# ECEC-413: Introduction to Parallel Computer Architecture
# CUDA Programming Assignment 2

Prof. Naga Kandasamy, ECE Department, Drexel University

October 14, 2011

The lab is due on October 26, 2011. 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.

```
1: procedure VEC_MAT_MULT(A, x, y)
2: int i, j;
3: for i := 0 to n − 1 do
4:    y[i] := 0;
5:    for j := 0 to n − 1 do
6:       y[i] := y[i] + A[i, j] × x[j];
7:    end for
8: end for
```

Edit the `vec_mat_mult_on_device()` function in `vec_mat_mult.cu` and the `vec_mat_mult_kernel()` function 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 $4096 \times 4096$ and the size of the vector will be $4096 \times 1$. 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 kernel is invoked, it will then compute the correct solution 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.

E-mail me all of the files needed to run your code as a single zip file called `lab1.zip`.

This question will be graded on the following parameters:

- *Correctness*: 50 points. Use the shared memory on the GPU to get your code working.
- *Report*: 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.
- *Theory*: 20 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 kernel, in terms of GFLOPs?