ハードウェアTCP/IPスタックを使用したROS準拠FPGAコンポーネントの通信性能評価

菅田 悠平[1,a)]　大川 猛[1,b)]　大津 金光[1,c)]　横田 隆史[1,d)]

概要：画像処理などの処理データ量が多い組込みアプリケーションを，FPGA を用いて加速することが期待されている．FPGA を活用するためにはソフトウェア・ハードウェアの両面に渡る開発が必要であり，組込みシステムへの導入が困難である．この問題に対し，ロボット開発に広く使われているミドルウェアである ROS(Robot Operating System) に準拠したコンポーネントとして FPGA を活用する技術が提案された．しかし，これまでに提案されたプログラマブル SoC(System on Chip) を使用したコンポーネントでは，ARM プロセッサで通信処理が行われるため通信遅延が大きい問題があった．本稿ではハードウェア TCP/IP を用いて ROS の通信処理をハードウェア化し，画像処理を想定した数 Mbyte のデータの転送を行った際の通信性能を評価した結果について述べる．評価の結果，ARM 上で動作させた通常のソフトウェア ROS が 6M バイト受信・送信するための遅延時間が 290ms であるのに対し，提案するハードウェア ROS 準拠 FPGA コンポーネントは 175ms であることが分かった．

キーワード：FPGA，ROS，ロボット，組込みシステム，ハードウェア TCP/IP

Evaluation of communication performance of ROS-compliant FPGA component using hardware TCP/IP stack

Sugata Yuhei[1,a)]　Ohkawa Takeshi[1,b)]　Ootsu Kanemitsu[1,c)]　Yokota Takashi[1,d)]

Abstract: It is expected to accelerate embedded applications with a large amount of processing data such as image processing using the FPGA. In order to utilize the FPGA, it is necessary to develop both software and hardware and it is difficult to introduce it to the embedded system.
To solve this problem, a technology was proposed to utilize FPGA as a component compliant for Robot Operating System (ROS), which is widely used for robot development. However, in the components using programmable SoC (System on Chip) proposed so far, since the communication processing is performed by the ARM processor, there is a problem that the communication delay is large. In this paper, we present the result of evaluating the communication performance when transferring data of several megabytes assuming image processing by hardware processing of ROS communication processing using hardware TCP/IP. As a result, while the latency to receive and send 6M byte by normal software ROS on ARM processror was 290ms, the latency was 175ms in the case of proposed Hardware ROS-comliant FPGA component.

Keywords: FPGA, ROS, Robot, embededd system, Hartware TCP/IP,

1. はじめに

組込みシステムにおいて画像処理・画像認識などを行う，エンベデッドビジョンにおいては，限られた消費電力において要求性能を満たすための処理基盤として，FPGA（Field Programmable Gate Array）が注目されている [1]. FPGA は，ユーザがディジタル論理回路を自由にプログラムすることが可能な LSI チップであり，アプリケーションに応じたハードウェア並列処理が実現でき，汎用的なプロセッサや GPU と比較した場合，電力あたりの性能が高い [2]．そのため，画像処理のようなソフトウェアでは時間がかかる処理を FPGA にオフロードすることで，システム全体の性能向上が期待できる．しかし，FPGA を活用す

[1] 宇都宮大学大学院工学研究科
Graduate School of Engineering Utsunomiya University
7-1-2 Yoto, Utsunomiya, Tochigi, 321-8585, Japan
[a)] sugata@virgo.is.utsunomiya-u.ac.jp
[b)] ohkawa@is.utsunomiya-u.ac.jp
[c)] kim@is.utsunomiya-u.ac.jp
[d)] yokota@is.utsunomiya-u.ac.jp

るにはハードウェア設計の知識が必要不可欠であるため，ソフトウェアシステムへの導入が難しい．[3]

この問題に対して，現在ロボット開発に広く使われているソフトウェア開発プラットフォームである ROS（Robot Operating System) を対象として，FPGA を使用した処理回路をコンポーネント化する，ROS 準拠 FPGA コンポーネントが提案された [4]．画像処理等を効率的に処理する FPGA をコンポーネント化することにより，容易に HW/SW 協調のロボットシステムを構築することができる．しかし，これまでに提案されたプログラマブル SoC(System on Chip) を使用したコンポーネントでは，ARM プロセッサで通信処理が行われるため通信遅延が大きかった．通信遅延が大きいために，FPGA を用いてアプリケーションを高速化しても，通信がボトルネックとなり，システム性能の向上に繋がらないという課題があった [4]．

本稿では，通信処理の高速化のために，ハードウェア TCP/IP スタックを使用した ROS 準拠 FPGA コンポーネントの設計・実装を行い，通信性能評価を行った結果について報告する．性能評価の際には，画像処理を想定した数 M バイトのデータ転送の遅延時間を評価する．第 2 章にて HW/SW 協調システムにおける FPGA コンポーネント技術について述べる．第 3 章にて本研究が対象としている ROS の概要，従来の ROS 準拠 FPGA コンポーネントの課題について述べる．第 4 章にて，ハードウェア化について説明し，第 5 章にて，その通信性能の評価を行う．最後に第 6 章で本論文をまとめる．

2. FPGA コンポーネント技術

2.1 FPGA 設計の課題

組込みシステムにおいて，画像処理などの要求性能を満たすために，電力効率の高い FPGA を用いて処理を高速化することが望まれている．しかし，組込みソフトウェアと FPGA を用いてシステムを構成するためには，FPGA の開発に関して問題がある．

