Vol.2021-HPC-178 No.6
2021/3/15

HBM2メモリを持つFPGAボードの性能評価

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

概要：近年，高位合成 (High Level Synthesis: HLS) と呼ばれる技術が発展してきており，Field Programmable Gate Array (FPGA) 開発の障壁が低下しつつある．しかしながら，FPGA の持つメモリ帯域は他のアクセラレータと比べて低く，HPC 分野で FPGA を利用する際の障壁となることがあった．しかし，High Bandwidth Memory 2 (HBM2) を搭載した FPGA チップがベンダーからリリースされ始めており，最大で 512GB/s のメモリ帯域を有する．依然として，Graphics Processing Unit (GPU) のアクセラレータと比べると，1/4 倍性能の開きがあるものの，性能が一桁以上違うという状況からは改善しつつある．本稿では，Intel Stratix10 FPGA に搭載された HBM2 メモリの性能評価および HPC アプリケーションに適用する手法について述べる．

1. はじめに

近年，高性能計算 (HPC) の分野で Field Programmable Gate Array (FPGA) が注目されている．これまで，FPGA の回路を記述するためには，Verilog HDL を始めとしたハードウェア記述言語 (Hardware Description Language: HDL) を用いるのが一般的であった．HDL はソフトウェアにおけるアセンブラに相当するものであり，記述する際にハードウェアの構造や動作に関する知識が求められていた．また，FPGA の大規模化に伴い，HDL ですべての回路を設計することが難しい状況になりつつある．

近年，高位合成 (High Level Synthesis: HLS) と呼ばれる技術が発展してきており，開発の障壁が低下しつつある．HLS はソフトウェアで使われている言語（C，C++，OpenCL など）を用いて，ハードウェアの動作を記述できるものである．HDL がクロック・ビット単位の粒度で動作を記述しなければならないのに対して，HLS は回路の動作を HDL より高位のレベルで記述できる．

また，近年の FPGA は最大 100Gbps の外部通信なポートを扱える能力を有しており，高速な FPGA 間通信が実現できる．筑波大学計算科学研究センターではスーパーコンピューター Cygnus を運用している．Cygnus には 64 枚の FPGA が搭載され，8x8 の 2D トーラスネットワークが構築されている．これまでの研究で，我々は複数の FPGA を用いた並列計算を実現してきた [1], [2].

これらの研究で，FPGA の外部通信能力は高く，並列計算に有用であることは明らかとなった．しかしながら，FPGA の持つメモリ帯域は他のアクセラレータと比べて低く，FPGA ボードに搭載されていたメモリは DDR4 メモリであり 76.8GB/s の帯域しかない．最先端の Graphics Processing Unit (GPU) である NVIDIA A100 80GB[3] が 2TB/s のメモリ帯域を持つことと比較すると，FPGA が有するメモリ帯域は非常に弱いと言わざるを得ず，HPC 分野で FPGA を利用する際の障壁となることがあった．しかし，High Bandwidth Memory 2 (HBM2) を搭載した FPGA チップがベンダーからリリースされ始めており，最大で 512GB/s のメモリ帯域を有する．A100 と比較すると 1/4 のメモリ帯域でしかないものの，10 倍やそれ以上の帯域差がある状況よりは改善している．

本研究の目的は，HBM2 を搭載した FPGA ボードの性能評価を行うことである．HBM2 を持つ FPGA ボードとして，インテル Stratix 10 MX FPGA 開発キットを用いる．このボードは Intel Stratix 10 MX FPGA (SKU: 1SM21CHU2F53E1VG) を搭載しており，16GB の HBM2 を FPGA パッケージ内に含む．

本稿の貢献は以下の通りである．

- HBM2 持つを FPGA の性能特性を明らかにする．
- HPC でよく用いられるブロックストライドのメモリアクセス性能の評価を行い，アプリケーションで利用する際の性能指標を示す．
- HPC アプリケーションを HBM2 FPGA 環境に移植し，高い性能を達成する方法について考察する．

2. 関連研究

FPGA に搭載されている HBM2 を活用する研究として

[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

は，従来型のメモリ帯域が必要なアプリケーション [4] に加えて，ニューラルネットワークに適用した研究 [5], [6] が知られている．また，本研究会においても [7] の報告がある．これらの研究では，演算回路が特定の HBM2 メモリ Channel に接続されており，それ以外の channel にアクセスすることはできない．

