Vol.2021-ARC-244 No.5
Vol.2021-SLDM-194 No.5
Vol.2021-EMB-56 No.5
2021/3/25

オイラー動画像誇張処理を対象としたCPU-FPGAハイブリッドシステムの実装と評価

上野 麟[1,a)] 谷本 輝夫[1,b)] 後藤 孝行[1,c)] 丸岡 晃[2,d)] 川上 哲志[1,e)] 小野 貴継[1,f)] 飯塚 拓郎[3,g)] 井上 弘士[1,h)]

概要：オイラー動画像誇張処理 (EVM) は動画像における見えの微小な変化を数値計算により増幅し誇張することでそれらの検出を容易にするアプリケーションである．本研究では，組み込み機器における EVM のリアルタイム実行の実現を目指しており，そのため CPU–FPGA ハイブリッドシステムとしての設計及び実装に取り組んでいる．本稿では，CPU–FPGA システムの基本実装（ベースライン）からボトルネックとなる FPGA オフロード部分を抽出し，これに対して，1) CPU–FPGA 間のデータ転送性能の向上，2) FPGA 処理における演算並列度の改善，を検討する．そして，3) 当該オフロード部分の実装に基づく性能電力モデルを構築し，改善方式の効果を評価する．評価結果に基づくモデリングの結果，全ての IP コアへ本改善手法を適用できれば電力投資効果の観点から GPU 利用に対し 1.129 倍の性能向上を達成できる可能性が明らかになった．

1. はじめに

近年，低遅延リアルタイム・アプリケーションの普及やネットワーク帯域の節約などを背景に，センサーやアクチュエータといった入出力デバイスの近傍で情報処理するエッジコンピューティングの重要性が高まっている [5]．その代表的な適用例として，拡張現実やインテリジェント・カメラ，自動運転支援など，カメラ入力に対するリアルタイム動画像処理が挙げられる．その特徴は，低消費電力化や小型化，低コスト化といった従来の組込みシステム開発において重要視された設計制約に加え，膨大なデータをリアルタイム処理するための高い実効性能が求められる点にある．

一般に，エッジコンピューティングでは様々なアプリケーションの実行が求められるため，より柔軟性の高い実行処理基盤の構築が鍵となる．そこで我々は，再構成可能ハードウェアの一種である FPGA (Field Programmable Gate Array) に着目し，次世代エッジコンピューティング・プラットフォームとして CPU–FPGA ハイブリッドシステムに関する研究を進めている．そしてその一環として，オイラー動画像誇張処理（Eulerian Video Magnification, EVM [7]）のリアルタイム処理実現に向けた実装評価を行った [8]．EVM は，カメラから入力する動画像中の色や位置の微小な変化を数値計算により増幅し，その検出を容易にするアプリケーションである．応用例として，顔色の微小な変化に基づく脈拍推定や，胸部の動きの監視による呼吸判定などが挙げられ，今後の更なる普及が見込まれる．この既存研究では EVM のハードウェア/ソフトウェア分割を基本とする実装可能性を示したものの，その実効性能は必ずしも十分ではなかった．

そこで本研究では，上記既存研究でのベースライン実装からボトルネックとなる FPGA オフロード部分を抽出し，これに対して，1) CPU–FPGA 間のデータ転送性能の向上，2) FPGA 処理における演算並列度の改善，を検討する．そして，3) 当該オフロード部分の実装に基づく性能電力モデルを構築し，改善方式の効果を評価する．本評価においては，比較対象として CPU-GPU ハイブリッド実装を追加し，Ultra96V2 評価ボードならびに Jetson Nano を対象としたボードレベルの消費電力測定環境を構築した．実験の結果，当該オフロード部分に関し，既存研究でのベースライン実装と比較して 2.3 倍の高速化を達成した．一方，モデリングに基づく CPU–GPU ハイブリッド実装との消費電力

