2022 Tenth International Symposium on Computing and Networking Workshops (CANDARW) | 978-1-6654-7532-7/22/$31.00 ©2022 IEEE | DOI: 10.1109/CANDARW57323.2022.00090

FPGA-Based Control of Remote Robotic Hands and ROS2 Integration

Ludi Wang
Tokai University
Japan
ludiwangjp@gmail.com

Takeshi Ohkawa
Tokai University
Japan
ohkawa.takeshi@tokai.ac.jp

***Abstract*— The remote robotic hand is required to imitate the hand movements of hand joints with low latency. In this report, the design and implementation of an FPGA-based robotic hand with remote control is proposed. To reduce the latency of controlling the robot hand, the waiting time for processing data needs to be shortened. Therefore, the authors use FPGA to accelerate the data calculations and ROS2 integration system. The authors carried out a preliminary evaluation of the processing speed of using an FPGA to accelerate the control of a remote robotic hand. The measured processing time of calculating data for the angle between finger joints was 2 milliseconds.**

Keywords—FPGA, ROS2, robot hand

I. Introduction

In recent years, reconfigurable high-speed processing and low-power devices, such as field-programmable gate arrays (FPGAs), which allow users to customize configurations after manufacturing, have become popular due to the slowdown in CPU performance improvements [1]. Because, unlike CPUs, FPGAs are able to continue to follow the Moore's Law curve, providing more logic and memory resources with each new generation of FPGAs [3]. For example, faster processing has been reported by using FPGA devices for image processing and recognition, pose and position estimation from sensor values, conversion of 3D coordinates, plotting, and self-position estimation. It is possible to perform the above data processing with CPU devices, but the increase in power consumption is a serious issue. The increased power consumption means a shorter battery life, so the robot must be equipped with a large and heavy battery. FPGAs can achieve high-performance processing that is not possible with software using hardware parallel processing with low power consumption and small space. It means that FPGA has a longer service life than CPU devices in the same processing performance. Therefore, more and more smart FPGA devices are replacing CPU devices, entering our lives and bringing many conveniences.

For a doctor who needs to operate on a patient in a remote location or scientists who need to accurately mix dangerous chemical reagents, remote robotic hands can solve many human problems. To realize remote robotic hands, there are many software processing which takes long time to process. For example, currently, the common control methods for robotic hands are command input by the user on a virtual console or a physical joystick manipulation. Motion capture has an advantage that does not need any physical contact, however, it needs complex processing for precise motion detection, which is expected to be accelerated with FPGA.

The development of robot system needs integration framework to reduce the development complexity. The Robot Operating System (ROS) is a set of software libraries and tools for building robot applications. The Robot Operating System (ROS) is a set of software libraries and tools for building robot applications. From drivers and state-of-the-art algorithms to powerful developer tools, which is a necessary feature for remote robotic hands [1].

Currently, there are many studies on FPGA, such as FPGA-based visual neuro-mimetic robot vision systems [4] and FPGA implementation of closed-loop control system for small-scale robot [5]. There are also studies using FPGA robots to compute vehicle coordinates. There is also an increasing number of studies on developing combinations of ROS2 and FPGA, for example, automated integration of high-level synthesis FPGA modules with ROS2 Systems [6].

In this study, an FPGA acceleration and the system integration of a control processing for a remote robotic hand is discussed. This study is characterized by the fact that each finger joint of the robot can move independently and its motion is controlled by the coordinates of the user's hand and the angle of the finger joints. The average delay that can be perceived by human vision and touch is 50-60 milliseconds [4]. Therefore, the total delay including ROS2 communication time is expected to be within 50-60 milliseconds.

II. Proposed Design of Remote Robotic Hands

In this section, the design of remote robotic hands is divided into a software section, a hardware section and a ROS2 communication section. In section A, the software part is presented, with a flowchart of the program designed in Python and the way the program is assembled. In section B, the hardware part is presented, describing the creation of the RTL Libraries and the IP core. In section C, ROS2 is presented, describing the system architecture and its role.

