Vol.2018-MPS-119 No.13
2018/7/30

FPGA 実装されたリアルタイム超解像回路の改良

松本尚† 眞田麻代† 安浪涼花† 城和貴†

概要：本稿では，我々が以前に開発したリアルタイム超解像システムを将来の高機能化に備えて改良する方法について報告する．映像を ICBI（Interactive Curvature Based Interpolation）アルゴリズムを用いてリアルタイムで超解像処理を行うため, 我々は ICBI を FPGA(Field Programmable Gate Array)によってハードウェア実装したシステムを開発した．しかし,このシステムにおいてフレームバッファに使用しているブロック RAM の容量制限が厳しく，更なる機能を追加することが困難である．最近の高性能な FPGA チップは SoC(System on Chip)化が進み、高性能 CPU、高性能バス、高速 DDR インタフェースをハードマクロとして内蔵している．これらハードマクロの機能を援用することにより，この超解像システムを改良し，容量の制約を取り払い，システムのさらなる高機能化に備える．

キーワード：FPGA，ICBI，GPU

Refinement of a real-time super-resolution FPGA circuit

TAKASHI MATSUMOTO†
MAYO SANADA† SUZUKA YASUNAMI† KAZUKI JOE†

1. はじめに

近年パソコンやテレビのディスプレイなど表示デバイスの解像度が向上している一方で，小型カメラや古い映像など解像度の低いビデオデータが多く存在しており，入力デバイスやコンテンツの解像度が追い付いていないことが問題になっている．解像度の低い入力データは，解像度の高い表示デバイスで表示すると解像度の差により粗くぼやけたように表示されてしまう．この解像度の差を補うために解像度の低い画像を補間し，解像度の高い画像を生成する超解像という技術が誕生し，様々な補間アルゴリズムが考案されている．

解像度を向上するということが重要視されるようになるまでにも，同一解像度において画像を拡大するために補間技術は古くから各種開発されてきた．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]．しかし、これらの画像補間技術は補間後の画質が十分なクオリティを持っていなかった．こうした中で NEDI（New Edge-Directed Interpolation）アルゴリズムというエッジを意識して見やすく補間する超解像アルゴリズムが開発された[2]．しかし，超解像技術は防犯カメラや内視鏡カメラや検査カメラなど，社会に取り入れる際は短い処理時間で人間に見やすく拡大できることが重要視されるため，計算量が多く時間のかかる NEDI は実時間処理に不向きであった．

そこに NEDI より大幅に少ない計算量で同等の結果が得られる ICBI（Interactive Curvature Based Interpolation）アルゴリズムが開発された[3]. ICBI アルゴリズムは NEDI に比べると大幅に計算量が削減されていたが, 高性能 CPU によってソフトウェア処理する場合には、画像の大きさにもよるが、1 画像当たり数秒オーダーの処理時間が掛かり，リアルタイム処理が可能になるのにはほど遠い状況であった.

超解像アルゴリズムは，計算量が多いため，ソフトウェア面における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）を用いて, ICBI アルゴリズムをハードウェア実装するための手法を我々は提案した[4][5].

以下，2 章では我々が提案し，実際に試作した FPGA による実時間超解像ハードウェアシステムを説明する．3 章では，提案ハードウェアの問題点を示し，最近一般的になってきた SoC(System on Chip)タイプの FPGA を用いた解決方法を述べる. 4 章では, 解決方法の詳細について述べる. 5 章では改良された超解像ハードウェアの高機能化方式について議論する.

2. 改良前の実時間超解像システム

2.1 ハードウェアシステムの基本構造

実時間超解像ハードウェアを設計試作する前に，安価で資料の入手容易な OmniVision 社の OV7670[6]を搭載したカメラモジュールを使用して，解像度を縦横ともに倍に高解像度化してディスプレイに実時間表示することを開発目標に定めた. このシステムは, 入力側信号の従うクロックと、出力側信号の従うべきクロックの最低二種類のクロックを

