2019 IEEE/ACM 2nd International Workshop on Robotics Software Engineering (RoSE)

High Level Synthesis of ROS Protocol Interpretation and Communication Circuit for FPGA

Takeshi Ohkawa
Graduate school of Engineering
Utsunomiya University
Utsunomiya, Japan
ORCID: 0000-0002-6536-6439

Yuhei Sugata
Graduate school of Engineering
Utsunomiya University
Utsunomiya, Japan
sugata@virgo.is.utsunomiya-u.ac.jp

Harumi Watanabe
Department of Embedded Software
Tokai University
Tokyo, Japan
harumi@wing.ncc.u-tokai.ac.jp

Nobuhiko Ogura
Faculty of Informatics
Tokyo City University
Yokohama, Japan
ogura@tcu.ac.jp

Kanemitsu Ootsu
Graduate school of Engineering
Utsunomiya University
Utsunomiya, Japan
kim@is.utsunomiya-u.ac.jp

Takashi Yokota
Graduate school of Engineering
Utsunomiya University
Utsunomiya, Japan
yokota@is.utsunomiya-u.ac.jp

***Abstract* — This paper proposes a method with encapsulating hardware description on ROS nodes for improving the productivity of robot development. To realize intellectual robots, we should satisfy constraints involving high performance, low power consumption, and high energy efficiency. FPGA (Field Programmable Gate Array) is well-known to satisfy the constraints. In conventional methods, to apply FPGA into software description of ordinary ROS system, we need to describe codes for performing the conversion process between the abstraction level of the ROS message level and the abstraction degree of the low FPGA level. Thus, the describing cost causes to the productivity problem. This proposal contributes to simplifying the describing part of FPGA in the conversion process. In this process, we provide a generating mechanism from C/C++ programs into circuits in High-Level Synthesis and integrating communication in the ROS protocol. To evaluate whether the method contributes to productivity, we compare a C/C++ program in the new method with a conventional description in HDL. As a result, the size of the new method was 127 lines, while the conventional was 860 liens. Therefore, we consider this method contributes to improving productivity.**

Keywords— ROS, FPGA, component, High Level Synthesis, Design productivity

I. Introduction

Various hardware accelerators are used to improve insufficient performance of software processing. Most of the general processing has already been realized as dedicated hardware, and dedicated hardware processing is performed in the computer system without consciousness of software developers. For example, GPU (Graphics Processing Unit), which is an LSI(Large Scale Integration) chip dedicated to graphics processing, performs most of graphics processing for screen display and basic image processing of a camera input.

Software libraries such as OpenCV, which is for computer-vision image processing, generally hide these hardware processes. Therefore, software developers can benefit from hardware acceleration without explicitly writing the software description, which is required for acceleration of processing. On the other hand, it is necessary for dedicated software processing used for intelligent robots to write use hardware processing such as GPU. For example, processes used for intelligent robots include stochastic matching processing used for SLAM (Simultaneous Localization and Mapping), Kalman filter, particle filter for sensor input data, and so on. Processing for these intelligent robots has many computations and does not reach the required performance, especially in the case of embedded computing in a robot. Although there are many processing which needs acceleration, it is not always possible to prepare dedicated hardware accelerator. For this reason, robot software is customized so that it can cope with requests such as thinning out data, low resolution images and changing algorithms.

Utilization of FPGA , which is an LSI capable of real time processing with high energy efficiency, is expected to solve the above issue. Since FPGA is an LSI chip that can realize arbitrary digital circuits by a software-like program, it is possible to freely realize parallel processing and memory-hierarchical architecture specialized for the application. Since any circuit can be freely programmed, FPGA can be applied to unprecedented advanced processing such as robot-specific software processing with high energy efficiency.

