Vol.2023-ARC-252 No.27
Vol.2023-SLDM-202 No.27
Vol.2023-EMB-62 No.27
2023/3/24

極低温不揮発FPGAを対象とした誤り耐性量子コンピュータ向け表面符号復号器のRTL設計

中村 徹舟[1,a)]　宮村 信[2]　井上 弘士[1]　川上 哲志[1]　阪本 利司[2]　多田 宗弘[2]　谷本 輝夫[1]

概要：量子ハードウェアは高いエラー率を示すため，量子誤り訂正技術の実現が不可欠である．特に，表面符号は高いエラー訂正性能をもつ誤り訂正符号として注目されている．本研究では，極低温環境で動作可能な NanoBridge-FPGA への実装を目指し，iterative greedy アルゴリズムを用いた表面符号復号器の RTL 設計を行った．設計した復号器は，先行研究と同じ誤りシミュレータを用いて動作検証を行い，レイテンシ・使用リソース量の評価も行った．さらに，NanoBridge-FPGA への論理合成・配置配線も行い，使用リソース量を確認した．

1. はじめに

新たなコンピューティング技術として，量子コンピュータが注目されている．量子コンピュータは，重ね合わせ状態や量子もつれといった量子力学的特性を利用することで，大規模な演算を高速で実現できるとされている．しかしながら，既存の量子デバイスが示すエラー率は，ほとんどの量子アプリケーションの実行に必要な値と比べて非常に高く，実用化に向けた障壁となっている．

量子ハードウェアに生じるエラーの影響を排し，高精度な演算を実現する量子誤り訂正 (Quantum Error Correction: QEC) 手法として，量子誤り訂正符号が提案されている．特に，物理量子ビットを 2 次元格子平面上に並べた表面符号 (Surface code: SC) は，高い信頼性をもつ量子誤り訂正符号として注目されている．表面符号のエラー訂正は，エラーシンドロームから最も可能性の高いエラー鎖の位置を推論することで行われる．この操作は，重み付きグラフの最小重み完全グラフマッチング (Minimum Weight Perfect Matching: MWPM) として扱うことができる．このような処理を行う論理ユニットは復号器 (Decoder) と呼ばれ，CMOS や SFQ といった古典的動作を行うハードウェアによって実現される．

本研究で対象とする超伝導量子回路は，極低温環境での動作が前提であり，復号器などの制御ユニットも極低温環境に置くことで配線コストを小さくする手法が検討されている．しかしながら，極低温環境においては，厳しい電力制約や面積制約が課せられる．例えば，4K の温度帯において許容される消費電力は約 1W 程度，面積は約 $620cm^2$ とされる [1]．したがって，低消費電力かつ小さい面積で実装可能な復号器を実現することが必要となる．

極低温環境で動作する復号器の実装デバイスとして注目されている NanoBridge-FPGA [2] は，回路を実装する際，論理合成には Synopsys 社の Design Compiler を使用しているため，実装する回路の RTL 記述が必要となる．しかしながら，これまでに提案された中で最も有効な復号アルゴリズムの 1 つである QECOOL [3] の iterative greedy アルゴリズムの基本的な復号器実装は，SFQ 回路を想定したもの [3] や，C++記述からの高位合成を利用したもの [4] であり，RTL で設計されたものはほとんど存在しない．

本研究では，十分な復号性能をもち，NanoBrige-FPGA への実装が可能な復号器の RTL 設計を行うことにより，将来的な極低温環境での動作検証を可能にすることを目的とする．ハードウェア記述言語である Verilog-HDL を用いて iterative greedy アルゴリズムのオンライン復号器の RTL 設計を行った．設計した回路は，先行研究 [4] と同じ C++による誤りシミュレータを用いた動作検証をパスすることを確認した．また，Verilog によるテストベンチを作成し，C++による検証環境に対応していないデバイスでの検証を可能にした．さらに，NanoBridge-FPGA をターゲットにした論理合成および配置配線を行い，使用リソース量を確認した．

本論文の構成は以下の通りである．まず，第 2 章で表面符号の基本事項を述べる．次に，第 3 章で NanoBridge-FPGA の説明を行う．第 4 章では，関連研究を述べる．第 5 章で

[1] 九州大学
Kyushu University, Nishi, Fukuoka 819–0395, Japan
[2] ナノブリッジ・セミコンダクター株式会社
NanoBridge Semiconductor, Inc., Tsukuba, Ibaraki 305-8564, Japan
[a)] tesshu.nakamura@cpc.ait.kyushu-u.ac.jp

Vol.2023-ARC-252 No.27
Vol.2023-SLDM-202 No.27
Vol.2023-EMB-62 No.27
2023/3/24

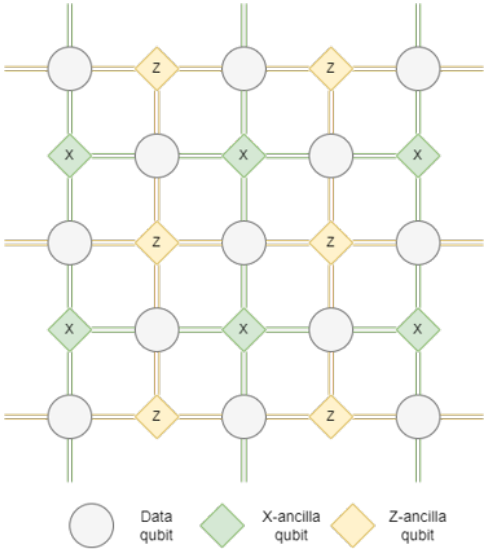

図 1: 表面符号（符号距離 3）

は，設計復号アルゴリズムである iterative greedy アルゴリズムについて述べる．第 6 章では，本研究で行った復号器の RTL 設計について述べ，第 7 章で動作検証，第 8 章で復号器の性能評価について述べる．最終章では，まとめと今後の展望について述べる．

2. 表面符号を用いた誤り訂正

2.1 表面符号の構成

表面符号は，複数の物理量子ビットを 2 次元格子状に配置し，1 つの論理量子ビットを構成する符号である．表面符号上の物理量子ビットは，論理量子ビットの状態を表現するデータ量子ビット (Data qubit) と，データ量子ビットに生じたエラーを検知するための観測用ビットである補助量子ビット (Ancilla qubit) から構成される．各補助量子ビットは，隣接するデータ量子ビットとエンタングル状態を形成しており，隣接する 4 つのデータ量子ビットのパリティとしてはたらく．これらの測定値はエラーシンドロームと呼ばれ，この情報をもとに訂正操作が行われる．これにより，データ量子ビットの状態を観測によって破壊することなく，補助量子ビットの観測結果からデータ量子ビットに生じたエラーの発生箇所を推定できる．

