CB-001

再結合光子の輻射輸送大規模計算に向けた HBM-FPGA 実装への考察
The evaluation for the HBM-FPGA implementation considering a large-scale radiative transfer simulation of recombination photons

古川 和輝 * Kazuki Furukawa*
横野 智也 † Tomoya Yokono†
山口 佳樹 ‡ Yoshiki Yamaguchi‡
吉川 耕司 § Kohji Yoshikawa§
藤田 典久 § Norihisa Fujita§
小林 諒平 § Ryohei Kobayashi§
朴 泰祐 § Taisuke Boku§
梅村 雅之 § Masayuki Umemura§

1. 概要

本稿は，宇宙輻射輸送シミュレーション ARGOT (Accelerated Radiative transfer on Grids using Oct-Tree) [1] の高速化に向け，FPGA による演算加速を検討した．

ARGOT は，筑波大学計算科学研究センターが天体現象解明のためのプロジェクトの一環として開発を進めているシミュレーションコードである．ARGOT は，恒星などの点光源から放射されるエネルギーを演算する ARGOT スキームと星間媒質など空間に広がった光源からの輻射エネルギーを演算するための ART (Authentic Radiation Transfer) [3] スキームの 2 つのより構成されている．そして，これまで，GPU による両スキームの高速化が検討されてきた．しかし，ART スキームはランダムに近いメモリアクセスを要求するため GPU による飛躍的な高速化が困難であることが明らかになってきた．そこで，自由度の高い，FPGA による ART スキームの加速検討が開始された．

関連研究 [4] では，HLS 設計により，ART スキームを FPGA に移植し，その実装や性能比較について詳細に報告されている．関連研究 [5] では，HLS 設計に加え，RTL 設計による演算性能評価が追加された．FPGA は CPU による実装と比較して最大 17.5 倍の大幅な性能向上を達成したことが示されている．しかし，RTL 実装はメモリ帯域の制約から FPGA のオンチップメモリ (Block RAM[10]) を積極的に用いた設計となっており，シミュレーション空間に大きな制約を与えていた．

そこで本研究では，メモリバンド幅に起因するシミュレーションサイズの制約に対する解決策として，HBM (High Bandwidth Memory) [2] を採用することにした．しかし，Xilinx Alveo UltraScale+ U280 (以下 Alveo U280) [11] が搭載する HBM を用いて ART スキームの実装を試みたが期待する十分な性能向上を得ることができなかった．例えば，本研究における HBM 評価では最大 426 GB/s のスループットが確認された．一方，ランダムアクセス時は，RAMA IP を利用した場合で最低約 20.6GB/s（read 時），RAMA IP を利用しない場合は 約 4.7GB/s となり，スループットに大きなバラツキがあることが確認された．

ART スキームの演算加速部は 1 モジュール約 9.6GB/s のデータ入力を必要とする．つまり，この結果は，効率的に HBM を利用しない場合，演算加速部の 1 モジュールで要求されるスループットの高々半分しか使用できない可能性を示唆している．また，並列度を 32 とした場合はデータ入力に約 307GB/s のスループットが必要であり，HBM にシミュレーション空間を保存することを考えると，そのアクセスが大きなボトルネックとなる可能性が高い．そこで，メモリ空間を小さく区切り，ランダム性を極力排除するメモリアラインメントと，それを実現するための ART スキーム用メモリアドレス制御について将来的に検討する．

2.ART

2.1. 宇宙輻射輸送方程式の概要

宇宙にある多くの天体はガスでできており，電磁波である光を放射する．したがって，その光の放射と物質の相互作用を考えることは宇宙で起こる物理現象を理解する上で大変重要である．一般に無数の光子の流れの振る舞いや伝搬を扱う輻射輸送方程式を利用して，宇宙輻射輸送シミュレーションを行うことになる．

粒子の速度に比べて光速は遥かに早く，宇宙輻射輸送シミュレーションでは系を定常的として扱うため，解くべき輻射輸送方程式は，

$$I_\nu(\tau_\nu) = I_\nu(0)e^{-\tau_\nu} + \int_0^{\tau_\nu} e^{-(\tau_\nu - \tau'_\nu)} S_\nu(\tau'_\nu) d\tau'_\nu \quad (1)$$

とすることができる．ただし，ν は光線の振動数，I_ν はエネルギーの強さを表す輻射強度，S_ν は源泉関数と呼ばれる無次元の物理量である．また，τ_ν を光学的厚みといい，周波数 ν の電磁波が物質を透過する際に吸収されるエネルギーの程度を示す．この変化量 $d\tau_\nu$ は，ρ を物質密度として電磁波が距離 ds だけ進んだとすると，$(d\tau_\nu \propto \rho ds)$ のように表すことができる (参考 [6]).