[8] で Choi らは Xilinx 製 FPGA ボードである Alveo U280 を用いて，Bucket Sort と Merge Sort を実装し，性能評価を行った．Vivado HLS で実装した Sorter 部分と彼らが HBM Connect と呼んでいるメモリネットワークが実装されている．HBM Connect によって Sorter 回路とメモリ間が接続されており，Sorter 回路が全ての HBM2 Channel にアクセスできる．ボード開発環境の制限により，ボードにある HBM2 の半分でしか性能評価ができていないものの，理論ピークの 9 割程度の性能が達成されている．性能の代償として，HBM Connect を構成するために回路リソースを消費しており，その分アプリケーションのために使える回路リソースが減っている．

Xilinx FPGA 向けに Shuhai[9] というマイクロベンチマークが開発されているが，我々が研究対象としている Intel FPGA 向けに開発されたベンチマークは知られていない．本稿では，Intel FPGA を対象にして HBM2 の性能評価を行う．また，将来に HPC アプリケーションを FPGA 向けに最適化することを前提とし，HPC アプリケーションでよく用いられるメモリアクセスパターンで性能評価を行う．

3. HBM2 のアーキテクチャ

Intel は Stratix 10 FPGA の世代から Multi Chip Module (MCM) 構造を基本に FPGA パッケージを構成している．FPGA の再構成可能なロジックを含むダイを中心として，特定の機能に特化したダイを接続することで，様々なバリエーションの製品を構築する．ダイ間は Embedded Multi-Die Interconnect Bridge (EMIB) [10] と呼ばれる技術で接続される．そして，FPGA 内に外部ダイで実装されている機能を利用する Intellectual Property (IP) を実装すると，FPGA 内部回路から利用できる．Intel Stratix 10 FPGA では，EMIB を用いた MCM 実装によって HBM2 が接続されており，

図 1 に Stratix 10 MX FPGA における HBM2 のアーキテクチャを示す．最大で 2 つ (Top, Bottom) のメモリダイ[*1]が接続され，各ダイで 4GB or 8GB のサイズを持つ．各ダイは最大で 256GB/s[*2]の帯域を持ち，2 ダイをまとめた Aggregated Bandwidth では最大で 512GB/s に達する．

また，FPGA 固有の制限として，最大の性能を得るためには，メモリコントローラを駆動してデータを供給する FPGA 側のロジックも，HBM2 の持つ広い帯域を使い切れる性能がなければならない．

HBM2 では，（比較的）遅いメモリを多数並列に駆動することで，低消費電力・広帯域なメモリが実現されている．図 1 にある様に，ダイあたり 8 つの物理メモリチャンネル (32GB/s/CH) があり，さらに各物理チャンネルが 2 つの Pseudo-channel (16GB/s/pCH) に分割されている構造を持つ．Pseudo-channel の物理的接続は 64bit 幅であるが，FPGA の動作周波数レンジでは 1:1 で駆動することが難しいため，256bit バス幅で動作周波数を 1/4 という形で実装されている．ただし，HBM2 の仕様として，4 要素のバーストアクセス (Burst-Length 4: BL4) が必須となっているため，バス幅を拡張しても，メモリに対するアクセス粒度は変わらない．同じメモリチャンネルに属する Pseudo-channel は，それぞれ独立のデータバスを持つが，コマンドバスは 2 つで共有する．したがって，同時に片方の Pseudo-channel にしか命令を発行できないが，データアクセス（読み書き）は同時に行える．交互にコマンドを発行することで，メモリアクセスに関するレイテンシを隠蔽できる．

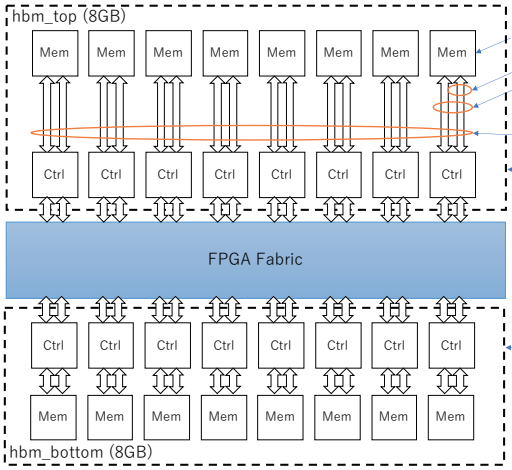
ここで，データの観点で見ると，全ての Pseudo-channel は独立しており，それぞれが独立したメモリのように見える．すなわち，ある Pseudo-channel から書き込んだデータは，同じ Pseudo-channel からしかアクセスできず，他の Pseudo-channel を用いてアクセスできない．一般的なプロセッサ環境では，データがどのメモリに書き込まれたか意識する必要はなく，システムにある全てのメモリにアクセスできる．これが可能なのは，プロセッサ内にアドレスを基準としたメモリアクセスネットワークが存在するからである．FPGA 環境で同様の挙動を実現しようとすると，同様の機能をもったネットワークを FPGA 内に実装しなければならない．

