Vol.2018-MPS-117 No.14
2018/3/1

FPGA 実装された ICBI の性能評価

戸田菜津子† 石川由羽† 松本尚† 城和貴†

概要：本稿では，ICBI（Iterative Curvature Based Interpolation）の FPGA（Field Programmable Gate Array）によるハードウェア実装の性能と，ICBI 開発者によって実装された GPU 版の ICBI プログラムを組み込み用 GPU に移植した際の性能を比較する．ICBI とは，超解像アルゴリズムであり，低解像度画像を高解像度画像に画像処理するための手法の 1 つである．ICBI をハードウェア実装することによって実時間での処理を可能にするために，FPGA への実装方法が提案されている．ICBI のハードウェア化にともない，改変を行った超解像アルゴリズムの性能を確認するために，GPU へ移植を行う．FPGA と GPU の実行結果を比較し，評価を行う．

キーワード：FPGA，ICBI，GPU

Performance evaluation of ICBI mounted on FPGA

TODA NATSUKO†
YU ISHIKAWA† TAKASI MATSUMOTO† KAZUKI JOE†

1. はじめに

近年，信号処理 LSI の発達に伴いパソコンやテレビなどの画面の大型化，高解像度化，低価格化が進み，表示デバイスの性能が向上し, 1920×1080 画素の HD（High Difinition）が一般的になっている．放送コンテンツや専門的な高解像度のカメラからの映像など，録画解像度が表示デバイスより高解像である場合は，画像や映像を特に支障なく閲覧することが可能である．しかし，放送コンテンツや専門的な高解像度のカメラ以外からの映像や，内視鏡カメラや各種検査用の超小型カメラからの映像，監視カメラやロボットに搭載されている安価な小型のカメラからの映像などの低解像度のコンテンツを，より高解像度の表示デバイスで閲覧する場合には，解像度の粗さが際立って見えてしまう問題がある．この低解像度のコンテンツの画像の粗さを目立たなくするためには，実時間で画像処理を行う技術が求められている．

最も単純で短時間に処理を行う技術として，画像補間技術がある．画像補間技術とは，画像を拡大する際に，ピクセル間を補完する技術で，代表的なものとしてニアレストネイバー法，バイリニア法，バイキュービック法などがある[1]．これらの技術は比較的短時間で画像の処理を行うことが可能である. しかし, 実時間処理を行うことは難しい．

そこで，超解像アルゴリズムの研究が行われている．超解像アルゴリズムとは，解像度の低い画像を，鮮明な高解像度の画像に復元する技術である．解像度の高い画像へ復元するためには，低解像度の画像で失われている高周波成分を補間することが必要である．高周波成分とは，画素の濃度値が急激に変化している画像の輪郭部分などのことである．超過像度アルゴリズムには，イメージエンハンサー[2]，再構成超解像[3]，学習型超解像[4]，NEDI（New Edge-Directed Interpolation）[5]，FCBI（Fast Curvature Based Interpolation）[5], ICBI(Iterative Curvature Based Interpolation）[5]などがある．

超解像アルゴリズムは，計算量が多いため，ソフトウェア面の改良のみでは実行時間を短縮することは困難である. そのため, 超解像アルゴリズム専用のハードウェアとして, 安価に手に入れられ，必要に応じて何度も回路を変更できる FPGA（Field Programmable Gate Array）[6]を用いてハードウェア実装するための手法の提案[7]がされている．

本稿では, FPGA による ICBI のハードウェア実装を行った FPGA 実装の性能評価を行う．ICBI とは，超解像アルゴリズムの 1 つである．評価方法として，FPGA と同様に画像処理に優れた GPU（Graphics Processing Unit）においても ICBI を実装し，コストやパフォーマンスの点で両者を比較する．GPU は，画像処理を専門とする演算ユニットとして登場し，単純かつ膨大なデータに対し，並列処理を適用することで高速な処理を行う．監視カメラ動画等，既にインフラとして使用されているものを対象とし，如何に低いコストで実用化できるかという観点で性能評価を行う．GPU の多くは高価な製品であるが，今回は比較的安価な組み込み GPU である Jetson TX1[8][9]を使用する. Jetson 上で実行するプログラムは，ICBI 開発者によって実装された GPU 版のプログラムを組み込み用 GPU に移植したものである．

