CUDA and OpenCL --- Development Interfaces for Multicore Programming

Dr. Jun Ni, Ph.D.
Associate Professor of Radiology, Biomedical Engineering, Mechanical Engineering, Computer Science
The University of Iowa, Iowa City, Iowa, USA

Outline

- CUDA, Navida-based Programming environment for GPGPU
- OpenCL, open source programming environment for multicore processor systems for Cell/BE and GPU

Introduction to CUDA

- **CUDA** (an acronym for **Compute Unified Device Architecture**)
  - a parallel computing architecture developed by NVIDIA
- CUDA is the computing engine in NVIDIA graphics processing units (GPUs)
  - accessible to software developers through industry standard programming languages
- Programmers use 'C for CUDA' (C with NVIDIA extensions)
  - compiled through a PathScale Open64 C compiler
  - to code algorithms for execution on the GPU

Introduction to CUDA

- CUDA architecture supports a range of computational interfaces
  - OpenCL
  - DirectCompute
- Third party wrappers
  - Python
  - Fortran
  - Java
  - Matlab
- The latest drivers all contain the necessary CUDA components CUDA works with all NVIDIA GPUs G8X series onwards:
  - GeForce
  - Quadro
  - Tesla

Introduction to CUDA

- CUDA gives developers access to the native instruction set and memory of the parallel computational elements in CUDA GPUs.

Introduction to CUDA

- Using CUDA, the latest NVIDIA GPUs effectively become open architectures like CPUs.
- Unlike CPUs, GPUs have a parallel "many-core" architecture, each core capable of running thousands of threads simultaneously - if an application is suited to this kind of an architecture, the GPU can offer large performance benefits
- Multi-threads computing becomes more efficient within multicore architecture

Introduction to CUDA

- In the computer gaming industry, in addition to graphics rendering, graphics cards are used in game physics calculations (physical effects like debris, smoke, fire, fluids)
  - PhysX
  - Bullet
- CUDA has also been used to accelerate non-graphical applications by an order of magnitude or more
  - computational biology
  - Cryptography
  - Other fields

Introduction to CUDA

- Example:
  - BOINC distributed computing client
- CUDA provides both a low level API and a higher level API
- The initial CUDA SDK was made public on 15 February 2007, for Microsoft Windows and Linux
- Mac OS X support was later added in version 2.0. which supersedes the beta released February 14, 2008

Advantages

- CUDA has several advantages over traditional general purpose computation on GPUs (GPGPU) using graphics APIs.
  - Scattered reads – code can read from arbitrary addresses in memory.
  - Shared memory – CUDA exposes a fast shared memory region (16KB in size) that can be shared amongst threads.
  - This can be used as a user-managed cache, enabling higher bandwidth than is possible using texture lookups
  - Faster downloads and readbacks to and from the GPU
  - Full support for integer and bitwise operations, including integer texture lookups.

Limitations and Challenges

- It uses a recursion-free, function-pointer-free subset of the C language, plus some simple extensions.
- However, a single process must run spread across multiple disjoint memory spaces, unlike other C language runtime environments.
- Texture rendering is not supported.
- For double precision (only supported in newer GPUs like GTX 260[12]) there are some deviations from the IEEE 754 standard: round-to-nearest-even is the only supported rounding mode for reciprocal, division, and square root.

Limitations and Challenges

- In single precision, Denormals and signalling NaNs are not supported
- Only two IEEE rounding modes are supported (chop and round-to-nearest even)
- Those are specified on a per-instruction basis rather than in a control word
- Precision of division/square root is slightly lower than single precision.