Difficulty of using FPGA in robots is that designing high-performance circuit is time-consuming since it is done at digital circuit level. In general, application development of FPGA is done at circuit level by RTL (Register Transfer level) using HDL (Hardware Description Language). RTL describes the clock-cycle accurate behavior of all registers which work in parallel so that each line of source code works in parallel. It is good to describe circuit in detail, however, it is humble to describe sequential behavior. So development productivity is much lower than ordinary software. Recently, HLS (High-Level Synthesis) [1] has become popular for generating HDL from software language such as C/C++. The major benefit of HLS is program comprehension. For example, an image processing library [2] in C language for HLS is available, which is easy to understand and reuse. However, a high-performance circuit using HLS still needs expertise knowledge of FPGA [3].

On the other hand, in a wide field of robot engineering, it is not realistic to grasp many necessary technologies for all the specialized fields in robot development. We have proposed ROS-compliant FPGA component [4] in order to introduce any FPGA processing circuit easily to robot systems. ROS (Robot Operating System) [15] is a kind of software platform which supports component-oriented development. By using ROS-compliant FPGA component, FPGA can be easily introduced to robots so reusability of FPGA circuits can be improved. We also proposed an automated design tool [12] which generates the communication path between ROS message and circuit as software. However, it is still a problem that its communication performance is very low because of embedded processor, which needs less power but low in processing performance. Therefore, the development of ROS protocol interpretation and communication circuit for high-performance data communication is expected to be automated.

The contributions of this paper are:

- Description of the detailed ROS-compliant FPGA component co-operated by software and hardware
- Proposal of a novel design flow by HLS (High-Level Synthesis) with better program comprehension
- Feasibility study and evaluation in source lines of code to show improvement in design productivity

II. Development of ROS-Compliant FPGA Component

This section describes the development of ROS-compliant FPGA component. Expected role of the FPGA is accelerating an application processing such like computer-vision algorithm, filter and so on. At the same time, the application processing on the FPGA must communicate with the outer system, i.e. ROS software nodes surrounding the application processing. Therefore, the FPGA component must have two interfaces in addition to the "Application processing", that are "Interface for input" and "Interface for output" as shown in Figure 1.

"Interface for input" must have the following functions:

- Input a message from a topic subscribed in advance
- Interpret the message in ROS protocol and extract/marshal data for the application processing
- Send the marshaled data to the application processing

Also, "Interface for output" must have the following:

- Receive result from the application processing
- Generate a message in ROS protocol from the result of the application processing
- Publish the result to a topic advertised in advance

"Application processing" can be developed in various ways for each application domain. The examples for computer vision and machine learning application are xfOpenCV [5] in reVISION Platforms from Xilinx Inc., OpenVINO [6] Toolkit from Intel, and so on. For other domains, OpenCL [7], VivadoHLS [8], and any other HLS tools can be a generic solution for computing tasks like scientific computations [7], and complex applications including dynamic data structure [8].

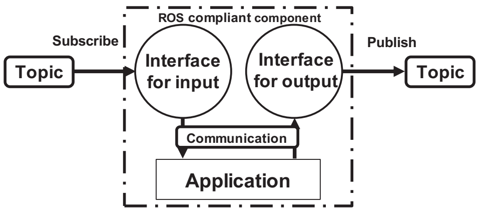


Figure 1 Structure of ROS-compliant FPGA component

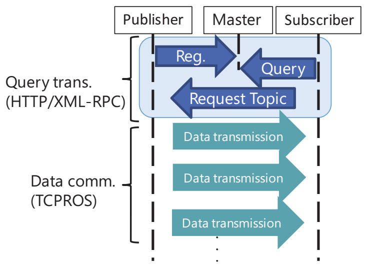


Figure 2 Sequence of ROS message communication

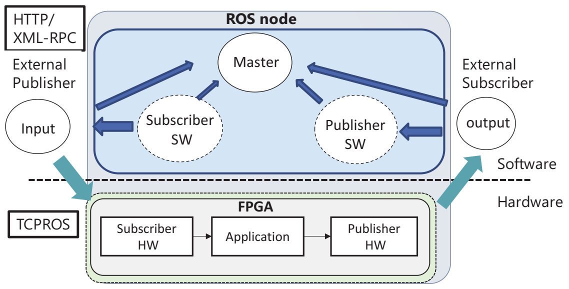


Figure 3 Co-operation of software and hardware to realize ROS node by separating XML-RPC and TCPROS protocol

This paper, though, focus on how to develop the two interfaces which (1) communicate with other ROS node, (2) convert data between ROS protocol message and application processing in FPGA, and (3) send/receive data to/from application circuit. The three STEPs are all done in FPGA.