FPGA のハードウェア実装にあたって，ICBI のアルゴリズムの一部を変更している．この変更は，開発ボードである ZedBoard[11]の回路設計と使用する OV7670 カメラの色情報出力の仕様の制約によるものである．本稿では，FPGA

† 奈良女子大学
Nara Women's University

実装時の改変アルゴリズムと処理内容や順序，データを同様の条件となるように，GPU 用の ICBI プログラムを組み込み用 GPU である Jetson へ移植する際に変更を加えて，比較実験を行う．

以下，2 章では超解像アルゴリズム ICBI の説明をする．3 章では，FPGA を用いた ICBI の実装方法を述べる．4 章では，その手法に基づき，GPU を用いた ICBI の実装方法を述べる．5 章では実行結果と比較結果について述べる．

2. ICBI

超解像アルゴリズムとして，FCBI と ICBI がある．FCBI は ICBI アルゴリズムの前段階の初期値計算として用いられるもので，このアルゴリズムだけでもピクセル補間をすることができるが，ICBI ではその後さらに平滑化処理を反復的に行い，精度を向上させる．これらのアルゴリズムは，フレーム内超解像技術と呼ばれる，複数フレームではなく，1 つのフレームのみで高解像度画像を生成するアルゴリズムである．

FCBI では，まず補間したいピクセルに接する 8 つのピクセルが含まれる斜め方向を 2 つ定め，それぞれの方向に対して局所近似二次導関数を以下の式のようにしてそれぞれ求める．FCBI のイメージを図 1 に示す．図 1 の左図は，補間ピクセルに対して対角線の右下方向への勾配を求めるイメージ図である．同様に右図は，対角線の右下方向への勾配を求めるイメージ図である．ここで，二次導関数をそれぞれ図 1 左の斜め方向を$I11(2i+1,2j+1)$，図 1 右の斜め方向を$I12(2i+1,2j+1)$とおくと，それらを求める式は以下のようになる．

$$\begin{aligned} I_{11}(2i+1,2j+1) = {} & I(2i-2,2j+2)+I(2i,2j) \\ & +I(2i+2,2j-2)-3I(2i,2j+2) \\ & -3I(2i+2,2j)+I(2i,2j+4) \\ & +I(2i+2,2j+2)+I(2i+4,2j) \end{aligned}$$

$$\begin{aligned} I_{12}(2i+1,2j+1) = {} & I(2i,2j-2)+I(2i+2,2j) \\ & +I(2i+4,2j+2)-3I(2i,2j) \\ & -3I(2i+2,2j+2)+I(2i-2,2j) \\ & +I(2i,2j+2)+I(2i+2,2j+4) \end{aligned}$$

それぞれの方向のうち，局所近似二次導関数が小さい方の 2 つの近接したピクセルの輝度の平均値を求め，その値を補間するピクセルに代入する．$I_{11}(2i+1,2j+1) > I_{12}(2i+1,2j+1)$のときは，

