A Generic Verification Environment for Video Processing Hardware

Takayoshi Abe^{a,*}, Hiroyuki Tanabe^a, Akira Yamawaki^a

^aKyushu Institute of Technology, 1-1 Sensui-cho, Tobata-ku, Kitakyushu-shi, Fukuoka, 804-8550, Japan

*Corresponding Author. abe@boss.ecs.kyutech.ac.jp

Abstract

The logic simulation cannot verify the video processing hardware completely. Therefore, the prototyping on an FPGA is performed. However, the effort to develop the interfaces for a camera, an image memory, and a host PC is a large load for developers. Therefore, we provide a generic verification environment standardized the input and output of video processing hardware and can be mounted on any FPGA. By using our environment, it is expected to reduce the load to the developer. This paper shows the system configuration of our environment. Then, through a case study of the actual prototyping of the video processing hardware, we show that our environment can perform the prototyping correctly and occupies small spaces on the FPGAs.

Keywords : FPGA, video processing.

1. Introduction

In recent years, the demand for higher functionality to embedded devices has become increasingly strong. The video processing has become necessary to achieve high performance. For example, to control the speed of the vehicle automatically, high-speed obstacle recognition is required [1]. The digital video camera is required to recognize the human face in high quality image [2]. They must also meet low power consumption, low-cost, high-performance and compact embedded devices. Therefore, the hardware implementation of image processing is required.

However, logic simulation for video processing hardware is difficult to perform enough verification and will need a huge test pattern that makes simulation time long significantly. In addition, it is also impossible with a visual confirmation to the actual operation such as object tracking. Therefore, the prototyping using an FPGA is traditionally performed [3~4]. However, the effort to

Fig.1. Verification Environment

develop the interfaces for a camera, an image memory, and a host PC is a large load for developers.

Therefore, we provide a generic verification environment, and can be mounted on any FPGA, standardized the peripheral circuits. The developer can concentrate designing the video processing hardware. So, our environment can contribute to reducing the burden of the developer. Downloading our generic verification environment into the FPGAs owned by the developers, they have only to put their video processing hardware in our environment.

This paper shows the system configuration of our environment. Then, through a case study of the actual prototyping of the video processing hardware, we show that our environment can perform the prototyping correctly and occupies small spaces on the FPGAs.

2. Generic Verification Environment

2.1 System Overview

Fig. 1 shows the organization of our verification environment. The verification environment provides some standardized interfaces to the user's video processing hardware. For the camera module, there are two interfaces which are for the camera setting and for the video data

transferring captured by the camera. The VRAM interface is for visual confirmation (VRAM, VGA IF). Since there are FPGA boards which have the memory, the memory interface as Wishbone bus [5] is provided. To communicate the host PC, the serial communication interface (SCI) is also provided.

The camera module is set from the host PC via the IF for the camera settings. Images acquired by the camera is sent to the image processing hardware via the camera data IF.

A mailbox (MB) is a register file used for communication with the host PC. The MB becomes the memory-mapped IO of the control/status registers for the video processing hardware in any system-on-chip (SoC). Since MB is dual-port memory, the video processing hardware and a host can read and write the MB at the same time. Between the host PC and the mailbox, the Wishbone bus interface is used.

The video processing hardware with the mailbox has the master Wishbone bus interface to the memory and the slave Wishbone bus interface to the mailbox. So, the video processing hardware with mailbox can be connected easily to any on-chip bus in the SoC of the user product.

Contents of VRAM appears on the display that is connected to the FPGA board. The image size of visual confirmation via a display depends to the capacity of the VRAM.

The communication between the host PC and the verification environment can be established by any USB-Serial Communication converter.

The typical verification flow is shown in Fig.1 as dotted arcs with the number. First, through the serial communication IF as ①, the setting of the camera module is performed by the host PC. Once setting of the camera finishes, the camera sent the captured image to the image processing hardware via the camera data IF ②. The host PC invokes the video processing hardware by storing some parameters to the mailbox ③. Then, the invoked video processing hardware processes the captured image from the camera. If the user like to confirm the processed result visually via the display, the video processing hardware puts the processed data to the VRAM ④. If the video processing hardware has to store the intermediate data into the memory, the video processing hardware can put the processed data into the on-board memory ⑤. The VRAM and the on-board memory can be loaded and stored by the host PC ⑥.

(a) CAM IF

(b) VRAM IF

(c) MB IF

(d) Memory IF(read)

(e) Memory IF(write)

Fig.2. Timing chart

Fig.3. Interface of MB

2.2 Interface Format of Hardware Module

We show timing chart of IF for the camera data, VRAM interface, MB interface and memory IF with reading and writing in Fig.2.

Let us describe the brief operation of each interface. Video processing hardware acquires the camera data via the camera IF. The pixel data reaches with the 1-bit data enabling signal. The hardware module can write the processed data into the VRAM to confirm the result by human eyes. The VRAM can be accessed via a simple and traditional memory interface with the data and address port, and the write enable signal. The MB interface is the same as read and write to the register file. The memory IF uses Wishbone bus. Thus, the hardware module can access any memory by the simple and traditional bus protocol.

