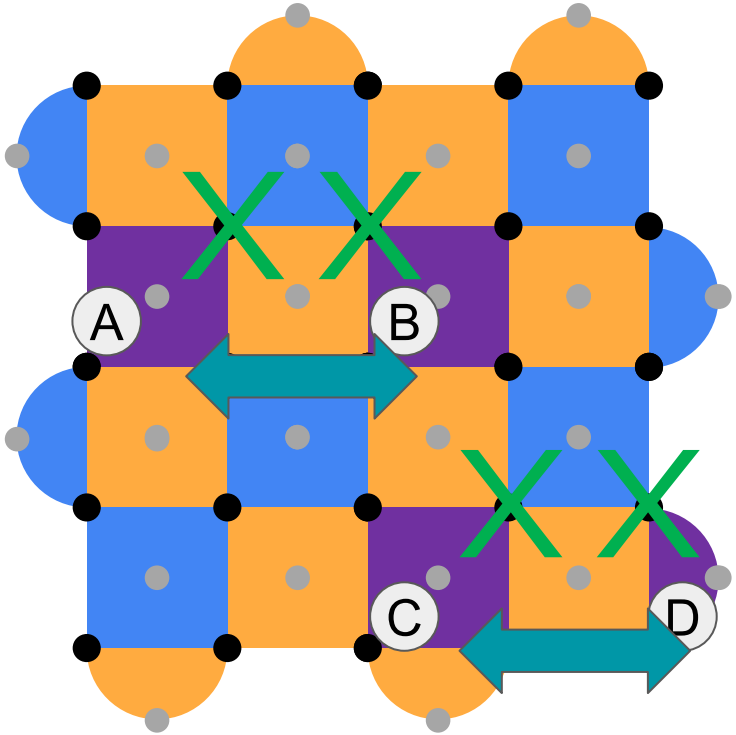
Minimum Weight Perfect Matching (MWPM)



V: Hot syndromes
W_e: Manhattan distance

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

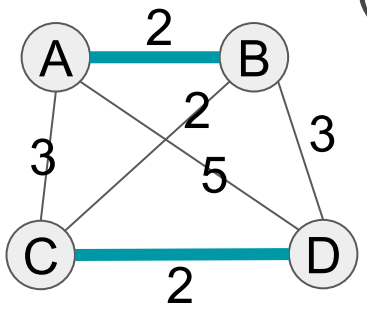
表面符号におけるエラー推定（復号）



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

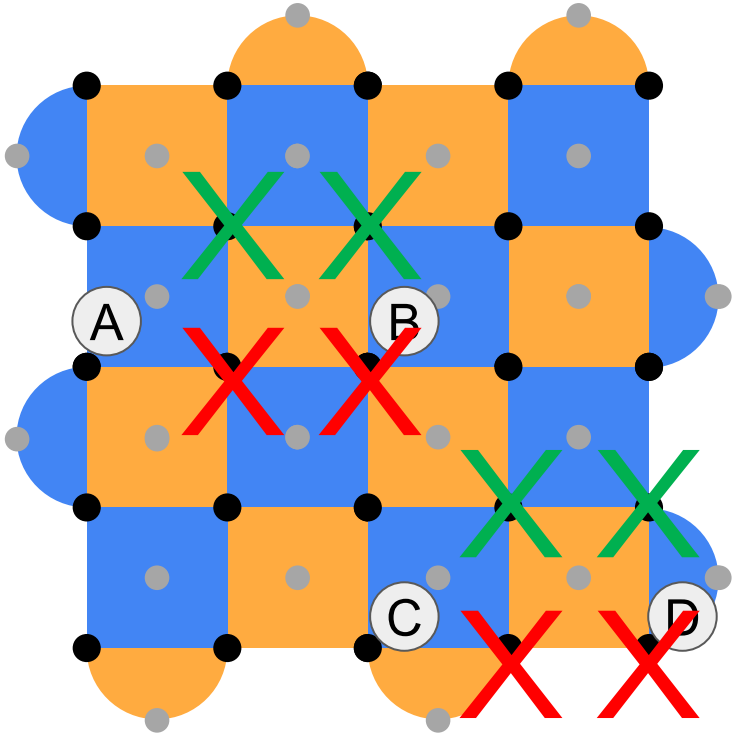
Minimum Weight Perfect Matching (MWPM)



V: Hot syndromes
W_e: Manhattan distance

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

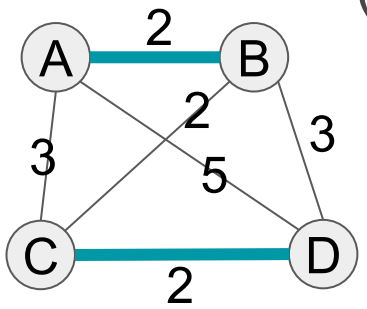
表面符号におけるエラー推定（復号）



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

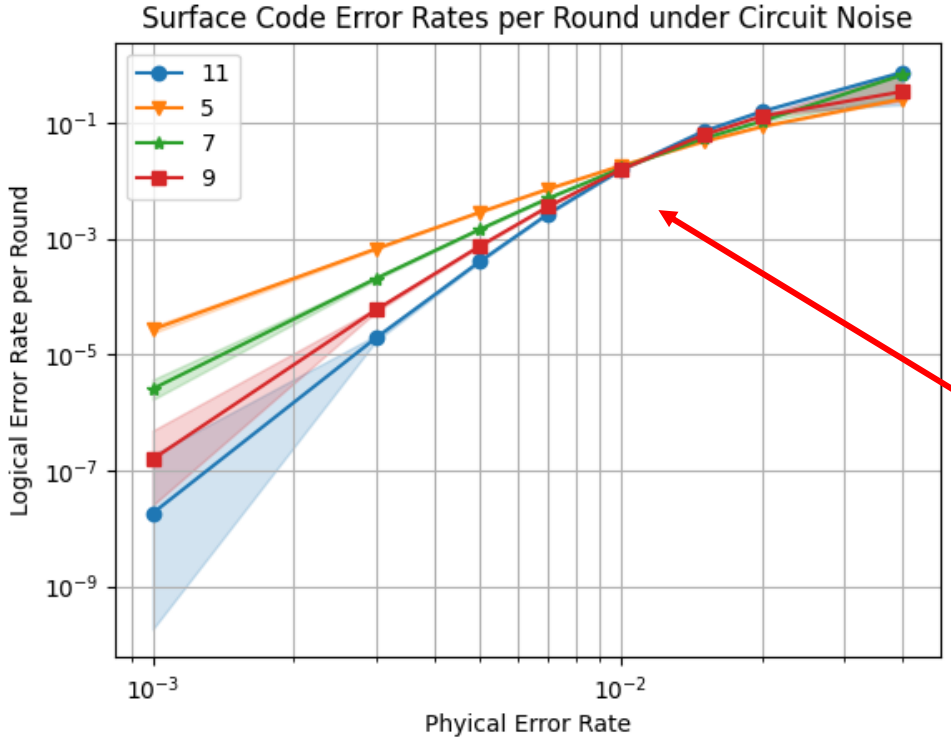
Minimum Weight Perfect Matching (MWPM)



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

QECOOL: On-Line Quantum Error Correction with a Superconducting Decoder for Surface Code [DAC'21]
&
QULATIS: A Quantum Error Correction Methodology toward Lattice Surgery [HPCA'22]

Yosuke Ueno[1], Masaaki Kondo[1], Masamitsu Tanaka[2],
Yasunari Suzuki[3], Yutaka Tabuchi[4]

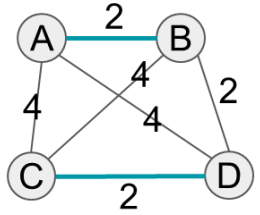
[1]The University of Tokyo [2]Nagoya University [3]NTT [4]RIKEN

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

誤り推定器
（古典計算機）

グラフマッチング



冷凍機

Error corr.

Qubit meas.

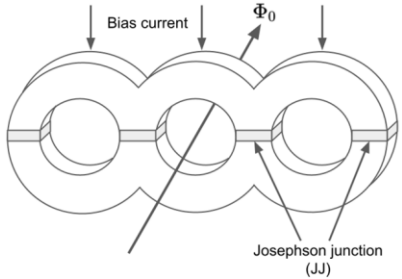
Qubits

提案アーキテクチャ

推定器

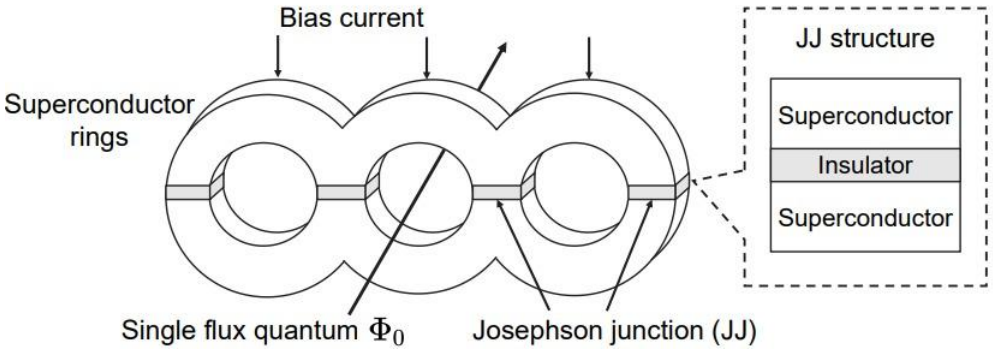
Qubits

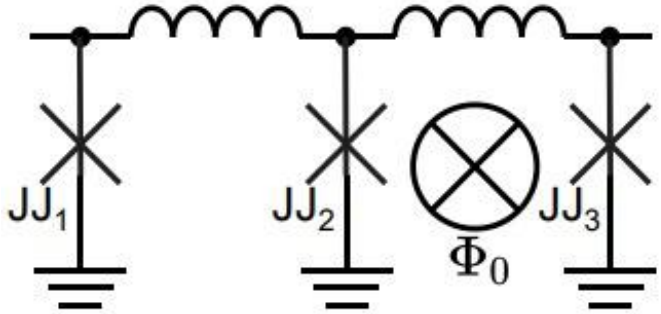
SFQ circuits



- 高速・低消費電力
- パルス論理
- RAMは高コスト

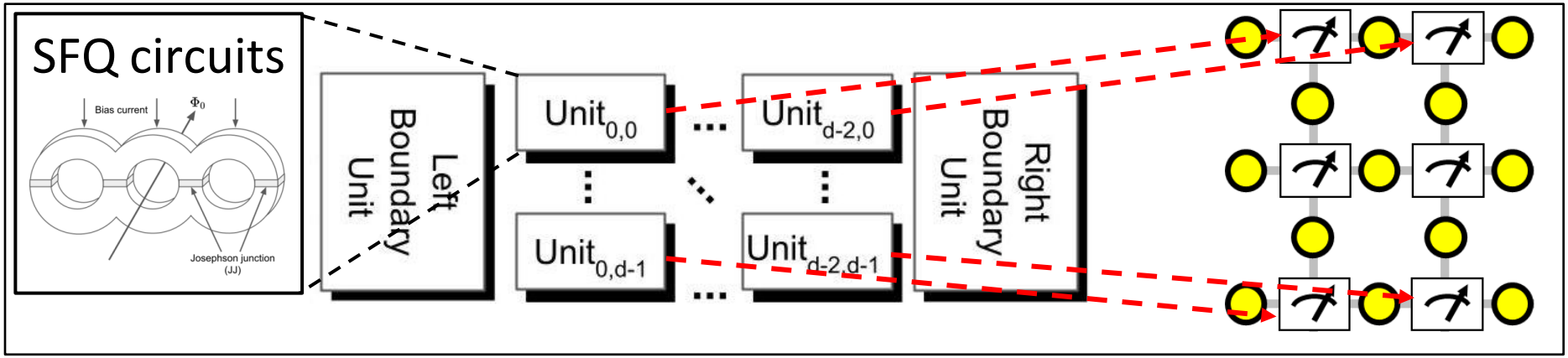
超伝導古典回路によるエラー推定





- 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: QECOOL

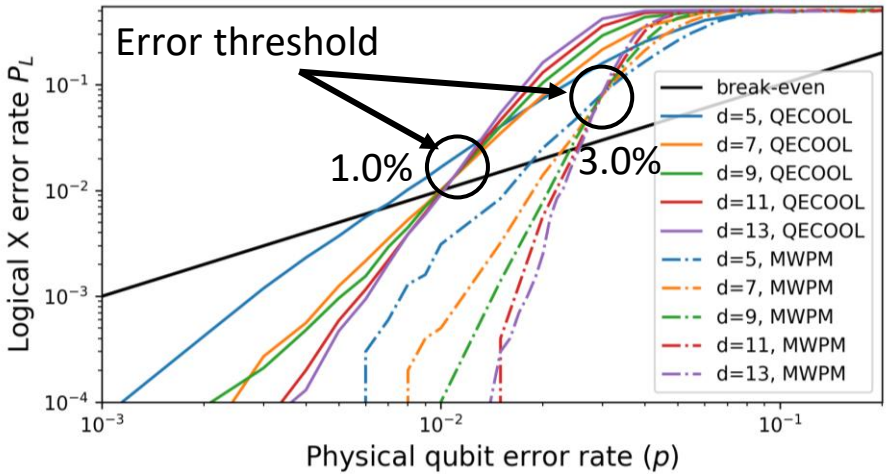


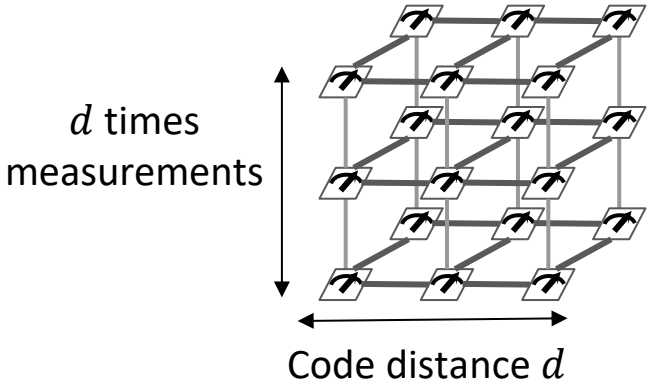
QECOOLのアーキテクチャ

- Quantum Error COrrection 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, “QECOOL: 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





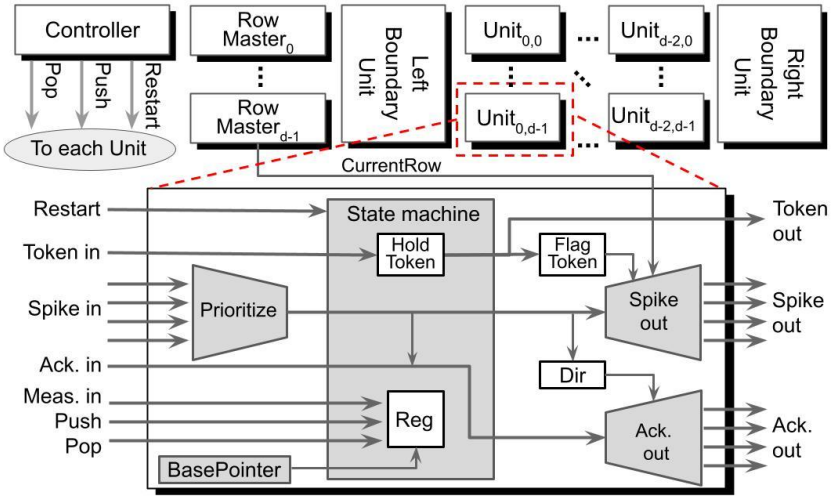
Target lattice shape

Experimental condition

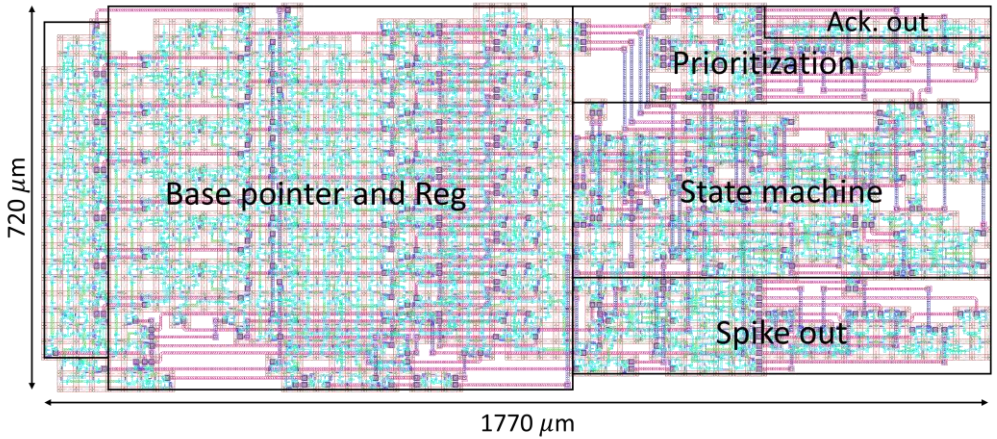
- Measurement process is performed **once every 1 μs**
- Each QECOOL Unit has a **7-bit** buffer to store syndrome values
- If buffer entry size is greater than $K = 3$, QECOOL is performed; otherwise, each Unit waits for measurement process
- MWPM operates with batch-QEC manner

- 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 mm^2	Latency: 215 ps	Power cons.: **2.78 μW**

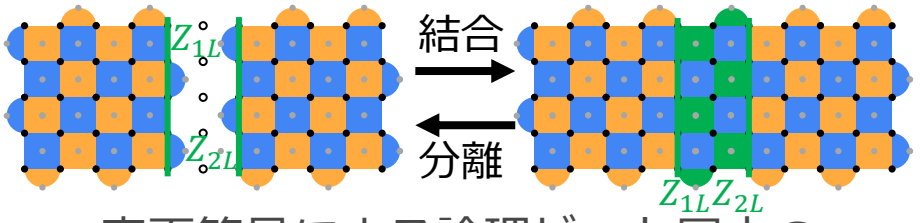
Decoder power consumption per one logical qubits

Suppose $d = 9$,

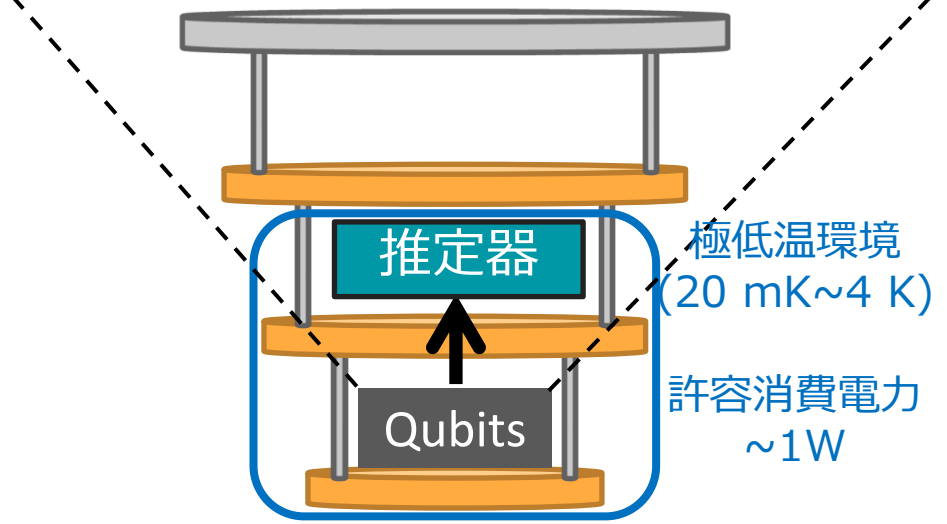
$$9 \times 8 \times 2 \times 2.78_{[\mu W]} = \boxed{400\ \mu\mathrm{W}}$$

QULATIS: 論理演算を行う場合への拡張

格子手術

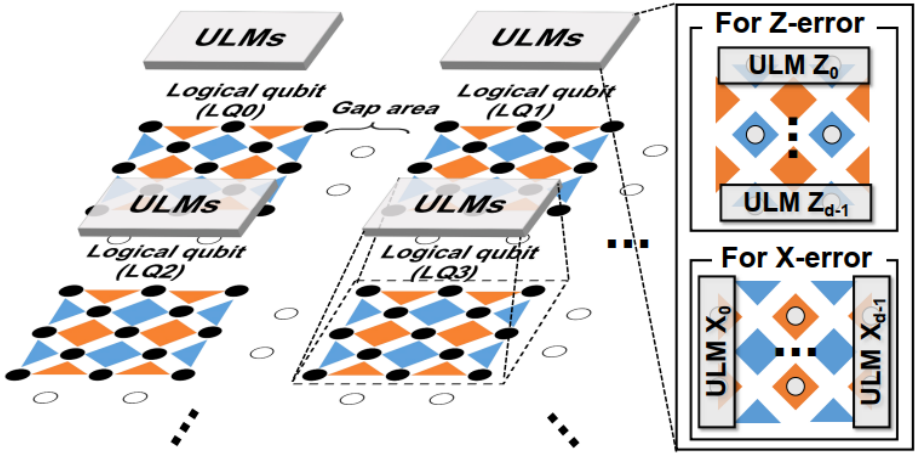


表面符号による論理ビット同士の
結合・分割により**論理演算**を実現



提案手法：QULATIS

- より複雑な形状のグラフマッチング問題を解く**SFQ誤り推定アーキテクチャ**
- 格子手術の誤り推定を具体的に考えた**世界初のアーキテクチャ**



Ueno, Kondo, Tanaka, Suzuki, and Tabuchi. QULATIS: A Quantum Error Correction Methodology toward Lattice Surgery. HPCA2022.