$$\frac{I(2i,2j)+I(2i+2,2j+2)}{2}$$

を代入し，$I_{11}(2i+1,2j+1) < I_{12}(2i+1,2j+1)$の場合は，

$$\frac{I(2i+2,2j)+I(2i,2j+2)}{2}$$

を代入する．

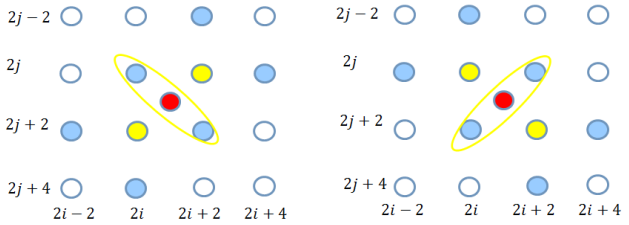

図 1: FCBI のイメージ図

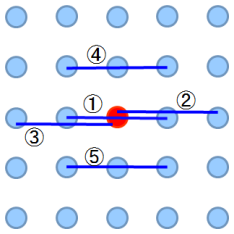

図 2: ICBI の後工程のイメージ図

ICBI の後工程のアルゴリズムでは，FCBI で補間処理を行った後に，曲率平滑化によってさらに画像を滑らかにする．ICBI の後工程のイメージを図 2 に示す．FCBI で求めたピクセル周辺の二階微分の差分の総和の絶対値を求め，FCBI で補間したピクセルの輝度値を少しずつ変更して変更前と変更後の大小を比較して小さいほうを採用する．図 2 における①～⑤を周辺ピクセルの二階微分とすると，差分の総和の絶対値は以下の式で表される．

$$|①-②|+|①-③|+|①-④|+|①-⑤|$$

ICBI も FCBI と同じく，斜め方向と縦横方向どちらに対してもこの操作を行う．この時，エッジと思われる方向は総和から除外する．

3. FPGA を用いた ICBI の実装方法

3.1 基本構成

FPGA 実装では，ICBI をハードウェア化するために，集積回路 FPGA を使用する．基本構成の要素を図 3 に表す．構成要素は，Camera モジュール，ZedBoard，ディスプレイである．FPGA は，あらかじめ集積されている論理回路の組み合わせや配線を，設計者が変更することで，独自の論理回路を構築することができる集積回路である．フリップフロップ回路を記憶部として使用した SRAM という半導体メモリを用いることで，設計した回路を変更することが可能になっている．

FPGA 実装におけるデータの一連の流れを図 3 に沿って説明する．まず，カメラから出力されたラスタデータを Dual Port RAM（frame buffer）で一時的に保存して，カメラとディスプレイの非同期信号の同期合わせを行う．次に，画像処理パイプライン（Image Processing Pipeline）で画像処理をする．画像処理パイプラインには，ピクセル補間パイプライン（Interpolation Pipeline）が含まれている．なお，

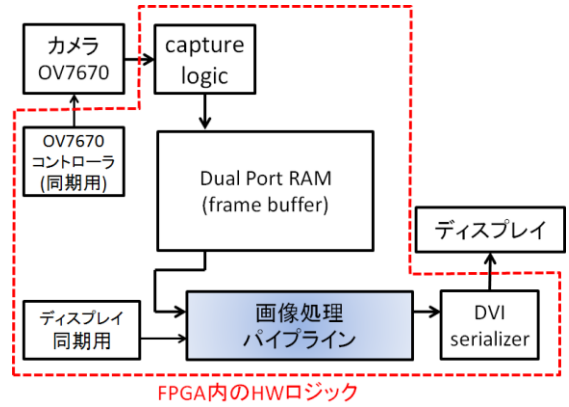

図 3: FPGA の基本構成要素

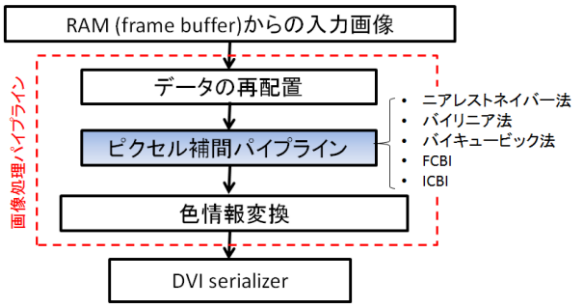

図 4: データ入力から色情報変換までの一連の流れ

画像処理パイプラインが対象とする処理は，ニアレストネイバー法，バイリニア法，バイキュービック法，FCBI，ICBI である．ピクセル補間パイプラインの詳細については，3.2 節で説明する．FPGA 実装にあたり画像処理パイプラインの改変を行っている．画像処理パイプラインの詳細と改変内容については，3.3 節で説明する．最後に，超解像を行った画像データをディスプレイに出力する．入力を行うカメラ，出力を行うディスプレイの詳細については，3.4 節で説明を行う．