Limitations and Challenges

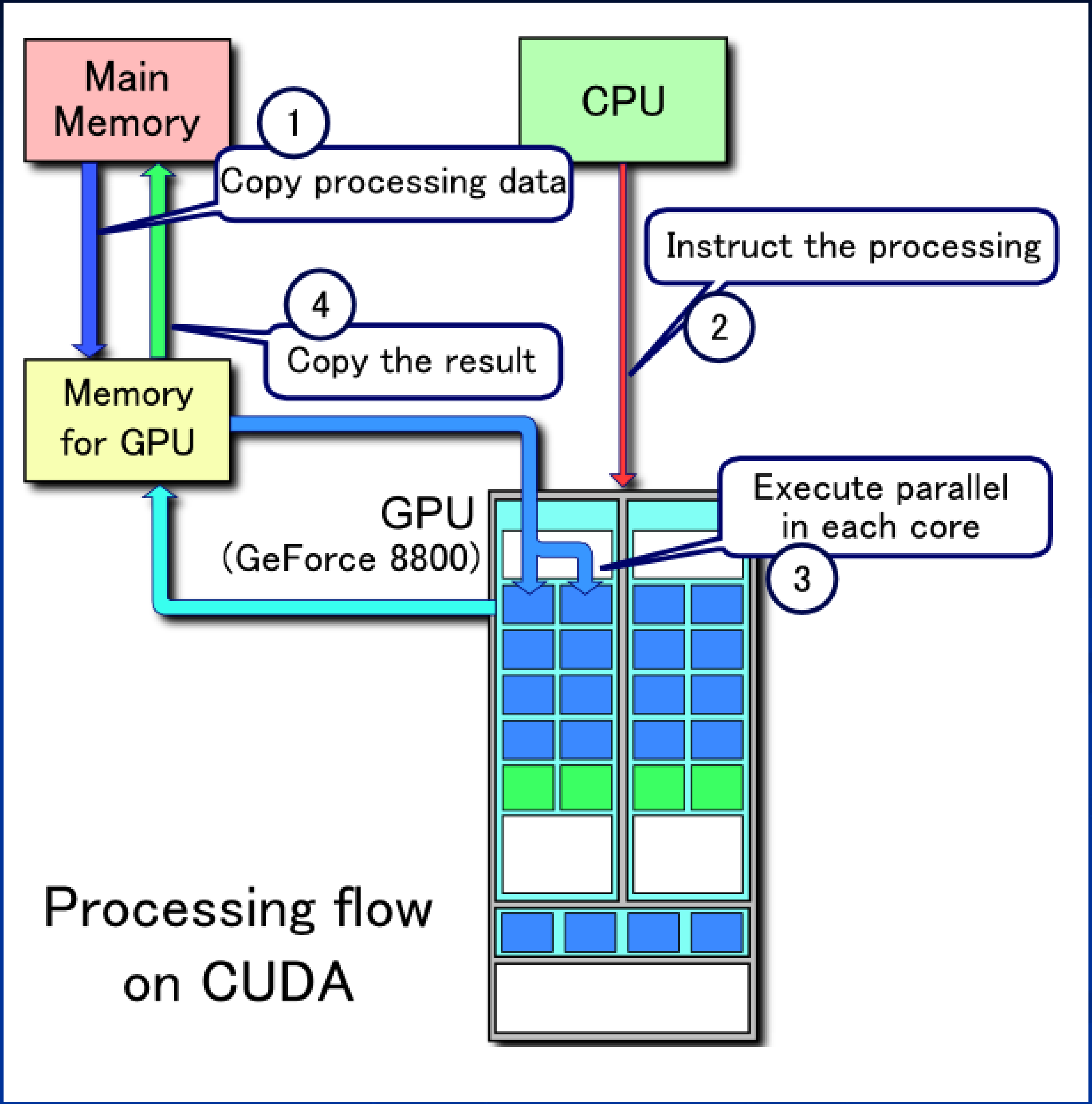
- The bus bandwidth and latency between the CPU and the GPU may be a bottleneck.
- Threads should be running in groups of at least 32 for best performance, with total number of threads numbering in the thousands.

Limitations and Challenges

- Branches in the program code do not impact performance significantly, provided that each of 32 threads takes the same execution path; the SIMD execution model becomes a significant limitation for any inherently divergent task (e.g. traversing a space partitioning data structure during raytracing).

