Development of a simple verification environment using FPGA for image processing hardware created by High-Level-Synthesis using TCP/IP

Atsushi Shojima and Akira Yamawaki

Kyushu Institute of Technology, 1-1 Sensui, Tobata, Kitakyushu 804-8550, Japan

*Corresponding Author: yama@ecs.kyutech.ac.jp

Abstract

The development of image processing hardware using FPGA requires various peripherals such as cameras, memory, and displays. Commercially available FPGA boards have various peripherals, but they cannot be used without developing and implementing their own interface circuits. In addition, since the on-board peripherals are different for each FPGA board, new interface circuits must be developed every time when employing different FPGA boards. Therefore, we are developing a general-purpose verification environment that can be imported into commercial FPGA boards, including CPUs, without the need for peripherals on the FPGA board. The feature of the proposed verification environment is that it provides virtual peripherals on a PC. In addition, the proposed verification environment can directly mount hardware modules that are automatically converted from software programs by High-Level Synthesis (HLS). As a result, the design of interface circuits with peripheral devices can be omitted. In this paper, to realize the above verification environment, we developed the software to be executed on the PC and the CPU on the FPGA board, respectively. The communication between the PC and FPGA was initially implemented using serial communication, but in this paper, Linux is installed on the FPGA board's CPU, and TCP/IP communication is implemented between the PC and FPGA. Using these software, we investigated whether it is possible to verify images such as 4K for the image processing hardware created by HLS.

Keywords: peripheral devices, interface circuit, FPGA, image processing hardware, High-Level-Synthesis, TCP/IP.

1. Introduction

In recent years, a variety of embedded devices, such as

TVs and stationary video games, have been converted to 4K. For embedded devices with image processing and limited power supply, high functionality with low power consumption is required. Therefore, it is desirable to implement image processing with high computational load in hardware⁽¹⁻³⁾. To reduce the development burden of image processing hardware, FPGA and High-Level Synthesis (HLS)⁽⁴⁻⁶⁾ are effective because FPGA can rewrite any digital circuits many times, and HLS can automatically turn software description into hardware description.

Hardware verification using FPGA boards requires a variety of peripherals. Commercial FPGA boards have a variety of peripherals. However, in order to use these peripherals, it is necessary to develop interface circuits in a hardware language and implement them in the FPGA. In addition, the peripherals mounted on FPGAs differ among various boards. Therefore, it is necessary to develop a new interface circuits that is suitable for each FPGA board. It is a big burden for developers to support all these peripherals.

Using HLS, it is easy to obtain the hardware description from the software description. However, HLS does not automatically generate peripheral device interfaces that require detailed operations at the clock level. The HLS image processing hardware has its own interface, and therefore the interface has to be developed.

We are developing an environment that enables easy verification of HLS-generated image processing hardware on an arbitrary FPGA board. Our environment consists of an FPGA board, a PC, and management software that runs on the PC and FPGA board. The peripherals to be installed in the target product are realized virtually on the PC; since the peripherals on the FPGA board are not used, the design of the interface circuit can be omitted.

The flow of the proposed verification is as follows: the PC sends an image to the FPGA, and after hardware

Fig. 1. Conventional hardware verification environment.

processing, the result is sent to the PC. In this process, we initially used serial communication⁽⁷⁾. However, as the image size became larger, it took a huge amount of communication time, and we thought it was difficult to put it into practical use. Therefore, in this paper, we installed Linux on the FPGA board's CPU, and developed network programs using TCP/IP⁽⁸⁾ those runs on Linux and on a PC. Using these programs, we conducted experiments to see if the proposed verification environment is feasible.

2. Hardware verification system

2.1 Conventional hardware verification environment

Conventional development of image processing hardware using FPGAs involves purchasing a commercially available FPGA board, implementing the hardware on the FPGA on the board, and verifying it. The input and output to the hardware is done from the peripheral devices mounted on the board. Therefore, it is necessary to design new interface circuits with the peripheral devices. Fig. 1 shows the conventional verification environment.

