OpenCL Overview
SIGGRAPH Asia, November 2012

Neil Trevett
President, The Khronos Group
Vice President Mobile Content, NVIDIA
Processor Parallelism

CPUs
Multiple cores driving performance increases

GPUs
Increasingly general purpose data-parallel computing

OpenCL is a programming framework for heterogeneous compute resources
OpenCL – Heterogeneous Computing

• Cross-platform/vendor standard for harnessing all system compute resources

• Native framework for programming diverse parallel computing resources
  - CPU, GPU, DSP – as well as hardware blocks(!)

• Define N-dimensional computation domain
  - Execute ‘C’ kernel at each point in computation domain

• Powerful, low-level flexibility
  - Foundational access to compute resources for higher-level engines, frameworks and languages

• Embedded profile
  - No need for a separate “ES” spec
  - Reduces precision requirements

One code tree can be executed on CPUs or GPUs
OpenCL Working Group Members

- Diverse industry participation — many industry experts
  - Processor vendors, system OEMs, middleware vendors, application developers
  - Academia and research labs, FPGA vendors
- NVIDIA is chair, Apple is specification editor
OpenCL Overview

• **C Platform Layer API**
  - Query, select and initialize compute devices

• **Kernel Language Specification**
  - Subset of ISO C99 with language extensions
  - Well-defined numerical accuracy: IEEE 754 rounding with specified max error
  - Rich set of built-in functions: cross, dot, sin, cos, pow, log ...

• **C Runtime API**
  - Runtime or build-time compilation of kernels
  - Execute compute kernels across multiple devices
OpenCL Platform Model

- **One Host + one or more Compute Devices**
  - Each Compute Device is composed of one or more Compute Units
  - Each Compute Unit is further divided into one or more Processing Elements
OpenCL Execution Model Details

- **Kernel**
  - Basic unit of executable code ~ C function
  - Data-parallel or task-parallel

- **Program**
  - Collection of kernels and functions
    ~ dynamic library with run-time linking

- **Command Queue**
  - Applications queue kernels & data transfers
  - Performed in-order or out-of-order

- **Work-item**
  - An execution of a kernel by a processing element
    ~ thread

- **Work-group**
  - A collection of related work-items that execute on
    a single compute unit ~ core

Example of parallelism types
An N-dimension domain of work-items

- Kernels executed across a global domain of *work-items*
- Work-items grouped into local *workgroups*
- Define the “best” N-dimensioned index space for your algorithm
  - Global Dimensions: 1024 x 1024 (whole problem space)
  - Local Dimensions: 128 x 128 (work group ... executes together)

Synchronization between work-items possible only within workgroups: barriers and memory fences

Cannot synchronize outside of a workgroup
OpenCL Memory Model

- **Private Memory**
  - Per work-item

- **Local Memory**
  - Shared within a workgroup

- **Global/Constant Memory**
  - Visible to all workgroups

- **Host Memory**
  - On the CPU

Memory management is Explicit
You must move data from host -> global -> local ... and back
Programming Kernels: OpenCL C

• Derived from ISO C99
  - But without some C99 features such as standard C99 headers, function pointers, recursion, variable length arrays, and bit fields

• Language Features Added
  - Work-items and workgroups
  - Vector types
  - Synchronization
  - Address space qualifiers

• Also includes a large set of built-in functions
  - Image manipulation
  - Work-item manipulation,
  - Math functions, etc.
OpenCL Execution Model

- OpenCL application runs on a host which submits work to the compute devices
- Context - the environment within which work-items execute
  - Includes devices and their memories and command queues
- Applications queue kernel execution
  - Executed in-order or out-of-order
Synchronization: Queues & Events

- Events can be used to synchronize kernel executions between queues
- Example: 2 queues with 2 devices

Diagram:

- GPU: Enqueue Kernel 1, Kernel 1, Enqueue Kernel 2
- CPU: Enqueue Kernel 1, Kernel 1, Enqueue Kernel 2

Kernel 2 starts before the results from Kernel 1 are ready.

