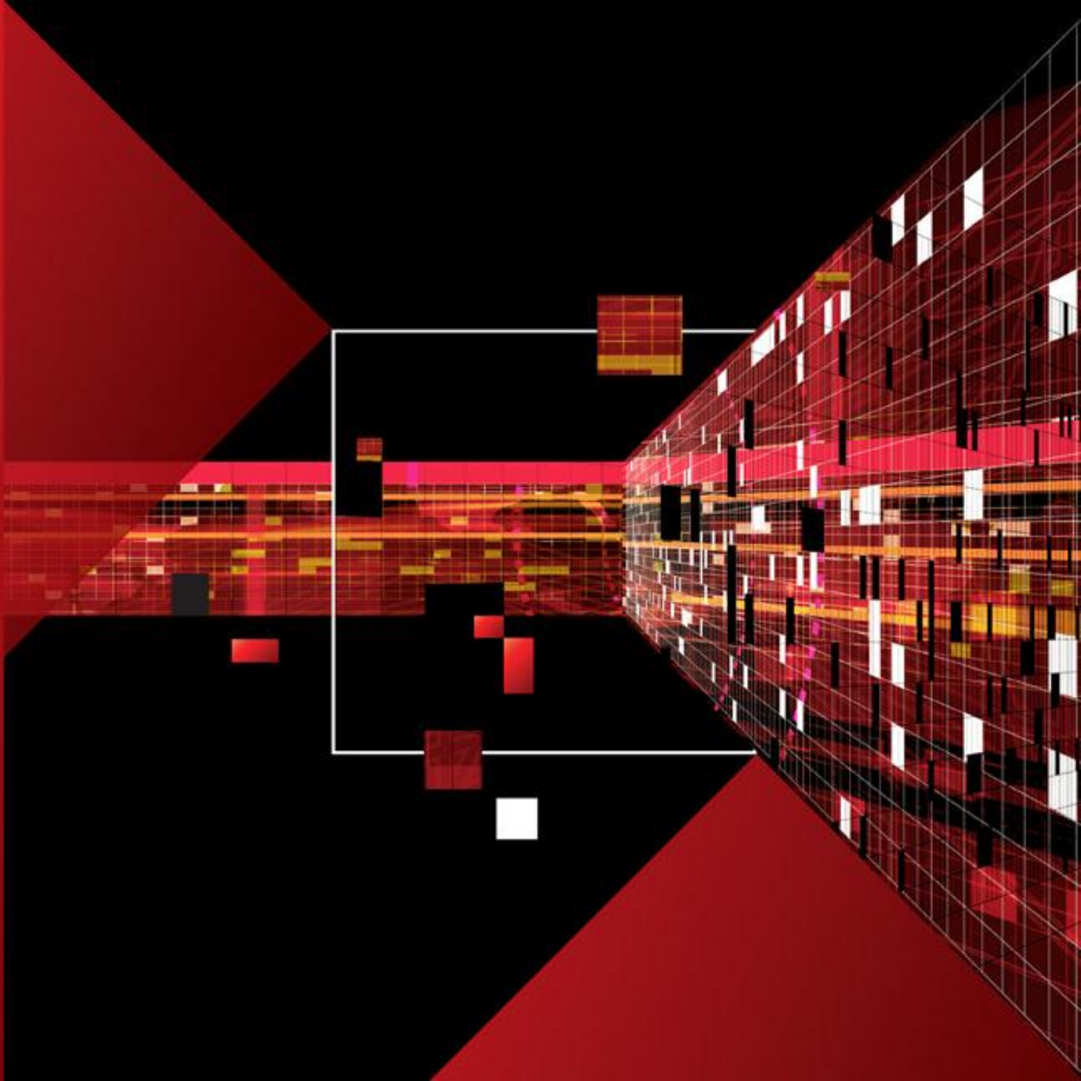
AMD

Fusion11

DEVELOPER SUMMIT



AMD

Fusion11

DEVELOPER SUMMIT

ADVANCED OPENCL™ DEBUGGING AND PROFILING – A CASE STUDY

Yaki Tebeka
Advanced Micro Devices
Fellow, Developer Tools

Budirijanto Purnomo
Advanced Micro Devices
Technical Lead, GPU Compute Tools

ABOUT OPENCL

OpenCL is FUN!

- New programming language
- Exposes the massively multithreaded GPU
- And the CPU
- A lot of horse power, optimized for parallel computing
- Order of magnitude performance improvement!

OPENCL DEBUGGING AND PROFILING MODEL

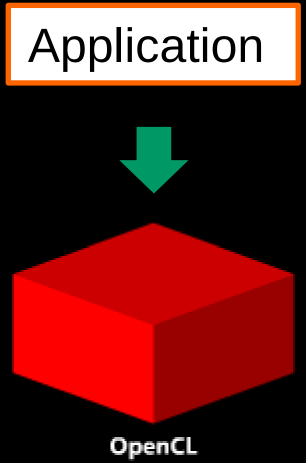
However,

- Debugging and profiling parallel processing applications is hard
- On-time delivery of robust (bug-free) OpenCL applications is challenging
- It is almost impossible to optimize an OpenCL based application to fully utilize the available parallel processing system resources

OPENCL DEBUGGING AND PROFILING MODEL

OpenCL is a “Black Box”

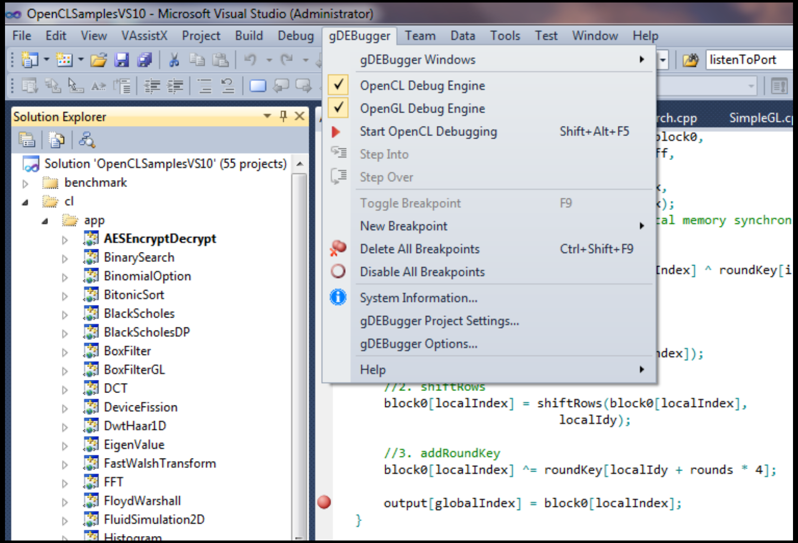
- The application enqueues OpenCL commands
- OpenCL’s runtime executes the commands
- The developer cannot
  - Debug the OpenCL kernels
  - See the execution details
  - View runtime loads