Limitations and Challenges

- Unlike OpenCL, CUDA-enabled GPUs are only available from NVIDIA (GeForce 8 series and above, Quadro and Tesla).
- AMD/ATI only supports GPGPU on its high-end chips


Main Memory
CPU
1
Copy processing data
Instruct the processing
2
4
Copy the result
Memory for GPU
GPU (GeForce 8800)
Execute parallel in each core
3
Processing flow on CUDA

Example C++ loads a texture from an image into an array on the GPU:

```
cudaArray* cu_array; texture<float, 2> tex;
// Allocate array
cudaChannelFormatDesc description = cudaCreateChannelDesc<float>();
cudaMallocArray(&cu_array, &description, width, height); // Copy image data to array
cudaMemcpy(cu_array, image, width*height*sizeof(float), cudaMemcpyHostToDevice);
// Bind the array to the texture
cudaBindTextureToArray(tex, cu_array);
// Run kernel
dim3 blockDim(16, 16, 1);
dim3 gridDim(width / blockDim.x, height / blockDim.y, 1);
kernel<<< gridDim, blockDim, 0 >>>(d_odata, width, height);
cudaUnbindTexture(tex);
__global__ void kernel(float* odata, int height, int width) {
unsigned int x = blockIdx.x*blockDim.x + threadIdx.x;
unsigned int y = blockIdx.y*blockDim.y + threadIdx.y;
float c = tex2D(tex, x, y);
odata[y*width+x] = c;
 }
```

Example in Python: the product of two arrays on the GPU

```
import pycuda.driver as drv
import numpy
import pycuda.autoinit mod = drv.SourceModule(""" __global__
    void multiply_them(float *dest, float *a, float *b)
{ const int i = threadIdx.x; dest[i] = a[i] * b[i];
} """)
multiply_them = mod.get_function("multiply_them")
a = numpy.random.randn(400).astype(numpy.float32)
b = numpy.random.randn(400).astype(numpy.float32)
dest = numpy.zeros_like(a)
multiply_them( drv.Out(dest), drv.In(a), drv.In(b), block=(400,1,1))
print dest-a*b
```

Example in Python: matrix multiplication in GPU

```
import numpy
from pycublas import CUBLASMatrix
A = CUBLASMatrix(
   numpy.mat([[1,2,3],[4,5,6]],numpy.float32) )
B = CUBLASMatrix(
   numpy.mat([[2,3],[4,5],[6,7]],numpy.float32) )
C = A*B print C.np_mat()
```

Language Bindings

- Unlike OpenCL, CUDA-enabled GPUs are only available from NVIDIA (GeForce 8 series and above, Quadro and Tesla).
- AMD/ATI only supports GPGPU on its high-end chips

Limitations and Challenges

- Python - PyCUDA
- Java - jCUDA, JCublas, JCufft
- .NET - CUDA.NET

References

- NVIDIA Clears Water Muddied by Larrabee Shane McGlaun (Blog) - August 5, 2008 - DailyTech
- First OpenCL demo on a GPU at YouTube (requires Adobe Flash)
- DirectCompute Ocean Demo Running on NVIDIA CUDA-enabled GPU at YouTube (requires Adobe Flash)
- Giorgos Vasiliadis, Spiros Antonatos, Michalis Polychronakis, Evangelos P. Markatos and Sotiris Ioannidis (September 2008, Boston, MA, USA). "Gnort: High Performance Network Intrusion Detection Using Graphics Processors" (PDF). *Proceedings of the 11th International Symposium On Recent Advances In Intrusion Detection (RAID).* http://www.ics.forth.gr/dcs/Activities/papers/gnort.raid08.pdf.
- Schatz, M.C., Trapnell, C., Delcher, A.L., Varshney, A. (2007). "High-throughput sequence alignment using Graphics Processing Units.". *BMC Bioinformatics* **8:474**: 474. doi:10.1186/1471-2105-8-474. http://www.biomedcentral.com/1471-2105/8/474.
- Manavski, Svetlin A.; Giorgio Valle (2008). "CUDA compatible GPU cards as efficient hardware accelerators for Smith-Waterman sequence alignment". *BMC Bioinformatics* **9(Suppl 2):S10**: S10. doi:10.1186/1471-2105-9-S2-S10. http://www.biomedcentral.com/1471-2105/9/S2/S10.