論理量子ビットに生じるエラー率（論理エラー率）は，物理量子ビットに生じるエラー率（物理エラー率）と符号距離に依存する．この符号距離は，論理パウリ X 演算・論理パウリ Z 演算を定義するために必要な物理量子ビットのビット反転または位相反転の最小数として定義される．

図 1 に符号距離 d=3 の場合の模式図を示す．補助量子ビットには X と Z の 2 種類が存在し，それぞれパウリ X エラー（ビット反転エラー），パウリ Z エラー（位相反転エラー）の検出に用いられる．また，パウリ Y エラーは，これら 2 つのエラーが同時発生した場合に対応する．実際の量子ビットには連続的な量のエラーが起こりうるが，量子状態の測定時に起こる射影から，パウリ X，Y，Z エラーに離散化することができるため，上記のような簡略化が可能である．

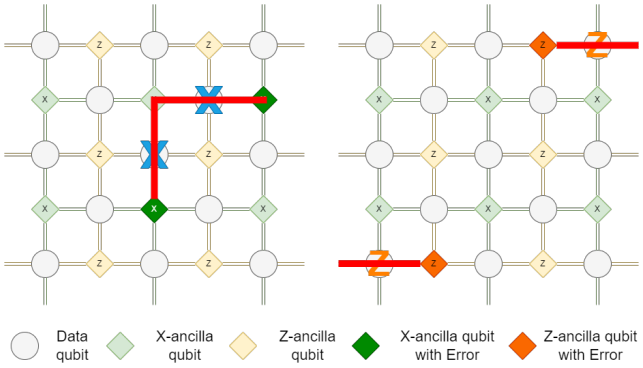

図 2: エラー位置の推定

2.2 表面符号の復号

X 補助量子ビットおよび Z 補助量子ビットは，隣接するデータ量子ビットのうち奇数個にエラーが生じた場合に $|1\rangle$ となり，“1”が観測されることが期待される．“1”が観測された補助量子ビットをアクティブビットと呼ぶ．量子回路実行中，各補助量子ビットは一定間隔ごとに観測され，それらの観測値がエラーシンドロームを形成する．この観測間隔はコードサイクルと呼ばれ，量子ビットの寿命より十分に短い必要がある．超伝導量子ビットの場合，コードサイクルは一般的に約 1µs と仮定される [4, 5].

表面符号の復号とは，エラーシンドロームの情報をもとに，量子回路実行中に生じたエラーの位置および種類を推定し，それらを訂正するプロセスである．表面符号の復号器は，ノイズモデルとエラーシンドロームの情報をもとに，得られたシンドロームを実現するエラーのうち最も可能性の高いものを推定することが望ましいが，このような推定を求める問題は NP 困難であることが知られている [6]．そのため，近似的なアルゴリズムを用いて問題の複雑さを多項式時間に還元することが一般的である．最も有望なアプローチの 1 つは，各アクティブビットを頂点とし，アクティブビット間のマンハッタン距離を重みとしたグラフの最小重み完全マッチング問題 [7] に帰着する方法である．図 2 に示すように, 復号器はアクティブビット間の経路のコストが最小となるアクティブビットの組を求めることで，それらを繋ぐ最短経路上にエラーが生じていると推定する．エラー訂正操作においては，実際にエラーが生じている経路とは別の経路を選んだ場合でも，選んだ経路とエラーが生じている経路で閉じたループが形成されていれば，符号語の状態に影響は無く，復号成功となることが知られている．X 補助量子ビットと Z 補助量子ビットは交互に配置されているため，それぞれのエラーシンドロームに対して独立に復号を行うことが可能である．

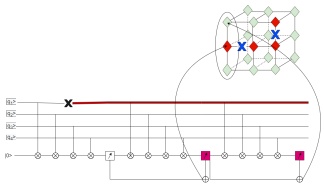

図 3: データ量子ビットにエラーが生じた場合

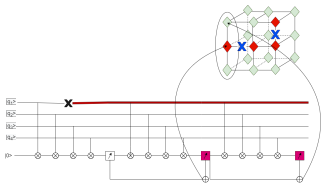

図 4: 補助量子ビットの観測エラーが生じた場合

量子ビットに生じるエラーには，データ量子ビットに生じるエラーだけでなく，補助量子ビットの観測エラーも含まれる．正確な誤り訂正の実現のためには，これらのエラーの訂正も考える必要がある．観測エラーの訂正は，補助量子ビットの観測を複数回行うことで対応可能である．図 3，図 4 に示すように，各時刻の観測結果とその前時刻の観測結果の XOR 値を垂直方向に積み重ねることにより，3 次元のシンドローム格子を構成できる．この 3 次元のシンドローム格子上のマッチング問題を解くことで，観測エラーも含めたエラー訂正が可能となる．

3. NanoBridge-FPGA

NanoBrige は半導体スイッチと半導体メモリを同時に置き換え可能な不揮発性スイッチである．ポリマー固体電解質を，不活性電極であるルテニウム (Ru) と活性電極である銅 (Cu) で挟んだ構造をとり，銅電極側に正の電圧を印加した場合には，イオン化した銅により架橋が形成され，高抵抗（オフ）状態から低抵抗（オン）状態へと変化する．また，負の電圧を印加した場合には，架橋を形成する銅原子が回収され，オン状態からオフ状態へと変化する．NanoBridege は以下のような特徴をもつ．

- 繰り返し書き換えが可能であり，状態を維持するための電力が不要である（不揮発性）
- 半導体スイッチに比べて負荷容量が小さい（半導体スイッチの 10 分の 1 程度）
- オン・オフ抵抗差が高い（オン状態：500 Ω，オフ状態：1 GΩ）
- 銅架橋の太さは数 nm と推定される．

NanoBridge-FPGA は，LUT のメモリや配線を切り替えるスイッチに用いられている半導体メモリや半導体スイッチを NanoBridge に置き換えた FPGA である．論理ブロックの面積が小さく，低消費電力かつ高速動作を実現可能であることや，宇宙空間・極低温などの過酷環境下での動作が可能であるといった特徴をもつ．このため，量子計算をサポートする古典ユニットである量子制御ユニット (Quantum Control Processor: QCP) の実現デバイスとしての利用が議論されている．現在, NanoBridge-FPGA は 4K での抵抗性スイッチングが確認されており, QCP 実現デバイスとしての利用に向けた開発が行われている [8].