4. ベンチマークプログラム

4.1 ベンチマークの構造

図 2 にベンチマークの全体構造を示す．Pseudo-channel 毎に Tester モジュールを接続しメモリアクセスを行う．また，全体の制御コントローラーとして NIOS II プロセッサを配置する．NIOS II は Intel が提供している独自のプロセッサで，FPGA 開発環境に統合されており，容易に FPGA に組み込め，NIOS II に対応する gcc コンパイラが提供されている．プロセッサの演算性能は低いものの制御用としては十分であり，FPGA 開発環境との親和性を重視して NIOS II を選択した．NIOS II プロセッサは，Joint Test Action Group (JTAG) ケーブル経由で Universal Asynchronous Receiver/Transmitter (UART) 通信を

[*1] ダイの中で DRAM が積層されているため，Stack とも呼ばれる．
[*2] 最大帯域は SKU (Speed Grade のランク) に依存する．256GB/s/die を得るには-1 グレードが必要．-2 では最大 204.8GB/s/die，-3 では 153.6GB/s/die となる．



Total	512GB/s	
ダイあたり	256GB/s	x2
Channelあたり	32GB/s	x8
Pseudo-channelあたり	16GB/s	x16

図 1: Intel FPGA における HBM2 の構造.

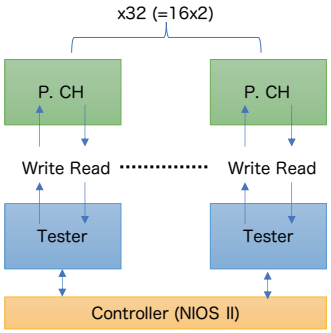


図 2: ベンチマークの全体構造.

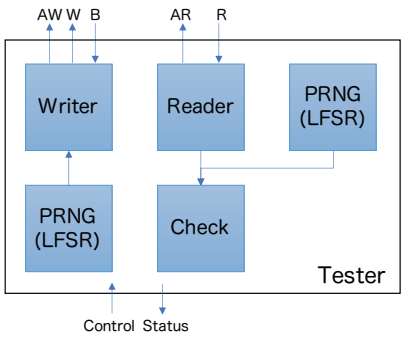


図 3: Tester モジュールの構造. AW, W, B, AR, R はそれぞれ AXI-4 バスにおける信号名.

ホストと行える. NIOS II 用の Terminal ソフトを用いてホスト– NIOS II 間の通信ができ，例えば printf 関数や scanf 関数を実行すると，その入出力はホスト側と繋がる.

Tester モジュールの構造を図 **3** に示す. ここで，AW, W, B, AR, R はそれぞれ AXI-4 における信号名に由来しており，Write Address, Write Data, Write Response, Read Address, Read Response Channel を意味する. Intel FPGA でメモリ関係の IP を扱う場合は，Avalon Memory Mapped (Avalon-MM) バスを使うことが一般的であるが，Intel より提供されている HBM2 Controller IP は AXI-4 バスを使う実装となっているため，Tester モジュールも AXI-4 バスを扱うように実装する.

Tester モジュールは Linear Feedback ShiftRegister (LFSR) で生成される擬似乱数 Psuedo-Random Generator (PRNG) 列をメモリに書き込む Writer と，メモリからデータを読み出し PRNG 列と比較する Reader モジュールから構成される. 書き込みと読み込みをペアで実行すると，メモリアクセスが正しいかどうかが検証できる. また，図では省略しているが，書き込みと読み込みにそれぞれ何クロックかかったかを測定するタイマーが含まれており，タイマーで測定した時間を元に転送帯域を計算できる. Write，Read どちらも Request を発行してからそれに対する Response が帰ってくるまでを処理時間と定義する. ただし，Tester モジュールの動作開始と完了の間で時間測定を行っているため，Tester モジュールの動作遅延が含まれている. コントローラの内部構造が不明なため，Write Resonse の応答が来た時点でどこまでデータが書き込まれているのかわからないが，Write Response が発行されるタイミングは，Write したデータを Read できる時点であるとされている.