現状，FPGA の開発を行う際，一般的にはクロック単位でレジスタに書き込む論理演算を行う RTL(Register Transfer Level) の設計を行わなければならない．また，論理演算の記述のためには，Verilog-HDL や VHDL などの HDL(Hardware Description Language) を用いる必要がある．また，開発したハードウェア回路の機能検証では，クロック単位での波形シミュレーションなどを用いる必要がある．そのため，デバッグにも時間がかかり FPGA の開発生産性はソフトウェアと比較すると低い [5]．

最近では，C 言語からハードウェアを合成する高位合成ツールが製品化 [6] されて広く用いられるようになり、更には Java・Python などのソフトウェア言語からハードウェアを合成することも可能 [7][8] になった，しかし，既存の C 言語等のソフトウェアがそのまま合成可能というわけではなく，高位合成可能なソフトウェア記述に変更する必要がある．そのため，FPGA の性能を引き出し，最適なハードウェア回路を設計するには，ハードウェア設計に対する深い理解が必要不可欠である．また，生成されるハードウェア回路は必ずしも最適なものとは限らず，課題は残っている．

このように，通常のソフトウェア技術者が FPGA 上に最適なハードウェア回路を実装するのは困難である．そのため，熟練の FPGA 技術者によって実装された最適なハードウェアを，部品としてソフトウェアから使用可能にする技術が求められている．

2.2 FPGA コンポーネントを用いた HW/SW 協調設計

そこで，FPGA コンポーネントを用いた HW/SW 協調システム開発手法が行われている [4][9][10]．FPGA コンポーネントを用いた HW/SW 協調システムの概念を**図 1** に示す．近年の FPGA デバイスの大容量化・低価格化により，多くの組込みシステムにおいて FPGA デバイスの導入が現実的となっている．そのため，FPGA を用いて，HW/SW 協調システムを構築することにより，短い開発日数でシステムの要求性能を満たすことが期待できる．しかし，FPGA 上の HW を直接操作するソフトウェアを開発するためには，物理メモリアドレス上に配置されたレジスタを操作するため，OS カーネル (特権) モードのデバイスドライバ開発を行うこととなり，開発コストがかかる．

そのため，FPGA によるハードウェア処理のコンポーネント化（部品化）を行う．コンポーネント化によって，FPGA 上のハードウェアの実装の詳細をカプセル化して隠蔽し、外部との通信により入力・出力のインターフェイスを明確化・抽象化して FPGA 上の HW の操作を容易にすることが出来る [3][9]．FPGA をコンポーネント化することにより，ソフトウェア開発者は FPGA 上のハードウェア回路を部品として扱うことができる．そのため，一般的なソフトウェア開発者が FPGA を扱うことが可能となると期待される．

ここで用いる FPGA コンポーネントには，入力にかかる通信時間，処理自体の時間，出力にかかる通信時間を全て加味したうえで，要求される処理性能を満たす必要がある．そのため，通信による遅延時間は極力小さくする必要がある．また，通信を行うために FPGA のハードウェアリソースを大量に消費してしまうことも問題となる．アプリケーション処理を実現するための FPGA のハードウェアリソースを確保するために，通信に用いるハードウェアリソース量を極力小さくする必要がある．

一般にコンポーネント技術が実際のシステムで採用されるためには，コンポーネント化による，性能・コスト面のオーバーヘッドを最小化する必要がある．何故なら，いくら設計がしやすいと言っても，高速な FPGA 処理をコン

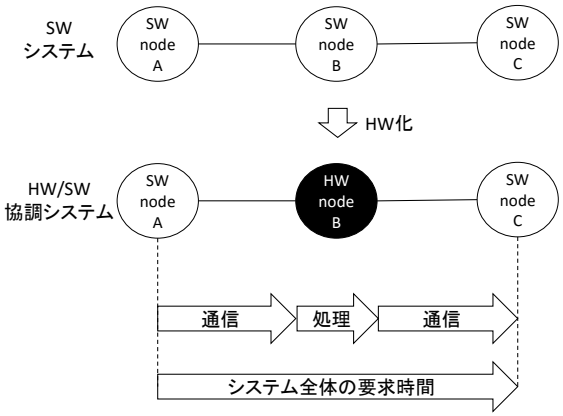

図 1 FPGA コンポーネントを用いた HW/SW 協調システム
Fig. 1 concept of HW/SW cooperative system using FPGA component

ポーネント化することによって，外部・内部での通信が加わることで遅延時間が大きくなってしまってはシステム全体としての性能改善にはつながらない．また，コンポーネント化のために大幅にコストが増加する（例えば高価な FPGA が必要になる）ことも受け入れられない．すなわち，コンポーネント化による設計の容易化による設計生産性の向上は，性能・コストのオーバーヘッドとトレードオフの関係にあることに注意すべきである．

