FPGA による量子コンピュータシミュレーションシステムの開発

1. 背景

実際の量子コンピュータの量子ビット数は増加傾向にあり，2020 年現在，古典コンピュータによるシミュレーションが困難な規模となっている．一方で，実際の量子コンピュータは量子ビットの安定性や量子ビット同士の接続の制約などから様々なアルゴリズム・アプリケーションの実装やデバッグに容易に利用できるものではなく，シミュレータの必要性は大きい．

量子コンピュータでは，量子ビット数が n 倍になる場合，それをシミュレーションする古典コンピュータで必要なメモリおよび計算回数は一般に 2^n倍となる．そのため，高速化・大規模化のために古典コンピュータを多数用意すればよいというわけではない．また，仮に必要な古典コンピュータを用意できたとしても，その相互接続網およびメモリ帯域のために演算性能は指数関数的に低下する．

そのため，シミュレーションの高速化・大規模化のためには，量ではなく質の向上を目指したアーキテクチャが必要であると考えられる．

2. 目的

本プロジェクトでは，FPGA を使った量子コンピュータシミュレータのアーキテクチャを検討・実装を目的とする．FPGA 上では，欲しい演算およびデータ型をアプリケーションに応じてユーザーが作り込むことができる．この特性を活用することで，量子コンピュータのシミュレーションに必要なリソースを，量から質に転換することを目指す．

これにより，現在の古典コンピュータを並べるだけでは不可能な規模の量子コンピュータのシミュレーションを現実的な時間で処理できる環境を構築することが本プロジェクトの目的である．

3. ソフトウェア開発内容

本プロジェクトでは，FPGA を使った量子コンピュータシミュレーションの実装に取り組んだ．汎用プロセッサあるいは GPGPU のようなアクセラレータと比較して高速・大規模・高精度のシミュレーションの実現を目標として開発を行った．実装するシミュレータの構成を図 1 に示す．実行環境として AWS-F1 のような環境を想定する．

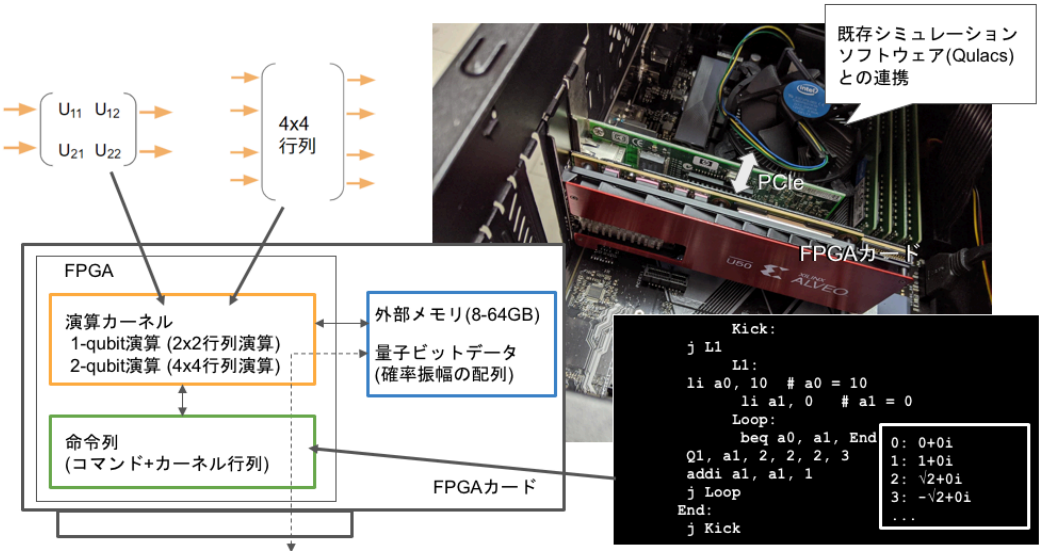


図 1: プロジェクトで目標とするシミュレータの構成

量子コンピュータシミュレータは，汎用プロセッサに対するアクセラレータとして実装することを想定し，汎用プロセッサから，適宜，実行制御できるように実装する．FPGA 上には，後述する 1 量子ビット演算用の 2x2 行列演算および，2 量子ビット演算用の 4x4 行列演算をカーネルとして実装する．
シミュレーションの対象量子ビット数は，FPGA ボードのオンボードメモリ(8 から 64GB 程度)に格納できる 20 から 30 量子ビット程度を想定する．

プロジェクトにおける開発内容

実装した量子コンピュータシミュレータに関して，(1)シミュレーションモデル，(2)演算カーネルの実装，(3)演算カーネル駆動部分の実装，(4)演算カーネルの基本性能，および，(5)データ供給・排出機構について述べる．

(1) シミュレーションモデル

量子コンピュータのシミュレーションを FPGA に実装するにあたって，シミュレーションモデルを検討した．Qulacs や幾つかのオープンソースのシミュレータ，教科書で解説されているシミュレーションモデルを比較した結果， Qulacs の 1 量子ビット向け演算および 2 量子ビット向け演算を参考に，図 2 のような 2x2 行列演算および 4x4 行列演算を FPGA 上に実装することとした．

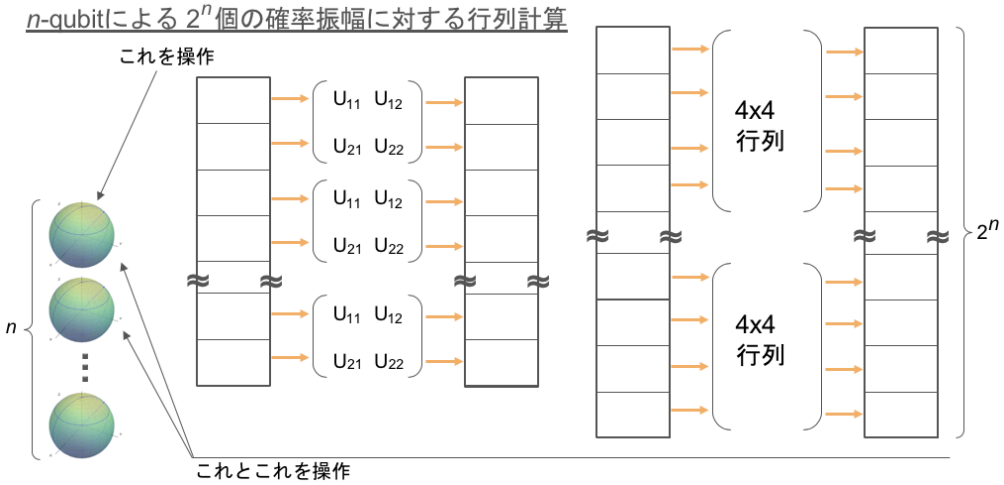


図 2: 実装するシミュレーションモデル

(2) 演算カーネルの実装

FPGA の特性を有効に活用するため，シミュレーションの肝である複素数演算数の積和演算処理向けのパイプライン処理カーネルを HDL により RTL で実装した.