これらのベンチマーク用モジュールは，Chisel[11] で記述されている. Chisel は scala 上に構築されたドメイン特化言語 (Domain Specific Language: DSL) であり，中間表現である Firrtl[12] を経由して，Verilog HDL が生成される. Chisel 上の記述はほぼ 1 : 1 で Verilog HDL に変換される. したがって，Chisel の表現能力は Verilog HDL と同等であり，サイクルレベルの粒度で回路動作が記述できる. 我々のこれまでの研究 [1], [2] では，OpenCL を用いた高位合成で回路を記述していた. しかしながら，OpenCL を用いてしまうと性能予測が難しく，メモリの生のパフォーマンスを測定することが困難であるため，本稿では Chisel による低レベル記述を行う. OpenCL 等高位合成環境における HBM2 の性能評価は今後の課題である.

4.2 メモリコントローラの設定

HBM2 からデータを読み出す際，メモリコントローラは性能を最適化するためにリクエストの順番を並び替えて実行する．したがって，コントローラーから出力されるデータの順番はリクエストの順番と一致しない．この仕様では FPGA 内でデータを扱う回路が煩雑になってしまうため，リクエストの順番とレスポンスの順番を一致されるバッファ (Reorder Buffer: ROB) がコントローラ内に実装されている．そして，コントローラ内の ROB を有効にするか無効にするかはオプションで切り替えられる．

Tester は読みだしたデータが正しいかどうか PRNG 列と比較を行うため，リクエスト順序通りにデータが得られないと不都合がある．したがって，本稿の範囲では ROB 有効で HBM2 コントローラを利用している．ただし，ROB をデータを一度バッファリングして並び替える処理が入るため，リードのレイテンシが増加する．例えば，メモリ⇔メモリ間転送のように，バルク転送を行うケースであれば，データの順序は一致してなくても問題ないため，ROB を無効化することで性能向上が期待できる．

3 章で述べたように，HBM2 の仕様はバースト長＝ 4 となっており，64bit×4 の範囲に連続してアクセスを行う．したがって，メモリに対するアクセス粒度は 256bit となる．HBM2 コントローラ IP の設定に “Enable Pseudo BL8 for performance” という項目がある．ドキュメント [13] によると，Pseudo BL8 モードを有効にすると，バースト長＝ 8 となり，アクセス粒度が 512bit に拡大される替わりに，効率が改善されると書かれている．本稿では，性能を重視しているため，この機能を有効にしてコントローラを実装している．512bit というデータ粒度は，CPU で実装されている Single-Instruction Multiple-Data (SIMD) 命令のデータ幅でよく使われているサイズであり，HPC アプリケーションとの親和性も問題ないと考えられる．

4.3 動作周波数

2 章で述べた通り，HBM2 の性能を最大限発揮するには，FPGA 側の制御回路（今回は Tester）は 500MHz で動作することが求められる．しかしながら，最適化が十分できておらず，全 32 Channel を駆動する回路を構築した際に 500MHz 動作を達成できなかった．

core clock（FPGA 側インターフェイスの動作周波数）の 500MHz は動作はサポートされているものの，十分な最適化を行わないと達成できないものと思われる．HBM2 IP ライブラリ上で，core clock を 500MHz に設定すると，“Core timing closure may be difficult at the selected frequency. The maximum recommended frequency for this configuration and speedgrade is 405MHz.” という警告文が表示されるからである．本稿では，IP ライブラリの警告文と実際に回路合成をした際にえられる動作周波数から，memory clock を 1000MHz，Core clock を 400MHz 動作をターゲットとすることとした．この場合，メモリコントローラはフルスピードで動作しているが，core clock は 8 割の速度で動作している．したがって，メモリコントローラへデータを供給する部分に帯域不足が生じるため，シーケンシャルアクセスの帯域が低下する．一方で，ランダムアクセスの性能については，HBM2 側の性能に律速されるため，影響は少ないと考えられる．

表 1: 評価環境 (PPX)

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 19.4.0.64
FPGA	インテル Stratix 10 MX FPGA 開発キット (1SM21CHU2F53E1VG)
FPGA Memory	HBM2 16GB (8GB × 2)
Quartus Prime	Quartus Prime Pro 19.4.0.64

5. 性能測定

5.1 測定環境

本稿では，筑波大学計算科学研究センターで運用中の実験クラスタ Pre-PACS-X (PPX) を性能評価に用いる．PPX の 1 ノードにインテル Stratix 10 MX FPGA 開発キットを搭載し，性能評価を行う．開発キットは，1SM21CHU2F53E1VG を搭載しており，Speedgrade-1 のロジック部と，16GB の HBM2 を搭載した MCM となっている．