3. ROS 準拠 FPGA コンポーネント

3.1 ROS の概要

本研究が対象とする ROS は，ロボット開発用のミドルウェアであり，現在ロボット開発に広く使われている．ROS は OSRF(Open Source Robotics Foundation) によってオープンソースで公開されており，主に UNIX OS 上で動作し，Ubuntu が公式にサポートされている [20]．ROS は主にソフトウェア開発用のビルドツール，ソフトウェア間の通信フレームワークを提供する．ROS を使用したロボット開発はコンポーネント指向設計に基づいており，ロボット開発者は様々なソフトウェアコンポーネントを組み合わせることでロボットシステムを構築することができる．また ROS の公式 wiki では，アクチュエータの制御や画像処理などの多くのソフトウェアコンポーネントが公開されており，ユーザはダウンロードして使用することができる．

ROS の通信モデルは，Publish/Subscribe メッセージングに基づいた非同期な通信方法である．Publish/Subscribe メッセージングの概要を**図 2** に示す．この通信方法は，サーバー・クライアントモデルと比較して，ノード同士の結合が低いという特徴があり，動的なネットワークを構成することができる．そのため，ノードの追加や変更などを容易に行うことができる特徴がある．Publish/Subscribe メッセージングでは，データを配信するノードを Publisher，

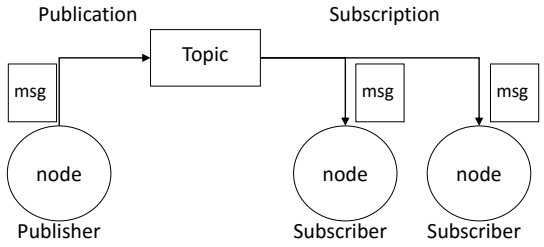

図 2 Publish/Subscribe メッセージング概要
Fig. 2 concept of Publish/Subscribe messaging in ROS

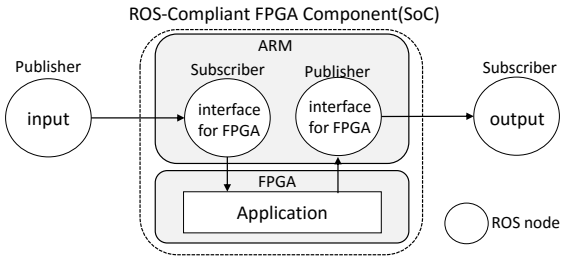

図 3 プログラマブル SoC を用いた ROS 準拠 FPGA コンポーネントの概要
Fig. 3 a structure overview of ROS-Compliant FPGA Component implemented on Programmable SoC

データを購読するノードを Subscriber という．送受信するデータはメッセージ単位で行われ，メッセージには予め定義されてる形式を使用するか，ユーザ自身で定義することが可能である．また，各ノードはトピック単位で通信を行い，Publisher ノードはトピックを購読している Subscriber に対してメッセージの送信を行い，Subscriber ノードは購読しているトピックからメッセージの受信を行う．このようにして ROS システムにおいては多くのノードからなる分散システムを構成する．

3.2 プログラマブル SoC を用いた ROS 準拠 FPGA コンポーネント

最初の提案における ROS 準拠 FPGA コンポーネントは，ARM プロセッサと FPGA を搭載したプログラマブル SoC(System on Chip) を用いることを前提として設計された [4]．プログラマブル SoC を用いた ROS 準拠 FPGA コンポーネントの概要を**図 3** に示す．ARM プロセッサ上に実装された，”interface for FPGA ”ノードは，他の ROS プロセスとの Publish/Subscribe 通信と HW/SW 間の通信を行う．具体的には，input ノードが配信するトピックを購読し，入力データを受信する．受信したデータを FPGA に渡し，処理結果を受け取る．その後 output ノードが購読するトピックに対して配信する．また，このような HW/SW 間の通信インタフェースを自動生成する cReComp(creator for Reconfigurable Component) も提案されており，コンポーネントの生産性向上も図られている [11]．

しかし，プログラマブル SoC を用いた ROS 準拠 FPGA

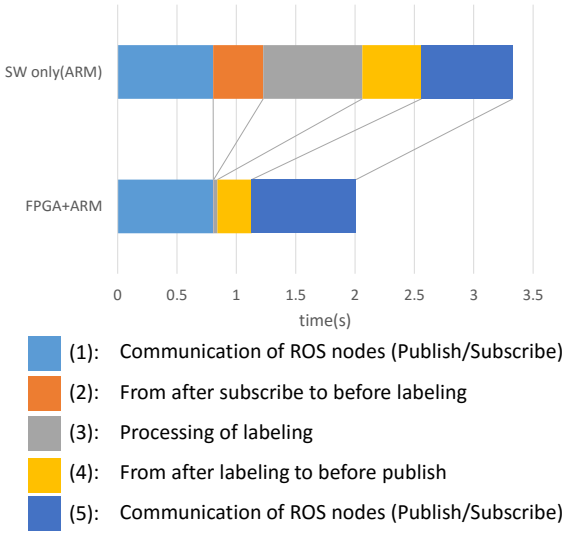

図 4 プログラマブル SoC を用いた ROS 準拠 FPGA コンポーネントの性能評価 [4]

Fig. 4 latency comparison of ROS-Compliant FPGA Component implemented on Programmable SoC

