Design of the convolution layer using HDL and evaluation of delay time using a camera signal

1st Ryoki KAMESAKA
System of Engineering
Kochi University of Technology
Kochi, Japan
235119g@gs.kochi-tech.ac.jp

2nd Yukinobu HOSHINO
System of Engineering
Kochi University of Technology
Kochi, Japan
hoshino.yukinobu@kochi-tech.ac.jp

***Abstract*—Research of the deep-learning has been applied to several problems, such as image recognition, voice recognition, and natural language processing. Those are achievement results in the world. At the same time, it is used in various applications and research fields. This technology is expected for using embedded systems of IoT devices such as smartphones and tablet-pc. This technology is expected for using the embedded system of IoT devices such as smartphones and tablet-pc. Deep-learning requires large computational resources and high power consumption. GPU is a kind of solution to computational resources. In general, most systems need to use GPU acceleration. However, it is a hard point to implement on tiny embedded devices under the required GPU. Also, in recent years, FPGAs have been applied to image processing technology fields such as defect inspection of automobiles, security systems, and industrial products. Hardware acceleration is one of the techniques to improve processing speed. This technique is often used in the field of image processing. Our research tried to design the hardware acceleration approach for the convolutional neural network and we compared the software processing approach and our hardware approach. The software processing approach is conventional. As a result, our hardware-based convolutional layer can show high-speed performance than the software-based design. These hardware-based modules were implemented on FPGA. We used HDL(Hardware Description Language) to implement on FPGA. By using this, you can create a logic circuit like programming. It is usually used when designing the FPGA circuit. In this paper, we show the detail of the architecture and the comparison result of processing time with CPU.**

***Index Terms*—Convolutional Neural Network, FPGA, Pipeline**

I. Introduction

Deep learning is a method of machine learning that the computer is able to learn the tasks that humans naturally perform. This technology supports the development of artificial intelligence (AI), and this technology is beginning applicate in various fields. In recent years, deep learning has been applied to various problems such as image recognition, voice recognition, and natural language processing, and has achieved results.At the same time, it is used in various applications in any field and is expected to be used in embedded systems such as IoT devices [1]–[3]. Accelerators for deep learning include CPU, GPU, FPGA, and ASIC. Basically, AI technology, which like deep learning, is hard to implement on small embedded devices such as low power systems.

Because deep learning has a high computational cost, and generally requires a GPU [4]–[6]. Most computing libraries of AI technology has designed for using a GPU system. Thus, target architecture should implement a GPU as the acceleration of AI. Especially, GPU is indispensable for real-time object detection by the currently deep learning technology. As a reason, the largest amount of processing in deep learning is convolution processing and matrix operation by GPU architecture. Additionally, High-speed execution requires high-speed memory access and parallel computing performance of multiplication and addition. Embedded systems have become very important because they are this system is supporting in the backside of our life. It can be easy to find everywhere our daily lives. For example, consumer electronics, heating control systems, traffic lights, engine control systems, etc. As a requirement for embedded systems, it is necessary to balance power consumption and performance. Also, FPGAs have been applied to image processing technology fields such as defect inspection of automobiles, security systems, and industrial products [7]–[9]. FPGAs have better power performance than CPUs and GPUs, and their significance is increasing, day by day. Hardware acceleration is one kind of technique to improve the processing speed and it is often used in the field of image processing. Because each processing such as image processing filters and feature extractions are usually independent.

There are many studies related to FPGAs, and it has been reported that FPGAs perform as the processing speed [10]–[13]. Those perform are better compared to CPUs by implementing hardware image processing filters using FPGAs in most cases. It is able to describe the differences between the processing methods of FPGA and CPU. In the case of processing RAW data directly from a camera, CPU and GPU is necessary to store all camera data to RAM once. But, FPGA don't need to store all data to RAM once as shown in Fig .1. The time to access the memory can be reduced, and the data from the camera can be processed directly. So, image processing can be performed at the register transfer level. Also, when performing real-time processing using a camera, speed up can be expected by using pipeline processing.