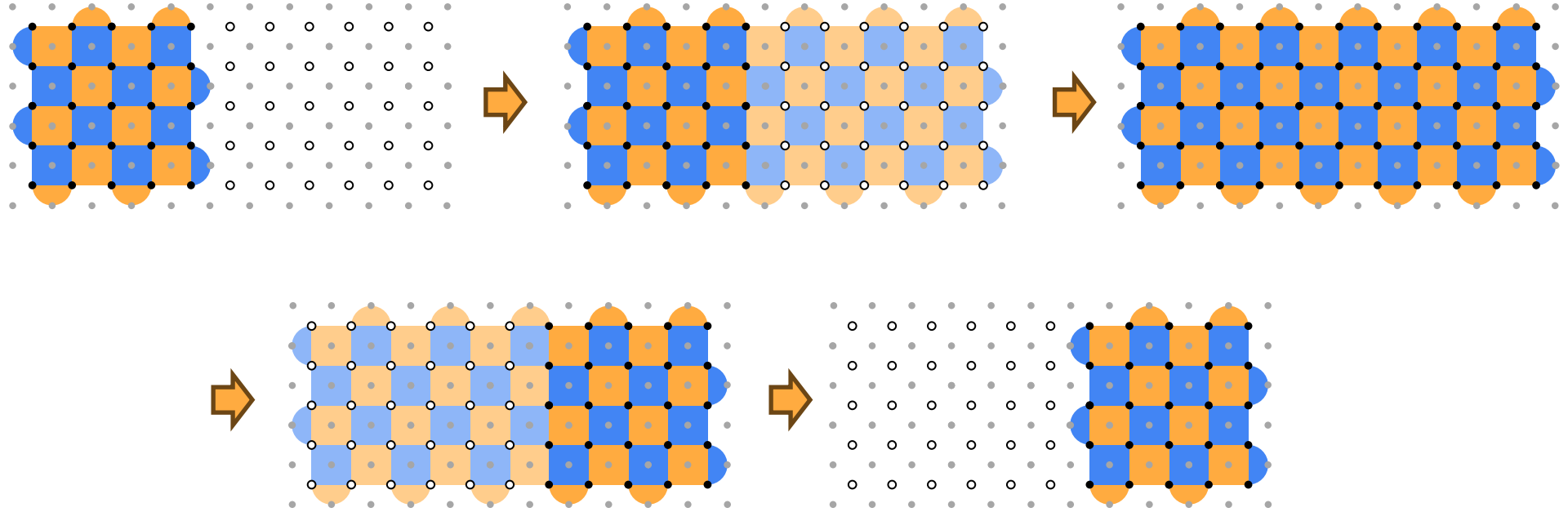
QECOOL/QULATISまとめ

- 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$400\ \mu\mathrm{W}$
  - 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できる
- QULATIS: 格子手術に基づく論理演算へとQECOOLを拡張
  - 論理ビットの境界が動的に変わる状況に対処

今日の内容

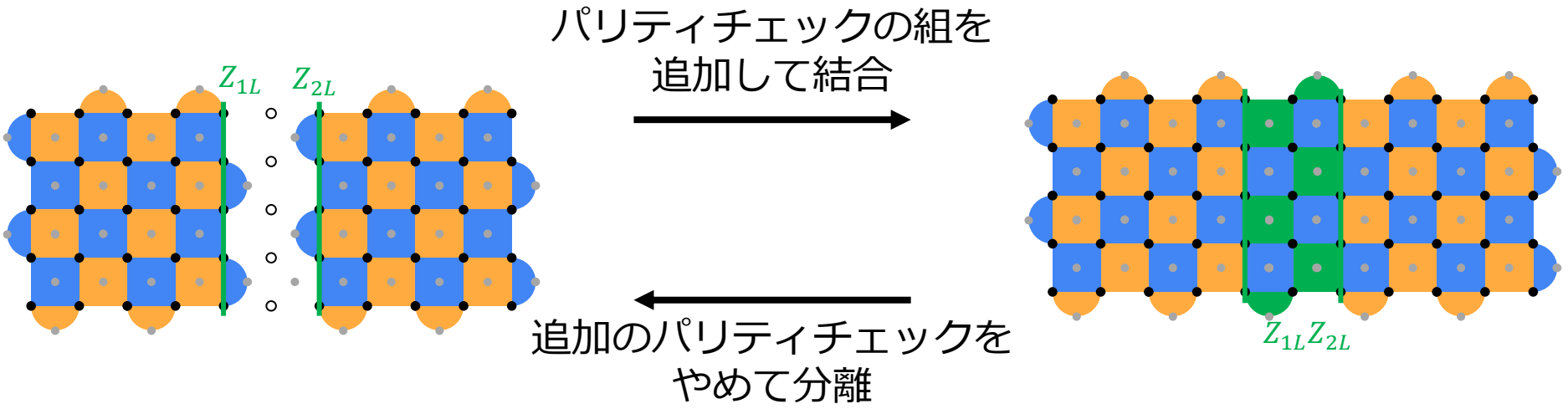
- 量子計算機アーキテクチャ分野の研究動向
  - 計算機アーキテクチャの研究対象・隣接分野
  - 計算機アーキテクチャ分野における量子計算機関連の研究動向
- 量子計算機アーキテクチャに関連する研究紹介
  - 超伝導ディジタル回路を用いた量子誤り推定機構（博論、DAC'21、HPCA'22）
  - 誤り耐性量子計算（FTQC）におけるロードストア型アーキテクチャ（HPCA'25）
  - 未出版内容のため公開版では削除
  - 中性原子を対象としたフルスタックFTQCシミュレーションフレームワーク（MICRO'26）
- 量子計算機アーキテクチャ研究の魅力・まとめ

符号の変形



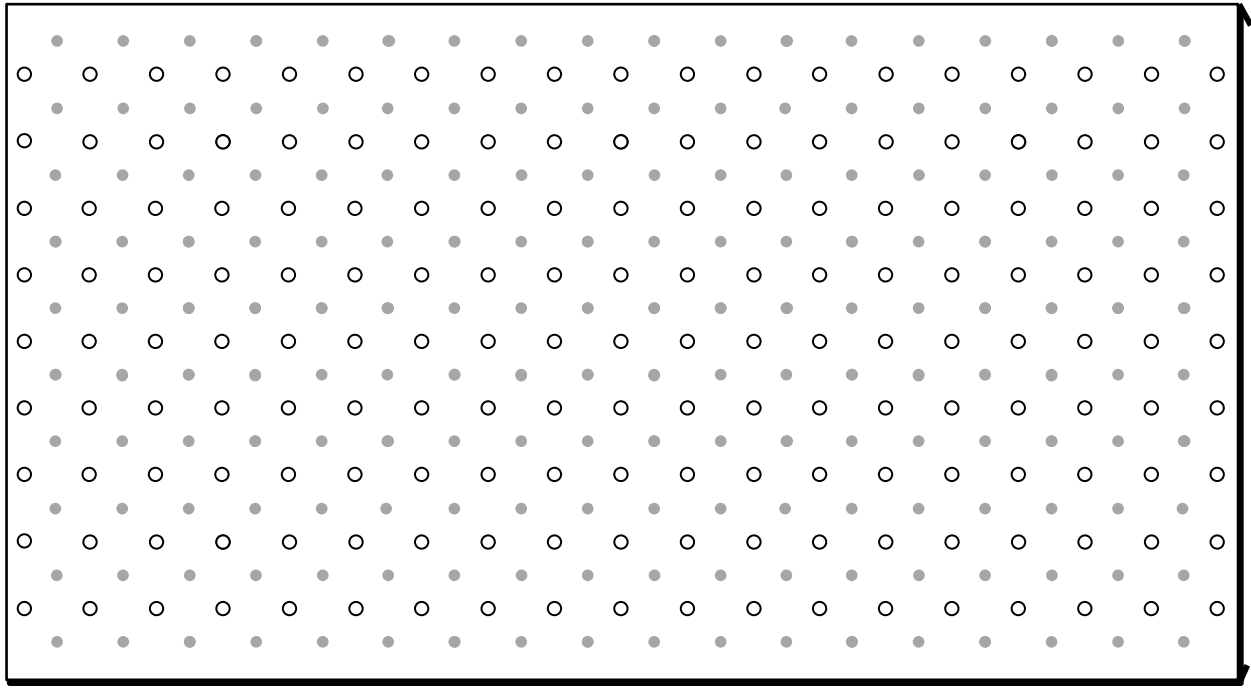
- パリティチェックの組を変えることで符号を変形
  - 論理状態を保ったまま変形 -> 拡大、縮小、**移動**、回転
  - 論理状態を変えながら変形 -> 論理S, CNOTゲート

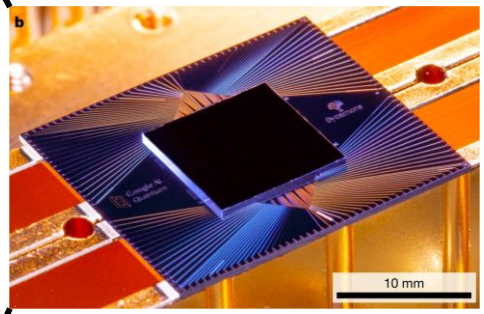
格子手術による多体Pauli測定



- 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

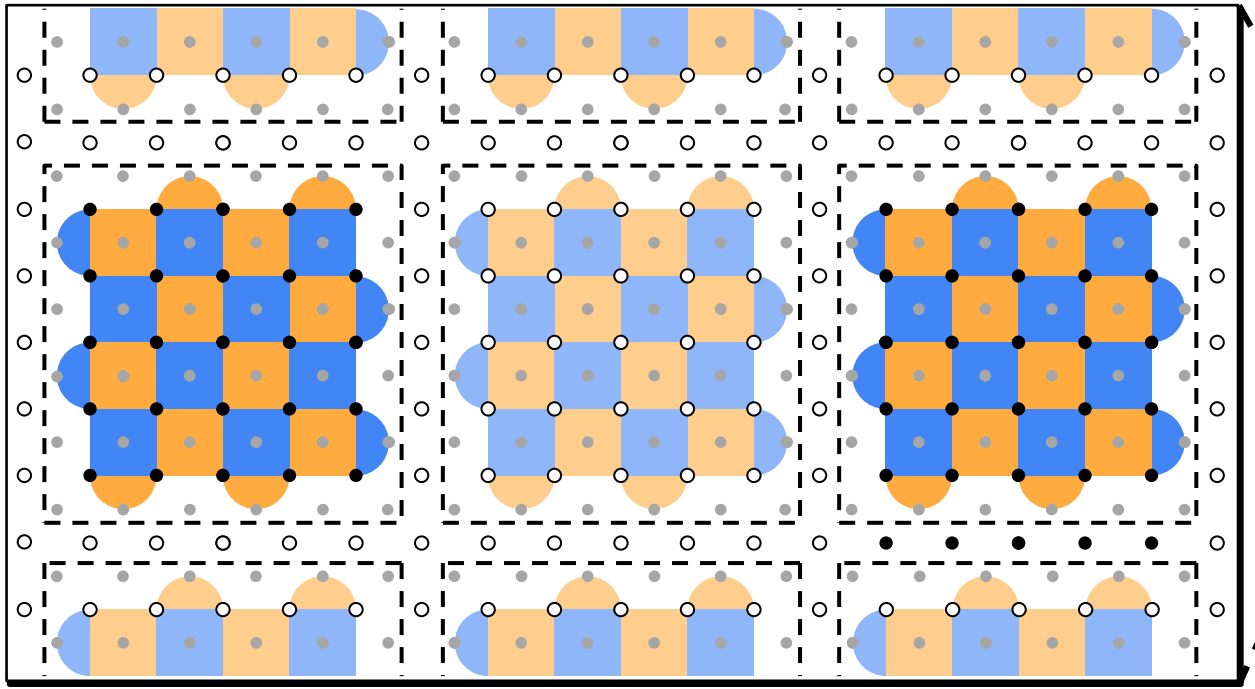
実際の量子ビットでの実装

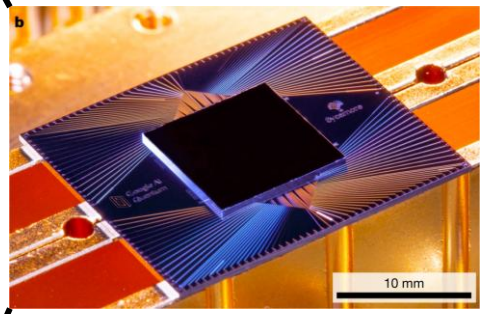




- 表面符号で計算を実行できるように区画を分ける
- 一部に論理状態を確保して一部は計算用に空けておく

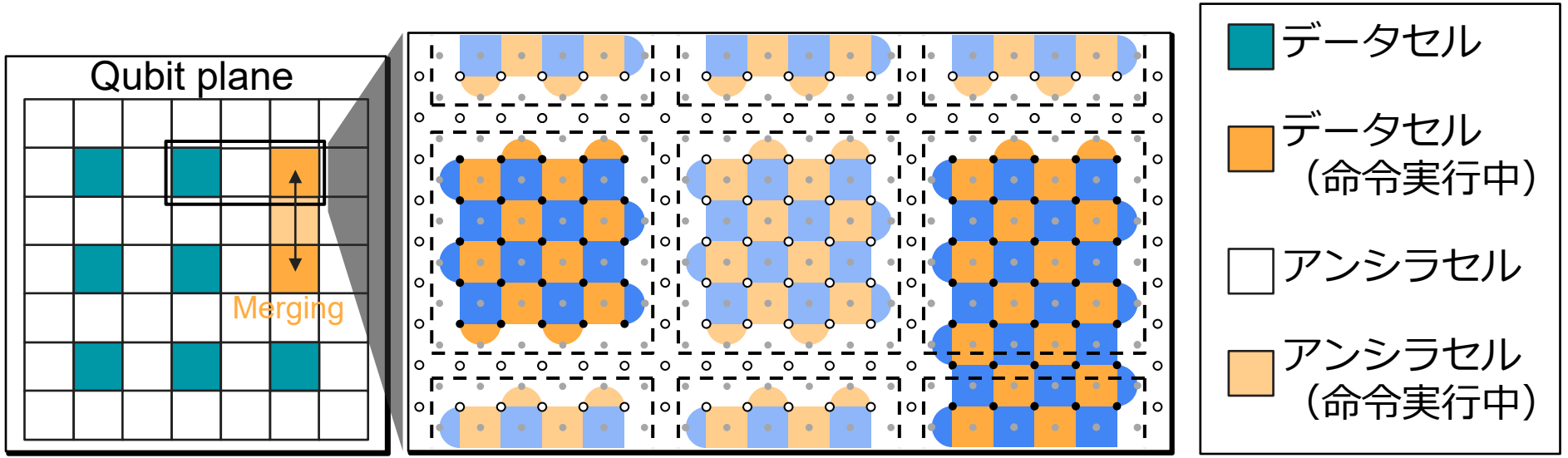
実際の量子ビットでの実装





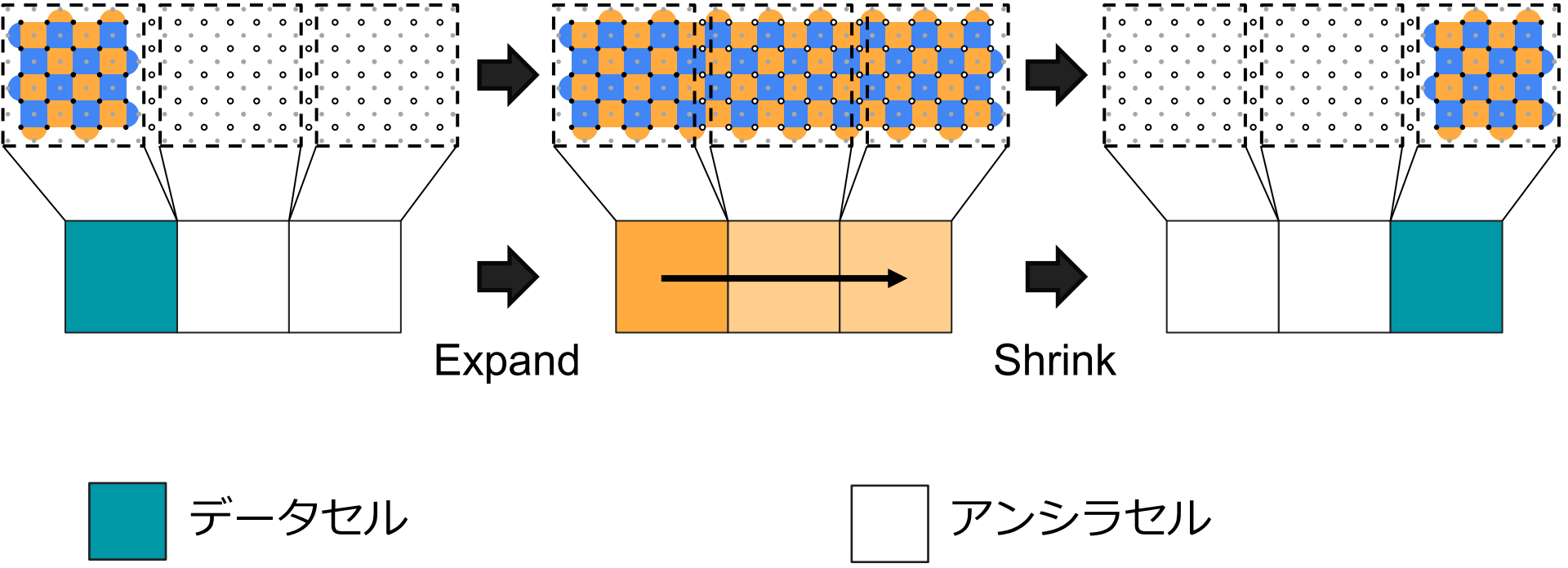
- 表面符号で計算を実行できるように区画を分ける
- 一部に論理状態を確保して一部は計算用に空けておく

表面符号による論理演算を実行するQubit plane

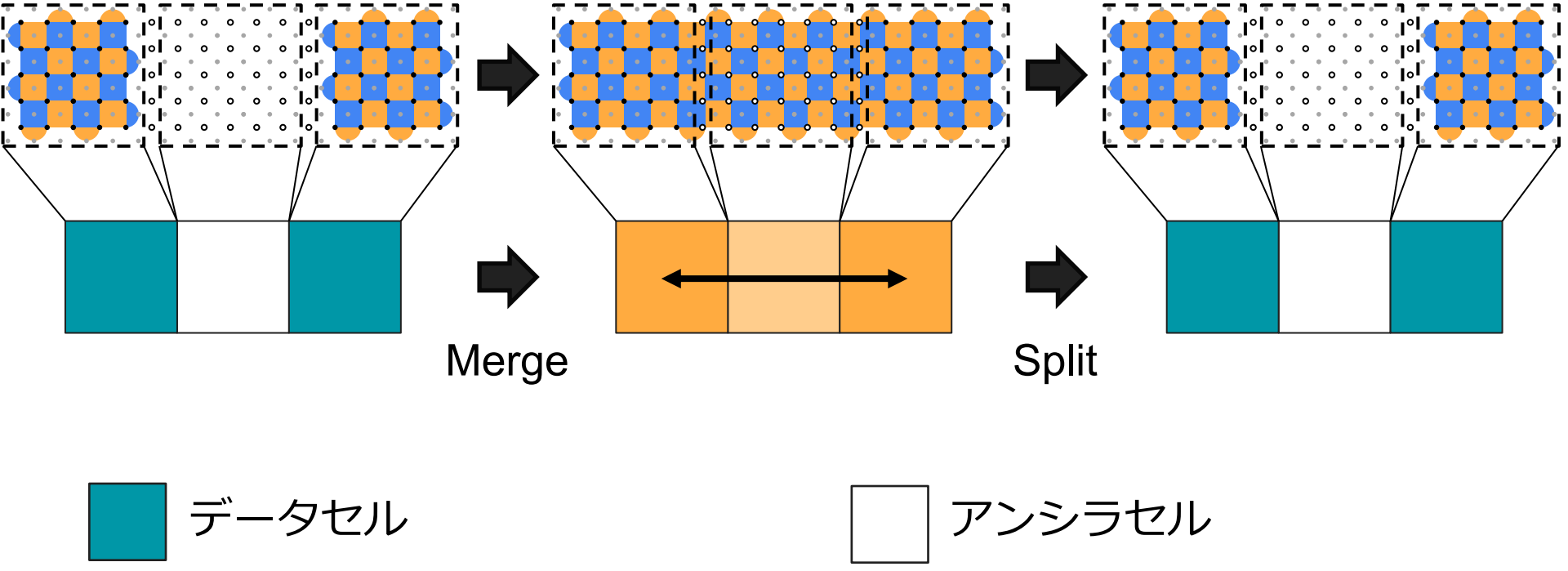


- 1つ1つのセルが符号距離dの表面符号を構成できる物理量子ビットを持つ
- **データセル**に計算用の論理状態を確保
- 空きスペース（**アンシラセル**）を使って
  計算（＝データセルの**拡大・縮小**、**結合・分離**）を行う

例：論理ビットの移動

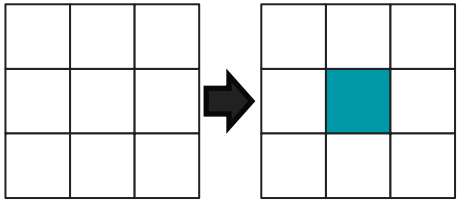


例：格子手術

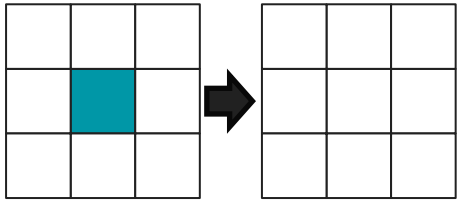


格子手術命令セットの一例

INIT_Z, INIT_X (0 code beat)

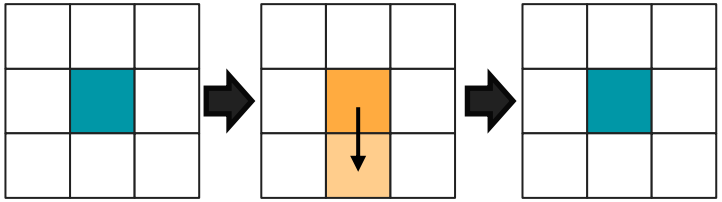


MEAS_X, MEAS_Z (0 code beat)



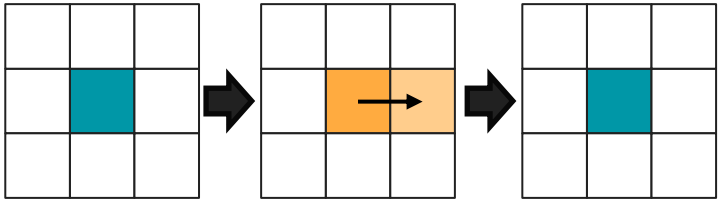
+Classical output

OP_H (3 code beats)



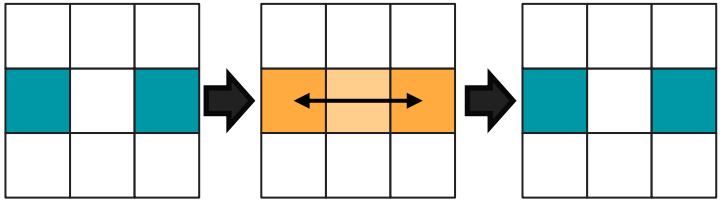
Any direction

OP_S (2 code beats)



Horizontal direction

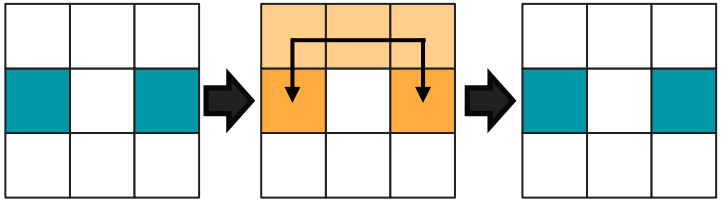
MEAS_ZZ (1 code beat)



Any path

+Classical output

MEAS_XX (1 code beat)



Any path

+Classical output

Slides available in https://yuteno.github.io/

LSQCA: Resource-Efficient Load/Store Architecture for Limited-Scale Fault-Tolerant Quantum Computing

Takumi Kobori[2], Yasunari Suzuki[1], **Yosuke Ueno**[1, 2]

Teruo Tanimoto[3], Synge Todo[1], Yuuki Tokunaga[4]

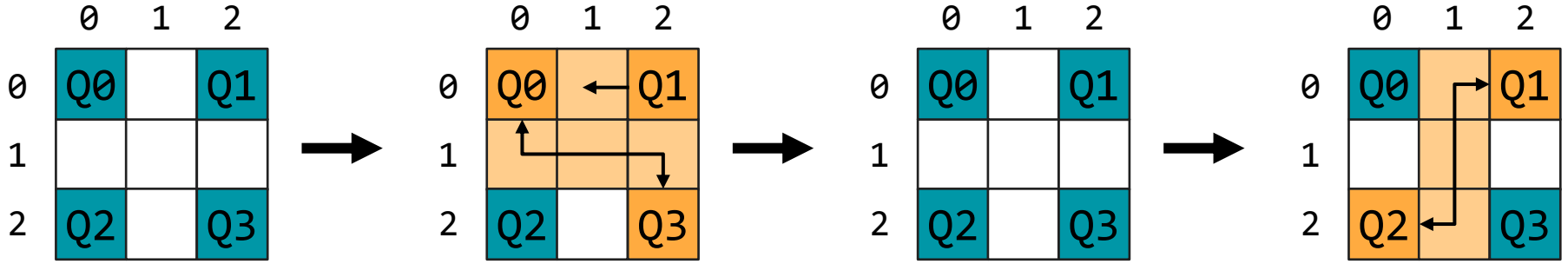
Physicists and Computer architects

[1]RIKEN, [2]The University of Tokyo, [3]Kyushu University, [4]NTT

arXiv 2412.20486

Takumi Kobori, Yasunari Suzuki, **Yosuke Ueno**, Teruo Tanimoto, Synge Todo, Yuuki Tokunaga, LSQCA: Resource-Efficient Load/Store Architecture for Limited-Scale Fault-Tolerant Quantum Computing, IEEE International Symposium on High-Performance Computer Architecture (HPCA) 2025, pp. 304-320.

