Acceleration of Publish/Subscribe Messaging in ROS-compliant FPGA Component

Yuhei Sugata†, Takeshi Ohkawa‡, Kanemitsu Ootsu‡, Takashi Yokota‡

Graduate School of Engineering, Utsunomiya University
7-1-2 Yoto, Utsunomiya, Tochigi, 321-8585, Japan

†sugata@virgo.is.utsunomiya-u.ac.jp
‡{ohkawa, kim, yokota}@is.utsunomiya-u.ac.jp

ABSTRACT

Intelligent robots demand complex information processing such as SLAM (Simultaneous Localization and Mapping) and DNN (Deep Neural Network). FPGA (Field Programmable Gate Array) is expected to accelerate these applications with high energy efficiency. Introducing FPGA into robots is difficult due to its high development costs. In order to introduce FPGA easily to a system on ROS (Robot Operating System) development platform, ROS-compliant FPGA Component has been proposed. However, large communication latency between ROS components is a severe problem. This research aims to reduce the latency by implementing Publish/Subscribe messaging of ROS as hardware. Based on result of network packets analysis in ROS system, we propose a method of implementing a Hardware ROS-compliant FPGA Component. It is done by separating registration part (XMLRPC) and data communication part (TCPROS) of Publish/Subscribe messaging. Evaluation result shows that the proposed Hardware ROS-compliant FPGA component can reduce delay time of Publish/Subscribe messaging even with small amount of hardware.

1. INTRODUCTION

Autonomous robots with high intelligence are expected to take an active role in disaster recovery under ultimate condition or in daily life support with human interaction [1]. Such robots are required to recognize their surroundings by using compute-intensive algorithms, for example, SLAM (Simultaneous Localization and Mapping) [2][3], object recognition, DNN (Deep Neural Networks) [4] and so on. To process these algorithms in real-time conditions, it is necessary for robots to compute and process large amount of data. However, high-performance processors are hard to be employed into robots due to its battery operation.

FPGA (Field Programmable Gate Array) exhibits high power-efficiency compared to high-performance processors, i.e. CPU or GPU (Graphics Processing Unit) [5][6]. Optimized parallel processing of hardware can be implemented by using FPGA. Acceleration of application, such as preprocessing of SLAM [3] and hardwired neural network [4], has been researched actively.

On the other hand, development costs of FPGA are the major problems because FPGA development basically requires RTL (Register Transfer Level) circuit description in HDL (Hardware Description Language). To ease the difficulty in RTL development, HLS (High-Level Synthesis) [7] design using C language has been proposed, however, FPGA developer still needs deep knowledge of both hardware and software. Therefore, FPGA is difficult for robot developers to use since they are busy with designing mechanics, electronics and software.

In order to solve the above-mentioned problem, ROS-compliant FPGA component was proposed [8]. ROS (Robot Operating System) [9][10][20] is a software platform for robot development. It provides communication environment among components, build tools for component development, debugging/logging tools and so on. In robot development, component oriented development is essential [11][17]. ROS-compliant FPGA component is a componentized FPGA, which complies with ROS communication protocol, i.e. Publish/Subscribe messaging. Any componentized FPGA can be introduced into ROS system easily [18] since it is functionally equivalent to software ROS component. However, the ROS-compliant FPGA component has a severe overhead in performance due to communication latency. The communication overhead of Publish/Subscribe messaging of ROS-compliant FPGA component, which is implemented on Programmable SoC (System-on-Chip) [12][22], is very large [8][19].

In this paper, we propose a method of implementing a hardware ROS-compliant FPGA Component without using Programmable SoC. Based on analysis of network packets traffic of ROS system, we design a structure of Hardware ROS-compliant FPGA Component and evaluate its performance.

2. ROS-compliant FPGA component

2.1 ROS Overview

ROS (Robot Operating System) is released by OSRF (Open Source Robotics Foundation) as an open source project [20]. It is a software platform which provides a framework of communication layer and builds system for robotic

software. Communication model of ROS is based on Publish/Subscribe messaging (Figure 1). Publish/Subscribe messaging is an asynchronous messaging model in which ROS nodes communicate through a topic with other nodes. In ROS development, a robot system is designed with a set of components called “node” and its communication channel called “topic”'. A publisher node publishes a message to a topic. And any subscriber node, which has subscribed to the topic in advance, can receive the message. The biggest advantage of Publish/Subscribe messaging is a dynamic network configuration. Since the ROS nodes are bound loosely, it is possible to add/remove a node to/from the system easily.