2.2.ARGOT プログラム

ARGOT プログラムは，ray tracing を用いた宇宙輻射輸送シミュレーションプログラムであり，星などの点光源付近の輻射輸送を扱う ARGOT スキームと星間媒質などの空間的に広がった輻射輸送を扱う ART スキームによって成り立つ．

ARGOT スキームでは，3 次元メッシュを利用して無

* 筑波大学大学院システム情報工学研究群, Graduate School of Science and Technology, University of Tsukuba
† 筑波大学大学院システム情報工学研究科 (現:日本電信電話株式会社 ソフトウェアイノベーションセンタ), Graduate School of Systems and Information Engineering, University of Tsukuba (Presently with NTT Software Innovation Center, Nippon Telegraph and Telephone Corporation)
‡ 筑波大学システム情報系, Faculty of Engineering, Information and Systems, University of Tsukuba
§ 筑波大学計算科学研究センター, Center for Computational Sciences, University of Tsukuba

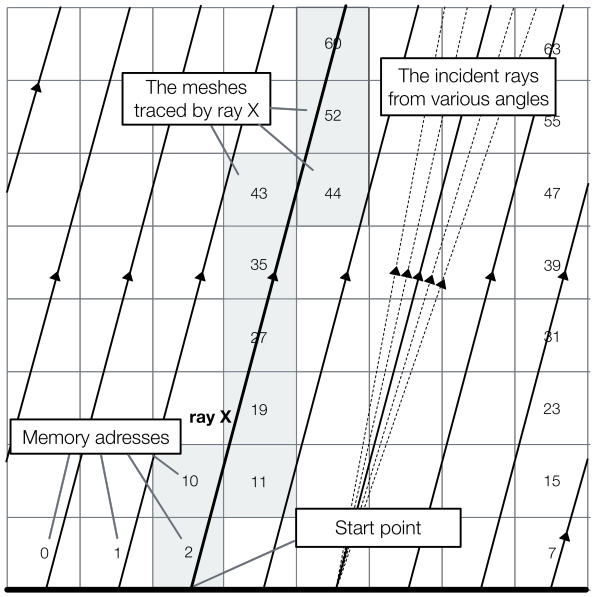

図 1: ART スキームの Ray tracing のイメージ (2D)

数にある点光源を集約する近似アルゴリズムを提案しており，輻射輸送を式 (2) を利用して計算する．

$$I_\nu(\tau_\nu) = I_\nu(0)e^{-\tau_\nu} \tag{2}$$

これは，点光源付近であるため電磁波の吸収を考慮しない形となっている．集約された点光源の中心から，ターゲットメッシュまで至る光線が通る軌跡でエネルギーが吸収されるため，最後までに残っているエネルギーを計算する．無数にある光源がスーパーメッシュに集約されることで，アルゴリズムによる計算コストはメッシュの数を N_m，光源の数を N_s とすると，$N_\mathrm{m}^{4/3}\log(N_\mathrm{s})$ まで抑えられる．

2.3.ART スキーム

ART スキームは，ARGOT と同様に空間を 3 次元メッシュに分割することによって並列演算を行う．宇宙空間に漂う星間媒質である中性ガスは，紫外線以上のエネルギーを受け取った際に電離した後，電子と再結合し電磁波を放射する．この電磁波のエネルギーは再結合光子と呼ばれ，宇宙輻射輸送シミュレーションを行う上で重要な要素となる．ART スキームはこのような空間的に広がった輻射を考慮したアルゴリズムとなっている．

空間的に広がった光源 (星間媒質) に対しては，吸収だけでなく散乱を考慮する必要があるため，式 (1) を利用してシミュレーションする．ただし源泉関数は，ε_ν と κ_ν をそれぞれ散乱係数と吸収係数として，$S_\nu = \varepsilon_\nu/\kappa_\nu$ を利用する．すると，S_ν は τ_ν に依存せず，光線上では一定であると近似できるため，ベクトル $\hat{\boldsymbol{n}}$ 上を進む光線が距離 ΔL を進んだ時の光学的厚みを $\Delta\tau$ として，

$$I_\nu^{out}(\hat{\boldsymbol{n}}) = I_\nu^{in}(\hat{\boldsymbol{n}})e^{-\Delta\tau} + S_\nu(1-e^{-\Delta\tau}) \tag{3}$$

