Khronos Update

OpenCL, SYCL and SPIR - The Next Steps

Neil Trevett | Khronos President
NVIDIA Vice President Developer Ecosystem
OpenCL Working Group Chair
ntrevett@nvidia.com | @neilt3d
Boston, May 2019
OpenCL Update from the Khronos Perspective

1. Widening support and industry usage
   Heterogenous compute is the ‘new Moore’s Law’
   Critical to new-generation mobile/embedded systems

2. Strengthening the OpenCL Ecosystem
   Increasing community engagement
   Leveraging the power of open source resources

3. Deploying New Functionality
   Extensions and core specs
   Processor deployment flexibility
   Community-based kernel language tooling

4. Platform Deployment Flexibility
   Enabling OpenCL with no native drivers
OpenCL Heterogeneous Computing

A programming and runtime framework for heterogeneous compute resources
Low-level control over memory allocation and parallel task execution
Simpler and relatively lightweight compared to GPU APIs

Fragmented GPU API Landscape

Growing number of optimized OpenCL libraries
Vision and Imaging
Machine Learning and Inferencing
Linear Algebra and Mathematics
Physics

Many mobile SOCs and Embedded Systems becoming increasingly heterogeneous
Autonomous vehicles
Vision and inferencing

Heterogeneous Compute Resources

© Copyright Khronos® Group 2019 - Page 3
OpenCL Conformant Implementations

Vendor timelines are first conformant submission for each specification.

Adoption Since Last IWOCL
Intel: 2.1 for latest processors on Windows and Linux
Intel: 1.0 on Arria 10 GX FPGA
NVIDIA: 1.2 for Turing GPUs on Windows and Linux
Qualcomm: 2.0 on Adreno GPUs on Android
OpenCL User Adoption

# OpenCL-based GitHub Repos

70% Increase in two years

OpenCL

Desktop Creative Apps

Parallel Computation Languages

Linear Algebra Libraries

Vision and Imaging Libraries

Math and Physics Libraries

OpenCL is Pervasive!

© Copyright Khronos® Group 2019 - Page 5
OpenVX
A high-level graph-based abstraction for Portable, Efficient Vision Processing
Can be implemented on almost any hardware or processor

NNEF (Neural Network Exchange Format)
For transferring trained Neural Networks into inferencing accelerators
Provides stability needed by hardware vendors through true multicompany governance
Extending OpenVX for Custom Nodes

OpenVX/OpenCL Interop
- Provisional Extension
  - Enables custom OpenCL acceleration to be invoked from OpenVX User Kernels
  - Memory objects can be mapped or copied

Kernel/Graph Import
- Provisional Extension
  - Defines container for executable or IR code
  - Enables arbitrary code to be inserted as a OpenVX Node in a graph

OpenVX user-kernels can access command queue and cl_mem objects to asynchronously schedule OpenCL kernel execution.

Application

OpenVX data objects

Copy or export cl_mem buffers into OpenVX data objects

cl_mem buffers

Map or copy OpenVX data objects into cl_mem buffers

OpenVX/OpenCL Interop
Inferencing Acceleration

Many inferencing stacks use OpenCL to access hardware acceleration. OpenCL has excellent ML acceleration: Subgroups, small types, built-in functions for fixed function hardware.

To mix inferencing with vision and other custom processing.
SYCL Single Source C++ Parallel Programming

- SYCL 1.2.1 Adopters Program released in July 2018 with open source conformance tests soon

- Multiple SYCL libraries for vision and inferencing
  - SYCL-BLAS, SYCL-DNN, SYCL-Eigen

- Multiple Implementations shipping: triSYCL, ComputeCpp, HipSYCL
  - http://sycl.tech

C++ Kernel Fusion can gives better performance on complex apps and libs than hand-coding

Accelerated code passed into device OpenCL compilers

User application code

C++ template libraries

C++ template libraries

C++ template libraries

SYCL for OpenCL

OpenCL Devices

CPU

GPU

DSP

FPGA

Other technologies

CPU

Custom Processor
SYCL Implementations

- clang
  - OpenCL + SPIR-V
    - Intel CPUs/GPUs
    - other SPIR-V devices?
  - OpenCL + SPIR(-V)
    - Intel CPUs/GPUs
    - AMD GPUs (depending on driver stack)
    - ARM Mali
    - Renesas R-Car

- ComputeCpp
  - PTX devices
    - NVIDIA GPUs

- triSYCL
  - OpenCL + SPIR-df
    - Intel CPUs
    - GPU vendors
    - poCL (CPUs, NVIDIA GPUs)
    - Xilinx FPGAs