STEP 1) Communication with other ROS node

The communication sequence to work as a publisher or a subscriber in ROS system is show in Figure 2. In the beginning, *Publisher* registers its topic information to *Master*. This corresponds to *advertiseTopic*() of ROS-API call. After this, *Subscriber* can query for the registered topic name by *subscribe*() ROS-API call. *Master* works as a name service of topic in ROS system like this. These query transactions are done in HTTP/XML-RPC protocol [13]. After the query transactions, message communication of application data starts in TCPROS protocol. In our previous report [9], a method of accelerating ROS message communication for application data is proposed and exhibited by using FPGA hardwired TCP/IP stack based on the architecture here.

Figure 3 illustrates how the FPGA works as ROS node. The query transactions in XML-RPC are done by software on microprocessor since the transactions do not need high-speed communication. Instead, the data communication in TCPROS protocol is done in FPGA at high-performance.

STEP 2) Conversion of data between ROS and FPGA

Message in TCPROS protocol is almost raw binary data of application [15]. Therefore, the conversion of data is only extraction and marshalling processing. Extraction process includes interpretation of ROS message based on the ROSTCP protocol, which has structured data fields with variable length array. Marshalling process is necessary if the data format used the circuit of the application processing is different from ROS message.

STEP 3) Send/Receive data to/from application circuit

Communication between the Interface circuit and Application processing has several choices. The most simple one is direct connection using register, however, it has problem if the coming data overwrites the register. Therefore, inserting FIFO buffer is a realistic approach. Anyway, connection between circuit modules are not tough work.

III. Proposed Design Flow using High Level Synthesis

The development of data conversion/communication processing is an error prone task. In addition, the development TAT (Turn-Around Time) of FPGA is much longer than ordinary software development in general. Therefore, it is necessary to reduce iterations of error and try in the development flow for the ROS/FPGA data conversion processing. In this section we propose a novel design flow of ROS/FPGA data conversion processing using High-Level Synthesis to ease the development.

In our original work [4], the communication was the development bottleneck since the Interface was implemented in software. Our previous work [9], the three STEPs in the FPGA can be connected through hardwired, however, the development was done by describing circuits in HDL (Hardware Description Language). In this paper, we propose a design flow in which all the three STEPs are described in C/C++ language and use High-Level-Synthesis (HLS) to synthesize circuit for the three STEPs.

The parameters of Publisher and Subscriber circuit used for realizing ROS node are different depending on application. That is, the type of ROS messages used is dependent on application. Another parameters are information about node and topic, for example, topic name, node name, network information, and so on. Therefore, the circuit used in the ROS node must be generated for each application. Even a small parameter change, conventional HDL design need long time for compilation and verification. Therefore, automating the process is important to avoid re-spin of the design/implementation process. Therefore, a design flow is necessary to generate ROS node which is capable of communicate with other ROS nodes and utilized high-performance and highly effective FPGA processing.

We propose a design flow is illustrated in Figure 4. The developer of ROS-compliant FPGA component prepares (a) ROS message definition and (b) ROS node configuration. The ROS message definition (a) is a commonly used style in ROS system development. It defines application specific custom message type. Most of popular message types used in robotic software are pre-defined and distributed by official ROS releases, for example, sensor_msgs for various sensor data including camera image and IMU. The ROS node config (b) is used to indicate A generator is used to specify topic name, node name, network information to communicate with other publishers/subscribers/master in the ROS system. Application (c) is provided as C/C++ for HLS. After generator, C/C++ descriptions of subscriber (d) and publisher (e) are obtained. Finally, HLS tool synthesize circuits (f) for FPGA to work as ROS node.

