Vol.2019-SLDM-189 No.17
Vol.2019-EMB52 No.17
2019/11/14

ROS 2のための協調システム設計支援フレームワーク

田村 爽[1,a)]　新田 泰大[1]　高瀬 英希[1,2]

概要：近年のロボット開発では，その開発を促進するプラットフォームである ROS および ROS 2 が広く用いられている．しかし，自律移動ロボットにおける消費電力の制約や，人と同じ空間で動くロボットの安全性を保証するためのリアルタイム性など，様々な課題が存在している．これらのロボットにおける問題を解決するために，FPGA を活用する方策が挙げられる．しかし，FPGA におけるハードウェアの設計コストは高く，ロボットシステムに組み込むことは容易ではない．そこで本稿では，ROS 2 のための協調設計フレームワークを提案する．フレームワークによってハードウェアおよび SW/HW 間のインタフェースを作成することで，ロボット設計者は ROS 2 アプリケーションから通常のデバイスと同じようにハードウェアを制御することが可能となる．提案するフレームワークはハードウェアによるシステムの高性能化に貢献する．提案するフレームワークの有効性を評価するため，ロボットシステムで頻用される線形識別機による物体識別処理のハードウェア化を実装した．この結果，FPGA を活用することによって，ARM プロセッサとの通信時間を含めた場合でも約 275 倍の高速化を実現した．

キーワード：プログラマブル SoC，FPGA，ROS 2

A supporting framework to codesign for ROS 2

So TAMURA[1,a)]　Yasuhiro NITTA[1]　Hideki TAKASE[1,2]

Abstract: In recent robot development, ROS and ROS 2, which are platforms that promote the development, are widely used. However, there are various problems in robot development, such as restrictions on power consumption and real-time performance to guarantee the safety of robots that move in the same space as humans. The problem with these robots can be solved by using FPGA. However, the difficulty in designing hardware in FPGAs becomes an obstacle to incorporation in robot systems. In this paper, we propose a collaborative design framework for ROS 2. Create HW and SW / HW interfaces in the proposed framework. Robot software designers can control hardware from a ROS 2 application just like a USB device. The proposed framework contributes to high-performance hardware systems. Evaluate the proposed framework. We implemented hardware for object identification processing using a linear classifier frequently used in robot systems. As a result, the FPGA achieved about 275 times faster than the ARM processor.

Keywords: Programmable SoC，FPGA，ROS 2

1. はじめに

近年，ロボットは様々な状況で活用されており，その技術開発および研究も盛んに行われている．無人ドローンや AI ロボットなどは，周囲の状況を分析することによって，人間の判断および指示を介在せずに自身を制御することが求められる．制御のための複雑なソフトウェアを実行するためには，一般には高消費電力なプロセッサを必要とする．しかし，ロボットの小型化や長時間稼働のためには，より低消費電力な演算処理装置が必要とされる．

組込みシステムにおいて高度な処理を低消費電力で実現する方法として，プロセッサと FPGA を同一チップ上に集積したデバイスであるプログラマブル SoC を用いることが挙げられる．プログラマブル SoC を用いたシステムでは，

[1] 京都大学大学院情報学研究科
[2] JST さきがけ
[a)] emb@lab3.kuis.kyoto-u.ac.jp

負荷が大きい処理を FPGA 上のハードウェアで，柔軟性の求められる処理をプロセッサ上のソフトウェアで実行するように設計することで，高性能化と低消費電力化の両立を図ることが可能となる．しかし，プログラマブル SoC の利点を活かしたシステムの設計には，ハードウェアとソフトウェアのそれぞれの深い知識が必要となる．Xilinx プログラマブル SoC 向けに提供されている，Xilinx SDSoC[1] はソフトウェア志向での協調設計を実現する協調システム統合設計環境である．SDSoC を活用することによって，設計者はハードウェアやインタフェースの知識に精通することなく，ハードウェアを活用したシステムを容易に構築できるようになることが期待される．

ロボット開発においても，さらに多岐にわたる専門知識が要求される．ロボット開発においては生産性を向上させるために様々なフレームワークやミドルウェアが提供されている．ROS 1[*1][2] および ROS 2[3] はロボット用ソフトウェアの設計生産性の向上に資するプラットフォームである．ソフトウェア部品はノードとして表現され，ノード間の通信レイヤのフレームワークが提案されている．また，様々なシミュレータやデバッグツールによるロボット開発がサポートされ，再利用性や保守性に優れたエコシステムが構築されている．