のように表すことができ，これが ART スキームで解くべき近似式となる．プログラムでは空間の終端面にある格子数と同じ数だけ光線を束ねた “レイ” を仮定し, レイの進行方向に向かって逐次的に演算を行う必要がある．

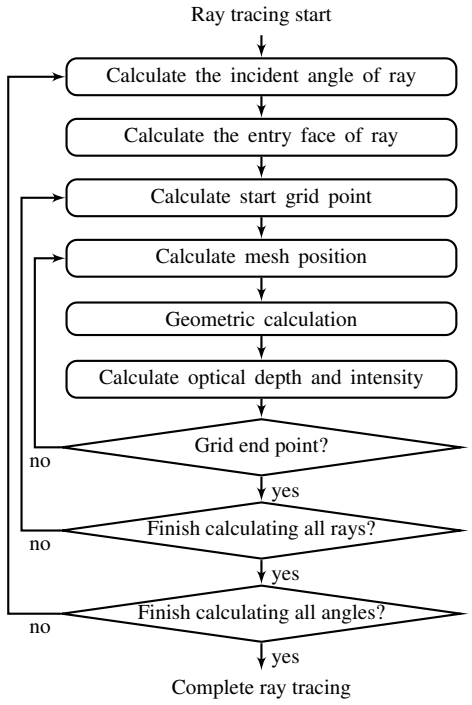

図 2: ART スキームにおける処理フロー

2.4.ART スキームの性質

ART では光線を束ねたレイ同士は，互いに干渉せず独立していると近似して演算を行うため，GPU で並列処理を行うのに適した演算スキームとすることが目指された．独立したレイを計算する際には，レイが平行に直進するとみなすことで各レイを演算単位として並列処理を行うことができるということが特徴である．これによってすべての計算量は $O(N^5)$ となる。さらに, レイベクトルの成分は HEALPix アルゴリズム [7] によって決定され，生成される典型的なレイの角度は 768 通りあるとされる．アルゴリズムは図 2 のようになり，一番内側のループがメッシュごとの計算を表す．

2.5.FPGA を利用する理由

元々 ART スキームは GPU によるシミュレーションコードとして開発されたが，ART スキームはメッシュのデータをメモリより入出力する際に，3 次元空間を演算対象としている事に起因してランダムアクセスが発生する．図 1 のように ART スキームではレイの方向に演算を進める．したがって，一見規則的なメモリアクセスではあるが，3 次元空間を進むベクトルの方向にメモリアドレスが線形的に並んでいるわけではないため，ランダムアクセスが発生することになる．

各メッシュはエネルギー等の物理現象に関する情報を保持し，演算を行う際にはその情報を取り出す必要がある．GPU の場合は，高バンド幅の共有メモリを利用する事になるが，隣のメッシュを跨ぐ場合に発生するランダムアクセスにおいてはキャッシュミスを多数誘発し，これがボトルネックになってしまう．そこで，低レイテンシかつ高バンド幅の BRAM を有した FPGA を利用し，メモリアクセスの処理を工夫した回路を作る事によって解決することが期待されている．

3. ハードウェア

3.1.AXI インターフェース

AXI (Advanced eXtensible Interface) [17] は，ARM 社が 1996 年に公開した，マイクロコントローラ・バス・ファミリである ARM AMBA の一部である．AXI には read/write 合わせて合計 5 チャネルを有している．各チャネルには，READY 信号と VALID 信号によりハンドシェイクを行う．AXI ではバースト転送がサポートされており，より高速なデータ転送を行うためには積極的に利用する必要がある．ハンドシェイクが成立したときに AXBURST 信号によってバースト方式，AXLEN 信号によってバースト長を指定する．

Xilinx 社の提供する IP で今日一般的に採用されているインターフェースは AXI Gen4(AXI4) であるが, HBM IP で利用可能なインターフェースは AXI Gen3(AXI3) であり，AXI3 は AXI4 に比べて制限されているものがある．特に AXI3 の AXLEN 信号は AXI4 が 8bit に対して 4bit であり，バースト長が短い．

表 1: AXI におけるアドレスチャネルの信号 (一部)

