Homework Assignment 03

Convolution Engine for Image Processing

Design a simple convolution engine for the edge detection or other image processing applications. Please refer to Wikipedia for more details. In addition, you can also learn some more Verilog coding and also software programming skills:

1. Verilog coding skill with file IO and array;
2. Verilog design style and hand-shaking with timing consideration;
3. Verilog design style with SRAM block;
4. The usage of Makefile.
5. (Optional: use the revision control tool.)

Note:

a. For Makefile, you may refer to online resources such as

   鳥哥的 Linux 私房菜 http://linux.vbird.org/linux_basic/1010index.php

b. Source code is the best document!
c. Starting from this homework assignment, I try to write the problem description in a way that may encourage you to raise a discussion.

Problem Description and Design Specification

We are going to design a 5x5 convolution filter (kernel) of edge detection. The input is a gray-scale 256x256 image. The filter coefficients should be able to change via the test stimulus.

1. Please refer to the slides (hw03-2.pdf) for the further details.
2. The design template can be found in `hw03_vX.tgz`, where vX denotes the version (which can be v1, v2, etc., in case we will update it later). Always check for the latest version. Under Linux/Unix, use the follow command to extract the tar-gzipped file:

```
$ tar zxvf hw03_vX.tgz
```

3. Look into `load_bmp.v`, which is the main testbench. Learn what the testbench does to load the image, convert the pixels into the gray-scale ones, and store into the SRAM by using a Verilog trick.
4. Study the IO signals, block diagram and timing.
5. **HW03a**
   I. Design the engine in `hw03a.v` with only one multiplier.
   II. For a block of storage, the SRAM is usually much smaller than flip-flop

implementation. You may try to use SRAM instances in your design when you need a buffer for intermediate results. Adopt `sram.v` to create your own SRAM instances if necessary. However, make sure that the SRAM instances come with data widths of 8, 16, 32, etc. Their address depth can only be 2^n, where n is a positive integer.

6. **HW03b**
   I. How do you improve the latency by adding more multipliers? Can you achieve the 2x speedup by using two multipliers? Can you achieve the 4x speedup by using even more multipliers? Design an improved engine with more than 2x speedup as compared with HW03a.
   II. You have the degree of freedom to alter the original architecture (filter, sram65535x8, and so on). For example, you may use two separate, smaller SRAM instances to store the data, each of 32768x8. In this case, you should modify the testbench as well.
7. **HW03c**
   I. Use dual-port SRAM instance to design an alternative engine with at least 2x speedup as compared with HW03a. Is it easier with dual-port SRAM? Or is it the same?
   II. You can find a Verilog template of dual-port SRAM. You should modify and verify it when necessary.
8. Learn how to use Makefile to integrate the simulation.
9. Complete and verify your RTL/gate-level designs with the golden responses. The testbench will also convert your outputs in the SRAM to a BMP file. You may look into the details.

   Note:
   I. There are alternative behavior descriptions of single-port/dual-port SRAM instances in the Verilog templates. You should verify all the different versions before using them.
   II. Please leave the SRAM instances un-synthesized.
   III. Instead, estimate the area of SRAM:
      a. The area of a single-port 8192x8 SRAM is 347,200 um^2.
      b. The area of a dual-port 8192x8 SRAM is 766,600 um^2.
      c. As the reference, the smallest NAND2 is 5.09 um^2; a DFF with asynchronous active-low reset is 32.25 um^2, in our cell library.
10. Discuss the number of flip-flops, number of SRAM blocks and their sizes in your designs.
11. What is the speedup between your different designs?
12. Write a report to summarize all the discussions.

Note

➔ This is a graduate-school-level homework assignment. Make reasonable assumptions or raise a discussion if there is any detail needed to be more specific.

➔ For each assignment, you are requested to write a report with your name and student ID. The topics should include, but are not limited to, the following items:

a. The design concept with figure and description;
b. Simulation result with explanation, including the discussion about problems you encounter, and the way you solve them (or not);
c. A brief summary, including suggestions for us (or this course).

DO NOT put the entire source code or boring waveform screenshots into the report without proper explanation!

➔ Submit the source files and the electrical report based on TA's instructions.

➔ The source files may include the followings (I assume that you can know what these files are for from their naming) for **HW=hw03a** and **HW=hw03b** and **HW=hw03c**:

- RTL designs: ${HW}.v
- Gate-level implementation: ${HW}_syn.v
- SDF file from the Design Compiler: ${HW}_syn.sdf
- FSDB waveforms: ${HW}.fsdb, ${HW}_syn.fsdb
- Document: README.txt to briefly describe how to perform all the simulations
- (Optional) Makefile (if you use your own Makefile)
- (Optional) ${HW}.f (if you have your own one)
- (Optional) Other files or documents for your designs and simulations to fulfill the requirement of this assignment. Put the details in your report and README.txt
- The PDF report: ${HW}_YourStudentID.pdf

DO NOT hand in compressed files.

Assignment due on 4/06/2019, 23:30

1. Submit your source designs and the report.
   - No overdue is allowed.
   - Submission rules will be strictly enforced.