Kernel 2 waits for an event from Kernel 1 and does not start until the results are ready.
Creating an OpenCL Program

- CPU
- GPU
- DSP

- Context
- Programs
- Kernels
- Memory Objects
- Command Queue

- Programs
  - Kernel0
  - Kernel1
  - Kernel2
- Images
- Buffers
- In order & out of order

Compile → Create data and arguments → Send for execution
OpenCL Milestones

- Six months from proposal to released OpenCL 1.0 specification
  - Due to a strong initial proposal and a shared commercial incentive

- Multiple conformant implementations shipping on desktop
  - For CPUs and GPUs on multiple OS

- 18 month cadence between dot releases
  - Backwards compatibility protects software investment
OpenCL 1.2 Announced in December 2011

• Significant updates - Khronos being responsive to developer requests
  - Updated OpenCL 1.2 conformance tests available
  - Multiple implementations underway

• Backward compatible upgrade to OpenCL 1.1
  - OpenCL 1.2 will run any OpenCL 1.0 and OpenCL 1.1 programs
  - OpenCL 1.2 platform can contain 1.0, 1.1 and 1.2 devices
  - Maintains embedded profile for mobile and embedded devices
Partitioning Devices

- Devices can be partitioned into sub-devices
  - More control over how computation is assigned to compute units
- Sub-devices may be used just like a normal device
  - Create contexts, building programs, further partitioning and creating command-queues
- Three ways to partition a device
  - Split into equal-size groups
  - Provide list of group sizes
  - Group devices sharing a part of a cache hierarchy
OpenCL Built-in Kernels

• Used to control non-OpenCL C-capable resources on an SOC – ‘Custom Devices’
  - E.g. Video encode/decode, Camera ISP ...

• Represent functions of Custom Devices as an OpenCL kernel
  - Can enqueue Built-in Kernels to Custom Devices alongside standard OpenCL kernels

• OpenCL run-time a powerful coordinating framework for ALL SOC resources
  - Programmable and custom devices controlled by one run-time

Built-in kernels enable control of specialized processors and hardware from OpenCL run-time
Installable Client Driver

- Analogous to OpenGL ICDs in use for many years
  - Used to handle multiple OpenGL implementations installed on a system

- Optional extension
  - Platform vendor will choose whether to use ICD mechanisms

- Khronos OpenCL installable client driver loader
  - Exposes multiple separate vendor installable client drivers (Vendor ICDs)
  - Open source released! [http://www.khronos.org/registry/cl/](http://www.khronos.org/registry/cl/)

- Application can access all vendor implementations
  - The ICD Loader acts as a de-multiplexor

ICD Loader ensures multiple implementations are installed cleanly

ICD Loader enables application to use any of the installed implementations

```
Application -> ICD Loader
           ^
           | Vendor #1
           | OpenCL
           v
Vendor #2
OpenCL

```

```
Vendor #3
OpenCL

```

© Copyright Khronos Group, 2012 - Page 18
Other Major New Features in OpenCL 1.2

- **Separate compilation and linking of objects**
  - Provides the capabilities and flexibility of traditional compilers
  - Create a library of OpenCL programs that other programs can link to

- **Enhanced Image Support**
  - Added support for 1D images, 1D & 2D image arrays
  - OpenGL sharing extension now enables an OpenCL image to be created from an OpenGL 1D texture, 1D and 2D texture arrays

- **DX9 Media Surface Sharing**
  - Efficient sharing between OpenCL and DirectX 9 or DXVA media surfaces

- **DX11 surface sharing**
  - Efficient sharing between OpenCL and DirectX 11 surfaces

- And many other updates and additions..
OpenCL 1.2 Update – Optional Extensions

• Create an OpenCL image from a OpenGL multi-sampled texture
  - Provides more flexibility in interoperating 3D graphics and compute

• Create 2D images from an OpenCL buffer
  - Process memory structures using the advanced properties of OpenCL images

• Security features for WebCL implementations layered over OpenCL
  - Initialize local and private memory before a kernel begins execution
  - Query and API to terminate an OpenCL context to ensure a long running kernel does not affect system stability

• Load an OpenCL program object from a Standard Portable Intermediate Representation (SPIR) instance
  - Increased tool chain flexibility and avoids the need to ship kernel source in commercial applications
OpenCL Roadmap

OpenCL-HLM (High Level Model)
Exploring high-level programming model, unifying host and device execution environments through language syntax for increased usability and broader optimization opportunities

Long-term Core Roadmap
Significant enhancements to memory and execution model:
- Better handle irregular work loads
- Reduce overhead of host/device data exchange
- Better image handing and API interop
- Enhanced language constructs and built-in functions for ease of use

