University of Arkansas
Department of Computer Science & Computer Engineering
CSCE 4643/5693 – GPU Programming

Lab 4: Tiled 2-D Convolution

Due Date: March 2, 2020 (Monday) at midnight 11:59pm
Required for all students

1. Objective

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

2. Procedure

Step 1: Download the lab 4 material from blackboard to your home folder at the Karpinski computer cluster. Unzip it.

```
unzip lab4-convolution.assignment.zip
```

Step 2: Edit `main.cu` to add the constant memory copy and set the block and grid dimensions correctly. Edit `kernel.cu` to implement the shared memory tiled convolution. To handle halo cells, treat them as having a value of zero.

Step 3: Compile and test your code.

```
make
./convolution                  # Uses the default input image size
./convolution m                # Uses a square m x m input image
./convolution m n              # Uses an (m x n) input image matrix
```

Step 4: Answer the following questions in a new file named `answers.txt`:

1. 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 input images of different sizes and calculate the rate for each case using the timing measurements provided in the code. Make sure to justify your choice of input sizes.
2. 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?

Step 5: Submit your assignment. You should only submit the following files:

- `main.cu`
- `kernel.cu`
- `answers.txt`

Compress the files and name them as following:

```
tar -cvf lab4_<your lastname>.tar main.cu kernel.cu answers.txt
```

3. Grading:

Your submission will be graded based on the following criteria.

- Functionality/knowledge: 80 points
  - Correct code and output results
  - Correct usage of shared memory in the kernel to hide global memory access latencies
  - Correct handling of boundary cases
  - Check return values of CUDA APIs
- Answers to questions: 20 points
  - Correct answer to questions in Step 4
  - Sufficient work is shown
  - Neatness and clarity