Board A in Fig. 1 has camera A, display A, microphone A, and SD card A. Therefore, the interface circuits for these peripherals, Camera A_IF, Display A_IF, Microphone A_IF, and SD Card A_IF, have been designed and installed. On the other hand, for board B in Fig. 1, the peripherals are camera B, display B, microphone B, and SD card B. Therefore, new interface circuits (camera B_IF, display B_IF, microphone B_IF, and SD card B_IF) must be designed and implemented for these peripherals. In addition to image processing hardware, designing interface circuits for peripheral devices on each FPGA board is a big burden for developers.

2.2 Proposed hardware verification environment

Fig. 2 shows an overview of our proposed verification system. The proposed verification environment has an

Fig. 2. Proposed hardware verification system.

industry-standard on-chip bus (AXI) as an interface to the image processing hardware in the FPGA. The FPGA board is connected to the PC by using common communication such as UART and LAN. In this paper, we use TCP/IP to communicate between the FPGA board and the PC. The CPU on the FPGA board is equipped with Linux and executes the software for communication between the host PC and the image processing hardware to be verified. In addition, commercially available general-purpose peripherals such as displays are connected to the PC. By connecting these general-purpose peripherals to the PC, there is no need to develop new interfaces even if the infrastructure changes.

2.3 Communication method between FPGA board and host PC

The FPGA board is connected to the PC through general communication such as UART and LAN. Initially, the board was implemented using UART communication, which is one of the serial communication methods.

Serial communication is a communication method that uses one or two transmission lines, and sends and receives data one bit at a time in succession. UART communication is an asynchronous serial communication method that uses two signal lines, one for transmission and the other for data reception. The problem with using serial communication is that the communication time is huge. It takes a lot of time to send a 640 x 640 pixel image file, which is about 1.17 MB

Fig. 3. Software system design.

in size, from the PC to the FPGA board, assuming a communication speed of 115200 bit/s, which takes about 26 seconds. It is expected to take the same amount of time to send the processing results from the FPGA board to the PC. Therefore, it takes about one minute. This means that it is not possible to use huge images such as 720 p, 1080 p, and 4K for verification.

Therefore, in this paper, we used TCP/IP communication between FPGA and PC, which is a standard communication method used in the Internet consisting of two protocols, TCP and IP. The transmission and reception speed is 1000 Mbps, which is faster than serial communication. In order to shorten the communication time, we investigated the implementation of Linux on a CPU with FPGA board, running software with a description of TCP/IP communication.

3. Software development

3.1 Software system design

In order to enable verification using only the FPGA board and the host PC, software that provides virtual peripherals is required. The PC side is controlled by this management software, and the CPU on the board runs Linux, which performs TCP/IP communication with the PC and controls the hardware. In the FPGA, the CPU on the board runs Linux and executes the software for TCP/IP communication with the PC and hardware control. The overview is shown in Fig. 3.

The explanation about the flow of the proposed image processing hardware verification using FPGA is given below. First, the HLS image processing hardware to be verified is mounted on the FPGA, and the host PC and the CPU on the board execute the software described above. Next, the host PC selects an image to be processed and sends it to the FPGA,

which receives the image and performs hardware processing on it. The FPGA then receives the image, performs hardware processing on it, and sends the hardware processed data from the FPGA to the host PC to display the result on the PC. In this way, our proposed verification environment can perform hardware verification without using the peripherals on the FPGA board by executing the software described above. 3.2 and 3.3 below describe the software running on the FPGA board and the host PC.

3.2 Software on FPGA board

On the FPGA board, Linux executes the software that controls the HLS hardware and receives and sends image data to and from the PC.