[1] 九州大学
[2] 株式会社フィックスターズ
[3] Fixstars Solutions Inc.
[a)] rin.ueno@cpc.ait.kyushu-u.ac.jp
[b)] tteruo@kyudai.jp
[c)] takayuki.goto@cpc.ait.kyushu-u.ac.jp
[d)] akira.maruoka@fixstars.com
[e)] satoshi.kawakami@cpc.ait.kyushu-u.ac.jp
[f)] takatsugu.ono@cpc.ait.kyushu-u.ac.jp
[g)] t_iizuka@fixstars.com
[h)] inoue@ait.kyushu-u.ac.jp

Vol.2021-ARC-244 No.5
Vol.2021-SLDM-194 No.5
Vol.2021-EMB-56 No.5
2021/3/25

ならびに性能に関する比較においては優位性を確認することができず，さらなる改善の必要性が明らかとなった．また，電力投資効果（同一ボードに搭載された CPU を基準とした，アクセラレーションのために消費する電力当りの性能向上率）の観点から評価した結果， 全ての IP コアへ本改善手法を適用できれば，GPU 利用に対し 1.129 倍の性能向上を達成できる可能性が明らかになった．

本稿の構成は以下の通りである．第 2 章では EVM について説明する．第 3 章では EVM の CPU–FPGA ハイブリッドシステム実装について述べる．第 4 章では比較対象である CPU–GPU ハイブリッドシステムの実装について述べる．第 5 章では実験と評価を行い，第 6 章で設計改善の適用について検討する．最後に第 7 章で本稿をまとめる．

2. オイラー動画像誇張処理

2.1 概要

オイラー動画像誇張処理は，通常肉眼で視認不可能な微小変化を含んだ動画像に対して，複数段の画像フィルタ処理を施すことで微小変化を誇張・増幅し，視認可能にする画像処理アプリケーションである．このアプリケーションを構築する処理は微小な色変化増幅加工を行うか，微小な物体動作の増幅加工を行うかでアプリケーションを構成する処理が異なる．本研究では微小な物体動作の増幅加工を対象としたオイラー動画像誇張処理について説明を行う．オイラー動画像誇張処理を構成する処理の中でも特に増幅加工に重要となる処理が 2 つ存在し，それぞれラプラシアンピラミッド生成処理と IIR (Infinite Impulse Response) フィルタ処理と呼ぶ．

ラプラシアンピラミッド生成処理は画像処理アプリケーションでしばしば利用される．オイラー動画像誇張処理では動画像中に含まれる増幅対象が含まれた物体境界を特定周波数帯成分ごとに抽出するために利用される．ガウシアンフィルタによる畳み込み処理と画像縮小処理，画像拡大処理を複数回繰り返すことで，1 枚の入力画像から，物体境界を抽出した異なる解像度の出力画像を複数得ることができる．

IIR フィルタ処理は，ある時刻に入力された信号の影響が，無限先の出力信号に影響するようなシステムである無限インパルス応答システムを応用して構築された画像フィルタ処理である．そのため，ある IIR フィルタ処理の出力結果は，それ以前の IIR フィルタ処理の出力結果の影響を受けている．また IIR フィルタ処理はラプラシアンピラミッド生成処理の直後の処理であり，ラプラシアンピラミッドで生成された物体境界を含む画像から，微振動のみを含んだ物体境界を抽出するために利用される．オイラー動画像誇張処理ではユーザは微小変化に応じて IIR フィルタの設定が可能であり，増幅対象とする微小変化を選択することができる．

図 1 ベースライン設計における IP とその入出力（1 データが 8-bit の場合）

2.2 処理フロー

オイラー動画像誇張処理は入力動画からフレームを 1 枚読み込み，複数の画像フィルタ処理を施し増幅フレームとして出力する．各画像フィルタ処理について説明する．はじめに，ラプラシアンピラミッドの生成を行い，微小変化を含んだ物体境界のみを抽出する．そしてラプラシアンピラミッド各段の画像に IIR フィルタ処理を施し物体境界に含まれた微小変化を抽出する．次に変化量増幅処理により微小変化を視覚的に誇張する加工を施す．変化量増幅画像生成処理では，ピラミッド状の画像データから拡大と画像の足し合わせを繰り返し行い，1 枚の画像を生成する．そして画像再構成処理により，元画像を足し合わせ増幅フレームとする．そして増幅フレームを動画化することで微小変化を誇張した動画を生成する．

