Implement of Many-Core System Homework 7

109061564 馬婕芸

**Introduction**

In this homework, we port the Gaussian blur module to 2-core RISC-V virtual prototype platform. Also, since DMA/Memory is an exclusive resource, we need to use mutex lock to constrain the access, and use barrier to synchronize the processing between two cores.

**Implementation details**

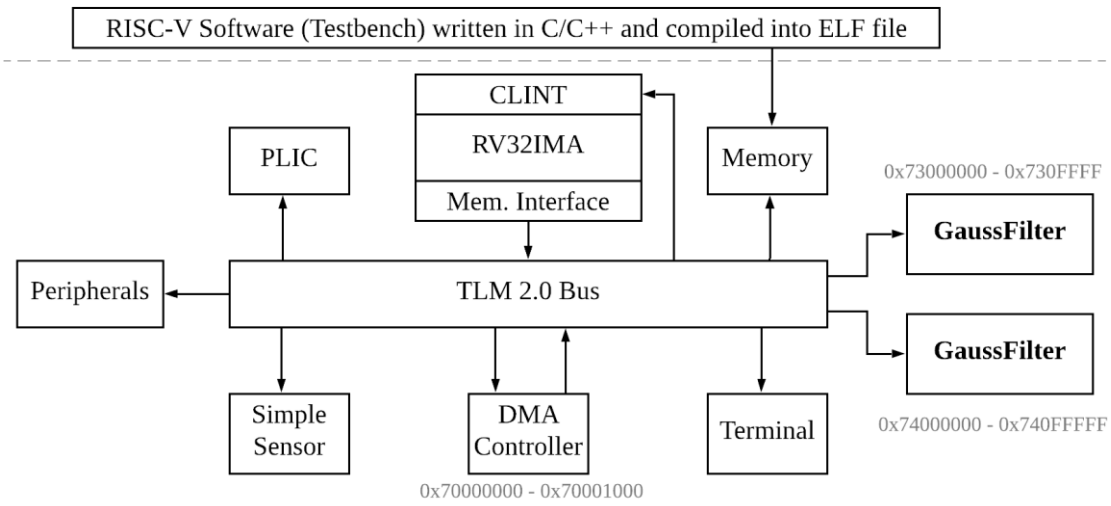
1. Gaussian blur filter is an image filter in digital signal processing. In this homework, we will implement a 3x3 Gaussian blur filter. The kernel is defined as shown below:

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

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

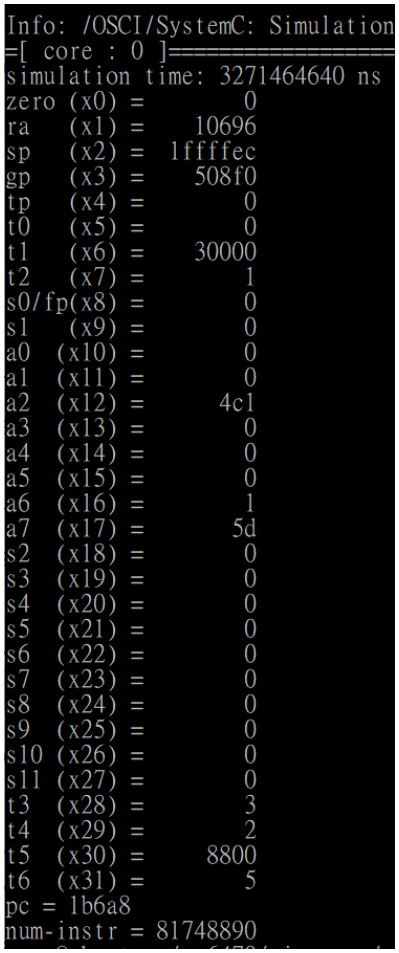


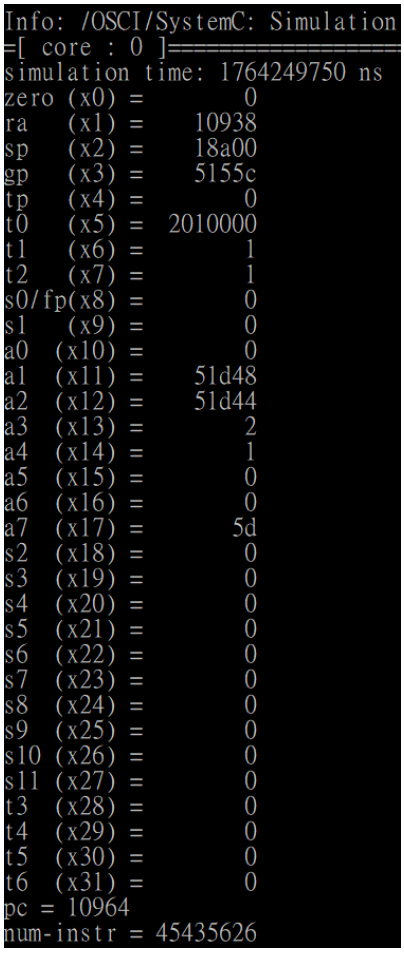

The VP platform contains main memory, CLINT (core-level interrupt controller), PLIT (platform-level interrupt controller), peripherals (ethernet, flash, etc.), sensors, DMA controller, terminal, and also customized GaussFilter. Inside GaussFilter module, we perform 2-D convolution on 9-pixel array.

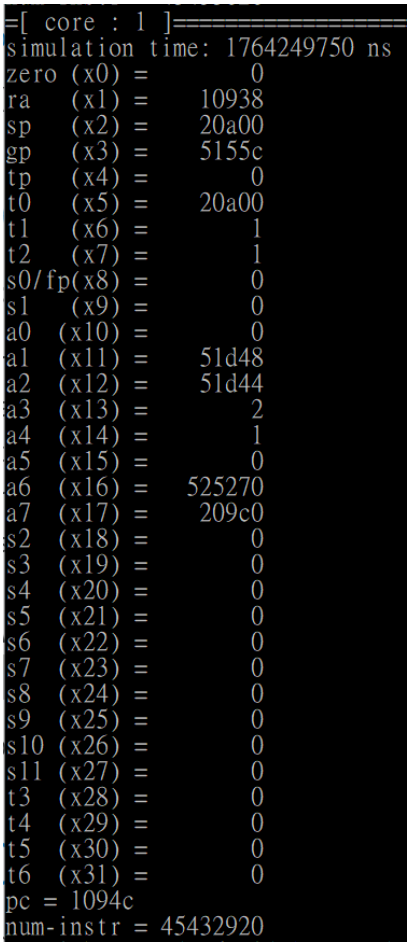
With DMA controller, we can directly pass data from memory to GaussFilter and from GaussFilter back to memory. Whole platform are written based on SystemC TLM 2.0, and each components and the bus are connected by TLM 2.0 sockets, and the data transaction for GaussFilter are implemented using blocking 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. Comparison between single-core and 2-core simulation is shown below.

<table>
<tr><th>Single-core</th><th colspan="2">2-core</th></tr>
<tr>
<td>


</td>
<td>


</td>
<td>


</td>
</tr>
<tr><td>3271464640 ns</td><td colspan="2">1764249750 ns</td></tr>
<tr><td>81748890 instructions</td><td colspan="2">45432920 instructions</td></tr>
</table>

In 2-core simulation, we optimize the utilization of DMA bandwidth. Though there will be more computing unit in multi-core system, the simulation time is incredibly reduced using only one DMA.

**Conclusion**

In this lab, we ported our SystemC module to multi-core 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.