2.2 Performance issue of ROS

Figure 2 shows a structure overview of ROS-compliant FPGA component implemented on Programmable SoC, which is reported in [8]. To implement a component, a programmable SoC of Xilinx Zynq-7020 is used, which equips one ARM processor and FPGA. In the SoC, software ROS nodes which run on the ARM processor (interface for FPGA), carry out communication between HW-SW and that of other ROS nodes. The component receives a ROS message from a subscribed topic in advance, then processes data of the message on FPGA. After the data processing on FPGA, the result is published to a topic for other ROS nodes.

For performance evaluation, connected-component labeling was implemented as a HDL circuit and componentized as a ROS-compliant FPGA component on a Programmable SoC [8]. The result of Full-HD (1920x1080) image is shown in Figure 3. Processing of labeling (3) is drastically accelerated by using FPGA, and other parts take much long time. Especially, Publish/Subscribe messaging ((1) and (5)) occupies 85% (1.686 (s)) of the whole latency (1.998 (s)) of the component.

To study ROS performance, we have compared communication latency of (1) PC-PC and (2) PC-ARM. Figure 4 shows measured latency in two different data size (1M and 6M). Two computer nodes are connected each other through Gigabit Ethernet. It is a problem that the latency in (2) PC-ARM environment is much larger than (1) PC-PC. Therefore, the performance in embedded processor environment, such as ARM processor, should be improved. The next generation of ROS, ROS2[13] is proposed to improve performance in embedded systems and under review of specification currently. However, we now focus on the current version of ROS because there are many active users and abundant component library.

The challenge for ROS-compliant FPGA component is to reduce large overhead in communication latency. If communication latency is reduced, the ROS-compliant FPGA component can be used as an accelerator for processing in robotic applications/systems.

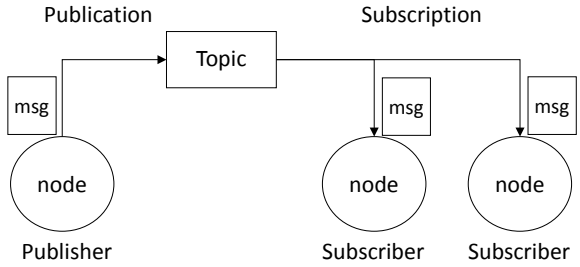

Figure 1 concept of Publish/Subscribe messaging in ROS

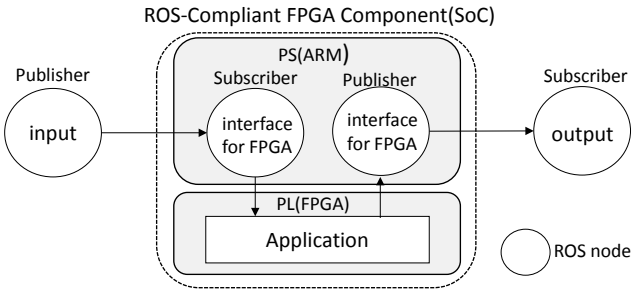

Figure 2 a structure overview of ROS-Compliant FPGA Component implemented on Programmable SoC

