2019 International Workshop on Smart Info-Media Systems in Asia (SISA 2019),Sep.4-6,2019

Component-based FPGA Development for Intelligent Robotics

Takeshi Ohkawa*

*Tokai University, Japan

Abstract— **Intelligent robots need versatile kinds of processing such as image processing, probabilistic matching, neural network and so on. Processing which is enough standardized can be done on a hardwired LSI chip, however, advanced features need reconfigurability of algorithm by using FPGA chip. However, the development cost of FPGA is still high even with using the HLS (High-Level Synthesis). This paper proposes a novel approach to improve the reusability of FPGA IPs by introducing a component-based design method by inter-FPGA component and intra-FPGA component.[1]**

I. Introduction

Versatile kinds of tasks are required to construct an intelligent robot. Also, Various hardware accelerators are used to improve the 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 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 the 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 kinds of processing which need acceleration, it is not always possible to prepare dedicated hardware accelerator.

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.

The difficulty of using FPGA in robots is that designing a high-performance circuit is time-consuming and needs hardware-expertise since it is done at the 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++. 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. I proposed ROS-compliant FPGA component [4] in order to introduce any FPGA processing circuit easily to robot systems. ROS (Robot Operating System) 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.

The above FPGA component treats FPGA communication with the outside the FPGA device. Another aspect of FPGA development is the component-based development of inside FPGA device. MATLAB/Simulink or LabView are the widely-used tools of model-based and component-based development platform for FPGA. They are well optimized for their target applications since they are commercial products. As a general solution, a component-based design method by using FPGA circuit modules based-on Publish-Subscribe communication is also proposed [7], inspired by the ROS-compliant FPGA component.

This paper proposes a novel method to develop intelligent robots using the previously proposed FPGA component technologies. The contributions of this paper are:

- Proposal of a novel idea of inter/intra-FPGA component based on Publish/Subscribe architecture
- Proposal of a novel design approach of an intelligent robot using the intra/inter-FPGA component

[1] 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.

II. Component-based FPGA Development

Component-based development is a widely-accepted development method to improve the design productivity of software system. On the other hand, development environment for FPGA is not so useful compared to one for software. In this section, several aspects of component-based development of FPGA for intelligent robotics are discussed.

The idea of component-based development can be widely applied. Therefore, one can imaging many things for the word "component". Here I categorize the methods of component-based FPGA development into two:

- (A) Inter-FPGA component
- (B) Intra-FPGA component

The inter-FPGA component is for inter-FPGA development with inter-communication among FPGAs or other software. On the other hand, intra-FPGA component is for intra-FPGA development inside FPGA. Ideally, the two should be integrated into one, however, performance requirement does not allow inter-FPGA communication sometimes. That is, the communication overhead between component is the issue for this kind of component technology. Therefore, the discussion in this paper categorize the component technology into two.

A. Inter-FPGA component for inter-FPGA development

In a system which has multiple FPGAs, or a system which has an FPGA device and other microprocessor, there is a need of communication from/to FPGA. In such cases, the idea of component-based development is effective to improve the design reusability of FPGA.

For example, FPGA which can communicate through TCP/IP can be an FPGA component, which can be operated from outside FPGA. However, TCP/IP is an infrastructural communication protocol below application layer. It means the semantics and the interpretation of the message totally depends on the target application.

To realize a generic FPGA-component which can be operated by software, several studies have been done. A hardware/software (hw/sw) complex system [11] that combines hw (an FPGA) and sw (a CPU). This is an idea to have software interface for hardware (FPGA) and other software can operate FPGA as a software. ORB Engine [11] is an approach to make FPGA a CORBA object. CORBA [17] (Common Object Request Broker Architecture) is a standard software technology by which a method of remote object can be called through network message, in the manger of object-oriented programming (OOP). A succeeding specification which is independent from OOP is the DDS [18] (data distribution service). DDS focuses on data flow and the communication QoS (Quality of Service) rather than object method call.

