Digital Systems

EEE4084F

Practical 6 – OpenCL

[30 Marks]

Introduction

The focus of this practical is investigating the performance metrics of an OpenCL matrix multiplication implementation.

The Programming Model

OpenCL uses a programming and memory model similar to OpenGL. The CPU must copy data to the GPU and then tell a kernel (OpenCL worker) to process the data. When the kernel is finished, the CPU can read back the result.

Download and inspect the practical source code (available from Vula resources). Familiarise yourself with the program and what the OpenCL wrapper is actually doing.

If you modify the code, be very careful with how you allocate and free memory. You need to allocate CPU and GPU memory separately, and then free them separately as well.

The local group size has to be an integer fraction of the global group size, in every dimension. Every time you change N, you have to recalculate the local group size.

Installation

If you want to run this practical from home, you'll need to install the OpenCL toolkit for your GPU and modify the compiler and linker search paths accordingly.

Part 1: Speed-up

The default program multiplies two 3×3 matrices. This process takes much longer on the GPU than the CPU, mainly due to overhead. At what matrix size does it become worth while to use the GPU instead of the CPU? Support your argument with measured results (presented in graphs and/or tables). You may also, for instance, make assumptions regarding speed-up expected were the CPU version multi-threaded (use results from previous practicals to support your assumptions).

Part 2: Data Transfer Overhead

The asynchronous nature of the OpenCL interface makes it difficult to obtain accurate timing information of the various steps of the process. Try to come up with a way to measure the data transfer overhead and processing time separately (**hint:** make use of the `clFinish` function in the `OpenCL_WriteData` and `OpenCL_ReadData` functions).

Use this new information to comment on the sources of OpenCL processing delay. Also comment on the speed-up factor achieved when transfer overhead is not taken into account. Does this relate well to the number of threads that are running on the GPU? If not, provide an argument for why this is the case.

Do this for large N (pick a value that takes long enough to dominate the transfer overhead, but does not let you finish a whole coffee between runs).

Part 3: Report

Compile your experiments and findings into an IEEE-style conference paper. Make sure you include ALL hardware details, including GPU and CPU clock rates, if available. Of particular importance is the local work group size and number of compute units.

The page limit is 3 pages. Submit your paper to the Vula Assignment for this practical.

END OF ASSIGNMENT