Phase 1.1:: OpenCL with Intel

Configaration

- Syestem: Windows
- OS: Windows 10
- Cpu: Intel
- GPU: iGpu
- Installed Softwares:
    - Intel Graphics Driver
    - OpenCL Runtime
    - Visual Studio 2022
    - Visual Studio Code
    - CodeBlocks

Sample Code:

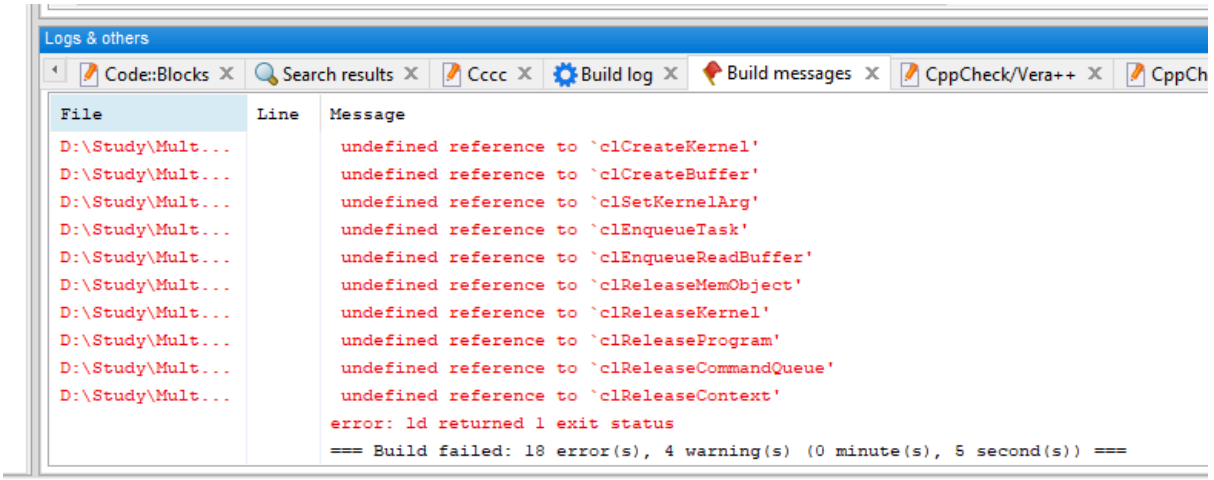
- Hello_World.c (sourced from moodle)

Guidelines: (From Moodle)

- 01 OpenCL_SDK_Installation_2024
- 02 OpenCL_IDE_Integration_2024

Result:

- VsCode: Error while building.
- CodeBlocks: Build Failed with 18 errors



- Possible Issue: SDK or IDE Inegration

Debugging:

- Opencl sdk setup tutorial: https://youtu.be/HGWNxq1vrs4?si=OvasKRphAprT-giU
- Ms .net framework (v3.5) : https://dotnet.microsoft.com/en-us/download/dotnet-framework/net35-sp1
-
- intel driver and suport assitant: https://www.intel.com/content/www/us/en/support/detect.html

- OpenCL™ Runtimes for Intel® Processors: https://www.intel.com/content/www/us/en/developer/articles/tool/opencl-drivers.html

- download intel opencl sdk and intel tbb:
    - https://intel-sdk-for-opencl-applications.en.softonic.com/download
    - https://www.intel.com/content/www/us/en/developer/articles/tool/intel-system-studio-download-and-install-intel-c-compiler.html?wapkw=system%20studio%20package
    - https://www.softpedia.com/get/Programming/SDK-DDK/Intel-SDK-for-OpenCL-Applications.shtml
    - https://www.intel.com/content/dam/develop/external/us/en/documents/iss-icc-download-install-cmdline-780679.pdf
- Sample Code
    - https://sourcecode.talkplayfun.com/electromagnetics/100-HelloOpenCL.zip
- C++ LC++ Library Extension
    - https://sourcecode.talkplayfun.com/extension/CppExtension.zip

Update:

- No positive outcome. Will try a different method

Phase 1.2:: OpenCL with Nvidia (Remote PC)

Configuration:

- Windows 10
- CPU: AMD 5900x
- GPU: Nvidia 2080 Xtreme
- Installed Softwares:
    - Nvidia Graphics Driver
    - Nvidia CUDA Toolkit
    - Visual Studio Code
    - C/C++ extension for VS Code
    - MinGW GCC Compiler

Guideline:

- Moodle Instructions

Additional Guidelines (Hlghtly Recommended)