4. 関連研究

表面符号の復号を高速・低消費電力かつ正確に実行するために，これまで多くの復号アルゴリズムが提案されている．最も広く用いられている Blossom アルゴリズム [9] は，MWPM の厳密解を提供する高い復号性能を示すが，符号距離について多項式にスケールすること [7] やシンドロームが全て得られてからしか復号処理を開始できないため，大規模化に不向きであるとされる．これに対し，素集合データ構造に対する union-find アルゴリズムを用いる手法 [10–12]，ノイズモデルをニューラルネットワークで学習する手法 [13]，LUT を用いてマッチングコストの計算コストを削減する手法 [10, 14] などが提案されている．特に，MWPM の近似アルゴリズムである Greedy アルゴリズム [15] は，スケーラビリティと小さなレイテンシを示すため，多くの復号器提案 [3–5, 14] において用いられている．本研究では，Greedy アルゴリズムのバリエーションの 1 つである Iterative greedy アルゴリズム [3, 4, 14] を設計対象としている．Greedy アルゴリズムおよび Iterative greedy アルゴリズムの詳細は, 第 5 章にて述べる．

また，復号器を実装するデバイスに関する提案も行われている．300K CMOS は，現行の CMOS 技術を用いた，室温動作する最も標準的なデバイスである．現在の量子制御プロセッサは，この 300K CMOS を用いて実現されているが，4K で動作する QC インターフェースとの 300K-4K 間データ転送は，スケーラビリティが制限される要因となっている [16, 17]．このため，QC インターフェースと同じ約 4K の極低温環境で制御プロセッサを動作させる提案がなされている．4K の極低温環境で動作する CMOS デバイスは Cryogenic CMOS (Cryo-CMOS) と呼ばれ，復号器の実装デバイスとして検討されている [16, 18, 19]．また，ジュセフソン接合を基本素子とし，同じく 4K で動作する単一磁束量子 (SFQ) 回路を用いた手法 [4, 20] なども提案されている．

5. 設計対象復号アルゴリズム

本研究においては，MWPM の近似アルゴリズムである Iterative greedy アルゴリズムを用いた復号器の RTL 設計を行った．本章では，はじめに基本的な Greedy アルゴリズムを説明したのち，その修正アルゴリズムである Iterative greedy アルゴリズムを説明する．

5.1 Greedy アルゴリズム

Blossom アルゴリズムを用いた MWPM 復号器は, MWPM の厳密解を与えるが，量子ハードウェアから要請される時間制約以内に処理を終えることが難しいため，近似解を求める代替アプローチが必要である．このような代替アルゴリズムの 1 つである Greedy アルゴリズム [15] は，スケーラビリティと小さなレイテンシを示すため，これまで様々な提案 [3, 5, 14] において用いられている．Greedy アルゴリズムの疑似コードをアルゴリズム 1 に示す．

入力 S は，復号対象となるすべての時間軸にわたるエラーシンドロームを格納した配列である．各シンドロームは古い物から順（z 成分の昇順）に格納されており，エントリサイズは符号距離を d，コードサイクル数を c として $d * d * c$ となる．また，出力 P は，マッチングが決定したペアを格納した配列である．この結果をもとに実際の訂正操作が行われる．

動作の概要は以下の通りである．

(1) 1 番目のノードを選ぶ.

(2) 手順 1 で選んだノードと，全ノードとの間のコストを求める．このとき，選んだノード自身とマッチングした場合のコストも求める．

(3) 最もコストが小さくなるノードの組を選ぶ．（コストが同じ場合は，自身以外でインデックスが最も小さいノードを優先する）

(4) 選んだノードの組（自分自身とのマッチングを含む）をマッチング結果として送出する．

(5) 2 番目以降のノードについても手順 2〜4 を順に繰り返す．

第 4 章に述べたように，コストの計算手法は復号の定式化手法によって異なる．本研究で設計した復号器では，重み付けを考慮しないシンプルな実装としている．コスト計算手法の詳細は 6.2.2 に述べる．

アルゴリズム 1 Greedy algorithm

```
Input: S[0...N-1]
Output: P[0...N-1]
function MATCH_GREEDY(S)
    Initialize:
        A[0...N-1] ← 0
    for i = 0 to N-1 do
        if A[i] == 0 then
            c_m ← GET_COST(i, i)
            t_m ← i
            for t = i+1 to N-1 do
                if A[t] == 0 then
                    c ← GET_COST(i, t)
                    if c < c_m or (c == c_m and t_m == i) then
                        c_m ← c
                        t_m ← t
            end for
            if t_m == i then
                A[i] ← 1
                P.end ← MAKE_PAIR(i, -1)
            else
                A[i] ← 1
                A[t_m] ← 1
                P.end ← MAKE_PAIR(i, t_m)
    end for
    return P
end function
```

5.2 Iterative greedy アルゴリズム

Greedy アルゴリズムには様々なバリエーションがあるが，その中でも，許容コストに関して繰り返し Greedy アルゴリズムを用いたマッチングを行う Iterative greedy と呼ばれるアルゴリズムが有望視されている [3, 14]．Iterative greedy アルゴリズムの疑似コードをアルゴリズム 2 に示す．

Iterative greedy アルゴリズムの動作を以下に示す．

(1) 許容コストを 1 とする．

(2) 第 5.1 節と同様の手順でマッチングを行う．ただし，各サイクルにおいて最小コストが許容コストより大きければ，マッチング不成立とする．

(3) 許容コストを 2 とし，手順 2 を実行した後の出力ノード配列に対し，同様に 5.1 の手順でマッチングを行う．

(4) 許容コストを 3 以降とした場合についても，同様に上記の処理を繰り返す．

6. 表面符号復号器の RTL 設計

本研究では，3 次元のシンドローム格子を対象としたオンライン復号器の RTL 設計を行った．復号アルゴリズムには，第 5 章で述べた iterative greedy アルゴリズムを用いた．本章では，設計環境，設計した復号器の概要，およびテストベンチ記述を用いた動作検証の結果を述べる．

