C-002

ROS2-FPGA ノード生成による高位合成 FPGA モジュール動作検証手法の提案
A Proposal of Operation Verification Method for High-Level-Synthesized FPGA Modules by ROS2-FPGA Node-Generation

森 隼人† 大川 猛‡ 菅谷 みどり†

1. はじめに

Society5.0 において，FPGA（Field-programmable gate array）モジュールをネットワーク上のクラウド，MEC（Multi-access Edge Computing）などの様々な場所に配置することで，IoT デバイスでは困難な高性能処理を準リアルタイム的に行う事が期待されている．MEC とは，5G の無線基地局に配置した計算リソースで，IoT デバイスやエッジデバイスからの要求に応じた計算負荷の大きい処理を行う方式である[1].

クラウドには大規模な FPGA を配置する事が可能な為，高負荷な処理を行う事が期待される．一方，MEC やエッジ，IoT デバイスは電力制約により FPGA の回路規模が限られる．これにより，回路規模と処理性能のトレードオフが発生する為，動作検証の際には実際に動作する環境を考慮した回路設計と開発が必要になる．また，ネットワーク上に配置する FPGA モジュールの動作検証には，回路の検証に加えて実際に動作する環境に合わせた通信機能の実装が必要である．しかし，こうした FPGA モジュールを実行環境のネットワーク上の任意の場所に配置して動作検証する手法は確立されていない．そこで，ノードをネットワークへ容易に追加・削除する事が可能なロボットソフトウェア向けミドルウェアである ROS2（Robot Operating System 2）[2]を用いる事で，ネットワーク構成に依存しない FPGA モジュールの配置を行う事が容易になると考えられる．

本研究の目的は，HLS（High Level Synthesis）ベースで生成した FPGA モジュールを実行環境のネットワーク上の任意の場所に配置して動作検証する手法を確立することである．これによって，ネットワーク上に配置する事を想定した FPGA モジュールの動作検証を容易に行う事を目指す．本稿では，C/C++ 言語からの高位合成によって生成した FPGA モジュールを，ROS2 ノードとして動作させる事で効率的に動作検証を行う手法を提案する．また，実装した FPGA モジュールを同一マシン内でのローカル通信，有線 LAN 接続による通信，無線 LAN 接続による通信の 3 つの異なるネットワーク環境で動作検証を行い，提案手法の有効性を評価する．

本稿の構成は以下の通りである．第 2 節にて関連研究，第 3 節にて提案手法に用いた FOrEST の紹介，第 4 節にて提案手法について述べる．第 5 節にて提案手法の評価を述べ，第 6 節にて本稿の結論を述べる．

2. 関連研究

IoT やエッジデバイスに ROS や FPGA を活用する研究は高い関心を集めている．IoT デバイスを始めとする組込みデバイス向けの ROS2 ノード軽量実行環境として文献[3]では mROS2 を提案している．mROS2 では，最小限な出版購読型の通信においては高い通信性能とメモリ軽量性を示した．メモリアクセス性能がボトルネックとなる SoC FPGA ボード上での動作が可能となれば，本研究で用いた M-KUBOS[4]のみならず小規模な FPGA モジュールも IoT デバイスやエッジデバイスとして活用の幅が広がると考えられる．文献[5]では，FPGA を代表とする再構成可能デバイス向け OS である ReconOS 上で動作する ReconROS を提案している．ReconROS は ROS2 ベースのシステム上で FPGA を用いた容易なハードウェアアクセラレーションを提供するプラットフォームであり，ROS2 ノードをハードウェアにマッピングする事で FPGA モジュールの ROS2 ノード化を実現した．

一方で，既存の HLS ツールでは，ハードウェア設計の検証の為のシミュレーションや最悪のケースを想定したタイミング情報を提供している事から，FPGA モジュールのオンボード検証について提案されている．文献[6]では，HLS ベースで生成した FPGA モジュールのオンボード検証用フレームワークである RC-Unity を提案した．提案フレームワークでは，FPGA モジュール単一での検証には有用であるが，FPGA モジュールをシステムに統合した際の動作検証は行えていない．文献[7][8]では，ホスト PC とはシリアル通信を想定した実機での動作検証環境を提案した．また文献[8]では，シリアル通信における画像転送にかかる時間が性能に影響しているが，リアルタイム性が求められる IoT やエッジ環境においては通信遅延を含む動作検証が期待される為，通信遅延が性能に影響しない事が求められる．