本稿では，ROS 2 を用いたロボットシステムにおいて FPGA の導入を促進する協調設計のフレームワークを提案する．提案するフレームワークを用いることで，FPGA および ROS 2 それぞれの利点を活用できる．FPGA の利用は ROS 2 のノードの中の一部の機能としてラップすることにより，FPGA を利用するための ROS 2 アプリケーション設計を考慮することなくハードウェア化による高速化を行うことができる．フレームワークを活用したハードウェア化による高速化については，2 種類の方式を提案する．1 つ目はセンサからのデータを直接 FPGA で受け取り，CPU に送る前に前処理を行う方式である．これにより，画像データの歪曲補正，二値化およびエッジ検出などの処理の FPGA へのオフロードおよび高速化を可能とする．2 つ目は，CPU で処理しているデータの一部を FPGA に送り，その結果を FPGA から受け取る方式である．これは，CPU と FPGA の間での通信がオーバーヘッドとなる可能性があるが，CPU で柔軟に前処理したデータを FPGA で処理することができる．

さらに，SDSoC を用いたソフトウェア志向のワークフローも提案する．SDSoC を用いることによって，設計者は ROS 2 および SDSoC の知識のみでハードウェアを含むシステムの設計を行うことができる．

本フレームワークにおける、ROS 2 において提供されるシミュレータやデバッグツール用いた協調設計についても議論する．ROS 2 のツールを活用することで，FPGA を含むシステムにおいてもロボット開発を効率的に行うことが可能となることを示す．

[*1] 本稿では，ROS 2 と明確に区別する際には ROS 1 と記す．

2. ROS 2（Robot Operating System 2）

2.1 概要

ROS 1（Robot Operating System）は，WillowGarage とスタンフォード大学が共同で開発したロボットソフトウェア開発ミドルウェアである．Open Robotics によってオープンソースコミュニティで開発が進められている．ロボットソフトウェアの再利用性を高めることを目標として開発された．

ROS 2 は ROS 1 における課題を解決するために，2014 年から開発されている．要点は以下のようになる．

- 協調ロボット環境への対応
- リアルタイム性の担保されたシステムへの対応
- 様々な通信状況に対応するための QoS のサポート
- マイコン等の組込みでの動作

実装面の特徴として，通信層に国際ミドルウェア標準である DDS が採用されている．DDS は様々なベンダや，もしくはオープンソースとして提供されている．ROS 2 では，様々な DDS に対応する rmw（ROS middleware）が実装されており，rmw の API を介してユーザーアプリケーションや rcl（ROS client library）から DDS を利用することが可能である．

2.2 ROS の通信モデル

ROS では，プログラム部品をノードとして表現し，複数のノードを組み合わせることでロボットシステムを構築する．ノード間の通信層を提供することで，これによって複雑なアルゴリズムを実行するアプリケーションノードとハードウェアの制御を行うノードを分離することが可能となる．ノード間の通信は出版購読通信と呼ばれる方式で実現される．トピックとしてデータの種類を分別し，パブリッシャがトピックに対して出版したデータを各サブスクライバノードが自身のタイミングで購読する一対他の通信モデルである．

図 1 にトピックによるノード間での出版購読通信のモデルを表す．扱うトピック名が同一であれば，ROS ロボットシステムにおいて，ハードウェアとそのノードを他のハードウェアと対応するノードに容易に変更できる．

ROS 1 ではノード毎に一つのプロセスとして管理されており，プロセス間の通信は TCP/IP 層を介する．これは，低性能な PC において画像等の大きなデータをやりとりする際のオーバヘッドとなっていた．そこで ROS 2 では，必要に応じて複数のノードを一つのプロセスで実行する Executor 機能が提供されている．これにより，複数のノードを一つのプロセスで実行することによるゼロコピーを実

