Impact of PS Load on FPGA Object Detection System Performance

Yusuke Watanabe
CRAFT WORK Co., Ltd,
5F OS Bldg., 3-5-15 Shibasaki-cho, Tachikawa, Tokyo, 190-0023, Japan
Graduate School of Life Science and Systems Engineering, Kyushu Institute of Technology, 2-4 Hibikino, Wakamatsu-ku, Kitakyushu, Fukuoka, 808-0196, Japan

Hakaru Tamukoh
Graduate School of Life Science and Systems Engineering, Kyushu Institute of Technology,
2-4 Hibikino, Wakamatsu-ku, Kitakyushu, Fukuoka, 808-0196, Japan
E-mail: watanabe.yusuke898@mail.kyutech.jp, tamukoh@brain.kyutech.jp
http://www.lsse.kyutech.ac.jp/english/

Abstract

A field-programmable gate array (FPGA) device which has a Zynq architecture becomes popular these days. It is featured by integration of processing system (PS) and programmable logic (PL) into a single chip. While we tend to focus on the performance of PL, we can not ignore PS load completely. In this paper, using a Zynq FPGA board, we explore how our object detection system performance changes with PS load and report our experiment results.

Keywords: FPGA, Zynq, PL, PS, Parallel Processing, System Performance.

1. Introduction

In robotics, demands for running neural networks such as object detection on a low energy consumption device are extremely high. We can soon find papers about relationship between neural networks and energy consumption [1], [2] rogrammable gate arrays (FPGAs) meet the requirement as they are well known for their low energy consumption [3], [4]. Thanks to the recent advancement of FPGAs, architectures such as Zynq which are featured by integration of the software programmability of a processing system (PS) and the hardware programmability of a programmable logic (PL) into a single device have become popular.

When we employ FPGA devices, though we tend to focus on the PL performance with regard to the system performance [5], [6], we can not ignore the PS performance completely. The larger a system becomes, the larger its PS load tends to be. Therefore, we research an impact of PS load on the system performance by measuring the execution time. Considering both the PS and the PL performance together, we intend to improve the whole system performance.

2. Method

We use our object detection system shown in Fig. 1 to research influence of the PS load on PL and system performances. Our object detection in PL is implemented based on Ref. 7 and Ref. 8 and performs convolution and max pooling operations which are widely used in a neural network. Our System works on an FPGA board having a Zynq architecture to which a USB camera and a monitor are directly connected. An image file and the USB camera image are used as input data to our system. We change the PS load in our system by switching these two input methods. After receiving input data, our system

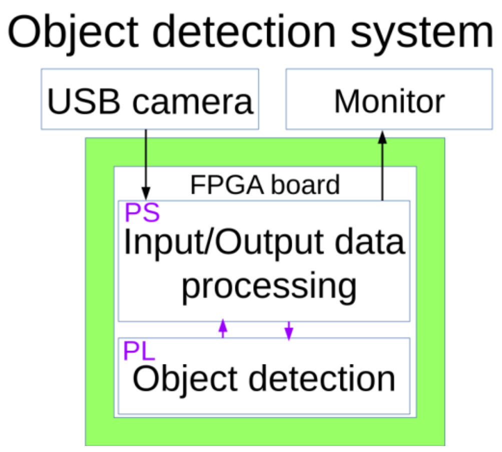

Fig. 1: Our object detection system on a FPGA board

then detects objects and pass an output image on which bounding boxes are drawn to the monitor. We measure execution time of both object detection in PL and inference in the application to evaluate our system performance. The inference time represents a period of time from getting input image data to displaying output image data.

We conduct several experiments, collect the execution time and compare the time to evaluate the impact of the PS load on our object detection system performances.

3. Experiment

We employ a Zynq UltraScale+ MPSoC ZCU102 evaluation board, Logicool C270 HD WEBCAM and BenQ GW2480T as our experiment environment and conduct three experiments executing the following applications.

(i) FPGA application using multiple threads.
(ii) C++ and FPGA applications.
(iii) FPGA application using only the main thread.

Our FPGA applications in all experiments are generated by AMD Xilinx tool and perform completely the same object detection in PL and the other operations in PS as shown in Fig. 1. While there is no difference in the PL operations, the PS operations vary in each application. As execution time, we measure CPU time by C library clock() function and measure wall time by C library clock_gettime() function. CPU time refers to the time CPU is busy in processing the program's instructions. Wall time refers to the elapsed time during measurement. OpenCV imshow() function is called to display an output image from our system.

