Vol.2019-HPC-171 No.6
2019/9/20

再構成可能なハードウェアを用いた演算と通信を融合する手法の提案と性能評価

藤田 典久[1]　小林 諒平[1,2]　山口 佳樹[2,1]　朴 泰祐[1,2]

概要：近年，高性能計算の分野で再構成可能なハードウェアである Field Programmable Gate Array (FPGA) が次世代の演算加速装置として注目されている．FPGA を高性能計算で用いる際の障壁は開発の困難さであったが，高位合成手法の発展に伴いこの問題は解決しつつある．最新の FPGA は最大で 100Gbps×4 の通信性能を有しており，我々はその強力な通信性能に注目している．FPGA の絶対性能は他のアクセラレータよりも低いが，FPGA が持つ演算能力と通信能力を組み合わせることでより広い範囲の問題に FPGA が適用できると考えている．本研究の目的は，高位合成で記述された FPGA アプリケーションから通信機構を操作し並列処理システムを実現することである．通信のスループットやレイテンシだけでなく，通信と演算を一体化したパイプラインが FPGA 内に構築される点も評価を行い，高位合成で記述した FPGA アプリケーションで並列計算が可能なことを示す．我々は FPGA 間で直接通信を行う環境として CoE というシステムを開発しており，バンド幅は最大で 90.7Gbps を達成し，最小レイテンシは 429.2ns であった．また，パイプライン評価においても，良好な結果が得られ，通信と演算を一体化したパイプラインを構築できていることを確認した．

1. はじめに

近年，高性能計算の分野で再構成可能なハードウェアである Field Programmable Gate Array (FPGA) が次世代の演算加速装置として注目されている．アクセラレータは多数のコアと広い演算幅の Single Instruction Multiple Data (SIMD) 演算器を持つため，ヘテロジニアスなシステムでは Central Processing Unit (CPU) はレイテンシコア，アクセラレータはスループットコアとして用いられる．それらのアクセラレータに加え，FPGA は CPU とアクセラレータのトレードオフの中間にあるデバイスである．

従前，FPGA を高性能計算で用いる際の障壁は開発の困難さであったが，高位合成 (High Level Synthesis: HLS) 手法の発展に伴いこの問題は解決しつつある．FPGA 開発では，Verilog HDL や VHDL といったハードウェア記述言語 (Hardware Description Language: HDL) を用いて回路を記述することが一般的であった．HDL は 1 クロック・1 ビットの粒度で回路の動作を記述するもので，利用に際してハードウェアの知識が必要となり，計算科学者が FPGA を用いる際の障壁となっていた．一方，高位合成は C や C++等のソフトウェア開発で用いられる言語を用いてハードウェアを記述するものであり，ハードウェア開発に精通していない科学者が FPGA を利用することを可能とするものである．しかしながら，演算性能・メモリバンド幅の両面で，現在の FPGA はアクセラレータとして広く用いられている GPU に敵わない．したがって，既に GPU で高速に計算できる問題を FPGA に適用しても，GPU の性能を超えられるとは言い難い．FPGA を高性能計算で用いる場合は，アプリケーションの中のどの計算が FPGA に適するかを見極めて適用することが重要だと言える．

我々は FPGA の持つ強力な通信性能に注目している．最新の FPGA は最大で 100Gbps×4 の通信性能を有している．また，それらの通信機構は直接 FPGA の内部のルーティング回路に接続されており，オーバーヘッドの少ない FPGA 間通信を可能とする．NVIDIA GPU では GPUDirect for RDMA (GDR) [1] と呼ばれる技術があり，外部の通信機構が GPU のメモリに直接アクセスができる．しかしながら，通信の制御は CPU が行う必要があり，GPU が主体となって通信できる技術ではない．したがって，CPU-GPU 間の同期コストや，通信データが PCI Express (PCIe) を経由して通信機構とやり取りされるオーバーヘッドは残されている．FPGA を単体のアクセラレータとして見ると適用できる範囲は小さいが，FPGA が持つ演算能力と通信能力を組み合わせることでより広い範囲の問題に FPGA が適用できると考えている．

[1] 筑波大学 計算科学研究センター
[2] 筑波大学 システム情報工学研究科

本研究の目的は，高位合成で記述された FPGA アプリケーションから通信機構を操作し，高位合成で記述した FPGA アプリケーションで並列計算が可能なことを示すことである．高位合成から通信できるシステムを開発し，それの性能評価を行う．通信のスループットやレイテンシの評価だけでなく，通信と演算を一体化したパイプラインが FPGA 内に構築される点も評価を行う．通信と演算がパイプライン化されることにより，通信と演算がオーバーラップされ通信隠蔽が実現できる．

本稿で用いる通信システムは，これまでの研究 [2] の実装を元にして改良を加えたものである．本稿で新たに実装された機能は主に以下の通りである．

- 100Gbps 通信対応
- アプリケーションに応じた通信コードの自動生成
- 直結網を前提とし，軽量なプロトコルの採用

2. 関連研究

OpenCL を FPGA で用いてアプリケーションやベンチマークの性能評価を行った論文はいくつか報告されている．[3] では，元々 GPU 向けに作成されたコードをそのまま FPGA 向けに用いても性能が悪く，OpenCL コードが FPGA 向けに最適化されている必要があると述べられている．[4] で，著者らは Maxwell 方程式の解を求めるプログラムを OpenCL を用いて FPGA に実装した．このアプリケーションは非構造格子を用いるため，メモリアクセスが規則的な格子を用いる場合よりも複雑になる．FPGA 向けのメモリアクセス最適化を適用し，マルチスレッド化された CPU 実装とくらべて，約 2 倍の性能を達成した．HPC 研究会においても，[5] や [6] で FPGA と OpenCL を用いた研究報告がなされているが，どちらでも OpenCL の最適化の困難さ，すなわち，CPU や GPU と異なる記述スタイルが必要であると述べられている．FPGA の絶対性能は GPU など他のアクセラレータと比べると低く，どのような種類の処理を FPGA にオフロードするのかが重要となる．