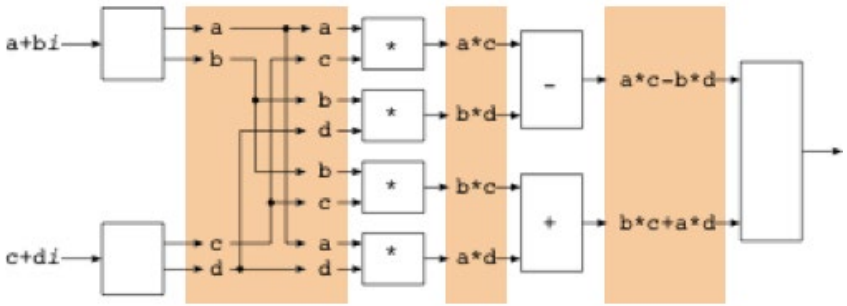


図 3: 複素数の演算パイプライン

図 3 は，二つの複素数の積を求める演算器の構成である. 図中の網掛け部分にレジスタを挿入することで，この演算器をパイプライン動作させることができる.

複素数の積和で構成されるシミュレーションの演算カーネルの実行の様子を図 4，図 5 に示す. 入力データが投入されはじめた後，間断なくデータが供給される場合，演算結果も間断なく出力される. 1 量子ビットに対する演算では，1 サイクルに 2 個のデータを受け取り 2 個のデータを出力する. 同様に，2 量子ビットに対する演算では 1 サイクルに 4 個のデータを受け取り 4 個のデータを出力する.