Another movement is application domain specific approach of realizing FPGA a component. ROS (Robot Operating System) is a kind of software platform which supports component-oriented development. ROS-compliant FPGA component [4] is proposed to easily introduce FPGA into robots. An automated design tool was also proposed [5] 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, a hardwired ROS protocol interpretation and communication circuit for high-performance data communication is proposed [6]. COMTA (Connective Object for Middleware to Accelerator) [13] provides communication channel as Data object and SHM (Shared memory) object to FPGA by using ARM processor on SoC-FPGA, such as Xilinx Zynq or Intel SoC-FPGA device.

The above technologies are to communicate FPGA with software outside the FPGA device.

B. Intra-FPGA component for intra-FPGA development

Even inside an FPGA device, there are multiple circuit modules which have various functionality. The benefits of component-based development are also applicable to the intra-FPGA development.

MATLAB/Simulink is the widely-used tool of model-based and component-based development platform for FPGA. LabView is also a well-known tool by NI. Since the tools are commercial, the tools are very optimized to their products.

A generic approach of model-based and component-based development is inspired by the ROS-compliant FPGA component technology. A design method of realizing Publish/Subscribe communication on the FPGA is proposed [7]. The effectiveness of Publish/Subscribe communication model is widely recognized as the model to express data flow of application with the Internet or intelligent robots, namely in ROS. In other words, the model that Subscribers receive only the necessary topics among the information sent by Publishers represents the natural structure of information processing in the era of big data. For example, ROS (Robot Operating System) improves scalability, expandability and reusability of robot software parts by designing and implementing communication among a lot of software processes with the Publish/Sub-scribe communication model.

III. Inter-FPGA Component technology

This section describes the previously proposed ROS-compliant FPGA component, which can be categorized into the inter-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. Any HLS tools can be a generic solution for computing tasks like scientific computations, and complex applications including dynamic data structure.

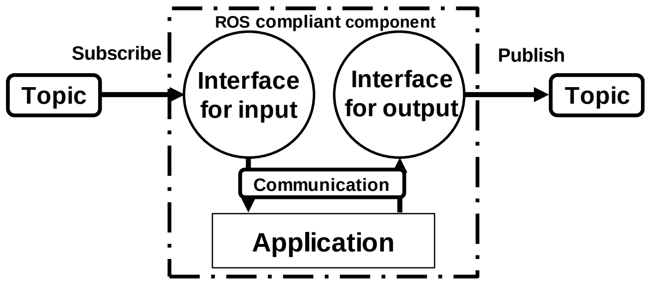


Figure 1 Structure of ROS-compliant FPGA component

Here, two interfaces are described, 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 another 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 [19]. After the query transactions, message communication of application data starts in TCPROS protocol. In our previous report [6], 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. 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 simplest 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.

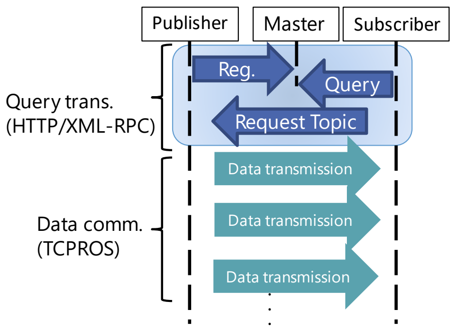


Figure 2 Sequence of ROS message communication

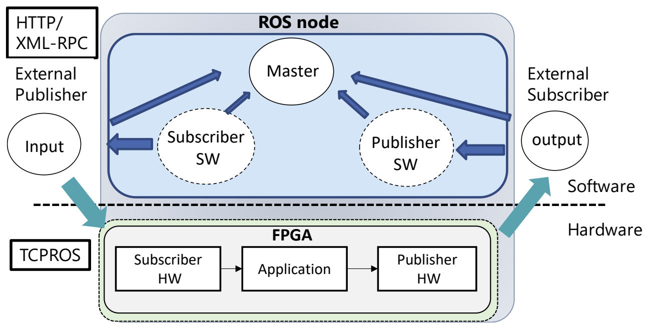


Figure 3 Co-operation of SW/HW to realize ROS node by separating XML-RPC and TCPROS protocol