複数の FPGA をネットワークで接続して利用する研究はいくつか知られている．[7] では，データセンター内に 2D トーラスネットワークを持つ FPGA クラスタを構築し，Bing Web 検索エンジンのアクセラレータとして用いている．また，エジンバラ大学では Maxwell FPGA クラスタ [8] を構築している．しかし，Maxwell クラスタでは独自の Parallel Toolkit (PTK) と呼ばれる HDL を用いたフレームワークでアプリケーションを開発しており，高位合成は用いていない．

本研究の独自性は，高位合成が持つ高い生産性と，FPGA が持つ高い通信能力を併せて利用し，並列計算を行うことである．FPGA と OpenCL を用いる研究は行われているが，1 FPGA のみを用いているものであり，並列計算が行われている例は知られていない．

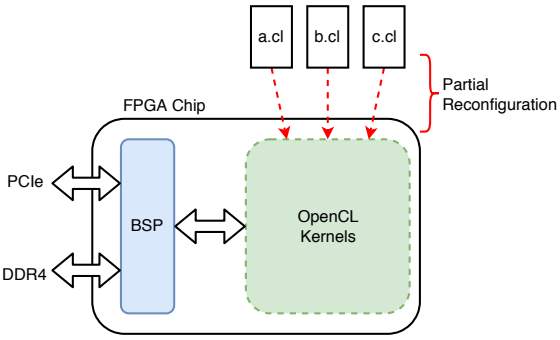

図 1: OpenCL 利用時における FPGA の内部構造の概略図．

3. Intel FPGA SDK for OpenCL

3.1 概要

Intel 社は自社の FPGA 向けの高位合成の処理系として，Intel FPGA SDK for OpenCL を公開している．この SDK を用いることで，OpenCL 言語を用いて FPGA をプログラミングできる．OpenCL から FPGA 上のハードウェアを生成するだけでなく，ホスト CPU で用いるドライバやランタイムライブラリも提供されており，この SDK のみでシステム全体を構築できる．

OpenCL 利用時の FPGA ハードウェアの構造の概略図を**図 1** に示す．図からわかるように，FPGA の内部構造は，大きくわけて 2 つの部分から構成される．1 つは Board Support Package (BSP) 由来の部分，もう一つはコンパイル対象の OpenCL コードから由来する部分である．

BSP は本 OpenCL SDK 固有の要素であり，異なる FPGA ボード上で同じ OpenCL プログラムを扱うために存在する．OpenCL 対応の FPGA ボードは多数あり，各ボードでハードウェアの構成が異なる．例えば，それぞれのボードで搭載されているメモリの種類や FPGA チップが異なる場合がある．したがって，それぞれのボードに固有の情報を OpenCL コンパイラに与える必要があり，BSP がその役割を担う．BSP にはペリフェラルコントローラが含まれており，OpenCL のアクセラレータとして利用するための最低限のものとして，PCIe コントローラと Dual Data Rate (DDR) メモリコントローラが含まれる．

3.2 OpenCL コードから生成されるパイプライン

Intel FPGA SDK for OpenCL を用いて OpenCL コードからハードウェアを作成すると，FPGA 内に OpenCL コードに対応するパイプラインが生成される．すなわち，FPGA 上に何らからの CPU を実装し，そのプロセッサ上でソフトウェアとして OpenCL コードが実行されるものではない．

図 2 と**図 3** に OpenCL コードからコンパイラによって生成されるパイプラインの概要を示す．Intel FPGA SDK

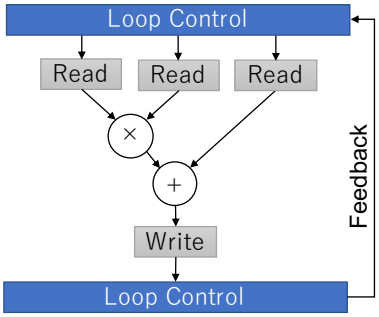

図 2: OpenCL コンパイラによって作られるパイプライン構造の例.

```
for (int i = 0; i < n; i++) {
    d[i] = a[i] * b[i] + c[i]
}
```

図 3: 図 2 の元となった OpenCL コードの一部.

for OpenCL のコンパイラ (aoc コマンド) の基本的なハードウェア生成ルールは，ループをパイプライン構造に変換することである．ループ内のコードが一つのパイプラインになり，その前後にループを制御するためのハードウェアが組み込まれる．また，メモリアクセスはコード内のアクセス一箇所づつ Load Store Unit (LSU) に置き換えられ，パイプラインに組み込まれる．