Figure 1 shows the proposed system overview of remote robotic hands. The coordinates of the user's hand collected by the motion capture device can be used to reconstruct the hand motion. The acquired coordinate data is published by ROS2 as a node, and FPGA device acquires the necessary data through the user. The image data is acquired by the camera and sent to the FPGA device via the NODE Image. The key point analysis of the hand is performed by the FPGA device. It is sent to the coordinate calculation module via the NODE Key point. Based on the calculation results, control signals are generated and sent to the robot hand via NODE Control signals. This speeds up data communication, makes the control system of the robot hand simpler and more efficient and reduces the delays caused by data processing.

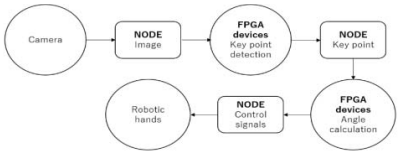

Fig. 1 Proposed design of a remote robotic hands

2832-1324/22/$31.00 ©2022 IEEE
DOI 10.1109/CANDARW57323.2022.00090

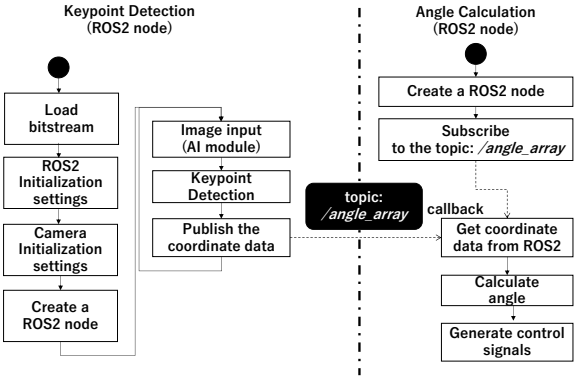

Fig. 2 Flowchart of the proposed system

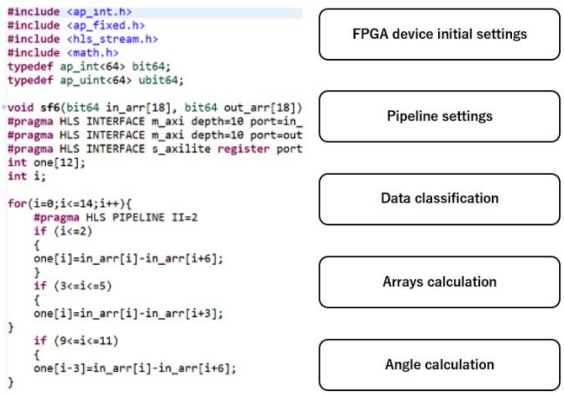

Fig. 3 Coordinate calculation of IP core source code

A. Software

Software development is carried out using Python. As shown in Figure 2, There are two main node components, Key point detection and Angle Calculation. In Key point detection, the first step is to first read the bitstream file and initialize the ROS2 and camera settings. The second step makes a node (Key point detection). The third step puts the images collected by the camera into the AI module, detects the key points of the hand and publishes them in topic (Angle array). After that step 3 is repeated and the data is continuously sent. In Angle Calculation, a ROS2 node is first made to receive the coordinate information from the topic. After that, the FPGA is used to calculate the angle. Finally the control signal is generated based on the calculated angle.

B. Hardware

The hardware part uses Vivado HLS 2022.1, one of the advanced synthesis tools offered by Xilinx, to automatically generate the HDL for Xilinx FPGAs from source codes written primarily in C, C++, or System C [7]. It is not as efficient as Verilog's use of FPGA performance but is more conducive to the development of this experiment.

a) RTL Libraries

MMIO (Memory-Mapped IO) (PS and PL share memory addresses) reads the array IN_ARR and outputs the result in OUT_ARR when the calculation is finished.

As shown in Figure 3, the host protocol is m_axi (AXI master), which is used to transmit a large amount of hand coordinate data. A set of coordinate data is 153 bytes.