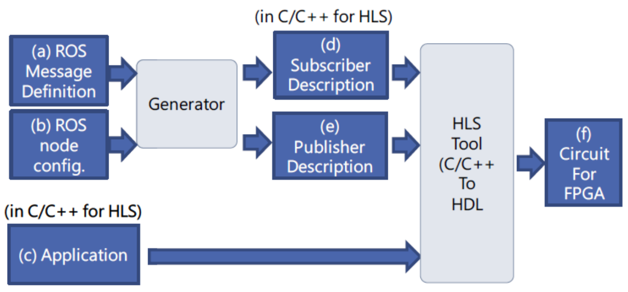


Figure 4 Design flow of ROS-compliant publish/subscribe FPGA node

A design flow of ROS-compliant FPGA component with HLS is described 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 the publisher/subscriber node information such as topic name and node name. 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.

ROS2 [14], a newer version of ROS, employs DDS (Data Distribution Service) [18] 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 [20] 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.

In summary, ROS-compliant FPGA component is an example of inter-FPGA component, which can be an accelerator of various kind of processing. The communication overhead can be minimized by hardware communication (TCP/IP). In ROS2, the hardware communication would be simpler because it uses UDP/IP.

IV. Intra-FPGA Component Technology

This section describes the previously proposed FPGA development framework [7][8] by integrating circuit modules through Publish/Subscribe communication, which can be categorized into the intra-FPGA component.

ROS is a successful example of software platform which employs Publish/Subscribe model. The benefit of Publish/Subscribe model is sparse connection between nodes, since “topic” plays the role of buffer and nodes connected to buffer can work more freely compared from a model which directly connects nodes. A system based on Publish/Subscribe communication model is constructed by multiple nodes. Nodes communicate each other via topics, which are logical queue. Each node communicates through a logic queue, which is called “topic” Data of a specific “topic” is published by “publisher”. And “subscriber” who would like to receive the data of the topic subscribes to the topic. After subscription, published data is distributed to subscribers. Adding new nodes and/or removing nodes are easy because multiple nodes publish or subscribe a topic.

The overview of the framework of intra-FPGA component is shown in Figure 5. This framework consists of FIFO buffer and switch in order to implement topic. In this framework, Publisher publishes data by writing data into input-FIFO buffer. On the other hand, Subscriber receives data by reading data from output-FIFO buffer. The switch in the framework reads data from input-FIFO buffer, which is mapped to a topic, and writes the data into all the output-FIFO buffer. Thus, each subscriber can receive data which was published by publishers.

As a design example using the above framework, the image resizing system was designed. Image resizing is used in the image recognition processing with local feature amounts. Figure 6 shows the Publish/Subscribe communication model of an image resizing system. A process “Master” publishes source images to processes “Worker”. Workers generate resized images of different scales from each other.

Figure 7 shows the image resizing system, designed using the proposed framework. This framework has 5 FIFO buffers for a Publisher and 8 FIFO buffers for Subscribers. The number written on the FIFO buffers in Figure 2 are the numbers of topics. Topic 1 is a channel for source images. Topic 2 to 5 are channels for resized images.

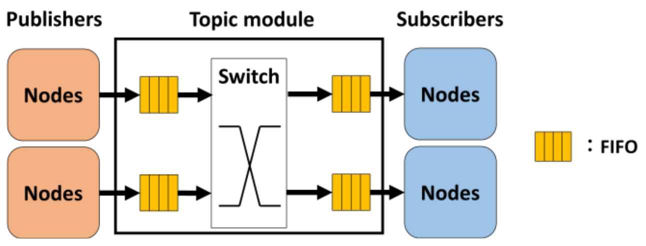


Figure 5 Concept of intra-FPGA component and communication via Topic module

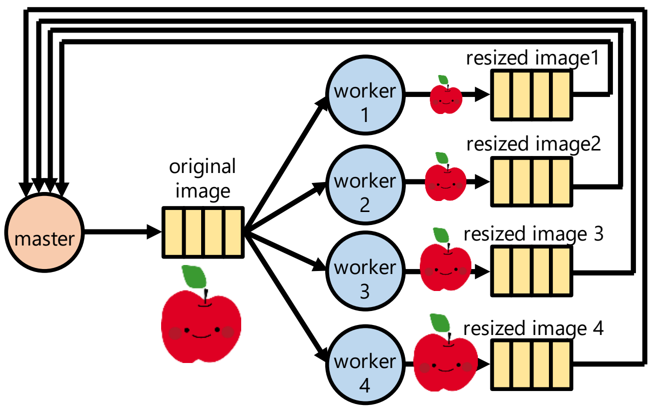


Figure 6 Example Publish/Subscribe model of multiple image resizing with the intra-FPGA component