Therefore, in this study, we aim to improve the calculation speed by designing a Convolutional Neural Network, as shown Fig .2.

978-1-7281-6932-3/20/$31.00 ©2020 IEEE

Convolutional Neural Networks in the field of image recognition works have achieved great performance in many tasks and are attracting attention. In particular, the features extracted from the hidden layer in CNN, that is trained using a large-scale object recognition data set represented by ImageNet. This algorithm is extremely versatile and is possibly useful in various fields.

We focus on the implementation of hardware the Convolutional Neural Networks. Additionally, we report on the implementation of the convolution layer and pooling layer using FPGA, and the result of the comparison of processing time with CPU.

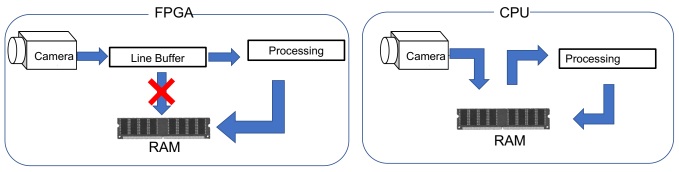

Fig. 1. Differences between CPU and FPGA processing. In case of processing RAW data directly from a camera, CPU and GPU is necessary to store all camera data to RAM once. But, FPGA don't need to store all data to RAM once.

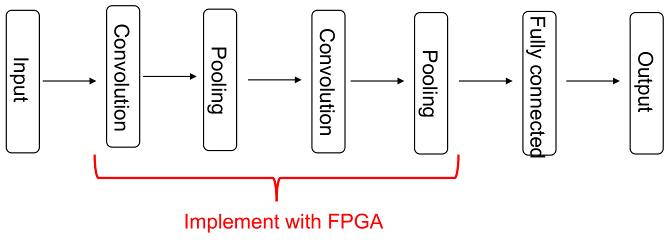

Fig. 2. The purpose of this research. Aim to improve the calculation speed by replacing the convolutional layer with FPGA.

II. Hardware Architecture

A. SoC FPGA and pipeline processing

This section explains an overview of the FPGA and details of the processing implemented. FPGA is an abbreviation of Field Programmable Gate Array. Users can design any circuit in HDL or circuit diagram, and write a program to FPGA. The flexibility of rewriting makes logical circuits of algorithms. There is possible to support new algorithms. Also, it is possible to adjust the granularity of pipeline processing and parallel processing. Because it is possible to design at the hardware level. CPU and GPU is necessary to store all camera data to RAM once. But, FPGA don't need to store all data to RAM once The method of executing processing one by one in time series is called sequential processing. CPU processing corresponds to this. Describes pipeline processing. For example, if the processing result of the instruction 1 is the input data of the instruction 2. Instead of waiting for the completion of the execution of the instruction 1 for all data, instruction 2 is executed the data for instruction 1 processing has been completed is sequentially performed. Although FPGAs have lower operating frequencies than CPUs and GPUs, they are expected to exhibit better performance than CPUs and GPUs by making good use of parallel and pipeline processing as shown Fig .3.

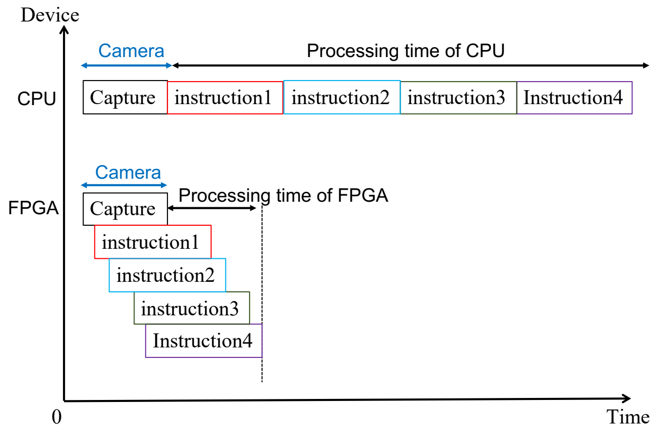

Fig. 3. Pipeline Processing. FPGA is expected to exhibit higher performance than CPUs and GPUs at lower frequencies by utilizing pipeline processing.

