Implement of Many-Core System Homework 4

109061564 馬婕芸

Introduction

Transaction-Level Modeling (TLM2.0) buses are commonly used between multiple masters and slave IPs. In this homework, we model a bus component using TLM 2.0 buses between testbench (master) and Gaussian blur (slave). Also, we define a memory map and model a random-access memory (RAM) for the whole system. All the functionality and test data are similar to previous labs.

Implementation details

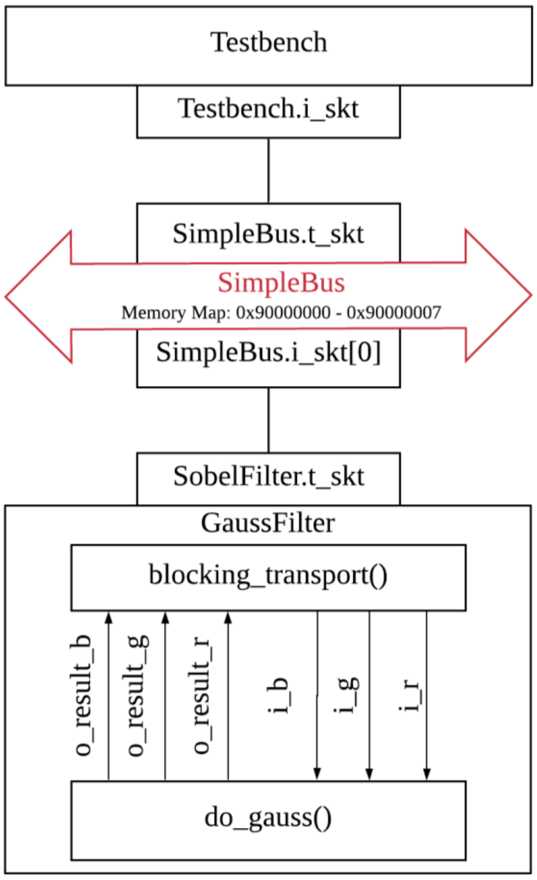
1. Gaussian blur filter is an image filter in digital signal processing. In this homework, we will implement a 3x3 Gaussian blur filter. The kernel is defined as shown below:

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

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.

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

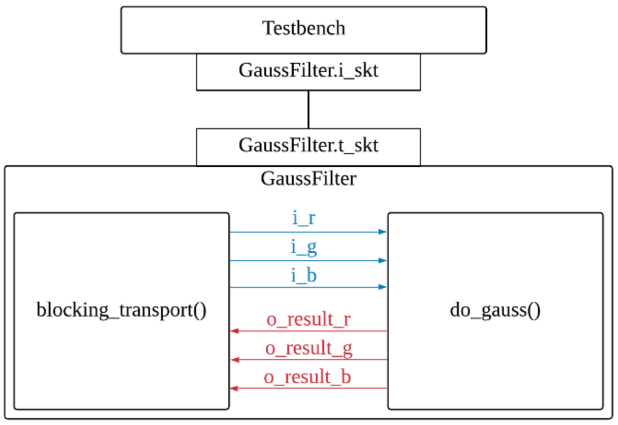
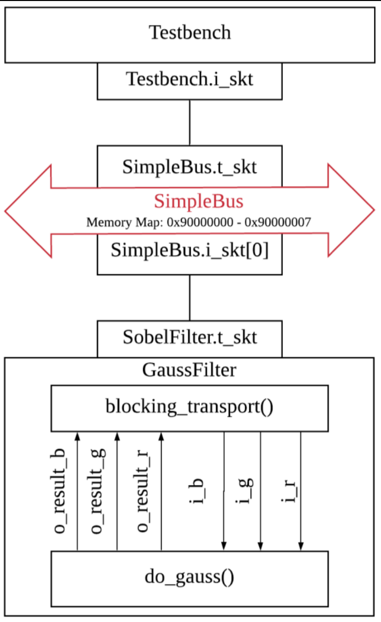
2. Architecture of the whole system.



There is a new component, SimpleBus, which compared to Lab03. Between Testbench, SimpleBus and GaussFilter modules, there are point-to-point TLM 2.0 interfaces, including initiator and target socket. The module Testbench works as TLM initiator, using SystemC module to handle TLM transaction; SimpleBus uses both initiator and target socket to bind sockets in Testbench and GaussFilter; the module GaussFilter acts as TLM target, and additionally using SystemC FIFO to handle data usage between two function. Also, we use self-defined memory map, which starts from 0x9000_0000 with size of 0x0000_0008.

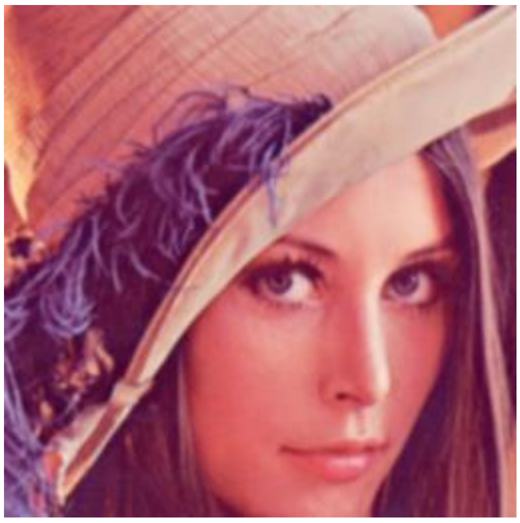
Inside Testbench module, every pixel is sent through SimpleBus and GaussFilter module put it in FIFO and give to do_gauss() function to will do 2D convolution to each pixel and then output into FIFO so as to send back to Testbench through SimpleBus

3. Comparison of architecture with and without TLM 2.0 bus and memory map:
   Since there are only one master module and one slave module in the whole system, it is hard to see the advantage of TLM2.0 bus and memory map. Assume there are more than one slave module, we can use memory map and several sockets in TLM bus to handle data transaction in standard way.

Homework3 architecture	Homework3 architecture
 Testbench GaussFilter.i_skt GaussFilter.t_skt GaussFilter blocking_transport() i_r i_g i_b o_result_r o_result_g o_result_b do_gauss()	 Testbench Testbench.i_skt SimpleBus.t_skt SimpleBus Memory Map: 0x90000000 - 0x90000007 SimpleBus.i_skt[0] SobelFilter.t_skt GaussFilter blocking_transport() o_result_b o_result_g o_result_r i_b i_g i_r do_gauss()

Results

1. lena_std_short.bmp:

Before Gaussian blur	After Gaussian blur
	

Conclusion

In this homework, TLM-2.0 bus and memory map are standard way to handle data transaction between multiple modules. Without changing the functionality and the data structure inside modules, we can address the data transaction issue more clearly using sockets, standard generic payload, memory map and blocking transport.