6.1 設計環境

設計においては，Vitis HLS，Vivado，Design Compiler を用いた．以下に，これらのツールの概要と本研究における役割を述べる．

6.1.1 Vitis HLS

Vitis HLS は，Xilinx 社が提供する高位合成ツールである．C/C++などによる抽象度の高い記述（動作記述）を元に，ハードウェア記述言語 (Hardware Description Language: HDL) による RTL 記述を自動生成することが可能である（高位合成）．さらに，C/C++で記述されたテストベンチによる回路の動作検証，RTL 記述の論理合成，対応している FPGA への配置配線など，FPGA 開発の各種フローを一括して行える．さらに，RTL Blackbox の機能を用いること

Vol.2023-ARC-252 No.27
Vol.2023-SLDM-202 No.27
Vol.2023-EMB-62 No.27
2023/3/24

アルゴリズム 2 iterative greedy algorithm

Input: $S[0 \ldots N-1]$
Output: $P[0 \ldots N-1]$

```
1:  function MATCH_ITERATIVE_GREEDY(S)
2:      Initialize:
3:          A[0 ... N − 1] ← 0
4:      for a = 1 to S[N − 1].z do
5:          for i = 0 to N − 1 do
6:              if A[i] == 0 then
7:                  c_m ← GET_COST(i, i)
8:                  t_m ← i
9:                  for t = i + 1 to N − 1 do
10:                     if A[t] == 0 then
11:                         c ← GET_COST(i, t)
12:                         if c < c_m or (c == c_m and t_m == i) then
13:                             c_m ← c
14:                             t_m ← t
15:                 end for
16:                 if c_m <= a then
17:                     if t_m == i then
18:                         A[i] ← 1
19:                         P.end ← MAKE_PAIR(i, −1)
20:                     else
21:                         A[i] ← 1
22:                         A[t_m] ← 1
23:                         P.end ← MAKE_PAIR(i, t_m)
24:         end for
25:     end for
26:     return P
27: end function
```

で，自作の RTL 記述を高位合成プロジェクトに組み込むことも可能である．

本研究では，先行研究と同じ誤りシミュレータ [4] を使用した動作検証を行う際に Vitis HLS を用いた．この誤りシミュレータは C++で記述されている．また，FPGA をターゲットにした論理合成・配置配線を行い，復号器の実行周波数・使用リソースなどの確認も行った．動作検証に用いた誤りシミュレータの詳細は第 7.1 節において述べる．本研究で使用したバージョンは Vitis HLS 2021.2 であり，ターゲットデバイスは XCZU7EV-2FFVC1156I とした．

6.1.2 Vivado

Vivado は，Xilinx 社が提供する HDL デザイン合成・解析ツールである．HDL により記述されたテストベンチ（RTL テストベンチ）を用いて，RTL 記述の動作検証，論理合成，配置配線などを行える．また，波形ビューワーの機能をもち，設計した回路のデバッグを視覚的に行うことが可能である．

本研究では，モジュールごとの単体テストを行う際に使用した．また，本研究で新たに作成した RTL テストベンチは，Vivado 上で動作検証を行えることを確認した．使用したバージョンは Vivado 2021.2 である．

6.1.3 Design Compiler

Design Compiler は，Synopsys 社が開発した論理合成ツールである．Verilog などによる RTL 記述からの論理合成が可能であり，AISC 開発におけるデファクトスタンダードとなっている．本研究では，設計した復号器が ASIC などをターゲットにした場合にも合成可能であることの確認に用いた．使用したバージョンは Design Compiler H-2013.03-SP2 である．

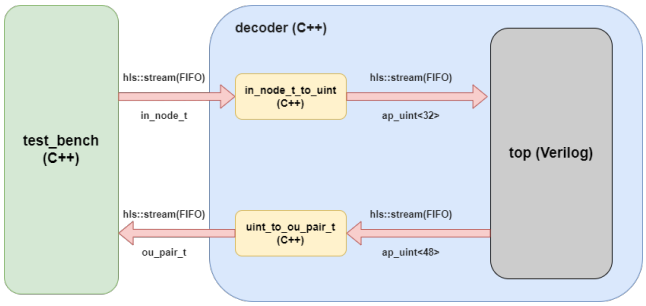

図 5: Vitis HLS プロジェクト構成

6.2 復号器の詳細

6.2.1 Vitis HLS プロジェクトの構成

Vitis HLS におけるプロジェクトの構成を図 5 に示す．本プロジェクトに含まれるファイルには，復号器本体と入出力インタフェースが記述された decoder.cpp および top.v と，鈴木らの Q3DE と同じ誤りシミュレータ [4] とその関連ファイル群がある．decoder.cpp には，最上位関数である decoder が定義されており，また，その下位関数として in_node_t_to_uint，uint_to_ou_pair_t，top が定義されている．これらの下位関数は HLS stream オブジェクトで接続されている．このオブジェクトは，入出力データが順次的に消費または生成される場合に用いられ，FIFO の BRAM としてインプリメントされる．また，誤りシミュレータと接続された decoder の入出力ストリームも，同様に HLS stream オブジェクトを用いて実装されている．

下位関数 top は復号器本体の記述である．本研究では，RTL Blackbox の機能を用いて，decoder.cpp の top 記述を本研究で作成した RTL 記述と置換する．この RTL 記述は top.v に定義されており，トップモジュールとその下位モジュールがすべて定義されている．

また，in_node_t_to_uint，uint_to_ou_pair_t は，誤りシミュレータからの入出力ストリームを，RTL 記述の入出力プロトコルに適合する形式へ変換する関数である．これは，RTL Blackbox を使用する場合には入出力ストリームに構造体を指定することができないという Vitis HLS の制約に対応するための処理である．in_node_t_to_uint は，構造体 in_node_t を 32 ビット任意精度符号なし整数型へ変換する関数であり，uint_to_ou_pair_t は，48 ビット任意精度符号なし整数型から構造体 ou_pair_t へ変換する関数である．