To realize the pipeline processing with the image processing filter, we used a line buffer structure [14]. The line buffer is a memory that stores the pixels of one horizontal line of one image. If the filter that performs image processing is 3×3, it is necessary to prepare three line buffers. Behavior of pipeline processing is shown in Fig .4. One pixel data sent from the camera is stored in Buf0. At the same time, the pixel data stored at address 0 of Buf0 is shifted to address 1 of Buf0. Similarly, the pixel data stored at the address 1 of Buf0 is shifted to the address 2 of Buf0. The process of shifting the pixel data to the next address is performed for all addresses. Each time data of one pixel is transferred from the camera, filter processing is performed on the pixel data in the part surrounded by the broken line frame. Each time one pixel of data is sent from the camera, filter processing is performed on the pixel data in the area enclosed by the broken line frame. It is possible to execute the filtering process in pixel units, not frame units. Because, the pixel data stored in the line buffer shifts.

In this study, we used Intel Cyclone V SoC (5CSE-MAF31C6) [15]–[18]. We used the board of Terasic DE1-SoC as shown in Fig .5. This SoC FPGA is equipped with a CPU (ARM-A9 processor) that can execute the OS and 1GB of DDR3 SDRAM. It means that it can realize hardware-software (HW/SW) co-design. It can make the most of the strengths of hardware and software. For example, by sharing the DDR3 SDRAM between the HPS side and the FPGA side, the image processing result on the FPGA can be confirmed from the HPS side usign OpenCV etc.

B. Overview of image processing system

In this section, Explain the image processing system that we designed. Fig .6 shows the image processing circuit designed for the FPGA in this experiment. In this experiment, two

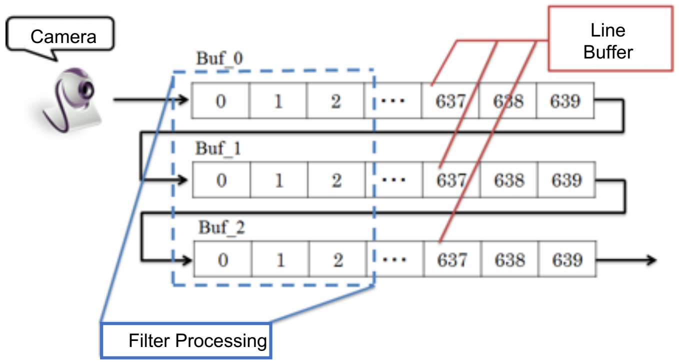

Fig. 4. Line Buffer. The pixel data stored in the line buffer shifts in synchronization with the camera clock, so, it is possible to execute the filtering process not on a frame basis but on a pixel basis.

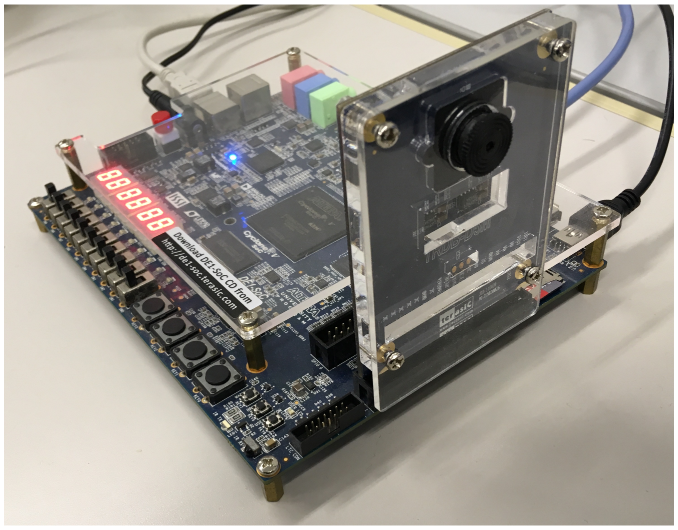

Fig. 5. FPGA DE1-SoC platform interfaced with the TRDB-D5M GPIO camera module.

convolutional layers were designed. When implementing a deeper layer model, the number of arithmetic units may be enormous. In addition, the implementation of the fully-connected layer involves floating-point arithmetic and may use a lot of memory. It may become insufficient hardware resources. This may affect the operation speed. Therefore, one layer was implemented with one FPGA, and a total of two convolutional layers were implemented using two FPGAs. This means that there are multiple Linux systems. It also makes it easier to analyze the output results in each layer.