3. CPU–FPGA ハイブリッドシステム設計

3.1 ベースライン設計

本節では先行研究 [8] で作成した Ultra96V2 評価ボードにおける CPU–FPGA ハイブリッドシステム設計について説明する．先行研究では，汎用プロセッサを用いたソフトウェア処理と FPGA によるハードウェア処理に分割して構築するハードウェア–ソフトウェアパーティショニング [6] に基づき，オイラー動画像誇張処理を構成する各処理を Halide コンパイラ FPGA バックエンド [1] を利用してハードウェア化し，ソフトウェア処理と組み合わせた設計を行った．Halide コンパイラ FPGA バックエンドは Halide プログラム [4] から高位合成向けの C++ プログラムを生成するツールである．プログラム記述に用いる Halide は画像処理に特化したドメイン特化言語であり，アルゴリズム記述とスケジュール記述を分離して記述する特徴を持つ．

ラプラシアンピラミッド生成処理はダウンサンプリング処理とアップサンプリング処理で構築され，それぞれを独立した IP として設計した．IIR フィルタ処理は前半部と後半部に処理を分け，前半を IIR フィルタ前処理，後半を変化量増幅処理と合わせて IIR フィルタ増幅処理として設計した．変化量増幅画像処理はアップサンプリング処理を再利用する．画像再構成処理は後段の画素値変換処理と合わせて画像再構成+画素値変換処理として設計した．

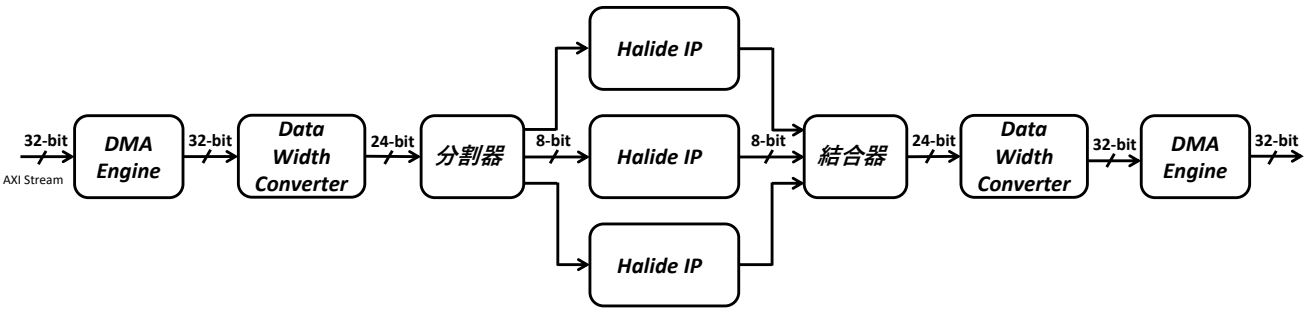


図 2 高データ並列設計における IP とその入出力（1 データが 8-bit の場合）

図 1 に IP の入出力の概要を示す．それぞれの IP はデータ入出力に AXI stream インタフェースを用い，毎サイクル演算結果が出力されるようパイプライン化した．IP への入出力は，FPGA に実装される DMA エンジンにより行われる．

ベースライン設計で IP 化した処理が入出力するデータは全て画像フレームであり，チャネル，横，縦の 3 次元データである．それぞれの処理は，チャネルに対して全て独立な処理であり，データ並列性を持つ．Halide コンパイラ FPGA バックエンドは `hls_burst()` や `unroll()` 指示によりデータ並列性のある処理を空間的に展開し，入出力のバス幅や演算器の数を増やすことでスループットを向上することができる．しかしながら，本執筆時に利用可能なバージョンでは指定可能な値が 2 冪値に限定されており，3 チャネルのカラー画像に対して適用できない．そのため，ベースライン設計における各 IP は毎サイクル 1 画素の 1 チャネル分を処理するよう作られている．したがって，各 IP の処理時間は（チャネル数×縦×横＋パイプライン段数）のサイクル数である．

