Implement of Many-Core System Homework 2

109061564 馬婕芸

Introduction

In this homework, FIFO (first-in-first-out) channels and row buffers are used to build a Gaussian blur processor. The functionality of Gaussian blur and system architecture are quite the same as the previous homework. However, the differences are the order of input data defined in testbench and additional buffer used in Gaussian blur processor. The implementation details and simulation results of both will be discussed in the following sections.

Implementation details

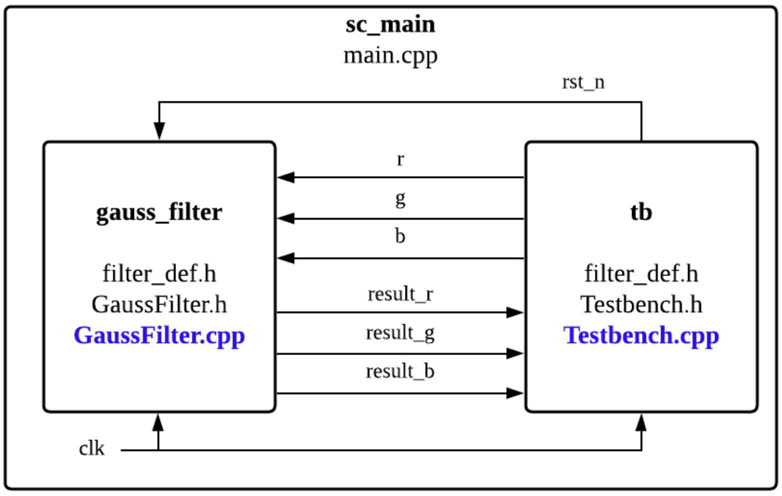
1. Gaussian blur filter is an image filter in digital signal processing. In this homework, we will implement a 3x3 Gaussian blur filter. The kernel is defined as shown below:

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

   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.

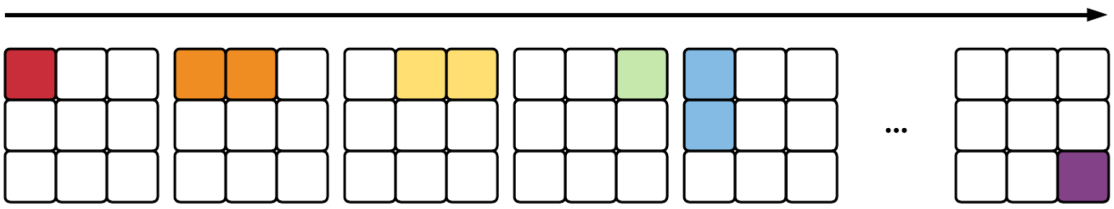
$$\text{result}[m, n] = \sum_{j} \sum_{i} \text{original}[i, j] \cdot \text{filter}[m - i, n - j], \qquad 0 \le m, n, i, j \le 2$$

2. SystemC process: The architecture of the Gaussian blur filter and testbench is shown as follows.

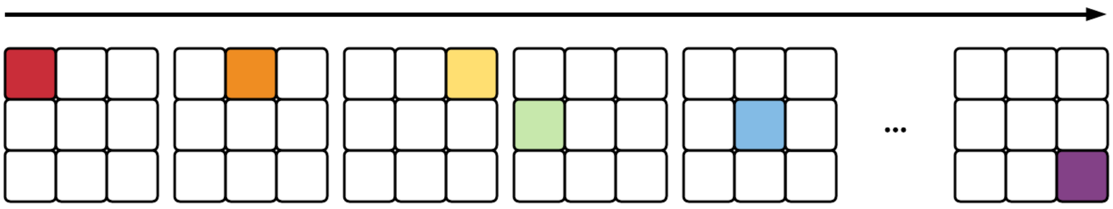


   Similar to the previous homework, both gauss_filter and tb modules are instantiated in sc_main and connected to each other based on FIFO (first in, first out) channels.
   For the file description, the main.cpp define sc_main, two sc_module and the connection between two modules. The filter_def.h defines the dimension of kernel. The GaussFilter.h and Testbench.h define variables and functions used in two modules. Lastly, the GaussFilter.cpp and Testbench.cpp define function behavior in two modules.
   Both GaussFilter.cpp and Testbench.cpp are marked in blue, which means there are new updates for this homework. The details will be discussed later.

3. In Testbench.cpp, input data order and size are changed: (take 3x3 picture with 2x2 kernel for example) In the previous homework, each pixel is sent for four times in this example.



In this homework, every pixel is sent only for one time in this example.

