一般社団法人 電子情報通信学会
THE INSTITUTE OF ELECTRONICS,
INFORMATION AND COMMUNICATION ENGINEERS

信学技報
IEICE Technical Report

プログラマブル SoC を用いた画像処理ハードウェアの ROS 準拠コンポーネント化

山科 和史† 大川 猛† 大津 金光† 横田 隆史†

†宇都宮大学大学院工学研究科 〒321-8585 栃木県宇都宮市陽東 7-1-2
E-mail: †kazushi@virgo.is.utsunomiya-u.ac.jp

あらまし 近年，ロボットには自律性が求められ，制御ソフトウェアは高度化・複雑化している．こうしたロボットはバッテリー駆動が望ましく，高性能なマイクロプロセッサは搭載できないため処理性能が不足する問題がある．一方，FPGA を用いて特定のロボット向け処理専用のハードウェアを実現することで，ロボットシステムの性能向上を見込める．しかしソフトウェア開発に比べて FPGA の開発コストが高く，ロボットシステムへの導入は容易ではない．したがって本研究では，FPGA を容易にロボットシステムへ導入可能とするため，FPGA を用いたハードウェアのコンポーネント化を提案する．この際，ロボットシステムの開発支援のソフトウェアプラットフォームとして主流となりつつある ROS に準拠する．本稿ではプログラマブル SoC を用いた ROS 準拠コンポーネント化手法の検討を行なう．そのケーススタディとしてラベリング処理を FPGA に実装したハードウェアを，ROS に準拠したコンポーネントとして扱う事例を示す．

キーワード ROS，プログラマブル SoC，コンポーネント指向開発，FPGA，ラベリング処理

ROS Compliant Componentizing of Image Processing Hardware on a Programmable SoC

Kazushi YAMASHINA†, Takeshi OHKAWA†, Kanemitsu OOTSU† and Takashi YOKOTA†

†Graduate School of Engineering, Utsunomiya University

7-1-2 Yoto, Utsunomiya, Tochigi, 321-8585 Japan
E-mail: †kazushi@virgo.is.utsunomiya-u.ac.jp

Abstract In recent years, robots are expected to be autonomous, and their control software become complex and highly functional. The robots have a problem of processing performance shortage because they are not able to equip with a high performance processor due to battery operation. On the other hand, by realizing hardware dedicated for a robot processing using an FPGA, it is expected to improve the performance of robot systems. But, it is difficult to introduce an FPGA into a robot system since development of FPGA has high cost compared to software development. In this study, we propose componentizing of FPGA hardware in order to promote the introduction of an FPGA into a robot system. The component is compliant to ROS, which is becoming mainstream of a software platform for development assistance of a robot system. This paper reports a study of ROS compliant componentizing using programmable SoC. As a case study, an example of ROS compliant component of labeling hardware implemented an FPGA is described.

Keywords ROS, programmable SoC, component oriented development, FPGA, labeling

1. はじめに

近年，世界中でロボット構築するための設計技術が盛んに研究されている．たとえば，災害救助ロボットや無人ドローン，パートナーロボットなどのロボットは，ロボット自身が周りの環境を認識し，状況判断して動作する自律性が求められ，それらの処理のためのソフトウェアは高度化・複雑化している．こうしたロボットはバッテリー駆動で運用されることが望まれているため，低消費電力化のために高性能なマイクロプロセッサを搭載できない．一方，画像処理やネットワーク処理の分野においては FPGA (Field Programmable Gate Array) を用いることで，処理の高性能化が盛んに行われている．FPGA は開発者が任意のプログラミングによって回路の機能や仕様の変更が可能である集積回路である．また，ソフトウェアでは汎用的な逐次処理であるのに対し，FPGA を用いたハードウェアの処理では，目的に特化した並列処理や制御向けの専用回路をプログラムして実現することが可能である．しか

し，一般的に FPGA の開発においてはハードウェア記述言語による実装が必要であり，普通のソフトウェアと比べると開発が難しいという問題がある．

ロボット工学は非常に多岐にわたる専門分野の上に成り立ち，これは各開発者の能力において開発が可能な範囲をはるかに凌駕する[1]．したがってロボットシステムへ FPGA を導入するためには FPGA の開発コストを削減し，ロボットシステムへの統合を容易にできることが必要である．ロボット向けのソフトウェア開発にかかるコストを削減するためには，コンポーネント指向開発が有効である[1]．コンポーネント指向開発の一環として，ロボット用のソフトウェア開発プラットフォームである ROS (Robot Operating System) が提案されている[2]．ROS とは通信レイヤーのフレームワークとロボットソフトウェアのためのビルドシステムを提供する．