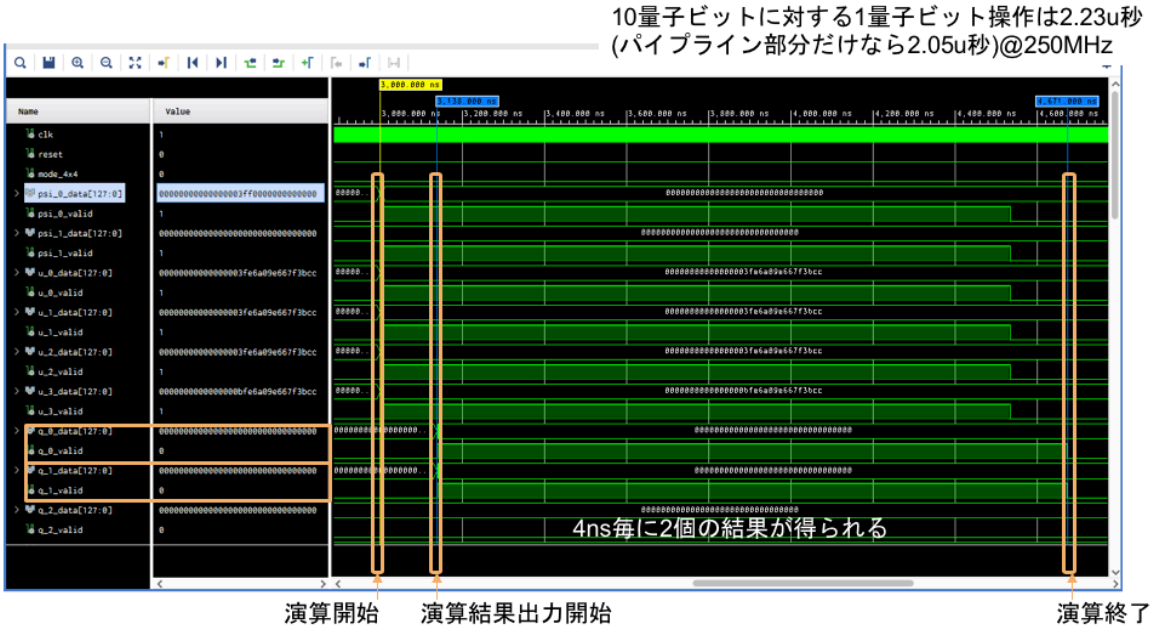


図 4: 10 量子ビットに対する 1 量子ビット操作の演算動作の様子

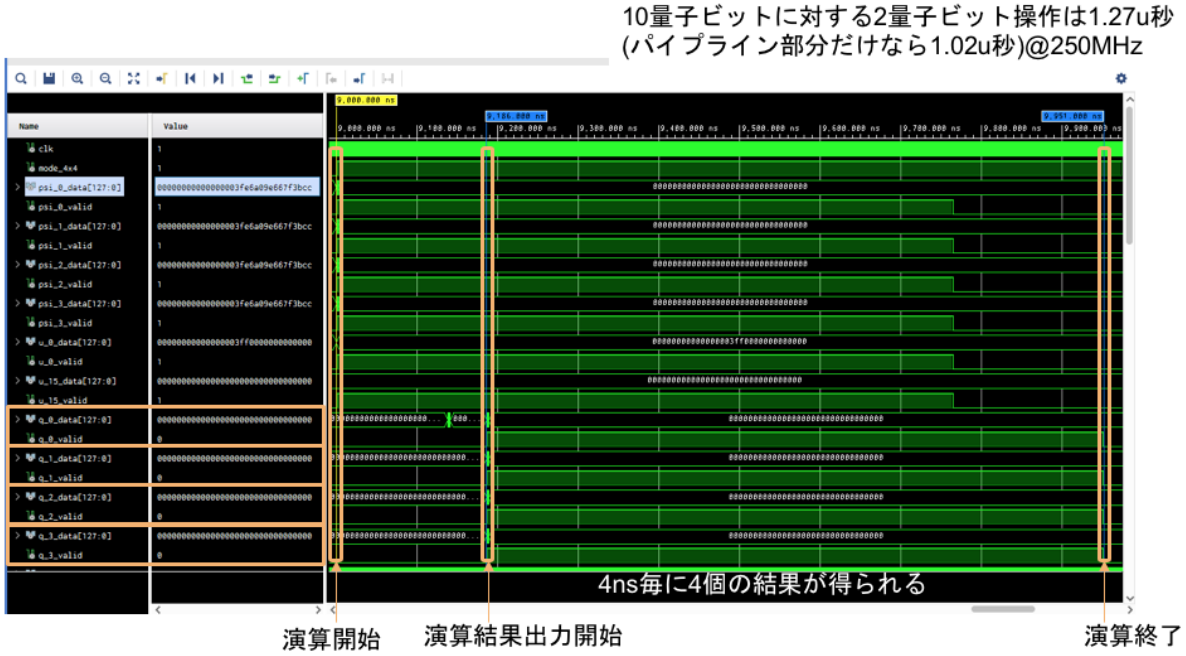


図 5: 10 量子ビットに対する 2 量子ビット操作の演算動作の様子

Vivado による合成，配置配線の結果，設計したモジュールは 250MHz で動作させられることがわかった．そのため，10 量子ビットからなる量子コンピュータの 1 量子ビットに対する演算には，2.23u 秒，2 量子ビットに対する演算には 1.27u 秒で完了する．

(3) 演算カーネルの駆動機構

演算カーネルは，2 量子ビットや 1 量子ビットを都度，ホスト PC から駆動することができるほか，内部の RISC-V 風の制御コントローラで実行制御できるように実装した．

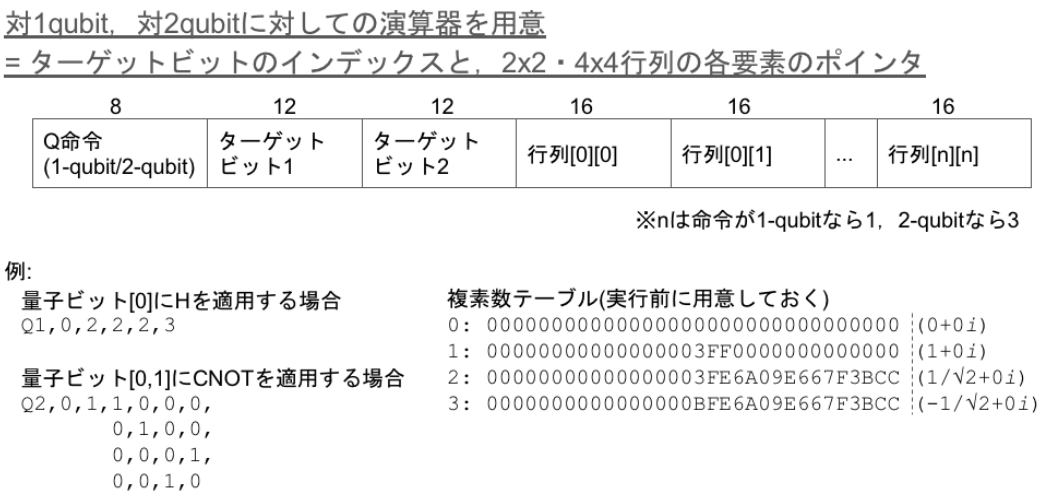


図 6: 量子コンピュータ用の演算命令フォーマット

図 6 に，シミュレータ内部に保持されるカーネル演算用の命令フォーマットに示す．様々な 1 量子ビット操作あるいは 2 量子ビット操作に対応できるように，2x2 行列および 4x4 行列の要素すべてをオペランドとして指定することができる点が特徴である．図 6 のように，あらかじめ複素数テーブルに（汎用プロセッサのプログラムにおけるテキスト領域に相当）使用する複素数を用意しておき，そのインデックスを，オペランドに指定し，アダマール演算や CNOT 演算などに相当する行列演算を実行する．
また，FPGA 単体でループ処理などの制御処理を実行できるよう RISC-V の基本命令セットである RV32I 程度の命令を持つ小さな制御コアを実装した．この制御コアを使用することで，図 7 のように，複数の量子ビットに対して操作を行うプログラムを実行させられる．

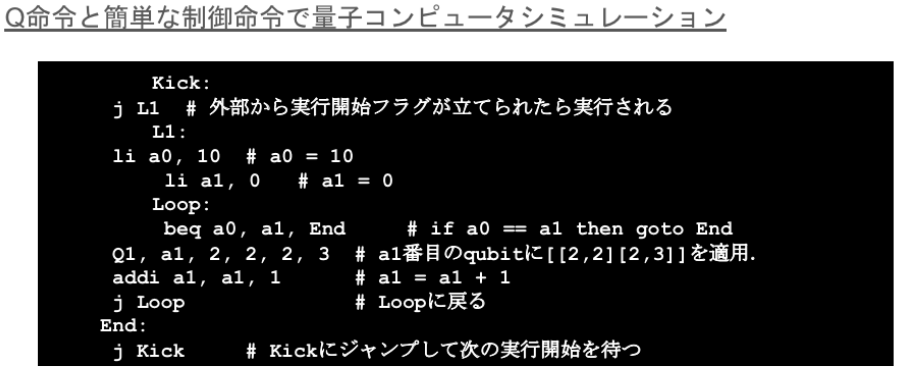


図 7: 量子コンピュータシミュレーション命令と制御命令のプログラムの例

(4) 演算カーネルの基本性能

演算カーネルの演算性能が最大になるのは，データを間断なく供給および排出できる場合であり，その場合，1 量子ビット演算では毎サイクル 2 量子ビット分の，2 量子ビット演算では毎サイクル 4 量子ビット分の演算結果が得られる．FPGA 上で必要となるリソース使用量を図 8 に示す．

	使用量	VU9P(AWS-F1)	Alveo U50	XC7Z020(ZYBO)
LUT	75,111(6.35%)	1,182,240	872,000	53,200
FF	127,552(5.39%)	2,364,480	1,743,000	106,400
DSP	684(10%)	6,840	5,952	220
BRAM		75.9Mb	47.3Mb	4.9Mb
URAM		270.0Mb	180.0Mb	