† 奈良女子大学
Nara Women's University

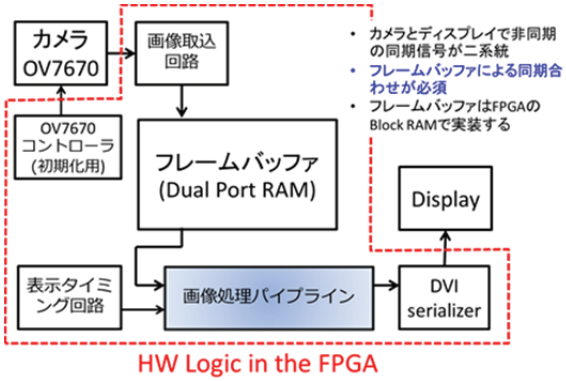

図 1 実時間超解像システムの基本構成

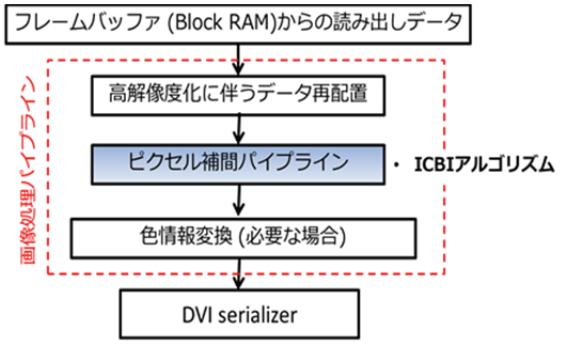

図 2 画像処理パイプラインのデータの流れ

取り扱う必要がある．また、入力のビデオ信号と出力のビデオ信号のドットクックが独立であるため，お互いの水平同期信号や垂直同期信号は互いに独立であると考えざるを得ない．このため，同一時点の入力信号の画面内の位置と出力信号の画面内の位置の間に関連がつけられないことになり，最低でも入力画像情報を 1 フレーム分溜めておけるフレームバッファがないとビデオ出力が不可能となる．

このフレームバッファの最低容量を見積もると, OV7670 はピクセル当たり 16bit のデータを出力し，解像度は 640×480 であるため，307,200×16bit の容量となる．試作に使用した FPGA 評価ボード(ZedBoard[7])に搭載された Xilinx 社 XC7Z020-1CLG484C[8]の内蔵ブロック RAM は 16bit 幅で最大 262,144 エントリしか構成できない．このシステムの設計検討段階では，FPGA 内蔵以外のメモリを使用することは考えていなかったため，OV7670 の出力のうち，262,144×16bit に納まる部分のみを使用することにした．つまり、OV7670 は 480 ラインまで使用可能ではあるが，400 ライン程度までしか使用せず，それを越える入力データは捨てることにした．ブロック RAM は読み出しと書き込みを独立させた 2 ポートメモリとして使用することができるため, 入力側クロックと出力側クロックの載せ替えも，フレームバッファにおいて行ってしまうことにした．超解像のための画像補間処理は出力クロックで動く画像処理パイプライン内において実行することにした．

超解像システムにおけるデータの流れを図 1 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不フレームバッファ(Dual Port RAM)で一時的に保存して，カメラの画像入力とディスプレイの画像出力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．

2.2 画像処理パイプラインの構成

画像処理パイプラインの構成を図 2 に示す．フレームバッファから読み出された画像情報は，縦横に解像度が 2 倍に増えることに伴って必要となるデータの再配置を行い，ピクセル補間パイプラインにおいて補間（超解像）処理を施され，RGB 形式の色情報に変換されて，DVI 出力に変換されてディスプレイに送られる．具体的な話を進めるためには，ピクセルデータの表現形式を定める必要がある．OV7670 の出力がピクセル当たり 16bit であるため，RGB 形式だと輝度情報の情報量をピクセル当たり 8bit 確保することができない．人間の視覚は色情報に対する解像能力よりも，輝度情報に対する解像能力の方が高いため，輝度情報をピクセル当たり 8bit 確保可能な YUV422 形式をピクセルデータとして採用する． YUV422 においては，各ピクセルに輝度情報(Y)が 8bit で出力され，色情報（色差情報）である U,V は 1 ピクセルおきに 8bit ずつ出力される. つまり，色情報の U,V に関しては隣り合った 2 つのピクセルで共有することになる．超解像システムのフレームバッファにおいては、前述のようにメモリ容量が逼迫しているため，R,G,B 各 8bit 等の標準的なフォーマットを採用せずに，OV7670 の輝度 8bit による出力である YUV422 形式のまま画像データを格納する. YUV422 形式を使用しているため，縦横 2 倍の解像度に上げる場合，隣り合うピクセルが高解像度化後に隣り合わなくなってしまう．このことからデータの再配置が必要になる．また，ピクセル補間パイプライン内でも YUV422 形式のまま補間処理を行うと，DVI 形式のビデオ出力はデジタル形式の RGB 各 8bit であるために、この形式に変換するために, YUV422 から RGB への色情報変換のステージが必要となる．ピクセル補間パイプラインには提案ハードウェアシステムでは ICBI アルゴリズムを改変して実装した．なお、今回試作したシステムは縦横 2 倍の解像度への変換しか行わないため, 輝度情報(Y)のみを補間して色情報はニアレストネイバー方式で対応している．