3.3 Channel 拡張

Intel FPGA SDK for OpenCL は OpenCL 言語に対していくつかの FPGA に特化した拡張を加えている．拡張の 1 つに，“Channel” 機構がある．Channel は OpenCL カーネル間でデータを直接交換するパイプのようなものである．カーネル間でデータをやりとりする場合，グローバルメモリに経由してデータを交換する方法が一般的であるが，Channel を用いる場合は FPGA 内部にデータパスが構成され，チップ外にあるメモリにアクセスすることなく通信ができる．したがって，従来手法と比べて高性能であり，メモリアクセスに関する回路が生成されないため省リソースとなる．Channel には “I/O Channel” と呼ばれる種類の Channel があり，これはカーネル間ではなく，BSP とカーネルの間を接続するために存在し，主に BSP にあるペリフェラルコントローラと OpenCL カーネル間の接続に用いられる．

3.4 通信機構に対応する BSP

通信機構を持つ OpenCL 対応の FPGA ボードであっても，そのボード用の BSP にコントローラが含まれていることは一般的ではなく，OpenCL から通信機構を操作し並列計算を行うためには，制御用のロジックを BSP に組み込む必要がある．我々は，これまでの研究 [9], [10] で，

表 1: これまでの研究との比較.

	これまでの研究 [2]	提案手法
FPGA	Arria 10	Stratix 10
通信速度	40Gbps	100Gbps
プロトコル	Ethernet	SerialLite III
トポロジー	Switch	Peer-to-Peer

OpenCL の BSP に通信機構のコントローラを追加し，それを OpenCL コードから制御できることを明らかにしている．

本稿で実装した OpenCL 通信機構は，これらの研究で得られた知見を元に開発したものである．なお，BSP に対してコントローラを追加し，OpenCL カーネルから用いる手法の詳細については本稿では省略する．前述した研究会報告と論文を参照して頂きたい．

4. CoE 2.0

4.1 概要

我々は FPGA 間で直接通信を行う環境として CoE というシステムを開発している．CoE の基本コンセプトは，前述した Channel による通信を FPGA 内だけでなく，ノード間に拡張するものである．CoE システムは，**図 4** にある様に，BSP に追加された外部通信コントローラ (本稿では SerialLite III [11]) および周辺回路 (HDL で記述)，OpenCL カーネルで記述された制御カーネル群と，それらの間をつなぐ Channel，CoE システムとアプリケーションの境界にある Channel である “CoE Channel” から構成される．

図 5 に CoE を用いて通信する際の簡単な例を示す．2 つのカーネル関数は異なる FPGA 上で動作しており，送信側の “`simple_out`” channel と受信側の “`simple_in`” channel が CoE システムを通じて繋っており，通信が可能となる．

本稿で用いる CoE システムは，1 章で述べた通り，これまでの研究 [2] の実装を元にして改良を加えたものであり，CoE 2.0 と呼称している．**表 1** にこれまでの研究との比較を示す．CoE 2.0 では主に，100Gbps 通信への対応と，様々なアプリケーション開発に利用できるようにコード生成システムを改良を加えた．また，通信プロトコルを Ethernet から SerialLite III に変更している．SerialLite III は Intel が開発している高効率な Peer-to-Peer なプロトコルであり，直結網に限定されるが Ethernet プロトコルと比べて低オーバヘッドで利用できる．

4.2 coe-gen: CoE コードジェネレータ

CoE では “CoE Channel” を通じて通信を行うが，何個の CoE Channel を実装するか，それぞれの Channel が扱うデータ型を何にするか，は CoE を用いるアプリケーショ

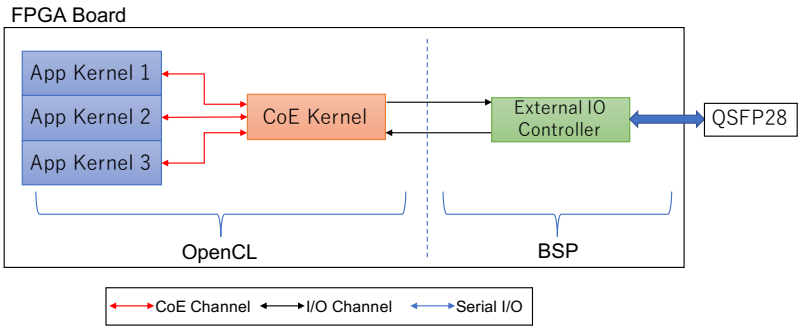

図 4: CoE システムの全体図．赤矢印は CoE Channel，黒矢印は I/O Channel，青矢印は高速シリアル通信を表す．

sender code on FPGA1

```
__kernel void sender(__global float* restrict x, int n) {
    for (int i = 0; i < n; i++) {
        float v = x[i];
        write_channel_intel(simple_out, v);
    }
}
```

receiver code on FPGA2

```
__kernel void receiver(__global float* restrict x, int n) {
    for (int i = 0; i < n; i++) {
        float v = read_channel_intel(simple_in);
        x[i] = v;
    }
}
```

図 5: CoE を用いて通信を行うコード例．

ンによって異なる．したがって，CoE 2.0 では，XML ファイルにこれらの情報を記述し，ジェネレータプログラム (coe-gen) を用いて CoE カーネルを生成する．

coe-gen を用いる場合の開発フローを**図 6** に示す．この図の中に示されているファイルの内容は以下の通りである．

app.xml
　定義ファイル
app_coe.cpp, app_coe.h
　CoE ランタイム (for Host)
app_cl.h
　CoE カーネル (OpenCL)
app.cpp
　ユーザアプリケーション本体 (for Host)
app.cl
　ユーザアプリケーションカーネル (OpenCL)