Since each finger has eight joints, two sets of coordinates are transmitted. To do this, the CPU sequentially increases the address offset and transfers the coordinate data to the IP core. 5 fingers have a total of 20 joints, so a total of 612 bytes is transferred. The PIPELINE pragma is also used to speed up the loading of data to the IP core. This approach can be effective in improving the performance of the FPGA. Fmax is improved by introducing registers in critical paths and reduces the number and the distance of gates, where a signal must pass through in a cycle [8]. It will reduce the time interval between function and cycle start. From the three-dimensional coordinates of the eight joints of the two fingers, the corresponding ACOS values are obtained. Finally, the angle needed for the final control of the robot is calculated by using radians.

The control signal, the input signal (in arr_v) and the output signal (out arr_v) are generated. They are used in the next part of the logic synthesis.

b) IP core

As shown in Figure 4, the RTL Libraries are the KV260_0. The HLS input port INPUT is merged into AXI port m_axi_INPUT_r, and the output port OUTPUT is merged into m_axi_OUTPUT_r and control port s_axi_CTRL. This experiment must use two AXI Smart Connects for data input and output. Data transfer between interfaces with different data widths is automatically converted by AXI SmartConnect [8]. It acts as a logical connection between PS and PL for data input and data output by connecting to the ZYNQ processing system.

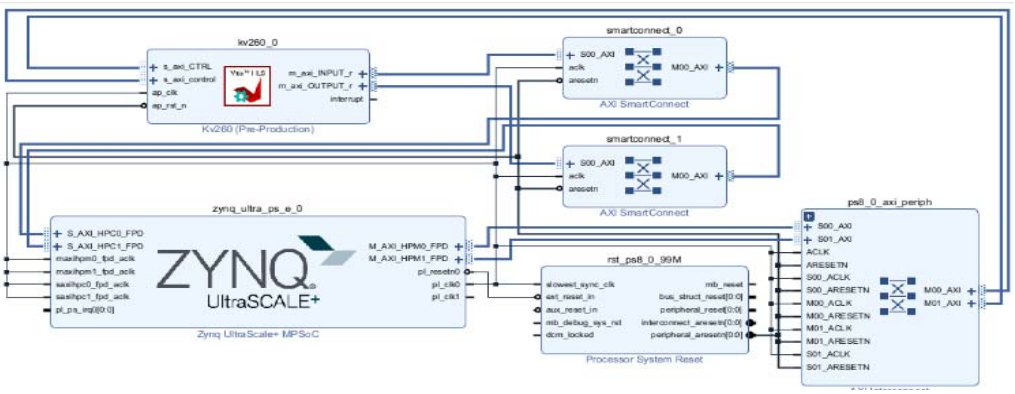

Fig. 4 Hardware block diagram of IP integration

C. ROS2 Communication

ROS2 uses DDS (Data Distribution Service) as a communication middleware with the aim of operating multiple robots in an environment with limited communication resources, such as wireless communication, and operating in an embedded microcomputer environment to achieve real-time control.

III. Experimental Results

The performance evaluation was done for the CPU, ARM and the FPGA comparing their processing speeds. The boards used for the three devices are listed in Table 2. The AMD Ryzen 9 5900HS is a high performance CPU with 8 cores and 16 threads, the Raspberry Pi 4B has a 4 core ARM Cortex-A72 CPU. The PYNQ-Z2 is a low cost Zynq 7000 FPGA development board. The Xilinx KV260 is based on the Zynq Ultrascale + high performance development board.

The program used to test the CPU is a python version with sixteen threads of processing. This may not take advantage of the CPU's full performance. The processing time for the same data on the PC, Raspberry Pi 4B, PYNQ-Z2, and KV260 was recorded. The calculations were repeated 100, 6000 and 10,000 times. The difference in calculation speed between them is shown in Table 3. The bigger the number of calculations becomes, the more the difference in the time spent on the calculation increases. The KV260 takes 2 milliseconds to process one iteration. 10,000 iterations take 1.03 seconds. Relatively high-performance CPU devices consume more power and have more computational power than FPGA, but the FPGA can still do the computation faster.

As shown in Figure 5, the latency was measured using TCP/IP and ROS2 when data was sent from the KV260 end node and received from the robot hand end node respectively. As shown in Table 1, the size of each data set containing coordinate information about the hand was 153 bytes and the average latency was 8 milliseconds for data transmission using TCP/IP and 8 milliseconds for data transmission using ROS2.