3. ROS2-FPGA ノード生成ツール FOrEST

3.1 ROS/ROS2

ROS（Robot Operating System）とはオープンソースで提供される，ロボットソフトウェア開発向けのミドルウェアである．ROS の特徴は，多くのプロセス間通信を roscore と呼ばれる ROS マスタが管理することで，ノードと呼ばれる各プロセスが通信相手の情報を必要としない，動的な非同期通信が可能な事である．また，ソフトウェア部品として自由に ROS システムに追加・削除可能な事からロボットソフトウェアの再利用性の向上が期待できる．

ROS2 とは，ROS の次世代バージョンであり，通信ミドルウェアに DDS(Data Distribution Service)

† 芝浦工業大学 Shibaura Institute of Technology
‡ 東海大学 Tokai University

Copyright © 2022 by
The Institute of Electronics, Information and Communication Engineers and
Information Processing Society of Japan All rights reserved.

を採用しており，リアルタイムの組込みシステムに適したミドルウェアである．DDS は，ROS の通信と似たデータ転送を提供するミドルウェアであり，送信元と受信元を自動で探索して通信を行う．

3.2 ROS2 における通信モデル

ROS2 において一般的に用いられるトピック通信は，メッセージの送受信を行うノード，通信経路であるトピック，ノード同士が送受信するデータであるメッセージの 3 つで構成される．メッセージの送信者を Publisher，受信者を Subscriber としてトピックを介して通信を行う．トピックではメッセージの内容が定義されており．Publisher と Subscriber は予め定義した型に従ったメッセージとトピックを指定することで，同じメッセージ型とトピックを指定するノードと通信することができる．これにより，通信相手の情報を知らずに通信する事が可能であり，ROS2 ネットワークへのノードの追加・削除が容易なため，動的なネットワークの構築が可能である．本研究では，ROS2 を用いて動的なネットワークを構築することで，FPGA モジュールが様々な場所で動作する事を想定した動作検証を行う．

3.3 FOrEST による自動生成

FOrEST[9]は，HLS ベースの FPGA を ROS2 ノードとして動作可能なプログラムを自動生成するオープンソースツールであり，Xilinx 社が提供する PYNQ フレームワーク上で動作する．図 1 に FOrEST によって生成される ROS2-FPGA ノードのシステム構造を示す．FOrEST の利用者はテンプレートを基にして，パッケージ名やユーザ IP コア名前を始めとする ROS パッケージ生成に必要な情報と，FPGA の入出力信号の情報を設定ファイルに記述する．その後 Python スクリプト（forest.py）を実行し，HLS ベースで生成した FPGA モジュールと通信する PYNQ ドライバ及び，C 言語で書かれた FPGA 回路の関数呼び出しパラメータを基に，他の ROS2 ノードと通信する為のカスタムメッセージを自動生成する事ができる．これにより，FPGA 駆動に必要なプログラムを記述する工程を省略し，FPGA モジュールをネットワーク上の ROS2 ノードとして動作検証する事が可能となる．本研究では，FOrEST を用いて FPGA モジュールを ROS2 ノード化する事で，効率的な動作検証手法の確立を目指す．

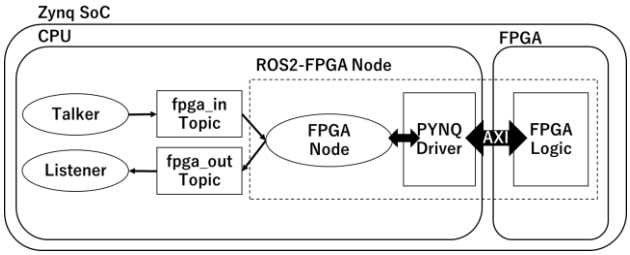

図 1 ROS2-FPGA ノードの構成

4. 提案手法

4.1 従来の動作検証

FPGA モジュールの動作検証手法として，HLS ツール上で動作検証を行う場合には，C/C++のテストベンチによるアルゴリズムの検証，C/RTL(Register Transfer Level)による協調シミュレーションによる IP コアの検証が挙げられる． 一方，実機における検証では，論理合成によって得られた回路を駆動する為のドライバを実装して回路の検証を行う．また，ネットワーク上に配置する FPGA 開発においては，回路の動作検証に加えて，実際の動作環境に合わせた通信機能の実装が必要である．