このボードは開発キットとして設計されているが，PCIe エッジコネクタを有しており，ホストサーバーの PCIe スロットに接続している．ただし，本実験では PCIe Controller IP は FPGA 内に実装されておらず，電力供給および固定用としてのみ用いられる．FPGA 内のベンチマーク回路の制御は，4.1 節で述べたように JTAG ケーブル（USB 接続）経由で行われる．また，ボードには DDR4 メモリスロットが 1 スロット実装されているが，こちらも利用しておらず，メモリアクセスは HBM2 に関するもののみを扱う．

5.2 同時アクセスする Channel 数を変化させる測定

図 4 に同時アクセスする Channel 数を増やした時の性能評価の結果を示す．転送サイズを Channel あたり 1MB に固定し，同時アクセスする Channel 数を変化させる．また，アクセスパターンはシーケンシャルアクセスである．

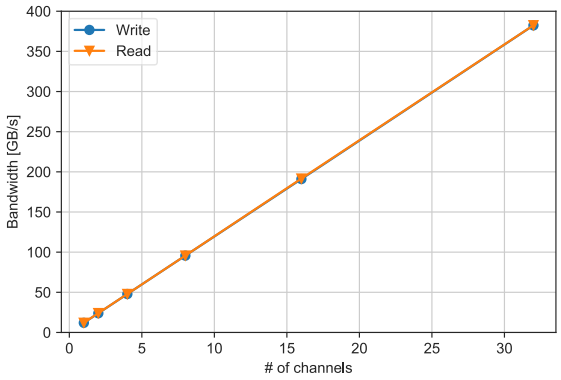


図 4: 同時アクセスする Channel 数を増やした時の性能変化.

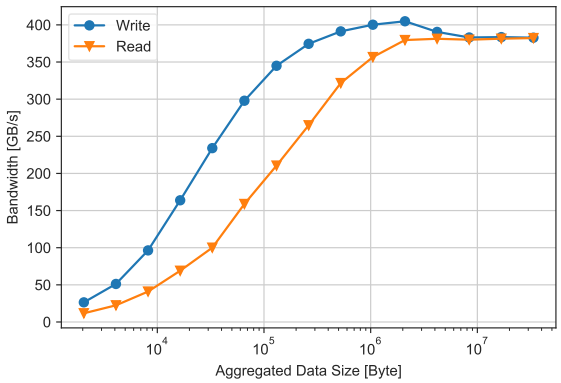


図 5: シーケンシャルアクセスの性能測定結果.

1 Channel の時, Write = 11.97GB/s, Read = 12.04GB/s の結果が得られ，また，32 Channel の時，Write = 382.36GB/s, Read = 382.53GB/s の結果が得られていることがわかる．1 Channel の性能と 32 Channel の性能の比は，それぞれ 31.93 倍，31.77 倍である．ほぼ線形に性能が増加していることから，HBM2 が持つ全ての Channel が独立して動作できていることがわかる．

5.3 シーケンシャルアクセスの測定

シーケンシャルアクセスの性能測定結果を**図 5** に示す．この実験では，全 32 Channel に対して同時にメモリアクセスを行う．また図 5 の横軸は 1 Channel 毎の転送量ではなく，32 Channel トータルの転送量を示す．転送量は Channel あたり 64Byte ～ 1MB までの測定を行う．最小値が 64Byte なのは，Channel あたりのメモリアクセス粒度が 64Byte なためである．図 5 の結果より，シーケンシャルアクセスの場合は，書き込み時 256KB (= 4KB/ch)，読み込み時 2MB (= 64KB/ch) の転送サイズがあれば十分な性能が得られるということがわかる．

最大で，シーケンシャルアクセス時に Write で 404GB/s, Read で 382GB/s の性能を達成した．400MHz 動作時のメモリバンド幅の理論ピークは 409.6GB/s であり，これに近い性能が得られているとわかる．

転送サイズが小さい時，Write に対して Read の性能が低いが，これは根本的に Read の方がレイテンシが長いことと，ROB の影響があるものと考えられる．また，Write の転送性能が 64KB の点にピークがあり，それを増えると若干低下する傾向が見られる．したがって，HBM2 コントローラの中に小さいサイズのバッファがあり，Write 時はそこに書き込んだ時点で完了と見做されていると考えられる．

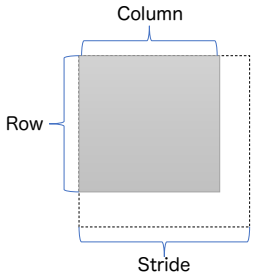


図 6: ブロックストライドアクセスの範囲.