3.2 高データ並列設計

CPU–FPGA ハイブリッドシステムを設計する際に重要となるのは，CPU–FPGA 間のデータ通信を可能な限り高速化することである [2][3]．ベースライン設計では，画像データの 1 画素 1 チャネル分，すなわち，データ型に応じてサイクル当たり 8-bit または 32-bit の配線でデータが入出力されており，このスループットがボトルネックとなっていた．アルゴリズム，設計制約の両観点で本来利用できるデータ並列性が活用されていない状況であり，さらなる性能向上のため優先的に改善する余地がある．

そこで，データ幅及び演算スループットをチャネル数分，つまり 3 倍にすることを考える．これは AXI stream を分割または結合することにより実現でき，図 2 に示すように分割器及び結合器を用いれば良い．また，Xilinx Logicore IP の DMA は入出力ストリームのデータ幅が 2 冪値に限定されるため，Data Width Converter を挿入する．

この時，各機能のアルゴリズムを担う IP はベースライン設計と同様 Halide コンパイラ FPGA バックエンドを用いて生成できる．一方で，分割器及び結合器は高位合成用 C++ プログラムを用いて作成し，高位合成により作成する．

表 1 CPU - FPGA システム環境 (Ultra96V2)

CPU	Cortex-A53 64bit Quad-core In-Order processor 1.2 GHz
メモリ	2GB LPDDR4
LUT	70,560
FF	141,120
DSP	360
BRAM (36 Kb)	216

4. CPU - GPU ハイブリッドシステム実装

本章では EVM の CPU–FPGA ハイブリッドシステム実装の有効性評価を行うために行った， CPU–GPU ハイブリッドシステム実装について述べる．実装にあたっては，EVM を構成する複数回の画像処理を GPU により実行する．これらのソフトウェア実装には OpenCV ライブラリを用いており，その GPU モジュールを利用することで容易に GPU で実行できる．このモジュールは NVIDIA 製 GPU のみをサポートしており，CUDA Runtime API を使用して実装されている．本研究で用いる Jetson Nano には NVIDIA 製の GPU が搭載されており，OpenCV の GPU モジュールを利用できる．

しかしながら，単純に GPU モジュールを連続して呼び出すと，GPU 計算の度に VRAM 確保処理が行われ性能低下するため，GPU モジュールとして提供される `cuda::Stream` クラスを利用した．`cuda::Stream` クラスは GPU 上の処理を管理するキューであり，計算順序の制御やホスト・デバイス間のデータ転送のオーバーラップを可能にする．これにより，VRAM 確保処理の回数を削減し，アプリケーション性能を改善した．

5. 評価

5.1 概要

本節では，オイラー動画像誇張処理を対象とした CPU–FPGA ハイブリッドシステムの設計の有効性を確認するた

Vol.2021-ARC-244 No.5
Vol.2021-SLDM-194 No.5
Vol.2021-EMB-56 No.5
2021/3/25

表 2　CPU - GPU システム環境 (Jetson Nano)

CPU	Cortex-A57 64bit Quad-core Out-of-Order processor 1.4 GHz
メモリ	4GB LPDDR4
GPU	Maxwell 128 CUDA core 921 MHz

