CS4402 Due: 3rd of April 2024

Problem Set 2

CS9635 *Submission instructions on last page*

Problem 1.

The goal of this problem is to realize a CUDA implementation of univariate polynomial multiplication. This example was used in Section 2.3 of Dependence Analysis and (Automatic) Parallelizati It is also part of the next chapter, namely Parallel Random-Access Machines. To simplify the implementation, we will make the same assumption as in these two chapters, that is, the two input polynomials have the same degree, thus. these two polynomials can be regarded as two vectors of he same length. Note that Slide 41 of the slides of the chapter Dependence Analysis and (Automatic) Parallelization is essentially a CUDA kernel. However, this is not an efficient CUDA kernel for reasons that will become clear in the chapter Optimizing CUDA code.

1. Your CUDA program should use multiple thread-blocks and allow the degree n to reach (at least) 2^{16}. Note that the PRAM model does not have a concept of thread-block. So the n^2 processors of the PRAM algorithm must translate to n^2/B thread-blocks with B threads per block, for a well-chosen B.

2. In fact, if your CUDA kernel uses 1D thread-blocks of size B, you must experiment with the value of B, say from $B \in \{32, 64, 128, 246, 512\}$ and pick one that minimizes running time.

3. Note that the PRAM algorithm for polynomial multiplication is somehow different from that of Slide 41 of Chapter Dependence Analysis and (Automatic) Parallelization. We shall use the algorithm of Slide 41 as our starting point towards an optimized kernel.

4. The input polynomials must have coefficients of type `int`, randomly chosen in $\{-1, 0, 1\}$ so as to avoid overflows.

5. Like some of the examples of the collection **simple_examples** your CUDA program should include a C function calculating the same product of polynomials as your kernel, so as to verify the result. of the computations performed on the GPU.

6. Your program should work correctly on one of two GPU clusters: `gpu1.gaul.csd.uwo.ca` and `gpu2.gaul.csd.uwo.ca`.

Question 1. [80 points] Realize a first version of your CUDA program using $\lceil (2n+1)/B \rceil$ thread-blocks with $B = 32$ threads. Indeed, the number of coefficients in the product is $2n + 1$.

Question 2. [20 points] For $n \in \{2^{14}, 2^{16}\}$ experiment with $B \in \{32, 64, 128, 246, 512\}$ and report on the collected data.

Submission instructions.

Format: *Input tests* and a `Makefile` (for compiling and running) are required. Please provide a `README` describing how to compile and test your code. Please submit source code only, that is, do not provide compiled code! All the files should be archived using the UNIX `tar` command.

Submission: The assignment should be submitted via OWL.

Collaboration. You are expected to do this assignment *on your own* without assistance from anyone else in the class. However, you can use literature and if you do so, briefly list your references in the assignment. Be careful! You might find on the web solutions to our problems which are not appropriate. For instance, because the parallelism model is different. So please, avoid those traps and work out the solutions by yourself. You should not hesitate to contact me if you have any questions regarding this assignment. I will be more than happy to help.

Marking. This assignment will be marked out of 100. A 10 % bonus will be given if your submission is clearly organized. Messy assignments (no Makefile, no README, no tests) will receive a 10 % malus in addition to possible deductions (e.g. if the code cannot be compiled).