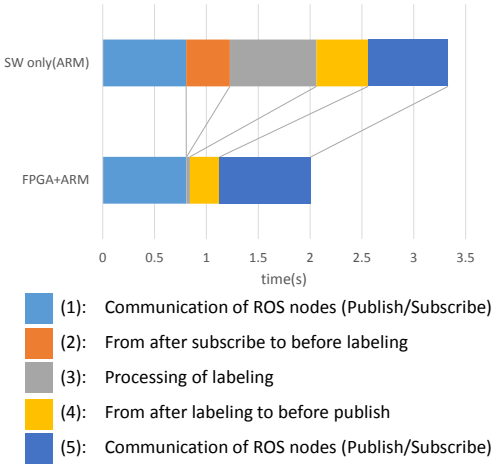

Figure 3 latency comparison of ROS-Compliant FPGA Component and software ROS

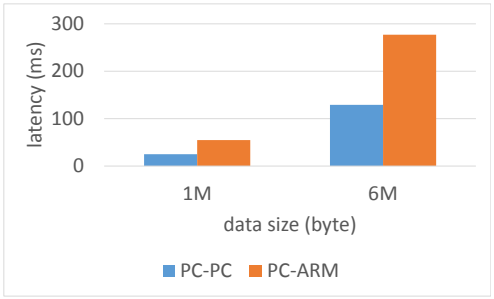

Figure 4 measured latency of ROS topic communication between remote nodes on (1) PC-PC and (2) PC-ARM

3. Hardware Design of ROS node

3.1 Analysis

In order to implement Publish/Subscribe messaging of ROS as hardware, we analyzed network packets which flowed in Publish/Subscribe messaging in ROS system of ordinary software. WireShark [21], a software for network packet analysis, was used.

Figure 5 shows a communication procedure in ROS system which consists of, as minimum structure, a master, a publisher and a subscriber node.

***STEP(1)*:** Publisher and Subscriber nodes register their nodes and topic information to Master. The registration is done by calling methods like *registerPublisher*, *hasParam* and so on, using XMLRPC [14].

***STEP(2)*:** Master notifies topic information to Subscriber nodes by calling *publisherUpdate* (XMLRPC).

***STEP(3)*:** Subscriber sends a connection request to Publisher by using *requestTopic* (XMLRPC).

***STEP(4)*:** Publisher returns IP address and port number, TCP connection information for data communication, as a response to the *requestTopic* (XMLRPC).

***STEP(5)*:** Subscriber establishes a TCP connection by using the information and sends connection header to the TCP connection. Connection header contains important metadata about a connection being established, including typing and routing information, using TCPROS [20].

***STEP (6)*:** If it is a successful connection, Publisher sends connection header (TCPROS).

***STEP (7)*:** Data transmission repeats. This data is written with little endian and header information (4 byte) is added to the data (TCPROS).

After this analysis we found out that network packets which flowed in Publish/Subscribe messaging in ROS system can be categorized into two parts, that is, registration part and data transmission part. Registration part uses XMLRPC [14] ((1) to (4)), while data transmission part uses TCPROS [20] ((5) to (7)), which is almost raw data of TCP communication with very small overhead. In addition, once data transmission (7) starts, only data transmission repeats without steps (1) – (6).

TCP/IP communication capability, which is necessary for implementing publisher and subscriber, is summarized in Table 1. Based on the network packet analysis, we the server ports used in XMLRPC and TCPROS are assigned differently. In addition, a client TCP/IP connection of XMLRPC for master node is necessary for publisher. For subscriber, two client TCP/IP connections of XMLRPC and one client connection of TCPROS are necessary. Therefore, two or three TCP ports are necessary to implement Publish/Subscribe messaging. It is a problem to implement ROS node using hardware TCP/IP stack. This issue is discussed in the following subsection.

Table 1 TCP/IP communication capability necessary for implementing a ROS node based on network packet analysis

	Publisher		*Subscriber*	
Server port for XMLRPC	1 port	(3)	1 port	(2)
Client port for XMLRPC	1 port	(1)	2 ports	(1), (3)
Server port for TCPROS	1port	(5, 6, 7)	-	
Client port for TCPROS	-		1 port	(5, 6, 7)