6.2.2 各モジュールの詳細

設計した復号器のブロック図を図 6 に示す．本復号器の

Vol.2023-ARC-252 No.27
Vol.2023-SLDM-202 No.27
Vol.2023-EMB-62 No.27
2023/3/24

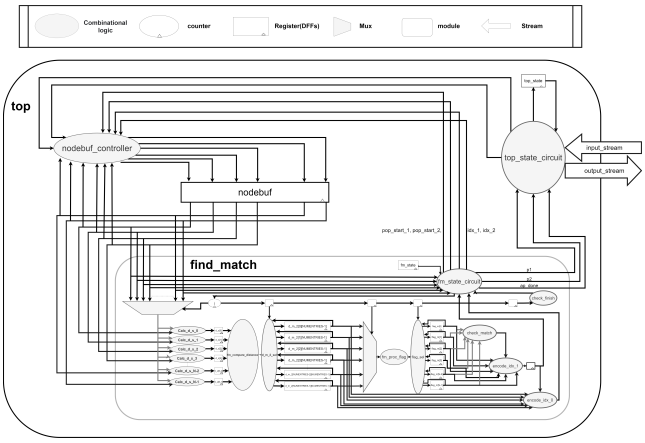

図 6: 設計した復号器の概略図

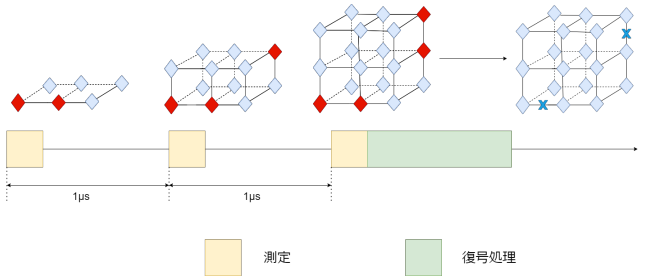

図 7: バッチ処理の概略図

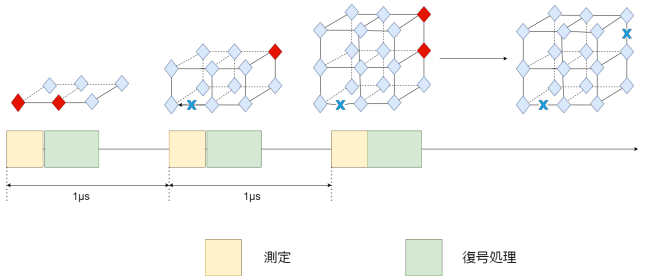

図 8: オンライン処理の概略図

トップモジュールは，nodebuf，find_match という 2 つのモジュールと，それらの制御記述からなる．

復号アルゴリズムには，第 5.2 節に述べた iterative greedy アルゴリズムを用いている．本復号器では，観測結果がすべて得られるまで待ってから処理を開始する手法（バッチ処理）ではなく，マッチングを確定するのに十分な観測結果が得られた順にマッチングを行う手法（オンライン処理）を採用している．バッチ処理とオンライン処理の模式図は，図 7，図 8 に示す．また，nodebuf と find_match の概要を以下に示す．

- nodebuf
  入力ストリームから受け取られたアクティブノードは，Active Node Queue (ANQ) と呼ばれる領域に記録される．アクティブノードは構造体として定義されており，メンバー変数として 3 次元空間座標 (x, y, z) と，最も近傍の境界までの距離 d をエントリにもつ．find_match は，ANQ 内に保存されているアクティブノード間のコストを求め，マッチングを決定する．Verilog 記述では，ANQ のエントリサイズをパラメータ化しており，任意のサイズで合成することが可能である．ただし，ANQ のエントリサイズが十分でない場合，バッファオーバーフローの発生確率が論理エラー率よりも高くなる．この場合，エラー復号に失敗し，量子プログラムの実行を初めから再実行する必要がある．また，ANQ のエントリサイズの増加とともに，コスト評価回路や比較演算回路が増加する．

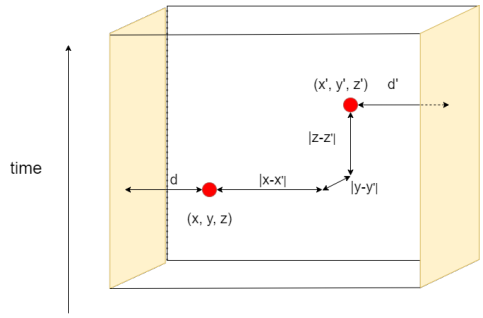

図 9: 3 次元のシンドローム格子上のアクティブノード

- find_match
  ANQ 内のエントリに対し，全対のコストをパイプラインで並列計算し，最小のコストとなるペアを選んで出力する．また，選んだペア間のコストが，その時点の閾値条件を満たすかを確認する．閾値条件とは「ANQ 内の最も新しいノードとパンプライン処理で選ばれたノードについて，それぞれの z 成分のみ考えた z 方向のコスト差（閾値）が，選んだペア間のコストより大きい」という条件である．閾値条件を満たさない場合にはマッチングは行われない．マッチング決定部の回路規模は ANQ のエントリサイズに比例し，レイテンシはその時点の ANQ 内のエントリ数に比例する．

また，コストの計算方法は以下の通りである．

- 異なる 2 ノード間のコスト
  図 9 に示すように，
  ノード A: 座標 (x, y, z)，境界までの距離 d
  ノード B: 座標 (x', y', z')，境界までの距離 d'
  とすると

$$c_1 = |x - x'| + |y - y'| + |z - z'|$$

$$c_2 = d + d'$$

$$cost = min(c_1, c_2)$$

- 1 つのノードが自分自身とマッチングする際のコスト

$$cost = 2 * d$$

Vol.2023-ARC-252 No.27
Vol.2023-SLDM-202 No.27
Vol.2023-EMB-62 No.27
2023/3/24

7. 動作検証

7.1 C++テストベンチによる動作検証

本研究において設計した復号器は，C++で記述された誤りシミュレータ（C++テストベンチ）を用いて動作検証を行った．この誤りシミュレータは，以下のような仮定を用いている．

(1) 誤り訂正符号として表面符号を用いる．符号距離は d とする．

(2) コードサイクルごとに，パウリエラーがデータ量子ビットおよび補助量子ビット上に発生するノイズモデルを仮定する．また，物理エラー率は p とする．

(3) ノイズマップは各コードサイクルの始めに挿入される．

(4) 各ノイズマップにおいて，パウリ X エラー，パウリ Y エラー，パウリ Z エラーが発生する確率はそれぞれ $p/2$ である．

(5) エラー推定には，Edmonds の blossom アルゴリズム [9] の Kolmogorov による実装 [21] を用いる．

以上は，誤り訂正性能を評価する場合の一般的な仮定である [3,5,22].

