Vol.2018-HPC-163 No.24
2018/3/2

宇宙輻射輸送計算におけるHDL設計とOpenCL設計の比較

横野 智也[1]　藤田 典久[2]　山口 佳樹[1,2]　大畠 佑真[1]　小林 諒平[2,1]　朴 泰祐[2,1]　吉川 耕司[2,3]
安部 牧人[3]　梅村 雅之[2,3]

概要：半導体の高集積化は，FPGA の大規模化・高機能化・低価格化をもたらし，組み込みシステム用途だけでなく高性能計算用途においても導入が検討されるようになった．しかし，FPGA 開発はハードウェア記述言語（HDL: Hardware Description Language）による設計が主流であり，FPGA の利用可能性は開発の困難さによって大きく制約を受けている．FPGA の高性能計算応用を考えたとき，C 言語や OpenCL 言語を初めとする高位記述による設計が考えられるが，開発効率などの定性的な議論はあるものの，演算性能を定量的に比較した報告は少ない．そこで本論文では，宇宙輻射輸送計算をベンチマークに，高位記述設計（OpenCL 言語による HLS 設計）と低位記述設計（Verilog HDL による RTL 設計）とを比較し，高性能計算応用からみた FPGA の利用可能性と演算性能について議論する．具体的には，原始銀河形成シミュレーションにおいて再結合光子の輻射輸送を解く ART（Authentic Radiation Transfer）法を FPGA に実装し，その演算性能について比較を行った．細かな演算回路の調整や外部インタフェースを含むシステムとしての設計を除くと，XILINX 社と Intel 社という利用デバイスの違いがあるものの，記述方法によらず同程度の性能を得ることができることを確認できた．

1. はじめに

近年，高速計算および電力対性能比の向上を目的に FPGA を利用する事例が増加している．この流れを加速した代表例として Microsoft 社の Bing 検索 [1] を挙げることができるが，FPGA の普及が進んでいる要因の一つはその回路規模が飛躍的に増加したことと考えられている．XILINX 社の FPGA を例にとると，2010 年から現在まで，7-series, Ultrascale, Ultrascale+の 3 世代が登場している．この 3 世代の間に，ハイエンド製品である Virtex FPGA において，10 倍以上の回路規模の増大が確認できる（図 1）．この急激な変化は，FPGA の設計方法にも大きな変化をもたらした．

FPGA の回路規模が数万システムゲートの場合，ハードウェア記述言語（HDL: Hardware description Language）による RTL (Register Transfer Level) 設計が妥当であった．これは，少ない回路資源を高効率に利用する観点から，細粒度を含む全ての演算について最適化可能な記述方法が求められたためである．しかし，FPGA 1 チップの回路規模が 100 万システムゲートを超えた現在，その全ての動作を把握し，RTL 設計により完全な最適化を達成するのは困難になりつつある．そこで，高位記述言語による HLS (High Level Synthesis) 設計に注目が集まっている．Intel 社の Intel SDK for OpenCL，Xilinx 社の Vivado HLS および SDAccel など HLS 設計・開発環境は整いつつある．

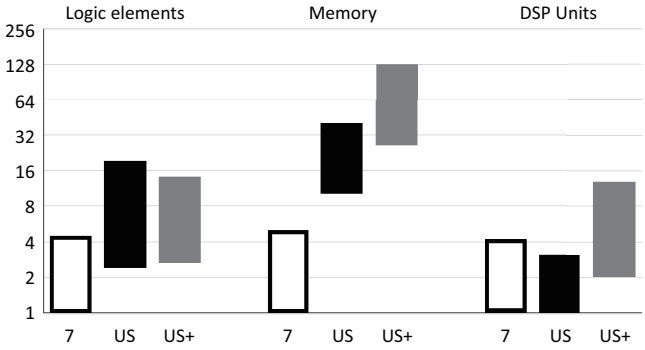

図 1 Xilinx 社 Virtex FPGA の回路規模の変遷．7 シリーズの最小 Virtex FPGA を 1 として正規化し，各シリーズ (7, US, US+) の最大と最小の範囲を図示した．

筑波大学計算科学研究センターでは，HA-PACS プロジェクトにおける密結合並列計算機構 (TCA:Tightly Coupled Accelerators) として，システム内通信を加速する目的で FPGA を用いた PEACH2 (PCI Express Adaptive Communication Hub Ver.2) を開発している [2], [3]．そして，PEACH2 に続くコンセプトとして AiS (Accelerator in Switch) が提案されており，これは通信機構の中にアプリケーションに特化した演算機構をより積極的に組み込むこ

[1] 筑波大学大学院システム情報工学研究科
[2] 筑波大学計算科学研究センター
[3] 筑波大学数理物質科学研究科

とを検討している [4], [5].