本研究では FPGA を用いたハードウェアを，ROS に準拠したコンポーネント化することによって，FPGA をロボット技術者が容易にロボットシステムへ導入できるような設計基盤を提案する．これはすなわち，ロボット開発者の開発生産性を下げることなく，FPGA によってロボットシステムの性能，その開発に貢献するということである．

本稿ではコンポーネント化のケーススタディとして，画像処理の一種であるラベリング処理を行うハードウェア回路をコンポーネント化し，プログラマブル SoC 上に実装する．また，ROS プロセス（ソフトウェア）のみによるラベリング処理と ROS + FPGA（ソフトウェア + ハードウェア）による処理を比較し，処理性能を検証する．

2. ロボット向けフレームワーク ROS

2.1. ROS の概要

ROS は OSRF (Open Source Robotics Foundation)によってオープンソースで公開されている[2]，通信レイヤーのフレームワークとロボットソフトウェアのためのビルドシステムを提供するソフトウェアプラットフォームである．主な動作プラットフォームは Linux である．ROS の目的はロボット工学分野の研究・開発におけるソフトウェアの“再利用”を支援するとともに，処理や制御のソフトウェアコンポーネントによってシステムの構築をすることである．図 1 に ROS の公式サイト[2]に登録されているソフトウェアパッケージ数の推移を示す．ROS の初期リリースをした 2007 年からパッケージ数は急激に伸びている．表 1 に示したロボット向けソフトウェアプラットフォーム別のパッケージ数と論文引用数を比較しても，ROS はロボット向けのソフトウェアプラットフォームの中でも主流となりつつあることがわかる．

表 1　ロボット向け設計プラットフォームの登録パッケージと論文引用の数(2015 年 5 月現在)

プラットフォーム名	パッケージ数	論文引用数*
ROS	3699	4610
RT-middleware [3]	321	1090
OROCOS [4]	不明	1540

*google scholar 2015/05/27

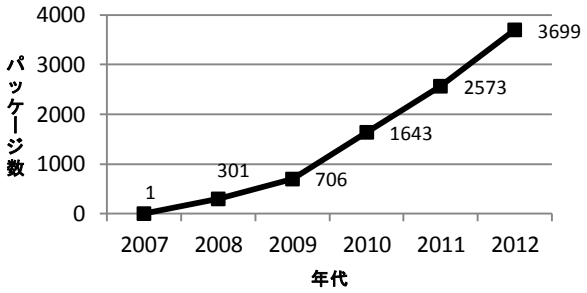

図 1　ROS のソフトウェアパッケージ数の推移

2.2. ROS のシステムモデル

ROS 上でのシステム開発では，機能や処理のまとまりごとに ROS プロセスをつくる．この際，各 ROS プロセスがどのような処理を行うか，またどのような機能を持たせるかはユーザに委ねられる．この ROS プロセスを複数作成し，プロセス同士で通信をすることでロボットの処理や制御を行う．

ROS における通信モデルは Publish (配信) / Subscribe (購読)通信モデルである．Publish / Subscribe 通信モデル(図 2)は，プロセスとプロセス同士がトピックを介して Publish(配信) / Subscribe(購読) を行う非同期メッセージングモデルである．Publish / Subscribe 通信モデルの最大の利点は，サーバ・クライアント通信モデルと比べて，通信対象となるプロセス同士の結合性が低いため，動的なネットワーク構成が可能となることである．すなわち，必要に応じて，「システムへの機能の追加＝ROS プロセスの導入」が容易に行える点である．

プロセスには 2 つの役割が存在し，Publisher(配信者)と Subscriber(購読者)があり，Publisher はトピックへメッセージの配信を行い，Subscriber はトピックからメッセージの内容を購読する．またトピックとは，各プロセスが Publish/Subscribe するメッセージの内容を区別するための名前であると同時に，メッセージの通信経路の名前である．トピックの中にはトピック自身の名前に属するメッセージのみが存在する．各プロセスは自分のプロセスに関連したメッセージが存在するトピックへの Publish / Subscribe を行う．このような Publisher と Subscriber となる ROS プロセスがそれぞれトピックに対して Publish/Subscribe することによって ROS プロセス同士のデータのやり取りを行う．メッセージの Publisher(配信者)は特定の Subscriber(購読

者) を想定せずにメッセージをトピックに配信するため, 各プロセス同士は通信相手の情報は持たない(持つ必要がない). このように ROS プロセス同士は結合性が低く, 動的なネットワーク構成が可能である.