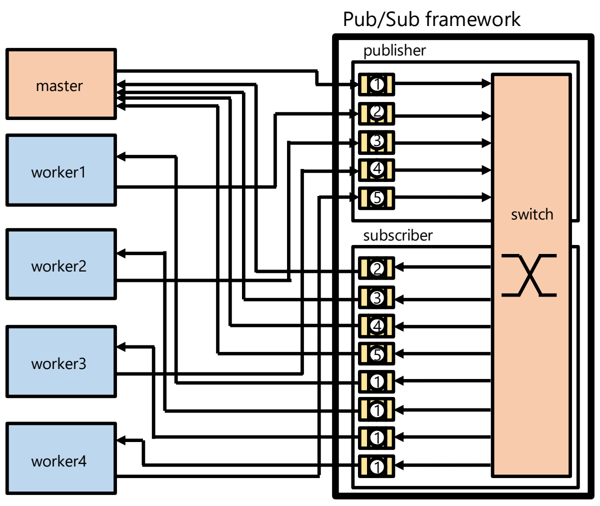
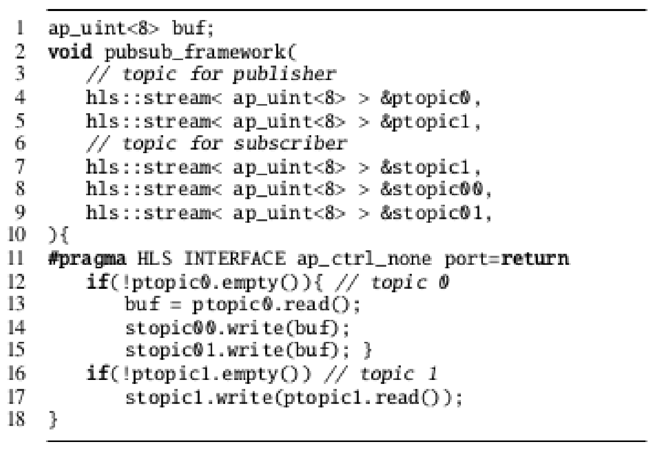


Figure 7 Example design with framework: block diagram



```
ap_uint<8> buf;
void pubsub_framework(
    // topic for publisher
    hls::stream< ap_uint<8> > &ptopic0,
    hls::stream< ap_uint<8> > &ptopic1,
    // topic for subscriber
    hls::stream< ap_uint<8> > &stopic1,
    hls::stream< ap_uint<8> > &stopic00,
    hls::stream< ap_uint<8> > &stopic01,
){
#pragma HLS INTERFACE ap_ctrl_none port=return
    if(!ptopic0.empty()){ // topic 0
        buf = ptopic0.read();
        stopic00.write(buf);
        stopic01.write(buf); }
    if(!ptopic1.empty()) // topic 1
        stopic1.write(ptopic1.read());
}
```

Figure 8 An example C++ code of the topic module for HLS

An evaluation was made on hardware resource utilization and clock cycles. In this evaluation, the proposed framework is implemented in XC7K325T-2FFG900C, is Xilinx FPGA, on the Genesys 2 board (Digilent inc.). The switch was implemented with C++ language using a development environment Vivado HLS (Xilinx inc.). Figure 8 shows the implementation of the topic module. In this sample code, two publishers and three subscribers are connected as written at the arguments. Using type hls stream<> in the function argument, an interface for FIFO is generated.

Table 1 shows hardware resource utilization. The data dos not include resources of FIFO buffers. As shown in the table, the hardware resource is very small in the FPGA device. After synthesis and P&R, the clock period is estimated. The estimated clock period of the framework is 4.38ns, which is capable of work at 200MHz.