There are two types of design: HDL design and high-level synthesis. In this study, we designed with HDL, due to maximizing the performance of the chip. We used Verilog HDL and System Verilog for the design on the FPGA side, and we used a TRDB-D5M camera manufactured by Terasic. Verilog HDL and System Verilog are one of the HDL. First, the RAW data is transferred from the camera. This RAW data is transferred by the Bayer pattern from the camera, so it is necessary to convert. The RAW image captured from the camera is converted to RGB and GRAY by a color conversion circuit. In the image processing system designed in this experiment, the camera operates at 25 MHz and the FPGA operates at 50 MHz. A FIFO(First In First Out) was used to absorb the difference between these two timings. Convolution and MAX pooling processing is performed on this GRAY image by the filter circuit. The filter size of the convolution layer in this experiment is 3×3. Also, in the pooling layer, 2×2MAXpooling is performed. Then, the data is transferred to the second FPGA and convolution and pooling are performed. This output image that filtered is stored in the On-Chip Memory provided inside the FPGA. On-Chip Memory is a SRAM based small-capacity memory embedded inside FPGA and it is used as a buffer to temporarily store image-processed data. The result for one frame is DMA-transferred from On-Chip Memory to DDR3 SDRAM via HPS. DMA(Direct Memory Access) transfer is a method that DMAC(DMA Controller) transfers data between main memory and NIC instead of CPU. Uses the interface between the main memory called the DMA channel and peripheral devices without going through the CPU. As a result, the load on the CPU is small and generally high speed is possible. By storing one line of image data in On-Chip Memory. If one line image data is stored, generates an interrupt signal. A transfer instruction is issued from the Nios CPU to the DMA controller and is transferred from the On-Chip Memory to the DDR3 SDRAM via the HPS. The on-chip memory adopts a double buffer method that prepares two. In this way, when one on-chip memory is DMA transfer, processing results can be stored to the other on-chip memory as shown in Fig .7. That is, it can hide the delay.

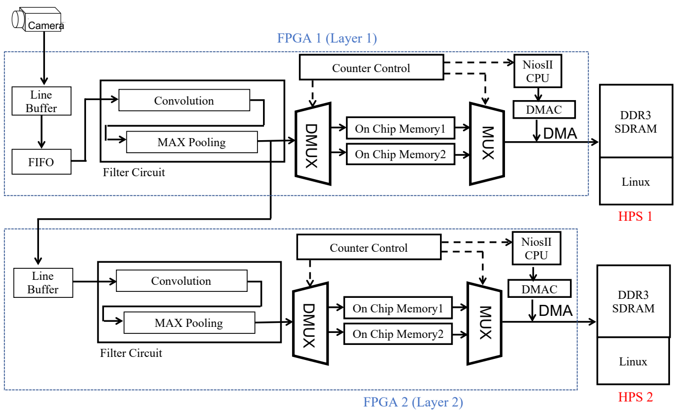

Fig. 6. Circuit diagram of the image processing system designed for this experiment.