GDEBUGGER

gDEBugger™ for Microsoft Visual Studio®

- An OpenCL Debugger
  - API Level
  - Kernel Source Code
- An OpenGL API level debugger
- Integrated into Microsoft Visual Studio
- Provides the information a developer needs to find bugs and optimize the application's performance



```
                                  plock0,
                                  ff,

                                  k,
                                  k);
                                  cal memory synchron

                                  Index] ^ roundKey[i

                                  ndex]);
        //2. shiftRows
        block0[localIndex] = shiftRows(block0[localIndex],
                                       localIdy);

        //3. addRoundKey
        block0[localIndex] ^= roundKey[localIdy + rounds * 4];

        output[globalIndex] = block0[localIndex];
    }
```

DEMO

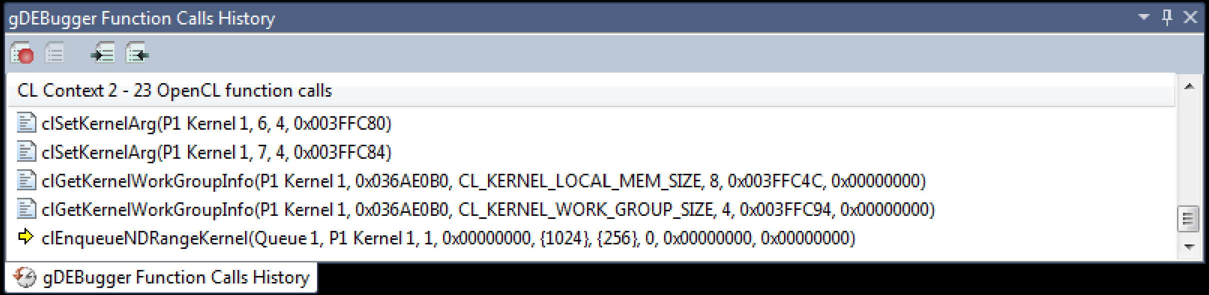
Uniform Random Noise Generator Sample

- A sample application provided with AMD APP SDK samples
- Each thread generates a uniform random deviation and applies to a pixel
- OpenCL kernel computes the deviation using a linear congruential generator proposed by Park and Miller
- The average of four neighboring pixels is used as the seed for each pixel

GDEBUGGER

API Calls History View

- Displayed a log of OpenCL and OpenGL API calls
- Call details are displayed in the Properties View



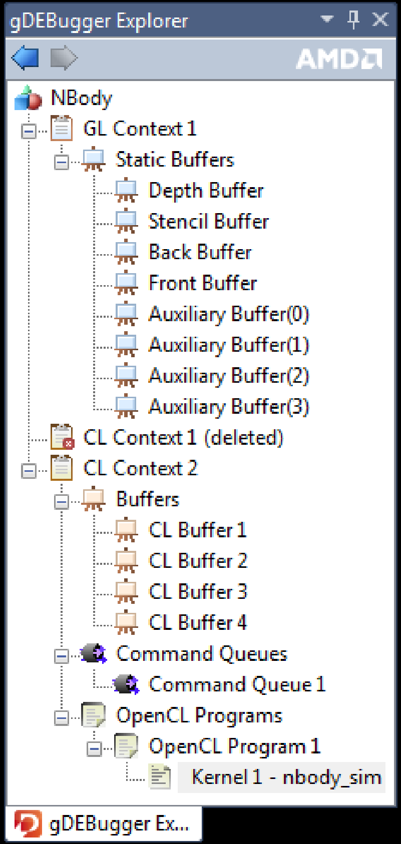
gDEBugger Function Calls History

```
CL Context 2 - 23 OpenCL function calls
clSetKernelArg(P1 Kernel 1, 6, 4, 0x003FFC80)
clSetKernelArg(P1 Kernel 1, 7, 4, 0x003FFC84)
clGetKernelWorkGroupInfo(P1 Kernel 1, 0x036AE0B0, CL_KERNEL_LOCAL_MEM_SIZE, 8, 0x003FFC4C, 0x00000000)
clGetKernelWorkGroupInfo(P1 Kernel 1, 0x036AE0B0, CL_KERNEL_WORK_GROUP_SIZE, 4, 0x003FFC94, 0x00000000)
clEnqueueNDRangeKernel(Queue 1, P1 Kernel 1, 1, 0x00000000, {1024}, {256}, 0, 0x00000000, 0x00000000)
```

GDEBUGGER

gDEBugger Explorer

- Displays OpenCL and OpenGL allocated objects
- Marks OpenGL-OpenCL shared contexts
- Focuses the GUI views on objects
- Double-click displays each object in the appropriate view