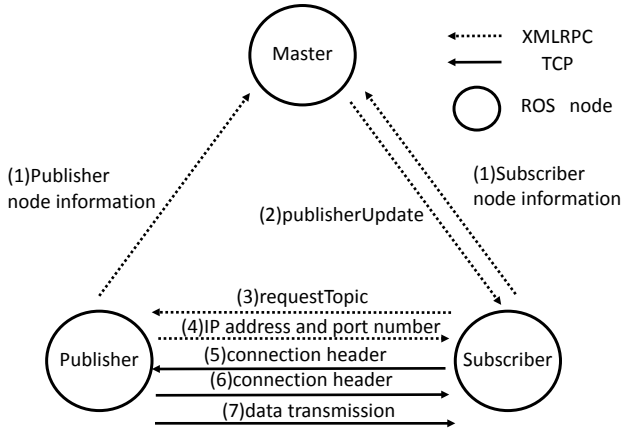

Figure 5 procedure of Publish/Subscribe messaging

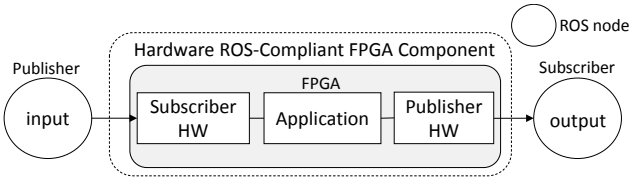

Figure 6 structure overview of Hardware ROS-compliant FPGA Component

3.2 Design

Conventionally, publication or subscription of topics was done by software (interface for FPGA node in Figure 2). By implementing this node as a hardwired circuit, direct communication between the ROS node and the FPGA becomes possible, and it is expected to reduce the communication latency. In order to implement hardware ROS nodes, we designed Subscriber HW and Publisher HW separately as shown in Figure 6. Subscriber HW is responsible to subscribe to a topic of another ROS node and to receive ROS messages from the topic; Publisher HW is responsible to publish ROS messages to a topic of another ROS node. Several hardware TCP/IP stacks for performing TCP/IP communication with FPGA, have been proposed [15][16]. It should be noted that the functionality of hardware TCP/IP stack is restricted because hardware resource of FPGA is generally limited. More resource is needed if more functionality (the number of ports, sessions, and corresponding protocols) is necessary.

In this paper, we studied an implementation method of ROS nodes as hardware on an FPGA, using a simple TCP/IP stack which has only one port and one session communication capability. In order to accelerate data communication part, only TCPROS is implemented not in software but in hardware. TCP connection of data transmission part ((5) to (7)) is established based on IP address and port number of connection response (4). So, it is possible to separate registration part and data transmission part by changing IP address and port number. In other words, XMLRPC part is done by software, and TCPROS part is done by hardware.

Functionalities for separating XMLRPC part and TCPROS part are summarized below, and execution sequence of Subscriber HW and SW is shown in Figure 7 and execution sequence of Publisher HW and SW is shown in Figure 8,.

- Subscriber SW processes ***STEP(1)*** to ***(4)*** and sends IP address and port number to Subscriber HW.
- Subscriber HW establishes TCP connection to a publisher based on IP address and port number given by Subscriber SW, and receives ROS message after sending connection header.
- Publisher SW processes ***STEP(1)*** to ***(4)*** and returns IP address and port number of Publisher HW at ***STEP(4)***.
- Publisher HW sends connection header and ROS message after receiving the header from subscriber.

By separating the registration part (XMLRPC) and the data communication part (TCPROS) as described above, it is possible to implement Publish/Subscribe messaging as hardware with hardware TCP/IP stack which has only one port and one session.

3.3 Hardware Implementation

A block diagram of Hardware ROS-Compliant FPGA Component is shown in Figure 9. SiTCP [16] is used for hardware TCP/IP stack, which is originally designed for large sensor networks in high-energy physical experiments. By using SiTCP, TCP/IP communication on Gigabit Ethernet can be performed with a small amount of hardware. SiTCP has one TCP port and one UDP port for remote control of the status of the SiTCP itself. SiTCP works as a server mode, at default, which listens to a connection request. It can also work as a client mode, after configuration, which requests a connection for a server.

Subscriber HW works as a client mode. The sequence shown in Figure 7 is executed by *Subscriber Logic*. UDP control packets are used to store destination IP address and port number in RBCP_REG. When *Subscriber HW* requests TCP connection, the stored values are used. After TCP connection is established, Subscriber Logic writes connection header to FIFO and sends TCP packets. The response header and data from topic are sent from the destination as reply packets.

Application Logic processes data given by Subscriber HW, and sends result data to Publisher HW.

Publisher HW works as a server mode. The sequence shown in Figure 8 is executed by *Publisher Logic*. SiTCP listens to a connection request from subscriber. After TCP connection is established, Publisher Logic reads connection header from FIFO, prepares replying connection header and sends it as a TCP packet. After the connection, data for topic are sent.