ここで，データセンターのような多くのユーザが利用しかつ複数の FPGA が並列に動作する環境において，RTL 設計のみを唯一の選択肢とし続けることはユーザビリティの点から現実的ではない．一方，高性能演算と言う観点で設計手法をみたとき，HLS 設計のみを選択肢とするのは，現時点では時期尚早と考えられる．

そこで本論文では，HDL 設計と HLS 設計の現状を等距離から評価し議論することで，次世代のヘテロジニアス高性能計算およびそこに FPGA が存在する可能性について検討する．この評価において，初期天体に関する宇宙輻射輸送シミュレーションをベンチマークとして選択した．これは，当該シミュレーションが GPU のような加速機構により高速化される部分とそうでない部分をほぼ等しく含んでいるため，GPU とは異なるアーキテクチャとの協調計算が求められるからである．本論文では，フルスクラッチ開発による HDL 設計および HLS 設計により宇宙輻射輸送シミュレーションの一部の計算を FPGA 上に実装し，その性能について検証した．

以下，本論文の構成は次のようになっている．第 2 章では，本論文において高速化の対象とした宇宙輻射輸送シミュレーションについて述べる．第 3 章では，宇宙輻射輸送シミュレーションを FPGA に実装する際，HDL 設計と HLS 設計でどのように異なるかについて述べる．第 4 章では，各実装によって得られた結果について比較・評価をおこない，続く第 5 章で本論文の結論を述べる．

2. 宇宙輻射輸送シミュレーション

2.1 概要

筑波大学計算機科学研究センターでは，宇宙輻射輸送シミュレーションコード ARGOT (Accelerated Radiative transfer on Grids using Oct-Tree) の開発を行っている．ARGOT は輻射輸送問題を解く際に，点光源からの輻射輸送計算と空間的に広がった光源からの輻射輸送計算の 2 つのスキームを組み合わせたものである．

点光源からの輻射輸送を計算する ARGOT [6], [7] は以下の式を用いて計算される．この時，ν は振動数，$I(\nu)$ は輻射強度，$\tau(\nu)$ は光学的厚みを表す．

$$\frac{dI(\nu)}{d\tau(\nu)} = -I(\nu) \tag{1}$$

次に，空間的に広がった点光源からの輻射輸送を計算する ART (Authentic Radiation Transfer) [8] は以下の式を用いて表される．

$$\frac{dI(\nu)}{d\tau(\nu)} = -I(\nu) + S(\nu) \tag{2}$$

ここで，$S(\nu)$ は源泉関数を表す．源泉関数とはあるメッシュから放射される輻射の強さを表す．本研究ではこの ART 法の部分に注目し FPGA に実装する．このため，本節以降は ART 法についてのみ述べる．

2.2 ART 法

ART 法はレイトレージングを用いた演算スキームであり，空間を 3 次元メッシュに分割して扱う．レイは，図 2 に示すように，境界から発射され直進する．

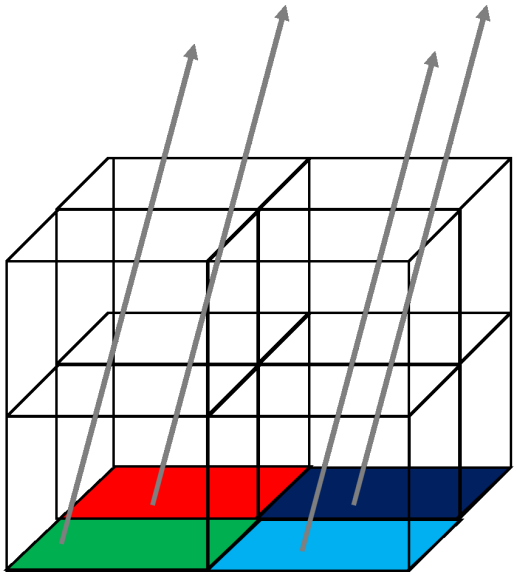

図 2 ART 法によるレイトレーシング

ART 法で扱うレイは平行光のみであり，屈折や反射などは発生しない．レイの角度や向きは NASA が提供している HEALPix アルゴリズム [9] を用いて決定される．レイに沿って輻射輸送計算が実行され，複数のレイが通過するメッシュではその演算結果が積算される．レイトレーシングの 2 次元イメージを図 3 に示す．

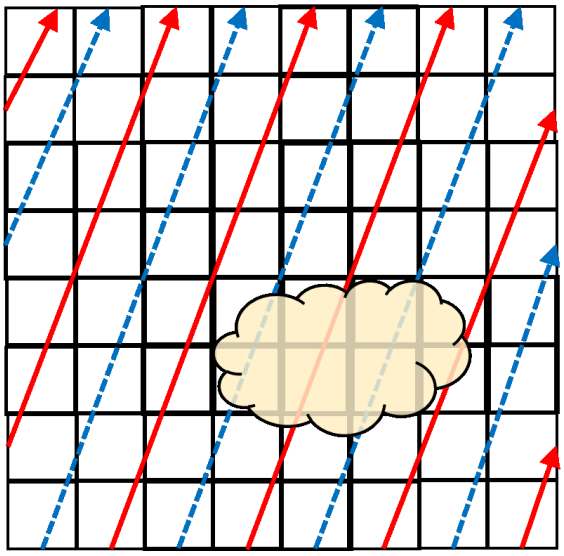

図 3 レイトレーシングの 2 次元イメージ
レイが通るガスの反応を計算する

また，メッシュとレイとの関係について考えると，あるレイがメッシュに与える影響（ないし与えられる影響）はメッシュを通過した距離に依存している．ART 法による 2 次元でのレイトレーシングのイメージを図 4 に示す．

