AMD **Accelerated** Parallel Processing TECHNOLOGY

SAMPLE

Simple Multi GPU

1 Overview

1.1 Location

`$<APPSDKSamplesInstallPath>\samples\C++Amp\`

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 default executables are placed in `$<APPSDKSamplesInstallPath>\samples\C++Amp\bin\x86\` for 32-bit builds, and `$<APPSDKSamplesInstallPath>\samples\C++Amp\bin\x86_64\` for 64-bit builds.

Type the following command(s).

1. `SimpleMultiGPU`

   This runs the program with the default options: s = (1600 * 1200 * 3).

2. `SimpleMultiGPU -h`

   This 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`	Show all command options and their respective meaning.
`-q`	`--quiet`	Quiet mode. Suppresses text output.
`-e`	`--verify`	Verify results against reference implementation.
`-t`	`--timing`	Print timing-related statistics.
`-v`	`--version`	AMD APP SDK version string.
`-x`	`--samples`	Number of example input values (multiples of three).

2 Introduction

An accelerator represents a “target” on which C++ AMP code is meant execute thread-parallel and data-parallel operations assisting the host application running on CPU. In most scenarios, an accelerator is a GPU device. Accelerators are represented in C++ AMP as objects of the accelerator class. The C++ AMP runtime supports the notion of a default accelerator to run C++ AMP code on a target that can be the best accelerator in the system. However C++ AMP also supports choosing a specific accelerator available in the system. This becomes necessary in a scenario where the C++ AMP code must be executed on at least two accelerators.

C++ AMP supports the following predefined accelerators.

- `accelerator::default_accelerator` represents the default accelerator that the C++ AMP runtime has chosen.
- `accelerator::direct3d_ref` represents an extremely slow reference rasterizer emulator that simulates a direct3d device on the CPU (useful for debugging).
- `accelerator::cpu_accelerator` represents the CPU.
- `accelerator::direct3d_warp` is an accelerator that uses multi-core and SSE2, SSE3, and SSE 4.1, depending on the CPU capabilities. This is the current CPU fallback.

C++ AMP also enumerates hardware accelerators that can be chosen as targets by user programs.

3 Implementation Details

This example shows how data can be initialized, processed, and synchronized between multiple GPUs using C++ AMP. At minimum, there must be two hardware accelerators. This example identifies two hardware accelerators (GPUs), splits the input data evenly across them, and executes an RGB-to-YUV 4:4:4 color conversion algorithm on both in parallel. Checking that the dedicated memory is not 0 kB, and ensuring that the is_emulated flag of the accelerator is false, helps identify hardware accelerators. The input RGB data then is split into two AMP `array_view`s, which then become inputs to the two selected accelerators. By using the overload of `parallel_for_each()`, which allows specifying of the target `accelerator_view`, the same lambda is run on both the accelerators with the first half of the input given to the first `accelerator_view`, and the second half of the input given to the second accelerator_view. The outputs of both accelerators are then verified.

Although this example is targeted at multiple GPUs, it easily can be modified to execute in parallel on a CPU and GPU accelerators, instead of two GPUs. This is left as an exercise for the user.

4 Recommended Input Option Settings

For best performance, enter the following on the command line: `-x 5760000 -q -t -e`

5 References

1. http://www.danielmoth.com/Blog/concurrencyaccelerator.aspx
2. http://en.wikipedia.org/wiki/YUV

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

© 2012 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.