図 6 の app.xml ファイルが CoE のネットワーク構造を定義するファイルであり，この情報を元にして coe-gen コマンドが各種ファイルを生成する．

coe-gen へ入力する XML ファイルの例 (一部のみ抜粋) を**図 7** に示す．channel タグによって CoE チャンネルの仕様が定義され，これに合うようにカーネルなどが生成される．name 属性は CoE channel の OpenCL コードにおける識別子，type タグはその channel の型，read-from/write-to はどの外部ポートから読み出す/書き込むか，width は CoE channel の幅 (byte) を表す．width は OpenCL コードを解析すれば type から導けるが，現時点の coe-gen にはその機能がないため，明示的に与えている．

4.3 通信と演算のパイプライン化

高性能計算のアプリケーションでは，通信時間を隠蔽するために，通信と演算をオーバーラップする最適化が用いられる．特に，ステンシル計算では一般的な最適化の一つとして用いられており，袖領域のデータを必要としない計算と袖領域の通信をオーバーラップし通信隠蔽を行う [12]．

例えば，オーバーラップを Message Passing Interface (MPI) を用いて実装する場合，MPI_Isend や MPI_Recv などの非同期通信を利用したり，通信用スレッドと演算用スレッドを分けてプログラミングしたりする手法が用いられる．これらの方法は，どちらも明示的にアプリケーションプログラマが通信隠蔽用のコードを書く必要があり，実装コストがかかる．

一方で，FPGA と OpenCL を用いる場合，3.2 節にあるように，演算はパイプラインに変換され FPGA に実装される．また，CoE は channel 機構に基づく通信を提供するものであり，通信に関する処理もパイプラインが構築される．したがって，両者を組み合わせると，通信と演算の両方を融合したパイプラインを構築できる．言い換えると，クロックサイクルレベルで細粒度に通信と演算がオーバーラップされているといえる．この特徴は他のアクセラレータにはなく FPGA 特有のものであり，我々は FPGA 上で演算と通信を融合させることで，HPC アプリケーションのさらなる演算加速が可能であると考えている．

4.4 制限事項

CoE は開発中のシステムであり，いくつかの制限事項が残っている．

- 直結網を想定しているが，ルーター機能はまだないため，隣接ノードにしかデータが送れない．
- フロー制御や再送制御の実装はなく，受信側のバッファが不足した場合はパケットが脱落する．

これらの制限事項は引き続き開発を行い解消してく予定である．物理層では SerialLite III プロトコルを用いているが，SerialLite III にはフロー制御の機能がないため CoE

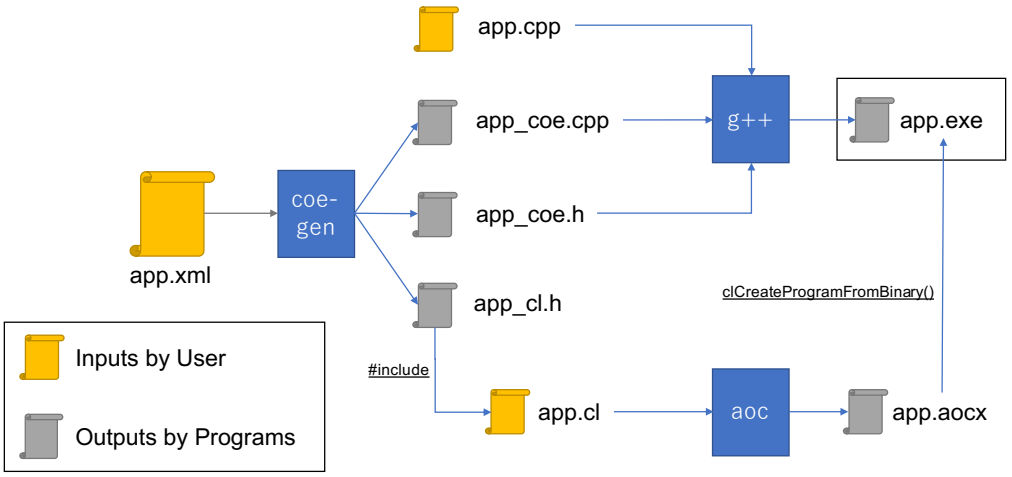

図 6: coe-gen を用いる際の開発フロー図.

```
<channel name="fwd_x_neg_in">
  <type>uint16</type>
  <read-from id="5">x_neg</read-from>
  <width>64</width>
</channel>
<channel name="fwd_x_neg_out">
  <type>uint16</type>
  <write-to id="6">x_neg</write-to>
  <width>64</width>
</channel>
<channel name="back_x_neg_in">
  <type>uint16</type>
  <read-from id="7">x_neg</read-from>
  <width>64</width>
</channel>
<channel name="back_x_neg_out">
  <type>uint16</type>
  <write-to id="8">x_neg</write-to>
  <width>64</width>
</channel>
```

図 7: coe-gen へ入力する XML ファイルの一部.