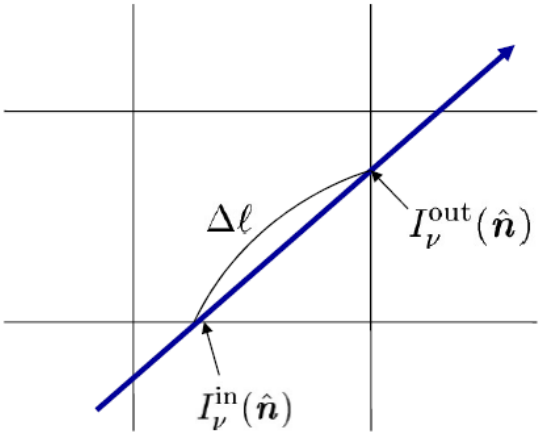

図 4　ART 法によるレイトレーシングの幾何学計算 (2 次元)

図 4 において，メッシュの計算は以下の式で表される．

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

$$\Delta\tau_\nu = \sigma(\nu)n\Delta\ell \quad (4)$$

図 4 に示すように，ART 法は各レイの計算を進行方向に向かって逐次的に行わなければならない．一方，異なるレイの間に計算の依存関係はなく，レイ単位であれば並列に計算ができる．ART 法の輻射輸送計算において，レイの入射角度は 700〜数千種類，またレイの本数が入射境界のメッシュ数に比例する．つまり，ART 法の高速化は，この並列化をどのように実現するかにかかっている．

並列化における一つの問題点は，ART 法は計算するレイの角度によって，メッシュデータへのアクセスパターンが複雑に変化することである．このため CPU や GPU などで利用される SIMD 演算を高効率に適用することが難しい．FPGA を用いる場合，ART 法に適した低レイテンシかつ高バンド幅でアクセス可能なオンチップメモリを一時保存領域として定義し，アクセスパターンを多種多様にカスタマイズ可能である．FPGA で利用可能なオンチップメモリ容量は急速に増大しており，FPGA は ART 法を加速する有力な候補となっている．

2.3　ART 法における処理フロー

ART 法では，まず，レイの角度を決定するループがあり，そこで各レイの入射面が計算される．次に，決定されたレイの角度と入射面に従い，各レイを計算するループが実行される．このループ内では，各レイに対してメッシュ中のガスの計算を行うループが存在する．つまり，ART 法の中心的な演算は 3 重ループにより実現されている．この ART 法の処理フローを図 5 に示す．

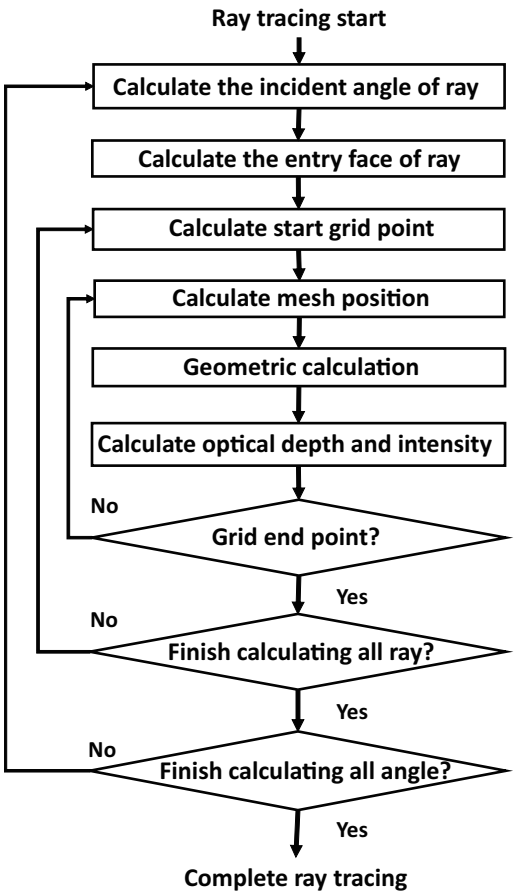

図 5　ART 法における処理フロー

ここで ART 法の計算量は以下より見積もることができる．一つの辺を $N(=N_x=N_y=N_z)$ とすると，角度のループが $O(N^2)$，レイのループが $O(N^2)$，メッシュのループが $O(N)$ となる．したがって，すべての計算量は $O(N^5)$ となる．前節でも述べたとおり，各レイは独立しているため，内側の 2 重ループは計算が独立している．この性質に注目することによって高速化を図っていく．