格子手術のコンパイルと問題点




```
OP_S Q1 left
MXX  Q0 Q3
     (0,1) (1,1) (2,1)
MZZ  Q1 Q2
     (1,0) (1,1) (1,2)
```

```
OP_S Q1 left
MXX  Q0 Q3
     (0,1) (1,1) (2,1)
MZZ  Q1 Q2
     (1,0) (1,1) (1,2)
```

```
OP_S Q1 left
MXX  Q0 Q3
     (0,1) (1,1) (2,1)
MZZ  Q1 Q2
     (1,0) (1,1) (1,2)
```

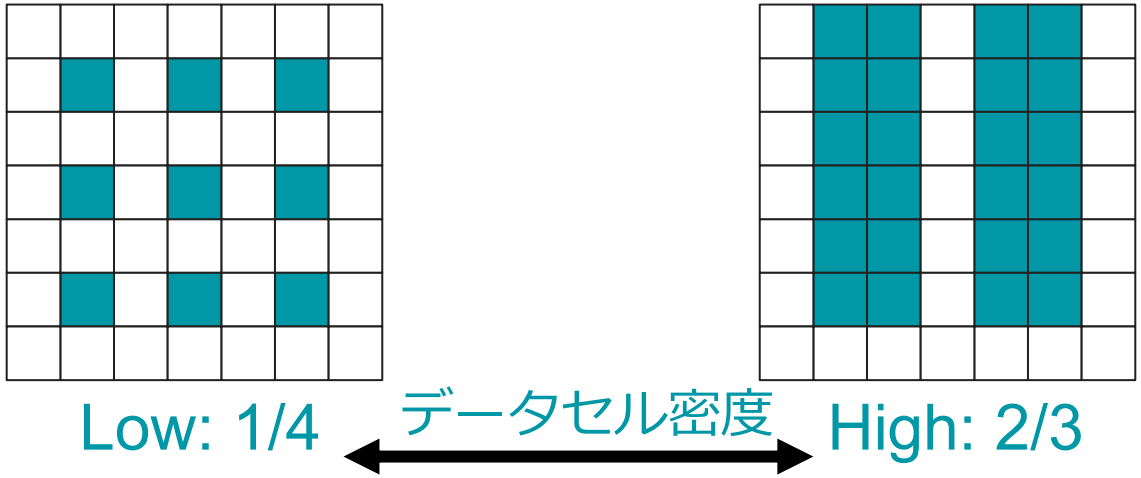
```
OP_S Q1 left
MXX  Q0 Q3
     (0,1) (1,1) (2,1)
MZZ  Q1 Q2
     (1,0) (1,1) (1,2)
```

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

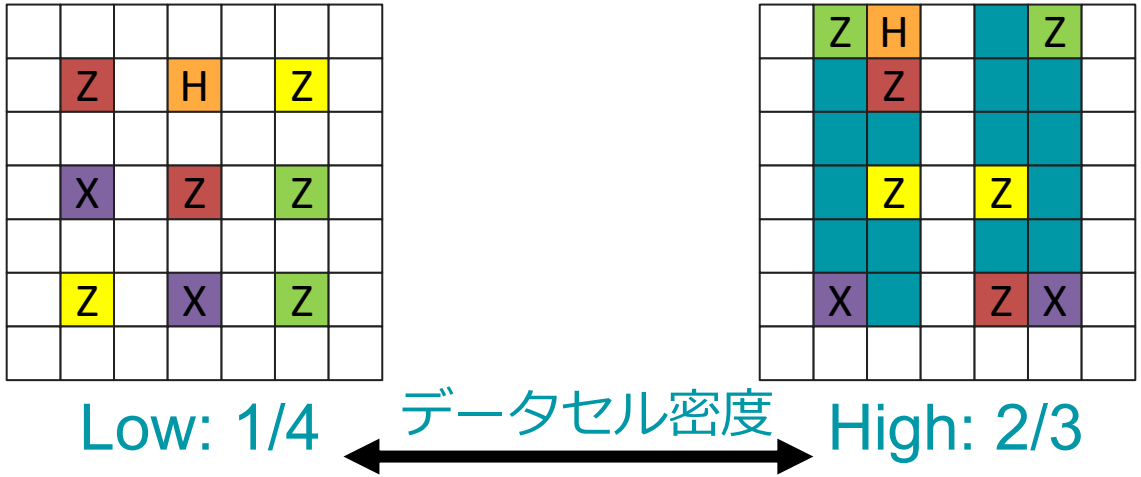
データセル配置と実行時間

- **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



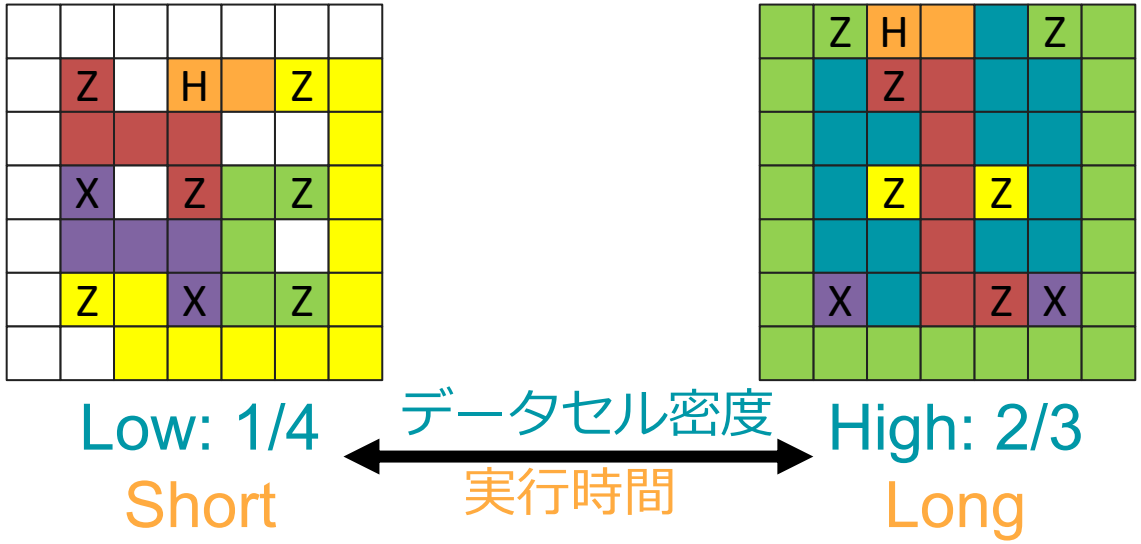
データセル配置と実行時間

- **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

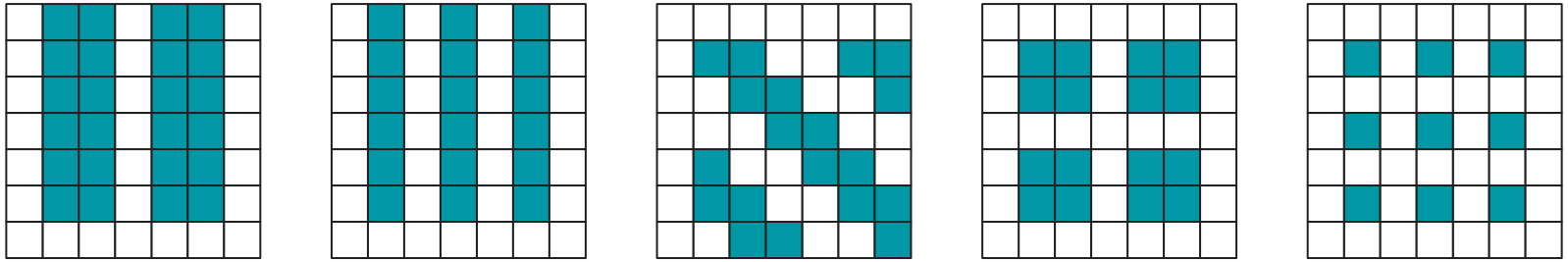


データセル配置と実行時間

- **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



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

↕

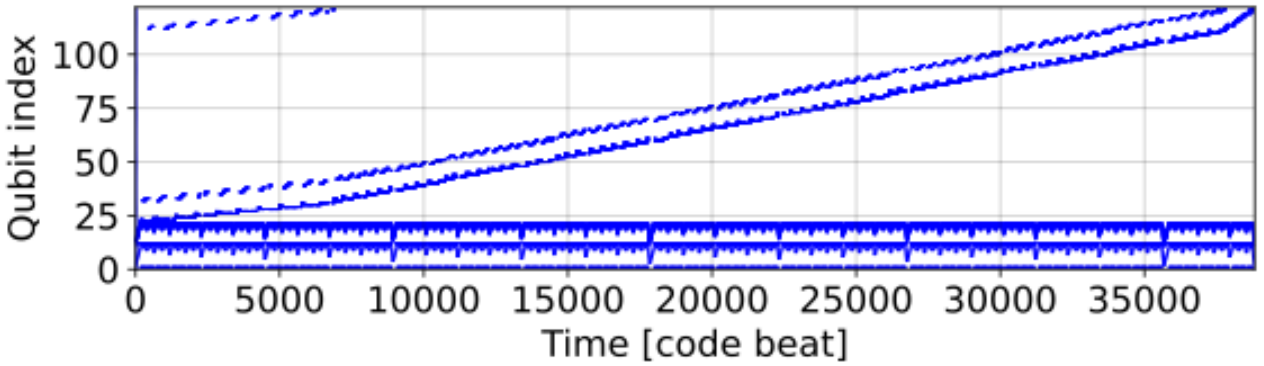
従来の格子手術コンパイルの問題点2

- 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

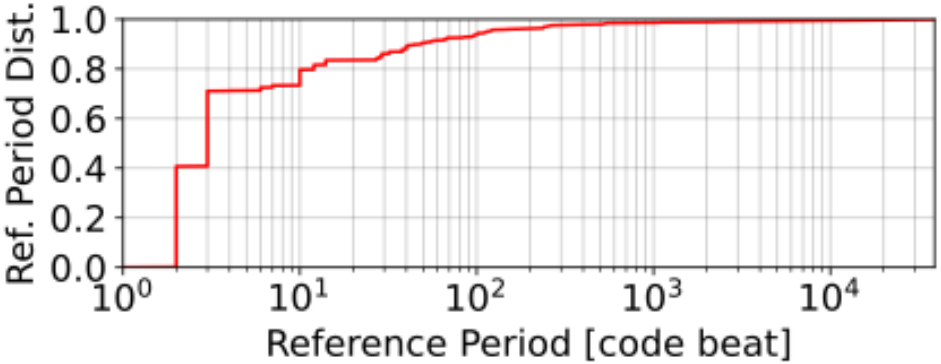
[1] J. Lee et al., PRX Quantum 2, 030305 (2021). [2] M. Beverland et al., arXiv:2211.07629 (2022). [3] Y. Ueno et al., arXiv:2411.17519 (2024).
[4] C. Chamberland and E. T. Campbell, PRX Quantum 3, 010331 (2022). [5] M. Beverland et al., PRX Quantum 3, 020342 (2022).

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

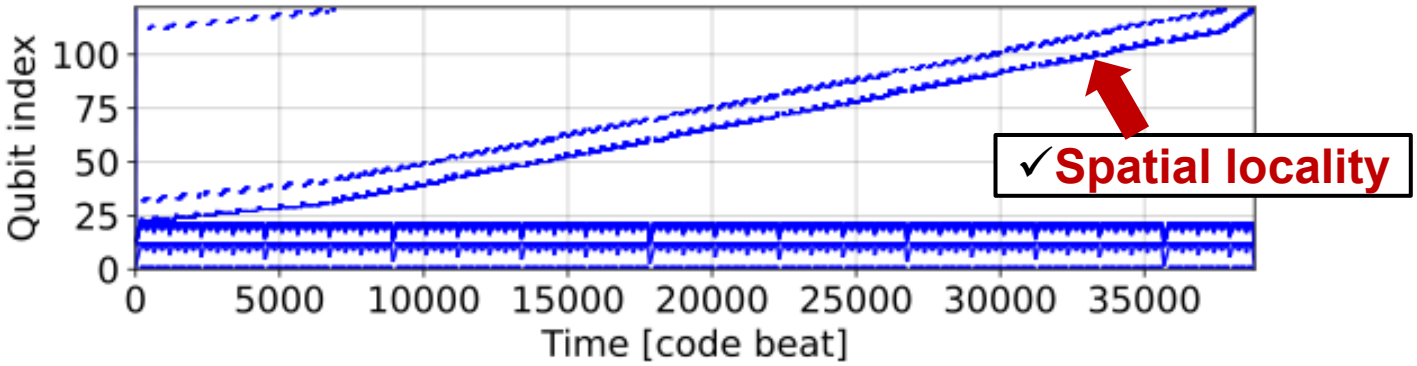


Cumulative distribution of average reference period for SELECT circuit

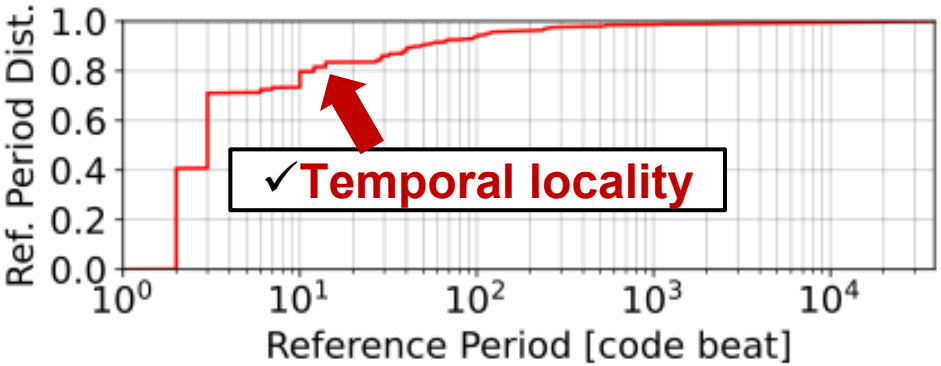


QPEにおけるデータセルへのアクセスパターン

Reference time stamp for SELECT subroutine of quantum phase estimation

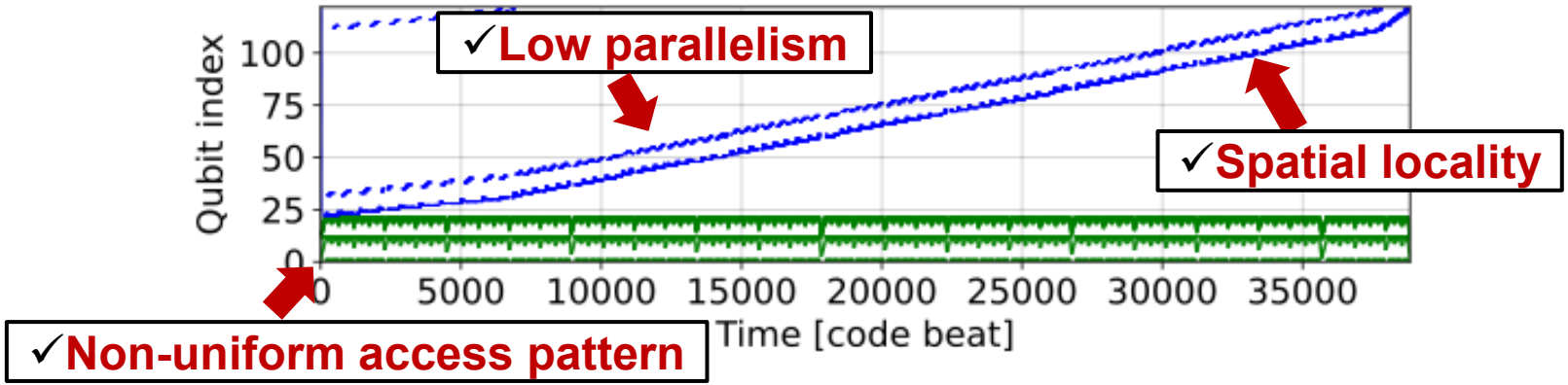


Cumulative distribution of average reference period for SELECT circuit

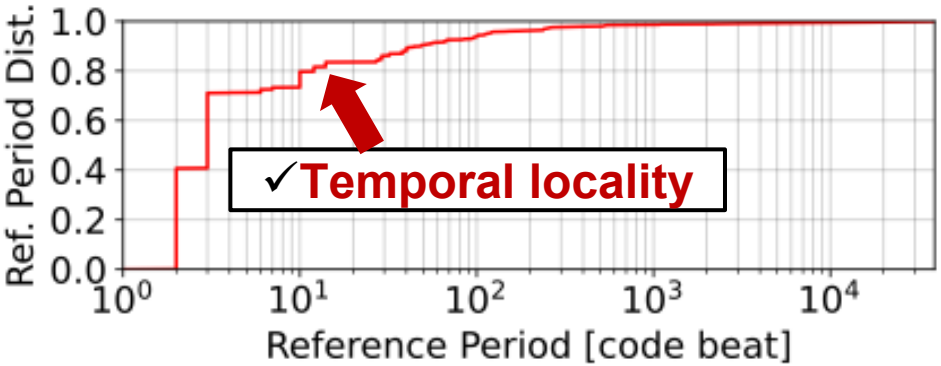


QPEにおけるデータセルへのアクセスパターン

Reference time stamp for SELECT subroutine of quantum phase estimation

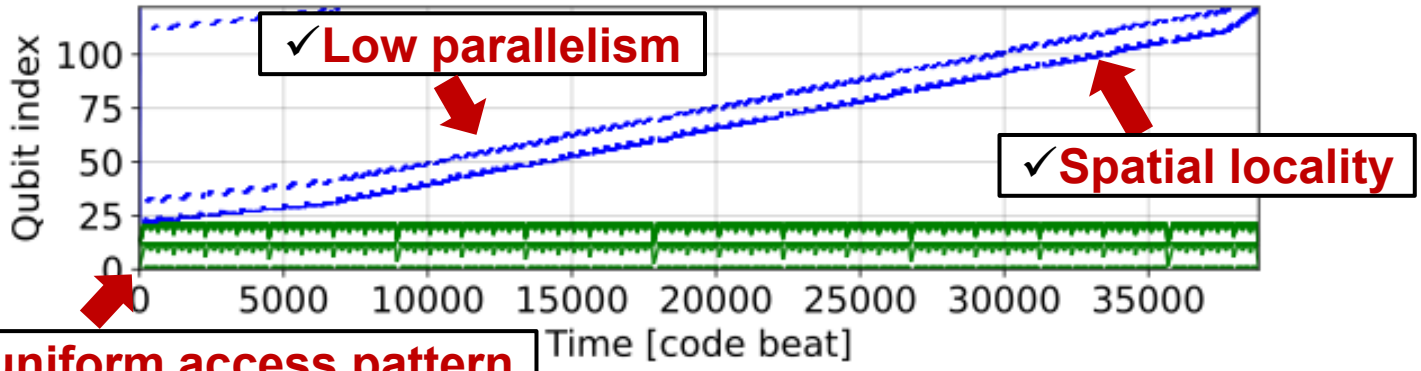


Cumulative distribution of average reference period for SELECT circuit

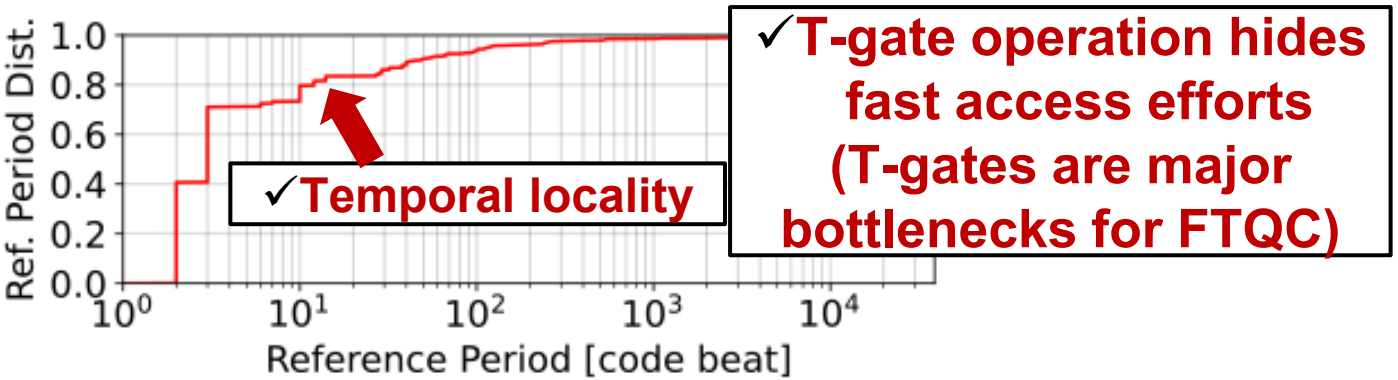


QPEにおけるデータセルへのアクセスパターン

Reference time stamp for SELECT subroutine of quantum phase estimation

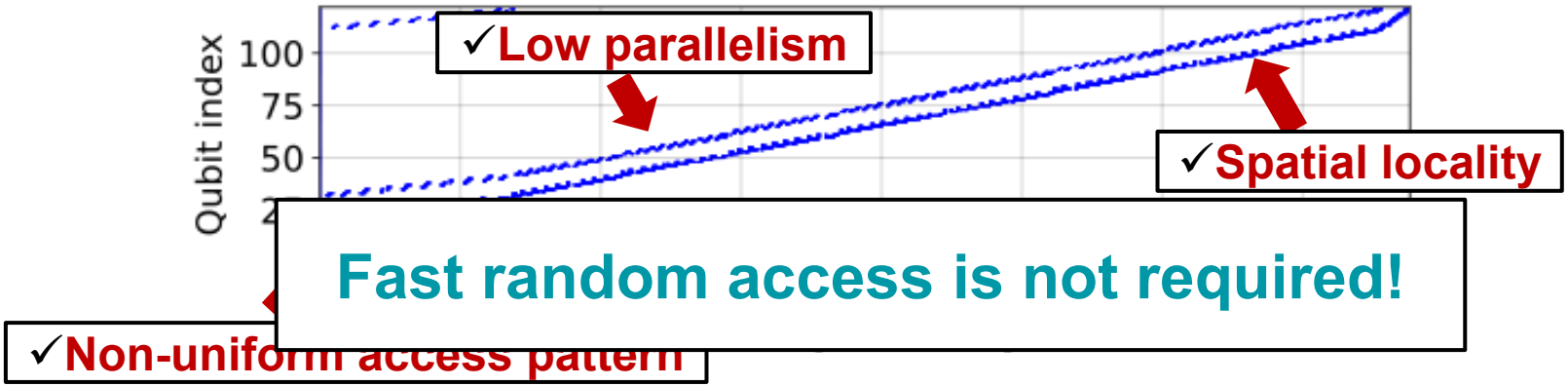


Cumulative distribution of average reference period for SELECT circuit

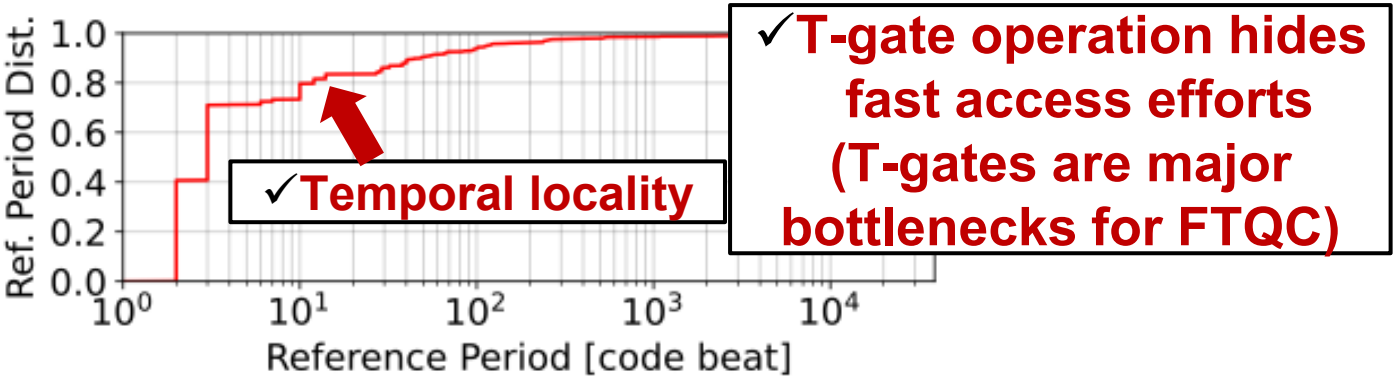


QPEにおけるデータセルへのアクセスパターン

Reference time stamp for SELECT subroutine of quantum phase estimation

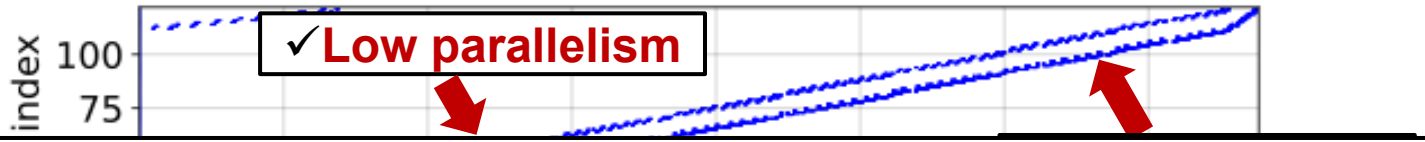


Cumulative distribution of average reference period for SELECT circuit



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

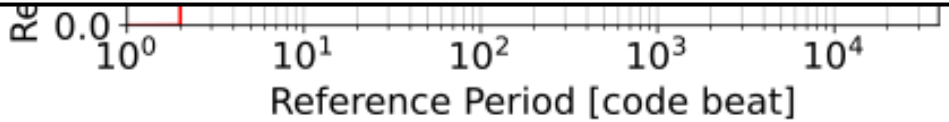


Our goal

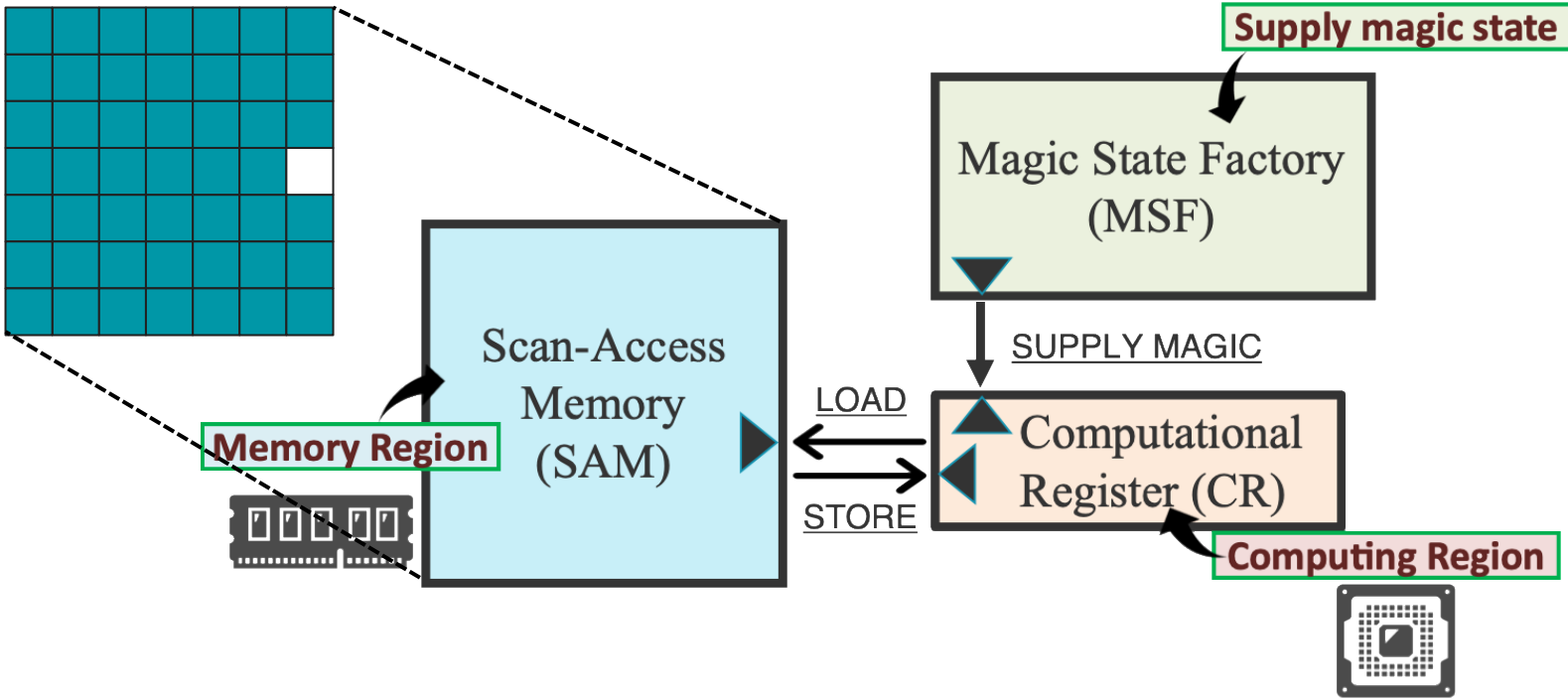
Higher memory density with a small time overhead