- hipSYCL
  - ROcm
    - AMD GPUs
    - NVIDIA GPUs

- sycl-gtx
  - OpenCL 1.2
    - pretty much anything

(Non-standard macros required)
Ecosystem Engagement

Contribute to open source specs, CTS and tools

OpenCL Forum and Slack Channel

Advisory Panel

Working Group

Spec fixes and suggestions made under the Khronos IP Framework. Open source contributions under repo’s CLA - typically Apache 2.0
https://github.com/KhronosGroup

Advisors under the Khronos IP Framework can comment and contribute to requirements and draft specifications

Khronos members under IP Framework and NDA can participate and vote in working group meetings

Open to all!
https://community.khronos.org/c/opencl
www.khr.io/slack

www.khr.io/slack
Khronos has open sourced OpenCL Specifications and Conformance Tests

Merge requests welcome from the community (subject to review by OpenCL working group)
File bugs for specifications, headers etc.
Mix your own documentation!

Contributions and Distribution under Apache 2.0

Spec Build System and Scripts

Spec and Ref Language Source
Redistribution under CC-BY 4.0

Contributions and Distribution under Apache 2.0

Khronos builds and Ratifies Canonical Specification under Khronos IP Framework. No changes or re-hosting allowed

Spec and Ref Language Source and derivative materials. Re-mixable under CC-BY by the industry and community

Community built documentation and tools

Anyone can test any implementation at any time

Conformant Implementations can use trademark and are covered by Khronos IP Framework

Conformance Test Suite Source

Source Materials for Specifications and Reference Documentation CONTRIBUTED Under Khronos IP Framework (you won’t assert patents against conformant implementations, and license copyright for Khronos use)
Underway - Unified OpenCL Specification

• Unified OpenCL API specification will describe the API for all versions of OpenCL
  - Rather than having a separate specification per version

• OpenCL SPIR-V environment, extension and SPIR-V specs are already unified
  - Working well - good developer feedback

• Easier for developers to navigate
  - And to consistently apply specification fixes and clarifications

• Working Group has started prototyping the spec work
  - Short introductory section describing the unified aspects
  - "missing before X.Y" and "deprecated by X.Y" language
  - Roughly as SPIR-V specification
    - [Link](https://github.com/KhronosGroup/OpenCL-Docs/issues/77)

• DRAFT unified spec is already uploaded!
  - [Link](https://github.com/KhronosGroup/OpenCL-Docs/files/3170333/OpenCL_API.pdf)
  - Feedback welcome!

  Eases opportunity to coherently include deprecation and version evolution rationale in specification
  - as requested in yesterday's BOF
OpenCL Evolution

**OpenCL Extension Specs**
- Vulkan / OpenCL Interop
- Scratch-Pad Memory Management
- Extended Subgroups
- SPIR-V 1.4 ingestion for compiler efficiency
- SPIR-V Extended debug info

**Spec Maintenance Updates**
- Regular updates for spec clarifications and bug fixes

**Integration of Extensions plus New Core functionality**
- Vulkan-like loader and layers
- ‘Flexible Profile’ for Deployment Flexibility

**Repeat Cycle for next Core Specification**

**OpenCL**
- May 2017
- OpenCL 2.2

**Target 2020**
- ‘OpenCL Next’
Embedded Processors & OpenCL Conformance

- The embedded market is a new frontier needing advanced heterogenous compute
  - E.g. Vision and inferencing using a wide range of processor architectures
- BUT OpenCL is currently monolithic - and arguably desktop/HPC-centric
  - E.g. a processor without 32-bit IEEE floating point cannot realistically be conformant
  - Vendors and developers do not want software emulation of higher precisions
- Many functionality requirements change between different markets and processors

OpenCL is disenfranchising one of its most important emerging market opportunities!

<table>
<thead>
<tr>
<th>Supported Precisions</th>
<th>DSP A</th>
<th>DSP B</th>
<th>DSP C</th>
</tr>
</thead>
<tbody>
<tr>
<td>8-bit integer</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>16-bit integer</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>32-bit integer</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>64-bit integer</td>
<td>✗</td>
<td>✓</td>
<td>✗</td>
</tr>
<tr>
<td>16-bit float</td>
<td>✗</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>32-bit float</td>
<td>✗</td>
<td>✗</td>
<td>✓</td>
</tr>
<tr>
<td>64-bit float</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>Possible to be OpenCL Compliant?</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
</tbody>
</table>
OpenCL Next ‘Flexible Profile’

Goals
Enable Conformant OpenCL implementations on more diverse processors and platforms
Enable vendors to ship functionality precisely targeting their customers and markets
A conformant OpenCL can expose precisely what is available in the hardware
Enable incremental feature adoption

Design Philosophy
More OpenCL features become optional for enhanced deployment flexibility
Optionality includes both API and language features e.g. floating point precisions
Enhanced query mechanisms - precisely which features are supported by a device?