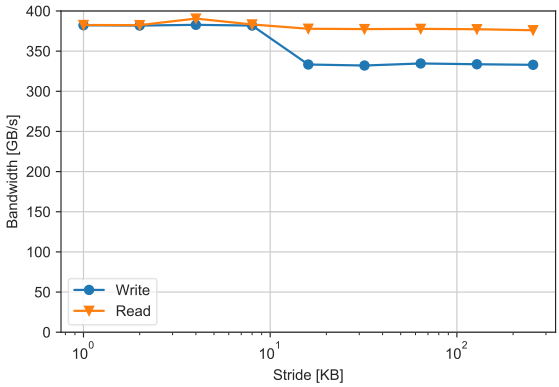


図 7: ストライド幅を変えた時の性能評価.

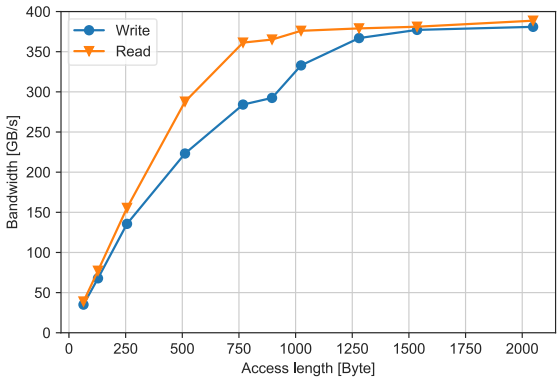


図 8: 連続アクセス量を変えた時の性能評価.

表 2: 動作周波数を変化させた時の性能比較.

Fmax	Write [GB/s]	Read [GB/s]
400MHz	382.36	382.53
500MHz	465.85	462.54

5.4 ブロックストライドアクセスの測定

図 6 にブロックストライドアクセスの範囲を示す．ブロックストライドアクセスには列，行，ストライド幅のパラメータがあり，ある行の中では連続メモリアクセスとなるが，行の間はストライド幅分の間隔が空いているメモリアクセスパターンを指す．ブロックストライドアクセスの列を 1024Byte，行を 1024 に固定して，ストライド幅を変えた時の性能評価を**図 7** に示す．図 7 より，列の長さが十分あれば，ブロックストライドアクセスをしても性能が低下しないことがわかる．ただし，16KB 以上のストライド幅でアクセスした場合，Write の性能が低下しているが，これは**図 8** でわかるように列の長さを大きくすると改善することがわかる．

図 8 にブロックストライドアクセスの列長を変えた場合の性能評価の結果を示す．こちらの結果は，アクセスの行を 1024 にストライド幅を 256KB に固定し，列の長さを変えたものである．図 8 より，Read の場合では 768Byte あれば十分な性能が得られていることがわかる．一方で，Write の場合，Read よりも長い列長のアクセスをしなければ性能が低下する．1024Byte でもやや不十分で (85%程度)，1280Byte 以上の列長が必要ということがわかる．

5.5 Overclock 状態での測定

FPGA の回路の最大動作周波数 (fmax) は，コンパイルされた結果（回路）のデータから，専用のツールで求められる．fmax の計算は，温度や個体差などの状況が最も悪い状態を想定して計算される．したがって，fmax を上回る周波数で動作させても問題なく動作する場合がある．これは，一般的なプロセッサにおける Overclock と等価な状態であり，他のボード個体や温度状況が変化すると正しく動作する保証はない．

4.3 節で述べたように，本来であればベンチマーク回路は 500MHz で動作させたいのであるが，現時点の実装で得られる fmax は 400〜450MHz である．本節では，Core clock の動作周波数の不足が性能にどの程度影響を与えているのかを明らかにするために，fmax が 445.24MHz の回路を Overclock で 500MHz 動作させた際の結果を示す．Overclock 状態では，前述した通り動作保証はないのであるが，書き込んだデータが正しく読めていることはベンチマーク回路で検証している．

測定結果を**表 2** に示す．500MHz の時の性能を 1 とすると，400MHz の時の性能は Read，Write それぞれ 0.799，0.803 だとわかる．この比率は動作周波数の比 0.8 とほぼ一致しており，動作周波数が低下した分だけ性能も低下している．動作周波数の性能に対する影響が大きいことがわかる．