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

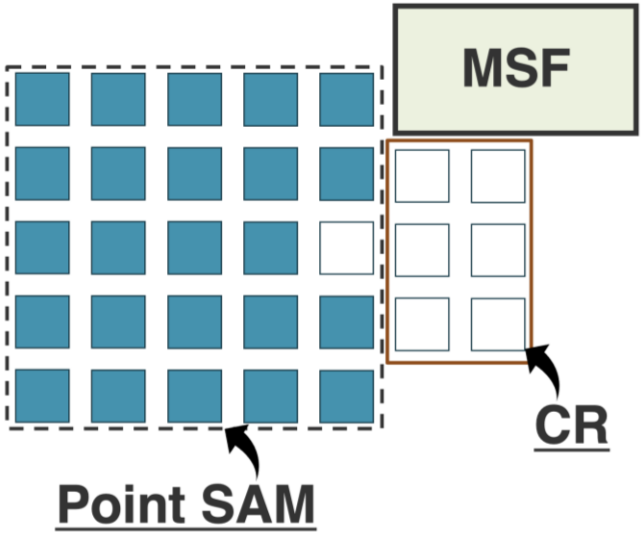
Load/ Store

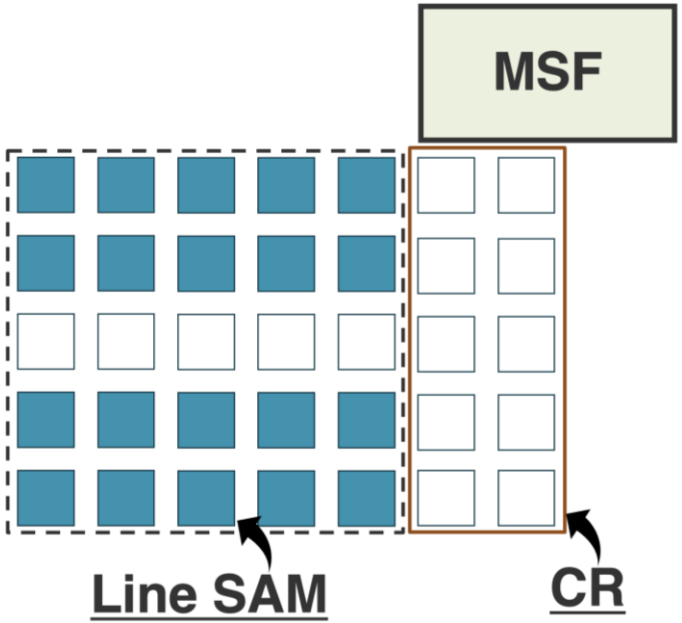
Type	Syntax	Latency	Description
Memory	`LD M C`	variable	(Load) Load logical qubit from SAM to CR
	`ST C M`	variable	(Store) Store logical qubit from CR to SAM
Preparation	`PZ.C C`	0 beat	(Zero-Init) Initialize a logical qubit to $\|0\rangle$ state
	`PP.C C`	0 beat	(Plus-Init) Initialize a logical qubit 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