信号名	説明
ARLEN/AWLEN	バースト転送の回数
ARBURST/AWBURST	バースト転送の方式
ARSIZE/AWSIZE	転送するデータのサイズ
ARADDR/AWADDR	アドレス

3.2.HBM

Xilinx 社が提供する Virtex UltraScale+ HBM FPGA シリーズから利用できる HBM2 は，帯域幅の理論値が最大 460GB/s の転送を可能にする．この FPGA シリーズに搭載されているのは，Samsung Semiconductor 社が Xilinx 社に提供している高帯域幅メモリであり，積層された DRAM のダイとチップが接続された基盤を高帯域幅で接続することで，低消費電力かつ低製造コストで高帯域幅のアクセスを可能にしたものである．現在主流の外部メモリ DDR4 SDRAM に比べ高価であるが，従来存在したの高帯域幅のメモリ群に比べ安価に入手することを可能にしており，FPGA の高集積化，高速化に対してボトルネックとなってきたメモリアクセスのスループット向上に大きく貢献することが予想され，実際に HPC 分野では NEC SX-Aurora TSUBASA [8] や Fujitsu A64FX [9] などにも利用され始めている．

本研究で利用した Xilinx Alveo U280 Datacenter Accelerater Card [11] は 4GB の HBM スタックを 2 つ搭載しており，各スタックには 8 つの独立したメモリチャンネル (MC) が存在する．各 MC からは 2 つの 64bit 疑似チャンネルがつながり，それぞれが 2Gb のメモリ領域にアクセスすることができる (図 3)．また，FPGA 側より利用可能なデータ幅は 256bit で固定されている．

この機構により，各 MC からアクセスできるメモリ領域は限られているため，Alveo U280 では 32x32 のクロスバースイッチがハードウェア化されて搭載され，ユーザが利用可能なポートから直接接続している MC だけでなく，全 MC にアクセスすることができる．

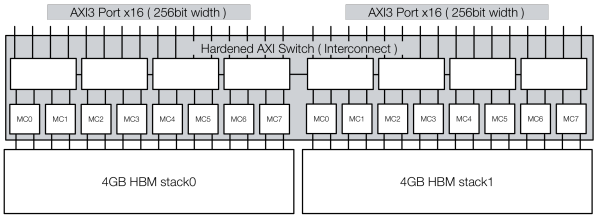

図 3: HBM

表 2: FPGA ボードの概要 ([11][15] を参考に作成)

	本研究	関連研究 [5]
FPGA Board	**Alveo U280**	**KCU1500**
Chip	Virtex XCU280	Kintex XCKU115
Registers	2,607K	1,327K(FFs)
LUTs	1,204K	663K
BRAM (MB)	43.0	9.5(75.9Mb)
HBM	4GB HBM stack x2	-

3.3.HBM に対する既知の特性

HBM は DRAM を用いて構成されているため，ランダムアクセスを行う場合 ACT コマンドと PRE コマンドによって BANK と ROW アドレスを頻繁に指定しなければならず，この操作がボトルネックになる．また，Xilinx 社は各 AXI3 ポートから直接接続された MC にアクセスする場合，最も速度を出すことができるとしており，逆に MC を 2 つずつ束ねるインターコネクトをまたぐようなアクセスがあった場合に，スループットが大きく下がるとしている [12]．これに加えて，より細かいデータブロックを扱う場合にスループットが下がることになるため，HBM を利用する際には大きなデータサイズを扱うことが高速に運用するために不可欠となってくるということが言える．

実際に Shuhai ベンチマーク [13] においては 1 ポートからの転送は検証が行われており，バースト転送の回数が少なくなるほどスループットが低くなる傾向がある．さらに，アドレスの範囲を 0x1000000, 0x1000000 で固定して計測している．しかし，より広いアドレス範囲やポート数を限らずに利用した際にどのようにスループットを確保することができるのかどうかは，実際に FPGA をアクセラレータとして用いる場合に非常に重要な要素になる．

4. ベンチマーク回路の設計

4.1. 設計環境

Vivado Design Suite 2019.2 は Xilinx 社が提供する FPGA 設計の統合ソフトウェアであり，本研究ではこのソフトウェアを利用して Verilog HDL のコンパイルから論理回路の配置配線まですべてを行った．表 2 は，本研究にて用いた Alveo U280 FPGA ボードと先行研究で利用した KCU1500 FPGA ボードの計算資源を比較したものである．本研究で利用した FPGA ボードが関連研究よりも多くの LUT や BRAM を利用可能であり，理論的にはより多くの演算モジュールを並列に配置することでシミュレーションの演算加速を行うことができる．