従来手法と提案手法の検証フローの比較を図 2 に示す．FPGA モジュールの動作検証における共通部分は C/C++言語での FPGA 設計から回路の論理合成までである．まず高位合成を行う前に，C/C++言語を用いたテストベンチによる C シミュレーションを行う．これにより，高位合成をする為に C/C++言語で記述したアルゴリズムが期待通りに動作するかを検証する．従来の実機での動作検証では，通信機能の実装例として 2 つの手法が挙げられる．

1. 手法 1：Python による Socket 通信の実装
2. 手法 2：ROS2 を用いた通信の実装

従来手法 1 では，Python を用いた Socket 通信の実装を行う．これは，一般的な Socket 通信のプログラムを記述するだけであるが，FPGA への入力パラメータの仕様に変更があった際に，通信機能も仕様に合わせて変更する必要がある． 従来手法 2 では通信に ROS2 を用いることで，通信機能の実装にかかるコストを削減することが可能である．従来手法 1 では通信相手の情報が必要であったが，ROS2 では通信相手の情報は不要であり，共通のトピック名を決めておくだけで良い．また，転送データについても ROS2 のカスタムメッセージとして定義しておくことで 1 つのメッセージ内に画像と string 型の文字列など複数のデータを含むことができる．一方で，これらの従来手法ではどちらも FPGA モジュールを駆動する為に PYNQ ライブラリを用いたドライバの実装が必要という課題があった．

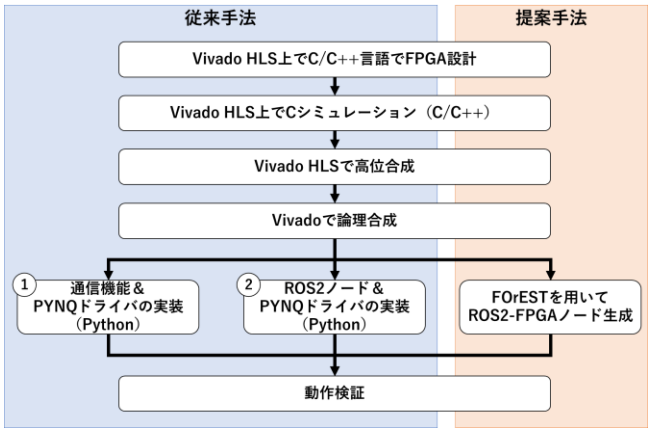

図 2　動作検証フローの比較

Copyright © 2022 by
The Institute of Electronics, Information and Communication Engineers and
Information Processing Society of Japan All rights reserved.

4.2 提案手法による動作検証

提案手法では，HLS ベースで生成した FPGA モジュールを対象に，FOrEST を用いて ROS2 ノード化する事で，実行環境に応じた通信機能及び PYNQ ドライバの実装を省いた動作検証を行う．

提案手法では，図 3 に示す FOrEST の設定ファイルを記述し，その情報を基に予め用意された Jinja2 テンプレートファイルに基づいて PYNQ ドライバや ROS2 ノードに必要なファイルが生成される．設定ファイルには ROS2 のパッケージ情報や FPGA の入出力のデータ型や通信プロトコルを記述する．

ROS2 を用いる従来手法 2 と提案手法の検証フローの比較を図 4 に示す．従来手法 2 では，ROS2 の独自メッセージ用パッケージや ROS2-FPGA ノード用パッケージの生成から PYNQ ドライバの実装を含む ROS2-FPGA ノード実装までをユーザが行う必要がある．一方で，提案手法におけるユーザが必要な主な作業は，図 3 に示した設定ファイルの記述とファイル生成に必要な Python スクリプトの実行である．