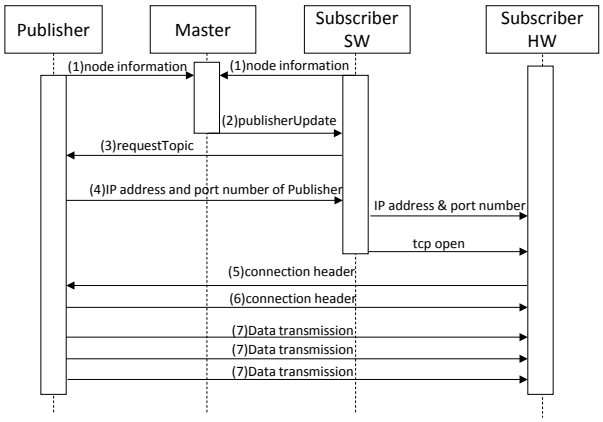

Figure 7 execution sequence of Subscriber SW and HW

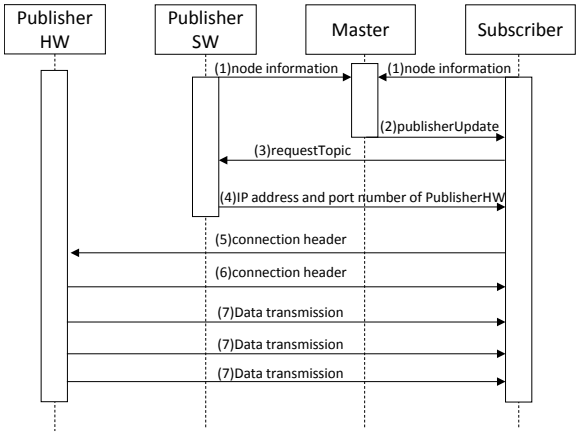

Figure 8 execution sequence of Publisher SW and HW

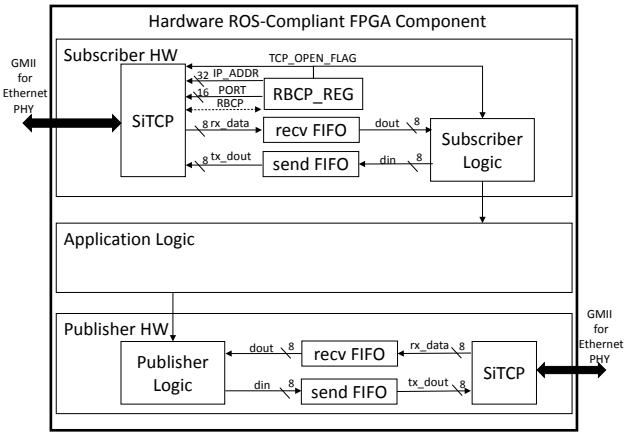

Figure 9 Hardware ROS-Compliant FPGA Component

4. Evaluation

4.1 Evaluation system

In order to evaluate the proposed hardware implementation method, we have developed a ROS system shown in Figure 10. ROS message type used in this evaluation has two int32 values. Each node publishes and/or subscribes a message of the type as follows.

- *Para_in* node publishes a message to *input_data* topic.
- *Add_para* node receives the message from the *input_data* topic, and publishes a message based on the received message to *output_data* topic.
- *Add* node receives the message from the *output_data* topic.

4.2 Measurement environment

Measurements were done for the latency from publishing the *para_in* node until receiving the message of the add node. We prepared three types of evaluation environments shown in Table 2. Each nodes are connected with Gigabit Ethernet. Host PC used in environment (1) to (3) is equipped with Intel Core i7 870(2.93GHz), 4GB memory and its OS is Ubuntu16.04.

- Environment (1), a ROS node is executed on a PC equipped with Intel Core i7 920(2.67GHz), 12GB memory and its OS is Ubuntu16.04.
- Environment (2), a ROS node is executed on an ARM Cortex-A9 processor (666MHz) of ZC7Z020 (Xilinx Ltd) on Zedboard [23].
- Environment (3), a HW ROS node is implemented by using Spartan-6 FPGA (XC6SLX110, Xilinx Inc.) and two SiTCP instances for Gigabit Ethernet ports on exTri-CSI board (e-trees.Japan, Inc.).