3.　ART 法の FPGA 実装

本論文では，ART 法を FPGA に実装にあたってハードウェア記述言語 (HDL:Hardware Description Language) である Verilog-HDL と高位記述言語である OpenCL による実装 [10] について比較する．以下，Verilog-HDL 設計と OpenCL 設計についてそれぞれ説明する．

3.1　RTL 設計と HLS 設計

FPGA 上にアプリケーションを実装する手段は，HDL を用いた RTL (Register Transfer Level) 設計と高位記述言語による HLS (High Level Synthesis) 設計とに大きくわけることができる．表 1 にその特徴を示す．

表 1　FPGA 実装における RTL 設計と HLS 設計

	RTL 設計	HLS 設計
利点	HDL による論理回路レベルでの細粒度記述	高位記述言語による動作レベルでの記述
欠点	設計時間の長期化	回路レベルの最適化が困難

これまで FPGA の回路開発には，HDL を用いた RTL 設計を適用するのが一般的であった．この理由として，FPGA の回路量がそれほど大きくなく，演算回路の最適化による恩恵（演算性能の向上，消費電力の削減，回路の小規模化）が開発に要する時間を大きく上回ったからである．一方，FPGA の回路規模が 100 万ゲートを超えた現在，FPGA の全演算回路を RTL 設計により使いこなすのは非常に難しくなりつつある．このため，C 言語や OpenCL 言語などによりアルゴリズムの動作を直接的に記述し，そこから RTL 記述を生成する HLS 設計に注目が集まっている．そして，これはユーザビリティを高めることにつながり，一部の組み込み系システムだけでなく高性能演算を初めとする他のシステムにおいても利用する可能性を高めている．

HLS 設計では，高位合成言語から中間言語として HDL を生成して RTL 設計に繋げるため，開発期間は RTL 設計と比べ大幅に短縮できる．一方，その演算性能は高位合成言語から HDL を生成するパーサーの性能に依存する部分が大きく，期待する性能を得るにはパーサーが処理しやすい HLS 記述を推測して記述する必要がある．加えて，より高い性能を出すには，FPGA アーキテクチャも考慮しての記述が要求される．このため，CPU や GPU 向けに記述・最適化されたコードをそのまま用いても性能が出ないことがわかっており，FPGA 向けの最適化をどう行うかが課題となる [11],[12].

3.2　RTL 設計による ART 法の FPGA 実装

FPGA 実装において，FPGA の空間並列度を活かす形でレイ計算の並列度を高めることが重要となる．そこで本論文では，レイを計算するアクセラレータについて最適化した RTL 設計を準備し，それを増やす形で高速化を図った．RTL 設計によるブロック図を図 6 に示す．

アクセラレータは完全パイプライン化され，レイの情報を FIFO (First In, First Out) から毎クロック読み込み，演算を実行する．このとき，ART 法の空間は FPGA のオンチップメモリより非常に大きいため，RTL 設計では将来のオフチップメモリ利用を考慮した設計となっている．つまり，メモリに直接ではなく，アクセラレータはレイの情報およびメッシュの計算結果を FIFO を通じてアクセスしている．コントロールモジュールは，FIFO と FPGA のオンチップメモリとのスイッチとして機能し，円滑に計算が進むように全体の計算を制御している．

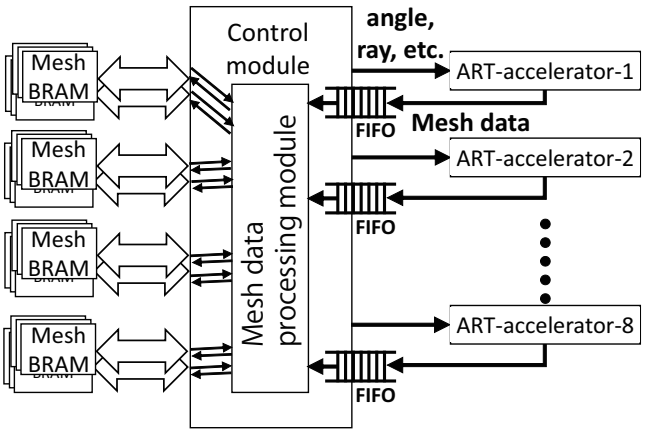

図 6　RTL 設計における実装のブロック図

3.3　HLS 設計による ART 法の FPGA 実装 [10]

HLS 設計による ART 法の実装では，RTL 設計と同様に，演算を加速するアクセラレータを複数用意することで並列化を行う．RTL 設計と異なる点は，レイに対して並列に処理をするのではなく，図 7 に示すようにメッシュを中心に考え，分割されたメッシュ空間と PE (Proccessing Element) を対応させて計算を記述した点にある．