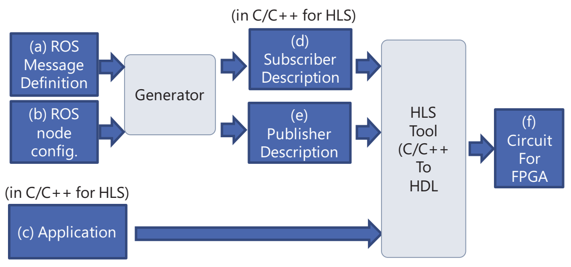


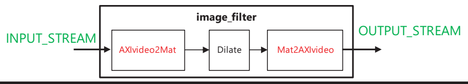
Figure 4 Proposed design flow of generating ROS-compliant FPGA component with HLS tool

IV. Feasibility Study and Evaluation

In the proposed development flow, a generator of Subscriber and Publisher description in C/C++ for HLS is necessary. Before making the generator, it is necessary to understand the description for Subscriber and Publisher well.

To evaluate the feasibility of the proposed design flow, we have implemented ROS node with FPGA including Application, Subscriber and Publisher in C/C++ for HLS. As an HLS tool, Vivado HLS 2017.2 is used. Application is Dilate image processing as an example of simple image processing. The hardware structure and code fragment for the Application are shown in Figure 5. By using Vivado HLS, some OpenCV functions, such as *cv::Dilate()*, can be synthesized into hardware. The *image_filter()* function is synthesized as hardware, which is described in verilog HDL. In order to use *hls::Dilate()* function, type conversion from *AXI_STREAM* to *hls::Mat* is necessary. *AXI_STREAM* represents a data stream on AXI4-Stream point-to-point bus protocol, while *hls::Mat* corresponds to *cv::Mat* in OpenCV.

Conversion of data between ROS message and hardware can be described in C/C++ for HLS. The ROS message format used in the system is *sensor_msgs/Image*, which is a popular data type in ROS message, as shown in Figure 6. The code fragment of the Publisher, which includes ROS communication procedure, conversion processing between ROS protocol and FPGA, and communication with application processing circuit is shown in Figure 7.



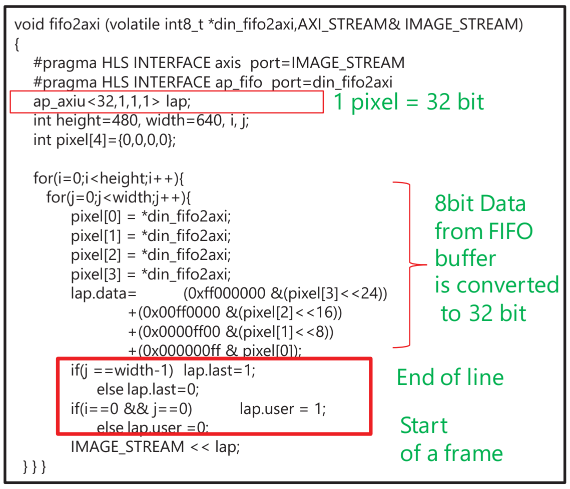
```
void image_filter (AXI_STREAM& INPUT_STREAM,
AXI_STREAM& OUTPUT_STREAM)
{
#pragma HLS INTERFACE axis port=OUTPUT_STREAM
#pragma HLS INTERFACE axis port=INPUT_STREAM
hls::Mat<MAX_HEIGHT, MAX_WIDTH, HLS_8UC3> img_0(rows, cols);
hls::Mat<MAX_HEIGHT, MAX_WIDTH, HLS_8UC3> img_1(rows, cols);

hls::AXIvideo2Mat(INPUT_STREAM, img_0);
hls::Dilate(img_0, img_1);
hls::Mat2AXIvideo(img_1, OUTPUT_STREAM); }
```

Figure 5 Code fragment for Dilate image filter

```
ROS message
(sensor_msg/Image)
std_msgs/Header header
uint32    height
uint32    width
string    encoding
uint8     is_bigendian
uint32    step
uint8[ ]  data
```

