

## Double Precision Matrix Multiplication

### 1 Overview

**1.1 Location** `$<APPSDKSamplesInstallPath>\samples\opencl\cpp_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 default executables are placed in `$<APPSDKSamplesInstallPath>\samples\opencl\bin\x86` for 32-bit builds and `$<APPSDKSamplesInstallPath>\samples\opencl\bin\x86_64` for 64-bit builds.

Type the following command(s).

1. `MatrixMulDouble`  
Multiplies two matrices of size 64 x 64 with blockSize = 16.
2. `MatrixMulDouble -h`  
This prints the help message.

**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 meaning.
	--device	Devices on which the program is to be run. Acceptable values are <code>cpu</code> or <code>gpu</code> .
-q	--quiet	Quiet mode. Suppresses all text output.
-e	--verify	Verify results against reference implementation.
-t	--timing	Print timing.
	--dump	Dump binary image for all devices.
	--load	Load binary image and execute on device.
	--flags	Specify compiler flags to build the kernel.
-p	--platformId	Select platformId to be used (0 to N-1, where N is the number of available platforms).
-d	--deviceId	Select deviceId to be used (0 to N-1, where N is the number of available devices).
-v	--version	AMD APP SDK version string.
-x	--heightA	Height of matrix A.
-y	--widthA	Width of matrix A and height of matrix B.

Short Form	Long Form	Description
-z	--widthB	Width of matrix B.
-i	--iterations	Number of iterations for kernel execution.
	--eAppGflops	Prints GFLOPS calculated from transfer plus kernel time.

## 2 Implementation Details

This sample computes the following relation among matrices using double precision floating point:

$$\mathbf{C} = \mathbf{AB}$$

Dimensions of matrix A = (y, x) {width0, height0}.

Dimensions of matrix B = (z, y) {width1, width0}.

This results in a matrix C, with dimension = {z, x}.

There are two versions of matrix multiply kernel in this sample:

- One for the Radeon HD™ 4XXX series (7xx series GPUs), which does not use local memory and does not expose hardware LDS in OpenCL.
- One for the Radeon HD™ 5XXX series, which uses local memory (32 kB per compute unit for Evergreen series of GPUs).

Note that the extension `cl_amd_fp64` is currently supported only on RV770 and Cypress GPUs.

### 2.1 7xx Stream Kernel

For (width A / 4) iterations of the loop: Each thread reads four float4s from matrix A, then the corresponding four float4s from matrix B. Each thread then calculates the partial matrix multiplication and updates the partial sum. Thus, each thread computes four float4s (16 floating values of matrix C).

The number of global threads = {widthC / 4, heightC / 4}.



**Figure 1 Matrix Multiplication on the 7xx-Series GPUs**

## 2.2 Evergreen Stream Kernel

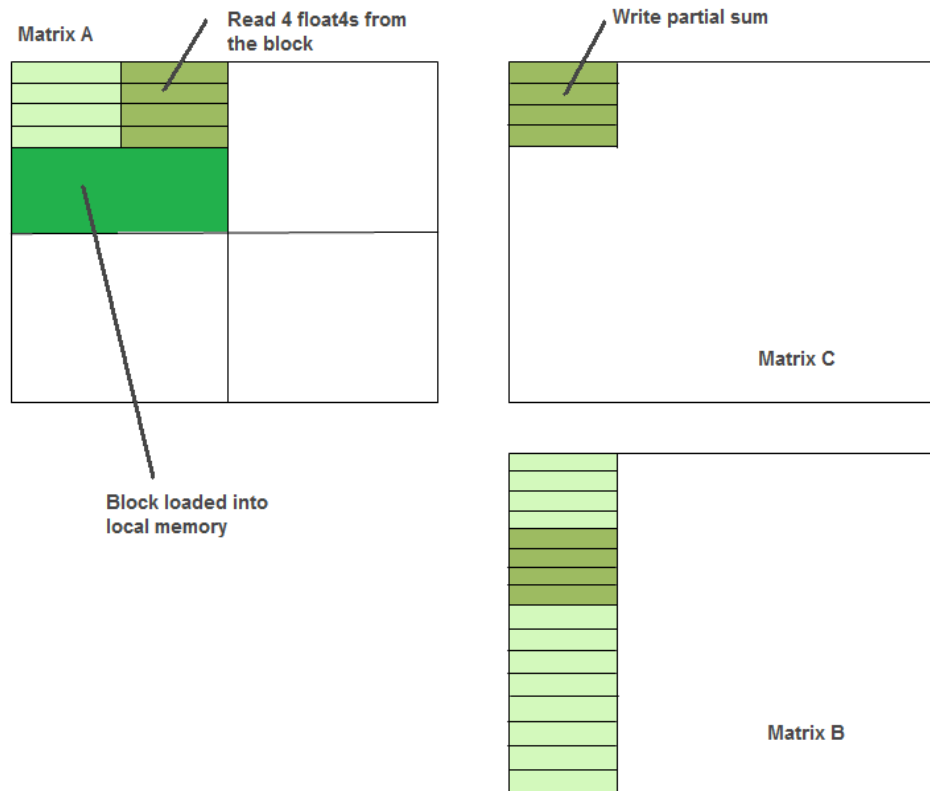
The access pattern for matrix A, above, is strided; for matrix B it is linear. Thus, a tile of matrix A is loaded into shared memory, which supports broadcasting of values.

A work-group loads a tile of matrix A into shared memory. The outer loop runs for the number of work-groups that can fit in matrix A's width. (See Figure 2.)

The inner loop consists of loading four float4 values from matrix A's tile, and the corresponding values from matrix B's global memory, to compute a partial sum.

Each work-item computes four float4s values, and writes to matrix C's global memory.

The number of global work-items = {widthC / 4, heightC / 4}.



**Figure 2 Matrix Multiplication on the Evergreen-Series GPUs**

#### 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](http://developer.amd.com/appsdk)  
Developing: [developer.amd.com/](http://developer.amd.com/)  
Support: [developer.amd.com/appsdksupport](http://developer.amd.com/appsdksupport)  
Forum: [developer.amd.com/openclforum](http://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.