OpenCL Next aims to be a flexible run-time framework that can be pervasively and cost-effectively deployed across a wider range of heterogenous devices
Flexible Profile and Feature Sets

- In OpenCL Next Flexible Profile features become optional for enhanced deployment flexibility
  - API and language features e.g. floating point precisions

- Feature Sets reduce danger of fragmentation
  - Defined to suit specific markets - e.g. desktop, embedded vision and inferencing

- Implementations are conformant if fully support feature set functionality
  - Supporting Feature Sets will help drive sales - encouraging consistent functionality per market
  - An implementation may support multiple Feature Sets

OpenCL 2.2 Functionality

- = queryable, optional feature

- Khronos-defined
  - OpenCL 2.2 Full Profile Feature Set
  - OpenCL 1.2 Full Profile Feature Set

- Industry-defined Feature Set E.g. Embedded Vision and Inferencing
OpenCL Next Feature Set Discussions

- OpenCL Next Flexible Profile will leverage the new Unified OpenCL Specification
  - Feature Sets can easily select from any previous functionality

- What could be useful Feature Sets?
  - Feature sets for previous spec versions e.g. OpenCL 1.2?
  - ‘Desktop’ Feature Set to raise the universally available baseline above OpenCL 1.2?
  - Set of OpenCL functionality that runs efficiently over Vulkan?
  - Vertical market focused - e.g. inferencing, vision processing

- Some vertically integrated markets don’t care about cross-vendor app portability
  - But still want to use industry standard programming framework - reduces costs for engineering, tooling, training etc.
  - Allow conformance for any combination of features - no Feature Sets
  - Enables minimal footprint OpenCL per system - ideal for Safety Certification

In some vertically integrated embedded markets, application portability is not so important
OpenCL Tooling Ecosystem Subgroup

- Coordinating collaborative opportunities between SPIR-V and LLVM ecosystems
  - Encouraging joint development of new features and tooling integration
- Active open source projects - making SPIR-V a first-class LLVM citizen
  - Extending SPIR-V<->LLVM Translation for OpenCL - release 8.0 is out!
    - https://github.com/KhronosGroup/SPIRV-LLVM-Translator
  - Libclc: implementation of the OpenCL C 1.1 library for use with Clang
    - https://libclc.llvm.org/
  - Upstream SPIR-V backend translation from Clang/LLVM - in discussion
    - https://clang.llvm.org/
  - Front-end support for all OpenCL C language versions in Clang
- C++ for OpenCL in Clang
  - Experimental support for C++ in OpenCL

Khroros increasing efforts around developing, coordinating and releasing open source tooling

Examples and Tutorials
Make it approachable

Tools
Make it usable

Conformance Tests
Make it reliable

Specifications
Make it possible
C++ for OpenCL in Clang Project

- Front end with OpenCL C 2.0 and C++17 capabilities
  - Experimental support in Clang 9.0
  - Expect Alpha in September 2019

- Existing OpenCL C code is valid and fully compatible
  - Enables gradual transition to C++ for existing apps

- Offline compilation into SPIR-V or device binary
  - Generates SPIR-V 1.0 for most features
  - Uses SPIR-V 1.2 where necessary

- Works with any OpenCL 2.0 driver
  - Possible future driver updates may take advantage of enhanced language capabilities

- Check it out in Compiler Explorer
  - https://godbolt.org/z/nGvxAC

```
template<class T> T add( T x, T y )
{
    return x + y;
}

__kernel void test( __global float* a, __global float* b) 
{
    auto index = get_global_id(0);
    a[index] = add(b[index], b[index+1]);
}
```

clang -std=c++ test.cl

Good example of the power and flexibility of offline compilation using SPIR-V - how can we embrace and support this class of language project?
Current OpenCL Compilation Flow

Frontend and backend compilers ship with OpenCL driver
Adding new language capabilities → new driver version
Offline Compilation with SPIR-V

