Implement of Many-Core System Homework 6

109061564 馬婕芸

Introduction

In this homework, we port the Gaussian blur module to RISC-V virtual prototype platform. Also, we port our testbench developed before on SystemC module to RISC-V software. RISC-V virtual prototype is supported by RV32IMAFDC core and implemented in SystemC TLM 2.0. This VP allow flexible prototyping and mixture hardware and software running. Moreover, the surrounding systems consist of sensors, many kinds of peripheral, direct-memory-map controller, and also interrupt controller, etc. Using VPs, we can do early simulation on software development and verification.

Implementation details

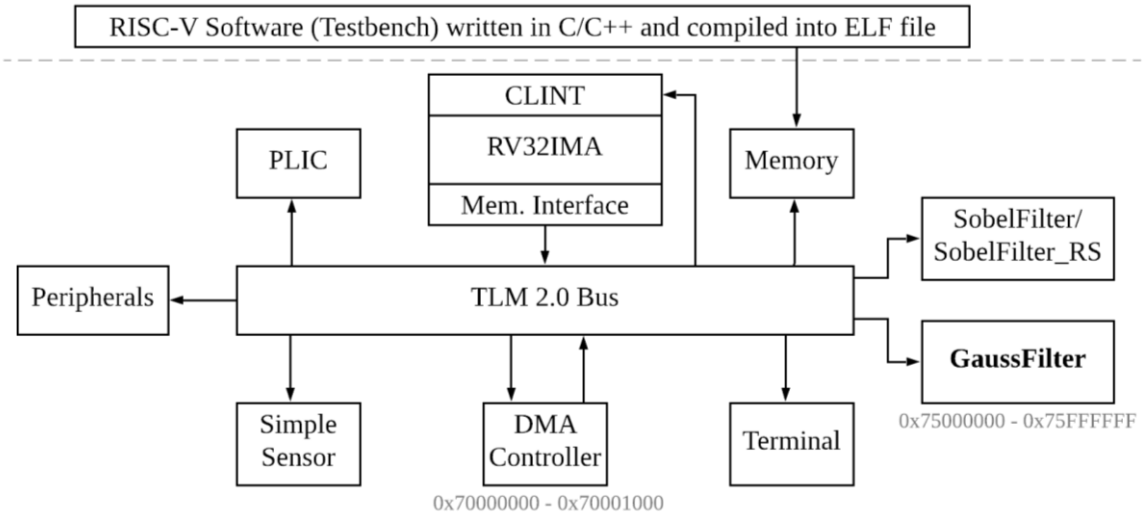
1. Gaussian blur filter is an image filter in digital signal processing. In this homework, we will implement a 3x3 Gaussian blur filter. The kernel is defined as shown below:

$$\text{kernel} = \begin{bmatrix} 1, & 2, & 1 \\ 2, & 4, & 2, \\ 1, & 2, & 1 \end{bmatrix}$$

After 2-D convolution with the original image (lena_std_short.bmp), Gaussian blur process finishes. Also, the 2-D convolution is defined as shown below: since there are three values (r, g, b) in one pixel, the convolution will be done on each value one by one.

$$\text{result}[m,n] = \sum_{j}\sum_{i} \text{original}[i,j] \cdot \text{filter}[m-i, n-j], \qquad 0 \le m,n,i,j \le 2$$

2. Architecture of RISC-V VP platform with Gaussian blur mudule



The VP platform contains main memory, CLINT (core-level interrupt controller), PLIT (platform-level interrupt controller), peripherals (ethernet, flash, etc.), sensors, DMA controller, terminal, and also customized GaussFilter. Inside GaussFilter module, we perform 2-D convolution on 9-pixel array.

With DMA controller, we can directly pass data from memory to GaussFilter and from GaussFilter back to memory. Whole platform are written based on SystemC TLM 2.0, and each components and the bus are connected by TLM 2.0 sockets, and the data transaction for GaussFilter are implemented using blocking transport function, and then use FIFO (first-in-first-out) to pass data to do_filter() function in GaussFilter module. The testbench ported as RISC-V software will be complied as ELF

(executable linkable format) file and load into main memory in the platform.

Results

1. We use lena_std_short.bmp for verifying the functionality of Gaussian filter. Since there is no input/output with files, we print the result of each pixel as below:

```
user@ubuntu:~/ee6470/riscv-vp/sw/gauss$ make sim
riscv32-unknown-elf-g++ -std=c++14 main.cpp -o main -march=rv32ima -mabi=ilp32
~/ee6470/riscv-vp/vp/build/bin/riscv-vp-acc --intercept-syscalls main

        SystemC 2.3.3-Accellera --- May 14 2021 07:00:32
        Copyright (c) 1996-2018 by all Contributors,
        ALL RIGHTS RESERVED
====================================
          Reading from array
====================================
 input_rgb_raw_data_offset      = 54
 width                          = 256
 length                         = 256
 bytes_per_pixel                = 3
====================================
send r: 0, g: 0, b: 0
send r: 0, g: 0, b: 0
send r: 0, g: 0, b: 0
send r: 0, g: 0, b: 0
send r: 122, g: 69, b: 125
send r: 125, g: 76, b: 127
send r: 0, g: 0, b: 0
send r: 156, g: 101, b: 143
send r: 165, g: 102, b: 146
gen  r: 135, g: 81, b: 131
send r: 0, g: 0, b: 0
send r: 0, g: 0, b: 0
send r: 0, g: 0, b: 0
send r: 122, g: 69, b: 125
send r: 125, g: 76, b: 127
send r: 148, g: 91, b: 138
send r: 156, g: 101, b: 143
send r: 165, g: 102, b: 146
send r: 143, g: 89, b: 136
gen  r: 139, g: 84, b: 133
```

Since the pixel are sent from upper left corner, the first pixel in (0, 0) location is (r, g, b) = (122, 69, 125), and the second pixel in (1, 0) location is (125, 76, 127). Considering the zero-padding issue, the 2D convolution process of first output will be as follows:

(0, 0, 0)	(0, 0, 0)	(0, 0, 0)		1	2	1		
(0, 0, 0)	(122, 69, 125)	(125, 76, 127)	*	2	4	2	=	(135, 81, 131)
(0, 0, 0)	(156, 101, 143)	(165, 102, 146)		1	2	1		

The result of the first pixel will be (r, g, b) = (135, 81, 131), and so on.

Conclusion

In this lab, we ported our SystemC module to RISC-VP, which can co-simulate on both hardware and software. With the same software module, we can directly do hardware simulation without changing anything. In this way, we can easily do simulation and verification on our software in the early stage of design.