C. Camera interface

This section explains the camera module that we used. The camera used was the TRDB-D5M manufactured by Intel Corporation. This CMOS sensor support 2592H x 1994V active pixels. In the case of a full resolution, the captured output is in the RGB Bayer Pattern format and a frame rate of up to 15 frames per second(FPS). These parameters can

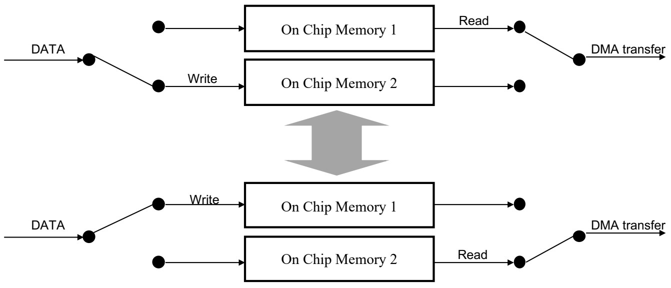

Fig. 7. Double buffer.When one on-chip memory is DMA transfer, processing results can be stored to the other on-chip memory, it can hide the delay

be changed by manipulating the camera registers through I2C communication. Fig .8 shown the block diagram that two parts of the system, the D5M CMOS image sensor, and the camera controller.

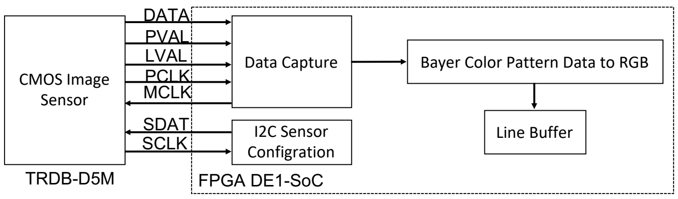

Fig. 8. The camera interface block diagram.

III. Experiment

A. Experimental method

In this section, we describe experiments. In this experiment, we measured the latency of the convolution layer and the MAX pooling layer. It is the time from the end of image data capture for one frame to the time the filter processing and transferred to DDR3SDRAM. The Fig 9 shows the definition of CPU latency. The Fig 10 shows the definition of the latency of FPGA. A logic analyzer was used to measure the latency of the FPGA. The size of the camera image is VGA (640x480), it is a general resolution. Also, it is difficult to compare CPU and FPGA that the camera used in this experiment under the same conditions. Therefore, the CPU environment in this experiment was made to read an image prepared in RAM in advance, and process the image. The software image processing program on the CPU side was created using C / C ++ language and OpenCV library. We used GCC for compiling. The CPU is Intel (R) Core (TM) i5-2400S CPU (2.5GHz). The processing speed of the CPU measured the processing time from the end of image reading to the end of image processing ten times, and the average time was taken as the processing time.

B. Results

Here, we explain the experimental results. Table .III shows the processing results for one frame in each layer. Fig . 12 shows the timing chart measured by the logic analyzer. It shows the end of camera capture and the end of DMA transfer in the first layer. Indicates the time until the FPGA writes

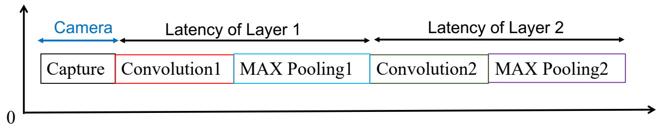

Fig. 9. Definition of CPU latency.

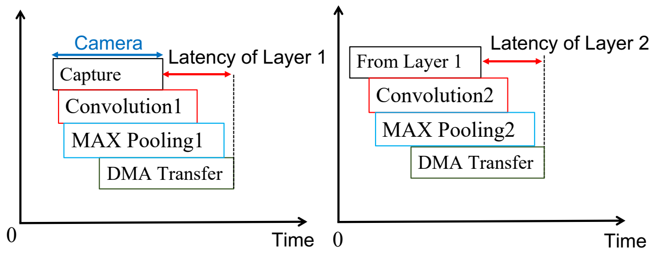

Fig. 10. Definition of FPGA latency.

data to DDR3 SDRAM by DMA transfer. The filter size in this experiment was 3×3 for both layers, and convolution was performed with the filter shown in Fig .11. Fig .13 is image data captured from the camera. Fig .15 shows the processing results of the first-layer FPGA. And Fig .14 shows the simulation results in the C language bu CPU. Comparing these results, it can be confirmed that convolution processing and MAX pooling processing are performed correctly in the hardware circuit.

5	0	-5
10	0	-10
5	0	-5

Fig. 11. Filters that prepared in this experiment.