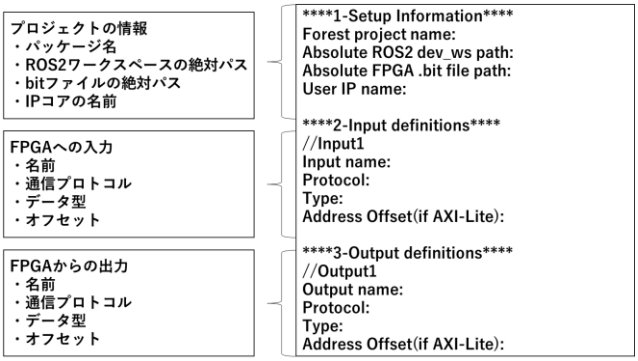

図 3 設定ファイルのテンプレート

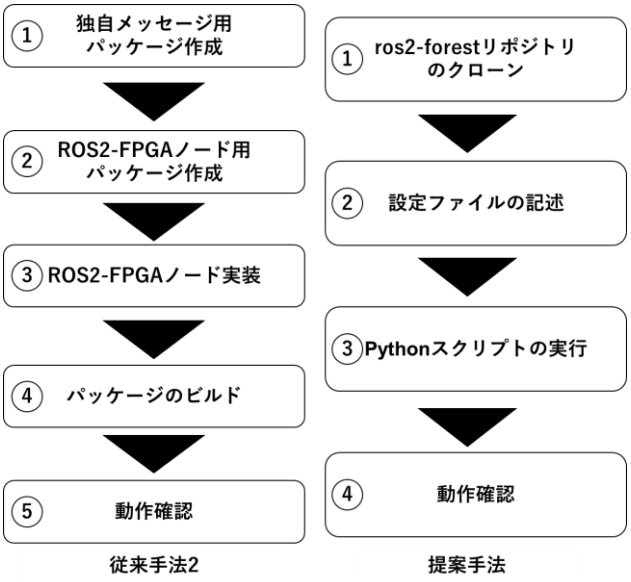

図 4 従来手法 2 と提案手法の比較

本稿では，FPGA 開発環境として Vivado HLS による高位合成及び Vivado を用いた論理合成の例を挙げる．FPGA 開発フロー及び FPGA 回路の C シミュレーションは従来と同様であるが，実際にネットワーク上に配置する際に，提案手法では FOrEST を用いて通信機能及び PYNQ ドライバの実装工程を省略可能な為，従来と比較して容易に動作検証可能であると言える．5 章にて従来手法との比較及び，ネットワーク構成の異なる 3 つの環境で提案手法による FPGA モジュールの動作検証を行い提案手法の有効性を評価する．

5. 評価

5.1 評価環境

本評価では，FPT'21 FPGA Design Competition に向けた自律移動ロボット開発[10]における白線認識 FPGA の動作検証を題材とした．

評価環境を**表 1** に示す．評価環境には，大規模な FPGA 環境として M-KUBOS[4]を使用した．M-KUBOS は PALTEK 社が提供する，Zynq UltraScale +シリーズ最大の回路規模である Zynq UltraScale+ xczu19eg を搭載した MEC として配置する事が期待されている FPGA ボードである．

5.2 ROS2 における画像転送遅延の計測

始めに予備評価として，ROS2 における通信遅延時間の計測結果を図 5 に示す．ここでは 3 つのサイズ（1920x1080，640x480，320x240）の画像を転送した際の遅延時間と，白線認識 FPGA の出力結果である要素数 20 の float64 配列をそれぞれ 100 回転

表 1　評価環境

	M-KUBOS
CPU	ARM Cortex A53 ARM Cortex　R5
FPGA	Zynq UltraScale+ xczu19eg
LUT	522720
FF	1045440
BRAM_18K	984
DSP48E	1968

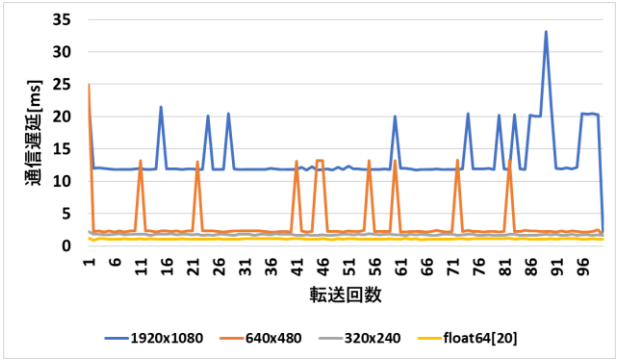

図 5 ROS2 における通信遅延時間

Copyright © 2022 by
The Institute of Electronics, Information and Communication Engineers and
Information Processing Society of Japan All rights reserved.