gDEBugger Explorer

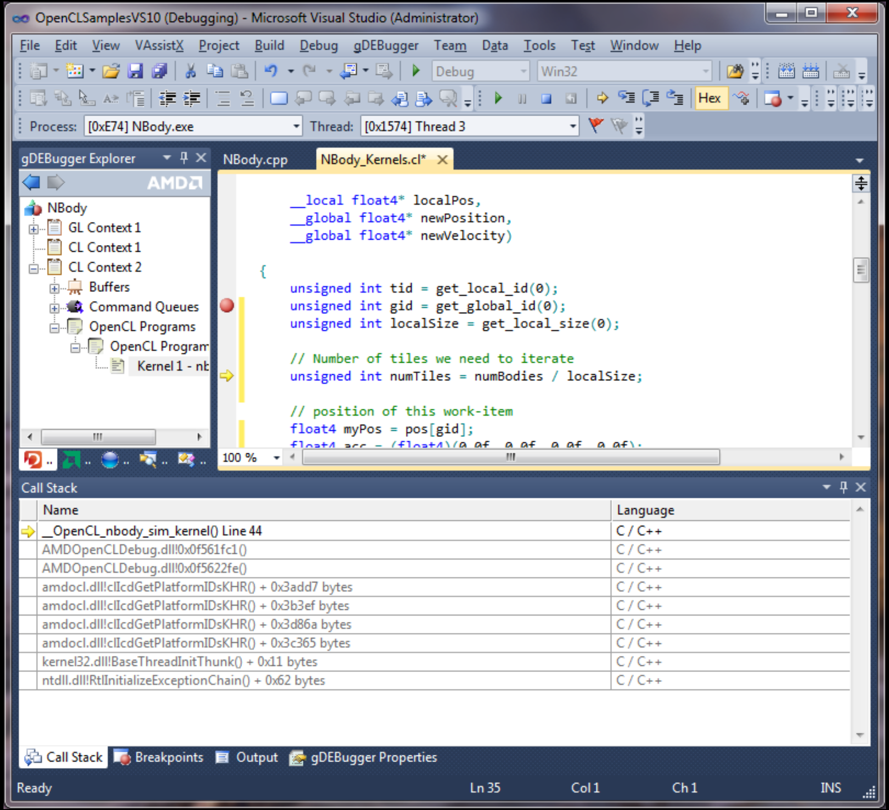
- NBody
  - GL Context 1
    - Static Buffers
      - Depth Buffer
      - Stencil Buffer
      - Back Buffer
      - Front Buffer
      - Auxiliary Buffer(0)
      - Auxiliary Buffer(1)
      - Auxiliary Buffer(2)
      - Auxiliary Buffer(3)
  - CL Context 1 (deleted)
  - CL Context 2
    - Buffers
      - CL Buffer 1
      - CL Buffer 2
      - CL Buffer 3
      - CL Buffer 4
    - Command Queues
      - Command Queue 1
    - OpenCL Programs
      - OpenCL Program 1
        - Kernel 1 - nbody_sim

gDEBugger Ex...

GDEBUGGER

Source Code and Call Stack

- Displays C, C++ and OpenCL C source code
- Enables setting source code breakpoints
- Displays a combined C, C++ and OpenCL C call stack



Process: [0xE74] NBody.exe

Thread: [0x1574] Thread 3

```
        __local float4* localPos,
        __global float4* newPosition,
        __global float4* newVelocity)

    {
        unsigned int tid = get_local_id(0);
        unsigned int gid = get_global_id(0);
        unsigned int localSize = get_local_size(0);

        // Number of tiles we need to iterate
        unsigned int numTiles = numBodies / localSize;

        // position of this work-item
        float4 myPos = pos[gid];
        float4 acc = (float4)(0.0f, 0.0f, 0.0f, 0.0f);
```

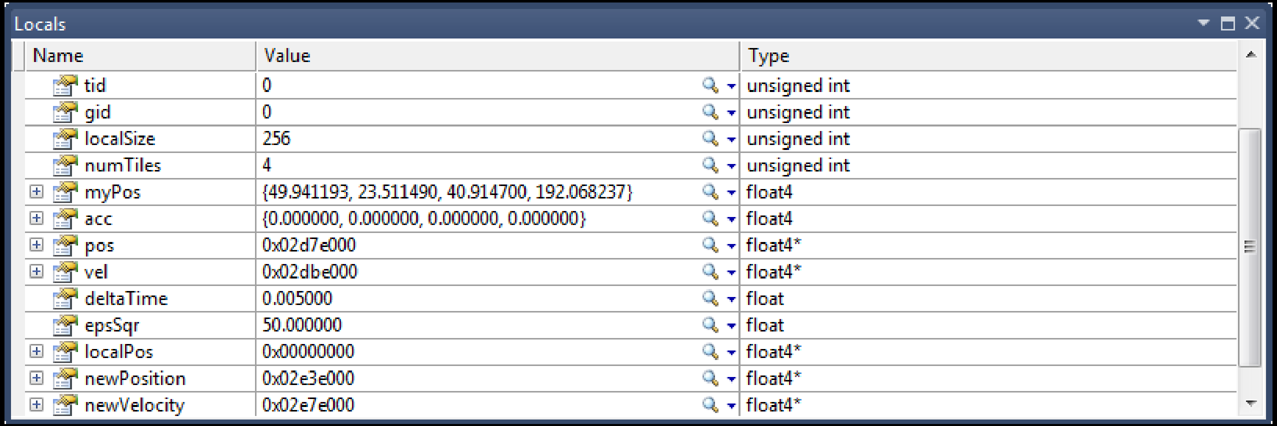
Call Stack

Name	Language
__OpenCL_nbody_sim_kernel() Line 44	C / C++
AMDOpenCLDebug.dll!0x0f561fc1()	C / C++
AMDOpenCLDebug.dll!0x0f5622fe()	C / C++
amdocl.dll!clIcdGetPlatformIDsKHR() + 0x3add7 bytes	C / C++
amdocl.dll!clIcdGetPlatformIDsKHR() + 0x3b3ef bytes	C / C++
amdocl.dll!clIcdGetPlatformIDsKHR() + 0x3d86a bytes	C / C++
amdocl.dll!clIcdGetPlatformIDsKHR() + 0x3c365 bytes	C / C++
kernel32.dll!BaseThreadInitThunk() + 0x11 bytes	C / C++
ntdll.dll!RtlInitializeExceptionChain() + 0x62 bytes	C / C++