Table 3 shows the amount of hardware resource of Hardware ROS-compliant FPGA component. It can be implemented with small amount of hardware. Please note that Application Logic is not implemented in this evaluation. The rest of FPGA resource can be used for Application Logic.

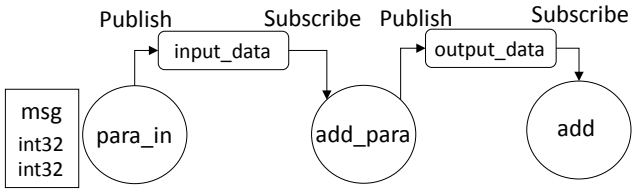

Figure 10 nodes and topics of the ROS system used for measurement of communication latency

Table 2 deployment of ROS nodes

environment	*para in*	*add para*	*add*
(1)PC	host PC	PC	host PC
(2)ARM	host PC	ARM	host PC
(3)FPGA	host PC	FPGA	host PC

Table 3 resource used for implementing Hardware ROS-compliant FPGA component (Xilinx Spartan-6, XC6SLX110)

RESOURCE	*UTILIZATION*
Slice Registers	9253/126576(7%)
Slice LUTs	7421/63288(11%)
RAMB16BWERs	54/268(20%)

4.3 Evaluation results

Figure 11 shows the average of latency in each evaluation environment. The measurement was done by using gettimeofday() standard C library function in the software and repeated 10 times. In the environment (1), latency of Publish/Subscribe messaging between PC and PC was 0.9ms. In the environment (2), latency between PC and ARM processer was 1.0ms. On the other hand, in the environment (3), latency between PC and FPGA was 4.0ms. The latency (3) increased even though it used hardware TCP/IP stack (SiTCP) on FPGA. The reason is that SiTCP does not send packet immediately. When transmission data does not reach the maximum segment size (MSS, usually 1460byte), SiTCP waits for input of additional transmission data. To ensure this, latency was measured by configuring the MSS of SiTCP to 12byte, which is the same size as transmission data (12byte). Dummy data was added on in order to send a packet immediately. The measured latency is shown as (4) in Figure 11. The latency between the PC and FPGA can be 0.5ms by configuring the Publisher HW to write the transmission data and transmit it immediately. Therefore, it is clear that the latency of hardware ROS node can be reduced compared to software ROS node.

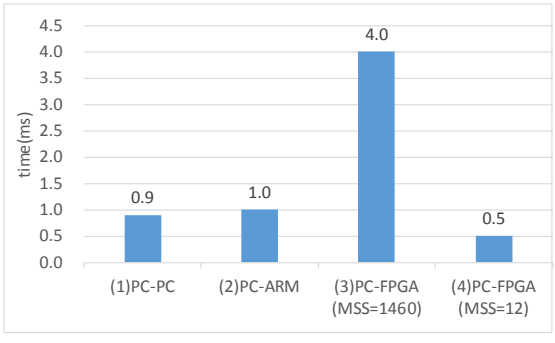

Figure 11 measured latency of ROS node (1)PC-PC (2)PC-ARM (3)PC-FPGA (MSS=1460) (4) PC-FPGA (MSS=12)

5. Conclusion

In this paper, we have discussed hardware design of Publish/Subscribe communication in ROS node by using FPGA. Based on the network packet analysis results, we have proposed to separate ROS Publish/Subscribe communication structure into registration part (XMLRPC) and data communication part (TCPROS). By employing the proposed separation method, Hardware ROS-compliant FPGA component can be implemented, even with a TCP/IP hardware stack (SiTCP) which only requires small amount of hardware resource and has single TCP/IP port and single session. Then the communication latency was evaluated using the Hardware ROS-compliant FPGA component. The result indicated that latency of the Hardware ROS-compliant FPGA component can be reduced to 0.5ms by the proposed implementation method, while the latency of an ordinary software ROS component is about 1.0ms. Forthcoming challenges are to process high-resolution image data and to integrate registration part (XMLRPC) and data communication part (TCPROS) into single FPGA.

6. ACKNOWLEDGMENTS

This research and development work was supported by the MIC/SCOPE#152103014.

7. REFERENCES

[1] Siegwart, R., Nourbakhsh, I. R., and Scaramuzza, D., "Introduction to autonomous mobile robots" MIT press, 2011.