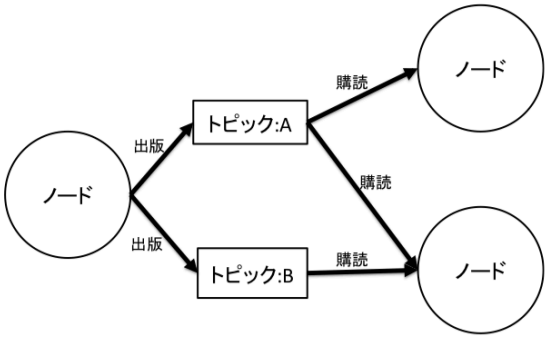

図 1 ROS におけるノード間の出版購読通信モデル

現し，ノード間通信のオーバーヘッドが解消された．

2.3 関連研究

[4] では，ROS 1 に FPGA を導入可能とする技術として，ROS 準拠 FPGA コンポーネント技術が提案された．この論文では，(1) プログラマブル SoC を用いて構築する手法および (2) FPGA 回路のみで ROS ノードを実現する手法が提案された．(1) では Xillinux および Xillybus[5] を用いており，通常のソフトウェア ROS 1 ノードが ARM プロセッサ上で動作するため，ソフトウェア ROS 1 システムと高い親和性，相互運用性を持つ．しかし，Xilibus は FPGA 上の回路と ARM プロセッサ間のインタフェースに FIFO を用いているため，連続画像などの通信には適していない．(2) では通信遅延削減のためのハードウェア TCP/IP スタックを用いて FPGA 回路のみで ROS ノードを実現し，ROS の通信における大幅な性能向上が可能であることを示した．一方，現状では ROS 1 メッセージ解釈・生成のハードウェア実装が一部のメッセージ型にしか対応していないため，通信に用いることができる型が限定されている．

また，[6] では，FPGA ボードとラップトップを搭載し，ラップトップ上の ROS 1 ノードから FPGA ボード上の資源を扱う手法を提案している．PC の ROS 1 ノードと FPGA ボードの CPU の間のインタフェース，および FPGA ボードの PS/PL 間のインタフェースを自動生成することによって，ROS 1 システムのハードウェア化による高速化を実現している．

3. 協調設計フレームワーク

3.1 フレームワークにおけるワークフロー

提案するフレームワークにおいて，ユーザーが行うべき手順は次の通りとなる．

(1) ROS 2 API を含むアプリケーションから，ハードウェア化したいタスクを選択し，そのソースコードを切り分ける．
(2) ハードウェアを C++で記述し，高位合成によってハードウェアを作成する．
(3) 任意のハードウェア構成を設計し，論理合成を行う．
(4) ハードウェアを利用するための SW/HW インタフェース（以下 IF）を作成する．論理合成の結果から，ハードウェアが接続される先の CPU 物理アドレスを取得できるため，デバイスドライバもしくは UIO（User space I/O）を作成する．これらを ROS 2 アプリケーションから呼び出すことによって，ソフトウェアからハードウェアが利用できる．
(5) 他の ROS 2 ノードと合わせて実行することで，ハードウェアを利用したシステムを構築する．

以上により，ROS 2 がインストールされたプログラマブル SoC 上において，ハードウェアを利用したアプリケーションを実行することが可能となる．設計者の技術に応じて，高位合成において pragma を用いた高性能なハードウェア設計や，DMA による SW/HW 間のデータ通信なども実装可能である．

3.2 フレームワーク活用のためのシステム構成

提案するフレームワークを活用するための，OS を含めたシステム構成について議論する．本節ではハードウェアによるシステムの高性能化を目指す場合，またリアルタイム性を求める場合のそれぞれについて，解決するためのシステム構成を提案する．

3.2.1 ハードウェアによるシステムの高性能化

ROS 2 アプリケーションを動作させる OS としては，Ubuntu が広く採用されている．そこで，本項では，CPU で動作させる OS に Ubuntu を用いた場合のシステム構成を図 2 のように提案する．ARM には Linux をインストールし，ROS 2 を動作させる．ROS 2 からはドライバノードやデバイスドライバを介することによって，FPGA 上に構築した回路を操作することができる．この時，図 2 の (a) のように，センサデータを FPGA で受け取り，ハードウェアで処理されたセンサデータを一方的に受信する方法と，(b) のように計算のためのデータを CPU から送り，ハードウェアで処理された結果をまた受け取る方法が存在する．(a) は，画像処理のための前処理など，入力データに対して一律に処理すべきことが決まっている場合に適している．(b) は，データに対して柔軟な処理を行う必要があるが，そのうちの一部の処理をアクセラレートする場合に適している．

(a) では常にハードウェアを動作させることができるが，(b) の場合，CPU と FPGA が通信をしている間は，ハードウェアが動作していないこととなる．そのため，CPU と FPGA の通信時間が大きい場合は，ハードウェアによる効果が十分に得られない可能性がある．