≦ 20 qubit 分

参考

図 8: 演算カーネルの実装に必要なリソース使用量

FPGA に内蔵されたメモリであり，毎サイクル読み書きできる BRAM を利用すると，演算データの供給・排出は問題ないが，その場合には高々20 量子ビット程度しかシミュレーションすることができない．目標で述べたように 30 量子ビット程度のシミュレーションを FPGA で実行するためには FPGA の外にあるオフチップメモリの使用は不可欠である．

(5) データ供給・排出機構

FPGA のオフチップメモリとの転送速度は，たとえば，1GHz DDR で 64bit 接続のメモリであれば 16GBps である．想定する実行環境である AWS F1 に搭載されている FPGA ボードの場合メモリが 4 個搭載されているため，入出力をうまく別メモリに割り当てることで，最大 32GBps でデータを流すことができる．
一方，演算カーネルは，毎サイクル 4 入力 4 出力の complex double のデータを扱うため，動作周波数が 250MHz の場合，常に，16GBps でデータ入出力を捌く必要がある．
オフチップメモリの帯域と，カーネル演算に必要な帯域を比べると，必要なデータを毎回オフチップメモリから読み出すような非効率的な実装ではなく，図 9 および図 10 のようなキャッシュ機構が必要になる．

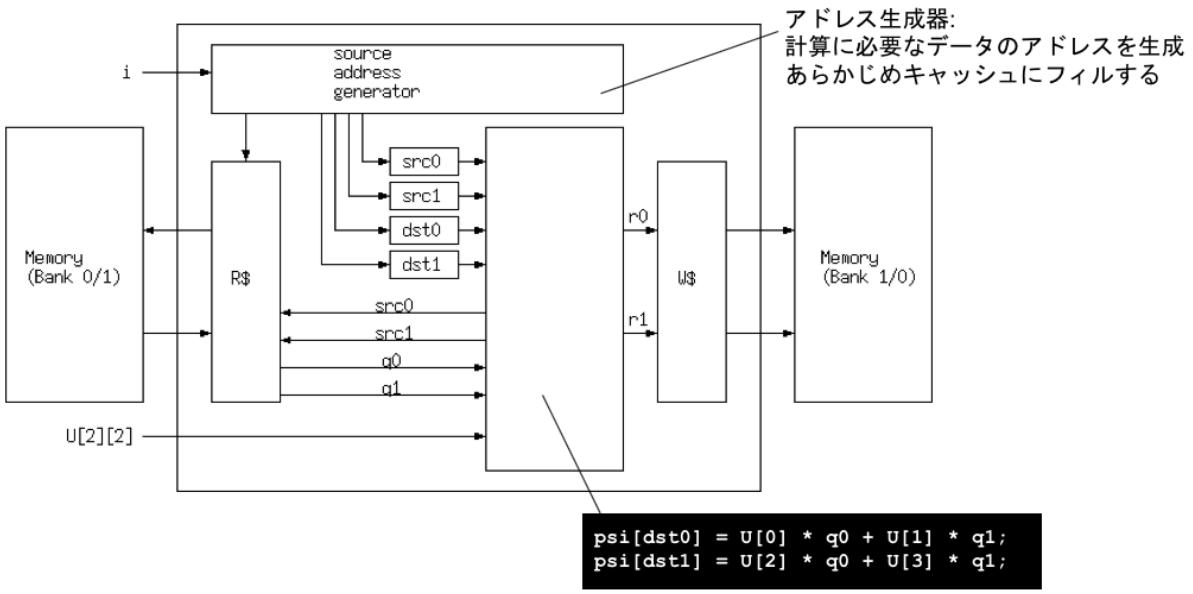


図 9: キャッシュと演算カーネルからなるシステム構成(1 量子ビット操作用)

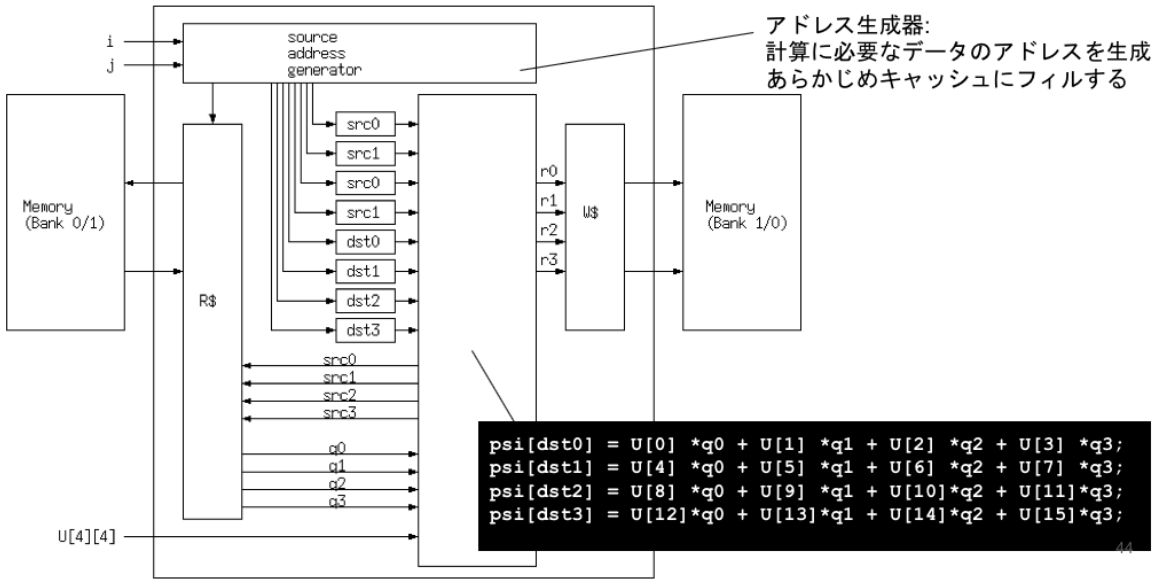


図 10: キャッシュと演算カーネルからなるシステム構成(2 量子ビット操作用)

FPGA からオフチップメモリを読み書きするコントローラとのインターフェースである AXI の仕様と現実的に実装可能な動作周波数を考慮すると，512bit 幅で動作周波数 300MHz の構成をとることで 19.2GBps 程度が利用できる.