表 2: 評価環境

CPU	Intel Xeon E5-2690 v4 × 2
CPU Memory	DDR4 2400 MHz 64 GB (8 GB × 8)
Infiniband	Mellanox ConnectX-4 EDR
Host OS	CentOS 7.3
Host Compiler	gcc 4.8.5
OpenCL SDK	Intel FPGA SDK for OpenCL 19.1.0.240
FPGA	Nallatech 520N (1SG280HN2F43E2VG)
FPGA Memory	DDR4 2400 MHz 32 GB (8 GB × 4)
Communication Port	QSFP28 × 4 (up to 100 Gbps × 4)

レイヤーでもフロー制御が実装できていない．SerialLite III とは独立してフロー制御を実装するか，フロー制御の機能があるプロトコルへの変更を予定している．

5. 性能評価

5.1 評価環境

性能評価には Pre-PACS version X (PPX) クラスタシステムを用いる．PPX は筑波大学 計算科学研究センターで運用中のシステムであり，同センターが開発を計画している PACS シリーズ・スーパーコンピュータ次世代機 (システム名 Cygnus[13]) のプロトタイプシステムである．

PPX は実験用システムなため，様々な仕様のノードが混在しているが，本研究の性能評価は，PPX システムの Intel Stratix 10 FPGA ボード搭載ノードから最大 3 ノードを用いて行う．**表 2** に PPX システムの Stratix 10 FPGA ノードの仕様を示す．各ノードに Broadwell Xeon CPU ×2, InfiniBand EDR HCA×1, Nallatech FPGA ボード ×1 が搭載されている．

評価実験に用いる環境は CPU 向け汎用の InfiniBand ネットワークと FPGA 向け直結網の 2 種類の高速なネットワークを持つ．InfiniBand ネットワークは Mellanox 製のスイッチを用いてノード間が接続されており，FPGA 間ネットワークは**図 8** にあるように 2 × 2 のトーラスネットワークが構築されている．Nallatech 520N FPGA ボードは最大で 100Gbps の通信が行える QSFP28 ポートを 4 つ持ち，それらのポート間を光ケーブルで接続している．各 FPGA が 4 ポート持つため，トーラスではなく，4 FPGA をすべて直結する接続を取ることも出来るが，PPX は Cygnus システム (64 FPGA で 8 × 8 のトーラスを構成) の試験用として構築・運用しているため，Cygnus と同じネットワークトポロジーを用いる．

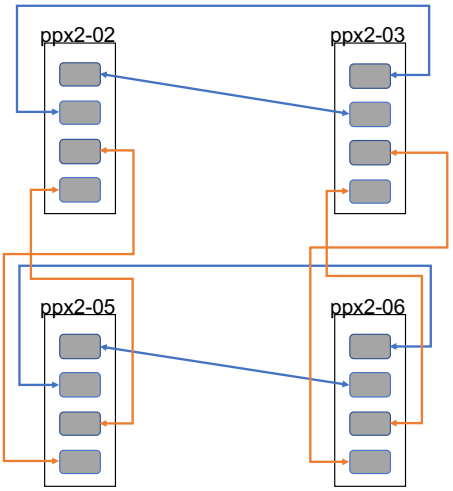

図 8: PPX の FPGA 間ネットワーク図．青色の線は横方向の接続，オレンジ色の線は縦方向の接続を表す．ppx2-02, 03, 05, 06 はそれぞれノード名である．

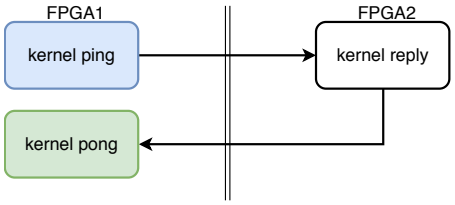

図 9: pingpong ベンチマークの構成．

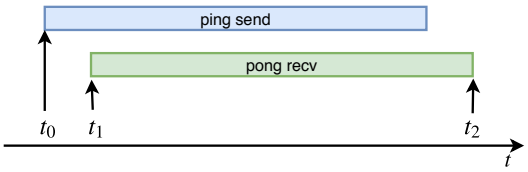

図 10: pingpong ベンチマークにおける時間計測．

5.2 pingpong ベンチマーク

CoE 通信機構の基礎性能を評価するために pingpong ベンチマークの評価を行う．pingpong ベンチマークは**図 9** の様に実装しており，2 つの FPGA 間でデータを交換し性能を測定する．この性能評価には ppx2-02, ppx2-03 の 2 ノードにある FPGA 間を用いて行った．直結された FPGA 間の性能を測定しているため，光通信にかかる時間と OpenCL 由来の通信オーバーヘッドが測定される．

性能の測定には，**図 10** にあるように，3 箇所を OpenCL カーネルの動作クロック精度で時間を測定し，そこから性能を求める．通信レイテンシおよびバンド幅は片道レイテンシに通信時間を足した $\frac{t_1 - t_0}{2} + (t_2 - t_1)$[s] を元に計算する．これらの時間は同一の FPGA（送信側）で測定されるため，クロックサイクルカウンターを用いて測定できる．OpenCL で FPGA プログラミングを行う場合，カーネルの動作周波数はコードによって変動するが，Stratix 10 を用いる場合一般的に 250MHz~350MHz の範囲で動作するため，2.85ns~4.0ns の精度で時間を測定できる．