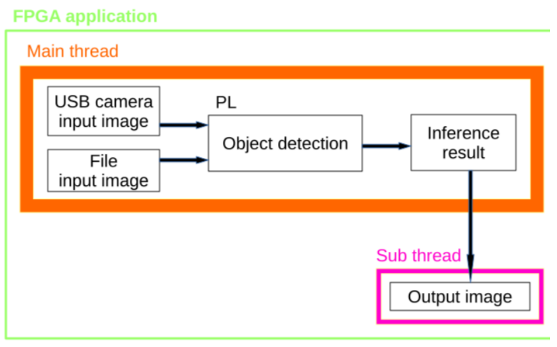

Fig. 2: An overview of the first experiment application

Fig. 2 represents an overview of processing in the FPGA application executed in the first experiment. We switch two input methods by changing arguments passed to the application. We call pthread_create() function in the POSIX thread libraries to display an output image in a sub thread.

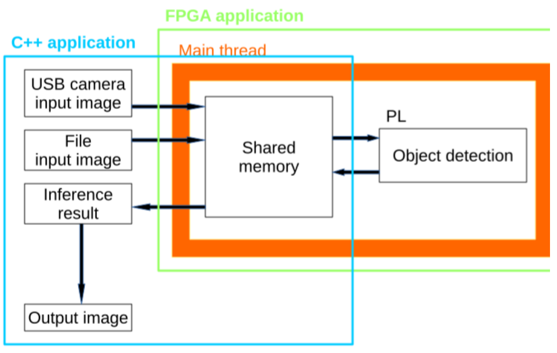

Fig. 3: An overview of the second experiment

Fig. 3 represents an overview of the second experiment application. We execute the C++ application which is generated by cmake and automatically executes the FPGA application. Shared memory is used to pass data between the C++ and the FPGA applications. Whatever the input methods are, input data to the FPGA application are passed through the shared memory and therefore we only employ the image file to measure the execution time. In the second experiment, we also measure the execution time of the FPGA application which is the time from right after the beginning of the FPGA application to right before the end of the FPGA application.

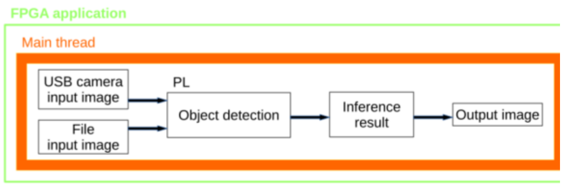

Fig. 4: An overview of the third experiment application

Fig. 4 shows an overview of the third experiment application which is almost the same as the first experiment application. The only difference is that the function to display an output image is called in the main thread instead of called in a sub thread.

We execute each application in the three experiments for ten times and record its execution time. When we use the USB camera, we regard the time when any object is detected as the execution time. We prepare an image file so that any object is detected by the system and we use it in all experiments.

4. Results

Table 1, Table 2, Table 3, Table 4 and Table 5 show the results of our experiments. Min. is the minimum execution time and Max. is the maximum time. Ave. means the mean time of the ten measurement. The time is showed in the second.

Table 1. CPU time in the first experiment

			Image input	
			Camera	File
CPU time (sec)	Inference	Min.	1.344	5.978 x10^{-1}
		Ave.	1.348	5.988 x10^{-1}
		Max.	1.350	5.997 x10^{-1}
	PL	Min.	1.042 x10^{-1}	1.016 x10^{-1}
		Ave.	1.046 x10^{-1}	1.028 x10^{-1}
		Max.	1.080 x10^{-1}	1.048 x10^{-1}

Table 2. Wall time in the first experiment

		Image input	
		Camera	File
Inference	Min.	1.215	5.136 x10^{-1}
	Ave.	1.219	5.145 x10^{-1}
	Max.	1.221	5.161 x10^{-1}
PL	Min.	5.231 x10^{-2}	5.271 x10^{-2}
	Ave.	5.232 x10^{-2}	5.273 x10^{-2}
	Max.	5.233 x10^{-2}	5.281 x10^{-2}

Table 1 is the CPU time of the first experiment. Table 2 is the wall time of the first experiment. From Table 1 and Table 2, the PS load of the USB camera is higher than the image file and the execution time in PL is almost the same in both input methods. These results indicate that there is no impact of the PS load on the PL execution.

Table 3. CPU time in the second experiment

			Image input
			File
CPU time (sec)	Inference	Min.	5.797
		Ave.	5.808
		Max.	5.817
	FPGA application	Min.	4.071
		Ave.	4.073
		Max.	4.074
	PL	Min.	5.285 x10^{-2}
		Ave.	5.294 x10^{-2}
		Max.	5.301 x10^{-2}

Table 3 is the CPU time of the second experiment. We do not show the wall time because both the CPU and wall time are almost the same. From Table 3, the CPU time in PL is shorter than the first experiment application though the inference CPU time is longer, and the wall time in PL is almost the same.