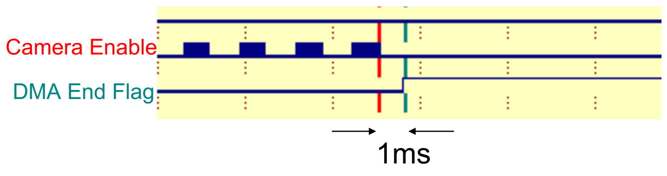

Fig. 12. the timing chart measured by the logic analyzer in first layer.

IV. Discussion

In this section, we verified real-time performance. Here, real-time processing is defined as completing the processing

TABLE I
CPU EXPERIMENTAL CONDITIONS

CPU	Language	Compiler
Intel(R) Core(TM) i5-2400S CPU(2.5GHz)	C/C++	GCC

TABLE II
FPGA EXPERIMENTAL CONDITIONS

FPGA	Language	Environment
Cyclone V	Verilog HDL System Verilog	Quartus ii13.1 Web Edition

before the image transfer of the next frame image is completed. The operation clock is converted to 12.5MHz by the color conversion circuit used in this experiment, so, it is transferred at 12.5MHz to the first layer. Therefore, the transfer is performed at 80 ns per pixel. It takes 1 ms from the end of capturing one frame of image data to the first layer to the end of processing and the end of DMA transfer. This means that the processing is completed before the 12,500 pixels of the next frame image, that is, 20 lines of pixel data are transferred from the camera. The frame size(VGA) is 640×480, so one frame is 480 lines. It is a 4% latency of one frame. It means that processing has been completed by the end of the capture of the next frame image as shown Fig 16. Also, real-time processing is possible even if the number of layers is further increased. The model implementation with multiple convolutional layers would be possible. The FIFO is used between the color conversion circuit and the filter circuit. It operates at 50 MHz after the camera captures, so data is transferred in 20 ns to the second layer. The line buffer structure used in this experiment has a stride width of 1. MAX pooling is realized by skipping one pixel and skipping one line and enabling the processing enable signal. Therefore, the second and subsequent layers operate in the synchronization signal of the MAX pooling circuit of the first layer. In this experiment, only two convolutional layers were implemented. However, if the number of layers is further increased, the delay may increase. It needs to be considered. By inserting FIFO between each layer and separating the clock of each layer, processing speed can be increased.

V. CONCLUSION

In this paper, we implemented two convolutional layers on FPGA. And we discussed the availability of implementation CNN to FPGA. There are for the hardware implementation of convolutional neural networks. Also, we designed layers by software and compared them with CPU processing and FPGA's. About convolution and pooling operations, FPGA processing speed was confirmed faster than CPU. Also, we confirmed that the FPGA was able to perform real-time processing. In this experiment, only the convolution layer was implemented. The size of the convolution used in this experiment is the smallest. The pooling is only the simplest implementation and verification of max pooling. It is necessary to implement more convolutional layers and more pooling layers for the multi-layered CNN. And we have to complicated coefficients required for convolutional neural networks. Also, the full connected layer has never designed yet. Thus, it is necessary to implement a full connected layer. As future work, implement more convolutional layers and evaluate the latency for the camera. It is also necessary to consider the implementation of full connections. The general full connected layer occurs floating-point processing. Also, the implementation of an FPGA requires consideration. In recent years, it has been conducted that an Adaptive Neuro-Fuzzy Inference System (ANFIS) on FPGA [19]–[22]. In future plans, we would like to implement ANFIS and verify its practicality. Also, in this experiment, the camera speed was low. So, it is also necessary to increase the speed of the camera to perform the experiment.

TABLE III
MEASUREMENT RESULTS OF THE PROCESSING SPEED OF FPGA AND CPU

Device	Layer 1[ms]	Layer 2[ms]
CPU	19	4
FPGA	1	2

Fig. 13. Camera data

REFERENCES

[1] Nicholas D Lane, Sourav Bhattacharya, Petko Georgiev, Claudio Forlivesi, and Fahim Kawsar. An early resource characterization of deep learning on wearables, smartphones and internet-of-things devices. In *Proceedings of the 2015 international workshop on internet of things towards applications*, pages 7–12. ACM, 2015.