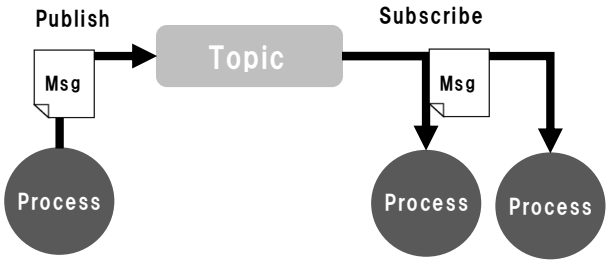

図 2 Publish / Subscribe 通信モデル

3. ROS 準拠の FPGA コンポーネント

本章では, ROS 準拠の FPGA コンポーネントを実現する上での要求, 実装すべき機能, また実装するハードウェアプラットフォームについて説明する.

3.1. ROS 準拠コンポーネントの要求

本研究における ROS 準拠とは, ROS における Publish / Subscribe 通信モデルのメッセージ形式に対応し, ほかの ROS プロセスと相互に接続可能であるということと定義する. ROS 準拠コンポーネントを実現する上で求められる要求を以下の 2 点にまとめた.

- ROS 準拠コンポーネントによる処理は ROS プロセスのみで処理した場合と等しい処理結果であること
- ROS で使用するメッセージ形式と FPGA で使用するデータ形式が統一されていること

ROS で構成されたシステムに FPGA を導入するには, ソフトウェアのみで構成された ROS プロセスと交換した際に, 機能的に等価であることが第一条件である. そのため, ROS 準拠コンポーネントは ROS プロセスと FPGA 間の通信において, FPGA で使用するデータ形式と ROS プロセスで使用しているメッセージにおけるデータ形式の統一を行う必要がある. これらの要求を満たしつつ FPGA のハードウェア回路による処理性能の向上も満たすシステムを目指す.

3.2. ROS 準拠コンポーネントの機能

実現する ROS 準拠コンポーネントのモデルを図 3 に示す. 前節で示した要求を元に実装すべき機能は以下の 4 点である.

- FPGA を用いた処理
- FPGA と ROS プロセス間の通信を行うインターフェイスとなる ROS プロセス
- 入力データの用意を行う ROS プロセス
- 出力データの配信(他プロセスへ) を行う ROS プロセス

FPGA においてハードウェア化の対象となる処理は, たとえば知的画像処理などの大量の計算を必要とする処理である. コンポーネント化する上で実装するソフトウェアは大きく分けて 2 種類に分けられる. 1 種類目は, コンポーネントへの入力データを取得するプロセスと, コンポーネント外へ処理結果の出力を行なう 2 つの入出力用のプロセスである. 2 種類目は入出力プロセスと FPGA とのデータの整合と通信を担うインターフェイスとなるプロセスである. 以上の構成にすることで, ハードウェア(HW)の処理とソフトウェア(SW)の処理が混在する HW / SW 協調処理を行うシステムが構築可能となる.

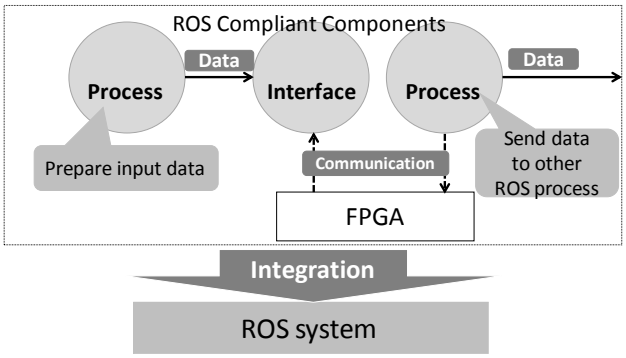

図 3 ROS 準拠コンポーネントのモデル

3.3. プログラマブル SoC の選択

ROS 準拠コンポーネントの処理と, 従来の ROS プロセスのみによる処理の最大の違いは, ROS プロセスと FPGA 上に実装した回路が混在することである.