パラメータは鈴木らによる 40-BASE，80-BASE [4] と同様のもの ($p = 10^{-2}$，$d = 5$) を用いた．いずれのエントリーサイズの場合も，任意のシード値・試行回数について，正しい結果が得られることを確認した．

7.2 RTL テストベンチによる動作検証

7.1 で用いたような C++テストベンチは，Vitis HLS が対応している FPGA をターゲットとした場合のみ動作検証が可能である．しかしながら，実際にハードウェアへの実装を考えた場合，ASIC や Xilinx 環境に対応していない FPGA なども選択肢として想定される．したがって，ターゲットに依存しない検証を可能にすることは重要である．

そこで本研究では，iterative greedy アルゴリズムを用いた復号器のテストベンチを Verilog を用いて記述し，動作検証が可能な環境を作成した．図 10 に作成したテスト環境の構成図を示す．入力には，第 7.1 節の誤りシミュレータからサンプルした複数のエラーパターンを与えた．結果として，いずれの入力に対しても，復号器からの出力が正解データと一致することを確認した．

8. 設計した復号器の評価

8.1 評価手法

8.1.1 Xilinx FPGA を対象とした評価

Vitis HLS では，第 7.1 節で述べた検証を行う際に，入力

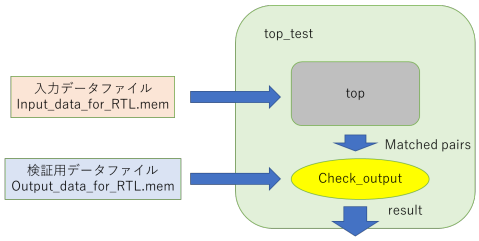

図 10: RTL テストベンチを用いた動作検証

	C synthesis	Run implementation
40-BASE	2.0 ns	2.0 ns
80-BASE	3.0 ns	3.0 ns

表 1: Vitis HLS を用いた合成時のクロック制約

テストデータの処理にかかったレイテンシ（クロックサイクル数）が出力される．複数パターンの入力テストデータを与えた場合，最良パターンと最悪パターン，および平均値をそれぞれ得ることができる．また，RTL 記述の論理合成および指定した FPGA ターゲットへの配置配線を行うことで，クロック周期やリソース使用量を確認することができる．

本研究の性能評価においては，Q3DE の高位合成による復号器実装 [4] が最小のスループットを示したクロック制約（表 1）を与えたうえで論理合成と配置配線を行い，クロック周期とリソース使用量を確認した．また，その時のレイテンシ（処理時間）を求め，Q3DE の復号器実装との比較を行った．評価に用いた Vitis HLS のバージョンは Vitis HLS 2021.2 であり，ターゲット FPGA には XCZU7EV-2FFVC1156I を用いた．また，クロック制約を除くオプションについては全てデフォルトの値に指定した．

8.1.2 ASIC を対象とした評価

設計した復号器が Design Compiler を用いて論理合成可能であるかを検証するため，実際に ASIC をターゲットとした論理合成を行った．また，その時の消費電力と面積を確認した．セルライブラリには HIT018 を使用し，合成時の各種最適化指定は行わなかった．

8.1.3 NanoBridge-FPGA を対象とした評価

将来の極低温環境での実測を目指し，本研究では NanoBridge-FPGA をターゲットとした論理合成および配置配線を行い，使用リソース量を確認した．論理合成には Design Compiler を用いており，現時点では面積方向の最適化を重視した合成を行っている．また，配置配線には，一般的な SA(Simulated Annealing) を用いた配置と，遅延を考慮した経路選択による配線を行った．

8.2 評価結果

8.2.1 Xilinx FPGA を対象とした評価

表 2 に，C++の動作記述から合成された復号器と本研究

	Avg Latency	Max Latency	Min Latency
40-BASE(HLS)	5,619	6,100	5,022
40-BASE(RTL)	3,706	4,165	3,079
80-BASE(HLS)	5,083	5,932	4,021
80-BASE(RTL)	3,706	4,165	3,079

表 2: 復号器のレイテンシ (クロックサイクル数)

	clock period
40-BASE (HLS)	2.027 ns
40-BASE (RTL)	2.967 ns
80-BASE (HLS)	2.979 ns
80-BASE (RTL)	3.836 ns

表 3: 配置配線後のクロック周期

	Avg Latency	Max Latency	Min Latency
40-BASE (HLS)	11,390 ns	12,365 ns	1,0179 ns
40-BASE (RTL)	10,996 ns	12,358 ns	9,135 ns
80-BASE (HLS)	15,142 ns	17,671 ns	11,979 ns
80-BASE (RTL)	14,216 ns	15,977 ns	11,811 ns

表 4: 復号器のレイテンシ (処理時間)

で RTL 設計した復号器のレイテンシ（平均値，最大値，最小値）を示す．単位はクロックサイクル数である．本研究で設計した復号器では，各マッチングサイクルにおける ANQ のエントリ数に応じた制御を行うことにより，パイプライン実行時の不要なコスト計算を避け，より小さいレイテンシを目指した．これにより，いずれのエントリサイズの場合も HLS より RTL の方が少ないクロックサイクル数で処理を完了していることが確認できた．

表 3 に配置配線後のクロック周期を示す．また，表 4 にテストデータ対する処理時間を示す．クロック周期については HLS と比較して増加していることがわかるが，処理に要するクロックサイクル数が減少しているため，テストデータに対する全体の処理時間は減少した．また，誤りシミュレータの乱数シードを変更し，複数パターンのテストデータについても同様に処理時間を求めた．これらの処理時間についても，本研究の復号器と同程度であることが確認できた．

表 5 に本研究で RTL 設計した回路と高位合成による回路の使用リソースを示す．いずれのエントリサイズの場合も HLS と比較して RTL の使用リソースの方が大きいことがわかる．これは C++記述から Verilog 記述を合成する際に，Vitis HLS の処理系によってターゲット FPGA の最適化が行われた結果だと考えられる．しかしながら，いずれの場合もリソース使用率は 10%未満に留まっており，組み込みレベルの FPGA 上で動作させることが可能である．

8.2.2 ASIC を対象とした評価

本研究で作成した Verilog 記述からの論理合成が可能であることを確認できた．論理合成後の面積と消費電力見積もりを表 6 に示す．ただし，配線面積を含んでいないため，配線を考慮した場合，これらの値は増加することが予想される．