3.2 ピクセル補間パイプラインの構成

データ入力から色情報変換（Color conversion）までのイメージを図 4 に表す．入力によって得る色情報は，YUV422 型を使用する．

YUV422 型のイメージ図を図 5 に表す．U，V（色情報）は 2 ピクセルあたりに 8bit×1 個のみの情報が格納されているフォーマットである．そのため，パイプラインの前段階として，データの配置（Data Arrangement）を行う．具体的には，縦横 4 ピクセルに対して，輝度値についてはそれぞれのピクセルの値を代入し，色情報についてはすべて同じ値を代入する．

ピクセル補間パイプラインは画像処理パイプラインの一部である．FCBI は，斜め方向と横方向それぞれの方向から 2 回に分けて補間するピクセルの輝度値を算出するため，2 回目の計算は 1 回目で計算するピクセルの値がすべて求

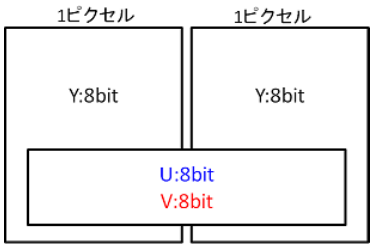

図 5: YUV422 型のイメージ図

まった後に行う必要がある．また，ICBI は FCBI 適用後に曲率の平滑化の計算を行う．そのため，1 回目の FCBI と曲率の平滑化の後に，2 回目の FCBI アルゴリズムおよび曲率の平滑化を行う．しかし，前述のように，1 回目と 2 回目の FCBI には依存関係が存在するため，逐次的に処理をする必要がある．そのため，ICBI をパイプライン処理に入れることができない．そこで，ICBI アルゴリズムを 1 回ずつ行うのではなく，2 回で 1 つの手順となるように改変を行う．具体的には，まず FCBI を 2 回連続で行い，FCBI 適用後に 2 回連続で曲率平滑化を求める．この改変では，曲率平滑化の部分がパイプライン回路を複数段接続することが必要とされるため，各段のハードウェア量が大きくなる．ICBI を FCBI と曲率平滑化に分けて処理するため，平滑化処理の収束性が悪化する可能性がある．したがって，FPGA 実装では 2 つの処理に分けてハードウェア実装を行っている．曲率平滑化の計算処理は 1 クロックで処理することができない．このため，曲率平滑化の処理に関しては 2 クロックで実行することにする．これにより，平滑化回路において WNS（Worst Negative Slack）が発生することはなくなる．WNS は，1 クロックでの計算量が多く，次のクロックまでに計算処理が間に合わないときに発生するものである．また，曲率平滑化の処理パイプラインは 8 回路まで接続して LSI CAD による配置配線が可能である．

3.3 画像処理パイプラインの構成

FPGA を利用して，超解像アルゴリズム ICBI を実行するためのハードウェア開発を行っている．ボードは，ザイリンクス ZedBoard（Zynq-7000 All Programmable SoC 搭載）という低コストの開発ボードを使用する．カメラは，OV7670 という VGA 形式のカラー映像を出力可能である，広く利用されている安価なカメラを用いる．自動ホワイトバランスや自動露出などの機能が組み込まれており，内臓レジスタの設定を変更することで，カメラの動作や機能を大幅に制御できるようになっている．出力は 1 ピクセル当たり 16bit であり，フォーマットは RGB 系，YUV 系，Raw 出力である Bayer 系を選択可能となっている．ディスプレイ出力フォーマットは 1280×800 付近の解像度で，出来るかぎりドット周波数の低い規格を採用する．これらを考慮し，ディスプレイの出力フォーマットとして，1280×768