References

- Pyrit - Google Code http://code.google.com/p/pyrit/
- Use your NVIDIA GPU for scientific computing, BOINC official site (December 18 2008)
- NVIDIA CUDA Software Development Kit (CUDA SDK) - Release Notes Version 2.0 for MAC OSX
- CUDA 1.1 - Now on Mac OS X- (Posted on Feb 14, 2008)
- Silberstein, Mark (2007). "Efficient computation of Sum-products on GPUs" (PDF). http://www.technion.ac.il/~marks/docs/SumProductPaper.pdf.
- [http://www.herikstad.net/2009/05/cuda-and-double-precision-floating.html CUDA and double precision floating point numbers]
- "CUDA-Enabled Products". *CUDA Zone*. NVIDIA Corporation. http://www.nvidia.com/object/cuda_learn_products.html. Retrieved 2008-11-03.
- CUDA-Enabled GPU Products
- http://www.nvidia.com/object/fermi_architecture.html The Next Generation CUDA Architecture, Code Named Fermi .

Useful Links

- Nvidia CUDA *official website*
- Nvidia Nexus, NVIDIA.
- Nvidia CUDA developer registration for professional developers and researchers
- Nvidia CUDA GPU Computing developer forums
- A conversation with Jen-Hsun Huang, CEO Nvidia Charlie Rose, February 5, 2009
- http://www.gpu4vision.org Scientific Publications, Videos and Software using CUDA
- Beyond3D – Introducing CUDA Nvidia's Vision for GPU Computing March 10, 2007
- University of Illinois Nvidia CUDA Course taught by Wen-mei Hwu and David Kirk, Spring 2009
- CUDA: Breaking the Intel & AMD Dominance
- A CUDA-based Engine for MATLAB
- NVidia CUDA Tutorial Slides (from DoD HPCMP2009)
- Ascalaph Liquid GPU molecular dynamics.
- CUDA implementation for multi-core processors

Useful Links

- Integrate CUDA with Visual C++, September 26, 2008
- CUDA.NET - .NET library for CUDA, Linux/Windows compliant
- Using NVIDIA GPU for scientific computing with BOINC software
- CUDA.CS.MSU.SU Russian CUDA developer community
- GPUmat: MATLAB(R) GPU Toolbox based on CUDA
- Enable Intellisense for CUDA in Visual Studio 2008, April 29, 2009
- CUDA Tutorials for high performance computing
- An introduction to CUDA (French)
- NVidia CUDA Tutorial & Examples (from ISC2009)
- GPUBrasil.com, First website on GPGPU in Portuguese
- [2], Implementation of CUDA in Cloth simulation
- DDJ CUDA series, First in the Doctor Dobb's Journal series teaching CUDA (over 13 articles by Rob Farber)

OpenCL – Cross-platform Intrerface for Multicore Programming

- **OpenCL** (**Open** **C**omputing **L**anguage)
  - a framework for writing programs that execute across heterogeneous platforms consisting of CPUs, GPUs, and other processors.
  - OpenCL includes a language (based on C99) for writing *kernels* (functions that execute on OpenCL devices)
  - APIs are used to define and then control the platforms.
  - OpenCL provides parallel computing using task-based and data-based parallelism.

OpenCL – Cross-platform Intrerface for Multicore Programming

- OpenCL is analogous to the open industry standards OpenGL and OpenAL, for 3D graphics and computer audio, respectively.
- OpenCL extends the power of the GPU beyond graphics (GPGPU).
- OpenCL is managed by the non-profit technology consortium Khronos Group.