3.2.2 リアルタイム制御を可能とする構成

協働ロボットや自動運転においては，大量のプログラムを動作させつつ，重要な安全性の保証に関わる部分にはリ

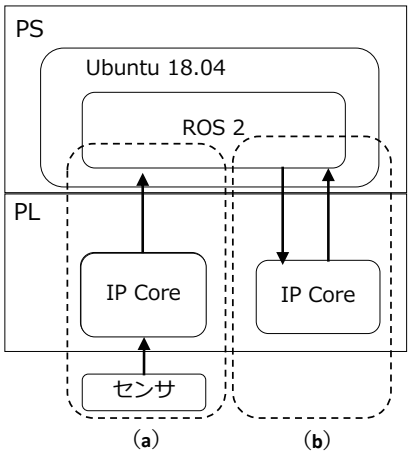

図 2　Ubuntu を用いた場合のハードウェア化による高速化

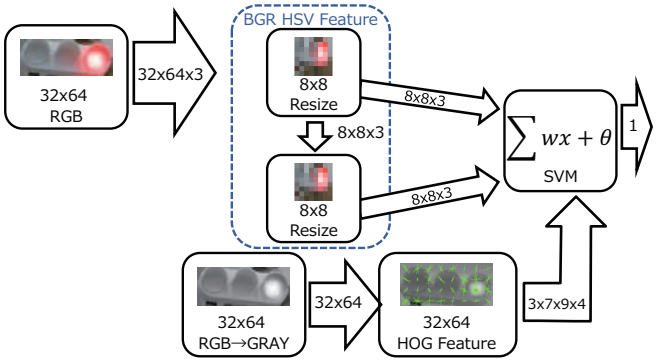

図 3　線形識別器の概要

アルタイム性が求められる．ROS 2 は RTOS 上でも動作させることが可能な設計となっており，リアルタイム性があるシステムを構築することが可能である．そのため，協働ロボットや自動運転車に ROS 2 を用いることが注目されている．

様々な ROS 2 ノードが存在するロボットの場合，全てのノードをリアルタイム制御することは困難であるため，RTOS と Linux のヘテロジニアスなアーキテクチャが適している．FPGA を RTOS 上の ROS 2 から制御することにより，リアルタイム性を求められる処理を FPGA で行うことが可能である．

提案するフレームワークを活用し，RTOS 上の ROS 2 からのハードウェア操作は現時点では達成できていない．今後，本フレームワークを RTOS 対応させるように拡張することで，FPGA を活用したリアルタイム性のある ROS 2 システムを設計することが可能となる．

3.3　適用事例

図 2 の（a）および（b）に相当する IP コアを作成し，それぞれの評価を行なった．ターゲットボードには Xilinx Zynq Ultrascale+ MPSoC ZU3EG を搭載したプログラマブル SoC 開発ボードである Ultra96 を使用し，いずれの処理も IP コアの実装には高位合成ツール Vivado HLS[7] を使用した．

（a）に相当する IP コアでは，センサに PCam 5C を用いた．PCam 5C は MIPI 規格により信号が PL 部に直接接続されている．画像データを受け取り，画像を 2 値化およびエッジ検出を行なった結果を CPU に送信する処理を行っている．V4L2 インタフェースに一部準拠したデバイスドライバを作成することで，Linux システム上では通常の USB Web カメラと同様に V4L2 API を用いて処理済みの画像を取得することができる．また，PCam 5C で取得できるフレームレートと同様に最大 30FPS が可能である．

（b）に相当する IP コアでは，USB Web カメラから取得された画像に対し，線形 SVM 識別器およびスライディングウインドウ法による赤信号検出器を実装した．スライディングウインドウ法では単一の識別器を全候補領域に対して適用する．ウインドウサイズは 32pix*64pix とし，ストライドを縦横ともに 8pix とした．設計した IP コアは 320pix*240pix のフレーム画像を入力として受け取り，全ウインドウ候補 891 個の識別結果（赤信号である確率）を出力する．線形識別器の処理を図 3 に示す．縮小されたウインドウ画像の BGR 画素および HSV 画素，さらに HOG(Histogram of Gradients) を特徴量として使用する．

本 IP コアの実装には，HOG および SVM を使用した物体検出器の HW 実装に関する研究 [8] を参考にした．線形 SVM では各特徴量の要素に関する計算が独立しているため，1 つのウインドウの一部の特徴量が計算された時点でそのウインドウに関する推論計算を開始することができる．これにより処理全体をパイプライン化することが可能になる．また，FPGA 向けに浮動小数点の代わりに固定小数点を用いることで計算を軽量化を施した．

