AMD Accelerated Parallel Processing TECHNOLOGY

SAMPLE

Discrete Cosine Transform

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 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. `DCT`
   This initializes a random matrix of size 64x64 and gets a discreet cosine transform of the matrix.
2. `DCT -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 cpu or gpu.
-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	--width	Width of the input matrix.
-y	--height	Height of the input matrix.
-i	--iterations	Number of iterations for kernel execution.

2 Introduction

Discrete Cosine Transform (DCT) is a common transform for compressions of 1D and 2D signals such as audio, images (JPEG, see reference 1), and video. It also is used often for image compression JPEG[3]. A detailed explanation of Discrete Cosine Transform theory is explained in [2,3].

DCT is of a square matrix f(x,y) of size N is given by Equation 1:

$$C(u,v) = \alpha(u)\,\alpha(v) \sum_{x=0}^{N-1} \cos(\pi(2x+1)u / 2N)\ \{ \sum_{y=0}^{N-1} f(x,y)\cos(\pi(2x+1)v / 2N) \}$$

where

a (u) = squareroot(1/N) , u = 0
squareroot(2/N) , $u \neq 0$

From the above equation it can be observed that for N=8 we can form a set of basis functions to be used in the calculation of the DST. Thus, the equation above can be rewritten as Equation 2:

$$C(u,v) = A^T \times A$$

where matrix A stores the cosine values of Equation 1. And X denotes the input signal matrix f(x,y).

This sample implements a two-dimensional variation of the DCT that operates on blocks of size 8x8. This variation, known as DCT8x8, is used often in image and video compression.

$$\begin{vmatrix} g & a & B & c & g & d & e & f \\ g & c & E & -f & -g & -a & -b & -d \\ g & d & -e & -a & -g & f & b & c \\ g & f & -b & -d & g & c & -e & -a \\ g & -f & -b & d & g & -c & -e & a \\ g & -d & -e & a & -g & -f & b & -c \\ g & -c & E & f & -g & a & -b & d \\ g & -a & B & -c & g & -d & e & -f \end{vmatrix}$$

where:

a = cos(p/16) /4
b = cos(p / 8) /4/
c = cos(3p/ 16) /4/

d = cos(5p/16) /4
e = cos(3p/ 8) /4
f = cos(7p/ 16) /4

g = 1/squareroot(8)

3 Implementation Details

We partition the input matrix into blocks of size 8x8 and each block is performed the DCT using Equation 2. Here we do two matrix multiplications:

- B = ATX
- C = BA

where C is the DCT of the block X, which of size 8x8.

The input matrix is first divided into a set of blocks of size 8x8. Each block has a horizontal and a vertical index in the matrix. Each element of the input matrix has global indices and local indices that are relative to the block. For example, an element that belongs to the input matrix at (i,j) belongs to a block for which the indices are (i/8, j/8) and the local ids within the block are (i%8, j%8)

Thus: globalindices = (i,j) , localindices= (i%8, j%8); these belong to a block with indices(i/8, j/8). For each invocation of the thread (i,j) we calculate B(i,j), then calculate C(i,j) using the B(i,j). We ensure that all the elements for a block, B(i,j), are calculated before proceeding to calculate C(i,j) in each thread by using the local memory fence that is supported in OpenCL.

Calculate B(i,j) using A and X by:

$$B(i,j) = \sum_{k=0}^{7} A^T(i,k)*X(k,j)$$

Ensure that all B(i,j) for the block are calculate using barrier (`CLK_LOCAL_MEM_FENCE`)

$$C(i,j) = \sum_{k=0}^{7} B(i,k)*A(k,j)$$

Thus, C stores the DCT8x8 transform of the block X, which belongs to the input matrix.

4 References

1. http://en.wikipedia.org/wiki/JPEG#Discrete_cosine_transform
2. Syed Ali Khayam. “The Discrete Cosine Transform (DCT): Theory and Application”. ECE 802-602: Information theory and Coding, March 10th 2003.
3. http://en.wikipedia.org/wiki/Discrete_cosine_transform

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**