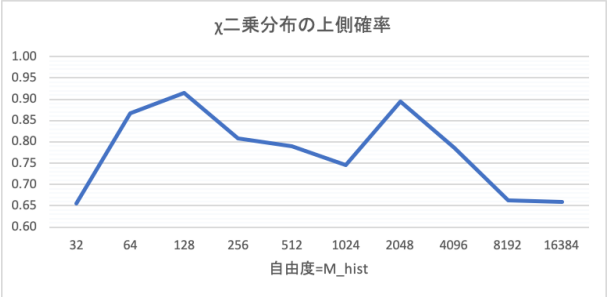

図 4: 擬似乱数の χ^2 値の検定結果

4.2. ベンチマークの目的

本研究では，Xilinx Virtex UltraScale+ HBM FPGA が搭載する高帯域幅メモリ (HBM2) を利用した ART スキームの演算回路実装を想定し，ベンチマークテストを実施した．ART スキームではメモリに対するランダムアクセスが頻繁に発生するため，使用したベンチマークでは，そのランダムアクセスが HBM 搭載の FPGA 上においてどの程度ボトルネックになりうるのかを検証する目的で行っている．また，ART スキームでは read が特に多いため，本実験では read に対して実験を行い，その性能に対して考察した．

4.3. ランダムアドレスの生成

HBM が備える AXI3 ポートのアドレスは，転送されるデータの最小ブロックサイズが 256bit であることから下位 5bit は all zero となる．したがって，本研究で生成すべき乱数は 33bit のうち最大長が上位 28bit となり，これを C 言語の random() を利用して生成した．また，生成した 10243 個の整数を一様分布に対するその χ^2 値を演算することによってそのランダム性を評価した．生成する乱数の個数を N とし，長さ M_{hist} のヒストグラムとして配列 H を用意し，乱数のばらつき度合についての評価式は以下のようになる (参考:[14]).

$$\chi^2 = \sum_{i=1}^{M_{\mathrm{hist}}} \frac{(H[i] - N/M_{\mathrm{hist}})^2}{N/M_{\mathrm{hist}}} \tag{4}$$

上記の式によって，生成した疑似乱数の χ^2 値が示す上側確率を計算した結果が図 4 であり，全体として 0.8 前後の上側確率があるので，この擬似乱数列のランダム性は十分であるとし，これをランダムアドレス生成に利用する．

4.4. 設計回路

ベンチマークでは，32Byte のデータをランダムアドレスに対して read を行うことによって評価する．32Byte は，HBM IP が一度に転送するブロックサイズであるためにその最大値として採用した．

C 言語の random() により疑似乱数を素数の 10243 個生成し，coe ファイルに入力する．この coe ファイルを FPGA の BRAM に初期値として入力することによって，乱数を BRAM より取り出し，ここからランダムアドレスを生成する．また，前述の通り AXI3 インターフェースを利用してデータを入出力することになるため，AXI インターフェースに対するマスターモジュールを作成した (図 5).

また，Xilinx 社は RAMA(Random Access Master Attachment) IP[19] というランダムアクセスをリオーダリングすることができるライブラリを提供している．これを利用した際のパフォーマンスに関しても計測を行った．

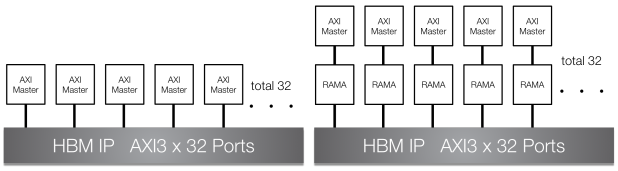

図 5: 設計図の概要

表 3: ランダムアドレス領域とアクセス対象ポート数の関係

Address bit	1Port	2Port	4Port	8Port	16Port	32Port
AXI:Memory Port	1:1	1:2	1:4	1:8	1:16	1:32
4H/4GB 領域	[28:10-6]	[29:10-6]	[30:10-6]	[31:10-6]	[32:10-6]	-
8H/8GB 領域	[28:10-6]	[29:10-6]	[30:10-6]	[31:10-6]	[32:10-6]	[33:10-6]