GDEBUGGER

Watch views

- Displays OpenCL kernel’s variable values and types



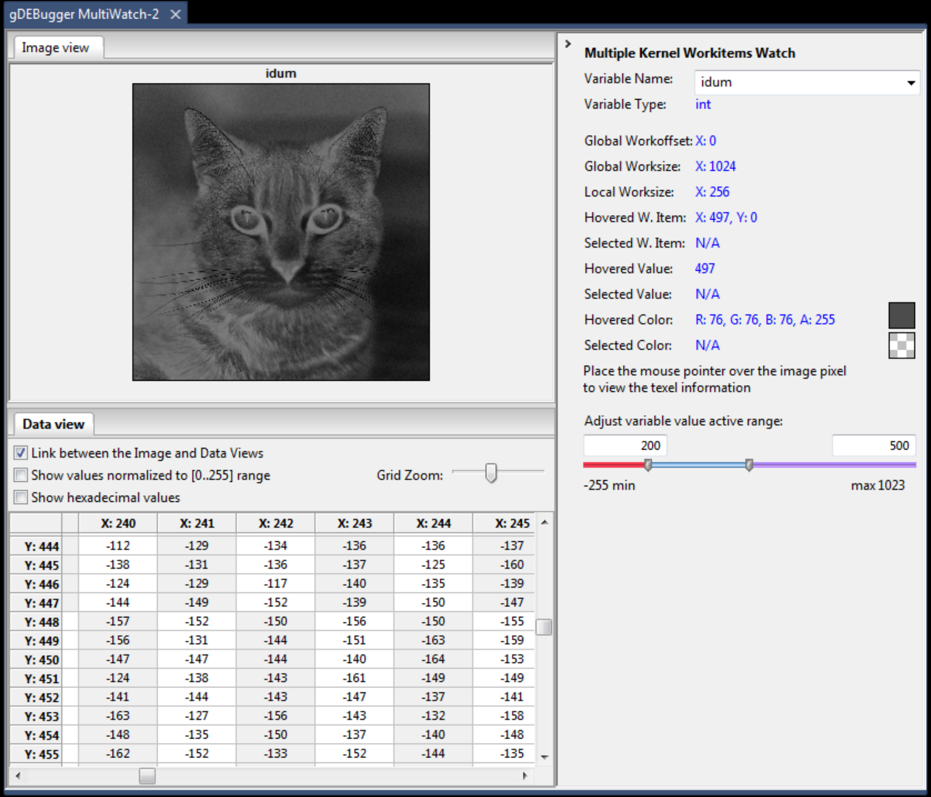
Locals

Name	Value	Type
tid	0	unsigned int
gid	0	unsigned int
localSize	256	unsigned int
numTiles	4	unsigned int
⊞ myPos	{49.941193, 23.511490, 40.914700, 192.068237}	float4
⊞ acc	{0.000000, 0.000000, 0.000000, 0.000000}	float4
⊞ pos	0x02d7e000	float4*
⊞ vel	0x02dbe000	float4*
deltaTime	0.005000	float
epsSqr	50.000000	float
⊞ localPos	0x00000000	float4*
⊞ newPosition	0x02e3e000	float4*
⊞ newVelocity	0x02e7e000	float4*

GDEBUGGER

Multi Watch View

- Displays the values of an OpenCL kernel variable across all work items and work groups
- The image view provides a graphics representation of the data (each pixel represents a single work item)



Multiple Kernel Workitems Watch

Variable Name: idum

Variable Type: int

Global Workoffset: X: 0

Global Worksize: X: 1024

Local Worksize: X: 256

Hovered W. Item: X: 497, Y: 0

Selected W. Item: N/A

Hovered Value: 497

Selected Value: N/A

Hovered Color: R: 76, G: 76, B: 76, A: 255

Selected Color: N/A

Place the mouse pointer over the image pixel
to view the texel information

Adjust variable value active range:

200 500

-255 min max 1023

☑ Link between the Image and Data Views

□ Show values normalized to [0..255] range

□ Show hexadecimal values

	X: 240	X: 241	X: 242	X: 243	X: 244	X: 245
Y: 444	-112	-129	-134	-136	-136	-137
Y: 445	-138	-131	-136	-137	-125	-160
Y: 446	-124	-129	-117	-140	-135	-139
Y: 447	-144	-149	-152	-139	-150	-147
Y: 448	-157	-152	-150	-156	-150	-155
Y: 449	-156	-131	-144	-151	-163	-159
Y: 450	-147	-147	-144	-140	-164	-153
Y: 451	-124	-138	-143	-161	-149	-149
Y: 452	-141	-144	-143	-147	-137	-141
Y: 453	-163	-127	-156	-143	-132	-158
Y: 454	-148	-135	-150	-137	-140	-148
Y: 455	-162	-152	-133	-152	-144	-135

PROFILING WITH AMD APP PROFILER

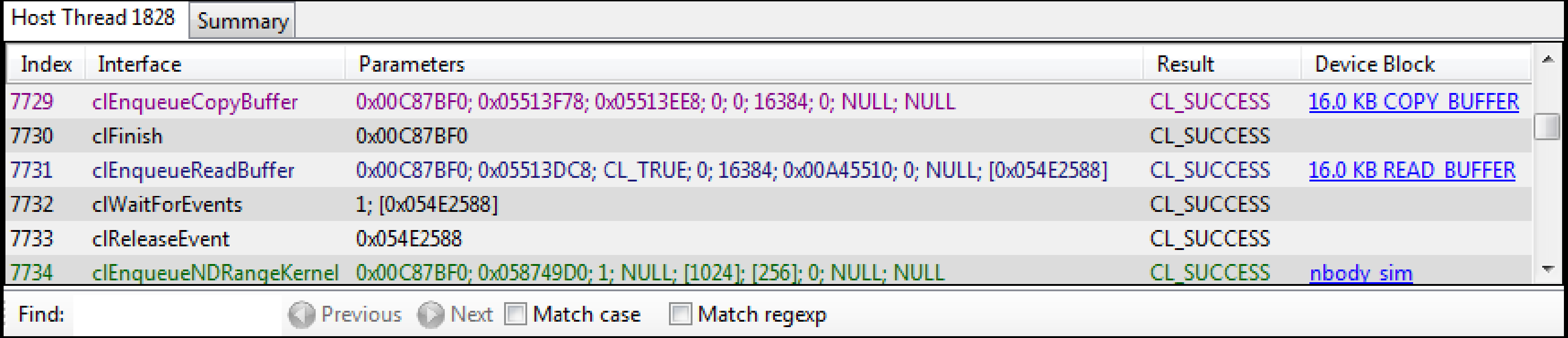
AMD APP PROFILER

- Analyzes and profiles OpenCL and DirectCompute application for AMD APUs and GPUs
- Integrates into Microsoft Visual Studio® 2008 and 2010
- Is available as a command line utility program for Windows and Linux platforms
- Does not require a custom driver
- Does not require source code or project modifications of the target application

PROFILING WITH AMD APP PROFILER

What can APP Profiler do for you?

- Analyze and Profile OpenCL applications
  - View API input arguments and output results
  - Find API hotspots
  - Determine top ten data transfer and kernel execution operations
  - Identify failed API calls, resource leaks and best practices