ここで，量子コンピュータのシミュレーションに必要なメモリのアドレスアクセスパタンが，操作対象の量子ビットによって，大きくことなることに注意が必要である.

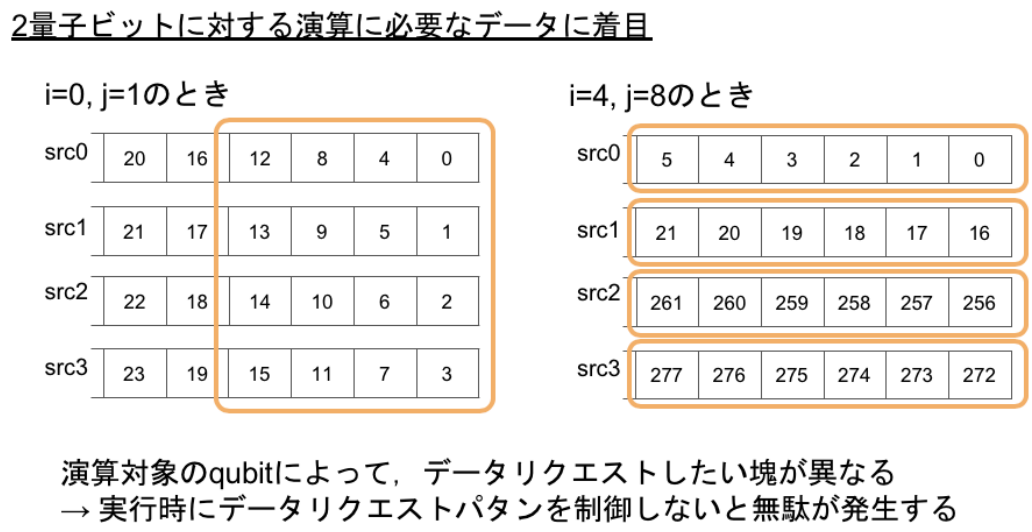


図 11: 操作対象の量子ビットとメモリアクセスパタンの違い

図 11 は，2 量子ビットに対する演算において，(a)量子ビット 0 と 1 に操作を適用する場合と，(b)量子ビット 4 と 8 に操作を適用する場合のメモリアクセスパタンを比較した図である．(a)の場合は，カーネルの 4 入力がまとまったデータを取るのに比べて，(b)の場合は，カーネルの 4 入力それぞれが異なったアドレスにアクセスしている．そのため，この 2 つのパタンでは，オフチップメモリからキャッシュに読み出す時のアドレス指定のパタンを変更する必要がある．

本プロジェクトでは，この問題を解決し，どのような量子ビット操作に対しても必要なメモリ転送性能が得られるアーキテクチャを検討した．

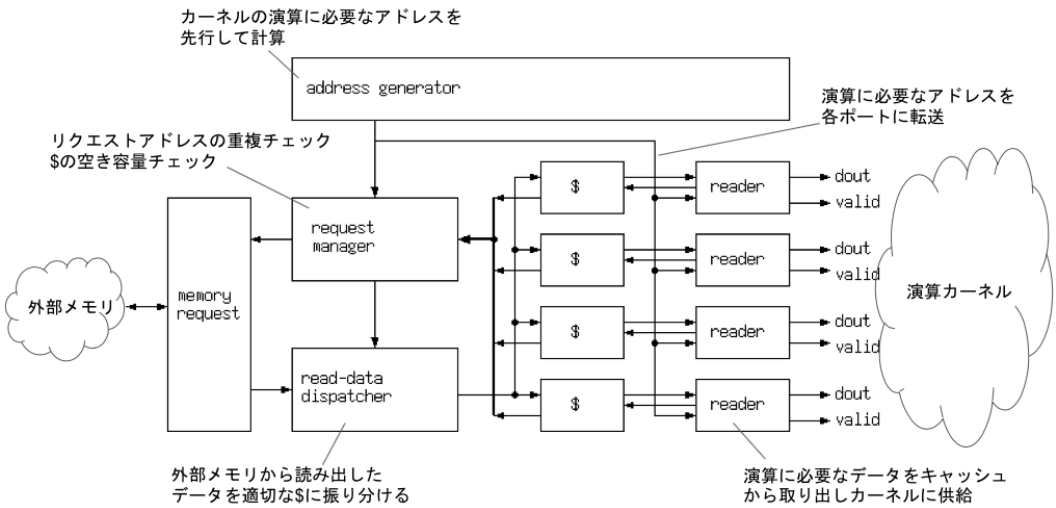


図 12: データアクセス機構

図 12 に提案するデータアクセス機構を示す．提案するデータアクセス機構は，演算に必要なデータが格納されているアドレスを先行して計算する address_generator，計算

されたアドレスから必要なオフチップアクセスを選択する request_manager，実際にオフチップメモリへのアクセスを発行する memory_request，オフチップから読み出したデータを必要な演算カーネルごとのキャッシュ($)に保存する read-data dispatcher から構成される．キャッシュさえデータが載ってしまえば，連続的にキャッシュからデータを読み出し演算カーネルに供給することで，間断なくシミュレーションに必要な演算処理を実行させることができる．

たとえば，オフチップメモリアクセスとのアクセス単位が 256 個であるとする．このとき，図 11 のように 0 量子ビット目と 1 量子ビット目に操作を適用する場合，address_generator は，0,1,2,3,…と必要なアドレスを生成する．この生成列に対して，request_manager は，アドレス 0 に対するデータアクセスを，memory_request に転送し，オフチップメモリアクセスを実行させる．一方，4 量子ビット目と 8 量子ビット目に対する操作の場合には，0,16,256,272,…というアドレス列を address_generator が生成し，request_manager によって，0 と 256 からの読み出しだけが実際のオフチップメモリアクセスとして実行される．0 から読み出したデータは，read-data dispatcher で，0 番目と 1 番目のキャッシュだけに書き込まれ，256 から読み出したデータは 2 番目と 3 番目のキャッシュに書き込まれる．

4. 新規性・優位性

本プロッジェクトで検討・実装をすすめたアーキテクチャについて，単体コアを FPGA に実装するときの演算性能の見積もり結果と汎用プロセッサや GPU 上で実行される既存シミュレータの処理性能を比較する．また，その結果を踏まえ，提案するアーキテクチャの高性能化・大規模化・高精度化の可能性と課題について述べる．