6. 考察

本章では，HPC アプリケーションを HBM2 メモリを用いて実装する際に，どのように実装すると高い性能が得られるかについて考察を行う．まず，シーケンシャルアクセスが主になるアプリケーション（例えば，STREAM ベンチマーク）では，素直にメモリアクセスを行えば良いと考えられる．シーケンシャルアクセスでは，アクセス長がそのまま性能に直結するからである．ただし，図 4 からわかるように，HBM2 は全ての Channel に対してアクセスを同時に行わなければ性能が出ない．STEAM の様に巨大な配列に対してアクセスする場合は，Channel 毎に 1 次元で領域分割を行えば良いと明らかであるが，複数の配列を用いる実際のアプリケーションではデータをどのようにメモリ Channel に割り当てるかが課題となる．

次に，ブロックストライドアクセスを行う場合を考える．ある程度の大きさを持ったブロック単位でアクセスしなければ十分な性能が得られないことがわかった．Read の場合，768Byte あれば十分な性能が得られたが，Write の場合，1024Byte でもやや不十分で (85%程度)，1280Byte 以上の列長が求められる．一般的なプログラムでは，Write よりも Read の方が割合が多い事を考慮すると（例えば，行列積では 3 : 1），Read の性能が十分得られる 768 バイト以上の連続アクセスを行うようにアプリケーションを設計するのが良いと考えられる．ブロックストライドアクセスの場合でも同様に，全てのメモリ Channel に対してアクセスしなければならない制約は残っており，アプリケーションのデータをどのようにメモリ Channel に割り当てるかが課題となる．

ここまででは，アプリケーションの回路から HBM2 に対して直接アクセスを行うケースを考えてきた．Intel FPGA には M20K と呼ばれる内臓メモリ (Block RAM: BRAM) が実装されている．これは，SRAM で実装されているため，ランダムアクセスが可能なメモリとなっている．したがって，HBM2 上のデータを直接操作するのではなく，計算に必要なデータを一旦 BRAM にコピーし，計算が終わったら結果を BRAM から書き戻すアプローチが考えられる．

BRAM を経由するため，STREAM のような単純なメモリアクセスを行うコードではレイテンシ面から性能が低下する．しかしながら，バルク転送となるため HBM2 コントローラの ROB を無効化できることや，BRAM がキャッシュのような立ち位置となりデータの再利用ができる利点がある．また，一度 BRAM に転送してしまえばランダムアクセスをしても性能が低下しない．ステンシル計算のような，ランダムアクセスを行う場合や，同じアドレスに複

数回アクセスする場合は BRAM 併用の実装の方が良いと考えられる．

全てのメモリ Channel に対してアクセスしなければならない制約は，BRAM を使う場合でも引き続き残る．細かい粒度でデータをインターリーブし，ある演算回路が全てのメモリ Channel にアクセスできるような機構を FPGA に実装すれば，この問題は緩和できる．しかしながら，FPGA 内部の回路リソースは有限であり，高機能なネットワークあるいはキャッシュを実装してしまうと，アプリケーションや通信といった部分にさける回路リソースが限られてしまう問題がある．

以上のことより，HBM2 の高い性能を FPGA から利用することは容易ではない．実用上，メモリアクセスの自由度をある程度制限しなければならないと考えられる．例えば，本稿で実装したように，ある計算回路はあるメモリ Channel にしかアクセスしないといった制限が考えられる．この状態は，MPI を用いて分散メモリ環境でプログラミングを行うことに似ている．

7. まとめと今後の課題

本稿では，HBM2 メモリに対してアクセスを行い性能測定を行うベンチマークを実装し，FPGA 上の HBM2 メモリの性能を評価した．400MHz 動作時のメモリバンド幅の理論ピークが 409.6GB/s であるところに対して，シーケンシャルアクセス時に Write で 404GB/s，Read で 382GB/s の性能を達成した．ただし，ベンチマーク部が 400MHz でしか動作できておらず，今回の結果では HBM2 の性能を全て引き出せたとは言い難い．Overclock を行い 500MHz で動作させた時の性能と比較すると，動作周波数の比 (500:400) の割合で性能が低下しており，動作周波数が HBM2 の性能に大きな影響を与えていることが判明した．最適化を行い，500MHz 動作で性能評価を行うことが今後の課題である．