ROS で構成されたシステムにおけるコンポーネントの導入には, Publish/Subscribe 通信が必要である. ROS 準拠コンポーネントでは FPGA を用いたハードウェアによる処理を HW/SW 混在のコンポーネントにすることで, ソフトウェアのみの ROS プロセスと同等に Publish/Subscribe 通信モデルを行い, ROS システムへの導入を容易にする.

そこで, ROS 準拠コンポーネントを実装するハードウェアプラットフォームとして, プログラマブル SoC を用いる. プログラマブル SoC 上に実装する ROS 準拠コンポーネントの処理の割り当てを図 4 に示した. SoC (System on Chip)とは 1 枚の半導体チップにシステムに必要な一連の機能を集積する方式である. 一般的に, FPGA によるハードウェア処理は低消費電力かつ, 高速であるという特徴がある. 先行研究[5]ではロボットの処理を, FPGA を用いたハードウェア回路による省電力化と処理性能の向上に試みている. 本研究では SoC の中でも ARM プロセッサと FPGA を同時に集積したプログラマブル SoC を用いる. ARM プロセッサでは Linux + ROS が動作し, FPGA ではロボット向けの処理を行う. ソフトウェアに適した処理はソフトウ

ェアに，ハードウェアに適した処理はハードウェアに割り当てて，ハードウェア開発の負担を最小限にすることで，コンポーネントの開発効率を保ったまま，処理性能の向上を図る．

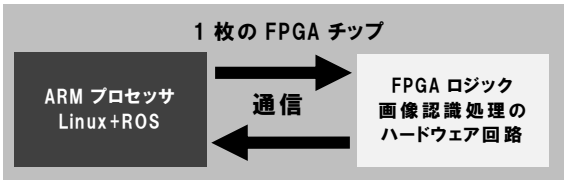

図 4　プログラマブル SoC 上の ROS 準拠コンポーネント

4. 実装事例：ラベリング処理を行う ROS 準拠コンポーネント

前章で説明した ROS 準拠コンポーネントのモデルに基づいてラベリング処理を行うコンポーネントを実装した．本章では，その詳細について説明する．

4.1. ラベリング処理

ラベリング処理とは二値化画像に対して，連続した白色の画素の集合体にラベル番号を付与する処理である．処理の用途としては，同一の番号の集合体の画素数や幅，高さなどの特徴量を求め，欠陥検査や分類処理などに用いることが出来る．二値化した画像に対してラベリング処理を施した様子を図 5 に示す．

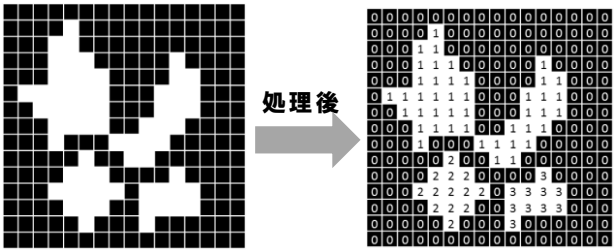

図 5　ラベリング処理の様子

4.2. ハードウェア実装の概要

ハードウェア全体図を図 7 に示す．処理のターゲットとする画像サイズは 1920×1080(幅×高さ)の 2Mbyte 程度の画像とした．この際， FPGA 上の BRAM には，画像 1 枚分のデータを保存できないで，画像の 1 ラインごとに処理を行う．性能向上のため 1 画素あたりのラベリング処理を 1 クロックで行えるように設計した．

4.2.1. FPGA－ARM 間の通信

FPGA と ARM プロセッサとの通信については，Xillybus 社[6]が無償で公開している，Zynq 向けの開発プラットフォームである Xillinux 試用版を使用した．Xillinux は Zynq の ARM プロセッサ上で動作する Linux(Ubuntu)である．Zynq 上で動作する Linux では，FPGA のハードウェア回路がデバイスファイルとしてアクセスできる．その通信の仕組みを図 6 に示す．FPGA と ARM 間の通信は ARM 上のソフトウェアからデバイスドライバを通じて FIFO に書かれたデータを，FPGA(ラベリングの回路)側から rd_en を制御して任意のタイミングで読み込む．