History of OpenCL Developments

- OpenCL was initially developed by Apple Inc., which holds trademark rights, and refined into an initial proposal in collaboration with technical teams at AMD, IBM, Intel, and Nvidia.
- Apple submitted this initial proposal to the Khronos Group.

History of OpenCL Developments

- On June 16, 2008 the Khronos Compute Working Group was formed with representatives from CPU, GPU, embedded-processor, and software companies.
- This group worked for five months to finish the technical details of the specification for OpenCL 1.0 by November 18, 2008. This technical specification was reviewed by the Khronos members and approved for public release on December 8, 2008.

History of OpenCL Developments

- OpenCL 1.0 has been released with Mac OS X v10.6 ("Snow Leopard"). According to an Apple press release:
- Snow Leopard further extends support for modern hardware with Open Computing Language (OpenCL), which lets any application tap into the vast gigaflops of GPU computing power previously available only to graphics applications.
- OpenCL is based on the C programming language and has been proposed as an open standard.

History of OpenCL Developments

- AMD has decided to support OpenCL (and DirectX 11) instead of the now deprecated Close to Metal in its Stream framework.
- RapidMind announced their adoption of OpenCL underneath their development platform, in order to support GPUs from multiple vendors with one interface.

History of OpenCL Developments

- On December 9, 2008, Nvidia announced its intention to add full support for the OpenCL 1.0 specification to its GPU Computing Toolkit
- On October 30, 2009, IBM released its first OpenCL implementation
- OpenCL specification is under development at Khronos, which is open to any interested company to join.

History of OpenCL Developments

- On December 10, 2008, AMD and Nvidia held the first public OpenCL demonstration, a 75-minute presentation at Siggraph Asia 2008.
  - AMD showed a CPU-accelerated OpenCL demo explaining the scalability of OpenCL on one or more cores while Nvidia showed a GPU-accelerated demo.
- On March 26, 2009, at GDC 2009, AMD and Havok demonstrated the first working implementation for OpenCL accelerating Havok Cloth on AMD Radeon HD 4000 series GPU.
- On April 20, 2009, Nvidia announced the release of its OpenCL driver and SDK to developers participating in its OpenCL Early Access Program.

History of OpenCL Developments

- On August 5, 2009, AMD unveiled the first development tools for its OpenCL platform as part of its ATI Stream SDK v2.0 Beta Program.
- On August 28, 2009, Apple released Mac OS X Snow Leopard, which contains a full implementation of OpenCL.
- OpenCL in Snow Leopard will initially be supported on
  - ATI Radeon HD 4850, ATI Radeon HD 4870 and NVIDIA's Geforce 8600M GT, GeForce 8800 GS, GeForce 8800 GT, GeForce 8800 GTS, Geforce 9400M, GeForce 9600M GT, GeForce GT 120, GeForce GT 130, GeForce GTX 285, Quadro FX 4800, and Quadro FX 5600.
- On September 28, 2009, NVIDIA released its own OpenCL drivers and SDK implementation.

History of OpenCL Developments

- On October 13, 2009, AMD released the fourth beta of the ATI Stream SDK 2.0, which provides a complete OpenCL implementation on both R700/R800 GPUs and SSE3 capable CPUs. The SDK is available for both Linux and Windows.
- On November 26, 2009, NVIDIA released drivers for OpenCL 1.0 (rev 48).
- The Apple, Nvidia, RapidMind and Mesa Gallium3D implementations of OpenCL are all based on the LLVM Compiler technology and use the Clang Compiler as its frontend.

History of OpenCL Developments

- On December 10, 2009, VIA released their first product supporting OpenCL 1.0 - ChromotionHD 2.0 video processor included in VN1000 chipset.
- On December 21, 2009, AMD released the production version of the ATI Stream SDK 2.0 which provides OpenCL 1.0 support for R800 GPUs and beta support for R700 GPUs.