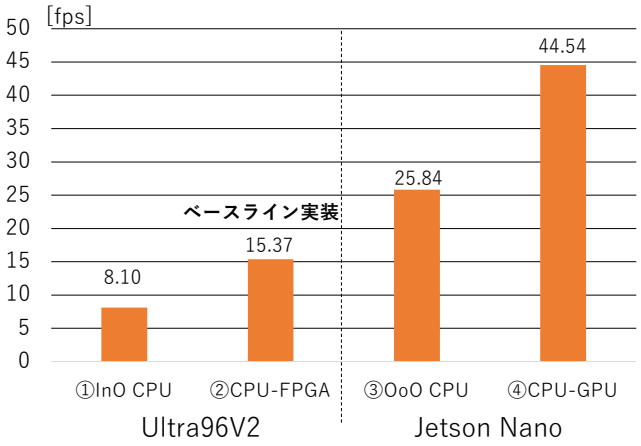


図 3　フレームレート性能評価結果

めの実装と評価を行う．先行研究 [8] で作成した Ultra96V2 評価ボードにおける CPU–FPGA システムをベースラインとし，比較対象として Jetson Nano に CPU–GPU ハイブリッドシステムを実装し，フレームレート性能と消費電力性能を比較し評価する．それぞれのハードウェア仕様を表 1 と表 2 に示す．また，消費電力計測には菊水電子の直流安定化電源 PMX18-5A を用いる．フレームレート及び消費電力の評価には入力データとして横 640，縦 480 ピクセルの画像 301 フレームからなる動画像ファイルを用いる．

5.2　フレームレート評価

図 3 にフレームレートの評価結果を示す．CPU–FPGA システム (Ultra96V2) の In-Order CPU のみを用いた場合（①InO CPU）は 8.10 FPS, FPGA による加速実行を適用した場合（②CPU-FPGA）は 15.37 FPS であった．一方，CPU–GPU システム (Jetson Nano) の Out-of-Order CPU のみを用いた場合（③OoO CPU）は 25.84 FPS, GPU による加速実行を適用した場合 44.54 FPS であった．

①に対する②のフレームレート向上比率は 1.90 倍，③に対する④のフレームレート向上比率は 1.72 倍となった．また，②と④を比較した場合， CPU–GPU システムの方がフレームレート性能が 2.90 倍高い．

CPU-GPU システムの方が FPS が高い理由として，下記複数の要因が挙げられる．

（1） ④は Out-of-Order プロセッサを採用しており CPU 性能が高い．

（2） 加速実行した処理が画像処理であり，GPU との親和性が高い．

（3） GPU の動作周波数は 921 MHz であるのに対し，FPGA は 300 MHz であり GPU のほうが動作周波数が高い．

（4） ②のベースライン実装では，データ転送幅が性能を律速している．

(4) については第 6 章で改善について検討する．

図 4 にフレーム毎の処理時間変化を示す．CPU による処理は処理時間のばらつきが大きく，CPU–FPGA システムによる処理が処理時間のばらつきが比較的小さいことが確認できる．CPU–GPU システムについては序盤の処理フレームの処理時間が，中盤以降の処理時間より長い．