2.3 ディスプレイ出力

高解像度化前の画像データを格納するフレームバッファが 640×400 前後の大きさであるという制約があるため，ディスプレイ出力フォーマットは 1280×800 付近の解像度で出力できて，出来るかぎりドット周波数の低い規格を採用することにした．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た．

3. システムの問題点と改良方針

3.1 システムの問題点

改良前システムの試作時点では，ICBI アルゴリズムを FPGA によってハードウェア実装可能なことを実証することに重点があったため，試作システムの検証容易性とか機能拡張性といった視点は考慮されていなかった．FPGA 内部のブロック RAM でフレームバッファを構成したために，フレームバッファ上に規則正しいパターンを表示して補間が正しく機能するかどうか試すだけでも毎回パターン発生回路を設計開発する必要があった．さらに，超解像（補間）結果はディスプレイに表示されるだけなので，数値的に完全に動作が正しいかどうかの検証が困難であった．また，OV7670 は 640×480 の VGA 信号出力が可能なカメラであるにも関わらず，ブロック RAM の容量制約のため，640×400 以下で使用せざるを得なかった．今後，システムの高機能化を探る上でも，入力や補間結果を問わず複数フレームを保存できるフレームバッファの必要性を痛感した．

3.2 SoC タイプの FPGA チップ

試作に使用した ZedBoard には 512Mbyte の DDR メモリが搭載されている．DDR メモリのデータパスやコントロール部は XC7Z020-1CLG484C チップの FPGA 部である PL(Programmable Logic)部に属しているのではなく，PS(Processing System)部に属している．改良前の試作時点では，FPGA の活用によって ICBI アルゴリズムがハードウェア実装可能なことを実証しようとしていたため，PS 部に属する資源を活用することは考えていなかった．また，PS 部はハードマクロの組込みプロセッサを中心に高性能バス等で構成されているため，FPGA 設計技術以外のスキルと知識が開発に要求される懸念もあった．

しかし，近年ハードマクロの CPU や通信インタフェースや周辺回路を多数組込んだ SoC タイプの FPGA チップが多数登場し，値段も安価になって入手性もよくなってきている．例えば，Xilinx 社であれば，ZedBoard に使用されている Zynq-7000 SoC[10]シリーズや Zynq UltraScale+ MPSoC[11]シリーズなどが，intel(旧 Altera)社であれば，Cyclone V SoC[12]シリーズや Arria 10 SoC[13]シリーズが該当する．この FPGA+SoC というチップ構成を採用する流れは単なる異種回路を組合せた品種拡充のための流れではなく，本質的なものであると考えられる．なぜなら，LSI の集積度が上がって莫大な回路が 1 チップに集積可能になった今日では，使用頻度の高い回路に関してはハードマクロで予め埋め込んでしまった方が効率良い．いくらハードウェア回路の合成技術や最適配置配線技術が進歩しても，ソフトマクロの回路ではハードマクロの回路に対して動作スピードも面積の小ささも太刀打ちできない．ハードマクロ回路が不要なユーザは使わなければいいだけである．回路を使わないユーザが存在することよりもチップの対象ユーザが広がることで大量生産可能になることの方がチップの開発製造コストを抑える影響が高い．これらの理由から，SoC タイプの FPGA チップが今後は主流となって行き，組込み CPU と組込み回路で十分に性能が足りる応用にはプログラムで対応し，処理速度がプログラムでは間に合わない応用にのみ FPGA 部分に構成される回路が援用されることになると考えられる．