OpenCL-SPIR (Standard Parallel Intermediate Representation)
Exploring LLVM-based, low-level Intermediate Representation for code obfuscation/security and to provide target back-end for alternative high-level languages
OpenCL as Parallel Compute Foundation

- **HLM**: C++ syntax/compiler extensions
- **WebCL**: JavaScript binding to OpenCL for initiation of OpenCL C kernels
- **Aparapi**: Java language extensions for parallelism
- **River Trail**: Language extensions to JavaScript
- **C++ AMP**: C++ syntax/compiler extensions

CUDA or DirectCompute may also be used as compiler targets - BUT OpenCL provides cross-platform, cross-vendor coverage

Intel Shevlin Park Project using Clang/LLVM and OpenCL

[http://llvm.org/devmtg/2012-11/#talk10](http://llvm.org/devmtg/2012-11/#talk10)
Many advanced photo apps today run on a single CPU
  - Suboptimal performance and power

OpenCL is a platform to harness CPUs/GPUs for advanced imaging
  - Even if code is ‘branchy’

“The tablet ... has new multimedia capabilities, including a computational camera, which lets devs tap directly into its computational capability through new application programming interfaces such as OpenCL. That access enables next-generation use cases such as light-field cameras for mobile devices.”
OpenCL Rollout on Mobile Starting

If you're looking to create that perfect multi-threaded, NFC, GPS-based OpenCL app (and who isn't?), but found your development board options too limited, Samsung has good news. It's just launched the Arndale community development board around its Exynos 5 Dual Soc, with the ARM Cortex-A15 dual-core CPU and ARM Mali T604 GPU. Those specs give the board "an order of magnitude lift in performance" from the last model and full profile OpenCL capability, according to Samsung, on top of NFC, GPS and camera sensor features. That'll let developers go to town on new games, security and multimedia apps next month for $250 -- if that's you, check the PR after the break or coverage below.
Adobe at SIGGRAPH 2012

Adobe ❤️ OpenCL

- Compute API supported across vendors
- Programming model familiar to C programmers
- Demonstrated performance
- Same compute kernels on CPU and GPU!

- Adobe is now active member of OpenCL working group
  - Contributing Adobe’s experience and minds to continue OpenCL evolution
OpenCL and OpenGL Compute Shaders

- OpenGL compute shaders provide access from GLSL to all GL pipe memory
  - Memory buffer and textures

- OpenGL compute shaders and OpenCL support different use cases
  - OpenCL provides a significantly more powerful and complete compute solution

1. Fine grain compute operations inside OpenGL
2. GLSL Shading Language
3. Execute on single GPU only

Developer driven decision

1. Full ANSI C programming of heterogeneous CPUs and GPUs
2. Utilize multiple processors
3. Coarse grain, buffer-level interop with OpenGL

Enhanced 3D Graphics apps “Shaders++”
Imaging Video Physics AI
Pure compute apps touching no pixels

© Copyright Khronos Group, 2012 - Page 26
OpenCL Desktop Implementations

OpenCL Books – Available Now!

- **OpenCL Programming Guide** - The “Red Book” of OpenCL
- **OpenCL in Action**
- **Heterogeneous Computing with OpenCL**
- **The OpenCL Programming Book**
Spec Translations

- **Japanese OpenCL 1.1 spec translation available today**
  - Valued partnership between Khronos and CUTT in Japan

- **Working on OpenCL 1.2 specification translations**
  - Japanese, Korean and Chinese
Khronos OpenCL Resources

• OpenCL is 100% free for developers
  - Download drivers from your silicon vendor

• OpenCL Registry
  - www.khronos.org/registry/cl/

• OpenCL 1.2 Reference Card
  - PDF version

• Online Man pages
  - http://www.khronos.org/registry/cl/sdk/1.2/docs/man/xhtml/

• OpenCL Developer Forums
  - Give us your feedback!
  - www.khronos.org/message_boards/
Expanding Platform Reach for Graphics and Computation

**Desktop**
- **Graphics**: OpenGL
- **Compute**: OpenCL (Full Profile)

**Mobile**
- **Graphics**: OpenGL ES
- **Compute**: OpenCL (Full Profile and Embedded Profile)

**Web**
- **Graphics**: WebGL
- **Compute**: WebCL

**WebGL** on majority of production desktops now. WebGL pervasively available on mobile in next 12 months.

WebCL will start deploying in next 12-18 months.
Thank you

• Any questions?