	FF	LUT
40-BASE (HLS)	8,991	14,679
40-BASE (RTL)	9,111	14,191
80-BASE (HLS)	13,211	36,668
80-BASE (RTL)	21,065	35,811

表 5: Xillinx FPGA 使用リソース

	消費電力	面積
40-BASE (RTL)	130 mW	6.83 mm^2
80-BASE (RTL)	225 mW	14.17 mm^2

表 6: Design Compiler 論理合成結果

	FF	LUT
8-BASE (RTL)	2,148	5,180
16-BASE (RTL)	4,522	12,472

表 7: NanoBridge FPGA 使用リソース

4K の領域における消費電力バジェットは $1mW$ 程度 [1] であり，面積バジェットは $620cm^2$ 程度である [23] が，これは QCP の他ユニットを含めた合計値であり，各ユニットを実装する温度帯などの QCP アーキテクチャに依存する．また，極低温における MOSFET の性能は室温における性能と大きく異なるため，既存のモデルが極低温動作に適用できる保証がない点に注意が必要である．このようなアーキテクチャや温度を考慮した検討には，XQsim [24] といったシミュレーションツールを用いる必要がある．

8.2.3 NanoBridge-FPGA を対象とした評価

配置配線後のリソース使用量を表 7 に示す．LUT 使用率は 8-BASE と 16-BASE それぞれ 15%，37%程度であり，これらのエントリサイズでの実装が可能であることがわかる．極低温環境における実装時のクロック周波数や消費電力に関しては，静的タイミング解析 (STA) ツールによって見積もり可能である．現在，常温および 4K 環境における STA ライブラリを制作中であり，完了後，4K 環境における動作可能性の検証を行う予定である．

9. おわりに

本稿では，ハードウェア記述言語である Verilog を用いて，iterative greedy アルゴリズムによる復号器の RTL 設計を行った．また，先行研究にて提案された誤りシミュレータを用いて，設計した復号器の動作検証および処理レイテンシの評価を行い，十分な復号性能をもつことを示した．さらに，NanoBridge-FPGA 実装後の動作検証を行うため，設計した復号器の RTL テストベンチも作成した．実際にエントリサイズ 8 と 16 の場合について論理合成および配

置配線を行い，NanoBridge-FPGA 実装時の使用リソースを確認した．

今後は，STA ツールによる周波数・電力見積もりを行い，将来的なチップでの実測を目指す．

謝辞 本研究における誤りシミュレータを作成された鈴木泰成氏に感謝いたします．また，本研究の実施にあたり，ご助言いただいた上野洋典氏に感謝いたします．なお，本研究は一部，JST ムーンショット型研究開発事業 JPMJMS2067，JPMJPR21B3，JSPS 科研費 JP20K19771，JP22H05000，JP22K17868 の支援を受けたものである．

参考文献

[1] Hornibrook, J. M., Colless, J. I., Conway Lamb, I. D., Pauka, S. J., Lu, H., Gossard, A. C., Watson, J. D., Gardner, G. C., Fallahi, S., Manfra, M. J. and Reilly, D. J.: Cryogenic Control Architecture for Large-Scale Quantum Computing, *Phys. Rev. Appl.*, Vol. 3, p. 024010 (online), DOI: 10.1103/PhysRevApplied.3.024010 (2015).

[2] Miyamura, M., Sakamoto, T., Bai, X., Tsuji, Y., Morioka, A., Nebashi, R., Tada, M., Banno, N., Okamoto, K., Iguchi, N., Hada, H., Sugibayashi, T., Nagamatsu, Y., Ookubo, S., Shirai, T., Sugai, F. and Inaba, M.: NanoBridge-Based FPGA in High-Temperature Environments, *IEEE Micro*, Vol. 37, No. 5, pp. 32–42 (online), DOI: 10.1109/MM.2017.3711648 (2017).

[3] Ueno, Y., Kondo, M., Tanaka, M., Suzuki, Y. and Tabuchi, Y.: QECOOL: On-Line Quantum Error Correction with a Superconducting Decoder for Surface Code, *Proceedings of the 58th ACM/IEEE Design Automation Conference*, DAC '21, IEEE, pp. 451–456 (online), DOI: 10.1109/DAC18074.2021.9586326 (2021).

[4] Suzuki, Y., Sugiyama, T., Arai, T., Liao, W., Inoue, K. and Tanimoto, T.: Q3DE: A fault-tolerant quantum computer architecture for multi-bit burst errors by cosmic rays, *Proceedings of the 55th IEEE/ACM International Symposium on Microarchitecture*, MICRO-55, IEEE, pp. 1110–1125 (online), DOI: 10.1109/MICRO56248.2022.00079 (2022).

[5] Holmes, A., Jokar, M. R., Pasandi, G., Ding, Y., Pedram, M. and Chong, F. T.: NISQ+: Boosting Quantum Computing Power by Approximating Quantum Error Correction, *Proceedings of the ACM/IEEE 47th Annual International Symposium on Computer Architecture*, ISCA '20, IEEE, pp. 556–569 (online), DOI: 10.1109/ISCA45697.2020.00053 (2020).

[6] Hsieh, M.-H. and Le Gall, F. m. c.: NP-hardness of decoding quantum error-correction codes, *Phys. Rev. A*, Vol. 83, p. 052331 (online), DOI: 10.1103/PhysRevA.83.052331 (2011).

[7] Fowler, A. G.: Minimum Weight Perfect Matching of Fault-Tolerant Topological Quantum Error Correction in Average O(1) Parallel Time, *Quantum Info. Comput.*, Vol. 15, No. 1–2, pp. 145–158 (2015).

[8] Okamoto, K., Tanaka, T., Miyamura, M., Ishikuro, H., Uchida, K., Sakamoto, T. and Tada, M.: Cryogenic operation of NanoBridge at 4 K for controlling qubit, *Japanese Journal of Applied Physics*, Vol. 61, No. SC, p. SC1049 (2022).

[9] Edmonds, J.: *Paths, Trees, and Flowers*, Vol. 17, pp. 449–467 (online), DOI: 10.4153/CJM-1965-045-4, Cambridge University Press (1965).

[10] Das, P., Pattison, C. A., Manne, S., Carmean, D., Svore, K., Qureshi, M. and Delfosse, N.: A Scalable Decoder Micro-architecture for Fault-Tolerant Quantum Computing, arXiv:2001.06598 [quant-ph] (2020).

[11] Delfosse, N. and Nickerson, N. H.: Almost-linear time decoding algorithm for topological codes, *Quantum*, Vol. 5, p. 595 (online), DOI: 10.22331/q-2021-12-02-595 (2021).