3.3 問題点の改良方針

SoC タイプの FPGA チップへの流れが本質的なものであると考えると，PS 部の機能を使うことを避けてシステムを構成する理由は，PS 部を使用するための知識を習得する手間以外には，なくなってしまう．ZedBoard の DDR メモリ上にフレームバッファを取ることにすれば，容量制約は実際上なくなってしまう．また，フレームバッファへの読み書きが組込み CPU から可能であるため，超解像（補間）すべき入力をカメラ以外から供給することや，変換結果を取り出して PC 等に転送することが容易に可能になると考えられる．DDR メモリ上にフレームバッファを取る場合は，データ転送スピードが組込み CPU 上のソフトウェアによって間に合うレベルではないため，書き込みも読み出しも DMA で実現する必要がある．Xilinx 社の開発支援ツールである Vivado[9]を使用することにより，DMA 転送回路のテンプレートを自動生成してくれることが判ったため，この機能を使って生成されたソースを修正して DMA 転送機能を開発することにした．

4. 改良方法の詳細

4.1 DMA 転送の仕様

試作システムで利用している XC7Z020-1CLG484C を含む Zynq-7000 SoC シリーズでは，DDR メモリと DMA 転送を行うのに AXI (Advanced eXtensible Interface)[14]を利用する．DMA 転送回路は PL 部に FPGA によって構成されるため，DDR メモリへの DMA 転送には，PS-PL 間のインタフェースである高性能 AXI ポートが使用される．Zynq チップの高性能 AXI ポート自体は DMA 転送の転送幅として 32bit/64bit が選択可能という仕様のようではあるが，ZedBoard の環境では 32bit 幅の DMA 転送回路の生成しか Vivado 上で選択できなかったため，32bit 幅の DMA 転送を使用することにした．DMA 転送におけるデータ転送レート（転送クロック）に関しても，バンド幅を考えればデータ転送クロックは高速であるに越したことはないが，ZedBoard での動作実績を考慮して 200MHz とした．Vivado

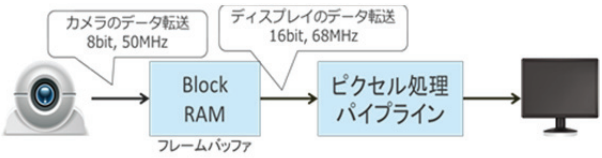

図 3 改良前システムのデータの流れ

で半自動生成される DMA 回路において, DMA 転送のバースト転送長は固定数かつ 256 転送まで設定可能であり，バースト長が長いほど転送に伴うオーバヘッドの比率を下げて転送バンド幅を大きくできる．しかし，画像のライン長単位で DMA 転送を行うため，1 回のバースト転送サイズがライン長の公約数になっていないと転送無駄によるオーバヘッドが発生する．1 ラインサイズは入力が 640×16bit の 1280byte であるため，バースト長は 64（64×32bit = 256byte）とすることにした．フレームバッファ上の画像データ表現は，将来の機能拡張に向けて DDR メモリのバンド幅に余裕を持たせるために，従来通り YUV422 形式の 1 ピクセル当たり 16bit のデータフォーマットを踏襲することにした．これにより，画像処理パイプライン以降の構成は改良前システムと同じになる．

4.2 三種のデータ転送クロック間の載せ替え

改良前のシステムは 1280×768 60Hz ドットクロック 68MHz のディスプレイ規格を使用していたが, 改良後はカメラ入力が 640×480 をフルサイズで使えるため, ディスプレイのビデオ規格に 1280×960 で表示可能でドットクロックがなるべく低い規格を採用することにした．この結果，1280×960 60Hz ドットクロック 108MHz Progressive を採用した．

改良前のシステムは，FPGA 回路内において，カメラからのデータ出力クロックとディスプレイ表示用のドットクロックの 2 種類のクロックに基づいて動いており，クロック間の載せ替えがブロック RAM によるフレームバッファで行われていた（図 3 参照）．なお、図中でカメラ出力が 8bit 幅になっているのは，カメラモジュールが YUV422 の 16bit データを 8bit ずつ 2 クロックで出力しているためである．

これに対して改良後システムでは, DDR メモリへのデータ書き込みと読み出しが 200MHz のクロックに基づいて動作を行い, データ書き込み前にカメラ出力の 8bit 幅 50MHz のデータ転送を 32bit 幅に変換して 12.5MHz の転送に変換して, その後で DDR メモリへの書き込み DMA 転送のために 200MHz クロックに載せ替えてやる必要がある．カメラのデータ出力クロックと DDR メモリの DMA 転送クロックは基本的に非同期であるため，このデータ転送に伴うクロックの載せ替えにはブロック RAM を使った dual port memory による FIFO バッファを挿入する必要がある. また、