送した際の遅延時間を計測した．320x240 画像及び float64 配列の遅延時間はどちらも平均 2ms 以下であり，白線認識 FPGA の処理時間と比較して影響しない範囲であると考えられる．一方，1920x1080，640x480 の 2 つの画像サイズでは遅延時間にばらつきがあることから，車体制御に影響を及ぼす可能性が考えられる．車体制御には安定した周期で処理を行う事が求められる為，遅延時間に 10~20ms のばらつきが発生する事は望ましくない．その為，通信遅延のばらつきによる車体制御の安定性の低下を回避する為に，ROS2-FPGA ノード化する際に入力画像は ROS2 通信のメッセージには載せず，動作検証の際には ROS2-FPGA ノードが直接カメラからの画像を受け取る設計にした．

5.3 従来手法との比較

従来手法 1，2 と提案手法について，それぞれ通信や FPGA への入出力における制約を表 2 に示す．従来手法 1 では，通信に TCP または UDP による Socket 通信を使用する為，通信相手の IP アドレスやポート番号と言った情報が必要であった．それに対して従来手法 2 及び提案手法では，ROS2 を用いる為通信相手の情報は不要であり，通信の際には同一のトピックを指定することでデータの送受信が可能である．また，通信に ROS2 を用いることで，rosbag や rostopic といったトピックを流れるデータを確認できるツールが利用可能な為，動作検証に有益であると考えられる．

次に，送受信するデータについて，従来手法 1 では送信の際 byte 形式にエンコードする必要があり，受信する際には受信データをデコードする必要があった．また，従来手法 2 では ROS2 のメッセージ型に準拠したデータや，ユーザが定義した独自のメッセージ型が利用可能であった．一方で，提案手法では FOrEST がサポートする int32, float64 といった ROS2 のプリミティブメッセージ型のうち一部のデータ型に限られる．また，FPGA の入出力においても FOrEST が用意する jinja テンプレートに含まれる一部のデータ型のみ使用可能である．すなわち，提案手法には FOrEST がサポートする一部のデータ型を扱うケースにおいて動作検証が可能という制約がある．

図 6 に各手法における実装上の作業量をコード量行を示す．従来手法では，いずれも PYNQ ドライバの実装において 60 行のコード量を要した．また，通信機能の実装においては，従来手法 1 では Socket 通信に 28 行，従来手法 2 では ROS2 のコールバック関数実装に 25 行のコード量が必要であった．一方提案手法に必要なコード量は，テンプレートに記述した 24 行のみである．総コード量を比較すると従来手法 1 の 72.7%減，従来手法 2 の 71.8%減であることがわかる．したがって，提案手法によって動作検証に必要なコード量を削減した事を示した．

表 2 各手法における実装上の制約

	従来手法1	従来手法2	提案手法
通信相手の情報（IPアドレスなど）	必要	不要	不要
送受信データ	byte形式	ROS2のメッセージ型に準拠	int32,float64などの一部のデータ型のみ
FPGAの入出力	無し	無し	int32,float64などの一部のデータ型のみ

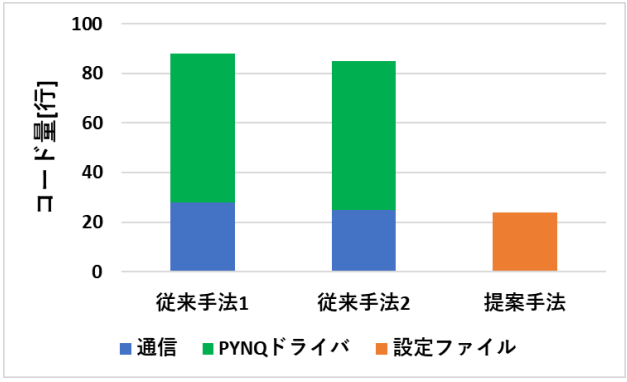

図 6 コード量の比較

5.4 異なるネットワーク環境下での動作検証

図 7 に TCP による Socket 通信を用いた SW（CPU）＋HW（FPGA）の場合，ROS2 を用いた SW+HW の場合，そして SW 実行（CPU のみ）の場合のシステム構成及び実行時間の計測範囲を示す．SW+HW では FPGA への書き込み，FPGA 実行時間，FPGA からの読み込み時間を計測し，SW 実行では白線検出ノードでの実行時間を計測した．なお，SW で行う前処理の実行時間は共通部分である為，実行時間には含まないものとした．

