ECEC 413: Introduction to Parallel Computer Architecture
CUDA Lab 1: Simple Matrix Multiplication

Prof. Naga Kandasamy, ECE Department, Drexel University

February 13, 2010

The Lab is due February 22, 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 `MatrixMulOnDevice()` function in `matrixmul.cu` and the `MatrixMulKernel()` function in `matrixmul_kernel.cu` to complete the functionality of the matrix multiplication on the GPU. Do not change the source code elsewhere. The size of the matrix is defined such that one thread block will be sufficient to compute the entire solution matrix. The input matrices must be loaded from GPU global memory. The use of shared memory is not required for this assignment.
- Your program should accept no arguments. The application will create two randomly initialized matrices to multiply. After the GPU-based multiplication kernel is invoked, it will then 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), the application 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.

- How many times is each element of the input matrices loaded during the execution of the kernel?
- What is the memory-access to floating-point computation ratio in each thread? Consider a multiply and addition as separate operations, and ignore the storing of the result. Only global memory loads should be counted towards your off-chip bandwidth.

Your assignment will be graded on the following parameters:

- Correctness: 25%.
- Functionality: 40%, that includes the correct usage of CUDA library calls and C extensions and the correct usage of thread id's in the matrix computation.
- Report: 35%. Answer to question 1: 15%, answer to question 2: 20%