@60Hz 15:9，dot clock：68MHz Progressive を採用して開発を行っている．

まず，パイプラインの段数による影響の確認を行っている．このために，4，7，8 回路それぞれでカスケード接続する．ICBI の平滑化処理を実装するにあたって，7，8 回路をカスケード接続した設計を作動させた場合，出力にノイズが発生してしまい，安定して動作しているとは言えない．これはパイプライン段数が大きすぎることが原因であると考えられる．回路数が多いときには安定して動作しないことから，ICBI において平滑化パイプライン回路を 4 回路程度に抑えないと ZedBoard では良好に動かないと判断する．このとき，平滑化処理パイプラインのカスケード接続の段数は，曲率平滑化の反復回数である．平滑化パイプライン回路数を4回路としてICBIアルゴリズムをソフトウェア実装して実行したところ，エッジ周辺にゴミのようなピクセルが見られる．これは FCBI アルゴリズムが発生させたゴミをICBIの曲率平滑化のパイプライン4回路では処理しきれないためである．

回路数を減少させる代わりに，ヒルクライム法の変化量の値を 2 倍強に大きくして平滑化処理の反復 1 回分の効果を増強させる．このことを FPGA 実装では ICBI ブースト版と呼んでいる．ブースト版の ICBI アルゴリズムを通常版のアルゴリズムに追加して実験を行った結果，回路数が 4 回路の場合でも良好な結果が得られている．ブースト版 ICBI 回路数 4 と通常版 ICBI 回路数 4 を比較すると，ブースト版 ICBI の解像度が高く，良好な出力結果が得られたことから，ICBI のハードウェア実装ではブースト版 ICBI のアルゴリズムを採用する．ハードウェア実装を行った結果，FCBI アルゴリズムは比較的計算量が少ないため，少ないハードウェア量で実時間処理が可能である．その一方で，ICBI アルゴリズムは曲率平滑化処理の部分がパイプライン回路を複数段接続する必要があるため，各段のハードウェア量が非常に嵩んでしまう．

3.4 カメラ出力，ディスプレイ出力

FPGA 実装では，解像度の低いカメラとして，1 ピクセル当たり 16bit のデータ量を出力するカメラを対象とする．1 ピクセル当たり 16bit であるため，RGB 系出力の最大情報量は RGB565（R:5bit，G:6bit，B:5bit）であり，YUV 系は YUV422（2 ピクセルあたり Y:8bit*2, U:8bit*1, V:8bit*1）である．FCBI アルゴリズムでは，輝度情報が重要となるため，FPGA 実装では YUV422 型の出力法が有用である．仮に，RGB の各色プレーンに対して FCBI アルゴリズムを適用した場合，細かい描写のある付近で補間する方向が各色で異なってしまい，偽色を生じる可能性がある．一方，Y（輝度）に基づいて補間する場合，色差情報をそれほど考慮しなくていいため，偽色を発生させる可能性が低い．以上の観点から，FPGA 実装では，輝度情報を多く含む YUV422 型出力を適用する．また，YUV422 型は 2 ピクセルあたりに 8bit×1 個のみ U（R），V（G）に関する情報がないフォーマットである．これは，人間の目が色差情報よりも輝度に関する解像度が高いことに由来している．UV 情報が 2 ピクセルあたり 1 個しか存在しないということは，Y の情報よりも間引かれた状態から超解像アルゴリズムを適用しなければならないことを意味している．ただし，輝度値で保存された値に，若干数値の異なる色差情報を与えても，視覚的に大きな影響は生じない．このことは，白黒画像では明白である．そこで，FPGA 実装では，輝度値 Y についてのみ着目し，FCBI アルゴリズムを実行している．

4. GPU を用いた改変版 ICBI の実装方法

4.1 基本構成

本稿では，FPGA 実装である改変版 ICBI に基づき，GPU への実装を行う．ICBI のソフトウェア実装には，ICBI の開発元で公開されている ICBI の CUDA 実装版パッケージ「ICBICUDA」を使用し，Jetson への移植の際にプログラムの改変を行う．ここで，元の ICBICUDA を通常版とし，改変を行ったものを改変版と呼ぶ．改変版 ICBICUDA のプログラムの流れを以下のリストに示す．

1. 画像の読み込み
2. RGB 型から YUV422 型へデータ変換
3. 倍率化画像用のグリッド上に画像のピクセルを再配置
4. デバイス側のメモリ初期化
5. ホスト側からデバイス側へデータ転送
6. カーネルを呼び出し演算
   - 6.1 FCBI
   - 6.2 曲率平滑化による反復修正
   - 6.3 YUV422 型から TColor 型へデータ変換
7. デバイス側からホスト側へデータ転送
8. 拡大画像出力