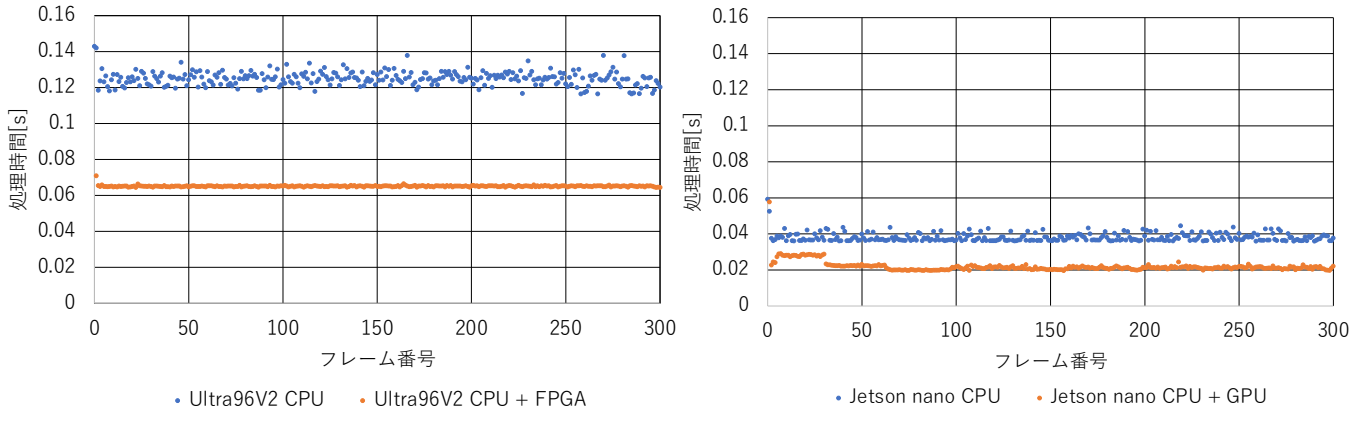
5.3　消費電力評価

図 5 にアプリケーション実行時の消費電力変化を示す．経過時間 約 10 [s] までは Ultra96V2 と Jetson Nano の両方でアプリケーションが動作していないアイドル状態にある．その後，アプリケーションが実行され処理が終了するまで消費電力が増加することが分かる．Ultra96V2 では青色で示す CPU 実行時と橙色で示す CPU-FPGA ハイブリッド実行時のアイドル状態の消費電力が異なるが，これは FPGA へのビットストリーム読み込みの有無に起因する．アプリケーション実行中の平均消費電力を算出すると，Ultra96V2 搭載 CPU によるソフトウェア処理時と，ベースライン実装による FPGA 加速実行時の平均消費電力はそれぞれ 4.225 W，5.246 W であった．また Jetson Nano 搭載 CPU によるソフトウェア処理時と，CPU–GPU ハイブリッド実装による GPU 加速実行時の平均消費電力はそれぞれ 3.660 W，3.982 W であった．ベースラインと CPU–GPU システムを比較した場合，CPU–GPU システムの方がベースラインより消費電力が 24.1% 低い．

FPGA における消費電力が大きい理由として，クロックゲーティングを適用していないことが考えられる．ベースライン実装では，6 種の処理が FPGA に実装されているが，それらは逐次的に利用されており，並列動作していない．したがって，それぞれの機能で見るとアイドルの時間が存在する．それぞれの機能に対し，アイドル時にはクロックを停止すれば，FPGA の消費電力を削減可能と考えられる．

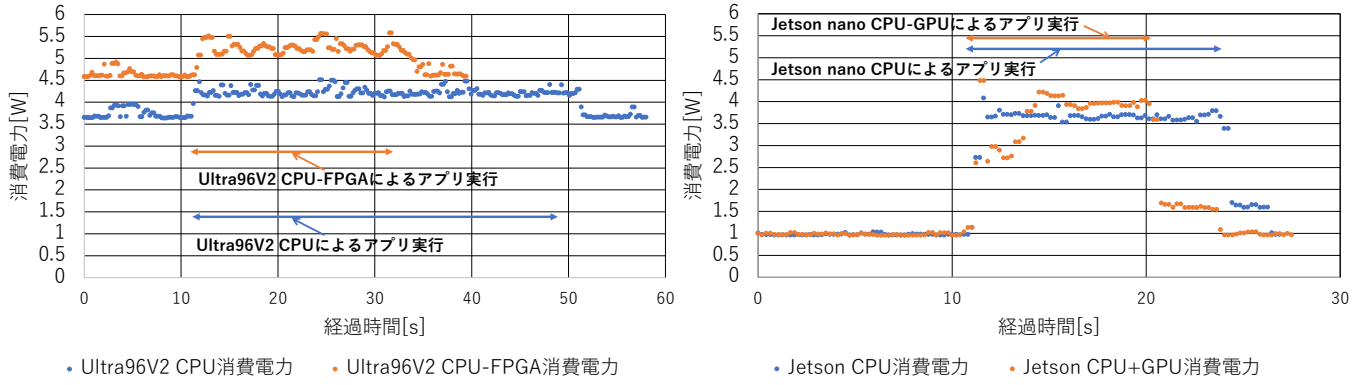
5.4　高データ並列化の性能評価

ここでは画素値変換処理について，カラー画像のチャネルのデータ並列性を活用した高データ並列化適用時の実装評価を行う．ベースラインにおける画素値変換及び，データ転送幅及び演算並列度をベースライン時の 3 倍にした画素値変換の処理時間はそれぞれ 4.32 ms，1.84 ms であった．このことから，データ幅及び演算スループットを 3 倍化した IP の性能は，ベースラインで設計された IP の性能の 2.3 倍の結果になることが期待される．この結果を元に，第 6 章で他の機能に対しても同様の手法を適用した時の効果を