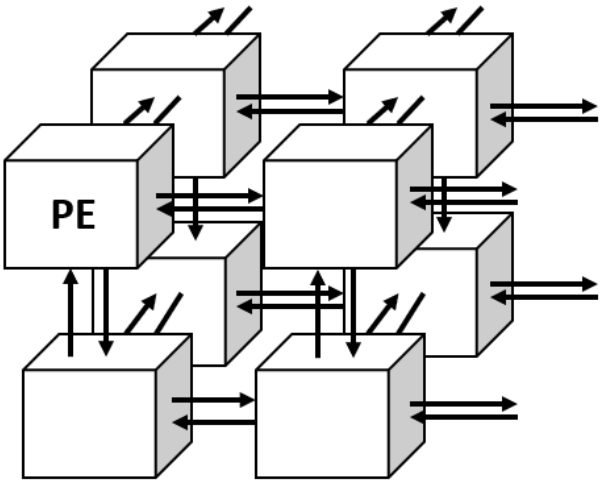

図 7　HLS 設計の空間分割と PE の配置

このため，レイがどの PE で演算されるかはレイの位置によって決定される．レイトレージングの計算が進むと，レイの情報は隣接する PE に Channel と呼ばれる FIFO を用いて隣接する PE に転送され，計算が継続される．このとき，PE の内部構造は図 8 に示すとおりである．

レイは，図 8 の Ray in より入力され，Ray data に格納されてメッシュの演算に用いられる．メッシュデータは予め Mesh RAM に格納されており，これらを利用することで担当する分割空間のループ演算を実現する．演算が終了すると，レイは Ray out を通じて隣接 PE に移動する．

この構造は，シンプルかつ並列化しやすいという特徴を

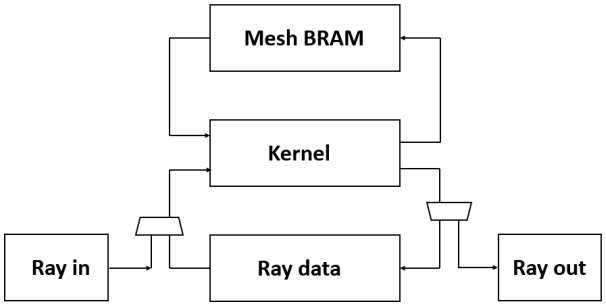

図 8　PE 内の計算処理

備えており，HLS 設計に適したものと考えられる．

4. FPGA 実装における各性能の評価

4.1 評価環境

RTL 設計において，本論文では，Xilinx 社の Kintex Ultra Scale KCU1500 (XCKU115-2FLVB2104E) を用いた．開発環境は，同 Xilinx 社の Vivado2017.4 を使用している．

HLS 設計には，BittWare 社の A10PL4 を用いた．このボードには，前述した XCKU115 と同規模の FPGA である Intel 社の Arria10 GX115 が搭載されている．開発環境は，Intel 社の FPGA SDK for OpenCL を用いた．この SDK を用いることで，OpenCL で記述したプログラムを FPGA 回路にコンパイルできる．

これら 2 つの実装は，異なる言語 (RTL 設計と HLS 設計)，異なる実装方式 (レイベース設計とメッシュベース設計)，異なる FPGA アーキテクチャ (Xilinx FPGA と Intel FPGA) であるため直接比較することは難しい．例えば，浮動小数点演算についても，Xilinx の DSP と Intel の DSP では構造が異なるため，同じ演算でも必要となる個数が異なる．しかし，上記を留意しつつ両手法を比較することで，RTL 設計および HLS 設計のそれぞれの特徴について確認することができる．上記より，本評価では RTL 設計および HLS 設計の優劣をつけるものではなく，あくまでも双方の特徴を浮き彫りにすることを目的とした．

最後に，CPU についても比較対象として用いることとした．本論文では，この実装に，Intel 社の Xeon E5-2660 v4 を用いた．本ソフトウェアプログラムは C 言語および OpenMP によって記述され，gcc4.8.5 を用いてコンパイルされている．

これら 3 つの設計についてまとめたものを表 2 に示す．

4.2 評価手法

性能評価には，ARGOT プログラムから ART 法の計算を行うコア部分のみを抜き出して作成したベンチマークプログラムを用いる．ARGOT から入力メッシュデータを得られないため，ART 法で使用する入力メッシュデータは乱数により生成した．性能評価において，ART 法の計算量は入力データに依存しないため，上記の処置は特に問題ないと考えられる．

次に，ARGOT は問題で扱うガスの組成を，水素のみ，水素＋ヘリウムの 2 通り扱える．今回の性能評価では，汎用性を考慮し，水素＋ヘリウムのガスを扱った．ART 法で取り扱う問題サイズは $(N_x, N_y, N_z) = (512, 512, 512)$ 以上が望ましい．しかし，今回の評価では FPGA 上のオンチップメモリで演算できる範囲としたため，$(N_x, N_y, N_z) = (16, 16, 16)$ で評価を行っている．また，HEALPix を用いてレイを生成する際に用いるパラメータ $N_{side} = 8$ としたので，生成されるレイの種類（角度）は 768 通りとなる．