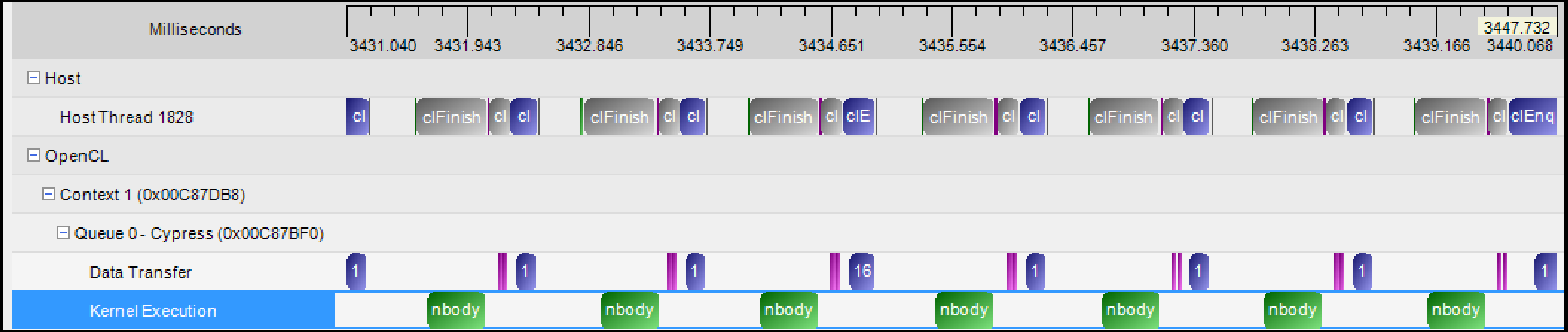


Index	Interface	Parameters	Result	Device Block
7729	clEnqueueCopyBuffer	0x00C87BF0; 0x05513F78; 0x05513EE8; 0; 0; 16384; 0; NULL; NULL	CL_SUCCESS	16.0 KB COPY_BUFFER
7730	clFinish	0x00C87BF0	CL_SUCCESS	
7731	clEnqueueReadBuffer	0x00C87BF0; 0x05513DC8; CL_TRUE; 0; 16384; 0x00A45510; 0; NULL; [0x054E2588]	CL_SUCCESS	16.0 KB READ_BUFFER
7732	clWaitForEvents	1; [0x054E2588]	CL_SUCCESS	
7733	clReleaseEvent	0x054E2588	CL_SUCCESS	
7734	clEnqueueNDRangeKernel	0x00C87BF0; 0x058749D0; 1; NULL; [1024]; [256]; 0; NULL; NULL	CL_SUCCESS	nbody_sim

PROFILING WITH AMD APP PROFILER

What can APP Profiler do for you?

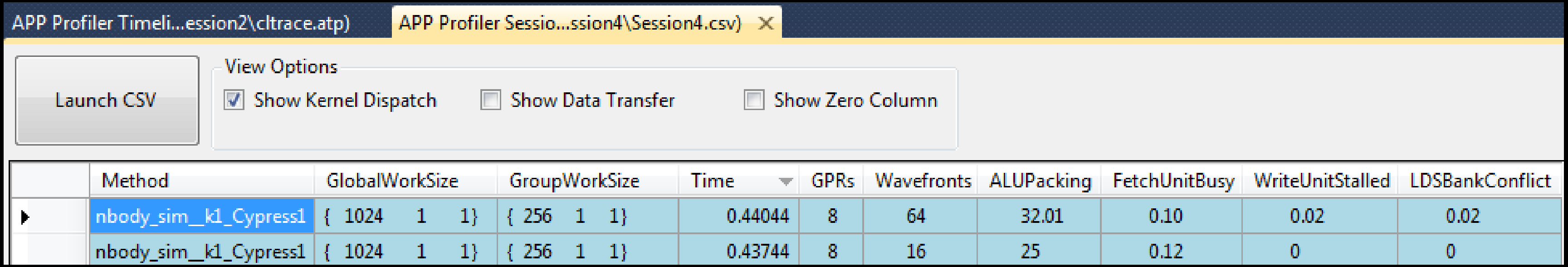
- Visualize OpenCL execution in a timeline chart
  - View number of OpenCL contexts and command queues created and the relationships between these items
  - View host and device execution operations
  - View data transfer operations
  - Determine proper synchronization and load balancing



PROFILING WITH AMD APP PROFILER

What can APP Profiler do for you?

- Analyze the OpenCL kernel execution for AMD Radeon GPUs
  - Collect GPU Performance Counters
    - The number of ALU, global and local memory instructions executed
    - GPU utilization and memory access characteristics
    - Shader Compiler VLIW packing efficiency
  - Show the kernel resource usages
  - View the AMD intermediate language (IL) and hardware disassembly (ISA)



APP Profiler Timeli...ession2\cltrace.atp)
APP Profiler Sessio...ssion4\Session4.csv) ✕
Launch CSV

View Options

☑ Show Kernel Dispatch ☐ Show Data Transfer ☐ Show Zero Column

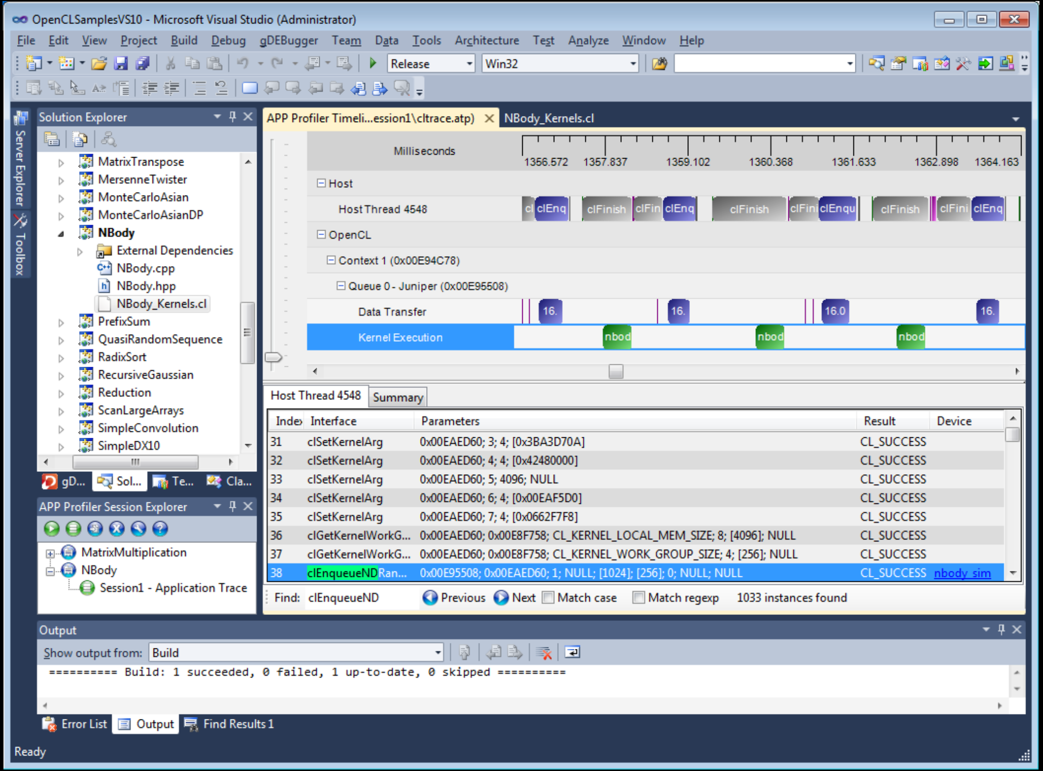
	Method	GlobalWorkSize	GroupWorkSize	Time	GPRs	Wavefronts	ALUPacking	FetchUnitBusy	WriteUnitStalled	LDSBankConflict
▸	nbody_sim__k1_Cypress1	{ 1024 1 1}	{ 256 1 1}	0.44044	8	64	32.01	0.10	0.02	0.02
	nbody_sim__k1_Cypress1	{ 1024 1 1}	{ 256 1 1}	0.43744	8	16	25	0.12	0	0

