ECE3270 Digital System Design
Lab 6: OpenCL Image Dilation 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 Dilation

Image dilation is taking similar portions of an image and "growing" them, in simplest terms.

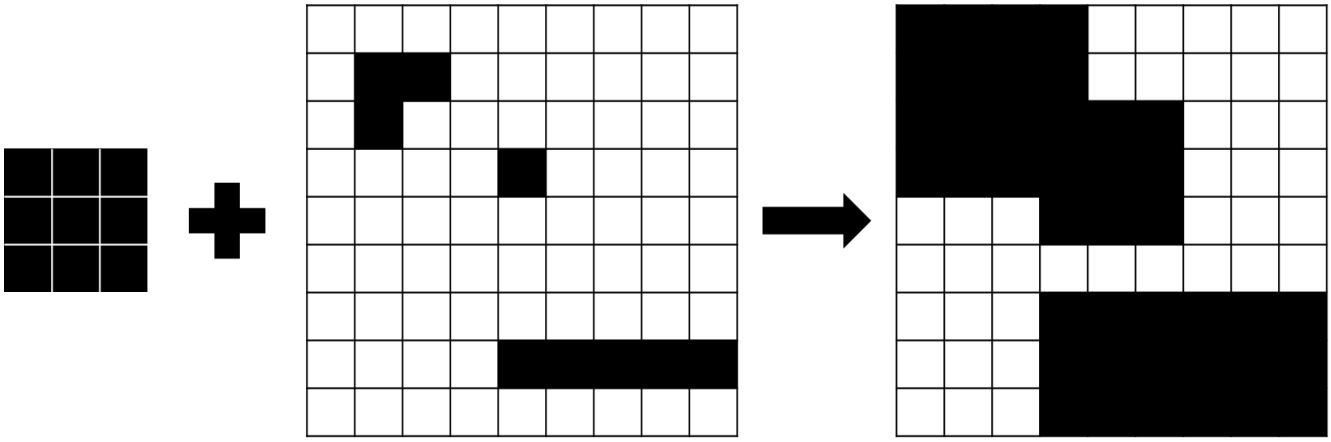

Figure 1: Diagram demonstrating dilation using a 3x3 dilation kernel. For each black pixel, the 3x3 kernel is centered on it. Each pixel the 3x3 kernel is centered on is then colored black in the output image. This causes shapes to "grow" or dilate. You can think of the kernel as a stamp that is centered on every "on" pixel, then it is stamped down to color extra parts of the image.

For this assignment, you will be required to create image dilation 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 dilation process.

C Program Requirements

The C program has the following requirements.

1) You will first need to threshold a PPM image, which means to convert an image to binary (black and white). To make this simple, a pixel will become white (on) when any of the RGB values is **ABOVE** 185. Else the pixel is black (off). You may work with grayscale images if you choose.

2) Next, use a **5x5 Square** to turn on surrounding pixels if the pixel you are checking is also on. Save the dilated image. Please note the example above shows black as "on" for ease of demonstration. You will be growing white areas in your design.

3) You must time the dilation 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

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 Dilation! (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) You can write this using either 1 or 2 kernels, the choice is yours.
4) 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.
5) 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).
6) You will also need to go into the host Makefile and change the project name.
7) 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. 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 - Proper format - All sections included - Valid images where applicable - Proper grammar, punctuation, and spelling	**50%** - 10% - 30% - 5% - 5%
Demo - Live Demonstration - Includes working code and answering questions from the TA - Comments o Thoughtful comments, not English translations of code	**40%** - 30% - 5% - 5%
Proper Assign Server Code Submission	**10%**