RTL 設計における演算時間は，FPGA 回路内に作成したカウントレジスタを用いて測定をした．カウントレジスタからのカウント値の取り出しは，Xilinx が提供しているロジック・アナライザ IP である ILA (Integrated Logic Analyzer) を用いている．計算終了信号をトリガとして読み出されたカウント値を動作周波数と掛けあわせることで，演算時間を求めることができる．OpenCL 実装における計算時間は，ホスト CPU 上で測定した．このため，OpenCL カーネルの起動時間及び同期の待機時間は含まれるが，メモリ転送時間は計算時間に含めていない．

4.3 FPGA リソース使用量

4.3.1 RTL 設計におけるリソース使用量

RTL 設計における各種リソース使用量を表 3 上段に示す．LUT (Look-up Table) は Xilinx FPGA における論理回路を実現する最小ブロック単位であり，表内の数値はその使用量を意味する．Register は LUT と対で配置されたレジスタてあり，数値は実際の使用量を意味する．BRAM (Block RAM) は，XILINX FPGA におけるオンチップメモリの呼称であり，今回の実装で用いた Kintex XCKU115 FPGA では 1 つの BRAM あたり 32Kbit の容量を持つ．DSP (Digital Signal Processing unit) は加算器や乗算器を含む数値演算のためのハードウェア IP である．浮動小数点演算はこの DSP を用いて実装されている．

4.3.2 HLS 設計におけるリソース使用量

FPGA OpenCL 実装における各種リソース使用量を表 3 に示す．ALM(Adaptive Logic Module) は Intel Arria10 FPGA における論理回路を実現する最小ブロック単位であり，Xilinx FPGA の LUT に対応したものである．Resister は各 ALM と対で配置されたレジスタであり，これも前節と同じである．M20K は，Intel FPGA におけるオンチップメモリの呼称であり，1 つの M20K ブロックは 20Kbit の容量を持つ．本論文で使用した Intel FPGA が持つ DSP は，IEEE 準拠の単精度浮動小数点数積和をサポートしている．OpenCL 上で float 型の演算を記述すると DSP を用いて実装されるため，HLS 設計と親和性の高いアーキテクチャを採用していると言える．

表 2　評価環境

	CPU	FPGA(HLS 設計)	FPGA(RTL 実装)
製品型番（FPGA）	Intel Xeon E5-2660 v4	Intel Arria10 GX115	Xilinx Kintex XCKU115
製品型番（ボード）	-	BittWare A10PL4	Xilinx KCU1500
基本回路 (Logic Element) [個]	-	1,150K	663K
演算コア（DSP）[個]	-	1,518	5,520
オンチップメモリ	280Mbit	53Mbit	75.9Mbit
開発言語	C/OpenMP	OpenCL	Verilog-HDL
コンパイラ	gcc 4.8.5	Intel Quartus Prime Pro. Intel FPGA SDK for OpenCL Version 16.1.2.203	Vivado2017.4
Price(*)	176,039 円 [13]	1,144,000 円 [14]	254,265 円 [15]

(*USD 1=110 円で計算)

4.3.3　リソース使用量の比較

表 3 より，RTL 設計と HLS 設計について比較すると，RTL 設計の方が HLS 設計よりも少ない回路使用量で実現していることがわかる．例外は DSP だが，DSP の使用数については，4.1 節でも述べたように，単純な比較はできない．ただ，単精度浮動小数点演算をサポートしている Intel 社の FPGA 上の設計が要求する DSP 数 (2 個) に対し，XILINX 社の FPGA 上の設計が要求する DSP 数が約 1.5 倍 (∼3 個) と考えると，表 3 の数値の違いは大きく見えるが，実際の回路使用量は同程度と考えることができる．

よって，HLS 設計は，HDL 設計の 2∼3 倍の規模で同程度性能を持つデザインを提供できる可能性を持っていると言える．そして，DSP を多く要求する数値演算に代表される応用問題においては，HDL 設計と HLS 設計の差異が出にくいと考えられる．つまり，HDL 設計と HLS 設計の性能差はより小さくなると考えられる．

RTL 設計の動作周波数の低下は，FPGA アーキテクチャに依存するところもあるため，ここでは議論の対象とせずに参考として掲載するに留める．

4.4　性能評価

性能評価において指標とする CPU の実装は FPGA 実装の基となったコードであり C 言語で記述されている．この実装は OpenMP を用いて並列化されており，レイ単位でスレッド並列化されている．なお，HDL 設計および HLS 設計 [10] ともにフルスクラッチ開発であり，何れかのコードを他の評価環境で再利用することは行っていない．