Figure 6 ROS message used in the evaluation system



```
void fifo2axi (volatile int8_t *din_fifo2axi,AXI_STREAM& IMAGE_STREAM)
{
  #pragma HLS INTERFACE axis  port=IMAGE_STREAM
  #pragma HLS INTERFACE ap_fifo  port=din_fifo2axi
  ap_axiu<32,1,1,1> lap;
  int height=480, width=640, i, j;
  int pixel[4]={0,0,0,0};

  for(i=0;i<height;i++){
    for(j=0;j<width;j++){
        pixel[0] = *din_fifo2axi;
        pixel[1] = *din_fifo2axi;
        pixel[2] = *din_fifo2axi;
        pixel[3] = *din_fifo2axi;
        lap.data=         (0xff000000 &(pixel[3]<<24))
                    +(0x00ff0000 &(pixel[2]<<16))
                    +(0x0000ff00 &(pixel[1]<<8))
                    +(0x000000ff & pixel[0]);
        if(j ==width-1)  lap.last=1;
            else lap.last=0;
        if(i==0 && j==0)          lap.user = 1;
            else lap.user =0;
        IMAGE_STREAM << lap;
 } } }
```

Figure 7 Code fragment for interpreting ROS message

Table 1 Source code lines of FPGA Publisher/Subscriber implementations by HDL and HLS (C/C++)

	Publisher	Subscriber	Total
HDL (Verilog)	415	445	860
HLS (C/C++)	57	70	127

The *fifo2axi()* function is synthesized as hardware, too. In the evaluation system, as TCP/IP packet arrived to the network interface port, the contents of the packet is pushed to a FIFO buffer by hardwired TCP/IP stack. The data of FIFO can be pop-up by reading the pointer variable *din_fifo2axi*. In the code fragment, four pixel values are pop-up from FIFO which has 8-bit width and marshalled into 32-bit width data. The control behavior such as end-of line and start-of frame can be described in C/C++ for HLS. The description is much simpler than writing RTL description in HDL.

The source code lines of the developed FPGA Publisher/Subscriber implementations are shown in Table 1. The lines are counted without comments. which was 860 lines (415 lines for publisher and 445 lines for subscriber), is described in 127 lines (57 lines for publisher and 70 lines for subscriber) in C/C++ language by using HLS. This reduction of description is achieved because C/C++ language can describe sequential behavior by line-by-line execution, while HDL need to clock-cycle level description to realize sequential description.

V. Discussion

Our proposal in this paper is communicating between software and FPGA through more loosely coupled way than the above mentioned tightly coupled, i.e. memory mapped access, coupling. There is a tradeoff between communication performance and usability, which includes the development, management and reuse cost. Communication between software and FPGA was an issue discussed for long time. The problem of tightly-coupled HW/SW is very long compilation time and necessity of expertise of FPGA to use FPGA. Our proposal is an encapsulation of FPGA as ROS node and make its interface sparse at network level. For many application, the latency of network level connection is hardly problematic [9].

ROS2 [10], a newer version of ROS, employs DDS (Data Distribution Service) [11] as a communication middleware. In DDS system, the Master process is distributed into the participant nodes. And the query transactions and data communication are done in RTPS (Real-time Publish-Subscribe) protocol [14] which uses broadcast/multicast using UDP/IP instead of TCP/IP for realizing QoS (Quality of Service) at better communication performance and reducing processing overhead for low-power embedded processors. There are many differential points between DDS and ROS, however, the principle of the communication sequence is similar. Therefore, it is expected to realize the same mechanism in ROS2 by developing the query process, too.

VI. Conclusion

A design flow of High Level Synthesis (HLS) of ROS protocol interpretation and communication circuit for FPGA is proposed. As a result of the feasibility study and evaluation, the conversion processing between ROS protocol and FPGA and the communication processing with other ROS nodes is described in 127 line in C/C++ language by using HLS, while it was 860 line in the description with the conventional hardware description language (HDL). This reduction of description is achieved because C/C++ language can describe sequential behavior by line-by-line execution, while HDL need to clock-cycle level description to realize sequential description. The proposed design flow is expected to improve program comprehension and contribute design productivity of ROS-compliant FPGA component for robot system.