The total latency for motion capture, ROS2 data communication, data processing, and motor control is estimated within 50 milliseconds. The measured processing time of calculating data was 2 milliseconds. The data communication used 8 milliseconds. The Combined processing time and communication time of 10 milliseconds and satisfy the requirement.

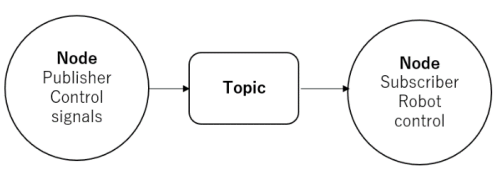

Fig.5 ROS2 node graph of the experimental setup

Table 1 Communication latency time

	Data size (Byte)	Latency (ms)
ROS2	153	8
TCP/IP	153	8

Table 2 Environment for the performance evaluation

	Board
PC	AMD Ryzen 9
Mini-computers	Raspberry Pi 4
FPGA	PYNQ-Z2 (Xilinx)
FPGA	KV260 (Xilinx)

Table 3 Coordinates calculation processing time (unit: s)

Number of times	AMD Ryzen 9	Raspberry Pi 4	PYNQ-Z2	KV260
1	0.003	0.005	0.002	0.002
100	0.031	0.0318	0.025	0.025
6000	1.782	2.830	1.253	0.682
10000	2.947	4.156	2.092	1.037

IV. Conclusion

This paper proposed an acceleration and an integration method of coordinates calculation for an FPGA-based remote robotic hand control. Modules including ROS2 data communication, data processing (FPGA module) and control signal generation functions were developed. RTL Libraries and IP cores were produced. A preliminary evaluation of the processing speed of FPGA accelerated control of a remote robotic hand was carried out. The measured processing time of calculating data for the angle between finger joints was 2 milliseconds. 10,000 coordinate calculations take 2.947 s on a conventional PC, but only 1.037 s on the FPGA. Compared to a traditional CPU, FPGAs are more efficient and therefore faster to compute.

Acknowledgment

This work was supported by JST CREST Grant Number JPMJCR19K1, Japan.

References

[1] "ROS 2 Documentation" [Online] Available: ROS 2 Documentation − ROS 2 Documentation: Rolling documentation

[2] M. Horowitz、"1.1 computing's energy problem (and what we can do about it)," in Solid-State Circuits Conference Digest of Technical Papers (ISSCC), 2014 IEEE International. IEEE, pp. 10–14, 2014. DOI: 10.1109/ISSCC.2014.6757323

[3] S. Kestur, J. D. Davis and O. Williams, "BLAS Comparison on FPGA, CPU and GPU," 2010 IEEE Computer Society Annual Symposium on VLSI, 2010, pp. 288-293, DOI: 10.1109/ISVLSI.2010.84.

[4] Rayner, K., Smith, T. J., Malcolm, G. L., & Henderson, J. M. (2009). Eye movements and visual encoding during scene perception. Psychological Science, 20(1), 6–10.

[5] Wei Zhao, Byung Hwa Kim, A. C. Larson and R. M. Voyles, "FPGA implementation of closed-loop control system for small-scale robot," ICAR '05. Proceedings., 12th International Conference on Advanced Robotics, 2005., 2005, pp. 70-77, doi: 10.1109/ICAR.2005.1507393.

[6] D. P. Leal, M. Sugaya, H. Amano and T. Ohkawa, "Automated Integration of High-Level Synthesis FPGA Modules with ROS2 Systems," 2020 International Conference on Field-Programmable Technology (ICFPT), 2020, pp. 292-293, doi: 10.1109/ICFPT51103.2020.00052.

[7] Ilya Ganusov, Henri Fraisse, Aaron Ng, Rafael Trapani Possignolo, Sabya Das, "Automated extra pipeline analysis of applications mapped to Xilinx UltraScale+ FPGAs," in 2016 26th International Conference on Field Programmable Logic and Applications (FPL), 2016 IEEE International. pp. 1–4, 2016 DOI: 10.1109/FPL.2016.7577344.

[8] "Vivado Design Suite" [Online] Available: SmartConnect (xilinx.com)