[12] Das, P., Pattison, C. A., Manne, S., Carmean, D. M., Svore, K. M., Qureshi, M. and Delfosse, N.: AFS: Accurate, Fast, and Scalable Error-Decoding for Fault-Tolerant Quantum Computers, *Proceedings of the IEEE International Symposium on High-Performance Computer Architecture*, HPCA '22, IEEE, pp. 259–273 (online), DOI: 10.1109/HPCA53966.2022.00027 (2022).

[13] Varsamopoulos, S., Bertels, K. and Almudever, C. G.: Comparing Neural Network Based Decoders for the Surface Code, *IEEE Transactions on Computers*, Vol. 69, No. 02, pp. 300–311 (online), DOI: 10.1109/TC.2019.2948612 (2020).

[14] Liao, W., Suzuki, Y., Tanimoto, T., Ueno, Y. and Tokunaga, Y.: WIT-Greedy: Hardware System Design of Weighted ITerative Greedy Decoder for Surface Code, *Prodeedings of the 28th Asia and South Pacific Design Automation Conference*, ASP-DAC '23, IEEE, pp. 209–215 (2023).

[15] Drake, D. E. and Hougardy, S.: A simple approximation algorithm for the weighted matching problem, *Information Processing Letters*, Vol. 85, No. 4, pp. 211–213 (online), DOI: https://doi.org/10.1016/S0020-0190(02)00393-9 (2003).

[16] Bardin, J. C., Jeffrey, E., Lucero, E., Huang, T., Das, S., Sank, D. T., Naaman, O., Megrant, A. E., Barends, R., White, T., Giustina, M., Satzinger, K. J., Arya, K., Roushan, P., Chiaro, B., Kelly, J., Chen, Z., Burkett, B., Chen, Y., Dunsworth, A., Fowler, A., Foxen, B., Gidney, C., Graff, R., Klimov, P., Mutus, J., McEwen, M. J., Neeley, M., Neill, C. J., Quintana, C., Vainsencher, A., Neven, H. and Martinis, J.: Design and Characterization of a 28-nm Bulk-CMOS Cryogenic Quantum Controller Dissipating Less Than 2 mW at 3 K, *IEEE Journal of Solid-State Circuits*, Vol. 54, No. 11, pp. 3043–3060 (online), DOI: 10.1109/JSSC.2019.2937234 (2019).

[17] Yamanashi, Y., Kainuma, T., Yoshikawa, N., Kataeva, I., Akaike, H., Fujimaki, A., Tanaka, M., Takagi, N., Nagasawa, S. and Hidaka, M.: 100 GHz Demonstrations Based on the Single-Flux-Quantum Cell Library for the 10 kA/cm^2 Nb Multi-Layer Process, *IEICE Transactions on Electronics*, Vol. E93-C, No. 4, pp. 440–444 (online), DOI: 10.1587/transele.e93.c.440 (2010).

[18] Park, J.-S., Subramanian, S., Lampert, L., Mladenov, T., Klotchkov, I., Kurian, D. J., Juarez-Hernandez, E., Perez-Esparza, B., Kale, S. R., Asma Beevi, K. T., Premaratne, S., Watson, T., Suzuki, S., Rahman, M., Timbadiya, J. B., Soni, S. and Pellerano, S.: 13.1 A Fully Integrated Cryo-CMOS SoC for Qubit Control in Quantum Computers Capable of State Manipulation, Readout and High-Speed Gate Pulsing of Spin Qubits in Intel 22nm FFL FinFET Technology, *Proceedings of the IEEE International Solid-State Circuits Conference*, ISSCC '21, Vol. 64, IEEE, pp. 208–210 (online), DOI: 10.1109/ISSCC42613.2021.9365762 (2021).

[19] Van Dijk, J. P. G., Patra, B., Subramanian, S., Xue, X., Samkharadze, N., Corna, A., Jeon, C., Sheikh, F., Juarez-Hernandez, E., Esparza, B. P., Rampurawala, H., Carlton, B. R., Ravikumar, S., Nieva, C., Kim, S., Lee, H.-J., Sammak, A., Scappucci, G., Veldhorst, M., Vandersypen, L. M. K., Charbon, E., Pellerano, S., Babaie, M. and Sebastiano, F.: A Scalable Cryo-CMOS Controller for the Wideband Frequency-Multiplexed Control of Spin Qubits and Transmons, *IEEE Journal of Solid-State Circuits*, Vol. 55, No. 11, pp. 2930–2946 (online), DOI: 10.1109/JSSC.2020.3024678 (2020).

[20] Jokar, M. R., Rines, R., Pasandi, G., Cong, H., Holmes, A., Shi, Y., Pedram, M. and Chong, F. T.: DigiQ: A Scalable Digital Controller for Quantum Computers Using SFQ Logic, *Proceedings of the IEEE International Symposium on High-*

Performance Computer Architecture, HPCA '22, IEEE, pp. 400–414 (online), DOI: 10.1109/HPCA53966.2022.00037 (2022).

[21] Kolmogorov, V.: Blossom V: a new implementation of a minimum cost perfect matching algorithm, *Mathematical Programming Computation*, Vol. 1, No. 1, pp. 43–67 (online), DOI: 10.1007/s12532-009-0002-8 (2009).

[22] Duckering, C., Baker, J. M., Schuster, D. I. and Chong, F. T.: Virtualized Logical Qubits: A 2.5D Architecture for Error-Corrected Quantum Computing, *Proceedings of the 53rd Annual IEEE/ACM International Symposium on Microarchitecture*, MICRO-53, IEEE, pp. 173–185 (online), DOI: 10.1109/MICRO50266.2020.00026 (2020).

[23] Krinner, S., Storz, S., Kurpiers, P., Magnard, P., Heinsoo, J., Keller, R., Luetolf, J., Eichler, C. and Wallraff, A.: Engineering cryogenic setups for 100-qubit scale superconducting circuit systems, *EPJ Quantum Technology*, Vol. 6, No. 1, p. 2 (2019).

[24] Byun, I., Kim, J., Min, D., Nagaoka, I., Fukumitsu, K., Ishikawa, I., Tanimoto, T., Tanaka, M., Inoue, K. and Kim, J.: XQsim: Modeling Cross-Technology Control Processors for 10+K Qubit Quantum Computers, *Proceedings of the 49th Annual International Symposium on Computer Architecture*, ISCA '22, ACM, pp. 366–382 (online), DOI: 10.1145/3470496.3527417 (2022).