4.5. 実験条件

ベンチマークを行うにあたって，ランダムアクセスを行った際のアクセス対象のポート数とアドレスの中でランダム数として割り振ったビット範囲を以下の表 3 に示す．この表に示されたように HBM stack を 1 つ使った場合と 2 つ使った場合でそれぞれランダムアクセスを行い，検証を行った．文献 [2] のように HBM は最小ブロックサイズが 256bit(32Byte) であるため，アドレスの下位 5bit[4:0] は不使用となり，バーストサイズによって不使用となるアドレスの範囲が変動する．アドレス幅は 8H では 34bit，4H では 33bit となり，使用する AXI ポート数は 8H で 32，4H で 16 となる．また，バースト長 (AXLEN) はそれぞれ 1,2,4,8,16 を試した，ただし，全ての実験において HBM IP におけるグローバルアドレッシングを選択した．スループットの計測には HBM IP に付属する Hardware Debug Core を利用した．また，RAMA IP を利用した場合の読み込み速度に関しても計測することで，RAMA IP の有用性も検証を行う．RAMA IP で設定したリオーダーキューは，深さ 512 とした．

5. ベンチマーク結果

前述の設計を実装し動作させ，デバッグコアを利用して結果を確認した結果，読み込みの際の速度は図 6，RAMA IP を利用した場合には図 7 のようになった．各図の左側 5 つのデータ群は HBM の stack を 2 つ利用して 8GB のメモリ空間 (8H)，右側 4 つは stack を 1 つ利用して 4GB のメモリ空間 (4H) のグローバルアドレッシングを行った場合の結果を表している．また，1 つのデータ群は各ポートからアクセスを行うポートの範囲を示している．例えば “8H 16Ports” は，2stack(8H) を用いた場合に AXI 32 Port に 32 個のマスターを接続し，各 AXI Port からは 16 Ports 分の MC に対するアドレス領域にランダムアクセスするということ表している 3.

ただし，グラフ上において 16 Burst において誤差範囲が大きくなっている値は，スループットが不安定になり正しくメモリアクセスが行われていない可能性が高い．原因は調査中であるが，文献 [13] においてもバーストサ

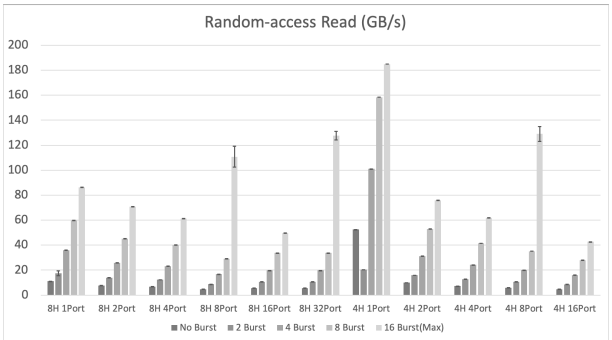

図 6: ベンチマーク結果 (GB/s) : read

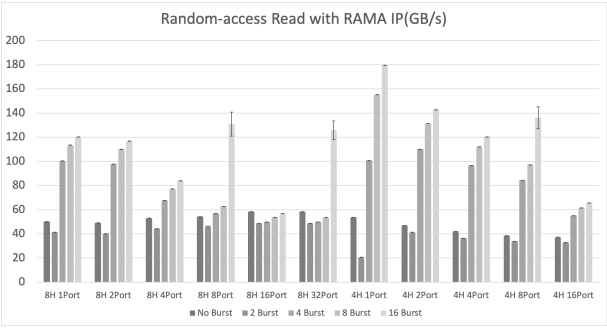

図 7: ベンチマーク結果 (GB/s) : read with RAMA IP

Resource Utilisation		Without RAMA IP		With RAMA IP	
Resource	Available	#	%	#	%
LUT	1303680	19320	1.4819587	52500	4.027062
LUTRAM	600960	49	0.01	8420	1.4010916
FF	2607360	11199	0.42951488	59763	2.2920885
BRAM	2016	356	17.65873	387	19.196428
IO	624	2	0.32051283	2	0.32051283
BUFG	1008	4	0.39682543	4	0.39682543
MMCM	12	1	8.333334	1	8.333334

表 4: リソース使用率 (HBM stack x2)

イズが 512Byte の時はベンチマークが行われていない．

結果を見ると，ランダムアクセスを行った場合読み込みはバーストサイズを上げるほどスループットが高くなっているのに対して，書き込み時にはバーストサイズはスループットに大きく関与していない．本研究では，U280 においては最大 426GB/s を出すことができることが確認されており，これとベンチマーク結果を比較するとスループットは著しく低いことがわかる．ただし，RAMA IP を利用するとランダムアクセス時には HBM IP のクロスバースイッチへの負荷が減るため，スループットは改善している．