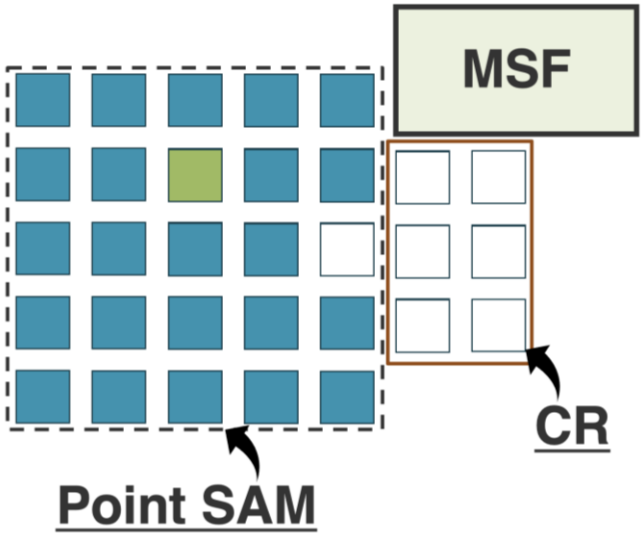
- Scan-Access Memory

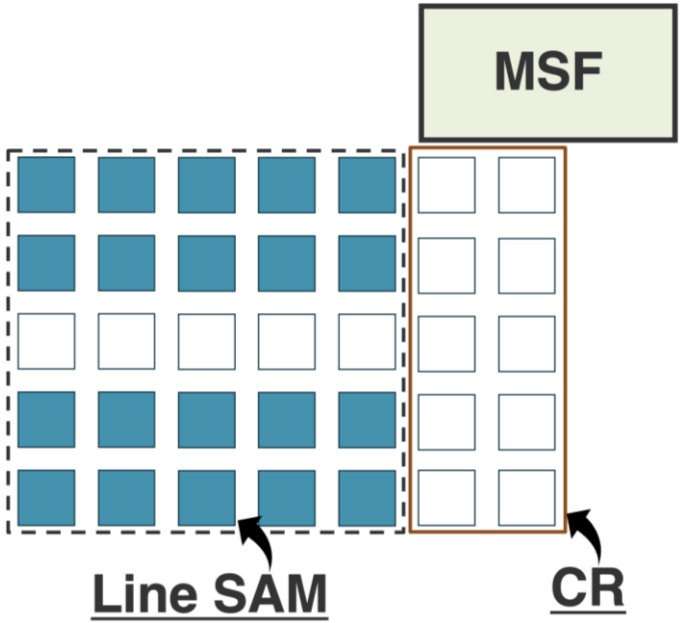




ロードストア型誤り耐性量子計算機アーキテクチャ

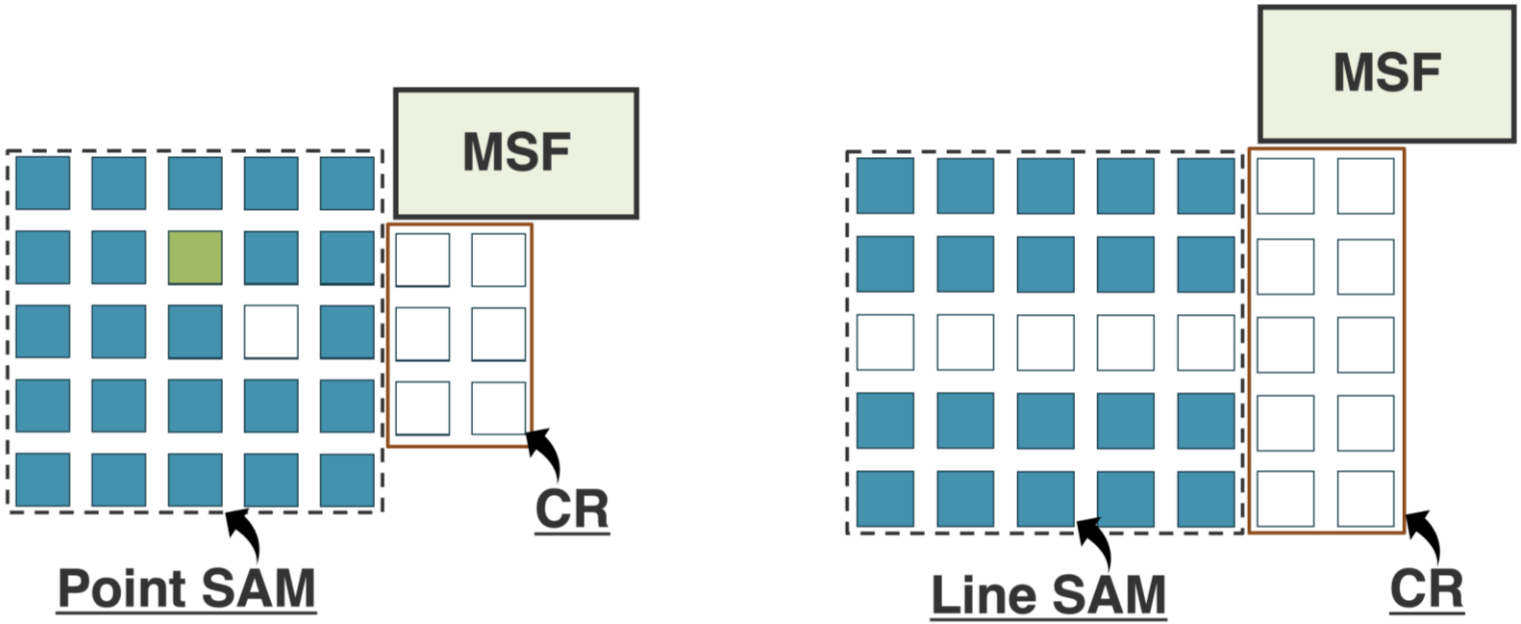
- Scan-Access Memory





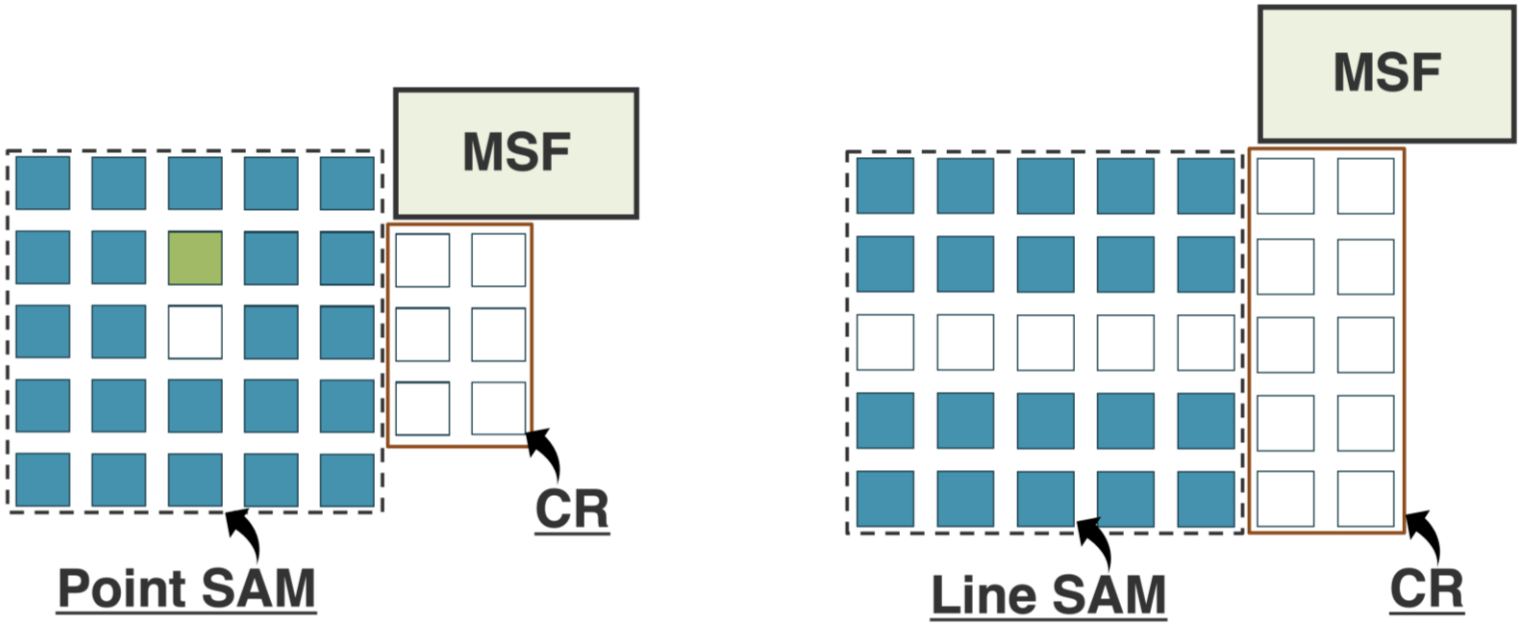
ロードストア型誤り耐性量子計算機アーキテクチャ

- Scan-Access Memory



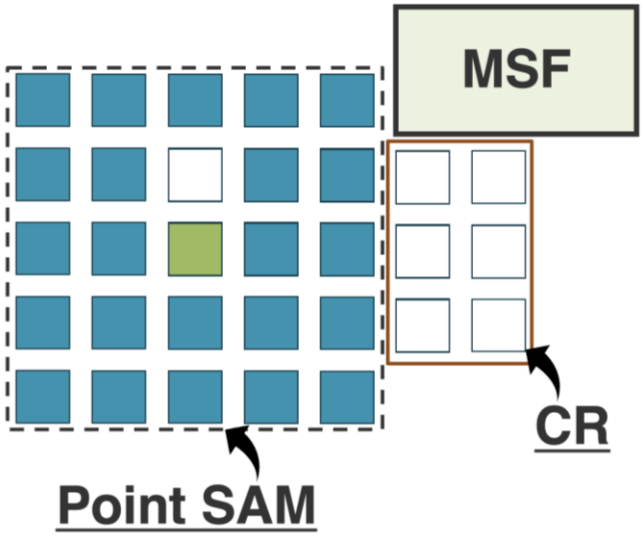
ロードストア型誤り耐性量子計算機アーキテクチャ

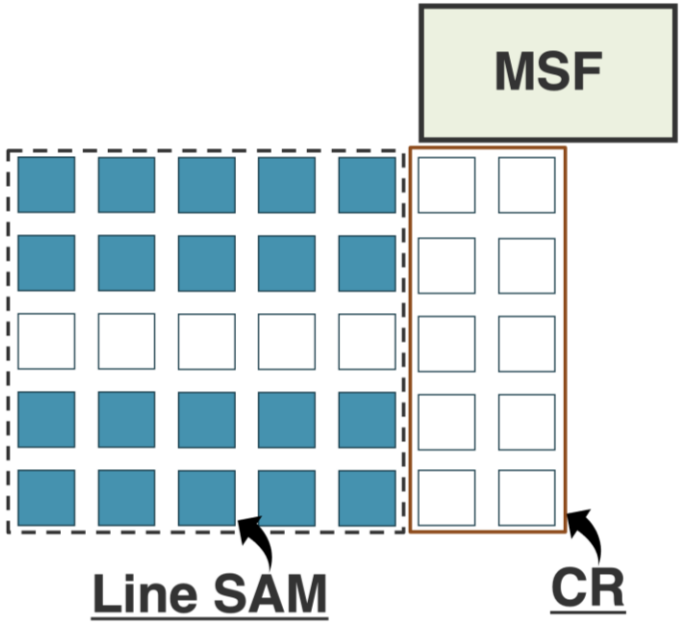
- Scan-Access Memory



ロードストア型誤り耐性量子計算機アーキテクチャ

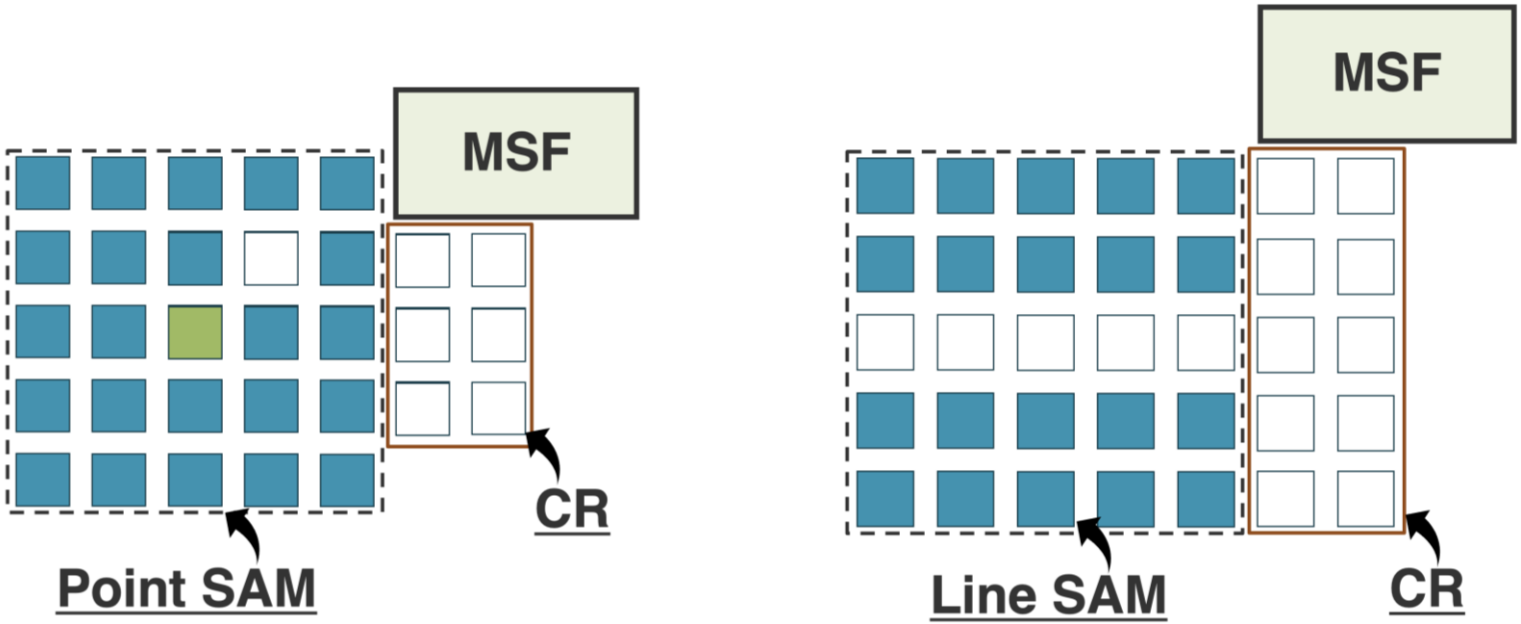
- Scan-Access Memory





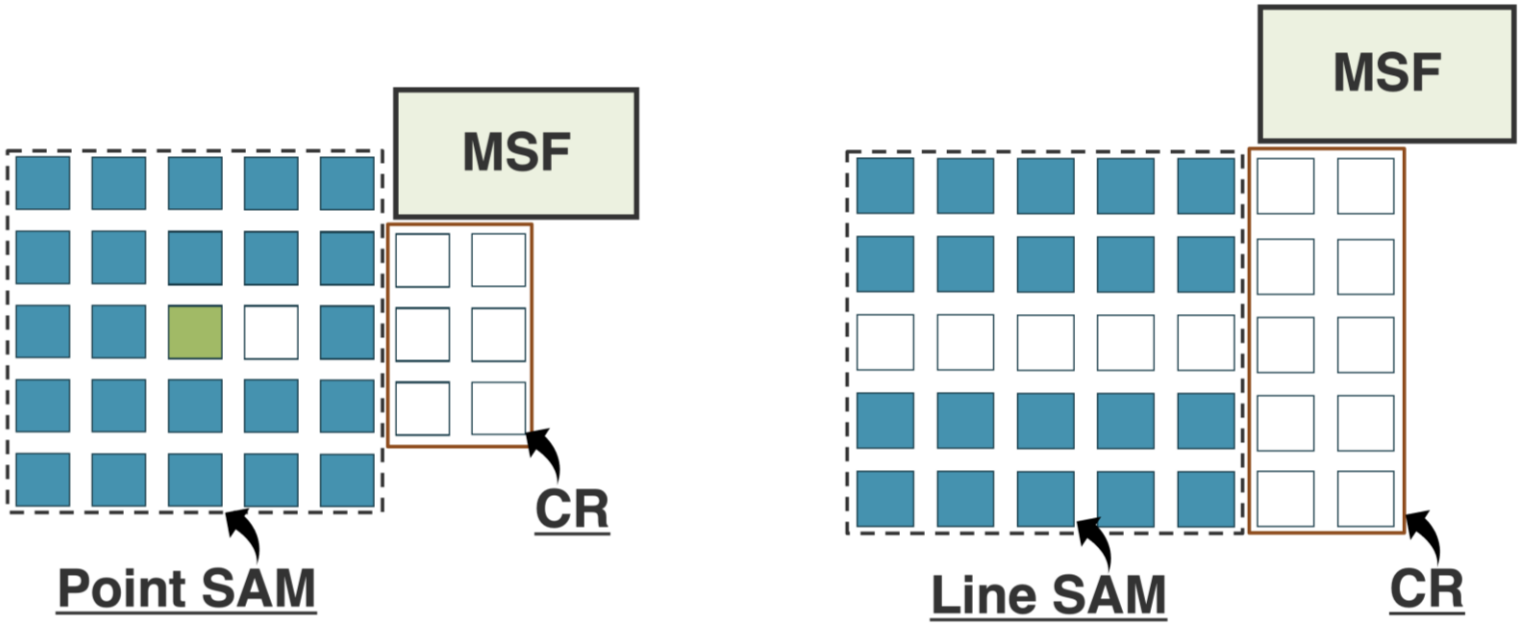
ロードストア型誤り耐性量子計算機アーキテクチャ

- Scan-Access Memory



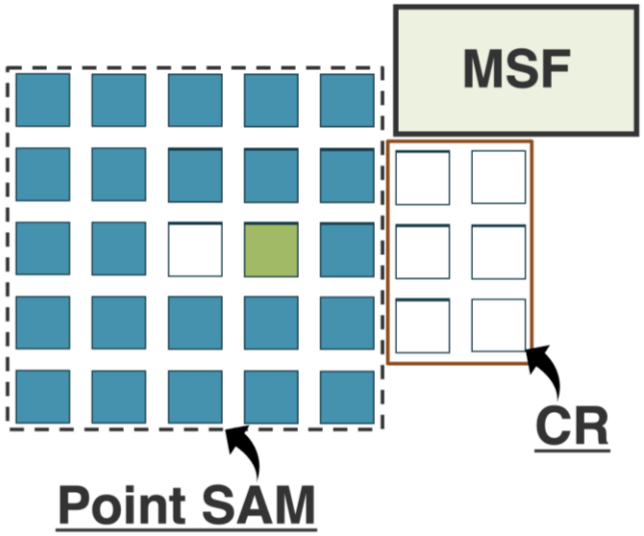
ロードストア型誤り耐性量子計算機アーキテクチャ

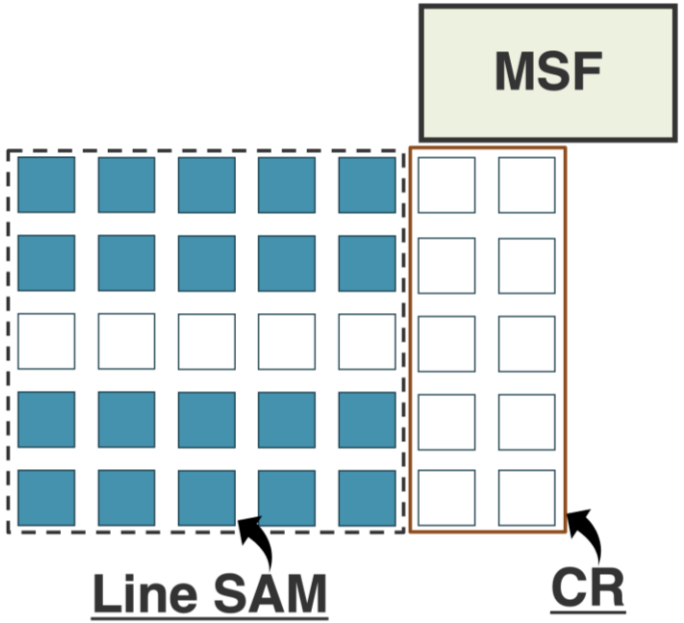
- Scan-Access Memory



ロードストア型誤り耐性量子計算機アーキテクチャ

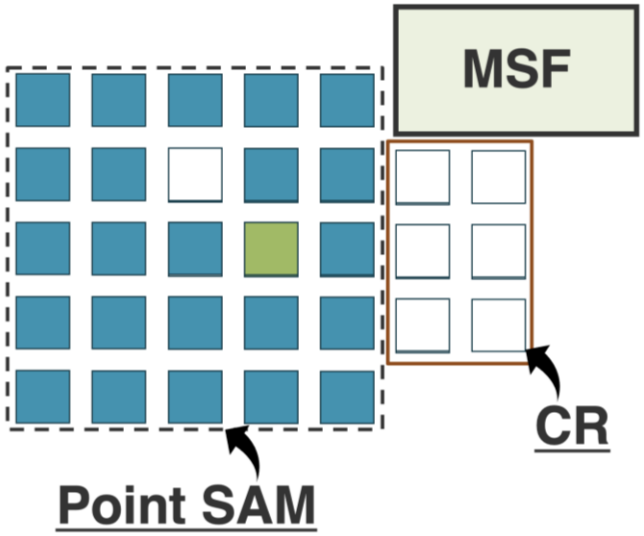
- Scan-Access Memory

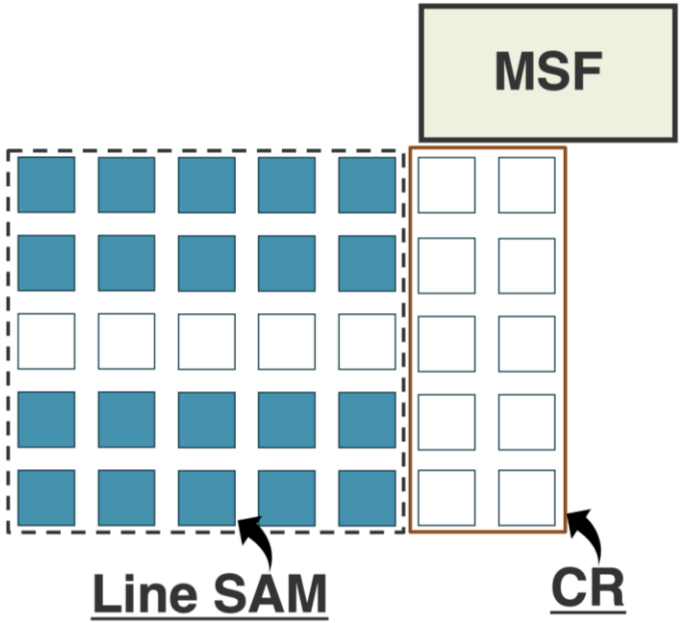




ロードストア型誤り耐性量子計算機アーキテクチャ

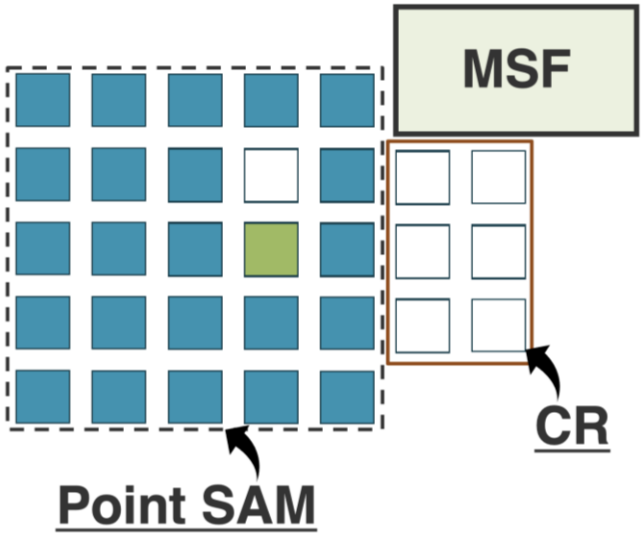
- Scan-Access Memory

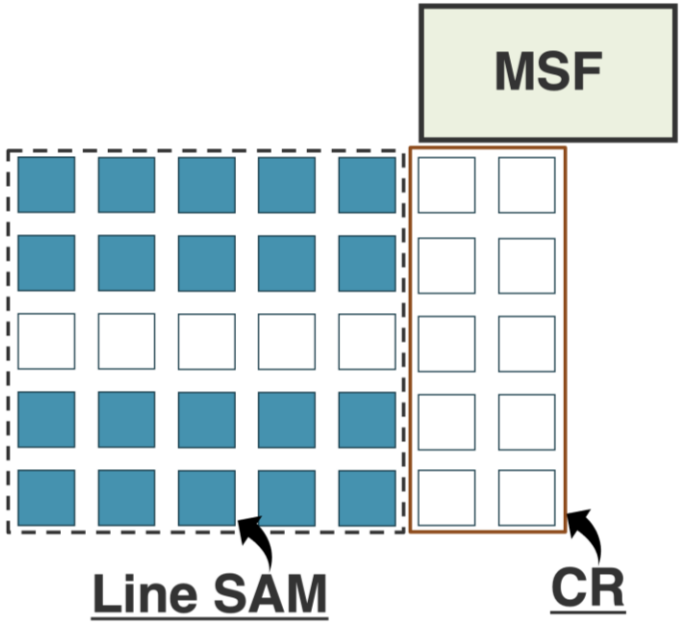




ロードストア型誤り耐性量子計算機アーキテクチャ

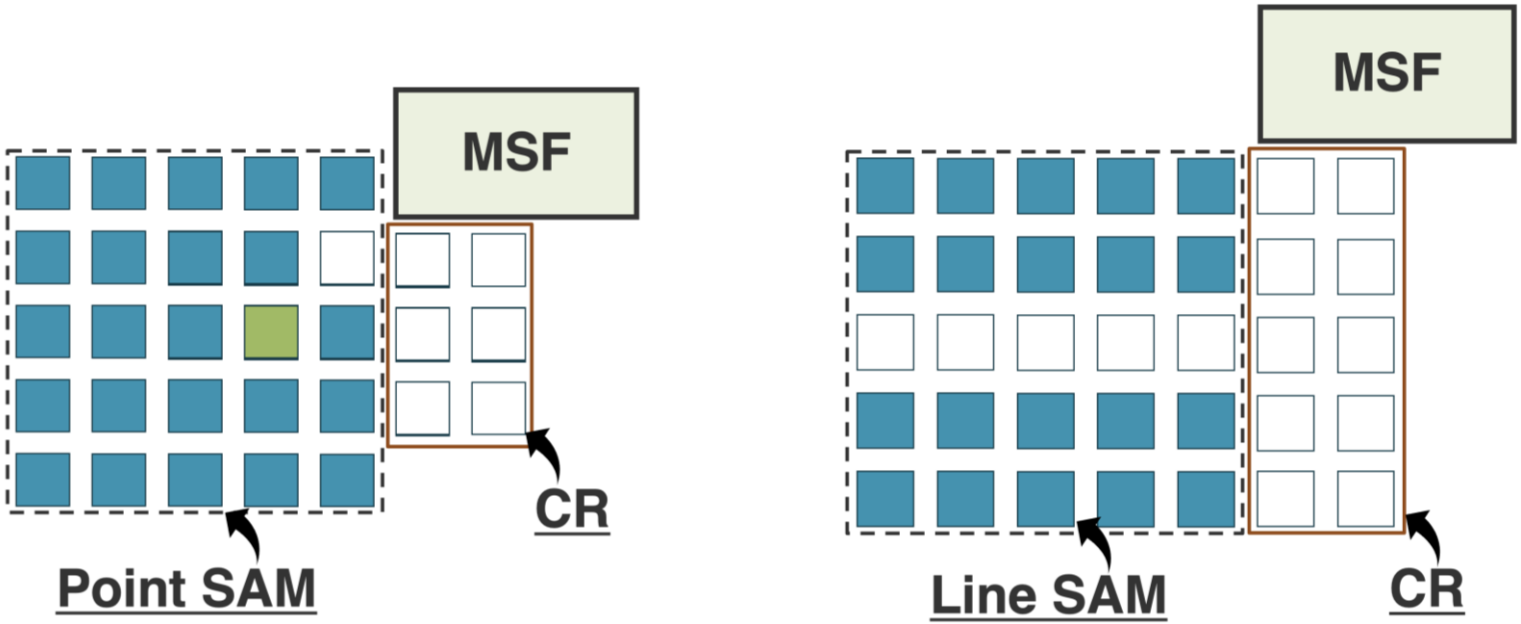
- Scan-Access Memory





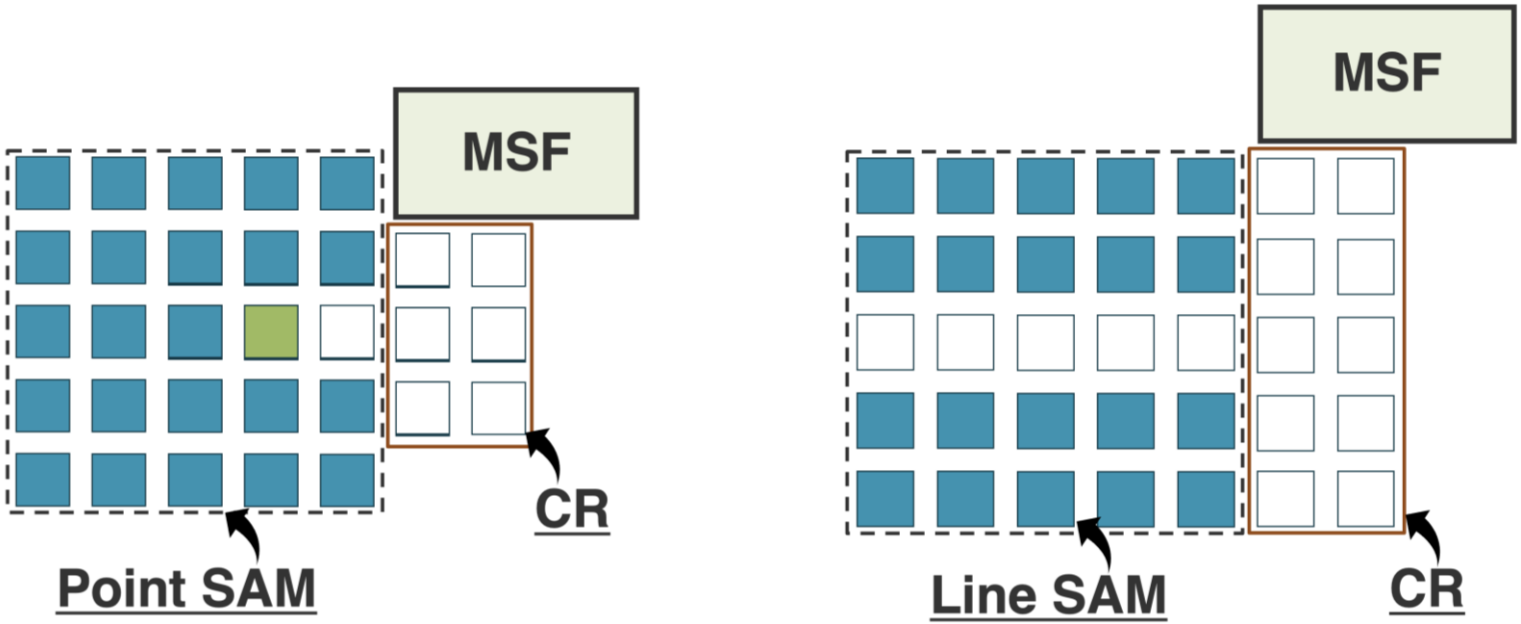
ロードストア型誤り耐性量子計算機アーキテクチャ

- Scan-Access Memory



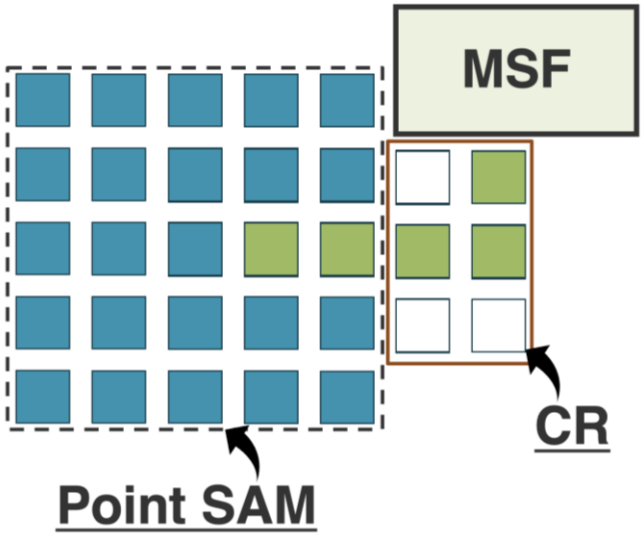
ロードストア型誤り耐性量子計算機アーキテクチャ

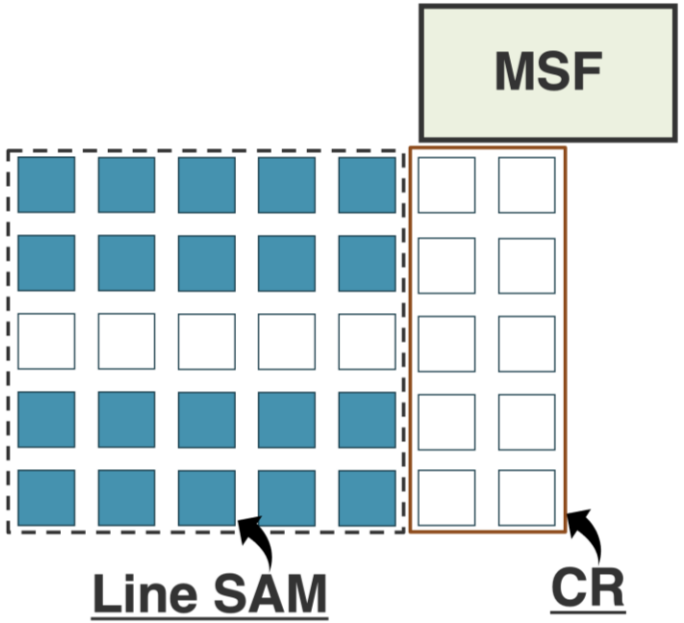
- Scan-Access Memory



ロードストア型誤り耐性量子計算機アーキテクチャ

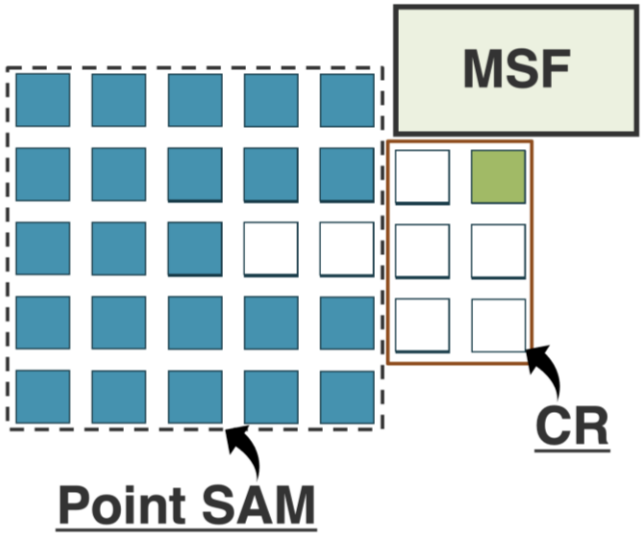
- Scan-Access Memory

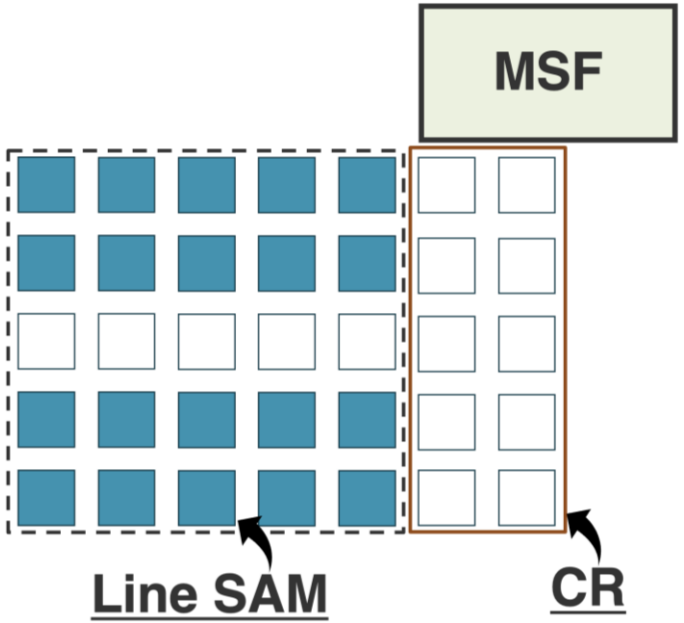




ロードストア型誤り耐性量子計算機アーキテクチャ

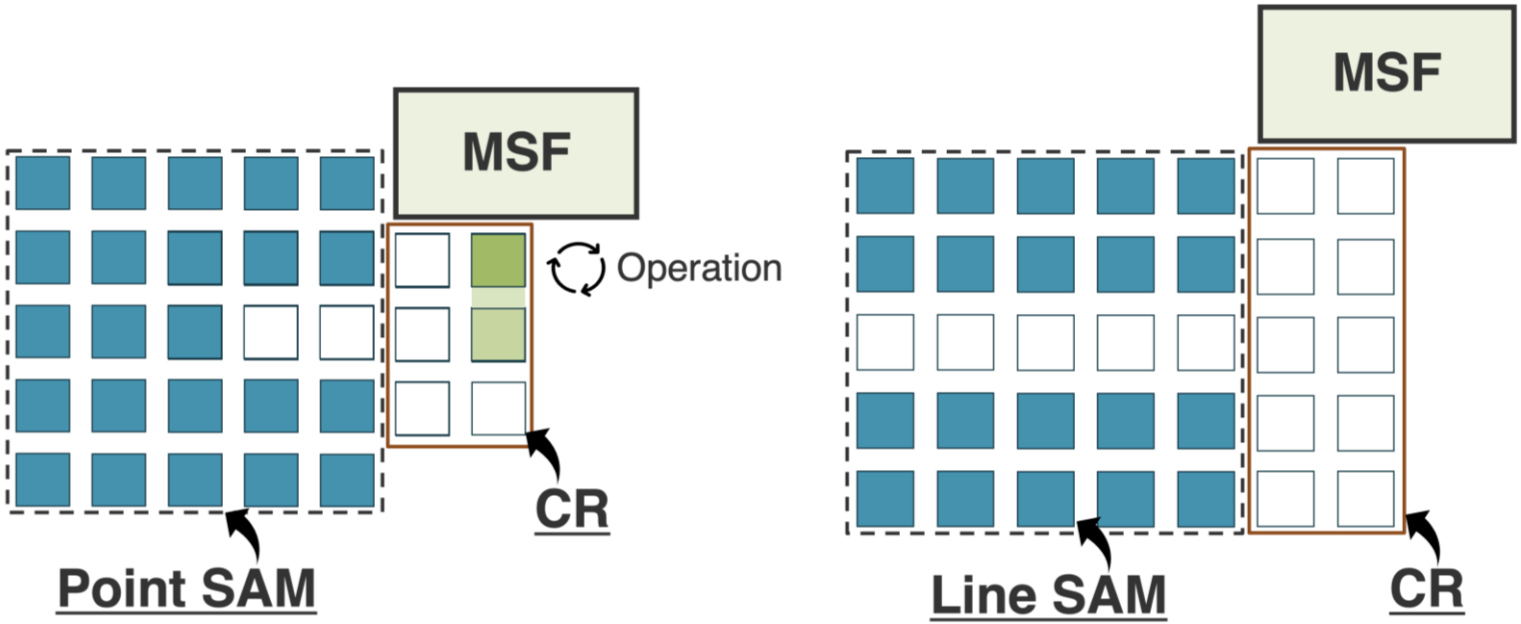
- Scan-Access Memory





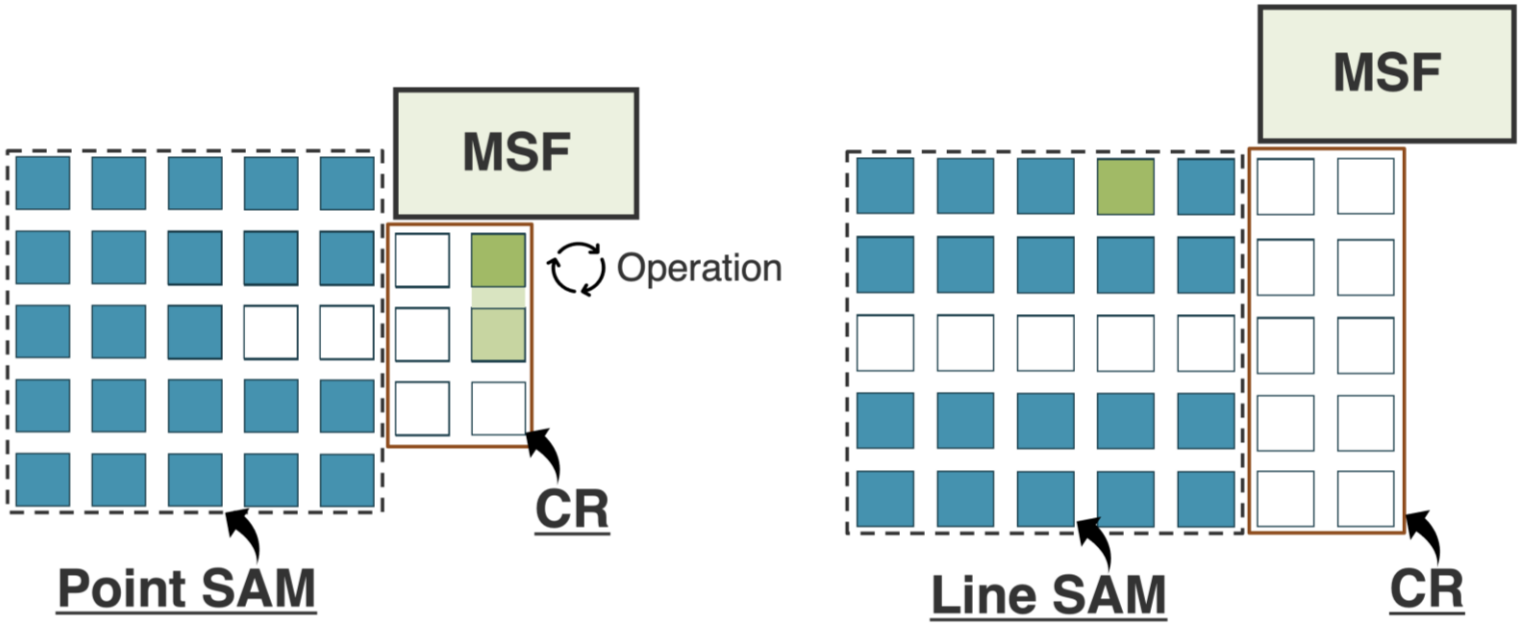
ロードストア型誤り耐性量子計算機アーキテクチャ

- Scan-Access Memory



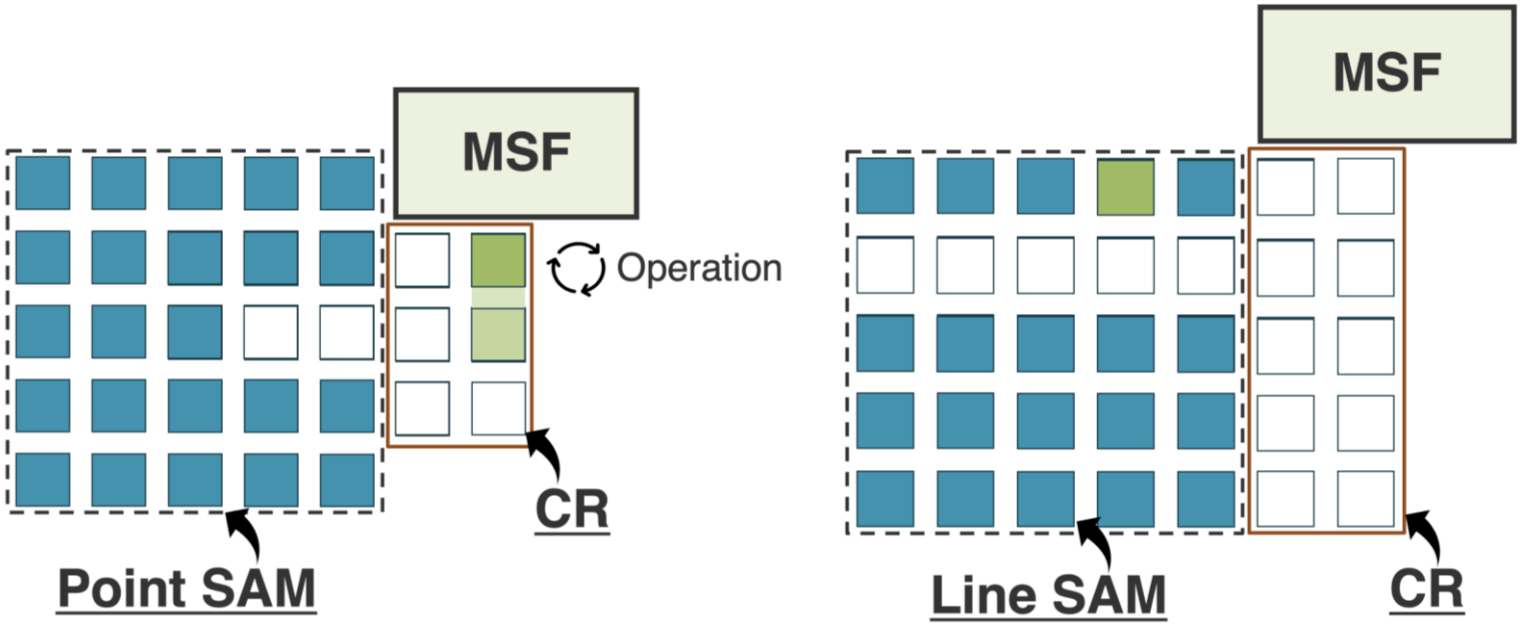
ロードストア型誤り耐性量子計算機アーキテクチャ

- Scan-Access Memory



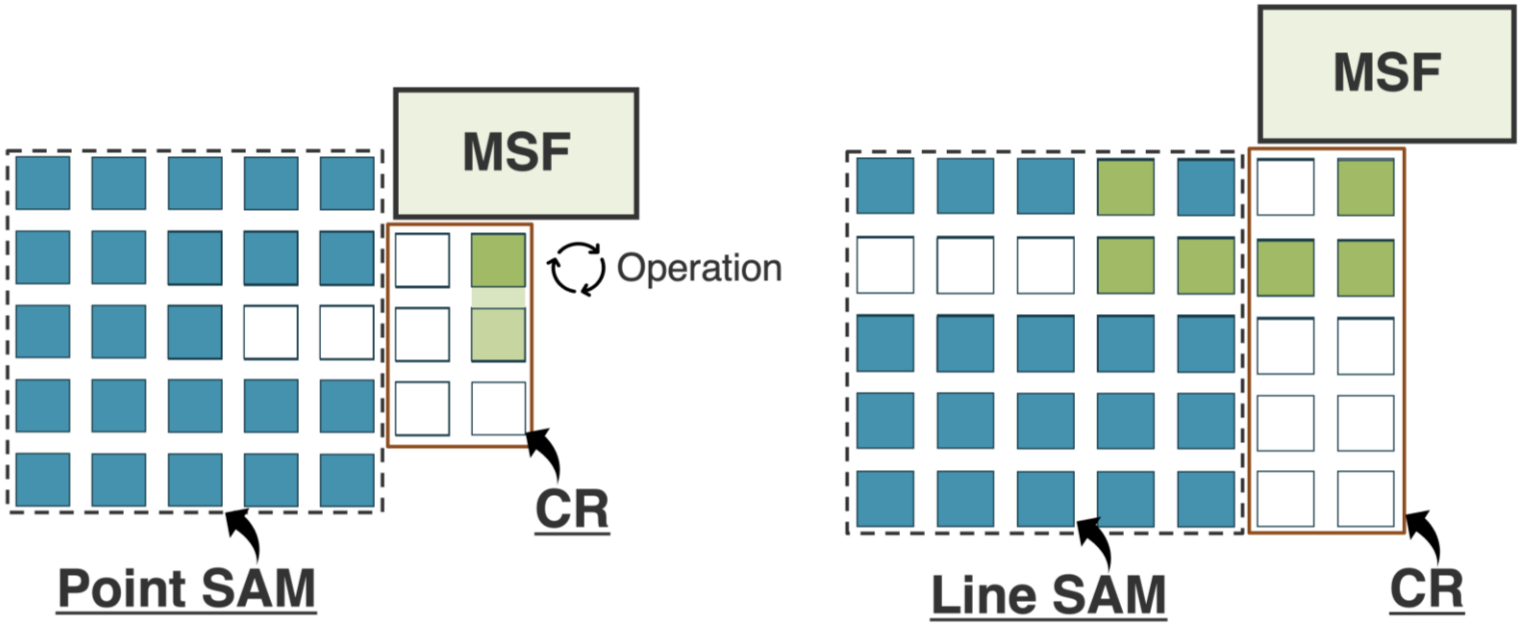
ロードストア型誤り耐性量子計算機アーキテクチャ

- Scan-Access Memory



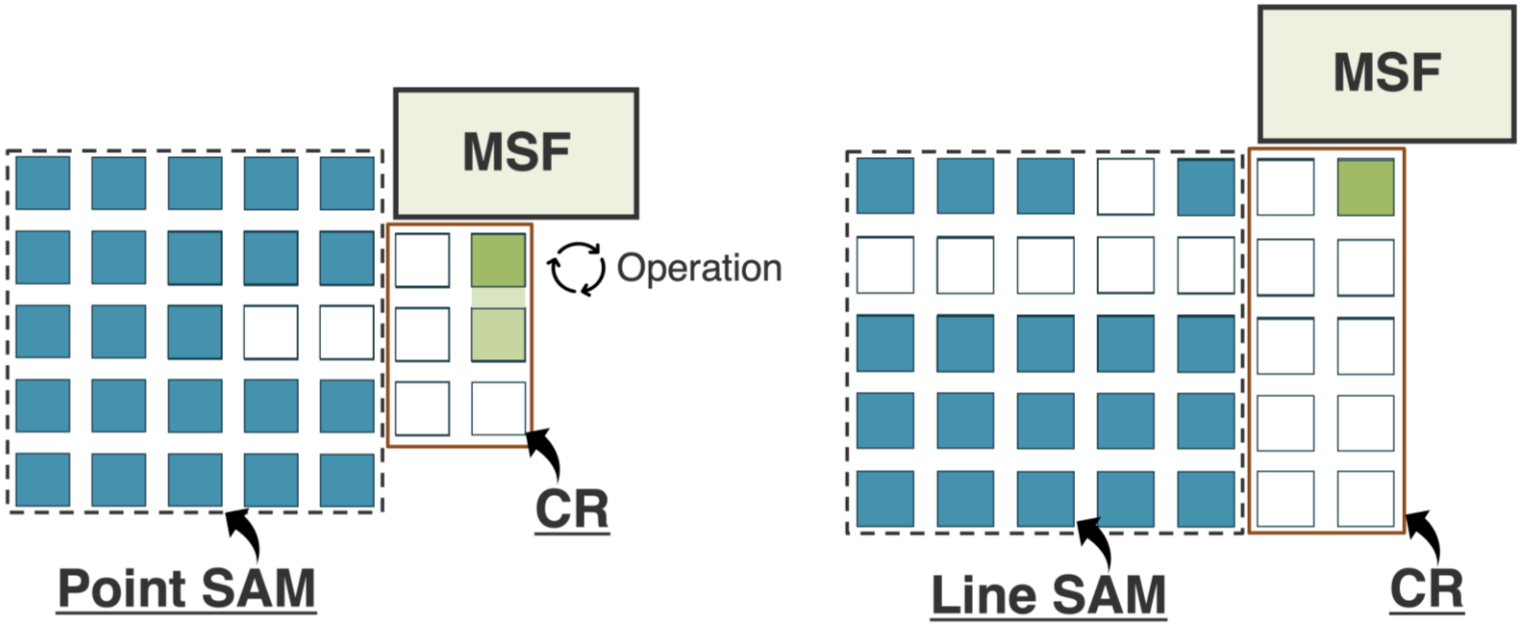
ロードストア型誤り耐性量子計算機アーキテクチャ

- Scan-Access Memory



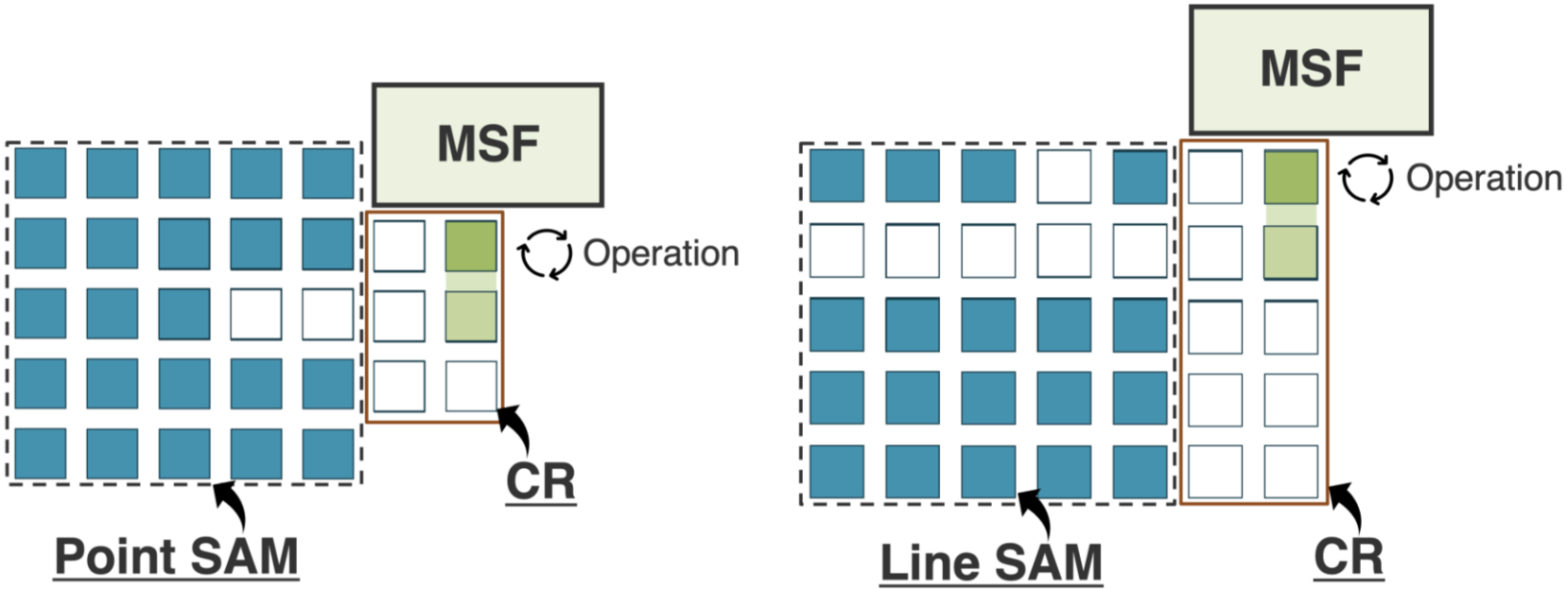
ロードストア型誤り耐性量子計算機アーキテクチャ

- 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

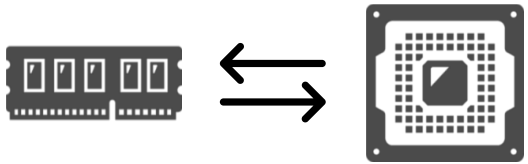
- Scan-Access Memory



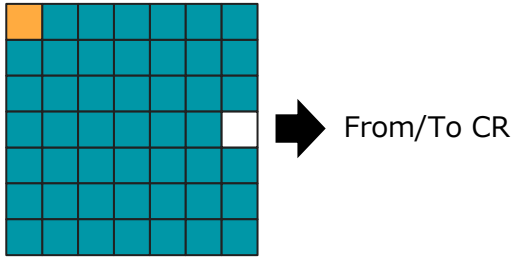