[2] Marian Verhelst and Bert Moons. Embedded deep neural network processing: Algorithmic and processor techniques bring deep learning to iot and edge devices. *IEEE Solid-State Circuits Magazine*, 9(4):55–65, 2017.

[3] Shuochao Yao, Yiran Zhao, Aston Zhang, Shaohan Hu, Huajie Shao, Chao Zhang, Lu Su, and Tarek Abdelzaher. Deep learning for the internet of things. *Computer*, 51(5):32–41, 2018.

[4] LC Yan, B Yoshua, and H Geoffrey. Deep learning. *nature*, 521(7553):436–444, 2015.

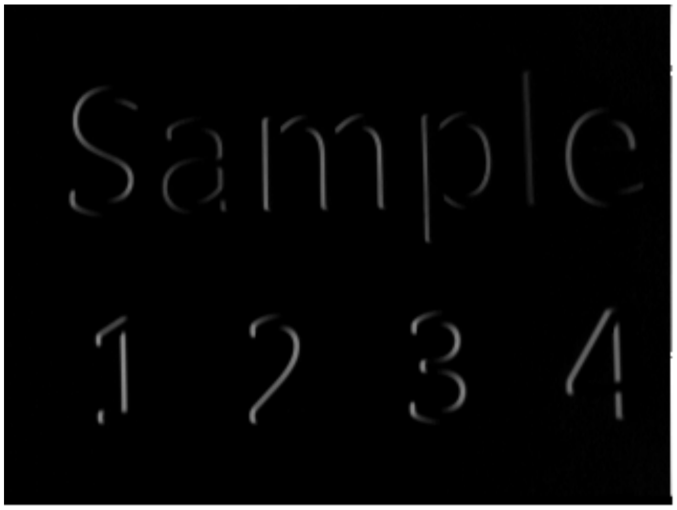

Fig. 14. Results of CPU

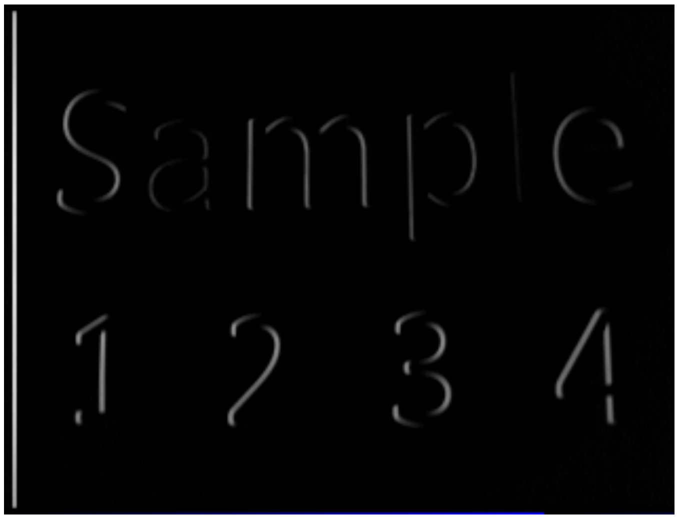

Fig. 15. Results of FPGA

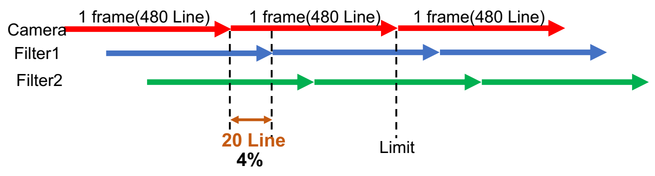

Fig. 16. Verification of real-time performance.

[5] Ammar Ahmad Awan, Khaled Hamidouche, Jahanzeb Maqbool Hashmi, and Dhabaleswar K Panda. S-caffe: Co-designing mpi runtimes and caffe for scalable deep learning on modern gpu clusters. In *Acm Sigplan Notices*, volume 52, pages 193–205. ACM, 2017.