DEMO

Uniform Random Noise Generator Sample

- A sample application provided with AMD APP SDK samples
- Each thread generates a uniform random deviation and applies to a pixel
- OpenCL kernel computes the deviation using a linear congruential generator proposed by Park and Miller
- The average of four neighboring pixels is used as the seed for each pixel

PROFILING WITH AMD APP PROFILER | DEMO



Index	Interface	Parameters	Result	Device
31	clSetKernelArg	0x00EAED60; 3; 4; [0x3BA3D70A]	CL_SUCCESS	
32	clSetKernelArg	0x00EAED60; 4; 4; [0x42480000]	CL_SUCCESS	
33	clSetKernelArg	0x00EAED60; 5; 4096; NULL	CL_SUCCESS	
34	clSetKernelArg	0x00EAED60; 6; 4; [0x00EAF5D0]	CL_SUCCESS	
35	clSetKernelArg	0x00EAED60; 7; 4; [0x0662F7F8]	CL_SUCCESS	
36	clGetKernelWorkG...	0x00EAED60; 0x00E8F758; CL_KERNEL_LOCAL_MEM_SIZE; 8; [4096]; NULL	CL_SUCCESS	
37	clGetKernelWorkG...	0x00EAED60; 0x00E8F758; CL_KERNEL_WORK_GROUP_SIZE; 4; [256]; NULL	CL_SUCCESS	
38	clEnqueueNDRan...	0x00E95508; 0x00EAED60; 1; NULL; [1024]; [256]; 0; NULL; NULL	CL_SUCCESS	nbody_sim

Find: clEnqueueND Previous Next □ Match case □ Match regexp 1033 instances found

```
========== Build: 1 succeeded, 0 failed, 1 up-to-date, 0 skipped ==========
```

- Included with the AMD APP SDK v2 package
- Available as a separate download from http://developer.amd.com/AMDAPPProfiler

SUMMARY

gDEBugger for Microsoft Visual Studio

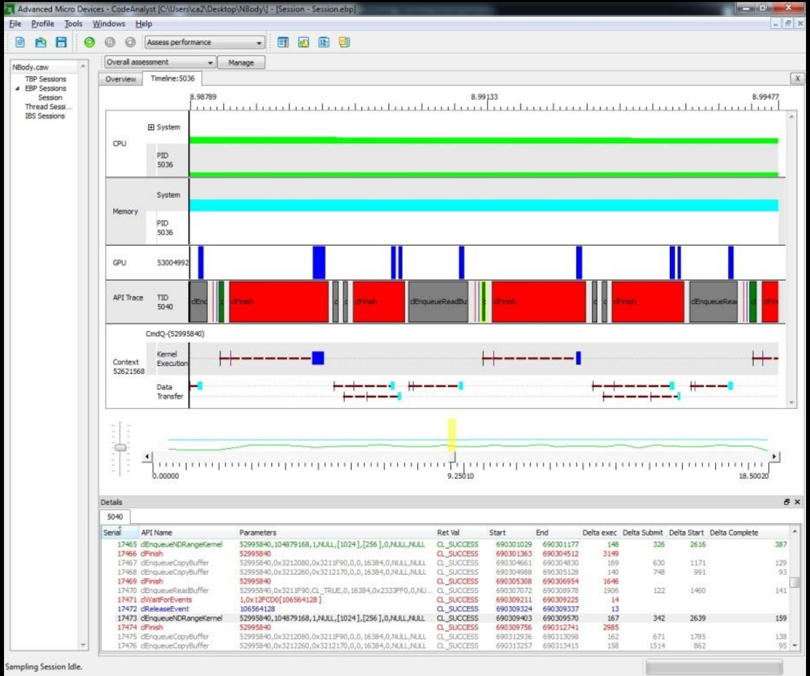
- API level debugging: view OpenCL buffers
- OpenCL kernel debugging on the GPU
  - Single step, set breakpoint and run to breakpoint
  - Inspect variables
- Call Stack and multi-watch view

AMD APP Profiler

- Trace OpenCL API calls and visualize OpenCL execution
- Determine kernel execution vs data transfer bottleneck
- Determine synchronization issues
- Identify failed API calls, resource leaks and best practices
- Collect and analyze GPU performance counters of an OpenCL kernel

OTHER AMD DEVELOPER TOOLS

- **CodeAnalyst**: a system wide profiler to analyze the performance of applications (OpenCL ,C, C++, Java, Fortran), drivers and system software on AMD CPU, GPU and APU.