通常版 ICBICUDA のプログラムに改変を加え，FPGA 実装の条件へ近づける．改変点は，大きく 2 か所である．1 つ目は，補間処理をする際の色情報が，RGB から算出された TColor 型ではなく，YUV422 型を使用している点である．改変版 ICBICUDA では，手順 1 の入力画像の読み込みを行った直後に RGB 型から YUV422 型への変換を行う．色情報改変点の詳細については，4.2 節で説明する．2 つ目は，ICBI の処理の一部である FCBI と曲率平滑化を行う順番を変更している点である．ICBI の処理手順の改変点の詳細については，4.3 節で説明する．また，手順 1 で用意する画像サイズは，640×384 ピクセルのものを用意する．これは，FPGA 実装時のカメラの最大出力画素の 1280×768 を 0.5

倍したもので，FPGA 実装で使用されている入力画像のサイズである．

通常版 ICBICUDA では，ICBI 処理を行う際に，色情報である RGB 型を利用している．しかし，FPGA 実装では，YUV422 型での入力を行い，補間処理後に YUV422 型から RGB 型へ再変換を行っている．これは，FPGA 実装時に使用する OV7670 カメラの入力時の色情報の仕様によるものである．そこで改変版 ICBICUDA では，画像データ読み込み後に RGB 型から YUV422 型へ変換を行い，補間処理後のデバイス側からホスト側へ画像データの転送を行った後に YUV422 型から RGB 型への変換を行う．デバイスとホスト間の転送を行う際にも，YUV422 型のデータに対応させる．GPU 上で行う輝度の勾配の計算と補間処理については，YUV422 型に対応した演算方法へ改変を行う．YUV422 型には，2 ピクセルあたりの情報量が，Y が 16bit であるのに対して，UV は，各 8bit しか存在しないため，2 ピクセルの間で同じ情報を共有することになる．3.4 節でも述べているように，人間の目にとっては，画像を見る際，色情報の中でも輝度値の情報がより高感度に知覚されている．また，ICBI は，輝度値の勾配を基にして補間元のピクセルの選択を行うため，輝度値 Y の情報量の重要度は色差情報の UV よりも大きい．以上の理由より，UV 情報は今回の補間処理に関しての影響は非常に小さいと考えられるため，ICBI で使用する色情報は輝度値 Y のみとする．RGB 型から YUV422 型への変換については，メインプログラムを含む interpolation.cpp 内で bmploader.cpp での画像読み込み処理の後に行う．RGB 型から YUV422 型への変換を行うことで，1 ピクセル辺りのデータが Y，U もしくは Y，V の輝度情報と色差情報が 1 対 1 で与えられ，2 変数で表現することができる．それに伴い，ホスト側からデバイス側へ転送するデータ量も減少し，転送可能な画像の最大サイズが大きくなると考えられる．

4.2 平滑化処理の改変点

FPGA 実装では，パイプライン段数を抑えるために，FCBI と曲率平滑化を分けて処理を行っている．具体的には，FCBI を 2 回連続で行い，FCBI 適用後に 2 回連続で曲率平滑化を求める．また，平滑化反復回数が 4 回，平滑化処理内で行われるヒルクライム法の内容が異なっている．

本稿では，改変版 ICBICUDA の流れの内，手順 6.1 と 6.2 の部分を通常版 ICBICUDA から変更をしている．まず，FPGA 実装と同様に FCBI を 2 回連続で行うようにする．通常版 ICBICUDA では，補間するピクセルの斜め方向の周辺ピクセルに対して FCBI を適用させ補間を行った後に，同じ斜め方向のピクセルに対して曲率平滑化による修正も同時に行う．上下左右のピクセルに関しても同様の処理を行う．改変版 ICBICUDA では，斜め方向の FCBI を行った後，上下左右の FCBI を連続で適用する．その後，斜め方向と上下左右の曲率平滑化処理を連続で行い，補間の精度を向上させる．入れ替えによるデータの同期ズレが起こる可能性があるため，斜め・縦横それぞれの補間終了後のタイミングで同期を行う．また，FPGA 実装に合わせて平滑化反復回数と一部処理を変更する．平滑化反復回数は ST 値で管理されている．ST 値を 4 とし，その代わりに，平滑化 1 回あたりの効果を大きくするため，ヒルクライム法の step 値を 2 倍強に変更をする．