(1) 演算性能見積もり

演算カーネルおよびデータ供給・排出機構により，FPGA のメモリ入出力帯域をほぼ使い尽くすシミュレータを実装できる見積もりが得られた．この場合の演算処理性能の見積もり結果を図 13 に示す．

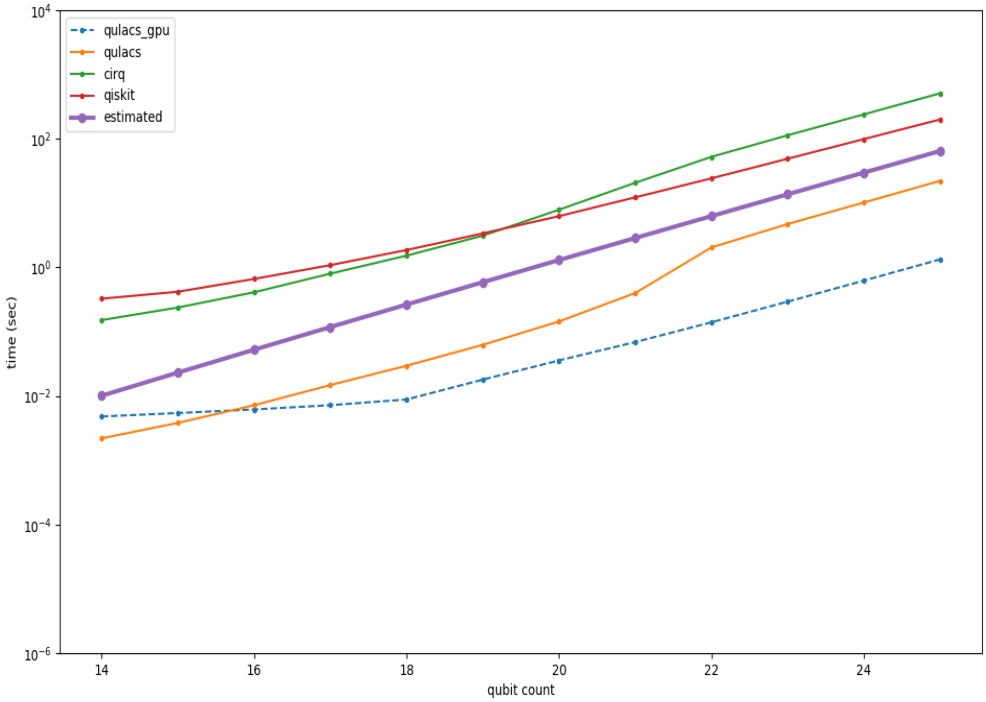


図 13: 既存のシミュレータの処理性能と FPGA シミュレータの処理性能見積もりの比較

これは，シミュレーション対象の n 個の量子ビットに対し，

1. n 個の量子ビットに対しランダムな角度の X 回転操作を行なう
2. 隣り合う量子ビット同士に CNOT 操作をおこなう
3. n 個の量子ビットに対しランダムな角度の X 回転操作を行なう
4. 2.から 3.を n 回繰り返す

という処理をベンチマークとして処理性能を測定，見積もりしたものである．

FPGA 以外での実装に関するベンチマーク結果は，Qulacs の Web サイト https://github.com/qulacs/qulacs より引用したものである．

図 13 の紫の実戦が FPGA によるシミュレータの処理性能の見積もり結果である．検討したアーキテクチャの演算カーネルの処理性能は，量子ビット数に依存しない．そのため，量子シミュレータを構成する量子ビット数に応じた計算回数にほぼ比例した計算時間となる．

図 13 から見てとれるよう，今回検討したアーキテクチャによって FPGA で達成可能な処理性能は，業界最速のソフトウェアシミュレータである Qulacs と比べて，汎用プロセッサ(Intel Xeon E5-2694 v4 2.6GHz) に対しては 1/8 程度，GPGPU(NVIDIA Tesla V100) に対しては 1/64 程度であることがわかった．

(2) 高性能化・大規模化・高精度化へのアプローチ

本プロジェクトの下で検討したアーキテクチャをベースにした高速化・大規模化・高精度化について，さらなる検討をおこなった．

(a) 高速化

本プロジェクトで検討した演算カーネルはパイプライン処理によって，1 量子ビットに対する操作では 2 サイクルに 2 個，2 量子ビットに対する操作では 4 サイクルに 4 個結果を得ることができる．処理スループットはデータ入出力帯域で律速されているため，HBM を搭載した次世代 FPGA を使うことで，全体の性能を向上させることができると考えられる．

単純には，HBM を搭載した Xilinx Alveo U50 をターゲット FPGA とする場合，メモリの最大データ転送帯域が 200GBps 程度になるため，同じアーキテクチャのまま演算カーネルを 8 個並列に実行できるように並べることで，8 倍の高速化ができることが期待できる．

また，単純に並べるだけでなく，操作対象の量子ビット数を増加させることも考えられる．その場合，必要になるデータ転送帯域は増加するが，1 個の結果を得るために必要な演算量が増加するため，相対的に，演算に対して必要なデータ転送帯域を削減できると考えられる．