ロードストア型を採用するコスト

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



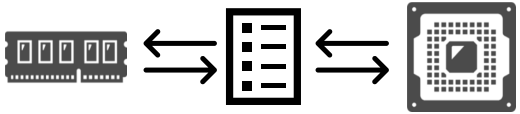
最悪ケースでは最もアクセスの遅いセルが
繰り返し呼び出される形になる



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

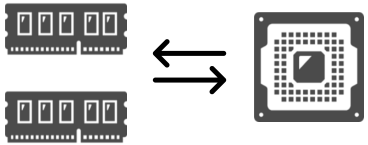
キャッシュ / 専用メモリ

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



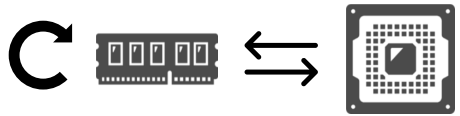
マルチチャンネルメモリ

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

メモリ内部に軽量な計算機構を入れ
一部の計算をメモリ内部で行う



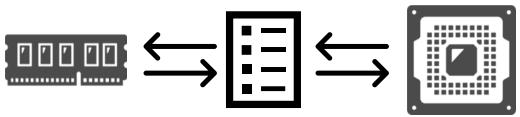
キャッシュなどの技術はメモリアクセスが典型的な特性を持っていることを仮定している　量子も同様の性質を持つか？

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

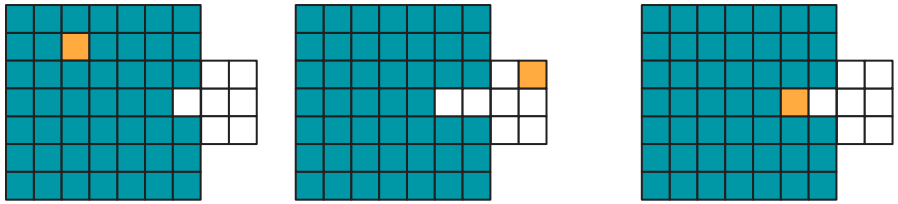
指針：キャッシュ/専用メモリ

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

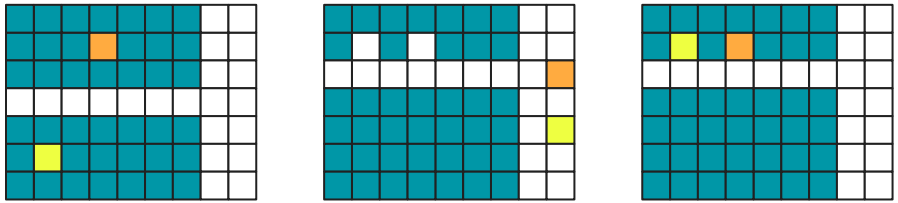


最適化１：Locality-aware store

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

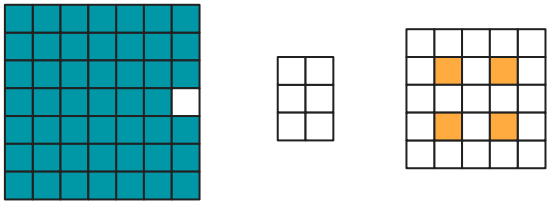


同時に利用したデータは同時にアクセスしやすい場所に戻す
これにより、空間的な局所性を活用し典型的ロードを早くする



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

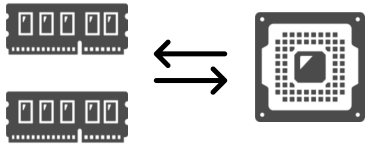


ロードストアの影響を軽減する最適化

最適化機構の提案

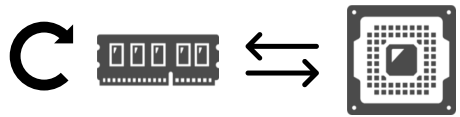
指針２：マルチチャンネルメモリ

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



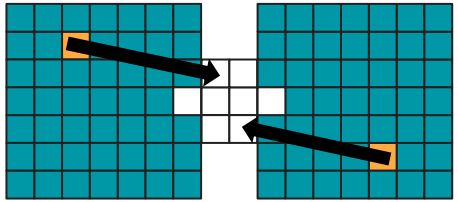
指針３：インメモリ計算

メモリ内部に軽量な計算機構を入れ
一部の計算をメモリ内部で行う



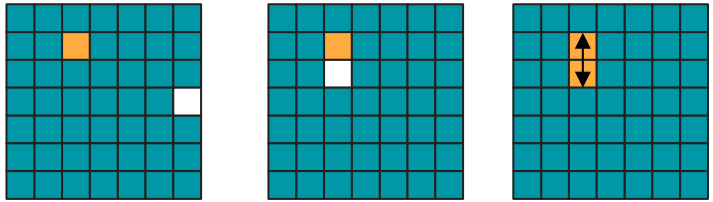
最適化３：Multi-bank SAM

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



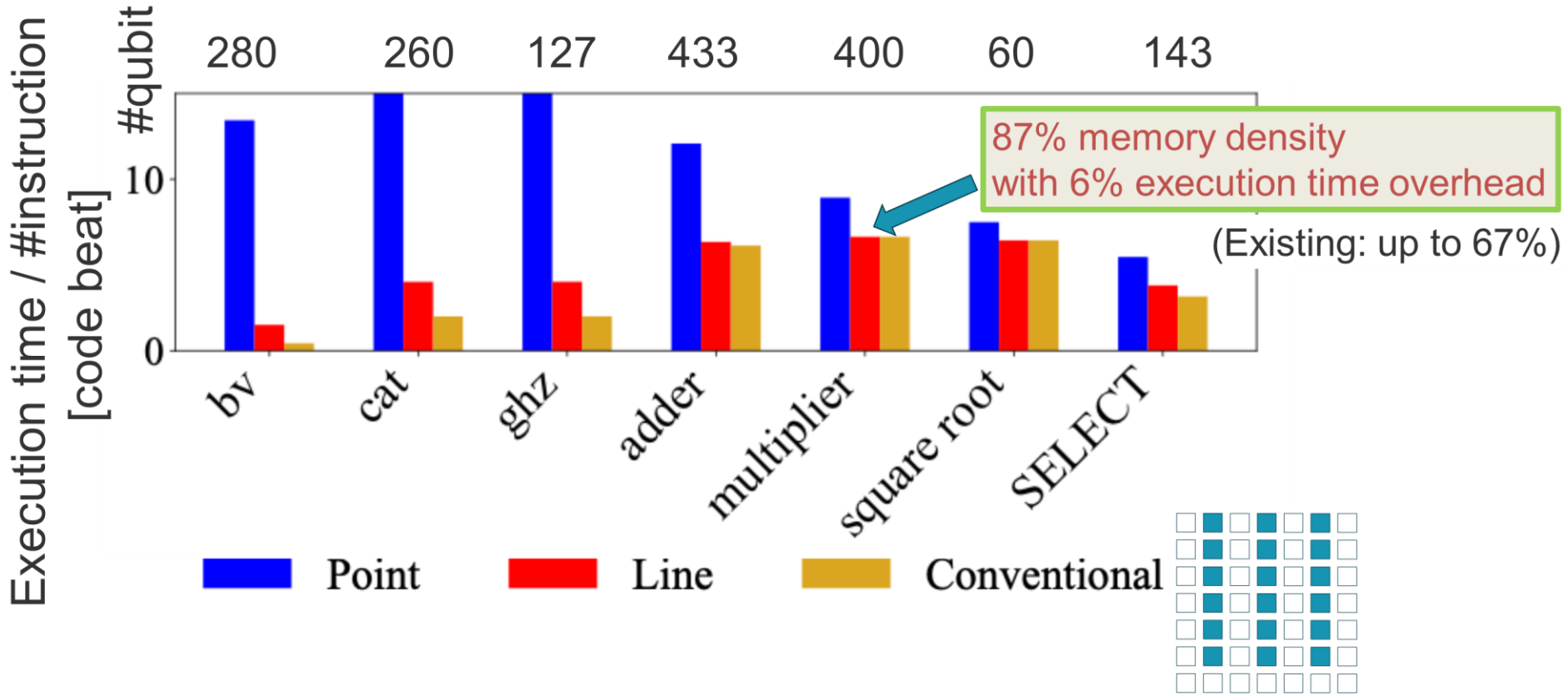
最適化４：In-memory operations

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

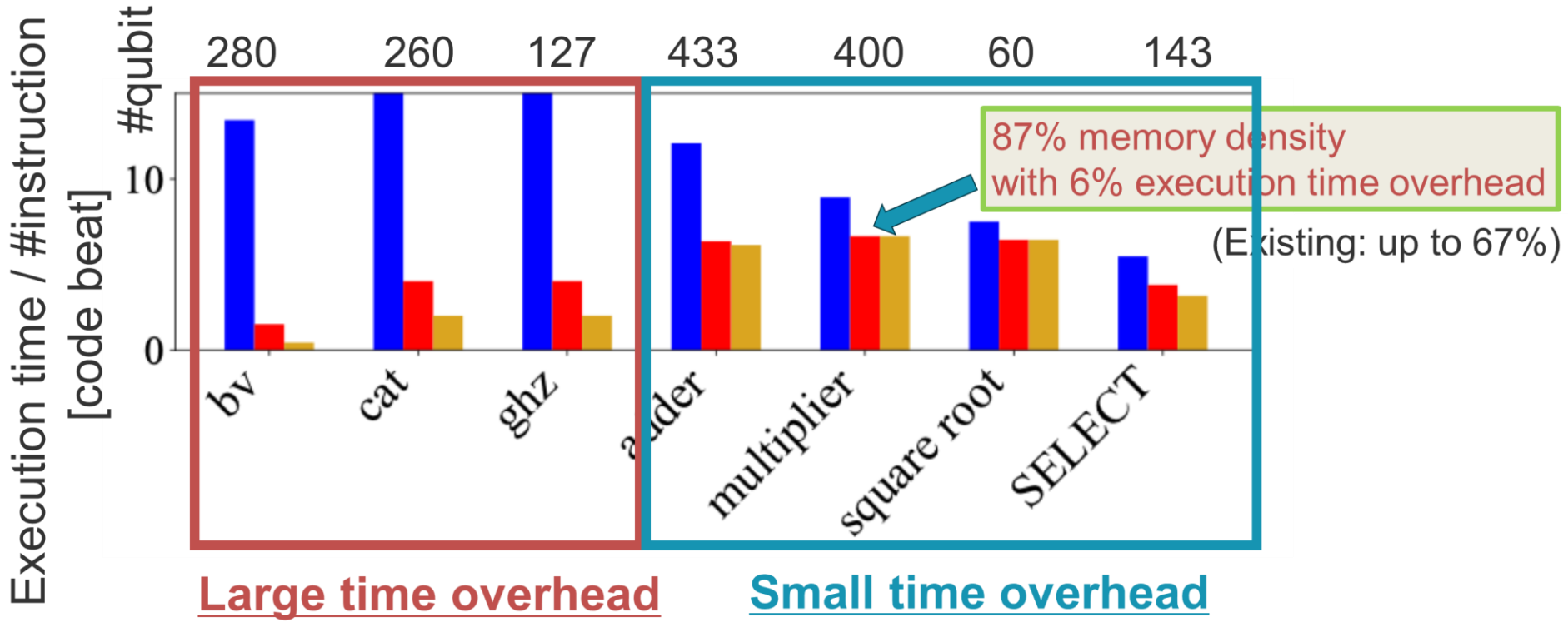


論理HゲートやSゲートをその場で実行

Basic evaluation for benchmark program



Basic evaluation for benchmark program



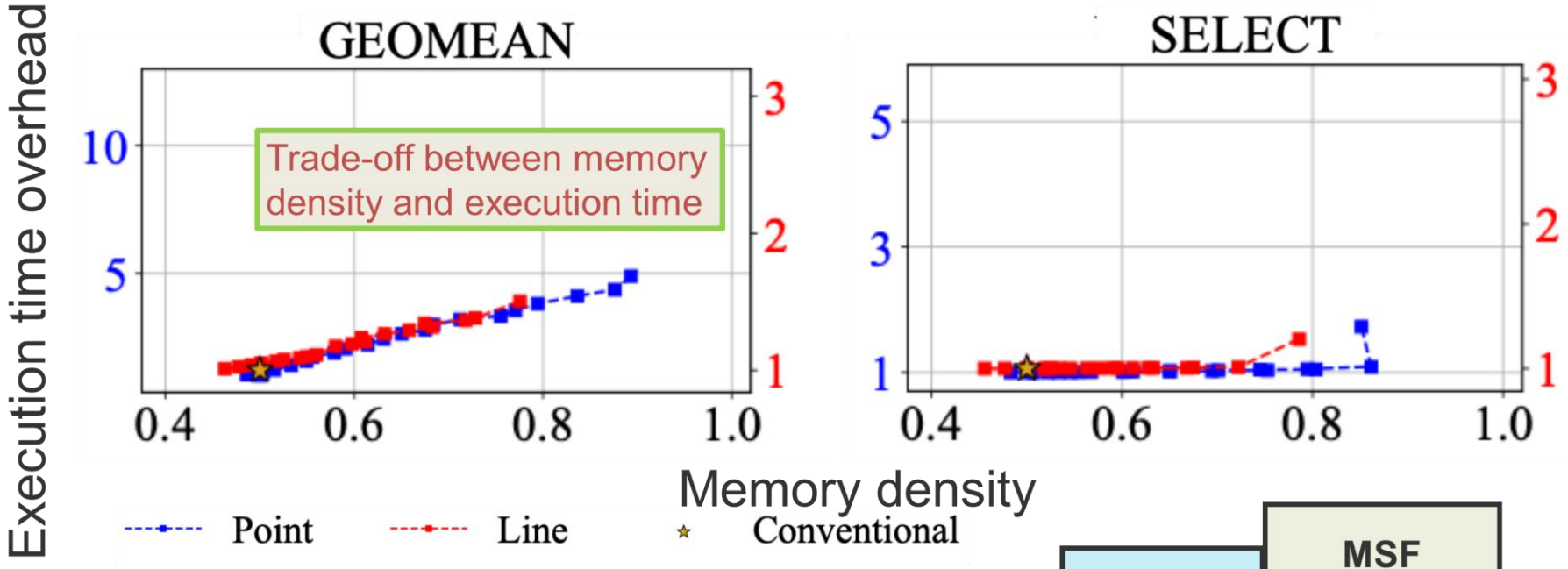
Large time overhead

- Less T-gates
- Not bottleneck in practice

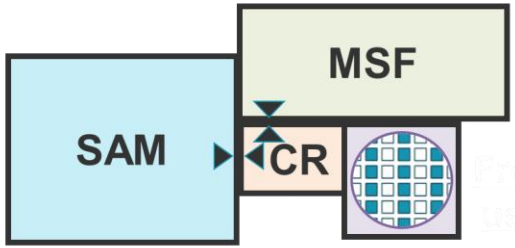
Small time overhead

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

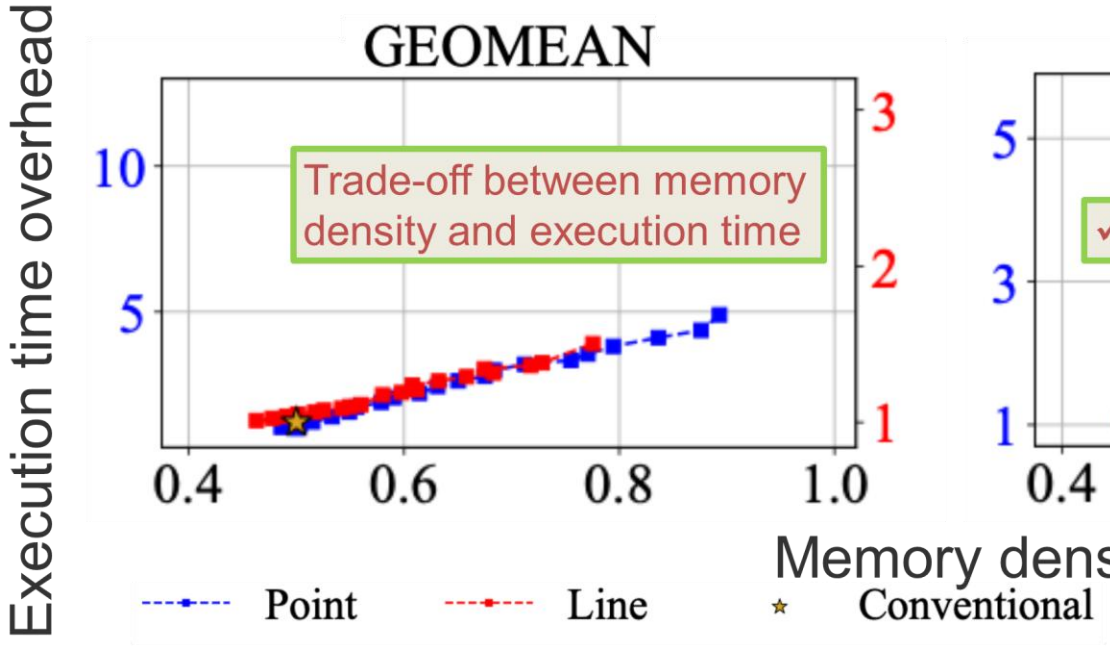
Hybrid floorplan for benchmark program



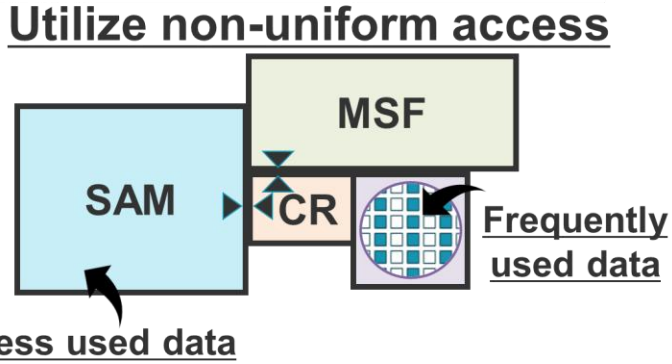
Each Points: how large the conventional floorplan in Hybrid floorplan (Left is larger)



