ECE3270 Digital System Design
Lab 5: OpenCL Convolution Design

Lab Overview: The purpose of this project is to further understanding of FPGA computing and to introduce OpenCL and different optimizations. 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 and optimizations used. You will submit all edited files (vhd, xml, cpp, cl, c) to the assign server as Assignment 5. When submitting your report, be certain this is the **FINAL VERSION** of your report. Multiple submissions **ARE NOT ALLOWED** and will **NO LONGER** be accepted for submitting the wrong version.

1D Convolution

1D Convolution can be thought of as the process of taking iterative dot products of two vectors at different offsets from one another. This process is often used to "smooth" a signal by averaging each pixel with neighbors from both directions by using weighted values as "kernel" elements. See Fig. 1 for further information.

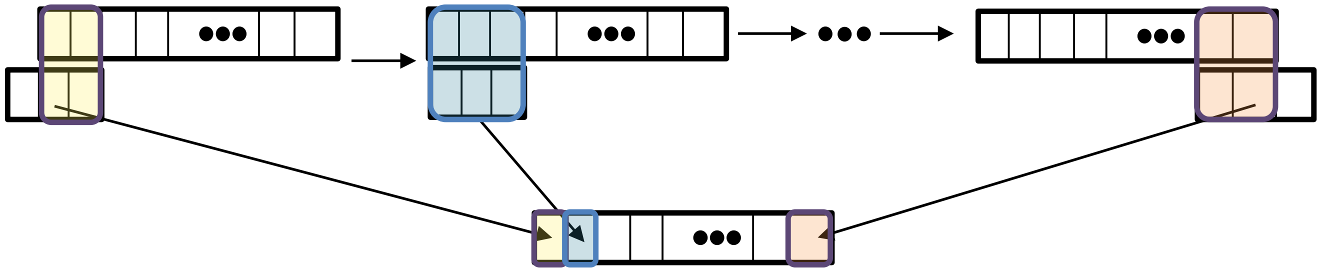

Figure 1: Diagram demonstrating convolution of a large vector and a vector of 3 elements. In the first and last stages, only the overlapping elements are multiplied and added (you can ignore the hanging element in the small vector or pretend it is multiplied by 0). This smaller vector is usually called 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 convolution 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 convolution process.

C Program Requirements

The C program has the following requirements.

1) You should hard code a kernel defined as: "float kernel[3] = {1.0/3.0, 1.0/2.0, 1.0/6.0};". You may use a simpler kernel for testing, but this should be your definition for the final code.

2) You must time the convolution algorithm. You **WILL NOT** time reading/writing the vectors, 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 ends of your vector. Inappropriately handling edge cases can cause invalid output.

OpenCL Requirements

The OpenCL program has the following requirements.

1) Complete Convolution! (Refer to C code).
2) You can utilize the emulation to verify result, but you cannot use this as your final timing results.
3) Compile for use on the DE1-SoC and verify performance on hardware.
4) Complete **AT LEAST** one OpenCL optimization found in the AOCL Best Practices Guide Document.
5) 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.
6) Modify the folder structure and all filenames to accurately reflect the name of this lab (executables, files, or functions named bitpair or traffic are not acceptable).
    a. You will also need to go into the host Makefile and change the project name.

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.

2) Explain the optimization you used in your code and why you chose this optimization. Provide details as to how this speeds up execution time.