図 14 は，演算カーネルを並列化できた場合の実行性能の見積もりと，既存シミュレータの実行性能を比較した結果である．

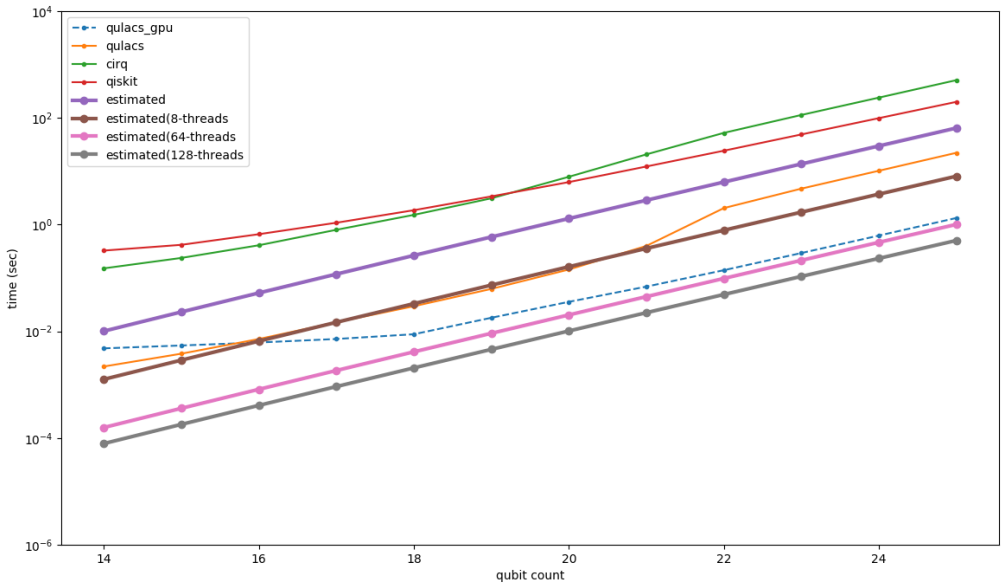


図 14: 演算カーネルを並列化できた場合の実行性能見見積もりと既存シミュレータの実行性能の比較

今回対象とした通常の DDR メモリを搭載した FPGA 向けの演算カーネルが一つの実装（estimated・紫）では，汎用プロセッサとくらべて低い演算性能であるが，
8 並列できた場合の見積もり（estimated 8-threads・茶）では，汎用プロセッサを凌駕する実行性能が得られる．
ただし，GPU(V100)の演算性能を超えるためには，さらに 8 倍の処理性能向上が必要になる．これを実現するためには，転送帯域に対する演算量を 8 倍程度に増やす必要があり，8 から 16 量子ビットに対する操作を，演算カーネルとして FPGA 上に実装する必要があると考えらえる．

(b) 大規模化

量子コンピュータのシミュレーションに必要なメモリ量は，対象とする量子ビット数が n 倍になると 2^n 倍になる．64GB のオフチップメモリを搭載する FPGA ボードの場合には，確率振幅を complex double で保持する場合には，30 量子ビット程度が限界となる．この限界を超えて大規模化するためには，複数 FPGA を並べる必要がある．汎用プロセッサあるいは GPU を使った場合でも，規模の大小は別にして，複数ノードを並べなければならない点には変わりはない．
複数ノードを並べてシミュレーション処理を行う場合に実行性能を律速するのはノード間のデータ転送帯域である．汎用プロセッサあるいは GPU の場合，アプリケーションレイヤのソフトウェアが他ノードにあるデータを読み書きする場合には，物理的なネットワークコネクションの上にあるソフトウェアプロトコルスタックを介する必要がある．そのため，レイテンシの観点でもスループットの観点でもオーバヘッドが生じる．一方で FPGA の場合，I/O と演算器を直結することができるため，ノード間通信に必要なオーバヘッドを削減できることが期待できる（図 15）．

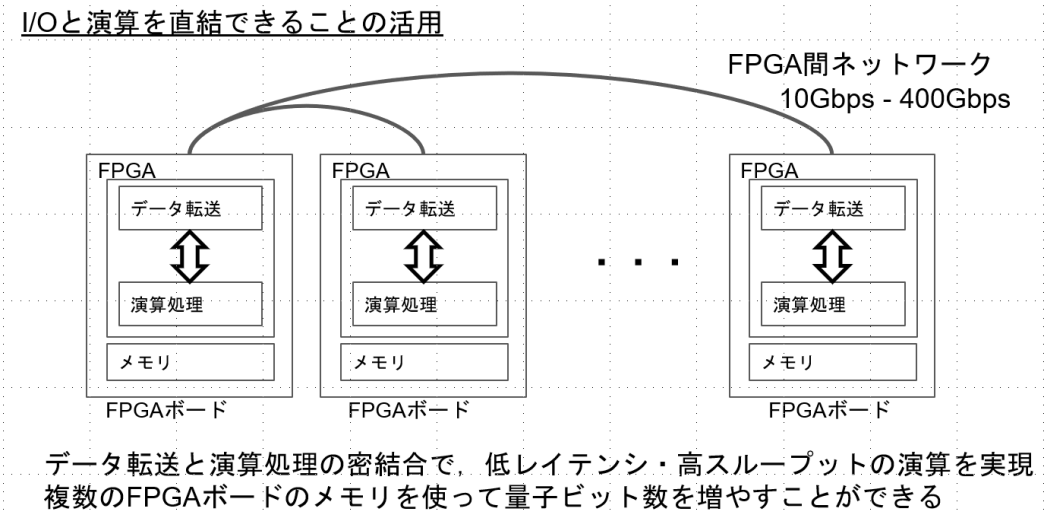
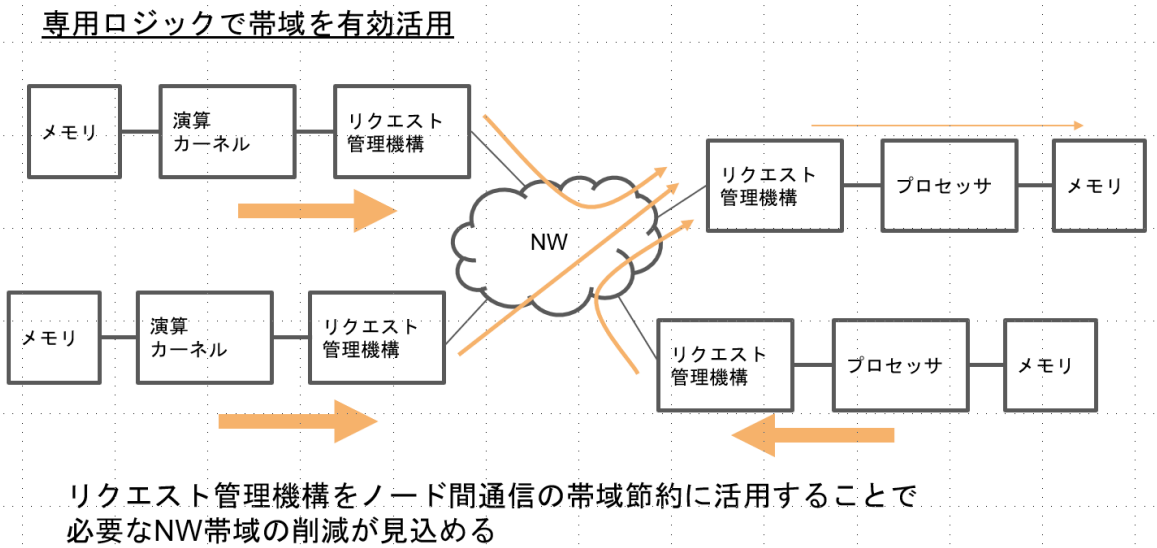


