AMD **Accelerated** Parallel Processing TECHNOLOGY

SAMPLE

Hello C++ AMP

1 Overview

1.1 Location

`$(AMDAPPDKSAMPLESROOT)\samples\opencl\cl\UserStory\`

where `AMDAPPDKSAMPLESROOT` is an environment variable pointing to the installation path of the AMD APP SDK samples.

1.2 How to Run

1. Before running the sample, install Microsoft® Visual Studio® 2012 or higher and the latest AMD APP SDK.
2. Compile the sample. Use the command line to change to the directory where the executable is located. The pre-compiled sample executable is at `$(AMDAPPDKSAMPLESROOT)\samples\opencl\bin\x86\` for 32-bit builds, and `$(AMDAPPDKSAMPLESROOT)\samples\opencl\bin\x86_64\` for 64-bit builds.

2 Introduction

This is an introductory sample for developers who want to learn C++ AMP programming. This sample introduces various C++ AMP concepts (array_view, parallel_for_each, extent, lambda function, etc.) through a simple kernel that computes SAXPY (z = ax + y) on two arrays.

3 Implementation Details

The input vectors vX and vY are first initialized with random values:

```
for(int i = 0; i < numElement; i++)
{
    vX[i] = (float)(rand() % numElement);
    vY[i] = (float)(rand() % numElement);
}
```

Here is the simple loop that computes the SAXY on the CPU in a serial fashion:

```
for(int i = 0; i < numElement; i++)
{
    vZ_CPU[i] = alpha * vX[i] + vY[i];
}
```

To perform the computations on the GPU, the input data must be transferred to the GPU. An array_view object is used to wrap around the data containers in the CPU memory:

```
array_view<const float, 1> avX(numElement, vX);
array_view<const float, 1> avY(numElement, vY);
```

The C++ AMP runtime does the data transfer when the GPU accesses the input data through the array_view objects.

Here is the code that computes the SAXPY on the GPU:

```
parallel_for_each(avX.extent, [=](index<1> idx) restrict(amp)
{
    // idx is the zero-based index of the current thread.
    // Thread #0 --> idx == 0, Thread #1 --> idx == 1,...
    avZ_GPU[idx] = alpha * avX[idx] + avY[idx];
});
```

The parallel_for_each instructs the runtime to create `numElement` of threads to execute the lambda function in parallel. The variable `idx` contains the ID of the thread.

To read the results on the CPU side, the synchronize method is called to ensure the data is copied from the GPU back to the CPU container.

```
avZ_GPU.synchronize();
```

Finally, the GPU output is compared to the CPU output.

```
for(int i = 0; i < numElement; i++)
{
    if(vZ_GPU[i] != vZ_CPU[i])
    {
        std::cout << "Verification failed !!!" << std::endl;
        exit(1);
    }
}
```

4 References

1. *C++ AMP: Language and Programming Model* (downloadable from microsoft.com)

Contact

Advanced Micro Devices, Inc.
One AMD Place
P.O. Box 3453
Sunnyvale, CA, 94088-3453
Phone: +1.408.749.4000

For AMD Accelerated Parallel Processing:
URL: developer.amd.com/appsdk
Developing: developer.amd.com/
Support: developer.amd.com/appsdksupport
Forum: developer.amd.com/openclforum