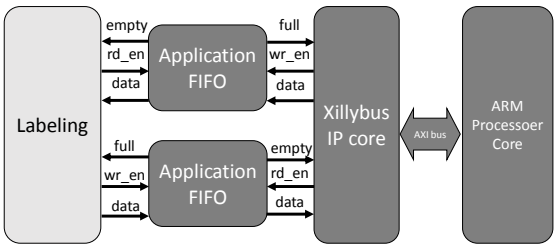

図 6　Xillinux における通信の仕組み

4.2.2. 各モジュールの機能

ラベリング処理の回路を構成する要素とその機能を表 2 に示す．回路を構成する要素は大きく分けて 5 つある．

memory_img，label_data0，label_data1 は処理に必要な BRAM モジュールである．回路はまず画像データ 1 ライン分を memory_img に格納し，格納が終わったと同時に label_generator へ画像の画素データを入力する．画素データともに，ラベリング処理では前ラインのラベル番号を必要とするため，ラベリング処理結果を格納する BRAM と前ライン分の結果を読み込むための BRAM として label_data0，label_data1 を用意した．すなわち，label_data0 の前ラインのラベリング処理結果を読み出しつつ，label_data1 に現ラインの処理結果を書き込む．また，次ラインでは label_data1 を読み出し，label_data0 に書き込む．

FIFO buffer は FPGA と ARM 間の通信を担う．FIFO buffer を入力と出力に用いることで，入出力時は回路の任意のタイミングでデータの送受信ができる．

なお，label_generator の詳細な動作は次項で説明する．

表 2　各モジュールの機能

モジュール名	機能
memory_img	入力画像データを 1 ライン分保存
label_generator	1 画素のラベリング処理
label_data0	ラベリング番号を 1 ライン分保存
label_data1	ラベリング番号を 1 ライン分保存
FIFO buffer	データの入出力バッファ

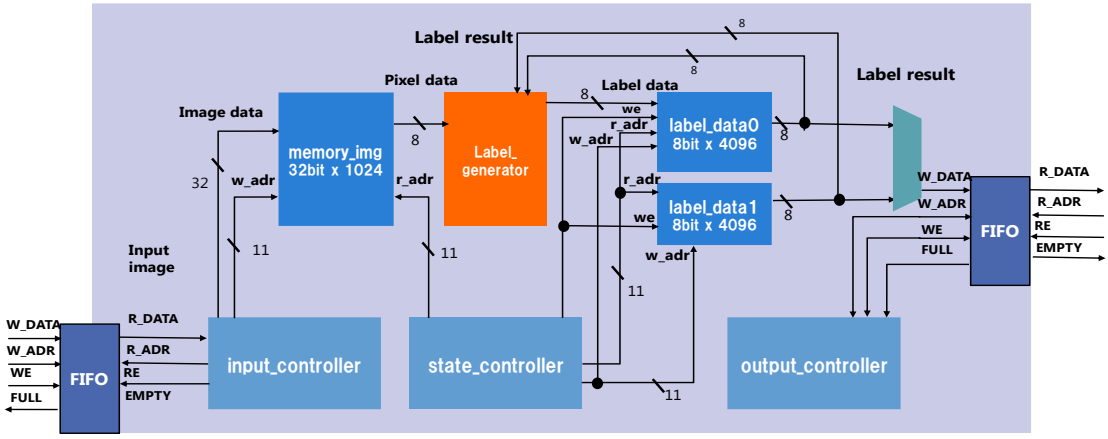

図 7　ラベリング処理を 1 ライン分行う回路のブロック図

4.2.3. ラベリング回路(label_generator)の詳細

FPGA を用いてラベリング処理を最適に行うため，1 画素あたりの処理を 1 クロックで行う方針で回路を設計した． label_generator のブロック図を図 8 に示す．入力は 5 つあり，いずれも 8 bit である．またその内訳は入力画像のピクセルデータのポートと 1 つ，ラベリング処理に必要な過去のラベル番号用のポート 4 つである．白色の画素に対して，参照した 4 つのラベル番号がすべて 0 ならば，直前につけたラベル番号＋1 の値をラベル番号として出力する．この際，参照した 4 つのラベル番号に 0 以外の数字があった場合，その中で 0 以外の最小値をラベル番号として出力する．なお，出力ポートも 8 bit である．