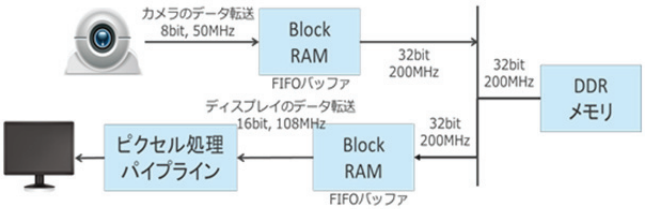

図 4 改良後システムのデータの流れ

dual port memory とは違い DDR メモリは 1 ポートで一度には一つのマスタからしかアクセスできない．カメラからのデータ入力もディスプレイへデータ出力も DDR メモリのポートの空き状態と無関係に定期的に行われるため，この意味からも FIFO バッファの挿入は必須である．FIFO バッファの容量は 1 ライン分で十分であるが，今回は 1024×32bit の FIFO バッファを使用することにした. 同様に DDR メモリからの読み出しデータ転送に関しても，ブロック RAM による FIFO メモリ（1024×32bit）を挿入することにより，DMA クロックの 200MHz からディスプレイのドットクロックの 108MHz に載せ替える．改良後システムのデータの流れを図 4 に示す.

4.3 Vivado による AXI DMA 回路の半自動生成

いろいろな文献や資料を調べていくうちに，Xilinx の設計支援ツールである Vivado によって DMA 回路が半自動生成可能であることが判り，今回はその機能（Create and Package IP）を活用して DMA 回路を開発した．ただし，少なくとも我々が使用している Vivado v2015.4.2 では AXI 側回路がテンプレートとして生成されるのみであるため，生成されたハードウェア記述言語のソースコードから読み解いてユーザ回路側のインタフェースの追加ならびにテンプレート回路記述の修正をする能力が必要とされる．また，AXI を使用する IP をブロック図に追加すると必要なリセット回路やメモリインターコネクト回路を自動生成して，AXI 関連信号の自動配線まで行ってくれる．非常にありがたい機能ではあるが，DMA 関連の問題が発生した場合には, 自分で設計していない回路は動作理解ができておらず, 却って原因追及の手間が増える要因ともなり得ると感じた.

5. 機能拡張の方向性について

本改良によって PL 部のブロック RAM から ZedBoard に搭載された DDR メモリにフレームバッファを移行したため，フレームバッファの容量制約はなくなった．これにより, カメラ入力以外の画像データを組込み CPU によってフレームバッファに描画して，補間パイプラインで処理することが可能である．補間結果をフレームバッファに書き込む DMA 回路を追加することにより，補間結果の検証も容易に行うことが可能である.

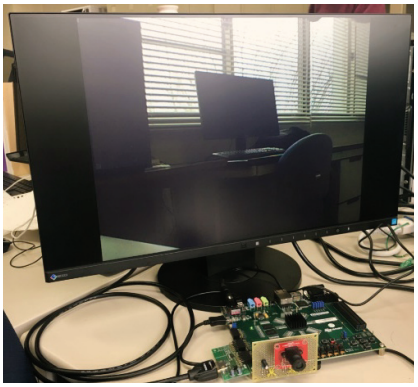
図 5 改良後のシステムによる表示

本超解像システムは VGA 解像度のカメラ入力を縦横 2 倍に解像度を向上させる機能しか現状では持ち合わせていない．防犯カメラや検査用カメラの表示装置として本システムを使用する場合には，注目個所をリアルタイムでさらに拡大表示したいという要件が望まれると思われる．この拡大処理においても，不自然なジャギーやエッジのボケが少ない ICBI アルゴリズムが適していると考えられる. 本システムを複数台カスケードして, 2 台で縦横 4 倍, 3 台で縦横 8 倍，4 台で縦横 16 倍という実現方法も考えられるが，コストも大きくなり，装置の規模も嵩む．本装置は ICBI アルゴリズムによる縦横2倍の補間を毎秒60枚の動画に対して遅れ無しに実現できる．しかし，監視カメラや検査カメラを見る人間にとって毎秒 60 枚の動画は不要である可能性が高い. そこで, 毎秒 30 枚表示の代わりに縦横 4 倍の補間，毎秒 20 枚表示の代わりに縦横 8 倍の補間，毎秒 15 枚表示の代わりに縦横 16 倍表示の拡大表示を行うシステムに拡張する方式が考えられる．補間度合が中間状態の画像をフレームバッファに書き戻し，1 フレーム分の表示期間ごとに縦横 2 倍に拡大していき，目標倍率の画像をディスプレイに出力すればよい. ディスプレイの表示は 1280×960 程度を維持するため, 4 倍以上の拡大表示の場合はカメラ入力の一部の領域のみがディスプレイに表示される．拡大対象位置はマウス等で指示できるようにする．ディスプレイのフレームレートは 60Hz で固定であるため，例えば 16 倍拡大画像の場合は, フレームバッファ上には 640×480 のカメラ入力画像, 640×480 の縦横 2 倍拡大画像, 640×480 の縦横 4 倍拡大画像，640×480 の縦横 8 倍拡大画像の 4 枚の画像が同時に存在することになる．カメラからの原画像以外は 4 フレームに一回しか更新されない.