さらに，今回の実験での設計におけるリソース使用率を表 4 で示す．乱数生成のために BRAM を比較的多く利用しているが，全体としてはアドレスを制御するのみであるため非常に小さな回路となった．

6. 結果を踏まえた評価

ART スキームでは，シミュレーション空間の 1 メッシュあたり 48Byte のデータを有しており，このアクセラレータを FPGA に実装した場合，1 モジュール 19flops/clk の演算能力があるため (表 5)，1 モジュールあたり 0.50GB/flop のデータ入力 (read 時) が必要となる．ただし，Xilinx 社提供の Floating-Point Operator IP を用いて演算すると，毎クロック結果が出力されるため，表 5 に示したようなレイテンシをほとんど無視することができる．したがって，FPGA を 200Mhz で動作させた場合，以下の計算を行うことで 1 モジュールあたり最大約 9.6GB/s のメモリアクセスが発生する事が分かる．

表 5: 関連研究 [5] で実装された ART スキームにおける Floating-Point Operator IP の性能概要

Operator	Mult	Add/Sub	Div	Comp	Exp	Total/mesh
Latency(clk)	4	4	9	1	6	36
Output(Byte) / clk	4	4	4	4	4	48
# of Op / mesh	8	5	1	4	1	19

$$48[\mathrm{Byte/clk \cdot module}] \times 0.2[\mathrm{Ghz}] = 9.6[\mathrm{GB/s \cdot module}]$$

メモリアクセスの工夫をせずに ART スキームをそのまま FPGA に実装した場合の実効性能を検証するために，4GB のメモリ空間に対してバースト転送なしのランダムアクセスによる読み込みが発生すると仮定してみる．その際の HBM のスループットは RAMA IP を利用して約 53.7GB/s，利用しないと約 4.7GB/s であるため，単純計算を行うと RAMA IP を利用しないと高々 1/2 並列，RAMA IP を利用しても 5 並列分の実効性能しか出すことができない事になってしまう．

もしも関連研究で実装されたのと同じ 32 並列を実装させるとすると，単純計算で 1 モジュールに必要なデータ入力速度の 32 倍の約 307GB/s のデータ入力が発生するため，HBM に対するメモリアクセスを工夫してもランダムアクセスである場合はスループットが不足する可能性がある．関連研究よりもさらに回路規模が大きな Alveo U280 などの FPGA を利用し並列度を上げて実装する場合には，たとえ HBM の最大スループットである 425GB/s を実現することができたとしても，メモリアクセスがボトルネックになる可能性が十分に考えられる．

7. 結論

本稿では，宇宙輻射輸送シミュレーションコード ARGOT で用いられる ART スキームを Alveo U280 に実装した場合に，シミュレーション空間を HBM を利用して保存すると仮定し，HBM に対して発生するメモリアクセスボトルネックの程度について評価する目的でベンチマークを行った．ベンチマークでは，様々なアドレス範囲とバーストサイズに対してランダムアクセスを行うことで，HBM の特性を探った．その結果，アドレス範囲が広くバーストサイズが小さい場合には HBM の内部のスイッチに大きな負荷がかかるためにスループットは著しく低下し，最大 426GB/s に対して read で約 4.7GB/s の最小速度となることが確かめられた．ただし，RAMA IP を利用すると約 20.6GB/s までスループットは改善されることも分かった．

この結果から，FPGA に実装された ART スキームの演算部において 1 モジュールあたりのデータ入力が約 307GB/s 必要であることから，HBM によるシミュレーション空間の保存にはスループットが不足する可能性がある．そのため，実際に HBM を利用して演算加速を行う際には，できる限りメモリ空間を小さく区切ったメモ

リアライメントにより HBM を利用することで，その性能を最大限引き出す必要があることが分かった．

謝辞

本研究の一部は，科学研究費補助金 JP17H01707，JP18H03246，文科省「次世代領域研究開発 (高性能汎用計算機高度利用事業)」における「次世代演算通信融合型スーパーコンピュータの開発」，TIA 連携プログラム探索事業「かけはし」2019，2020 年度の助成を受けたものである．また Xilinx 社より「Xilinx University Program」を通じて開発ソフトウェアの支援を受けておりここに謝意を表する．

参考文献

[1] Takashi Okamoto, Kohji Yoshikawa and Masayuki Umemura. ARGOT: accelerated radiative transfer on grids using oct-tree. Monthly Notices of the Royal Astronomical Society, Vol. 419, No. 4, pp. 2855–2866, 2012.

