ECEC 413: Introduction to Parallel Computer Architecture
CUDA Lab 3: Tiled Matrix Multiplication

Prof. Naga Kandasamy, ECE Department, Drexel University

March 1, 2010

The Lab is due March 19, 2010. You may work on the assignment in teams of up to two people.

Important: Before starting the lab assignment, please read the tutorial on getting the CUDA SDK working on the cluster. You will find the tutorial (written by Tony Shackleford) on webCT under the "Labs" folder.

Your lab assignment is as follows:

- Edit the source files `matrixmul.cu` and `matrixmul_kernel.cu` to complete the functionality of the matrix multiplication on the GPU. The two matrices could be any size, but the resulting matrix is guaranteed to have a number of elements less than 64,000. Use shared memory when multiplying the two matrices.
- Your program should accept no arguments. The application will create two randomly sized and initialized matrices such that the matrix operation $M \times N$ is valid, and P is properly sized to hold the result. After multiplication on the GPU is complete, the program will compute the correct solution matrix using the CPU, and compare that solution with the GPU-computed solution. If the solutions match within a certain tolerance, if will print out "Test PASSED" to the screen before exiting.
- You must e-mail me all of the files needed to compile and run your code as a single zip file.

Also, once you have completed the assignment, answer the following questions in a separate lab report.

- In your kernel implementation, how many threads can be simultaneously executing on a GeForce GTX 280 GPU, which contains 30 Streaming Multiprocessors? Use `nvcc --ptxas-options="-v" matrixmul_kernel.cu` to see the resource usage of your kernel (although the compilation will fail, it will only do so after compiling the kernel and displaying the relevant information.)
- How many times is each element of the input matrices loaded during the execution of the kernel?

Your assignment will be graded on the following parameters:

- Correctness: 25%.
- Functionality: 40%, shared memory is used in the kernel to mask global memory access latencies.
- Report: 35%. Answer to question 1: 20%, answer to question 2: 15%