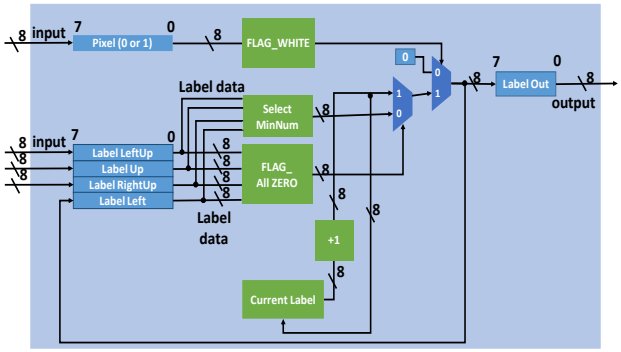

図 8　label_generator のブロック図

4.3. ROS プロセス(ソフトウェア) 実装の概要

実装した ROS 準拠コンポーネントのシステムの全体像を図 9 に示す．また ROS 準拠コンポーネントを構成する ROS プロセスを以下にまとめた．

- input_image：入力画像データを data_input に配信
- write2fpga：data_input から購読したデータを FPGA に送信
- read4fpga：FPGA から受信したデータを data_output に配信

ROS プロセスは 3 つあり，1 つ目の input_image は入力画像(bmp イメージ)を読み込み，画素データの取得を行い，トピック data_input へ配信を行なう．また，FPGA と ROS プロセス間のインターフェイスとなるのが，write2fpga，read4fpga である．たとえば，FPGA への入力用の ROS プロセスである write2fpga は，FPGA の FIFO をデバイスファイルとしてアクセスし，データを送信する．read4fpga も同様に読み込むデバイスファイルを FIFO に指定し，データを取得する．write2fpga で購読した ROS メッセージは，画像データとして FPGA へ送信する．ラベリング処理後，read4fpga で FPGA から受け取ったラベル番号を，同様の ROS メッセージのデータ形式にしてトピック data_input に配信する．他の ROS プロセスラベリング処理の結果を必要とする際は，トピック data_output を購読すればよい．

ROS におけるプロセス間通信のメッセージ形式は，図 10 に示すようなメッセージファイルにデータ形式を宣言することによって定義する．本研究では画像データを格納するための 32bit 整数配列と，フレーム番号を格納するための 32bit 整数型変数で構成した．

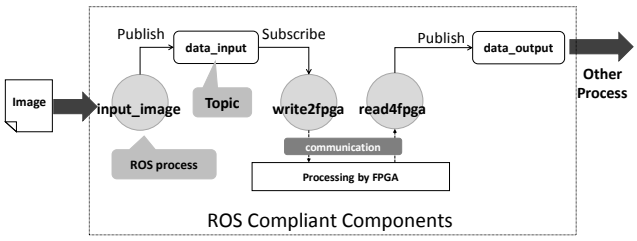

図 9　ROS 準拠コンポーネントのシステム全体

```
int32 value

int32[] data
```

図 10　メッセージファイルの内容

5. 性能評価

ROS 準拠コンポーネントの性能評価を行なった．評価は 3 つの条件で行った．

(1) ROS 準拠コンポーネント(ARM+FPGA)

(2) ソフトウェアのみによる処理(SW only:ARM)

(3) PC 環境上(SW only:PC)

(1)および(2)における評価環境は Xilinx 社製 ZC7Z020 搭載 Zedboard 上の ARM Cortex-A9 プロセッサ(666MHz)，OS は Ubuntu 12.04 LTS である．また(3)の PC 環境は Intel Core i7 870(2.93GHz), OS は Ubuntu 12.10 である．

図 11 にラベリング処理時間の評価結果を示す．測定は 10 フレーム行い，その平均値を示した．入力画像の解像度は 1920×1080 である．(1)は 1 フレームあたり 4.5 ms(ARM-FPGA 間の通信時間を含む)であり，もっとも時間が短いことがわかる．この際，処理時間の最大値は 6.0 ms，最小値は 2.0 ms であった．すなわち FPGA を用いた最適な専用ハードウェアの実装により，ARM 上でのソフトウェアのみの処理時間と比べ約 185 倍，PC 環境上と比べても約 16 倍の高速化に成功した．