In summary, an example of inner-FPGA component technology is introduced. FPGA circuit design by component, based on publish/subscribe communication has potential to ease the FPGA development thanks to reusability of the circuit modules.

Table 1 Hardware resource utilization

Name	FF	LUT
Expression	0	4
Multiplexer	-	130
Resister	2	-
Total	2/407,600	134/203,800

V. Proposal: Inter/Intra-FPGA component Framework

Based on the inter-FPGA and intra-FPGA component technology which were described in the previous sections, a novel development method of intelligent robots using FPGA acceleration is proposed. As a result, a novel approach to improve the reusability of FPGA IPs by introducing a component-based design method is enabled. Several simple case studies are described using the design methodology.

Figure 9 briefly shows the concept of the integration of inter-FPGA component and intra-FPGA component. In this example, a camera image is input to FPGA device from left side. Then, the image processing, which is a “node”, processes the image and some results output to a “topic”. Here, the meaning of “node” and “topic” is similar to ROS system. The results are received by two nodes. One is control logic (down) and the other is image report (right). The control logic receives the result from image processing and outputs some control value to a topic, which is sent to motor actuating. This is one feedback loop. This feedback loop is composed of the intra-FPGA components and it would provide a very short response time (micro seconds). At the same time, the image report node sends the results of image processing node to a cloud server in the right of the figure. The processing results of the cloud server is sent back to a node “control from cloud” and gives some value for motor actuation. The “control from cloud” may need several milli-seconds latencies, however, the cloud processing can utilize huge-database and scalable computing/storage resources. Therefore, the combination of edge and cloud computing is inevitable for realizing intelligent robotics.

The sending to cloud server or receive from cloud server need the functionality of inter-FPGA component technology. Thus, the inter-FPGA and intra-FPGA component technology works in a system which combines edge computing using FPGA and communicating with a cloud server.

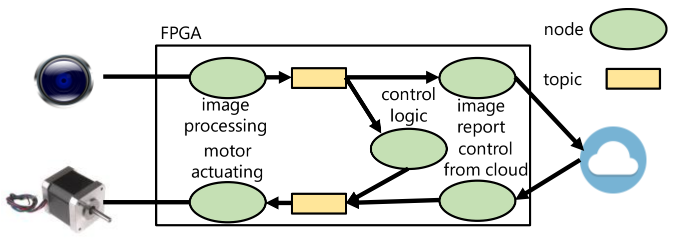


Figure 9 The concept of component-based FPGA development by inter/intra-FPGA component

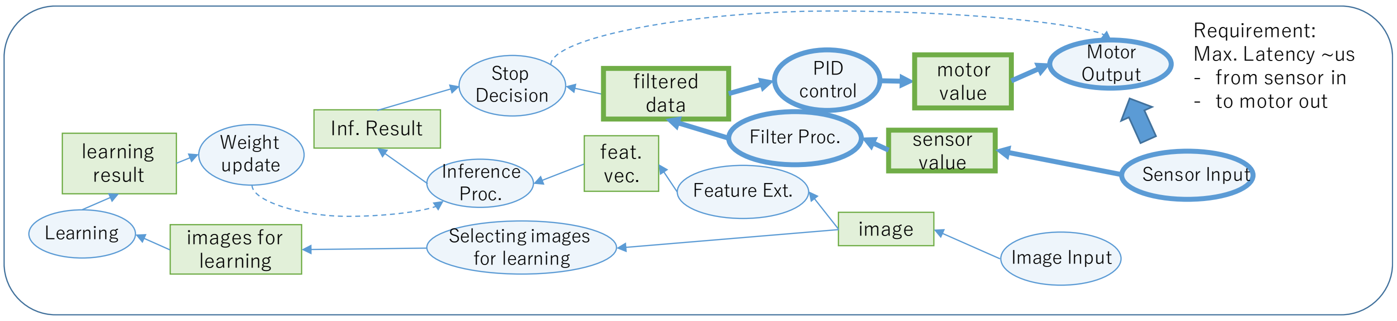


Figure 10 An example system model of an intelligent robot vision system with service call (dashed lines), with a latency requirement (sensor to motor, micro seconds)