コンポーネントには，ROS の Publish/Subscribe 通信の遅延時間が大きいという課題があった．先行研究における性能評価結果を**図 4** に示す [4]．これは，入力を行うノードがコンポーネントに入力画像 (1920*1080) を送信してから，出力ノードが処理結果を受信するまでの遅延時間を分析したものである．図中 (3) ラベリング処理の ARM プロセッサでのソフトウェア実行時間が 0.835 秒であるのに対して，FPGA での実行時間は 0.032 秒と大幅な速度向上を達成している．しかし，図中 (1)，(5) の ROS の Publish/Subscribe 通信にかかる時間が，(1)，(5) ともに約 0.8 秒と FPGA での実行時間と比較して大きいことが分かる．そのため，FPGA を使用して処理時間を高速化しても，システム全体の要求時間を満たすことは難しい．

4 章ではハードウェア ROS 準拠 FPGA コンポーネントについて述べる．ハードウェア ROS 準拠 FPGA コンポーネントとは，ROS の通信処理のハードウェア化，つまり従来 topic の購読/配信を行っていた ” interface for FPGA ” ノードを FPGA 上に実装することで，プログラマブル SoC を使用しないで FPGA による処理をコンポーネント化する手法である．

4. ハードウェア ROS 準拠 FPGA コンポーネント

4.1 通信分析

ROS の Publish/Subscribe 通信プロトコルの概要は，公式 wiki に記載されている [20]．ROS の通信処理のハードウェア化を具体的に検討するために，通常の ROS ソフトウェアで構成されたシステムを対象として，ノード間を流れるパケットの分析を行った．パケット分析環境としては，Wireshark を用いた [21]．パケット分析に基づいた，Master，Publisher，Subscriber の通信内容を**図 5** に示す．Master とは ROS システム内で必要不可欠なプロセスであり，システム内の名前解決を行う．以下に通信手順およびその内容を示す．

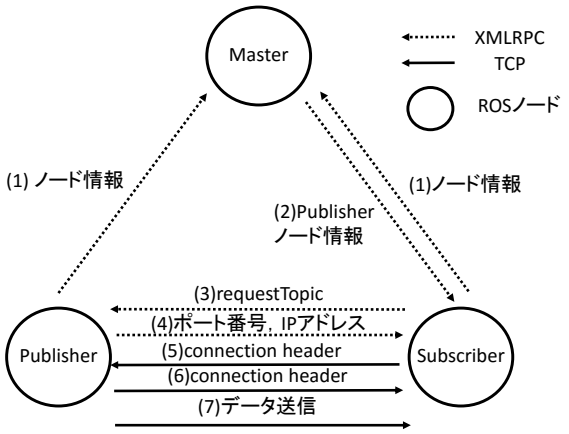

図 5 Publish/Subscribe 通信の手順

Fig. 5 procedure of Publish/Subscribe messaging

(1) Publisher，Subscriber は，XMLRPC を使用して [12]，API を呼び出し，自身のノード名，IP アドレス，配信/購読するトピック名を Master に登録する．XMLRPC は，RPC データ形式に XML を利用した遠隔手続き呼び出しのことであり，転送プロトコルは HTTP を使用する．

(2) Subscriber は自身が購読するトピックの配信元となる Publisher ノードの情報を得る．

(3) Subscriber ノードが Publisher ノードの API である requestTopic を呼び出す．

(4) Publisher ノードは，(3) に対して，データ通信を行う TCP のポート番号を返す．

(5) Subscriber ノードは，(4) の TCP ポートに対してコネクションを確立し，connection header を送信する．

(6) Publishern ノードが Subscriber に対して，connection header を送信する．

(7) Publisher ノードから Subscriber ノードに対して，データ送信が繰り返される．

分析の結果，Publish/Subscribe 通信は，XMLRPC プロトコルを使用した通信 (図中 (1〜4)) と TCP 通信による生データの送受信 (図中（5〜7）) からなることが分かった [18]．また，Master との通信 (図中 1,2)，Publisher と Subscriber 間での XMLRPC 通信 (3,4)，データ通信 (5,6,7) の様に ROS の通信には，複数のポートが使われる．

4.2 ハードウェア化方針

以上の分析結果に基づいてハードウェア化方針を検討した．XMLRPC が使用している転送プロトコルである HTTP は，TCP/IP の上位層に位置するため，ROS の通信

