LSU EE 7700-2 Homework 4 Due: 26 February 2013

Follow the instructions on the class procedures page, `http://www.ece.lsu.edu/gp/proc.html` *for account setup and homework, substituting* `hw4` *for* `hw1` *where appropriate. Also, the file to edit is* `hw4.cu`*, not* `hw4.cc`*. The assignment code is the same as the vertex transformation code used in class.*

Problem 1: Compile and run the Homework 4 assignment code. This code is similar to that used in Homework 3 except rather than finding the minimum vertex magnitude, it finds the sum of all the squared magnitudes. This makes the code simpler, and also makes the error-checking code more sensitive to errors (which is a good thing).

The first command-line option specifies the method of finding the global sum (a sum is an example of a reduction). The code currently works with methods 0 to 3, in the next problem method 4 will be implemented.

The program will accept three other command line arguments, those match the previous assignment. The first argument is the reduction method to use, the second is the number of blocks to launch, the thrid is the number of threads per block, the fourth is the size of the array in MiB. If the program is run with arguments r x y z it will launch with xy threads and an array with $\lfloor z2^{20} \rfloor$ elements, and use reduction method r to find a global sum.

As with the prior assignment, the timing is of the GPU portion only. When comparing timings, note that with reduction method 0 the GPU does not perform any reduction, so it will appear to be faster than the other methods (since the CPU time isn't being counted).

Run the program with methods 0 to 3. Prepare a table with a row for each method. Show the run time and the following other information: the number of synchronizations performed per thread, and the number of additions performed to find the global sum by the *critical thread*. The *critical thread* is the thread that performs the most work. In many of our examples, that will be thread 0.

Problem 2: Locate the routine `reduction_method_4`. Add code to this routine so that the reduction is performed by using two tree reductions. In the first reduction each warp will do its own reduction. That is, after the first reduction we'll have the sum of threads 0-31, a separate sum for 32-63, etc. If the block size were 1024 threads (the maximum block size in a Fermi device) we would have 32 separate sums. Use the second reduction to find the sum of these 32 (or fewer) sums.

This can be written so that only one synchronization is needed, between the two reductions.

Problem 3: Add the data for your reduction routine to the table. Comment on the performance of each method. Which do you think should be used?