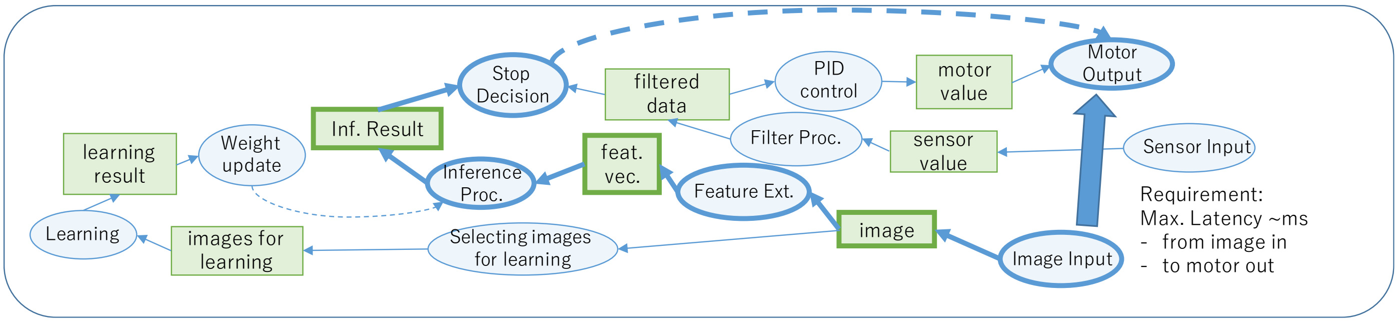


Figure 11 Another requirement of maximum latency from image to motor, milli second