Acknowledgment

This research and development work was supported by the MIC/SCOPE #152103014 and JSPS Kakenhi grant (17K00072). The authors also thank to Xilinx University Program and Intel University Program.

References

[1] Alexandre Cornu, Steven Derrien, Dominique Lavenier. “HLS Tools for FPGA : faster development with better performances.” Proceeding of the 7th International Symposium on Applied Reconfigurable Computing, Feb 2011, Belfast, United Kingdom. 6578, pp.67-78, 2011.

[2] M. A. Oezkan, O. Reiche, F. Hannig and J. Teich, "A Highly Efficient and Comprehensive Image Processing Library for C++-based High-Level Synthesis," FSP 2017; Fourth International Workshop on FPGAs for Software Programmers, Ghent, Belgium, 2017, pp. 1-10.

[3] J. Choi, Ruo Long Lian, S. Brown and J. Anderson, "A unified software approach to specify pipeline and spatial parallelism in FPGA hardware," 2016 IEEE 27th International Conference on Application-specific Systems, Architectures and Processors (ASAP), London, 2016, pp. 75-82. doi: 10.1109/ASAP.2016.7760775

[4] Takeshi Ohkawa, Kazushi Yamashina, Hitomi Kimura, Kanemitsu Ootsu, Takashi Yokota, "FPGA Component Technology for Easy Integration of FPGA into Robot Systems," IEICE Transactions on Information and Systems, Vol.E101-D, No.2, pp.363-375, Feb. 2018.

[5] H. Omidian and G. Lemieux. “An Accelerated OpenVX Overlay for Pure Software Programmers,” 2018 Field Programable Technology (FPT’18), Dec. 2018, pp. 293-296, DOI 10.1109/FPT.2018.00056

[6] H. Lam and D. Ojika, "Research Opportunities in Heterogeneous Computing for Machine Learning," 2018 International Conference on High Performance Computing & Simulation (HPCS), Orleans, 2018, pp. 559-560. doi: 10.1109/HPCS.2018.00094

[7] Jialiang Zhang and Jing Li. “Improving the Performance of OpenCL-based FPGA Accelerator for Convolutional Neural Network.” In Proceedings of the 2017 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays (FPGA '17). ACM, New York, NY, USA, 2017, pp. 25-34. DOI: https://doi.org/10.1145/3020078.3021698

[8] F. Winterstein, S. Bayliss and G. A. Constantinides, "High-level synthesis of dynamic data structures: A case study using Vivado HLS," 2013 International Conference on Field-Programmable Technology (FPT), Kyoto, 2013, pp. 362-365. doi: 10.1109/FPT.2013.6718388

[9] Yuhei Sugata, Takeshi Ohkawa, Kanemitsu Ootsu, and Takashi Yokota. 2017. Acceleration of Publish/Subscribe Messaging in ROS-compliant FPGA Component. In Proceedings of the 8th International Symposium on Highly Efficient Accelerators and Reconfigurable Technologies (HEART2017). ACM, New York, NY, USA, Article 13, 6 pages. DOI: https://doi.org/10.1145/3120895.3120904

[10] Yuya Maruyama, Shinpei Kato, and Takuya Azumi.“Exploring the performance of ROS2.” In Proceedings of the 13th International Conference on Embedded Software (EMSOFT '16). ACM, 10 pages. 2016, DOI: https://doi.org/10.1145/2968478.2968502

[11] G. Pardo-Castellote, "OMG Data-Distribution Service: architectural overview," 23rd International Conference on Distributed Computing Systems Workshops, 2003. Proceedings., 2003, pp. 200-206. doi: 10.1109/ICDCSW.2003.1203555

[12] Takeshi Ohkawa, Kazushi Yamashina, Takuya Matsumoto, Kanemitsu Ootsu, Takashi Yokota, “Automatic Generation Tool of FPGA Components for Robots,”, IEICE transactions on Information and Systems, to appear in the issue of Jun. 2019

[13] http://wiki.ros.org/xmlrpcpp

[14] https://www.omg.org/spec/DDSI-RTPS/

[15] http://wiki.ros.org/