Lab#3 Tiled Convolution

IMPORTANT:

- **Write collaborators name, if any.**
- **Test with different input sizes before submitting.**
- **Only submit ONE zip file (`FirstName_LastName_Lab3.zip`) that includes `report.pdf` and `other source files`.**

Due:

Oct. 31 11:59:59pm, 2023

If you are late (even by a minute – or heaven forbid, less than a minute late), you will receive 50% of your earned points for the designated grade as long as the assignment is submitted by 11:59pm the following day, based on the due date listed on the above and confirmed by the instructor. If you are more than 24 hours late, you will receive a zero for the assignment and your assignment will not be graded at all.

Goal:

The purpose of this lab is to get you more familiar with shared memory tiling techniques, handling complex boundary conditions, and using constant memory.

Instructions

1. Download and extract lab3-conv.zip from eLC. The folder should contain 5 files: **Makefile, conv_kernel.cu, conv_main.cu, support.cu, support.h**. Carefully study the code and ask questions on eLC if you have any. Run the `make` command to compile your files.
2. If you have any questions about connecting to a remote machine, please refer to the Lab0-setup manual.
3. Coding

   a. Edit **conv_main.cu** to add the constant memory copy and set the block and grid dimensions correctly. Edit **conv_kernel.cu** to implement the shared memory tiled convolution. To handle halo cells, treat them as having a value of zero. Check main() for a description of the modes.

4. Look for the statement "INSERT YOUR CODE HERE". Test different thread block sizes and `choose the one that works best for your case`. The performance will be an important factor for your final grade.
5. Your program should be able to accept valid matrix sizes as arguments. We also provide a `round` argument to help calculate an average latency and reduce fluctuations. Check the code carefully before starting.

Testing

After you are done with coding, answer the following questions and submit the `report (PDF)` and `5 source files` (all in one zip file) in the eLC:

1. Which CUDA machines have you tested?
2. Can your program compile properly?
3. Is your program working correctly and did it pass the test?
4. What is the floating-point computation rate for the GPU kernel in this application? How does it scale with the size of the input image? To answer this question, try multiple sized inputs and calculate the rate for each using the timing measurements provided in the code. Make sure to justify your choice of input sizes.
5. What percentage of time is spent as overhead for using the GPU? Consider as overhead: device memory allocation time and memory copy time to and from the device. Do not include problem setup time or result verification time in your calculations of overhead or total execution time. Try this with multiple input sizes and explain how the overhead scales with the size of your input?

Grading

1. Your submission will be graded based on the report and code (including but not limited to code quality, correctness, performance, and readability).

2. **Bonus (10%) will be awarded for the best performance in the lecturer's testing samples.**
    - One bonus will be given to graduate student(s) and another bonus will be given to undergraduate student(s).