AMD **Accelerated** Parallel Processing TECHNOLOGY

# SAMPLE

# Calc Pie

## 1 Overview

### 1.1 Location

`$<AMDAPPSDKSamplesInstallPath>\samples\opencl\cl\2.0`

### 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 `$<AMDAPPSDKSamplesInstallPath>\samples\opencl\bin\x86\` for 32-bit builds, and `$<AMDAPPSDKSamplesInstallPath>\samples\opencl\bin\x86_64\` for 64-bit builds.

Ensure that the OpenCL 2.0 environment is installed.

Type the following command(s).

1. `CalcPie`
   This command runs the program with the default options.
2. `CalcPie -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 for kernel execution. |
| `-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 atomics in OpenCL 2.0. The sample calculates PI value using Monte Carlo analysis using the area of circle within a unit square. The host sends a large number of random points (x,y) within (1,1) to the kernel. The kernel computes the number of times these random points lie within the circle. Using this ratio, we calculate the PI value.

## 3 Algorithm

This sample demonstrates the usage of 2.0 atomics for calculating the value of PI using Monte Carlo Analysis. Specifically it uses the `atomic_fetch_add` API in OpenCL 2.0. The sample workflow is as follows:

1. Create two buffers RandomX and RandomY for generating two arrays of random points between (0,0) and (1,1).
2. Send these two arrays to the kernel.
3. The kernel computes, for each point, whether the point lies within the circle or outside the circle but inside the unit square.
4. All work items do this parallel and increments this counter using atomics.

This sample must be run in the OpenCL 2.0 environment.

## 4 Implementation

The given point is counted to be inside the circle or outside by computing the distance of this point from the center. If the distance is more than the radius, it is outside the circle.

**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