[2] Samsung Semiconductor. High bandwidth memory is catching on for data-intensive computing. January 8, 2018. https://www.samsungsemiblog.com/memoryandssd/high-bandwidth-memory-is-catching-ond-for-data-intensive-computing/

[3] Satoshi TANAKA, Kohji YOSHIKAWA, Takashi OKAMOTO and Kenji HASEGAWA. A new ray-tracing scheme for 3D diffuse radiation transfer on highly parallel architectures. Publications of the Astronomical Society of Japan, Vol. 67, No. 4, pp. 62(1–16), 2015.

[4] 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. 宇宙輻射輸送コードにおける OpenCL による FPGA 演算加速最適化. 情報処理学会論文誌 コンピューティングシステム（ACS）Vol.12 No.3 64-75 (2019 年 3 月)

[5] 横野 智也. FPGA を用いた宇宙輻射輸送シミュレーションの高速化. 筑波大学大学院博士課程 システム情報工学研究科修士論文. (2019 年 3 月)

[6] 梅村雅之，福江純，野村英子.『輻射輸送と輻射流体力学』．日本評論社．2016 年

[7] Krzysztof M Gorski, Eric Hivon, AJ Banday, Benjamin D Wandelt, Frode K Hansen, Mstvos Reinecke, and Matthia Bartelmann. Healpix: a framework for high-resolution discretization and fast analysis of data distributed on the sphere. The Astrophysical Journal, Vol. 622, No. 2, p. 759, 2005.

[8] NEC. NEC SX Aurora TSUBASA Architecture. NEC Vector Engine Processor. https://www.hpc.nec/documents/guide/pdfs/Aurora_ISA_guide.pdf

[9] Fujitsu. Press releases. Fujitsu Presents Post-K CPU Specifications. August 22, 2018. https://www.fujitsu.com/global/about/resources/news/press-releases/2018/0822-02.html

[10] Xilinx. UltraScale Architecture Memory Resources User Guide, UG573 (v1.10) February 4, 2019. https://www.xilinx.com/support/documentation/user_guides/ug573-ultrascale-memory-resources.pdf

[11] Xilinx. Alveo Data Center Accelerator Card Platforms User Guide UG1120 (v1.1). April 22, 2020 https://www.xilinx.com/support/documentation/boards_and_kits/accelerator-cards/ug1120-alveo-platforms.pdf

[12] Chris Riley. Basic Tutorial for Maximizing Memory Bandwidth with Vitis and Xilinx UltraScale+ HBM Devices. Nov 11, 2019 https://developer.xilinx.com/en/articles/maximizing-memory-bandwidth-with-vitis-and-xilinx-ultrascale-hbm-devices.html

[13] Zeke Wang, Hongjing Huang, Jie Zhang, Gustavo Alonso. Shuhai: Benchmarking High Bandwidth Memory on FPGAs. 2020 IEEE 28th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM), p.111-119, 9 May 2020.

[14] 東京大学教養学部統計学教室編.『統計学入門』．東京大学出版会．1991 年

[15] Xilinx. UltraScale Architecture and Product Data Sheet: Overview. May 20, 2020. https://www.xilinx.com/support/documentation/data_sheets/ds890-ultrascale-overview.pdf

[16] Xilinx, AXI High Bandwidth Memory Controller v1.0, PG276 (v1.0) October 30, 2019. https://www.xilinx.com/support/documentation/ip_documentation/hbm/v1_0/pg276-axi-hbm.pdf

[17] ARM. AMBA AXI and ACE: Protocol Specification. https://static.docs.arm.com/ihi0022/g/IHI0022G_amba_axi_protocol_spec.pdf

[18] Xilinx. Vivado Design Suite User Guide Creating and Packaging Custom IP,UG1118 (v2019.1) June 12, 2019. https://www.xilinx.com/support/documentation/sw_manuals/xilinx2019_2/ug1118-vivado-creating-packaging-custom-ip.pdf

[19] Xilinx. RAMA 1.1 LogiCORE IP Product Guide Vivado Design Suite. PG310 (v1.1) November 14, 2018. https://japan.xilinx.com/support/documentation/ip_documentation/rama/v1_1/pg310-rama.pdf