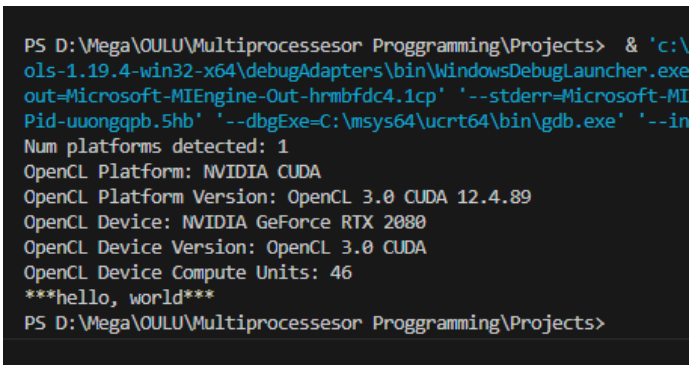
- MinGW GCC Setup for VsCode:
    - https://code.visualstudio.com/docs/cpp/config-mingw
    - Installing MinGW to build C++ Code on Windows
- OpenCL SDK and IDE(VsCode) Configuration: [path variables, cpp configuration, set task]
    - Tuto / Download Install OpenCL SDK / Compile Run C C++ / VSCode Windo...
- Nvidia OpenCL SDK Code Sample (For cl.h headers)
    - https://developer.nvidia.com/opencl

Setup Steps:

1. Install and update Nvidia driver, cuda toolkit, VsCode
2. Download OpenCL SDK and libraries
3. Set path variables
4. Download and setup MinGW GCC Compiler
5. Edit includePath and Set task in VSCode
6. Test with hello_world_opencl.c to test environment

Result

- Success. Expected Output with no error.



Update:

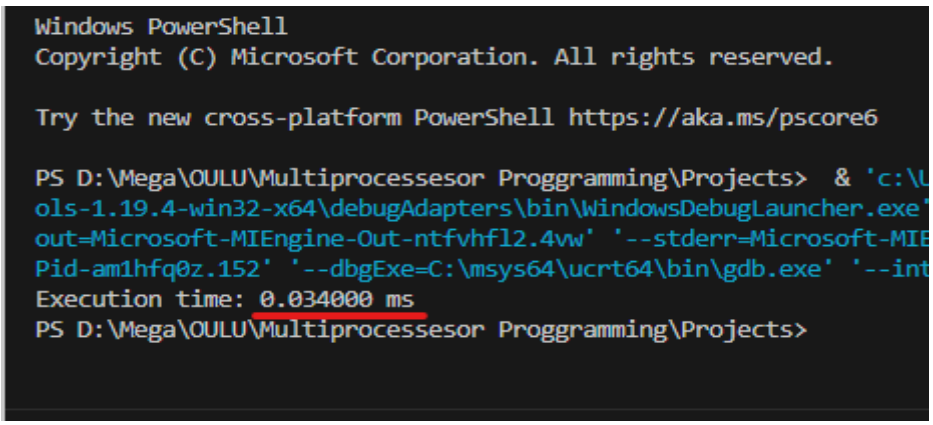
- Configured Remote Tunneling in Vscode so that it can be accessed remotely

Phase 2. :: OpenCL introduction. Matrix addition, filtering, profiling.

2.1.1 Matrix Addition

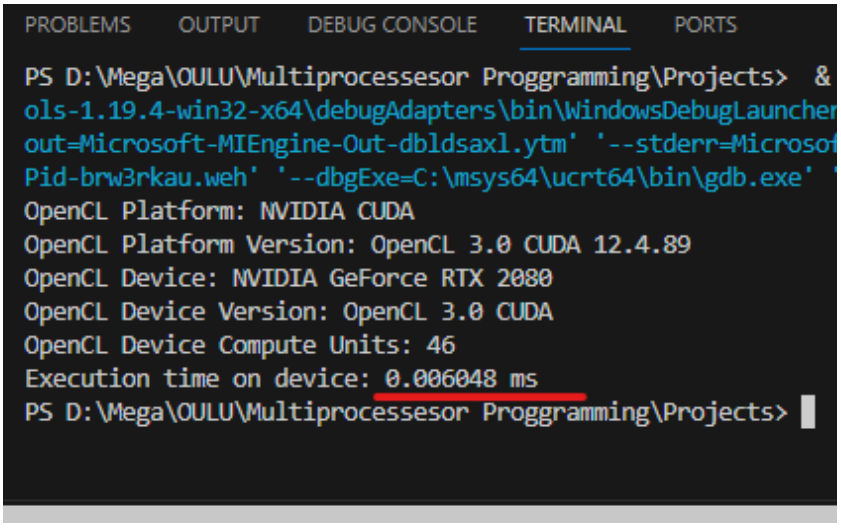
2.1.1.1 Addition without OpenCL

- Create matrix_addition.c to add two matrices (100x100) using c/c++ program
- Profile execution time using gettimeofday(..) function
- Result. The program ran successfully. It took 0.034000ms to execute the program.



2.1.2 Addition with OpenCL

- Create matrix_addition_opencl.c to add two matrices (100x100) using OpenCL programming
- Profile execution time using clGetEventProfilingInfo(..) function
- Make add_matrix.cl Kernel code for matrix addition.
- Result. The program ran successfully. It took 0.006048ms to execute the program.
- With OpenCL, the execution was 0.027952ms or 5.62x faster



2.1.2 Matrix Multiplication

2.1.2.1 Multiplication Without Opencl

- Create matrix_multiplication.c to add two matrices (100x100) using c/c++ program
- Profile execution time using gettimeofday(..) function
- Result. The program ran successfully. It took 2ms to execute the program.

