CS 635 - Spring 2012
Project 3
This homework assignment is due by 11:59 pm on Sunday, May 6, 2012.

CUDA Programming Assignment

Introduction

The original Game of Life (GOL) was developed in 1970 by John Conway at the University of Cambridge. The game demonstrates that some simple local rules can lead to interesting large-scale behavior. The game is played on a periodic grid of cells (i.e. a toroidal mesh). Each cell can either be occupied by an organism or it can be unoccupied.

The game begins either with a random selection of occupied cells, or with a pattern that is read in from a file. From this initial generation, the next generation is calculated using the following birth, death, and survival rules:

- If an occupied cell has 0 or 1 occupied neighbors, the organism dies due to loneliness.
- If an occupied cell has 4, 5, 6, 7, or 8 occupied neighbors, the organism dies due to overcrowding.
- If an occupied cell has 2 or 3 neighbors, the cell continues to be occupied, that is, the organism survives in the next generation.
- If an unoccupied cell has exactly 3 occupied neighbors, then this cell will be occupied in the next generation, that is, an organism is born.

One of the interesting problems associated with the Game of Life is to identify the existence of oscillating or limiting patterns to which a grid of cells might converge to. You can find additional information about the game on the web, including Wikipedia and the following websites:

- A *Scientific American* article:
  `http://www.ibiblio.org/lifepatterns/october1970.html`
- Game description and Java applet:
  `http://www.bitstorm.org/gameoflife`

Project Description

The purpose of this programming assignment is to *design, implement, and evaluate* a CUDA program that performs the two-dimensional Game of Life calculations. To gain a better understanding of these tasks, you should begin by programming a sequential version of the Game of Life before you start the CUDA version.

The parallel version where the calculations are divided among a set of threads may require you to examine ways in which the grid of cells could be decomposed into smaller parts. If necessary, you can review earlier class lectures where block and strip decomposition were discussed.

Input/Output

Your program should take the command line arguments: the number of generations (G) to iterate and the size (N) of the $N \times N$ Game of Life grid. You can also input the number of massively parallel threads (T) to be used, but if you prefer, T can be specified statically within your code. Note that the size of the grid, N can be much larger than the number of threads T.

The output of your program should include the size of the Game of Life grid, the number of generations performed, the number of threads/blocks that you used, and a list of resulting live cell coordinates (x, y) at the end of the Game of Life simulation.

GOL - Generation One

To start off your Game of Life simulation, you can employ a random number generator to create the initial life grid. Recall that we discussed three approaches in class that either use the CPU to generate the initial grid or use the GPU device to generate the initial grid.

To test your sequential and CUDA programs for correctness, you might also want to generate synthetic input that verifies that the flip-flop and blinker patterns that Conway identified (see the Scientific American article) occur in both your sequential and CUDA codes.

Thread/Block Experiments

Since this is the last project of the semester, your last task is to design and performa a series of experiments that measure the performance of your sequential and CUDA implementations. You should determine the best combination of CUDA grid and thread blocks for your implementation when Game of Life grid sizes range from $N = 2^2, 2^3, \ldots, 2^{12}$ (or as large as you can go.)

The design of these experiments is up to you- it's time to utilize all that you have learned this semester to find an optimal solution that performs the Game of Life simulation as quickly as possible on a parallel system.

Turn In:

- Project Write-Up:
  1. A brief overview of your implementation (in less than 5 pages) which includes the following:
     (a) A description of your CUDA implementation for the Game of Life program.
     (b) Quantitative evidence as to whether or not you observed speedups in your parallel implementation. Include a discussion of where/when bottlenecks that affected performance might have occurred.
     (c) Data from your execution experiments should presented in table or graph form.
  2. Annotated program listings
  3. Instructions on how to compile and run your CUDA code on Tesla1/Tesla2
  4. Evidence that your CUDA implementation is correct, e.g. sample output produced after $G = 3$ generations by your sequential and CUDA codes for 8×8 Game of Life grids starting with the same Generation One configuration. The number of threads used for the CUDA code should be $T = 64$.
- Electronic copy of your code: An electronic version of your code should be submitted along with the project write-up. This code should be uploaded to the class Moodle website with your documentation before the due date.