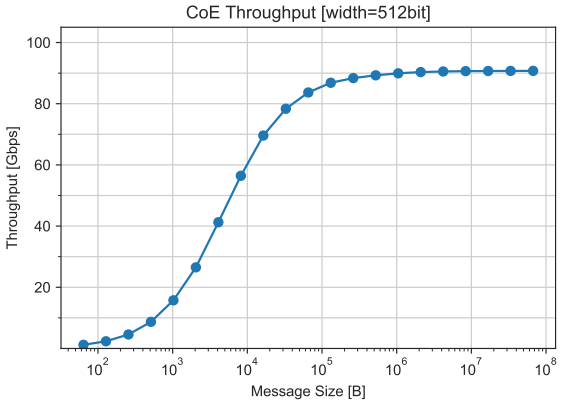

図 11: pingpong ベンチマークの測定結果．

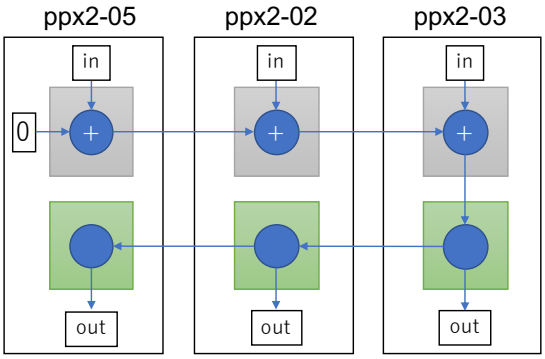

図 12: トイプログラムの構成．灰色の箱は往路の OpenCL カーネルを，緑色の箱は復路の OpenCL カーネルを表す．

pingpong ベンチマークの測定結果を**図 11** に示す．片道の最小レイテンシは 429.2ns，通信スループットは最大で 90.7Gbps の性能が得られた．この測定には，幅 512bit (uint16 型) の CoE Channel を用い，メッセージ長は 16byte (1 × uint16) から 64MB (1048576 × uint16) まで変化させ測定した．また，OpenCL カーネルの動作周波数は 360.0MHz であった．通信性能 (特に最小レイテンシ) は OpenCL カーネルの動作周波数に依存し，周波数が高い方が性能がよくなるため，全てのコードで pingpong ベンチマークと同じ性能が得られることを保証するものではない．

5.3 通信のパイプラインに関する評価

CoE では，通信と演算が一体化されたパイプラインが FPGA 内に構築される．したがって，通信と演算が自然とオーバーラップされ，通信時間が隠蔽される．この効果を測定するために，本稿では簡単なトイプログラムを作成し性能評価に用いる．

評価に用いたプログラムの構造を**図 12** に示す．このプログラムは MPI における MPI_Allreduce + MPI_SUM を模したものであり，すべての FPGA の入力データの和を取り，計算結果をすべてのノードに返す．OpenCL のカー

```
uint16 val = (uint16)(0);
if (in_port == 1) {
    val = read_channel_intel(fwd_x_neg_in);
} else if (in_port == 2) {
    val = read_channel_intel(fwd_x_pos_in);
} else if (in_port == 3) {
    val = read_channel_intel(fwd_y_neg_in);
} else if (in_port == 4) {
    val = read_channel_intel(fwd_y_pos_in);
}

val += (uint16)(
    v + 0, v + 1, v + 2, v + 3,
    v + 4, v + 5, v + 6, v + 7,
    v + 8, v + 9, v + 10, v + 11,
    v + 12, v + 13, v + 14, v + 15
);

ulong t_tmp = 0;
if (out_port == 1) {
    write_channel_intel(fwd_x_neg_out, clocktime(
        val, &t_tmp));
} else if (out_port == 2) {
    write_channel_intel(fwd_x_pos_out, clocktime(
        val, &t_tmp));
} else if (out_port == 3) {
    write_channel_intel(fwd_y_neg_out, clocktime(
        val, &t_tmp));
} else if (out_port == 4) {
    write_channel_intel(fwd_y_pos_out, clocktime(
        val, &t_tmp));
} else if (out_port == 5) {
    write_channel_intel(internal, clocktime(val, &
        t_tmp));
}
```

図 13: トイプログラムの OpenCL コードの一部.

ネルは 2 種類のカーネルで構成される．1 つは往路のデータ転送を行うカーネルであり，もう 1 つは復路のデータ転送を行うカーネルである．通信はバケツリレー方式で行われ，全体の計算が完了したら計算結果を全てのノードに返す．また，和の計算を行う処理は往路で行われ，復路でその計算結果をブロードキャストする．

図 13 にトイプログラムのコードの一部を示す．このコードは入力を受け取り，その結果に値を加算し，出力するというものであり，図 12 の灰色で示されているカーネルの一部である．read_channel_intel，write_channel_intel 関数はそれぞれ Channel から読み出し，書き出しを行う組み込み関数であり，clocktime 関数は時間を測定する独自の関数である．if 文で入出力する Channel を切り替えられるようになっているが，これは CoE にはルーティング機能がまだなく，FPGA ボードにあるどの外部リンクで通信を行うかを明示する必要があるためである．

性能評価の結果を**図 14** に示す．最小レイテンシは 2014ns，最大スループットは 181.4Gbps が得られた．本実

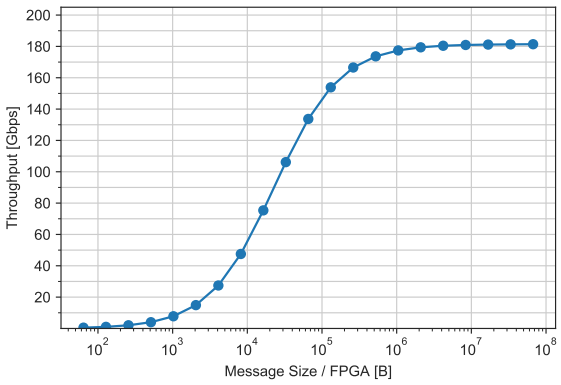

図 14: トイプログラムの測定結果.