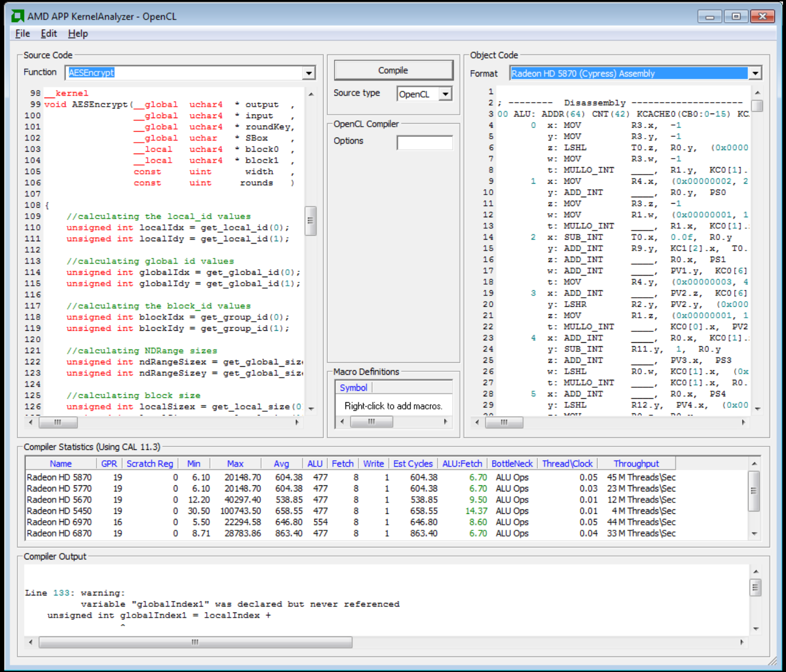


Serial	API Name	Parameters	Ret Val	Start	End	Delta exec	Delta Submit	Delta Start	Delta Complete
17465	dEnqueueNDRangeKernel	52995840,104879168,1,NULL,[1024],[256],0,NULL,NULL	CL_SUCCESS	690301029	690301177	148	326	2616	387
17466	dFinish	52995840	CL_SUCCESS	690301363	690304512	3149			
17467	dEnqueueCopyBuffer	52995840,0x3212080,0x3211F90,0,0,16384,0,NULL,NULL	CL_SUCCESS	690304661	690304830	169	630	1171	129
17468	dEnqueueCopyBuffer	52995840,0x3212260,0x3212170,0,0,16384,0,NULL,NULL	CL_SUCCESS	690304988	690305128	140	748	991	93
17469	dFinish	52995840	CL_SUCCESS	690305308	690306954	1646			
17470	dEnqueueReadBuffer	52995840,0x3211F90,CL_TRUE,0,16384,0x2333FF0,0,NU...	CL_SUCCESS	690307072	690308978	1906	122	1460	141
17471	dWaitForEvents	1,0x12FCD0[106564128]	CL_SUCCESS	690309211	690309225	14			
17472	dReleaseEvent	106564128	CL_SUCCESS	690309324	690309337	13			
17473	dEnqueueNDRangeKernel	52995840,104879168,1,NULL,[1024],[256],0,NULL,NULL	CL_SUCCESS	690309403	690309570	167	342	2639	159
17474	dFinish	52995840	CL_SUCCESS	690309756	690312741	2985			
17475	dEnqueueCopyBuffer	52995840,0x3212080,0x3211F90,0,0,16384,0,NULL,NULL	CL_SUCCESS	690312936	690313098	162	671	1785	138
17476	dEnqueueCopyBuffer	52995840,0x3212260,0x3212170,0,0,16384,0,NULL,NULL	CL_SUCCESS	690313257	690313415	158	1514	862	95

  - Optimize heterogeneous computing applications
  - Find performance hotspots and issues using AMD technology (time based profiling, event based profiling, instruction based profiling, thread profiling)
  - Tune both managed (Java) and native code (OpenCL, C/C++, Fortran)
  - Analyze programs on multi-core and NUMA platforms
  - Available as a standalone product (Windows, Linux) and a Visual Studio plug-in
  - http://developer.amd.com/CodeAnalyst/

OTHER AMD DEVELOPER TOOLS

- **AMD APP KernelAnalyzer**: a static analysis tool to compile, analyze and disassemble an OpenCL kernel for AMD GPU products



Function: AESEncrypt

Source type: OpenCL

Format: Radeon HD 5870 (Cypress) Assembly

```
__kernel
void AESEncrypt(__global  uchar4  * output  ,
                __global  uchar4  * input   ,
                __global  uchar4  * roundKey,
                __global  uchar   * SBox    ,
                __local   uchar4  * block0  ,
                __local   uchar4  * block1  ,
                const     uint      width   ,
                const     uint     rounds   )

{
    //calculating the local_id values
    unsigned int localIdx = get_local_id(0);
    unsigned int localIdy = get_local_id(1);

    //calculating global id values
    unsigned int globalIdx = get_global_id(0);
    unsigned int globalIdy = get_global_id(1);

    //calculating the block_id values
    unsigned int blockIdx = get_group_id(0);
    unsigned int blockIdy = get_group_id(1);

    //calculating NDRange sizes
    unsigned int ndRangeSizex = get_global_siz
    unsigned int ndRangeSizey = get_global_siz

    //calculating block size
    unsigned int localSizex = get_local_size(0
```

```

; --------  Disassembly --------------------
00 ALU: ADDR(64) CNT(42) KCACHE0(CB0:0-15) KC
     0  x: MOV        R3.x,  -1
        y: MOV        R3.y,  -1
        z: LSHL       T0.z,  R0.y,  (0x0000
        w: MOV        R3.w,  -1
        t: MULLO_INT  ____,  R1.y,  KC0[1].
     1  x: MOV        R4.x,  (0x00000002, 2
        y: ADD_INT    ____,  R0.y,  PS0
        z: MOV        R3.z,  -1
        w: MOV        R1.w,  (0x00000001, 1
        t: MULLO_INT  ____,  R1.x,  KC0[1].
     2  x: SUB_INT    T0.x,  0.0f,  R0.y
        y: ADD_INT    R9.y,  KC1[2].x,  T0.
        z: ADD_INT    ____,  R0.x,  PS1
        w: ADD_INT    ____,  PV1.y,  KC0[6]
        t: MOV        R4.y,  (0x00000003, 4
     3  x: ADD_INT    ____,  PV2.z,  KC0[6]
        y: LSHR       R2.y,  PV2.y,  (0x000
        z: MOV        R1.z,  (0x00000001, 1
        t: MULLO_INT  ____,  KC0[0].x,  PV2
     4  x: ADD_INT    ____,  R0.x,  KC0[1].
        y: SUB_INT    R11.y,  1,  R0.y
        z: ADD_INT    ____,  PV3.x,  PS3
        w: LSHL       R0.w,  KC0[1].x,  (0x
        t: MULLO_INT  ____,  KC0[1].x,  R0.
     5  x: ADD_INT    ____,  R0.x,  PS4
        y: LSHL       R12.y,  PV4.x,  (0x00
```

Compiler Statistics (Using CAL 11.3)

