AMD **Accelerated** Parallel Processing TECHNOLOGY

SAMPLE

ExtractPrimes

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 first must 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.

Ensure that the OpenCL 2.0 environment is installed.

Type the following command(s).

1. `ExtractPrimes`
   This command runs the program with the default options.
2. `ExtractPrimes -h`
   This command prints the help file.

1.3 Command Line Options

Table 1 lists, and briefly describes, the command line options.

Table 1 Command Line Options

Short Form	Long Form	Description
`-h`	`--help`	Shows all command options and their respective meanings.
	`--device [cpu|gpu]`	Devices on which the OpenCL kernel is to be run. Acceptable values are `cpu` or `gpu`.
`-q`	`--quiet`	Quiet mode. Suppresses all text output.
`-e`	`--verify`	Verify results against reference implementation.
`-t`	`--timing`	Print timing-related statistics.
`-v`	`--version`	AMD APP SDK version string.
	`--dump [filename]`	Dump the binary image for all devices.
	`--load [filename]`	Load the binary image and execute on the device.
	`--flags [filename]`	Specify the filename containing the compiler flags for building the kernel.
`-i`	`--iterations`	Number of iterations.
`-p`	`--platformId`	Select the platformId to be used[0 to N-1 where N is number platform s available].
`-d`	`--deviceId`	Select deviceId to be used[0 to N-1 where N is number devices available].

2 Introduction

This sample demonstrates the usage of new workgroup built-ins `work_group_scan_inclusive_add` (to compute the prefix sums at workgroup level) and `work_group_broadcast` (to broadcast the value from one workitem to all others in the workgroup).

The sample workflow is as follows:

1. The host creates the required buffers, such as input, primes (Boolean array for each input element), output (for holding the prefix sums on primes array), outPrimes (for final result). These are sent to the kernel.
2. The host launches four kernels in this order:
    i. `set_primes_kernel` computes whether each array element is prime and fills the primes array
    ii. `group_scan` and `global_scan` compute the prefix sums globally and puts the output prefix sums into output array
    iii. Finally, `get_primes_kernel` extracts and places them into the `outPrimes` array using the indexes using the prefix sums in output array.

This sample must be run in the OpenCL 2.0 environment.

3 Implementation

Kernels filling the primes Boolean array and group scan is obvious. For global scan, the same technique that was described in the BuiltInScan sample is used. The `get_primes` kernel uses these prefix sums to fill the `outPrimes` array for each prime found in the input array. The performance improvement found compared to the OpenCL 1.2 version is described in Chapter 6 of the *AMD OpenCL User Guide* document.

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