[6] Zhilu Chen, Jing Wang, Haibo He, and Xinming Huang. A fast deep learning system using gpu. In *2014 IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 1552–1555. IEEE, 2014.

[7] Shivank Dhote, Pranav Charjan, Aditya Phansekar, Aniket Hegde, Sangeeta Joshi, and Jonathan Joshi. Using fpga-soc interface for low cost iot based image processing. In *2016 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, pages 1963–1968. IEEE, 2016.

[8] Bryan H Fletcher. Fpga embedded processors. In *Embedded Systems Conference*, page 18, 2005.

[9] Stephen M Trimberger and Jason J Moore. Fpga security: Motivations, features, and applications. *Proceedings of the IEEE*, 102(8):1248–1265, 2014.

[10] Shuichi Asano, Tsutomu Maruyama, and Yoshiki Yamaguchi. Performance comparison of fpga, gpu and cpu in image processing. In *2009 international conference on field programmable logic and applications*, pages 126–131. IEEE, 2009.

[11] Nicolas Gac, StéPhane Mancini, Michel Desvignes, and Dominique Houzet. High speed 3d tomography on cpu, gpu, and fpga. *EURASIP Journal on Embedded systems*, 2008:5, 2008.

[12] Masahiro Shimasaki and Yukinobu Hoshino. Pipeline labeling algorithm with parallel calculation of gravity center on fpga. In *2016 Joint 8th International Conference on Soft Computing and Intelligent Systems (SCIS) and 17th International Symposium on Advanced Intelligent Systems (ISIS)*, pages 86–91. IEEE, 2016.

[13] Markos Papadonikolakis, Christos-Savvas Bouganis, and George Constantinides. Performance comparison of gpu and fpga architectures for the svm training problem. In *2009 International Conference on Field-Programmable Technology*, pages 388–391. IEEE, 2009.

[14] Masahiro Shimasaki and Yukinobu Hoshino. Design and verification of image processing filters on fpga. In *16th International Symposium on Advanced Intelligent Systems*, pages T5e–2, 2015.

[15] Saif N Ismail, Muataz H Salih, and Y Wahab. Design and implementation of embedded vision based tracking system for multiple objects using fpga-soc. *ARPN Journal of Engineering and Applied Sciences*, 13(3):1085–1097, 2018.

[16] Mudit Pradhan et al. *Evaluate the use of Fpga Soc for real time data acquisition and aggregate micro-texture measurement using laser sensors*. PhD thesis, 2016.

[17] Sahand Kashani and René Beuchat. Soc-fpga design guide de1-soc edition.

[18] Alexandre Muñiz Garcia, Roberto Fernández Molanes, Juan J Rodríguez-Andina, and José Fariña. 100fps camera-based ugv localization system using cyclone v fpsoc. In *IECON 2018-44th Annual Conference of the IEEE Industrial Electronics Society*, pages 2789–2794. IEEE, 2018.

[19] José Alejandro Galaviz-Aguilar, Patrick Roblin, José Ricardo Cárdenas-Valdez, Z Emigdio, Leonardo Trujillo, José Cruz Nuñez-Pérez, Oliver Schütze, et al. Comparison of a genetic programming approach with anfis for power amplifier behavioral modeling and fpga implementation. *Soft Computing*, 23(7):2463–2481, 2019.

[20] Alexey Romanov. High-efficient implementation of neural networks and anfis using direct bitstream processing. In *2019 IEEE 60th International Scientific Conference on Power and Electrical Engineering of Riga Technical University (RTUCON)*, pages 1–5. IEEE, 2019.

[21] Hocine Khati, Rabah Mellah, and Hand Talem. Neuro-fuzzy control of a position-position teleoperation system using fpga. In *2019 24th International Conference on Methods and Models in Automation and Robotics (MMAR)*, pages 64–69. IEEE, 2019.

[22] J Gnanavadivel, N Senthil Kumar, ST Christa, and S Muralidharan. Design and implementation of fpga-based high power led lighting system for ships. 2019.