Example of FFT

```
// create a compute context with GPU device
context = clCreateContextFromType(0, CL_DEVICE_TYPE_GPU, NULL,
    NULL, NULL);
// create a work-queue
queue = clCreateWorkQueue(context, NULL, NULL, 0);
// allocate the buffer memory objects
memobjs[0] = clCreateBuffer(context, CL_MEM_READ_ONLY |
    CL_MEM_COPY_HOST_PTR, sizeof(float)*2*num_entries, srcA);
 memobjs[1] = clCreateBuffer(context, CL_MEM_READ_WRITE,
    sizeof(float)*2*num_entries, NULL);
// create the compute program program =
    clCreateProgramFromSource(context, 1, &fft1D_1024_kernel_src, NULL);
// build the compute
program executable
clBuildProgramExecutable(program, false, NULL, NULL);
```

Example of FFT

```
// create the compute
kernel kernel = clCreateKernel(program, "fft1D_1024");
// create N-D range object with work-item dimensions
global_work_size[0] = num_entries;
local_work_size[0] = 64;
range = clCreateNDRangeContainer(context, 0, 1, global_work_size, local_work_size);
// set the args values
clSetKernelArg(kernel, 0, (void *)&memobjs[0], sizeof(cl_mem), NULL);
clSetKernelArg(kernel, 1, (void *)&memobjs[1], sizeof(cl_mem), NULL);
clSetKernelArg(kernel, 2, NULL, sizeof(float)*(local_work_size[0]+1)*16, NULL);
clSetKernelArg(kernel, 3, NULL, sizeof(float)*(local_work_size[0]+1)*16, NULL);
// execute kernel
clExecuteKernel(queue, kernel, NULL, range, NULL, 0, NULL);
```

References

- Khronos Group (2008-06-16). "Khronos Launches Heterogeneous Computing Initiative". Press release. http://www.khronos.org/news/press/releases/khronos_launches_heterogeneous_computing_initiative/. Retrieved 2008-06-18.
- "OpenCL gets touted in Texas". MacWorld. 2008-11-20. http://www.macworld.com/article/136921/2008/11/opencl.html?lsrc=top_2. Retrieved 2009-06-12.
- Khronos Group (2008-12-08). "The Khronos Group Releases OpenCL 1.0 Specification". Press release. http://www.khronos.org/news/press/releases/the_khronos_group_releases_opencl_1.0_specification/. Retrieved 2009-06-12.
- Apple Inc. (2008-06-09). "Apple Previews Mac OS X Snow Leopard to Developers". Press release. http://www.apple.com/pr/library/2008/06/09snowleopard.html. Retrieved 2008-06-09.

References

- AMD (2008-08-06). "AMD Drives Adoption of Industry Standards in GPGPU Software Development". Press release. http://www.amd.com/us-en/Corporate/VirtualPressRoom/0,,51_104_543~127451,00.html. Retrieved 2008-08-14.
- "AMD Backs OpenCL, Microsoft DirectX 11". eWeek. 2008-08-06. http://www.eweek.com/c/a/Desktops-and-Notebooks/AMD-Backing-OpenCL-and-Microsoft-DirectX-11/. Retrieved 2008-08-14.
- "HPCWire: RapidMind Embraces Open Source and Standards Projects". HPCWire. 2008-11-10. http://www.hpcwire.com/topic/applications/RapidMind_Embraces_Open_Source_and_Standards_Projects.html. Retrieved 2008-11-11.
- Nvidia (2008-12-09). "NVIDIA Adds OpenCL To Its Industry Leading GPU Computing Toolkit". Press release. http://www.nvidia.com/object/io_1228825271885.html. Retrieved 2008-12-10.
- "OpenCL Development Kit for Linux on Power". alphaWorks. 2009-10-30. http://www.alphaworks.ibm.com/tech/opencl. Retrieved 2009-10-30.
- "OpenCL Demo, AMD CPU". 2008-12-10. http://www.youtube.com/watch?v=sLv_fhQlqis. Retrieved 2009-03-28.