Fig. 4 shows the relationship between the memory map of Linux and the hardware on the FPGA side. When developing hardware, actual (physical) addresses are assigned to input/output ports such as AXI and MMR, etc. In order to control hardware with software running on Linux, the assigned actual address must be mapped to a virtual address visible to the program. Therefore, the mmap function, a UNIX system call, is used to map the required image size range from the specified file offset into memory. We mapped the addresses necessary for the verification of the image processing hardware, such as to the input and output images, hardware control, and system control.

The communication of the host PC was executed as a server program using `<netinet/in.h>` and other programs. The flow of communication is to first define a socket, specify the IP address and port number of the host, and put the TCP connection in the accepted state. When a connection is received from the client (host PC), the connection is established and image data is sent and received.

The software on the FPGA side operates in correspondence with the GUI of the host PC, and only when

Fig. 4. Virtual and real (physical) memory address.

"Send" is pressed on the GUI, the connection is probabilistically established and the image is received. Then, after hardware processing of the image, the result is sent to the host PC. In all other cases, the system is in standby mode.

3.3 GUI on host PC

In this research, the management software running on the host PC was converted to GUI. The GUI was developed using Processing, an open source project. To design the GUI, we used the <controlP5> library, which is a window control library. The designed GUI is shown in Fig. 5.

In "FREQUENCY", enter the operating frequency of the FPGA. By pressing "CONFIRM", the operating frequency is entered. In "IMAGE FILE", enter the name of the image file to be processed. Press "BMP" to display the specified image. Press "HW TEST" to send the selected image to the FPGA, and after hardware processing in the FPGA, the image and the number of execution clocks are received and displayed. When "TIME" is pressed, the

hardware processing time is calculated based on the operating frequency entered at the beginning and the number of clocks received. The measured processing time can be saved as a CSV file by pressing "CSV MAKE". The "EXIT" button is pressed when the verification is finished to close the window. If the selected image is larger than half the width of the display (width=3840, height=2160), only the selected image is displayed before processing, and only the image after hardware processing is displayed after processing.

The <processing.net> library was used to communicate with the FPGA board. It was designed to work as a client for TCP/IP communication by creating a client socket when the "Send" button is pressed.

4. Experiment and discussion

4.1 Experiment environment

We investigated whether the proposed verification

Fig. 5. GUI design.

Fig. 6. Hardware platform.

environment for image processing hardware using TCP/IP communication works properly. The verification was conducted for multiple size images. The flow of the verification is multiple images are sent from the GUI of the host PC to the FPGA by using TCP/IP communication, and hardware processing is performed. After that, the processing results are sent to the PC and displayed on the GUI. We also investigate whether the communication time can be reduced by changing from serial communication to TCP/IP communication.

Fig. 6 shows the hardware platform developed by Vivado 2018.3 for the actual device verification. This platform has a wrapper called "HW_Wrapper" that covers the HLS hardware. With this wrapper, the developed HLS hardware can be easily replaced. In addition, to measure the hardware execution time, a performance counter running at 100 MHz was included as a memory-mapped register on the FPGA.

In this operational verification, the image processing hardware was created from the development flow called High-Level-Synthesis (HLS) from the image processing software that inverts the R, G, and B values of the original image and implemented on the platform. HLS is a technology that automatically generates a hardware description language by inputting the description of a high-level programming language such as C or C++ into a high-level synthesis tool.

The board used in the experiment was the Zybo Z7 from Digilent. The FPGA installed on the board is ZYNQ (Zynq-7000) from Xilinx.

4.2 Operation check

Fig. 7 shows a GUI screen displaying the processing results of the verified image processing hardware. The image size is 640 x 640 pixels. Fig. 8 and Fig. 9 show the screen

Fig. 7. Hardware processing result (640 x 640).

Fig. 8. Before hardware processing (3000 x 2003).

Fig. 9. Hardware processing result (3000 x 2003).

before and after image processing with an image size of 3000 x 2003 pixels. It can be confirmed that the image specified by the GUI can be inverted in color, indicating that the image processing hardware and communication can be controlled properly. It was also confirmed that if the image to be verified was more than half the width of the display, it could be displayed separately.