Offline compilation stage has no direct driver interaction

If offline compiler generates valid SPIR-V, front-end can add language capabilities without requiring a driver update
Examples: Variadic Macros, Atomic Functions w/ Address Spaces, Templates are possible with no SPIR-V changes
Community OSS-Driven Language Evolution

- Language capabilities can move faster with offline compilation
  - Benefits Khronos: easier to explore and iterate on new features
  - Benefits implementers: no driver updates required for new language features
  - Benefits developers: one consistent tool enables all implementations

- Proposal -> accelerate towards community-based language ecosystem
  - Host non-ratified offline language documentation at Khronos - agile updates
  - Add Compiler Capabilities extension to core OpenCL - using preprocessor #defines
  - Code can choose to interrogate extension for enhanced compiler capabilities

- N.B. NOT proposing to remove OpenCL C online compilation!
  - Key to many use-cases - and can also absorb new features over time
  - https://github.com/KhronosGroup/OpenCL-Docs/issues/65
SPIR-V Ecosystem

- GLSL
- HLSL
- Third party kernel and shader languages
- glslang
- DXC

**SPIR-V**
- Khronos-defined cross-API IR
- Native graphics and parallel compute support
- Easily parsed/extended 32-bit stream
- Data object/control flow retained for effective code generation/translation

**SPIR-V (Dis)Assembler**
- SPIR-V (Dis)Assembler
- SPIR-V Validator
- SPIRV-opt | SPIRV-remap

**Optimization Tools**
- Environment spec used for compilation needs to match target runtime
- Vulkan
- OpenGL

**Front-ends**
- OpenCL C
- OpenCL C++
- SYCL for ISO C++

**Bi-directional Translators**
- LLVM to SPIR-V

**3rd Party Hosted OSS**
- Khronos-hosted Open Source Projects
  - https://github.com/KhronosGroup/SPIRV-Tools

**Community**
- Khronos cooperating with clang/LLVM Community

**Third party kernel and shader languages**
- MSL
- HLSL
- GLSL

**Front-ends**
- SPIRV-Cross
- GLSL
- HLSL
- MSL

**APIs**
- Vulkan
- OpenGL

**Projects**
- SPIR-V Tools
- LLVM
- glslang
- DXC

**Environments**
- Khronos-hosted Open Source Projects

**Benefits**
- Native graphics and parallel compute support
- Easily parsed/extended 32-bit stream
- Data object/control flow retained for effective code generation/translation
OpenCL Platform Deployment Flexibility

- **Clspv** - Google’s experimental compiler for OpenCL C to Vulkan SPIR-V
  - Open source - tracks top-of-tree LLVM and clang, not a fork
  - Originally tested on over 200K lines of Adobe OpenCL C production code
  - Sony is now working with Google to compile their production kernels
    - 355 kernels in 75 files - 40 files compiled successfully at first pass

- **Clvk** - experimental OpenCL to Vulkan API shim by Kevin Petit
  - Early days - but Halide’s OpenCL back-end successfully running over Vulkan

![Diagram showing OpenCL C Source, Clspv Compiler, Run-time API Translator, OpenCL Host Code, and Runtime connected to Clspv Compiler and OpenCL Host Code, with prototype open source projects links provided: https://github.com/google/clspv and https://github.com/kpet/clvk.](image-url)
Refining clspv with Diverse Workloads

Kernel repositories for use in long-term perf/regression testing - including kernels from open source OpenCL libraries
https://github.com/KhronosGroup/OpenCL-Sample-Kernels

Compile diverse OpenCL C kernel workloads

Efficient mapping to Vulkan SPIR-V?

Updates to compiler achieve efficient mapping?

Propose updates to Vulkan programming model

ISVs can compile and test commercially-sensitive kernels - and report issues on GitHub

Please let us know if you are interested to try running your kernels through clspv!

Vulkan is already expanding its compute model e.g. 16-bit storage, compact memory types and operations, Variable Pointers, Subgroups

Community assistance to add capabilities is welcome!

