University of Arkansas
Department of Computer Science & Computer Engineering
CSCE4643/5693 – GPU Programming

Programming Assignment 7: Reduction

Due Date: October 29 (Sunday) at midnight 11:59 pm
Required for ALL students

1. Objective

The objective of this programming assignment is to get you familiar with the parallel reduction algorithm.

2. Procedure

Step 1: Download the programming assignment 7 materials from blackboard to your home folder at Karpinski computer cluster. Unzip it.

```
unzip p7-reduction.assignment.zip
```

Step 2: Edit `kernel.cu` to implement the device kernel code for the parallel reduction algorithm, assuming that an input array of any size can be handled by your kernel. Each thread block will deal with a sub-array of 2×BLOCK_SIZE. You only have to produce the partial sum of each thread block for this part. We will sum up the partial results to the final result on the host.

Step 3: Compile and test your code.

```
make
./reduction             # Uses the default input size
./reduction <m>         # Uses an input with size m
```

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

- How many times does a single thread block synchronize to reduce its portion of the array to a single value?
- What is the minimum, maximum, and average number of "real" operations that a thread will perform? "Real" operations are those that directly contribute to the final reduction value.

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

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

Compress the files and name them after your last name like the following:

```
tar -cvf p7_<your last name>.tar kernel.cu answers.txt
```

If the last name is alan, the tar file name is p7_alan.tar

Submit the tar file in blackboard.

3. Grading:

Your submission will be graded based on the following criteria.

- Functionality/knowledge: 90 points
  - Correct code and output results (20 points)
  - Correct usage of shared memory in the kernel to hide global memory access latency (20 points)
  - Avoiding control flow divergences and shard memory bank conflicts (20 points)
  - Correct handling of boundary cases (10 points)
  - Checking return values of CUDA APIs (10 points)
  - Schedule the correct number of thread blocks (10 points)
- Answers to questions: 10 points
  - Correct answer to questions in Step 4
  - Sufficient work is shown
  - Neatness and clarity