章 6 で述べた HBM2 上のデータを BRAM にコピーするタイプの回路を実装する事も今後の課題の一つである．一旦 BRAM を経由するオーバーヘッドはあるものの，メモリアクセス部と計算部の回路が分離できるメリットが大きいと考えている．Intel FPGA では，回路実装をパーティションに分けて扱うことができ，パーティション単位でコンパイル結果の再利用が可能である．最適化した HBM2 関係パーティションを M，アプリケーションのパーティションを A，B とする．M を十分最適化して 500MHz 動作が可能になった場合，M を 500MHz が動作する状態を維持したまま，M+A や M+B といった FPGA 回路を作成する使い方ができる．

Intel の高位合成開発環境である Intel FPGA SDK for OpenCL でも HBM2 はサポートされている．今後は，高位合成環境におけるメモリアクセス性能の評価を行いたいと考えている．OpenCL から HBM2 を扱う際にも，Verilog HDL で扱う場合と同様な制限があることがわかっており [7]，OpenCL コードからどのようにして HBM2 を扱えば性能が出るのかを明らかにしなければならない．

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] 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：Stratix 10 FPGA を用いた ray-tracing 法による輻射輸送計算の高速化，研究報告ハイパフォーマンスコンピューティング（HPC），2020-HPC-175 (2020).

[2] 柏野隆太，小林諒平，藤田典久，朴 泰祐：OpenCL 対応 FPGA 間光リンク接続フレームワーク CIRCUS と SMI の性能評価，研究報告ハイパフォーマンスコンピューティング（HPC），2020-HPC-175 (2020).

[3] NVIDIA: VIDIA A100 | NVIDIA, `https://www.nvidia.com/ja-jp/data-center/a100/`.

[4] Meyer, M., Kenter, T. and Plessl, C.: Evaluating FPGA Accelerator Performance with a Parameterized OpenCL Adaptation of Selected Benchmarks of the HPCChallenge Benchmark Suite, *2020 IEEE/ACM International Workshop on Heterogeneous High-performance Reconfigurable Computing (H2RC)*, pp. 10–18 (online), DOI: 10.1109/H2RC51942.2020.00007 (2020).

[5] Venkataramanaiah, S. K., Suh, H. S., Yin, S., Nurvitadhi, E., Dasu, A., Cao, Y. and Seo, J. S.: FPGA-based Low-Batch Training Accelerator for Modern CNNs Featuring High Bandwidth Memory, *2020 IEEE/ACM International Conference On Computer Aided Design (ICCAD)*, pp. 1–8 (2020).

[6] Kuramochi, R. and Nakahara, H.: An FPGA-Based Low-Latency Accelerator for Randomly Wired Neural Networks, *2020 30th International Conference on Field-Programmable Logic and Applications (FPL)*, pp. 298–303 (online), DOI: 10.1109/FPL50879.2020.00056 (2020).

[7] 塙 敏博，三木洋平：宇宙物理アプリケーションのための FPGA 演算オフローディングの検討，研究報告ハイパフォーマンスコンピューティング（HPC），2020-HPC-172 (2019).

[8] kyu Choi, Y., Chi, Y., Qiao, W., Samardzic, N. and Cong, J.: HBM Connect: High-Performance HLS Interconnect for FPGA HBM, *FPGA '21* (2021).

[9] Wang, Z., Huang, H., Zhang, J. and Alonso, G.: Shuhai: Benchmarking High Bandwidth Memory On FPGAS, *2020 IEEE 28th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM)*, pp. 111–119 (online), DOI: 10.1109/FCCM48280.2020.00024 (2020).

[10] Intel: Intel Custom Foundry (EMIB), `https://www.intel.sg/content/www/xa/en/silicon-innovations/6-pillars/emib.html?countrylabel=Asia%20Pacific`.

[11] Bachrach, J., Vo, H., Richards, B., Lee, Y., Waterman, A., Avižienis, R., Wawrzynek, J. and Asanović, K.:

Chisel: Constructing hardware in a Scala embedded language, *DAC Design Automation Conference 2012*, pp. 1212–1221 (online), DOI: 10.1145/2228360.2228584 (2012).

[12] Izraelevitz, A., Koenig, J., Li, P., Lin, R., Wang, A., Magyar, A., Kim, D., Schmidt, C., Markley, C., Lawson, J. and Bachrach, J.: Reusability is FIRRTL ground: Hardware construction languages, compiler frameworks, and transformations, *2017 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pp. 209–216 (online), DOI: 10.1109/ICCAD.2017.8203780 (2017).

[13] Intel: IHigh Bandwidth Memory (HBM2) Interface Intel® FPGA IP User Guide, `https://www.intel.co.jp/content/dam/www/programmable/us/en/pdfs/literature/ug/ug-20031.pdf`.