For the CAM_IF shown in Fig. 2 (a), the CAM_WE_i is a signal to indicate that the pixel data from the CAM_IF is valid. The CAM_WD_i is the pixel data from the CAM_IF. The CAM_WA_i is the pixel address indicating the X-Y axis of pixel.

For the VRAM_IF shown in Fig. 2 (b), the VRAM_WE_o is a write request signal to the VRAM. The VRAM_WD_o is the data to write to the VRAM. The VRAM_WA_o is the address for the write data.

Fig. 2(c) shows the the mailbox (MB) IF. The HW_WE_o is a write request signal to the MB. The HW_A_o is the register number in the MB. The HW_DO_o is writing data to the MB. HW_DI_i is the read data from the MB.

The memory IF shown in Fig. 2 (d) and Fig. 2 (e) is equal to Wishbone bus IF. Thus, you can embed the hardware module into your SoC easily. The CYC_o indicates that the hardware module initiates the bus cycle.

The ADI_o is the memory address to access. The DAT_o is the data to write to memory. The DAT_i is data read from memory. The ACK_i is a bus completion signal from the memory.

2.3 Mailbox Interface

The MB is a group of 32bit $\times$ 32 dual-port registers for the control and status registers to the hardware module. Function of each register for the hardware module can be arbitrarily determined by the developer.

In our environment, the host PC reads and writes via the serial interface to the MB. Actually, this serial communication is converted to the Wishbone bus IF. That is, the MB becomes a bus slave IF for the hardware module in any SoC. Fig.3 shows the interface of the MB.

The HW_WE_i is a write request signal from the hardware module. The HW_A_i is the register number. The HW_DO_i is the write data from the hardware module and HW_DI_o is the read data to it.

The interface between the MB and serial communication circuit is the Wishbone bus IF as shown in Fig. 3.

3. Experiment and Discussion

3.1 Experiment

To verify the functionality of our verification environment, we have implemented a prototype of the proposed verification environment on the ML401 FPGA board of Xilinx. ML401 has Virtex4 FPGA.

In the experiment, we verified the path of ① to ④ shown in Fig.1. To do this, we have developed a video processing hardware extracting the R, B, or G from the camera data.

Fig. 4 shows the execution snapshots when the test hardware module runs. From the host PC, we changed the color to extract, writing the control register on the MB. As expected, the full color, the blue color, the green color and the red color are displayed on the LCD via the VRAM, according to the value of the control register.

3.2 Functionality Evaluation

To make large room for the user's hardware module, our verification environment implemented into the FPGA is desired to be small as much as possible. Thus, we evaluated the amount of our environment, varying the used FPGAs.

Fig.4. Execution snapshots

The used FPGAs are virtex4, virtex5 and virtex6 of Xilinx. Used resources to the total resources are 9.85%, 5.51%, and 0.08% in the virtex4, virtex5 and virtex6 respectively. That is, we found that our environment is small and can provide the large room to the user's hardware in the FPGAs used.

4. Summary

We have proposed a verification environment generic video processing hardware using FPGA. With this system, so many developers will be able to focus only on the development of HW video processing. And they will be able to reduce the development burden.

Through the preliminary experiment, we have confirmed that our environment can be utilized actually for developing the video processing hardware. Also, we have confirmed that since the hardware overhead of our environment is very small, any developer can employ it without worrying about the resources in the FPGA used.

As future work, we will create an access circuit to external memory SDRAM, SRAM, etc. that was not implemented this yet. Also, we plan to apply more actual application programs to our environment.

Acknowledgement: This work was partially supported by JSPS KAKENHI Grant Number 22700055.

References

- (1) Hanawa Keiji, Sogawa Yoshiyuki : "Development of Stereo Vision System for Automobile Environmental Perception", technology research paper of IEICE, Vol. 101, No. 302, 37-42, 2001
- (2) Toru Nishimura, Hirofumi Hirofumi : "Technologies for Higher Picture Quality of Digital Still Camera", The Journal of The Institute of Image Information and Television Engineers, Vol. 63, No. 6, 720-730, 2009
- (3) Nakajima Masami, Kondo Hiroyuki and Takata Hirokazu and more : "Development of the high-performance microcomputer "M32R" by prototyping method with FPGA", IEICE technology research paper, Vol. 201, No. 168, 61-66, 2002
- (4) Inoue Keita, Weijie Xing, Kimura Shinji : "Asseleration of Prototyping Design Verification Using Circuit Modification" , IPSJ SIG Notes SLDM, 2007, 27, 113-118, 2007
- (5) Zhu Jing, Cao Yunpeng, Yang Cuijun : "Design and FPGA Verification of Dual-Speed Adaptive Ethernet Controller " , WiCOM 2011 7th International Conference , No. 12302760, Print ISBN: 978-1-4244-6250-6, 2011