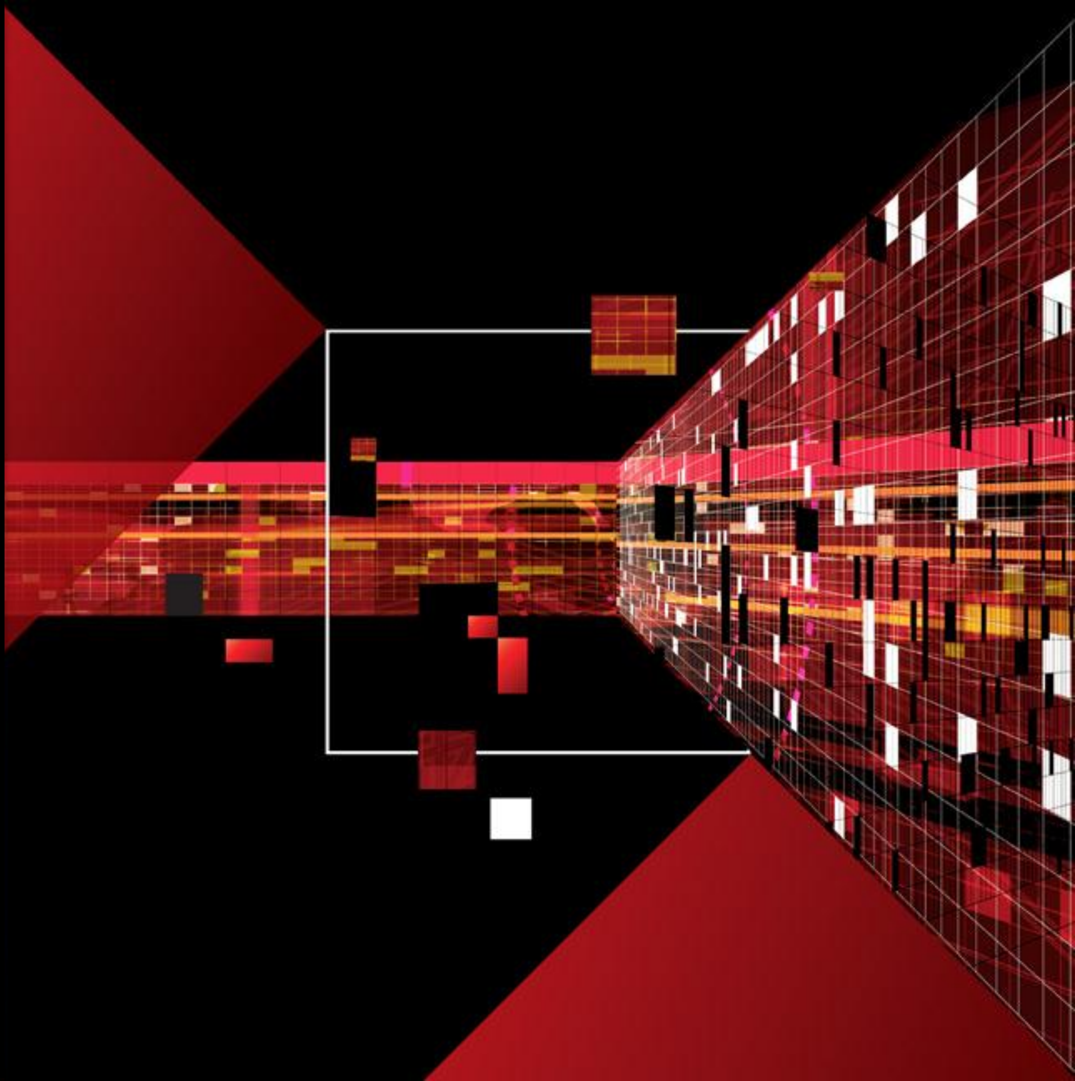
Name	GPR	Scratch Reg	Min	Max	Avg	ALU	Fetch	Write	Est Cycles	ALU:Fetch	BottleNeck	Thread\Clock	Throughput
Radeon HD 5870	19	0	6.10	20148.70	604.38	477	8	1	604.38	6.70	ALU Ops	0.05	45 M Threads\Sec
Radeon HD 5770	19	0	6.10	20148.70	604.38	477	8	1	604.38	6.70	ALU Ops	0.03	23 M Threads\Sec
Radeon HD 5670	19	0	12.20	40297.40	538.85	477	8	1	538.85	9.50	ALU Ops	0.01	12 M Threads\Sec
Radeon HD 5450	19	0	30.50	100743.50	658.55	477	8	1	658.55	14.37	ALU Ops	0.01	4 M Threads\Sec
Radeon HD 6970	16	0	5.50	22294.58	646.80	554	8	1	646.80	8.60	ALU Ops	0.05	44 M Threads\Sec
Radeon HD 6870	19	0	8.71	28783.86	863.40	477	8	1	863.40	6.70	ALU Ops	0.04	33 M Threads\Sec

Compiler Output

```
Line 133: warning:
          variable "globalIndex1" was declared but never referenced
    unsigned int globalIndex1 = localIndex +
                ^
```

- Compile and analyze for multiple Catalyst driver and GPU device targets
- View kernel compilation warning and error messages
- View AMD Intermediate Language (IL) and hardware disassembly (ISA) code
- View various statistics generated by analyzing the ISA code
- http://developer.amd.com/AMDAPPKernelAnalyzer/

QUESTIONS



Disclaimer & Attribution

The information presented in this document is for informational purposes only and may contain technical inaccuracies, omissions and typographical errors.

The information contained herein is subject to change and may be rendered inaccurate for many reasons, including but not limited to product and roadmap changes, component and motherboard version changes, new model and/or product releases, product differences between differing manufacturers, software changes, BIOS flashes, firmware upgrades, or the like. There is no obligation to update or otherwise correct or revise this information. However, we reserve the right to revise this information and to make changes from time to time to the content hereof without obligation to notify any person of such revisions or changes.

NO REPRESENTATIONS OR WARRANTIES ARE MADE WITH RESPECT TO THE CONTENTS HEREOF AND NO RESPONSIBILITY IS ASSUMED FOR ANY INACCURACIES, ERRORS OR OMISSIONS THAT MAY APPEAR IN THIS INFORMATION.

ALL IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR ANY PARTICULAR PURPOSE ARE EXPRESSLY DISCLAIMED. IN NO EVENT WILL ANY LIABILITY TO ANY PERSON BE INCURRED FOR ANY DIRECT, INDIRECT, SPECIAL OR OTHER CONSEQUENTIAL DAMAGES ARISING FROM THE USE OF ANY INFORMATION CONTAINED HEREIN, EVEN IF EXPRESSLY ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

AMD, the AMD arrow logo, and combinations thereof are trademarks of Advanced Micro Devices, Inc. All other names used in this presentation are for informational purposes only and may be trademarks of their respective owners.

OpenCL and OpenCL logo are trademarks of Apple Inc. used by permission by Khronos.

Microsoft and Visual Studio are registered trademarks of Microsoft Corporation in the United States and/or other jurisdictions.