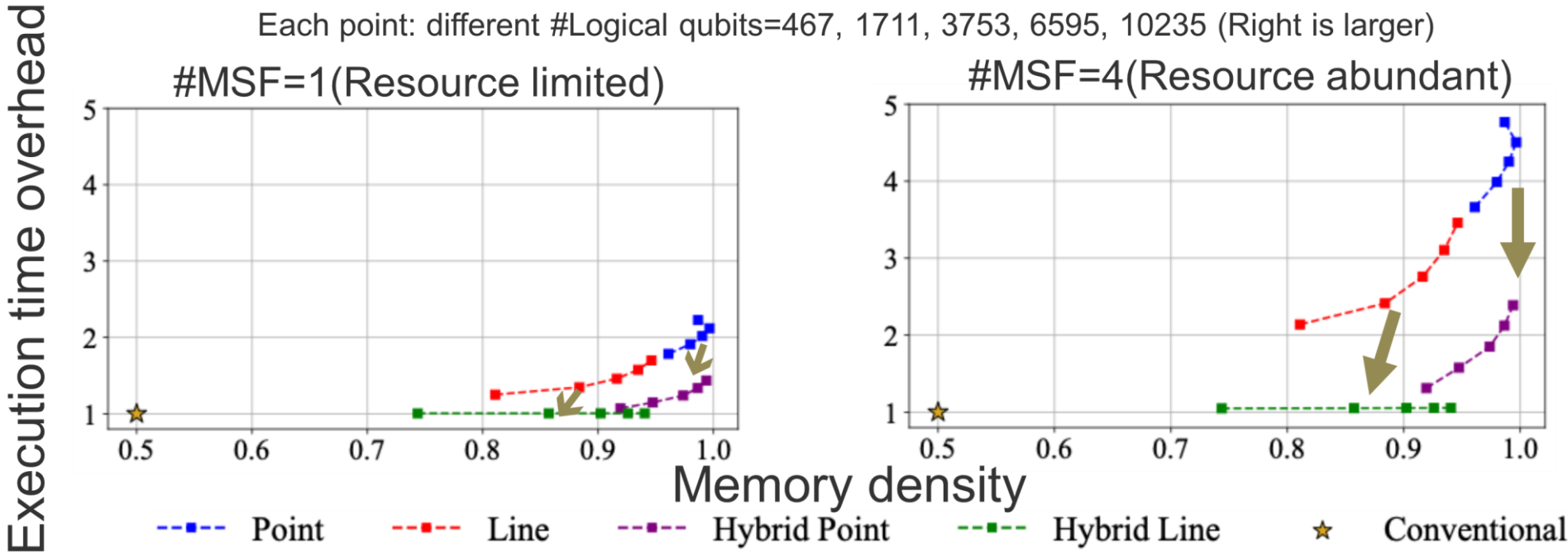
Hybrid floorplan for benchmark program



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

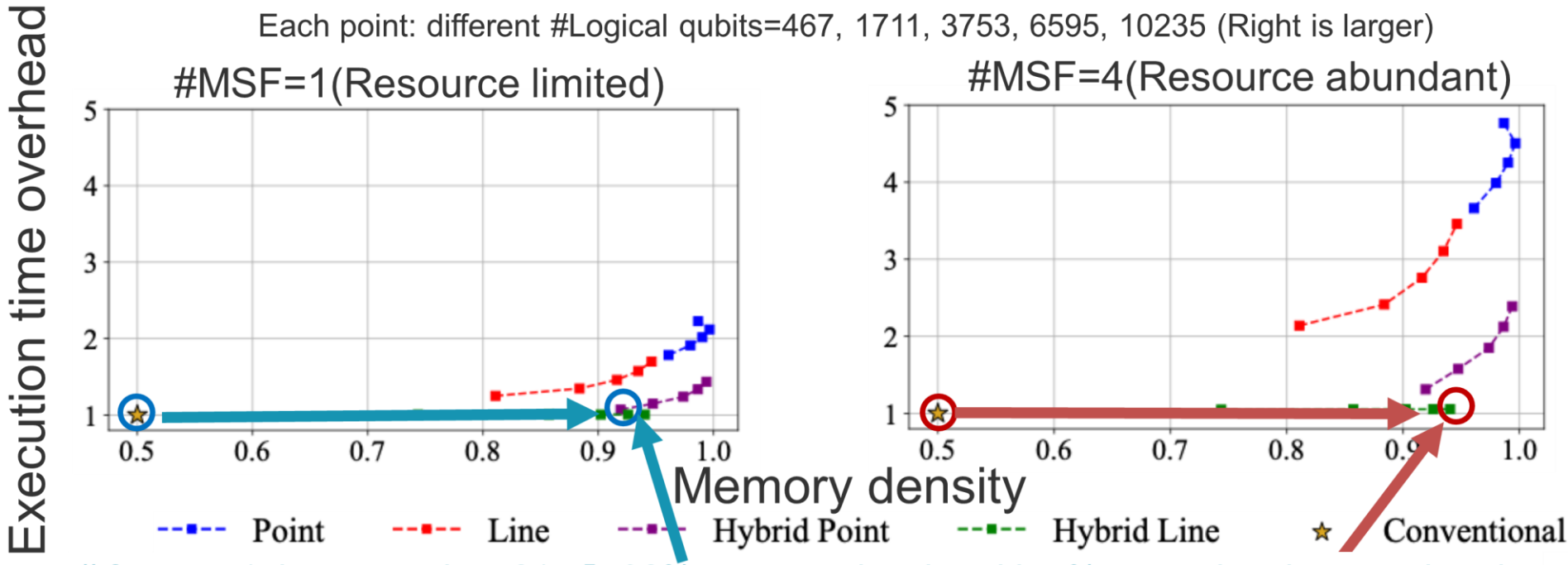


Optimized hybrid floorplan evaluation for SELECT circuit



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

Optimized hybrid floorplan evaluation for SELECT circuit

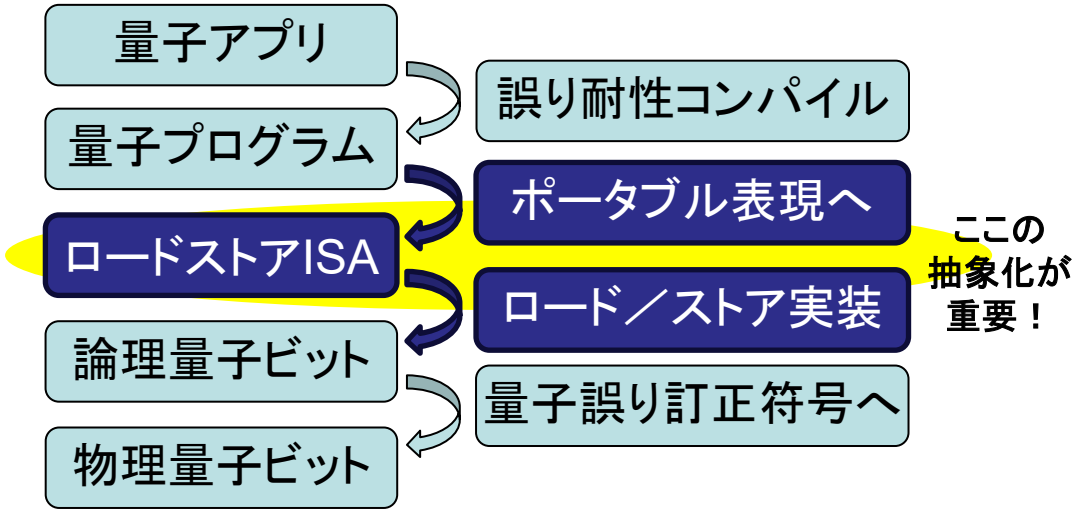


factory=1, instance size=21 ➔ 92% memory density with 7% execution time overhead by Hybrid point-SAM architecture

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

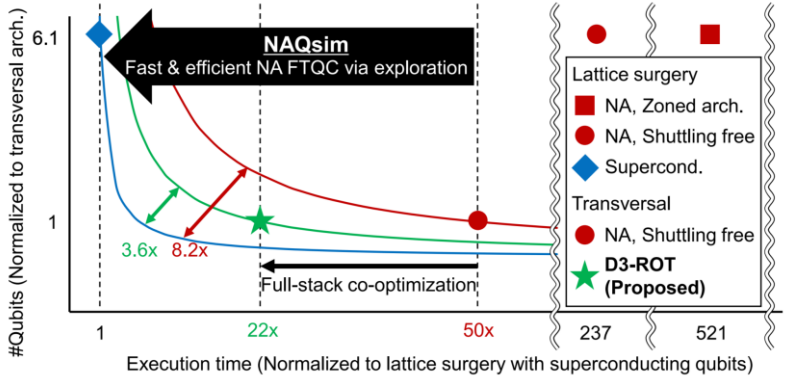
- 量子計算機アーキテクチャ分野の研究動向
  - 計算機アーキテクチャの研究対象・隣接分野
  - 計算機アーキテクチャ分野における量子計算機関連の研究動向
- 量子計算機アーキテクチャに関連する研究紹介
  - 超伝導ディジタル回路を用いた量子誤り推定機構（博論、DAC'21、HPCA'22）
  - 誤り耐性量子計算（FTQC）におけるロードストア型アーキテクチャ（HPCA'25）

未出版内容のため公開版では削除

  - 中性原子を対象としたフルスタックFTQCシミュレーションフレームワーク（MICRO'26）
- 量子計算機アーキテクチャ研究の魅力・まとめ

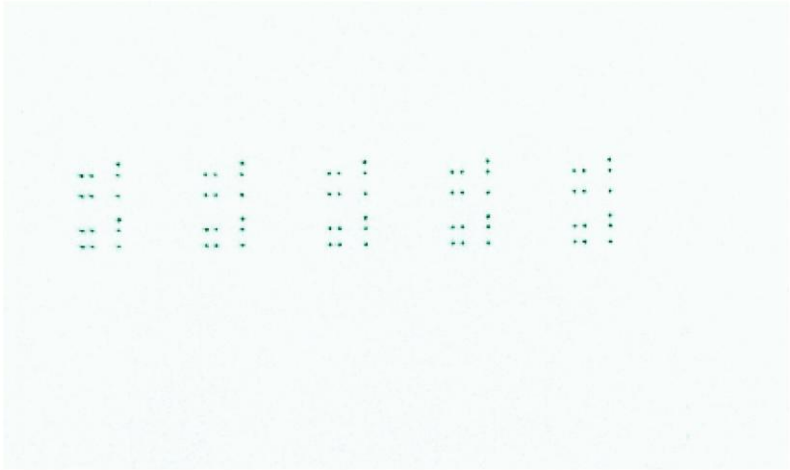
NAQsim

- 研究背景
  - 中性原子FTQCは超伝導に比べて237倍遅い（∵ 1000倍遅い物理ゲート操作）
  - 良いアーキテクチャを設計したいがシミュレーションフレームワークがない
    - 障壁：コンパイラ、制御ハードウェア、Qubit planeを同時に検討する必要あり
- 提案
  - フルスタックシミュレーション
    フレームワーク**NAQsim**を開発
  - NAQsimを用いたアーキテクチャ探索により
    D3-ROT architectureを提案
    - Space-time volumeの観点で
      超伝導との差を237倍 -> **3.6倍**に

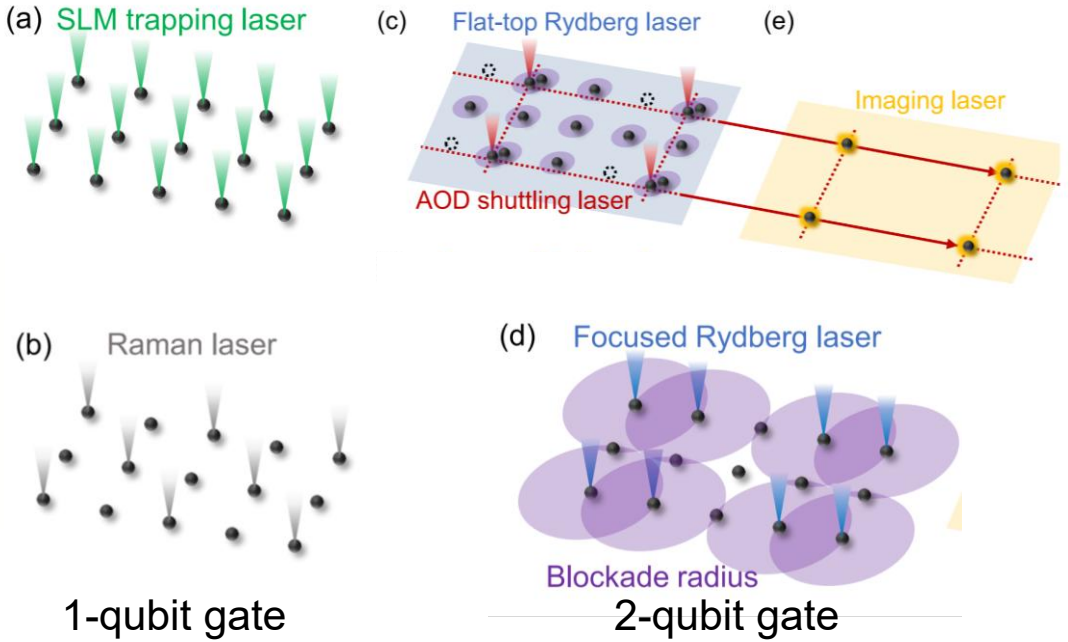


Y. Ueno, S. Sunami, T. Hinokuma, Y. Suzuki, A. Goban, H. Yamasaki, T. Teruo, I. Byun, “NAQsim: Full-Stack Architecture Simulation Framework for Fast and Space-Efficient Neutral Atom Quantum Computing”, MICRO’26.

中性原子量子コンピュータ



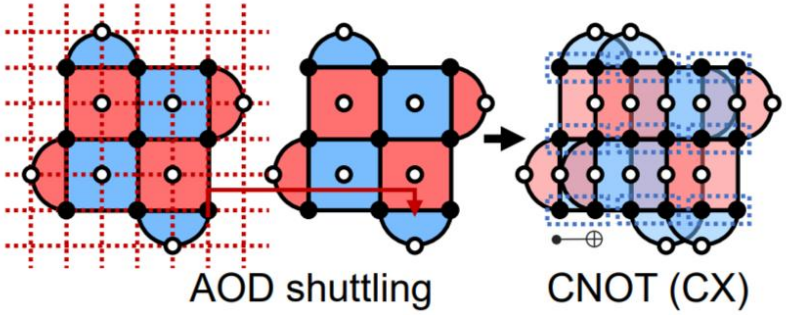
D. Bluvstein et al. “Logical quantum processor based on reconfigurable atom arrays”, Nature 626, 58-65



- 量子ビット＝中性原子を動かしながら計算を行う
  - 通常はspatial light modulator (SLM) に原子がトラップされている
  - Acousto-optic deflector (AOD) が原子を行単位・列単位でつまんで動かせる

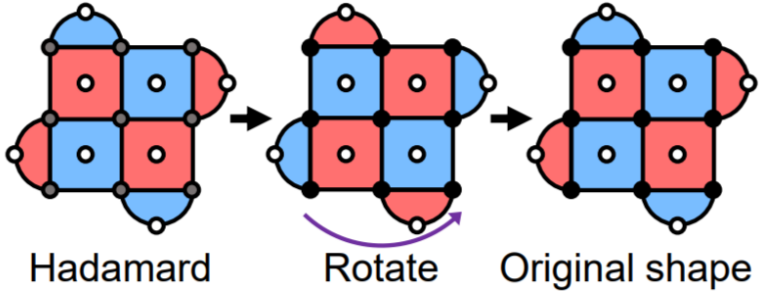
主な論理演算

Transversal CNOT



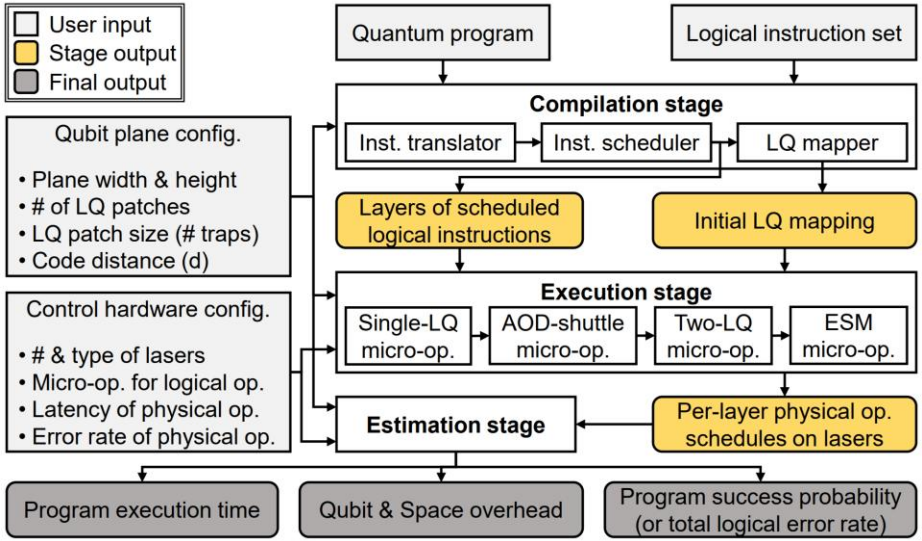
- 対象の論理ビットの片方をAODで移動させもう片方と重ねる
- 対応するデータ量子ビット間でそれぞれCNOTを行う

Transversal Hadamard (H) + 90-degree rotation



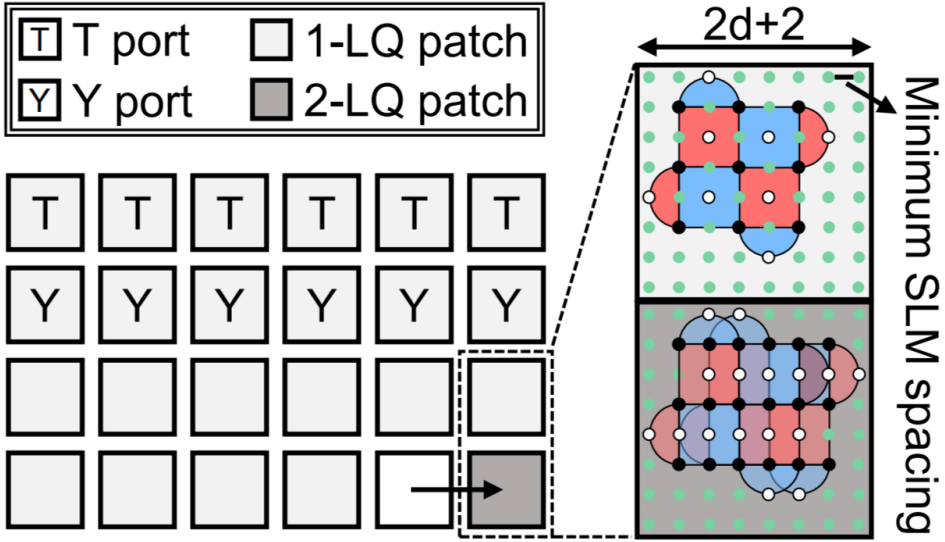
- 全データ量子ビットにH gateを実行
- H gateによりXとZの境界が入れ替わるので論理ビットを物理的に90°回転して補正
  - 回転の方法は色々ある（後述）

フルスタックFTQCシミュレータNAQsim



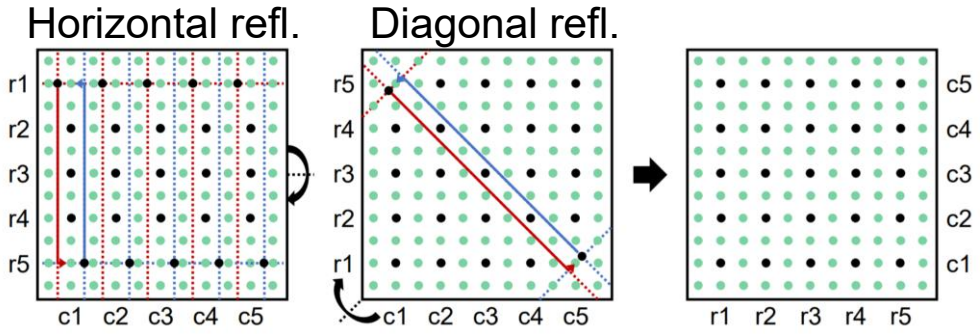
- 物理的なレーザー操作や原子の移動の制約までをサイクルレベルで評価できるシミュレーションフレームワーク
  - 入力：量子プログラム、命令セット、Qubit plane config.、Control HW config.
  - 出力：プログラム実行時間、必要な量子ビット数・面積、プログラム成功確率

Baseline qubit plane architecture

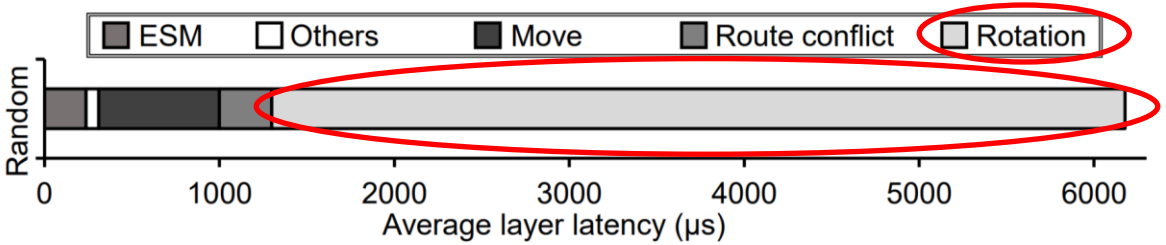


- 1パッチあたり2論理ビットを保持できるだけのSLMあり
  - Transversal CNOT実行時は一時的に1パッチに2論理ビット保持
- 1パッチあたりの面積はSLM間隔が最小となるように設定
  - RotationにはSpace-efficient reflectionしか使えない