表 3: プロトコルオーバヘッド

要素	ペイロード通信速度	効率
物理層速度	103.125Gbps	
67b/64b	98.484Gbps	×0.955
Meta Frame	98.287Gbps	×0.998
SL3 Burst	96.813Gbps	×0.985
CoE Header	90.762Gbps	×0.938

験には ppx2-02, ppx2-03, ppx2-05 の計 3 ノードを用いた．図 12 にあるように ppx2-05 が始点ノードとなり ppx2-03 で折り返す．測定結果のスループットは，始点ノードのデータ送信開始から始点ノードのデータ受信終了までの時間から求めた．また，横軸のデータサイズは，各ノードが持っているデータサイズを表しており，MPI_Allreduce における count 引数に相当する．pinpong ベンチマークの結果 90.7Gbps と比べて，181.4Gbps と約 2 倍の性能が得られているが，これは通信と演算がパイプライン化によって送信と受信が同時に行われるためである．

6. 考察

6.1 pingpong ベンチマーク

pingpong ベンチマークで得られた最大スループットは 90.7Gbps であり，物理層に 100Gbps を用いているのに対して約 90%の性能しか得られていない．しかしながら，この性能は設計の意図したとおりである．**表 3** に理論上の通信性能を示す．評価環境では物理層の速度は 103.125Gbps (4×25.78125Gbps) であり，この速度は 100Gb Ethernet の物理層と同じ速度を採用している．表 3 は，その物理層の速度に対して，プロトコル上のオーバヘッドがどの程度あるのかを示したものである．この中で，67b/64b, Meta Frame，SL3 Burst は SerialLite III に由来するオーバーヘッドであり，公式ドキュメント [11] に記載されている計算式を用いて求めた．CoE Header は CoE が付与するヘッダによるオーバーヘッドを示すものである．CoE のパケットは 64byte で構成されており，そこに 4byte のヘッ

ダと 60byte のペイロードを含む．

前述したオーバーヘッドの要素の中で，最も大きいものは CoE のパケット長が短いことに起因するものである．これは，CoE の通信フォーマットの設計は，低レイテンシに焦点を当てて最適化しているためである．アプリケーションからデータが渡されてから，実際に送信されるまでのバッファリング時間を短くすることができるため，通信レイテンシが短くなる．この特性はパイプラインに基づく通信を行う CoE システムでは特に重要となる．加えて，バッファリングに FPGA の回路リソースを使わなくて済むため回路効率が良い．また，次世代の FPGA では，Pulse Amplitude Modulation (PAM) 4 を用いた 200Gbps やそれ以上の高速通信が可能になると期待でき，より高性能な通信機構を用いることでこの問題は緩和できると考えている．

OpenCL 部で通信処理にかかる時間と，BSP に内の SerialLite III の通信処理にかかる時間を詳細に測定することは，CoE BSP にルーター機能を追加した際の性能予測などに重要である．しかしながら，レイテンシのブレイクダウンについてはまだ測定できておらず，今後の課題である．

6.2 通信のパイプラインに関する評価

通信のパイプラインに関する評価で最小レイテンシは 2014ns, 最大スループットは 181.4Gbps が得られた．まず，レイテンシについてであるが，このプログラムは図 12 にあるように外部通信を 4 回行う．これは, pingpong ベンチマークのレイテンシが 429.2ns であるため $2014 \div 429.2 = 4.7$ 回の通信に相当する．やや通信時間が長くかかっているが，整数加算とはいえ演算にもレイテンシがあることと，トイプログラムの動作周波数が 280.0MHz と，pingpong ベンチマークの 360.0MHz と比べて低いためであると考えている．

次にスループットに関する考察であるが，pinpong ベンチマークの結果 90.7Gbps と比べて，181.4Gbps と 2 倍の性能が得られており，想定通りといえる．また，通信リンク 2 本分の速度が得られていることから，送信と受信が同時に行われているとわかる．したがって，この結果は，通信と演算が一体となったパイプラインが構築され，正しくデータが流れていることを示すものであるといえる．

7. おわりに

本研究では，Intel OpenCL 開発環境が FPGA 向け拡張として実装している Channel API を，異なる FPGA 間に拡張するコンセプトである CoE 環境を開発し，性能評価を行なった．システムの性能を評価するために pingpong ベンチマークと MPI_Allreduce を模したトイプログラムに CoE を適用した．

CoE 通信のバンド幅は最大で 90.7Gbps を達成し，CoE を用いた 16 バイト通信時の最小レイテンシは 429.2ns であった．また，トイプログラムでは，181.4Gbps と pinpong ベンチマークの 2 倍の性能が得られており，通信リンク 2 本分の速度が得られていることから，通信と演算が一体となったパイプラインが構築され，送信と受信が同時に行われている結果が得られた．しかしながら，さらなる最適化の必要や，フロー制御やパケットルーティングなど，通信機構として実用するのに必要な機能がまだ不足しており，今後開発を進めていく．

CoE においては計算と通信が一体のパイプラインを形成しているため，計算をしながら通信を行うという挙動が自然に記述でき，通信隠蔽を行いやすい．したがって，CoE の通信モデルは FPGA で並列計算を行う際に適していると考えている．我々は宇宙物理学のアプリケーションの FPGA 向け最適化も行なっており [14]，今後，CoE の通信システムをそのアプリケーションに適用し，複数 FPGA を用いた並列計算を行う予定である．

筑波大学 計算科学研究センターでは，CPU + GPU + FPGA 混在ノードを持つマルチヘテロジニアスなスーパーコンピューター Cygnus を 2019 年 5 月より運用している．Cygnus システムには 64 枚の FPGA が搭載されており，それらが 8×8 のトーラスネットワークで接続されている．CoE は今後 Cygnus 上で利用することを目標として開発を続けていく．

謝辞 本研究の一部は，「高性能汎用計算機高度利用事業」における課題「次世代演算通信融合型スーパーコンピュータの開発」及び，文部科学省研究予算「次世代計算技術開拓による学際計算科学連携拠点の創出」による．また，本研究の一部は，「Intel University Program」を通じてハードウェアおよびソフトウェアの提供を受けており，Intel の支援に謝意を表する．

参考文献

[1] NVIDIA Corporation: GPUDirect for RDMA, (online), available from ⟨https://docs.nvidia.com/cuda/gpudirect-rdma/index.html⟩.