Increasing deployment options for OpenCL developers to any platform where Vulkan is a supported API e.g. Android
Vulkan Portability Initiative on Apple

Almost all mandatory Vulkan 1.0 functionality is supported:
- No Triangle Fans
- No separate stencil reference masks
- Events are not supported

Selected Optional Features and Extensions are added as required - driven by industry input and feedback
- Robust buffer access
- BC texture compressed formats
- Fragment shader atomics

https://github.com/KhronosGroup/MoltenVK

Open source SDK to build, run, and debug applications on macOS - including validation layer support
https://vulkan.lunarg.com/

SPIRV-Cross
Convert SPIR-V shaders to Metal Shaders

macOS / iOS Run-time
Maps Vulkan to Metal API

MoltenVK supports macOS 10.11 / iOS 9.0 and up

Open source beta release for macOS

OPEN SOURCE.
Free to use - no fees or royalties - including commercial applications

https://github.com/KhronosGroup/MoltenVK
Apps Shipping On Apple with Vulkan Backend

Forsaken Remastered was just updated with Vulkan support! If you’re on Linux, you’re probably hitting 60fps with the existing OpenGL renderer, but it’s good to be future proof. If you’re on a Mac, though, you definitely want to switch. On my MacBook, the framerate goes from around 15 to a solid 60!

Initial Vulkan Performance On macOS With Dota 2 Is Looking Very Good

Written by Michael Larabel in Valve on 1 June 2018 at 03:37 PM EDT 34 Comments

Yesterday Valve released Vulkan support for Dota 2 on macOS. Indeed, this first major game relying upon MoltenVK for mapping Vulkan over the Apple Metal drivers is delivering performance gains.

Production Dota 2 on Mac Ships - up to 50% more perf than Apple’s OpenGL

First iOS Apps using MoltenVK ship through app store

Qt Running on Mac through MoltenVK

June 2018

Multiple iOS and macOS apps organically ported - support through MoltenVK website e.g. Forsaken Remastered on Mac

Google Filament PBR Renderer on Mac

Artifact from Steam ships on macOS - first Vulkan-only Valve app on Mac

Diligent Engine runs on MacOS

September 2018

November 2018

January 2019

Valve Releases Artifact As Its Cross-Platform, Vulkan-Powered Digital Card Game

Written by Michael Larabel in Valve on 26 November 2018 at 14:04 PM EST 26 Comments

Valve managed to ship their latest game today as planned and without any major delays.

Artifact is now available with launch-day support for Linux, macOS, and Windows. Artifact is a competitive digital card game in order to play in 1v1 or 2v2 in genres like gothic fantasy, this isn’t just a new property and...
Valve - Vulkan Dota 2 on macOS

Shipping Now.
Vulkan delivering up to 50% performance increase over native OpenGL
Universal Deployment Flexibility?

Open source SPIRV-Cross converts SPIR-V to MSL or HLSL

Open source shims convert Vulkan to Metal or D3D API calls

Open source tools enable OpenCL and Vulkan apps to be increasingly deployed on many platforms

OpenCL Programs

Clspv and clvk

Native Vulkan Drivers

OpenCL

UWP and D3D based Consoles

macOS

iOS

Microsoft DirectX 11

DirectX 12

SteamOS

Switch

Android

iOS

redhat

Ubuntu

Windows 7

Windows 10

macOS

iOS
OpenCL Portability Initiative on Apple?

- OpenCL C Front-end
- OpenCL C++ Front-end
- SYCL for ISO C++ Front-end
- C++ for OpenCL in clang Front-end

*Applications*

- SPIR-V Cross
  - Convert SPIR-V kernels to Metal Shaders

*Kernels*

- LLVM
  - LLVM to SPIR-V Bi-directional Translators
- Vulkan SPIR-V
- OpenCL SPIR-V

*API Calls*

- macOS / iOS Run-time
  - Maps OpenCL to Metal API Calls

Is there community interest in cooperation over in a direct ‘OpenCL over Metal’ OSS project? Can Khronos host/help coordinate?

Would need runtime conversion of API calls. OpenCL is much closer to Metal than Vulkan - easier to shim.

SPIR-V Cross would need expanding to handle OpenCL dialect of SPIR-V.
Get Involved!

- **OpenCL is driving to new levels of usability and deployment flexibility**
  - We want to know what *you* need from OpenCL!

- **OpenCL Next and Feature Sets**
  - Let us know what you think!

- **Multiple ways to engage and help OpenCL evolve**
  - Join Khronos for a voice and a vote in any of these standards
  - Or ask about an invite to the OpenCL Advisory Panel
  - Or consider getting involved in OpenCL OSS projects
    - [https://github.com/KhronosGroup](https://github.com/KhronosGroup)
  - Or talk to us on Slack and the forums
    - [https://community.khronos.org/c/opencl](https://community.khronos.org/c/opencl)
    - [www.khr.io/slack](http://www.khr.io/slack)

- **Neil Trevett**
  - ntrevett@nvidia.com
  - @neilt3d
  - www.khronos.org

---

If you need OpenCL let your hardware vendors know! Your voice counts!