

1 Overview

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

1.2 How to Run See the *Getting Started* guide for how to build samples. You must first compile the sample.

1. Use the command line to change to the directory where the executable is located. The precompiled 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. Type the command `MemoryModel` to initialize the input from 1 to 256.

2 Introduction

This is a simple sample used to teach developers the concept and use of the four distinct memory regions in OpenCL.

3 Implementation Details

3.1 Background Work-item(s) executing a kernel have access to four distinct memory regions: Global Memory, Local Memory, Constant Memory, and Private Memory.

1. The global memory (`__global`) is the memory region that is accessible by all the work-items. Reads and writes can be cached, depending on the devices.
2. The local memory (`__local`) has local visibility to a work-group. This can be used to share data between work-items in that work-group.
3. The constant memory (`__constant`) is a region in the global memory; it remains constant over the execution of a kernel.
4. The private memory (`__private`) is a memory region private to a work-item; it is not visible to another work-item. Any variable declared without an address space qualifier is private by default.

3.2 OpenCL Kernel

```
#define GROUP_SIZE 64

__constant int mask[] =
{
    1, -1, 2, -2
};

__kernel void MemoryModel(__global int *outputbuffer, __global int *inputbuffer)
{
```

```

__local int localBuffer[GROUP_SIZE];
    __private int result=0;
    __private size_t group_id=get_group_id(0);
__private size_t item_id=get_local_id(0);
__private size_t gid = get_global_id(0);

// Each workitem within a work group initialize one element of the local buffer
__private int input_data = inputbuffer[gid];
localBuffer[item_id]= input_data;
// Synchronize the local memory
barrier(CLK_LOCAL_MEM_FENCE);

// add 4 elements from the local buffer
// and store the result into a private variable
for (int i = 0; i < 4; i++) {
    __private int t = localBuffer[(item_id+i)%GROUP_SIZE];
    result += t;
}
// multiply the partial result with a value from the constant memory
result *= mask[group_id%4];

// store the result into a buffer
    outputbuffer[gid]= result;
}

```

This sample uses a group size of 64.

The kernel starts by initializing the local array `localBuffer` with data from the global memory `inputbuffer`.

```
localBuffer[item_id]=inputbuffer[gid];
```

Note that `gid` is a private variable that holds the global ID unique to a work-item. For example, `gid==0` for work-item 0, `gid==1` for work-item 1, etc. The variable `item_id` is similar, but holds the local id.

The work-group has 64 work-items; all 64 slots of `localBuffer` are initialized in parallel.

Each work-item of this kernel starts by initializing the local memory using data from the global memory. This sample uses a group size of 64. In this case, one work-item initializes one slot of `localBuffer` in parallel.

A memory barrier is needed to ensure that the changes to local memory become visible to all the work-items of that work-group.

The loop that follows shows that each work-item loads four consecutive values from the local memory, then add them together. The partial result is different for every work-item, so it is being stored in a private variable.

The partial result then is multiplied by a value from the constant memory.

Finally, the result is stored into another global array `outputbuffer`, which can be transferred back to the host or used as input by another kernel.

4 References

1. *OpenCL Specification v1.2*, Memory Model (section 3.3), Address Space Qualifiers (section 6.5).

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



The contents of this document are provided in connection with Advanced Micro Devices, Inc. ("AMD") products. AMD makes no representations or warranties with respect to the accuracy or completeness of the contents of this publication and reserves the right to make changes to specifications and product descriptions at any time without notice. The information contained herein may be of a preliminary or advance nature and is subject to change without notice. No license, whether express, implied, arising by estoppel or otherwise, to any intellectual property rights is granted by this publication. Except as set forth in AMD's Standard Terms and Conditions of Sale, AMD assumes no liability whatsoever, and disclaims any express or implied warranty, relating to its products including, but not limited to, the implied warranty of merchantability, fitness for a particular purpose, or infringement of any intellectual property right.

AMD's products are not designed, intended, authorized or warranted for use as components in systems intended for surgical implant into the body, or in other applications intended to support or sustain life, or in any other application in which the failure of AMD's product could create a situation where personal injury, death, or severe property or environmental damage may occur. AMD reserves the right to discontinue or make changes to its products at any time without notice.

Copyright and Trademarks

© 2013 Advanced Micro Devices, Inc. All rights reserved. AMD, the AMD Arrow logo, ATI, the ATI logo, Radeon, FireStream, and combinations thereof are trademarks of Advanced Micro Devices, Inc. OpenCL and the OpenCL logo are trademarks of Apple Inc. used by permission by Khronos. Other names are for informational purposes only and may be trademarks of their respective owners.