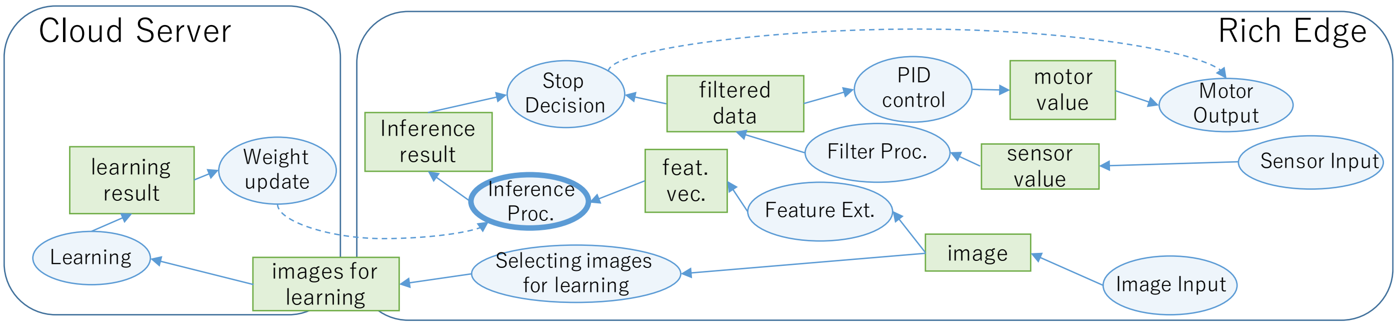


Figure 12 An example deployment among edge and cloud server

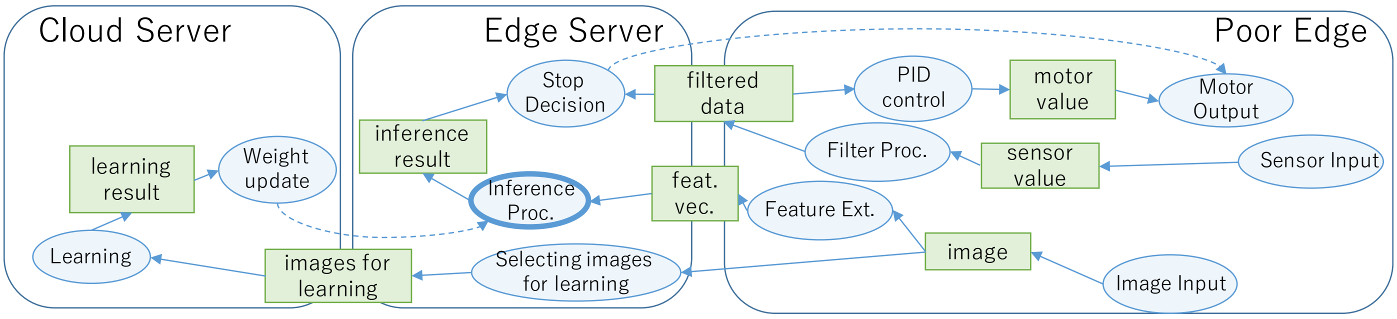


Figure 13 Another example deployment among edge, edge server and cloud server

Figure 10 shows a detailed example system model of an intelligent robot vision system with a service call, with a latency requirement. Since this system is more complex than the previous system (Fig. 9), only necessary parts are explained. From the right side, a sensor input comes in. Motor output is in the system, too. Let's assume the system is implemented in software, basically. FPGA would be installed only if it is necessary.

The requirement in this example is that the maximum latency time is several micro-seconds from the sensor input to the motor output. To achieve the requirement, the system should take care that the total latencies of "Filter Proc." and "PID control" nodes are below the requirement.

Figure 11 illustrates another example of a latency requirement using the same system. Here, the requirement is that the maximum latency time is several milli-seconds from the (camera) image input to the motor output. This indicates visual feedback to the motor. To achieve the requirement, the system should take care that the total latencies of "Feature Extraction", "Inference Processing" and "Stop Decision". The "Inference Processing" means Deep Neural Network (DNN) processing does some classification processing and outputs its result. This whole loop is intended that if the camera input image is somewhat dangerous, the motor is stopped emergently. Usually, DNN processing tends to take time. Therefore, the system should take care of the latency of the part is within the requirement. One may think using FPGA for the part is effective. Then, an inter-FPGA component can be used to accelerate the processing, since the inter-FPGA component can communicate with other software in ROS manner.

Deployment of a software component is another practical issue of building an intelligent robot. An example of deployment among edge and cloud server is shown in Figure 12. The system model is separately deployed into "Rich Edge" and "Cloud server" in this example. The requirements of Fig. 10 and Fig. 11 may be satisfied by this partitioning, only if the edge computing performance is enough rich. Or, some processing node should be offloaded into FPGA. The partitioning process needs to take care of the communication data amount and bandwidth among the target computing resources (i.e. edge node and cloud server) for the deployment.

Figure 13 is another example deployment among edge, edge server and cloud server. There three computing nodes in the deployment target environment: "Poor Edge", "Edge Server" and "Cloud Server". This is another typical case of building small mobile robot with a poor microprocessor which is low-power consumption. In such case, a server computer is prepared at near-edge location since the cloud server is too far to communicate with frequently. In this case, the "Edge Server" may oversee DNN processing to satisfy the latency requirement.

What we learned from this example is that using the system model make the latency requirement clear, and what processing needs to be speed up in the system. Furthermore, if we can use code-generation from the model to an implementation, we can modify the model and we can obtain the implementation for the target computing environment depending on the situation. Therefore, the complexity of the development of intelligent robot can be reduced by using the proposed inter-FPGA/intra-FPGA component technology.

VI. Conclusion

A component-based design method using inter/intra-FPGA component technology is proposed. The inter/intra-FPGA component technology is inspired by the ROS (Robot Operating System) and the Publish/Subscribe communication model, which improves the component reusability. It is also expected to promote model-driven development as shown in the example models of intelligent robots. The proposed design flow is expected to contribute design productivity of intelligent robots.

Acknowledgement

The author would like to thank to the people who concerns to this work in Utsunomiya University. Here list the names: Kazushi Yamashina, Takuya Matsumoto, Hitomi Kimura, Yuhei Sugata, Toki Kobayashi, Aoi Soya, Kenta Arai, Kanemitsu Ootsu and Takashi Yokota.