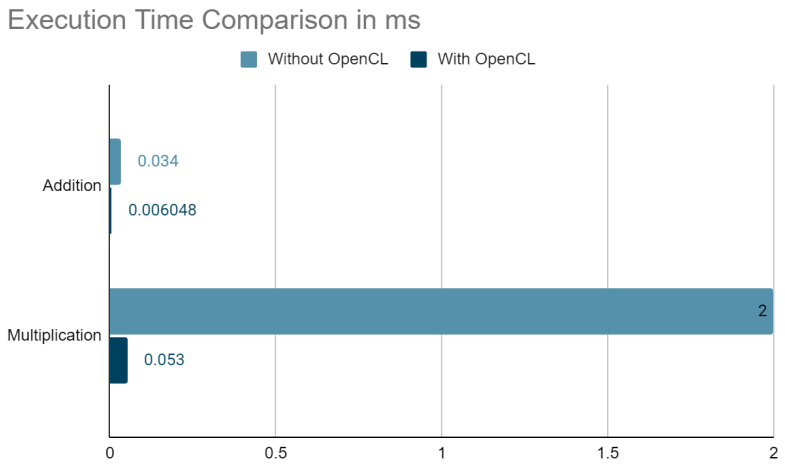
```
PS D:\Mega\OULU\Multiprocessesor Proggramming\Projects>  & 'c:\User
ols-1.19.4-win32-x64\debugAdapters\bin\WindowsDebugLauncher.exe' '-
out=Microsoft-MIEngine-Out-c0d43br4.mkp' '--stderr=Microsoft-MIEngi
Pid-ltxfpikt.jno' '--dbgExe=C:\msys64\ucrt64\bin\gdb.exe' '--interp
Host Execution Time: 2 ms
PS D:\Mega\OULU\Multiprocessesor Proggramming\Projects>
```

2.1.2.2 Multiplication Without Opencl

- Create matrix_multiplication_opencl.c to add two matrices (100x100) using OpenCL programming
- Profile execution time using clGetEventProfilingInfo(..) function
- Make multiply_matrix.cll Kernel code for matrix addition.
- Result. The program ran successfully. It took 0.053ms to execute the program.
- With OpenCL, the execution was 1.947ms or 38x faster

```
PS D:\Mega\OULU\Multiprocessesor Proggramming\Projects>  & 'c:\
ols-1.19.4-win32-x64\debugAdapters\bin\WindowsDebugLauncher.exe
out=Microsoft-MIEngine-Out-qjkko5dv.04x' '--stderr=Microsoft-MI
Pid-swwu03vl.pdq' '--dbgExe=C:\msys64\ucrt64\bin\gdb.exe' '--ir
Kernel Execution Time: 0.053 ms
PS D:\Mega\OULU\Multiprocessesor Proggramming\Projects>
```

2.1.3 Comparison



2.2: Read/save image, convert to grayscale, reduce size, apply filter in C/C++

Prerequisites

- Downloaded image_0.png and Created a source file image.c for the tasks
- Downloaded lodepng.c and lodepng.h and included as header in the source file.

Opening Image:

- Created a function named **ReadImage()**
- Use lodepng.c library to read the input images in RGBA format

Resizing:

- Created a function named **ResizeImage()** to resize the image
- took pixels from every fourth row and column.
- The resultant image was 1/16 of the original size (2940x2016 to 735x504).

Greyscaling

- Created **GrayScaleImage()** function to convert the image from RGB to gray.
- Transform the images to greyscale images (8 bits per pixel). By this eq, Y=0.2126R + 0.7152G + 0.0722B.

Filtering:

- Created a function named **ApplyFilter()**
- Applied a 5x5 moving filter (simple average) on the gray-scaled image matrix.

Encoding

- Created a function named **WriteImage()** to save the image
- used LodePNG to write the image by using **lodepng_encode_file().** The resulting image is normalized to grayscale having 8-bit depth.
- the resulting image was saved as **image_0_bw.png**

Profiling

- The profiling was done by **ProfileFunction()**
- Profiling was done by measuring the time of start and end time of executing each function and calculating their difference.
- Execution time is shown in the Result section.

Result

```
out=Microsoft-MIEngine-Out-ge51oy4z.um1' '--stderr=Microsof
Pid-sncw5ljj.02e' '--dbgExe=C:\msys64\ucrt64\bin\gdb.exe' '
ReadImage took 402 ms to execute
ResizeImage took 4 ms to execute
GrayScaleImage took 1 ms to execute
ApplyFilter took 15 ms to execute
WriteImage took 94 ms to execute
PS D:\Mega\OULU\Multiprocessesor Proggramming\Projects>
```

2.3: Read/save image, convert to grayscale, reduce size, apply filter in OpenCL