ECSE 420 – Parallel Computing – Fall 2020

Lab 1 – Logic Gates Simulation

Due: October 16, 2020 at 11:59 pm

In this lab, you will write the code emulating basic logic gates, parallelize it using CUDA, and write a report summarizing your experimental results on comparing different ways of allocating the memory.

Logic gates

Logic gates are the basic building blocks of computers. For this lab, you will only need to consider the following gates: AND, NAND, OR, NOR, XOR, and XNOR. You can find the truth table for those gates in Table 1.

Input		Output					
A	B	AND	NAND	OR	NOR	XOR	XNOR
0	0	0	1	0	1	0	1
0	1	0	1	1	0	1	0
1	0	0	1	1	0	1	0
1	1	1	0	1	0	0	1

Table 1 Truth Table of 2-Input Logic Gate

Detailed Instructions

You will be given three test files, along with the expected outputs, and a simple program that compares two output files (i.e., the one containing the results that we give to you, and another containing your program output) and reports number of errors.

For each step below, you should verify and benchmark your code against test cases provided and report the execution time obtained. You should report the times for kernel function only, and exclude reading and writing times.

For this lab, your program only needs to launch one thread per logic gate. That is, if you have N element in your input file, your program needs to launch N threads.

1. Write code that simulates logic gate sequentially. Your code should take input command like

   ./sequential <input_file_path> <input_file_length> <output_file_path>

2. Parallelize your code using explicit memory allocation in CUDA. In addition to execution time, you should also measure the explicit data migration (copy data from host to device) time. Your code should take input command like

   ./parallelal_explicit <input_file_path> <input_file_length> <output_file_path>

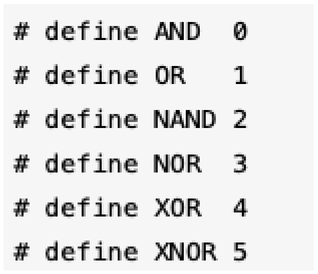
3. Parallelize your code using unified memory allocation in CUDA and compare its execution time with explicit memory method. Your code should take input command like

   ./parallelal_unified <input_file_path> <input_file_length> <output_file_path>

4. (Optional) Parallelize your code using unified memory allocation with data prefetching (moving data to GPU before the kernel launch) in CUDA. The implementation of data prefetching might differ in different system setups. It is important to include and discuss your implementation in the report. This step is mainly for students who are interested and students will not be penalized for omitting this step, but they can get bonus 5% of the grade.
5. Discuss the results and refer to the architecture on which you run your experiment. Tell us what conclusions you draw about the memory allocation alternatives.

Input and Output Data Format

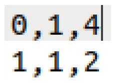
Your program should take an input of a CSV file and produce an output of a plain text file where each line contains a single binary output of a gate. Each line of your input file represents one logic gate: the first two columns represent the inputs in boolean values (0,1) and the third column indicates the gate type in integer (0 to 5). You should use the type definition in Figure 1 to decode the gate type.



```
# define AND  0
# define OR   1
# define NAND 2
# define NOR  3
# define XOR  4
# define XNOR 5
```

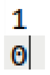
Figure 1 Type Definition of Logic Gate

For example, if you have an input file as shown in Figure 2, the first line represents a XOR gate with inputs of 0 and 1, and second line indicates a NAND gate with inputs of 1 and 1. The corresponding output file is shown in Figure 3.



```
0,1,4
1,1,2
```

Figure 2 Input File Example



```
1
0
```

Figure 3 Output File Example

You will also be given a simple program, "compareResults.c", to check your result. To run it,

./compareResults <path_to_solution> <path_to_output>

Submission Instructions:

Each group should submit a single zip file with the filename *Group_<your group number> _Lab_1.zip*. (Ex – Group_09_Lab_1.zip).

1. A **lab report** answering all the questions in the lab (please use a PDF format).
   a. Must be named *Group_<your group number>_Lab_1_Report*. (Ex – Group_03_Lab_1_Report.pdf).
   b. Must have a cover page (Include Group number, members' names and ID).
   c. Must follow the logical order for the lab discussions.
2. Add **your own source** code inside.