References

- "OpenCL Demo, NVIDIA GPU". 2008-12-10. http://www.youtube.com/watch?v=PJ1jydg8mLg. Retrieved 2009-03-28.
- "AMD and Havok demo OpenCL accelerated physics". PC Perspective. 2009-03-26. http://www.pcper.com/comments.php?nid=6954. Retrieved 2009-03-28.
- "NVIDIA Releases OpenCL Driver To Developers". NVIDIA. 2009-04-20. http://www.nvidia.com/object/io_1240224603372.html. Retrieved 2009-04-27.
- "AMD does reverse GPGPU, announces OpenCL SDK for x86". Ars Technica. 2009-08-05. http://arst.ch/5te. Retrieved 2009-08-06.
- Dan Moren; Jason Snell (2009-06-08). "Live Update: WWDC 2009 Keynote". *macworld.com*. MacWorld. http://www.macworld.com/article/140897/2009/06/keynote.html. Retrieved 2009-06-12.
- "Mac OS X Snow Leopard – *Technical specifications and system requirements*". Apple Inc. 2009-06-08. http://www.apple.com/macosx/specs.html. Retrieved 2009-08-25.
- "ATI Stream Software Development Kit (SDK) v2.0 Beta Program". http://developer.amd.com/GPU/ATISTREAMSDKBETAPROGRAM/Pages/default.aspx#one. Retrieved 2009-10-14.

References

- "Apple entry on LLVM Users page". http://llvm.org/Users.html#Apple. Retrieved 2009-08-29.
- "Nvidia entry on LLVM Users page". http://llvm.org/Users.html. Retrieved 2009-08-06.
- "Rapidmind entry on LLVM Users page". http://llvm.org/Users.html. Retrieved 2009-10-01.
- "Zack Rusin's blog post about the Mesa Gallium3D OpenCL implementation". http://zrusin.blogspot.com/2009/02/opencl.html. Retrieved 2009-10-01.
- http://www.via.com.tw/en/resources/pressroom/pressrelease.jsp?press_release_no=4327

References

- "ATI Stream SDK v2.0 with OpenCL™ 1.0 Support". http://developer.amd.com/gpu/ATIStreamSDK/Pages/default.aspx. Retrieved 2009-10-23.
- "OpenCL". SIGGRAPH2008. 2008-08-14. http://s08.idav.ucdavis.edu/munshi-opencl.pdf. Retrieved 2008-08-14.
- "Fitting FFT onto G80 Architecture" (PDF). Vasily Volkov and Brian Kazian, UC Berkeley CS258 project report. May 2008. http://www.cs.berkeley.edu/~kubitron/courses/cs258-S08/projects/reports/project6_report.pdf. Retrieved 2008-11-14.
- "OpenCL on FFT". Apple. 16 Nov 2009. https://developer.apple.com/mac/library/samplecode/OpenCL_FFT/index.html. Retrieved 2009-12-07.

Links

- Official site
- OpenCL for NVIDIA (Download page)
- OpenCL for AMD (Download page)
- OpenCL for IBM Cell broadband Engine (download page)
- OpenCL for S3 Chrome announcement
- OpenCL-Z
- PyOpenCL by Goncalo Carvalho
- PyOpenCL by Andreas Klöckner

Links

- The Open Toolkit library cross-platform C# OpenGL, OpenAL and OpenCL wrapper for Mono/.Net
- GPU Modeling and Development in OpenCL
- First public demonstration of OpenCL by NVIDIA on December 12, 2008 at Siggraph Asia
- First public demonstration of OpenCL by AMD on December 12, 2008 at Siggraph Asia
- OpenCL: What you need to know – article published in Macworld, August 2008
- HPCWire: OpenCL on the Fast Track
- OpenCL explained as part of a larger article on Snow Leopard (MacOS X 10.6), published at ArsTechnica in August 2009.