ここで，HDL 設計および HLS 設計による FPGA 実装と CPU との計算時間を図 9 に示す．また，1 スレッド処理の CPU を基準とした速度比を図 10 に示す．FPGA 実装において実行時間は，RTL 設計では 4.314(ms)，HLS 設計では 3.41(ms) となっている．動作周波数比 (XILINX@200MHz と Intel@236MHz) を考慮しても，HDL 設計と比較し HLS 設計の方が 7%程度性能が高い．また，CPU (multi-thread) と比較し，HDL 設計で 11.95 倍，HLS 設計で 15.12 倍の高速化が得られており，十分な性能を提供できるレベルまで達していると考えることができる．

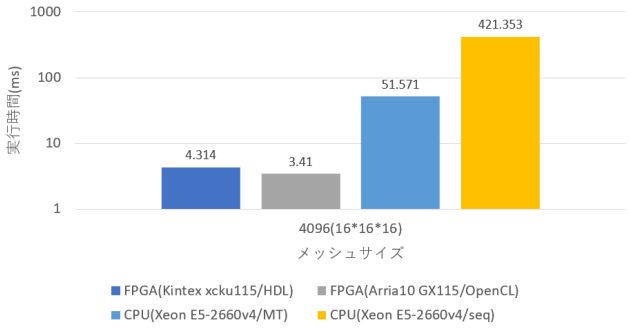

図 9　FPGA と CPU の実行時間 (ms)

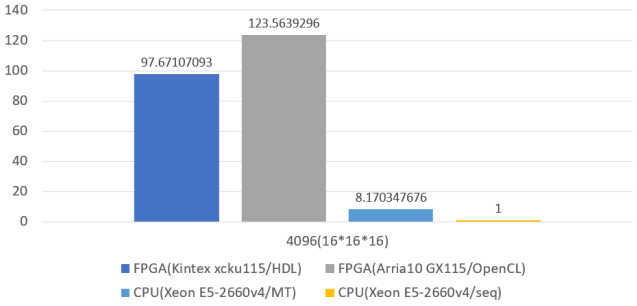

図 10　1 スレッド CPU を基準とした時の速度比

4.5　RTL 設計における性能低下について

図 11 に，アクセラレータ数を 1 から 8 まで変化させた時の高速化率を示す．

4.4 節において RTL 設計において十分な演算性能が得られていないことを指摘したが，この理由は図 11 より見て取れる．つまり，完全パイプライン化されたアクセラレータの性能ではなく，並列度の上昇に伴うメモリコントローラ部の処理遅延が原因であることがわかっている．メモリコントローラの最適化により，10%程度の性能向上が見込める．また，RTL 設計では回路規模に余裕があるため，16 並

表 3　リソース使用量とその差分 (括弧内の%は使用 FPGA における回路利用率を意味する)

	LUTs/ALMs	Registers	BRAM/M20K	BRAM/M20K·MLAB	DSP	Freq.[MHz]
RTL 設計 (Xilinx KCU1500)	132,200 (19.93%)	134,608 (10.15%)	320.5 (14.84%)	10,256Kbit	820 (14.86%)	200.00
HLS 設計 (Intel GX115)	173,282 (40.56%)	247,840 (14.50%)	1,296 (47.78%)	25,920K(M20K) +4,704K(MLAB)	512 (33.73%)	236.11
HLS/HDL 比	0.76	0.54	-	0.33∼0.40	1.60	0.847

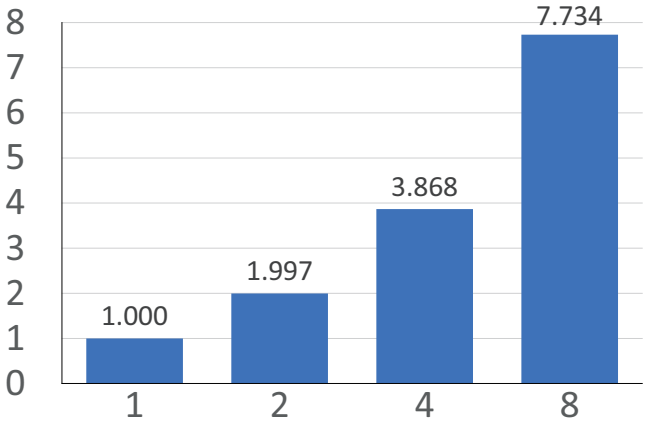

図 11　RTL 設計によるアクセラレータ数と高速化率の関係

列まで実装を拡張することが可能であることもわかっている．HLS 設計の条件に合わせて評価をおこなったが，RTL 設計においてはまだまだ性能向上の余地が残っている．

5. おわりに

本研究では，宇宙輻射輸送シミュレーションコード ARGOT で用いられている ART 法を，Kintex FPGA と Arria10 FPGA のそれぞれに対して RTL 設計と HLS 設計を適用し評価を行なった．FPGA は 1 スレッド版の CPU を基準とした時，RTL 設計で 97.67 倍，HLS 実装で 123.56 倍となった．また，マルチスレッド版 CPU を基準とした時はそれぞれ 11.95 倍，15.12 倍の高速化が達成された．