また ROS 準拠コンポーネントとソフトウェアのみによる処理での全体の経過時間を図 12 に示す．図 12 中の凡例と番号が表す時間は以下のとおりである．

1: ROS のプロセス間のデータ通信時間
2: データを購読後，ラベリング処理を行うまでの時間
3: ラベリング処理の時間(通信時間含む)
4: ラベリング処理終了後，他の ROS プロセスへ結果を出力するまでの時間
5: ROS のプロセス間のデータ通信時間

(1)の処理の経過時間は，1.871 s であり，(2)の処理と比べ，約 1.7 倍の高速化であった．(1)，(2)における経過時間の多くを占めるのは ROS プロセス間における通信時間である．また，(3)の経過時間において ROS プロセス間通信が占める時間は，(1)，(2)に比べて大幅に短い．ラベリング処理に要する処理時間の構成比としては，どの環境下においても同様である．そのうえで(1)，(2)の ROS プロセス間通信に大きな時間がかかる原因として，PC のプロセッサと ARM プロセッサの性能の差があげられる．(1)，(2)では動作周波数は 666MHz の ARM プロセッサ上であるが，(3)における環境ではプロセッサは動作周波数が 2.93GHz である．このプロセッサの性能差が ROS プロセス間における通信に大きく影響したと考えられる．

以上のことから，プログラマブル SoC 上における ROS 準拠コンポーネントでは，ハードウェア化したラベリング処理の性能向上が処理全体の性能向上に大きく貢献していることがわかる．

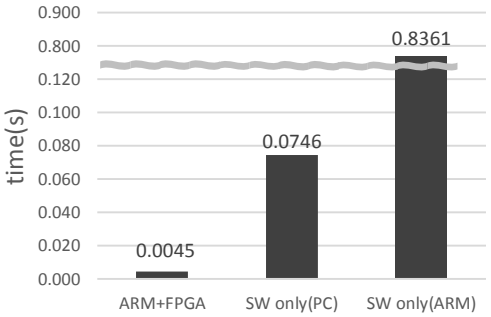

図 11　ラベリング処理時間比較

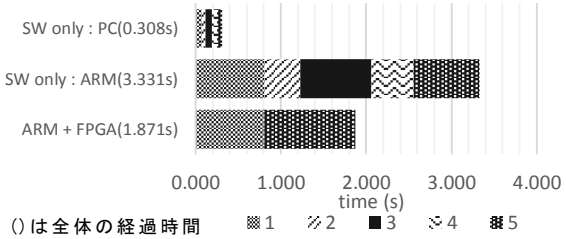

図 12　ROS 準拠コンポーネントの経過時間比較

6. おわりに

本稿では，プログラマブル SoC を用いた画像処理ハードウェアの ROS 準拠コンポーネント化について述べた．ROS 準拠コンポーネントの開発は ARM プロセッサと FPGA が同時に搭載されたプログラマブル SoC に実装し，ソフトウェアとハードウェアの協調処理を行うことによって，開発生産性を維持しつつ処理性能向上が可能であることを示した．

ROS 準拠コンポーネントの処理時間の多くは ROS プロセス間の通信時間が占める．今後，プログラマブル SoC 上で ROS プロセス間通信の時間を削減するための検討が必要である．

謝辞

本研究は，一部 JSPS 科研費 (基盤研究(C)24500055，同(C)15K00068，同(C)25330055，若手研究(B)25730026) の援助による．

文　献

[1] 日本ロボット学会(編)：“ロボットテクノロジー”，オーム社, 2011.

[2] Open Source Robotics Foundation: : http://wiki.ros.org/.

[3] OpenRTM-aist : http://www.openrtm.org/openrtm/ja.

[4] OROCOS : http://www.orocos.org/.

[5] 鈴木拓也，山崎優作，田向権，関根優年：“移動型ロボットに統合する知能処理回路”，電子情報通信学会技術研究報告，VLD2011−105, CPSY2011−68, RECONF2011−64, pp.83～88, 2012.

[6] Xillybus : http://xillybus.com/.

[7] 山科和史，大川猛，大津金光，横田 隆史：“ロボット応用のための ROS 準拠 FPGA 画像処理コンポーネントの基礎設計”，情報処理学会 第 77 回全国大会講演論文集, pp.1-169～1-170, 2015.