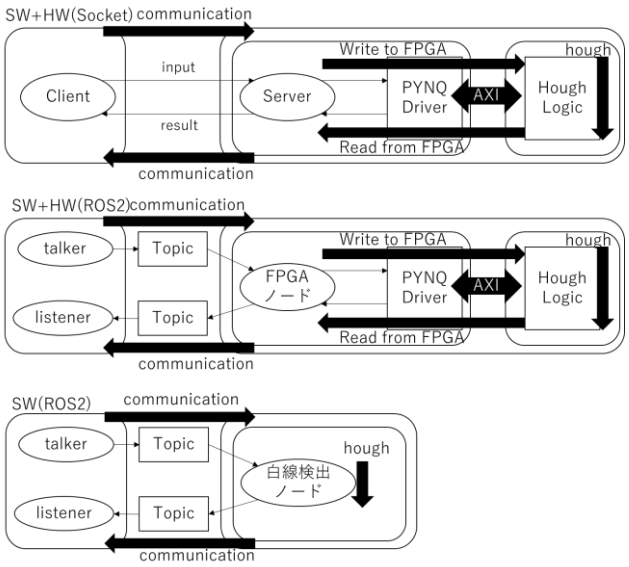

図 7 システム構成と計測範囲

Copyright © 2022 by
The Institute of Electronics, Information and Communication Engineers and
Information Processing Society of Japan All rights reserved.

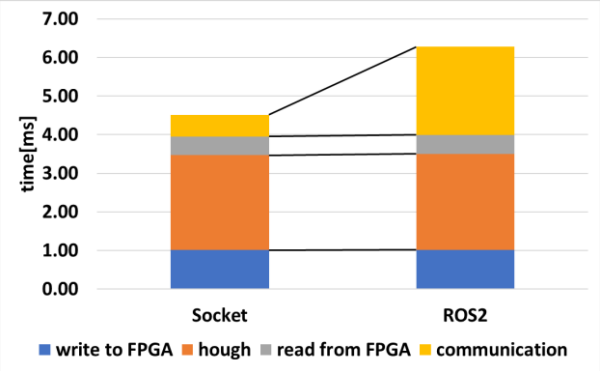

図 8　通信遅延の比較

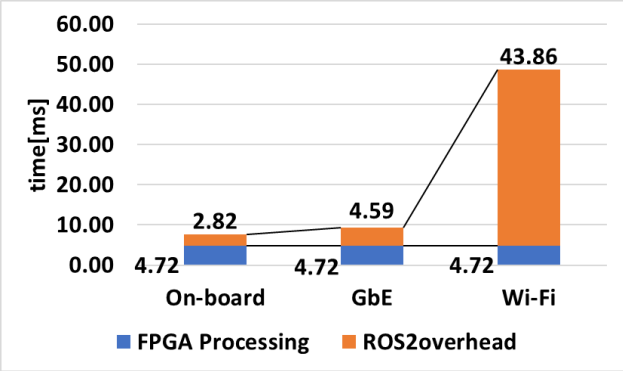

図 10 各環境の実行時間と遅延時間(QVGA)

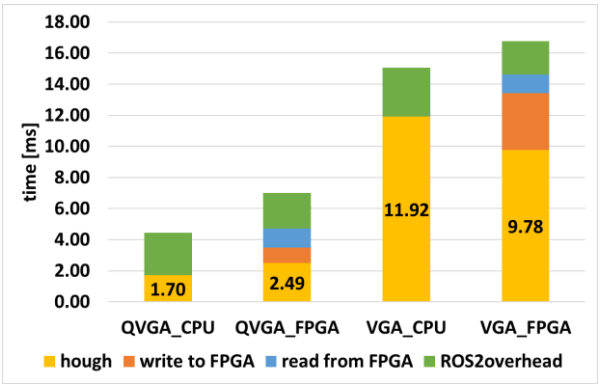

図 9　実行時間（CPU，FPGA）

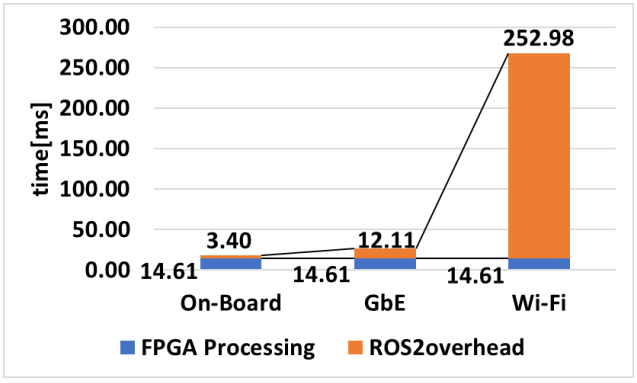

図 11 各環境の実行時間と遅延時間(VGA)