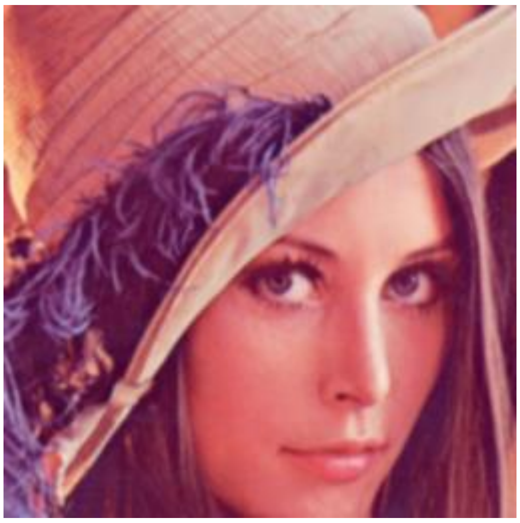


As a result, the number of data access will be reduced.

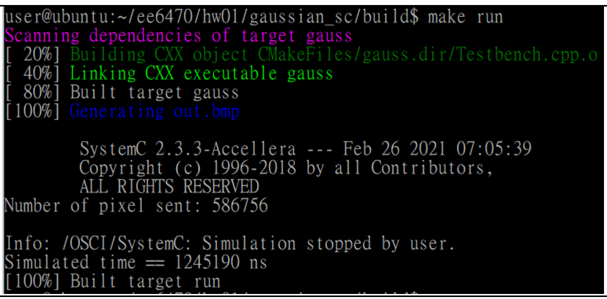
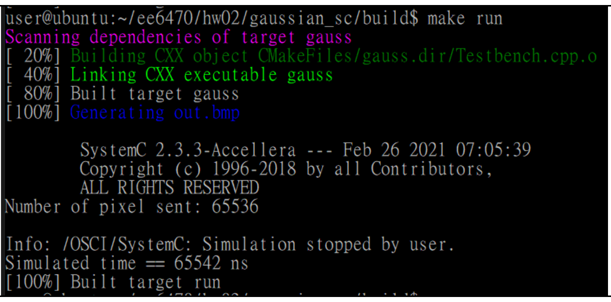
4. In GaussFilter.cpp, an additional buffer in the block: due to the nature of 2D convolution, each data(pixel) is used for about 9 times, as a result, there must be a buffer (256 pixels * 256 pixels * 3 color channels) to save data which is already input. However, the hardware system area may be increased in this way.

Results

1. lena_std_short.bmp:

Before Gaussian blur	After Gaussian blur
	

2. Number of pixels are sent to the Gaussian blur processor:

586,756 pixels are sent.	65,536 pixels are sent.
	

```
user@ubuntu:~/ee6470/hw01/gaussian_sc/build$ make run
Scanning dependencies of target gauss
[ 20%] Building CXX object CMakeFiles/gauss.dir/Testbench.cpp.o
[ 40%] Linking CXX executable gauss
[ 80%] Built target gauss
[100%] Generating out.bmp

        SystemC 2.3.3-Accellera --- Feb 26 2021 07:05:39
        Copyright (c) 1996-2018 by all Contributors,
        ALL RIGHTS RESERVED
Number of pixel sent: 586756

Info: /OSCI/SystemC: Simulation stopped by user.
Simulated time == 1245190 ns
[100%] Built target run
```

```
user@ubuntu:~/ee6470/hw02/gaussian_sc/build$ make run
Scanning dependencies of target gauss
[ 20%] Building CXX object CMakeFiles/gauss.dir/Testbench.cpp.o
[ 40%] Linking CXX executable gauss
[ 80%] Built target gauss
[100%] Generating out.bmp

        SystemC 2.3.3-Accellera --- Feb 26 2021 07:05:39
        Copyright (c) 1996-2018 by all Contributors,
        ALL RIGHTS RESERVED
Number of pixel sent: 65536

Info: /OSCI/SystemC: Simulation stopped by user.
Simulated time == 65542 ns
[100%] Built target run
```

$$\frac{65{,}536}{586{,}756} \approx 0.1117 = 11.17\ \%$$

From the reduction of number of data access, the memory bandwidth in the system can be dramatically expanded.

Discussion

Since a default number of FIFO in SystemC is 16, as a result, when FIFO is full, the output for FIFO will read out all the data in FIFO and wait for the next data. We can also modify the number of FIFO to improve our system simulation.

Conclusion

In this homework, the interface (order of input data, and data buffer) is important to the latency. Since memory (data) access is time-consuming in a real design. To reduce the number of pixel (data) transfer, the memory bandwidth and latency can be reduced. However, power and area may increase. This is also a key trade-off for a real design.