5. 性能評価

5.1 環境

本稿で実装において使用するハードウェアは，NVIDIA 社の組み込みコンピュータ Jetson TX1 である．Jetson は，NVIDIA Maxwell 世代の GPU を搭載した省電力統合型プロセッサの Tegra X1 を基盤とした，組み込みシステム向けモジュールである．NVIDIA CUDA®コアを 256 基搭載し，演算性能は 1 モデル前の製品である Jetson TK1 のおよそ 3 倍となる 1TFLOPS を実現している．一方で，消費電力はモジュール単体で 10W と低く抑えられており，機械学習におけるエネルギー効率は Intel Core i7-6700K の 10 倍に達している．CPU は Cortex-A57 のクアッドコア構成となっている．メモリは 4GB/LPDDR4，メモリのバス幅は 64bit，メモリ帯域幅は 29.2GB/秒である．ストレージは 16GB の eMMC で，SDIO・SATA のインターフェースにも対応しており，ユーザーが自前のストレージを活用するという使い方も可能となっている．Jetson TX1 の特徴として，ボードサイズが 50×87mm と非常に小型であることと，1TFLOPS の処理能力に対して 10W 以下の消費電力でパフォーマンスを発揮できる電力効率に優れている点が挙げられる．GPU を汎用計算で利用するための統合開発環境として，NVIDIA 社が提供している CUDA[11]を使用する．CUDA のバージョンは 7.0 である．使用した OS は Ubuntu14.04LTS である．

使用されている FPGA は Zynq-7020[12]である．PL（プログラマブルロジック）セル，LUT（ルックアップテーブル），FF（フリップフロップ），ブロック RAM の値は，それぞれ 85K，53200，106400，4.9Mb である．

5.2 比較結果

実験において，FPGA は，640×384 ピクセルの映像を静止画としてカメラ OV7670 によって読み込み，ZedBoard 上で，ICBI を画像に適用し，ディスプレイで表示を行う．Jetson TX1 は，640×384 ピクセルの画像を実験データとして入力し，GPU 上で 1280×768 ピクセルの画像へ 2 倍拡大を行い，出力する．

ICBI ブースト版を実行した際の GPU（Jetson TX1）と FPGA（Zynq-7020）の比較を行い，その結果を表 1 にまと

表 1: GPU と FPGA の比較結果

	GPU	FPGA
実測時間（s）	312.462	1.08
	313.416	
価格（円）	5 万～9 万	3 万～3 万 5 千
消費電力（W）	5～10	5～10
使用ゲート数	約 14 億	約 100 万

める．

実測時間については，FPGA の場合が 1.08 秒，GPU の場合が，反復数 20 回のとき 312.428 秒，反復数 4 回の ICBI ブースト版のときは，313.416 秒となり，FPGA が GPU と比べて，約 0.3%の実行時間となる．価格については，FPGA が流通していないため，FPGA は同様の性能を持つ製品の価格を参考価格とする．GPU 製品の中では，Jetson シリーズは比較的安価であるが，FPGA はより低価格で入手することができる．消費電力については，FPGA が 5～10W，GPU が 5～10W となり，双方とも低い消費電力での演算処理を行っている．使用ゲート数については，FPGA は現在公開されているマニュアルでは，正確な数値が記載されていないため，プログラマブルロジックセルやルックアップテーブル，フリップフロップなどの数を考慮して，約 100 万とする．GPU は，トランジスタ数をゲート数とし，最大 14 億のゲートを搭載している．