図 8 に Socket 通信と ROS2 通信における通信遅延の比較を示す．Socket 通信では提案手法よりも多いコード量を必要とするが，通信遅延はわずか 0.56ms であった．一方提案手法で用いた ROS2 通信では，通信遅延は 2.28ms であり，これは Socket 通信における通信遅延の約 4.1 倍に相当する．

次に，画像サイズを変えた場合の ROS2 における実行結果を図 9 に示す．VGA，QVGA どちらの画像サイズの場合でも，CPU のみの方が実行時間は短かった．一方で，直線検出処理を行っている Hough だけに着目すると VGA では FPGA の方が実行時間は短い事がわかる．これは FPGA への書き込みに時間がかっており，QVGA に比べて VGA の方が write の時間が占める割合は大きい為，性能ボトルネックになっていると言える．したがって，FPGA ノードとして処理全体を高速化する為には FPGA モジュールの性能だけでなくメモリアクセス性能を考慮してアプリケーションを選定する必要があると考えられる．

図 10，図 11 に白線検出 FPGA を同一マシン内（Onboard ： 同一 Zynq チップ上に配置した ROS 2 ノード間の通信），有線 LAN による通信（GbE：1000Base-T），無線 LAN による通信（Wi-Fi：IEEE802.11b）の 3 つのネットワーク環境で動作検証を行った際の通信時間の測定結果を示す．GbE 及び Wi-Fi 環境では ROS2-FPGA ノードのみを M-KUBOS 上で動作させ，他のノードは通信相手である PC 上で動作させ，入力画像は ROS2 のメッセージに含め，サイズは QVGA，VGA とした．なお，FPGA ノードにおける白線検出処理にかかる時間は図 9 で示した値を予測値として用いた．QVGA グレースケール画像は，FOrEST で ROS2 ノードの生成を行う際に ROS2 メッセージ型 int32 [19200]となるように，VGA グレースケール画像は int32[76800]となるように，ハードウェア化する C 言語の関数の引数を設定した．すなわち，入力画像データのサイズは，QVGA では 19200x4 = 76.8KB，VGA では 76800x4=307.2KB である．On-board では，いずれの画像サイズの場合にも通信遅延は FPGA 処理時間に比べて小さい為，ROS2 メッセージ転送にかかる通信遅延は許容範囲内であると言える．一方，GbE 環境では FPGA 処理時間と同等の通信遅延が生じており，Wi-Fi 環境では FPGA 処理時間と比較して QVGA においては 9.3 倍，VGA においては 17.3 倍の遅延時間であり，性能ボトルネックになっている事が分かる．また，Wi-Fi の場合の QVGA グレースケール画像転送にかかる通信遅延時間は，43.38ms であることから，通信速度を計算すると約 14Mbps となり，IEEE802.11b の規格である 11Mbps と比較しても妥当な値であると考えられる．一方，GbE の通信遅延時間は 4.59ms であり，通信速度は約 133.86Mbps であった．これは，1000Base-T で期待される 1000Mbps からは遥かに小さいため，

Copyright © 2022 by
The Institute of Electronics, Information and Communication Engineers and
Information Processing Society of Japan All rights reserved.

表 3 ハードウェアリソース使用量

Resource	VGA(640x480)	QVGA(320x240)
LUT	303781/522720(58%)	166040/522720(31%)
FF	95186/1045440(9%)	25159/1045440(2%)
BRAM_18K	735/1968(37%)	191/1968(9%)
DSP48E	0/1968(0%)	0/1968(0%)

通信以外の要因での遅延が発生していると考えられる．

Society5.0 における MEC，クラウド環境では 5G の高速無線通信によって FPGA モジュールをネットワーク上に配置する事が想定される為，今後の課題として 5G 環境での通信遅延の計測を行う事が考えられる．いずれの環境においても同様の ROS2 パッケージを用いて検証を行った結果，異なるネットワーク環境でも同様の動作を行える事が分かった．したがって，提案手法はネットワーク構成に依存しない FPGA モジュールの動作検証が可能である事を示した．

