ECE3270 Digital System Design
Lab 6: OpenCL Image Smoothing Design

Lab Overview: The purpose of this project is to further understanding of FPGA computing and to introduce OpenCL. Students will be required to write C code and an OpenCL kernel. You will complete a full report using the LaTeX template and include discussion about speedup. You will submit all edited files (vhd, xml, cpp, cl, c) to the assign server as Assignment 6.

## Image Smoothing

Image smoothing is commonly completed by computing the convolution of a kernel and an image. Convolution can be simplified into the dot product of the kernel and the pixels the kernel covers. Shown below is a 1D convolution. You will need to extend this idea into 2D data (specifically PPM images). You will line the center pixel of your kernel up with the pixel being computed, compute the convolution ("dot product"), and save the result in the same location of the output image.

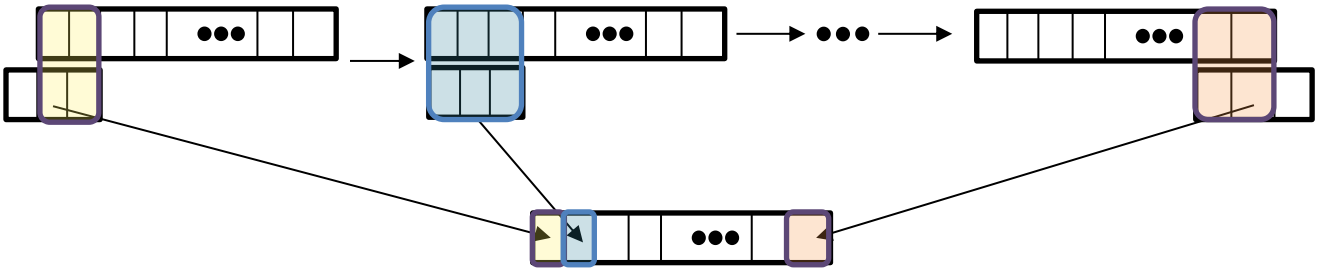

**Figure 1: Diagram demonstrating convolution of a large vector and a vector of 3 elements. In the first and last stages, only the overlapping elements are multiplied and added (you can ignore the hanging element in the small vector or pretend it is multiplied by 0). This smaller vector is usually called the "kernel" and often the sum of all elements of the kernel equals 1 to compute weighted averages. The highlighted elements in each stage are elements that take place in the dot product.**

For this assignment, you will be required to create image smoothing in C and OpenCL. It is best that you start on the C program as soon as possible and verify your output. You can use this as comparison with the OpenCL to verify correct implementation, plus it is good practice to verify you understand the smoothing process.

## C Program Requirements

The C program has the following requirements.

1) You should use a Gaussian Blur filter, which you can find here. Note, other filters can work as long as each number is divided by the sum of each number of the filter. Save the smoothed image. You may work with grayscale images if you choose.

2) You must time the smoothing algorithm. You **WILL NOT** time reading/writing the images, and if you choose to do so, keep this result separate from the calculations. Make sure to print these values for comparison with the OpenCL implementation. See the answer in the following link for timing details.
http://stackoverflow.com/questions/5248915/execution-time-of-c-program

3) Keep in mind that edges have less neighbors, so make sure to properly handle the corners and edges of your image. Inappropriately handling edge cases can cause invalid output.

4) You must test on at least 5 different image sizes. If you use an external source to read/write PPM images, please reference this in your report and in comments in your code! Failure to do so is plagiarism.

OpenCL Requirements

The OpenCL program has the following requirements.

1) Complete Smoothing! (Refer to C code). Note if the C code is complete, little modification is needed to make the OpenCL code work correctly.
2) You can utilize the emulation to verify your result, but you cannot use this as your final timing results.
3) Modify the device Makefile so that your executable is named appropriately and **DOES NOT** link to a library anymore. You will not be writing any VHDL code for this lab!
    a. This means remove all references to lib, the lib object, and the "-l $(PROJECT)_rtl.aoclib -L ." section of the default and optm targets.
4) Modify the folder structure and all filenames to accurately reflect the name of this lab (executables, files, or functions named bitpair, fuel, safe, etc. are not acceptable).
5) You will also need to go into the host Makefile and change the project name.
6) Compile for use on the DE1-SoC and verify performance on hardware.

Report Requirements

1) Discuss the speedup gained when running on hardware. Remember that speedup is defined as:

$$Speedup = \frac{Time_C}{Time_{OpenCL}}$$

You need to make sure your times ONLY record algorithm execution time. If you include the time for I/O operations, your results will be skewed toward the system with quicker file transfers and OpenCL overhead will add to this skew. It is also suggested you time your C algorithm on the ARM CPU instead of on a lab machine to make sure your comparisons are more valid!

**Extra Credit:** Implement one of the optimizations discussed in class from the Best Practices Guide. To receive full credit, you must implement the optimization and explain why this improves speedup. You must also compare this to the unoptimized version to show that it actually improved your algorithm. Your explanation must include discussion as to why you chose the specific optimization and steps taken to verify the optimization is fine tuned.

| Rubric | |
|---|---|
| **Report**<br><br>- Proper format<br>- All sections included<br>- Valid images where applicable<br>- Proper grammar, punctuation, and spelling | **50%**<br><br>- 10%<br>- 30%<br>- 5%<br>- 5% |
| **Demo**<br><br>- Live Demonstration<br>- Includes working code and answering questions from the TA<br>- Comments<br>  o Thoughtful comments, not English translations of code | **40%**<br><br>- 30%<br>- 5%<br>- 5% |
| **Proper Assign Server Code Submission** | **10%** |