全体の結果としては，ゲート数に関しては GPU の最大使用ゲート数は非常に大きい．対して FPGA は約 100 万である．しかしながら，GPU は最大ゲート数すべてを使用しているわけではない．一方で，FPGA は必要とされるゲート数などの資源量の回路配線を行うことで，ユーザーが使用用途に合わせた最大限のパフォーマンスを行うことができる．実測時間と価格に関しては，FPGA に優位性があることが言える．

6. まとめ

本稿では，ICBI の FPGA によるハードウェア実装の性能と，ICBI 開発者によって実装された GPU 版の ICBI プログラムを組み込み用 GPU に移植した際の性能を比較した．ICBI とは，超解像アルゴリズムであり，低解像度画像を高解像度画像に画像処理するための手法の 1 つである．ICBI をハードウェア実装することによって実時間での処理を可能にするために，FPGA への実装方法が提案されている．ICBI のハードウェア化にともない，改変を行った超解像アルゴリズムの性能を確認するために，組み込み用 GPU へ移植を行った．FPGA と GPU の実行結果を比較し，両者の性能を確認した．今回の対象として，監視カメラなどの低解像度の画像を実時間で高解像度化を行う際の実用化についての観点で評価を行った．

今回行った 640×384 ピクセルから 1280×768 ピクセルへの倍率化と高解像度画像の出力処理については，組み込み用 GPU は，実行時間の点から実時間処理を行うことが難しいと考えられる．対して，FPGA は問題なく実時間処理が行えることが確認できた．FPGA は，開発ボードであるその特性から，配線回路やメモリ，カメラなどシステムの多くをユーザー側で制御を行うことができるため，最低限の資源消費量で，開発を行うことができる．加えて，低消費電力かつ低価格で実装が可能であり，実時間処理にも対応する．以上の理由から，FPGA は GPU に比べて，実用化が非常に容易であり，資源量や価格を抑えながら必要な機能を持つ実装を行うことが可能である．

参考文献

[1]山本 裕 永原 正章 (2007) “デジタル制御理論による信号処理” 映像情報メディア学会誌 Vol. 61 No. 12 P 1710-1715
[2]武内 好男 尾毛谷高 (1984) “高速度テレビカメラ・VTR システムの開発” テレビジョン学会誌 Vol. 38 No. 11 P 994-1000
[3]松本 信幸 井田 孝(2010) “画像のエッジ部の自己合同性を利用した再構成型超解像” 電子情報通信学会論文誌 D Vol.J93-D No.2 pp.118-126
[4]石田 皓之 高橋 友和(2006) “携帯カメラ入力型文字認識におけるぼけやぶれに対処するための生成型学習法” 電子情報通信学会論文誌 D Vol.J89-D No.9 pp.2055-2064
[5]Andrea Giachetti Nicola Asuni (2011)“Real-Time Artifact-Free Image Upscaling”IEEE Transactions on Image Processing 20(10):2760 – 2768
[6]小林 優(2011)『FPGA ボードで学ぶ組込みシステム開発入門 ～Altera 編～』 技術評論社 384pp.
[7]松本 尚 山本 有紗 城 和貴(2016) “実時間超解像回路の試作―ICBI アルゴリズムの FPGA 実装―” 研究報告数理モデル化と問題解決（MPS）2016-MPS-109-11 p.1-4
[8]NVIDIA, “組み込みシステムの開発者キットとモジュール” http://www.nvidia.co.jp/object/embedded-systems-dev-kits-modules-jp.html, （参照 2018-02-01）
[9]マイナビニュース, “NVIDIA が組み込み向けモジュール「Jetson TX1」を国内販売”, https://news.mynavi.jp/article/20160303-tx1/, （参照 2018-02-01）
[10]Xilinx, “ZedBoard” https://japan.xilinx.com/products/boards-and-kits/1-8dyf-11.html, （参照 2018-02-01）
[11]NVIDIA, “What Is CUDA” https://blogs.nvidia.com/blog/2012/09/10/what-is-cuda-2/（参照 2018-02-01）
[12]Xilinx, “Zynq-7000 All Programmable SoC” https://japan.xilinx.com/products/silicon-devices/soc/zynq-7000.html, （参照 2018-02-01）