(a) Ultra96V2　(b) Jetson nano

図 4　フレーム毎の処理時間変化



(a) Ultra96V2　(b) Jetson nano

図 5　アプリケーション実行時の消費電力変化

見積もる．

6.　高データ並列設計適用の検討

本章では，第 5.4 節の評価結果を元に，高並列化を適用する機能の選定及びその効果見積もりを行う．十分な回路資源が利用可能な場合には，全ての IP を高並列化することが高速化のためには望ましいが，実際には利用可能な回路資源量は有限であり，かつ，組み込み用途を考えるとより少ない資源量での実装が求められる．

そこで，6 種の IP における高並列化する優先度を考え，高並列化を適用する IP の数とその効果を見積もる．本稿では，それぞれの IP の処理時間を計測し，1 フレーム処理当たりの処理時間に占める割合の大きいものから順に高並列化するものとする．1 フレーム当たりの処理時間に占める割合の大きい IP を順に並べると，アップサンプリング処理，IIR フィルタ増幅処理，IIR フィルタ前処理，ダウンサンプリング処理，画素値変換処理，画像再構成+画素値変換処理となった．

効果の見積もりは CPU 実行と比較するものとし，加速実行による追加電力当たり加速部性能向上率を指標とする．この指標を用いることで，CPU–GPU システムとの比較が可能であると考える．本稿における見積もりでは，以下の

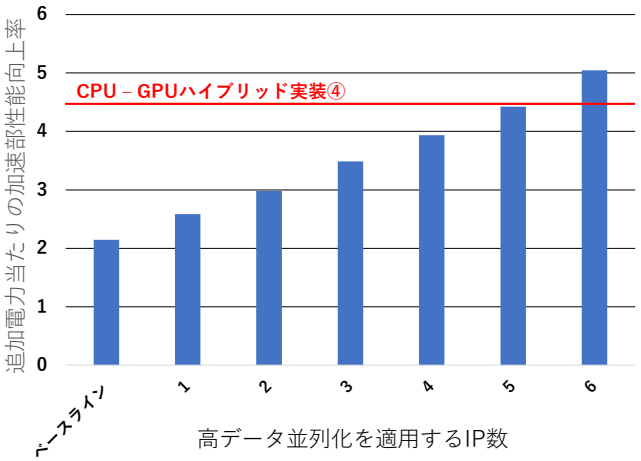


図 6　高データ並列化適用時の効果見積もり

仮定を置く。

- 追加電力は高並列化後も変化しないものとし，ベースラインの消費電力を用いる
- IP の種類，処理する画像データサイズに依らず，高並列化適用後の処理時間は 1/2.3 を乗じて算出する

見積もり結果を図 6 に示す．ベースラインに加え，高データ並列化を適用する IP の数を 1 から 6 まで変えた時

の CPU 実行と比較した追加電力当たりの加速部性能向上率を示している．CPU–GPU システムの場合を赤線で示しており，5 つの IP を高並列化することで GPU と同等，全ての IP を高並列化することで GPU より優れた加速実行効率を達成できる見込みがある．

この結果から，画像処理アプリケーションに対する CPU–FPGA システムの一定の適用可能性が認められる．しかしながら，見積もり時の仮定には非現実的なものも含まれており，さらなる検証が必要である．