RTL 設計と HLS 設計を比較すると，回路規模やオフチップメモリを考慮した実装という観点においては，従来の予想通り HDL 設計の方が望ましいという結果が得られた．しかし，同じ並列度の際には HLS 設計の方が実行性能が勝るという結果が得られた．データセンターなどにおいて大規模 FPGA が導入された場合，FPGA 内の全演算回路を使った HDL 設計を全アプリケーションに適用するのは困難であると考えられる．この結果は，そのときに HLS 設計を用いて一定の性能向上を得ておき，より計算速度を重視するアプリケーションにおいて HDL 設計を適用していくという使い分けの可能性を示唆している．

今後は，他アプリケーションおよびマルチ FPGA 環境下における評価などを継続し，より応用性の高いデータおよび HLS 設計における知見などを集めて行く予定である．

参考文献

[1] Putnam, A., Caulfield, A. M., Chung, E. S., Chiou, D., Constantinides, K., Demme, J., Esmaeilzadeh, H., Fowers, J., Gopal, G. P., Gray, J. et al.: A reconfigurable fabric for accelerating large-scale datacenter services, *Computer Architecture (ISCA), 2014 ACM/IEEE 41st International Symposium on*, IEEE, pp. 13–24 (2014).

[2] Hanawa, T., Kodama, Y., Boku, T. and Sato, M.: Tightly coupled accelerators architecture for minimizing communication latency among accelerators, *Parallel and Distributed Processing Symposium Workshops & PhD Forum (IPDPSW), 2013 IEEE 27th International*, IEEE, pp. 1030–1039 (2013).

[3] Kodama, Y., Hanawa, T., Boku, T. and Sato, M.: PEACH2: An fpga-based pcie network device for tightly coupled accelerators, *ACM SIGARCH Computer Architecture News*, Vol. 42, No. 4, pp. 3–8 (2014).

[4] Kuhara, T., Tsuruta, C., Hanawa, T. and Amano, H.: Reduction calculator in an FPGA based switching Hub for high performance clusters, *Field Programmable Logic and Applications (FPL), 2015 25th International Conference on*, IEEE, pp. 1–4 (2015).

[5] Tsuruta, C., Miki, Y., Kuhara, T., Amano, H. and Umemura, M.: Off-Loading LET Generation to PEACH2: A Switching Hub for High Performance GPU Clusters, *ACM SIGARCH Computer Architecture News*, Vol. 43, No. 4, pp. 3–8 (2016).

[6] Scannapieco, C. e. a., Wadepuhl, M., Parry, O., Navarro, J., Jenkins, A., Springel, V., Teyssier, R., Carlson, E., Couchman, H., Crain, R. et al.: The Aquila comparison project: the effects of feedback and numerical methods on simulations of galaxy formation, *Monthly Notices of the Royal Astronomical Society*, Vol. 423, No. 2, pp. 1726–1749 (2012).

[7] Okamoto, T., Yoshikawa, K. and Umemura, M.: ARGOT: accelerated radiative transfer on grids using oct-tree, *Monthly Notices of the Royal Astronomical Society*, Vol. 419, No. 4, pp. 2855–2866 (2012).

[8] Tanaka, S., Yoshikawa, K., Okamoto, T. and Hasegawa, K.: 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).

[9] Gorski, K. M., Hivon, E., Banday, A., Wandelt, B. D., Hansen, F. K., Reinecke, M. and Bartelmann, M.: 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).

[10] 藤田典久，小林諒平，山口佳樹，大畠佑真，　朴泰祐，吉川耕司，安部牧人，梅村雅之ほか：OpenCL を用いた FPGA による宇宙輻射輸送シミュレーションの演算加速，研究報告ハイパフォーマンスコンピューティング (HPC)，Vol. 2017, No. 12, pp. 1–9 (2017).

[11] 大島聡史，　塙敏博，片桐孝洋，中島研吾ほか：FPGA を

用いた疎行列数値計算の性能評価，研究報告ハイパフォーマンスコンピューティング (HPC)， Vol. 2016, No. 1, pp. 1–9 (2016).
[12] 塙敏博，伊田明弘，大島聡史，河合直聡ほか：FPGAを用いた階層型行列ベクトル積，研究報告ハイパフォーマンスコンピューティング (HPC)， Vol. 2016, No. 40, pp. 1–9 (2016).
[13] .: Xeon E5-2660 v4 BOX, Kakaku.com, Inc. (online), available from ⟨http://kakaku.com/item/K0000869941/⟩ (accessed 2018-Jan-25).
[14] .: Systems on FPGA - System on a Programmable Chip (SoPC).
[15] .: DK-U1-KCU1500-A-G - Development Kit, Kintex Ultrascale FPGA, Premier Farnell Limited (online), available from ⟨http://www.newark.com/xilinx/dk-u1-kcu1500-a-g/development-kit-kintex-ultrascale/dp/42AC9931⟩ (accessed 2016-Sep-08).