VivadoHLS による高位合成結果について述べる．合成後のレポートによるとクロックレイテンシは最小で 160936, 最大で 207736 となった．リソース使用率を表 1 に，SW/HW

表 1　赤信号検出器のリソース使用量

Resource	Utilization	Available	Utilization %
LUT	29848	70560	42.3
LUTRAM	1449	28800	5.03
FF	30219	141120	21.4
BRAM	77	216	35.6
DSP	112	360	31.1
BUFG	3	196	1.53

表 2　SW/HW 実行時間

	Processing Time[ms]	fps
SW	1700	0.58
HW	2.07	483
SW/HW IF + HW	6.22	160

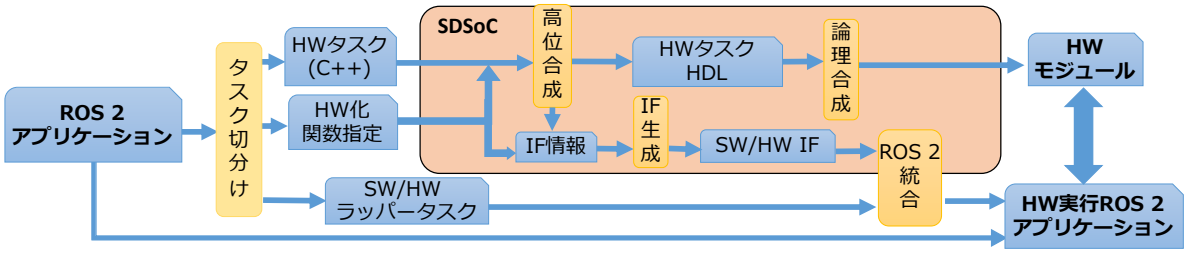

図 4　SDSoC を用いたフレームワークの全体像

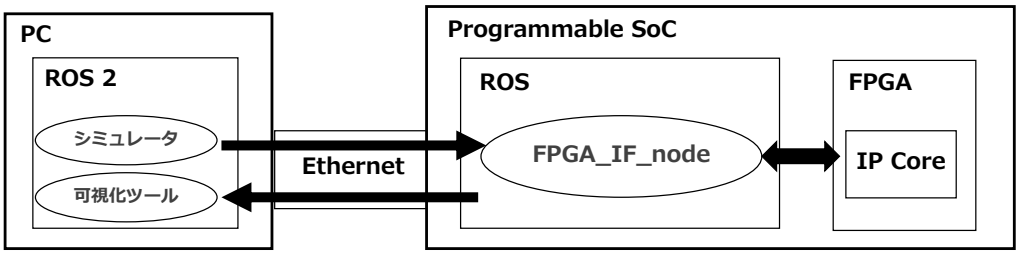

図 5　Gazebo とプログラマブル SoC の通信

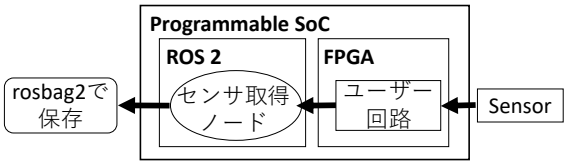

図 6　rosbag2 の利用例 1

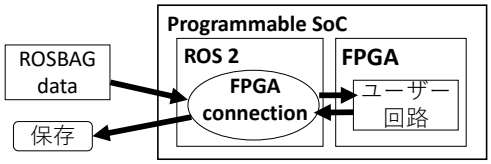

図 7　rosbag2 の利用例 2

の実行時間での比較を表 2 に示す．このリソース使用率は，PS/PL 間のデータ転送に必要な AXI DMA および学習済の重みを保存する BRAM を含めた設計全体のリソース使用率を表している．実行時間に関しては，SW/HW 間の通信時間を含めても 1 フレームの処理が 6.22milisec で完了し，Ultra96 ボード上のプロセッサのみで処理した場合に対して約 275 倍の高速化を実現した．

4.　ソフトウェア志向の協調システム開発

3.1 において提案したワークフローでは，SW/HW IF を作成するために Linux のドライバおよびハードウェア操作のための設計が要求されるため，開発コストが大きい．そこで，ハードウェアや Linux ドライバの知識を持たないソフトウェアエンジニアのためのフレームワーク活用として，図 4 のような SDSoC を活用したワークフローを提案する．