Fig. 10 shows a graph of the processing time of the image processing hardware verified in this research. According to the graph, as the image size increases, the processing time also increases proportionally. This is due to the increase in the amount of information processed.

Fig. 10. Hardware processing time.

Fig. 11. Serial communication time.

Fig. 12. TCP/IP communication time.

The graphs in Fig. 11 and Fig. 12 show the time required to send one image from the host PC to the FPGA when using serial communication and TCP/IP communication, respectively. The serial communication took 26.55 s for a 640 x 640 image. For the same size image, it took 0.02 s using TCP/IP. It was found that the communication time could be improved by about 1738 times. Also, with TCP/IP, a 3000 x 2003 image can be communicated in 0.23 s. Therefore, we believe that verification using huge images such as 4K is also possible.

5. Conclusion

The goal of this research is to develop a general-purpose hardware verification environment for FPGAs. In this paper, we developed software to run on the FPGA board and GUI to run on the host PC, and the communication between the PC and FPGA was implemented by TCP/IP communication. Using these software, we verified the image processing hardware for color inversion developed by HLS. As a result, we were able to confirm the results of the image processing hardware on the FPGA on the GUI of the PC. We also showed that huge image processing such as 4K can be verified by implementing it with TCP/IP communication.

Future work includes the verification of moving images and the implementation of dynamic partial reconfiguration. Since we were able to reduce the communication time by implementing TCP/IP communication, we believe that it is possible to verify the video. Also, by implementing dynamic partial reconfiguration, various image processing can be verified by sending hardware circuit data from the PC. By realizing these, it will become an even more versatile verification environment.

References

- (1) Robert Karam, Ruchir Puri, and Swarup Bhunia : “Energy-Efficient Adaptive Hardware Accelerator for Text Mining Application Kernels”, IEEE Transactions on Very Large Scale Integration (VLSI) Systems, Vol. 24, NO. 12, pp. 3526-3537, 2016.
- (2) Shih-Hsiang Lin, Pei-Yin Chen, and Yu-Ning Lin : “Hardware Design of Low-Power High-Throughput Sorting Unit”, IEEE Transactions on Computers, Vol. 66, No. 8, pp. 1383-1395, 2017.
- (3) J. M. Mora-Gutiérrez, C. J. Jiménez-Fernández, and M. Valencia-Barrero : “Multiradix Trivium Implementations for Low-Power IoT Hardware”, IEEE Transactions on Very Large Scale Integration (VLSI) Systems, Vol. 25, No. 12, pp. 3401-3405, 2017.
- (4) Marco Siracusa, Marco Rabozzi, Emanuele Del Sozzo, Marco D. Santambrogio and Lorenzo Di Tucci : “Automated Design Space Exploration and Roofline Analysis for FPGA-Based HLS Applications”, FCCM2019, pp. 314-314, 2019.
- (5) Adrien Prost-Boucle, Olivier Muller and Frederic Rousseau : “A Fast and Autonomous HLS Methodology for Hardware Accelerator Generation under Resource Constraints”, DSD2013, pp. 201-208, 2013.
- (6) Arthur Silitonga, Florian Schade, Guanru Jiang and Juergen Becker : “HLS-Based Performance and Resource Optimization of Cryptographic Modules”, ISPA-IUCC-BDCLOUD-SOCIALCOM-SUSTAINCOM 2018, pp. 1009-1016, 2018.
- (7) Atsushi Shojima and Akira Yamawaki : “Development of a simple verification environment using FPGA of image processing hardware created by High-Level-Synthesis”, AROB 26th 2021, pp. 422-426, 2021.
- (8) Atsushi Nakanishi, Kouki Uchikoshi, Kei Igarashi and Tomoaki Kimura : “Hardware implementation of embedded systems using the TCP/IP communication”, IEICE technical report, Vol. 114, No. 496, pp. 27-32, 2015.