[2] Dissanayake, M. W. M. G., et al. "A solution to the simultaneous localization and map building (SLAM) problem." Robotics and Automation, IEEE Transactions on 17.3, pp. 229-241, 2001.

[3] Janosch Nikolic, et al., "A synchronized visual-inertial sensor system with FPGA pre-processing for accurate real-time SLAM," 2014 IEEE International Conference on Robotics and Automation (ICRA) 2014, pp. 431-437, 2014.

[4] Zhang Chen, Li Peng, Sun Guangyu, Xiao, Yijin Guan, Bingjun Xiao, Jason Cong, "Optimizing fpga-based accelerator design for deep convolutional neural networks." In Proc. of the 2015 ACM/SIGDA International Symposium on Field- Programmable Gate Arrays, pp.161-170, 2015.

[5] Kentaro Sano, Wang Luzhou, Yoshiaki Hatsuda, Takanori Iizuka and Satoru Yamamoto, "FPGA-Array with Bandwidth-Reduction Mechanism for Scalable and Power-Efficient Numerical Simulations based on Finite Difference Methods," ACM Transactions on Reconfigurable Technology and Systems (TRETS), Vol.3, No.4, Article No.21, DOI:10.1145/1862648.1862651 (35 pages), 2010.

[6] Asano, S., Maruyama, T., and Yamaguchi, Y., "Performance comparison of FPGA, GPU and CPU in image processing", In Proc. of International Conference on Field Programmable Logic and Applications, FPL2009, pp. 126-131, 2009.

[7] P. Coussy, et al. "An introduction to high-level synthesis," IEEE Design & Test of Computers 26.4, pp. 8-17, 2009.

[8] 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.

[9] Quigley, M., Conley, K., Gerkey, B., Faust, J., Foote, T., Leibs, J., and Ng, A. Y., "ROS: an open-source Robot Operating System," In ICRA workshop on open source software, Vol. 3, No. 3.2, p. 5, 2009.

[10] Cousins, S., "Exponential growth of ros [ros topics]," IEEE Robotics and Automation Magazine, 1(18), pp.19-20, 2011.

[11] Mamat R., Jawawi, A., Dayang, N., and Deris, S., "A component oriented programming for embedded mobile robot software", International Journal of Advanced Robotic Systems, pp. 371-380, 2007.

[12] Lin, Z., and Chow, P., "Zcluster: A zynq-based hadoop cluster," In Proc. of International Conference on Field-Programmable Technology (FPT) 2013, pp.450-453, 2013.

[13] Yuya Maruyama, Shinpei Kato, and Takuya Azumi, "Exploring the Performance of ROS2," In Proc. of the ACM SIGBED International Conference on Embedded Software (EMSOFT), 2016.

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

[15] David Sidle, Zsolt Istvan, Gustavo Alonso, "Low-Latency TCP/IP Stack for Data Center Applications," In Proc.26th International Conference on Field Programmable Logic and Applications (FPL), pp. 1-4, 2016.

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

[17] Takeshi Ohkawa, Daichi Uetake, Takashi Yokota, Kanemitsu Ootsu and Takanobu Baba, "Reconfigurable and Hardwired ORB Engine on FPGA by Java-to-HDL Synthesizer for Realtime Application," Proc. 4th International Symposium on Highly Efficient Accelerators and Reconfigurable Technologies (HEART 2013), pp. 45-50, June 2013.

[18] 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.

[19] Takeshi Ohkawa, Kazushi Yamashina, Takuya Matsumoto, Kanemitsu Ootsu, Takashi Yokota, "Architecture Exploration of Robot System using ROS-Compliant FPGA Component," Proc. 27th IEEE International Symposium on Rapid System Prototyping (RSP), pp.72-78, Oct. 2016. DOI: http://dx.doi.org/10.1145/2990299.2990312

[20] Open Source Robotics Foundation : http://wiki.ros.org/ (Access: 2017/3/15)

[21] WireShark: https://www.wireshark.org/ (Access: 2017/3/15)

[22] Zynq-7000 All Programmable SoC, Xilinx, http://www.xilinx.com/products/silicon-devices/soc/zynq-7000.html, (Access: 2017/3/15)

[23] http://zedboard.org/