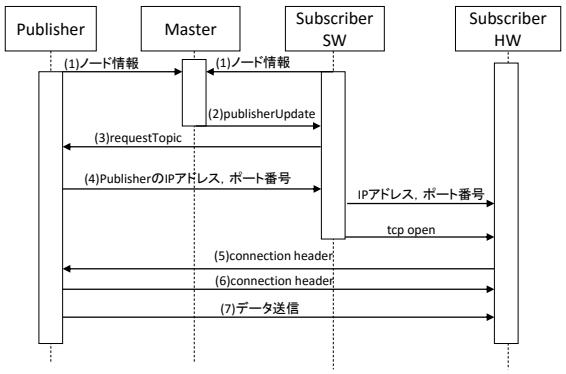

図 6 Subscriber HW の通信シーケンス
Fig. 6 communication sequence of Subscriber HW

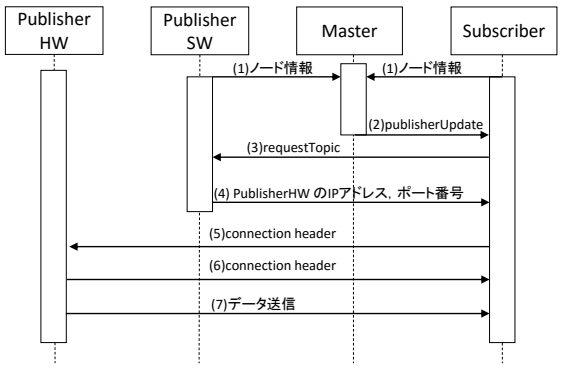

図 7 Publisher HW の通信シーケンス
Fig. 7 communication sequence of Publisher HW

処理をハードウェア化するためには，FPGA 上で TCP/IP 通信を行う必要がある．これまでに FPGA 上で TCP/IP 通信を行うためのハードウェア TCP/IP スタックが提案されてきた [13][14][15]．提案されているハードウェア TCP/IP スタックの多くは，TCP/IP プロトコルの全てをハードウェア化するとリソース使用量が増えることなどから，使用できるポート数，セッション数，使用できるプロトコルが限定されている [16]．そこで，通信機能のみを持つ 1 ポート，1 セッションのハードウェア TCP/IP スタックを使用することを想定した ROS の通信のハードウェア化手法を検討した [17]．検討の結果，XMLRPC 通信部分をソフトウェアで行い，データ通信部分をハードウェアで行うことで，ROS の Publish/Subscribe 通信をハードウェア化できることが明らかとなった [17][19]．

Subscriber ノードを分割した際の通信シーケンスを**図 6** に示す．分割後のソフトウェアを Subscriber SW，ハードウェアを Subscriber HW と呼ぶ．(1)〜(4) の XMLRPC の通信処理を Subscriber SW が行い，その結果得られた TCP コネクションのポート番号，IP アドレスを Subscriber HW に送信する．Subscriber HW は，その IP アドレス，ポート番号を基に Publisher に対してコネクションを確立し，(5)〜(6) の connection header を交換した後に，データを受信する (7)．

Publisher ノードを分割した際の通信シーケンスを**図 7** に示す．分割後のソフトウェアを Publisher SW，ハードウェアを Publisher HW とする．(1)，(4) の XMLRPC の通信処理を Publisher SW が行い，(4) のときに Publisher HW の TCP コネクションのポート番号，IP アドレスを Subscriber に送信する．Subscriber が Publisher HW に対してコネクションを確立し，connection header を送信する (5)．その後，Publisher HW は connection header の送信 (6)，データ送信 (7) を行う．

本稿では，上記方針に基づいて，ROS 準拠 FPGA コンポーネントの通信処理をハードウェア化し，性能評価を行った結果について説明する．

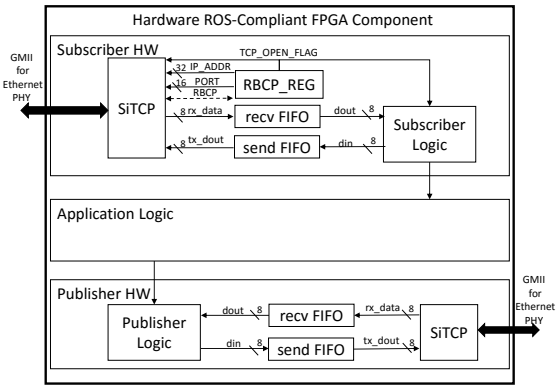

図 8 ハードウェア ROS 準拠 FPGA コンポーネントのブロック図
Fig. 8 block diagram of Hardware ROS-Compliant FPGA Component

4.3 実装方法

4.2 で説明した手法を用いて実装した，ハードウェア ROS 準拠 FPGA コンポーネントの構成を**図 8** に示す．Subscriber HW，Publisher HW の TCP/IP 通信を実現するためのハードウェア TCP/IP スタックには SiTCP を使用した [13]．SiTCP は，データ通信用の TCP ポートを一つと遠隔バス制御用の UDP ポートを一つ持っているハードウェア TCP/IP スタックであり，ギガビット Ethernet にて最大 950Mbps の転送速度で通信が可能である．また，開発者が使用可能な IP として配布されているため，本研究で使用することとした．

Subscriber HW の SiTCP はクライアントモードで動作し，Subscriber SW から受け取った IP アドレス，ポート番号を基に入力データの配信を行う Publisher ノードに対してコネクションを確立する．送受信の制御を Subscriber Logic が行い，コネクションを確立したら，送信用 FIFO に connection header の送信データを書き込みを行うことで，TCP パケットが送信される．その後，受信用 FIFO に書き込まれるデータを Application Logic へと渡す．

Publisher HW の SiTCP はサーバモードで動作し，Application Logic の処理結果を購読する Subscriber からのコ

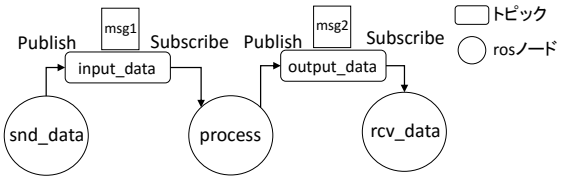

図 9 評価対象の ROS システムのノード構成
Fig. 9 nodes and topics of the ROS system used for measurement of communication latency

ネクションの確立を待つ．コネクションが確立され connection header を受信したら，Publisher Logic が送信用 FIFO に connection header を書き込む．その後 Application logic によって処理されたデータを送信用 FIFO に書きこむことで，Subscriber に処理結果を送信する．