設計者の行うべき作業は次の通りである．

(1) ROS 2 API を含むアプリケーションから，ハードウェア化したいタスクを選択し切り分ける．ハードウェア化部以外を SW/HW ラッパータスクとする．

(2) ハードウェア化したい C++記述と関数を SDSoC の入力とすることで，ハードウェアおよび，ハードウェアを利用するための SW/HW IF が作成される．

(3) SW/HW IF を，切り分けた SW/HW ラッパータスクと統合する．

(4) ラッパーノードを他の ROS 2 ノードと同時に起動することにより，HW を利用した ROS 2 システムを構築する．

ハードウェアにしたい C++記述を指定するだけで，ハードウェアと SW/HW IF が生成されるため，設計者は SDSoC と ROS 2 の知識のみでシステムを構築することが可能となる．

5.　ハードウェア開発における ROS ツール活用

ロボット開発においては，実機で動作する前にシミュレータを用いての動作確認や，デバッグツールを活用した実機での検証を行う．提案するフレームワークを用いてハードウェアモジュールを ROS 2 から利用できるように

した場合，FPGA を含むシステムでも ROS 2 のツールを用いた開発が可能になる．次のようにシステムを構築することで，ROS 2 のシミュレータでハードウェアモジュールへの入出力を実験することや，ハードウェアモジュールへの入出力の保存や再現を行うことが可能である．

5.1 Gazebo

ROS 2 では，同一ネットワーク間にあれば，他の PC 上のノードともローカルに存在するノードと同様に通信することができる．そのため，プログラマブル SoC を PC と同一ネットワークに繋ぐことにより，図 5 のように PC とプログラマブル SoC で ROS 2 メッセージのやりとりが可能となる. これにより，PC で 3D シミュレーター Gazebo[9] を動かし，生成したセンサーのデータ（画像，LiDAR 等）をプログラマブル SoC に送信し，処理を行った結果を PC でリアルタイムに確認することが可能である．

5.2 rosbag2

rosbag2 は，ROS ノード間でやりとりされるデータを保存し，再現できるデバッグツールである. 実ロボットで得られるセンサデータを保存し，同じデータを何度でも流すことができるため，実装のアルゴリズムの改善や比較，パラメータ調整などに活用される．これは FPGA の実装にも応用することが可能である．rosbag2 を用いることで，図 6 のように FPGA による処理の結果を保存することや，図 7 のように，複数の FPGA の実装に同じデータを再生し，結果を保存して比較することにより，容易に比較することができる．

6. おわりに

本稿では，ROS 2 のための協調システム設計支援フレームワークを提案し，有用性について述べた．FPGA を活用することによって，ハードウェア化による高速化，消費電力の削減や，リアルタイム性のあるシステムの高性能化を実現できる．

今後の課題としては，提案したフレームワークを用いて，RTOS 上で動作させた ROS 2 からの FPGA 利用についてが挙げられる．このためには，フレームワークが対象とする OS を ROS 2 が動作する RTOS に対応させる必要がある．また，RTOS 上で動作する ROS 2 については未だ開発途上の機能もあり，今後の ROS 2 の開発に合わせての対応が求められる．

謝辞

本研究の一部は，JST さきがけ JPMJPR18M8 の支援を受けたものである．

参考文献

[1] Xilinx，https://japan.xilinx.com/products/design-tools/software-zone/sdsoc.html
[2] Open Robotics, http://wiki.ros.org/
[3] Open Robotics, https://index.ros.org/doc/ros2/
[4] K. Yamashina, T. Ohkawa, K. Ootsu, T. Yokota, ”Proposal of ROS-compliant FPGA Component for Low-Power Robotic Systems -case study on image processing application-”, International Symposium on Foundations and Practice of Security The Computing Research Repository, 2015.
[5] Xillybus, http://xillybus.com/
[6] Y.Ishida, T.Morie, H.TamukohA, ”Hardware Accelerated Robot Middleware Package for Intelligent Processing on Robots”, IEEE International Symposium on Circuits and Systems (ISCAS), 2018
[7] Xilinx, https://www.xilinx.com/products/design-tools/vivado/integration/esl-design.html
[8] Luo, J. & Lin, C. (2018). Pure FPGA Implementation of an HOG Based Real-Time Pedestrian Detection System. Sensors. 18. 1174. 10.3390/s18041174.
[9] Gazebo, http://gazebosim.org/