Baseline architectureにおけるボトルネック



Space-efficient reflectionの手順

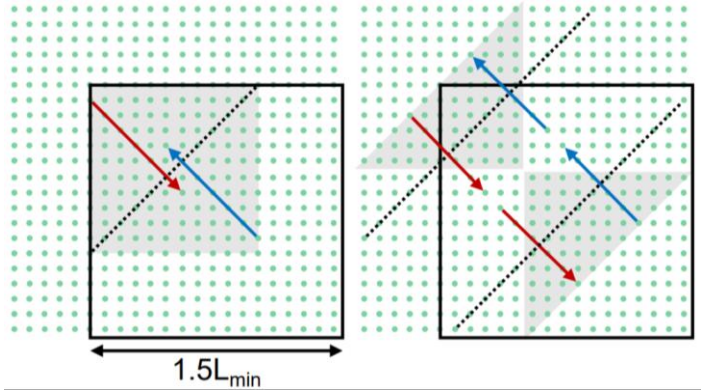


NAQsimで算出した実行時間の内訳

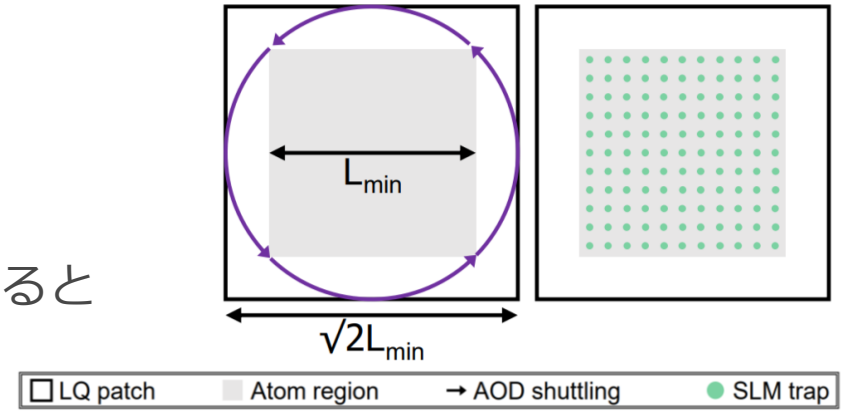
- Space-efficient reflectionは$O(d)$の操作時間がかかり遅い

より高速なRotation手法

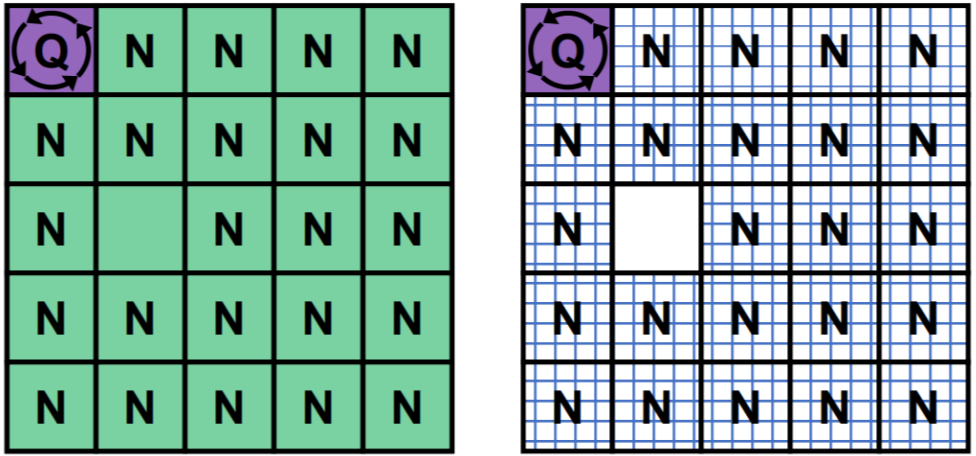
- Time-efficient reflection
  - + $O(\log(d))$ の操作時間
  - + SE refl. と比べて追加のAOD不要
  - - パッチサイズが2.25倍
  - - 各量子ビットへの操作数増 -> 高エラー率

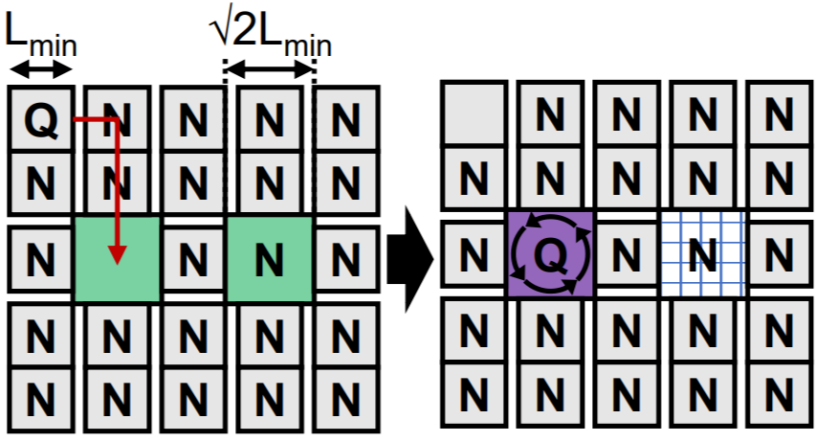


- Direct rotation
  - + $O(1)$の操作時間
  - - 専用AODが必要
  - - パッチサイズが2倍
  - - 全パッチでdirect rotationをサポートすると非効率的（次スライド）



Direct rotationの問題点

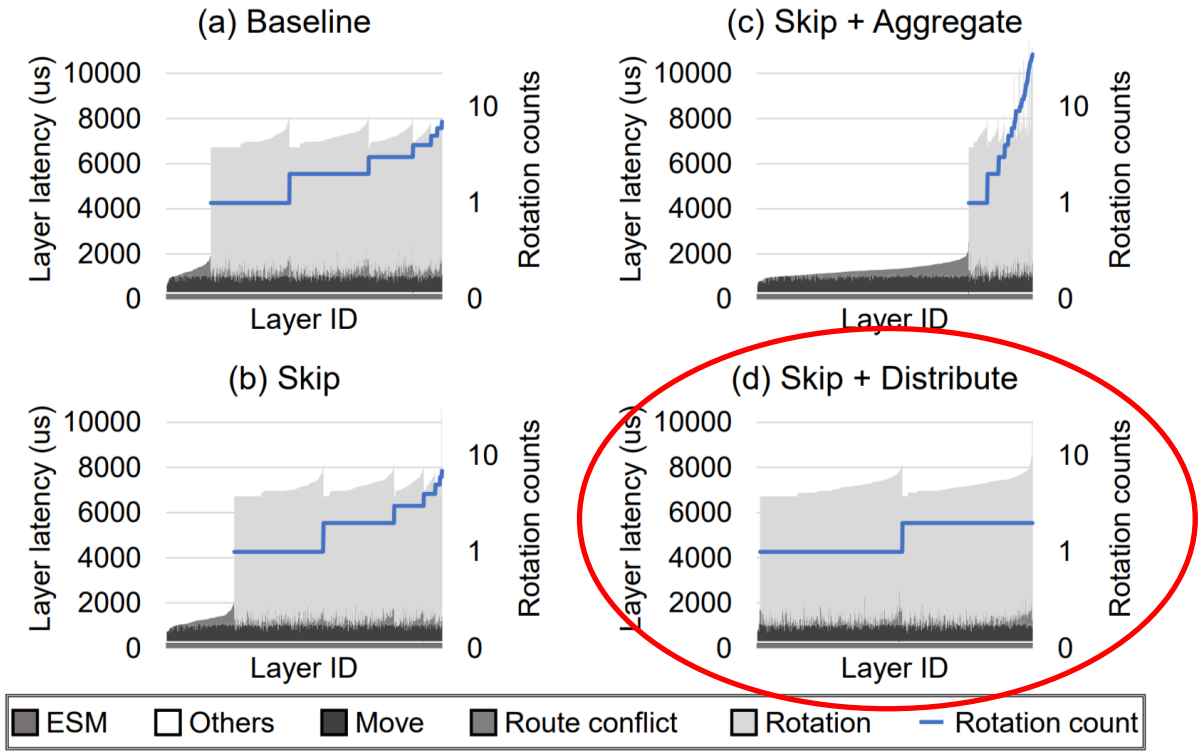




(a) 全パッチでdirect rotationをサポート
+ 同時に多くのdirect rotationを実行可
- Space overhead大
- Target以外のパッチのSLMと干渉

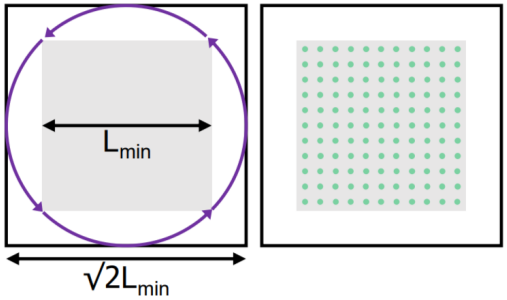
(b) 一部のパッチのみでサポート
+ Space overhead小
- 同時に実行可能なdirect rotationの数に制限

コンパイル最適化：Distributing rotations

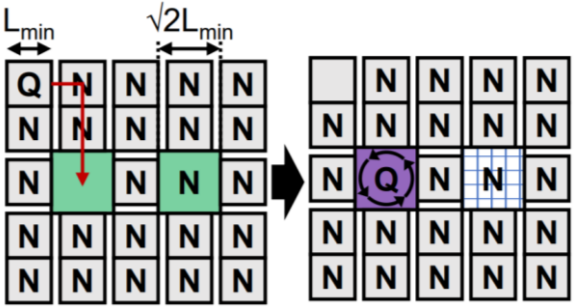


- コンパイル時の工夫により1サイクルあたりrotationを高々2つに抑える
  -> Direct rotationをサポートするパッチは2つで十分

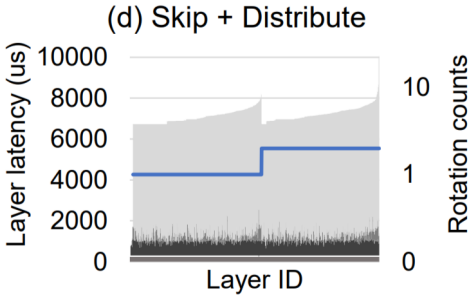
D3-ROT architecture



Direct rotation



Dedicated rotation patches

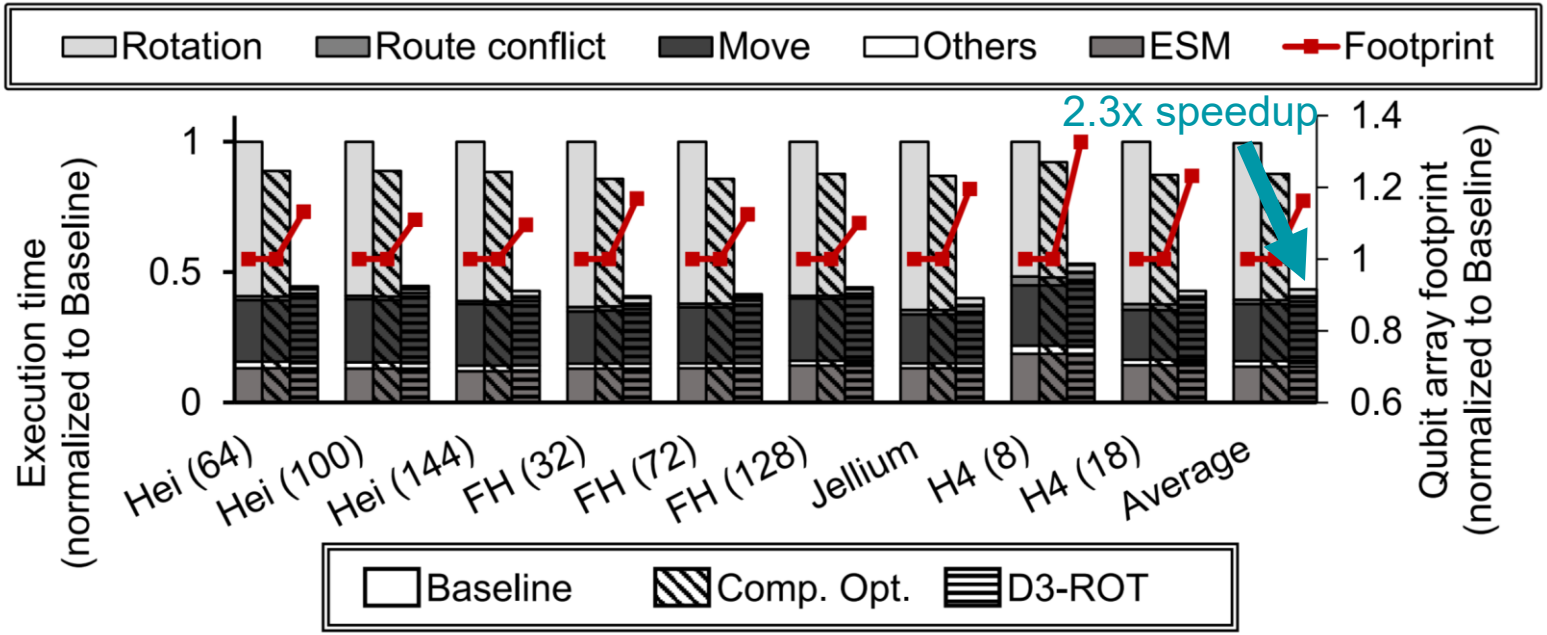


Distributing rotations

- D3-ROT: Near-rotation-free architecture with **(1) direct rotation** on the **(2) dedicated rotation patches** enabled by **(3) distributing rotations**
- 制御HW、Qubit plane、コンパイラのco-optimizationにより
  - 導入する専用AODの数および面積オーバーヘッドを最小化し
  - RotationによるbottleneckをDistributing rotationsで解消する

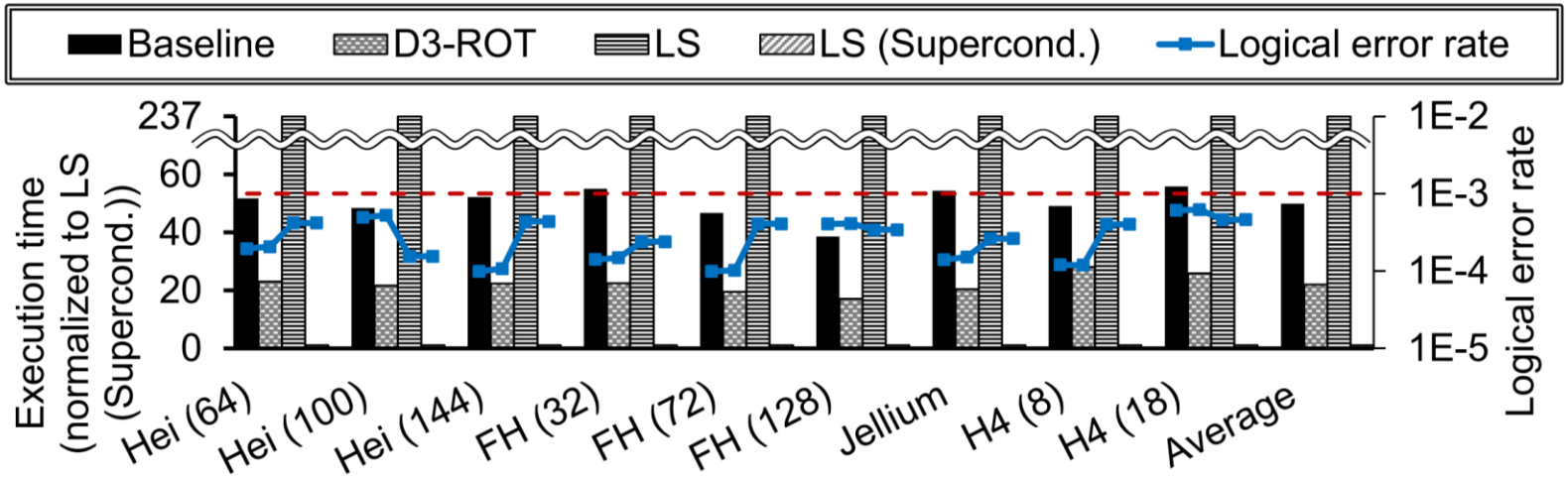
  アーキテクチャを提案

評価結果：中性原子アーキテクチャ同士の比較



- RotationのBottleneckを解消し、Baselineに比べて平均2.3倍の高速化
  - 面積オーバヘッドは16%程度

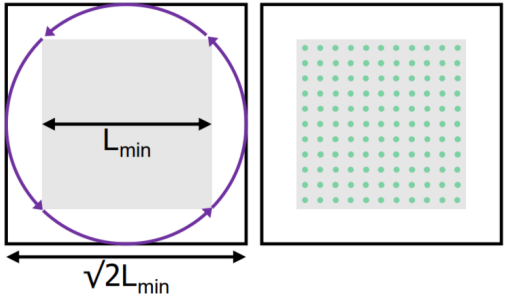
評価結果：超伝導＋格子手術との比較

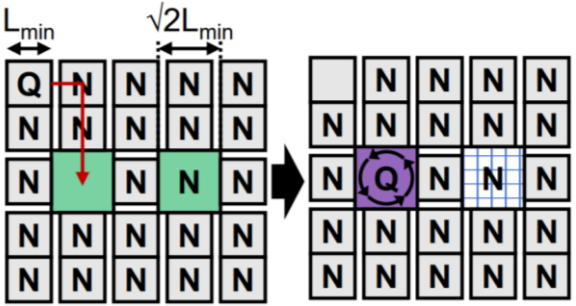


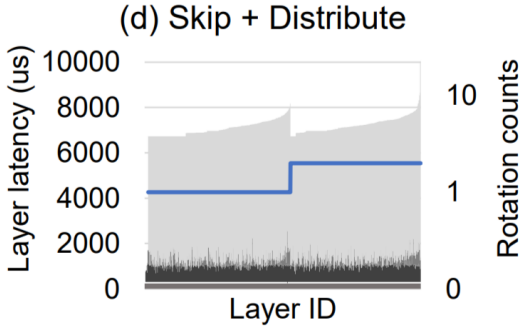
- 超伝導+格子手術アーキとの実行時間の差
  - Baseline: 50倍程度
  - D3-ROT: 22倍程度
- 必要な量子ビット数
  - 超伝導：521,352 qubits
  - D3-ROT：84,496 qubits

6.2倍改善 → Space-time overheadの差は **3.6倍**まで縮小

NAQsimまとめ







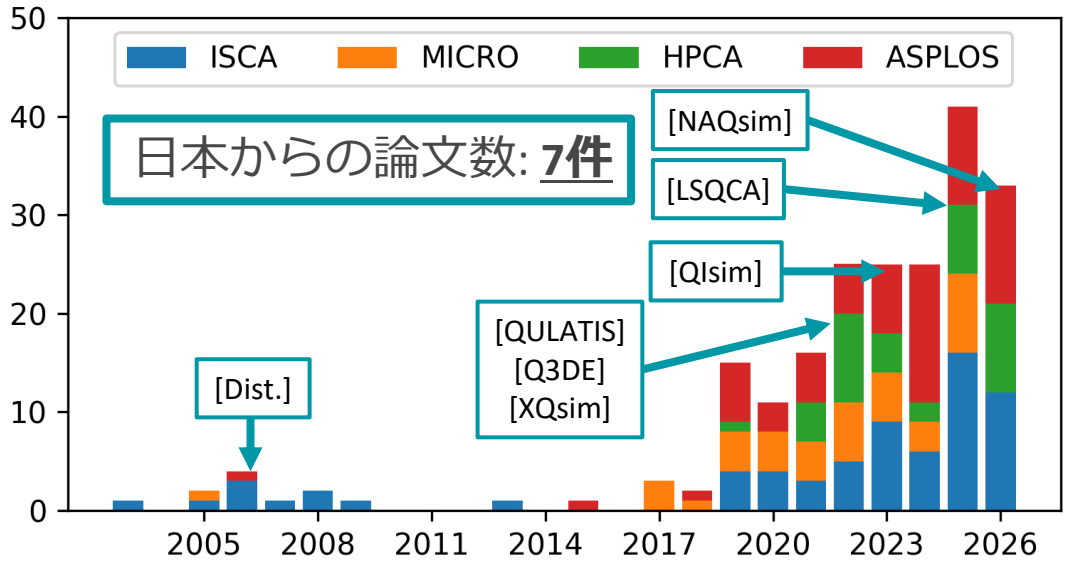
- フルスタックFTQCシミュレーションフレームワークを開発
  - コンパイラ、制御ハードウェア、Qubit planeを同時に考慮しつつアーキテクチャ探索可能
- Co-optimizationにより D3-ROT architectureを提案
  - Rotationのbottleneckを解消し、Baselineに比べて**2.3倍の高速化**
  - Space-time overheadの観点で、超伝導アーキテクチャとの差を3.6倍まで縮小
- 今後の展望：分散FTQC or そもそも回転不要なcolor codeのアーキテクチャ探索

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
- 量子計算機アーキテクチャに関連する研究紹介
  - 超伝導ディジタル回路を用いた量子誤り推定機構（博論、DAC'21、HPCA'22）
  - 誤り耐性量子計算（FTQC）におけるロードストア型アーキテクチャ（HPCA'25）

    未出版内容のため公開版では削除
  - 中性原子を対象としたフルスタックFTQCシミュレーションフレームワーク（MICRO'26）
- 量子計算機アーキテクチャ研究の魅力・まとめ

国内外の量子計算機アーキテクチャ研究動向

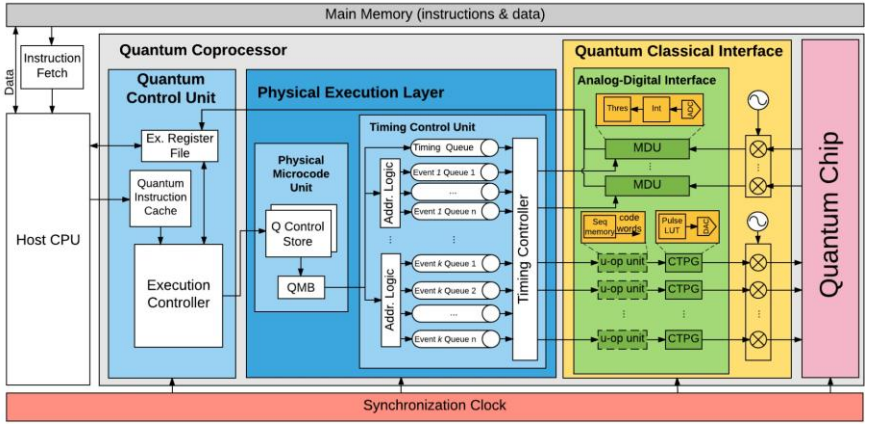


アーキ系のトップ国際会議（ISCA, MICRO, HPCA, ASPLOS）における量子計算機関連論文数（2003～2026年）

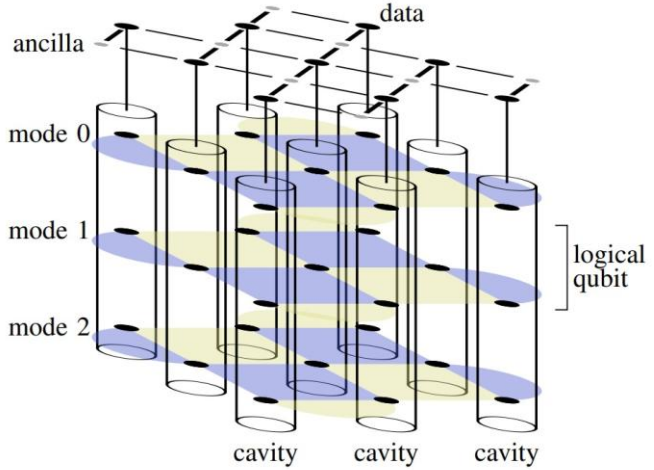
年	量子関連論文割合
2003~2018	0~1%程度
2019	5.4% (15/276)
2020	3.6% (11/308)
2021	5.0% (16/323)
2022	7.7% (25/325)
2023	6.1% (25/412)
2024	5.3% (25/475)
2025	7.6% (41/539)
2026	7.6% (33/434) MICRO除く

[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**.
[QIsim] D. Min, …, M. Tanaka, K. Inoue, J. Kim, “QIsim: 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.**
[NAQsim] **Y. Ueno**, S. Sunami, T. Hinokuma, Y. Suzuki, A. Goban, H. Yamasaki, T. Tanimoto, I. Byun “NAQsim: Full-Stack Architecture Simulation Framework for Fast and Space-Efficient Neutral Atom Quantum Computing”, **MICRO2026**.

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 (@UCSB until 2023))	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

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
  - 研究のアプローチ：**適切な**抽象化・モデル化
    - 量子計算機アーキテクチャ研究では物理の方々との共同研究が不可欠
- 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
  - まだ何も決まってないから色々と考える余地が多い、重要な問題も残っている
  - 需要の割には日本ではまだまだ人口が少ない -> チャンス！