The author also would like to thank to the committee members of the SCOPE project for giving valuable comments: Satoshi Funada (e-Trees.Japan), Masayuki Isobe (FiVE Solutions, Inc.), Noriaki Ando (AIST), Keitaro Naruse (University of Aizu), Yukinori Ishikawa (University of Aizu), Midori Sugaya (Shibaura Institute of Technology), Kentaro Sano (Tohoku University, RIKEN), Hakaru Tamukoh (Kyushu Institute of Technology) and Hideki Takase (Kyoto University).

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. http://doi.org/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. http://doi.org/10.1587/transinf.2017RCP0011

[5] Takeshi Ohkawa, Kazuzhi Yamashina, Takuya Matsumoto, Kanemitsu Ootsu, Takashi Yokota, "Automatic Generation Tool of FPGA Components for Robots,", IEICE transactions on Information and Systems, Vol.E102-D, No. 5, pp.1012-1019, May 2019. http://doi.org/10.1587/transinf.2018RCP0004

[6] Takeshi Ohkawa, Yuhei Sugata, Harumi Watanabe, Nobuhiko Ogura, Kanemitsu Ootsu, Takashi Yokota, "High Level Synthesis of ROS Protocol Interpretation and Communication Circuit for FPGA," 2nd Workshop of Robot Software Engineering 2019 (RoSE2019) held with ICSE2019

[7] Kenta Arai, Takeshi Ohkawa, Kanemitsu Ootsu and Takashi Yokota, "Proposal of Publish/Subscribe Communication Framework for Circuit Components on FPGA," In proc. of Asia Pacific Conference on Robot IoT System Development and Platform 2018 (APRIS2018), an extended abstract of ESS2018 (Embedded System Symposium 2018), 2018.

[8] Kenta Arai, Takeshi Ohkawa, Kanemitsu Ootsu and Takashi Yokota, "Component-based FPGA development of intelligent image processing for industrial IoT devices," In proc. of 1st International Workshop on Embedded Software for Industrial IoT 2018 (ESIIT2018) held with DATE18

[9] 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, 2013. https://doi.org/10.1145/2641361.2641374

[10] Takeshi Ohkawa, Takashi Yokota, Kanemitsu Ootsu, "A Prototyping System for Hardware Distributed Objects with Diversity of Programming Languages --- Design and Preliminary Evaluation," Proc. 2013 International Conference on Field Programmable Technology (ICFPT 13), pp.474-477, (2013.12) https://doi.org/10.1109/FPT.2013.6718418

[11] H. Tamukoh and M. Sekine, "Design of Networked hw/sw Complex System using Hardware Object Model and Its Application," Proc. of 39th Annual Conference of the IEEE Industrial Electronics Society, pp. 2250-2255, 2013.

[12] Takeshi Ohkawa, Daichi Uetake, Kanemitsu Ootsu, Takashi Yokota, "Component-based System Design of FPGA and Software for Intelligent Real-time Systems," Proc. 2014 International Workshop on Smart Info-Media Systems in Asia (SISA 2014), Ho Chi Minh City, Vietnam, pp.64-69, (2014.10) https://www.ieice-sisa.org/?page_id=402#tSS1-04

[13] Yutaro Ishida, Takashi Morie, Hakaru Tamukoh, "A Hardware Accelerated Robot Middleware Package for Intelligent Processing on Robots," 2018 IEEE International Symposium on Circuits and Systems (ISCAS), pp.1-8, 2018. http://doi.org/10.1109/ISCAS.2018.8351722

[14] 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, http://doi.org/10.1145/2968478.2968502

[15] Noriaki Ando, Takashi Suehiro, Kosei Kitagaki, Tetsuo Kotoku and Woo-Keun Yoon 2005, RT-middleware: distributed component middleware for RT (robot technology), In Proc. of IEEE/RSJ International Conference on Intelligent Robots and Systems (2-6 Aug. 2005), IROS 2005, 3933 - 3938. DOI= http://dx.doi.org/10.1109/IROS.2005.1545521

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

[17] https://www.omg.org/spec/CORBA/

[18] https://www.omg.org/spec/DDS/

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

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

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