[2] 藤田典久, 小林諒平, 山口佳樹, 朴 泰祐: OpenCL による FPGA 上の演算と通信を融合した並列処理システムの実装及び性能評価, 情報処理学会研究報告, 2018-HPC-167 (2018).

[3] Zohouri, H. R., Maruyama, N., Smith, A., Matsuda, M. and Matsuoka, S.: Evaluating and Optimizing OpenCL Kernels for High Performance Computing with FPGAs, *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, SC '16, Piscataway, NJ, USA, IEEE Press, pp. 35:1–35:12 (online), available from ⟨http://dl.acm.org/citation.cfm?id=3014904.3014951⟩ (2016).

[4] Kenter, T., Mahale, G., Alhaddad, S., Grynko, Y., Schmitt, C., Afzal, A., Hannig, F., Forstner, J. and Plessl, C.: OpenCL-Based FPGA Design to Accelerate the Nodal Discontinuous Galerkin Method for Unstruc-

tured Meshes, *2018 IEEE 26th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM)*, Vol. 00, pp. 189–196 (online), DOI: 10.1109/FCCM.2018.00037 (2018).

[5] 大島聡史，塙　敏博，片桐孝洋，中島研吾：FPGA を用いた疎行列数値計算の性能評価，情報処理学会研究報告，2016-HPC-153 (2016).

[6] 塙　敏博，伊田明弘，大島聡史，河合直聡：FPGA を用いた階層型行列ベクトル積，情報処理学会研究報告，2016-HPC-155 (2016).

[7] Putnam, A., Caulfield, A., Chung, E., Chiou, D., Constantinides, K., Demme, J., Esmaeilzadeh, H., Fowers, J., Gray, J., Haselman, M., Hauck, S., Heil, S., Hormati, A., Kim, J.-Y., Lanka, S., Peterson, E., Smith, A., Thong, J., Xiao, P. Y., Burger, D., Larus, J., Gopal, G. P. and Pope, S.: A Reconfigurable Fabric for Accelerating Large-Scale Datacenter Services, *Proceeding of the 41st Annual International Symposium on Computer Architecuture (ISCA)*, IEEE Press, pp. 13–24 (online), available from ⟨https://www.microsoft.com/en-us/research/publication/a-reconfigurable-fabric-for-accelerating-large-scale-datacenter-services/⟩ (2014).

[8] Baxter, R., Booth, S., Bull, M., Cawood, G., Perry, J., Parsons, M., Simpson, A., Trew, A., McCormick, A., Smart, G., Smart, R., Cantle, A., Chamberlain, R. and Genest, G.: Maxwell - a 64 FPGA Supercomputer, *Second NASA/ESA Conference on Adaptive Hardware and Systems (AHS 2007)*, pp. 287–294 (online), DOI: 10.1109/AHS.2007.71 (2007).

[9] 大畠佑真，小林諒平，藤田典久，山口佳樹，朴　泰祐：OpenCL と Verilog HDL の混合記述による FPGA 間 Ethernet 接続, 情報処理学会研究報告, 2017-HPC-160 (2017).

[10] Kobayashi, R., Oobata, Y., Fujita, N., Yamaguchi, Y. and Boku, T.: OpenCL-ready High Speed FPGA Network for Reconfigurable High Performance Computing, *Proceedings of the International Conference on High Performance Computing in Asia-Pacific Region*, HPC Asia 2018, New York, NY, USA, ACM, pp. 192–201 (online), DOI: 10.1145/3149457.3149479 (2018).

[11] Intel Corporation: SerialLite III ストリーミング・プロトコル，（オンライン），入手先 ⟨https://www.intel.co.jp/content/www/jp/ja/programmable/products/intellectual-property/ip/interface-protocols/m-alt-seriallite3.html⟩.

[12] Idomura, Y., Nakata, M., Yamada, S., Machida, M., Imamura, T., Watanabe, T., Nunami, M., Inoue, H., Tsutsumi, S., Miyoshi, I. and Shida, N.: Communication-overlap techniques for improved strong scaling of gyrokinetic Eulerian code beyond 100k cores on the K-computer, *The International Journal of High Performance Computing Applications*, Vol. 28, No. 1, pp. 73–86 (online), DOI: 10.1177/1094342013490973 (2014).

[13] 筑波大学 計算科学研究センター：スーパーコンピュータ – 筑波大学計算科学研究センター　Center for Computational Sciences)，（オンライン），入手先 ⟨https://www.ccs.tsukuba.ac.jp/supercomputer/#Cygnus⟩.

[14] 藤田典久，小林諒平，山口佳樹，朴　泰祐，吉川耕司，安部牧人，梅村雅之：並列 FPGA システムにおける OpenCL を用いた宇宙輻射輸送コードの演算加速，情報処理学会研究報告, 2018-HPC-165 (2018).