AMD **Accelerated** Parallel Processing
TECHNOLOGY

SAMPLE

Hello World

1 Overview

1.1 Location

`$<APPSDKSamplesInstallPath>\samples\opencl\cl\`

1.2 How to Run

1. Before running the sample, install the latest AMD GPU driver and the latest AMD APP SDK.
2. Import the necessary header files into the project. The header files are located in `$<APPSDKSamplesInstallPath>\include`, where `A<APPSDKSamplesInstallPath>` is the installation path of the AMD APP SDK.
3. Compile the sample. Use the command line to change to the directory where the executable is located. The pre-compiled sample executable is at `$<APPSDKSamplesInstallPath>\samples\opencl\bin\x86\` for 32-bit builds, and `$<APPSDKSamplesInstallPath>\samples\opencl\bin\x86_64\` for 64-bit builds.

2 Introduction

This is a stand-alone OpenCL sample, independent of any utility libraries in the SDK. It is an easy sample for a new user to start coding and learning OpenCL.

HelloWorld is an OpenCL kernel that modifies the input string `GdkknVnqkc`. Each kernel thread receives one character of the string and increments the assigned output by 1. For example: work-item 0 gets G, and G plus 1 is H in ASCII. So, the string changes to “HelloWorld” after kernel execution.

3 Implementation Details

The following steps guide you through the process of writing a simple OpenCL program.

Step 1. Get platform.

Query the available platforms, and choose an appropriate one. For information about the platforms, used `clGetPlatformIDs` and `clGetPlatformInfo`.

Step 2. Query devices.

Use `clGetDeviceIDs` to query the platform, and choose the first GPU device. If there is no GPU, use the CPU.

Step 3. Create context.

Use `clCreateContext` to create a context using the first device. This can be a GPU or CPU, depending on the available devices on the system.

Step 4. Create command queue.

Use `clCreateCommandQueue` to create a command queue on the context for the device.

Step 5. Create program.

Use `clCreateProgramWithSource` to create the program that uses the kernel file.

Step 6. Build program.

Use `clBuildProgram` to build the program.

Step 7. Create memory objects.

Define the initial input and output buffers for the host, and create memory objects for the kernel. Use `clCreateBuffer` to create cl_mem objects.

Step 8. Create kernel object.

Use `clCreateKernel` to create a kernel for the device.

Step 9. Set kernel arguments.

Use `clSetKernelArg` to set arguments for the kernel.

Step 10. Run the kernel.

Use `clEnqueueNDRangeKernel` to run the kernel.

Step 11. Read the output back to host memory.

Use clEnqueueReadBuffer to read the results of the executed kernel back to host buffer.

Step 12. Release the resources used by OpenCL.

a. Using API **clReleaseKernel** to release kernel.

b. Using API **clReleaseProgram** to release program.

c. Using API **clReleaseMemObject** to release buffer.

d. Using API **clReleaseCommandQueue** to release command queue.

e. Using API **clReleaseContext** to release context.

Use `free` or `delete` to free the resources used by the host.

If successful, the `errcode_ret` is set to `CL_SUCCESS`; otherwise, a different error codes is returned.

For more information about the API functions and error codes, see the latest *OpenCL Specification*.

4 OpenCL Kernel

```
__kernel void helloworld(__global char* in, __global char* out);
```

The `__kernel` denotes that the function is a kernel function. It has two arguments: the inputBuffer is passed as an argument to `in`, and the outputBuffer is passed as an argument to `out`.

```
int num = get_global_id(0);
```

This API provides the work-item id in the global execution space. For this kernel, the execution space is the same size as `MEM_SIZE(10)`. Thus, each instance of the kernel that is executed has associated with it an element in `*in` and `*out`.

```
out[num] = in[num] + 1;
```

Depending on the `threadid(num)`, each thread gets an element from `in`, and increments it by 1, then passes it to `out`.

5 References

1. *OpenCL Specification*, v. 1.2, available at http://www.khronos.org/opencl/ .

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/
Forum: developer.amd.com/openclforum