5. 性能評価

5.1 評価を行った ROS システム

ハードウェア ROS 準拠 FPGA コンポーネントを実装し，Publish/Subscribe 通信の遅延時間の評価を行った．評価を行う際に構築した ROS システムのノード構成を**図 9** に示す．snd_data ノードは，配信するトピックである，input_data トピックにメッセージ 1 を送信する．process ノードは inpud_data トピックを購読し，メッセージ 1 を受信したら，output_data トピックにメッセージ 2 を送信する．rcv_data ノードは output_data トピックを購読し，メッセージ 2 を受信する．メッセージ 1，メッセージ 2 はともに sensor_msgs/image 型である．データ構造を**図 10** に示す．このメッセージ型は，ROS で用意されているメッセージ型であり，画像の幅，高さ，エンコードに関する変数を持つ．変数 step に入れた値によって，配列である data のサイズが決まる．例えば step 数を 1 とすれば，1 バイトの配列が用意され，1000 とすれば，1000 バイトの配列となる．このシステムにおいて，process ノードと同等の動作をするハードウェア ROS 準拠 FPGA コンポーネントを実装し，snd_data ノードがメッセージ 1 を送信してから rcv_data ノードがメッセージ 2 を受信するまでの時間を計測した．測定に用いたメッセージ 1, メッセージ 2 のサイズは，画像サイズ 480*640 の各ピクセル 3 バイト (RGB) を想定した 1M バイト，2 値化されたフル HD 画像 (1920*1080*1) の 2M バイト，1920*1080*3 の 6M バイトで行った．

5.2 測定環境

ハードウェア ROS 準拠 FPGA コンポーネントの通信遅延を比較するために，汎用的なプロセッサを搭載した PC-PC，PC-ARM プロセッサ，PC-FPGA の 3 通りで計測を行った．測定環境を図 9 に示す．PC1，PC2，ARM，FPGA は全てギガビット Ethernet で接続されている．PC1

```
std_msgs/Header header
uint32      height
uint32      width
string      encoding
uint8       is_bigendian
uint32      step
uint8[ ]    data
```

図 10 sensor_msgs/image メッセージの内容
Fig. 10 contents of message of sensor_msgs/image

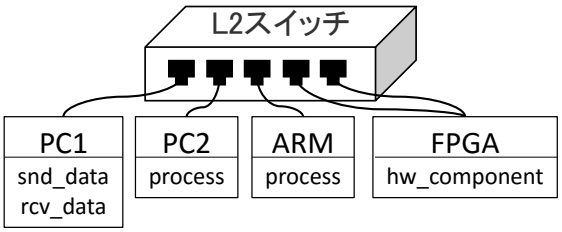

図 11 測定環境
Fig. 11 measurement environment

表 1 ハードウェアリソースの使用量
Table 1 resource used for implementing Hardware ROS-comliant FPGA component

RESOURCE	UTILIZATION
Slice Registers	9035/126576(7%)
Slice LUTs	7110/63288(11%)
RAMB16BWERs	129/268(48%)

で実行している snd_data ノードがデータを送信すると，レイヤ 2 スイッチを経由して，PC2 または ARM プロセッサで実行している process ノードがデータを受信する．データを全て受信したら，PC1 上で実行している rcv_data ノードに対してデータ送信を行う．PC1，PC2 には CPU が Intel Core i7 870(2.93GHz)，メモリが 4GB のものを使用した．OS は Ubuntu16.04 である．ARM プロセッサには，Xilinx 社製 ZC7Z020 搭載の Zedboard 上の ARM Cortex-A9 プロセッサを用いた [22]．FPGA ボードには，Xilinx 社の Spartan-6(XC6SLX100) を搭載した，株式会社 e-trees.Japan の exTri-CSI[23] を用いた．このボードは，ギガビット Ethernet を 3 ポートもっており，そのうちの 2 つと SiTCP のインスタンスを接続している．また実装したハードウェアのリソース量を**表 1** に示す．表 1 に示されているリソース量は，図 8 の Subscriber HW と Publisher HW の合計である．

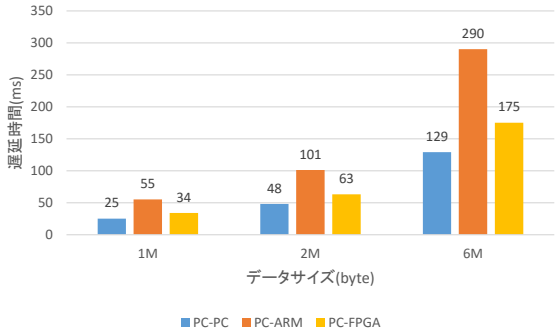

図 12 画像メッセージを送信してから画像メッセージを受信するまでの遅延時間

Fig. 12 measured latency from sending image message to receiving image message

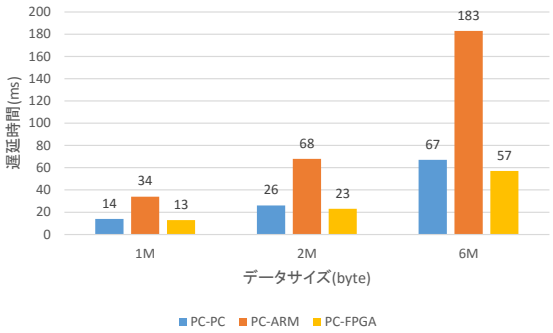

図 13 画像メッセージを送信してから終了メッセージを受信するまでの遅延時間

Fig. 13 measured latency from sending image message to receiving end message

5.3 測定結果

snd_data ノードが画像メッセージを送信してから，rcv_data ノードが画像メッセージを受信するまでの遅延時間の測定結果を**図 12** に示す．画像メッセージとはメッセージサイズを 1M,2M,6M バイトにしたものである．画像サイズが増加するにつれて，遅延時間も増加していることが分かる．6M サイズの送受信にかかる遅延時間は，PC-PC 間の場合は 129ms，PC-ARM 間の場合は 290ms，PC-FPGA 間の場合は 175ms であった．ARM プロセッサと比較して，ハードウェア ROS 準拠 FPGA コンポーネントの遅延時間が短くなることが分かった．