図 15: FPGA ノード間通信

2020 年時点では，FPGA 間のネットワーク帯域は 10Gbps から 400Gbps 程度が期待できる．本プロジェクトで検討したアーキテクチャであれば，ノード間通信の帯域に

128Gbps を確保することができれば，オンボードのオフチップメモリからデータを読み書きして演算カーネルを駆動させる場合と同様の演算性能を得ることができる．
図 11 に示した通り，量子シミュレータでは操作の対象となる量子ビットのインデックスに応じて，アクセスすべきメモリアドレスが異なるパタンを持つ．そのため，複数のノードを使って，シミュレーションに必要なメモリを分割した場合，メモリアクセスが特定のノードに集中する可能性が排除できない．
これを解決するためには，ノード間通信を適切に制御する，不要あるいは冗長なノード間通信を発生させない機構が必要になる．この機構は，本プロジェクトの下で検討したデータ取得・排出機構（図 12）を流用することで実装できる．

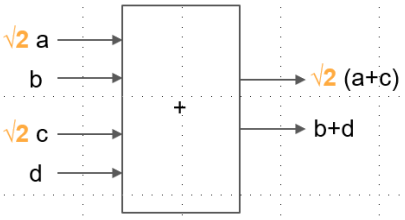


すなわち，それぞれのリクエスト管理機構でノード上の演算カーネルが必要なデータアクセスの冗長性を排除し，さらに複数のノードからのリクエストの冗長性も排除することで，実質的にネットワークおよびオフチップメモリへのアクセスに必要な帯域を節約することができる．

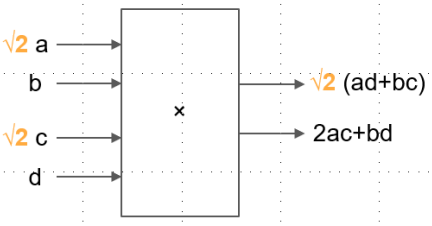
(c) 高精度化
FPGA を利用するメリットの一つに，アプリケーションに応じたデータ型を定義できること，パイプライン演算器を構成できることがある．本プロジェクトでは，複素数の積和演算のためのパイプライン演算器を実装，活用することで，メモリ帯域スループット相当の演算を実現することができた．
ここで，汎用プロセッサと同じ複素数演算を利用するのではなく，$1/\sqrt{2}$ を含むデータを基本型および，そのためのパイプライン演算器を実装することができる．$1/\sqrt{2}$ があれば，$\pi/8$ 相当の回転操作ができる．

FPGAならではの特殊演算器の実装 (ex. √2の演算器)

(a + 1/√2 b) + (c + 1/√2 d)
= 1/√2 (√2 a + b) + 1/√2 (√2 c + d)
= 1/√2 (√2 (a + c) + (b + d))



(a + 1/√2 b)(c + 1/√2 d)
= 1/√2 (√2 a + b) 1/√2 (√2 c + d)
= 1/2 (√2 a + b)(√2 c + d)
= 1/2 ((2ac + bd) + √2 (ad + bc))



あらわれる無理数を√2に制限すると，性能変わらず高精度化が実現できる

図 16: 1/√2 を基本型として持つ演算器

1/√2 演算の演算器の実現例を図 16 に示す．1/√2 の積和は複数の積和に展開して実行することになるが，FPGA 上に実装する場合，パイプライン並列性によってスループットが 1 の演算器を構成することができる．

5. 期待されるユーザー価値と社会へのインパクト

本プロジェクトのユーザー価値は，(1)FPGA を使った量子コンピュータシミュレータの実装事例と可能性，(2)FPGA を使ったランダムメモリアクセスアプリケーションの実装事例，(3)実際の量子コンピュータの制御方式への転用の可能性，を示した点があげられる．また，(4)FPGA コミュニティに対する量子コンピュータ関連の啓蒙活動にもなったことも本プロジェクトの価値と考えらえる．

(1) FPGA を使った量子コンピュータのシミュレータ実装事例と可能性
4.2 の成果で述べた通り，本プロジェクトでは量子コンピュータのシミュレータを実装するための，演算カーネルおよびデータ転送アーキテクチャを検討し，その実行性能の見積もりを得た．また，高性能化，大規模化，高精度化に向けての可能性について検討した．

(2) FPGA を使ったランダムメモリアクセスアプリケーションの実装事例として
量子コンピュータのシミュレーションでは，操作の対象とする量子ビットに応じて，アクセスするメモリパタンが大きく異なるという問題がある．本プロジェクトでは，メモリ取得・排出機構で，この問題に対する解を示した．このアーキテクチャは，量子コンピュータのシミュレーションのみならず，ランダムメモリアクセスが必要となるグラフ処理のような

アプリケーションを FPGA に実装する場合にも活用できる可能性があると考えられる.

(3) 実際の量子コンピュータの制御方式への転用可能性
超電導量子ビットをマイクロ波で制御するタイプの量子コンピュータをはじめとして，多くの量子コンピュータの制御には FPGA が利用されている. 今回は，アクセラレータとして，1 量子ビット操作，2 量子ビット操作を専用命令とするアーキテクチャを検討した. この命令セットは，シミュレータだけでなく実機であっても同様に活用できると考えられる.

(4) FPGA コミュニティへの量子コンピュータ関連の啓蒙活動
量子コンピュータのシミュレーションは，演算の並列性がある一方で，データアクセスパタンが複雑であるという特徴を持っている. そのため，汎用プロセッサや GPU に対して，FPGA によるアプリケーション特化アーキテクチャの活用できる可能性が高い. このプロジェクトの活動を通じて，FPGA コミュニティに対して，量子コンピュータシミュレーション開発の興味深さを啓蒙することができた.

6. 氏名（所属）

三好 健文（わさらぼ合同会社）