Table 4. CPU time in the third experiment

			Image input	
			Camera	File
CPU time (sec)	Inference	Min.	1.258 x10^{-1}	5.067 x10^{-1}
		Ave.	1.279 x10^{-1}	5.076 x10^{-1}
		Max.	1.285 x10^{-1}	5.082 x10^{-1}

	PL	Min.	5.218 x10^{-2}	5.262 x10^{-2}
		Ave.	5.227 x10^{-2}	5.272 x10^{-2}
		Max.	5.231 x10^{-2}	5.281 x10^{-2}

Table 5. Wall time in the third experiment

			Image input
			Camera
Wall time (sec)	Inference	Min.	1.209
		Ave.	1.213
		Max.	1.236
	PL	Min.	5.231 x10^{-2}
		Ave.	5.232 x10^{-2}
		Max.	5.232 x10^{-2}

Table 4 is the CPU time of the third experiment. Table 5 is the wall time of the third experiment. We do not show the wall time of the image file input because both the CPU and wall time are almost the same. From Table 4 and Table 5, the CPU time of both inference and PL is shorter than the first experiment application and the wall time is almost the same.

From the results of all experiments, we paid attention to the relationship between the number of threads in PS and the execution time in PL and show it as Table 6. The execution time in Table 6 is the mean time of the image file input.

Table 6. Relationship between the number of threads and the execution time in PL

		The number of threads	Mean CPU time in PL (sec)	Mean wall time in PL (sec)
Experiment	One	2	1.028 x10^{-1}	5.273 x10^{-2}
	Two	1	5.294 x10^{-2}	
	Three	1	5.272 x10^{-2}	5.232 x10^{-2}

5. Conclusion

There is no impact of the PS load on PL execution as for the wall time. Although the CPU time in PL increases when we increase the number of threads in PS, the wall time in PL does not change. We can conclude the system performance highly depends on the PS load from our experiments. To improve the system performance, we can consider the PS and PL separately because the PS load does not affect the wall time in PL.

As a future work, we need to further research whether the increase of the CPU time in PL affects the wall time in PL.

References

1. C. Profentzas, M. Almgren, O. Landsiedel, “Performance of deep neural networks onlow-power IoT devices”, Proceedings of the Workshop on BenchmarkingCyber-Physical Systems and Internet of Things, 2021.
2. L. Caballero, Á. Perafan, M. Rinaldy, W. Percybrooks, “Predicting the Energy Consumption of a Robot in an Exploration Task Using Optimized Neural Networks”, Electronics, 10, 920, 2021.
3. M. Qasaimeh, K. Denolf, J. Lo, K. Vissers, J. Zambreno and P. H. Jones “Comparing Energy Efficiency of CPU, GPU and FPGA Implementations for Vision Kernels”, *IEEE International Conference on Embedded Software and Systems (ICESS)*, pp. 1-8, 2019.
4. H. Nakahara and T. Sasao, “A High-speed Low-power Deep Neural Network on an FPGA based on the Nested RNS: Applied to an Object Detector”, IEEE International Symposium on Circuits and Systems (ISCAS), pp. 1-5, 2018.
5. L. Mo, C. Wu, L. He and G. Chen, “Layout driven FPGA packing algorithm for performance optimization”, IEICE Electronics Express, 2017.
6. T. Nguyen, C. MacLean, M. Siracusa, D. Doerfler, N. J. Wright and S. Williams, “FPGA-based HPC accelerators: An evaluation on performance and energy efficiency”, Concurrency Computat Pract Exper, 2022.
7. M. Courbariaux, I. Hubara, D. Soudry, R. El-Yaniv and Y. Bengio, “Binarized Neural Networks: Training Deep Neural Networks with Weights and Activations Constrained to +1 or -1”, 2016.
8. H. Nakahara, H. Yonekawa, H. Iwamoto and M. Motomura, “A Batch Normalization Free Binarized Convolutional Deep Neural Network on an FPGA (Abstract Only)”, pp. 290-290, 2017.

Authors Introduction

Mr. Yusuke Watanabe

He received his B.Eng. and M.Eng. degrees from Waseda University, Japan, in 2007 and 2009. He is pursuing his Ph.D. degree in Kyushu Institute of Technology. He is working for CRAFT WORK Co., Ltd.

Mr. Hakaru Tamukoh

He received his B.Eng. degree from Miyazaki University, Japan, in 2001. He received his M.Eng. and Ph.D. degrees from Kyushu Institute of Technology, Japan, in 2003 and 2006, respectively. He was a postdoctoral research fellow at Kyushu Institute of Technology, from 2006 to 2007. He was an assistant professor at Tokyo University of Agriculture and Technology, from 2007 to 2013. He is currently a professor in the graduate school of Life Science and Systems Engineering, Kyushu Institute of Technology, Japan. His research interest includes digital hardware design, soft-computing and home service robots. He was the author of works that won the Best Paper Award at IJCNN 2019, the Best Live Demonstration Award at ISCAS 2019, the Best Paper Award at ICONIP 2013. He is a member of IEEE, IEICE, JNNS.