しかし，SiTCP のスループットである 949Mbps と比較したときに，大きな速度向上を得られていないことから，受信のみ，送信のみに分けて，さらに性能評価を行った．snd_data ノードが画像メッセージを送信してから，rcv_data ノードが終了メッセージを受信するまでの遅延時間の測定結果を**図 13** に示す．終了メッセージとはメッセージ 2 の step 数を 1 としたものであり，メッセージサイズは 46byte である．すなわち図 13 の遅延時間は，主に実装したハードウェア ROS 準拠 FPGA コンポーネントが画像データを受信するためにかかった時間である．画像データ 6M バイトの場合，PC-PC 間の通信時間 (67ms) よりもさらに短い，57ms で通信が行えていることが分かる．

一方，snd_data ノードが開始メッセージを送信してから，rcv_data ノードが画像メッセージを受信するまでの遅延時間の測定結果を**図 14** に示す．開始メッセージとはメッセージ 1 の step 数を 1 としたものであり，上記で述べた終了メッセージと同様にメッセージサイズは 46byte である．すなわち図 14 の遅延時間は，主に実装したハードウェア ROS 準拠 FPGA コンポーネントが画像データを送信するためにかかった時間である．画像データ 6M バイトの場合，PC-ARM 間の通信時間 (106ms) よりも長い，116ms であった．つまり，PC-FPGA 間の通信において，ハードウェア ROS 準拠 PGA コンポーネントが画像データを受信する際の遅延 (57ms) とくらべて，送信する際の遅延が大きい (116ms) ことが分かる．

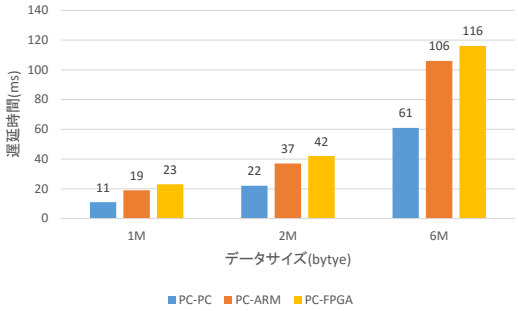

図 14 開始メッセージを送信してから画像メッセージを受信するまでの遅延時間

Fig. 14 measured latency from sending start message to receiving image message

表 2 画像メッセージを送信する際のパケット分析結果

Table 2 packet analysis result of sending image message

	データサイズ (byte)	パケット送出間隔 (ms)
PC-PC	65160	0.556
FPGA-PC	16060	0.294

送信時の遅延が大きい原因について調べるため，図 14 の 6M バイトの画像メッセージを送信する際の通信状況について，Wireshark を用いたパケットキャプチャを行った．画像メッセージを送信する際のパケット分析結果を**表 2** に示す．

PC-PC 間の通信 (61ms) の場合では，データサイズが 65160byte のパケットを平均約 0.55ms で送信していることが分かった．これは通信スループットとして約 937Mbps に相当する．一方，FPGA-PC 間の通信 (116ms) の場合では，データサイズが 16060byte のパケットを平均約 0.29ms で送信していることが分かった．これは通信スループットとして約 436Mbps に相当する．以上の測定から，SiTCP を使用した際の送信パケットのデータサイズを大きくするか，パケット送出の間隔を短くすることで通信スループッ

トを向上できれば，ハードウェア ROS 準拠 FPGA コンポーネントの通信性能を向上できると考えられる．

6. おわりに

本稿では，ROS を対象として，ハードウェア TCP/IP スタックを使用したハードウェア ROS 準拠 FPGA コンポーネントを提案し，FPGA 上に実装して画像データを含む ROS メッセージを送受信した際の通信性能を評価した．ハードウェア TCP/IP スタックとして用いた SiTCP は，1 ポート 1 セッションに機能を限定しており，少ないハードウェア量で TCP/IP 通信を実現可能であることが分かった．提案するハードウェア ROS 準拠 FPGA コンポーネントの，Publish/Subscribe 通信にかかる遅延時間を測定した．評価には，1M バイトから 6M バイトの画像データを含む ROS メッセージを使用した．通信路はギガビット Ethernet を用いた．評価の結果，ARM 上で動作させた通常のソフトウェア ROS が 6M バイト受信・送信するための遅延時間が 290ms であるのに対し，提案するハードウェア ROS 準拠 FPGA コンポーネントは 175ms であることが分かった．分析の結果，受信のみを行った場合は 57ms であるのに対し，送信のみを行った場合に 116ms かかっていることが分かった．今後の課題は送信時の遅延を削減することである．

謝辞 本研究開発は，総務省 SCOPE (受付番号 152103014) の委託を受けたものです．

参考文献

[1] 平岩丈治, 天野英晴. ”再構成可能なリアルタイムビジョンアーキテクチャの FPGA への実装 (応用システム, デザインガイア 2012-VLSI 設計の新しい大地-).” 電子情報通信学会技術研究報告. CPSY, コンピュータシステム 112.322 pp. 39-44, 2012.

