ECEC-622: Introduction to Parallel Computer Architecture CUDA Programming Assignment 1

Prof. Naga Kandasamy, ECE Department, Drexel University

April 28, 2013

The lab is due on May 6, 2013. You may work on the problems in teams of up to two people.

Matrix-Vector Multiplication. You will multiply a dense $n \times n$ matrix A with an $n \times 1$ vector x to yield the $n \times 1$ result vector y. The serial algorithm is shown below.

```
procedure VEC_MAT_MULT($A$, $x$, $y$)
int $i$, $j$;
for $i := 0$ to $n - 1$ do
    $y[i] := 0$;
    for $j := 0$ to $n - 1$ do
        $y[i] := y[i] + A[i, j] \times x[j]$;
    end for
end for
```

Edit the `vec_mat_mult_on_device_using_global_memory()` and `vec_mat_mult_on_device_using_shared_memory()` functions in `vec_mat_mult.cu` and the corresponding kernel functions in `vec_mat_mult_kernel.cu` to complete the functionality of the vector-matrix multiplication on the GPU. Do not change the source code elsewhere (except for adding timing-related code). The size of the matrix is guaranteed to be 4096×4096 and the size of the vector will be 4096×1. The CUDA source files for this question are available on webCT as a zip file.

Your program should accept no arguments. The application will create a randomly initialized matrix and a vector to multiply. After the GPU-based multiplication kernels are invoked, it will then compute the correct solution using the CPU and compare that solution with the GPU-computed solutions. If the solutions match within a certain tolerance, the application will print out "Test PASSED" to the screen before exiting.

E-mail me all of the files needed to run your code as a single zip file called `lab3.zip`.

This question will be graded on the following parameters:

- Use GPU global memory to get your code working. (20 points)
- Use GPU shared memory to improve performance. (30 points)
- A two/three page report describing how you designed your kernel (use code or pseudocode to clarify the discussion) and the amount of speedup obtained over the serial version for both GPU-based versions. (10 points)
- The GTX 275 GPU can achieve a peak processing rate of about 933 GFLOPs. The memory bandwidth on the device is 141.7 Gb/s. How many floating-point operations must be performed per load operation to achieve the peak processing rate? What is the performance of your kernels (both naive as well as the one that uses shared memory), in terms of GFLOPs? (20 points)