6. おわりに

本稿では，我々が以前に開発したリアルタイム超解像システムを将来の高機能化に備えて改良する方法について報告した. 映像に対して ICBI アルゴリズムを用いてリアルタイムで超解像処理を行うため，我々は ICBI を FPGA によってハードウェア実装したシステムを開発した．しかし，このシステムにおいてフレームバッファに使用しているブロック RAM の容量制限が厳しく，更なる機能を追加することが困難であった．最近の高性能な FPGA チップは SoC 化が進み、高性能 CPU、高性能バス、高速 DDR インタフェースをハードマクロとして内蔵している．これらハードマクロの機能を援用して, DDR メモリをフレームバッファとして使用することにより, この超解像システムを改良し，容量の制約を取り払った．この改良により，見易く拡大可能な ICBI アルゴリズムを使ったリアルタイムで表示拡大率を変更可能なシステムが実現可能であることを示した．図 5 は改良前のシステムでは不可能であった 1280×960 60Hz で画像を表示しており，改良に成功したことが判る.

参考文献

[1]奥富正敏 他 “ディジタル画像処理” CG-ARTS 協会, 2004.
[2]X. Li and M. T. Orchard. “ New edge-directed interpolation” IEEE Trans. on Image Proc., 10:1521–1527, 2001.
[3]Andrea Giachetti Nicola Asuni (2011)“Real-Time Artifact-Free Image Upscaling” IEEE Transactions on Image Processing 20(10):2760 – 2768
[4]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
[5]Takashi Matsumoto, Arisa Yamamoto, Kazuki Joe (2016) “Real-Time Super Resolution: FPGA Implementation for the ICBI Algorithm” In Proceedings of 2016 International Conference on Parallel and Distributed Processing Techniques and Applications(PDPTA 2016),p.415-422
[6]EPFL, “Extension module with ov7670 CMOS camera” https://wiki.epfl.ch/prsoc/ov7670, （参照 2018-05-06）
[7]Xilinx, “ZedBoard” https://japan.xilinx.com/products/boards-and-kits/1-8dyf-11.html, （参照 2018-02-01）
[8]Xilinx, “Zynq-7000 All Programmable SoC” https://japan.xilinx.com/products/silicon-devices/soc/zynq-7000.html, （参照 2018-02-01）
[9]Xilinx, “Vivado Design Suite” https://japan.xilinx.com/products/design-tools/vivado/vivado-webpack.html (参照 2018-05-08)
[10]Xilinx, “Zynq-7000 Programmable SoC” https://japan.xilinx.com/products/silicon-devices/soc/zynq-7000.html, (参照 2018-05-08)
[11]Xilinx, “ Zynq UltraScale+ MPSoC” https://japan.xilinx.com/products/silicon-devices/soc/zynq-ultrascale-mpsoc.html (参照 2018-05-08)
[12]intel. “Cyclone® V SoC” https://www.altera.co.jp/products/soc/portfolio/cyclone-v-soc/overview.html, (参照 2018-05-08)
[13]intel, “ Arria® 10 SoC” https://www.altera.co.jp/products/soc/portfolio/arria-10-soc/overview.html (参照 2018-05-08)
[14]Xilinx, “AXI リファレンスガイド” https://japan.xilinx.com/support/documentation/ip_documentation/j_ug761_axi_reference_guide.pdf (参照 2018-05-11)