[2] Asano, Shuichi, Tsutomu Maruyama, and Yoshiki Yamaguchi. ”Performance comparison of FPGA, GPU and CPU in image processing.” Field Programmable Logic and Applications, 2009. FPL 2009. International Conference on. IEEE, 2009.

[3] 東遼平, 高瀬英希, 高木一義, 高木直史.”プログラマブル SoC のためのシステム設計環境の検討と SW-HW インタフェース生成手法の実装”. 研究報告システム LSI 設計技術 (SLDM), 34, pp. 1-6, 2014.

[4] Kazushi Yamashina, Takeshi Ohkawa, Kanemitsu Ootsu and Takashi Yokota,“ Proposal of ROS-compliant FPGA Component for Low- Power Robotic Systems - case study on image processing application, ”Proceedings of 2nd International Workshop on FPGAs for Software Programmers, FSP2015, pp. 62-67, 2015.

[5] 三好健文, 船田悟史. ”Fpga 向け高位合成言語としての java の活用手法の検討.” 第 53 回プログラミング シンポジウム予稿集, pp. 59-68, 2012.

[6] XILINX INC.: https://japan.xilinx.com/

[7] 三好健文. ”企業における FPGA アプリケーション研究開発の一例.” 電子情報通信学会技術研究報告. RECONF, リコンフィギャラブルシステム 114.223, pp. 51-56, 2014.

[8] 高前田伸也. ”PyCoRAM による Python を用いたポータブルな FPGA アクセラレータ開発.” 組込みシステムシンポジウム 2014 論文集, pp. 2-2, 2014

[9] 工藤健慈，今中晴記，志賀裕介，関根優年,“ hw/sw 混載システムにおける hwObject モデルとその制御手法 ”，情報科学技術フォーラム一般講演論文集第 1 分冊 FT2002, C-3,pp.193-194，2002.

[10] 大川 猛，高野 創司，植竹 大地，横田 隆史，大津 金光，馬場 敬信,“ ソフトウェアから FPGA を容易に扱うための分散オブジェクトプラットフォーム ”，情報処理学会第 54 回プログラミング・シンポジウム論文集，pp.127-136, 2013.

[11] Kazushi Yamashina, Hitomi Kimura, Takeshi Ohkawa, Kanemitsu Ootsu, Takashi Yokota, “ cReComp: Automated Design Tool for ROS- Compliant FPGA Component, ” In proc. IEEE 10th International Symposium on Embedded Multicore/Many-core Systems-on-Chip (MCSoC-16), 2016.

[12] Allman, Mark. ”An evaluation of XML-RPC.” ACM sigmetrics performance evaluation review 30.4, pp. 2-11, 2003.

[13] Tomohisa Uchida. ”Hardware-Based TCP Processor for Gigabit Ethernet,” IEEE Transactions on Nuclear Science, Vol.55, No.SIG 3, pp.1631-1637, 2008 .

[14] David Sidle, Zsolt Istvan, Gustavo Alonso. ”Low-Latency TCP/IP Stack for Data Center Applications,”” 26th International Conference on Field Programmable Logic and Applications (FPL), 2016 .

[15] 田向権, ベルグシュタインナダヴ, and 関根優年. ”ネットワーク化された hw/sw 複合体のための Reconfigurable TCP/IP Offload Engine (RFID 関連技術, システムオンシリコン, 一般).” 電子情報通信学会技術研究報告. SIS, スマートインフォメディアシステム 111.342, pp. 47-50, 2011.

[16] 藤田琴子, 田向権, ベルグシュタインナダヴ, and 関根優年. ”WEB アプリに用いる FPGA 用 IP: TCP/IP 回路 (FPGA 応用, FPGA 応用及び一般).” 電子情報通信学会技術研究報告. CPSY, コンピュータシステム 111.398, pp. 1-6, 2012.

[17] 菅田 悠平，山科 和史，松本 拓也，田向 権，大川 猛，大津 金光，横田 隆史,“ ROS 準拠 FPGA コンポーネントにおける通信処理のハードウェア化 ”，第 61 回 システム制御情報学会研究発表講演会，pp. 324-3, 2017.

[18] 菅田 悠平，山科 和史，大川 猛，大津 金光，横田 隆史,“ ROS の Publish/Subscribe 通信のハードウェア実装による高速化の検討 ”，情報処理学会 第 79 回全国大会 講演論文集， pp.1-129〜1-130, 2017.

[19] Y.Sugata, T.Ohkawa, K.Ootsu, and T.Yokota. ”Acceleration of Publish/Subscribe Messaging in ROS-compliant FPGA Component.” In Proceedings of 8th International Symposium on Highly Efficient Accelerators and Reconfigurable Technologies (HEART 2017), 2017.

[20] ROS.org: http://wiki.ros.org/

[21] WireShark：http://www.wireshark.org/

[22] zedboard:http://zedboard.org/

[23] e-trees.Japan Inc.:http://e-trees.jp/