また，CPU–FPGA ハイブリッドシステム設計には性能面，消費電力面で最適化の余地がある．性能面では IP 間のパイプライン化などが考えられ，消費電力面ではクロックゲーティングによる低消費電力化が考えられる．今後は高データ並列化のみならず，さらなる最適化の可能性についても検討を進めていく必要がある．

7. おわりに

本稿は，オイラー動画像誇張処理のベースライン実装のボトルネックとなる FPGA オフロード部についてデータ転送性能向上と FPGA 処理の並列度を高めた際の性能電力モデルを構築し，改善効果を明らかにすることを目的とした．そこで，既存研究のベースライン実装と CPU–GPU ハイブリッド実装の比較のため，Ultra96V2 評価ボードならびに Jetson Nano を対象にフレームレート性能と消費電力を評価した．その結果，ベースライン実装は CPU–GPU ハイブリッド実装とフレームレート性能と消費電力の点で優位性が確認できず，ベースライン実装は設計改善する必要性が明らかとなった．また，既存研究におけるベースライン実装からデータ転送性能と FPGA 処理の並列性を高める改善手法を適用することで，ベースライン時の設計から 2.3 倍の高速化を達成した．これらの結果に基づくモデリングの結果，ベースライン設計した 6 種の IP コアへ改善手法を適用すると，電力投資効果の点で GPU 利用に対して 1.129 倍の性能向上を達成できることが確認できた．今後，ベースライン設計に実際に改善手法を適用し，効果を実測する予定である．

謝辞 本研究は一部，JSPS 科研費 JP19K20235, JP20K19771 の助成による．

参考文献

[1] Ishikawa, A., Fukushima, N., Maruoka, A. and Iizuka, T.: Halide and GENESIS for Generating Domain-Specific Architecture of Guided Image Filtering, *Proceedings of the 2019 IEEE International Symposium on Circuits and Systems*, ISCAS '19, pp. 1–5 (2019).

[2] Liang Feng, Hao Liang, Sinha, S. and Wei Zhang: HeteroSim: A heterogeneous CPU-FPGA simulator, *THe 26th International Conference on Field Programmable Logic and Applications*, pp. 1–1 (online), DOI: 10.1109/FPL.2016.7577386 (2016).

[3] Matthews, O., Manocha, A., Giri, D., Orenes-Vera, M., Tureci, E., Sorensen, T., Ham, T. J., Aragon, J. L., Carloni, L. P. and Martonosi, M.: MosaicSim: A Lightweight, Modular Simulator for Heterogeneous Systems, *2020 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*, pp. 136–148 (online), DOI: 10.1109/ISPASS48437.2020.00029 (2020).

[4] Ragan-Kelley, J., Barnes, C., Adams, A., Paris, S., Durand, F. and Amarasinghe, S.: Halide: A Language and Compiler for Optimizing Parallelism, Locality, and Recomputation in Image Processing Pipelines, *Proceedings of the 34th ACM SIGPLAN Conference on Programming Language Design and Implementation*, PLDI '13, New York, NY, USA, ACM, pp. 519–530 (2013).

[5] Satyanarayanan, M.: The Emergence of Edge Computing, *Computer*, Vol. 50, No. 1, pp. 30–39 (online), DOI: 10.1109/MC.2017.9 (2017).

[6] Vahid, F.: What is Hardware/Software Partitioning?, *SIGDA Newsl.*, Vol. 39, No. 6, pp. 1–1 (2009).

[7] Wu, H.-Y., Rubinstein, M., Shih, E., Guttag, J., Durand, F. and Freeman, W. T.: Eulerian Video Magnification for Revealing Subtle Changes in the World, *ACM Transactions on Graphics (Proc. SIGGRAPH 2012)*, Vol. 31, No. 4, pp. 13–15 (2012).

[8] 上野 麟，谷本輝夫，後藤孝行，丸岡 晃，川上哲志，小野貴継，飯塚拓郎，井上弘士：オイラー動画像誇張処理を対象とした Halide を用いた FPGA 加速実行の設計と実装評価，情報処理学会研究報告，Vol. 2020-ARC-241, No. 4, pp. 1–8 (2020).