表 3 に評価に用いた FPGA モジュールのハードウェアリソース使用量を示す．VGA 画像を対象とした回路では，LUT の使用率は 58%であり，M-KUBOS ボードの回路規模を考慮すると LUT 使用量は大きいと考えられる．白線検出 FPGA のようなロボットに搭載する FPGA モジュールの回路規模は小さい為，動作検証の段階で実行環境を考慮し，リソースを削減した回路設計をする必要があると言える．

6. 結論

本研究では，Society5.0 における IoT デバイス，MEC，クラウドにおいて，FPGA モジュールを様々な場所に配置することを想定し，ROS2-FPGA ノードの枠組みで動作検証を行う手法を提案した．

提案手法により，C/C++言語からの高位合成によって生成した FPGA モジュールを ROS2-FPGA ノード化するツール FOrEST を用いる事で，従来手法よりも動作検証に必要なコード量を 72.7%削減して検証可能な事を示した．一方で，動作検証に ROS2 を用いた事で従来手法の 0.56ms に比べて，約 4.1 倍の 2.28ms の通信遅延が生じることが分かった．また ROS2 における通信遅延は全体の 36.3%に相当することから，FPGA モジュールに実装した処理の遅延時間によっては通信遅延がボトルネックになる可能性があると言える．

FPGA モジュールの動作環境を同一マシン内のローカル通信，有線 LAN 接続による通信，無線 LAN による通信で動作検証を行い，提案手法では FPGA モジュールのネットワーク環境が異なる場合でも容易な動作検証が可能であることを示した．その際，QVGA グレースケール画像のメッセージ転送において，同一マシン上では通信遅延は十分許容範囲内である一方，GbE 環境では FPGA 処理時間と同等の遅延時間，Wi-Fi(IEEE802.11b)環境では 9.3 倍の遅延時間が発生しており，通信遅延がボトルネックである事を示した．今後の課題としては，HLS ベースの FPGA モジュールの動作検証だけでなく OpenCL カーネルを使用した FPGA モジュールの動作検証手法の確立が挙げられる．また，4G-LTE，5G などの実際に Society 5.0 に用いられる通信環境での評価が考えられる．

謝　辞

本研究は,JST,CREST,JPMJCR19K1 の支援を受けたものです.

文　献

[1] 稲毛琢己, 弘中和衛, 飯塚健介, &天野英晴, “M-KUBOS を用いた PYNQ クラスタの構築”, 研究報告システム・アーキテクチャ (ARC), 2021

[2] ROS2 github, https://github.com/ros2/ros2

[3] 高瀬英希, 祐源 英俊, “mROS 2：組込みデバイス向けの ROS 2 ノード軽量実行環境”, 2022

[4] PALTEK, M-KUBOS, https://www.paltek.co.jp/mcube/index.html (2022 年 6 月 21 日閲覧)

[5] Lienen, Christian, Marco Platzner, and Bernhard Rinner. "ReconROS: Flexible hardware acceleration for ROS2 applications." In 2020 International Conference on Field-Programmable Technology (ICFPT), pp. 268-276. IEEE, 2020.

[6] Caba, J., Rincon, F., Dondo, J., Barba, J., Abaldea, M., & López, J. C. Testing framework for on-board verification of HLS modules using grey-box technique and FPGA overlays. Integration, 68, 129-138.(2019).

[7] 庄嶋篤，山脇彰. "高位合成画像処理ハードウェアのための仮想周辺機器による汎用検証環境の開発." 産業応用工学会全国大会講演論文集 2021. 一般社団法人 産業応用工学会, 2021.

[8] 千原悠真, 山脇彰. "高位合成ハードウェアの実機における汎用検証環境の構築." 産業応用工学会全国大会講演論文集 2018 . 一般社団法人 産業応用工学会, 2018.

[9] D. Pinheiro Leal, M. Sugaya, H. Amano, T. Ohkawa, "Automated Integration of High-Level Synthesis FPGA Modules with ROS2 Systems", International Conference on Field Programmable Technology (FPT), 2020.

[10] Amano, Hayato, Hayato Mori, Akinobu Mizutani, Tomohiro Ono, Yuma Yoshimoto, Takeshi Ohkawa, and Hakaru Tamukoh. "A dataset generation for object recognition and a tool for generating ROS2 FPGA node." In 2021 International Conference on Field-Programmable Technology (ICFPT), pp. 1-4. IEEE, 2021.

Copyright © 2022 by
The Institute of Electronics, Information and Communication Engineers and
Information Processing Society of Japan All rights reserved.